Overview

The Map control lets you create geographic maps that display data about specific map areas. The data values, which are retrieved from a SAS data set, are represented either by the color of the corresponding map area or by markers that you place on the map at points of interest. These markers, which can be one of several different shapes, are scaled and colored according to the underlying data, and can be labeled as well. At runtime, users can probe for interesting details by creating and moving markers and rotating the map interactively.
Using the Map Control

A map object uses individual map areas (or markers placed on these areas) to convey up to four different pieces of information:

- the location of a marker on a map area
- the color of a map area or of a marker
- the size of a marker
- the label on a map area or marker.

Methods that return the exact value of each area allow you to write complete applications using the Map object as an interface tool. When a specific map area is selected, SCL code can be invoked to display graphs or generate reports based on the selection.

Parts of a Map
map area

an area within a map such as a country, state, or province. Each map area has its own identification (ID) value. The border of a map area is defined by X and Y variables for all observations in the map data set that have the same ID value.

In area maps, the color of each map area represents a response value for that area. If the area has no response value, it uses the default map color. By default the map color is blue. You can use the mapColor attribute or the _setColor method to change the map color.

In marker maps, all map areas are the same color; only the marker color changes.

marker (marker maps only)

a graphical symbol whose size and color represent response values. The size of the marker, for example the height of a bar, is determined by the value of the response variable assigned to marker size. The color of the marker is determined by the value of the response variable assigned to marker color. The marker label is also generated from response data set values.

label

the text associated with the map area or marker and displayed in conjunction with it. The text of the label is contained in the response variable assigned to the label.

legend

one or two symbols that relate map area or marker characteristics to the data:

- The color bar represents the values of the color variable or the values of the color RANGE entry. By default the legend displays the minimum and maximum values of the color variable. If you assign a RANGE entry, the legend displays the minimum and maximum values defined by the RANGE entry. The color bar appears on the legend for both area maps and marker maps.

- The wedge-shaped size symbol shows the minimum and maximum values from the size variable. The size symbol appears on the legend for marker maps only.

About Map Types

The Map control produces two types of maps: area maps and marker maps.

About Area Maps

Area maps are two-dimensional maps that display data by coloring the individual map areas according to the associated data values. For example, you can display data like this sales data for selected European countries:

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>VOLUME</th>
<th>REVENUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>350</td>
<td>France</td>
<td>10</td>
<td>25000</td>
</tr>
<tr>
<td>394</td>
<td>Germany</td>
<td>100</td>
<td>250000</td>
</tr>
</tbody>
</table>
On an area map of Europe where revenue is mapped to color, countries with the highest revenue values would be green and those with the lowest revenue values would be red, if the default RANGE entry is used.

Area maps are similar to the maps produced by the CHORO statement of the GMAP procedure in SAS/GRAPH software.

### About Marker Maps

Marker maps display the values of one to three variables as graphical symbols whose size, color, and label represent the data values. In the example of sales in Europe, the color of the marker could represent the volume of sales, and the size of the marker could represent sales revenue. In this case, tall green bars would indicate high sales volume and revenue; tall red bars would indicate a country that has low sales, but has high revenue from those sales. In most cases, the color and size variables should be related or the map may be confusing.

### Sources of Information

The map object uses information from three SAS data sets:
- map data set
- marker data set
- response data set
In addition, it uses the IDMATCH Info dialog to obtain data from the user and a color RANGE entry to define the colors that are associated with the ranges of response values.

### About Map Data Sets

A map data set contains coordinates that define the boundaries of map areas, such as states or counties. Institute supplied map data sets are stored in a library that is usually assigned the libref MAPS.

A map data set must contain these variables:

- a numeric variable named X containing the horizontal coordinates of the boundary points.
- a numeric variable named Y containing the vertical coordinates of the boundary points.
- an identification variable, usually named ID, that identifies the map area to which the coordinates belong. The map identification variable links the map data set and the response data set either directly, or indirectly via the marker data set. Map area identification variables can be either character or numeric.

The following is an observation from the MAPS.US map data set. Notice that the ID variable is named STATE.
You must specify a map data set. If the map data set is the only input to the map object, then the result is a simple marker map. In this case, all markers placed on such a map will use the default marker color and all be the same size because they have no response values.

**About Response Data Sets**

The response data set provides the data that you want the map to display, such as sales figures, population statistics, or income ranges.

The response data set must contain these variables:

- at least one identification variable that identifies the map area, such as a state, a country, or a business office, to which a response value belongs. For example, in the response data set of European sales data, the response identification variable is ID, which contains the country code for each country.

  The response identification variable links the response values with the corresponding map areas and must match the IDMATCH variable in the marker data set. It may also match the ID variable in the map data set. Although referred to as the ID variable, this variable can have any name and can be either numeric or character. Character variables have a maximum length of 40.

