Overview

The Chart control lets you create a multidimensional chart of two to six variables at a time. Any of the variables can be attached to one of three axes and one of three color ranges. For the Bar and Box type chart, each observation is displayed as an instance of the bar or the box. For the Line and Area types, the observations are linked based on the By variable, which must be a Classification variable. For multiple observations that have the same levels for the independent variables, the response values are summarized using the selected statistics (Sum/Mean). Once created, the object is fully interactive: you can zoom the chart in and out; rotate or tilt it; display, hide, move, or size its legend; manipulate color; and more.
Using Multidimensional Charts

About Multidimensional Charts

Creating a Multidimensional Chart

Specifying Data for the Analysis

Using Data Subsets

Specifying the Chart Type

Specifying Variables and a Viewing Mode

Displaying Titles and Footnotes

Displaying a Legend

Adding Reference Lines

Using Color

Setting the Action Mode

Setting the Animation Mode

Using a Pick List

Setting the Angle of Tilt or Rotation

About Multidimensional Charts

The Chart Control lets you create the following types of multidimensional charts:

- Bar charts, which can analyze up to six variables
- Box charts, which can analyze up to six variables
- Line charts, which can analyze up to three variables (no color variables)
- Area charts, which can analyze up to three variables (no color variables)
The simplest bar chart analyzes two variables in a two-dimensional view. The following bar chart shows a simple chart analyzing weight by age. Each observation in the data set is an instance of the bar.

![Weight by Age Chart](chart.png)

You can analyze a third variable in a two-dimensional bar chart by adding a color variable. Alternatively, rather than represent the third variable's values with color, you can view the chart in three dimensions and specify the third variable on the third axis.

In the following two-dimensional bar chart, the bar color represents the average level of fat for each age.
To analyze four variables, you have to use a three-dimensional view of the chart, specifying three axes variables and one color variable:

The Chart object lets you use up to three color variables. In the following box chart, color is used to represent the average fat, cholesterol, and blood pressure. For Line and Area Charts, the BY variable links the observations.
Creating a Multidimensional Chart

To create a multidimensional chart, open a frame and select a Chart Control from the Components window. This instantiates the object for you. You can name the object by opening the Property sheet and assigning a name on the name attribute.

You can then perform the following tasks in any order:
- Specify the data that contains the variables you want to analyze "Specifying Data for a Multidimensional Chart" on page 223
- Specify the chart type "Specifying the Chart Type for a Multidimensional Chart" on page 224
- Specify the variables and viewing mode for the analysis "Specifying Variables and a Viewing Mode for a Multidimensional Chart" on page 224

Specifying Data for a Multidimensional Chart

To specify the data for a multidimensional chart, use the dataSet or dataSetID attributes, or reference the SCL list that contains the data. Use the chart object’s dataSource attribute to indicate the source of the data.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataSet</td>
<td>Specify the data set in the form libref.filename. You can use this attribute in the property sheet, or in SCL code.</td>
</tr>
<tr>
<td>dataSetID</td>
<td>Specify the data set ID, which is the identifier returned by the open() method. The dataSetID attribute is useful when you want to create a subset of the data &quot;Using Subsets of Data for Multidimensional Graphs&quot; on page 232 before using it. This attribute is only available in SCL code.</td>
</tr>
</tbody>
</table>

For example, the following code sets the dataSet attribute to Sashelp.Revhub:

```plaintext
cart1.dataset = 'sashelp.revhub';
```

In the following example, the data set is first screened for those revenues that are over $500,000:
dsid=open('sashelp.revhub');
    rc=where(dsid,'revenue > 500000');
chart1.dataSource='datasetid';
chart1.dataSetID=dsid;

### Specifying the Chart Type for a Multidimensional Chart

A multidimensional chart object can generate a bar chart, box chart, line chart, and area chart. To specify the chart type, set the `chartType` attribute to one of the values:
- Bar
- Box
- Line
- Area

For example, for a chart object named `chart1`, the following code sets the `chartType` attribute to Bar:

```javascript
chart1.chartType = 'bar';
```

### Specifying Variables and a Viewing Mode for a Multidimensional Chart

A multidimensional chart can analyze from two to six variables. The object supports long variable names, and mixed-case variable names.

You can view the chart in two or three visual dimensions, called the viewing mode. The number of variables you specify must be coordinated with the visual dimensions, which you set with the object's `viewMode` attribute. The viewing mode also affects which actions are available for interacting with the object.

The following table shows how to coordinate the number of variables with the viewing dimensions. The table after it describes the viewing modes.
Using Color in a Multidimensional Chart

The following table lists the attributes you can use to control color in a multidimensional chart. For information on controlling the color in titles and footnotes, see “Displaying Titles and Footnotes in Multidimensional Graphs” on page 233.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Specifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>axisColor</td>
<td>Color for the axis, axis labels, tick marks, and legend.</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>Color outside of the chart axes but within the object’s border</td>
</tr>
<tr>
<td>barColor</td>
<td>Default bar colors</td>
</tr>
<tr>
<td>borderColor</td>
<td>Color for border around the chart object</td>
</tr>
<tr>
<td>borderTitleColor</td>
<td>Text color for the border’s title</td>
</tr>
<tr>
<td>chartBackgroundColor</td>
<td>Color within the chart axes</td>
</tr>
<tr>
<td>colorVariable, color2Variable, color3Variable</td>
<td>Bar colors. See Bar Color in Multidimensional Charts, below.</td>
</tr>
<tr>
<td>faceColorMode</td>
<td>Whether bar colors affect the entire bar or only the bar tops</td>
</tr>
<tr>
<td>gridColor</td>
<td>Color of the chart’s grid lines</td>
</tr>
<tr>
<td>highlightColor</td>
<td>Color a bar turns when it is selected</td>
</tr>
</tbody>
</table>
Attributes | Specifies
---|---
missingColor | Color used to represent missing values in the chart
outlineColor | Color for the outlines around markers

For all the attributes except the color variables, you can specify a SAS color name like red, green, or blue. Alternatively, you can set the color to match a color value that is set in the SAS environment. For example, you can set the color to match the color used for the SAS notes that print in the Log window.

To set colors in the SAS environment, use the SASCOLORS window, which you can open with the Tools menu:

Tools ➤ Options ➤ Colors

For a chart object named chart1, the following code sets the backgroundColor attribute to blue:

```javascript
chart1.backgroundColor = 'blue';
```

### Color Ranges in Multidimensional Charts

By default, the color ranges for bar colors in a multidimensional chart range as follows:

<table>
<thead>
<tr>
<th>Color Variable</th>
<th>Range (low to high)</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorVariable</td>
<td>yellow to Red</td>
</tr>
<tr>
<td>colorVariable2</td>
<td>dark green to light green</td>
</tr>
<tr>
<td>colorVariable3</td>
<td>dark blue to light blue</td>
</tr>
</tbody>
</table>

The colors can represent the Sum, Mean, or Frequency statistic for the analysis variables. To specify the statistic represented by the color range, use the colorVariableStatistic attribute.

To change the color values, you can specify alternative colors on one of the colorList attributes (colorList, color2List, or color3List), or use a color range object on one of the colorRangeObject attributes (colorRangeObject, color2RangeObject, or color3RangeObject). Use the colorSource attribute to indicate the color source.

You can also specify a numeric color variable whose values are indexed into the defined color range to determine the specific color used for each bar. To assign the color variable, use the colorVariable, colorVariable2, and colorVariable3 attributes. You can use the faceColorMode attribute to determine whether the color applies to the entire bar or just the bar tops.

For a chart object named chart1, the following code sets the colorVariable to Revenue, the colorVariableStatistic to Mean, and the faceColorMode to Top:

```javascript
chart1.colorVariable = 'revenue';
chart1.colorVariableStatistic = 'mean';
chart1.faceColorMode = 'top';
```
Setting the Action Mode for Multidimensional Graphs

The multidimensional chart, histogram, pie, and scatter objects are interactive. For example, you can zoom the object in or out to change its displayed size, or rotate it to see a different view of it. To perform simple actions, like Pick and Probe, simply click the object. To perform animation actions, like Zoom and Tilt, click the object and drag it within its border.

