User Interface Examples

Measurement Studio delivers several examples for User Interface. Click on the following links for example descriptions and locations:

- 2D Graph Events
- Advanced Button
- Advanced Knob
- Advanced Slide
- Annotation Bookmarking
- Axes Advanced
- Axis
- Clock
- Color Substitution
- Cursors
- CWBinding Features
- CWBinding Reader
- Graph Styles
- Import/Export Styles
- Knob Format
- Lissajous
- Lorenz Attractors
- Range Checking
- Scaled Plots
- Simple Annotations
- Simple Button
- Simple CWBinding
- Simple Graph
- Simple Knob
- Simple Numeric Edit
- Simple Slide
- Slide Format
- Sweep Chart
- UI Formats
- UI Statistics
- Visualizing Data
- XYGraph
Click the **Hide/Show** button in the toolbar to hide or show the Visual Basic table of contents.
**Button Overview**

Represents different types of Boolean controls on a user interface. A Boolean control displays an on or off state (True or False). Buttons are often used to input or output Boolean information or initiate an action in your program.

- Different display styles: toggle switches, LEDs, push buttons, slides, and on/off buttons.
- Custom bitmap buttons.
- Button modes specify how the Button control responds to user input. For example, you can make a button respond only programmatically (that is, not respond to any user input). Or you can click the button to temporarily change its value and then release to revert the button to its original state (called a latch). Finally, you can click on the button to change its value until you click on it again.
- Built-in format styles for the labels, including scientific, symbolic engineering, scaling, time, and date.
- Animation - you can animate different parts of the control. For example, you might want to animate the text on a stop button.
- Custom background images.
- Three-dimensional button style.
Notes

- As you are working in the Images property page, you are setting properties for the button in the state (On or Off) that it appears in the preview window. If the button is off in the preview window, then you are setting the caption, off text, on text, and image properties for when the button is off. If the button is on in the preview window, then you are setting the caption, off text, on text, and image properties for when the button is on. For example, click the button on in the preview window and load the bitmap for the button's On state. Click the button off in the preview window to load the bitmap for the Off state. It doesn't matter what state the button is in when you are setting the background image. The background image is either visible or invisible for both the On and Off state.
Tips

- To get information about any of the properties in the button control's property pages, right click on the property and select **What's This?**. For complete reference information about this control and its properties, click on the Visual Basic or Visual C++ Reference link.

- In Visual C++, if you select the **ALL** tab in the property pages, the What's This? help is disabled.
Click the **Hide/Show** button in the toolbar to hide or show the Visual Basic table of contents.
Graph Overview

Plots and charts two-dimensional data on a user interface.

- Plots or charts. Plotting data refers to the process of taking a large number of points and updating one or more plots on the graph with new data; the old plot is replaced with the new plot. Charting data appends new data points to an existing plot over time. Charting is used with slow processes where only few data points per second are added to the graph. When the number of data points exceeds the number of points that can be displayed on the graph, the graph scrolls so that new points are added to the right side of the graph while old points disappear to the left.

- Multiple plot styles: point, line, line-point, and bar.

- Multiple plots with individual properties such as name, line and point style, width, and base value.

- Cursors, which display a crosshair on a graph to mark a specific point or region on the graph or highlight data.

- Configurable axes, including customizable ticks, labels, value pairs, and captions. You can also have multiple Y axes and use axis autoscaling.

- Built-in format styles for labels, including scientific, symbolic engineering, scaling, time, and date.

- Panning and zooming at runtime. Panning is useful when the graph displays only a subset of the data that has been plotted. You can scroll through all data plotted on the graph, essentially shifting the graph's display to different portions of the plot. You can use zooming to enlarge or diminish a portion of the plot displayed by the graph. For example, if you zoom on a section of a plot, the graph displays a smaller portion of the plot in the same amount of display area, which enlarges the detail of that section.

- Three-dimensional style.
Notes

- Set the TrackMode property to allow user interaction with the graph, such as panning, zooming, and moving cursors, while your program is running.
Tips

- To get information about any of the properties in the graph control's property pages, right click on the property and select **What's This?**. For complete reference information about this control and its properties, click on the Visual Basic or Visual C++ Reference link.

- In Visual C++, if you select the **ALL** tab in the property pages, the What's This? help is disabled.
Click the **Hide/Show** button in the toolbar to hide or show the Visual Basic table of contents.
Knob Overview

Represents different types of circular displays on a user interface. A knob accepts and displays individual or multiple scalar values.

- Different display styles: dials, gauges, and meters.
- Automatic axis labeling with numeric scales (log or inverted) and values (continuous or discrete).
- Multiple pointers, each one representing one scalar value. A pointer indicates the current value of the knob.
- Custom ticks, labels, and value pairs. Ticks are the divisions that represent increments on the knob. Labels display the value of each tick. Value pairs are names paired with a value. For example, use a value pair to add the text label "Boiling Point" to a numeric axis at the value 212 °F.
- Built-in format styles for the labels, including scientific, symbolic engineering, scaling, time, and date.
- Animation. You can animate different parts of the control. For example, you might want to animate a pointer if its value exceeds a safe limit in the application.
- Custom images. Add images to different parts of the control, including the background and pointer(s).
- Three-dimensional knob style.
**Tips**

- To get information about any of the properties in the knob control's property pages, right click on the property and select **What's This?**. For complete reference information about this control and its properties, click on the Visual Basic or Visual C++ Reference link.

- In Visual C++, if you select the **ALL** tab in the property pages, the What's This? help is disabled.
Click the **Hide/Show** button in the toolbar to hide or show the Visual Basic table of contents.
Numeric Edit Overview

Displays numbers on a user interface as they would be displayed in a text box. The numeric edit control uses scalars rather than ASCII characters, so you don't have to coerce strings into numbers. Furthermore, the values are automatically formatted as numbers, and you can apply built-in numeric format styles.

- Range checking.
- Increment and decrement buttons.
- Control or indicator display style - A control accepts input from users, while an indicator only displays its current value.
- Built-in numeric format styles, including scientific, symbolic engineering, scaling, time, and date.
- Three-dimensional numerical edit style.
Tips

- To get information about any of the properties in the numeric edit control's property pages, right click on the property and select What's This?. For complete reference information about this control and its properties, click on the Visual Basic or Visual C++ Reference link.
- In Visual C++, if you select the ALL tab in the property pages, the What's This? help is disabled.
Click the **Hide/Show** button in the toolbar to hide or show the Visual Basic table of contents.
Slide Overview

Represents different types of linear displays on a user interface. A slide accepts and displays individual or multiple scalar values.

- Different display styles: vertical and horizontal slides, tanks, and thermometers.
- Automatic axis labeling with numeric scales (log or inverted) and values (continuous or discrete).
- Multiple pointers, each one representing one scalar value.
- Custom ticks, labels, and value pairs - ticks are the divisions that represent values on the slide. Labels display the value of each tick. Value pairs are names paired with a value. For example, use a value pair to add the text label "Boiling Point" to the value 212 °F.
- Built-in format styles for the labels, including scientific, symbolic engineering, scaling, time, and date.
- Animation - you can animate different parts of the control. For example, you might want to animate a pointer if its value exceeds a safe limit in the application.
- Custom images - add images to different parts of the control, including the background and pointer(s).
- Three-dimensional slide style.
Tips

- To get information about any of the properties in the slide control's property pages, right-click on the property and select **What's This?**. For complete reference information about this control and its properties, click on the Visual Basic or Visual C++ Reference link.

- In Visual C++, if you select the **ALL** tab in the property pages, the What's This? help is disabled.
**CWAnnotation**

The CWAnnotation object displays text, a shape, and/or an arrow on a CWGraph to mark a specific point or region on the graph or highlight something programmatically. CWAnnotation properties define the position and appearance of the annotation. The annotation position corresponds either to the coordinate space of the associated plot or to the screen coordinates of the CWGraph control. To interact with the annotations while the program is running, set the TrackMode property to cwGTrackDragAnnotation.
### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrow</td>
<td>Returns a CWArrow object, which specifies the arrow part of the annotation.</td>
</tr>
<tr>
<td>Caption</td>
<td>Returns a CWCaption object, which specifies the text part of the annotation.</td>
</tr>
<tr>
<td>CoordinateType</td>
<td>Specifies how the text and shape coordinates are scaled when the annotation is drawn.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies if the CWAnnotation generates mouse events or if you can drag the annotation.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies the name of the annotation.</td>
</tr>
<tr>
<td>Plot</td>
<td>Specifies the plot that the annotation is associated with.</td>
</tr>
<tr>
<td>PointIndex</td>
<td>Specifies the index of a plot point to center the shape around.</td>
</tr>
<tr>
<td>Shape</td>
<td>Returns a CWShape object, which specifies the shape part of the annotation.</td>
</tr>
<tr>
<td>SnapMode</td>
<td>Specifies if the annotation's shape is centered around the plot point defined by the Plot and PointIndex properties and how the shape is dragged by the mouse.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies if an annotation is visible.</td>
</tr>
</tbody>
</table>
Methods

SetBuiltInStyle  Sets many properties of the annotation to represent the new style specified.
See Also

CWAnotations
CWAAnnotations

CWAAnnotations is a collection of CWAAnnotation objects.
Properties

**Count**

Returns the number of objects in the collection.
Methods

- **Add**: Adds an object to the collection and returns the new object.
- **Item**: Returns the specified object from the collection.
- **Remove**: Removes the specified item from the collection.
- **RemoveAll**: Removes all objects from the collection.
See Also

CWAAnnotation
**CWArrow**

The CWArrow object defines the arrow part of the CWAnnotation object. CWArrow properties define the appearance of the arrow. If the arrow is visible, it points from the text part of the annotation to the shape part.
Properties

**Color**
Specifies the color of the arrow.

**HeadStyle**
Specifies the style of the arrowhead that points to the CWShape object.

**LineStyle**
Specifies the line style of the arrow.

**TailStyle**
Specifies the style of an arrowhead that points to the caption.

**Visible**
Specifies if the arrow is visible.

**Width**
Specifies the width of the arrow.
See Also

CWAnnotation
**CWAxes**

CWAxes is a collection of CWAxis objects. A graph has one X axis and a varying number of Y axes. Usually, the X axis is at index 1, and the Y axes are at subsequent indices.
Properties

Count

Returns the number of objects in the collection.
Methods

- **Add**  
  Adds an object to the collection and returns the new object.

- **Item**  
  Returns the specified object from the collection.

- **Remove**  
  Removes the specified item from the collection.

- **RemoveAll**  
  Removes all objects from the collection.
See Also

CWGraph.Axes
CWAxis
**CWAxis**

The CWAxis object defines the axis for slide, knob, and graph controls. The CWAxis properties determine how data values are scaled to the range displayed on the control, where ticks and grid lines are drawn (if used), and what kinds of labels are placed on the axis.

While some properties are valuable features in a particular control, such as grid lines on the graph, those properties might not be useful or valid for other controls.
## Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoScale</td>
<td>Determines if the minimum and maximum limits of the axis are automatically set.</td>
</tr>
<tr>
<td>Caption</td>
<td>Specifies the text to draw on the axis.</td>
</tr>
<tr>
<td>CaptionColor</td>
<td>Specifies the color used to draw the caption.</td>
</tr>
<tr>
<td>Discrete</td>
<td>Represents only discrete values on the axis, according to the base and interval properties.</td>
</tr>
<tr>
<td>DiscreteBase</td>
<td>Specifies the base value for discrete axes.</td>
</tr>
<tr>
<td>DiscreteInterval</td>
<td>Specifies the interval between discrete values.</td>
</tr>
<tr>
<td>FormatString</td>
<td>Specifies the format string for formatting the labels on the axis.</td>
</tr>
<tr>
<td>Inverted</td>
<td>Specifies if the direction of an axis is inverted.</td>
</tr>
<tr>
<td>Labels</td>
<td>Returns a CWLabel object, which specifies how labels appear on the axis.</td>
</tr>
<tr>
<td>Log</td>
<td>Specifies if the axis has a Log10 scale.</td>
</tr>
<tr>
<td>Maximum</td>
<td>Specifies the maximum value of the axis.</td>
</tr>
<tr>
<td>Minimum</td>
<td>Specifies the minimum value of the axis.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies the name of the axis.</td>
</tr>
<tr>
<td>ScaleStyleValuePairsOnly</td>
<td>Specifies if only the value pairs are displayed on the axis.</td>
</tr>
<tr>
<td>Ticks</td>
<td>Returns a CWTicks object, which specifies how divisions and ticks appear on this axis.</td>
</tr>
<tr>
<td>ValuePairs</td>
<td>Returns a CWValuePairs collection of CWValuePair objects, which specify labels for particular points on the axis.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies if the axis is visible or hidden.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>AutoScaleNow</td>
<td>Causes the axis to rescale immediately.</td>
</tr>
<tr>
<td>SetMinMax</td>
<td>Sets both the minimum and the maximum values of the axis at the same time.</td>
</tr>
</tbody>
</table>
See Also

CWAxes
**CWBinding**

The CWBinding object binds a control to an external data source or target. External data sources and targets might be data items on HTTP, FTP, OPC, DSTP, or file servers located anywhere on the Internet. The CWBinding object associates a property with the external data item. For example, you can create a CWBinding to bind the value of a slide display to a data item on a DataSocket Server to display changes in the data source on the user interface of your program. The CWBinding associates CWSlide.Value with the data location "dstp://localhost/wave".

You can bind most Measurement Studio User Interface control properties to external data sources or targets. For example, you can bind the color of a data plot, a caption name, or a background color to external data items.

The CWBindingDataUpdated event is generated when data is ready to be sent (in write mode) or has just been received (in read mode). Use the CWBindingDataUpdated event to manipulate data values as they are sent or read. For example, you might want to scale values or ignore them if they don't meet your criteria.

A CWBinding object uses National Instruments DataSocket technology to connect to data sources and targets and share live measurements over the Internet. Refer to the DataSocket Web site at www.ni.com/datasocket for more information about DataSocket technology and sharing live data over the Internet.

Note: For security reasons, the CWBinding functionality is disabled when a Measurement Studio User Interface control is running directly within a Web page because the Web page could initialize or script the control to connect to potentially dangerous data sources. To use the CWBinding functionality within a Web page, embed the Measurement Studio User Interface control in a custom control using Visual Basic or Visual Basic Control Creation Edition, and place the custom control on the Web page.
### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AccessMode</strong></td>
<td>Specifies the read or write connection to make when connecting to the data source.</td>
</tr>
<tr>
<td><strong>ActualURL</strong></td>
<td>Identifies the actual URL of the current data source.</td>
</tr>
<tr>
<td><strong>AutoConnect</strong></td>
<td>Specifies if the binding automatically connects to the source as soon as the program is run.</td>
</tr>
<tr>
<td><strong>BindProperty</strong></td>
<td>Specifies the name of the property that you are binding to an external data item.</td>
</tr>
<tr>
<td><strong>DataUpdated</strong></td>
<td>Indicates if value or attributes on the CWBinding object have been set since they were last read.</td>
</tr>
<tr>
<td><strong>DataUpdatedEnabled</strong></td>
<td>Indicates if the binding generates the CWBindingDataUpdated event when the bound data changes.</td>
</tr>
<tr>
<td><strong>LastError</strong></td>
<td>Returns the last error code used in an CWBindingStatusUpdated event.</td>
</tr>
<tr>
<td><strong>LastMessage</strong></td>
<td>Stores the last message used in a CWBindingStatusUpdated event.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Specifies the name of the CWBinding object.</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Specifies the current status of the data connection.</td>
</tr>
<tr>
<td><strong>StatusUpdated</strong></td>
<td>Indicates if the binding status has changed or if an error has occurred.</td>
</tr>
<tr>
<td><strong>TimerInterval</strong></td>
<td>Automatically calls the Update method at a regular interval specified in milliseconds.</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td>Specifies the location as a URL of the data source or target to which you are binding.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connect</strong></td>
<td>Connects the CWBinding to a data source or target.</td>
</tr>
<tr>
<td><strong>ConnectTo</strong></td>
<td>Connects the CWBinding object to a data source or target.</td>
</tr>
<tr>
<td><strong>Disconnect</strong></td>
<td>Disconnects the CWBinding from the data item to which it is currently connected.</td>
</tr>
<tr>
<td><strong>SelectURL</strong></td>
<td>Enables interactive browsing and selection of data items and files on DataSocket and OPC servers. With this method you can easily create data URLs, rather than constructing the URLs yourself, or display a simple text box in which you can enter HTTP and FTP URLs.</td>
</tr>
<tr>
<td><strong>SetBindObject</strong></td>
<td>Object whose property will bind to a data source or target. This object might be either a control or one of the objects on that control.</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Causes the CWBinding to read from a data source or write to a data target.</td>
</tr>
</tbody>
</table>
See Also

CWBindings
CWBIndings

CWBIndings is a collection of CWBinding objects.
Properties

Count

Returns the number of objects in the collection.
## Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Adds an object to the collection and returns the new object.</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Returns the specified object from the collection.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Removes the specified item from the collection.</td>
</tr>
<tr>
<td><strong>RemoveAll</strong></td>
<td>Removes all objects from the collection.</td>
</tr>
</tbody>
</table>
See Also

CWBinding
CWBButton

CWBButton is the top-level object for the Button control.
Properties

- **BackColor**
  Specifies the background color of the button.

- **BackgroundImage**
  Specifies the image to use for the background of the button.

- **Caption**
  Specifies the text that appears in the button.

- **CaptionColor**
  Specifies the color of the caption for the button.

- **CWBindings**
  Returns a collection of CWBinding objects.

- **Enabled**
  Specifies if the control responds to user input.

- **Font**
  Specifies the font for the caption and the two text fields in the button.

- **ImmediateUpdates**
  Specifies if the control draws new data as soon as it is available, or if the form refreshes the control when it draws other controls.

- **KeyboardMode**
  Specifies how the control handles keyboard input from the user.

- **Mode**
  Specifies how the button responds to user input.

- **OffColor**
  Specifies the color of the button in the Off state.

- **OffImage**
  Specifies the picture for the button in the Off state.

- **OffText**
  Specifies the text for the button in the Off state.

- **OffTextColor**
  Specifies the color of the text for the button in the Off state.

- **OnColor**
  Specifies the color of the button in the On state.

- **OnImage**
  Specifies the picture for the button in the On state.

- **OnText**
  Specifies the text for the button in the On state.

- **OnTextColor**
  Specifies the color of the text for the button in the On state.

- **ReadyState**
  Returns the ready state.

- **ShowFocusMode**
  Specifies how the control indicates it has the focus.

- **Value**
  Specifies the current value of the button.

- **Windowless**
  Specifies if the control has a window.
Methods

- **AboutBox**: Displays the About Box for the control.
- **ControlImage**: Returns an image of the entire control.
- **ExportStyle**: Exports the style of the Measurement Studio control to a file.
- **ImportStyle**: Imports a previously exported style.
- **OffImages**: Provides access to the CWImage objects in the CWButton control in the Off state.
- **OnImages**: Provides access to the CWImage objects in the CWButton control in the On state.
- **Refresh**: Redraws the CWButton control.
- **SetBuiltinStyle**: Sets many properties of the control to represent the new style specified.
Events

- **Click**
  Generates when you click the mouse on the control.

- **CWBindingDataUpdated**
  Generated when the binding data is updated.

- **CWBindingStatusUpdated**
  Generated when the status of the binding connection changes.

- **DbClick**
  Generates when you double-click the mouse on the control.

- **KeyDown, KeyUp**
  KeyUp generates when you release a key while the control has the input focus.
  KeyDown generates when you press a key while the control has the input focus.

- **KeyPress**
  Generated when the control has focus and you press a key.

- **MouseDown, MouseMove, MouseUp**
  MouseDown generates when you click the mouse on the control.
  MouseMove generates when you move the mouse over the control.
  MouseUp generates when you release the mouse on the control.

- **ReadyStateChange**
  Generated when the ready state changes.

- **ValueChanged**
  Generates when the value of the button changes.
**CWCaption**

The CWCaption object defines the text part of the CWAnnotation object. CWCaption properties define the position and appearance of the text.
## Properties

<table>
<thead>
<tr>
<th><strong>Alignment</strong></th>
<th>Specifies the alignment of the caption relative to CWCaption.XCoordinate and CWCaption.YCoordinate properties.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Angle</strong></td>
<td>Specifies the angle of the text.</td>
</tr>
<tr>
<td><strong>Color</strong></td>
<td>Specifies the color of the caption text.</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Specifies the font of the caption text.</td>
</tr>
<tr>
<td><strong>Text</strong></td>
<td>Specifies the text of the caption.</td>
</tr>
<tr>
<td><strong>XCoordinate,</strong></td>
<td>Specifies the X and Y coordinates that the caption appears at on the graph.</td>
</tr>
<tr>
<td><strong>YCoordinate</strong></td>
<td></td>
</tr>
</tbody>
</table>
Methods

SetCoordinates

Sets the X and Y coordinates of the caption.
See Also

CWAnotation
CWCursor

The CWCursor object displays a crosshair on a CWGraph to mark a specific point or region on the graph or highlight something programmatically. CWCursor properties define the position and appearance of the cursor. The cursor position corresponds to the coordinate space of the associated plot or to the CWGraph.PlotTemplate if the cursor is not associated with a specific plot. To interact with the cursors while the program is running, set the TrackMode property.
## Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Color</strong></td>
<td>Specifies the color of the cursor crosshair and point.</td>
</tr>
<tr>
<td><strong>CrosshairStyle</strong></td>
<td>Specifies the type of lines that identify the cursor position.</td>
</tr>
<tr>
<td><strong>Enabled</strong></td>
<td>Specifies if the CWCursor object generates mouse events or if you can drag the cursor in cursor tracking mode.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Specifies the name of the cursor.</td>
</tr>
<tr>
<td><strong>Plot</strong></td>
<td>Specifies the plot associated with the cursor.</td>
</tr>
<tr>
<td><strong>PointIndex</strong></td>
<td>Specifies the point associated with the cursor on the plot.</td>
</tr>
<tr>
<td><strong>PointStyle</strong></td>
<td>Specifies the cursor point style.</td>
</tr>
<tr>
<td><strong>SnapMode</strong></td>
<td>Specifies the coordinates available for cursors to align with on a plot.</td>
</tr>
<tr>
<td><strong>Visible</strong></td>
<td>Specifies if the cursor is visible or hidden.</td>
</tr>
<tr>
<td><strong>XPosition</strong></td>
<td>Specifies the current x-axis position of the cursor. The SnapMode property can cause the value to be coerced to a point on one of the current plots.</td>
</tr>
<tr>
<td><strong>YPosition</strong></td>
<td>Specifies the current y-axis position of the cursor. The SnapMode property can cause the value to be coerced to a point on one of the current plots.</td>
</tr>
</tbody>
</table>
Methods

SetPosition  Sets the x- and y-axis positions of the cursor at the same time.
See Also

CWCursors
CWCursors

CWCursors is a collection of CWCursor objects.
Properties

- **Count**: Returns the number of objects in the collection.
Methods

**Add**
Add an object to the collection and returns the new object.

**Item**
Returns the specified object from the collection.

**Remove**
Removes the specified item from the collection.

**RemoveAll**
Removes all objects from the collection.
See Also

CWCursor
CWDData

The CWDData object holds a value and attributes associated with the DataSocket connection.
<table>
<thead>
<tr>
<th>Properties</th>
<th>Value</th>
<th>Specifies the value of a CWData object.</th>
</tr>
</thead>
</table>
## Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyFrom</td>
<td>Copies the value and attributes from another CWData object.</td>
</tr>
<tr>
<td>DeleteAttribute</td>
<td>Deletes an existing attribute if it exists.</td>
</tr>
<tr>
<td>GetAttribute</td>
<td>Gets a CWData object for an attribute.</td>
</tr>
<tr>
<td>GetAttributeNames</td>
<td>Returns the names of the attributes on the CWData object.</td>
</tr>
<tr>
<td>HasAttribute</td>
<td>Returns True if an attribute exists.</td>
</tr>
<tr>
<td>Reset</td>
<td>Clears value and all attributes.</td>
</tr>
<tr>
<td>SetAttribute</td>
<td>Sets the value of an attribute.</td>
</tr>
</tbody>
</table>
CWGraph

The CWGraph object is the top-level object in the graph control and it has properties that affect the overall appearance of the control.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotations</td>
<td>Specifies a collection of CWAnnotation objects.</td>
</tr>
<tr>
<td>AnnotationTemplate</td>
<td>Returns the CWAnnotation object to use as a template for new annotations.</td>
</tr>
<tr>
<td>Axes</td>
<td>Specifies a collection of CWAxis objects.</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color for the graph caption.</td>
</tr>
<tr>
<td>Caption</td>
<td>Specifies the caption that appears on the graph.</td>
</tr>
<tr>
<td>CaptionColor</td>
<td>Specifies the color of the caption.</td>
</tr>
<tr>
<td>ChartLength</td>
<td>Specifies how many points the graph stores when charting before it deletes old data.</td>
</tr>
<tr>
<td>ChartStyle</td>
<td>Specifies how chart methods update the display as new data is added to the plot.</td>
</tr>
<tr>
<td>Cursors</td>
<td>Specifies a collection of CWCursor objects.</td>
</tr>
<tr>
<td>CWBindings</td>
<td>Returns a collection of CWBinding objects.</td>
</tr>
<tr>
<td>DefaultPlotPerRow</td>
<td>Specifies the default value used by the Plot or Chart method if the optional bPlotPerRow or bChartPerRow parameter is omitted.</td>
</tr>
<tr>
<td>DefaultxFirst</td>
<td>Specifies the default X value of the first point in the plot when using the PlotY method. Set this property only if the optional xFirst parameter is omitted for the PlotY method.</td>
</tr>
<tr>
<td>DefaultxInc</td>
<td>Specifies the default value for incrementing when using the PlotY or ChartY method. Set this property only if the optional xInc parameter is omitted for either method.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies if the graph generates any events.</td>
</tr>
<tr>
<td>Font</td>
<td>Specifies the font for labels on all axes.</td>
</tr>
<tr>
<td>GraphFrameColor</td>
<td>Specifies the color for the graph frame.</td>
</tr>
<tr>
<td>GraphFrameImage</td>
<td>Specifies the image for the background of the graph frame.</td>
</tr>
<tr>
<td>GraphFrameStyle</td>
<td>Specifies the style of the graph frame.</td>
</tr>
<tr>
<td>ImmediateUpdates</td>
<td>Specifies if the graph draws new data as soon as it is available, or if the form refreshes the graph when it draws other controls.</td>
</tr>
<tr>
<td>KeyboardMode</td>
<td>Specifies how the control handles keyboard input from the user.</td>
</tr>
<tr>
<td>PlotAreaColor</td>
<td>Specifies the background color of the plot area.</td>
</tr>
<tr>
<td>PlotAreImage</td>
<td>Specifies the image used for the background of the plot area.</td>
</tr>
<tr>
<td>Plots</td>
<td>Specifies a collection of CWPlots.</td>
</tr>
<tr>
<td>PlotTemplate</td>
<td>Returns the CWPlot object to use as a template for new plots.</td>
</tr>
<tr>
<td>ReadyState</td>
<td>Returns the ready state.</td>
</tr>
<tr>
<td>TrackMode</td>
<td>Determines how the mouse interacts with the graph.</td>
</tr>
<tr>
<td>Windowless</td>
<td>Specifies if the control has a window.</td>
</tr>
<tr>
<td>XYData</td>
<td>Displays data from an external source in a plot on a graph as it would if you called the PlotXY method.</td>
</tr>
<tr>
<td>XYDataAppend</td>
<td>Displays data from an external source in a chart as it would if you called the ChartXY method.</td>
</tr>
<tr>
<td>YData</td>
<td>Displays data from an external source in a plot on a graph as it would if you called the PlotY method.</td>
</tr>
<tr>
<td>YDataAppend</td>
<td>Displays data from an external source in a chart as it would if you called the ChartY method.</td>
</tr>
</tbody>
</table>
Methods

**AboutBox**  Displays the About Box for the control.

**ChartXvsY**  Charts a one- or two-dimensional array of Y data against a one-dimensional array of X data. This method is similar to the PlotYvsX method, except that the previously plotted data is not deleted until the amount stored for each chart is greater than the CWGraph.ChartLength property specifies.

**ChartXY**  Charts a two-dimensional array of data. This method is similar to the PlotXY method, except that the previously plotted data is not deleted until the amount stored for each chart is greater than the CWGraph.ChartLength property specifies.

**ChartY**  Charts Y data on one or more plots relative to the index of the data.

**ClearData**  Clears data in all plots.

**ControlImage**  Returns an image of the entire control.

**ExportStyle**  Exports the style of the Measurement Studio control to a file.

**Images**  Provides access to the CWImage objects in the CWGraph control.

**ImportStyle**  Imports a previously exported style.

**PlotXvsY**  Plots a 1D or 2D array of Y data against a 1D array of X data. Each row of Y values generates one plot.

**PlotXY**  Plots a 2D array of data. The first row is X values, and subsequent rows are Y values for each plot.

**PlotY**  Plots Y data evenly spaced on the x axis relative to the index in the array. Alternatively, you can use the xFirst and xInc parameters to specify the X value at the first data point and the incremental X value between data points.

**Refresh**  Redraws the CWGraph control.
Events

**AnnotationChange**
Generates when you reposition an annotation with the mouse.

**AnnotationMouseDown, AnnotationMouseMove, AnnotationMouseUp**
AnnotationMouseDown is generated when you click the mouse on an annotation.
AnnotationMouseMove is generated when you move the mouse over an annotation.
AnnotationMouseUp is generated when you release the mouse over an annotation.

**Click**
Generates when you click the mouse on the control.

**CursorChange**
Generated when you reposition a cursor with the mouse.

**CursorMouseDown, CursorMouseMove, CursorMouseUp**
CursorMouseDown is generated when you click the mouse on a cursor.
CursorMouseMove is generated when you move the mouse over a cursor.
CursorMouseUp is generated when you release the mouse over a cursor.

**CWBindingDataUpdated**
Generated when the binding data is updated.

**CWBindingStatusUpdated**
Generated when the status of the binding connection changes.

**DblClick**
Generates when you double-click the mouse on the control.

**KeyDown, KeyUp**
KeyUp generates when you release a key while the control has the input focus.
KeyDown generates when you press a key while the control has the input focus.

**KeyPress**
Generated when the control has focus and you press a key.

**MouseDown, MouseMove, MouseUp**
MouseDown generates when you click the mouse on the control.
MouseMove generates when you move the mouse over the control.
MouseUp generates when you release the mouse on the control.

**PlotAreaMouseDown, PlotAreaMouseMove, PlotAreaMouseUp**
PlotAreaMouseDown generates when you click the mouse on the plot area.
PlotAreaMouseMove generates when you move the mouse over the plot area.
PlotAreaMouseUp generates when you release the mouse over the plot area.

**PlotMouseDown, PlotMouseMove, PlotMouseUp**
PlotMouseDown generates when you click the mouse on a plot.
PlotMouseMove generates when you move the mouse over a plot.
PlotMouseUp generates when you release the mouse over a plot.
| ReadyStateChange | Generated when the ready state changes. |
**CWIImage**

The CWImage object determines how images are displayed within a control.
## Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnimateColumns</td>
<td>Specifies the number of columns in a bitmap that is being used for animation.</td>
</tr>
<tr>
<td>AnimateInterval</td>
<td>Specifies how often an image animates.</td>
</tr>
<tr>
<td>AnimateRows</td>
<td>Specifies the number of rows in a bitmap that is being used for animation.</td>
</tr>
<tr>
<td>BlinkInterval</td>
<td>Specifies how often the image blinks.</td>
</tr>
<tr>
<td>Color</td>
<td>Specifies the color of the image, or, if you are doing dynamic color substitution in Windows metafiles, the color in the image that you want to replace.</td>
</tr>
<tr>
<td>FlipH</td>
<td>Flips an image horizontally.</td>
</tr>
<tr>
<td>FlipV</td>
<td>Flips an image vertically.</td>
</tr>
<tr>
<td>Picture</td>
<td>Specifies the image used in the CWImage object.</td>
</tr>
<tr>
<td>ReverseAnimation</td>
<td>Specifies the direction of animation.</td>
</tr>
<tr>
<td>SaveLink</td>
<td>Specifies if the CWImage object saves the image itself or a link to the image.</td>
</tr>
<tr>
<td>Stretch</td>
<td>Specifies if the image is displayed normal size or stretched to fit the size available within the control.</td>
</tr>
<tr>
<td>Substitute</td>
<td>Specifies that you want to perform dynamic color substitution in the metafile image if set to True.</td>
</tr>
<tr>
<td>SubstituteColor</td>
<td>Specifies the color that will replace a specified color in the metafile image.</td>
</tr>
<tr>
<td>Tile</td>
<td>Specifies if the image is tiled.</td>
</tr>
<tr>
<td>Tolerance</td>
<td>Specifies the percentage for matching color hues on the image to the SubstituteColor property.</td>
</tr>
<tr>
<td>Transparent</td>
<td>Specifies if the image has a transparent color.</td>
</tr>
<tr>
<td>TransparentColor</td>
<td>Specifies the color in the image that must be drawn as transparent.</td>
</tr>
<tr>
<td>URL</td>
<td>Specifies the path to the image.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies if the image is visible or not.</td>
</tr>
</tbody>
</table>
Methods

Reload Loads the image from the given URL.
**CWKnob**

CWKnob is the top-level object for the Knob control.
## Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivePointer</td>
<td>Specifies the CWPointer object of the active pointer.</td>
</tr>
<tr>
<td>ArcEnd</td>
<td>Specifies the end angle, in degrees, for the knob axis arc.</td>
</tr>
<tr>
<td>ArcStart</td>
<td>Specifies the start angle, in degrees, for the knob axis arc.</td>
</tr>
<tr>
<td>Axis</td>
<td>Returns the CWAxis object for this control.</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color of the control.</td>
</tr>
<tr>
<td>BackgroundImage</td>
<td>Specifies an image to be used for the background of the control.</td>
</tr>
<tr>
<td>Caption</td>
<td>Specifies the text that appears in the control.</td>
</tr>
<tr>
<td>CaptionColor</td>
<td>Specifies the color of the caption.</td>
</tr>
<tr>
<td>CWBindings</td>
<td>Returns a collection of CWBinding objects.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies if the control responds to user input.</td>
</tr>
<tr>
<td>Font</td>
<td>Specifies the font for the caption and axis labels.</td>
</tr>
<tr>
<td>ForeColor</td>
<td>Specifies the foreground color in the CWSlide or CWKnob control.</td>
</tr>
<tr>
<td>ImmediateUpdates</td>
<td>Specifies if the control draws new data as soon as it is available or if the form refreshes the control when it draws other controls.</td>
</tr>
<tr>
<td>IncDecValue</td>
<td>Specifies the amount that the control is changed when the user clicks the increment or decrement buttons, or uses the keyboard to increment or decrement the control.</td>
</tr>
<tr>
<td>KeyboardMode</td>
<td>Specifies how the control handles keyboard input from the user.</td>
</tr>
<tr>
<td>Pointers</td>
<td>Returns a collection of Pointer objects.</td>
</tr>
<tr>
<td>ReadyState</td>
<td>Returns the ready state.</td>
</tr>
<tr>
<td>ScaleStyleValuePairsOnly</td>
<td>Specifies if only the value pairs are displayed on the axis.</td>
</tr>
<tr>
<td>ShowFocusMode</td>
<td>Specifies how the control indicates that it has the focus.</td>
</tr>
<tr>
<td>Statistics</td>
<td>Returns the Statistics object for this control.</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the value of the active pointer.</td>
</tr>
<tr>
<td>ValuePairIndex</td>
<td>Specifies the index of the value pair selected by the active pointer.</td>
</tr>
<tr>
<td>Windowless</td>
<td>Specifies if the control has a window.</td>
</tr>
</tbody>
</table>
Methods

AboutBox          Displays the About Box for the control.
ControlImage      Returns an image of the entire control.
ExportStyle       Exports the style of the Measurement Studio control to a file.
Images            Provides access to the CWImage objects in the CWKnob or CWSlide control.
ImportStyle       Imports a previously exported style.
Refresh           Redraws the knob control.
SetBuiltinStyle   Sets many properties of the control to represent the new style specified.
## Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Click</strong></td>
<td>Generates when you click the mouse on the control.</td>
</tr>
<tr>
<td>CWBindingDataUpdated</td>
<td>Generated when the binding data is updated.</td>
</tr>
<tr>
<td>CWBindingStatusUpdated</td>
<td>Generated when the status of the binding connection changes.</td>
</tr>
<tr>
<td>DblClick</td>
<td>Generates when you double-click the mouse on the control.</td>
</tr>
<tr>
<td><strong>KeyDown, KeyUp</strong></td>
<td>KeyUp generates when you release a key while the control has the input focus.</td>
</tr>
<tr>
<td><strong>KeyPress</strong></td>
<td>KeyDown generates when you press a key while the control has the input focus.</td>
</tr>
<tr>
<td><strong>MouseDown, MouseMove, MouseUp</strong></td>
<td>MouseDown generates when you click the mouse on the control.</td>
</tr>
<tr>
<td></td>
<td>MouseMove generates when you move the mouse over the control.</td>
</tr>
<tr>
<td></td>
<td>MouseUp generates when you release the mouse on the control.</td>
</tr>
<tr>
<td>PointerValueChanged</td>
<td>Generated as the value of a pointer changes from the user interface or the program.</td>
</tr>
<tr>
<td>PointerValueCommitted</td>
<td>Generated when the value of a pointer has stopped changing.</td>
</tr>
<tr>
<td>ReadyStateChange</td>
<td>Generated when the ready state changes.</td>
</tr>
</tbody>
</table>
CWLLabels

The CWLabels object determines how axis labels are drawn. Labels are the numbers displayed next to the ticks. Use the properties to define the placement and color of labels.
# Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Above</strong></td>
<td>Specifies if labels appear above the object.</td>
</tr>
<tr>
<td><strong>Below</strong></td>
<td>Specifies if labels appear below the object.</td>
</tr>
<tr>
<td><strong>Color</strong></td>
<td>Specifies the color of the labels.</td>
</tr>
<tr>
<td><strong>Left</strong></td>
<td>Specifies if labels appear to the left of the object.</td>
</tr>
<tr>
<td><strong>Radial</strong></td>
<td>Specifies if radial labels are drawn.</td>
</tr>
<tr>
<td><strong>Right</strong></td>
<td>Specifies if labels appear to the right of the object.</td>
</tr>
<tr>
<td><strong>Width</strong></td>
<td>Specifies the width of a label on the axis.</td>
</tr>
</tbody>
</table>
See Also

