GLG Widget Reference Manual

GLG Toolkit
Version 4.3

Generic Logic, Inc.

Generic Logic, Inc.
6 University Drive 206-125
Amherst, MA 01002
USA
Telephone: (413) 253-7491
FAX: (413) 241-6107
email: support@genlogic.com
web: www.genlogic.com

Copyrights and Trademarks

April 7, 2023
Software Release Version 4.3

GLG Widget Reference Manual

Table of Contents

Chapter 1 Using the GLG Widgets 11

Customizing GLG Widgets in the Graphics Builder 11

Using Scalable Text 12

Using Widgets in the Graphics Builder 12

Loading and Adding Widgets to the Drawing 12

Prototyping Widgets Using Predefined Animation Commands 13

Prototyping Dials and Meters With Custom Animation Commands 13

Prototyping Real-Time Charts with Custom Animation Commands 14

Prototyping 2D and 3D Graph Widgets with Custom Animation Commands 14

Storing Custom Animation Command in the Drawing 14

Real-Time Chart Widgets 14

Using Real-Time Chart Widgets 15

Input Widgets 16

Meters, Dials and Avionics Gauges 17

Knob Widgets 18

Slider and Linear Gauge Widgets 18

Native Slider and Scrollbar Widgets 20

Button and Toggle Widgets 20

Native Button and Toggle Widgets 21

Menu Widgets 22

Tabs Widgets 22

Text Widgets 23

Spinner Widgets 24

List and Option Menu Widgets 25

Clock and Stopwatch Widgets 25

Palette Widgets 26

Resource, Tag, Alarm and Font Browser Widgets 26

Value Display Widgets 26

State Indicator Widgets 28

Process Control Symbols 28

Tank Widgets 29

Column Widgets 31

Heat Exchanger Widgets 31

Valve Widgets 31

Process Control Equipment 32

Process Control Gauges and Indicators 32

Pipes and Lines 33

3D Pipes 33

Electrical and Electronic Circuit Symbols 34

Electric Switches and Relays 35

Transformers and Inductors 35

Resistors, Capacitors, Diodes 35

Transistors, Gates, Amplifiers 35

Miscellaneous Electrical Components 36

Special Widget Set 36

Layout Templates 36

Special Widgets 37

2D and 3D Graphs 38

Components of a GLG Graph 39

Naming Conventions 39

Performance Optimizations 40

Commonly Used Graph Resources 41

Data Samples and the Data Group 42

Data Area 46

Axes 47

Major And Minor Ticks 47

Grid Lines 48

Labels 48

Level Objects 51

Status Object 51

Titles 51

Legends 52

Multiple Axes 52

3D viewing 52

....52

Chapter 2 2D and 3D Graph Widget Resources 53

Bar Graph
Signed Bar Graph
Histogram Graph..54

Animation Data 54

Resource Sets..54

Horizontal Bar Graph..54

Animation Data 54

Resource Sets..55

Symmetrical Bar Graph 55

Animation Data 55

Resource Sets..55

Packed Bar Graph 56

Animation Data 56

Resource Sets..56

Stacked Bar Graph 57

Animation Data 57

Resource Sets..57

Step Graph..57

Animation Data 58

Resource Sets..58

Filled Step Graph 58

Animation Data 58

Resource Sets..58

Horizontal Step Graph 59

Animation Data 59

Resource Sets..59

Horizontal Filled Step Graph 60

Animation Data 60

Resource Sets..60

Multiline Step Graph..60

Animation Data 60

Resource Sets..61

Filled Multiline Step Graph 61

Animation Data 61

Resource Sets 62

Line Graph
Filled Line Graph
Line Segment Graph
Point Graph 62

Animation Data 62

Resource Sets 63

Multiline Graph
Filled Multiline Graph
Multiset Line Segment Graph
Multiset Point Graph 63

Animation Data 63

Resource Sets 64

XY Line Graph
XY Point Graph 64

Animation Data 64

Resource Sets 64

Multiline XY Graph
Multiset Point XY Graph 65

Animation Data 65

Resource Sets 65

Multiaxis Multiline XY Graph 66

Animation Data 66

Resource Sets 66

Polar Line Graph
Polar Filled Line Graph
Polar Line Segment Graph
Polar Point Graph 67

Animation Data 67

Resource Sets 67

Multiline Polar Graph 68

Animation Data 68

Resource Sets 68

XY Polar Line Graph 68

Animation Data 69

Resource Sets 69

Multiline Polar XY Graph 69

Animation Data 69

Resource Sets....70

Pie Chart....70

Animation Data 70

Resource Sets....70

Combination Graph....71

Animation Data 71

Resource Sets....71

Stock Graphs 71

Animation Data 72

Resource Sets....72

3D Bar Graph
3D Signed Bar Graph
3D Histogram Graph
Pyramid Graph
Prism Graph
Cylinder Graph....73

Animation Data 73

Resource Sets....73

Multiset 3D Bar Graph
Multiset 3D Signed Bar Graph
Multiset Pyramid Graph
Multiset Prism Graph
Multiset Cylinder Graph 74

Animation Data 74

Resource Sets....74

3D Packed Bar Graph
3D Packed Signed Bar Graph 75

Animation Data 75

Resource Sets....75

Multiset 3D Packed Bar Graph
Multiset 3D Packed Signed Bar Graph....75

Animation Data 75

Resource Sets....76

3D Stacked Bar Graph 76

Animation Data 77

Resource Sets....77

Multiset 3D Stacked Bar Graph....77

Animation Data 77

Resource Sets 78

3D Step Graph 78

Animation Data 78

Resource Sets 79

Multiset 3D Step Graph 79

Animation Data 79

Resource Sets 79

Ribbon Graph 80

Animation Data 80

Resource Sets 80

Multiset Ribbon Graph 80

Animation Data 80

Resource Sets 81

Framed Surface Graph
Unframed Surface Graph 81

Animation Data 82

Resource Sets 82

Gantt Chart 82

Animation Data 82

Resource Sets 83

Horizontal 3D Slider Graph 83

Animation Data 83

Resources 83

Resource Sets 83

Vertical 3D Slider Graph 84

Animation Data 84

Resources 84

Resource Sets 84

Chapter 3 2D and 3D Graph Resource Sets 85

Resource Sets 85

Title 85

Data Area 85

Polar Data Area 86

3D Data Area 86
Datagroup 86
Multiset Datagroup 86
Packed Datagroup 87
Line Datagroup 87
Multiline Datagroup 88
XY Line Datagroup 88
Multiline XY Datagroup 89
Filled Step Datagroup....89
Multiline Step Datagroup 90
Filled Multiline Step Datagroup 90
Pie Datagroup....91
Polar XY Line Datagroup 91
Multiline Polar XY Datagroup 91
3D Datagroup 92
Multiset 3D Datagroup 92
3D Packed Datagroup 93
Multiset 3D Packed Datagroup 93
Surface Datagroup 94
Level 95
Symmetrical Level 95
Polar Level 95
X Level 96
Y Level 96
Status Object 96
Value Axis 96
X Axis 97
Y Axis 97
Z Axis 98
X Value Axis 98
Y Value Axis 99
Y1 Value Axis 99
Y Value Multiaxis 99
Z Value Axis 100
X Time Axis 100

- Y Time Axis 101
- Tangent Ticks........ 102
- Tangent Labels 102
- Tangent Grid 102
- Radial Grid 102
- Legend 102
- Bevel 103

View Resources 103

- Viewing 103
- Zooming 103

Chapter 1

Using the GLG Widgets

1

The GLG Toolkit includes an extensive library of pre-built widgets that may be incorporated into a wide variety of applications. Each widget is a GLG drawing (".g" file) containing a collection of GLG graphical objects that implement the widget's graphical appearance and run-time behavior.

The widgets have no source code associated with them, and their functionality is completely defined in each widget's drawing. Widgets' drawings are cross-platform and may be used by GLG applications on various platforms and programming environments (C/C++, Java, C# / .NET, HTML5 & JavaScript, etc.) with no porting required.

Customizing GLG Widgets in the Graphics Builder

The widgets are highly customizable. A widget drawing may be loaded into the GLG Graphics Builder and edited as any other GLG drawing. Starting with the GLG release 4.2, all widgets have been redesigned to extensively use **public properties**, which greatly simplifies widget editing in both the Graphics Builder and HMI Configurator. The **Public Properties dialog**, accessed via the *Object, Public Properties* menu option or the *Public Properties* toolbar button, provides convenient access to the most commonly used widget properties.

The **Resource Browser** accessed via the *Object, Resources* menu option or the *Object Resources* toolbar button can also be used to modify all widget resources, including resources that are not accessible via public properties. The resource mechanism can be used to access widget resources via API at run time, and the Resource Browser can be used in the Graphics Builder to navigate a widget's resource hierarchy as described in the next section.

In addition to changing a widget's properties and resources, child components inside each widget can be edited directly by going "down" into the widget using the *Hierarchy Down* button of the Graphics Builder, selecting individual widget components and editing their attributes using the *Object, Properties* menu option or the *Properties* toolbar button. Each component's geometry can also be modified by using its control points.

While inside a widget, it can be customized by deleting any unnecessary elements and adding new custom elements, such as annotations, labels, additional active elements, etc. Custom dashboards may be created by combining several widgets into a single drawing, and new custom widgets can be created.

This chapter contains a general description of the categories of the GLG widgets: charts, meters, dials and other input widgets, process control and electrical symbols, as well 2D and 3D graph widgets.

Changing Widget Properties at Run Time via API

While the Public Properties dialog provides a convenient way to edit properties of a selected widget in the Graphics Builder, the GLG resource mechanism provides a universal access to all widget properties at run time. The GLG API provides a collection of *GetResource* and *SetResource* methods that can be used to query or change values of any widget property using a resource path to identify the widget and a resource inside it.

To identify widget resources that need to be accessed by the application code at run time, load the widget into the Graphics Builder and use the Resource Browser to browse the widget's resource hierarchy. The **Resource Browser** is accessed via the *Object, Resources* menu option or the *Object Resources* R toolbar button.

When a resource of interest is selected in the Resource Browser, the *Selection* field at the top of the Resource Browser dialog shows its resource path relative to the selected object. This resource path can be used in the GLG API methods to access the resource.

Using Scalable Text

Scalable widgets that contain text objects have the *TextScaling* resource that controls text scaling. If *TextScaling* is set to 1, the font size of the text will change when the widget size is increased or decreased. For the text scaling to work, the *BaseWidth* attribute of the parent viewport has to be set to a positive number that defines the default viewport size. When the viewport width changes, the text size will change by the ratio of the new viewport width to the *BaseWidth* value.

To make sure that text objects in all widgets in the drawing scale the same way, the *BaseWidth* should be set on the top viewport of the drawing instead of the viewports of the individual widgets.

Using Widgets in the Graphics Builder

Any widget may be loaded into the Graphics Builder by loading corresponding *".g"* files, located in one of the *widgets* directory subdirectories. However, the widget palettes are the most convenient way of adding the widgets to a drawing using the drag and drop functionality. To pop up a widget palette, click on the *Palettes* button of the main menu, then select the desired palette type.

Loading and Adding Widgets to the Drawing

There are two ways to load a widget drawing into the Graphics Builder:

- To add a widget to an existing drawing, simply click on the widget's icon in the palette. The widget will be added at the current location of the editing focus. When creating a panel containing several widgets in a *"$Widget"* viewport, use the *Hierarchy Down* button to go into the viewport and then add widgets to it; otherwise, the widgets will added at the top level outside of the viewport. When creating widget panels, individual widgets should be assigned unique names via the *Properties* dialog.
- To load the whole widget drawing into the Graphics Builder, *Ctrl-click* (click with the *Control* key held down) on the widget icon in the palette. This will discard the current drawing and replace it with the drawing of the selected widget. The widget drawing contains the widget, its icon and an animation command for prototyping the widget in the Run mode. The animation command is visible in the drawing as a text object.

Prototyping Widgets Using Predefined Animation Commands

All widgets may be prototyped using the Graphics Builder's Run mode. To start the Run mode, select *Run*, *Start* from the main menu or click on the *Start* toolbar button. This will activate a dialog for entering an animation command.

To simplify the process, all graph and control widget drawings include proper animation commands. To animate a widget with the animation command included in the widget drawing, *Ctrl+click* on the widget icon in the palette to load the widget's drawing, then start the Run mode and press the *OK* button in the animation command dialog to start the animation.

The Run mode toolbar provides controls for changing animation speed and pausing the animation. It also displays update performance data, such as updates per seconds and seconds per update.

For control widgets that can be used for both input and output, such as dials, sliders, toggles and other interface widgets, the Run mode may also be used to test the widget's interactive behavior. Press the *Pause* button in the Run mode toolbar to stop animating the widget with output data and use the mouse to interact with the widget. For example, you can move a dial or slider with the mouse or change the state of a toggle widget.

Prototyping Dials and Meters With Custom Animation Commands

Custom animation commands may be used to animate any dial or meter widget in the drawing. For example, the following command will provide a smooth animation of a dial by using a sinusoidal wave in the 0-100 range as a datasource for the widget's *Value* resource:

```
$datagen -sin d 0 100 $Widget/Value
```

$Widget/Value is a resource path for the resource that will be animated.

To enter the animation command, start the Run mode by selecting *Run*, *Start* from the main menu or clicking on the *Start* toolbar button. This will activate a dialog for entering an animation command. Enter the command and press *OK* to see animation.

If a dial is renamed or placed inside a panel containing several dials, the path may be different from the one used above. For example, the following path will animate a *Value* resource of a dial named *Dial2* placed inside a top level *$Widget* viewport:

$Widget/Dial2/Value

If the resource path is not correct, an error message will be generated. Use the Graphics Builder's Resource Browser (the *All Resources* toolbar button) to browse the drawing's resources and check the resource path.

To animate more than one dial, use several animation commands in one string. For example, the following command will animate dials in the drawing named Dial1 and Dial2:

```
$datagen -sin d 0 100 Dial1/Value
         -sin d 0  50 Dial2/Value
```

Refer to the *The Data Generation Utility* chapter on page 404 of the *GLG User's Guide and Builder Reference Manual* for a complete list of the animation command options.

Prototyping Real-Time Charts with Custom Animation Commands

Real-time charts are animated by supplying data sample values and time stamps into the chart's *ValueEntryPoint* and *TimeEntryPoint*. If a time stamp is not supplied, the chart will automatically use current time to generate a time stamp. The following prototyping command may be used for animating a real-time chart with a single plot:

```
$datagen -sin d 1 9 $Widget/Chart/Plots/Plot#0/ValueEntryPoint
```

A default animation command included in each chart widget's drawing provides a good starting point for experimenting with the chart's options. A modified animation command may be stored in the drawing using the *Run, Store Run Command* menu option described below.

Prototyping 2D and 3D Graph Widgets with Custom Animation Commands

Prototyping Graph Widgets in the Graphics Builder requires more elaborate prototyping commands that animate both the graph's labels and datasamples. If you want to use a custom animation command for a graph, use a default graph animation command defined in the graph's drawing as a starting point and customize it to explore various graph animation options. The modified animation command may be stored in the drawing using the *Run, Store Run Command* option from the main menu. The customized drawing may be saved for later reuse or added to the Builder's palettes.

Storing Custom Animation Command in the Drawing

The *Run, Store Run Command* menu option stores the current animation command by setting the *$DatagenString* resource of the drawing, if it is present. In most widget drawings, a text object at the top of the drawing displays the Run command. The *TextString* attribute of the text object is named *$DatagenString*. If a widget's drawing was loaded using *Ctrl*-click on the widget's icon in a palette, the *$DatagenString* resource is already present in the drawing; if a widget was added using a mouse click, the *$DatagenString* resource may be added to the drawing by placing a text object in the drawing and naming its *TextString* attribute *$DatagenString*.

Real-Time Chart Widgets

The Real-Time Charts widget set contains a variety of high-performance chart widgets. The charts are optimized for displaying a large number of lines with a huge number of data points. They also provide integrated zooming and scrolling functionality, tooltips and cursor feedback, as well as multiple Y axes, flexible tick labeling and many other features. Both time-based scrolling charts and XY Scatter widgets are provided.

The chart widgets are animated in the same way as any other GLG drawings, by setting their resources. Each chart has two types of resources: resources that control its appearance, such as plot colors, number of axis ticks, etc., and the entry point resources that are used to supply real-time chart data.

In addition to the real-time charts, legacy 2D and 3D Graph Widgets are also provided. The 2D and 3D graph widgets provide additional widget types, such as packed and stacked bar charts, as well as radial and pie charts. They also can use gradient shading and 3D rendering for presentation effects. The 2D and 3D Graph Widgets are described in the *2D and 3D Graphs* chapter on page 38.

Using Real-Time Chart Widgets

All real-time chart widgets use the GLG Chart object which implements all real-time and interaction features of each chart. The chart object contains integrated axes, plots, legend and other objects and has a number of attributes that control its behavior. Refer to the *Chart Objects* chapter on page 124 of the *GLG User's Guide and Builder Reference Manual* for a **detailed description of the chart's attributes and features**.

In addition to chart properties, a *Chart* object of every real-time chart widget provides *OffsetTop*, *OffsetBottom*, *OffsetLeft* and *OffsetRight* resources that control layout of the chart inside the widget.

An application supplies data for a chart using the chart's *ValueEntryPoint*, *TimeEntryPoint* and *ValidEntryPoint* described in the the *Chart Objects* chapter. An application can either provide a time stamp for every data sample, or let the chart use current time as a time stamp. The chart automatically positions and scrolls its data samples according to their time stamps. The chart also supports the use of invalid data samples, which are displayed as gaps in the plot lines.

A real-time chart maintains a data buffer of a variable size that keeps data samples accumulated in the chart. Users can zoom and scroll the chart in both vertical and horizontal direction using the chart's integrated zooming and scrolling features. The chart can also be scrolled by simply dragging it with the mouse.

The integrated zooming and scrolling behavior of the chart is controlled by the *ZoomMode* and the *Pan* attribute of the chart's parent viewport. Refer to the *Integrated Zooming and Panning* chapter of the *GLG User's Guide and Builder Reference Manual* on page 230 for more details.

The chart's integrated tooltips and cursor feedback features may be used to display a cross-hair cursor that follows the mouse, as well as display a tooltip showing the information about the data sample located under the current mouse position. The chart object also provides an API for querying chart selection from a program.

To activate the chart's tooltip, the *ProcessMouse* attribute of the chart's parent viewport must be set to TOOLTIP. This is the default for chart widgets in the Real-Time Charts palette.

Customizing Real-Time Chart Widgets

The *Chart* object located inside the chart widget viewport contains most of the chart properties. While the chart object attributes can be accessed by going "down" into the viewport and selecting the chart, the *Edit Chart* button of the Public Properties dialog (accessed via the *Object, Public Properties* menu option) provides a convenient access to the attributes of the chart without the need to go "down" into the chart's viewport. Refer to the *Chart Objects* chapter of the *GLG User's Guide and Builder Reference Manual* on page 124 for a **detailed description of the chart's attributes**.

The Public Properties dialog also has buttons for accessing the chart's background, legend and title properties, as well as the offsets that control the chart's layout. The following buttons may be present:

Background contains color and other rendering properties of the chart widget's viewport.
Edit Chart accesses attributes of the chart object inside the widget viewport.
Edit Legend accesses attributes of an optional chart legend inside the widget viewport.
Title accesses attributes of the chart title inside the widget viewport.
Offsets contains offset properties that control the chart widget layout.

Refer to the *Editing Chart Properties in the Builder* chapter on page 80 of the *GLG Builder and Animation Tutorial* for detailed information on **editing real-time chart widgets**.

Input Widgets

Input widgets include widgets that react to the user input, such as keyboard or mouse interaction. An interaction handler attached to an input widget's viewport handles the widget's user interaction. The widgets may be used for both input and output. To disable the widget's interactive capabilities and use it only for the output, delete the widget viewport's input handler.

Most of the input widgets, such as dials and meters, sliders, knobs and buttons, are grouped into the **Controls Widget Set**. The Controls Widget Set also includes slider-based linear indicators, output-only widgets, such as value display widgets (described in the *Value Display Widgets* section on page 26), state indicators and avionics gauges. Some other input widgets from the **Special Widget Set**, such as menus and checkboxes, spinners, text input and various browser widgets, are also described in this section.

The **meter** and **dial widgets**, as well as **knobs, switches, sliders** and **linear gauges** have an input handler attached and will react to the mouse interaction, such as moving a dial's needle with the mouse. To use these widgets for output only, delete the interaction handler attached to a widget.

The type of the input handler attached to the widget defines the widget type, such as a dial, a slider, a knob, etc. The type of the used input handler also defines resources present in the widget. Refer to the *Input Objects* chapter on page 255 of the *GLG User's Guide and Builder Reference Manual* for detailed information on resources of various input handlers.