Only one action at a time is available, and the action is set on the actionMode attribute. You can set the action in the object’s Property window or in SCL code. For example, for an object named mdViz, the following code sets the actionMode attribute to Zoom:

```javascript
mdViz.actionMode = 'zoom';
```

The following table describes all the actions:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick</td>
<td>Select a bar and add associated data to the pickList attribute. Selection is made by clicking the bar. Selection can be for a single bar, extended selection, or disabled, depending on setting for selectionMode attribute.</td>
</tr>
<tr>
<td>Probe</td>
<td>Display data associated with a bar. The data display is controlled by clicking and holding down the mouse button. As the mouse moves over different bars, the display changes.</td>
</tr>
<tr>
<td>Move</td>
<td>Move the chart within its borders. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>Rotate</td>
<td>Rotate the chart around the vertical axis. Movement is controlled by horizontal mouse movement during a click and drag.</td>
</tr>
<tr>
<td>Tilt</td>
<td>Tilt the chart around the horizontal axis. Movement is controlled by vertical mouse movement during a click and drag.</td>
</tr>
<tr>
<td>Spin</td>
<td>Change both the tilt and the rotation angles. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>Zoom</td>
<td>Zooms object in or out. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>Legend</td>
<td>Moves legend, or changes legend size. To move, click in the legend center and drag. To size, click just inside border you want to size, and then drag.</td>
</tr>
<tr>
<td>ScrollLegend</td>
<td>Scrolls legend values if they can’t all be displayed at once. Movement is controlled by vertical mouse movement during a click and drag.</td>
</tr>
<tr>
<td>DataMove</td>
<td>Scrolls the data within the axes. Only works for ViewMode=2D or 3D. For 3D, scrolls only in X-Axis. Also, only works after zooming the data using DataZoom or ViewPort. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>DataZoom</td>
<td>Zooms into the data to see a subset of the bars more clearly. After a DataZoom, you can use DataMove. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>Viewport</td>
<td>Zooms into part of the data by dragging a rectangle around it. After a Viewport, you can use DataMove. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
</tbody>
</table>
CubeRotate | Rotates each box around its own center. Available only for ChartType=BOX with ViewMode=3DPerspective. Movement is controlled by clicking and dragging the mouse.
---|---
Refline | Move a reference line. Movement is controlled by clicking and dragging the mouse. Available only at runtime. Not available for Pie Charts.

The actions that are available depend on the type of object, and how many data dimensions and viewing dimensions are set for it. For example, you can rotate or tilt multidimensional charts, but it only makes sense to do so when the chart’s viewMode is 3D, which provides three true data dimensions. If the viewMode is set to 3D, the actions Tilt, Rotate, and Spin are not possible. However, rather than generate an error or be ignored, each action simply changes the object’s projectionAngle attribute, which determines the angle of the projection that provides the illusion of a third dimension.

When you perform an animation action (Move, Rotate, Tilt, Spin, or Zoom), the object is displayed either as a wire frame or a filled object. This animation mode affects performance while the user interacts with the object. To control the animation mode, see “Setting the Animation Mode for Multidimensional Graphs” on page 228.

### Setting the Angle of Tilt or Rotation for Multidimensional Graphs

For multidimensional chart, histogram, and scatter objects, the object’s rotation angle is its current angle relative to the Y axis. Similarly, its tilt angle is its angle relative to the X axis.

To set a rotation angle or tilt angle, use the object’s rotationAngle or tiltAngle attributes. For example, for an object named mdViz, the following code sets the rotationAngle attribute to 35 degrees:

```javascript
mdViz.rotationAngle = 35;
```

You can also use the object’s actionMode attribute to adjust these angles interactively. Set the actionMode to Tilt to adjust the tiltAngle, or Rotate to adjust the rotationAngle (see “Setting the Action Mode for Multidimensional Graphs” on page 227). To set both angles at once, set the actionMode to Spin.

### Setting the Animation Mode for Multidimensional Graphs

The multidimensional chart, histogram, pie, and scatter objects have an actionMode attribute, which determines the action taken when the user interacts with the object. For information on setting the mode, see “Setting the Action Mode for Multidimensional Graphs” on page 227.

Some of the actions are animation actions (Move, Rotate, Tilt, Spin, or Zoom), which means the object moves or changes appearance as you drag the object within its border. For example, with the Zoom action, the object is zoomed in or out within its border.

During the animation action, you can display the object as an empty wire frame, or as an object filled with its assigned colors and shading. The wire frame yields better performance, but the filled object looks better and provides better analysis capabilities while the action is in progress.
To set this animation mode, use the animationMode attribute. Valid values for animateMode are Wireframe and Filled. For example, for an object named mdViz, the following code sets the animateMode attribute to Filled:

```javascript
mdViz.animateMode = 'filled';
```

### Specifying Color Ranges in Multidimensional Graphs

By default, the multidimensional histogram, pie, and scatter objects use colors that range from yellow to red, where red represents the highest values. Multidimensional charts can use up to three color ranges: yellow to red, light green to dark green, and light blue to dark blue.

You can modify the color range(s) in either of two ways:
- Define a different list of colors for the object to use. Do so on the colorList attribute.
- Create a color range object that applies color to specific ranges in a response variable's values. Name the object on the colorRangeObject attribute.

The object's colorSource attribute indicates which technique is being used.

### Using a List of Colors for Multidimensional Graphs

You can specify a list of colors for a multidimensional graph to use. To do so, define an SCL list of color values and assign the list to the object's colorList attribute. With this technique, be sure the object's colorSource attribute is set to colorList.

In build mode, the object's property sheet automatically opens a List Editor so you can define the SCL list.

In SCL code, you can assign the color list by first making an SCL list, populating it with a list of color values, and then assigning the list to the object's colorRange attribute.

For an object named mdViz, the following code defines a short colors list and assigns it to the object's colorRange attribute:

```javascript
MyColors = makelist(4);
   rc=insertc(mycolors, 'blue');
   rc=insertc(mycolors, 'red');
   rc=insertc(mycolors, 'yellow');
   rc=insertc(mycolors, 'green');
mdViz.colorSource = 'range';
mdViz.colorRange = myColors;
```

### Using a Pick List with Multidimensional Graphs

The multidimensional chart, histogram, pie, and scatter objects have an actionMode attribute that determines what action is taken when you click on the object. For information on setting the action mode, see “Setting the Action Mode for Multidimensional Graphs” on page 227.

One of the actions is Pick, which causes the object to store the clicked element's values in the object's pickList attribute. The pickList attribute contains an SCL list of lists, any of which can be a named list.
For example, assume a pie chart object’s actionMode attribute is set to Pick, and that the pie represents the population figures for a set of countries. If you click on the pie slice representing France, the pickList attribute stores the value identifying the country as France, and it stores France's population value. If, pressing the multiple selection key on the keyboard, you click on France, England, and Germany, the pickList attribute's list stores three lists; the first list identifies France and its population, the second identifies England and its population, and the third identifies Germany and its population.

To get the items from the pickList attribute, use the object’s _getPickList method. For an object named mdViz, the following code gets items from the object’s pickList attribute:

```python
myList = makelist();
mdViz._getPickList(myList);
```

To set the items in the pickList attribute, populate an SCL list with the values and assign the list to the pickList attribute:

```python
myList = makelist();
rc=insertn(myList, 100);
rc=insertn(myList, 200);
mdViz.pickList = myList;
```

---

**Displaying a Legend in Multidimensional Graphs**

The multidimensional chart, histogram, pie, and scatter objects have let you display a legend. To do so, set the showLegend attribute to Yes. For an object named mdViz, the following code displays a legend:

```python
mdViz.showLegend='yes';
```

For the legend to display, you must be using color in the graph to represent a response variable's values.

There are two types of legends: bar and list. Use the legendType attribute to specify either a Bar or a List legend:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar</td>
<td>Represents colors as a bar of graduated colors. Available when colorSource attribute is set to colorList or colorRangeObject.</td>
</tr>
<tr>
<td>List</td>
<td>Lists the color used to represent each value from the response variable. Available only if colorSource attribute is set to colorRangeObject.</td>
</tr>
</tbody>
</table>

The information displayed for the legend depends on the legend's size. If the size is large enough to display legend values and labels without overlap, the values and labels are displayed. If the legend size is too small to display the values or labels, they are automatically hidden so that the legend does not look cluttered.
Scrolling a Legend

If legendType is List and a legend's current size isn't large enough to display all legend values, you can scroll the legend values interactively. To do so, set the graph object's actionMode attribute to scrollLegend. You can then click in the legend, and drag the mouse up or down to scroll to the desired values.

Moving and Sizing a Legend Interactively

You can interactively move a legend or change its size. To do so, you must set the graph object's showLegend attribute to Yes, and its actionMode attribute to Legend.