CWAxis.Labels
**CWNumEdit**

CWNumEdit is the top-level object for the Numeric Edit control.
# Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AccelInc</strong></td>
<td>Determines the amount the value increments or decrements according to how long the user holds down the increment or decrement button.</td>
</tr>
<tr>
<td><strong>AccelTime</strong></td>
<td>Specifies the number of seconds the increment or decrement button must be held down before the value is changed by the AccelInc value instead of the IncDecValue.</td>
</tr>
<tr>
<td><strong>Alignment</strong></td>
<td>Specifies how the text within the CWNumEdit control is aligned.</td>
</tr>
<tr>
<td><strong>Appearance</strong></td>
<td>Specifies how the edit box portion of the CWNumEdit control is drawn.</td>
</tr>
<tr>
<td><strong>BackColor</strong></td>
<td>Specifies the frame color and the background color behind the increment and decrement buttons for the CWNumEdit control.</td>
</tr>
<tr>
<td><strong>BackColorText</strong></td>
<td>Specifies the background color of the edit box portion of the CWNumEdit control.</td>
</tr>
<tr>
<td><strong>BorderStyle</strong></td>
<td>Specifies the border around the edit box portion of the CWNumEdit control.</td>
</tr>
<tr>
<td><strong>ButtonColor</strong></td>
<td>Specifies the color of the increment and decrement buttons.</td>
</tr>
<tr>
<td><strong>ButtonStyle</strong></td>
<td>Specifies if the increment and decrement buttons have a three-dimensional style.</td>
</tr>
<tr>
<td><strong>CWBindings</strong></td>
<td>Returns a collection of CWBinding objects.</td>
</tr>
<tr>
<td><strong>Discrete</strong></td>
<td>Specifies if the CWNumEdit control is in discrete or continuous mode.</td>
</tr>
<tr>
<td><strong>DiscreteBase</strong></td>
<td>Specifies the initial value to use in a discrete CWNumEdit control.</td>
</tr>
<tr>
<td><strong>DiscreteInterval</strong></td>
<td>Specifies the interval to use in a discrete CWNumEdit control.</td>
</tr>
<tr>
<td><strong>Enabled</strong></td>
<td>Specifies if the user is able to interact with the control.</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Specifies the font the CWNumEdit control uses.</td>
</tr>
<tr>
<td><strong>ForeColorText</strong></td>
<td>Specifies the color of the text in the CWNumEdit control.</td>
</tr>
<tr>
<td><strong>FormatString</strong></td>
<td>Specifies the string that formats the value in the numeric edit control.</td>
</tr>
<tr>
<td><strong>IncDecButtonPosition</strong></td>
<td>Specifies the location of the increment and decrement buttons of the CWNumEdit control.</td>
</tr>
<tr>
<td><strong>IncDecButtonVisible</strong></td>
<td>Specifies if the increment and decrement buttons are visible.</td>
</tr>
<tr>
<td><strong>IncDecValue</strong></td>
<td>Specifies the amount the value increments or decrements when the user clicks the increment or decrement button.</td>
</tr>
<tr>
<td><strong>Maximum</strong></td>
<td>Specifies the maximum value of the CWNumEdit control for range checking.</td>
</tr>
<tr>
<td><strong>Minimum</strong></td>
<td>Specifies the minimum value of the CWNumEdit control for range checking.</td>
</tr>
<tr>
<td><strong>Mode</strong></td>
<td>Specifies how the CWNumEdit control responds to user input.</td>
</tr>
<tr>
<td><strong>RangeChecking</strong></td>
<td>Specifies if the CWNumEdit control enforces the Minimum and Maximum property values.</td>
</tr>
<tr>
<td><strong>ReadyState</strong></td>
<td>Returns the ready state.</td>
</tr>
<tr>
<td><strong>Text</strong></td>
<td>Specifies the text that is displayed in the edit box.</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Specifies the current value of the numeric edit control.</td>
</tr>
</tbody>
</table>
**Methods**

- **AboutBox**: Displays the About Box for the control.
- **ControlImage**: Returns an image of the entire control.
- **ExportStyle**: Exports the style of the Measurement Studio control to a file.
- **ImportStyle**: Imports a previously exported style.
- **SetMinMax**: Sets the Minimum and Maximum properties.
## Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click</td>
<td>Generates when you click the mouse on the control.</td>
</tr>
<tr>
<td>CWBindingDataUpdated</td>
<td>Generated when the binding data is updated.</td>
</tr>
<tr>
<td>CWBindingStatusUpdated</td>
<td>Generated when the status of the binding connection changes.</td>
</tr>
<tr>
<td>DblClick</td>
<td>Generates when you double-click the mouse on the control.</td>
</tr>
<tr>
<td>Error</td>
<td>Generated when the user enters an illegal value in the CWNNumEdit control.</td>
</tr>
<tr>
<td>IncDecButtonClicked</td>
<td>Generated when the user clicks the increment or decrement button on the numeric edit control.</td>
</tr>
<tr>
<td>KeyDown, KeyUp</td>
<td>KeyUp generates when you release a key while the control has the input focus.</td>
</tr>
<tr>
<td></td>
<td>KeyDown generates when you press a key while the control has the input focus.</td>
</tr>
<tr>
<td>KeyPress</td>
<td>Generated when the control has focus and you press a key.</td>
</tr>
<tr>
<td>MouseDown, MouseMove, MouseUp</td>
<td>MouseDown generates when you click the mouse on the control.</td>
</tr>
<tr>
<td></td>
<td>MouseMove generates when you move the mouse over the control.</td>
</tr>
<tr>
<td></td>
<td>MouseUp generates when you release the mouse on the control.</td>
</tr>
<tr>
<td>ReadyStateChange</td>
<td>Generated when the ready state changes.</td>
</tr>
<tr>
<td>ValueChanged</td>
<td>Generated when the value of the CWNNumEdit control has changed from the user interface or the program.</td>
</tr>
<tr>
<td>ValueChanging</td>
<td>Generated when the value of the CWNNumEdit control is about to change (the value of the control has been entered but not set).</td>
</tr>
</tbody>
</table>
**CWPictureDisp**

The CWPictureDisp object manipulates images within controls. This object is very similar to the Picture object within Visual Basic and other containers.
Properties

**CWImage**  Returns the CWImage object that allows advanced image operations.

**Handle**  Returns a handle to the graphic contained within a Picture object.

**Height**  Specifies the height of the picture in HiMetric units.

**hPal**  Specifies a handle to the palette of the picture.

**Type**  Returns the graphic format of a picture.

**Width**  Specifies the width of the picture in HiMetric units.
See Also

CWImage
**CWPlot**

The CWPlot object defines the plot for the CWGraph control. Its properties determine the appearance of the data. New plots use settings from the CWGraph.PlotTemplate object.

Use the plot methods on a CWPlot object to display one set of data. To pass more than one set of data at a time, use the plot methods on the CWGraph.
## Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoScale</td>
<td>Specifies if the extent of the data in the plot affect the extent of an autoscaling axis.</td>
</tr>
<tr>
<td>BasePlot</td>
<td>Specifies a plot to use for Y values when either the FillToBase or LineToBase property is enabled.</td>
</tr>
<tr>
<td>BaseValue</td>
<td>Specifies the Y value used to fill the space between the base plot and the x axis when either the FillToBase or LineToBase property is enabled.</td>
</tr>
<tr>
<td>DefaultPlotPerRow</td>
<td>Specifies the default value used by the PlotXY and ChartXY methods if the optional bXInFirstRow parameter is omitted.</td>
</tr>
<tr>
<td>DefaultxFirst</td>
<td>Specifies the default value used by the PlotY method if the optional xFirst parameter is omitted.</td>
</tr>
<tr>
<td>DefaultxInc</td>
<td>Specifies the default value used by the PlotY and ChartY methods if the optional xInc parameter is omitted.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies if the plot generates mouse events when CWGraph.TrackMode = cwGTrackAllEvents and the plot is visible.</td>
</tr>
<tr>
<td>FillColor</td>
<td>Specifies the color to use for filling to the specified BaseValue or BasePlot values.</td>
</tr>
<tr>
<td>FillToBase</td>
<td>Fills the area between the plot and the values set for CWPlot.BaseValue or CWPlot.BasePlot.</td>
</tr>
<tr>
<td>LineColor</td>
<td>Specifies the color of line that connects points in the plot.</td>
</tr>
<tr>
<td>LineStyle</td>
<td>Specifies the style of lines for connecting points on a plot.</td>
</tr>
<tr>
<td>LineToBase</td>
<td>Specifies if lines connect the data points to the values specified by CWPlot.BaseValue or CWPlot.BasePlot.</td>
</tr>
<tr>
<td>LineToBaseColor</td>
<td>Specifies the color of lines connecting data points to the values specified by CWPlot.BaseValue or CWPlot.BasePlot.</td>
</tr>
<tr>
<td>LineWidth</td>
<td>Specifies the width of the plotting line. The width range is 1 to 5.</td>
</tr>
<tr>
<td>MultiPlot</td>
<td>Determines if the CWGraph plot or chart methods can use this plot.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies the name of the plot.</td>
</tr>
<tr>
<td>PointColor</td>
<td>Specifies the color of the points on a plot.</td>
</tr>
<tr>
<td>PointStyle</td>
<td>Specifies the image drawn at each point on a plot.</td>
</tr>
<tr>
<td>TempPlot</td>
<td>Specifies if a plot can be discarded if the next CWGraph plot or chart method does not need it.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies if the plot is visible or hidden.</td>
</tr>
<tr>
<td>XAxis</td>
<td>Specifies the x axis for a plot. This is a read-only property.</td>
</tr>
<tr>
<td>XYData</td>
<td>Displays data from an external source in a plot on a graph as it would if you called the PlotXY method.</td>
</tr>
<tr>
<td>XYDataAppend</td>
<td>Displays data from an external source in a chart as it would if you called the ChartXY method.</td>
</tr>
<tr>
<td>YAxis</td>
<td>Specifies the y axis for a plot.</td>
</tr>
<tr>
<td>YData</td>
<td>Displays data from an external source in a plot on a graph as it would if you called the PlotY method.</td>
</tr>
<tr>
<td>YDataAppend</td>
<td>Displays data from an external source in a chart as it would if you called the ChartY method.</td>
</tr>
</tbody>
</table>
Methods

.ChartXvsY. Charts an array of Y data against an array of X data. This method is similar to the PlotXvsY method, except that the system does not delete the previously plotted data until the amount stored is greater than the CWGraph.ChartLength property specifies.

.ChartXY. Charts a two-dimensional array of data as an XY chart. This method is similar to the PlotXY method, except that the system does not delete the previously plotted data until the amount stored is greater than the CWGraph.ChartLength property specifies.

.ChartY. Charts a 1D array of data. This method is similar to the PlotY method, except that the system does not delete the previously plotted data until the amount stored is greater than the CWGraph.ChartLength property specifies.

.ClearData. Clears the data currently displayed in the plot.

.PlotXvsY. Plots an array of Y data against an array of X data.

.PlotXY. Plots a two-dimensional array of data as an XY plot. The first row is the X data and the second row is the Y data.

.PlotY. Plots a one-dimensional array of data.
See Also

CWPlots
CWPlots

CWPlots is a collection of CWPlot objects.
## Properties

| Count | Returns the number of objects in the collection. |
Methods

*Add*  
Adds an object to the collection and returns the new object.

*Item*  
Returns the specified object from the collection.

*Remove*  
Removes the specified item from the collection.

*RemoveAll*  
Removes all objects from the collection.
See Also

CWPlot
CWPointer

Represents a single value displayed on a knob or slide control. You can add multiple pointers to knobs and slides.
## Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>Specifies the color of the pointer.</td>
</tr>
<tr>
<td>FillColor</td>
<td>Specifies the fill color of the pointer.</td>
</tr>
<tr>
<td>FillStyle</td>
<td>Specifies how the pointer is filled.</td>
</tr>
<tr>
<td>Index</td>
<td>Specifies the index of the pointer in the CWPointers collection.</td>
</tr>
<tr>
<td>Mode</td>
<td>Specifies how the pointer behaves and responds to user input.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies the name of the pointer.</td>
</tr>
<tr>
<td>Style</td>
<td>Specifies the graphical style of the pointer.</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the value of the pointer.</td>
</tr>
<tr>
<td>ValuePairIndex</td>
<td>Specifies the index of the value pair selected by this pointer.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies if the pointer is visible or hidden.</td>
</tr>
</tbody>
</table>
See Also

CWPointers
**CWPointers**

CWPointers is a collection of CWPointer objects.
<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
</tr>
<tr>
<td>Returns the number of objects in the collection.</td>
</tr>
</tbody>
</table>
Methods

**Add**
Add an object to the collection and returns the new object.

**Item**
Returns the specified object from the collection.

**Remove**
Removes the specified item from the collection.

**RemoveAll**
Removes all objects from the collection.
See Also

CWPointer
Pointers
CWSHape

The CWSHape object defines the shape part of the CWAnnotation object. CWSHape properties define the type, position, and appearance of the shape. You can use shapes to mark regions of the plot area or graph. You also can use the CWSHape object to add images or any type of shape anywhere on the graph.
**Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>Specifies the color of the shape.</td>
</tr>
<tr>
<td>FillVisible</td>
<td>Specifies if the shape is filled with the value specified by the CWShape.Color property.</td>
</tr>
<tr>
<td>Image</td>
<td>Specifies the image of the shape.</td>
</tr>
<tr>
<td>LineColor</td>
<td>Specifies the line color of the shape.</td>
</tr>
<tr>
<td>LineStyle</td>
<td>Specifies the line style of the shape.</td>
</tr>
<tr>
<td>LineWidth</td>
<td>Specifies the line width of the shape.</td>
</tr>
<tr>
<td>PointStyle</td>
<td>Specifies the point style of the shape.</td>
</tr>
<tr>
<td>RegionArea</td>
<td>Specifies the region area of the shape.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of shape.</td>
</tr>
<tr>
<td>XCoordinates</td>
<td>Specifies the X and Y coordinates of the shape.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetCoordinates</td>
<td>Sets the X and Y coordinates of the shape.</td>
</tr>
</tbody>
</table>
See Also

CWAnotation
**CWSlide**

CWSlide is the top-level object for the Slide control.
### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ActivePointer</strong></td>
<td>Specifies the CWPointer object of the active pointer.</td>
</tr>
<tr>
<td><strong>Axis</strong></td>
<td>Returns the CWAxis object for this control.</td>
</tr>
<tr>
<td><strong>BackColor</strong></td>
<td>Specifies the background color of the control.</td>
</tr>
<tr>
<td><strong>BackgroundImage</strong></td>
<td>Specifies an image to be used for the background of the control.</td>
</tr>
<tr>
<td><strong>Caption</strong></td>
<td>Specifies the text that appears in the control.</td>
</tr>
<tr>
<td><strong>CaptionColor</strong></td>
<td>Specifies the color of the caption.</td>
</tr>
<tr>
<td><strong>CWBindings</strong></td>
<td>Returns a collection of CWBinding objects.</td>
</tr>
<tr>
<td><strong>Enabled</strong></td>
<td>Specifies if the control responds to user input.</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Specifies the font for the caption and axis labels.</td>
</tr>
<tr>
<td><strong>ForeColor</strong></td>
<td>Specifies the foreground color in the CWSlide or CWKnob control.</td>
</tr>
<tr>
<td><strong>ImmediateUpdates</strong></td>
<td>Specifies if the control draws new data as soon as it is available or if the form refreshes the control when it draws other controls.</td>
</tr>
<tr>
<td><strong>IncDecValue</strong></td>
<td>Specifies the amount that the control is changed when the user clicks the increment or decrement buttons, or uses the keyboard to increment or decrement the control.</td>
</tr>
<tr>
<td><strong>InteriorColor</strong></td>
<td>Specifies the color used to paint the interior of several types of slide control styles.</td>
</tr>
<tr>
<td><strong>KeyboardMode</strong></td>
<td>Specifies how the control handles keyboard input from the user.</td>
</tr>
<tr>
<td><strong>Pointers</strong></td>
<td>Returns a collection of Pointer objects.</td>
</tr>
<tr>
<td><strong>ReadyState</strong></td>
<td>Returns the ready state.</td>
</tr>
<tr>
<td><strong>ScaleStyleValuePairsOnly</strong></td>
<td>Specifies if only the value pairs are displayed on the axis.</td>
</tr>
<tr>
<td><strong>ShowFocusMode</strong></td>
<td>Specifies how the control indicates that it has the focus.</td>
</tr>
<tr>
<td><strong>Statistics</strong></td>
<td>Returns the Statistics object for this control.</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Specifies the value of the active pointer.</td>
</tr>
<tr>
<td><strong>ValuePairIndex</strong></td>
<td>Specifies the index of the value pair selected by the active pointer.</td>
</tr>
<tr>
<td><strong>Windowless</strong></td>
<td>Specifies if the control has a window.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AboutBox</td>
<td>Displays the About Box for the control.</td>
</tr>
<tr>
<td>ControlImage</td>
<td>Returns an image of the entire control.</td>
</tr>
<tr>
<td>ExportStyle</td>
<td>Exports the style of the Measurement Studio control to a file.</td>
</tr>
<tr>
<td>Images</td>
<td>Provides access to the CWImage objects in the CWKnob or CWSlide control.</td>
</tr>
<tr>
<td>ImportStyle</td>
<td>Imports a previously exported style.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Redraws the CWSlide control.</td>
</tr>
<tr>
<td>SetBuiltinStyle</td>
<td>Sets many properties of the control to represent the new style specified.</td>
</tr>
</tbody>
</table>
# Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Click</strong></td>
<td>Generates when you click the mouse on the control.</td>
</tr>
<tr>
<td><strong>CWBindingDataUpdated</strong></td>
<td>Generated when the binding data is updated.</td>
</tr>
<tr>
<td><strong>CWBindingStatusUpdated</strong></td>
<td>Generated when the status of the binding connection changes.</td>
</tr>
<tr>
<td><strong>DblClick</strong></td>
<td>Generates when you double-click the mouse on the control.</td>
</tr>
<tr>
<td><strong>KeyDown, KeyUp</strong></td>
<td>KeyUp generates when you release a key while the control has the input focus.</td>
</tr>
<tr>
<td><strong>KeyPress</strong></td>
<td>KeyDown generates when you press a key while the control has the input focus.</td>
</tr>
<tr>
<td><strong>KeyDown</strong></td>
<td>Generated when the control has focus and you press a key.</td>
</tr>
<tr>
<td><strong>MouseMove, MouseUp</strong></td>
<td>MouseDown generates when you click the mouse on the control.</td>
</tr>
<tr>
<td><strong>MouseDown</strong></td>
<td>MouseMove generates when you move the mouse over the control.</td>
</tr>
<tr>
<td><strong>MouseUp</strong></td>
<td>MouseUp generates when you release the mouse on the control.</td>
</tr>
<tr>
<td><strong>PointerValueChanged</strong></td>
<td>Generated as the value of a pointer changes from the user interface or the program.</td>
</tr>
<tr>
<td><strong>PointerValueCommitted</strong></td>
<td>Generated when the value of a pointer has stopped changing.</td>
</tr>
<tr>
<td><strong>ReadyStateChange</strong></td>
<td>Generated when the ready state changes.</td>
</tr>
</tbody>
</table>
**CWStatistics**

CWStatistics provides access to the statistics kept by the CWKnob and CWSlide controls. The three calculated statistics--minimum, maximum, and mean--are updated each time a pointer value is changed programmatically or graphically.

You can reset the minimum, maximum, and mean values with the Reset method. After resetting the values, minimum and maximum are calculated using only values collected since the last reset.

To continuously display any statistic for a specific pointer, set the mode on the CWPointer object either in the Pointer property page or programmatically.
## Properties

<table>
<thead>
<tr>
<th>Maximum</th>
<th>Returns the maximum value of any pointer since the statistics were last reset.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>Returns the mean of the last 10 pointer values.</td>
</tr>
<tr>
<td>Minimum</td>
<td>Returns the minimum value of any pointer since the statistics were last reset.</td>
</tr>
</tbody>
</table>
**Methods**

- **Reset** Resets the minimum, maximum, and mean statistics. After resetting the values, the system recalculates the minimum and maximum using only values collected since the last reset.
See Also

Statistics
CWPInter.Mode
CWTicks

The CWTicks object controls placement of tick marks on an axis. You configure tick marks by specifying how many divisions you want on the axis independent of the range of the axis or by specifying an interval for placing the ticks.

To make a grid similar to those found on an oscilloscope, set the MajorDivisions property on the X and Y Axes.Tick object to 10. Use the MajorDivisionUnits property to create tick marks and grid lines that scroll with the data on a strip chart. If both MajorDivisions and MajorDivisionUnits are set to nonzero values, the MajorDivisions property takes precedence.

The same rules apply to the minor division properties.
Properties

**Above**
Specifies if tick marks are drawn above the object.

**AutoDivisions**
Specifies if divisions are automatically calculated.

**Below**
Specifies if tick marks are drawn below the object.

**Inside**
Specifies if tick marks are drawn on the inside of the axis.

**Left**
Specifies if tick marks are drawn to the left of the object.

**MajorDivisions**
Specifies the number of major divisions.

**MajorGrid**
Specifies if major grid lines appear.

**MajorGridColor**
Specifies the color of major grid lines.

**MajorTickColor**
Specifies the color of major ticks.

**MajorTicks**
Specifies if major ticks appear.

**MajorUnisBase**
Specifies the base number for calculating ticks.

**MajorUnisInterval**
Specifies the number of units between major divisions.

**MinorDivisions**
Specifies the number of minor divisions for each major division.

**MinorGrid**
Specifies if minor grid lines appear.

**MinorGridColor**
Specifies the color of minor grid lines.

**MinorTickColor**
Specifies the color of minor ticks.

**MinorTicks**
Specifies if minor ticks appear.

**MinorUnisInterval**
Specifies the number of units between minor divisions.

**Outside**
Specifies if tick marks are drawn on the outside of the axis.

**Right**
Specifies if tick marks are drawn to the right of the object.
CWValuePair

The CWValuePair object configures an individual value pair, which consists of a name and a value. Use value pairs on the axis of a knob, slide, or graph control to associate a symbolic name with a value on the axis. For example, use value pairs as custom ticks, labels, or grid lines. You also can use value pairs on the slide and knob control to implement a ValuePairs only control, which limits the valid values of the control to the control's value pairs.

The value of a value pair can be the position of the value pair on the axis or the index of the value pair in the collection.
<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specifies the name of a value pair.</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the value of a value pair.</td>
</tr>
</tbody>
</table>
See Also

CWValuePairs
**CWValuePairs**

The CWValuePairs object is a collection of CWValuePair objects for an axis. A value pair marks specific points on an axis with a custom label.
Properties

- **Count**: Returns the number of objects in the collection.
- **GridLines**: Specifies if grid lines appear at value pair locations.
- **LabelType**: Specifies the type of labels to draw for the value pairs.
- **Location**: Specifies if CWValuePairs are placed on the axis by their value or by their index.
- **MajorTicks**: Specifies if major ticks are placed at the location of value pairs.
Methods

- **Add**: Adds an object to the collection and returns the new object.
- **Item**: Returns the specified object from the collection.
- **Remove**: Removes the specified item from the collection.
- **RemoveAll**: Removes all objects from the collection.
- **Swap**: Swaps two CWValuePair elements, altering their indices.
See Also

CWValuePair
Example: 2D Graph Events

Demonstrates interaction with data displayed on the Measurement Studio 2D Graph.
Description

This example displays the many ways you can interactively and programmatically interact with data displayed on a Measurement Studio 2D Graph in Visual Basic. Use the TrackMode property to pan, zoom, or generate cursor or plot events in the graph. See the description below the TrackMode listbox for information about the selected TrackMode.

You can set the TrackMode option to one of the following values either in the graph property pages or programmatically:

Plot Area Events
Plot & Cursor Events
Cursors (CursorChange event)

PanXY
PanX
PanY
ZoomXY
ZoomX
ZoomY

Plot Area Events

Select the Plot Area Events TrackMode to generate Mouse events anywhere on the graph. The Mouse events return the x position and y position of the mouse on the graph. As you move the mouse, notice how the values in Plot Area Events change accordingly. The Plot Area Mouse events also indicate when the left or right mouse button is clicked and released anywhere on the graph.

Plot and Cursor Events

Select the Plot & Cursor Events TrackMode to generate both Mouse events along plot and Cursor events anywhere on the graph. The PlotMouse events return the x position and y position of the cursor location and the point index, or the exact point on the plot. The PlotMouse events also indicate when the left or right mouse button is pressed or released on the plot. Unlike the PlotMouse events, the Cursor events are generated anywhere on the plot area. The Cursor events return the x position and y position of the crosshair and specify if the left...
or right mouse button is pressed or released in the graph area.

**Cursor Change**

Select Cursors to interactively drag a cursor anywhere on the graph and generate the CursorChange event. CursorChange returns the x position and y position of the crosshair.

**Note:** If you want to generate both Cursor and Plot events, you have to use the Plot & Cursor TrackMode and write event procedures to move the cursor programmatically when a CursorMouse event occurs.

**Pan**

Use the Pan modes to move the plot horizontally and/or vertically. Panning is useful when the entire plot is not visible in the plot area. Select PanXY to drag the plot both horizontally and vertically. Click and drag on the graph to pan the plot. Use PanX to limit interaction to horizontal panning. Select PanY to limit interaction to vertical panning. To reset the plot area to its original view, select the Reset After Zooming or Panning button.

**Zoom**

Use the Zoom modes to reset the graph view to a specific region of the plot, which is useful when you want to focus on a particular region of a graph. ZoomXY selects a region horizontally and vertically. Click and drag to the region you want to zoom in on. ZoomX selects an area of the plot horizontally only. ZoomY selects an area vertically only. Select the Reset After Zooming or Panning Button to reset the view.
Controls, Properties, Methods, and Events

This example demonstrates the following controls, properties, methods, and events:

- CWGraph
- TrackMode
- CWCursor
- SnapMode
Example Location
Samples\UI\Graph\Events
Example: Advanced Button
Demonstrates advanced button and Boolean concepts.
Description

This example demonstrates the advanced properties of the CWButton. The buttons shown in this example are custom bitmap buttons in the form of pipes, pumps, and valves. When the buttons are true, they are green, and when they are false, they are red. The tank continuously fills and empties during this example. You can interact with this example by turning both the valve and pump off and on.
Example Location
Samples\UI\Button\Advanced Button
Example: Advanced Knob
Demonstrates advanced settings on the CWKnob control.
Description

This example demonstrates the advanced properties of the CWKnob control. This user interface displays the knob as a knob, dial, and meter. You can interact with a few of the controls in this example. The knobs on the top row are non-interactive. On the bottom row, you can change the value of the Power and watch the change affect the speed. You can also change the angle of the rudder and see a change in the Ascent. Changes in the Frequency change the value of Signal.
Example Location

Samples\UI\Knob\Advanced Knob
Example: Advanced Slide

Demonstrates some advanced uses of the CWSlide control.
Description

This example demonstrates the advanced properties of the CWSlide control. The horizontal slide at the top of the user interface shows multiple pointers with the different fill options. As you move each pointer around, you can see how the options are different. The ramp and tank use the Advanced property page to replace the frame of the control with custom bitmaps. The bitmaps use a transparent color are moved forward in the Z-direction to hide other components that are part of the control. Move the pointers on the horizontal slide at the top of the interface to change the values of the ramp and tank. You can also change the values using your mouse. The vertical slide on the left side of the interface demonstrates a moving updated value. It is a value pair that is updated in the ValueChanged event. You can turn off the general labels without disabling the value pair label by setting the Tick Spacing to By Units and setting Major to 0. The thermometer on the right side of the interface demonstrates a range of fill colors. The thermometer uses one fill pointer per color which is updated in the ValueChanged event of the pointer.
Example Location
Samples\UI\Slide\Advanced Slide
Example: Annotation Bookmarking

Demonstrates use of several Annotation properties on the Measurement Studio CWGraph ActiveX control.
Description

This example demonstrates how to bookmark points on a plot with annotations, and then to scroll the graph to view those bookmarked points. In this example, an arrow and text style annotation marks several points along the plot. Click on one of the buttons to center the corresponding annotation.
Controls, Properties, Methods, and Events

This example demonstrates the following controls, properties, methods, and events:

- **CWAnnotation**
- **Plot**, **PointIndex**, **SnapMode**, **Caption**, **Name**, **Shape**
- **CWCaption**
- **SetCoordinates**
- **CWShape**
- **XCoordinates**
- **CWGraph**
- **PlotY**, **Axes**, **TrackMode**, **Annotations**
- **CWAxis**
- **Minimum**, **Maximum**
Example Location

Samples\UI\Graph\Graph Styles
See Also

Simple Annotations Example
Example: Axes Advanced

Programmatically controls axes on a CWGraph.
Description

This example demonstrates how the CWGraph can add and manipulate axes during runtime. You can interact with this example by adding and removing plots and axes from the graph. You can then assign plots to an axis using the Assign Plot to Axis area. You can also edit the axes using the Edit Axis area by changing the ticks, labels, and minimum and maximum.
Example Location
Samples\UI\Graph\Axes Advanced
Example: Axis
Programmatically controls the axes of a CWGraph.
Description

This example demonstrates how the CWGraph can manipulate axes during runtime. You can interact with this example by plotting either random points or a sine wave. You can then set the minimum and maximum for either axes or choose to autoscale.
Example Location

Samples\UI\Graph\Axes
Example: Clock
Uses the CWKnob control to implement a clock.
Description

This example uses a CWKnob to create an analog clock. A CWNumEdit control displays the time including the seconds.
Example Location
Samples\UI\Knob\Clock
Example: Color Substitution

Demonstrates color substitution in the CWImage object.
Description

You can use color substitution on metafiles to customize your control images while still being able to modify colors on the image.
Controls, Properties, Methods, and Events

This example demonstrates the following controls, properties, methods, and events:

- CWImage
- Substitute, SubstituteColor
Example Location

Samples\UI\Common\Color Substitution
**Example: Cursors**

Uses cursors to interact with data plotted on a CWGraph.
Description

This example demonstrates the different ways you can use cursors to interact with the graph.

To begin this example, select whether you want random data or a sine wave. Press the Generate Data button to see 100 points displayed on the graph. Using the Graph Operations on the user interface, you can zoom, pan, or have two different types of cursors. When in the zoom mode, you can use your mouse to zoom into a selected area. Use the Zoom Out button to see the original plot after zooming in. The Autoscale button will autoscale the axes. Using the pan mode, you can set the extents of the X and Y axes by dragging the plot area with the mouse. If you choose the Cursor Coordinates option, a crosshair appears on the graph. If the Cursors toggle is set to Float Free, then you can move the crosshair anywhere on the graph. The Snap to Plot mode only allows the crosshair to move along the curve. The value of the X and Y coordinates of the intersection appears in the Cursor Information area. If you choose the Cursor Measurements, another crosshair appears on the graph. This crosshair moves in the same way as the previous one, but it measures the distance between the X and Y coordinates of the intersection of the two crosshairs. This information is displayed in the Cursor Measurements area.
Example Location
Samples\UI\Graph\Cursors
Example: CWBinding Features

Demonstrates a wide variety of CWBinding features.
Description

The user interface controls in this example use integrated DataSocket connectivity to connect a property from the control to a data item. Integrated DataSocket connectivity works through CWBinding objects. The actual connection between the external data item and the control property is called a binding.

CWGraph1 uses two CWBinding objects to connect two separate plots to two separate data items on the National Instruments OPC Demo server: a square wave generator and a sine wave generator. Both bindings are connected in Read mode with an update interval of 100 ms. The initial data is read and charted as soon as the connection is made and then updated every 100 ms. This type of binding is useful if you want to chart y data with respect to actual time, rather than charting new data when ever it is available (the x-axis values becomes arbitrary). Check the Bindings property page to see how to specify interval updates. If you want to set intervals programmatically, use the CWBinding.TimerInterval property.

CWSlide1 creates a binding in the property page that configures a CWBinding object to write data to an item on the National Instruments OPC Demo server. CWGraph2 programmatically creates a CWBinding and connection to the same item in read mode so that the data from the slide is visualized in a plot on the graph. Try moving the value of the slide and see how the plot on the graph responds.

The binding on CWKnob1, which also is configured in the Bindings property page, connects to a sine generator on the National Instruments OPC Demo server and automatically reads new data. As each new data point is received, CWKnob1 generates the CWBindingDataUpdated event. This event procedure checks the range of each data point. If a data point exceeds 9, then the knob turns red. If the data point falls below 1, then the knob turns yellow. You can use the CWBindingDataUpdated event to manipulate, process, or save data from the data source. CWKnob1 uses the CWBindingStatusUpdated event to check the status of the binding connection and display it on the user interface.
Controls, Properties, Methods, and Events

This example demonstrates the following controls, properties, methods, and events:

**CWBinding**

**AccessMode**, **URL**, **BindProperty**, **SetBindObject**, **ConnectTo**, **Disconnect**, **Connect**

**CWGraph**
Example Location
Samples\UI\Common\CWBinding
See Also

Simple CWBinding Example
CWBinding Reader Example
Example: CWBinding Reader

Demonstrates how to display data from different external data sources in a graph using integrated DataSocket connectivity configured in the property pages and one line of code.
Description

The Graph contains one CWBinding object that connects to the source selected on the user interface in ReadAutoUpdate mode, so the graph displays new data when ever the source has new data available. The external data source is bound to the CWGraph.YData property. Setting the YData property is the same as calling the PlotY method to plot new data points. You can plot xy data by binding the XYData property or chart data by binding the YDataAppend or XYDataAppend property.

This example can connect to the following data sources:

A simple wave file on the National Instruments Web server:
http://www.natinst.com/cworks/datasocket/chirp.wav

A waveform in DSD format on the National Instruments FTP server:

A more complex waveform on the National Instruments Web server:
http://www.natinst.com/cworks/datasocket/schirp.dsd

Data from a tabbed text file on the FTP server:

Data from the local DataSocket server. To use this data source, you must start a DataSocket server on your computer and run the DSWriter example from the Samples/DataSocket directory.
dstp://localhost/wave

This example also displays the connection status using the CWBindingStatusUpdated event.
Controls, Properties, Methods, and Events

This example demonstrates the following controls, properties, methods, and events:

- CWBinding
- ConnectTo, Disconnect
- CWGraph
Example Location
Samples\UI\Common\CWBinding
See Also

Simple CWBinding Example
CWBinding Features Example
Example: Graph Styles

Demonstrates the Plot and Chart methods on the Measurement Studio CWGraph ActiveX control.
**Description**

The Graph is the most complex of the user interface objects. You can use it in two basic modes: Plot or Chart. You can select the mode by using different methods in your program to pass data to the graph.

This example uses the PlotY and ChartY methods, which are the two most common methods for passing data to the graph. However, there are two more Plot methods (PlotXY and PlotXvsY) and two more Chart methods (ChartXY and ChartXvsY) that affect how data is displayed on the graph.

The button control has several different display styles but maintains a single set of property sheets and properties, methods, and events. You can use the Button control as an input or output. When using it as an input, create a push button or a switch to initiate an action or switch between actions. For an output, create an LED to indicate a Boolean condition.
Example Location
Samples\UI\Graph\Graph Styles
Example: Import/Export Styles

Restores previously configured CWGraph controls using ImportStyle and ExportStyle.
Description

Run this example and change the CWGraph Style slide to import previously exported styles for the graph. This example demonstrates programmatic loading of styles for a control. The styles were created using the CWGraph property page.
Example Location

Samples\UI\Common\Import Export Styles
See Also

ExportStyle
ImportStyle
Example: Knob Format
Demonstrates CWKnob styles.
Description

This example demonstrates how highly configurable the CWKnob is. Five different knobs are shown on the front panel with four different looks.

The controls in the top half of the panel entirely correspond to the dial in the upper half. The axis control sets the maximum and minimum to the values in the CWNumEdit controls. You can add and remove pointers on the dial with the labeled buttons. The Value Changed area also displays information about the dial. The Index shows the number of the pointer that you have selected, and Value shows the number that the pointer is pointing to. The bottom half of the interface shows four more knobs. The dial on the right simply shows that you can make these knobs look like dials. You can move the pointer around the dial, but it does not appear to change anything else. The quarter-circle knob also does not alter anything, but it shows that a knob in that form is available. The two meters on the left show how the knob can also be a meter. The top meter does not alter anything, though you can interact with it by moving the pointer. The bottom meter moves continuously, and the pointer color even changes to red when it enters the yellow zones.
Example Location
Samples\UI\Knob\Knob Format
Example: Lissajous
Displays Lissajous figures using the Measurement Studio Graph and Slide ActiveX controls.
Description

French scientist Jules Lissajous is known for his work with vibrating bars, in which he tried to develop an optical method for studying vibrations. At first he studied waves produced by a tuning fork in contact with water. Later, he described a way of studying acoustic vibrations by reflecting a light beam from a mirror attached to a vibrating object onto a screen. The figures obtained by successively reflecting light from mirrors on two tuning forks vibrating at right angles are known as Lissajous figures or curves. The curves are seen only because of persistence of vision in the human eye. This example predefines five different Lissajous figures.

Try experimenting with the predefined figures and creating your own figures and curves by defining different functions, frequencies, and amplitudes.
Example Location
Samples\Apps\Lissajous
**Example: LorenzAttractors**

Demonstrates Lorenz attractors.
**Description**

This example shows the Lorenz Attractors in the different planes as well as in a three-dimensional graph. Plotted on a three-dimensional plane, a shape unlike any other forms. Instead of a simple geometric structure or even a complex curve, the structure now known as the Lorenz Attractor weaves in and out of itself. Projected on the X-Z plane, the attractor looks like a butterfly; on the Y-Z plane, it resembles an owl mask. The X-Y projection is useful mainly for glimpsing the three-dimensionality of the attractor; it looks something like two paper plates, on parallel but different planes, connected by a strand of string.

As the Lorenz Attractor is plotted, a strand will be drawn from one point, and will start weaving the outline of the right butterfly wing. Then it swirls over to the left wing and draws its center. The attractor will continue weaving back and forth between the two wings, its motion seemingly random, its very action mirroring the chaos which drives the process. The term "sensitive dependence on initial conditions" was coined to describe the phenomenon that small changes in a recursive system can drastically change the results of running that system.

To begin the example, press the Plot button. The X, Y, and Z values of the last point appear in the respective box. To see the attractor charted on the graphs, press the Chart button. You can see the current coordinates by pressing the Update XYZ button. Press the Stop Charting button to stop the charting. To see a detailed description of Lorenz Attractors, press the About Lorenz Attractors button.
Controls, Properties, Methods, and Events

This example demonstrates the following controls, properties, methods, and events:

- **CWGraph**
- **ClearData**, **PlotXvsY**, **ChartXvsY**, **Axes**, **Plots**
- **CWAxis**
- **SetMinMax**
- **CWPlot**
- **PlotY**, **PlotXvsY**, **ChartY**
Example Location

Samples\Apps\LorenzAttractors
Example: Range Checking

Demonstrates using the range checking features and events of the CWNumEdit control.
Description

This example shows the CWNumEdit and its ability to check for range values. The numeric edit control in the upper left corner is the input value that is compared to the minimum and maximum at the bottom left corner. As the value of the input numeric control is changing, the ValueChanging Event LED lights up and the boxes below it display the new, previous, and attempted values. If the value is out of range, the Out of Range LED lights up. Once the value has changed, the ValueChanged Event LED lights up and the boxes below it display the values. If the value is out of range, the Out of Range LED lights up.