Note that the input handlers attached to an input widget expects the widgets to have several predetermined resources depending on the type of the input handler. Altering the names of these resources may render the widget inoperable, and may generate an error message when the drawing is reset.

At run-time, the application's Input callback receives a message when a widget receives some input. Refer to the *Input Objects* chapter on page 255 of the *GLG User's Guide and Builder Reference Manual* for information on the message parameters of various input handlers.

Refer to the *Appendix B: Message Object Resources* chapter on page 461 of the *GLG User's Guide and Builder Reference Manual* for information on the input actions for various types of input handlers.

Meters, Dials and Avionics Gauges

All meter, dial and avionics widgets use the *Angular Axis* object that renders each widget's axis with ticks and labels. Refer to the *Angular Axis* section on page 155 of the *GLG User's Guide and Builder Reference Manual* for a detailed description of the angular axis' attributes and features. The axis can be accessed by going "down" into the widget, but most of its properties are more conveniently accessed via the Public Properties dialog.

The *GlgKnob* input handler is attached to the viewport of the meter and dial widgets, enabling the widgets to handle both input and output. If the widget's input handler is not disabled, the dial's needle can be moved with the mouse. To disable the widget's interactive capabilities and use it only for output, delete the widget viewport's input handler. The widgets inherit resources of the *GlgKnob* interaction handler. Refer to the the *GlgKnob* section on page 264 of the *GLG User's Guide and Builder Reference Manual* for a **list of the interaction handler's resources**.

The widgets also have an *ActiveElement* resource, which is an object used to render the dial's needle. The active element may have the *Truncate* resource that controls how input values that are out of range are displayed. When it is set to ON, the value is truncated to fit into the High-Low range of the widget. When the value is truncated, the active element is positioned at either the start or the end of the scale, depending on the input value.

Dial and meter widgets may have an optional *ScaleArea* resource, which is an object used to render the dial background.

Some of the avionics gauges use two sets of needles and/or two sets of angular axes.

The *Palettes, Make Widget Viewports Transparent* menu option may be checked to **make widget viewport transparent**. For example, a transparent dial widget may be used as a non-rectangular dial on top of a custom background image.

Customizing Meter and Dial Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the most commonly used widget properties.

The most important properties, such as *High* and *Low* ranges, *Value* and *Units*, are listed at the top level of the dialog. The rest of the properties are grouped into categories, with dialog buttons to access them. The following categories may be present:

Colors contains color properties of various widget elements.
Ticks contains properties that control the number of axis ticks and their sizes.
TickLabel
UnitLabel
ValueLabel

> Contain properties of the corresponding labels.

BandColors contain properties that control the number of color bands and their colors.

BandThresholds contains thresholds that control the start and end position of each color band. If the *AbsoluteRange* property is set to 0 (default), the thresholds are interpreted as percentages of the dial's range. If it is set to 1, the thresholds are interpreted as absolute values.

Knob Widgets

A **Knob Widget** converts an angular mouse position into a value, providing visual feedback in the form of the angular position of the knob's active element. When the left mouse button is pressed over the knob widget, the knob reacts by moving its active element to the mouse position. If the left mouse button is held down and dragged, the active element of the knob follows the mouse until the button is released.

A knob may also be used for output. When used this way, the widget converts a scalar value (specified with the *Value* resource) into the position of its active element. If the widget is used only for output, you may delete the input handler of the widget's viewport.

There are several different versions of knob widgets in the GLG Widget Library, each with a different look and feel. Each uses an object called *ActiveElement* to provide the visual feedback for the current knob value. The range of positions of the active element is associated with the range of the knob values, defined by the *Low* and *High* resources of the knob. Some knobs also have an indicator that shows the current knob value as a number.

The active element's *Truncate* resource controls how out of range values are displayed. By default, it is set to ON to truncate out of range values to force them in the range, so that the knob does not rotate outside its start and end positions.

A knob widget may also have control buttons for increasing or decreasing its value. A user may click on these buttons to move the knob pointer by a preset distance in some direction.

Same as the meter and dial widgets, knob widgets inherit resources of the *GlgKnob* interaction handler. Refer to the the *GlgKnob* section on page 264 of the *GLG User's Guide and Builder Reference Manual* for a **list of the interaction handler's resources**, as well as the names of optional increase and decrease buttons that may be present inside a knob widget.

Customizing Knob Widgets

The knob widgets have resources and properties similar to the dial and meter widgets. Refer to the *Customizing Meter and Dial Widgets* section on the previous page for information on editing knob widget properties.

Slider and Linear Gauge Widgets

A **Slider Widget** converts a linear mouse position into one or two scalar values, and provides some appropriate visual feedback by moving its active element. When the left mouse button is pressed over the slider widget, the slider reacts by moving its active element to the mouse position. If the left mouse button is held down and dragged, the active element of the slider follows the mouse until the button is released.

There are several different versions of slider widgets in the GLG Widget Library, each with a different look and feel. There are also two-dimensional versions that control two variables, one corresponding to the X coordinate, and the other corresponding to the Y coordinate. For one-dimensional sliders, there are horizontal and vertical versions of each slider. Native sliders and scrollbars are also provided as separate widgets listed in the next section.

The slider contains *ValueX* (for the horizontal slider) or *ValueY* (for the vertical slider) resource which contains the slider's value and is set by the slider each time its knob is moved. The two-dimensional slider contains both *ValueX* and *ValueY* resources, while horizontal and vertical sliders contain only one of them, depending on the slider direction. For convenience, one-dimensional sliders contain a *Value* resource which is an alias mapped to either *ValueX* or *ValueY.*

An object called *ActiveElement* is a slider thumb that provides a visual feedback for the current slider value. The range of positions of the active element is associated with the range of the slider values, defined by the *Low* and *High* resources of the slider. The active element's *Truncate* resource controls how out of range values are displayed. By default, it is set to ON to truncate the out of range values to force them in the range, so that the slider's knob does not move outside its start and end positions.

Some sliders also have an indicator showing the current slider value in digital form, as well as resources that control thumb size, named *SliderSize* or *LengthCoeff*. A slider may also have buttons. A user may click on these buttons to move the slider pointer the distance specified by the *Increment* resource in some direction.

Slider widgets inherit resources of the *GlgSlider* interaction handler. Refer to the *GlgSlider* section on page 259 of the *GLG User's Guide and Builder Reference Manual* for a **list of the slider resources** as well as names of buttons that may be placed inside the slider.

Some slider widgets use the *Axis* object to display an axis next to the slider track. Refer to the *Axis* section on page 143 of the *GLG User's Guide and Builder Reference Manual* for a **list of the axis object resources**.

A slider may also be used for output, displaying a value of its Value resource. When the slider's *ValueX*, *ValueY* or *Value* resources are set, the slider positions its active element to show the new value. If the widget is used only for output, you may delete the input handler of the widget's viewport.

A spinning slider is a combination of a slider and a spinner. In addition to the slider element, it also contains buttons for changing its value, as well as a text object that displays the current value and allows entering a new value by typing. The spinning slider inherits additional properties of the *GlgSpinner* interaction handler. Refer to the *GlgSpinner* section on page 271 of the *GLG User's Guide and Builder Reference Manual* for a **list of the spinner resources**.

Linear gauges may have the *ActiveElementSeries* series object containing stacked colored bars and the *LevelSeries* object containing level lines.

Customizing Slider Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the most commonly used widget properties.

The most important properties, such as *Value* and *High* and *Low* ranges, are listed at the top level of the dialog. The rest of the properties are grouped into categories, with dialog buttons to access them. The following categories may be present:

Thumb contains properties that control rendering of the slider's active element.

Track contains properties that control rendering of the slider track, which shows the range of the slider movement.

Tip contain properties of the thumb tip, if present.

ValueLabel and **TickLabel** contain properties of the corresponding labels, if present.

Ticks contains properties that control the number of axis ticks and their sizes for sliders with an axis.

Linear gauge widgets may contain the following additional categories:

Bar contains rendering properties of the gauge's bar.

BarColors contains properties that control number of colored bars, their thresholds and colors.

BarBox contains rendering properties of the background box.

Offsets contains offsets that control geometry of the widget's elements.

LevelColors contains properties that control number of level lines, their position and colors.

Level contains properties that define line attributes of level lines.

Native Slider and Scrollbar Widgets

Native slider and scrollbar widgets use the corresponding native widgets of the underlying deployment platform. They contain a minimal number of resources inherited from their *GlgNSlider* interaction handler described in the *GlgNSlider* section on page 271 of the *GLG User's Guide and Builder Reference Manual*.

In addition to the *Low* and *High* ranges, they contain a single *Value* resource regardless of the slider direction. The *SliderSize* resource controls the length of the scrollbar's thumb and may be set to a value in the range from 0 to 1.

Customizing Native Slider and Scrollbar Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the widget properties.

The *BackgroundColor* resource may be used to define the widget's color when the widget is deployed in the C/C++ environment on Linux/Unix platforms. In the rest of the environments, the widget's color is controlled by the platform's native environment.

Button and Toggle Widgets

A **Button Widget** is a widget activated with a mouse press. Buttons can be **push buttons** that produce some action when they are pushed, but do not have a "state," or **toggle buttons**, that have an internal state (*OnState* resource). Native push button and toggle widgets are also provided and are described below.

The type and behavior of different buttons are based on the resources present in the button’s viewport. The *ActOnPress* resource setting controls if the button is activated on mouse press or mouse release. If the *ArmedState* resource (*ActivateOnCtrl* property) is present and has a value different from -1, the button will react to the mouse interaction only if the *Control* key is pressed. If the *DelayInterval* resource is present and has a value greater than 0, the button will be activated only if it is kept pressed for the number of seconds specified by the resource.

Button widgets inherit resources of the *GlgButton* interaction handler. Refer to the *GlgButton* section on page 266 of the *GLG User’s Guide and Builder Reference Manual* for a **list of the button resources**.

Customizing Button Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the most commonly used widget properties.

The most important button properties, such as *LabelString, TooltipString* are listed at the top level of the dialog. The *ActOnPress* and *ActivateOnCtrl* properties may be used to change button behavior as described in the previous section. For the toggle buttons, the *Value* property provides access to the toggle state (*OnState* resource).

The rest of the properties are grouped into categories, with dialog buttons to access them. The following categories may be present:

Colors contains properties that control button colors and gradients.
Label contains button label properties.
Checkmark contains properties control rendering of a toggle button checkmark.

Native Button and Toggle Widgets

Native button and toggle widgets use the corresponding native widgets of the underlying deployment platform. They contain a minimal number of resources inherited from their *GlgNButton* interaction handler described in the *GlgNButton* section on page 268 of the *GLG User’s Guide and Builder Reference Manual*.

Customizing Native Button and Toggle Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the widget properties.

Same as the button and toggle widgets described above, native buttons and toggles have *LabelString, TooltipString* properties. Native toggle buttons also have the *Value* property that provides access to the toggle state (*OnState* resource).

The *BoxColor* resource may be used to define the widget’s color when the widget is deployed in the C/C++ environment on Linux/Unix platforms. In the rest of the environments, the widget’s color is controlled by the platform’s native environment.

Menu Widgets

A **Menu Widget** is an array of button widgets. The menu manages its buttons, allowing the application to manage an array of buttons as a single object. The menu can contain either push buttons or toggle buttons, provided that all menu buttons are of the same type.

A radio menu contains the *SelectedIndex* resource which is set to the index of the selected toggle button.

The use of the *GlgMenu* input handler and its resources are described on page 275 of the *GLG User's Guide and Builder Reference Manual*.

Customizing Menu Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the widget properties.

The *NumRows* and *NumColumns* properties control the number of buttons in the menu, and *BackgroundColor* specifies the color of the menu box. For radio menus, *InitSelectedIndex* controls which menu button will be selected on the initial appearance (when the widget is loaded or the drawing is reset).

The rest of the properties are grouped into categories, with dialog buttons to access them. The following categories may be present:

ButtonColors contains properties that control button colors and gradients.
ButtonLabel contains properties that specify text attributes of the button labels.
LabelStrings contains properties that specify button label strings.
Offsets contains properties that specify menu layout offsets.
Tooltips contains properties that specify button tooltips.
Checkmark contains checkmark rendering properties for menus with toggle buttons.

Tabs Widgets

A **Tabs Widget** is a specialized version of a radio menu widget that has can be used to switch between drawings displayed on a page. The widget inherit resources of a radio menu widget described above, including the *SelectedIndex* resource.

The widget also has a *GoTo* command action attached, which is activated when a tab is selected, invoking the application Input callback. An application can use the transformed value of the *DrawingFile* resource to load a drawing corresponding to the selected tab. The transformed value of the *DrawingFile* resource can be accessed from the message object in the Input callback using the *"Action/Command/DrawingFile/XfValue"* resource path.

The command's *DrawingFile* resource can also be constrained to the *SourcePath* resource of a *SubWindow* object to display a drawing corresponding to the selected tab without a need to handle the *GoTo* action in the program. If this technique is used, the *GoTo* action may be disabled to avoid interfering with other GoTo actions handled by the application.

Customizing Tabs Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the widget properties.

The *NumTabs* property defines the number of tabs, and *BackgroundColor* specifies the background color. The *InitSelectedIndex* property controls which tab will be selected on the initial appearance (when the widget is loaded or the drawing is reset).

The rest of the properties are grouped into categories, with dialog buttons to access them. The following categories may be present:

DrawingFiles contains properties that specify drawing files corresponding to each tab selection.

DrawingTitles contains a list of properties that specify drawings' titles.

Label contains properties that specify text attributes of the tab labels.

LabelList contains properties that specify a label string for each tab.

Tab contains properties that specify tab attributes.

Tooltips contains properties that specify a tooltip for each tab.

Text Widgets

A **Text Widget** used for entering a text string. There are versions of the text widget for entering numerical values, both integer and floating point. The widget uses a native text entry widget of the underlying deployment platform and adds additional functionality for entering numerical values with range validation.

The text widget inherit resources of the *GlgNText* interaction handler. Refer to the *GlgNText* section on page 269 of the *GLG User's Guide and Builder Reference Manual* for a **list of the text widget resources**.

Customizing Text Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the widget properties.

The *TextString* property (or *PasswordString* for the password text widget) specifies the text string that will be displayed when the widget is loaded. The *SelectAllOnFocus* property controls the widget's selection policy when it receives focus. If set to 1, all text in the widget will be selected when the user clicks on the widget or moves the focus into it by using the TAB key.

The *BackgroundColor* and *BorderColor* properties control widget colors when the widget is deployed in the C/C++ environment on Linux/Unix platforms. In the rest of the environments, the widgets' colors are controlled by the platform's native environment.

The numerical input text widgets also have additional *Value*, *MinValue* and *MaxValue* properties that defined the initial value on load and the accepted range of values. If entered value is outside the range, the color of the widget will change to indicate an error. The *EnforceRange* property may be set to 0 to allow entering values outside the range, otherwise the entered value will be forced inside the range when the user finishes entering the value. For floating point input the *ValueFormat* property specifies a C-style format for displaying the value.

The *Color* button provides access to properties that specify error colors. The color used when the widget has no errors (*ColorNoError*) is used only when the widget is deployed in the C/C++ environment on Linux/Unix platforms. In the rest of the environments, this color is controlled by the platform's native environment.

The *InputFormat* property specifies the type of the text input.

Spinner Widgets

A **Spinner Widget** is used for entering a numerical value by either entering a new value in a text object, or using arrows to increase or decrease a current value. There are several versions of spinners for entering integer or floating point values.

A spinner widget contains a numerical input text box, as well as increase and decrease buttons. Spinner resources are similar to those of the numerical text input widget described in the previous section, with addition of optional *Increment* and *PageIncrement* resources that specify the amount to adjust the value by the *Increment /Decrement* or *PageIncrement / PageDecrement* buttons.

The spinner widget inherit resources of the *GlgSpinner* interaction handler. Refer to the *GlgSpinner* section on page 271 of the *GLG User's Guide and Builder Reference Manual* for a **list of the spinner widget resources**.

A spinning slider described on page 19 is a combination of a spinner with a slider for changing the value with the mouse.

Customizing Spinner Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the widget properties.

Similar to the numerical text input widget described in the previous section, the spinner has the *Value*, *MinValue* and *MaxValue* properties that define the initial value on load and the accepted range of values. For floating point spinners, the *ValueFormat* property specifies a C-style format for displaying the value. The *Wrap* property controls wrap around when the value adjusted using the buttons reaches the minimum or maximum range limit. If the property is set to 0, wrap around is disabled.

The *BackgroundColor* property control widget background color.

The rest of the properties are grouped into categories, with dialog buttons to access them. The following categories may be present:

Color contains properties that specify error colors. The color used when the widget has no errors (*ColorNoError*) is used only when the widget is deployed in the C/C++ environment on Linux/Unix platforms. In the rest of the environments, this color is controlled by the platform's native environment.

Buttons contains button color properties.
ButtonLabels contains button labels.

List and Option Menu Widgets

The **List** and **Option Menu Widgets** provide access to the corresponding native list, option menu and combo box widgets. Refer to the *GlgNList* section on page 272 and the *GlgNOption* section on page 274 of the *GLG User's Guide and Builder Reference Manual* for information on resources of these widgets.

Clock and Stopwatch Widgets

The **Clock** and **Stopwatch Widgets** display the current time or the elapsed time. Both widgets use the same input handler. The handler takes care of updating the time and reacting to the input events for the stopwatch. The Clock Widget continues updating even while being edited.

The Clock Widget does not react to input events from a user and serves to display the current time only. If the *TimerState* resource is present, the clock measures the elapsed time instead of real time, and the widget becomes a stopwatch. The Stopwatch Widget can receive input through the *Start*, *Stop* and *Reset* buttons contained in the clock viewport to control the stopwatch.

If the input handler of the Clock Widget is deleted or disabled, the widget may be used to display user-defined time under control of an application program. In this case, the application program uses the widget's *Hour*, *Min* and *Sec* resources to drive the clock.

Refer to the *GlgClock* section on page 279 of the *GLG User's Guide and Builder Reference Manual* for the **list of the widget resources**.

The widgets use an angular axis object (an *Axis* resource) to render the dial of the clock. Refer to the *Angular Axis* section on page 155 of the *GLG User's Guide and Builder Reference Manual* for a list of the angular axis' attributes.

The hands and other elements of the widgets may be edited the same way as any other object.

Customizing Clock and Stopwatch Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the widget properties.

The following property categories may be present, with dialog buttons to access them:

Colors contains color properties of clock hands and other widget elements.
TickLabel contains properties that control rendering of the tick labels used to render dial numbers.
Ticks contains properties that control the number and size of the dial ticks.
Buttons contains rendering properties of the stopwatch buttons.
TimeLabel contains rendering properties of the stopwatch time label.

Palette Widgets

A **Palette Widget** is used for creating different palettes with which to present a user with choices of objects. It may be used to create different types of palettes, from palettes for selecting object attributes, such as colors, line widths or font sizes, to palettes of icons. The widget is designed to be used within a program using the GLG API.

A palette widget has a resource object named *PaletteObject*, which is a container object (a group or a series) with several members, each corresponding to a different possible choice. When the user selects one of the objects in the container with the left mouse button, the chosen object is returned to the program in the callback message object. The returned object is indicated with the *SelectedObject* resource of the message object.

Refer to the *GlgPalette* section on page 279 of the *GLG User's Guide and Builder Reference Manual* for more information.

Resource, Tag, Alarm and Font Browser Widgets

The **Resource Browser, Tag Browser** and **Alarm Browser Widgets** provide the application with the resource, tag and alarm browsing functionality similar to the one in the GLG Graphics Builder.

The **Font Browser Widget** is used to facilitate the font selection from the list of all available fonts. There are two font browser widgets: the XFT Font Browser for the FreeType XFT fonts and the Font Browser for the core X11 (XLFD) fonts used by the legacy X11 version of the Toolkit. The font browser widgets are only supported in the C/C++ environment on Linux/Unix. A native font browser can be used on Windows.

Refer to the *GlgBrowser* section on page 277 of the *GLG User's Guide and Builder Reference Manual* for more information on the browser widget. Refer to the *GlgFontBrowser* section on page 278 of the *GLG User's Guide and Builder Reference Manual* for more information on the font browser widget.

Value Display Widgets

Value Display widgets are used to display a value together with a label and measurement units. There are several value display widgets that have different layout of their components. There are also two variants of most value display widgets: fixed size and scalable.

Fixed size widgets have parameters that define their width and height in screen pixels, and their size does not change when the drawing is zoomed or resized. Fixed size widgets also have parameters that define position of their attachment point.

The size of scalable widgets is defined by their control points and can be adjusted using the mouse. Scalable widgets scale with the drawing when the drawing is zoomed or resized. The widget's *TextScaling* parameter described in the *Using Scalable Text* section on page 12 may be set to scale text objects inside the widget.

Most value display widgets can also be used to display status of the corresponding equipment by changing colors depending on the value of the *AlarmStatus* resource. The *NumStatusColors* resource may be used to change the number of available color states.