To move the legend, click on the middle of the legend, and then drag the legend to the desired location within the graph object's border.

To change the legend's size, click just inside the legend border that you want to size, and then enlarge or reduce the size. For example, to make the legend taller, click on the legend's top border and drag the border up.

Adding Reference Lines to Multidimensional Graphs

The multidimensional chart, histogram, and scatter objects let you emphasize a particular value on an axis. To do so, you can draw a reference line from that value across the axis area. You can add as many reference lines as you like to a graph.

Reference lines are interactive within the object. They have a square box at one end of the line, which you can drag toward either end of the axis when the actionMode attribute is set to Refline. The reference lines also have a label.

You can create an object's reference lines as restricted or unrestricted. Restricted reference lines cannot cross each other, whereas unrestricted lines can cross. For example, assume you have two reference lines on an axis, and that the lines are positioned at data values 10 and 35. If the reference lines are restricted, you cannot drag the line currently at value 35 to a position below 10 in the data, because it would require the line to cross the line at 10. If the lines are unrestricted, you can drag the line at 35 to a position below 10.

To create a reference line, use the object's _addReferenceLine method, which takes the following arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis</td>
<td>Axis that will contain the reference line. This can be the X, Y, or Z axis. The axis variable must be numeric.</td>
</tr>
<tr>
<td>refID</td>
<td>Reference line ID. A unique ID, which is returned by the method. Use this ID to identify the reference line when you want to query or delete the line.</td>
</tr>
<tr>
<td>vType</td>
<td>Value type. Specify Value to position the reference line according to data values. Specify Location to position the reference line according to a percentage of the length of the assigned axis.</td>
</tr>
</tbody>
</table>
### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>The reference line's initial position on the assigned axis. If the vType parameter is Value, the position must be a data value; for example, specify 5 to position the line at data value 5. If the vType parameter is Location, the position must be a percentage; for example, specify 0.5 to position the line in the middle of the assigned axis. The location value should be between 0 and 1.</td>
</tr>
<tr>
<td>label</td>
<td>A label to display above or next to the reference line to identify it for the user. For example, you might label it with the data value at the line's current position.</td>
</tr>
</tbody>
</table>

To determine whether the reference lines are restricted or unrestricted, set the referenceLineRestriction attribute to Yes to impose a restriction, or No not to impose the restriction.

For an object named mdViz, the following code adds a reference line to the middle of the Y axis:

```javascript
mdViz.referenceLineRestriction = 'yes';
mdViz._addReferenceLine('y', xRefId, 'location', 0.5, 'myLabel');
```

The following table shows the additional methods you can use to communicate with a reference line.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deleteReferenceLine</td>
<td>deletes a reference line</td>
</tr>
<tr>
<td>_getReferenceLineValue</td>
<td>gets the data value for the reference line's current position</td>
</tr>
<tr>
<td>_setReferenceLineLabel</td>
<td>sets a label for the reference line</td>
</tr>
<tr>
<td>_setReferenceLineValue</td>
<td>specifies the data value to use for setting the reference line's position. The value cannot be a location percentage.</td>
</tr>
</tbody>
</table>

---

### Using Subsets of Data for Multidimensional Graphs

To create subsets of the data for a multidimensional chart, histogram, pie, or scatter object, you can:

- Use the object’s dataSetID attribute when specifying data for the analysis. This technique lets you use a WHERE clause to create a data subset before displaying a graph. For more information on using this technique for one of the objects, see the help for that object.

- Provide reference lines so the user can interactively position the reference line at data points of interest. You can then query the reference lines to get their position, and create a data subset using those data values. (Reference lines are not available for pie charts.) For information on using reference lines, see “Adding Reference Lines to Multidimensional Graphs” on page 231.

Because you can query a reference line for its position, you can create an application that creates data subsets interactively. In this application, the user can drag the
reference lines to encompass the data range of interest. You can then query each line to determine its position, and generate a graph that uses those position values as the minimum and maximum values in the analysis.

For example, assume you have added two reference lines to the object mdviz, and that you stored their ID's in the variables refLine1 and refLine2. The following code queries the lines for their values and generates a histogram whose lowest data value is the value from line refLine1, and whose highest value is the value from line refLine2:

```plaintext
lowVal=0;
highVal=0;
mdviz._getReferenceLineValue(refLine1, lowVal);
mdviz._getReferenceLineValue(refLine2, highVal);
/* open a data set and use the reference-line */
/* values as the low and high values */
dsid=open('sasuser.mydata');
rc=where(dsid, 'x>lowVal','and x<highVal');
/* assume hist1 is an existing histogram object */
hist1.dataSource='dsid';
hist1.dataSetID=dsid;
```

---

### Displaying Titles and Footnotes in Multidimensional Graphs

The multidimensional chart, histogram, pie, and scatter objects use title and footnote objects to control their titles and footnotes. You can specify

- up to four titles using the attributes `title1`, `title2`, `title3`, and `title4`.
- up to two footnotes using attributes `footnote1` and `footnote2`.

Each title and footnote object has the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>Color for the title or footnote text</td>
</tr>
<tr>
<td>font</td>
<td>Font for the title or footnote text</td>
</tr>
<tr>
<td>text</td>
<td>The text for the title or footnote</td>
</tr>
<tr>
<td>justification</td>
<td>left, right, or center justification</td>
</tr>
</tbody>
</table>

Because the titles and footnotes are objects, they inherit additional attributes, which you will see in their property sheets. However, the attributes listed in the table above are the only ones you need.

When you set the title or footnote attributes in the chart, histogram, pie, or scatter object’s property sheet, an editor automatically opens so you can enter values for the title or footnote attributes.

To set titles or footnotes in SCL code, use the notation for complex attributes. For example, for a chart object named `chart1`, the following code sets the text and text color for one title and one footnote:

```plaintext
chart1.title1.text='My Title';
chart1.title1.color='red';
chart1.footnote1.text='My Footnote';
chart1.footnote1.color='blue';
```
Attributes

Attributes specified for the Chart Control class are described here.

Note: The following attributes are inherited from Widget class and appear in the Properties window in build mode, but they are not used by the Chart control: attachedInterface, CBTFramename, contentsUpdatedAttributes, help, helpText, model, and toolTipText.

Public Attributes
"viewportStackDepth "on page 260

Dictionary

XAxisLabelText

Returns or sets the label text for the X axis
Type: Character
Initial Value: (Object)
Valid Values:
Category: Appearance

XAxisValueLength

Returns or sets tick mark size for the X axis
Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

XDataMax

Returns or sets the maximum value for the X-axis
**Type:** Numeric  
**Initial Value:** (Object)  
**Valid Values:**  
**Category:** Data

---

**XDataMin**

Returns or sets the minimum value for the X-axis

**Type:** Numeric  
**Initial Value:** (Object)  
**Valid Values:**  
**Category:** Data

---

**XScale**

Returns or sets the X scaling value for changing the chart size

**Type:** Numeric  
**Initial Value:** (Object)  
**Valid Values:**  
**Category:** Appearance

---

**XVariable**

Returns or sets the analysis variable for the X axis

**Type:** Character  
**Initial Value:** (Object)  
**Valid Values:** \sashelp.dasses.chartvalues.sd  
**Category:** Data
YAxisLabelText

Returns or sets the label text for the Y axis
Type: Character
Initial Value: (Object)
Valid Values:
Category: Appearance

YAxisMode

Returns or sets the Y axis mode
Type: Character
Initial Value: (Object)
Valid Values: Relative Absolute
Category: Appearance

YAxisValueSpacing

Returns or sets tick mark size for the Y axis
Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

YDataMax

Returns or sets the maximum value for the Y axis
YVariable

Returns or sets the variable for the Y axis

Type: Character
Initial Value: (Object)
Valid Values: \sashelp.classes.chartvalues.scl
Category: Data
YVariableStatistic

Returns or sets the Y variable statistic
Type: Character
Initial Value: (Object)
Valid Values: Sum Mean Freq
Category: Data

YVariableType

Returns or sets the variable type for the Y axis
Type: Character
Initial Value: (Object)
Valid Values: Numerical Categorical
Category: Data

ZAxisLabelText

Returns or sets the label text for the Z axis
Type: Character
Initial Value: (Object)
Valid Values: 
Category: Appearance

ZDataMax

Returns or sets the maximum value for the Z axis
ZVariable

Returns or sets the variable for the Z axis

Type: Character
Initial Value: (Object)
Valid Values: \sashelp.classes.chartvalues.scl
Category: Data