To start this example, enter a number into the numeric edit control in the top left corner. The program will then go through the two events and update the information. If you click the Range Checking check box, the program will automatically put the value to 10 or 0 if you are greater than or less than the range.
Example Location
Samples\UI\NumericEdit\Range Checking
Example: Scaled Plots
Demonstrates the scaling parameters of the PlotY method on the CWGraph object.
Description

The PlotY method on the graph and plot objects optionally allows you to specify the initial value and the increment of the x data used to create the plot. You should use these parameters to plot your y data against the correct x data. For example, consider you are collecting 1000 temperatures at a rate of 10 kHz. To correctly plot the data, the x increment should be 1/10kHz, or .0001. As another example, consider a set of hourly temperatures collected every half hour beginning at 12 noon. In this case the initial x value should be 12 and the x increment should be .5. If you do not use these parameters, PlotY will assume an initial x value of 1 and an x increment of 1.
Example Location

Samples\UI\Graph\Scaled Plots
Example: Simple Annotations

Demonstrates basic uses of annotations on a Measurement Studio Graph (CWGraph) control.
Description

This example demonstrates how to programmatically create annotations to highlight data on a graph. In this example, a point-style annotation marks the minimum and maximum plot values of randomly generated data. To change the data, use the knob control to scale the data, then click the **Get Data** button to acquire and plot the data.

This example also demonstrates user interaction with annotations at run time. To reposition the annotation captions and lines, drag them with your mouse. Since the `CWAnnotation.SnapMode` property is set to `cwCSnapAnchoredToPoint`, you cannot move the annotation point. You can move the annotation captions and lines because the `CWGraph.TrackMode` property is set to `cwGTrackDragAnnotation`, and the `Enabled` property for both annotations is set to True.
Controls, Properties, Methods, and Events

This example demonstrates the following controls, properties, methods, and events:

- **CWGraph**
- **PlotY**
- **CWAAnnotation**
  - Enabled
  - Name
  - Plot
  - PointIndex
  - SnapMode
- **CWCaption**
  - SetCoordinates
  - Text
- **CWShape**
  - Type
- **CWKnob**
- **CWPointer**
  - Value
Example Location

Samples\UI\Graph\Annotations
Example: Simple Button
Demonstrates button styles.
Description

This example shows how highly configurable the CWButton is. Seventeen different buttons exist on the user interface. On the left side of the page, all of the buttons except for the floating donut and the lightening buttons change between their true and false modes randomly. Some of the buttons blink, and others are toggled. These buttons can be selected through the property pages. The remaining two buttons on the left side show how you can make customized bitmaps into a button. The toggles on the right side of the user interface control the LEDs underneath them. The Off and OK buttons control the LEDs below them. The Off button demonstrates a button that stays in the position that you place it in, while the OK button pops back up after you let go of the button.

You can interact with any of the buttons on the right side of the user interface. The buttons on the left side turn from true to false randomly, and the buttons with the customized bitmaps do not do anything. Press the Quit button to exit the example.
Example Location
Samples\UI\Button\Simple Button
Example: Simple CWBinding
Displays data from a data source specified by a URL on a Measurement Studio user interface control without any code.
Description

The graph in this example uses integrated DataSocket connectivity to connect a property from the control to a data item specified by a URL. Integrated DataSocket connectivity works through CWBinding objects. The actual connection between the external data item and the control property is called a binding.

In this example, the Graph.YDataAppend property is bound to the data source at opc://National Instruments.OPCDemo/sine, an item on the National Instruments OPC Demo server that was installed with Measurement Studio. Setting the YDataAppend property has the same effect as calling the ChartY method to chart new data.

Refer to the CWBinding Features and CWBinding Reader examples for more information about creating bindings and specifying URLs.
Example Location

Samples\UI\Common\CWBinding
See Also

- CWBinding Reader Example
- CWBinding Features Example
Example: Simple Graph

Demonstrates basic graph features, including plotting data, charting data, and setting axis scales and cursors.
Description

This example demonstrates the difference between plotting data (as new data is plotted, previous data on the graph is deleted) and charting data (new data is appended to existing data rather than overwriting existing data). The example lets users plot or chart a first data set, an additional data set, or multiple data sets (in this case 3) at the same time.

Experiment with the Plot and Chart buttons to see the difference between plotting and charting. Press the Clear Graph button to remove existing data from the graph.

You can customize the X-axis scale and the cursor from the user interface. To change either the minimum or maximum value on the X-axis scale, modify the appropriate value and press the Set Minimum or Set Maximum button. Press the Set Minimum and Maximum button to modify both values at the same time. To change the cursor (the blue crosshair on the graph), modify the X and Y values and press Set Cursor.
Controls, Properties, Methods, and Events

This example demonstrates the following controls, properties, methods, and events:

- **CWGraph**
- **PlotY**, **Axes**
- **CWAxis**
- **AutoScaleNow**, **SetMinMax**, **Minimum**, **Maximum**
Example Location

Samples\UI\Graph\Simple Graph
Example: Simple Knob
Demonstrates some simple uses of the CWKnob control.
Description

This example shows how highly configurable the CWKnob is. Eight different knobs are on the user interface. The left three dials on the top correspond to the leftmost dial on the bottom. As you change any one of the three top dials, the hands on the bottom dial change with the number indicated on the individual dials. Though you can change the position of the hands of the bottom dial, these values are not reflected in the upper dials. The fourth dial from the left corresponds with the meter directly below it, Top Meter. As you change the value on the dial, the value is displayed on the meter. The Right Meter behaves the same way as the Top Meter with the rightmost dial and meter.

You can interact with this example by changing the values of the dials on the top row. The values of the corresponding meters change with the values of the dials.
Example Location
Samples\UI\Knob\Simple Knob
Example: Simple Numeric Edit
Demonstrates many styles of the CWNumEdit control.
Description

This example demonstrates the versatility of the CWNNumEdit control. The five controls on the left and above the knob are related. They all display the same number with different notations and with the incremental buttons in different positions. The knob below this set shows how the CWNNumEdit control can display the value of a knob. The five controls in the upper middle of the user interface are related to each other and to the digital screen on the right. The top and bottom display the same value, though the bottom control turns red when the value is negative. The other boxes display the current from the digital screen, the value of the current using a monetary notation, and the value of the current in a frequency notation. The meter underneath the digital display shows how the control can display the value of a meter. Below the meter is a control displaying the time and date.
Example Location
Samples\UI\NumericEdit\Simple Numeric Edit
Example: Simple Slide
Demonstrates several types of CWSlide controls.
Description

This example shows how highly configurable the CWSlide is. Eight different slides are on the user interface. The three vertical slides above the three text boxes correspond to the leftmost vertical slide. As you move the pointers on the leftmost slide, the corresponding slide fills to that value, and the value is displayed in the text box. The two vertical slides on the right demonstrate the thermometer and tank types. They simply fill and empty during runtime. The two horizontal slides on the bottom correspond with each other. As you move the pointer, the pink slide above it will empty the same amount that you have moved the slide. The value of the pointer is also displayed in a text box.

To begin this example, you can change the pointers on the leftmost vertical slide and watch the corresponding slides react to the changes. You can also move the pointer of the horizontal slide.
Example Location
Samples\UI\Slide\Simple Slide
Example: Slide Format
Shows how to save slide formats.
Description

This example demonstrates the versatility of the CWSlide control. The two horizontal slides at the bottom of the user interface show the counting properties of the slide. The top slide counts how much time you have been in runtime. The bottom slide simply counts a percentage, and when it reaches the maximum, it resets to the minimum. The two vertical slides at the top left part of the user interface demonstrate the tank and thermometer. You can change their value with the mouse. The horizontal slide beneath them also interacts with them. When you change the value of that slide, the values in the tank and thermometer change. The buttons to the right of the rightmost vertical slide control it. You can add and remove pointers with the Pointer controls. The Value Changed controls show which pointer is currently selected and its value. The Axis Control area allows you to set the minimum and maximum of the slide.
Example Location
Samples\UI\Slide\Slide Format
Example: Sweep Chart

Demonstrates a special kind of data charting available on the CWGraph.
Description

This example demonstrates the sweep graph function of the CWGraph. A sweep chart replaces a previous set of data from the graph with a cursor to show where the new data begins. This sweeping shows a set number of points at any one time instead of all of the points on the same graph. Plots and charts are rather similar, though they are different. The difference between a plot and a chart is that a plot refers to taking a large number of points and updating one or more plots on the graph with new data. The old plot is replaced with a new plot. Charting data refers to appending new points to the end of an existing plot over time.

To begin this example, choose either Sine or Random data with the slide. Press the Sweep Chart button to see the chart plot data. Each time you press the Sweep Chart button, 250 data points are charted on the graph.
Example Location
Samples\UI\Graph\Sweep Chart
Example: UI Formats

Demonstrates different numeric formats available for the axes and values of different user interface controls.
Description

This example demonstrates how the format of different controls can be altered to include percent, symbolic engineering, scientific notation, percent, and currency. The most interactive part of this example is the section with the lightening bolt button. When the button is pressed, the value of the electric charge increases greatly, as shown by the Electric Charge meter and the graph. Over time, the charge decays. Next to the lightening generator area is a profit calculator. Depending on the value of the price slide, the profit and profit percentage change. Directly below the lightening generator is a frequency display. You can change the position of the knob and the box below it will display the current value of the knob. Below those controls are two numeric edit boxes with a format of date and time. The three slides on the bottom right demonstrate a thermometer, percentage slide, and scientific notation slide.

To interact with this example, you can do a variety of tasks. The lightening button can be pressed to simulate the electric charge and decay of a lightening strike. The Frequency dial can be turned, and the value will appear in the CWNumEdit box. The Price slide can be changed to see the profit and profit percentage. The other slides can be changed, but nothing occurs on the user interface as they are changed. The date and time can also be changed. To exit this example, press the Quit button.
Example Location

Samples\UI\Common\UI Formats
Example: UI Statistics

Uses the statistics mode of pointers on the CWSlide and CWKnob.
**Description**

With the statistics feature of the CWKnob and CWSlide controls, you can easily maintain the maximum, minimum, and mean of data passed to these controls. Select any of these statistics options in the Pointers property page of the control. To set a pointer to display one of the statistics, first set a different pointer as a control or indicator and then set the statistical pointer to one of the statistics modes.
Example Location
Samples\UI\Common\Statistics
Example: Visualizing Data

Demonstrates basic graph features and behavior, including how to work with the CWGraph control interactively in the Measurement Studio property pages and programmatically using Visual Basic code, charting real-time and historical data, plotting acquired signals, and customizing the CWGraph control to look exactly the way you want it to look.
Description

Choose one of the following options and press the Display Data button:

The Plot option uses the PlotY method on the CWGraph object to display a signal. Press the Display Data button several times to see how each call to the PlotY method plots a new signal.

The Chart option uses the ChartXY method on the CWGraph object to append a new signal to existing charts. In this case, the chart history is set to hold 3000 measurements, so pressing on the Display Data button several times appends new data to the chart and removes the oldest data to maintain a total measurement count of 3000 data points in the chart.

The Multiple plots option uses the PlotY method to create multiple plots at the same time by using a two-dimensional array of data. Each row of the array creates a plot in the graph.

The Compute average option creates multiple plots, then uses the Mean method on the CWStat1 control to compute the mean of the four plots and graph the result in a fifth plot. This option also sets the Multiplot property of the fifth plot to False so it is not replaced (or removed) when new plots are added to the graph. Click on the Multiple plots option again to see how the four original plots are replaced, but the fifth plot (average) remains untouched.

The Customize option demonstrates how you can use the properties of the CWGraph, CWPlot, and CWAxis objects to create a graph that looks exactly like you want. Here the graph has two y-axes, axes titles, and a graph title.

These options and the properties and methods that were used to create this example are fully explained in Application Note 150, Visualizing Data in Visual Basic.
Controls, Properties, Methods, and Events

This example demonstrates the following controls, properties, methods, and events:

**CWGraph**
- Axes, Plots, Caption, ChartLength, PlotY, ChartXY

**CWPlot**
- LineColor, MultiPlot, Name, PlotY, YAxis

**CWAxis**
- AutoScale, Caption, Name, Labels, Ticks, ValuePairs, AutoScaleNow, SetMinMax

**CWValuePair**
- Name, Value
Example Location

Samples\UI\Graph\_Getting Started
**Example: XYGraph**

Uses the PlotXY and PlotXvsY methods of the CWGraph.
Description

This example demonstrates the PlotXY and the PlotXvsY methods of the CWGraph. The difference between the two methods is that the PlotXY plots a 2D array of data. The first row is the x values while the subsequent rows are the y values for each plot. The PlotXvsY method plots a 1D or 2D array of y data against a 1D array of x data. Each row of y values generates a plot. This example also demonstrates how to chart points using the PlotY method. This method is used in the X Data and Y Data plots.

To begin this example, press any of the buttons. The PlotXY One Waveform and the PlotXY Multiple Waveforms buttons use the PlotXY method to plot a graph on the main graph and chart and x and y values on the XY Data graph. The PlotXvsY One Waveform and PlotXvsY Multiple Waveforms buttons use the PlotXvsY method to plot the graphs on the main graph. The x and y values appear on the X Data and Y Data charts respectively. You can also press the Plot Circle, Plot Octagon, Polar Plot, and Plot Spiral buttons to make those shapes. The x and y values will also appear on the X Data and Y Data graphs respectively.
Example Location
Samples\UI\Graph\XY Graph
Arrow Property (Read Only)

Syntax

`CWAnotation.Arrow`
Data Type

CWArrow
**Purpose**

Returns a CWAArrow object, which specifies the arrow part of the annotation.
See Also

CWArrow
Caption Property (Read Only)

Syntax

`CWAAnnotation.Caption`
Data Type

CWCaption
Purpose

Returns a CWCaption object, which specifies the text part of the annotation.
See Also

CWCaption
CoordinateType Property

Syntax

`CWAnnotation. CoordinateType`
Data Type

CWCWCoordinateTypes

You can use the following constants with this data type:

- cwAxesCoordinates–Scales the coordinates to the associated axes.
- cwPlotAreaCoordinates–Scales the coordinates in pixels relative to the upper-left corner of the plot area.
- cwScreenCoordinates–Scales the coordinates in pixels relative to the upper-left corner of the graph.
**Purpose**

Specifies how the text and shape coordinates are scaled when the annotation is drawn.
See Also

CWCaption.XCoordinate
CWCaption.YCoordinate
CWSave.XCoordinates
CWSave.YCoordinates
Enabled Property

Syntax

CWAnnotation.Enabled
Data Type

Boolean
Purpose

Specifies if the CWAnnotation generates mouse events or if you can drag the annotation.
Remarks

By default, the Enabled property is set to True.

If a user clicks on a disabled annotation, the annotation does not respond. Other annotations, cursors, plots, or the plot area can still generate an event.

You must set the CWGraph.TrackMode property to cwGTrackDragAnnotation before you can drag an annotation with your cursor.
See Also

CWGraph.TrackMode
<table>
<thead>
<tr>
<th>Name Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
</tr>
<tr>
<td>CWAnnotation.Name</td>
</tr>
</tbody>
</table>
Data Type

String
Purpose

Specifies the name of the annotation.
Plot Property

Syntax

Set CWAnnotation . Plot
Data Type

CWPlot
**Purpose**

Specifies the plot that the annotation is associated with.
Remarks

If CWAnnotation.CoordinateType is set to cwAxesCoordinates, the annotation uses the plot specified by the Plot property to scale text and shape coordinates. If CWAnnotation.SnapMode is set to True, the annotation shape is centered around the point on this plot with the index value of the CWAnnotation.PointIndex property.
Example

'Anchor the second annotation to the specified plot at point 25
CWGraph1.Annotations.Item(2).Plot = CWGraph1.Plots(1)
'Remember that PointIndex is zero-based
CWGraph1.Annotations.Item(2).PointIndex = 24
CWGraph1.Annotations.Item(2).SnapMode = cwCSnapPointsOnPlot
See Also

CWAnnotation.CoordinateType
CWAnnotation.SnapMode
CWAnnotation.PointIndex
PointIndex Property

Syntax

`CWAnnotation.PointIndex`
Data Type
Long
**Purpose**

Specifies the index of a plot point to center the shape around.
Remarks

The PointIndex property is zero-based. The property is best used with CWShape.Type set to cwShapeNone or cwShapePoint.

If CWAnnotation.SnapMode is set to True, the annotation shape is centered around or on this point index on the plot specified by the Plot property.
Example

'Anchor the second annotation to the specified plot at point 25
CWGraph1.Annotations.Item(2).Plot = CWGraph1.Plots(1)
'Remember that PointIndex is zero-based
CWGraph1.Annotations.Item(2).PointIndex = 24
CWGraph1.Annotations.Item(2).SnapMode = cwCSnapPointsOnPlot
See Also

CWAnnotation.SnapMode

CWAnnotation.Plot
Shape Property (Read Only)

Syntax

`CWAnnotation.Shape`
Data Type

CWShape
Purpose

Returns a CWSHape object, which specifies the shape part of the annotation.
See Also

CWShape
SnapMode Property

Syntax

CWAnnotation.SnapMode
Data Type

**CWCursorSnapModes**

You can use the following constants with this data type:

- **cwCSnapAnchoredToPoint**–Snaps the cursor or annotation to a specified point on a specific plot. The cursor or annotation cannot be moved.
- **cwCSnapFixed**–Sets the cursor or annotation position independent of any plot. The cursor or annotation cannot be moved.
- **cwCSnapFloating**–Sets the cursor or annotation independent of any plot. You can move the cursor or annotation anywhere in the graph area.
- **cwCSnapNearestPoint**–Snaps the cursor or annotation to the nearest point on any plot. You can move the cursor or annotation along any plot.
- **cwCSnapNearestYForFixedX**–For a fixed X location, the cursor's Y value is the Y value of the nearest point on the specified plot. For an annotation cwCSnapNearestYForFixedX is equivalent to cwCSnapPointsOnPlot. The cursor or annotation can be moved only along this plot.
- **cwCSnapPointsOnPlot**–Snaps the cursor or annotation to points on a specified plot. The cursor or annotation can be moved only along this plot.
**Purpose**

 Specifies if the annotation's shape is centered around the plot point defined by the Plot and PointIndex properties and how the shape is dragged by the mouse.
Remarks

If you set SnapMode to cwCSnapFloating, the shape is drawn at the values specified by CWShape.XCoordinates and CWShape.YCoordinates, and you can move the entire annotation anywhere in the graph area. If you set SnapMode to cwCSnapPointsOnPlot, the shape is centered around the plot point defined by the Plot and PointIndex properties, and you can move the shape along the plot.
Example

'Anchor the second annotation to the specified plot at point 25
CWGraph1.Annotations.Item(2).Plot = CWGraph1.Plots(1)
'Remember that PointIndex is zero-based
CWGraph1.Annotations.Item(2).PointIndex = 24
CWGraph1.Annotations.Item(2).SnapMode = cwCSnapPointsOnPlot
See Also

- CWAnnotation.Plot
- CWAnnotation.PointIndex
- CWShape
- CWShape.XCoordinates
- CWShape.YCoordinates
Visible Property

Syntax

CWAAnnotation. Visible
Data Type

Boolean
Purpose

Specifies if an annotation is visible.
SetBuiltinStyle Method

Syntax

`CWAnnotation.SetBuiltinStyle Style`
Purpose
Sets many properties of the annotation to represent the new style specified.
Parameters

Style As CWAnnotationStyles

The style of the annotation.
**Example**

CWAnnotation.SetBuiltInStyle cwAnnotationStyleText
Count Property (Read Only)

Syntax

Object.Count
Data Type

Integer
Applies To

CWAAnnotations
CWAxes
CWBindings
CW.Cursors
CWPlots
CWPPointers
CW.ValuePairs
**Purpose**

Returns the number of objects in the collection.
Example

'Same syntax applies to other collections
numPlots = CWGraph1.Plots.Count
Add Method

Syntax

Object.Add ()
Return Type

Object

Newly created object.
Applies To

CWAnotations
CWAxes
CWBbindings
CWCurvers
CWPlots
CWPointers
CWValuePairs
**Purpose**

Adds an object to the collection and returns the new object.
Example

'Add an annotation to the graph
CWGraph1.Annotations.Add
Item Method

Syntax

Object.Item (Item)
Return Type

Object

The object specified.
Applies To

- CWAnnotations
- CWAxes
- CWBindings
- CW Cursors
- CWPlots
- CWPointers
- CWValuePairs
**Purpose**

Returns the specified object from the collection.
Parameters

**Item As Variant**

Specifies either the name of the object or the index of the object in the collection. The index of the object is one-based.
Example

'Assign object in collection to variable
Dim Plot1 as CWPlot
Set Plot1 = CWGraph1.Plots.Item(1)

'Call property or method on object in collection
CWGraph1.Plots.Item(1).ChartY data, 1, True

'Access an item by name rather than index
CWSlide1.Pointers.Item("BoilerPressure")
Remove Method

Syntax

Object.Remove Item
Applies To

CWAnnotations
CWAxes
CWBindings
CWCursors
CWPlots
CWPointers
CWValuePairs
**Purpose**

Removes the specified item from the collection.
Parameters

Item As Variant

Represents either the name of the object or the index of the object in the collection. The index of the object is one-based.
Example

'Remove third pointer from slide
CWSlide1.Pointers.Remove 3
RemoveAll Method

Syntax

\textit{Object.RemoveAll}
Applies To

CWAAnnotations
CWAxes
CWBindings
CW Cursors
CWPlots
CWPointers
CWValuePairs
Purpose

Removes all objects from the collection.
Example

'Remove all value pairs from the knob
C WKnob1.Axis.ValuePairs.RemoveAll
Color Property

Syntax

\texttt{CWA\textcolor{Color}{arrow}}
Data Type

Color
**Purpose**

Specifies the color of the arrow.
HeadStyle Property

Syntax

CWArrow. HeadStyle
Data Type

**CWArrowHeadStyles**

You can use the following constants with this data type:

- `cwArrowHeadDiamond`– Specifies a diamond-shaped arrowhead.
- `cwArrowHeadLine`– Specifies an arrowhead with thin lines.
- `cwArrowHeadNone`– Specifies no arrowhead.
- `cwArrowHeadRound`– Specifies a round arrowhead.
- `cwArrowHeadSolid`– Specifies a solid arrowhead.
- `cwArrowHeadStealth`– Specifies a stylized arrowhead.
Purpose
Specifies the style of the arrowhead that points to the CWShape object.
See Also

CWAnnotation.Shape
LineStyle Property

Syntax

CWArow .LineStyle
Data Type

You can use the following constants with this data type:

- cwLineDash–Dashed line style.
- cwLineDashDot–Dash-dot line style.
- cwLineDashDotDot–Dash-dot-dot line style.
- cwLineDot–Dotted line style.
- cwLineNone–No line style.
- cwLineSolid–Solid line style.
- cwLineStepXY–Step XY line style.
- cwLineStepYX–Step YX line style.
**Purpose**

Specifies the line style of the arrow.
Remarks

You can only use cwLineStepXY and cwLineStepYX with the CWPlot.LineStyle property.
TailStyle Property

Syntax

\texttt{CWArrow.\ TailStyle}
Data Type

CWAArrowHeadStyles

You can use the following constants with this data type:

- cwArrowHeadDiamond–Specifies a diamond-shaped arrowhead.
- cwArrowHeadLine–Specifies an arrowhead with thin lines.
- cwArrowHeadNone–Specifies no arrowhead.
- cwArrowHeadRound–Specifies a round arrowhead.
- cwArrowHeadSolid–Specifies a solid arrowhead.
- cwArrowHeadStealth– Specifies a stylized arrowhead.
**Purpose**

 Specifies the style of an arrowhead that points to the caption.
See Also

CWAnnotation Caption
HeadStyle
Visible Property

Syntax

`CWArrow. Visible`
Data Type

Boolean
**Purpose**

Specifies if the arrow is visible.
Remarks

By default, the Visible property is set to True.
Width Property

Syntax

 CWArrow. Width
Data Type

Integer
Purpose

Specifies the width of the arrow.
Axes Property (Read Only)

Syntax

`CWGraph.Axes`
Data Type

CWAxes
Purpose

Specifies a collection of CWAxis objects.
Remarks

The collection contains only one X and at least one Y CWAxis object.
See Also

CWAxes
CWAxis
AutoScale Property

Syntax

`CWAxis.AutoScale`
Data Type

Boolean
Purpose
Determines if the minimum and maximum limits of the axis are automatically set.
Remarks

The AutoScale property applies only to the CWGraph control. It affects only CWPlots objects that also have the AutoScale property set to True.
Example

CWGraph1.Axes.Item(1).AutoScale = True
See Also

CWPlot.AutoScale
AutoScaleNow
Caption Property

Syntax

`CWAxis.Caption`
Data Type

String
**Purpose**

Specifies the text to draw on the axis.
Remarks

The caption is drawn where the labels are drawn on an axis. For example, if labels are drawn to the right of a vertical axis, the caption will be drawn only on the right. On a vertical axis, the caption is drawn vertically.

On the CWKnob control, the caption positioning is determined by the knob style.
See Also

Labels
CaptionColor
CaptionColor Property

Syntax

\texttt{CWAxis.CaptionColor}
Data Type

Color
Purpose

 Specifies the color used to draw the caption.
See Also

Caption
Discrete Property

Syntax

\texttt{CWA\textsubscript{axis}.Discrete}
Data Type

Boolean
Purpose

Represents only discrete values on the axis, according to the base and interval properties.
Remarks

A discrete axis limits the valid values. All values mapped to the axis, such as pointers and graph data, are assigned to the closest discrete value. For example, you can create a discrete axis of integers where the valid values are -1, 0, 1, 2, 3, and so on.

Typically, you create tick marks with the same base and interval values. This property does not affect the axis when its Log property is set to True.
**Example**

'Setup the CWSlide control to take only integer data
CWSlide1.Axis.Discrete = True
CWSlide1.Axis.DiscreteInterval = 1
CWSlide1.Axis.DiscreteBase = 1

'Setup the CWSlide control to round all data to the 
'nearest 0.5
CWSlide1.Axis.Discrete = True
CWSlide1.Axis.DiscreteInterval = .5
CWSlide1.Axis.DiscreteBase = 0
See Also

DiscreteBase
DiscreteInterval
DiscreteBase Property

Syntax

\texttt{CWAxis.DiscreteBase}
Data Type

Variant
**Purpose**

Specifies the base value for discrete axes.
Remarks

Valid values on a discrete axis are described by the equation:

\[ \text{ValidValue} = \text{DiscreteBase} + (n \times \text{DiscreteInterval}) \]

where \( n \) is any integer, positive or negative. The \text{DiscreteBase} and \text{DiscreteInterval} properties specify the discrete values that are valid on the axis. For example, for discrete values of integers, set \text{DiscreteBase} to 0 and \text{DiscreteInterval} to 1. For discrete values of 0.25, 0.5, and 0.75, set \text{DiscreteBase} to 0 and \text{DiscreteInterval} to 0.25.
**Example**

'Setup the CWSlide control to take only integer data
CWSlide1.Axis.Discrete = True
CWSlide1.Axis.DiscreteInterval = 1
CWSlide1.Axis.DiscreteBase = 1

'Setup the CWSlide control to round all data to the
'nearest 0.5
CWSlide1.Axis.Discrete = True
CWSlide1.Axis.DiscreteInterval = .5
CWSlide1.Axis.DiscreteBase = 0
See Also

Discrete
DiscreteInterval
DiscreteInterval Property

Syntax

CWAxis.DiscreteInterval
## Data Type

- **Variant**
**Purpose**

Specifies the interval between discrete values.
**Example**

'Setup the CWSlide control to take only integer data
CWSlide1.Axis.Discrete = True
CWSlide1.Axis.DiscreteInterval = 1
CWSlide1.Axis.DiscreteBase = 1

'Setup the CWSlide control to round all data to the
'nearest 0.5
CWSlide1.Axis.Discrete = True
CWSlide1.Axis.DiscreteInterval = .5
CWSlide1.Axis.DiscreteBase = 0
See Also

Discrete
DiscreteBase
FormatString Property

Syntax

`CWAxis.FormatString`
Data Type

String
Purpose

Specifies the format string for formatting the labels on the axis.
Remarks

The formatting string is similar to the string for the Format function in Visual Basic. The following formatting characters are available:

. (decimal point) Specifies the beginning of the number within the label. Use # and 0 to the right of the decimal point to specify the precision. A decimal point is assumed to be the first character in the format string if you do not include a decimal point in the format string.

0 Specifies the precision to the right of the decimal point. For example ".#0" always produces two digits to the right of the decimal point even when given an exact number such as 1.0.

# Specifies the precision to the right of the decimal point. For example ".##" produces up to two digits to the right of the decimal point. Thus 1.0 produces 1 while 1.025 produces 1.03.

e or E Specifies exponential notation. "E" specifies the capital letter. "e" specifies the lower case letter.

*nn Scales labels. For example ".*10" prints the value 1.0 as 10.

+nn or -nn Offsets labels. For example ".+20" prints the value 1.0 as 21.

k Specifies symbolic notation. For the format string ".k" 1.0 prints as 1 and 1000 prints as 1k and .001 prints as 1m.

"text" Specifies the addition of text to the label. For example ".V" adds a V to the right of every label. Thus 1.0 becomes 1 V.

$ or % Specifies to print these special characters.

: (colon) Time separator. The time separator uses hours:minutes:seconds when time values are formatted.

/ Date separator. The date separator uses day/month/year when date values are formatted.

d Displays the day as a number without a leading zero. For example: 1-3

,dd Displays the day as a number with a leading zero. For example: 01-31

ddd Displays the day as an abbreviation. For example: Sun - Sat.

dddd Displays the day as a full name. For example: Sunday - Saturday
w Displays the day of the week as a number. For example: 1 for Sunday through 7 for Saturday
ww Displays the week of the year as a number. For example: 1-53
m Displays the month as a number without a leading zero. For example: 1 - 12
mm Displays the month as a number with a leading zero. For example: 01 - 12
mmm Displays the month as an abbreviation. For example: Jan - Dec
mmmm Displays the month as a full month name January - December
q Displays the quarter of the year as a number. For example: 1 - 4
y Displays the day of the year as a number. For example: 1 - 366
yy Displays the year as a 2-digit number. For example: 00 - 99
yyyy Displays the year as a 4-digit number. For example: 1000 - 9999
h Displays the hour as a number without leading zeros. For example: 0 - 23
hh Displays the hour as a number with leading zeros. For example: 00 - 23
n Displays the minute as a number without leading zeros. For example: 0 - 59.
nn Displays the minute as a number with leading zeros. For example: 00 - 59
s Displays the second as a number without leading zeros. For example: 0 - 59
ss Displays the second as a number with leading zeros. For example: 00 - 59
AM/PM Uses the 12-hour clock and displays "AM" with any hour before noon. Displays "PM" with any hour between noon and 11:59 PM
am/pm Uses the 12-hour clock and displays "am" with any hour before noon. Displays "pm" with any hour between noon and 11:59 pm
A/P Uses the 12-hour clock and displays "A" with any hour before noon. Displays "P" with any hour between noon and 11:59 PM.
a/p Uses the 12-hour clock and displays "a" with any hour before noon.
Displays "p" with any hour between noon and 11:59 pm.

If you select a format string that includes date or time formatting, the property pages will display all values in a generic date and time format for editing.

Additionally, any property can be set programmatically with a string representing a date and/or time. Thus, a statement such as: `CWSlide1.Value = "01/08/1923 9:12:33.14 pm"` is a valid value. This statement can be abbreviated to represent just the time or just the date. For example, `CWSlide1.Value = "1/1/73"` and `CWSlide1.Value = "9:15 pm"` are valid as well.

With ActiveX controls, the date is implemented as a floating-point value, measuring days from midnight, 30 December 1899. So, midnight, 31 December 1899, is represented by 1.0. Similarly, 6 AM, 1 January 1900, is represented by 2.25, and midnight, 29 December 1899, is -1.0. However, 6 AM, 29 December 1899, is -1.25. To interpret the time portion, take the absolute value of the fractional part of the number. Thus, 1 second equals 1 / 24 hours / 60 minutes / 60 seconds, which is 1/86400 or approximately 1.157407e-5.
Example

'Display up to two digits to the right of the decimal on
'the x axis of the graph
CWGraph1.Axes.Item(1).FormatString = ".##"

'Use time formatting on the CWSlide control's axis
CWSlide1.Axis.FormatString = "hh:nn:ss"
CWSlide1.Value = "13:12:15"

'Use date formatting on the CWKnob control's axis
CWKnob1.Axis.FormatString = "mm/dd/yyyy"
CWKnob1.Value = "5/31/1986"
See Also

CWNumEdit.FormatString
Inverted Property

Syntax

\texttt{CWAxis.Inverted}
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
</tr>
</tbody>
</table>
Purpose
Specifies if the direction of an axis is inverted.
Remarks

An inverted vertical axis has the maximum value at the bottom of the axis. An inverted horizontal axis has the maximum value at the left of the axis. For a radial axis on a CWKnob control, an inverted axis has the maximum value clockwise from the minimum value. Inverted is the default position for a radial axis.
Labels Property (Read Only)

Syntax

`$\text{CWAxis}.Labels$`
Data Type

CWLabels
Purpose

Returns a CWLabel object, which specifies how labels appear on the axis.
See Also

Ticks
ValuePairs
CWLabels
Log Property

Syntax

\texttt{CWAaxis.Log}
Data Type

Boolean
**Purpose**

Specifies if the axis has a Log10 scale.
Remarks

When you change the Log property from a property page, you can change the settings for minimum values, maximum values, and tick mark placement to work better with a log scale.
Example

'Set the y axis of the graph to log mode
CWGraph1.Axes.Item(2).Log = True
Maximum Property

Syntax

`CWAxis(Maximum)`
Data Type

Variant
Purpose

Specifies the maximum value of the axis.
Remarks

Use the SetMinMax method to set the minimum and maximum values simultaneously.
Example

'Set the minimum and maximum values of the x axis
CWGraph1.Axes.Item(1).Minimum = 0
CWGraph1.Axes.Item(1).Maximum = 100

'Alternately, call SetMinMax to specify both values.
CWGraph1.Axes.Item(1).SetMinMax 0, 100
See Also

SetMinMax
Minimum
Inverted
Minimum Property

Syntax

\texttt{CWAxis.Minimun}
Data Type

Variant
Purpose

Specifies the minimum value of the axis.
Remarks
Use the SetMinMax method to set the minimum and maximum values simultaneously.
**Example**

'Set the minimum and maximum values of the x axis
CWGraph1.Axes.Item(1).Minimum = 0
CWGraph1.Axes.Item(1).Maximum = 100

'Alternately, call SetMinMax to specify both values.
CWGraph1.Axes.Item(1).SetMinMax 0, 100
See Also

SetMinMax
Maximum
Inverted
Name Property

Syntax

`CWAxis.Name`
Data Type

String
Purpose

Specifies the name of the axis.
Remarks

Use this name for indexing the Axes collection on a CWGraph object. If more than one axis has the same name, the first matching axis is used.
ScaleStyleValuePairOnly Property

Syntax

`CWAxis.ScaleStyleValuePairOnly`
Data Type

Boolean
Purpose

Specifies if only the value pairs are displayed on the axis.
Remarks

If you set this property to true, the value pairs are displayed on the axis.
Ticks Property (Read Only)

Syntax

```
CWAxis.Ticks
```
Data Type

CWTicks
**Purpose**

Returns a CWTicks object, which specifies how divisions and ticks appear on this axis.
ValuePairs Property (Read Only)

Syntax

`CWAxis.ValuePairs`
Data Type

CWPValuePairs
**Purpose**

Returns a CWValuePairs collection of CWValuePair objects, which specify labels for particular points on the axis.
Remarks

A ValuePair object is a label paired with a value. A slide control with text labels such as "Off", "Slow", and "Fast" uses CWValuePairs to define the labels. For example, use a CWValuePair object to add the text label "BoilingPoint" to a numeric axis at the value 212.

Refer to the CWValuePair topics for examples.
Example

'Add a value pair to the graph's x-axis
CWGraph1.Axes.Item(2).ValuePairs.Add

'Set the name and value of the value pair we just added
CWGraph1.Axes.Item(2).ValuePairs.Item(1).Name = "Test"
CWGraph1.Axes.Item(2).ValuePairs.Item(1).Value = 8
See Also

CWValuePairs
CWValuePair
Visible Property

Syntax

\texttt{CWAxis.Visible}
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
</tr>
</tbody>
</table>
**Purpose**

Specifies if the axis is visible or hidden.
Remarks

On a graph control, the CWPlot object resizes the plot area to accommodate the change in visibility. The size of a slide or knob control does not change.
AutoScaleNow Method

Syntax

`CWAxis.AutoScaleNow`
**Purpose**

Causes the axis to rescale immediately.
Remarks
The axis is autoscaled, regardless of the setting of the AutoScale property. This method applies only to axes on a CWGraph.
Example

'Force the axis to rescale
CWGraph1.Axes.Item(1).AutoScaleNow
See Also

AutoScale
SetMinMax Method

Syntax

CWAxis.SetMinMax Maximum, Minimum
**Purpose**

Sets both the minimum and the maximum values of the axis at the same time.
Remarks

Use this method to avoid setting a minimum value that is greater than the maximum, or a maximum value that is less than the minimum.

To make the axis appear inverted, set the Inverted property.
Parameters

**Maximum** As **Variant**
The new maximum value for the axis.

**Minimum** As **Variant**
The new minimum value for the axis.
Example

'Set the minimum and maximum values of the x axis
CWGraph1.Axes.Item(1).Minimum = 0
CWGraph1.Axes.Item(1).Maximum = 100

'Alternately, call SetMinMax to specify both values.
CWGraph1.Axes.Item(1).SetMinMax 0, 100
See Also

Minimum
Maximum
Inverted
AccessMode Property

Syntax

`CWBinding.AccessMode`
Data Type

You can use the following constants with this data type:

- **cwdsRead**– Reads once when connected. You can trigger another read by calling the Update method.
- **cwdsReadAutoUpdate**– Reads when connected and automatically reads again if the data at the data source is updated.
- **cwdsReadWriteAutoUpdate**– Writes the current data once connected. The data is rewritten automatically if any attribute or value is set. Reads when connected and automatically reads again if the data at the data source is updated.
- **cwdsWrite**– Writes the current data once connected. You can trigger the write again by calling the Update method.
- **cwdsWriteAutoUpdate**– Writes the current data once connected. The data is rewritten automatically if any attribute or value is set.
Purpose

Specifies the read or write connection to make when connecting to the data source.
Remarks

Alternatively, you can set this property when you call the ConnectTo method.
Example

'Create a binding that charts wave data
'from a local DataSocket Server on a graph.
Dim Binding As CWBinding

'Add new CWBinding
CWGraph1.CWBindings.Add
Set Binding = CWGraph1.CWBindings.Item(1)

'Set AccessMode to automatically read and update
Binding.AccessMode = cwdsReadAutoUpdate
'Specify the data source on the DataSocket Server
Binding.URL = "dstp://localhost/wave"

'Bind the URL to CWGraph control, YData property
Binding.SetBindObject CWGraph1
Binding.BindProperty = "YData"

'Connect to data source
Binding.Connect
See Also

Connect
ConnectTo
ActualURL Property (Read Only)

Syntax

`CWBinding.ActualURL`
Data Type

String
Purpose

Identifies the actual URL of the current data source.
Remarks

Once connected to a data source or target, the ActualURL property might be different from the URL property if the original URL pointed to a link that redirected the DataSocket to a new URL. If DataSocket is not connected, the value of ActualURL is an empty string.
Example

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWBindings.Item(1)

'Display actual URL in a text box on user interface
Text1.Text = Binding.ActualURL
See Also

URL
Connect
ConnectTo
AutoConnect Property

Syntax

CWBinding.AutoConnect
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
</tr>
</tbody>
</table>
Purpose

Specifies if the binding automatically connects to the source as soon as the program is run.
Remarks

Set the URL, Connection mode, and Auto Connect properties in the property pages at design time to automatically connect the CWBinding when the program is run or the Web page is loaded. You do not have to call the Connect or ConnectTo methods.