Customizing ValueDisplay Widgets

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the widget properties.

The most important properties, such as *Value*, *Description, Units* and *AlarmStratus* are listed at the top level of the dialog. The *ValueFormat* property specifies a C-style format used to display the widget's value. Optional separator properties that may be used to specify spaces or other separators between the value and its unit or description label may also be included.

The rest of the properties are grouped into categories, with dialog buttons to access them. The following categories may be present:

Colors contains widget color properties. There are several background and value label colors (*BGColor0, BGColor2*, etc., and *ValueColor0, ValueColor2*, etc.) that are used with corresponding *AlarmStates* values. The *NumStatusColors* property defines the number of alarm states and the corresponding colors.

DescriptionLabel
UnitLabel
ValueLabel
Contain text properties of the corresponding text objects.

Size & Alignment contains properties that control widget layout (the proportion of space used by each label and an optional color indicator). For fixed value display widgets it also contains properties that specify widget dimensions and position of the anchor point.

Font contains font type and font size properties for widgets with a single text object.

Date and Time Display Widget

The **Date and Time Display Widget** displays current date and time, updating it every second. Its properties are similar to the rest of the value display widgets, with additional properties that specify the format of the displayed date and time:

DateFormat
TimeFormat
specify the date and time formats. Refer to the description of the axis object's *Time Format* attribute on page 146 for information on the supported time and date formats.

UTC Flag may be set to 1 to show UTC time instead of local time.

Lat/Lon Display Widgets

Lat/Lon Display Widgets display a latitude or longitude value (or both), converting it from decimal degrees, minutes and seconds to sexagesimal. For example, a latitude value -40.504167 will be displayed as 40º30'15"S.

Depending on the widget, the latitude and longitude values are supplied via either separate resources, or a single resource of G type, with its X coordinate interpreted as longitude and Y as latitude.

The lat/lon display widget properties are similar to the rest of the value display widgets, with additional properties that specify lat/lon values and their description labels.

LED Widgets

The **LED Widget** displays a digit in the form of a 7-segment display.

In LED widgets with a bright background, the LED widget's *LedColor* and *LedEdgeColor* properties specify colors used to render the active segments, and inactive segments are turned invisible. In LED widgets with a dark background, the *LedOnColor* and *LedOffColor* properties specify colors of the active and inactive segments. The *Value* property specifies the digit to be displayed.

In multi-digit LED widgets the *NumDigits* property specifies the number of LED digits to display. LEDs of individual digits are named Led0, Led1, and so on. The *Value* resource of each LED may be used at run time to supply the digit to be displayed in each LED.

State Indicator Widgets

State Indicator widgets change their color depending on the incoming value.

The Public Properties dialog accessed via the *Object, Public Properties* menu option provides convenient access to the following widget properties:

Value specifies the incoming value.
NumColors specifies number of colors.
Colors button provides access to the widget colors.

Process Control Symbols

The process control symbol palettes provide the user with an easy way to create process control drawings by simply selecting the pre-built objects from a palette. Most of the objects have a built-in dynamic behavior to be used for displaying values or animation.

Tank and valve widgets have a resource named *Value*, as well as *Low* and *High* resources that specify the value range. For example, setting the *Value* of a tank object to a value in the range from *Low* to *High* changes the level of the tank's filling. In **valve widgets**, the *Value* resource controls its opening. The *Value* resource of the **process control indicators** controls the value displayed in the widget.

Fans, pumps and other dynamic equipment have an attached animation dynamics that is controlled by the *Enabled* resource. The *Interval* resource specifies the animation timer interval in seconds, which controls the rate of animation.

Each process control widget also has other resources to control its appearance, depending on the object type. The resources may include colors, line and text attributes, as well as geometry parameters. Some attributes, such as gradient coefficient, are relative in the range from 0 to 1.

Most of the process control symbols have several attachment points which can be used for attaching pipes or lines. For example, a tank has several connection points that can be used to attach pipes connected to the tank. When creating a pipe in the HMI Configurator, the tank’s attachment points are automatically highlighted when a mouse moves over the tank. In the Graphics Builder, attachment points are highlighted when constraining mode is activated while creating a pipe.

The process control symbols also include connector-style objects such as various pipes and lines. The connector-style objects have multiple control points which define their shape. Some of the lines, such as pneumatic or hydraulic line, have various elements replicated along the line and have the *NumElements* property that controls the number of elements.

Customizing Process Control Symbols

When a process control widget is selected in the drawing, the Public Properties dialog, accessed via the *Object, Public Properties* menu option, provides convenient access to the widget properties. To reduce cluttering, some widgets’ properties are grouped into categories, with dialog buttons for accessing properties in each category. The following sections describe properties of various process control widgets.

The Resource Browser may be used to list all resources of the selected process control object that can be used via API at run time. Tags may be assigned to the resource names to facilitate connecting dynamic data to the objects. Refer to the *Using Tags* chapter on page 321 of the *GLG User’s Guide and Builder Reference Manual* for more information about tags.

The following sections provide information on different types of process control widgets and their properties, accessible via the Public Properties dialog.

Tank Widgets

The **Tank Widget** displays a tank filled with liquid whose level changes depending on the supplied value. Some tanks include a label that displays liquid level as a numerical value. There are also tanks that use a separate Level Window object with an axis to display the tank’s liquid level.

Tank widgets have *Value*, *Low* and *High* properties that control the tank level and the range of values of the *Value* resource. For tanks with a separate Level Window object, the *Value*, *Low* and *High* properties are the properties of that object and can be accessed when the Level Window object is selected.

The rest of the properties are grouped into categories, with dialog buttons to access them. The following categories may be present:

Base
Body
Fill
Float
Contain properties of the corresponding tank elements.

TankColors contains tank color properties.

TankGeometry
TankShape
Contains properties that control tank geometry.

Tank Value Label Properties

Tanks that contain value labels displaying the tank fill level have additional **properties of Value Display widgets** described on page 26.

In some tanks, the value label is integrated with the tank object and moves up and down with the liquid level. The high and low range of the value displayed in the label may differ from the high and low range of the value that controls the tank level. For example, the tank level value may be supplied in the range from 0 to 1, while the displayed value should be displayed as percentage in the range from 0 to 100. The *DisplayLow* and *DisplayHigh* properties specify the range of the value displayed in the label and may be set to values different from the *Low* and *High* ranges. In our example, *DisplayLow* and *DisplayHigh* may be set to 0 and 100, while *Low* and *High* may be set to 0 and 1.

Same as the Value Display widgets, the value label can change color to indicate alarm condition. The *AlarmState* property controls the colors used by the label, and the *ValueColors* button provides access to the colors used to indicate alarm condition.

In some other tanks the label is a separate object that can be positioned with the mouse, while it is still connected to the tank's level value. The label properties can be accessed when the label is selected. These tanks also have a separate Level Window object that can be resized and positioned independently of the tank. The properties of the Level Window are described in the next section.

Both the tank label and the level window can display independent alarm conditions using separate variables and different colors. The level window indicates an alarm by changing color of the filled liquid, for example to indicate a temperature alarm. The label may also change its color to indicate the level alarm when the level is too high or too low.

If a level window is present, the label's alarm status resource is named *LabelAlarmStatus* and the alarm status of the level window is named *LevelAlarmStatus* to avoid name collision.

Level Window Properties

The Level Window object has the *Value*, *High* and *Low* properties that control the liquid level and the range of its values, as well as the *LevelAlarmStatus* property that controls colors used to display an alarm condition

The rest of the properties are grouped into categories, with the following dialog buttons to access them:

LevelWindow contains the level window color property.

LevelSlider contains properties that control the geometry and colors of the slider used to display liquid level. The *NumStatusColors* property specifies the number of colors used to display alarm conditions.

LevelLines contains properties controlling position and color of level lines used to display the low and high liquid levels.

Ticks contains properties that control the number of major, middle and minor ticks as well as their sizes and color.

TickLabel contains tick label properties.

Column Widgets

Column Widgets are static objects that do not have any dynamics. The Public Properties dialog for these widgets contains self-documented properties that control their attributes, such as *FillColor*, *EdgeColor*, *GradientCoefficient*, etc.

For a column containing plate or donut elements inside them, the properties are grouped into categories to separate the column and element properties.

Heat Exchanger Widgets

Heat Exchanger Widgets change their color to indicate their temperature. The *Value*, *High* and *Low* properties supply the temperature value and its range. When the *Value* changes from *Low* to *High*, the fill color of the widget gradually changes from black to the color specified by the FillColor resource of the widget's *Body.*

The **Body** button of the Public Properties dialog provide access to the properties of the heat exchanger's body. The **Element** button accesses properties of the heating element. Other buttons, such as **Pipes** or **Spray**, access properties of miscellaneous heat exchanger elements depending on the symbol type.

Valve Widgets

Valve Widgets represent various types of valves used in the process control equipment. Each valve has dynamics that changes the valve appearance to indicate its state, open or closed. The *Value*, *High* and *Low* properties supply the valve's opening value and its range. Control valves gradually open as their *Value* changes from *Low* to *High*, while on-off valves have only two states: closed or open depending on their *Value*.

The rest of the valve properties are grouped into categories, with dialog buttons to access them. The **Colors** button provides access to the valve color properties, the rest of the buttons depend on the valve type and provide access to the rendering properties of the valve's elements.

Process Control Equipment

These widgets include **motors, pumps, fans** and other **process control equipment** widgets. The Public Properties dialog of these widgets contains buttons for accessing rendering properties of individual widget elements, such as **Body, Drum, Plates**, etc.

For widgets with animation dynamics, the **Animation** button of the dialog provides access to the following animation properties:

Enabled controls animation state. Setting it to 0 disables animation.
Interval controls the rate of animation by specifying an animation timer interval in seconds.
Inversed controls animation direction. Setting it to 1 reverses animation direction.
Period specifies how many iterations it takes to fulfill one complete animation cycle.

Conveyor Widgets

The conveyor widgets have elaborate animation that simulates movement of the conveyor belt and the items on top of it. In addition to the **Enabled** parameter, the widgets have additional **Animation** parameters that can be used to enable or disable animation of individual elements:

BeltEnabled controls belt animation. If set to 0, the belt will be rendered as a solid line, otherwise it will animate the belt using line type dynamics.

BlocksEnabled controls animation of the blocks on top of the belt. If set to 0, the blocks will not be moving.

The **Blocks** button accesses parameters of the blocks positioned on top of the belt, including the number of blocks, their length and height, as well as their rendering attributes. The *BlockVisibility* parameter may be used to hide the blocks.

The **Geometry** button accesses parameters that define conveyor geometry, such as length, wheel radius, slope angle, etc. The **StartOffset** parameter controls position of the first block on the belt. The **FlipDirection** parameter can be used to change direction of the animation flow.

The **Belt**, **Spoke** and **Wheel** buttons access rendering attributes of the corresponding conveyor elements.

Process Control Gauges and Indicators

These widgets are similar to Value Display widgets and have similar properties. The *Value* property specifies the value to be displayed, and the *State* property specifies the widget’s alarm state that causes the widget to change its background color.

The rest of the properties are grouped into categories, with dialog buttons to access them. The following categories may be present:

Box
Body
Contain rendering properties of the gauge body.
StateColors contains alarm color properties.

ValueLabel
UnitLabel
Contain rendering properties of the corresponding labels.

Depending on the indicator type, the Public Properties dialog may also contain buttons for accessing rendering properties of miscellaneous widget elements.

Pipes and Lines

This palette includes a variety of pipes and lines, as well as bend and crossing symbols. The pipes and lines have multiple control points. To create a pipe or a line, click on the desired palette icon and select several points in the drawing to define the objects' shape.

Some of the lines, such as pneumatic or hydraulic line, have various elements replicated along the line and have the *NumElements* property that controls the number of elements.

The Public Properties dialog for these widgets has properties for editing symbol's rendering attributes. For composite symbols, the dialog contains buttons for accessing rendering properties of their elements, such as **Body, Line, Dot, Element, Anchor, Arrow**, etc.

3D Pipes

3D pipes use gradient shading to create a 3D effect. Some pipes have an animated dashed line used to indicate the flow inside the pipe, while other pipes use color to indicate the flow or some other state condition. The palette also contains matching elbows and flanges that can be used with the pipes.

The pipe's *LineWidthScaling* property may be set to 1 to change the pipe's line width when the drawing is resized. Line width scaling works the same way as text scaling described on page 12 and requires setting the *BaseWidth* attribute of the parent viewport.

OpenGL Note: When the drawing containing pipes is rendered using the OpenGL driver in the C/C++ environment, the maximum pipe width may be limited depending on the graphics card and the OpenGL version it supports. Older graphics cards that use compatibility profile (OpenGL version lower than 3.00) may limit the maximum pipe width to 10 pixels.

Customizing 3D Pipes

Single-Color 3D Pipes have the following properties:

PipeWidth specifies pipe width in pixels.
PipeColor specifies pipe color.
HighlightColor specifies gradient color used to create a 3D effect.
LineWidthScaling controls line width scaling policy as described above.

Flow Lines properties are grouped into two categories: **Flow** and **FlowAnimation**. The **Flow** category contains the following properties:

LineWidth specifies the width of the flow line.

Color specifies flow line color.
EnabledLineType specifies the line type used when the flow is enabled.
DisabledLineType specifies the line type used when the flow is disabled.
ArrowType specifies an optional arrow type that can be used to show flow direction.

The **FlowAnimation** category contains the following properties:

Enabled controls the flow animation state, enabled or disabled.
Interval controls animation speed by specifying animation timer interval in seconds.
Inversed controls animation direction.

Single-Color 3D Pipes with Flow Lines have properties of both the single-color pipes (accessible via the **Pipe** button on the Public Properties dialog) and the flow lines.

Color Flow Pipes change their color based on the value of their *State* property (0 or 1). Their colors are accessible via the **Colors** button and contain two pipe color properties (*PipeColor0* and *PipeColor1*) that are used to render corresponding values of the *State* property (0 or 1). The *EdgeCoefficient* property in the range from 0 to 1 specifies the darkness of the gradient used to create 3D effect. In the flat version of this pipe, *EdgeCoefficient* controls darkness of the pipe edge.

Pipe Elbows have the same set of properties as the corresponding pipes and also have an additional *Radius* property that specifies its radius.

Flanges and Elbows have the same set of color properties as the corresponding pipes and also have the following additional properties:

PipeWidth specifies the width of the pipe the flange or the elbow is used with.
PipeWidthScaling property specifies scaling policy and should match the value of the pipe's *LineWidthScaling* property.
ScaleLength
ScaleWidth
Specify scaling factors of a flange's length and width.
ScaleFactor specifies an elbow's scaling factor.
Angle is an optional property of rotated flanges that specifies the rotation angle.

Electrical and Electronic Circuit Symbols

This widget set includes several palettes containing electric switches and relays, transformers, fuses, resistors, capacitors, diodes, transistors, logic gates and many other electric and electronic symbols.

The switches, relays, fuses and circuit breaker symbols have a resource named *Value* that controls the widget's dynamic state: open or closed. The rest of the symbols are static and do not have dynamic resources.

The Public Properties dialog and the Resource Browser may be used to edit properties and resources of a selected widget as described on page 11.

Electric Switches and Relays

These widgets have animation that shows their state, such as open or closed, which is controlled by the widget's *Value* resource. All widgets have similar properties and contain the following buttons in the Public Properties dialog:

Actuator accesses rendering attributes of the widget's dynamic element.

Contacts accesses attributes used to render contacts, including their marker size.

The dialog may also include buttons that access rendering attributes of other widget elements. For example, **Link**, **Relay** and **Wire** buttons are present for relay widgets.

The following parameters that control widget geometry may also be present depending on widget type:

Length defines distance between contacts.

Width defines distance between parallel contact groups.

Angle controls the angle of rotating actuators, or the maximum rotation angle of a selector switch.

StartAngle

Radius

NumContacts

Control geometry and a number of contacts of a selector switch.

Transformers and Inductors

The Public Properties dialog for these static widgets contains **Contacts** and **Wire** buttons that access rendering properties of the corresponding widget elements. The **Arrow**, **Body**, **Phase Marks** and other buttons for editing widget elements may be present depending on the widget type.

Resistors, Capacitors, Diodes

The Public Properties dialog for these static widgets may contain **Contact, Element** and **Wire** buttons that access rendering properties of the corresponding widget elements. The **Arrow**, **Body**, other buttons for editing widget elements may be present depending on the widget type.

Transistors, Gates, Amplifiers

The Public Properties dialog for these static widgets may contain **Body, Contact, Element** and **Wire** buttons that access rendering properties of the corresponding widget elements. The **Arrow**, **Body** and other buttons for editing widget elements may be present depending on the widget type.

For gate widgets with a label, the dialog will also contain a **Label** button for accessing label rendering attributes. The **TextScaling** property may be used to enable label text scaling.

Miscellaneous Electrical Components

These widgets contain both static and dynamic electrical symbols.The static widgets have properties similar to the transistors, gates and amplifiers described in the previous section.

The dynamic symbols include fuses and circuit breakers which have animation that shows their state by using different colors. The animation is controlled by the widget's *Value* resource. The Public Properties dialog for these widgets also contains properties that specify colors used to show normal and blown fuse states, or open and closed circuit breaker states.

Special Widget Set

The Special Widget Set includes two palettes with interface input objects:

Menus and Check Boxes contain menus, check boxes and radio menus.

Spinners and Miscellaneous Input Widgets contain spinner and list widgets, as well as several types of text input widgets, including numerical input widgets.

The properties of these widgets are described in the the *Input Widgets* section.

The Special Widget Set also includes **Layout Templates** and **Special Widgets** palettes described below.

Layout Templates

Layout templates provide a collection of templates for an application's top-level window. Each template contains a viewport named *DisplayArea*, as well as optional fixed-size (non-resizable) viewports on the left, right, top and bottom of *DisplayArea* that can be used to host a toolbar, a menu or a message area. The optional areas are named *AreaLeft, AreaRight, AreaTop* and *AreaBottom*. The *OffsetLeft, OffsetRight, OffsetTop* and *OffsetBottom* properties shown in the Public Properties dialog define dimension of optional areas in screen pixels. These properties can also be accessed as resources at run time.

For applications that use a menu to navigate between different drawings, the *DisplayArea* viewport can be replaced with a *SubWindow* object by selecting the viewport and using the *Arrange, Replace Viewport with SubWindow* menu option available in the Enterprise Edition of the Graphics Builder. The subwindow's *Source* attribute can be changed to FILE, and the *Source Path* attribute can be used to supply a filename of the drawing to be displayed in the subwindow.

Fixed Size Dialogs

The last two widgets in the Layout Templates palette are fixed size viewports that can be used for non-resizable popup dialogs, with anchor point in the upper left corner or in the center of the widget. The Public Properties dialog for these widgets contain properties that define their dimensions and the anchor point position.

Special Widgets

This palette contains various specialized components. It includes linear and angular axes that can be used to design custom widgets, stand-alone legend symbols that can be used with the 2D and 3D graphs (the real-time chart widgets use a different integrated legend), and a few other miscellaneous widgets.

The **Axis** widgets use the Axis object and can be edited using the object's Properties dialog. The **Angular Axis** widgets use the Angular Axis object that can be edited via either the Properties dialog or the Public Properties dialog. The legend widgets can be edited using resources.

The palette also includes various **palette** and **browser widgets** described in the the *Input Widgets* section. The widgets can be customized using the Graphics Builder's Resource Browser dialog.

Alarm Bar Widget

The **Alarm Bar Widget** displays multi-colored stacked bars whose height is controlled by thresholds. It has the following properties in the Public Properties dialog:

Low
High
Specify the low and high range.
Height specifies the bar height that corresponds to the high range.
NumThresholds defines the number of stacked bars.
Thresholds accesses threshold values in the increasing order.
Colors defines threshold colors.

Flow Line Template

The flow line template widget provides an example of the flow line dynamics with configurable animation properties. The flow animation transformation is attached to the object's *LineType* attribute. The animation can be reused by marking the transformation and attaching a copy of it to a *LineType* attribute of another object via the Use Marked option of adding dynamics. Alternatively, a LineType attribute of other objects in the drawing can be constrained to the *LineType* of the template widget.

The widget has the following animation properties:

FlowEnabled enables or disables animation.
FlowInterval specifies the animation timer interval in seconds.
FlowInversed controls the flow direction (0 or 1).
EnabledLineType
DisabledLineType
Specify the line type used when animation is enabled or disabled.

Message List

This widget presents a simple example of a scrollable message list and has properties that control the number of message rows, the height of each message row, as well as rendering attributes of its elements. The SCADA Viewer demo includes a more elaborate example of a scrollable custom alarm table.

The *TextEntryPoint* resource can be used to push text strings to be displayed in the list. The new entries will appear at the top of the list, and the rest of the entries will be shifted down. Alternatively, an application can set individual list elements directly via resources.