ZDataMin

Returns or sets the minimum value for the Z axis

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Data

ZScale

Returns or sets the Z scaling value for changing the chart size

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance
actionMode

**Returns or sets the action mode**

**Type:** Character  
**Initial Value:** (Object)  
**Valid Values:** Pick Probe Move Rotate Tilt Spin Zoom Legend ScrollLegend DataMove DataZoom Viewport CubeRotate Refline  
**Category:** Appearance

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick</td>
<td>Select a bar and add associated data to the pickList attribute. Selection is made by clicking the bar. Selection can be for a single bar, extended selection, or disabled, depending on setting for selectionMode attribute.</td>
</tr>
<tr>
<td>Probe</td>
<td>Display data associated with a bar. The data display is controlled by clicking and holding down the mouse button. As the mouse moves over different bars, the display changes.</td>
</tr>
<tr>
<td>Move</td>
<td>Move the chart within its borders. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>Rotate</td>
<td>Rotate the chart around the vertical axis. Movement is controlled by horizontal mouse movement during a click and drag.</td>
</tr>
<tr>
<td>Tilt</td>
<td>Tilt the chart around the horizontal axis. Movement is controlled by vertical mouse movement during a click and drag.</td>
</tr>
<tr>
<td>Spin</td>
<td>Change both the tilt and the rotation angles. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>Zoom</td>
<td>Zooms object in or out. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>Legend</td>
<td>Moves legend, or changes legend size. To move, click in the legend center and drag. To size, click just inside border you want to size, and then drag.</td>
</tr>
<tr>
<td>ScrollLegend</td>
<td>Scrolls legend values if they can't all be displayed at once. Movement is controlled by vertical mouse movement during a click and drag.</td>
</tr>
<tr>
<td>DataMove</td>
<td>Scrolls the data within the axes. Only works for ViewMode=2D or 3D. For 3D, scrolls only in X-Axis. Also, only works after zooming the data using DataZoom or ViewPort. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>DataZoom</td>
<td>Zooms into the data to see a subset of the bars more clearly. After a DataZoom, you can use DataMove. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>Viewport</td>
<td>Zooms into part of the data by dragging a rectangle around it. After a Viewport, you can use DataMove. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CubeRotate</td>
<td>Rotates each box around its own center. Available only for ChartType=BOX with ViewMode=3DPerspective. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
<tr>
<td>Refline</td>
<td>Move a reference line. Movement is controlled by clicking and dragging the mouse.</td>
</tr>
</tbody>
</table>

**animateMode**

Returns or sets the animation mode

Type: Character
Initial Value: (Object)
Valid Values: Filled Wireframe
Category: Appearance

**autoScale**

Returns or sets the auto scale state, which determines whether chart size is automatically scaled to fit within its borders

Type: Character
Initial Value: (Object)
Valid Values: On Off
Category: Appearance

**axisColor**

Returns or sets the tick-mark, axis, label, and legend color

Type: Character
Initial Value: (Object)
Valid Values: 
Category: Appearance
axisLabelFont

Returns or sets the axis-label font

Type: List
Initial Value: (Object)
Valid Values:
Category: Appearance

barColor

Returns or sets the default color for bars

Type: Character
Initial Value: (Object)
Valid Values:
Category: Appearance

chartBackgroundColor

Returns or sets the chart background color

Type: Character
Initial Value: (Object)
Valid Values:
Category: Appearance

chartStyle

Returns or sets the chart style, which determines chart appearance when the chart is not in motion

Type: Character
**chartType**

Returns or sets the chart type

Type: Character
Initial Value: (Object)
Valid Values: Bar Box Line Area
Category: Appearance

**color2List**

Returns or sets the color2 range for data mapping

Type: List
Initial Value: (Object)
Valid Values:
Category: Appearance

**color2RangeObject**

Returns or sets the color2 range object

Type: Character
Initial Value: (Object)
Valid Values:
Category: Data
color2Variable

Returns or sets the color2 variable

Type: Character
Initial Value: (Object)
Valid Values: \sashelp.classes.chartvalues.scl
Category: Data

color2VariableStatistic

Returns or sets the color2 variable statistic

Type: Character
Initial Value: (Object)
Valid Values: Sum Mean Freq
Category: Data

color2VariableType

Returns or sets the color2 variable type

Type: Character
Initial Value: (Object)
Valid Values: Numerical Categorical
Category: Data

color3List

Returns or sets the color3 range for data mapping

Type: List
color3RangeObject

Returns or sets the color3 range object

Type: Character
Initial Value: (Object)
Valid Values:
Category: Data

color3Variable

Returns or sets the color3 variable

Type: Character
Initial Value: (Object)
Valid Values: \sashelp.classes.chartvalues.scl
Category: Data

color3VariableStatistic

Returns or sets the color3 variable statistic

Type: Character
Initial Value: (Object)
Valid Values: Sum Mean Freq
Category: Data
color3VariableType

Returns or sets the color3 variable type
Type:  Character
Initial Value:  (Object)
Valid Values:  Numerical Categorical
Category:  Data

colorList

Returns or sets the color range for data mapping
Type:  List
Initial Value:  (Object)
Valid Values:
Category:  Appearance

colorRangeObject

Returns or sets the color range object
Type:  Character
Initial Value:  (Object)
Valid Values:
Category:  Data

The Chart control does not support the color attributes (Blinking, Highlight, Reverse, Underline, HiRev, and None) that are available on a colorRangeObject.

colorSource

Returns or sets the color source
Type:  Character
Chart Control

### colorVariable

Returns or sets the color variable

- **Type:** Character
- **Initial Value:** (Object)
- **Valid Values:** ColorList, ColorRangeObject
- **Category:** Appearance

### colorVariableStatistic

Returns or sets the color variable statistic

- **Type:** Character
- **Initial Value:** (Object)
- **Valid Values:** Sum, Mean, Freq
- **Category:** Data

### colorVariableType

Returns or sets the color variable type

- **Type:** Character
- **Initial Value:** (Object)
- **Valid Values:** Numerical, Categorical
- **Category:** Data
cubeXRotation

Returns or sets the cube X rotation value
Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

cubeYRotation

Returns or sets the cube Y rotation value
Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

cubeZRotation

Returns or sets the cube Z rotation value
Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

dataSet

Returns or sets the dataset to be charted
Type: Character
Initial Value: (Object)
Valid Values:
Category: Data

**dataSetID**

Returns or sets the ID value for the dataset to be charted

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Data

**dataSource**

Returns or sets the data source

Type: Character
Initial Value: (Object)
Valid Values: Dataset, DatasetID
Category: Data

**defaultAttribute**

Returns or sets the attribute name used for the object’s default attribute

Type: Character
Initial Value: (Object)
Valid Values: \sashelp.classes.defaultattributevalues.scl
Category: Data
**dragInfo**

Returns or sets the information that is transferred from the control when a drag operation occurs

- **Type:** List  
- **Initial Value:** (Object)  
- **Valid Values:** None  
- **Category:** Drag & Drop

**dropInfo**

Returns or sets the information that defines the data representations that can be dropped on the control

- **Type:** List  
- **Initial Value:** (Object)  
- **Valid Values:** None  
- **Category:** Drag & Drop

**faceColorMode**

Returns or sets face color mode state, which determines whether color is applied to all chart surfaces, or just the top of the chart

- **Type:** Character  
- **Initial Value:** (Object)  
- **Valid Values:** Top All  
- **Category:** Appearance

**footnote1**

Returns or sets the footnote1 attributes
footnote2

Returns or sets the footnote2 attributes

Type: SASHELP.CLASSES.CTEXTLABEL_C класс
Initial Value: (Object)
Valid Values:
Category: Appearance

gridColor

Returns or sets the color of the grid that is formed if tick marks are extended through the chart area

Type: Character
Initial Value: (Object)
Valid Values:
Category: Appearance

groupByAxis

Returns or sets group by axis state

Type: Character
Initial Value: (Object)
Valid Values: X/Z
Category: Appearance
**highlightColor**

Returns or sets the selection color used when bars are selected with actionMode set to Pick

- **Type:** Character
- **Initial Value:** (Object)
- **Valid Values:**
- **Category:** Appearance

---

**legendFont**

Returns or sets the legend font

- **Type:** List
- **Initial Value:** (Object)
- **Valid Values:**
- **Category:** Appearance