- one or more variables with information about the map areas. These variables control the characteristics of the map:
  - marker or map area color
  - marker size
  - marker or map area label

The following is an example of an observation from a response data set.

**Example Code 17.2** Single Observation from a Response Data Set

<table>
<thead>
<tr>
<th>FIPS</th>
<th>STORES</th>
<th>AVGSALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>15</td>
<td>250000</td>
</tr>
</tbody>
</table>

The response variable assigned to the colorVariable attribute affects the color of the markers on a marker map or of the map areas on an area map. This variable must be numeric. Each value is compared to the ranges specified in the color RANGE entry. The markers will be the color specified for the particular range that the value fits into. For example, you could assign the AVGSALES variable to the colorVariable attribute.

The response variable assigned to the sizeVariable attribute affects the height or width of the markers. This variable must be numeric. For example, you could assign the STORES variable to the sizeVariable attribute.

The response variable assigned to the labelVariable attribute contains the text displayed at the base of the markers or on the map area in area maps. This variable can be either numeric or character.
About Marker Data Sets

The marker data set is a standard SAS data set that determines or records the position of the markers on the map. It also serves to link the map and response data sets when they do not share an identification variable.

The marker data set contains these four variables:

- X and Y are numeric variables that contain the coordinates of the location of each marker. These variables get their values from the map data set X and Y variables.
- an identification variable named ID that matches the identification variable in the map data set in both type and content. The corresponding variable in the map data set may have a different name, such as STATE. This variable can be either numeric or character. Character variables have a maximum length of 40.
- another identification variable named IDMATCH that matches a variable in the response data set in type and content. The corresponding variable in the response data set may have a different name, such as OFFICE. This variable can be either numeric or character. Character variables have a maximum length of 40.

A marker data set can be temporary or permanent.

Each observation in the marker data set specifies the location (X and Y) of a marker and its ID and IDMATCH values. The marker ID value must match the map ID value of the area where the marker is placed. The IDMATCH value must match the response ID value, which may also be the same as the map ID.

The following is an example of an observation from a Marker data set.

Example Code 17.3  Single Observation from a Marker Data Set

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>ID</th>
<th>IDMATCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.31618</td>
<td>0.14411</td>
<td>23</td>
<td>23</td>
</tr>
</tbody>
</table>

Notice that in this example the value of the IDMATCH variable, which comes from the response data set, is the same as the ID value, which comes from the map data set.

Creating a marker data set

You can create a marker data set from a SAS session or interactively from within the map object. Once you have created a marker data set you can assign it to the map object with the markerDataSetName or markerDataSetID attributes.

To create a marker data set in a SAS session, use a DATA step or FSEDIT to enter the X and Y location, the map identification value of each marker, and the IDMATCH variable. You can use map data sets like MAPS.USCENTER to obtain the X and Y locations for the center of each state as well as the state's ID value. You must supply the values for the IDMATCH variable. These values must correspond to the identification values in the response data set.

You can also create a marker data set interactively when your map data set and response data set have different identification values and you do not already have a marker data set. In this case the map object automatically enters values in a marker data set as you add markers to the map.
These are the steps:

1. Be sure the IDMATCH Info dialog is enabled by setting the IDQuery attribute value to On.
2. In the frame, use the popup menu to select Map Control -> Add.
3. Click on a map area to add a marker to the map. The map object copies the X, Y, and ID values from the map data set to the current observation in the marker data set.
4. When the IDMATCH Info dialog opens, enter the value of the response ID variable that corresponds to the currently selected map area. The map object assigns that value to the IDMATCH variable.

By default it saves the marker data set as WORK.DATA. To create a permanent data set enter a libref and data set name for the markerDataSetName attribute, and then save the frame.

Another way to create a marker data set of locations and IDMATCH values is to simply specify a map data set and a marker data set but no response data set. Place the markers on the map at the locations you plan to use and enter the IDMATCH value in the IDMATCH Info dialog.

Note: The map object automatically assigns the type of the IDMATCH variable so that it is the same type as the map identification variable. Therefore, your IDMATCH values must be the same type as the map ID values.

Saving the marker data set

While you are in BUILD mode, the map object saves the current marker information to the marker data set whenever you save the frame (File --> Save), open the Property sheet, or exit the frame. When the application is running, you can save the marker data set at any time with the _saveAs method.