Like the Connect and ConnectTo methods, this property causes the control to generate the CWBindingStatusUpdated event when the connection is made and the CWBindingDataUpdated event when data is updated.
See Also

AccessMode
Connect
ConnectTo
URL
CWBindingDataUpdated
CWBindingStatusUpdated
BindProperty Property

Syntax

CWBinding.BindProperty
Data Type

String
Purpose

Specifies the name of the property that you are binding to an external data item.
Remarks

When you are binding a property to a data item, first set the bind control or object with the SetBindObject method, and then use this property to specify the bind property on that control or object.
Example

'Create a binding that charts wave data
'from a local DataSocket Server on a graph.
Dim Binding As CWBinding

'Add new CWBinding
CWGraph1.CWBindings.Add
Set Binding = CWGraph1.CWBindings.Item(1)

'Set AccessMode to automatically read and update
Binding.AccessMode = cwdsReadAutoUpdate
'Specify the data source on the DataSocket Server
Binding.URL = "dstp://localhost/wave"

'Bind the URL to CWGraph control, YData property
Binding.SetBindObject CWGraph1
Binding.BindProperty = "YData"

'Connect to data source
Binding.Connect
See Also

SetBindObject
DataUpdated Property (Read Only)

Syntax

CWBinding.DataUpdated
Data Type

Boolean
**Purpose**

Indicates if value or attributes on the CWBinding object have been set since they were last read.
Remarks

This property is automatically set to True when the value or attributes of the CWBinding data have been set and automatically reset to False after this property is queried.
Example

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWBindings.Item(1)

Dim bDataUpdated As Boolean
'See if the binding data has been updated
bDataUpdated = Binding.DataUpdated
If bDataUpdated Then
  'Do something
End If
See Also

CWBindingDataUpdated
DataUpdatedEnabled
DataUpdatedEnabled Property

Syntax

`CWBinding.DataUpdatedEnabled`
Data Type

Boolean
**Purpose**

Indicates if the binding generates the CWBindingDataUpdated event when the bound data changes.
Remarks

Set this property to True to generate the CWBindingDataUpdated event. Use this event to view, manipulate, analyze, and even ignore updated data.

To enable the CWBindingDataUpdated event programmatically, set this property to True before connecting to the data source.
Example

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWBindings.Item(1)

'Enable the CWBindingDataUpdated event
Binding.DataUpdatedEnabled = True
See Also

CWBindingDataUpdated
LastError Property (Read Only)

Syntax

`CWBinding.LastError`
| Data Type | Long |
Purpose

Returns the last error code used in an CWBindingStatusUpdated event.
Remarks

The value of LastError is 0 if there was no error the last time the CWBindingStatusUpdated event was generated.

Some common errors include errors caused by incorrect network configurations, insufficient access privileges to connect to the data source, or an incorrectly formed URL.

The value of the error code is an HRESULT, the ActiveX data type used for reporting errors. It might include errors detected by the CWBinding or by the operating systems networking services. If an error is encountered, the message that goes with the last error is stored in the LastMessage property.

To determine the task the CWBinding object was performing when the error occurred, check the LastMessage and Status properties.
Example

'Display the current status in the form
'[status:error] Message

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWBindings.Item(1)

Dim str As String
If (Binding.StatusUpdated) Then
    str = "[" & CWBinding.Status & ":" & _
            CWBinding.LastError & "]" & _
            CWBinding.LastMessage
    MsgBox str
End If
See Also

CWBindingStatusUpdated
LastMessage
Status
StatusUpdated
LastMessage Property (Read Only)

Syntax

CWBinding.LastMessage
| Data Type  | String |
Purpose
Stores the last message used in a CWBindingStatusUpdated event.
Remarks

The message describes either the last error encountered or the last step taken in connecting or updating the data.
Example

'Display the current status in the form
'[status:error] Message

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWBindings.Item(1)

Dim str As String
If (Binding.StatusUpdated) Then
    str = "[" & CWBinding.Status & ":" & _
        CWBinding.LastError & "]" & _
    CWBinding.LastMessage
    MsgBox str
End If
See Also

`CWBindingStatusUpdated`
`LastError`
Name Property

Syntax

_CWBinding_.Name
Data Type

Boolean
Purpose

Specifies the name of the CWBinding object.
Example

CWGraph1.CWBindings(2).Name = "SineData"
Status Property (Read Only)

Syntax

CWBinding.Status
Data Type

CWDSStatus

You can use the following constants with this data type:

- cwdsConnecting–DataSocket is in the process of connecting to the data source or target.
- cwdsConnectionActive–DataSocket is in the process of transferring the data or waiting for an update.
- cwdsConnectionError–DataSocket encountered an error connecting to the data source or target.
- cwdsConnectionIdle–DataSocket has connected to the data source and transferred the data.
- cwdsUnconnected–DataSocket is not connected to any data source or data target.
Purpose
Specifies the current status of the data connection.
Remarks

The value of this property is the same as the last status value passed to the CWBindingStatusUpdated event.

If an error is encountered while connecting to the source or target, the status indicates the last step attempted. The LastError and LastMessage properties describe the error.
Example

'Display the current status in the form
'[status:error] Message

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWBindings.Item(1)

Dim str As String
If (Binding.StatusUpdated) Then
    str = "[" & CWBinding.Status & ":" & _
        CWBinding.LastError & "]" & _
        CWBinding.LastMessage
    MsgBox str
End If
See Also

CWBindingStatusUpdated
LastError
LastMessage
StatusUpdated Property (Read Only)

Syntax

CWBinding.StatusUpdated
Data Type

Boolean
**Purpose**
Indicates if the binding status has changed or if an error has occurred.
Remarks

The property is set to True when the CWBindingStatusUpdated event is generated and reverts to False when it is queried.
**Example**

'Display the current status in the form
'[status:error] Message

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWBindings.Item(1)

Dim str As String
If (Binding.StatusUpdated) Then
  str = "[" & CWBinding.Status & ":" & _
  CWBinding.LastError & "]" & _
  CWBinding.LastMessage
  MsgBox str
End If
See Also

Status

CWBindingStatusUpdated
TimerInterval Property

Syntax

CWBinding.TimerInterval
Data Type
Long
Purpose

Automatically calls the Update method at a regular interval specified in milliseconds.
Remarks

If you connect to a data source with the Read or Write access mode (and not ReadAutoUpdate or WriteAutoUpdate), use this property to receive data updates in regular intervals. ReadAutoUpdate and WriteAutoUpdate update the date every time the data changes.
Example

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWBindings.Item(1)

'Have Binding return new data once per second
Binding.TimerInterval = 1000
Binding.AccessMode = cwdsRead

'Connect to data source
Binding.Connect
See Also

AccessMode
Update
URL Property

Syntax

 CWBinding.URL
Data Type

String
Purpose

Specifies the location as a URL of the data source or target to which you are binding.
Remarks

You also can set the URL with the ConnectTo method call.

The CWBinding can connect to different data sources or targets, depending on the specified URL. The AccessMode property determines if the CWBinding is reading a data source or writing to a data target.

If the data source or target pointed to by the URL redirects the CWBinding to a new URL, the ActualURL property is set to the new URL.

The following types of URLs are supported:

DataSocket Server ("dstp:")
Examples: dstp://localhost/wave, dstp://machine/item

Standard Web Server ("http:")
Example: http://www.ni.com/cworks/datasocket/tone.wav

Standard FTP Sites ("ftp:"). The ftp site should allow anonymous connections.

Files directly accessible from your file system ("file:")
Examples: file:ping.wav, file:c:\mydata\ping.wav, file:\machine\mydata\ping.wav

OLE for Process Control (OPC) Servers ("opc:")
Examples: opc:\National Instruments.OPCDemo\sine, opc:\National Instruments.OPCDemo\sine?Accesspath=sine, opc:\machine\National Instruments.OPCModbus\Modbus Demo Box.4:0, opc:\machine\National Instruments.OPCModbus\Modbus Demo Box.4:0?updaterate=100&deadband;=0.7
Example

'Create a binding that charts wave data 
'from a local DataSocket Server on a graph. 
Dim Binding As CWBinding

'Add new CWBinding 
CWGraph1.CWBindings.Add 
Set Binding = CWGraph1.CWBindings.Item(1)

'Set AccessMode to automatically read and update 
Binding.AccessMode = cwdsReadAutoUpdate 
'Specify the data source on the DataSocket Server 
Binding.URL = "dstp://localhost/wave"

'Bind the URL to CWGraph control, YData property 
Binding.SetBindObject CWGraph1 
Binding.BindProperty = "YData"

'Connect to data source 
Binding.Connect
See Also

AccessMode
ActualURL
Connect
ConnectTo
Connect Method

Syntax

`CWBinding.Connect`
**Purpose**

Connects the CWBinding to a data source or target.
Remarks

You must set the URL and AccessMode properties before you can use the Connect method. Use the ConnectTo method to specify the URL and AccessMode properties as part of the method call.

If you connect a reading client to a DataSocket item that does not exist, the DataSocket Server creates the item with a default value of 0.
'Create a binding that charts wave data
'from a local DataSocket Server on a graph.
Dim Binding As CWBinding

'Add new CWBinding
CWGraph1.CWBindings.Add
Set Binding = CWGraph1.CWBindings.Item(1)

'Set AccessMode to automatically read and update
Binding.AccessMode = cwdsReadAutoUpdate
'Specify the data source on the DataSocket Server
Binding.URL = "dstp://localhost/wave"

'BIND the URL to CWGraph control, YData property
Binding.SetBindObject CWGraph1
Binding.BindProperty = "YData"

'Connect to data source
Binding.Connect
See Also

- AccessMode
- ConnectTo
- URL
ConnectTo Method

Syntax

`CWBinding.ConnectTo URL, AccessMode`
**Purpose**

Connects the CWBinding object to a data source or target.
Remarks

The ConnectTo method sets the URL and AccessMode properties to the values of the URL and AccessMode parameters passed to the method.

If you connect a reading client to a DataSocket item that does not exist, the DataSocket Server creates the item with a default value of 0.

If you want to manipulate data as it is written or read, set the DataUpdatedEnabled property to True so that the CWBindingDataUpdated event is generated. The CWBindingDataUpdated event is generated when data is ready to be sent (in write mode) or has just been received (in read mode). You might use the CWBindingDataUpdated event to scale values or ignore them if they don't meet your criteria.
Parameters

**URL As String**

URL of the data source or target to which you are connecting.

**AccessMode As CWDSAccessModes**

Access mode of the connection.
**Example**

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWBindings.Item(1)

'Connect Binding to wave item on local DataSocket Server
Binding.ConnectTo "dstp://localhost/wave", cwdsReadAutoUpdate
See Also

AccessMode
Connect
CWBindingDataUpdated
DataUpdated
URL
Disconnect Method

Syntax

`CWBinding.Disconnect`
**Purpose**

Disconnects the CWBinding from the data item to which it is currently connected.
Remarks

Disconnect has no effect if the binding is not connected.
Example

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWbindings.Item(1)

Binding.Disconnect
See Also

Connect
ConnectTo
SelectURL Method

Syntax

`CWBinding.SelectURL ( [startURL] [, Title] [, Options] [, Filters])`
Return Type

Boolean

Returns True if you selected a data item; False if the user cancelled. If True, the URL property is set to the URL that the user selected.
**Purpose**

Enables interactive browsing and selection of data items and files on DataSocket and OPC servers. With this method you can easily create data URLs, rather than constructing the URLs yourself, or display a simple text box in which you can enter HTTP and FTP URLs.
Parameters

**startURL** As **Variant**

[Optional] String containing the URL. If startURL is "opc:“, "dstp:“, or "file“, this method opens an OPC server browser, DataSocket server browser, or a file browser to enable users to interactively find and select data items. You can prompt users to create a Web or FTP location with the "http:“ or "ftp:“ URL schemes. If you omit this parameter, a simple text box is opened, and users can enter a URL with a different scheme. You also can use an entire URL, such as "dstp://localhost/wave".

**Title** As **Variant**

[Optional] Title of the SelectURL dialog box.

**Options** As **Variant**

[Optional] Flags to pass to the dialog:

0x00002000 (CREATEGUARD): Opens a dialog box to prompt the user for permission to create the file if that file does not already exist. If the user chooses to create the file, the dialog box closes and the function returns the specified name; otherwise, the dialog box remains open.

0x00001000 (FILEMUSTEXIST): Specifies that the user can type only names of existing files in the File Name entry field. If this flag is specified and the user enters an invalid name, the dialog box procedure displays a warning in a message box.

0x00000004 (HIDEREADONLY): Hides the Read Only check box.

0x00000008 (NOCHANGEDIR): Restores the current directory to its original value if the user changed the directory while searching for files.

0x00100000 (NODEREFERENCELINKS): Directs the dialog box to return the path and filename of the selected shortcut (.LNK) file. If this value is not given, the dialog box returns the path and filename of the file referenced by the shortcut.

0x00008000 (NOREADONLYRETURN): Specifies that the returned file does not have the Read Only check box checked and is not in a write-protected directory.

0x00010000 (NOTESTFILECREATE): Specifies that the file is not created
before the dialog box is closed. This flag should be specified if the application saves the file on a create-nonmodify network share. When an application specifies this flag, the library does not check for write protection, a full disk, an open drive door, or network protection. Applications using this flag must perform file operations carefully, because a file cannot be reopened once it is closed.

0x00000002 (OVERWRITEPROMPT): Causes the Save As dialog box to generate a message box if the selected file already exists. The user must confirm whether to overwrite the file.

0x00000800 (PATHMUSTEXIST): Specifies that the user can type only valid paths and filenames. If this flag is used and the user types an invalid path and filename in the File Name entry field, the dialog box function displays a warning in a message box.

0x00000001 (READONLY): Causes the Read Only check box to be checked initially when the dialog box is created.

0x00000020 (SAVEAS): Causes the file dialog to be a Save As file dialog, rather than an Open file dialog.

**Filters As Variant**

[Optional] Filter string to pass to the dialog. For example, All Files(*.*)|*.*
**Example**

'Create a reference to a CWBinding object
'on CWGraph1
Dim Binding As CWBinding
Set Binding = CWGraph1.CWBindings.Item(1)

'Browse for an OPC item
Binding.SelectURL "opc:"

'Browse for a file
Binding.SelectURL "file:"

'Select an entire URL
Binding.SelectURL "dstp://localhost/wave"
See Also

URL
SetBindObject Method

Syntax

CWBinding.SetBindObject pDisp
Purpose

Object whose property will bind to a data source or target. This object might be either a control or one of the objects on that control.
Parameters

pDisp As Object

Measurement Studio control or one of the objects on that control.
Example

'Create a binding that charts wave data
'from a local DataSocket Server on a graph.
Dim Binding As CWBinding

'Add new CWBinding
CWGraph1.CWBindings.Add
Set Binding = CWGraph1.CWBindings.Item(1)

'Set AccessMode to automatically read and update
Binding.AccessMode = cwdsReadAutoUpdate
'Specify the data source on the DataSocket Server
Binding.URL = "dstp://localhost/wave"

'Bind the URL to CWGraph control, YData property
Binding.SetBindObject CWGraph1
Binding.BindProperty = "YData"

'Connect to data source
Binding.Connect
See Also

BindProperty
Update Method

Syntax

CWBinding.Update
Purpose

Causes the CWBinding to read from a data source or write to a data target.
Remarks

Use the AccessMode property to determine if Update reads or writes data.

When using the cwdsRead or cwdsWrite access mode, call the Update method when you want a read or write to occur. If the DataUpdated property is set to False, it is set to True after the update is completed, and the CWBindingDataUpdated event is generated. If you want to read or write data every time new data is available, use the ReadAutoUpdate or WriteAutoUpdate access modes.
Example

'Get new data
Binding.Update
See Also

AccessMode
CWBindingDataUpdated
DataUpdated
BackColor Property

Syntax

 CWButton.BackColor
Data Type

Color
Purpose

Specifies the background color of the button.
BackgroundImage Property

Syntax

Set CWButton.BackgroundImage
Data Type

CWPictureDisp
Purpose

Specifies the image to use for the background of the button.
Remarks

By default, the background image is blank.
Caption Property

Syntax

`CWButton.Caption`
Data Type

String
**Purpose**

Specifies the text that appears in the button.
Remarks

By default, the caption is empty.
Example

CWButton1.Caption = "Value"
See Also

Font
CaptionColor Property

Syntax

 CWButton.CaptionColor
Data Type

Color
**Purpose**

Specifies the color of the caption for the button.
See Also

Caption
CWBindings Property

Syntax
Set Object.CWBindings
Data Type

CWBindings
Applies To

- CWButton
- CWGraph
- CKnob
- CWNumEdit
- CWSlide
**Purpose**

Returns a collection of CWBinding objects.
Example

'Set the URL on the first CWBinding object
CWGraph1.CWBindings(1).URL = "opc:/National Instruments.OPCDemo/sine"
Enabled Property

Syntax

Object.Enabled
Data Type

Boolean
Applies To

CWButton
CWKnob
CWSlide
Purpose

Specifies if the control responds to user input.
Remarks

If you set the Enabled property to False, the control appears disabled.
Font Property

Syntax

CWButton.Font
Data Type
Purpose

Specifies the font for the caption and the two text fields in the button.
Example

CWButton1.Font.Bold = True
'Make CWButton1 have the same font as CWButton2
Set CWButton1.Font = CWButton2.Font
See Also

Caption
OnText
OffText
ImmediateUpdates Property

Syntax

`CWButton, ImmediateUpdates`
Data Type

Boolean
Purpose

Specifies if the control draws new data as soon as it is available, or if the form refreshes the control when it draws other controls.
**Remarks**

Set this property to True to guarantee that the control is redrawn every time a change is made.

Set this property to False to skip redrawing for other events. When this property is set to False, the container, such as a Visual Basic form, controls the update rate.
Example

'Redraws the control every time you make a change.
CWBbutton1.ImmediateUpdates = True
KeyboardMode Property

Syntax

`CWBouton, KeyboardMode`
Data Type

CWKeyboardModes

You can use the following constants with this data type:

- `cwKeyboardHandled`– Keystrokes are processed by the control.
- `cwKeyboardNone`– Keystrokes are ignored by the control.
Purpose

Specifies how the control handles keyboard input from the user.
Mode Property

Syntax

`$\text{CWButton}. \text{Mode}$`
Data Type

CWButtonModes

You can use the following constants with this data type:

- cwModeIndicator—The CWButton does not respond to user input. In this mode you can only change the value of the CWButton programmatically.
- cwModeSwitchUntilReleased—The value of the CWButton changes when you click the CWButton. The value of the CWButton changes back to its original state when you release the mouse button.
- cwModeSwitchWhenPressed—The value of the CWButton changes when you click the CWButton. The value remains unchanged until you click the CWButton again.
**Purpose**

Specifies how the button responds to user input.
Example

'Button acts only as indicator
'It does not respond to user interaction
CWBButton1.Mode = cwModeIndicator
OffColor Property

Syntax

 CWButton. OffColor
Data Type

Color
**Purpose**

Specifies the color of the button in the Off state.
See Also

OnColor
OffImage Property

Syntax

Set CWButton.OffImage
Data Type

CWPictureDisp
**Purpose**

Specifies the picture for the button in the Off state.
OffText Property

Syntax

CWButton, OffText
Data Type

String
Purpose

Specifies the text for the button in the Off state.
Remarks
The location and visibility of the text depend on the style of the CWButton.
See Also

OnText
Caption
Font
OffTextColor Property

Syntax

 CWButton. OffTextColor
Data Type

Color
Purpose

Specifies the color of the text for the button in the Off state.
See Also

OffText
OnColor Property

Syntax

CWButton. OnColor
Data Type

Color
Purpose

Specifies the color of the button in the On state.
See Also
OffColor
OnImage Property

Syntax
Set CWButton, OnImage
Data Type

CWPictureDisp
Purpose

Specifies the picture for the button in the On state.
OnText Property

Syntax

CWButton, OnText
Data Type

String
**Purpose**

Specifies the text for the button in the On state.
Remarks

The location and visibility of the text depend on the style of the CWButton.
See Also

OffText
Font
OnTextColor Property

Syntax

 CWButton. OnTextColor
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Color</th>
</tr>
</thead>
</table>

Purpose

Specifies the color of the text for the button in the On state.
See Also

OnText
ReadyState Property (Read Only)

Syntax

Object.ReadyState
Data Type

Long
Applies To

CWButton
CWGraph
CWKnob
CWNumbEdit
CWSlide
Purpose

Returns the ready state.
Remarks

The ready state provides a mechanism for interacting with a control before all its data has been loaded. For example, you can specify that a CWButton control use an external image file for its background image. When the CWButton control is loaded, the image might take a long time to load into the control.
See Also

ReadyStateChange
ShowFocusMode Property

Syntax

`CWButton.ShowFocusMode`
Data Type

CWShowFocusModes

You can use the following constants with this data type:

- cwShowFocusControl–Indicates graphically if the control has the focus.
- cwShowFocusNone–Does not indicate graphically if the control has the focus.
**Purpose**

Specifies how the control indicates it has the focus.
Value Property

Syntax

CWBButton, Value
Data Type

Boolean
**Purpose**

Specifies the current value of the button.
Example

CWBUTTON1.Value = True
' Read the current value of the button
AlarmState = CWAlarmLED.Value
Windowless Property

Syntax

`CWButton.Windowless`
Data Type

Boolean
**Purpose**

Specifies if the control has a window.
Example

'The control has a window
CWBButton1.Windowless = False
See Also

CWButton.BackColor
AboutBox Method

Syntax

`Object.AboutBox`
Applies To

CWButton
CWGraph
CWKnob
CWNumEdit
CWSlide
Purpose
Displays the About Box for the control.
Remarks

The About Box displays the version number of the controls and information about contacting National Instruments. You can also access the About Box from a separate tab on the property pages.
Example

'Display the graph's About Box
CWGraph1.AboutBox
ControlImage Method

Syntax

Object.ControlImage ()
Return Type

Picture

The image of the entire control.
Applies To

CWButton
CWGraph
CWKnob
CWNumEdit
CWSlide
Purpose

Returns an image of the entire control.
Remarks

You can use the image of the control for printing. The image returned is in the form of an enhanced metafile (EMF).
Example

' Set the image of the picturebox control in VB to
' An image of the entire graph
Set Picture1.Picture = CWGraph1.ControlImage
ExportStyle Method

Syntax

Object.ExportStyle FileName
Applies To

CWButton
CWGraph
CWKnob
CWNumEdit
CWSlide
Purpose
Exports the style of the Measurement Studio control to a file.
**Remarks**

A style is the complete current state of a control. For example, on a CWSlide control the current values of pointers, all colors, all fonts, and all imported images are examples of a current state that is saved.

Use ExportStyle to save the style of a control after you have configured it. For example, if you often create a new CWGraph control and set properties such as tick colors, axes ranges, and background colors, you can save the style of a graph once you have configured it. Then, when you want to create a graph of the same style, create a new graph, access the property pages, and import the exported style.

A style can be imported or exported from the property page or programmatically using ExportStyle.
Parameters

FileName As String

The name of the style file to save.
Example

'Export a style
CWGraph1.ExportStyle "c:\LogGraph.cwx"

'Import a style into graph2
CWGraph2.ImportStyle "c:\LinearGraph.cwx"
See Also

ImportStyle
**ImportStyle Method**

**Syntax**

```
Object.ImportStyle FileName
```
Applies To

CWButton
CWGraph
CWKnob
CWNumEdit
CWSlide
Purpose

Imports a previously exported style.
Remarks
By default, Measurement Studio style files have a .cwx extension. You can export styles from the property page, or programmatically using ExportStyle.
**Parameters**

**FileName As String**

The name of the style file to load.
**Example**

'Export a style
CWGraph1.ExportStyle "c:\LogGraph.cwx"

'Import a style into graph2
CWGraph2.ImportStyle "c:\LinearGraph.cwx"
See Also

ExportStyle
OffImages Method

Syntax

`CwButton, OffImages (Item)`
Return Type

 CWImage

The specified image.
Purpose

Provides access to the CWImage objects in the CWButton control in the Off state.
**Remarks**

Use the OffImages method to perform operations such as loading custom bitmaps and adjusting blink or animation speeds for the Off state. Use the OnImages method to access images in the On state.

For the CWButton control, the following images are available: "Caption", "Background", "On Text", "Off Text", "Image".
Parameters

**Item As Variant**

The string of the CWImage or the number of the CWImage (starting at 1).
Example

'Make the caption blink quickly when the button is On
CWButton1.OnImages("Caption").BlinkInterval = cwSpeedFast

'Make the caption blink quickly when the button is On
'Use the index of the image instead of the name
CWButton1.OnImages(1).BlinkInterval = cwSpeedFast
See Also

OnImages
OnImages Method

Syntax

 CWButton, OnImages (Item)
Return Type

 CWImage

The specified image.
Purpose

Provides access to the CWImage objects in the CWButton control in the On state.
Remarks

Use the OnImages method to perform operations like loading custom bitmaps and adjusting blink or animation speeds for the On state. Use the OffImages method to access images in the Off state.

For the CWButton control, the following images are available: "Caption", "Background", "On Text", "Off Text", "Image".
Parameters

Item As Variant

The string of the CWImage or the number of the CWImage (starting at 1).
Example

'Make the caption blink quickly when the button is On
CWBLLbutton1.OnImages("Caption").BlinkInterval = cwSpeedFast

'Make the caption blink quickly when the button is On
'Use the index of the image instead of the name
CWBLLbutton1.OnImages(1).BlinkInterval = cwSpeedFast
See Also

OffImages
Refresh Method

Syntax

CWButton, Refresh
Purpose

Redraws the CWButton control.
SetBuiltInStyle Method

Syntax

CWButton.SetBuiltInStyle Style
Purpose

Sets many properties of the control to represent the new style specified.
Parameters

Style As CWButtonStyles

The style of the button.
Example
CWButton1.SetBuiltinStyle cwButtonStyleVToggle
Click Event

Syntax

Sub ControlName_Click()
Applies To

- CWButton
- CWGraph
- CWKnob
- CWNumEdit
- CWSlide
**Purpose**
Generates when you click the mouse on the control.
Remarks

The DblClick event does not call this event. Double-clicking on a control generates the Click event for the first click and the DblClick event for the second click. For more information, refer to Visual Basic help.
See Also

DbtClick
MouseDown
MouseMove
MouseUp
CWBindingDataUpdated Event

Syntax

Sub ControlName_CWBindingDataUpdated( Index As Integer, Data As CWData, Ignore As Boolean)
Applies To

CWButton
CWGraph
CWKnob
CWNumEdit
CWSlide
Purpose
Generated when the binding data is updated.
Remarks

The control generates a CWBindingDataUpdated event when it receives updated data. The event passes a reference to the data, so you can modify it in the event. For example, you can change the value of the updated data or even ignore it.

Rather than waiting for this event, you can check for new data. To determine if data has been updated since it was last read, query the value of the CWBinding.DataUpdated property. When a new value is loaded, DataUpdated is set to True. After the DataUpdated property is queried, DataUpdated reverts to False.

Use the DataUpdated property to check for updates if you are not using events or if you have set the CWBinding.DataUpdatedEnabled property to False.
**Parameters**

**Index** As [Integer](#)
Index in the CWBinding collection of the binding that generated this event.

**Data** As [CWData](#)
CWData object containing new data. You can modify this parameter to change the new data.

**Ignore** As [Boolean](#)
Ignores new data when set to True.
Example

Private Sub CWGraph1_CWBindingDataUpdated(ByVal Index As Integer, ByVal Data As CWUIControlsLib.CWData, Ignore As Boolean)
    If (Data.Value < 0) Then
        'Don't plot negative values--ignore them
        Ignore = True
    Else
        'Scale each value by a factor of 10
        Data.Value = Data.Value * 10
    End If
End Sub
See Also

CWBindings
CWBinding
CWData
CWBinding.DataUpdatedEnabled
CWBinding.DataUpdated
CWBindingStatusUpdated Event

Syntax

Sub ControlName_CWBindingStatusUpdated( Index As Integer, Status As Long, Error As Long, Message As String)
Applies To

CWButton
CWGraph
CWKnob
CWNumEdit
CWSlide
**Purpose**

Generated when the status of the binding connection changes.
**Remarks**

The status changes as the connection is made and data is downloaded. The event also is generated if an error occurs.

The CWBindingStatusUpdated event occurs every time you try to connect to the data source specified by the URL. The event returns parameters for the status, a system error code, and a string describing the most recent progress or error. You can use these parameters to identify the cause of a problem or to report the progress. The CWBindingStatusUpdated event might be generated several times, depending on the data source to which you are trying to connect.
**Parameters**

**Index As** Integer

Index in the CWBinding collection of the binding that generated this event.

**Status As** Long

Status of the binding connection.

**Error As** Long

Error of the binding connection.

**Message As** String

Descriptive message of the connection status.
Example

Private Sub CWGraph1_CWBindingStatusUpdated(ByVal Index As Integer, ByVal Status As Long, ByVal Error As Long, ByVal Message As String)

    'Display the status of connection 1
    If Index = 1 Then
        Text1.Text = Message
    End If

End Sub
See Also

CWBindings
CWBinding
CWBinding.Status
CWBinding.LastError
CWBinding.LastMessage
DblClick Event

Syntax
Sub ControlName_DblClick()
Applies To

- CWButton
- CWGraph
- CWKnob
- CWNumEdit
- CWSlide
Purpose
Generates when you double-click the mouse on the control.
Remarks

Double-clicking on a control generates the Click event for the first click and the DblClick event for the second click. For more information, refer to Visual Basic help.
See Also

Click
MouseDown
MouseMove
MouseUp
KeyDown, KeyUp Events

Syntax

Sub ControlName_KeyDown( KeyCode As Integer, Shift As Integer)
Sub ControlName_KeyDown( KeyCode As Integer, Shift As Integer)
Applies To

CWButton
CWGraph
CWKnob
CWNumEdit
CWSlide
**Purpose**

KeyUp generates when you release a key while the control has the input focus.
KeyDown generates when you press a key while the control has the input focus.
Remarks

For more information, refer to Visual Basic help.
Parameters

KeyCode As Integer

A code representing the key that was pressed. This code represents the key on the keyboard that was pressed rather than the ASCII value of the key. For example, pressing the <A> key produces the value 65. Pressing <@> or <SHIFT-2> produces the code for "2".

Shift As Integer

Specifies the state of the <SHIFT> key, <ALT> key, and <CTRL> key. The Shift argument can be any combination of the following values: 1 for <SHIFT>, 2 for <CTRL>, or 4 for <ALT>. For example, if the <SHIFT> and <CTRL> keys are pressed, the value is 3 (1 + 2).
See Also

KeyPress
KeyPress Event

Syntax
Sub ControlName_KeyPress( KeyAscii As Integer)
Applies To

CWButton
CWGraph
CWKnob
CWNumEdit
CWSlide
**Purpose**

Generated when the control has focus and you press a key.
Remarks

For more information, refer to Visual Basic help.
Parameters

KeyAscii As Integer

The ASCII value of the key that was pressed. In Visual Basic, use the Chr() function to translate KeyAscii into a character.
See Also

KeyDown
KeyUp
MouseDown, MouseMove, MouseUp Events

Syntax

Sub ControlName_MouseDown( Button As Integer, Shift As Integer, x As OLE_XPOS_PIXELS, y As OLE_YPOS_PIXELS)

Sub ControlName_MouseMove( Button As Integer, Shift As Integer, x As OLE_XPOS_PIXELS, y As OLE_YPOS_PIXELS)

Sub ControlName_MouseUp( Button As Integer, Shift As Integer, x As OLE_XPOS_PIXELS, y As OLE_YPOS_PIXELS)
Applies To

CWButton
CWGraph
CWKnob
CWNumEdit
CWSlide
Purpose

MouseDown generates when you click the mouse on the control.
MouseMove generates when you move the mouse over the control.
MouseUp generates when you release the mouse on the control.
Remarks
For more information, refer to Visual Basic help.
Parameters

**Button As Integer**

Button which was pressed, released, or, in the case of a mouse move, which buttons are down. The Button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2).

**Shift As Integer**

State of the <SHIFT> key, <ALT> key, and <CTRL> key. The Shift argument can be any combination of the following values: 1 for <SHIFT>, 2 for <CTRL>, or 4 for <ALT>. For example, if the <SHIFT> and <CTRL> keys are pressed, the value is 3 (1 + 2).

**x As OLE_XPOS_PIXELS**

Specifies the X coordinate of the current location of the pointer.

**y As OLE_YPOS_PIXELS**

Specifies the Y coordinate of the current location of the pointer.
See Also

Click
DbiClick
ReadyStateChange Event

Syntax
Sub ControlName_ReadyStateChange()
Applies To

- CWButton
- CWGraph
- CWKnob
- CWNumEdit
- CWSlide
**Purpose**

Generated when the ready state changes.
**Remarks**

When this event is generated, use the ReadyState property to determine the control’s new ready state.
See Also

ReadyState
ValueChanged Event

Syntax

Sub ControlName_ValueChanged( Value As Boolean)
Applies To

CWButton
**Purpose**
Generates when the value of the button changes.
Remarks

This event is generated if the button is in switch mode (switch value when clicked on) or in command mode (switch value until released).
Parameters

Value As **Boolean**

Specifies the current value of the button.
Alignment Property

Syntax

\texttt{CWCaption.Alignment}
Data Type

CWAlignments

You can use the following constants with this data type:

- `cwuiBottomCenter`–Bottom-center align.
- `cwuiBottomLeft`–Bottom-left align.
- `cwuiBottomRight`–Bottom-right align.
- `cwuiCenter`–Center.
- `cwuiLeftJustify`–Left align.
- `cwuiRightJustify`–Right align.
- `cwuiTopCenter`–Top-center align.
- `cwuiTopLeft`–Top-left align.
- `cwuiTopRight`–Top-right align.
Purpose

Specifies the alignment of the caption relative to CWCaption.XCoordinate and CWCaption.YCoordinate properties.
Example

'Mark the plot with a bold-cross point at (10,5)
CWGraph1.Annotations.Add
CWGraph1.Annotations.Item(1).Shape.Type = cwShapePoint
CWGraph1.Annotations.Item(1).Shape.PointStyle = cwPointBoldCross
'Specify the coordinates of the point
CWGraph1.Annotations.Item(1).XCoordinates = Array(10)
CWGraph1.Annotations.Item(1).YCoordinates = Array(5)
'Add a caption with the text "Peak"
CWGraph1.Annotations.Item(1).Caption.Text = "Peak"
'Position the caption above the point at (10,6)
CWGraph1.Annotations.Item(1).Caption.XCoordinate = 10
CWGraph1.Annotations.Item(1).Caption.YCoordinate = 6
'Align the caption so that the bottom-right corner of the caption appears at (10,6)
CWGraph1.Annotations.Item(1).Caption.Alignment = cwuiBottomRight
See Also

CWCaption XCoordinate
CWCaption YCoordinate
Angle Property

Syntax

CWCaption.Angle
Data Type

Double
Purpose

Specifies the angle of the text.
**Remarks**

The angle is measured in degrees counterclockwise from the default horizontal text position.
Color Property

Syntax

CWCaption. Color
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Purpose

Specifies the color of the caption text.
Font Property

Syntax

CWCaption.Font
Data Type
**Purpose**

Specifies the font of the caption text.
Text Property

Syntax

CWCaption.Text
Data Type

String
Purpose

Specifies the text of the caption.
Example

'Mark the plot with a bold-cross point at (10,5)
CWGraph1.Annotations.Add
CWGraph1.Annotations.Item(1).Shape.Type = cwShapePoint
CWGraph1.Annotations.Item(1).Shape.PointStyle = cwPointBoldCross
'Specify the coordinates of the point
CWGraph1.Annotations.Item(1).XCoordinates = Array(10)
CWGraph1.Annotations.Item(1).YCoordinates = Array(5)
'Add a caption with the text "Peak"
CWGraph1.Annotations.Item(1).Caption.Text = "Peak"
'Position the caption above the point at (10,6)
CWGraph1.Annotations.Item(1).Caption.XCoordinate = 10
CWGraph1.Annotations.Item(1).Caption.YCoordinate = 6
'Align the caption so that the bottom-right corner of the caption appears at (10,6)
CWGraph1.Annotations.Item(1).Caption.Alignment = cwuiBottomRight
XCoordinate, YCoordinate Properties

Syntax

```java
CWCaption.XCoordinate
CWCaption.YCoordinate
```
Data Type

Double
Purpose

Specifies the X and Y coordinates that the caption appears at on the graph.
Remarks

The alignment of the caption text depends on the values you specify with the CWCaption.Alignment property.
**Example**

'Mark the plot with a bold-cross point at (10,5)
CWGraph1.Annotations.Add
CWGraph1.Annotations.Item(1).Shape.Type = cwShapePoint
CWGraph1.Annotations.Item(1).Shape.PointStyle = cwPointBoldCross
'Specify the coordinates of the point
CWGraph1.Annotations.Item(1).XCoordinates = Array(10)
CWGraph1.Annotations.Item(1).YCoordinates = Array(5)
'Add a caption with the text "Peak"
CWGraph1.Annotations.Item(1).Caption.Text = "Peak"
'Position the caption above the point at (10,6)
CWGraph1.Annotations.Item(1).Caption.XCoordinate = 10
CWGraph1.Annotations.Item(1).Caption.YCoordinate = 6
'Align the caption so that the bottom-right corner of the caption appears at (10,6)
CWGraph1.Annotations.Item(1).Caption.Alignment = cwuiBottomRight
See Also

CWCaption Alignment
SetCoordinates Method

Syntax

CWCaption.SetCoordinates XCoordinate, YCoordinate
Purpose

Sets the X and Y coordinates of the caption.
Parameters

**XCoordinate** As [Double](#)
The X coordinate of the caption.

**YCoordinate** As [Double](#)
The Y coordinate of the caption.
See Also

XCoordinate
YCoordinate
Color Property

Syntax

 CWCursor. Color
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
</tr>
</tbody>
</table>
Purpose

Specifies the color of the cursor crosshair and point.
CrosshairStyle Property

Syntax

`CWCursor.CrosshairStyle`
Data Type

CWCrosshairStyles

You can use the following constants with this data type:

- cwCrosshairMajorX–If the point style is cwPointNone, a solid line is drawn.
- cwCrosshairMajorXMajorY–The crosshair has a long horizontal line and a long vertical line.
- cwCrosshairMajorXMinorY–The crosshair has a long horizontal line and a short vertical line.
- cwCrosshairMajorY–If the point style is cwPointNone, a solid line is drawn.
- cwCrosshairMinorX–The crosshair is a short horizontal line.
- cwCrosshairMinorXMajorY–The crosshair is a short horizontal line and a long vertical line.
- cwCrosshairMinorXMinorY–The crosshair is a short horizontal line and a short vertical line.
- cwCrosshairMinorY–The crosshair is a short vertical line.
- cwCrosshairNone–There is no crosshair.
Purpose

Specifies the type of lines that identify the cursor position.
Remarks

The enumeration permits a single horizontal line or a vertical line that stretches the width or height of the plot, short lines that are drawn close to the cursor position, and combinations of the two.