---

**legendType**

Returns or sets the legend type

- **Type:** Character
- **Initial Value:** (Object)
- **Valid Values:** Bar List
- **Category:** Appearance

---

**missingColor**

Returns or sets the color used to represent missing values
**outlineColor**

Returns or sets the bar outline color

<table>
<thead>
<tr>
<th>Type:</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value:</td>
<td>(Object)</td>
</tr>
<tr>
<td>Valid Values:</td>
<td></td>
</tr>
<tr>
<td>Category:</td>
<td>Appearance</td>
</tr>
</tbody>
</table>

**pickHighlightMode**

Returns or sets the pick highlight mode, which determines whether the outline or the entire bar is colored upon selection

<table>
<thead>
<tr>
<th>Type:</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value:</td>
<td>(Object)</td>
</tr>
<tr>
<td>Valid Values:</td>
<td>Outline Filled</td>
</tr>
<tr>
<td>Category:</td>
<td>Appearance</td>
</tr>
</tbody>
</table>

**pickList**

Returns or sets the picklist, which stores data associated with a selected bar with actionMode set to Pick

<table>
<thead>
<tr>
<th>Type:</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value:</td>
<td>(Object)</td>
</tr>
<tr>
<td>Valid Values:</td>
<td></td>
</tr>
<tr>
<td>Category:</td>
<td>Appearance</td>
</tr>
</tbody>
</table>
projectionAngle

Returns or sets projection angle, which is the angle of the projected surface when viewMode is set to 3D

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

Valid values for the projectionAngle are between 30 and 150 degrees. If you set a value outside of this range, the closest value within the range is used instead.

referenceLineRestriction

Returns or sets the reference line mode

Type: Character
Initial Value: (Object)
Valid Values: No Yes
Category: Appearance

rotationAngle

Returns or sets the angle to rotate around the vertical axis when viewMode is set to 3DPerspective. If viewMode is not 3DPerspective, rotation isn't possible and this attribute is ignored.

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

Valid values for the rotationAngle are between -180 and +180 degrees. If you set a value outside of this range, the closest value within the range is used instead.
scale

Returns or sets the scale value for changing the size of the chart

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

selectionMode

Returns or sets the selection mode state, which determines whether multiple selections are allowed when actionMode is set to Pick

Type: Character
Initial Value: (Object)
Valid Values: None/Single/Extended
Category: Appearance

showGrid

Shows or hides the grid that is formed if tick marks are extended through the chart area

Type: Character
Initial Value: (Object)
Valid Values: Yes No
Category: Appearance

showLegend

Determines whether the legend is displayed

Type: Character
**showLegendValue**

Returns or sets the display status of selected data values in the legend

Type: Character

Initial Value: (Object)

Valid Values: Yes No

Category: Appearance

**sortMode**

Returns or sets the sort type

Type: Character

Initial Value: (Object)

Valid Values: None FormattedAscending UnformattedAscending

Category: Appearance

**tickmarkFont**

Returns or sets the tick mark font

Type: List

Initial Value: (Object)

Valid Values:

Category: Appearance
**tiltAngle**

Returns or sets the angle to tilt around the horizontal axis when viewMode is set to 3DPerspective. If viewMode is not 3DPerspective, tilting isn’t possible and this attribute is ignored.

Type: Numeric

Initial Value: (Object)

Valid Values:

Category: Appearance

Valid values for the tiltAngle are between –180 and 0 degrees. If you set a value outside of this range, the closest value within the range is used instead.

**title1**

Returns or sets the title1 attributes

Type: SASHELP.CLASSES.CTEXTLABEL_C.CLASS

Initial Value: (Object)

Valid Values:

Category: Appearance

**title2**

Returns or sets the title2 attributes

Type: SASHELP.CLASSES.CTEXTLABEL_C.CLASS

Initial Value: (Object)

Valid Values:

Category: Appearance

**title3**

Returns or sets the title3 attributes
title4

Returns or sets the title4 class attributes

Type: SASHELP.CLASSES.CTEXTLABEL_C.CLASS
Initial Value: (Object)
Valid Values:
Category: Appearance

viewMode

Returns or sets the view mode

Type: Character
Initial Value: (Object)
Valid Values: 2D 3D 3D perspective
Category: Appearance

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>Height and width</td>
</tr>
<tr>
<td>3D</td>
<td>Height, width, and the illusion of depth</td>
</tr>
<tr>
<td>3D perspective</td>
<td>Height, width, and true depth</td>
</tr>
</tbody>
</table>

viewpointDistance

Returns or sets the distance of the eye point from the viewpoint

Type: Numeric
viewpointX

Returns or sets the X coordinate viewpoint location

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

viewpointY

Returns or sets the Y coordinate viewpoint location

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

viewpointZ

Returns or sets the Z coordinate viewpoint location

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance
**viewportStackDepth**

Returns the number of active viewports on the viewport stack

- **Type:** Numeric
- **Initial Value:** 0
- **Valid Values:**
- **Category:** Appearance

The viewportStackDepth attribute is read-only at runtime; you cannot set it, and it is not used in build mode. At runtime, viewportStackDepth is automatically incremented when the actionMode attribute is set to Viewport and you zoom in on part of the data by dragging a rectangle around it. Each time you drag a rectangle, viewportStackDepth is incremented by 1.

To return to the previous viewport, use the _popPreviousViewport method.

**Methods**

Methods specified for the Chart Control class are described here.

**Public Methods**

**_addReferenceLine**

Create a reference line

**Syntax**

```
objectName_addReferenceLine( axis, refid, vtype, value, label );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis</td>
<td>Character</td>
<td>I</td>
<td>refline axis. The axis variable must be numeric.</td>
</tr>
<tr>
<td>refid</td>
<td>Numeric</td>
<td>O</td>
<td>unique ID returned for the reference line. Used to query or delete the line</td>
</tr>
<tr>
<td>Argument</td>
<td>Type</td>
<td>Use</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>-----</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>vtype</td>
<td>Character</td>
<td>I</td>
<td>indicates whether the value parameter specifies a data value or a location on the axis. Use keyword Value for data values, or keyword Location for a location given as a percentage of the distance along the axis.</td>
</tr>
<tr>
<td>value</td>
<td>Numeric</td>
<td>I</td>
<td>specifies the reference line's initial position on the axis. If the vtype parameter is set to Value, use a data value to position the line; for example, specify 5 to position the line at axis value 5. If the vtype parameter is set to Location, use a percentage to position the line; for example, specify .5 to position the line in the middle of the axis. Location values are expected to range from 0 to 1.</td>
</tr>
<tr>
<td>label</td>
<td>Character</td>
<td>I</td>
<td>label to display for the reference line to identify it for the user.</td>
</tr>
</tbody>
</table>

_binit

Initializes an object at design time

Syntax

```c
objectName_binit( );
```
**_deleteReferenceLine_**

Delete a reference line

----------

**Syntax**

objectName_deleteReferenceLine( refId );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>refId</td>
<td>Numeric</td>
<td>I</td>
<td>refId for the reference line to be deleted</td>
</tr>
</tbody>
</table>

**_getPickList_**

Return the picklist

----------

**Syntax**

objectName_getPickList( picklist );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pickList</td>
<td>List</td>
<td>I</td>
<td>list of values stored when chart element was selected with actionMode set to Pick.</td>
</tr>
</tbody>
</table>

**_getPickedItemCount_**

Return the number of items in the picklist

----------

**Syntax**

objectName_getPickedItemCount( pickitems );
**Argument Type Use Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pickitems</td>
<td>Numeric</td>
<td>0</td>
<td>number of values stored when chart element was selected with actionMode set to Pick.</td>
</tr>
</tbody>
</table>

**_getReferenceLineValue**

Return a reference line value

**Syntax**

`objectName_getReferenceLineValue( refId, value );`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>refId</td>
<td>Numeric</td>
<td>I</td>
<td>refId of the reference line whose value will be returned</td>
</tr>
<tr>
<td>value</td>
<td>Numeric</td>
<td>0</td>
<td>the data value at the reference line's current position</td>
</tr>
</tbody>
</table>

**_init**

Initializes an object

**Syntax**

`objectName_init( );`
_popPreviousViewport

Set current viewport to previous viewport

Syntax

objectName_popPreviousViewport( );

_print

print the chart

Syntax

objectName_print( );

_resetCubes

Set the cubes back to the default position

Syntax

objectName_resetCubes( );

_resetView

Set the view back to the default setting
Syntax