2D and 3D Graphs

The **2D Graphs** and **3D Graphs** widget sets contain legacy 2D and 3D graph widgets that provide a variety of graph types, such as bar, pie, polar and other graphs with presentation-style graphics.

While the Real-Time Charts are implemented using a single performance-optimized *Chart* object with an integrated data buffer, the 2D and 3D graphs (including their axes and legends) are constructed using a collection of individual graphical objects, such as polygons, text, series and other objects. The 2D and 3D graphs support gradient shading and 3D bar rendering, as well as packed, stacked and other multi-set bar graphs.

Each graph widget can be prototyped in the Graphics Builder as described in the *Prototyping Widgets Using Predefined Animation Commands* section on page 13.

The 2D and 3D graph widgets do not have public properties and are customized using resources. When a graph widget is loaded in the Graphics Builder or HMI Configurator, the Resource Browser accessed via the *Object, Resources* menu option or the *Object Resources* R toolbar button can be used to view and edit widget resources.

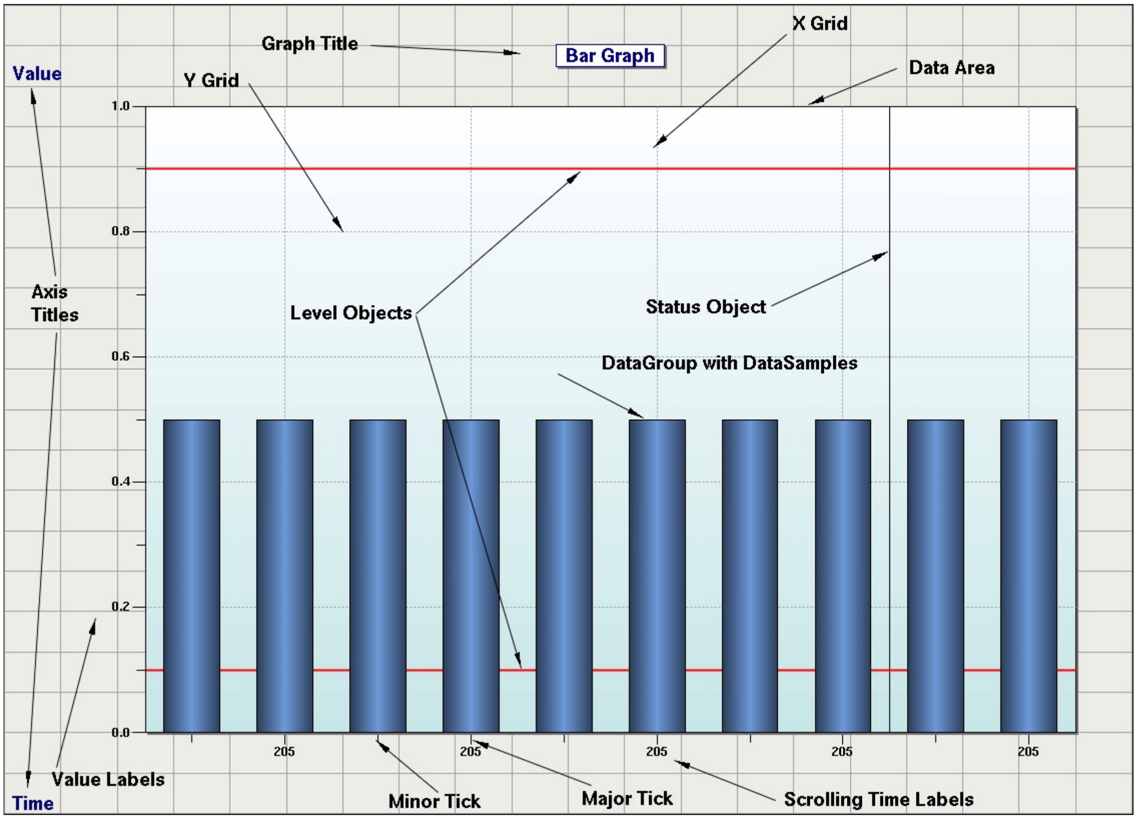
The following chapters describe components and resource hierarchies of the 2D and 3D graph widgets.

OpenGL Note

The 3D version of graph widgets is optimized for the OpenGL driver by default. It uses alpha-blending and transparency by setting the *Visibility* attribute of the graph's *DataGroup* to a fractional value. Set *DataGroup/Visibility* to 1 when a 3D graph widget is used in C/C++ applications with a GDI driver that doesn't support alpha-blending.

Components of a GLG Graph

The graphs of the GLG Widget Library are all constructed using similar elements to represent graph components such as axes, ticks, labels and data. A typical graph has a data area, axes, grids, titles, minor and major ticks, major tick labels for each axis, a data group containing graph data samples, a group of level lines and a status object, as shown in the picture below.



Components of a Graph Widget

For convenience, these elements have identical names in all graphs, although some names may have additional suffixes in certain cases. This means that editing any particular element is similar for all graphs.

The widgets in the GLG Widget Library are intended for use as templates. They may be used as they are or edited with the GLG Graphics Builder to produce customized versions.

Naming Conventions

The names of resources play an important role, since they are used to access objects inside a widget. There are a few naming conventions used for the Graph Widget components:

- The names of the elements associated with an axis start with the name of the axis (X or Y), such as *XLabel* or *YMajorTick*.
- The *series object* is used to create a variable number of objects in a graph, such as a variable number of datasamples, labels and ticks. The name of the series object ends with the *Group* suffix, such as *DataGroup*, *XLabelGroup* or *YMajorGroup* for major ticks on the Y axis.

- Each series replicates an object defined as its template. The value of the series' *Factor* attribute defines the number of the created template copies. For example, *DataGroup/Factor* defines the number of datasamples in a graph, and *XLabelGroup/Factor* defines the number of major tick labels. The label and tick series also have the *MinorFactor* attribute which controls the number of minor ticks, such as *XLabelGroup/MinorFactor.*
- The name of a template of a series object does not have any numerical suffix, such as *DataSample*. The names of the created instances of the template in a series object have a numerical suffix corresponding to the sequential number of the instance, such as *DataSample0, DataSample1*, and so on. All indexes are zero based, and *DataSample1* is actually the second object in a series.
- Nested series may be used for creating several groups of groups with a variable number of elements. For example, a multi-line graph uses the first series to create a line with a variable number of points. This series is then used as a template to the second series to create a variable number of lines.

  In the case of nested series objects, the name of top level series object is formed by adding a numerical suffix to the name of the bottom level composite object. This suffix indicates the level of the hierarchy and is spelled out as a word. For example, if there are two levels of series objects and the name of the inner series is *DataGroup*, the name of the outer series is *DataGroupOne*. The final hierarchy and the corresponding names look like this:

```
DataGroupOne        (series of lines)
   DataGroup        (line template)
      DataSample    (datasample template)
   DataGroup0       (first line instance)
      DataSample    (datasample template)
      DataSample0   (datasample instance)
      DataSample1   (datasample instance)
      ...
   DataGroup1       (second line instance)
      DataSample    (datasample template)
      DataSample0   (datasample instance)
      DataSample1   (datasample instance)
      ...
   ...
```

Performance Optimizations

The Real-Time Chart widget described on page 14 is optimized for performance with large data sets and should be used instead of the line and bar charts when possible. The rest of this section describes performance optimizations for 2D and 3D graph widgets.

Gradients

By default, the 2D version of graph widgets uses gradients for the graph's *DataArea*, line markers, bars of the bar graphs and other graph elements. When a large number of datasamples needs to be displayed with a fast update rate, the gradients may be disabled by editing object's rendering attributes to minimize CPU utilization and increase graph's update rate.

RollBack Scrolling

The *RollBack* attribute of the graph's *DataGroup* may be used in conjunction with the *WRAPPED* scrolling type for implementing scrolling behavior which scrolls the graph only once every **n** iterations, as defined by the value of the *RollBack* attribute.

For example, consider a graph with the *WRAPPED* scroll type, 200 datasamples, 10 X axis major ticks with labels, and 20 minor ticks per one major tick interval. Setting *DataGroup/Rollback=40* and *XLabelGroup/RollBack=2* will "roll" the graph back by 2 major tick and label intervals (which corresponds to 40 datasamples) when the graph gets completely filled with data.

The use of the *RollBack* limits the CPU-intensive scrolling operation to be performed only once on every 40th data update, compared with every data update in the regular scrolling graph with the *SCROLLED* scrolling type.

A special case of the rollback may be used to implement the graph which switches from the WRAPPED behavior to the SCROLLED behavior when the graph gets completely filled with data the first time. For example, for a graph with the WRAPPED scroll type, 200 datasamples, 10 X axis major ticks and labels, and 20 minor ticks per one major tick interval, the following settings may be used: *DataGroup/Rollback=1* and *XLabelGroup/Rollback=0.05* (which corresponds to one minor tick - 1/20).

Auxiliary Elements

The update speed of a graph also depends on the number and type of objects rendered on every update iteration. Deleting auxiliary elements of a graph such as grids or minor ticks, as well as filled data area can substantially increase the graph's maximum update speed.

Batch Update

For graphs with a large number of data samples and fast update rates, rendering performance may be optimized by reducing the number of times the graph is redrawn. For example, if a graph receives a new data value every 10 milliseconds and redraws after each new data value, it will be redrawn 100 times per second. Instead of refreshing the graph each time a new data point is received, the application may update the graph just several times per second. Each update would push all data points accumulated since the last update and redraw the graph just once for all accumulated new data values.

Commonly Used Graph Resources

Before proceeding with the detailed description of all graph resources, let's consider the typical resources used to animate a bar graph:

DataGroup/Factor	Defines the number of datasamples in the graph
DataGroup/DataSample/Low	Defines the low range of data
DataGroup/DataSample/High	Defines the high range of data
DataGroup/DataSample/Value	Defines an initial value for graph's samples
DataGroup/ScrollType	Defines the graph's scroll type (WRAPPED or SCROLLED)
DataGroup/EntryPoint	An entry point for pushing data values into the graph.

XLabelGroup/Factor	Defines the number of X labels, major ticks and grids
XLabelGroup/MinorFactor	Defines the number of minor ticks per each major tick interval
XLabelGroup/XLabel/String	Defines an initial value for the label, usually set to an empty string
XLabelGroup/EntryPoint	An entry point for pushing X axis labels into the graph
YLabelGroup/Factor	Defines the number of Y labels, major ticks and grids
YLabelGroup/MinorFactor	Defines the number of minor ticks per each major tick interval
YLabelGroup/Low	Defines the low range of the Y axis labels, same as DataGroup/DataSample/Low
YLabelGroup/High	Defines the low range of the Y axis labels, same as DataGroup/DataSample/High

A line graph has a slightly different resource hierarchy for its *DataGroup*:

DataGroup/Factor	Defines the number of datasamples in the graph
DataGroup/Marker/DataSample/Low	Defines the low range of data
DataGroup/Marker/DataSample/High	Defines the high range of data
DataGroup/Marker/DataSample/Value	Defines an initial value for graph's samples
DataGroup/ScrollType	Defines the graph's scroll type
DataGroup/EntryPoint	An entry point for pushing data values into the graph

A multi-line graph has a different *DataGroup* hierarchy as well:

DataGroupOne/Factor	Defines the number of lines in the graph
DataGroupOne/Persistent	May be set to PERSISTENT to preserve attribute settings of individual lines
DataGroupOne/DataGroup/Factor	Defines a number of point in each line
DataGroupOne/DataGroup/Marker/DataSample/Low	Defines the low range
DataGroupOne/DataGroup/Marker/DataSample/High	Defines the high range
DataGroupOne/DataGroup/Marker/DataSample/Value	Defines an initial value
DataGroupOne/DataGroup/ScrollType	Defines the graph's scroll type
DataGroupOne/EntryPoint	An entry point for pushing data values into all lines
DataGroupOne/DataGroupN/EntryPoint	An entry point for pushing values into the Nth line

The top level *DataGroupOne/EntryPoint* resource may be used to push values into all lines of the graph. The entry points of each line (the *DataGroupOne/DataGroupN/EntryPoint* resources, where N is the index of a line) may be used to push datasamples into each line separately.

Data Samples and the Data Group

Data samples are objects used by a graph to display its data. The type of object used as a data sample may be different depending on the type of the graph. For example, a data sample in a bar graph is represented by a polygon whose height changes with the value of the data value displayed. For a line graph, the data value is represented by the position (usually just the Y-coordinate) of a point on the line.

The method used to create multiple instances of a data sample object depends on the type of the graph. For example, a bar graph uses a *series* object to create a variable number of bar datasamples, and a line graph uses a *polyline* object to create a line with a variable number of points. In both cases, the *Factor* resource controls the number of bars or line points. Regardless of the type of an object used to replicate graph's datasamples, such a collection of data samples is called a **data group**, and these are identified with the *DataGroup* resource name.

If a series object is used to produce multiple copies of data samples, a template of the series object is usually named *DataSample*. The names of the created copies are formed by adding a consecutive digital index of the copy to the name of the template: *DataSample0*, *DataSample1*, and so on.

Each *DataGroup* has an *EntryPoint* resource which is used to push new data into the graph, while the graph automatically scrolls the data, via a history object attached to the *DataGroup*. The *ScrollType* resource allows changing the graph's scroll type, while the *Inversed* resource controls the update direction.

Resources of a DataSample

The type of object used as a data sample depends on the type of the graph. However, there are three resources present in the *DataSample* resource no matter the type of object. These are:

Value

The value to be represented by the data sample object. The *Value* resource of the template *DataSample* defines the initial value of all datasamples, while the *Value* of other datasamples defines the value of the datasample displayed in the graph.

The graph always works in a linear coordinate system. If data has to be displayed in the logarithmic format, it must be converted to the logarithmic scale before setting the *Value* resource.
Changes made to the *Value* resource are not permanent, and will revert to an initial value when the graph is reset. To make *Value* resource changes permanent, explode the *DataGroup* object with the GLG Graphics Builder. This may be used to save the data in the graph, and not as general technique, as the graph looses the ability to change the number of its datasamples after its datagroup has been exploded.

Low* and *High

These resources define the range of data displayed by the graph. Data sample shapes that extend beyond these limits may be clipped. The default values are 0 and 1, respectively. These resources may be changed dynamically without losing data currently displayed in a graph. If the *Low* or *High* resources are changed, the graph automatically adjusts all currently displayed data to the new range. This allows the application program to change the range of the data on the fly when the amplitude of the data increases or decreases beyond the current graph's data range. It also allows the program to scale the data samples when the values get too small.

Truncate

Controls how the data samples with values outside of the graph's range are plotted. If set to YES, the values will be truncated to fit inside the graph's range. For vertical graphs, the data samples whose values are truncated will appear at the top or the bottom of *DataArea*.

The data sample objects displayed in a graph are dynamically created objects whose attributes are inherited from the template *DataSample*. The attributes of the *DataSample* template may be changed to define the appearance of all datasamples in a graph.

The program can change attributes of any datasample at any time as well, but as with any series object, changes to the template instances are not permanent, and last only until the drawing is reset or reloaded, at which time the instances of the datasamples are recreated by copying the template datasample. This behavior may be prototyped in the GLG Graphics Builder by changing resources of datasample instances: the changes will be discarded after resetting the graph by pressing the *Reset* toolbar button.

Local and Global Attributes

Any attribute of a datasample or any other template object may be made local or global by setting the value of its *Global* flag. If the attribute is global, it is constrained for all datasamples in the series. Changing the value of a global attribute will affect all datasamples as well as the series' template. To change an attribute of individual datasamples independently, set the *Global* flag for that attribute to *LOCAL*.

If a template attribute is *GLOBAL*, changing the attribute immediately affects all instances of the template in the series. If a template attribute is *LOCAL*, the drawing needs to be reset by pressing the *Reset* toolbar button after changing the attribute in the GLG Graphics Builder to see the new values. In a program, the values of local attributes must be set before the hierarchy setup, since the hierarchy setup creates instances of the template and copies the settings of the template's attributes.

Attribute Transformations

In some graphs, various attributes of the data samples may have a transformation attached to them. For example, in a multi-line graph, the line's *EdgeColor* attribute has a *List* transformation to change the color depending on the line index, so that each line is rendered in a different color. A similar transformation is used to vary colors of the individual bars in packed and stacked bar graphs.

In this case, the value of the attribute can not be edited directly, as it is affected by the transformation attached to it. Instead, the Color0, Color1, Color2, ... resources will be provided to define a color of each line or bar.

To add more colors to the list of colors, select the line template of the polyline, select its *EdgeColor* attribute and edit a transformation attached to the attribute.

Nested Data Groups

Some types of graphs contain data groups whose elements are themselves composite objects. For example, a line graph with multiple lines or a packed bar graph that contain several "packs" of data, each of which contain several data samples.

In a multi-line graph, the series object used to create multiple lines is named *DataGroupOne.* It contains a template object named *DataGroup* which is used as a template for each line. The created line instances are named *DataGroup0*, *DataGroup1*, and so on. Each line instance contains datasamples named *DataSample0, DataSample1*, etc., located inside the line's *Points* group: *Points/DataSample0*, *Points/DataSample1*, and so on. Each line also contains *DataSample* template located inside its *Marker* object: *Marker/DataSample*.

In graphs such as a packed or stacked bar graph, the top level series object named *DataGroup* is used to replicate a desired number of "packs" or "stacks" of data, defined in its template named *Pack*. The created instances of packs will have names *Pack0*, *Pack1*, and so on. Each pack is a series object too, used to replicate its *DataSample* template to create several datasamples in each pack, named *DataSample0, DataSample1*, and so on.

For multi-line and other multi-set graphs, the *DataGroupOne/Persistency* attribute may be set to PERSISTENT to preserve attribute settings of individual lines. By default, the *Persistent* attribute is set to VOLATILE, and only the color setting is saved, while the rest of the line attributes have to be set at runtime. When the PERSISTENT setting is used, the attributes of individual lines may be set in the Builder.

DataGroup Resource Hierarchy

A *DataGroup* of a bar graph and other graphs that use the series object to replicate datasamples have the following resource hierarchy:

```
DataGroup         (series object)
   EntryPoint     (entry point for pushing data values into the graph)
   Factor         (number of datasamples)
   DataSample     (datasample template)
      Low         (low range of the graph)
      High        (high range of the graph)
      Truncate    (controls how out of range values are plotted)
      Value       (inital value)
   DataSample0    (datasample instance)
   DataSample1    (datasample instance)
   ...            (more datasample instances)
```

A packed or stacked bar graph will have a slightly different resource hierarchy:

```
DataGroup         (series object)
   EntryPoint     (entry point for pushing values into the graph)
   Factor         (number of datasample packs in the graph)
   Pack           (template of a pack containing several datasamples)
      Factor      (number of datasamples in a pack)
      Persistent  (May be set to preserve attribute settings of
                   individual samples in a pack)
      DataSample  (datasample template)
         Low      (low range of the graph)
         High     (high range of the graph)
         Truncate (controls how out of range values are plotted)
         Value    (inital value)
   Pack0          (pack instance)
   Pack1          (pack instance)
   ...            (more pack instances)
```

A typical line graph and other graphs that use the polyline object with a variable number of points have the following resource hierarchy:

```
DataGroup              (polyline object)
   EntryPoint          (entry point for pushing values into the graph)
   Factor              (number of datasamples)
   Marker              (template used for a point of the graph)
      DataSample       (datasample)
         Low           (low range of the graph)
         High          (high range of the graph)
         Truncate      (controls how out of range values are plotted)
         Value         (inital value)
   Points              (group containing all line’s datasamples)
      DataSample0      (datasample instance)
      DataSample1      (datasample instance)
      ...              (more datasample instances)
```

Finally, multi-line graphs that display several lines have the following resources:

```
DataGroupOne           (series object used to replicate line instances)
   EntryPoint          (entry point for pushing values into all lines)
   Factor              (number of lines)
   Persistent          (May be set to preserve attribute settings of
                        individual lines)
   DataGroup           (polyline object used as a line template)
      Factor           (initial number of points in each line)
      Marker           (template used for a point of the graph)
         DataSample    (datasample)
            Low        (low range of the graph)
            High       (high range of the graph)
            Truncate   (controls how out of range values are plotted)
            Value      (inital value)
   DataGroup0          (line instance)
      EntryPoint       (entry point for pushing values into this line)
      Factor           (number of datasamples in this line)
      Marker           (template used for a point of this line)
         DataSample    (datasample)
            Low        (low range of this line)
            High       (high range of this line)
            Value      (inital value)
   DataGroup1          (line instance)
      ...              (line resources, same as in DataGroup0)
   ...                 (more lines instances)
```

Data Area

The Data Area is a filled polygon used as a background for 2D graphs. It allows the graph to paint the area behind data samples in a color different from the background color. In 3D versions of the graphs, there are several data areas, representing the three planes of the graph’s frame.