If the PointStyle is cwPointNone and the CrosshairStyle is cwCrosshairMajorX or cwCrosshairMajorY, the cursor appears as a solid line.
Enabled Property

Syntax

[CWCursor].Enabled
Data Type

Boolean
**Purpose**

Specifies if the CWCursor object generates mouse events or if you can drag the cursor in cursor tracking mode.
Remarks

If a user clicks on a disabled cursor, the cursor does not respond. Other cursors, plots, or the plot area can still generate an event.
See Also

CWGraph.TrackMode
Name Property

Syntax

`CWCursor.Name`
Data Type

String
**Purpose**

Specifies the name of the cursor.
Remarks

Use the name for indexing the Cursors collection on a CWGraph. If more than one cursor has the same name, the first matching cursor is used.
See Also

CWGraph.Cursors
Plot Property

Syntax
Set CWCursor, Plot
Data Type

CWPlot
Purpose
Specifies the plot associated with the cursor.
Remarks

Set this property to a plot in the CWGraph.Plots collection to associate the CWCursor with a specific CWPlot.

Check the value of this property to determine where the user has snapped the cursor. If CWCursor.SnapMode is set to cwCSnapNearestPoint, the value of Plot can change when the cursor is repositioned. If CWCursor.SnapMode is set to cwCSnapPointsOnPlot, the value changes only when the Plot property is explicitly set, or the plot is removed from the collection. If CWCursor.SnapMode is set to cwCSnapFloating, the cursor is not associated with any plot.
See Also

SnapMode
PointIndex Property

Syntax

\texttt{CWCursor.\texttt{PointIndex}}
Data Type

Long
Purpose

Specifies the point associated with the cursor on the plot.
Remarks
Change this value to position the cursor to a specific point on a plot. If the index is invalid or there is no associated plot, the new value is retained, but the cursors are not repositioned. The new value is used the next time you set the Plot property.
See Also

SnapMode
Plot
PointStyle Property

Syntax

`CWCursor::PointStyle`
Data Type

CWPointStyles

You can use the following constants with this data type:

- cwPointAsterisk–Asterisk
- cwPointBoldCross–Bold cross
- cwPointBoldX–Bold X
- cwPointCross–Cross
- cwPointDottedEmptyCircle–Dotted empty circle
- cwPointDottedEmptySquare–Dotted empty square
- cwPointDottedSolidCircle–Dotted solid circle
- cwPointDottedSolidSquare–Dotted solid square
- cwPointEmptyCircle–Empty circle
- cwPointEmptyDiamond–Empty diamond
- cwPointEmptySquare–Empty square
- cwPointEmptySquareWithCross–Empty square with cross
- cwPointEmptySquareWithX–Empty square with X
- cwPointNone–None
- cwPointSimpleDot–Simple dot
- cwPointSmallCross–Small cross
- cwPointSmallEmptySquare–Small empty square
- cwPointSmallSolidSquare–Small solid square
- cwPointSmallX–Small X
- cwPointSolidCircle–Solid circle
- cwPointSolidDiamond–Solid diamond
- cwPointSolidSquare–Solid square
- cwPointX–X
Purpose

Specifies the cursor point style.
Remarks

The point is drawn where the crosshair lines meet.
SnapMode Property

Syntax

\texttt{CWCursor.SnapMode}
### Data Type

**CWCursorSnapModes**

You can use the following constants with this data type:

- **cwCSnapAnchoredToPoint**–Snaps the cursor or annotation to a specified point on a specific plot. The cursor or annotation cannot be moved.
- **cwCSnapFixed**–Sets the cursor or annotation position independent of any plot. The cursor or annotation cannot be moved.
- **cwCSnapFloating**–Sets the cursor or annotation independent of any plot. You can move the cursor or annotation anywhere in the graph area.
- **cwCSnapNearestPoint**–Snaps the cursor or annotation to the nearest point on any plot. You can move the cursor or annotation along any plot.
- **cwCSnapNearestYForFixedX**–For a fixed X location, the cursor's Y value is the Y value of the nearest point on the specified plot. For an annotation cwCSnapNearestYForFixedX is equivalent to cwCSnapPointsOnPlot. The cursor or annotation can be moved only along this plot.
- **cwCSnapPointsOnPlot**–Snaps the cursor or annotation to points on a specified plot. The cursor or annotation can be moved only along this plot.
**Purpose**

Specifies the coordinates available for cursors to align with on a plot.
Remarks

The SnapMode property controls the available coordinates when you change the cursor position by dragging with the mouse, setting the XPosition or YPosition properties, or using the SetPosition method. When you change the SnapMode value, the current cursor position changes according to the new settings.
Visible Property

Syntax

\texttt{CWCursor, Visible}
Data Type

Boolean
Purpose

Specifies if the cursor is visible or hidden.
Remarks

Hidden cursors do not generate events, even if they are enabled. However, they still apply the current SnapMode value as new data is plotted. When the cursors are shown, they might be in a new position. If you have several cursors, but only show a few at a time, setting the SnapMode property to cwSnapFree improves performance when plotting new data.
See Also

SnapMode
XPosition Property

Syntax

CWCursor.XPosition
Data Type

Variant
Purpose

Specifies the current x-axis position of the cursor. The SnapMode property can cause the value to be coerced to a point on one of the current plots.
Remarks

If you are setting both the X and Y positions of a cursor, you should use the SetPosition method.
Example

'Access the X position of the second cursor
x = CWGraph1.Cursors.Item(2).XPosition
See Also

CWCursor.SnapMode
CWCursor.SetPosition
YPosition Property

Syntax

 CWCursor, YPosition
Data Type

Variant
**Purpose**

Specifies the current y-axis position of the cursor. The SnapMode property can cause the value to be coerced to a point on one of the current plots.
Remarks

If you are setting both the X and Y positions of a cursor, you should use the SetPosition method.
Example

'Set the Y position of the first cursor
CWGraph1.Cursors.Item(1).YPosition = YLimit
See Also

SnapMode
SetPosition
SetPosition Method

Syntax

```cpp
CWCursor.SetPosition XPosition, YPosition
```
**Purpose**

Sets the x- and y-axis positions of the cursor at the same time.
Remarks

The SnapMode property affects the actual position of the cursor.
Parameters

XPosition As Variant
The X coordinate of the cursor.

YPosition As Variant
The Y coordinate of the cursor.
See Also

SnapMode
XPosition
YPosition
Value Property

Syntax

\texttt{CWData.Value}
Data Type

Variant
Purpose

Specifies the value of a CWDData object.
Remarks

If the CWData object is owned by a CWDataSocket and the CWDataSocket is connected to a data target in a write access mode, setting this property triggers the CWDataSocket.Update method. If you set this property (while connected in a write mode), the CWDataSocket.DataUpdated property is set to True.

If the CWData is owned by a CWDataSocket, reading this property sets the CWDataSocket.DataUpdated property to False.
Example

dim CWDData1 as new CWDData
dim v as variant
dim vector(100) as single
dim matrix(100,100) as long

' CWDData can hold many types of data
CWDData1.Value = 1
CWDData1.Value = "A String"
CWDData1.Value = vector
CWDData1.Value = matrix

' CWDData can be assigned to a varaint
v = CWDData1.Value

'Since value is the default property you can often
'Omit the "Value" keyword
CWDData1 = 10
v = CWDData1
CopyFrom Method

Syntax

\texttt{CWData.CopyFrom Source}
**Purpose**

Copies the value and attributes from another CWData object.
Remarks

If the destination CWData is associated with a CWDataSocket connection, copying to it triggers the CWDataSocket.Update method if it is connected with the AccessMode set to cwdsWriteAutoUpdate.
Parameters

Source As CWData

The CWData object to copy data from.
Example

'Copy the current value and attributes from a CWDataSocket's data

dim CWData1 as new CWData
CWData1.CopyFrom CWDataSocket1.Data

'Make some changes to the copy
CWData1.Value = 1
CWData1.SetAttribute("SampleRate", 10000)

'Copy data back to a DataSocket's data
CWDataSocket1.Data.CopyFrom CWData1
DeleteAttribute Method

Syntax

CWData.DeleteAttribute Name
Purpose
Deletes an existing attribute if it exists.
Remarks

If the CWData object is owned by a CWDataSocket and the CWDataSocket is connected to a data target in a write access mode, deleting an attribute triggers the CWDataSocket.Update method. If the CWData is owned by a CWDataSocket, deleting an attribute sets the CWDataSocket.DataUpdated property to True.
Parameters

Name As Variant

Name of the attribute to delete.
Example

'Delete an attribute
CWDatasocket1.Data.DeleteAttribute("SampleRate")
See Also

GetAttribute
HasAttribute
GetAttribute Method

Syntax

`CWData.GetAttribute (Name, Default)`
**Return Type**

`CWData`
**Purpose**

Gets a CWData object for an attribute
Remarks
If the attribute does not exist, the default value passed in is returned.
Parameters

**Name** As [Variant](#)
Name of the attribute to get.

**Default** As [Variant](#)
Default value to return if the attribute does not exist.
Example

'Get an attribute, or 0 if the attribute does not exist
Text1.Text = CWDataSocket1.Data.GetAttribute("SampleRate",0)
See Also

HasAttribute
GetAttributeNames
DeleteAttribute
setAttribute
GetAttributeNames Method

Syntax

`CWData.GetAttributeNames()`
Return Type

Variant

Attribute names.
Purpose

Returns the names of the attributes on the CWDData object.
Remarks

Returns a VARIANT that contains an array of strings. The array could be empty if there are no attributes on the CWDData object.
Example

'Print the names of the attributes
v = CWDataSocket1.Data.GetAttributeNames
for i = lbound(v) to ubound(v)
    text1.text = text1.text + " " + v(i)
next i
See Also

- GetAttribute
- HasAttribute
- DeleteAttribute
- SetAttribute
HasAttribute Method

Syntax

```
CwData.HasAttribute ( Name)
```
Return Type

Boolean

Returns True if an attribute exists.
Purpose

Returns True if an attribute exists.
Parameters

Name As Variant

Name of the attribute.
Example

'Check for an attribute
If CWData1.HasAttribute("SampleRate") Then
  Text1.Text = "Has attribute"
End If
See Also

DeleteAttribute
GetAttribute
GetAttributeNames
Reset Method

Syntax

\texttt{CWData.Reset}
**Purpose**

Clears value and all attributes.
Remarks

If the CWData object is owned by a CWDataSocket and the CWDataSocket is connected to a data target in a write access mode, resetting the CWData object triggers the CWDataSocket.Update method and sets the CWDataSocket.DataUpdated property to True.
Example

' Clear the current value and all attributes held by a CWDdataSocket
CWDdataSocket1.Data.Reset
SetAttribute Method

Syntax

```
CWData.SetAttribute Name, Value
```
Purpose
Sets the value of an attribute.
Remarks

If the CWData object is owned by a CWDataSocket and the CWDataSocket is connected to a data target in a write access mode, setting an attribute triggers the CWDataSocket.Update method and sets the CWDataSocket.DataUpdated property to True.
Parameters

**Name** As **Variant**
Name of the attribute to set.

**Value** As **Variant**
Value for the attribute.
Example

'Set the SampleRate Attribute to 1000
CWDatasocket1.Data.SetAttribute "SampleRate", 1000
Annotations Property (Read Only)

Syntax

`CWGraph.Annotations`
Data Type
CWAnotations
**Purpose**

Specifies a collection of CWAnnotation objects.
See Also

CWAnnotations
CWAnnotation
AnnotationTemplate Property (Read Only)

Syntax

CWGraph. AnnotationTemplate
Data Type

CWAAnnotation
**Purpose**

Returns the CWAnnotation object to use as a template for new annotations.
Remarks

You can use the Annotations property page to set properties on the template annotation. The default values for the template annotation are used as default property values for newly created annotations created with the CWAnnotations.Add method.
See Also

CWAnnotation
BackColor Property

Syntax

CWGraph.BackColor
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Color</th>
</tr>
</thead>
</table>

**Purpose**

Specifies the background color for the graph caption.
Caption Property

Syntax

\texttt{CWGraph, Caption}
Data Type

String
**Purpose**

Specifies the caption that appears on the graph.
Example

CWGraph1.Caption = "Temperature vs. Time"
See Also

Images

CaptionColor
CaptionColor Property

Syntax

`CWGraph. CaptionColor`
Data Type

Color
Purpose

Specifies the color of the caption.
See Also

Caption
ChartLength Property

Syntax

```
CWGraph.ChartLength
```
Data Type

Long
**Purpose**

Specifies how many points the graph stores when charting before it deletes old data.
**Remarks**

Set this property to 0 to store only enough points to fill the plot area.

If the AutoScale property is enabled on the x axis, the system does not recalculate the chart length.

Only the ChartY method supports automatic chart length calculation. The ChartXY and ChartXvsY methods do not.

If you change the x-axis range (the XInc parameter) between calls to ChartY, there might not be enough data to fill the plot area.
See Also

ChartY
CWPlot.ChartY
ChartStyle Property

Syntax

```cwgraph ChartStyle```
Data Type

You can use the following constants with this data type:

- cwChartScope—When the X values for new data reach the end of the display, the plot is cleared and the x axis updates according to its current range. For example, if the x-axis range is 0 to 100, it becomes 100 to 200. No old data is displayed.

- cwChartStrip—When a trace reaches the edge of the plot area, the x axis and plot start to scroll from right to left. New data points are appended on the right side while old data scrolls off the left side of the graph.
Purpose

Specifies how chart methods update the display as new data is added to the plot.
Remarks

Setting this property affects only the display. It does not affect how much data is stored.

If data is charted to plots at different rates, the plot that is furthest in front, that is, the plot that has the greatest X values, determines when and how much the x axis is updated. If the data is in decreasing order, the plot with the smallest X value is used.

Use this property only with the CWGraph.ChartY or CWPlot.ChartY method, and only when autoscaling is not enabled for the x axis.
See Also

ChartY
CWPlot.ChartY
ChartLength
CWAxis.AutoScale
Cursors Property

Syntax
Set CWGraph. Cursors
Data Type

CWCursors
Purpose

Specifies a collection of CWCursor objects.
See Also

CW Cursors
CW Cursor
DefaultPlotPerRow Property

Syntax

```
CWGraph.DefaultPlotPerRow
```
Data Type

Boolean
Purpose

Specifies the default value used by the Plot or Chart method if the optional bPlotPerRow or bChartPerRow parameter is omitted.
See Also

PlotY
PlotXY
ChartY
ChartXY
DefaultxFirst Property

Syntax

`CWGraph.DefaultxFirst`
Data Type

Double
**Purpose**

Specifies the default X value of the first point in the plot when using the PlotY method. Set this property only if the optional xFirst parameter is omitted for the PlotY method.
See Also

PlotY
DefaultxInc Property

Syntax

\texttt{CWGraph \textbackslash DefaultxInc}
Data Type

Double
**Purpose**

Specifies the default value for incrementing when using the PlotY or ChartY method. Set this property only if the optional xInc parameter is omitted for either method.
See Also

ChanY
PlotY
Enabled Property

Syntax

\texttt{CWGraph.\texttt{Enabled}}
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
</tr>
</tbody>
</table>
Purpose

Specifies if the graph generates any events.
Remarks

If you set the Enabled property to False, the control appears disabled.
Font Property

Syntax

`CAGraph.Font`
Data Type

Font
**Purpose**

Specifies the font for labels on all axes.
GraphFrameColor Property

Syntax

CWGraph.GraphFrameColor
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Color</th>
</tr>
</thead>
</table>

**Purpose**

Specifies the color for the graph frame.
GraphFrameImage Property

Syntax
Set CWGraph.GraphFrameImage
Data Type

CWPictureDisp
Purpose

Specifies the image for the background of the graph frame.
GraphFrameStyle Property

Syntax

CWGraph. GraphFrameStyle
Data Type

CWGraphFrameStyles

You can use the following constants with this data type:

- cwGraphFrame3D–Specifies a three-dimensional graph frame style.
- cwGraphFrameClassic–Specifies a classic graph frame style.
**Purpose**

Specifies the style of the graph frame.
Remarks

By default, the GraphFrameStyle property is set to cwGraphFrame3D. Previous versions of the 2D graph control default to cwGraphFrameClassic.
ImmediateUpdates Property

Syntax

CWGraph. ImmediateUpdates
Data Type

Boolean
**Purpose**

Specifies if the graph draws new data as soon as it is available, or if the form refreshes the graph when it draws other controls.
Remarks

Set this property to True to guarantee that the graph plots all available data.
Set this property to False to skip graph updates for other events. When this property is False, the container, such as a Visual Basic form, controls the update rate.
KeyboardMode Property

Syntax

CWGraph. KeyboardMode
Data Type

CWKeyboardModes

You can use the following constants with this data type:

- cwKeyboardHandled–Keystrokes are processed by the control.
- cwKeyboardNone–Keystrokes are ignored by the control.
Purpose

Specifies how the control handles keyboard input from the user.
PlotAreaColor Property

Syntax

`CWGraph.PlotAreaColor`
Data Type

Color
**Purpose**

Specifies the background color of the plot area.
PlotAreaImage Property

Syntax

Set CWGraph.PlotAreaImage
Data Type

CWPictureDisp
Purpose

Specifies the image used for the background of the plot area.
Remarks

By default, the image stretches to fit the entire plot area.
Plots Property (Read Only)

Syntax

`CWGraph.Plots`
| Data Type | CWPlots |
**Purpose**

Specifies a collection of CWPlots.
Remarks

If you pass data to the CWGraph object for more plots than there are CWPlot objects, the CWGraph control will dynamically create CWPlot objects to display the data.
See Also

CWPlot
PlotTemplate Property

Syntax

Set CWGraph. PlotTemplate
Data Type

CWPlot
**Purpose**

Returns the CWPlot object to use as a template for new plots.
Remarks

You can use the Plots property page to set properties on the template plot. The default values for the template plot are used as default property values for newly created plots created with the CWPlots.Add method, and the axes of the template plots determine the coordinates for plot area events.
Example

'Set a plot template property programmatically
CWGraph1.PlotTemplate.LineColor = vbRed
See Also

CWPlot
PlotAreaMouseDown
TrackMode Property

Syntax

`CWGraph, TrackMode`
Data Type

CWGraphTrackModes

You can use the following constants with this data type:

- cwGTrackAllEvents—Generates mouse events for all objects in the plot area.
- cwGTrackDragAnnotation—Positions annotations with the mouse.
- cwGTrackDragCursor—Positions cursors with the mouse.
- cwGTrackPanPlotAreaX—Sets the extent of the x axis by dragging the plot area with the mouse.
- cwGTrackPanPlotAreaXY—Sets the extent of the x and y axes by dragging the plot area with the mouse.
- cwGTrackPanPlotAreaY—Sets the extent of the y axis by dragging the plot area with the mouse.
- cwGTrackPlotAreaEvents—Generates mouse events for only the plot area.
- cwGTrackZoomRectX—Sets the extent of the x axis by selecting a region with the mouse.
- cwGTrackZoomRectXY—Sets the extent of the x and y axes by selecting a region with the mouse.
- cwGTrackZoomRectY—Sets the extent of the y axis by selecting a region with the mouse.
Purpose
Determines how the mouse interacts with the graph.
Remarks

If the CWGraph.Enabled property is set to False, all tracking and events are disabled. Use the TrackMode property to turn on zooming and panning for the graph.
Example

'Enable panning for the X and Y axes
CWGraph1.TrackMode = cwGTrackPanPlotAreaXY
Windowless Property

Syntax

\texttt{CWGraph. Windowless}
Data Type

Boolean
Purpose

Specifies if the control has a window.
Example

'The control has a window
CWGraph1.Windowless = False
See Also

CWGraph.BackColor
XYData Property

Syntax

CWGraph.XYData
Data Type

Variant
Purpose
Displays data from an external source in a plot on a graph as it would if you called the PlotXY method.
Remarks

Use this property only with a CWBinding object to link to and display an external data source. If you want to plot XY data from your application, use the PlotXY method.

Set the DefaultPlotPerRow property value to specify how you want the array of data plotted.
See Also

CWBinding
DefaultPlotPerRow
PlotXY
XYDataAppend Property

Syntax

CWGraph.XYDataAppend
Data Type

Variant
**Purpose**

Displays data from an external source in a chart as it would if you called the ChartXY method.
Remarks

Use this property only with a CWBinding object to link to and display an external data source. If you want to chart XY data from your application, use the ChartXY method.

Set the DefaultPlotPerRow property value to specify how you want the array of data plotted.
See Also

CWBinding
ChartXY
DefaultPlotPerRow
YData Property

Syntax

\texttt{CWGraph, YData}
| Data Type | Variant |
Purpose

Displays data from an external source in a plot on a graph as it would if you called the PlotY method.
Remarks

Use this property only with a CWBinding object to link to and display an external data source. If you want to plot data from your application, use the PlotY method.

Set the DefaultPlotPerRow, DefaultxFirst, and DefaultxInc property values to specify how you want the array of data plotted, the X value for the first point in each plot, and the amount to increment X in each plot.
See Also

CWBinding
DefaultPlotPerRow
DefaultxInc
DefaultsFirst
PlotY
YDataAppend Property

Syntax

`CWGraph. YDataAppend`
Data Type

Variant
**Purpose**

Displays data from an external source in a chart as it would if you called the ChartY method.
Remarks

Use this property only with a CWBinding object to link to and display an external data source. If you want to chart data from your application, use the ChartY method.

Set the DefaultPlotPerRow and DefaultxFirst property values to specify how you want the array of data plotted and the X value for the first point in each plot.
See Also

- CWBinding
- ChartY
- DefaultPlotPerRow
- DefaultsInc
ChartXvsY Method

Syntax

`CWGraph.ChartXvsY xData, yData [, bChartPerRow = True]`
**Purpose**

Charts a one- or two-dimensional array of Y data against a one-dimensional array of X data. This method is similar to the PlotYvsX method, except that the previously plotted data is not deleted until the amount stored for each chart is greater than the CWGraph.ChartLength property specifies.
Remarks

Use the Plot and Chart methods on a CWPlot object if you want to modify a single plot without affecting the other plots. Use the Plot and Chart methods on the CWGraph object if you want to modify all existing plots.
**Parameters**

**xData** As [Variant](#)
The X data.

**yData** As [Variant](#)
The Y data.

**bChartPerRow** As [Variant](#)
[Optional] If True, each row of the yData array is equivalent to one plot. If False, each column of the yData array is equivalent to one plot.
This parameter has a default value of True.
See Also

PlotXvsY
ChartLength
CWPlot.ChartY
CWPlot.PlotY
ChartXY Method

Syntax

CWGraph.ChartXY xyData [, bChartPerRow = True]
**Purpose**
Charts a two-dimensional array of data. This method is similar to the PlotXY method, except that the previously plotted data is not deleted until the amount stored for each chart is greater than the CWGraph.ChartLength property specifies.
Remarks

Use the Plot and Chart methods on a CWPlot object if you want to modify a single plot without affecting the other plots. Use the Plot and Chart methods on the CWGraph object if you want to modify all existing plots.
Parameters

**xyData** As **Variant**
The data to chart.

**bChartPerRow** As **Variant**
[Optional] If True, the first row in the data array contains the X data and subsequent rows contain plots of Y data. If False, the first column in the data array contains the X data and subsequent columns contain plots of Y data.
This parameter has a default value of True.
See Also

PlotXY
PlotY
ChartLength
CWPlot.ChartY
CWPlot.PlotY
ChartY Method

Syntax

`CWGraph.ChartY yData [, xInc = 1] [, bChartPerRow = True]`
**Purpose**

Charts Y data on one or more plots relative to the index of the data.
**Remarks**

The xInc parameter determines the X spacing between points passed to a plot. The X value of the first point is the last point's X value + xInc. If no data has been plotted, the first X value is 0. This method is similar to the PlotY method, except that the previously plotted data is not deleted until the amount stored for each chart is greater than the CWGraph.ChartLength property specifies.

Use the Plot and Chart methods on a CWPlot object if you want to modify a single plot without affecting the other plots. Use the Plot and Chart methods on the CWGraph object if you want to modify all existing plots.
Parameters

yData As Variant

The data to chart. yData can be a scalar value adding one point to the first plot, a one-dimensional array adding n points to the first plot or one point to n plots, or a two-dimensional array adding multiple points to multiple plots.

xInc As Variant

[Optional] The amount to increment X for points in each plot. The value can be positive or negative.

This parameter has a default value of 1.

bChartPerRow As Variant

[Optional] If True and the yData array is one-dimensional, all the values in the yData array are appended to a single plot; if False and the yData array is one-dimensional, each value in the yData array is appended to its own plot. If True and the yData array is two-dimensional, each row of the yData array is appended to its own plot; if False and the yData array is two-dimensional, each column of the yData array is appended to its own plot.

This parameter has a default value of True.
See Also

PlotY
ChartLength
CWPlot.ChartY
CWPlot.PlotY
ClearData Method

Syntax

\texttt{CWGraph.CClearData}
**Purpose**

Clears data in all plots.
See Also

CWPlot.ClearData
Images Method

Syntax

\texttt{CWGraph.Images (Item)}
Return Type

`CWImage`

The specified image.
Purpose

Provides access to the CWImage objects in the CWGraph control.
Remarks

Use the Images method to perform operations like loading custom bitmaps and adjusting blink or animation speeds.

For the CWGraph control, the following images are available: "Caption", "Graph Frame", "Plot Area".
Parameters

**Item** As [Variant](#)

The string of the CWImage or the index of the CWImage (starting at 1).
Example

'Make the graph's caption blink fast
CWGraph1.Images("Caption").BlinkInterval = cwSpeedFast

'Access the caption using its index rather than the
'string "Caption"
CWGraph1.Images(1).BlinkInterval = cwSpeedFast
PlotXvsY Method

Syntax

`CWGraph.PlotXvsY xData, yData [, bPlotPerRow = True]`
**Purpose**

Plots a 1D or 2D array of Y data against a 1D array of X data. Each row of Y values generates one plot.
Remarks

The PlotXvsY method uses plots from the CWGraph.Plots collection that are marked as multiplots. If there are not enough plots available, it uses temporary multiplots. These plots and their settings are lost if a subsequent plot operation requires fewer plots.
Parameters

**xData** As Variant
The X data.

**yData** As Variant
The Y data.

**bPlotPerRow** As Variant

[Optional] If True, each row of the yData array is equivalent to one plot; if False, each column of the yData array is equivalent to one plot.

This parameter has a default value of True.
See Also

CWPlot.PlotXvsY
PlotXY Method

Syntax

`CWGraph. PlotXY xyData [, bPlotPerRow = True]`
Purpose

Plots a 2D array of data. The first row is X values, and subsequent rows are Y values for each plot.
Remarks
The PlotXY method uses plots from the CWGraph.Plots collection that are marked as multiplots. If there are not enough plots available, it uses temporary multiplots.
Parameters

**xyData** As **Variant**
The data to plot.

**bPlotPerRow** As **Variant**
[Optional] If True, the first row in the data array contains the X data and subsequent rows contain plots of Y data. If False, the first column in the data array contains the X data and subsequent columns contain plots of Y data.
This parameter has a default value of True.
See Also

CWPlot PlotXY
PlotY Method

Syntax

`CWGraph.PlotY yData [, xFirst = 0] [, xInc = 1] [, bPlotPerRow = True]`
**Purpose**

Plots Y data evenly spaced on the x axis relative to the index in the array. Alternatively, you can use the xFirst and xInc parameters to specify the X value at the first data point and the incremental X value between data points.
Remarks

Pass in a 2D array to plot several rows of data at once.

The PlotY method uses plots from the CWGraph.Plots collection that are marked as multiplots. If there are not enough plots available, it uses temporary multiplots.
Parameters

**yData** As Variant
The data to plot. yData can be a one-dimensional array that updates the first plot on the graph or a two-dimensional array that updates the first n plots on the graph.

**xFirst** As Variant
[Optional] The X value for the first point in each plot.
This parameter has a default value of 0.

**xInc** As Variant
[Optional] The amount to increment X for points in each plot. The value can be positive or negative.
This parameter has a default value of 1.

**bPlotPerRow** As Variant
[Optional] If True, each row of the yData array is equivalent to one plot. If False, each column of the yData array is equivalent to one plot.
This parameter has a default value of True.
See Also

CWPlot PlotY
Refresh Method

Syntax

`CWGraph Refresh`
**Purpose**

Redraws the CWGraph control.
AnnotationChange Event

Syntax

Sub ControlName_AnnotationChange( AnnotationIndex As Long, AnnotationPart As CWAnnotationParts, bTracking As Boolean)
Applies To

CWGraph
**Purpose**
Generates when you reposition an annotation with the mouse.
Remarks

This event is generated only when the TrackMode property is set to cwGTrackDragAnnotation.
**Parameters**

**AnnotationIndex** As [Long](#)

The index of the annotation (one-based).

**AnnotationPart** As [CWAnnotationParts](#)

Part of the annotation that was moved.

**bTracking** As [Boolean](#)

True if the mouse button was pressed when the event occurred.
AnnotationMouseDown, AnnotationMouseMove, AnnotationMouseUp Events

Syntax

Sub ControlName_AnnotationMouseDown( Button As Integer, Shift As Integer, XPos As Double, YPos As Double, AnnotationIndex As Long, AnnotationPart As CWAnnotationParts)

Sub ControlName_AnnotationMouseMove( Button As Integer, Shift As Integer, XPos As Double, YPos As Double, AnnotationIndex As Long, AnnotationPart As CWAnnotationParts)

Sub ControlName_AnnotationMouseUp( Button As Integer, Shift As Integer, XPos As Double, YPos As Double, AnnotationIndex As Long, AnnotationPart As CWAnnotationParts)
Applies To

CWGraph
Purpose

AnnotationMouseDown is generated when you click the mouse on an annotation.

AnnotationMouseMove is generated when you move the mouse over an annotation.

AnnotationMouseUp is generated when you release the mouse over an annotation.
Remarks

These events generate only if CWGraph.TrackMode is set to cwGTrackAllEvents. If the TrackMode property is set to cwGTrackDragAnnotation, the AnnotationChange event is generated.

In Visual Basic, use the following constants with the Button and Shift parameters:

- vbLeftButton 1 Left button is pressed
- vbRightButton 2 Right button is pressed
- vbMiddleButton 4 Middle button is pressed
- vbShiftMask 1 <Shift> key is pressed
- vbCtrlMask 2 <Ctrl> key is pressed
- vbAltMask 4 <Alt> key is pressed
Parameters

**Button As Integer**

State of the mouse buttons. The Button argument can be any combination of the following values: 1 for the left button, 2 for the right button, or 4 for the middle button. For example, if both the left and right mouse buttons are pressed, the Button argument is 3 (1 + 2). In the AnnotationMouseUp event, the Button argument corresponds to the state of the buttons after the triggering button is released.

**Shift As Integer**

State of the <SHIFT>, <CTRL>, and <ALT> keys when the button specified in the Button argument is pressed, released, or moved. The Shift argument is a bit field with the least-significant bits corresponding to the <SHIFT> key (bit 0), the <CTRL> key (bit 1), and the <ALT> key (bit 2). Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed.

**XPos As Double**

X position of the mouse.

**YPos As Double**

Y position of the mouse.

**AnnotationIndex As Long**

Index of the annotation (zero-based).

**AnnotationPart As CWAnnotationParts**

Part of the annotation on which the event occurred.
See Also

Annotations
AnnotationChange
CWAAnnotationParts
CursorChange Event

Syntax

Sub ControlName_CursorChange( CursorIndex As Long, XPos As Variant, YPos As Variant, bTracking As Boolean)
Applies To

CWGraph
Purpose
Generated when you reposition a cursor with the mouse.
Remarks

This event is generated only when TrackMode is set to cwGTrackDragCursor.
Parameters

**CursorIndex** As *Long*
The index of the cursor (one-based).

**XPos** As *Variant*
The X position of the cursor.

**YPos** As *Variant*
The Y position of the cursor.

**bTracking** As *Boolean*
True if the mouse was pressed when the event occurred.
Example

Private Sub CWGraph1_CursorChange(CursorIndex As Long, XPos As Variant, 'Store the new X and Y position of the cursor
    xDisplay = XPos
    yDisplay = YPos
End Sub
CursorMouseDown, CursorMouseMove, CursorMouseUp Events

Syntax

Sub ControlName_CursorMouseDown( Button As Integer, Shift As Integer, XPos As Variant, YPos As Variant, CursorIndex As Integer, CursorPart As CWCursorParts)

Sub ControlName_CursorMouseMove( Button As Integer, Shift As Integer, XPos As Variant, YPos As Variant, CursorIndex As Integer, CursorPart As CWCursorParts)

Sub ControlName_CursorMouseUp( Button As Integer, Shift As Integer, XPos As Variant, YPos As Variant, CursorIndex As Integer, CursorPart As CWCursorParts)
Applies To

CWGraph
Purpose

CursorMouseDown is generated when you click the mouse on a cursor.
CursorMouseMove is generated when you move the mouse over a cursor.
CursorMouseUp is generated when you release the mouse over a cursor.
Remarks

These events generate only if CWGraph.TrackMode is set to cwGTrackAllEvents. If the TrackMode property is set to cwGTrackDragCursor, the CursorChange event is generated.

In Visual Basic, use the following constants with the Button and Shift parameters:

- `vbLeftButton` 1 Left button is pressed
- `vbRightButton` 2 Right button is pressed
- `vbMiddleButton` 4 Middle button is pressed
- `vbShiftMask` 1 `<Shift>` key is pressed
- `vbCtrlMask` 2 `<Ctrl>` key is pressed
- `vbAltMask` 4 `<Alt>` key is pressed
Parameters

**Button** As [Integer]

State of the mouse buttons. The Button argument can be any combination of the following values: 1 for the left button, 2 for the right button, or 4 for the middle button. For example, if both the left and right mouse buttons are pressed, the Button argument is 3 (1 + 2). In the CursorMouseUp event, the Button argument corresponds to the state of the buttons after the triggering button is released.

**Shift** As [Integer]

State of the `<SHIFT>`, `<CTRL>`, and `<ALT>` keys when the button specified in the Button argument is pressed, released, or moved. The Shift argument is a bit field with the least-significant bits corresponding to the `<SHIFT>` key (bit 0), the `<CTRL>` key (bit 1), and the `<ALT>` key (bit 2). Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed.

**XPos** As [Variant]

X position of the cursor.

**YPos** As [Variant]

Y position of the cursor.

**CursorIndex** As [Integer]

Index of the cursor (zero-based).

**CursorPart** As [CWCursorParts]

Part of the cursor on which the event occurred.
See Also

Cursors
CursorChange
CWCursorParts
PlotAreaMouseDown, PlotAreaMouseMove, PlotAreaMouseUp Events

Syntax

Sub ControlName_PlotAreaMouseDown( Button As Integer, Shift As Integer, XPos As Variant, YPos As Variant)
Sub ControlName_PlotAreaMouseMove( Button As Integer, Shift As Integer, XPos As Variant, YPos As Variant)
Sub ControlName_PlotAreaMouseUp( Button As Integer, Shift As Integer, XPos As Variant, YPos As Variant)
Applies To

CWGraph
Purpose

PlotAreaMouseDown generates when you click the mouse on the plot area.
PlotAreaMouseMove generates when you move the mouse over the plot area.
PlotAreaMouseUp generates when you release the mouse over the plot area.
Remarks

These events generate only if CWGraph.TrackMode is set to cwGTrackAllEvents or cwGTrackPlotAreaEvents.

If you have multiple y axes, the y axis for the template plot is used. The point returned is the point clicked on, not the nearest point on a plot.

In Visual Basic, use the following constants with the Button and Shift parameters:

vbLeftButton 1 Left button is pressed
vbRightButton 2 Right button is pressed
vbMiddleButton 4 Middle button is pressed
vbShiftMask 1 <Shift> key is pressed
vbCtrlMask 2 <Ctrl> key is pressed
vbAltMask 4 <Alt> key is pressed
Parameters

**Button As Integer**

State of the mouse buttons. The Button argument can be any combination of the following values: 1 for the left button, 2 for the right button, or 4 for the middle button. For example, if both the left and right mouse buttons are pressed, the Button argument is 3 (1 + 2). In the PlotAreaMouseUp event, the Button argument corresponds to the state of the buttons after the triggering button is released.

**Shift As Integer**

State of the `<SHIFT>`, `<CTRL>`, and `<ALT>` keys when the button specified in the Button argument is pressed, released, or moved. The Shift argument is a bit field with the least-significant bits corresponding to the `<SHIFT>` key (Bit 0), the `<CTRL>` key (Bit 1), and the `<ALT>` key (Bit 2). Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed.

**XPos As Variant**

The X coordinate as defined by the X axis of the template plot.

**YPos As Variant**

The Y coordinate as defined by the Y axis of the template plot.
See Also

PlotTemplate
PlotMouseDown, PlotMouseMove, PlotMouseUp
Events

Syntax

Sub ControlName_PlotMouseDown( Button As Integer, Shift As Integer, XData As Variant, YData As Variant, PlotIndex As Integer, PointIndex As Long)
Sub ControlName_PlotMouseMove( Button As Integer, Shift As Integer, XData As Variant, YData As Variant, PlotIndex As Integer, PointIndex As Long)
Sub ControlName_PlotMouseUp( Button As Integer, Shift As Integer, XData As Variant, YData As Variant, PlotIndex As Integer, PointIndex As Long)
Applies To

CWGraph
Purpose

PlotMouseDown generates when you click the mouse on a plot.
PlotMouseMove generates when you move the mouse over a plot.
PlotMouseUp generates when you release the mouse over a plot.
Remarks

These events generate only if CWGraph.TrackMode is set to cwGTrackAllEvents.

In Visual Basic, use the following constants with the Button and Shift parameters:

- `vbLeftButton` 1 Left button is pressed
- `vbRightButton` 2 Right button is pressed
- `vbMiddleButton` 4 Middle button is pressed
- `vbShiftMask` 1 `<Shift>` key is pressed
- `vbCtrlMask` 2 `<Ctrl>` key is pressed
- `vbAltMask` 4 `<Alt>` key is pressed
**Parameters**

**Button As Integer**
State of the mouse buttons. The Button argument can be any combination of the following values: 1 for the left button, 2 for the right button, or 4 for the middle button. For example, if both the left and right mouse buttons are pressed, the Button argument is 3 (1 + 2). In the PlotMouseUp event, the Button argument corresponds to the state of the buttons after the triggering button is released.

**Shift As Integer**
State of the <SHIFT>, <CTRL>, and <ALT> keys when the button specified in the Button argument is pressed, released, or moved. The Shift argument is a bit field with the least-significant bits corresponding to the <SHIFT> key (Bit 0), the <CTRL> key (Bit 1), and the <ALT> key (Bit 2). Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed.

**XData As Variant**
The X value of the point nearest to the mouse pointer when the event occurred.

**YData As Variant**
The Y value of the point nearest to the mouse pointer when the event occurred.

**PlotIndex As Integer**
The index of the plot (zero-based).

**PointIndex As Long**
The index of the point (zero-based).
See Also

Plots
AnimateColumns Property

Syntax

CwImage.AnimateColumns
| Data Type | Long |
Purpose

Specifies the number of columns in a bitmap that is being used for animation.
Remarks

The AnimateColumns and AnimateRows properties are used together to enable animation of bitmap images. Use AnimateColumns and AnimateRows to specify the number and layout of animation frames within a bitmap.

For example, setting AnimateColumns = 10 and AnimateRows = 1 indicates that the bitmap image you've selected must be divided into 10 columns. During animation, the control cycles through the portion of the image in each column.

This property does not apply to CWImage objects representing captions.
See Also

AnimateRows
AnimateInterval Property

Syntax

`CWImage.AnimationInterval`
**Data Type**

**CWSpeeds**

You can use the following constants with this data type:

- cwSpeedFast–600 ms
- cwSpeedFastest–150 ms
- cwSpeedMedium–900 ms
- cwSpeedOff–Off
- cwSpeedSlow–1200 ms
- cwSpeedSlowest–1800 ms
- cwSpeedVeryFast–300 ms
- cwSpeedVerySlow–1500 ms
Purpose

Specifies how often an image animates.
Remarks

This property does not apply to CWImage objects representing captions.
AnimateRows Property

Syntax

CWImage.AnimateRows
Data Type

Long
Purpose

Specifies the number of rows in a bitmap that is being used for animation.
Remarks

This property does not apply to CWImage objects representing captions.
See Also
AnimateColumns
BlinkInterval Property

Syntax