```
oBJECT_NAME._resetView( );
```

=setBarColors

Syntax

```
return =เอเชxM_NAME.setBarColors( );
```

=setReferenceLineLabel

**Set the reference line label**

Syntax

```
objectM_NAME.setReferenceLineLabel( refId,label );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>refId</td>
<td>Numeric</td>
<td>I</td>
<td>refId of the reference line whose label will be set</td>
</tr>
<tr>
<td>label</td>
<td>Character</td>
<td>I</td>
<td>the label to assign to the reference line</td>
</tr>
</tbody>
</table>

=setReferenceLineValue

**Set a reference line value**
### Syntax

```
objectName._setReferenceLineValue(refid, value);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>refId</td>
<td>Numeric</td>
<td>I</td>
<td>refId of the reference line whose value will be set</td>
</tr>
<tr>
<td>value</td>
<td>Character</td>
<td>I</td>
<td>the data value to use for setting the reference line's position. The value cannot be a location percentage. To set a location, delete the reference line and add a new one with _addReferenceLine.</td>
</tr>
</tbody>
</table>

---

### _setcamShowLegendValue

determines whether legend values are displayed

```
objectName._setcamShowLegendValue(arg1);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>Character</td>
<td>I</td>
<td></td>
</tr>
</tbody>
</table>

---

### _setcamActionMode

sets the chart’s action mode

```
objectName._setcamActionMode(mode);
```
_setcamAnimateMode

the chart’s animation mode

Syntax

objectName_setcamAnimateMode( mode );

Argument | Type    | Use | Description |
----------|---------|-----|-------------|
mode      | Character | I   | the animation mode to set on the animationMode attribute |

_setcamAutoScale

sets the autoScale attribute

Syntax

objectName_setcamAutoScale( arg1 );

Argument | Type    | Use | Description |
----------|---------|-----|-------------|
ARG1      | Character | I   | Specify Yes or No |
**_setcamAxisColor_**

sets the color of the axis

**Syntax**

```
objectName_setcamAxisColor(color);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>Character</td>
<td>I</td>
<td>the color to use for the axis</td>
</tr>
</tbody>
</table>

**_setcamAxisLabelFont_**

sets the font for the axis label

**Syntax**

```
objectName_setcamAxisLabelFont(font);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>font</td>
<td>List</td>
<td>I</td>
<td>SCL list defining the font</td>
</tr>
</tbody>
</table>

**_setcamBarColor_**

Specify the default color for bars

**Syntax**

```
objectName_setcamBarColor(color);
```
_setcamChartBackgroundColor

Sets the color for the chart background

Syntax

objectName_setcamChartBackgroundColor( color );

Argument Type Use Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>Character</td>
<td>I</td>
<td>the color to set for the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>background</td>
</tr>
</tbody>
</table>

_setcamChartStyle

the chart's visual appearance while it is not in motion

Syntax

objectName_setcamChartStyle( style );

Argument Type Use Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>style</td>
<td>Character</td>
<td>I</td>
<td>indicates whether the chart appears as a Filled object or a Wireframe</td>
</tr>
</tbody>
</table>

_setcamChartType

the type of chart
### Syntax

```plaintext
objectName_setcamChartType( type );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Character</td>
<td>I</td>
<td>indicates whether the chart is a Bar, Box, Line, or Area chart</td>
</tr>
</tbody>
</table>

---

### _setcamColor2List

Sets the color range for color2

### Syntax

```plaintext
objectName_setcamColor2List( colorList );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorList</td>
<td>List</td>
<td>I</td>
<td>the colors list</td>
</tr>
</tbody>
</table>

---

### _setcamColor2RangeObject

Identifies the colorRangeObject for color2

### Syntax

```plaintext
objectName_setcamColor2RangeObject( object );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Character</td>
<td>I</td>
<td>the color range object</td>
</tr>
</tbody>
</table>
### _setcamColor2Variable_

Sets the variable for color2

**Syntax**

```plaintext
objectName_setcamColor2Variable( varName );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>varName</td>
<td>Character</td>
<td>I</td>
<td>the name of the variable whose values will index into the color range</td>
</tr>
</tbody>
</table>

### _setcamColor2VariableStatistic_

Sets the statistic represented by color2

**Syntax**

```plaintext
objectName_setcamColor2VariableStatistic( stat );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stat</td>
<td>Character</td>
<td>I</td>
<td>the statistic: Sum, Mean, or Freq</td>
</tr>
</tbody>
</table>

### _setcamColor2VariableType_

Sets the variable type for the color2 variable

**Syntax**

```plaintext
objectName_setcamColor2VariableType( type );
```
### _setcamColor3List

sets the colorList attribute

---

**Syntax**

```plaintext
objectName_setcamColor3List(list);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>List</td>
<td>I</td>
<td>color list</td>
</tr>
</tbody>
</table>

### _setcamColor3RangeObject

sets the colorRangeObject attribute

---

**Syntax**

```plaintext
objectName_setcamColor3RangeObject(arg1);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>Character</td>
<td>I</td>
<td>object's name</td>
</tr>
</tbody>
</table>

### _setcamColor3Variable

sets the colorVariable attribute
Syntax

objectName_setcamColor3Variable( varName );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>varName</td>
<td>Character</td>
<td>I</td>
<td>variable's name</td>
</tr>
</tbody>
</table>

_setcamColor3VariableStatistic

sets the color3VariableStatistic attribute

Syntax

objectName_setcamColor3VariableStatistic( stat );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stat</td>
<td>Character</td>
<td>I</td>
<td>specify Sum, Mean, or Freq</td>
</tr>
</tbody>
</table>

_setcamColor3VariableType

the color3Variable's type: numerical or categorical

Syntax

objectName_setcamColor3VariableType( arg1 );

_setcamColorList

sets the colorList attribute
### Syntax

```
objectName_setcamColorList( list );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>List</td>
<td>I</td>
<td>color list</td>
</tr>
</tbody>
</table>

### _setcamColorRangeObject

**Syntax**

```
objectName_setcamColorRangeObject( object );
```

### _setcamColorSource

**Syntax**

```
objectName_setcamColorSource( source );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Character</td>
<td>I</td>
<td>Indicates whether the color source is from a ColorList or a ColorRangeObject</td>
</tr>
</tbody>
</table>
_setcamColorVariable

sets the color variable

Syntax

objectName_setcamColorVariable( varName );

_setcamColorVariableStatistic

sets the colorVariable statistic: Sum, Mean, or Freq

Syntax

objectName_setcamColorVariableStatistic( stat );

_setcamColorVariableType

sets the colorVariable type

Syntax

objectName_setcamColorVariableType( type );

_setcamCubeXRotation

sets the cubeXRotation angle
Syntax
objectName_setcamCubeXRotation( angle );

_setcamCubeYRotation

sets the cubeYRotation angle

Syntax
objectName_setcamCubeYRotation( angle );

_setcamCubeZRotation

sets the cubeZRotation angle

Syntax
objectName_setcamCubeZRotation( angle );

_setcamDataSetID

Specifies the id for the analysis data set

Syntax
objectName_setcamDataSetID( id );
<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Numeric</td>
<td>I</td>
<td>the id returned by the open() function when the data set was opened</td>
</tr>
</tbody>
</table>

_setcamDataSetName

Specifies the name of the analysis data set

Syntax

objectName_setcamDataSetName( dsname );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsname</td>
<td>Character</td>
<td>I</td>
<td>the data set name</td>
</tr>
</tbody>
</table>

_setcamDataSource

Specifies the source of the analysis data set

Syntax

objectName_setcamDataSource( source );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Character</td>
<td>I</td>
<td>indicates whether the data set is being specified by a data set name or data set id</td>
</tr>
</tbody>
</table>

_setcamFaceColorMode

Specifies where color is displayed on a bar or cube
Syntax

```csharp
objectName_setcamFaceColorMode( mode );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Character</td>
<td>I</td>
<td>Specifies All if color is displayed on all bar surfaces, or Top if color is displayed only on the top surface</td>
</tr>
</tbody>
</table>

_setcamGridColor

Sets the color for the grid within the chart area if showGrid attribute is Yes

Syntax

```csharp
objectName_setcamGridColor( color );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>Character</td>
<td>I</td>
<td>the grid color</td>
</tr>
</tbody>
</table>

_setcamGroupByAxis

the axis to group charts by

Syntax

```csharp
objectName_setcamGroupByAxis( axis );
```
**argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis</td>
<td>Character</td>
<td>I</td>
<td>Indicates whether to group by the X, Y, or Z axis</td>
</tr>
</tbody>
</table>

**_setcamHighlightColor_**

Sets the color to display when a bar is selected

**Syntax**

objectName._setcamHighlightColor( color );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>Character</td>
<td>I</td>
<td>the highlight color</td>
</tr>
</tbody>
</table>

**_setcamLegendFont_**

Sets the font to use for legends

**Syntax**

objectName._setcamLegendFont( font );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>font</td>
<td>List</td>
<td>I</td>
<td>the SCL list the defines the font</td>
</tr>
</tbody>
</table>

**_setcamLegendType_**

Sets the legend type
**_setcamLegendType**

Sets the legend type.

**Syntax**

```
ObjectName._setcamLegendType( type );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Character</td>
<td>I</td>
<td>Indicates whether the legend is displayed as a Bar of graduated colors or List of discreet colors</td>
</tr>
</tbody>
</table>

**_setcamMissingColor**

Sets the color that represents missing values.