By selecting the Data Area polygon and editing it, you may change its fill color, border color and border width. You may also change the geometry of the whole graph by moving control points of the Data Area. The Data Area is rendered as a parallelogram object, and has three control points.

Like any parallelogram object, the three unconstrained control points of the Data Area may be named and then accessed as resources. This gives you the ability to change the geometry of the graph dynamically from a program. Changing the geometry of the Data Area adjusts the geometry of all datasamples displayed in the graph.

Axes

An **axis** of a graph is just a polygon. Axes are named by the corresponding coordinate: *XAxis*, *YAxis* and *ZAxis* (for 3D graphs).

Major And Minor Ticks

A major tick is represented by a polygon object named *XMajorTick* for the X axis, *YMajorTick* for the Y axis, and *ZMajorTick* for the Z axis of 3D graphs. Everything described below in this chapter for the X axis ticks is also applicable to the ticks on the Y and Z axes.

The *XMajorTick* polygon is used as a template for the *XMajorGroup* series object to produce multiple copies of a major tick. The number of ticks is defined by the factor of the series object, and their appearance is defined by the series object template.

A minor tick is represented by a polygon named *XMinorTick* for the X axis. The graphs use two nested series objects to produce the minor ticks. The inner series object, named *XMinorGroup*, uses *XMinorGroup* polygon as a template to produce the necessary number of minor ticks for one major tick interval. The series object on the top level, named *XMinorGroupOne*, uses the XMinorGroup series as a template reproduce it the number of times equal to the number of major tick intervals.

The templates of both the major and minor ticks have the *EndPoint* resource which controls the length of the tick. This resource may be used to adjust the tick length when the graph is stretched and ticks on one of the axes become too long or too short.

Setting the Number of Major and Minor Ticks

For convenience, the *Factor* and *MinorFactor* resource of both the major and minor ticks and the grids are constrained to the corresponding factors of the labels and may be edited in one place - the *XLabelGroup* series. Its *Factor* resource controls the number of labels, major ticks and grids, and the *MinorFactor* resource controls the number of minor ticks per each major tick. The total number of minor ticks, which is the product of the number of major ticks times the number of minor ticks, must match the number of the graph’s datasamples, or the ticks will be misaligned:

```
XLabelGroup/Factor * XLabelGroup/MinorFactor = DataGroup/Factor
```

Performance Optimization

In graphs with a large number of datasamples, the minor ticks series may be deleted to simplify the graph and increase its update performance.

Logarithmic Ticks

Ticks can be positioned on a logarithmic scale if the graph displays logarithmic data. In this case, only minor ticks are drawn with uneven intervals. Major ticks are still drawn evenly, since they correspond to the integer powers of the logarithm base. It is the responsibility of the program or the process supplying data to provide logarithmic data sample values.

To make minor ticks logarithmic, set the *LogType* attribute of the lower level series object of Y minor ticks (*YMinorGroup*) to *LOG*. The factor of this series should be set to a value of the base of the logarithm minus 1. For example, for the logarithm base ten, the factor should be set to nine.

Grid Lines

The background grid lines in a GLG Graph are rendered as polygons named *XGrid* and *YGrid*. The polygons are used as templates in the series objects named *XGridGroup* and *YGridGroup*, which produce the necessary number of grid lines. The factors of the grid's series are constrained to have the same values as the number of major ticks on the corresponding axis.

Labels

Axis labels to be displayed at each major tick are represented by text objects named *XLabel*, *YLabel* and, for 3D graphs, *ZLabel*. These text objects are used as templates for the series objects named *XLabelGroup*, *YLabelGroup* and *ZLabelGroup*, which produce the desired number of labels for the corresponding axis. Each label is positioned at the corresponding major tick.

The values of the *Factor* attribute of each label series defines the number of labels, major ticks and grids. The *MinorFactor* resource defines the number of minor ticks per one major tick interval. For the time axis with scrolling labels (the X axis for horizontal graphs), the total number of minor ticks, which is the product of the number of major ticks times the number of minor ticks, must match the number of the graph's datasamples, or the ticks and labels will be misaligned with the datasamples.

As with any other series, every individual label may be accessed with the index of the label as a suffix. For example, *XLabelGroup/XLabel3* accesses the fourth label on the X axis (since the index is zero-based).

The *XLabelGroup/XLabel* and *YLabelGroup/YLabel* access the label templates which define the initial appearance of all labels for the X and Y axes. The attributes of each individual label of the label series may be changed dynamically at run time, but these changes are volatile and disappear after resetting or reloading the drawing.

The labels' font can be changed by editing the *FontType* and *FontSize* attributes of the label's template, i.e. *XLabelGroup/XLabel/FontType*.

By default, the labels use SCALED text and the FontSize resource controls the maximum size. The label will use a smaller font if the window is resized to a small size or if the label is too long. To use a label with a constant font size, change the label template to a FIXED font and adjust its control point (the *XLabel/Point* resource) to position it below the corresponding major tick.

If the type of the text object template for a label series is SCALED, the *Global* flag of the *SizeConstraint* attribute of the template label controls whether or not all axis labels are displayed using the same font size when labels have different length.

If the *Global* flag is set to LOCAL for the template label, label text strings are fitted into their boxes individually and may be displayed in different font sizes if labels have different length. If the flag is set to GLOBAL, fitting the text of one label into the corresponding box affects all other labels. It means that if the text of a label does not fit into its box, a font of the smaller size is chosen for that label. This change of a font size affects all labels on the axis, so that all of them are displayed in this smaller font size. After the conditions that caused the change of the labels' font size are eliminated, the labels stay in their current small font size until the viewport is resized or reset.

Value Labels and the Format Resource

The labels on the value axis (the Y axis for horizontal graphs) are constrained to the values associated with the corresponding major ticks. The strings for the labels are supplied automatically depending on the graph's data range and the factor of major ticks.

For example, if the data range of the graph is [0,1] and the graph has six major ticks (the factor of major ticks is 5), then the strings displayed in the labels are"*0.0*", "*0.2*", "*0.4*", "*0.6*", "*0.8*" and "*1.0*".

Every value label has resources named *Low* and *High* that define the range used for labeling. These resources are constrained to have the same values as the *Low* and *High* ranges of the graph's DataSample. Changing these resources affects both the labeling range and the data range of the graph, rescaling currently displayed data and displaying new label values.

The *String* attribute of a value label has a format transformation attached to the attribute to produce the proper value. The *Format* attribute of the transformation is exported as the *Format* resource of the label and may be edited to change the label format.

The *Format* resource defines a C-style format used to display a label's data value. The format string should have no more than one conversion specification requiring an argument. This conversion specification should take a double-precision value as an argument, therefore only **f**, **g** and **e** format specifications are allowed. The conversion specification is optional, and may be omitted, allowing you to use the *Format* resource to replace the label string.

For example, to specify two digits of precision for all labels in the Y label series, change the *Format* resource of the label series template (*YLabelGroup/YLabel/Format*) to "*%.2f*". The *Format* may also be set to the following string "*Value=%.2f*" to display the *"Value="* in front of each label value.

The *"%"* character has a special meaning and defines the start of a format specification. If you want to display a "%" character in the string, use "%%" in its place. Refer to any C compiler documentation for a complete explanation of string formatting capabilities.

NOTE: It is the responsibility of the programmer to supply a valid C format for the *Format* string. The GLG Toolkit does not check the format for validity before displaying a drawing. Any entry is treated as a valid C format, in exactly the same way as any C program does. Defining the wrong format may cause unpredictable results, including a program crash.

To disable automatic labelling and supply a string to be displayed in each label at run-time, set the value of the *Global* flag of the *Format* resource of the template label to *LOCAL* (and reset the drawing if it is done in the GLG Graphics Builder). Then set the *Format* resource of each label instance to a desired string. For example, setting a format string of a label instance to "*Level 1*" simply displays "*Level 1*" in the label.

Scrolling Time Axis Labels

The labels on the time axis (X axis horizontal graphs) annotate the datasamples of the graph and scroll with them. The *XLabelGroup* time label series is equipped with the *EntryPoint* resource for convenience in setting graph labels. There are two methods of supplying strings for updating time axis labels.

The first method is to set the *String* attribute of the label directly using the name of the label text object. For example, *XLabelGroup/XLabel3/String* would be the resource name used to access the string of the fourth label on the X axis. Although straightforward, this method requires the application program to keep track of the current update position and to update labels differently depending on the scrolling behavior of the graph.

The second method is to use the *EntryPoint* resource of the label group. This method allows the program to supply label strings without defining where to place them. This keeps the labels synchronous with the corresponding data samples.

The *XLabelGroup/EntryPoint* resource is used for updating X axis labels. Setting this resource to different values in a loop animates the X axis labels according to the graph scrolling behavior. Querying the value of the resource yields the last string used for updating.

To keep scrolling labels synchronized with the corresponding datasamples, the *Factor* and *MinorFactor* resources must be set to values that match the number of datasamples in the graph. The *XLabelGroup/Factor* resource controls the number of labels and major, while the *XLabelGroup/MinorFactor* resource controls the number of minor ticks per one major tick interval. The product of these two values yields the total number of minor ticks in the graph and should equal to the number of datasamples:

*DataGroup/Factor = XLabelGroup/Factor * XLabelGroup/MinorFactor*

When this condition is met, the program can simply supply one time label per each datasample, and the graph will handle the scrolling, displaying only the label values that correspond to the major tick.

For example, let's consider a graph with 200 datasamples, 10 major ticks and labels, and 20 minor ticks per each major tick interval. Since 10 * 20 == 200, the program may supply one label and one datasample value per each update, and the graph will scroll the datasamples and labels automatically.

For a multi-line graph with the same resource settings and three lines, the program would supply three datasample values (one for each line) and one label per each update. The same would also be true for a packed or stacked bar graph with 3 datasamples in each pack.

For the graphs with a large number of datasamples, minor ticks may be deleted from the drawing to decrease the number of objects used by the graph. The *MinorFactor* of the label series should be still set to a proper value to maintain the equation *Factor x MinorFactor = Number of DataSamples*.

The *ScrollType* and Inversed resources of the *XLabelGroup* object is constrained to the corresponding resources of the graph's *DataGroup* to ensure that the scrolling behavior of the labels is consistent with the scrolling behavior of the graph. The values of these resources may be set in either place.

Level Objects

In addition to grids, many of the GLG graphs have **level objects**. These are horizontal lines usually used to mark some special thresholds in the graph. While grid lines are always positioned automatically on the levels defined by the major ticks, level objects may be positioned arbitrarily to annotate application-specific thresholds.

A level object is represented by a polygon named *LevelObject* which is used as a template for the *LevelObjectGroup* series to produce several copies of the level object. A factor of the series defines the number of level objects created. Each level object has a *Value* resource, controlling its position, or level.

Attributes of polygons representing every level line may be changed to distinguish different levels, for example using different colors. As with all series, the *Global* flags of the template attributes that need to be changed must be set to LOCAL, otherwise all level objects are affected. The level's series object may be exploded with the GLG Graphics Builder to make the changes permanent, otherwise the attributes have to be changed dynamically at run-time.

Level objects may be set invisible or deleted from the graph drawing if they are not used.

Status Object

The **status object** (the *StatusObject* resource) is used to indicate the current update position for the graphs with the WRAPPED scrolling type. It moves after every update to the next data sample, wrapping back to the starting position after reaching the last one.

A status object may be set invisible or deleted if not used. When a graph is just loaded or after a reset, the position of the status object coincides with the Y axis for horizontal graphs.

Titles

A few freely positioned titles are used to annotate a graph and its axes. The X and Y axis titles are named *XAxisLabel* and *YAxisLabel*, respectively. The graph's title text object is named *Title*. A title's position, as well as direction, text string, and other attributes may be changed by moving control points or editing attributes of the text object indicated by the resource name. An unlimited number of additional titles may be created in the GLG Graphics Builder by creating new text objects and positioning them.

Legends

Graphs with multiple datasets, such as multi-line graph or packed bar graph, contain legends that annotate the entities displayed by the datasets of the graph. Every legend item shows the name of the entity displayed by the corresponding data set and also the color or line attributes used to annotate it. The *Special* widget set also contains optional legends that may be added to any graph.

A legend consist of a series object named *LegendGroup* used to create multiple copies of a legend template. The legend template is made of a group named *Legend*. It contains a text object named *LegendLabel* and two polygons named *LegendBox* and *LegendLine*. Two marker objects named *LegendMarker1* and *LegendMarker2* are constrained to the ends of the *LegendLine* polygon.

The *LegendLabel* text object is used to name entities in the graph. The polygons and markers are used to display the color, line width or line type associated with a particular entity.

Multiple Axes

The *Special* widget set contains optional axis objects that may be added to any graph. Both the value and scrolling time axes are provided, including the vertical and horizontal versions.

3D viewing

3D Graph Widgets have a few resources designed to control the user's view of the graph in 3D space:

ShearFactor, ShearX, ShearY

Controls a shear transformation that may be applied to the whole drawing. The ShearFactor defines the degree of shear, and is meant to take values between 0 and 1 (although other values may be used for special effects). The ShearX and ShearY attributes determine the proportion of the shear in the X and Y direction. These two attributes should add to 1.

XAngle, YAngle, ZAngle

Control the rotation of the whole drawing around the specified axis.

To obtain the best effect, use either a rotate or shear transformation but not both, since they may interfere with each other. Most of the 3D graphs use the Shear viewing transformation by default, which provides the best 3D effect.

Chapter 2

2D and 3D Graph Widget Resources

2

This chapter contains descriptions of the 2D and 3D Graph widgets. Refer to the *Using the GLG Widgets* chapter on page 11 for information on using all other widget sets.

Each widget description includes information on the widget usage, data animation and widget resources. Because there is a lot of duplication between the widgets, the resources are grouped into "sets", and the descriptions below just list the resource sets used by each widget. For lists of the resources that make up the resource sets, see the *2D and 3D Graph Resource Sets* chapter on page 85 of this manual.

You may also want to consult the *Using the GLG Widgets* chapter on page 11 and the *2D and 3D Graphs* section on page 38 for general information on using 2D and 3D Graph widgets.

To test each widget, you can use the GLG Graphics Builder to animate the widget as described in the *Prototyping Widgets Using Predefined Animation Commands* section on page 13.

At run time, a graph is animated by setting its entry point resources. The resource path of entry points for animating a graph is listed in each graph's description. Each description also lists the graph's default data range, which can be changed by setting the graph's *Low* and *High* resources.

As an example, a graph widget might contain the following entry points:

Data Values Entry Point

$Widget/DataGroup/EntryPoint

Time Labels Entry Point

$Widget/XLabelGroup/EntryPoint

If the default data range is between 0 and 1, the widget can be animated using the following animation command:

```
$datagen d 0 1 "$Widget/DataGroup/EntryPoint"
         s 0 0 "$Widget/XLabelGroup/EntryPoint"
```

Refer to the *Prototyping 2D and 3D Graph Widgets with Custom Animation Commands* section on page 14 for information on using custom animation commands.

Refer to the *The Data Generation Utility* chapter on page 404 of the *GLG Programming Reference Manual* for a complete list of the prototyping command options.

The rest of this chapter describes the resource sets of each widget. For lists of the resources that make up the resource sets, see the *2D and 3D Graph Resource Sets* chapter on page 85.

Bar Graph
Signed Bar Graph
Histogram Graph

The Bar Graph is used to display one set of data as an arrangement of vertical bars. The height of a bar corresponds to the value of the data it represents. The Signed Bar Graph is similar to the bar graph, but data samples may have negative values as well as positive. The Histogram Graph is similar to the Bar Graph, but with no gaps between the bars.

Animation Data

The entry points for these graphs are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1. The default range for the Signed Bar Graph is between -1 and 1.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set

Horizontal Bar Graph

The Horizontal Bar Graph displays one set of data in 2D as horizontal bars. This graph is the same as the Bar Graph, except that the bars are horizontal. The graph resources are the same too, with the only distinction that X and Y prefixes are swapped.

Animation Data

The graph’s entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/YLabelGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Value Axis Resource Set
- Y Time Axis Resource Set
- Zooming Resource Set

Symmetrical Bar Graph

The Symmetrical Bar Graph displays one set of data in 2D as vertical bars centered around X axis. The height of a bar corresponds to the value of the data it represents. This is not the same as a Signed Bar Graph. The height of a bar above the X axis is the same as it's depth below it; it is symmetrical.

Animation Data

The graph's entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set

- Datagroup Resource Set
- Symmetrical Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set

Packed Bar Graph

The Packed Bar Graph displays several sets of data in 2D in clusters or “packs” of vertical bars. One pack of bars contains one sample of data that includes several datasamples. The height of each bar corresponds to the value of its data sample.

Animation Data

The graph’s entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1. There are three bars in each pack in the default graph.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Packed Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Multiaxis Resource Set
- Zooming Resource Set
- Legend Resource Set

Stacked Bar Graph

The Stacked Bar Graph displays several sets of data in 2D as stacked vertical bars. This is comparable to a packed bar graph except that the bars that make up each cluster are placed on top of each other. Each bar, composed of a pack of several smaller bars, contains one iteration of data from each data set. The height of every component of a stack corresponds to the value of its data sample. The total height of a stack is equal to the sum of the heights of the individual components. The range of the graph is equal to the range of the sum of the individual components. This means that the sum of all values corresponding to one stack should be less than or equal to the range of the graph.

Animation Data

The graph's entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 0.2, and there are, by default, 5 bars in each stack.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Packed Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set
- Legend Resource Set

Step Graph

The Step Graph displays one set of data in 2D as a step line. The height of a step corresponds to the value of a data sample.

Animation Data

The graph’s entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set

Filled Step Graph

The Filled Step Graph is the same as the Step Graph (see page 57), but with a filled line.

Animation Data

The graph’s entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set

- Data Area Resource Set
- Filled Step Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set

Horizontal Step Graph

The Horizontal Step Graph displays one set of data in 2D as a step line. The horizontal position of a step corresponds to the value of a data sample. This graph is the same as the Step Graph (see page 57), but is rotated on its side.

Animation Data

The graph's entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/YLabelGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Value Axis Resource Set
- Y Time Axis Resource Set
- Zooming Resource Set

Horizontal Filled Step Graph

The Horizontal Filled Step Graph displays one set of data in 2D as a vertical step line. The horizontal position of a step corresponds to the value of a data sample. This graph is the same as the Filled Step Graph (see page 58), but is rotated on its side.

Animation Data

The graph's entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/YLabelGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Filled Step Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Value Axis Resource Set
- Y Time Axis Resource Set
- Zooming Resource Set

Multiline Step Graph

The Multiline Step Graph displays multiple sets of data in 2D as step lines. The height of a step corresponds to the value of a data sample.

Animation Data

The graph's entry points are as follows:

Data Entry for All Lines

$Widget/DataGroupOne/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1. The default number of data groups is two.

Data Entry for One Line

To animate just one of the data groups, use this resource in place of the above data entry point:

$Widget/DataGroupOne/DataGroup<N>/EntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Multiset Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set
- Legend Resource Set

Filled Multiline Step Graph

The Filled Multiline Step Graph is the same as the Multiline Step Graph (see page 60), but with a filled step line.

Animation Data

The graph’s entry points are as follows:

Data Entry for All Lines

$Widget/DataGroupOne/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1. The default number of data groups is two.

Data Entry for One Line

To animate just one of the data groups, use this resource in place of the above data entry point:

$Widget/DataGroupOne/DataGroup<N>/EntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Filled Multiline Step Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set
- Legend Resource Set

Line Graph
Filled Line Graph
Line Segment Graph
Point Graph

The Line Graph displays one set of data in 2D as a polyline with marker objects identifying each data value. The Y coordinate of a polyline point corresponds to the value of a data sample. Markers may be switched off if desired.

The Filled Line Graph is the same as the Line Graph, but with a filled line. The Line Segment Graph is also similar, but the segments of the polyline may have different colors, widths and line types.

A Point Graph is simply a Line Graph without the lines. Only the markers are displayed.

Animation Data

The entry points for these graphs are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Line Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set

Multiline Graph
Filled Multiline Graph
Multiset Line Segment Graph
Multiset Point Graph

These four graphs are versions of the graphs on page 62 equipped to display data from multiple data sets. They have multiple lines and multiple Y axes with which to chart them.

Animation Data

These graphs' entry points are as follows:

Data Entry for All Lines

$Widget/DataGroupOne/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1. The default number of data groups is two.

Data Entry for One Line

To animate just one of the data groups, use this resource in place of the above data entry point:

$Widget/DataGroupOne/DataGroup<N>/EntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Multiline Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Multiaxis Resource Set
- Zooming Resource Set
- Legend Resource Set

XY Line Graph
XY Point Graph

The XY Line Graph displays one set of (X,Y) data pairs in 2D as a polyline with markers identifying each data pair. The markers may be switched off if desired. The X and Y coordinates of the polyline's points correspond to X and Y values of the input data pairs. The XY Point Graph is the same as the XY Line Graph, but only the markers are displayed. The XY Graph is also known as a scatter graph.