```
CWImage.BlinkInterval
```
Data Type

**CWSpeeds**

You can use the following constants with this data type:

- `cwSpeedFast`–600 ms
- `cwSpeedFastest`–150 ms
- `cwSpeedMedium`–900 ms
- `cwSpeedOff`–Off
- `cwSpeedSlow`–1200 ms
- `cwSpeedSlowest`–1800 ms
- `cwSpeedVeryFast`–300 ms
- `cwSpeedVerySlow`–1500 ms
Purpose

Specifies how often the image blinks.
Color Property

Syntax

CWImage. Color
Data Type

Color
**Purpose**

Specifies the color of the image, or, if you are doing dynamic color substitution in Windows metafiles, the color in the image that you want to replace.
Remarks

For dynamic color substitution in Windows metafiles, use with the Substitute, SubstituteColor, and Tolerance properties.
Example

'Set button's off image to red

'Dynamic color substitution of metafiles
'Load the metafile into the OffImage of the CWButton
Set CWButton1.OffImage.CWImage.Picture = LoadPicture("c:\pics\stop.wmf")
'Replace all red in image with blue
CWButton1.OffImage.CWImage.Substitute = True
CWButton1.OffImage.CWImage.Color = vbRed
CWButton1.OffImage.CWImage.SubstituteColor = vbBlue
'Substitute only very close hue matches
CWButton1.OffImage.CWImage.Tolerance = 10
See Also

- Substitute
- SubstituteColor
- Tolerance
FlipH Property

Syntax

CImage.FlipH
Data Type

Boolean
Purpose

Flips an image horizontally.
Remarks

This property works only for metafiles.
Example

'Flip button's off image horizontally and vertically
CWBButton1.OffImage.CWImage.FlipH = True
CWBButton1.OffImage.CWImage.FlipV = True
See Also
FlipV
FlipV Property

Syntax

CWImage.FlipV
| Data Type | Boolean |
Purpose

Flips an image vertically.
Remarks
This property works only for metafiles.
Example

'Flip button's off image horizontally and vertically
CWBButton1.OffImage.CWImage.FlipH = True
CWBButton1.OffImage.CWImage.FlipV = True
See Also

FlipH
Picture Property

Syntax

`<cwl:image>` Picture
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture</td>
<td></td>
</tr>
</tbody>
</table>
**Purpose**

Specifies the image used in the CWImage object.
Remarks

This property does not apply to CWImage objects representing captions.
Example

'Load a picture into the OffImage of the CWButton
Set CWButton1.OffImage.CWImage.Picture = LoadPicture("C:\WINNT\Maple Trails.bmp")
ReverseAnimation Property

Syntax

`CWImage.ReverseAnimation`
Data Type

Boolean
Purpose

Specifies the direction of animation.
Remarks

This property does not apply to CWImage objects representing captions.
See Also

AnimateColumns
AnimateRows
SaveLink Property

Syntax

 CWImage. SaveLink
Data Type

Boolean
Purpose

Specifies if the CWImage object saves the image itself or a link to the image.
Remarks

If you load an image into the control through the property page, there are two ways the control can save the image. By default, the control saves the entire image file as part of the Visual Basic form file. This is what the control does if SaveLink is set to False. Saving the image as part of the Visual Basic form file means that the form file might get large and contain duplicate images.

Alternatively, you can save a link (essentially a path) to the image file. When the control is loaded, it will load its images from the paths (URLs) given. The Visual Basic form file remains small, but the images must be available to load when an application is built.

This property does not apply to CWImage objects representing captions.
Example

'Set the URL to circles.bmp
CWButton1.OffImage.CWImage.URL = "c:\windows\circles.bmp"

'The Off Image must be loaded from the file each
time the control is loaded.
CWButton1.OffImage.CWImage.SaveLink = True

'Reload the image from the new URL
CWButton1.OffImage.CWImage.Reload
See Also

URL

Reload
Stretch Property

Syntax

*CImage. Stretch*
Data Type

Boolean
**Purpose**

Specifies if the image is displayed normal size or stretched to fit the size available within the control.
Remarks

This property only affects only bitmap and icon images. Metafile images stretch by default.

When Stretch is set to True, the image stretches to fit the bounding rectangle. If the bounding rectangle is smaller than the image, the image shrinks to the bounding rectangle regardless of the value of Stretch. If Stretch is set to False, the image continues to expand with the bounding rectangle until the bounding rectangle is larger than the image. If you use large images, consider resizing your image with a graphics application.

The Tile property takes precedence over the Stretch property. Thus, the Stretch property applies only when Tile is set to False.

This property does not apply to CWImage objects representing captions.
Example

'Turn stretching off for the CWButton's OffImage
CWButton1.OffImage.CWImage.Stretch = False
See Also

Tile
Substitute Property

Syntax

CWLmage. Substitute
Data Type

Boolean
Purpose

Specifies that you want to perform dynamic color substitution in the metafile image if set to True.
Remarks

You can perform dynamic color substitution only in Windows metafiles. Use the Color, SubstituteColor, and Tolerance properties to specify the color you want to replace, the replacement color, and the tolerance at which to match the color.
Example

'Load the metafile into the OffImage of the CWButton
Set CWButton1.OffImage.CWImage.Picture = LoadPicture("c:\pics\stop.wmf")

'Replace all red in image with blue
CWButton1.OffImage.CWImage.Substitute = True
CWButton1.OffImage.CWImage.Color = vbRed
CWButton1.OffImage.CWImage.SubstituteColor = vbBlue
'Substitute only very close hue matches
CWButton1.OffImage.CWImage.Tolerance = 10
See Also

Color
SubstituteColor
Tolerance
SubstituteColor Property

Syntax

CWImage.SubstituteColor
Data Type

Color
**Purpose**

Specifies the color that will replace a specified color in the metafile image.
Remarks

Use the Color property to specify the color in the metafile image that you want to replace. Use the SubstituteColor property to specify the color that will replace Color. Use the Tolerance property to specify the percentage at which you want the color matched.
Example

'Load the metafile into the OffImage of the CWButton
Set CWButton1.OffImage.CWImage.Picture = LoadPicture("c:\pics\stop.wmf")

'Replace all red in image with blue
CWButton1.OffImage.CWImage.Substitute = True
CWButton1.OffImage.CWImage.Color = vbRed
CWButton1.OffImage.CWImage.SubstituteColor = vbBlue
'Substitute only very close hue matches
CWButton1.OffImage.CWImage.Tolerance = 10
See Also

Substitute
Color
Tolerance
Tile Property

Syntax

CWImage. Tile
Data Type

Boolean
**Purpose**

Specifies if the image is tiled.
Remarks

Use the Tile property to repeat a small square image multiple times to fill a larger area. For the Tile property to work properly, the image size must be smaller than the size available for the image to draw within the control.

The Tile property takes precedence over the Stretch property.

This property does not apply to CWImage objects representing captions.
Example

'Tile the image in the plot area of the graph
CWGraph1.PlotAreaImage.CWImage.Tile = True
See Also

Stretch
Tolerance Property

Syntax

CWImage. Tolerance
Data Type

Long
**Purpose**

Specifies the percentage for matching color hues on the image to the SubstituteColor property.
**Remarks**

Use the Color property to specify the color in the metafile image that you want to replace. Then use the SubstituteColor property to specify the color that will replace Color. Use the Tolerance property to specify the percentage at which you want Color matched.

If you set the Tolerance to 100, then all colors on the image (that is, 100 %) are substituted. If you set this property to 0, then no color on the image (0 %) is substituted. Use a small tolerance to replace close matches to SubstituteColor.
Example

'Load the metafile into the OffImage of the CWButton
Set CWButton1.OffImage.CWImage.Picture = LoadPicture("c:\pics\stop.wmf")

'Replace all red in image with blue
CWButton1.OffImage.CWImage.Substitute = True
CWButton1.OffImage.CWImage.Color = vbRed
CWButton1.OffImage.CWImage.SubstituteColor = vbBlue
'Substitute only very close hue matches
CWButton1.OffImage.CWImage.Tolerance = 10
See Also

Substitute
Color
SubstituteColor
Transparent Property

Syntax

`CWImage. Transparent`
Data Type

Boolean
**Purpose**

Specifies if the image has a transparent color.
**Remarks**

You can use the Transparent and TransparentColor properties to create transparent regions within a bitmap, icon, or metafile. The TransparentColor property applies only when Transparent is set to True. When the control draws the image, it does not draw any pixels that are the color specified with the TransparentColor property. Thus, the graphics that are underneath the image show through.

When the control is printed, it ignores the Transparent property in some ActiveX control containers.

This property does not apply to CWImage objects representing captions.
Example

'Make the color black transparent in the image
CWBButton1.OffImage.CWImage.TransparentColor = vbBlack
CWBButton1.OffImage.CWImage.Transparent = True
See Also

TransparentColor
TransparentColor Property

Syntax

CWImage. TransparentColor
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Color</th>
</tr>
</thead>
</table>

**Purpose**

Specifies the color in the image that must be drawn as transparent.
Remarks

You can use the Transparent andTransparentColor properties to create transparent regions within a bitmap, icon, or metafile. The TransparentColor property applies only when Transparent is set to True. When the control draws the image, it does not draw any pixels that are the color specified with the TransparentColor property. Thus, the graphics that are underneath the image show through.

When the control is printed, it ignores the Transparent property in some ActiveX control containers.

This property does not apply to CWImage objects representing captions.
Example

'Make the color black transparent in the image
CWB1.OffImage.CWImage.TransparentColor = vbBlack
CWB1.OffImage.CWImage.Transparent = True
See Also

Transparent
URL Property

Syntax

 CWImage, URL
| Data Type | String |
**Purpose**

Specifies the path to the image.
Remarks

The path is in the form of a URL (Uniform Resource Locator). The simplest form of a URL string is a path and a filename. However, URLs can include files retrieved by FTP or HTTP over the internet or the World Wide Web.

Listed below are some example URLs:

- `c:\windows\circles.bmp`
- `http://www.ni.com/image.bmp`

This property does not apply to CWImage objects representing captions.
Example

'Set the URL to circles.bmp
CWButton1.OffImage.CWImage.URL = "c:\windows\circles.bmp"

'The Off Image must be loaded from the file each
time the control is loaded.
CWButton1.OffImage.CWImage.SaveLink = True

'Reload the image from the new URL
CWButton1.OffImage.CWImage.Reload
See Also

SaveLink
Reload
Visible Property

Syntax

`CWLimage. Visible`
Data Type

Boolean
**Purpose**

Specifies if the image is visible or not.
Example

'Hide the graph's caption.
CWGraph1.Images("Caption").Visible = False
Reload Method

Syntax

`CWImage.RLoad`
Purpose

Loads the image from the given URL.
Remarks
This method is valid only when the SaveLink property is set to True and a URL is specified with the URL property.
This property does not apply to CWImage objects representing captions.
Example

'Set the URL to circles.bmp
CWButton1.OffImage.CWImage.URL = "c:\windows\circles.bmp"

'The Off Image must be loaded from the file each
time the control is loaded.
CWButton1.OffImage.CWImage.SaveLink = True

'Reload the image from the new URL
CWButton1.OffImage.CWImage.Reloading
See Also

URL
SaveLink
ActivePointer Property

Syntax

Set Object. ActivePointer
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWPointer</td>
</tr>
</tbody>
</table>
Applies To
CWKnob
CWSlide
Purpose
Specifies the CWPointer object of the active pointer.
Remarks

Clicking on the slide sets the value of the active pointer to the value of the axis at the mouse click. Also, pressing the increment or decrement buttons moves the active pointer.

If there is only one pointer defined for the control, that pointer is always the active pointer.

You should usually set pointers with mode = cwuiPointerModeControl to be the active pointer.
Example

'Set the value of the active pointer
CWSlide1.ActivePointer.Value = 6

'Set pointer 2 to be the active pointer
Set CWSlide1.ActivePointer = CWSlide1.Pointers.Item(2)
See Also

Value
ValuePairIndex
ArcEnd Property

Syntax

\texttt{C WKneb\_ArcEnd}
Data Type

Double
Purpose

Specifies the end angle, in degrees, for the knob axis arc.
Remarks

The axis is drawn counter-clockwise from the start angle to the end angle.
Example
CWKnob1.ArcStart = 160
CWKnob1.ArcEnd = 200
See Also

ArcStart
ArcStart Property

Syntax

_CWKnob_. ArcStart
Data Type

Double
Purpose

Specifies the start angle, in degrees, for the knob axis arc.
Remarks
The axis is drawn counter-clockwise from the start angle to the end angle.
Example

CWKnob1.ArcStart = 160
CWKnob1.ArcEnd = 200
See Also

ArcEnd
Axis Property (Read Only)

Syntax

Object.Axis
Data Type
CWAxis
Applies To

CWKnob
CWSlide
**Purpose**

Returns the CWAxis object for this control.
Example
CWSlide1.Axis.Minimum = -10
BackColor Property

Syntax

Object.BackColor
Data Type

Color
Applies To

CWKnob
CWSlide
**Purpose**

Specifies the background color of the control.
Example

CWSlide1.BackColor = vbRed
BackgroundImage Property

Syntax
Set Object.BackgroundImage
| Data Type     | CWPictureDisp |
Applies To

CWKnob
CWSlide
**Purpose**

Specifies an image to be used for the background of the control.
Example

'Set the background image to be stretched
CWSlide1.BackgroundImage.CWImage.Stretch = True
Caption Property

Syntax

Object.Caption
Data Type

String
Applies To

CWKnob
CWSlide
**Purpose**

Specifies the text that appears in the control.
Example
CWSlide1.Caption = "Frequency"
CWSlide1.CaptionColor = vbRed
See Also

Font
CaptionColor
CaptionColor Property

Syntax

\texttt{Object.CaptionColor}
Data Type

Color
Applies To

CWKnob
CWSlide
Purpose

Specifies the color of the caption.
Example
CWSlide1.Caption = "Frequency"
CWSlide1.CaptionColor = vbRed
See Also

Caption
Font Property

Syntax

Object.Font
Data Type
Applies To

CWKnob
CWSlide
Purpose

Specifies the font for the caption and axis labels.
See Also

Caption
ForeColor Property

Syntax

*Object.ForeColor*
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Applies To

CWKnob
CWSlide
Purpose

Specifies the foreground color in the CWSlide or CWKnob control.
Example
CWSlide1.ForeColor = vbRed
ImmediateUpdates Property

Syntax

Object.ImmediateUpdates
Data Type

Boolean
Applies To

CWKnob
CWSlide
**Purpose**

Specifies if the control draws new data as soon as it is available or if the form refreshes the control when it draws other controls.
Remarks

Set this property to True to guarantee that the control is redrawn every time a change is made. Set this property to False to skip redrawing for other events. When this property is set to False, the container, such as a Visual Basic form, controls the update rate. You also can set this property to False to update several pointers without the control redrawing for each pointer update.
Example

CWSlide1.ImmediateUpdates = False

'Update two pointers
CWSlide1.Pointers(1) = Rnd
CWSlide1.Pointers(2) = Rnd

'The control will redraw to reflect the new pointers' positions
'when the system isn't busy
'Or, you can force the control to redraw
CWSlide1.ImmediateUpdates = True
IncDecValue Property

Syntax

Object.IncDecValue
Data Type

Variant
Applies To

CWKnob

CWSlide
**Purpose**

Specifies the amount that the control is changed when the user clicks the increment or decrement buttons, or uses the keyboard to increment or decrement the control.
Remarks

The CWKnob control does not have increment or decrement buttons.
See Also

KeyboardMode
KeyboardMode Property

Syntax

Object.KeyboardMode
Data Type

CWKeyboardModes

You can use the following constants with this data type:

- cwKeyboardHandled–Keystrokes are processed by the control.
- cwKeyboardNone–Keystrokes are ignored by the control.
Applies To

CWKnob
CWSlide
Purpose
Specifies how the control handles keyboard input from the user.
Pointers Property (Read Only)

Syntax

\textit{Object.Pointers}
Data Type

CWPointers
Applies To

CWKnob
CWSlide
Purpose

Returns a collection of Pointer objects.
Example

'Set the color of pointer 1 to yellow
CWSlide1.Pointers.Item(1).Color = vbYellow
ScaleStyleValuePairsOnly Property

Syntax

`Object.ScaleStyleValuePairsOnly`
Data Type

Boolean
Applies To

C WKnob
CWSlide
**Purpose**

Specifies if only the value pairs are displayed on the axis.
Remarks

If you set this property to true, the value pairs are displayed on the axis.
ShowFocusMode Property

Syntax

\textit{Object.ShowFocusMode}
**Data Type**

**CWShowFocusModes**

You can use the following constants with this data type:

- `cwShowFocusControl`–Indicates graphically if the control has the focus.
- `cwShowFocusNone`–Does not indicate graphically if the control has the focus.
Applies To

CWKnob
CWSlide
Purpose

Specifies how the control indicates that it has the focus.
Example

'Set the slider to show a focus rectangle
CWSlide1.ShowFocusMode = cwShowFocusControl
Statistics Property (Read Only)

Syntax

*Object.Statistics*
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWStatistics</td>
</tr>
</tbody>
</table>
Applies To

CWKnob
CWSlide
Purpose

Returns the Statistics object for this control.
Example

'Display the minimum value the slider has had
MsgBox CWSlide1.Statistics.Minimum
Value Property

Syntax

Object.Value
Data Type

Variant
Applies To

CWKnob
CWSlide
Purpose

Specifies the value of the active pointer.
Remarks

When the control is in ValuePairs Only mode, this property is the value of the value pair selected by the pointer. You must use the ValuePairIndex property to set the value.
Example

'Set the value of the active pointer on the slide
CWSlide1.Value = 5

'Set the value of the active pointer on the knob
CWKnob1.Value = 7

'Access the Value property of the active pointer
x = CWKnob1.Value
See Also

ActivePointer
ValuePairIndex
CWPointerValuePairIndex
ValuePairIndex Property

Syntax

ObjectValuePairIndex
Data Type

Long
Applies To

CWKnob
CWSlide
**Purpose**

Specifies the index of the value pair selected by the active pointer.
Remarks
This property is used only when the control is in ValuePairs Only mode.
When the control is in ValuePairs Only, a pointer can only have a value that is predefined by a value pair.
Example

'Select the second value pair
CWSlide1.ValuePairIndex = 2
See Also

ActivePointer
Value
CWPointer.ValuePairIndex
CWValuePairs
Windowless Property

Syntax

Object.Windowless
Data Type

Boolean
Applies To

CWKnob
CWSlide
**Purpose**

Specifies if the control has a window.
Example

'The control has a window
CWGraph1.Windowless = False
See Also

BackColor
Images Method

Syntax

Object.Images (Item)
**Return Type**

`CWImage`

The specified image.
Applies To

CWKnob
CWSlide
Purpose

Provides access to the CWImage objects in the CWKnob or CWSlide control.
Remarks

Use the Images method to perform operations like loading custom bitmaps and adjusting blink or animation speeds.

For the CWSlide control, the following images are available: "Caption", "Background", "Border", "Interior", "Frame", "Increment Button", "Decrement Button", "Pointer-1", and so on.

For the CWKnob control, the following images are available: "Caption", "Background", "Border", "Interior", "Frame", "Pointer-1", etc.
Parameters

Item As Variant

The string of the CWImage or the number of the CWImage (starting at 1).
Example

'Set the blink speed of the caption
CWSlide1.Images("Caption").BlinkInterval = cwSpeedFast

'Set the speed of the caption using an index instead
'Of the string "Caption"
CWSlide1.Images(1).BlinkInterval = cwSpeedFast
Refresh Method

Syntax

\texttt{CWKnob \textbf{Refresh}}
Purpose
Redraws the knob control.
SetBuiltInStyle Method

Syntax

`CWKnob.SetBuiltInStyle Style`
Purpose
Sets many properties of the control to represent the new style specified.
Parameters

Style As CWKnobStyles

The specified style of the knob.
Example

CWKnob1.SetBuiltInStyle cwKnobStyleDial
**PointerValueChanged Event**

**Syntax**

Sub *ControlName* _PointerValueChanged*( Pointer As Long, Value As Variant)*
Applies To

CWKnob
CWSlide
**Purpose**
Generated as the value of a pointer changes from the user interface or the program.
Remarks

Programmatic or graphical changes to the pointer cause this event. The CWSlide or CWKnob controls generate this event every time the value of a pointer changes while it is being set. For example, if you click and hold the mouse button in the middle of the control and drag the mouse, the value changes and you get a PointerValueChanged event. As you continue to drag and the value changes again, this event is generated again.
Parameters

**Pointer** As *Long*
The index of the pointer that changed.

**Value** As *Variant*
The new value of the pointer.
Example

'Digitally display the value of a slide in a CWNumEdit control
'When the value of the slide changes
Private Sub CWSlide1_PointerValueChanged(ByVal Pointer As Long, Value As Variant)
    CWNumEdit1.Value = CWSlide1.Value
End Sub
See Also

PointerValueCommitted
**PointerValueCommitted Event**

**Syntax**

Sub *ControlName_PointerValueCommitted*( Pointer As Long, Value As Variant)
Applies To

CWKnob
CWSlide
Purpose
Generated when the value of a pointer has stopped changing.
Remarks

The CWSlide or CWKnob control generates this event when the user releases the mouse button after clicking on the control, releases the keys used to change the value of the control, or sets the value programmatically. For some applications, it is better to wait for the PointerValueCommitted event rather than the PointerValueChanged event.
**Parameters**

**Pointer** As *Long*
The index of the pointer that changed.

**Value** As *Variant*
The new value of the pointer.
See Also

PointerValueChanged
Above Property

Syntax

\texttt{CWLabels, Above}
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
</tr>
</tbody>
</table>
**Purpose**

Specifies if labels appear above the object.
Remarks
This property is valid only when the axis is horizontal.
Example

'Turn on all axis labels on the graph
'Work on the x axis
CWGraph1.Axes.Item(1).Labels.Above = True
CWGraph1.Axes.Item(1).Labels.Below = True

'Work on the y axis
CWGraph1.Axes.Item(2).Labels.Left = True
CWGraph1.Axes.Item(2).Labels.Right = True
Below Property

Syntax

\texttt{CWLabels, \textit{Below}}
Data Type

Boolean
Purpose

Specifies if labels appear below the object.
Remarks
This property is valid only when the axis is horizontal.
Example

'Turn on all axis labels on the graph
'Work on the x axis
CWGraph1.Axes.Item(1).Labels.Above = True
CWGraph1.Axes.Item(1).Labels.Below = True

'Work on the y axis
CWGraph1.Axes.Item(2).Labels.Left = True
CWGraph1.Axes.Item(2).Labels.Right = True
Color Property

Syntax

\texttt{CWLabels, Color}
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Purpose
Specifies the color of the labels.
Example

'Make the x-axis labels red
CWGraph1.Axes.Item(1).Labels.Color = vbRed
Left Property

Syntax

\texttt{CWLabels, Left}
Data Type

Boolean
Purpose

Specifies if labels appear to the left of the object.
Remarks

This property is valid only when the axis is vertical.
Example

'Turn on all axis labels on the graph
'Work on the x axis
CWGraph1.Axes.Item(1).Labels.Above = True
CWGraph1.Axes.Item(1).Labels.Below = True

'Work on the y axis
CWGraph1.Axes.Item(2).Labels.Left = True
CWGraph1.Axes.Item(2).Labels.Right = True
Radial Property

Syntax

CWLabels, Radial
Data Type

Boolean
**Purpose**

Specifies if radial labels are drawn.
Remarks
This property is valid only when the axis is radial, such as a CWKnob.
Example

'Turn off radial labels
CWKnob1.Axis.Labels.Radial = False
See Also

CWKnob
Right Property

Syntax

CWLlabels. Right
Data Type

Boolean
Purpose

Specifies if labels appear to the right of the object.
Remarks

This property is valid only when the axis is vertical.
Example
'Turn on all axis labels on the graph
'Work on the x axis
CWGraph1.Axes.Item(1).Labels.Above = True
CWGraph1.Axes.Item(1).Labels.Below = True

'Work on the y axis
CWGraph1.Axes.Item(2).Labels.Left = True
CWGraph1.Axes.Item(2).Labels.Right = True
Width Property

Syntax

CWLabels. Width
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Long</th>
</tr>
</thead>
</table>

Purpose

Specifies the width of a label on the axis.
**Remarks**

Use the Width property to ensure that the axis is wide enough for a complete label. The Width property also determines the placement of automatic tick marks.

Set the Width to 0 for the axis to automatically set the width. Set the Width to a value greater than 0 to specify the number of digits to reserve on the axis for each label.

On the CWKnob control, the width controls the size of the knob by allowing more or less room for labels. Setting the width to 0 causes the CWKnob to select a default size independent of the labels.
Example

'Fix the width of the y axis at 5 characters
CWGraph1.Axes.Item(2).Labels.Width = 5

'Fix the size of the knob to account for labels of 3 characters
CWKnob1.Axis.Labels.Width = 3
See Also

CWTicks.AutoDivisions
AccelInc Property

Syntax

CWNumEdit.AccelInc
Data Type

Variant
Purpose

Determines the amount the value increments or decrements according to how long the user holds down the increment or decrement button.
Remarks

Use the AccelTime property to specify how many seconds the user must press down on the increment or decrement button before the CWNumEdit control uses the AccelInc value to increment or decrement the value.
Example

'Make the CWNumEdit control increment by 1 until the user
'Has held down the increment button for 5 seconds -
'Then increment by 10 at a time
CWNumEdit1.AccelInc = 1
CWNumEdit1.AccelInc = 10
CWNumEdit1.AccelTime = 5
See Also

IncDecValue
AccelTime
AccelTime Property

Syntax

CWNumEdit.AccelTime
Data Type

Long
**Purpose**

Specifies the number of seconds the increment or decrement button must be held down before the value is changed by the AccelInc value instead of the IncDecValue.
Example
'Make the CWNumEdit control increment by 1 until the user
'Has held down the increment button for 5 seconds -
'Then increment by 10 at a time
CWNumEdit1.AccelInc = 1
CWNumEdit1.AccelInc = 10
CWNumEdit1.AccelTime = 5
See Also

AccelInc
IncDecValue
Alignment Property

Syntax

CWNumEdit, Alignment
Data Type

CWAlignments

You can use the following constants with this data type:

- cwuiBottomCenter–Bottom-center align.
- cwuiBottomLeft–Bottom-left align.
- cwuiBottomRight–Bottom-right align.
- cwuiCenter–Center.
- cwuiLeftJustify–Left align.
- cwuiRightJustify–Right align.
- cwuiTopCenter–Top-center align.
- cwuiTopLeft–Top-left align.
- cwuiTopRight–Top-right align.
**Purpose**

Specifies how the text within the CWNumEdit control is aligned.
Remarks
Currently, you can only use the cwuiCenter, cwuiLeftJustify, and cwuiRightJustify constants with the CWNumEdit.Alignment property.
Example

'Center the text in the CWNNumEdit control
CWNNumEdit1.TextAlignment = cwuiCenter
Appearance Property

Syntax

CWNumEdit: Appearance
Data Type

CWAppearances

You can use the following constants with this data type:

- cwuiAppearanceFlat–Specifies a flat (two dimensional) control style.
- cwuiAppearanceThreeD–Specifies a three-dimensional control style.
**Purpose**

Specifies how the edit box portion of the CWNumEdit control is drawn.
BackColor Property

Syntax

CWNumEdit. BackColor
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
</tr>
</tbody>
</table>
Purpose

Specifies the frame color and the background color behind the increment and decrement buttons for the CWNumEdit control.
Remarks

To use the BackColor property, you must set the ButtonStyle property to cwuiNumEditButtonStyle3D and the BorderStyle property to cwuiBorderStyle3D. Setting the BackColor property has no effect on CWNumEdit controls that use the classic control style.
See Also

BorderStyle

ButtonStyle
BackColorText Property

Syntax

CWinEdit. BackColorText
Data Type

Color
Purpose

 Specifies the background color of the edit box portion of the CWNumEdit control.
BorderStyle Property

Syntax

CWndNumEdt, BorderStyle
Data Type

You can use the following constants with this data type:

- `cwuiBorderStyle3D`– Specifies a three-dimensional border.
- `cwuiBorderStyleFixedSingle`– Specifies a fixed-single border.
- `cwuiBorderStyleNone`– Specifies no border.
**Purpose**

Specifies the border around the edit box portion of the CWNumEdit control.
See Also

ButtonStyle
ButtonColor Property

Syntax

CWNumEdit.ButtonColor
Data Type

Color
**Purpose**

Specifies the color of the increment and decrement buttons.
**Remarks**

To use the ButtonColor property, you must set the ButtonStyle property to cwuiNumEditButtonStyle3D. This property cannot be used with CWNumEdit controls that use the classic control style.
See Also

ButtonStyle
ButtonStyle Property

Syntax

`CWNumEdit.ButtonStyle`
Data Type

CWNumEditButtonStyles

You can use the following constants with this data type:

- cwuiNumEditButtonStyle3D–Specifies a three-dimensional button style.
- cwuiNumEditButtonStyleClassic–Specifies a classic button style.
Purpose

Specifies if the increment and decrement buttons have a three-dimensional style.
Remarks
By default, the ButtonStyle property is set to cwuiNumEditButtonStyle3D. Previous versions of the numeric edit control default to cwuiNumEditButtonStyleClassic.
See Also

BorderStyle
Discrete Property

Syntax

\texttt{CWNNumEdit, \textit{Discrete}}
| Data Type | Boolean |
**Purpose**

Specifies if the CWNumEdit control is in discrete or continuous mode.
Remarks

When the CWNumEdit control is in discrete mode, the valid values are limited. The value of the control is always assigned to the closest discrete value. For example, you can set the CWNumEdit control to allow only integers, where the valid values are -1, 0, 1, 2, 3, and so on, by using a discrete axis.
Example

'Setup the CWNumEdit control to take only integer data.
CWNumEdit1.Discrete = True
CWNumEdit1.DiscreteInterval = 1
CWNumEdit1.DiscreteBase = 1

'Setup the CWNumEdit control to round all data to the nearest .5
CWNumEdit1.Discrete = True
CWNumEdit1.DiscreteInterval = .5
CWNumEdit1.DiscreteBase = 0
See Also

DiscreteBase
DiscreteInterval
DiscreteBase Property

Syntax

CWNumEdit, DiscreteBase
Data Type

Variant
Purpose

Specifies the initial value to use in a discrete CWNumEdit control.
Remarks

This property is used only when CWNumEdit.Discrete is set to True.
Example

'Setup the CWNumEdit control to take only integer data.
CWNumEdit1.Discrete = True
CWNumEdit1.DiscreteInterval = 1
CWNumEdit1.DiscreteBase = 1

'Setup the CWNumEdit control to round all data to the nearest .5
CWNumEdit1.Discrete = True
CWNumEdit1.DiscreteInterval = .5
CWNumEdit1.DiscreteBase = 0
See Also

Discrete
DiscreteInterval Property

Syntax

CWNumEdit: DiscreteInterval
Data Type

Variant
**Purpose**

Specifies the interval to use in a discrete CWNumEdit control.
Remarks
This property is used only when CWNumEdit.Discrete is set to True.
Example

'Setup the CWNumEdit control to take only integer data.
CWNumEdit1.Discrete = True
CWNumEdit1.DiscreteInterval = 1
CWNumEdit1.DiscreteBase = 1

'Setup the CWNumEdit control to round all data to the nearest .5
CWNumEdit1.Discrete = True
CWNumEdit1.DiscreteInterval = .5
CWNumEdit1.DiscreteBase = 0
See Also

Discrete
Enabled Property

Syntax

`CWNumEdit, Enabled`
Data Type

Boolean
**Purpose**

Specifies if the user is able to interact with the control.
Remarks

If you set the Enabled property to False, the control appears disabled.
Font Property

Syntax

`CWNumEdit.Font`
Data Type
Purpose
Specifies the font the CWNumEdit control uses.
ForeColorText Property

Syntax

CWNumEdit.ForeColorText
Data Type

Color
Purpose

Specifies the color of the text in the CWNumEdit control.
FormatString Property

Syntax

`CWNumEdit.FormatString`
Data Type

String
Purpose

Specifies the string that formats the value in the numeric edit control.
Remarks

The format string controls how the value of the CWNumEdit control is displayed. For example, you can use a format string to specify digits of precision, use exponential notation, or display time or date values.

Refer to CWAxis.FormatString for complete documentation of the FormatString property.
Example

'Make the CWNumEdit control display up to 3 digits to the right of the decimal
CWNumEdit1.FormatString = ".###"

'Use date formatting and set the value
CWNumEdit1.FormatString = "mm/dd/yyyy"
CWNumEdit1.Value = "1/8/1973"

'Use time formatting and set the value
CWNumEdit1.FormatString = "hh:nn:ss"
CWNumEdit1.Value = "10:30 pm"
See Also

CWAxis.FormatString

Text
IncDecButtonPosition Property

Syntax

 CWNumEdit, IncDecButtonPosition
Data Type

**CWPositions**

You can use the following constants with this data type:

- `cwuiBottom`–Bottom
- `cwuiLeft`–Left
- `cwuiRight`–Right
- `cwuiTop`–Top
**Purpose**

Specifies the location of the increment and decrement buttons of the CWNumEdit control.
See Also

IncDecButtonVisible
IncDecButtonVisible Property

Syntax

```
CWNumEdit.IncDecButtonVisible
```
Data Type

Boolean
Purpose

Specifies if the increment and decrement buttons are visible.
See Also

IncDecButtonPosition
IncDecValue Property

Syntax

`CWndEdit.IncDecValue`
Data Type

Variant
**Purpose**

Specifies the amount the value increments or decrements when the user clicks the increment or decrement button.
Example

'Make the CWNumEdit control increment by 1 until the user
'Has held down the increment button for 5 seconds -
'Then increment by 10 at a time
CWNumEdit1.AccelInc = 1
CWNumEdit1.AccelInc = 10
CWNumEdit1.AccelTime = 5
See Also

AccelInc
AccelTime
Maximum Property

Syntax

CWNumEdit Maximum
Data Type

Variant
**Purpose**

Specifies the maximum value of the CWNumEdit control for range checking.
Example

'Set the minimum to 0 and the maximum to 100
CWNumEdit1.Minimum = 0
CWNumEdit1.Maximum = 100
See Also

RangeChecking
Minimum
SetMinMax
Minimum Property

Syntax

CWNumEdit, Minimum
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variant</td>
</tr>
</tbody>
</table>
Purpose

Specifies the minimum value of the CWNumEdit control for range checking.
Example

'Set the minimum to 0 and the maximum to 100
CWNumEdit1.Minimum = 0
CWNumEdit1.Maximum = 100
See Also

RangeChecking
Maximum
SetMinMax
Mode Property

Syntax

CWinNumEdit. Mode
Data Type

You can use the following constants with this data type:

- cwEdModeControl–The CWNumEdit control responds to user input.
- cwEdModeIndicator–The CWNumEdit control does not respond to user input. In this mode you can only change the value of the CWNumEdit control programmatically.
Purpose

Specifies how the CWNumEdit control responds to user input.
Example

'Set the CWNnumEdit control to be read-only to the user
CWNNumEdit1.Mode = cwEdModeIndicator
RangeChecking Property

Syntax

CWNumEdit.RangeChecking
Data Type

Boolean
Purpose

Specifies if the CWNumEdit control enforces the Minimum and Maximum property values.
Example

'Range checking enforced
CWNumEdit1.RangeChecking = True
See Also

Minimum
Maximum
Text Property (Read Only)

Syntax

CWNumEdit.Text
Data Type

String
Purpose

Specifies the text that is displayed in the edit box.
Remarks

Use the Text property to get the formatted value currently in the edit box. The FormatString property value affects how the numeric edit control displays its value.
See Also

Value

FormatString
Value Property

Syntax

\texttt{CWNumEdit.Value}
Data Type

Variant
Purpose

Specifies the current value of the numeric edit control.
Remarks

The value might be different than what is displayed in the numeric edit control. Use the Text property to retrieve the exact string shown in the numeric edit control.
Example

'Set the numedit value to 5
CWNumEdit1.Value = 5.0

'Retrieve the value of the numedit control
x = CWNumEdit1.Value
See Also

Text
SetMinMax Method

Syntax

\texttt{CWNumEdit.SetMinMax Minimum, Maximum}
Purpose

Sets the Minimum and Maximum properties.
Remarks

Use the Minimum and Maximum properties to implement range checking on the value of the CWNumEdit control. If the RangeChecking property is set to True, the CWNumEdit control prevents the user from entering a value outside the range specified by the minimum and maximum values. When RangeChecking is set to False, the ValueChanging and ValueChanged events both pass back the OutOfRange parameter, which returns True if the new value of the control is out of range.
Parameters

**Minimum** As [Variant](#)

The minimum range checking value.

**Maximum** As [Variant](#)

The maximum range checking value.
Example

'Set the min to 0 and the max to 100
CWNumEdit1.SetMinMax 0, 100
See Also

RangeChecking
Minimum
Maximum
ValueChanged
ValueChanged
**Error Event**

**Syntax**

Sub *ControlName* _Error( Number As Integer, Description As String, Scode As Long, Source As String, HelpFile As String, HelpContext As Long, CancelDisplay As Boolean)
Applies To

CWNNumEdit
**Purpose**
Generated when the user enters an illegal value in the CWNumEdit control.
Remarks

The CWNumEdit control generates this event when the user types in an illegal value.
**Parameters**

**Number** As *Integer*

The lower 16-bits of the Scode value.

**Description** As *String*

Error description returned by the control. By default, this parameter returns one of three strings: "Type Mismatch", "Out of Present Range", or "Invalid Input". You can override these values by providing your error description inside the Error event handler. If CancelDisplay is set to True, no error messages are displayed.

**Scode** As *Long*

The Windows error code number.

**Source** As *String*

Application name.

**HelpFile** As *String*

Help file name.

**HelpContext** As *Long*

Context ID for the help file.

**CancelDisplay** As *Boolean*

Displays error descriptions in a message box when set to False (default value). If set to True, the control does not display the Description parameter default values or any values you specify for Description.
IncDecButtonClick Event

Syntax
Sub ControlName_IncDecButtonClick( IncButton As Boolean)
Applies To

CWNumEdit
**Purpose**

Generated when the user clicks the increment or decrement button on the numeric edit control.
**Parameters**

**IncButton** As [Boolean](#)

True if the increment button was pressed, or false if the decrement button was pressed.
See Also

ValueChanged
ValueChanged Event

Syntax

Sub ControlName_ValueChanged( Value As Variant, PreviousValue As Variant, OutOfRange As Boolean)
Applies To

CWNumEdit
Purpose

Generated when the value of the CWNumEdit control has changed from the user interface or the program.
Parameters

**Value** As [Variant](#)
Specifies the new value of the control.

**PreviousValue** As [Variant](#)
Specifies the value of the CWNumEdit control before the value was changed.

**OutOfRange** As [Boolean](#)
True if Value is not within the range specified by the Minimum and Maximum properties.
See Also

ValueChanging
SetMinMax
ValueChanging Event

Syntax

Sub ControlName_ValueChanged( NewValue As Variant, AttemptedValue As Variant, PreviousValue As Variant, OutOfRange As Boolean)
Applies To

CWNumEdit
**Purpose**
Generated when the value of the CWNumEdit control is about to change (the value of the control has been entered but not set).
Remarks
Modify the NewValue parameter within your event handler to change the value that will be set in the numeric edit control.
Parameters

**NewValue** As [Variant]

Specifies the new value the control will have. If you alter the NewValue parameter within your event code, the CWNumEdit control will take on the altered NewValue.

**AttemptedValue** As [Variant]

Specifies the value the user tried to set into the control. This value is often the same as NewValue and is most useful for custom range checking or validation.

**PreviousValue** As [Variant]

Specifies the value of the CWNumEdit control before the value changed.

**OutOfRange** As [Boolean]

True if NewValue is not within the range specified by the Minimum and Maximum properties.
Example

Sub CWNumEdit1_ValueChanged(NewValue as Variant, ByVal AttemptedValue as Variant, ByVal PreviousValue as Variant, ByValOutOfRange as Boolean)
    'If the new value is outside the range 0-100
    If NewValue < 0 or NewValue > 100 Then
        'Set the value to 50
        NewValue = 50
    End If
End Sub
See Also

ValueChanged
SetMinMax
CWImage Property (Read Only)

Syntax