**Syntax**

```
ObjectName._setcamMissingColor( color );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>Character</td>
<td>I</td>
<td>the color to use</td>
</tr>
</tbody>
</table>

**_setcamOutlineColor**

Sets the color to use for outlines around bars.

**Syntax**

```
ObjectName._setcamOutlineColor( color );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>Character</td>
<td>I</td>
<td>the color to use</td>
</tr>
</tbody>
</table>
_setcamPickHighlightMode

Sets bar appearance when it is selected with actionMode set to Pick

Syntax

objectName_setcamPickHighlightMode( mode );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Character</td>
<td>I</td>
<td>indicates whether highlight color is used for the bar Outline, or whether the bar is Filled with the highlight color</td>
</tr>
</tbody>
</table>

_setcamPickList

Sets values in the pickList attribute

Syntax

objectName_setcamPickList( list );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>List</td>
<td>I</td>
<td>the list of values to store when a bar is selected with actionMode set to Pick</td>
</tr>
</tbody>
</table>

_setcamProjectionAngle

Sets the projectionAngle attribute

Syntax

objectName_setcamProjectionAngle( angle );
<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle</td>
<td>Numeric</td>
<td>i</td>
<td>the angle to use</td>
</tr>
</tbody>
</table>

_setcamReferenceLineRestriction

Sets the referenceLineRestriction attribute

Syntax

objectName_setcamReferenceLineRestriction( restriction );

_setcamRemoveXVariableValue

removes the x variable value

Syntax

objectName_setcamRemoveXVariableValue( arg1 );

_setcamRemoveYVariableValue

removes the y variable value

Syntax

objectName_setcamRemoveYVariableValue( arg1 );
_setcamRemoveZVariableValue

removes the z variable value

Syntax

objectName_setcamRemoveZVariableValue( arg1 );

_setcamRotationAngle

Sets the rotationAngle attribute

Syntax

objectName_setcamRotationAngle( angle );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle</td>
<td>Numeric</td>
<td>I</td>
<td>the angle to rotate around the vertical axis when viewMode is set to 3DPerspective. If viewMode is not 3DPerspective, rotation isn't possible and the rotationAngle isn't used.</td>
</tr>
</tbody>
</table>

_setcamScale

sets the scale attribute

Syntax

objectName_setcamScale( scale );
**_setcamSelectionMode**

Sets the selection mode with actionMode set to Pick

---

**Syntax**

```
objectName_setcamSelectionMode( mode );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Character</td>
<td>I</td>
<td>Indicates whether only a Single bar can be selected, Extended selection of multiple bars, or None for no selections.</td>
</tr>
</tbody>
</table>

---

**_setcamShowGrid**

Shows or hides the grid, which is the extension of tick marks into the chart area between the axis lines

---

**Syntax**

```
objectName_setcamShowGrid( state );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>Character</td>
<td>I</td>
<td>Specifies Yes to show the grid and No to suppress the grid</td>
</tr>
</tbody>
</table>

---

**_setcamShowLegend**

Shows or hides the legend
**Syntax**

```c
objectName_setcamShowLegend( state );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>Character</td>
<td>I</td>
<td>Specifies Yes to show the legend, or No to suppress the legend</td>
</tr>
</tbody>
</table>

**setcamSortMode**

_sets the sort mode to None, FormattedAscending, or FormattedDescending

**Syntax**

```c
objectName_setcamSortMode( mode );
```

**setcamTickmarkFont**

_sets the font for tick mark labels

**Syntax**

```c
objectName_setcamTickmarkFont( font );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>font</td>
<td>List</td>
<td>I</td>
<td>SCL list that defines the font to use</td>
</tr>
</tbody>
</table>
_setcamTiltAngle

Sets the tiltAngle attribute

Syntax

objectName_setcamTiltAngle( angle );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle</td>
<td>Numeric</td>
<td>I</td>
<td>the angle to tilt around the horizontal axis when viewMode is set to 3DPerspective. If viewMode is not 3DPerspective, tilting isn't possible and the tiltAngle isn't used.</td>
</tr>
</tbody>
</table>

_setcamViewMode

Sets the viewMode attribute

Syntax

objectName_setcamViewMode( mode );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Character</td>
<td>I</td>
<td>Specifies one of the modes 2D (height and width), 3D (height, width, and the illusion of depth), or 3DPerspective (height, width, and true depth).</td>
</tr>
</tbody>
</table>
**_setcamViewpointDistance_**

Sets the `viewpointDistance` attribute.

**Syntax**

```
objectName_setcamViewpointDistance(distance);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>distance</td>
<td>Numeric</td>
<td>I</td>
<td>the distance from the center, measured in pixels</td>
</tr>
</tbody>
</table>

**_setcamViewpointX_**

Sets the `viewpointX` attribute.

**Syntax**

```
objectName_setcamViewpointX(x);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Numeric</td>
<td>I</td>
<td>the X coordinate viewpoint location</td>
</tr>
</tbody>
</table>

**_setcamViewpointY_**

Sets the `viewpointY` attribute.

**Syntax**

```
objectName_setcamViewpointY(y);
```
Chapter 7

_setcamViewpointZ

Sets the viewpointZ attribute

Syntax

objectName_setcamViewpointZ( z );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>Numeric</td>
<td>I</td>
<td>the Z coordinate viewpoint location</td>
</tr>
</tbody>
</table>

_setcamXAxisLabelText

Sets the XAxisLabelText attribute

Syntax

objectName_setcamXAxisLabelText( text );

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Character</td>
<td>I</td>
<td>the label text</td>
</tr>
</tbody>
</table>

_setcamXAxisValueLength

Sets the XAxisValueLength attribute
**Syntax**

```csharp
objectName_setcamXAxisValueLength( length );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>Numeric</td>
<td>I</td>
<td>length for tick marks</td>
</tr>
</tbody>
</table>

---

**_setcamXDataMax**

Sets the XDataMax attribute

**Syntax**

```csharp
objectName_setcamXDataMax( maxValue );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxValue</td>
<td>Numeric</td>
<td>I</td>
<td>maximum value to display on the X axis</td>
</tr>
</tbody>
</table>