Animation Data

The entry points for these graphs are as follows:

Data Entry

$Widget/DataGroup/XEntryPoint
$Widget/DataGroup/YEntryPoint

The default data range is between -1 and 1.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Xy Line Datagroup Resource Set

- X Level Resource Set
- Y Level Resource Set
- Status Object Resource Set
- X Value Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set

Multiline XY Graph
Multiset Point XY Graph

These two graphs are versions of the graphs on page 64 equipped to display data from multiple data sets. There is only one set of axes with this graph, so the various data sets should have the same data range. For different data ranges, see page 66.

Animation Data

The entry points for these graphs are as follows:

Data Entry for All Lines

$Widget/DataGroupOne/XEntryPoint
$Widget/DataGroupOne/YEntryPoint

The default data range is between -1 and 1. The default number of data groups is two.

Data Entry for One Line

To animate just one of the data groups, use these resource in place of the above data entry points:

$Widget/DataGroupOne/DataGroup<N>/XEntryPoint
$Widget/DataGroupOne/DataGroup<N>/YEntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Multiline Xy Datagroup Resource Set
- X Level Resource Set
- Y Level Resource Set
- Status Object Resource Set

- X Value Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set
- Legend Resource Set

Multiaxis Multiline XY Graph

The Multiaxis Multiline XY Graph displays several sets of (X,Y) data pairs in 2D as polylines with markers identifying each data pair. Markers may be switched off if desired. The X and Y coordinates of the polyline's points correspond to X and Y values of data pairs. The graph has several axes to display different sets of data in different ranges. See page 64 for the single data set version of this graph.

The Special widgets set contain additional axes objects that may be added to any graph.

Animation Data

The graph's entry points are as follows:

Data Entry for All Graphs

$Widget/DataGroupOne/XEntryPoint
$Widget/DataGroupOne/YEntryPoint

The default data range is between -1 and 1. The default number of data groups is two.

Data Entry for One Line

To animate just one of the data groups, use these resource in place of the above data entry points:

$Widget/DataGroupOne/DataGroup<N>/XEntryPoint
$Widget/DataGroupOne/DataGroup<N>/YEntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Multiline Xy Datagroup Resource Set
- X Level Resource Set
- Y Level Resource Set
- Status Object Resource Set

- X Value Axis Resource Set
- Y Value Multiaxis Resource Set
- Zooming Resource Set
- Legend Resource Set

Polar Line Graph
Polar Filled Line Graph
Polar Line Segment Graph
Polar Point Graph

The Polar Line Graph displays one set of data in polar coordinates as a polyline with markers at each data value. The value of the data sample controls the radial distance of the polyline's point from the center of the graph. Markers may be switched off if desired.

The Polar Filled Line Graph is the same as the Polar Line Graph, but with a filled line. The Polar Line Segment Graph is also similar, but the segments of the polyline may have different colors, widths and line types.

The Polar Point Graph is the same as the Polar Line Graph, but displays just the markers.

Animation Data

The entry points for these graphs are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- Polar Data Area Resource Set
- Line Datagroup Resource Set
- Polar Level Resource Set
- Status Object Resource Set
- Tangent Ticks And Labels Resource Set
- X Axis Resource Set
- Y Axis Resource Set
- Zooming Resource Set

Multiline Polar Graph

The Multiline Polar Graph displays several sets of data in polar coordinates as polylines with markers at each data value. The value of the data sample controls the radial distance of the polyline's point from the center of the graph. Markers may be switched off if desired. This is a multiline version of the graph on page 67.

Animation Data

The entry points for these graphs are as follows:

Data Entry for All Lines

$Widget/DataGroupOne/EntryPoint

The default data range is between -1 and 1. The default number of data groups is two.

Data Entry for One Line

To animate just one of the data groups, use these resource in place of the above data entry points:

$Widget/DataGroupOne/DataGroup<N>/EntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Polar Data Area Resource Set
- Multiline Datagroup Resource Set
- Polar Level Resource Set
- Status Object Resource Set
- Tangent Ticks And Labels Resource Set
- X Axis Resource Set
- Y Axis Resource Set
- Zooming Resource Set

XY Polar Line Graph

XY Polar Line Graph: Displays one set of data pairs in 2D in polar coordinates as a polyline with markers identifying each data value. One value of a pair controls the angle and the other controls the radial distance of the polyline's point from the center of the graph. Markers may be switched off if desired.

Animation Data

The graph's entry points are as follows:

Data Entry

$Widget/DataGroup/AngleEntryPoint
$Widget/DataGroup/RadiusEntryPoint

The default radius range is between 0 and 1. The default angle range is between 0 and 360.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Polar Data Area Resource Set
- Polar Xy Line Datagroup Resource Set
- Polar Level Resource Set
- Radial Grid Resource Set
- Tangent Grid Resource Set
- Tangent Labels Resource Set
- Zooming Resource Set

Multiline Polar XY Graph

The Multiline Polar XY Graph is a multiple data set version of the Polar XY Graph described on page 68.

Animation Data

The graph's entry points are as follows:

Data Entry for All Lines

$Widget/DataGroupOne/AngleEntryPoint
$Widget/DataGroupOne/RadiusEntryPoint

The default radius data range is between 0 and 1, while the angle spans between 0 and 360. The default number of data groups is two.

Data Entry for One Line

To animate just one of the data groups, use these resources in place of the above data entry points:

$Widget/DataGroupOne/DataGroup<N>/AngleEntryPoint
$Widget/DataGroupOne/DataGroup<N>/RadiusEntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/AngleEntryPoint*.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Polar Data Area Resource Set
- Multiline Polar Xy Datagroup Resource Set
- Polar Level Resource Set
- Radial Grid Resource Set
- Tangent Grid Resource Set
- Tangent Labels Resource Set
- Zooming Resource Set

Pie Chart

The Pie Chart displays one set of data as arc segments. A pie segment selection may be indicated by shifting the segment outside of the pie chart.

Animation Data

The entry points for these graphs are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

The default data range is between 0 and any positive number.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- Pie Datagroup Resource Set
- Zooming Resource Set
- Legend Resource Set

Combination Graph

The Combination Graph is a Bar Graph Widget with a Multiset Line Graph overlaid on top of the bars.

Animation Data

The entry points for the bar graph part of the combination are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1.

The entry point for the Multiset Line Graph component of the Combination Graph is as follows:

$Widget/DataGroupOne/EntryPoint

The default number of data sets is two.

Resource Sets

This widgets use resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Datagroup Resource Set
- Multiline Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set

Stock Graphs

The Stock Graphs are used to show current and past values of a data series. In addition to displaying current data with a line or marker, a stock graph can show the range of data over one or two given time periods. There are two stock graphs. The simpler graph, *stock2.g*, displays a marker at some “current” value, and can display a high and low limit with a vertical line behind that marker. The more complex stock graph, *stock1.g*, displays a multiset line graph with bars and lines to display two different sets of limit information in the background.

Animation Data

The entry points for the *stock2.g* graphs are as follows:

Data Entry

$Widget/DataGroup/MarkEntryPoint
$Widget/DataGroup/HighEntryPoint
$Widget/DataGroup/LowEntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 20000 and 100000.

The entry points for the *stock1.g* graph are as follows:

Data Entry

$Widget/DataGroupOne/EntryPoint
$Widget/DataGroup/OpenEntryPoint
$Widget/DataGroup/CloseEntryPoint
$Widget/DataGroup/HighEntryPoint
$Widget/DataGroup/LowEntryPoint
$Widget/DataGroup/MarkEntryPoint

The label entry point is the same as for the *stock2.g* graph.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Multiline Datagroup Resource Set
- Level Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Zooming Resource Set

3D Bar Graph
3D Signed Bar Graph
3D Histogram Graph
Pyramid Graph
Prism Graph
Cylinder Graph

All the graphs in this group display one set of data as an array of 3D objects. The height of an object corresponds to the value of its data sample. The 3D Signed Bar displays negative values as well as positive. The difference between the other graphs is only the shape of the objects representing the data. The Pyramid Graph uses little pyramids, and the Cylinder Graph uses cylinders and so on.

Animation Data

The entry points for these graphs are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- 3D Datagroup Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Viewing Resource Set

Multiset 3D Bar Graph
Multiset 3D Signed Bar Graph
Multiset Pyramid Graph
Multiset Prism Graph
Multiset Cylinder Graph

These graphs are multiset variations of that graphs described on page 73.

Animation Data

The entry points for these graphs are as follows:

Data Entry for All DataGroups

$Widget/DataGroupOne/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1. The default number of data groups is two.

Data Entry for One DataGroup

To animate just one of the data groups, use this resource in place of the above data entry point:

$Widget/DataGroupOne/DataGroup<N>/EntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- Multiset 3D Datagroup Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Z Axis Resource Set
- Viewing Resource Set

3D Packed Bar Graph
3D Packed Signed Bar Graph

The 3D Packed Bar Graph displays several sets of data as packs of vertical 3D bars. One pack contains one iteration of data from each of the data sets. The height of a bar corresponds to the value of its data sample. The signed version accepts negative values as well as positive.

Animation Data

The entry points for these graphs are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1, and the default number of bars in a pack is three.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- 3D Packed Datagroup Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Viewing Resource Set

Multiset 3D Packed Bar Graph
Multiset 3D Packed Signed Bar Graph

Displays several sets of data as several series of packs of vertical 3D bars. One pack of bars keeps corresponding samples of one set. The height of a bar corresponds to the value of a data sample.

The signed version accepts negative values as well as positive.

Animation Data

The entry points for these graphs are as follows:

Data Entry for All DataGroups

$Widget/DataGroupOne/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1. The default number of data groups is two, with three bars in each pack.

Data Entry for One DataGroup

To animate just one of the data groups, use this resource in place of the above data entry point:

$Widget/DataGroupOne/DataGroup<N>/EntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

To animate just one of the data packs, you might use a resource that looks like this:

$Widget/DataGroupOne/DataGroup1/Pack2/EntryPoint

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- Multiset 3D Packed Datagroup Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Z Axis Resource Set
- Viewing Resource Set

3D Stacked Bar Graph

The 3D Stacked Bar Graph displays several sets of data as packs of stacked vertical 3D bars. The corresponding samples of each set are placed on top of one another to form one pack of stacked bars. Each stack contains one iteration of data from each data set. The height of every component of a stack corresponds to the value of its data sample, and the total height of the stack equals the sum of the heights of its individual components.

The range of the graph should accommodate the range of the sum of the individual component data sets. This means that the sum of all values corresponding to one stack should be less than or equal to the total graph range.

Animation Data

The graph's entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 0.2, and there are five bars in a stack, by default.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- 3D Packed Datagroup Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Viewing Resource Set

Multiset 3D Stacked Bar Graph

The Multiset 3D Stacked Bar Graph is a multiset version of the Stacked Bar Graph described on page 76.

Animation Data

The graph's entry points are as follows:

Data Entry for All DataGroups

$Widget/DataGroupOne/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 0.2. The default number of bars in a stack is 5 and the number of data groups is 2.

Data Entry for One DataGroup

To animate just one of the data groups, use this resource in place of the above data entry point:

$Widget/DataGroupOne/DataGroup<N>/EntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- Multiset 3D Packed Datagroup Resource Set
- Status Object Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Z Axis Resource Set
- Viewing Resource Set

3D Step Graph

The 3D Step Graph displays one set of data in 3D as a step line. The height of a step corresponds to the value of a data sample.

Animation Data

The graph's entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- 3D Datagroup Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Viewing Resource Set

Multiset 3D Step Graph

The Multiset 3D Step Graph displays several sets of data in 3D as step lines. The height of a step corresponds to the value of a data sample.

Animation Data

The graph's entry points are as follows:

Data Entry for All DataGroups

$Widget/DataGroupOne/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1. The default number of data groups is two.

Data Entry for One DataGroup

To animate just one of the data groups, use this resource in place of the above data entry point:

$Widget/DataGroupOne/DataGroup<N>/EntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- Multiset 3D Datagroup Resource Set
- X Time Axis Resource Set

- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Z Axis Resource Set
- Viewing Resource Set

Ribbon Graph

The Ribbon Graph displays one set of data in 3D as a set of points connected by a ribbon. The height of a point corresponds to the value of a data sample. You can adjust the width of the ribbon by editing the polygon that is its series template .

Animation Data

The graph's entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- Datagroup Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Viewing Resource Set

Multiset Ribbon Graph

The Multiset Ribbon Graph displays several sets of data in 3D as data points connected by a ribbon. The height of a point corresponds to the value of a data sample.

Animation Data

The graph's entry points are as follows:

Data Entry for All DataGroups

$Widget/DataGroupOne/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1. The default number of data groups is two.

Data Entry for One DataGroup

To animate just one of the data groups, use this resource in place of the above data entry point:

$Widget/DataGroupOne/DataGroup<N>/EntryPoint

Use the number of the desired data group in place of <*N*>, as in *$Widget/DataGroupOne/DataGroup3/EntryPoint*.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- Multiset Datagroup Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Z Axis Resource Set
- Viewing Resource Set

Framed Surface Graph Unframed Surface Graph

These graphs display one set of data in 3D as a surface. The surface area is defined by a parallelogram. The parallelogram is then divided into number of nodes by the grid lines parallel to its sides. Every node is elevated by the amount corresponding to the value of a data sample. Drawing markers can be drawn on the graph. Markers are always drawn on the top of the surface.

To form a surface graph, every set of four nodes is connected by a line, creating a parallelogram. This parallelogram is close to a planar one only if the number of data samples is big and the value changes from one data sample to the next is small in both X and Y directions.

The difference between two graphs is in the number of context objects. The Unframed Surface Graph has only the axes, ticks and labels in addition to the surface. The Framed Surface Graph also has a frame made from the polygons and horizontal grids.

Animation Data

The entry points for these graphs are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

The default data range is between 0 and 1. The default graph has 11 column and row nodes, giving 121 total nodes. Use these parameters to control the *datagen* test program.

Resource Sets

These widgets use resources from the following sets:

- Title Resource Set
- 3D Data Area Resource Set
- Surface Datagroup Resource Set
- X Time Axis Resource Set
- Y Value Axis Resource Set
- Y1 Value Axis Resource Set
- Z Value Axis Resource Set
- Viewing Resource Set

Gantt Chart

Displays constrained time intervals as horizontal bars. The beginning of the next interval is constrained to coincide with the end of the previous interval.

The sum of all time interval values of the graph should be less than or equal to the total range of the graph.

Animation Data

The graph’s entry points are as follows:

Data Entry

$Widget/DataGroup/EntryPoint

Label Entry

$Widget/XLabelGroup/EntryPoint

The default data range is between 0 and 1. However, the default graph has 10 time intervals in it, so you should test the program with a data range between 0 and 0.1 so that the sum of all the time intervals is less than 1.

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Data Area Resource Set
- Datagroup Resource Set
- X Axis Resource Set
- Y Axis Resource Set
- Zooming Resource Set

Horizontal 3D Slider Graph

The Horizontal 3D Slider Graph is identical to the Horizontal Slider Graph described on page 83, except that the active element is a 3D bar. It displays one data value at a time, and the horizontal dimension of the bar corresponds to the value of the data.

Animation Data

The graph's entry points are as follows:

Data Entry

$Widget/DataSample/ValueX

The default data range is between 0 and 1.

The 3D Slider Graph may also be used as a control widget, in which case it can be animated without any data by using its reaction to mouse movement.

Resources

ActiveElementGroup (GROUP): data sample
  ActiveElement (POLYGON): one polygon of a data sample bar
    ValueX (DDATA): the value defining a position of a data sample
    Low (DDATA): low range of graph values
    High (DDATA): high range of graph values
    StartColor (GDATA): color for the minimum value
    EndColor (GDATA): color for the maximum value

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- X Value Axis Resource Set

- Value Label Resource Set

Vertical 3D Slider Graph

The Vertical 3D Slider Graph is identical to the Vertical Slider Graph described on page 83, except that the active element is a 3D bar. It displays one data value at a time, and the horizontal dimension of the bar corresponds to the value of the data.

Animation Data

The graph's entry points are as follows:

Data Entry

$Widget/DataSample/ValueY

The default data range is between 0 and 1.The widget may also be used as a control widget, in which case it can be animated without any data by using its reaction to mouse movement.

Resources

ActiveElementGroup (GROUP): data sample
  ActiveElement (POLYGON): one polygon of a data sample bar
    ValueY (DDATA): the value defining a position of a data sample
    Low (DDATA): low range of graph values
    High (DDATA): high range of graph values
    StartColor (GDATA): color for the minimum value
    EndColor (GDATA): color for the maximum value

Resource Sets

This widget uses resources from the following sets:

- Title Resource Set
- Y Value Axis Resource Set
- Value Label Resource Set

Chapter 3

2D and 3D Graph Resource Sets

3

This chapter contains descriptions of resource sets of the 2D and 3D Graph widgets. Refer to the *Using the GLG Widgets* chapter on page 11 for information on all other widget sets.

The resources of the 2D and 3D Graph widgets are arranged in groups for convenience. These groupings are only conventions established by the widget creators; only a few of them have a functional meaning.

For example, if it is noted that a widget has the resources from the "Datagroup Resource Set," this means that all the resources named in that set are used in the resource hierarchy of that widget. Since many of the widgets are similar in nature, they share the same sets of resources.

The lists of resources in this chapter list the name of the resource, and then (in parentheses) the type of the object named by the resource. If the object has children objects, these are listed immediately below the parent. The description of the Datagroup Resource Set is shown below:

DataGroup (SERIES): The graph data samples.
  DataSample (POLYGON): The data sample template object.
    Value (DDATA): A value defining the height of a data sample.
    Low (DDATA): The lower limit of graph values.
    High (DDATA): The upper limit of graph values.
    Truncate (DDATA): Controls rendering values outside of the range.
  Factor (DDATA): The number of data samples.
  ScrollType (DDATA): The scrolling type of the graph.
  EntryPoint (DDATA): The entry point for updating graph data samples.

The *DataGroup* resource indicates a series containing four resources: *DataSample*, *Factor*, *ScrollType*, and *EntryPoint*. The *DataSample* object itself contains four resources: *Value*, *Low*, and *High*. The *Value* resource may thus be indicated by *DataGroup/DataSample/Value*. The *Value* resource is a double-precision scalar value, indicated by the "DDATA" in the following parentheses. (Geometrical values are indicated by GDATA, and string values by SDATA.) If a resource is optional or constrained to equal another value, these facts are indicated in the parentheses.

Resource Sets

The resource sets used by the 2D and 3D Graph widgets are listed below:

Title

Title (TEXT): The title of the graph.

Data Area

DataArea (POLYGON): The area where data are displayed.

Polar Data Area

DataArea (ARC): The area where data are displayed.

3D Data Area

To specify a three-dimensional data area, three mutually perpendicular rectangles are used.

DataArea (POLYGON): The back data area. In its normal position, this is a rectangle in the XY plane.

DataAreaLeft (POLYGON): The left data area, a rectangle in the YZ plane.

DataAreaBottom (POLYGON): The bottom data area, a rectangle in the XZ plane.

Datagroup

- *DataGroup* (SERIES): The graph data samples.
  - *DataSample* (POLYGON): The data sample template object..
    - *Value* (DDATA): A value defining the height of a data sample..
    - *Low* (DDATA): The lower limit of graph values.
    - *High* (DDATA): The upper limit range of graph values.
    - *Truncate* (DDATA): Controls rendering values outside of the range.
  - *Factor* (DDATA): The number of data samples.
  - *ScrollType* (DDATA): The scrolling type of the graph.
  - *EntryPoint* (DDATA): The data entry point for updating graph data samples.

Multiset Datagroup

A Multiset datagroup is simply a series of Datagroups.

- *DataGroupOne* (SERIES): The graph's data sets.
  - *DataGroup* (SERIES): The graph data samples.
    - *DataSample* (POLYGON): The template polygon for the data samples.
      - *Value* (DDATA): A value defining the height of a data sample.
      - *Low* (DDATA): The lower limit of graph values.
      - *High* (DDATA): The upper limit of graph values.
      - *Truncate* (DDATA): Controls rendering values outside of the range.
    - *Factor* (DDATA): The number of data samples.
    - *ScrollType* (DDATA): The scrolling type of the graph.
    - *Inversed* (DDATA): Controls the scrolling direction.
    - *EntryPoint* (DDATA): The data entry point for updating graph data samples within a single Datagroup.
  - *Factor* (DDATA): The number of datagroups in the series.
  - *Persistent* (DDATA): May be set to preserve attribute settings of individual lines.
  - *EntryPoint* (DDATA): The entry point for updating all datagroups.

Packed Datagroup

DataGroup (SERIES): The graph data samples.

Pack (SERIES): The template for one cluster (pack) of data samples.