```text
CWImage
```
Data Type

CWImage
**Purpose**

Returns the CWImage object that allows advanced image operations.
Example

'Turn stretching on for the CWButton's OffImage
CWBotton1.OffImage.CWImage.Stretch = True
Handle Property

Syntax

\texttt{CWPictureDisp.Handle}
Data Type
Long
**Purpose**

Returns a handle to the graphic contained within a Picture object.
**Remarks**

The Handle property is useful when you need to pass a handle to a graphic as part of a call to a function in a dynamic link library (DLL) or the Windows API.
Height Property

Syntax

```
CWPictureDisp.Height
```
Data Type

Long
Purpose
Specifies the height of the picture in HiMetric units.
See Also

Width
hPal Property

Syntax

`CWPictureBoxDisp.hPal`
Data Type

Long
**Purpose**

Specifies a handle to the palette of the picture.
Remarks

The hPal property is useful when you need to pass a handle to a palette as part of a call to a function in a dynamic link library (DLL) or the Windows API.
Type Property

Syntax

`CWPictureDisp.Type`
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long</td>
</tr>
</tbody>
</table>
Purpose

Returns the graphic format of a picture.
Remarks

The type can be one of the following values:

0 - Picture is empty.
1 - Bitmap
2 - Metafile
3 - Icon
4 - Enhanced Metafile
Width Property

Syntax

```
CWPictureDisp. Width
```
Data Type

Long
Purpose

Specifies the width of the picture in HiMetric units.
See Also

Height
AutoScale Property

Syntax

`CWPlot.AutoScale`
Data Type

Boolean
Purpose

Specifies if the extent of the data in the plot affect the extent of an autoscaling axis.
Remarks

To make the graph autoscale to a plot, set the CWAxis.AutoScale property to True.
See Also

CWAxis.AutoScale
BasePlot Property

Syntax

Set `CWPlot.BasePlot`
Data Type

CWPlot
Purpose

Specifies a plot to use for Y values when either the FillToBase or LineToBase property is enabled.
Remarks

Points in the plot are paired with points in the base plot. Filling or drawing lines between plots works best when the X coordinates for each pair are the same.
Example

'Fills the space between the first and second plot on the graph with red
CWGraph1.Plots.Item(1).FillToBase = True
Set CWGraph1.Plots.Item(1).BasePlot = CWGraph1.Plots.Item(2)
CWGraph1.Plots.Item(1).FillColor = vbRed
See Also

Base Value
FillToBase
LineToBase
BaseValue Property

Syntax

`CWPlot.BaseValue`
Data Type

Variant
Purpose

Specifies the Y value used to fill the space between the base plot and the x axis when either the FillToBase or LineToBase property is enabled.
Remarks

The value of this property is ignored if the BasePlot property is set.
See Also

- BasePlot
- FillToBase
- LineToBase
DefaultPlotPerRow Property

Syntax

CWPLOT.DefaultPlotPerRow
Data Type

Boolean
Purpose

Specifies the default value used by the PlotXY and ChartXY methods if the optional bXInFirstRow parameter is omitted.
See Also

PlotXY
ChartXY
Default x First Property

Syntax

\texttt{CWPlo\_Default\_x\_First}
Data Type

Double
Purpose

Specifies the default value used by the PlotY method if the optional xFirst parameter is omitted.
See Also

PlotY
DefaultxInc Property

Syntax

CWPlot.DefaultxInc
Data Type

Double
**Purpose**

Specifies the default value used by the PlotY and ChartY methods if the optional xInc parameter is omitted.
See Also

ChanY
PlotY
Enabled Property

Syntax

\texttt{CWPlot.Enabled}
Data Type

Boolean
Purpose

Specifies if the plot generates mouse events when CWGraph.TrackMode = cwGTrackAllEvents and the plot is visible.
See Also

Visible

CWGraph.TrackMode
FillColor Property

Syntax

CWPPlot.FillColor
Data Type

Color
**Purpose**

Specifies the color to use for filling to the specified BaseValue or BasePlot values.
Example

'Fills the space between the first and second plot on the graph with red
CWGraph1.Plots.Item(1).FillToBase = True
Set CWGraph1.Plots.Item(1).BasePlot = CWGraph1.Plots.Item(2)
CWGraph1.Plots.Item(1).FillColor = vbRed
See Also

FillToBase
FillToBase Property

Syntax

CWPJob.FillToBase
Data Type

Boolean
**Purpose**

Fills the area between the plot and the values set for CWPlot.BaseValue or CWPlot.BasePlot.
Example

'Fills the space between the first and second plot on the graph with red
CWGraph1.Plots.Item(1).FillToBase = True
Set CWGraph1.Plots.Item(1).BasePlot = CWGraph1.Plots.Item(2)
CWGraph1.Plots.Item(1).FillColor = vbRed
See Also

- BaseValue
- BasePlot
- FillColor
LineColor Property

Syntax

```c
CWPlot.LineColor
```
Data Type

Color
Purpose

Specifies the color of line that connects points in the plot.
**Example**

'Set the plot line color to blue
CWGraph1.Plots.Item(1).LineColor = vbBlue
See Also

LineStyle
LineStyle Property

Syntax

CWPlot.LineStyle
Data Type

CWLineStyles

You can use the following constants with this data type:

- `cwLineDash`– Dashed line style.
- `cwLineDashDot`– Dash-dot line style.
- `cwLineDashDotDot`– Dash-dot-dot line style.
- `cwLineDot`– Dotted line style.
- `cwLineNone`– No line style.
- `cwLineSolid`– Solid line style.
- `cwLineStepXY`– Step XY line style.
- `cwLineStepYX`– Step YX line style.
Purpose

Specifies the style of lines for connecting points on a plot.
Remarks

Dashed line styles work only when the LineWidth property is set to 1. The dashed lines appear solid for all other line widths.
LineToBase Property

Syntax

`CWPlot.LineToBase`
Data Type

Boolean
Purpose

Specifies if lines connect the data points to the values specified by CWPlot.BaseValue or CWPlot.BasePlot.
See Also

Base Value
BasePlot
LineToBaseColor
LineToBaseColor Property

Syntax

```
CWPlot.LineToBaseColor
```
Data Type

Color
Purpose

Specifies the color of lines connecting data points to the values specified by CWPlot.BaseValue or CWPlot.BasePlot.
See Also

LineEditBase
LineWidth Property

Syntax

\texttt{CWPlot.LineWidth}
Data Type

Integer
Purpose
Specifies the width of the plotting line. The width range is 1 to 5.
Remarks
The line width affects the size of the points. If the width is greater than 1, dashed line styles appear solid.
See Also

LineStyle
MultiPlot Property

Syntax

\texttt{CWPlot. MultiPlot}
Data Type

Boolean
**Purpose**
Determines if the CWGraph plot or chart methods can use this plot.
Remarks

Multiplot refers to plotting several sets of data at once. For example, consider an array dimensioned (3,10) that holds three sets of Y data, each with 10 points. You can plot this array with the CWGraph.PlotY method. The CWGraph.PlotY method uses existing plots to display the data sets but only if their MultiPlot property is set to True. If the available plots do not exist, it uses temporary multiplots.

Set this property to False to create a plot the CWGraph plot or chart methods cannot use.
See Also

CWGraph.PlotY
<table>
<thead>
<tr>
<th>Name</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CWPlot.Name</td>
</tr>
</tbody>
</table>
Data Type

String
**Purpose**

Specifies the name of the plot.
Remarks

Use this name for indexing the Plots collection on a CWGraph object. If more than one plot has the same name, the first matching plot is used.
PointColor Property

Syntax

\texttt{CWPlot.PointColor}
Data Type

Color
Purpose

Specifies the color of the points on a plot.
See Also

PointStyle
PointStyle Property

Syntax

CWPlot.PointStyle
Data Type

**CWPointStyles**

You can use the following constants with this data type:

- cwPointAsterisk–Asterisk
- cwPointBoldCross–Bold cross
- cwPointBoldX–Bold X
- cwPointCross–Cross
- cwPointDottedEmptyCircle–Dotted empty circle
- cwPointDottedEmptySquare–Dotted empty square
- cwPointDottedSolidCircle–Dotted solid circle
- cwPointDottedSolidSquare–Dotted solid square
- cwPointEmptyCircle–Empty circle
- cwPointEmptyDiamond–Empty diamond
- cwPointEmptySquare–Empty square
- cwPointEmptySquareWithCross–Empty square with cross
- cwPointEmptySquareWithX–Empty square with X
- cwPointNone–None
- cwPointSimpleDot–Simple dot
- cwPointSmallCross–Small cross
- cwPointSmallEmptySquare–Small empty square
- cwPointSmallSolidSquare–Small solid square
- cwPointSmallX–Small X
- cwPointSolidCircle–Solid circle
- cwPointSolidDiamond–Solid diamond
- cwPointSolidSquare–Solid square
- cwPointX–X
**Purpose**

Specifies the image drawn at each point on a plot.
Example

' use an asterisk to mark each point of the plot
CWGraph1.Plots.Item(1).PointStyle = cwPointAsterisk
TempPlot Property

Syntax

`CWPlot.TempPlot`
Data Type

Boolean
**Purpose**

Specifies if a plot can be discarded if the next CWGraph plot or chart method does not need it.
Remarks

Check the value of this property to see if you can discard a plot. Plots allocated at design time or by using the CWPlots.Add method have this property set to False. Plots allocated by a CWGraph plot or chart method have this property set to True.
See Also

CWGraph.PlotY
Visible Property

Syntax

\texttt{CWPlot}. \texttt{Visible}
Data Type

Boolean
**Purpose**

Specifies if the plot is visible or hidden.
Remarks
Set this property to False to hide a plot without discarding its data. An invisible plot does not generate mouse events or affect autoscaling.
See Also

Enabled
XAxis Property

Syntax

Set $\texttt{CWPlo}t.X\texttt{Axis}$
Data Type

CWAxis
**Purpose**

Specifies the x axis for a plot. This is a read-only property.
XYData Property

Syntax

 CWPlot.XYData
Data Type

Variant
**Purpose**

Displays data from an external source in a plot on a graph as it would if you called the PlotXY method.
Remarks

Use this property only with a CWBinding object to link to and display an external data source. If you want to plot data from your application, use the PlotXY method.

Set the DefaultPlotPerRow property value to specify how you want the array of data plotted.
See Also

- CWBinding
- DefaultPlotPerRow
- PlotXY
XYDataAppend Property

Syntax

`CWPlot.XYDataAppend`
Data Type

Variant
Purpose
Displays data from an external source in a chart as it would if you called the ChartXY method.
Remarks

Use this property only with a CWBinding object to link to and display an external data source. If you want to chart data from your application, use the ChartXY method.

Set the DefaultPlotPerRow property value to specify how you want the array of data plotted.
See Also

CWBinding
ChartXY
DefaultPlotPerRow
YAxis Property

Syntax

Set CWPlot.YAxis
Data Type

CWAxis
**Purpose**

Specifies the y axis for a plot.
Remarks

You can set this property to any Y CWAxis object in the CWGraph.Axes collection. Use the Visual Basic Set function to assign a CWAxis object to this property.
Example

'Associate a third axis (second y axis) with second plot
Set CWGraph1.Plots(2).YAxis = CWGraph1.Axes(3)
YData Property

Syntax

`CWPlot.YData`
Data Type

Variant
**Purpose**

Displays data from an external source in a plot on a graph as it would if you called the PlotY method.
Remarks

Use this property only with a CWBinding object to link to and display an external data source. If you want to plot data from your application, use the PlotY method.

Set the DefaultxFirst and DefaultxInc property values to specify the X value for the first point in each plot and the amount to increment X in each plot.
See Also

CWBinding
DefaultsFirst
DefaultInc
PlotY
YDataAppend Property

Syntax

CWPlo. YDataAppend
Data Type

Variant
**Purpose**

Displays data from an external source in a chart as it would if you called the ChartY method.
Remarks

Use this property only with a CWBinding object to link to and display an external data source. If you want to chart data from your application, use the ChartY method.

Set the DefaultxInc property value to specify the amount to increment X in each plot.
See Also

CWBinding
ChartY
DefaultInc
ChartXvsY Method

Syntax

\texttt{CWPlot.ChartXvsY \ xData, yData}
Purpose

Charts an array of Y data against an array of X data. This method is similar to the PlotXvsY method, except that the system does not delete the previously plotted data until the amount stored is greater than the CWGraph.ChartLength property specifies.
Remarks

Use the Plot and Chart methods on a CWPlot object if you want to modify a single plot without affecting the other plots. Use the Plot and Chart methods on the CWGraph object if you want to modify all existing plots.
Parameters

xData As Variant
One-dimensional array of X data.

yData As Variant
One-dimensional array of Y data.
Example

CWGraph1.Plots.Item(2).ChartXvsY xData, yData
See Also

PlotXvsY

CWGraph.ChartLength
ChartXY Method

Syntax

\texttt{CWPlot.ChartXY xyData [, bXInFirstRow = True]}
Purpose
Charts a two-dimensional array of data as an XY chart. This method is similar to the PlotXY method, except that the system does not delete the previously plotted data until the amount stored is greater than the CWGraph.ChartLength property specifies.
Remarks

Use the Plot and Chart methods on a CWPlot object if you want to modify a single plot without affecting the other plots. Use the Plot and Chart methods on the CWGraph object if you want to modify all existing plots.
Parameters

xyData As Variant
Two-dimensional data array with exactly two rows or two columns to chart.

bXInFirstRow As Variant
[Optional] If True, the first row of the array contains the X data and the second row of the array contains a plot of Y data. If False, the first column of the array contains the X data and the second column of the array contains a plot of Y data. This parameter has a default value of True.
Example

'Chart the first row data as X data
'and the second row as Y data
CWGraph1.Plots.Item(4).ChartXY xyData, True
See Also

PlotXY
CWGraph.ChartLength
ChartY Method

Syntax

```
CWPPlot.ChartY yData [, xInc = 1]
```
Purpose
Charts a 1D array of data. This method is similar to the PlotY method, except that the system does not delete the previously plotted data until the amount stored is greater than the CWGraph.ChartLength property specifies.
Remarks

Use the Plot and Chart methods on a CWPlot object if you want to modify a single plot without affecting the other plots. Use the Plot and Chart methods on the CWGraph object if you want to modify all existing plots.
Parameters

**yData** As **Variant**
One-dimensional array of data to chart.

**xInc** As **Variant**
[Optional] The amount to increment X for each point.
This parameter has a default value of 1.
See Also

PlotY
ClearData Method

Syntax

 CWPlot.ClearData
**Purpose**

Clears the data currently displayed in the plot.
PlotXvsY Method

Syntax

```
CWPlot. PlotXvsY xData, yData
```
Purpose

Plots an array of Y data against an array of X data.
**Remarks**

Use the Plot and Chart methods on a CWPlot object if you want to modify a single plot without affecting the other plots. Use the Plot and Chart methods on the CWGraph object if you want to modify all existing plots.
Parameters

**xData** As **Variant**
One-dimensional array of X data.

**yData** As **Variant**
One-dimensional array of Y data.
See Also

CWGraph.PlotXvsY
PlotXY Method

Syntax

`CWPlot.PlotXY xyData [, bXInFirstRow = True]`
Purpose
Plots a two-dimensional array of data as an XY plot. The first row is the X data and the second row is the Y data.
Remarks

Use the Plot and Chart methods on a CWPlot object if you want to modify a single plot without affecting the other plots. Use the Plot and Chart methods on the CWGraph object if you want to modify all existing plots.
Parameters

**xyData** As [Variant](#)

Two-dimensional data array with exactly two rows or two columns to plot.

**bXInFirstRow** As [Variant](#)

[Optional] If True, the first row of the array contains the X data and the second row of the array contains a plot of Y data. If False, the first column of the array contains the X data and the second column of the array contains a plot of Y data.

This parameter has a default value of True.
Example

'Plot first row data as X data
'and second row as Y data
CWGraph1.Plots.Item(4).PlotXY xyData, True
See Also

CWGraph.PlotXY
**PlotY Method**

**Syntax**

```
CWPlot.PlotY yData [, xFirst = 0] [, xInc = 1]
```
**Purpose**

Plots a one-dimensional array of data.
Remarks

Use the Plot and Chart methods on a CWPlot object if you want to modify a single plot without affecting the other plots. Use the Plot and Chart methods on the CWGraph object if you want to modify all existing plots.
**Parameters**

**yData** As [Variant](#)
One-dimensional array of data to plot.

**xFirst** As [Variant](#)
[Optional] The X value for the first point in the plot.
This parameter has a default value of 0.

**xInc** As [Variant](#)
[Optional] The amount to increment X for each point.
This parameter has a default value of 1.
Example

'Plot the 1D array Voltages as plot 4
CWGraph1.Plots.Item(4).PlotY Voltages
See Also

CWGraph.PlotY
Color Property

Syntax

`CWPointer, Color`
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Color</th>
</tr>
</thead>
</table>

Color
Purpose

Specifies the color of the pointer.
Example

CWSlide1.Pointers.Item(1).Color = vbRed
**FillColor Property**

**Syntax**

`CWPPoint.FillColor`
Data Type

Color
Purpose

Specifies the fill color of the pointer.
Example

CWSlide1.Pointers.Item(1).FillColor = vbRed
See Also

FillStyle
FillStyle Property

Syntax

`CWPointer.FillStyle`
Data Type

CWPointerFillStyles

You can use the following constants with this data type:

- cwFillNone–No fill is drawn.
- cwFillToGreater–Fills the area between this pointer and the pointer with the next larger value. If there are no pointers with larger values, the system fills the area between this pointer and the maximum value.
- cwFillToLess–Fills the area between this pointer and the pointer with the next smaller value. If there are no pointers with smaller values, the system fills the area between this pointer and the minimum value.
- cwFillToMax–Fills the area between this pointer and the maximum value.
- cwFillToMin–Fills the area between this pointer and the minimum value.
**Purpose**

Specifies how the pointer is filled.
Example

'Fill the area between the pointer 1 and the minimum value
CWSlide1.Pointers.Item(1).FillStyle = cwFillToMin
See Also

FillColor
Index Property (Read Only)

Syntax

`CWPointer.Index`
Data Type

Long
**Purpose**

Specifies the index of the pointer in the CWPointers collection.
Example

'Display pointer 1's index - it will be 1
MsgBox CWSlide1.Pointers.Item(1).Index
See Also

CWPointers
Mode Property

Syntax

CWPointer.Mode
Data Type

**CWPointerModes**

You can use the following constants with this data type:

- **cwPointerModeControl**–The pointer responds to mouse clicks, clicks on the increment and decrement buttons, and programmatic changes.
- **cwPointerModeControlExact**–The pointer responds to mouse clicks directly on the pointer, mouse clicks on the increment and decrement buttons, and programmatic changes.
- **cwPointerModeControlSmooth**–The pointer responds to mouse clicks anywhere on the control, mouse clicks on the increment and decrement buttons, and programmatic changes. The pointer does not snap to the clicked location as it does when the mode is cwPointerModeControl. This mode prevents abrupt changes in the pointer value.
- **cwPointerModeIndicator**–The pointer does not respond to mouse clicks. You can only change the value of the pointer programatically.
- **cwPointerModeMaximum**–The pointer always contains the maximum value that other pointers have had since statistics were reset. The pointer does not respond to user input, and you cannot set its value programatically.
- **cwPointerModeMean**–The pointer always contains the mean of the values that other pointers have had since statistics were reset. The pointer does not respond to user input, and you cannot set its value programatically.
- **cwPointerModeMinimum**–The pointer always contains the minimum value that other pointers have had since statistics were reset. The pointer does not respond to user input, and you cannot set its value programatically.
Purpose
Specifies how the pointer behaves and responds to user input.
Example

CWSlide1.Pointers.Item(1).Mode = cwPointerModeIndicator
<table>
<thead>
<tr>
<th>Name Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
</tr>
<tr>
<td><code>CWPointer.Name</code></td>
</tr>
</tbody>
</table>
Data Type

String
**Purpose**

Specifies the name of the pointer.
Remarks

Use the name to access the pointer in the CWPointers collection.
Example

' Set pointer 1's name
CWSlide1.Pointers.Item(1).Name = "Value"

' Access the pointer by name
CWSlide1.Pointers.Item("Value").Value = 10
See Also

CWPointers
Style Property

Syntax

`CWPointer.Style`
Data Type

**CWPointerStyles**

You can use the following constants with this data type:

- **cwPointer3D**–The pointer appears as a three-dimensional needle. This style is available only on the CWKnob control.
- **cwPointerCustom**–A custom image represents the pointer. This style is available only on the CWSlide control.
- **cwPointerLeftBottomArrow**–The pointer appears as an arrow facing left or to the bottom, depending on whether the CWSlide is vertical or horizontal. This style is available only on the CWSlide control.
- **cwPointerNone**–No pointer image is used.
- **cwPointerNormal**–On a CWKnob control, the pointer appears as a thin line. On a CWSlide control, the pointer appears as a slider "thumb."
- **cwPointerRightTopArrow**–The pointer appears as an arrow facing right or to the top, depending on whether the CWSlide is vertical or horizontal. This style is available only on the CWSlide control.
Purpose

Specifies the graphical style of the pointer.
Example

CWSlide1.Pointers.Item(1).Style = cwPointerLeftBottomArrow

CWKnob1.Pointers.Item(1).Style = cwPointer3D
Value Property

Syntax

`CWPointer, Value`
| Data Type | Variant |
Purpose

Specifies the value of the pointer.
Remarks

A pointer can have a value larger than the maximum value on a scale or smaller than the minimum value, though the pointer is drawn within the scale values. For example, if the scale range is 1 to 10 and the pointer value is 20, the pointer appears at 10. When the control is in ValuePairs Only mode, this property is the value of the value pair selected by the pointer. You must use ValuePairIndex to set the value.
Example

CWSlide1.Pointers.Item(1).Value = 3
See Also

ValuePairIndex
Value
ValuePairIndex
ValuePairIndex Property

Syntax

`CWPointer.ValuePairIndex`
Data Type
Long
Purpose

Specifies the index of the value pair selected by this pointer.
Remarks

This property is valid only when the control is in ValuePairs Only mode.
Example

CWSlide1.Pointers.Item(1).ValuePairIndex = 2
See Also

Value
ValuePairIndex
Visible Property

Syntax

`CWPointer.Visible`
Data Type

Boolean
**Purpose**

Specifies if the pointer is visible or hidden.
Example

'Make the pointer visible
CWSlide1.Pointers.Item(1).Visible = True
Color Property

Syntax

\texttt{CWShape. Color}
Data Type

Color
**Purpose**

Specifies the color of the shape.
Remarks

This property specifies the fill color for cwShapeRectangle, cwShapePolygon, cwShapeOval, cwShapeArc, cwShapeRegion, and cwShapeMinMaxRegion. This property is not used with cwShapeLine and cwShapeImage.
Example

'Highlight in blue all the values that are below 5 on the y axis.
CWGraph1.Annotations.Item(1).Shape.Type = cwShapeMinMaxRegion
CWGraph1.Annotations.Item(1).Shape.RegionArea = cwRegionInverted
CWGraph1.Annotations.Item(1).Shape.XCoordinates = Array()
CWGraph1.Annotations.Item(1).Shape.YCoordinates = Array(5)
CWGraph1.Annotations.Item(1).Shape.FillVisible = True
CWGraph1.Annotations.Item(1).Shape.Color = vbBlue
See Also

CWShape.Type
CWShape.LineColor
FillVisible Property

Syntax

\texttt{CWShape.FillVisible}
Data Type

Boolean
Purpose

Specifies if the shape is filled with the value specified by the CWShape.Color property.
Remarks

By default, the FillVisible property is set to False, which displays only an outline of the shape.
Example

'Highlight in blue all the values that are below 5 on the y axis.
CWGraph1.Annotations.Item(1).Shape.Type = cwShapeMinMaxRegion
CWGraph1.Annotations.Item(1).Shape.RegionArea = cwRegionInverted
CWGraph1.Annotations.Item(1).Shape.XCoordinates = Array()
CWGraph1.Annotations.Item(1).Shape.YCoordinates = Array(5)
CWGraph1.Annotations.Item(1).Shape.FillVisible = True
CWGraph1.Annotations.Item(1).Shape.Color = vbBlue
See Also

CWShape Color
**Image Property**

**Syntax**

Set CwShape.Image
Data Type

CWPictureDisp
**Purpose**

Specifies the image of the shape.
Remarks

You can use bitmaps, icons, metafiles, GIFs, and JPEGs with the Image property
LineColor Property

Syntax

`CWShape.LineColor`
<table>
<thead>
<tr>
<th>Data Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td></td>
</tr>
</tbody>
</table>
**Purpose**

Specifies the line color of the shape.
See Also

CWSHape Color
LineStyle
LineWidth
LineStyle Property

Syntax

`CWShape.LineStyle`
**Data Type**

[CWLLineStyles](#)

You can use the following constants with this data type:

- `cwLineDash`–Dashed line style.
- `cwLineDashDot`–Dash-dot line style.
- `cwLineDashDotDot`–Dash-dot-dot line style.
- `cwLineDot`–Dotted line style.
- `cwLineNone`–No line style.
- `cwLineSolid`–Solid line style.
- `cwLineStepXY`–Step XY line style.
- `cwLineStepYX`–Step YX line style.
Purpose
Specifies the line style of the shape.
Remarks
You can only use the cwLineStepXY and cwLineStepYX constants with the CWPlot.LineStyle property.
See Also

LineColor
LineWidth
CWPlot.LineStyle
LineWidth Property

Syntax

\texttt{CWShape, LineWidth}
Data Type

Integer
Purpose
Specifies the line width of the shape.
See Also

LineColor
LineStyle
PointStyle Property

Syntax

` CWShape.PointStyle`
Data Type

You can use the following constants with this data type:

- cwPointAsterisk–Asterisk
- cwPointBoldCross–Bold cross
- cwPointBoldX–Bold X
- cwPointCross–Cross
- cwPointDottedEmptyCircle–Dotted empty circle
- cwPointDottedEmptySquare–Dotted empty square
- cwPointDottedSolidCircle–Dotted solid circle
- cwPointDottedSolidSquare–Dotted solid square
- cwPointEmptyCircle–Empty circle
- cwPointEmptyDiamond–Empty diamond
- cwPointEmptySquare–Empty square
- cwPointEmptySquareWithCross–Empty square with cross
- cwPointEmptySquareWithX–Empty square with X
- cwPointNone–None
- cwPointSimpleDot–Simple dot
- cwPointSmallCross–Small cross
- cwPointSmallEmptySquare–Small empty square
- cwPointSmallSolidSquare–Small solid square
- cwPointSmallX–Small X
- cwPointSolidCircle–Solid circle
- cwPointSolidDiamond–Solid diamond
- cwPointSolidSquare–Solid square
- cwPointX–X
**Purpose**

 Specifies the point style of the shape.
Remarks
You must set the CWShape.Type property to cwShapePoint to use the PointStyle property.
RegionArea Property

Syntax

\texttt{CwShape.RegionArea}
Data Type

CWRegionAreas

You can use the following constants with this data type:

- cwRegionInverted—Specifies an inverted region area.
- cwRegionNormal—Specifies a normal region area.
**Purpose**

Specifies the region area of the shape.
Remarks

Typically, an inverted region area highlights values less than the XCoordinates or YCoordinates values. A normal region area usually highlights values greater than the XCoordinates and YCoordinates values.
**Example**

'Highlight in blue all the values that are below 5 on the y axis.
CWGraph1.Annotations.Item(1).Shape.Type = cwShapeMinMaxRegion
CWGraph1.Annotations.Item(1).Shape.RegionArea = cwRegionInverted
CWGraph1.Annotations.Item(1).Shape.XCoordinates = Array()
CWGraph1.Annotations.Item(1).Shape.YCoordinates = Array(5)
CWGraph1.Annotations.Item(1).Shape.FillVisible = True
CWGraph1.Annotations.Item(1).Shape.Color = vbBlue
See Also

XCoordinates
YCoordinates
Type Property

Syntax

\texttt{CWShape \_ Type}
Data Type

You can use the following constants with this data type:

- `cwShapeArc`—Specifies an arc.
- `cwShapeImage`—Specifies an image.
- `cwShapeLine`—Specifies a line.
- `cwShapeMinMaxRegion`—Specifies a minimum-maximum region.
- `cwShapeNone`—Specifies no shape.
- `cwShapeOval`—Specifies an oval.
- `cwShapePoint`—Specifies a point.
- `cwShapePolygon`—Specifies a polygon.
- `cwShapeRectangle`—Specifies a rectangle.
- `cwShapeRegion`—Specifies a region.
Purpose

Specifies the type of shape.
Example

Dim shape As CWShape
...
'Specify a triangle
shape.Type = cwShapePolygon
shape.XCoordinates = Array(1, 2, 3)
shape.YCoordinates = Array(1, 2, 1)
'Specify a rectangle from (1,1) to (2,2)
shape.Type = cwShapeRectangle
shape.XCoordinates = Array(1, 2)
shape.YCoordinates = Array(1, 2)
'Specify a quarter turn arc of radius 3 centered at (1,1)
shape.Type = cwShapeArc
shape.XCoordinates = Array(1, 3, 0, 0)
shape.YCoordinates = Array(1, 3, 0, 90)
'Specify the region x > 5
shape.Type = cwShapeMinMaxRegion
shape.XCoordinates = Array(5)
shape.YCoordinates = Array()
See Also

XCoordinates
YCoordinates
XCoordinates, YCoordinates Properties

Syntax