---

**_setcamXDataMin**

Sets the XDataMin attribute

**Syntax**

```csharp
objectName_setcamXDataMin( minValue );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minValue</td>
<td>Numeric</td>
<td>I</td>
<td>the minimum value to display on the X axis</td>
</tr>
</tbody>
</table>
_setcamXScale

Sets the XScale attribute

Syntax

objectName_setcamXScale(scale);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scale</td>
<td>Numeric</td>
<td>I</td>
<td>width of bars relative to X axis. Expects values from 0 to 1. 0 uses bars</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to represent lines, 1 sets width so the bars fill the entire X axis.</td>
</tr>
</tbody>
</table>

_setcamXVariable

Sets the XVariable attribute

Syntax

objectName_setcamXVariable(varName);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>varName</td>
<td>Character</td>
<td>I</td>
<td>the analysis variable to use for the X axis</td>
</tr>
</tbody>
</table>

_setcamYAxisLabelText

sets text for y axis label

Syntax

objectName_setcamYAxisLabelText(text);
_setcamYAxisMode

sets YAxisMode attribute

Syntax

objectName_setcamYAxisMode( mode );

_setcamYAxisValueSpacing

sets YAxisValueSpacing attribute

Syntax

objectName_setcamYAxisValueSpacing( spacing );

_setcamYDataMax

sets YDataMax attribute

Syntax

objectName_setcamYDataMax( max );
_setcamYDataMin

sets YDataMin attribute

Syntax

objectName_setcamYDataMin( min );

_setcamYScale

sets y scale

Syntax

objectName_setcamYScale( scale );

_setcamYVariable

sets y variable

Syntax

objectName_setcamYVariable( varName );

_setcamYVariableStatistic

sets y variable statistic: Mean, Sum, or Freq
Syntax

objectName_setcamYVariableStatistic( stat );

_setcamYVariableType

sets type for y variable

Syntax

objectName_setcamYVariableType( type );

_setcamZAxisLabelText

sets text for z axis label

Syntax

objectName_setcamZAxisLabelText( text );

_setcamZDataMax

sets ZDataMax attribute

Syntax

objectName_setcamZDataMax( max );
_setcamZDataMin

sets ZDataMin attribute

Syntax

objectName_setcamZDataMin( min );

_setcamZScale

sets z scaling factor

Syntax

objectName_setcamZScale( scale );

_setcamZVariable

sets z variable

Syntax

objectName_setcamZVariable( varName );

Events

Events specified for the Chart Control class are described here.
<table>
<thead>
<tr>
<th>Event</th>
<th>Enabled</th>
<th>Description</th>
<th>Handler Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAxisLabelText changed</td>
<td>Yes</td>
<td>Occurs when XAxisLabelText attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>XAxisValueLength changed</td>
<td>Yes</td>
<td>Occurs when XAxisValueLength attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>XDataMax changed</td>
<td>Yes</td>
<td>Occurs when XDataMax attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>XDataMin changed</td>
<td>Yes</td>
<td>Occurs when XDataMin attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>XScale changed</td>
<td>Yes</td>
<td>Occurs when XScale attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>XVariable changed</td>
<td>Yes</td>
<td>Occurs when XVariable attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>YAxisLabelText changed</td>
<td>Yes</td>
<td>Occurs when YAxisLabelText attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>YAxisMode changed</td>
<td>Yes</td>
<td>Occurs when YAxisMode attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>YAxisValueSpacing changed</td>
<td>Yes</td>
<td>Occurs when YAxisValueSpacing attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>YDataMax changed</td>
<td>Yes</td>
<td>Occurs when YDataMax attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>YDataMin changed</td>
<td>Yes</td>
<td>Occurs when YDataMin attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>YScale changed</td>
<td>Yes</td>
<td>Occurs when YScale attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>YVariable changed</td>
<td>Yes</td>
<td>Occurs when YVariable attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>Event Name</td>
<td>Action Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>YVariableStatistic changed</td>
<td>Yes</td>
<td>Occurs when YVariableStatistic attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>YVariableType changed</td>
<td>Yes</td>
<td>Occurs when YVariableType attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>ZAxisLabelText changed</td>
<td>Yes</td>
<td>Occurs when ZAxisLabelText attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>ZDataMax changed</td>
<td>Yes</td>
<td>Occurs when ZDataMax attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>ZDataMin changed</td>
<td>Yes</td>
<td>Occurs when ZDataMin attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>ZScale changed</td>
<td>Yes</td>
<td>Occurs when ZScale attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>ZVariable changed</td>
<td>Yes</td>
<td>Occurs when ZVariable attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>actionMode changed</td>
<td>Yes</td>
<td>Occurs when actionMode attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>animateMode changed</td>
<td>Yes</td>
<td>Occurs when animateMode attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>autoScale changed</td>
<td>Yes</td>
<td>Occurs when autoScale attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>axisColor changed</td>
<td>Yes</td>
<td>Occurs when axisColor attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>axisLabelFont changed</td>
<td>Yes</td>
<td>Occurs when axisLabelFont attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>barColor changed</td>
<td>Yes</td>
<td>Occurs when barColor attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>chartBackgroundColor changed</td>
<td>Yes</td>
<td>Occurs when chartBackgroundColor attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>Event Description</td>
<td>Value</td>
<td>Occurs when</td>
<td>Attribute Value Changed</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------</td>
<td>--------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>chartStyle changed</td>
<td>Yes</td>
<td>chartStyle attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>chartType changed</td>
<td>Yes</td>
<td>chartType attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>color2List changed</td>
<td>Yes</td>
<td>color2List attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>color2RangeObject changed</td>
<td>Yes</td>
<td>color2RangeObject attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>color2Variable changed</td>
<td>Yes</td>
<td>color2Variable attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>color2VariableStatistic changed</td>
<td>Yes</td>
<td>color2VariableStatistic attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>color2VariableType changed</td>
<td>Yes</td>
<td>color2VariableType attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>color3List changed</td>
<td>Yes</td>
<td>color3List attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>color3RangeObject changed</td>
<td>Yes</td>
<td>color3RangeObject attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>color3Variable changed</td>
<td>Yes</td>
<td>color3Variable attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>color3VariableStatistic changed</td>
<td>Yes</td>
<td>color3VariableStatistic attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>color3VariableType changed</td>
<td>Yes</td>
<td>color3VariableType attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>colorList changed</td>
<td>Yes</td>
<td>colorList attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>Event Description</td>
<td>Occurs When</td>
<td>Attribute Value Changed</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------</td>
<td>-------------------------</td>
<td></td>
</tr>
<tr>
<td>colorRangeObject changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>colorSource changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>colorVariable changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>colorVariableStatistic changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>colorVariableType changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cubeXRotation changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cubeYRotation changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cubeZRotation changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dataSet changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dataSetID changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dataSource changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>faceColorMode changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>footnote1 changed</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Yes-No</td>
<td>Occurs When</td>
<td>Untrapped</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>footnote2 changed</td>
<td>Yes</td>
<td>Occurs when footnote2 attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>gridColor changed</td>
<td>Yes</td>
<td>Occurs when gridColor attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>groupByAxis changed</td>
<td>Yes</td>
<td>Occurs when groupByAxis attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>highlightColor changed</td>
<td>Yes</td>
<td>Occurs when highlightColor attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>legendFont changed</td>
<td>Yes</td>
<td>Occurs when legendFont attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>legendType changed</td>
<td>Yes</td>
<td>Occurs when legendType attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>missingColor changed</td>
<td>Yes</td>
<td>Occurs when missingColor attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>outlineColor changed</td>
<td>Yes</td>
<td>Occurs when outlineColor attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>pickHighlightMode changed</td>
<td>Yes</td>
<td>Occurs when pickHighlightMode attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>pickList changed</td>
<td>Yes</td>
<td>Occurs when pickList attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>projectionAngle changed</td>
<td>Yes</td>
<td>Occurs when projectionAngle attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>rotationAngle changed</td>
<td>Yes</td>
<td>Occurs when rotationAngle attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>scale changed</td>
<td>Yes</td>
<td>Occurs when scale attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>selectionMode changed</td>
<td>Yes</td>
<td>Occurs when selectionMode attribute value is changed</td>
<td>Untrapped</td>
</tr>
<tr>
<td>Event</td>
<td>Status</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>showGrid changed</td>
<td>Yes</td>
<td>Occurs when showGrid attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>showLegend changed</td>
<td>Yes</td>
<td>Occurs when showLegend attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>showLegendValue changed</td>
<td>Yes</td>
<td>Occurs when showLegendValue attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>sortMode changed</td>
<td>Yes</td>
<td>Occurs when sortMode attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>tickmarkFont changed</td>
<td>Yes</td>
<td>Occurs when tickmarkFont attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>tiltAngle changed</td>
<td>Yes</td>
<td>Occurs when tiltAngle attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>title1 changed</td>
<td>Yes</td>
<td>Occurs when title1 attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>title2 changed</td>
<td>Yes</td>
<td>Occurs when title2 attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>title3 changed</td>
<td>Yes</td>
<td>Occurs when title3 attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>title4 changed</td>
<td>Yes</td>
<td>Occurs when title4 attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>viewMode changed</td>
<td>Yes</td>
<td>Occurs when viewMode attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>viewpointDistance changed</td>
<td>Yes</td>
<td>Occurs when viewpointDistance attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>viewpointX changed</td>
<td>Yes</td>
<td>Occurs when viewpointX attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>viewpointY changed</td>
<td>Yes</td>
<td>Occurs when viewpointY attribute value is changed</td>
<td></td>
</tr>
<tr>
<td>viewpointZ changed</td>
<td>Yes</td>
<td>Occurs when viewpointZ attribute value is changed</td>
<td></td>
</tr>
</tbody>
</table>