DataSample (POLYGON): The data sample template polygon.

Value (DDATA): A value defining the height of a data sample.

Low (DDATA): The lower limit of graph values.

High (DDATA): The upper limit of graph values.

Truncate (DDATA): Controls rendering values outside of the range.

Factor (DDATA): The number of data samples in a pack.

Persistent (DDATA): May be set to preserve attribute settings of individual samples in a pack.

Factor (DDATA): The number of data sample clusters.

ScrollType (DDATA): The scrolling type of the graph.

Inversed (DDATA): Controls the scrolling direction.

EntryPoint (DDATA): The entry point for updating graph data samples.

Line Datagroup

These resources are used for simple line graphs.

DataGroup (POLYLINE): The graph data samples.

Marker (MARKER): A template object defining the marker to use at each data point.

DataSample (GDATA): The position of the marker.

Value (DDATA): A value defining the height of a data sample.

Low (DDATA): The lower limit of graph values.

High (DDATA): The upper limit of graph values.

Truncate (DDATA): Controls rendering values outside of the range.

Polygon (POLYGON): A template polygon defining the characteristics of the polyline line segments.

Markers (GROUP, optional): The created copies of the template marker. These will not appear unless the *Marker* resource is present.

Polygons (GROUP, optional): The created instances of line segments. These will not appear unless the *Polygon* resource is present.

Points (GROUP): The group of dynamically created points.

DataSample<n> (GDATA): The created instances of the data sample template.

Angle (DDATA, optional): The angle spanned by a polar polyline.

Factor (DDATA): The number of data samples on the polyline.

ScrollType (DDATA): The scrolling type of the graph.

Inversed (DDATA): Controls the scrolling direction.

EntryPoint (DDATA): The entry point for updating graph data samples.

Multiline Datagroup

These resources are used for graphs with more than one line of data drawn in the same data area.

- *DataGroupOne* (SERIES): The graph’s data sets.
  - *DataGroup* (POLYLINE): The template for one data set.
    - *Marker* (MARKER): The template for the parent object’s markers.
      - *DataSample* (GDATA): The template point of the polyline.
        - *Value* (DDATA): A value defining the height of a data sample.
        - *Low* (DDATA): The lower limit of graph values.
        - *High* (DDATA): The upper limit of graph values.
        - *Truncate* (DDATA): Controls rendering values outside of the range.
    - *Polygon* (POLYGON): The template for the parent object’s line segments.
    - *Markers* (GROUP, optional): The dynamically created copies of the template marker. These will not appear unless the *Marker* resource is present.
    - *Polygons* (GROUP, optional): The created instances of line segments. These will not appear unless the *Polygon* resource is present.
    - *Points* (GROUP): The group of dynamically created points.
      - *DataSample*<n> (GDATA): The dynamically created copies of the data sample template.
    - *Angle* (DDATA, optional): The angle spanned by a polar polyline.
    - *Factor* (DDATA): The number of data samples.
    - *ScrollType* (DDATA): The scrolling type of the graph.
    - *Inversed* (DDATA): Controls the scrolling direction.
    - *EntryPoint* (DDATA): The entry point for updating the data samples of one datagroup.
  - *Factor* (DDATA): The number of datagroups in the graph.
  - *Persistent* (DDATA): May be set to preserve attribute settings of individual lines.
  - *EntryPoint* (DDATA): The data entry point for updating the data samples of all the datagroups in the graph.

XY Line Datagroup

- *DataGroup* (POLYLINE): The graph data samples.
  - *Marker* (MARKER): The template for the parent object’s markers.
    - *DataSample* (GDATA): The position of the marker.
      - *XValue* (DDATA): The X coordinate of a data sample.
      - *XLow* (DDATA): The lower limit of X values.
      - *XHigh* (DDATA): The upper limit of X values.
      - *YValue* (DDATA): The Y coordinate of a data sample.
      - *YLow* (DDATA): The lower limit of Y values.
      - *YHigh* (DDATA): The upper limit of Y values.
      - *Truncate* (DDATA): Controls rendering values outside of the range.
  - *Polygon* (POLYGON): The template for the parent object’s line segments.
  - *Markers* (GROUP, optional): The dynamically created copies of the template marker. These will not appear unless the *Marker* resource is present.
  - *Polygons* (GROUP, optional): The created instances of line segments. These will not appear

unless the *Polygon* resource is present.

Points (GROUP): The group of dynamically created points.

DataSample<n> (GDATA): The dynamically created copies of the data sample template.

Factor (DDATA): The number of data samples.

ScrollType (DDATA): The scrolling type of the graph.

XEntryPoint (DDATA): The data entry point for the X values.

YEntryPoint (DDATA): The data entry point for the Y values.

Multiline XY Datagroup

Title (TEXT): The title of the graph.

DataArea (POLYGON): The area in which data are displayed.

DataGroupOne (SERIES): The series containing all the graph's data sets.

DataGroup (POLYLINE): The template datagroup for the graph.

Marker (MARKER): The marker template for the polyline.

DataSample (GDATA): The position of the marker.

XValue (DDATA): The X coordinate of a data sample.

XLow (DDATA): The lower limit of X values.

XHigh (DDATA): The upper limit of X values.

YValue (DDATA): The Y coordinate of a data sample.

YLow (DDATA): The lower limit of Y values.

YHigh (DDATA): The upper limit of Y values.

Truncate (DDATA): Controls rendering values outside of the range.

Polygon (POLYGON): The template for the parent object's line segments.

Markers (GROUP, optional): The dynamically created copies of the template marker. These will not appear unless the *Marker* resource is present.

Polygons (GROUP, optional): The created instances of line segments. These will not appear unless the *Polygon* resource is present.

Points (GROUP): The group of dynamically created points.

DataSample<n> (GDATA): The dynamically created copies of the data sample template.

Factor (DDATA): The number of data samples.

ScrollType (DDATA): The scrolling type of the graph.

XEntryPoint (DDATA): The data entry point for the X values of one set.

YEntryPoint (DDATA): The data entry point for the Y values of one set.

Factor (DDATA): The number of datagroups in the graph.

Persistent (DDATA): May be set to preserve attribute settings of individual lines.

XEntryPoint (DDATA): The data entry point for the X values of all sets.

YEntryPoint (DDATA): The data entry point for the Y values of all sets.

Filled Step Datagroup

DataGroup (SERIES): The graph data samples.

DataSample (GROUP): The template group.

Fill (POLYGON): The filled part of a data sample.

    - *Edge* (POLYGON): The edge of a data sample.
    - *Value* (DDATA): A value defining the height of a data sample.
    - *Low* (DDATA): The lower limit of graph values.
    - *High* (DDATA): The upper limit of graph values.
    - *Truncate* (DDATA): Controls rendering values outside of the range.
  - *Factor* (DDATA): The number of data samples.
  - *ScrollType* (DDATA): The scrolling type of the graph.
  - *Inversed* (DDATA): Controls the scrolling direction.
  - *EntryPoint* (DDATA): The entry point for updating graph data samples.

Multiline Step Datagroup

- *DataGroupOne* (SERIES): The series of the data sets that compose the graph.
  - *DataGroup* (SERIES): The template data set for the graph.
    - *DataSample* (POLYGON): The data sample template object.
      - *Value* (DDATA): A value defining the height of a data sample.
      - *Low* (DDATA): The lower limit of graph values.
      - *High* (DDATA): The upper limit of graph values.
      - *Truncate* (DDATA): Controls rendering values outside of the range.
    - *Factor* (DDATA): The number of data samples.
    - *ScrollType* (DDATA): The scrolling type of the graph.
    - *Inversed* (DDATA): Controls the scrolling direction.
    - *EntryPoint* (DDATA): The data entry point for updating samples of one datagroup.
  - *Factor* (DDATA): The number of datagroups in the graph.
  - *Persistent* (DDATA): May be set to preserve attribute settings of individual lines.
  - *EntryPoint* (DDATA): The data entry point for updating samples of all the datagroups.

Filled Multiline Step Datagroup

- *DataGroupOne* (SERIES): The collection of datagroups that make up the graph.
  - *DataGroup* (SERIES): The template datagroup.
    - *DataSample* (GROUP): The data sample template object.
      - *Fill* (POLYGON): The filled part of a data sample.
      - *Edge* (POLYGON): The edge of a data sample.
      - *Value* (DDATA): A value defining the height of a data sample.
      - *Low* (DDATA): The lower limit of graph values.
      - *High* (DDATA): The upper limit of graph values.
      - *Truncate* (DDATA): Controls rendering values outside of the range.
    - *Factor* (DDATA): The number of data samples.
    - *ScrollType* (DDATA): The scrolling type of the graph.
    - *Inversed* (DDATA): Controls the scrolling direction.
    - *EntryPoint* (DDATA): The data entry point for updating samples of one datagroup.
  - *Factor* (DDATA): The number of datagroups in the graph.
  - *Persistent* (DDATA): May be set to preserve attribute settings of individual lines.

EntryPoint (DDATA): The data entry point for updating samples of all the datagroups.

Pie Datagroup

DataGroup (SERIES): The graph data samples.

- *DataSample* (GROUP): The data sample template object.
  - *Sector* (ARC): The arc segment of a pie chart.
  - *Label* (TEXT): A label showing the value of that segment.
    - *Format* (SDATA): The format (C-style) for displaying the *Label* value.
  - *SplitFactor* (DDATA): When set to 1, the pie segment will be drawn separated from the rest of the pie. When set to zero, the pie will contact its neighbors.
- *Factor* (DDATA): The number of data samples (pieces of the pie).
- *EntryPoint* (DDATA): The entry point for updating data samples.

Polar XY Line Datagroup

DataGroup (POLYLINE): The graph data samples.

- *Marker* (MARKER): The marker to use for the series template.
  - *DataSample* (GDATA): The position of the marker.
    - *RadValue* (DDATA): The data point’s distance from the center of the graph.
    - *AngleValue* (DDATA): The angle described by the data point, the center of the graph, and the X axis.
    - *Low* (DDATA): The lower limit of the data point’s radius value.
    - *High* (DDATA): The upper limit of the data point’s radius value.
    - *Truncate* (DDATA): Controls rendering values outside of the range.
- *Polygon* (POLYGON): The polyline’s template polygon.
- *Markers* (GROUP, optional): The dynamically created copies of the template marker. These will not appear unless the *Marker* resource is present.
- *Polygons* (GROUP, optional): The created instances of line segments. These will not appear unless the *Polygon* resource is present.
- *Points* (GROUP): The group of dynamically created points.
  - *DataSample*<n> (GDATA): The dynamically created copies of the data sample template.
- *Factor* (DDATA): The number of data samples.
- *ScrollType* (DDATA): The scrolling type of the graph.
- *AngleEntryPoint* (DDATA): The entry point for updating the *AngleValue* resource of a the collection of data samples
- *RadiusEntryPoint* (DDATA): The entry point for updating the *RadValue* resource of the collection of data samples.

Multiline Polar XY Datagroup

DataGroupOne (SERIES): The collection of polylines that make up the graph.

- *DataGroup* (POLYLINE): The polyline template.
  - *Marker* (MARKER): The template for the parent object’s markers.
    - *DataSample* (GDATA): The position of the marker.

RadValue (DDATA): The data point's distance from the center of the graph.

AngleValue (DDATA): The angle described by the data point, the center of the graph, and the X axis.

Low (DDATA): The lower limit of the data point's radius value.

High (DDATA): The upper limit of the data point's radius value.

Truncate (DDATA): Controls rendering values outside of the range.

Polygon (POLYGON): The template for the parent object's line segments.

Markers (GROUP, optional): The dynamically created copies of the template marker. These will not appear unless the *Marker* resource is present.

Polygons (GROUP, optional): The created instances of line segments. These will not appear unless the *Polygon* resource is present.

Points (GROUP): The group of dynamically created points.

DataSample<n> (GDATA): The dynamically created copies of the data sample template.

Factor (DDATA): The number of data samples.

ScrollType (DDATA): The scrolling type of the graph.

AngleEntryPoint (DDATA): The entry point for updating the *AngleValue* resource of one set of data samples

RadiusEntryPoint (DDATA): The entry point for updating the *RadValue* resource of one set of data samples.

Factor (DDATA): The number of datagroups (polylines) in the graph.

Persistent (DDATA): May be set to preserve attribute settings of individual lines.

AngleEntryPoint (DDATA): The angle entry point for updating all of the data samples.

RadiusEntryPoint (DDATA): The radius entry point for updating all of the data samples.

3D Datagroup

DataGroup (SERIES): The graph data samples.

DataSample (GROUP): The data sample template object. This is a collection of several polygons arranged in three-dimensional space.

Element (POLYGON): The template polygon whose attributes are constrained to the attributes of all the polygons in the *DataSample* group.

Value (DDATA): A value defining the height of a data sample.

Low (DDATA): The lower limit of graph values.

High (DDATA): The upper limit of graph values.

Truncate (DDATA): Controls rendering values outside of the range.

Factor (DDATA): The number of data samples.

ScrollType (DDATA): The scrolling type of the graph.

Inversed (DDATA): Controls the scrolling direction.

EntryPoint (DDATA): The entry point for updating graph data samples.

Multiset 3D Datagroup

DataGroupOne (SERIES): The collection of the graph's data sets.

DataGroup (SERIES): A template datagroup.

DataSample (GROUP): The data sample template object. This is a collection of several polygons arranged in three-dimensional space.

Element (POLYGON): The template polygon whose attributes are constrained to the attributes of all the polygons in the *DataSample* group.

Value (DDATA): A value defining the height of a data sample.

Low (DDATA): The lower limit of graph values.

High (DDATA): The upper limit of graph values.

Truncate (DDATA): Controls rendering values outside of the range.

Factor (DDATA): The number of data samples.

ScrollType (DDATA): The scrolling type of the graph.

Inversed (DDATA): Controls the scrolling direction.

EntryPoint (DDATA): The entry point for updating one data set.

Factor (DDATA): The number of datagroups in the graph.

EntryPoint (DDATA): The entry point for updating all data sets.

3D Packed Datagroup

DataGroup (SERIES): The graph data samples.

Pack (SERIES): The template for one cluster (pack) of data samples.

DataSample (GROUP): The data sample template object.

Element (POLYGON): One polygon component of a composite data sample object. The attributes of all the component polygons are constrained to attributes of this element.

Value (DDATA): A value defining the height of a data sample.

Low (DDATA): The lower limit of graph values.

High (DDATA): The upper limit of graph values.

Truncate (DDATA): Controls rendering values outside of the range.

Factor (DDATA): The number of data samples. in a pack

Factor (DDATA): The number of data sample clusters.

ScrollType (DDATA): The scrolling type of the graph.

Inversed (DDATA): Controls the scrolling direction.

EntryPoint (DDATA): The entry point for updating graph data samples.

Multiset 3D Packed Datagroup

DataGroupOne (SERIES): The collection of data groups that make up the graph.

DataGroup (SERIES): The collection of data samples.making up one component of the graph.

Pack (SERIES): The template for one cluster (pack) of data samples.

DataSample (GROUP): The data sample template object.

Element (POLYGON): One polygon component of a composite data sample object. The attributes of all the component polygons are constrained to the attributes of this element.

Value (DDATA): A value defining the height of a data sample.

Low (DDATA): The lower limit of graph values.

High (DDATA): The upper limit of graph values.

Truncate (DDATA): Controls rendering values outside of the range.

Factor (DDATA): The number of data samples. in a pack

Factor (DDATA): The number of data sample clusters.

ScrollType (DDATA): The scrolling type of the graph.

Inversed (DDATA): Controls the scrolling direction.

EntryPoint (DDATA): The entry point for updating graph data samples.

Factor (DDATA): The number of datagroups in the graph.

EntryPoint (DDATA): The data entry point for updating all data sets.

Surface Datagroup

DataGroup (SERIES): The graph data samples.

DataSample (GDATA): The data sample for a surface graph is the 3-D coordinates of a vertex point of the surface.

Value (DDATA): A value defining the height (Y-coordinate) of a data sample.

Low (DDATA): The lower limit of graph values.

High (DDATA): The upper limit of graph values.

Truncate (DDATA): Controls rendering values outside of the range.

Polygon (POLYGON): The template polygon for the surface facets. The colors and line characteristics are copied from this template.

Polygons (GROUP): The group that contains the dynamically created facets of the surface.

Polygon<n> (POLYGON): Each facet of the surface is individually addressable with this resource. Note that, as with all GLG indexes, this index starts at zero.

Marker (MARKER): The template for the markers that appear at the surface vertices.

Markers (GROUP): The group containing the dynamically markers (if markers are enabled).

Marker<n> (MARKER): Each marker of a surface is individually addressable with this resource.

Points (GROUP): The group containing the positions of each vertex in the graph surface.

DataSample<n> (GDATA): An instance of the data sample with index <n> (zero based).

Value (DDATA): A value defining the height of a data sample.

Low (DDATA): The lower limit of graph values.

High (DDATA): The upper limit of graph values.

Rows (DDATA): The number of rows of data samples.

Columns (DDATA): The number of columns of data samples.

EntryPoint (DDATA): The entry point for updating graph data samples.

Level

Level objects are used to draw horizontal lines on a bar graph. They will often be used to indicate some externally defined data level.

LevelObjectGroup (SERIES): This is the series of level lines for the graph.

LevelObject (POLYGON): The template level line provides the line characteristics for the line.

Value (DDATA): The height of a level line

Low (DDATA, constrained): The lower limit of graph values.

High (DDATA, constrained): The upper limit of graph values.

Factor (DDATA): The number of level lines in the graph.

LevelObject<n> (POLYGON): The nth level line.

Value (DDATA): The height of a level line.

Low (DDATA, constrained): The lower limit of graph values.

High (DDATA, constrained): The upper limit of graph values.

Symmetrical Level

These level lines are used to mark pairs of level lines that are symmetrical around a given level.

LevelObjectOne (SERIES): A series of level line pairs.

LevelObjectGroup (GROUP): The template for a pair of level lines.

LevelObject1 (POLYGON): The upper level line.

Value (DDATA): The height of the upper level line.

High (DDATA, constrained): The upper limit of graph values.

LevelObject2 (POLYGON): The lower level line.

Value (DDATA, constrained): The height of the lower level line.

High (DDATA, constrained): The upper limit of graph values.

Factor (DDATA): number of level line sets

Polar Level

These lines are used on polar graphs.

LevelObjectGroup (SERIES): A series of level lines.

LevelObject (ARC): The template level line. The line’s characteristics, like weight and type, are taken from this template.

Value (DDATA): The radius of a level line.

Low (DDATA, constrained): The lower limit of graph values.

High (DDATA, constrained): The upper limit of graph values.

Factor (DDATA): The number of level lines.

X Level

XLevelObjectGroup (SERIES): A series of vertical (X) level lines.
  XLevelObject (POLYGON): The template X level line.
    Value (DDATA): The X-coordinate of the line.
    XLow (DDATA, constrained): The lower limit of X values in the graph.
    XHigh (DDATA, constrained): The upper limit of X values in the graph.
  Factor (DDATA): The number of X level lines in the graph.

Y Level

YLevelObjectGroup (SERIES): A series of horizontal (Y) level lines.
  YLevelObject (POLYGON): The template Y level line.
    Value (DDATA): The Y-coordinate of the line.
    YLow (DDATA, constrained): The lower limit of Y values in the graph.
    YHigh (DDATA, constrained): The upper limit of Y values in the graph.
  Factor (DDATA): The number of Y level lines.

Status Object

A status object is used to point to the update position for a wrapped graph.

StatusObject (POLYGON): Shows current update state. The origin of the polygon object is placed at the axis intercept of the current update position.

Value Axis

This is the general-purpose axis resource group. More specific resource groups (e.g. X-axis, Y-axis, X-axis for data values, X-axis for time values, and so on) are described below.

Axis (POLYGON): The line representing the axis itself.
AxisLabel (TEXT, optional): The label for the axis.
Base (POLYGON, optional): A polygon defining the positioning of axis ticks. The position of this polygon’s first point coincides with the first tick on the axis.
MajorGroup (SERIES): The series of major ticks.
  MajorTick (POLYGON): A template for the axis major ticks.
  Factor (DDATA): The number of major ticks.
MinorGroupOne (SERIES, optional): The series that contains all the axis minor ticks.
  MinorGroup (SERIES): The template for one set of minor ticks (between two major ticks).
    MinorTick (POLYGON): The template for an individual minor tick.
    Factor (DDATA): The number of minor ticks between major ticks.
    LogType (DDATA): Controls logarithmic positioning of minor ticks.
  Factor (DDATA, constrained): The number of X major ticks.
LabelGroup (SERIES): The series of major tick labels.
  Label (TEXT): A template for the major tick labels.
    Format (SDATA): The C-style format to use for displaying label values.

Low (DDATA, constrained): The lower limit of the graph's data.

High (DDATA, constrained): The upper limit of the graph's data.