```
CWShape.XCoordinates
CWShape.YCoordinates
```
| Data Type | Variant |
**Purpose**

Specifies the X and Y coordinates of the shape.
Remarks

Assign arrays of values to the XCoordinates and YCoordinates properties to specify the coordinates of the shape.

For cwShapePoint, cwShapeLine, cwShapePolygon, and cwShapeRegion, the pairs of X and Y coordinates are used as vertices.

For cwShapeRectangle, cwShapeOval, and cwShapeImage, the first two pairs of X and Y coordinates are used as corner vertices.

For cwShapeArc, the first X and Y coordinates are used as the center of the rotation of the arc. The second X and Y coordinates are used as the major and minor axes lengths. If specified, the third X coordinate is used as the rotation, in degrees, of the arc. If specified, the fourth X and Y coordinates are used as the starting and ending angles of the arc.

For cwMinMaxRegion, the first X coordinate, if specified, is used as an X minimum. If specified, the first Y coordinate is used as a Y minimum. If specified, the second X coordinate is used as an X maximum. If specified, the second Y coordinate is used as a Y maximum.
Example

Dim shape As CWShape
...
'Specify a triangle
shape.Type = cwShapePolygon
shape.XCoordinates = Array(1, 2, 3)
shape.YCoordinates = Array(1, 2, 1)
'Specify a rectangle from (1,1) to (2,2)
shape.Type = cwShapeRectangle
shape.XCoordinates = Array(1, 2)
shape.YCoordinates = Array(1, 2)
'Specify a quarter turn arc of radius 3 centered at (1,1)
shape.Type = cwShapeArc
shape.XCoordinates = Array(1, 3, 0, 0)
shape.YCoordinates = Array(1, 3, 0, 90)
'Specify the region x > 5
shape.Type = cwShapeMinMaxRegion
shape.XCoordinates = Array(5)
shape.YCoordinates = Array()
See Also

CWShape.Type
SetCoordinates Method

Syntax

CWShape.SetCoordinates XCoordinates, YCoordinates
**Purpose**

Sets the X and Y coordinates of the shape.
Remarks

The SetCoordinates method assigns arrays of values to specify the coordinates of the shape.

For cwShapePoint, cwShapeLine, cwShapePolygon, and cwShapeRegion, the pairs of X and Y coordinates are used as vertices.

For cwShapeRectangle, cwShapeOval, and cwShapeImage, the first two pairs of X and Y coordinates are used as corner vertices.

For cwShapeArc, the first X and Y coordinates are used as the center of the rotation of the arc. The second X and Y coordinates are used as the major and minor axes lengths. If specified, the third X coordinate is used as the rotation, in degrees, of the arc. If specified, the fourth X and Y coordinates are used as the starting and ending angles of the arc.

For cwMinMaxRegion, the first X coordinate, if specified, is used as an X minimum. If specified, the first Y coordinate is used as a Y minimum. If specified, the second X coordinate is used as an X maximum. If specified, the second Y coordinate is used as a Y maximum.
Parameters

**XCoordinates** As *Variant*
The X coordinates of the shape.

**YCoordinates** As *Variant*
The Y coordinates of the shape.
See Also

XCoordinates
YCoordinates
InteriorColor Property

Syntax

CWSlide. InteriorColor
Data Type

Color
**Purpose**

Specifies the color used to paint the interior of several types of slide control styles.
See Also

SetBuiltinStyle
ForeColor
BackColor
Refresh Method

Syntax

CWSlide.Refresh
Purpose
Redraws the CWSlide control.
SetBuiltinStyle Method

Syntax

CWSlide.SetBuiltinStyle Style
Purpose

Sets many properties of the control to represent the new style specified.
Parameters

Style As CWSlideStyles

The specified slide style.
Example
CWSlide1.SetBuiltInStyle cwSlideStyleTank
Maximum Property (Read Only)

Syntax

`CWStatistics.Maximum`
Data Type

Variant
**Purpose**

Returns the maximum value of any pointer since the statistics were last reset.
Example

'Output the min, max, and mean
MsgBox CWSlide1.Statistics.Maximum
MsgBox CWSlide1.Statistics.Minimum
MsgBox CWSlide1.Statistics.Mean

'Reset the statistics
CWSlide1.Statistics.Reset
See Also

Reset
Minimum
Mean
CWSlide
CWKnob
Mean Property (Read Only)

Syntax

`CWStatistics.Mean`
Data Type

Variant
Purpose

Returns the mean of the last 10 pointer values.
Example

'Output the min, max, and mean
MsgBox CWSlide1.Statistics.Maximum
MsgBox CWSlide1.Statistics.Minimum
MsgBox CWSlide1.Statistics.Mean

'Reset the statistics
CWSlide1.Statistics.Reset
See Also

- Reset
- Minimum
- Maximum
- CWSlide
- CWKnob
Minimum Property (Read Only)

Syntax

CWStatistics. Minimum
Data Type

Variant
**Purpose**

Returns the minimum value of any pointer since the statistics were last reset.
Example

'Output the min, max, and mean
MsgBox CWSlide1.Statistics.Maximum
MsgBox CWSlide1.Statistics.Minimum
MsgBox CWSlide1.Statistics.Mean

'Reset the statistics
CWSlide1.Statistics.Reset
See Also

Reset
Maximum
Mean
CWSlide
CWKnob
Reset Method

Syntax

 CWStatistics. Reset
Purpose
Resets the minimum, maximum, and mean statistics. After resetting the values, the system recalculates the minimum and maximum using only values collected since the last reset.
Example

'Output the min, max, and mean
MsgBox CWSlide1.Statistics.Maximum
MsgBox CWSlide1.Statistics.Minimum
MsgBox CWSlide1.Statistics.Mean

'Reset the statistics
CWSlide1.Statistics.Reset
See Also

Minimum
Maximum
Mean
CWSlide
CWKnob
Above Property

Syntax

CWTicks. Above
Data Type

Boolean
Purpose

Specifies if tick marks are drawn above the object.
Remarks

This property is valid only when the axis is horizontal.
Example

'Turn on all ticks on the graph

'Work on the x axis
CWGraph1.Axes.Item(1).Ticks.Above = True
CWGraph1.Axes.Item(1).Ticks.Below = True

'Work on the y axis
CWGraph1.Axes.Item(2).Ticks.Left = True
CWGraph1.Axes.Item(2).Ticks.Right = True
AutoDivisions Property

Syntax

\texttt{CWTicks.AutoDivisions}
Data Type

Boolean
**Purpose**

Specifies if divisions are automatically calculated.
Example

'Autocalculate divisions on the graph's x axis
CWGraph1.Axes.Item(1).Ticks.AutoDivisions = True
See Also

MajorDivisions
MajorUnitsInterval
Below Property

Syntax

CWTicks: Below
Data Type

Boolean
**Purpose**

Specifies if tick marks are drawn below the object.
Remarks

This property is valid only when the axis is horizontal.
Example

'Turn on all ticks on the graph

'Work on the x axis
CWGraph1.Axes.Item(1).Ticks.Above = True
CWGraph1.Axes.Item(1).Ticks.Below = True

'Work on the y axis
CWGraph1.Axes.Item(2).Ticks.Left = True
CWGraph1.Axes.Item(2).Ticks.Right = True
Inside Property

Syntax

CWTicks, Inside
Data Type

Boolean
Purpose

Specifies if tick marks are drawn on the inside of the axis.
**Example**

'Draw ticks on the inside and outside of the axis
CWGraph1.Axes.Item(1).Ticks.Inside = True

CWGraph1.Axes.Item(1).Ticks.Outside = True
Left Property

Syntax

CWTicks. Left
Data Type

Boolean
Purpose

Specifies if tick marks are drawn to the left of the object.
Remarks
This property is valid only when the axis is vertical.
**Example**

'Turn on all ticks on the graph

'Work on the x axis
CWGraph1.Axes.Item(1).Ticks.Above = True
CWGraph1.Axes.Item(1).Ticks.Below = True

'Work on the y axis
CWGraph1.Axes.Item(2).Ticks.Left = True
CWGraph1.Axes.Item(2).Ticks.Right = True
MajorDivisions Property

Syntax

CWTicks, MajorDivisions
Data Type

Variant
Purpose

Specifies the number of major divisions.
Remarks

If MajorDivisions is 0, the MajorUnitsInterval property takes effect.
Set MajorDivisions to 10 to split the axis range into 10 equal parts, like the grid on an oscilloscope.
Example

'Set 10 major divisions on the graph's x axis
CWGraph1.Axes.Item(1).Ticks.MajorDivisions = 10

'Set 2 minor divisions per major division
CWGraph1.Axes.Item(1).Ticks.MinorDivisions = 2
See Also

MajorUnitsInterval
MinorDivisions
MajorGrid Property

Syntax

`CWTicks, MajorGrid`
Data Type

Boolean
**Purpose**

Specifies if major grid lines appear.
Example

'Enable major and minor grid lines on the graph's x axis
CWGraph1.Axes.Item(1).Ticks.MajorGrid = True
CWGraph1.Axes.Item(1).Ticks.MinorGrid = True

'Set the grid line colors
CWGraph1.Axes.Item(1).Ticks.MajorGridColor = vbRed
CWGraph1.Axes.Item(1).Ticks.MinorGridColor = vbYellow
MajorGridColor Property

Syntax

`CWTicks.MajorGridColor`
<table>
<thead>
<tr>
<th>Data Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td></td>
</tr>
</tbody>
</table>
Purpose

Specifies the color of major grid lines.
Example

'Enable major and minor grid lines on the graph's x axis
CWGraph1.Axes.Item(1).Ticks.MajorGrid = True
CWGraph1.Axes.Item(1).Ticks.MinorGrid = True

'Set the grid line colors
CWGraph1.Axes.Item(1).Ticks.MajorGridColor = vbRed
CWGraph1.Axes.Item(1).Ticks.MinorGridColor = vbYellow
See Also

MinorGridColor
MajorTickColor Property

Syntax

CWTicks.MajorTickColor
Data Type

Color
Purpose

Specifies the color of major ticks.
Example

'Set the tick colors of the graph's x axis
CWGraph1.Axes.Item(1).Ticks.MajorTickColor = vbRed
CWGraph1.Axes.Item(1).Ticks.MinorTickColor = vbYellow
See Also

MinorTickColor
MajorTicks Property

Syntax

CWTicks, MajorTicks
Data Type

Boolean
**Purpose**

Specifies if major ticks appear.
Example

'Turn major and minor ticks on
CWGraph1.Axes(1).Ticks.MajorTicks = True
CWGraph1.Axes(1).Ticks.MinorTicks = True
See Also

MinorTicks
MajorUnitsBase Property

Syntax

CWTicks, MajorUnitsBase
Data Type

Variant
**Purpose**

Specifies the base number for calculating ticks.
Remarks

When ticks are spaced by units, ticks are placed at locations that fit the following equation:

\[ \text{MajorUnitsBase} + (n \times \text{MajorUnitsInterval}) \]

where \( n \) is any integer.
**Example**

'Specify that ticks are determined by MajorUnitsInterval
CWGraph1.Axes(1).Ticks.MajorDivisions = 0

'Place major ticks every 2 units (even numbers)
CWGraph1.Axes(1).Ticks.MajorUnitsInterval = 2

'Place minor ticks every .5 units
CWGraph1.Axes(1).Ticks.MinorUnitsInterval = .5

'The units base is 0
CWGraph1.Axes(1).Ticks.MajorUnitsBase = 0

'To specify ticks at odd numbers, change the base to 1
CWGraph1.Axes(1).Ticks.MajorUnitsBase = 1

'Now, there will be major ticks at ..., -3, -1, 1, 3, 5, ...
See Also

- MajorUnitsInterval
- MinorUnitsInterval
- MajorDivisions
MajorUnitsInterval Property

Syntax

`CWTicks.MajorUnitsInterval`
Data Type

Variant
Purpose

Specifies the number of units between major divisions.
Remarks

MajorDivisions must be set to 0 for this property to take effect. Set the MajorDivisions and MajorUnitsInterval properties to 0 to have no major ticks. Set this property to 5 and MajorUnitsBase to 0 to place major tick marks at multiples of five.
Example

'Specify that ticks are determined by MajorUnitsInterval
CWGraph1.Axes(1).Ticks.MajorDivisions = 0

'Place major ticks every 2 units (even numbers)
CWGraph1.Axes(1).Ticks.MajorUnitsInterval = 2

'Place minor ticks every .5 units
CWGraph1.Axes(1).Ticks.MinorUnitsInterval = .5

'The units base is 0
CWGraph1.Axes(1).Ticks.MajorUnitsBase = 0

'To specify ticks at odd numbers, change the base to 1
CWGraph1.Axes(1).Ticks.MajorUnitsBase = 1

'Now, there will be major ticks at ..., -3, -1, 1, 3, 5, ...
See Also

MajorUnitsBase
MinorUnitsInterval
MajorDivisions
MinorDivisions Property

Syntax

CWTicks.MinorDivisions
Data Type
Variant
Purpose

Specifies the number of minor divisions for each major division.
Remarks

Set the MinorDivisions property to 0 to disable minor divisions.
Example

'Set 10 major divisions on the graph's x axis
CWGraph1.Axes.Item(1).Ticks.MajorDivisions = 10

'Set 2 minor divisions per major division
CWGraph1.Axes.Item(1).Ticks.MinorDivisions = 2
See Also

MajorDivisions
MinorUnitsInterval
MinorGrid Property

Syntax

CWTicks, MinorGrid
Data Type

Boolean
**Purpose**

Specifies if minor grid lines appear.
Example

'Enable major and minor grid lines on the graph's x axis
CWGraph1.Axes.Item(1).Ticks.MajorGrid = True
CWGraph1.Axes.Item(1).Ticks.MinorGrid = True

'Set the grid line colors
CWGraph1.Axes.Item(1).Ticks.MajorGridColor = vbRed
CWGraph1.Axes.Item(1).Ticks.MinorGridColor = vbYellow
See Also

MajorGrid
MinorGridColor Property

Syntax

CWTicks.MinorGridColor
Data Type

Color
**Purpose**

Specifies the color of minor grid lines.
**Example**

'Enable major and minor grid lines on the graph's x axis
CWGraph1.Axes.Item(1).Ticks.MajorGrid = True
CWGraph1.Axes.Item(1).Ticks.MinorGrid = True

'Set the grid line colors
CWGraph1.Axes.Item(1).Ticks.MajorGridColor = vbRed
CWGraph1.Axes.Item(1).Ticks.MinorGridColor = vbYellow
See Also

MajorGridColor
MinorTickColor Property

Syntax

CWTicks.MinorTickColor
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Color</th>
</tr>
</thead>
</table>

Purpose
Specifies the color of minor ticks.
Example

'Set the tick colors of the graph's x axis
CWGraph1.Axes.Item(1).Ticks.MajorTickColor = vbRed
CWGraph1.Axes.Item(1).Ticks.MinorTickColor = vbYellow
See Also

MajorTickColor
MinorTicks Property

Syntax

CWTicks, MinorTicks
Data Type

Boolean
**Purpose**

Specifies if minor ticks appear.
Example

'Turn major and minor ticks on
CWGraph1.Axes(1).Ticks.MajorTicks = True
CWGraph1.Axes(1).Ticks.MinorTicks = True
See Also

MajorTicks
MinorUnitsInterval Property

Syntax

CWTicks. MinorUnitsInterval
Data Type

Variant
Purpose

Specifies the number of units between minor divisions.
Remarks
Set the MinorDivisions property to 0 to disable minor divisions.
**Example**

'Specify that ticks are determined by MajorUnitsInterval
`CWGraph1.Axes(1).Ticks.MajorDivisions = 0`

'Place major ticks every 2 units (even numbers)
`CWGraph1.Axes(1).Ticks.MajorUnitsInterval = 2`

'Place minor ticks every .5 units
`CWGraph1.Axes(1).Ticks.MinorUnitsInterval = .5`

'The units base is 0
`CWGraph1.Axes(1).Ticks.MajorUnitsBase = 0`

'To specify ticks at odd numbers, change the base to 1
`CWGraph1.Axes(1).Ticks.MajorUnitsBase = 1`

'Now, there will be major ticks at ..., -3, -1, 1, 3, 5, ...
See Also

MajorUnitsBase
MajorUnitsBase
MajorDivisions
Outside Property

Syntax

CWTicks. Outside
Data Type

Boolean
**Purpose**

Specifies if tick marks are drawn on the outside of the axis.
Example

'Draw ticks on the inside and outside of the axis
CWGraph1.Axes.Item(1).Ticks.Inside = True

CWGraph1.Axes.Item(1).Ticks.Outside = True
Right Property

Syntax

\texttt{CWTicks. Right}
Data Type

Boolean
Purpose

Specifies if tick marks are drawn to the right of the object.
Remarks

This property is valid only when the axis is vertical.
**Example**

'Turn on all ticks on the graph

'Work on the x axis
CWGraph1.Axes.Item(1).Ticks.Above = True

CWGraph1.Axes.Item(1).Ticks.Below = True

'Work on the y axis
CWGraph1.Axes.Item(2).Ticks.Left = True

CWGraph1.Axes.Item(2).Ticks.Right = True
Name Property

Syntax

`CWValuePair.Name`
| Data Type | String |
Purpose

Specifies the name of a value pair.
Example

'Create a value pair
CWGraph1.Axes.Item(1).ValuePairs.Item(1).Add

'Set the name to "Boiling Point" and the value to 212
CWGraph1.Axes.Item(1).ValuePairs.Item(1).Name = "Boiling Point"
CWGraph1.Axes.Item(1).ValuePairs.Item(1).Value = "212"
See Also

Value
Value Property

Syntax

CWValuePair.Value
Data Type

Variant
Purpose

Specifies the value of a value pair.
Example

'Create a value pair
CWGraph1.Axes.Item(1).ValuePairs.Item(1).Add

'Set the name to "Boiling Point" and the value to 212
CWGraph1.Axes.Item(1).ValuePairs.Item(1).Name = "Boiling Point"
CWGraph1.Axes.Item(1).ValuePairs.Item(1).Value = "212"
See Also

Name
GridLines Property

Syntax

`CWValuePairs.GridLines`
Data Type

Boolean
Purpose

Specifies if grid lines appear at value pair locations.
Example

Turn on major ticks and grid lines at value pair locations
CWSlide1.Axis.ValuePairs.MajorTicks = True
CWSlide1.Axis.ValuePairs.GridLines = True
See Also

Location
CWTicks
LabelType Property

Syntax

CWValuePairs.LabelType
### Data Type

**CWValuePairLabels**

You can use the following constants with this data type:

- `cwVPLabelName`–The axis draws the name of the value pairs.
- `cwVPLabelNone`–The axis does not draw the value pair.
- `cwVPLabelValue`–The axis draws the value of the value pair.
Purpose

Specifies the type of labels to draw for the value pairs.
Example

'Display the value pair's name, not its value
CWSlide1.Axis.ValuePairs.LabelType = cwVPLabelName
Location Property

Syntax

\texttt{CWValuePairs.Location}
Data Type

**CWValuePairLocations**

You can use the following constants with this data type:

- cwVPLocationIndex–The axis draws the value pairs at their index on the axis.
- cwVPLocationValue–The axis draws the value pairs at their value on the axis.
**Purpose**

Specifies if CWValuePairs are placed on the axis by their value or by their index.
Example

'Locate the value pairs by their value, not their index
CWSlide1.Axis.ValuePairs.Location = cwVPLocationValue
MajorTicks Property

Syntax

CWValuePairs.MajorTicks
Data Type

Boolean
**Purpose**

Specifies if major ticks are placed at the location of value pairs.
Example

'Turn on major ticks and grid lines at value pair locations
CWSlide1.Axis.ValuePairs.MajorTicks = True
CWSlide1.Axis.ValuePairs.GridLines = True
See Also

Location
GridLines
CWTicks
Swap Method

Syntax

\texttt{\texttt{CWValuePairs}.Swap\ element1,\ element2}
Purpose
Swaps two CWValuePair elements, altering their indices.
Remarks
This method is useful on CWSlide and CWKnob controls in ValuePairs Only mode.
Parameters

**element1** As *Variant*
Specifies the index of a value pair to be swapped.

**element2** As *Variant*
Specifies the index of a second value pair to be swapped.
Example

'Swap value pairs 1 and 2
CWSlide1.Axis.ValuePairs.Swap 1, 2
**CWCoordinateTypes Enumeration**

CWCoordinateTypes are the constants for the CWAnnotation.CoordinateType property.

You can use the following constants with this data type:

- **cwAxesCoordinates**—Scales the coordinates to the associated axes.
- **cwPlotAreaCoordinates**—Scales the coordinates in pixels relative to the upper-left corner of the plot area.
- **cwScreenCoordinates**—Scales the coordinates in pixels relative to the upper-left corner of the graph.
See Also

CWAnnotation.CoordinateType
## Data Types for CWUI

<table>
<thead>
<tr>
<th>Visual Basic</th>
<th>C/C++</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>VARIANT_BOOL</td>
<td>Has the value True (-1) or False (0).</td>
</tr>
<tr>
<td>Color</td>
<td>OLECOLOR</td>
<td>A color. In many containers, colors are treated as 32-bit integers.</td>
</tr>
<tr>
<td>Double</td>
<td>double</td>
<td>64-bit floating point number.</td>
</tr>
<tr>
<td>Double Array</td>
<td>LPSAFEARRAY</td>
<td>An array of doubles.</td>
</tr>
<tr>
<td>Font</td>
<td>LPFONTDISP</td>
<td>An OLE Automation Interface to a font.</td>
</tr>
<tr>
<td>Integer</td>
<td>short</td>
<td>16-bit signed integer.</td>
</tr>
<tr>
<td>Long</td>
<td>long</td>
<td>32-bit signed integer.</td>
</tr>
<tr>
<td>LPUNKNOWN</td>
<td>LPUNKNOWN</td>
<td></td>
</tr>
<tr>
<td>Object</td>
<td>LPDISPATCH</td>
<td>A generic OLE Automation Interface.</td>
</tr>
<tr>
<td>Object Array</td>
<td>SAFEARRAY</td>
<td>An array of objects.</td>
</tr>
<tr>
<td>OLE_XPOS_PIXELS</td>
<td>long</td>
<td></td>
</tr>
<tr>
<td>OLE_YPOS_PIXELS</td>
<td>long</td>
<td></td>
</tr>
<tr>
<td>Picture</td>
<td>LPPICUREIDISP</td>
<td>An OLE Automation Interface to a picture or image.</td>
</tr>
<tr>
<td>Single</td>
<td>single</td>
<td>32-bit floating point number.</td>
</tr>
<tr>
<td>String</td>
<td>BSTR</td>
<td>A string.</td>
</tr>
<tr>
<td>String Array</td>
<td>SAFEARRAY</td>
<td>An array of strings.</td>
</tr>
<tr>
<td>Variant</td>
<td>LPVARIANT</td>
<td>A variant.</td>
</tr>
<tr>
<td>Void</td>
<td>void</td>
<td>Nothing.</td>
</tr>
<tr>
<td>Window</td>
<td>OLE_HANDLE</td>
<td>A handle to a window.</td>
</tr>
</tbody>
</table>
**CWCursorSnapModes Enumeration**

CWCursorSnapModes are the constants for the CWCursor.SnapMode and the CWAnnotation.SnapMode properties.

You can use the following constants with this data type:

- **cwCSnapAnchoredToPoint**–Snaps the cursor or annotation to a specified point on a specific plot. The cursor or annotation cannot be moved.
- **cwCSnapFixed**–Sets the cursor or annotation position independent of any plot. The cursor or annotation cannot be moved.
- **cwCSnapFloating**–Sets the cursor or annotation independent of any plot. You can move the cursor or annotation anywhere in the graph area.
- **cwCSnapNearestPoint**–Snaps the cursor or annotation to the nearest point on any plot. You can move the cursor or annotation along any plot.
- **cwCSnapNearestYForFixedX**–For a fixed X location, the cursor's Y value is the Y value of the nearest point on the specified plot. For an annotation cwCSnapNearestYForFixedX is equivalent to cwCSnapPointsOnPlot. The cursor or annotation can be moved only along this plot.
- **cwCSnapPointsOnPlot**–Snaps the cursor or annotation to points on a specified plot. The cursor or annotation can be moved only along this plot.
CWAnnotationStyles Enumeration

CWAnnotationStyles are the constants for the Style parameter of the CWAnnotation.SetBuiltinStyle method.

You can use the following constants with this data type:

- cwAnnotationStyleArrow—Arrow annotation style.
- cwAnnotationStyleHalfSpace—Half-space annotation style.
- cwAnnotationStyleLine—Line annotation style.
- cwAnnotationStylePicture—Picture annotation style.
- cwAnnotationStyleRange—Range annotation style.
- cwAnnotationStyleRectangle—Rectangle annotation style.
- cwAnnotationStyleText—Text annotation style.
- cwAnnotationStyleTextAndArrow—Text and arrow annotation style.
See Also

CWAAnnotation.SetBuiltInStyle
**CWArrowHeadStyles Enumeration**

CWArrowHeadStyles are the constants for the CWArrow.HeadStyle and CWArrow.TailStyle properties.

You can use the following constants with this data type:

- `cwArrowHeadDiamond`– Specifies a diamond-shaped arrowhead.
- `cwArrowHeadLine`– Specifies an arrowhead with thin lines.
- `cwArrowHeadNone`– Specifies no arrowhead.
- `cwArrowHeadRound`– Specifies a round arrowhead.
- `cwArrowHeadSolid`– Specifies a solid arrowhead.
- `cwArrowHeadStealth`– Specifies a stylized arrowhead.
See Also

CWArrow.HeadStyle
CWArrow.TailStyle
**CWLineStyles Enumeration**

CWLineStyles are the constants for the CWPlot.LineStyle and CWArrow.LineStyle properties. You can only use cwLineStepXY and cwLineStepYX with the CWPlot.LineStyle property.

You can use the following constants with this data type:

- `cwLineDash`—Dashed line style.
- `cwLineDashDot`—Dash-dot line style.
- `cwLineDashDotDot`—Dash-dot-dot line style.
- `cwLineDot`—Dotted line style.
- `cwLineNone`—No line style.
- `cwLineSolid`—Solid line style.
- `cwLineStepXY`—Step XY line style.
- `cwLineStepYX`—Step YX line style.
**CWDSAccessModes Enumeration**

CWDSAccessModes are the constants for the AccessMode property on CWBinding.

You can use the following constants with this data type:

- **cwdsRead**—Reads once when connected. You can trigger another read by calling the Update method.
- **cwdsReadAutoUpdate**—Reads when connected and automatically reads again if the data at the data source is updated.
- **cwdsReadWriteAutoUpdate**—Writes the current data once connected. The data is rewritten automatically if any attribute or value is set. Reads when connected and automatically reads again if the data at the data source is updated.
- **cwdsWrite**—Writes the current data once connected. You can trigger the write again by calling the Update method.
- **cwdsWriteAutoUpdate**—Writes the current data once connected. The data is rewritten automatically if any attribute or value is set.
**CWDSStatus Enumeration**

The CWDSStatus object holds constants for the Status property on CWBinding. You can use the following constants with this data type:

- **cwdsConnecting**–DataSocket is in the process of connecting to the data source or target.
- **cwdsConnectionActive**–DataSocket is in the process of transferring the data or waiting for an update.
- **cwdsConnectionError**–DataSocket encountered an error connecting to the data source or target.
- **cwdsConnectionIdle**–DataSocket has connected to the data source and transferred the data.
- **cwdsUnconnected**–DataSocket is not connected to any data source or data target.
CWKeyboardModes Enumeration

Constants that specify how a control responds to keyboard input.

You can use the following constants with this data type:

- `cwKeyboardHandled`– Keystrokes are processed by the control.
- `cwKeyboardNone`– Keystrokes are ignored by the control.
See Also

CWButton.KeyboardMode
KeyboardMode
**CWButtonModes Enumeration**

CWButtonModes are the constants that specify how a CWButton responds to user input.

You can use the following constants with this data type:

- **cwModeIndicator**– The CWButton does not respond to user input. In this mode you can only change the value of the CWButton programmatically.
- **cwModeSwitchUntilReleased**– The value of the CWButton changes when you click the CWButton. The value of the CWButton changes back to its original state when you release the mouse button.
- **cwModeSwitchWhenPressed**– The value of the CWButton changes when you click the CWButton. The value remains unchanged until you click the CWButton again.
**CWShowFocusModes Enumeration**

CWShowFocusModes are the constants for the ShowFocusMode property of the CWPump, CWValve, CWMotor and CWVessel controls.

You can use the following constants with this data type:

- `cwShowFocusControl`–Indicates graphically if the control has the focus.
- `cwShowFocusNone`–Does not indicate graphically if the control has the focus.
CWButtonStyles Enumeration

CWButtonStyles are the constants for the Style parameter of the CWButton.SetBuiltinStyle method.

You can use the following constants with this data type:

- cwButtonStyle3DBitmap–3D custom bitmap button style.
- cwButtonStyle3DCommandOk–3D command button style.
- cwButtonStyle3DHRocker–3D horizontal rocker style.
- cwButtonStyle3DHSlideSwitch–3D horizontal slide switch style.
- cwButtonStyle3DHToggle–3D horizontal toggle style.
- cwButtonStyle3DPushButton–3D push button style.
- cwButtonStyle3DRoundLED–3D round LED style.
- cwButtonStyle3DSquareLED–3D square LED style.
- cwButtonStyle3DToggleOnOff–3D On/Off toggle style.
- cwButtonStyle3DVRocker–3D vertical rocker style.
- cwButtonStyle3DVSSlideSwitch–3D vertical slide switch style.
- cwButtonStyle3DVToggle–3D vertical toggle style.
- cwButtonStyleBitmap–Classic custom bitmap button style.
- cwButtonStyleCommandOk–Classic command button style.
- cwButtonStyleH3dSlide–3D horizontal slide style.
- cwButtonStyleHSlide–Classic horizontal slide style.
- cwButtonStyleHToggle–Classic horizontal toggle style.
- cwButtonStyleRoundLED–Classic round LED style.
- cwButtonStyleRoundPushButton–Classic round push button style.
- cwButtonStyleRoundStarLED–Classic round star LED style.
- cwButtonStyleSquareLED–Classic square LED style.
- cwButtonStyleSquarePushButton–Classic square push button style.
- cwButtonStyleSquareStarLED–Classic square star LED style.
- cwButtonStyleToggleOnOff–Classic On/Off toggle style.
- cwButtonStyleV3dSlide–3D vertical slide style.
- cwButtonStyleVSlide–Classic vertical slide style.
- cwButtonStyleVToggle–Vertical toggle style.
See Also

CWButton.SetBuiltinStyle
CWAlignments Enumeration

CWAlignments are the constants for the CWNumEdit.Alignment and CWCaption.Alignment properties. The CWNumEdit.Alignment property currently uses only cwuiCenter, cwuiLeftJustify, and cwuiRightJustify.

You can use the following constants with this data type:

- cwuiBottomCenter—Bottom-center align.
- cwuiBottomLeft—Bottom-left align.
- cwuiBottomRight—Bottom-right align.
- cwuiCenter—Center.
- cwuiLeftJustify—Left align.
- cwuiRightJustify—Right align.
- cwuiTopCenter—Top-center align.
- cwuiTopLeft—Top-left align.
- cwuiTopRight—Top-right align.
**CWCrosshairStyles Enumeration**

CWCrosshairStyles are the constants for the CWPlot.CrosshairStyle property.

You can use the following constants with this data type:

- `cwCrosshairMajorX`—If the point style is cwPointNone, a solid line is drawn.
- `cwCrosshairMajorXMajorY`—The crosshair has a long horizontal line and a long vertical line.
- `cwCrosshairMajorXMinorY`—The crosshair has a long horizontal line and a short vertical line.
- `cwCrosshairMajorY`—If the point style is cwPointNone, a solid line is drawn.
- `cwCrosshairMinorX`—The crosshair is a short horizontal line.
- `cwCrosshairMinorXMajorY`—The crosshair is a short horizontal line and a long vertical line.
- `cwCrosshairMinorXMinorY`—The crosshair is a short horizontal line and a short vertical line.
- `cwCrosshairMinorY`—The crosshair is a short vertical line.
- `cwCrosshairNone`—There is no crosshair.
**CWPointStyles Enumeration**

CWPointStyles are the constants for the CWCursor.PointStyle and CWPlot.PointStyle properties.

You can use the following constants with this data type:

- `cwPointAsterisk`–Asterisk
- `cwPointBoldCross`–Bold cross
- `cwPointBoldX`–Bold X
- `cwPointCross`–Cross
- `cwPointDottedEmptyCircle`–Dotted empty circle
- `cwPointDottedEmptySquare`–Dotted empty square
- `cwPointDottedSolidCircle`–Dotted solid circle
- `cwPointDottedSolidSquare`–Dotted solid square
- `cwPointEmptyCircle`–Empty circle
- `cwPointEmptyDiamond`–Empty diamond
- `cwPointEmptySquare`–Empty square
- `cwPointEmptySquareWithCross`–Empty square with cross
- `cwPointEmptySquareWithX`–Empty square with X
- `cwPointNone`–None
- `cwPointSimpleDot`–Simple dot
- `cwPointSmallCross`–Small cross
- `cwPointSmallEmptySquare`–Small empty square
- `cwPointSmallSolidSquare`–Small solid square
- `cwPointSmallX`–Small X
- `cwPointSolidCircle`–Solid circle
- `cwPointSolidDiamond`–Solid diamond
- `cwPointSolidSquare`–Solid square
- `cwPointX`–X
See Also

CWCursor::PointStyle
CWPlot::PointStyle
CWChartStyles Enumeration

CWChartStyles are the constants for the CWGraph.ChartStyle property.

You can use the following constants with this data type:

- **cwChartScope**–When the X values for new data reach the end of the display, the plot is cleared and the x axis updates according to its current range. For example, if the x-axis range is 0 to 100, it becomes 100 to 200. No old data is displayed.
- **cwChartStrip**–When a trace reaches the edge of the plot area, the x axis and plot start to scroll from right to left. New data points are appended on the right side while old data scrolls off the left side of the graph.
**CWGraphFrameStyles Enumeration**

CWGraphFrameStyles are the constants for the CWGraph.GraphFrameStyle property.

You can use the following constants with this data type:

- `cwGraphFrame3D`—Specifies a three-dimensional graph frame style.
- `cwGraphFrameClassic`—Specifies a classic graph frame style.


**CWGraphTrackModes Enumeration**

CWGraphTrackModes are the constants for the CWGraph.TrackMode property. You can use the following constants with this data type:

- `cwGTrackAllEvents`—Generates mouse events for all objects in the plot area.
- `cwGTrackDragAnnotation`—Positions annotations with the mouse.
- `cwGTrackDragCursor`—Positions cursors with the mouse.
- `cwGTrackPanPlotAreaX`—Sets the extent of the x axis by dragging the plot area with the mouse.
- `cwGTrackPanPlotAreaXY`—Sets the extent of the x and y axes by dragging the plot area with the mouse.
- `cwGTrackPanPlotAreaY`—Sets the extent of the y axis by dragging the plot area with the mouse.
- `cwGTrackPlotAreaEvents`—Generates mouse events for only the plot area.
- `cwGTrackZoomRectX`—Sets the extent of the x axis by selecting a region with the mouse.
- `cwGTrackZoomRectXY`—Sets the extent of the x and y axes by selecting a region with the mouse.
- `cwGTrackZoomRectY`—Sets the extent of the y axis by selecting a region with the mouse.
CWAnnotationParts Enumeration

CWAnnotationParts are the constants for CWGraph annotation events. You can use the following constants with this data type:

- cwAnnotationAll—Specifies the entire annotation.
- cwAnnotationShape—Specifies the shape part of the annotation.
- cwAnnotationShapePoint—Specifies a point on the shape part of the annotation.
- cwAnnotationText—Specifies the text part of the annotation.
**CWCursorParts Enumeration**

CWCursorParts are the constants for CWGraph cursor events.

You can use the following constants with this data type:

- `cwCursorCrosshair`—Sets cursor near the intersection point.
- `cwCursorXLine`—Sets cursor near the X (vertical) line.
- `cwCursorYLine`—Sets cursor near the Y (horizontal) line.
CWSpeeds Enumeration

CWSpeeds are the constants for the Style parameter of the CWSlide.SetBuiltInStyle method.

You can use the following constants with this data type:

- `cwSpeedFast`–600 ms
- `cwSpeedFastest`–150 ms
- `cwSpeedMedium`–900 ms
- `cwSpeedOff`–Off
- `cwSpeedSlow`–1200 ms
- `cwSpeedSlowest`–1800 ms
- `cwSpeedVeryFast`–300 ms
- `cwSpeedVerySlow`–1500 ms
**CWKnobStyles Enumeration**

CWKnobStyles are the constants for the Style parameter of the CWKnob.SetBuiltInStyle method.

You can use the following constants with this data type:

- `cwKnobStyle3DBottomMeter`– 3D bottom meter style.
- `cwKnobStyle3DDial`– 3D dial style.
- `cwKnobStyle3DKnob`– 3D knob style.
- `cwKnobStyle3DLeftMeter`– 3D left meter style.
- `cwKnobStyle3DRightMeter`– 3D right meter style.
- `cwKnobStyle3DTopMeter`– 3D top meter style.
- `cwKnobStyleDial`– Classic dial style.
- `cwKnobStyleHBottomGauge`– Classic horizontal bottom gauge style.
- `cwKnobStyleHTopGauge`– Classic horizontal top gauge style.
- `cwKnobStyleKnob`– Classic knob style.
- `cwKnobStyleVLeftGauge`– Classic vertical left gauge style.
- `cwKnobStyleVRightGauge`– Classic vertical right gauge style.
See Also

CWKnob.SetBuiltInStyle
**CWAppearances Enumeration**

CWAppearances are the constants for the CWNNumEdit.Appearance property. The CWAppearances enumerations specify if the CWNNumEdit edit box style is two- or three- dimensional.

You can use the following constants with this data type:

- `cwuiAppearanceFlat`– Specifies a flat (two dimensional) control style.
- `cwuiAppearanceThreeD`– Specifies a three-dimensional control style.
**CWBorderStyles Enumeration**

CWBorderStyles are the constants for the CWNumEdit.BorderStyle property.

You can use the following constants with this data type:

- cwuiBorderStyle3D—Specifies a three-dimensional border.
- cwuiBorderStyleFixedSingle—Specifies a fixed-single border.
- cwuiBorderStyleNone—Specifies no border.
**CWNumEditButtonStyles Enumeration**

CWNumEditButtonStyles are the constants for the CWNumEdit.ButtonStyle property.

You can use the following constants with this data type:

- cwuiNumEditButtonStyle3D—Specifies a three-dimensional button style.
- cwuiNumEditButtonStyleClassic—Specifies a classic button style.
**CWPositions Enumeration**

CWPositions are the numeric edit box increment or decrement button positions.

You can use the following constants with this data type:

- cwuiBottom–Bottom
- cwuiLeft–Left
- cwuiRight–Right
- cwuiTop–Top
**CWNumbEditModes Enumeration**

CWNumbEditModes are the constants for the Mode property of the CWNumbEdit control.

You can use the following constants with this data type:

- **cwEdModeControl**–The CWNumbEdit control responds to user input.
- **cwEdModeIndicator**–The CWNumbEdit control does not respond to user input. In this mode you can only change the value of the CWNumbEdit control programmatically.
See Also

CWNumEdit Mode
CWPointerFillStyles Enumeration

CWPointerFillStyles are the constants that specify how a pointer is filled.

You can use the following constants with this data type:

- cwFillNone—No fill is drawn.
- cwFillToGreater—Fills the area between this pointer and the pointer with the next larger value. If there are no pointers with larger values, the system fills the area between this pointer and the maximum value.
- cwFillToFLess—Fills the area between this pointer and the pointer with the next smaller value. If there are no pointers with smaller values, the system fills the area between this pointer and the minimum value.
- cwFillToMax—Fills the area between this pointer and the maximum value.
- cwFillToMin—Fills the area between this pointer and the minimum value.
See Also

CWPointer.FillStyle
CWPointerModes Enumeration

CWPointerModes are the constants that specify how a CWPointer responds to user input.

You can use the following constants with this data type:

- `cwPointerModeControl`–The pointer responds to mouse clicks, clicks on the increment and decrement buttons, and programmatic changes.
- `cwPointerModeControlExact`–The pointer responds to mouse clicks directly on the pointer, mouse clicks on the increment and decrement buttons, and programmatic changes.
- `cwPointerModeControlSmooth`–The pointer responds to mouse clicks anywhere on the control, mouse clicks on the increment and decrement buttons, and programmatic changes. The pointer does not snap to the clicked location as it does when the mode is `cwPointerModeControl`. This mode prevents abrupt changes in the pointer value.
- `cwPointerModeIndicator`–The pointer does not respond to mouse clicks. You can only change the value of the pointer programatically.
- `cwPointerModeMaximum`–The pointer always contains the maximum value that other pointers have had since statistics were reset. The pointer does not respond to user input, and you cannot set its value programmatically.
- `cwPointerModeMean`–The pointer always contains the mean of the values that other pointers have had since statistics were reset. The pointer does not respond to user input, and you cannot set its value programmatically.
- `cwPointerModeMinimum`–The pointer always contains the minimum value that other pointers have had since statistics were reset. The pointer does not respond to user input, and you cannot set its value programmatically.
See Also

CWPPointer Mode
**CWPointerStyles Enumeration**

CWPointerStyles are the constants that specify how a pointer appears graphically.

You can use the following constants with this data type:

- **cwPointer3D**–The pointer appears as a three-dimensional needle. This style is available only on the CWKnob control.
- **cwPointerCustom**–A custom image represents the pointer. This style is available only on the CWSlide control.
- **cwPointerLeftBottomArrow**–The pointer appears as an arrow facing left or to the bottom, depending on whether the CWSlide is vertical or horizontal. This style is available only on the CWSlide control.
- **cwPointerNone**–No pointer image is used.
- **cwPointerNormal**–On a CWKnob control, the pointer appears as a thin line. On a CWSlide control, the pointer appears as a slider "thumb."
- **cwPointerRightTopArrow**–The pointer appears as an arrow facing right or to the top, depending on whether the CWSlide is vertical or horizontal. This style is available only on the CWSlide control.
See Also

CWPointerStyle
**CWRegionAreas Enumeration**

CWRegionAreas are the constants for the CWSHAPE.RegionArea property. You can use the following constants with this data type:

- `cwRegionInverted`—Specifies an inverted region area.
- `cwRegionNormal`—Specifies a normal region area.
See Also

CWShape.RegionArea
CWShapeTypes Enumeration

CWShapeTypes are the constants for the CWShape.Type property. You can use the following constants with this data type:

- cwShapeArc—Specifies an arc.
- cwShapeImage—Specifies an image.
- cwShapeLine—Specifies a line.
- cwShapeMinMaxRegion—Specifies a minimum-maximum region.
- cwShapeNone—Specifies no shape.
- cwShapeOval—Specifies an oval.
- cwShapePoint—Specifies a point.
- cwShapePolygon—Specifies a polygon.
- cwShapeRectangle—Specifies a rectangle.
- cwShapeRegion—Specifies a region.
See Also

CWShape.Type
**CWSlideStyles Enumeration**

CWSlideStyles are the constants for the Style parameter of the CWSlide.SetBuiltinStyle method.

You can use the following constants with this data type:

- cwSlideStyle3DHFillSlide–3D horizontal fill slide style.
- cwSlideStyle3DGraduatedBar–3D horizontal graduated bar style.
- cwSlideStyle3DPointerSlide–3D horizontal pointer slide style.
- cwSlideStyle3DProgressBar–3D horizontal progress bar style.
- cwSlideStyle3DTank–3D tank slide style.
- cwSlideStyle3DThermometer–3D thermometer slide style.
- cwSlideStyle3DVFillSlide–3D vertical fill slide style.
- cwSlideStyle3DGraduatedBar–3D vertical graduated bar style.
- cwSlideStyle3DPointerSlide–3D vertical pointer slide style.
- cwSlideStyle3DProgressBar–3D vertical progress bar style.
- cwSlideStyleH–Classic horizontal 1 slide style.
- cwSlideStyleH2–Classic horizontal 2 slide style.
- cwSlideStyleHBar–Classic horizontal bar slide style.
- cwSlideStylePointer–Classic horizontal pointer slide style.
- cwSlideStyleTank–Classic tank slide style.
- cwSlideStyleThermometer–Classic thermometer slide
- cwSlideStyleV–Classic vertical 1 slide style.
- cwSlideStyleV2–Classic vertical 2 slide style.
- cwSlideStyleVBar–Classic vertical bar slide style.
- cwSlideStyleVPointer–Classic vertical pointer slide style.
See Also

CWSlide.SetBuiltinStyle
CWWValuePairLabels Enumeration

CWWValuePairLabels specify how value pairs appear on an axis.

You can use the following constants with this data type:

- cwVPLabelName–The axis draws the name of the value pairs.
- cwVPLabelNone–The axis does not draw the value pair.
- cwVPLabelValue–The axis draws the value of the value pair.
See Also

CWValuePairs.LabelType
CWValuePairLocations Enumeration

CWValuePairLocations specify where value pairs appear on an axis. You can use the following constants with this data type:

- cwVPLocationIndex—The axis draws the value pairs at their index on the axis.
- cwVPLocationValue—The axis draws the value pairs at their value on the axis.