Factor (DDATA, constrained): The number of major ticks.

X Axis

These resources control the appearance of a graph's X axis. Note that the axis labels in this group and other similar ones are arranged in a series with an enclosed history object. (For more about the history object, see [*].) This arrangement is provided as a convenient way to set individual attributes of series members, and has no functional relation to the graph's *EntryPoint* resource.

XAxis (POLYGON): The line representing the X axis itself.

XMajorGroup (SERIES): The series of X major ticks.

XMajorTick (POLYGON): The template for the X major ticks.

Factor (DDATA): The number of X major ticks.

XMinorGroupOne (SERIES): The series of X minor ticks.

XMinorGroup (SERIES): template for one set of X minor ticks

XMinorTick (POLYGON): The template for the X minor ticks.

Factor (DDATA): The number of X minor ticks.

Factor (DDATA, constrained): The number of X major ticks.

XGridGroup (SERIES, optional): The X axis grid lines.

XGrid (POLYGON): The template grid line.

Factor (DDATA, constrained): The number of X major ticks.

XLabelGroup (SERIES, optional): The series of major tick labels.

XLabel (TEXT): A template major tick label.

Factor (DDATA, constrained): The number of major ticks.

EntryPoint (SDATA, optional): Entry point for simplified setting of label strings.

Y Axis

YAxis (POLYGON): The line representing the Y axis itself.

YMajorGroup (SERIES): The series of Y major ticks.

YMajorTick (POLYGON): The template for the Y major ticks.

Factor (DDATA): The number of Y major ticks.

YMinorGroupOne (SERIES): The series of Y minor ticks.

YMinorGroup (SERIES): template for one set of Y minor ticks

YMinorTick (POLYGON): The template for the Y minor ticks.

Factor (DDATA): The number of Y minor ticks..

Factor (DDATA, constrained): The number of Y major ticks.

YGridGroup (SERIES, optional): The series of Y axis grid lines.

YGrid (POLYGON): The template grid line.

Factor (DDATA, constrained): The number of Y major ticks.

YLabelGroup (SERIES, optional): The series of major tick labels.

YLabel (TEXT): The template major tick label.

Factor (DDATA, constrained): The number of major ticks.

EntryPoint (SDATA, optional): Entry point for simplified setting of label strings.

Z Axis

ZAxis (POLYGON): The line representing the Z axis itself.

ZMajorGroup (SERIES): The series of Z major ticks.

ZMajorTick (POLYGON): The template for the Z major ticks.

Factor (DDATA): The number of Z major ticks.

ZMinorGroupOne (SERIES, optional): The series of Z minor ticks.

ZMinorGroup (SERIES): The template for one set of Z minor ticks.

ZMinorTick (POLYGON): The template for a single Z minor tick.

Factor (DDATA): The number of Z minor ticks.

Factor (DDATA, constrained): The number of Z major ticks.

ZLabelGroup (SERIES, optional): The set of major tick labels.

ZLabel (TEXT): A template major tick label.

Factor (DDATA, constrained): The number of major ticks.

EntryPoint (SDATA, optional): Entry point for simplified setting of label strings.

X Value Axis

XAxis (POLYGON, optional): The line representing the X axis itself.

XAxisLabel (TEXT, optional): The title for the X axis.

XMajorGroup (SERIES): The series of X major ticks.

XMajorTick (POLYGON): The template for the X major ticks.

Factor (DDATA): The number of X major ticks.

XMinorGroupOne (SERIES): The series of X minor ticks.

XMinorGroup (SERIES): template for one set of X minor ticks

XMinorTick (POLYGON): The template for the X minor ticks.

Factor (DDATA): The number of X minor ticks.

LogType (DDATA): Controls logarithmic positioning of minor ticks.

Factor (DDATA, constrained): The number of X major ticks.

XLabelGroup (SERIES): X major tick labels

XLabel (TEXT): The template for the X major ticks. label

Format (SDATA): The C-style format to use for displaying label values.

Low (DDATA, constrained): The lower limit of the graph's data.

High (DDATA, constrained): The upper limit of the graph's data.

Factor (DDATA, constrained): The number of X major ticks.

XGridGroup (SERIES): The series of X axis grid lines.

XGrid (POLYGON): The template grid line.

Factor (DDATA, constrained): The number of X major ticks.

Y Value Axis

- *YAxis* (POLYGON, optional): The line representing the Y axis itself.
- *YAxisLabel* (TEXT, optional): The title for the Y axis.
- *YMajorGroup* (SERIES): The series of Y major ticks.
  - *YMajorTick* (POLYGON): The template for the Y major ticks.
  - *Factor* (DDATA): The number of Y major ticks.
- *YMinorGroupOne* (SERIES): The series of Y minor ticks.
  - *YMinorGroup* (SERIES): template for one set of Y minor ticks
    - *YMinorTick* (POLYGON): The template for the Y minor ticks.
    - *Factor* (DDATA): The number of Y minor ticks.
    - *LogType* (DDATA): Controls logarithmic positioning of minor ticks.
  - *Factor* (DDATA, constrained): The number of Y major ticks.
- *YLabelGroup* (SERIES): Y major tick labels
  - *YLabel* (TEXT): The template for the Y major ticks. label
    - *Format* (SDATA): The C-style format to use for displaying label values.
    - *Low* (DDATA, constrained): The lower limit of the graph's data.
    - *High* (DDATA, constrained): The upper limit of the graph's data.
  - *Factor* (DDATA, constrained): The number of Y major ticks.
- *YGridGroup* (SERIES): The series of Y axis grid lines.
  - *YGrid* (POLYGON): The template grid line.
  - *Factor* (DDATA, constrained): The number of Y major ticks.

Y1 Value Axis

An auxiliary set of ticks and labels. It is the same as *Y Value Axis Resource Set* except that all the resources at the top level of the resource hierarchy that begin with "Y" are replaced with names that begin with "Y1" instead.

Y Value Multiaxis

This set of several Y-axes is designed to be used in graphs that display several different sets of data at the same time.

- *YAxisGroupOne* (SERIES): A series containing several vertical axes.
  - *Factor* (DDATA): The number of Y axes displayed.
  - *YAxisGroup* (GROUP): The template for one vertical axis with ticks and labels.
    - *YAxis* (POLYGON): The line representing the Y axis itself.
    - *YAxisLabel* (TEXT): The title for the Y axis.
    - *YMajorGroup* (SERIES): The series of Y major ticks.
      - *YMajorTick* (POLYGON): The template for the Y major ticks.
      - *Factor* (DDATA): The number of Y major ticks.
    - *YMinorGroupOne* (SERIES): The series of Y minor ticks.
      - *YMinorGroup* (SERIES): template for one set of Y minor ticks

- - - *YMinorTick* (POLYGON): The template for the Y minor ticks.
      - *Factor* (DDATA): The number of Y minor ticks.
      - *LogType* (DDATA): controls logarithmic positioning of minor ticks
    - *Factor* (DDATA, constrained): The number of Y major ticks.
- *YLabelGroup* (SERIES): Y major tick labels
    - *YLabel* (TEXT): The template for the Y major ticks. label
        - *Format* (SDATA): The C-style format to use for displaying label values.
        - *Low* (DDATA, constrained): The lower limit of the graph's data.
        - *High* (DDATA, constrained): The upper limit of the graph's data.
    - *Factor* (DDATA, constrained): The number of Y major ticks.
- *YGridGroup* (SERIES): The Y axis grid lines.
    - *YGrid* (POLYGON): The template grid line.
    - *Factor* (DDATA, constrained): The number of Y major ticks.

Z Value Axis

- *ZAxis* (POLYGON, optional): The line representing the Z axis itself.
- *ZAxisLabel* (TEXT): The title for the Z axis.
- *ZMajorGroup* (SERIES): The series of Z major ticks.
    - *ZMajorTick* (POLYGON): The template for the Z major ticks.
    - *Factor* (DDATA): The number of Z major ticks.
- *ZMinorGroupOne* (SERIES): The series of Z minor ticks.
    - *ZMinorGroup* (SERIES): The template for one set of Z minor ticks.
        - *ZMinorTick* (POLYGON): The template for a single Z minor tick.
        - *Factor* (DDATA): The number of Z minor ticks.
    - *Factor* (DDATA, constrained): The number of Z major ticks.
- *ZLabelGroup* (SERIES): The Z major tick labels.
    - *ZLabel* (TEXT): The template for the Z major ticks.
        - *Format* (SDATA): format to use for displaying label values
        - *Low* (DDATA): The lower limit of the Z axis.
        - *High* (DDATA): The upper limit of the Z axis.
    - *Factor* (DDATA, constrained): The number of Z major ticks.

X Time Axis

- *XAxis* (POLYGON, optional): The line representing the X axis itself.
- *XAxisLabel* (TEXT): The title for the X axis.
- *XMajorGroup* (SERIES): The series of X major ticks.
    - *XMajorTick* (POLYGON): The template for the X major ticks.
    - *Factor* (DDATA): The number of X major ticks.
    - *ScrollType* (DDATA, constrained): The scrolling type of the graph.
    - *Inversed* (DDATA, constrained): Controls the direction of the graph's scrolling behavior.
- *XMinorGroupOne* (SERIES): The series of X minor ticks.
    - *XMinorGroup* (SERIES): The template for one set of X minor ticks.

XMinorTick (POLYGON): The template for the X minor ticks.

Factor (DDATA): The number of X minor ticks.

Factor (DDATA, constrained): The number of X major ticks.

XLabelGroup (SERIES): X major tick labels

XLabel (TEXT): The template for the X major ticks labels.

Factor (DDATA, constrained): The number of X major ticks.

MinorFactor (DDATA, constrained): The number of X minor ticks.

ScrollType (DATA, constrained): The scrolling type of the graph.

Inversed (DDATA, constrained): Controls the direction of the labels' scrolling behavior.

EntryPoint (SDATA, optional): The entry point for setting labels' strings.

XGridGroup (SERIES): The series of X axis grid lines.

XGrid (POLYGON): The template grid line.

Factor (DDATA, constrained): The number of X major ticks.

MinorFactor (DDATA, constrained): The number of X minor ticks.

ScrollType (DATA, constrained): The scrolling type used for the grid lines.

Inversed (DDATA, constrained): Controls the direction of the grid lines' scrolling behavior.

Y Time Axis

YAxis (POLYGON, optional): The line representing the Y axis itself.

YAxisLabel (TEXT): The title for the Y axis.

YMajorGroup (SERIES): The series of Y major ticks.

YMajorTick (POLYGON): The template for the Y major ticks.

Factor (DDATA): The number of Y major ticks.

ScrollType (DATA, constrained): The scrolling type of the graph.

Inversed (DDATA, constrained): Controls the direction of the axis ticks' scrolling behavior.

YMinorGroupOne (SERIES): The series of Y minor ticks.

YMinorGroup (SERIES): template for one set of Y minor ticks

YMinorTick (POLYGON): The template for the Y minor ticks.

Factor (DDATA): The number of Y minor ticks.

Factor (DDATA, constrained): The number of Y major ticks.

YLabelGroup (SERIES): Y major tick labels

YLabel (TEXT): The template for the Y major ticks. label

Factor (DDATA, constrained): The number of Y major ticks.

MinorFactor (DDATA, constrained): The number of Y minor ticks.

ScrollType (DATA, constrained): The scrolling type of the graph.

Inversed (DDATA, constrained): Controls the direction of the labels' scrolling behavior.

EntryPoint (SDATA, optional): The data entry point provided for simplified setting of label strings.

YGridGroup (SERIES): The Y axis grid lines.

YGrid (POLYGON): The template grid line.

Factor (DDATA, constrained): The number of Y major ticks.

MinorFactor (DDATA, constrained): The number of Y minor ticks.

ScrollType (DATA, constrained): The scrolling type of the graph.

Inversed (DDATA, constrained): Controls the direction of the labels' scrolling behavior.

Tangent Ticks

The tangent and radial groups are used for polar graph grids. The tangent axes are the ones that go in a circle around the origin, while the radial axes radiate outwards from the origin.

- *TangentMajorGroup* (SERIES): The series of tangent major ticks.
  - *TangentMajorTick* (POLYGON): The template tangent major tick.
  - *Factor* (DDATA): The number of tangent major ticks.
- *TangentMinorGroupOne* (SERIES): The series of tangent minor ticks.
  - *TangentMinorGroup* (SERIES): The template for one set of tangent minor ticks (to be drawn between an adjacent pair of major ticks).
    - *TangentMinorTick* (POLYGON): The template for a single tangent minor tick.
    - *Factor* (DDATA): The number of tangent minor ticks between adjacent major ticks.
  - *Factor* (DDATA, constrained): The number of tangent major ticks.

Tangent Labels

- *TangentLabelGroup* (SERIES): The series of tangent labels.
  - *TangentLabel* (TEXT): The template tangent label.
    - *Format* (SDATA): The C-style format to use for displaying label values.
    - *Low* (DDATA, constrained): The lower limit of the label values.
    - *High* (DDATA, constrained): The upper limit of the label values.
  - *Factor* (DDATA, constrained): The number of tangent labels.

Tangent Grid

- *TangentGridGroup* (SERIES): A series of tangent grid lines.
  - *TangentGrid* (POLYGON): The template grid line.
  - *Factor* (DDATA): The number of tangent grid lines.

Radial Grid

- *RadialGridGroup* (SERIES): A series of radial grid lines.
  - *RadialGrid* (ARC): The template grid line.
  - *Factor* (DDATA): The number of radial grids.

Legend

- *LegendObject* (VIEWPORT): legend viewport
  - *LegendGroup* (GROUP): a group of legend elements
    - *Legend* (GROUP): template legend element
      - *LegendBox* (POLYGON, optional): legend box
      - *LegendLine* (POLYGON, optional): legend line
      - *LegendMarker1* (POLYGON, optional): legend line marker
      - *LegendMarker2* (POLYGON, optional): legend line marker
      - *LegendLabel* (TEXT): legend text object

Bevel

Body (GROUP): set of bevels
  Element (POLYGON, optional): bevel polygon
  BevelSize (DDATA, optional): the width of button bevels in pixels
Insert (VIEWPORT, optional): a viewport to place objects inside bevels
  FrameSize (DDATA): the width of the gap between the bevels and the Insert viewport.
  InverseShading (DDATA, optional): inverses bevel shading when set to 1.

View Resources

These resource sets control the view a user has of the widget.

Viewing

The angles used to specify the viewing transformation are defined relative to the "main view. " This view of a drawing has the X-axis pointing to the right, the Y-axis pointing up, and the Z-axis pointing directly at the viewer.

PanCenter (GDATA): This point is drawn in the center of the top-level viewport.
Scale (DDATA): The zoom factor for a drawing. A value of 1.0 indicates that the corners of the top-level viewport will have the coordinates *(-1000 -1000)* and *(1000 1000)*.
XAngle (DDATA): The angle of rotation around the X axis from the main view.
YAngle (DDATA): The angle of rotation around the Yaxis from the main view.
ZAngle (DDATA): The angle of rotation around the Z axis from the main view.
ShearFactor (DDATA): The degree of shear along the Z axis.
ShearX (DDATA): The proportion of the specified shear in the X direction.
ShearY (DDATA): The proportion of the specified shear in the Y direction.

Zooming

PanCenter (GDATA): This point is drawn in the center of the top-level viewport.
Scale (DDATA): The zoom factor for a drawing. A value of 1.0 indicates that the corners of the top-level viewport will have the coordinates *(-1000 -1000)* and *(1000 1000)*.

Index

3D bar graph, 73
3D color datagroup resource set, 92
3D data area resource set, 86
3D datagroup resource set, 92
3D histogram graph, 73
3D packed bar graph, 75
3D packed signed bar graph, 75
3D signed bar graph, 73
3D stacked bar graph, 76
3D step graph, 78

A

Alarm Browser, 26
axis, 47

B

bar graph, 54
- 3D multiple data sets, 74
- 3D packed, 75
- 3D stacked, 76
- horizontal, 54
- multiset 3D packed, 75
- packed, 56
- stacked, 57
- symmetrical, 55

bevel resource set, 103
button widget, 20

C

chart widgets, 14
clock widget, 25
Commonly Used Graph Resources, 41
cylinder graph, 73

D

data area, 46
data area resource set, 85
data samples, 42
datagroup resource set, 86

F

filled line graph, 62
filled multiline graph, 63
filled multiline step datagroup resource set, 90
filled multiline step graph, 61
filled step graph, 58
font browser widget, 26
framed surface graph, 81

G

gantt chart, 82
graph components, 39

H

histogram graph, 54
horizontal 3D slider graph, 83
horizontal bar graph, 54
horizontal filled step graph, 60
horizontal step graph, 59

K

knob widget, 18

L

labels, 48
Layout Templates, 36
legend resource set, 102
legends, 52
level objects, 51
level resource set, 95
line datagroup resource set, 87
line graph, 62
- multiline polar, 69
- multiple lines, 63
- polar, 67, 68

line segment graph, 62
logarithmic ticks, 48

M

major ticks, 47
minor ticks, 47
multiaxis multiline XY graph, 66
multiline datagroup resource set, 88
multiline graph, 63
multiline polar graph, 68
multiline polar XY datagroup resource set, 91
multiline polar XY graph, 69
multiline step datagroup resource set, 90
multiline step graph, 60
multiline XY datagroup resource set, 89
multiline XY graph, 65
Multiple Axes, 52

multiset 3D bar graph, 74
multiset 3D datagroup resource set, 92
multiset 3D packed bar graph, 75
multiset 3D packed datagroup resource set, 93
multiset 3D packed signed bar graph, 75
multiset 3D signed bar graph, 74
multiset 3D stacked bar graph, 77
multiset 3D step graph, 79
multiset cylinder graph, 74
multiset datagroup resource set, 86
multiset line segment graph, 63
multiset point graph, 63
multiset point XY graph, 65
multiset prism graph, 74
multiset pyramid graph, 74
multiset ribbon graph, 80

N

naming conventions, 39

P

packed bar graph, 56
packed color datagroup resource set, 87
palette widget, 26
pie chart, 70
pie datagroup, 91
point graph, 62
polar data area resource set, 86
polar filled line graph, 67
polar level resource set, 95
polar line graph, 67
polar line segment graph, 67
polar point graph, 67
polar XY line datagroup resource set, 91
prism graph, 73
pyramid graph, 73

R

radial grid resource set, 102
real-time chart, 14
resource
  naming conventions, 39
Resource Browser Widget, 26
resource set
  3D data area, 86
  3D datagroup, 92
  3D packed datagroup, 93
  bevel, 103
  data area, 85
  datagroup, 86
  filled multiline step datagroup, 90
  legend, 102
  level, 95
  line datagroup, 87
  multiline datagroup, 88
  multiline polar XY datagroup, 91
  multiline step datagroup, 90
  multiline Xy datagroup, 89
  multiset 3D datagroup, 92
  multiset 3D packed datagroup, 93
  multiset datagroup, 86
  packed color datagroup, 87
  pie datagroup, 91
  polar data area, 86
  polar level, 95
  polar Xy line datagroup, 91
  radial grid, 102
  status object, 96
  surface datagroup, 94
  symmetrical level, 95
  tangent grid, 102
  tangent labels, 102
  tangent ticks and labels, 102
  title, 85
  value axis, 96
  viewing, 103
  X axis, 97
  X level, 96
  X time axis, 100
  X value axis, 98
  Xy line datagroup, 88
  Y axis, 97
  Y level, 96
  Y time axis, 101
  Y value axis, 99
  Y value multiaxis, 99
  Y1 value axis, 99
  Z axis, 98
  Z value axis, 100
  zooming, 103
ribbon graph, 80

S

Scrolling Time Axis Labels, 50
signed bar graph, 54
slider
  horizontal 3D graph, 83
  horizontal graph, 83
  vertical 3D graph, 84
  vertical graph, 83

slider widget, 18
spinner widget, 24
stacked bar graph, 57
status object resource set, 96
status objects, 51
step graph, 57
stopwatch widget, 25
surface datagroup resource set, 94
symmetrical bar graph, 55
symmetrical level resource set, 95

T

Tag Browser Widget, 26
tangent grid resource set, 102
tangent labels resource set, 102
tangent ticks and labels resource set, 102
text objects, 48
text widget, 23
title resource set, 85
titles, 51
Truncate, 43

U

unframed surface graph, 81
updating time axis labels, 50

V

value axis resource set, 96
value labels, 49
vertical 3D slider graph, 84
viewing resource set, 103

X

X axis resource set, 97
X level resource set, 96
X time axis resource set, 96
Xy line datagroup resource set, 88
XY line graph, 64
XY point graph, 64
XY polar line graph, 68

Y

Y axis resource set, 97
Y level resource set, 96
Y time axis resource set, 101
Y value axis resource set, 99
Y value multiaxis resource set, 99
Y1 value axis resource set, 99

Z

Z axis resource set, 98
Z value axis resource set, 100
zooming resource set, 103