You can save the default marker data set WORK.DATA as a permanent data set in one of these ways:

- Assign a libref and use the DATA step, for example
  
  ```
  libname mylib 'pathname';
  
  data mylib.marker1;
  set work.data1;
  run;
  ```

- Use the _saveAs method from your SCL program, for example
  
  ```
  map._saveAs('mylib.marker1');
  ```

About the IDMATCH Info Dialog

The IDMATCH Info dialog allows you to enter the value of the response ID variable that corresponds to the currently selected map area.
About the Color RANGE Entry

The color RANGE entry determines which color is assigned to the response value that is represented by color. These colors appear either on the map areas (in area maps) or as the marker color (in marker maps).

By default the response values are divided into three ranges and each range is assigned a color: values in the lowest third are red, values in the middle third are yellow, and values in the highest third are green.

You can define your own color ranges by creating a color RANGE entry and assigning it to the map object's colorRange attribute.

How the Data Sets Work Together

If the identification variables in the map data set and response data set contain the same values, then the link between the two is direct. The example below shows a response data set containing sales data for three states and uses FIPS codes to identify the states Maine (23), New Hampshire (33), and Vermont (50).

The example below also shows representative observations from the corresponding map data set, MAPS.US. MAPS.US also uses FIPS codes in a variable named STATE to identify each state.

Example Code 17.4  Response Data Set and Map Data Set with Matching ID Values

<table>
<thead>
<tr>
<th>Response Data Set</th>
<th>Map Data Set (MAPS.US)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIPS STORES AVGSALES</td>
<td>STATE X Y</td>
</tr>
<tr>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td>33</td>
<td>8</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>33</td>
<td>0.31596</td>
</tr>
</tbody>
</table>
When the identification variables match, as they do here, you can link the two data sets directly by specifying STATE as the ID variable for the map data set and FIPS as the ID variable for the response data set. In this case, when the map object builds the marker data set it automatically uses STATE as the ID variable and FIPS as the IDMATCH variable, as shown below. So long as the two variables contain the same values, it doesn't matter if they are named the same or not.

**Example Code 17.5**  Marker Data Set with Matching ID Values

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>ID</th>
<th>IDMATCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.32609</td>
<td>0.18508</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>0.33324</td>
<td>0.14181</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>0.28606</td>
<td>0.16714</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

However, the identification values in the response data set may be different from those in the map data set. For example, if the response ID value contains the names of cities instead of FIPS codes, the map and response ID values do not match. In this case, you must specify the city names as IDMATCH values, so the map object can use the ID and IDMATCH variables in the marker data set to build the link between the map and response data sets, as shown below.

**Example Code 17.6**  Marker Data Set with Different ID Values

<table>
<thead>
<tr>
<th>Map Data Set</th>
<th>Marker Data Set</th>
<th>Response Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE</td>
<td>ID</td>
<td>IDMATCH</td>
</tr>
<tr>
<td>------</td>
<td>--</td>
<td>-------</td>
</tr>
<tr>
<td>23</td>
<td>23</td>
<td>'Portland'</td>
</tr>
<tr>
<td>33</td>
<td>33</td>
<td>'Dover'</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>'Burlington'</td>
</tr>
</tbody>
</table>

If you place a marker on Maine (FIPS code 23) the marker data set associates 23 with 'Portland' and displays the response data for Portland on the map.

### Creating a Map Object

This section describes the steps for creating a simple map object and displaying response data. Because the steps depends on whether the response data set contains an ID variable that matches the ID variable in MAPS.AUSTRAL, there are two versions of the process. Both versions use the map data set of Australia, MAPS.AUSTRAL, which contains information for these map areas:

<table>
<thead>
<tr>
<th>Area Name</th>
<th>Identification Value (ID)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORTHERN TERRITORY</td>
<td>1</td>
</tr>
<tr>
<td>NEW SOUTH WALES</td>
<td>2</td>
</tr>
</tbody>
</table>
Version 1: Map ID and Response ID Match

The response data set MYLIB.AUSREV shown below contains an ID variable that matches in type and content the ID variable in the map data set.

**Example Code 17.7  Response Data Set MYLIB.AUSREV**

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>VOLUME</th>
<th>REVENUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NORTHERN TERRITORY</td>
<td>10</td>
<td>25000</td>
</tr>
<tr>
<td>2</td>
<td>NEW SOUTH WALES</td>
<td>100</td>
<td>250000</td>
</tr>
<tr>
<td>4</td>
<td>SOUTH AUSTRALIA</td>
<td>25</td>
<td>100000</td>
</tr>
<tr>
<td>5</td>
<td>TASMANIA</td>
<td>25</td>
<td>50000</td>
</tr>
<tr>
<td>6</td>
<td>VICTORIA</td>
<td>75</td>
<td>150000</td>
</tr>
</tbody>
</table>

Note that not all map areas have data.

1. In a frame, create a map object by selecting a Map Control from the Components window.

2. Set the following attribute values for the map information:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapDataSetName</td>
<td>Maps.Austral</td>
</tr>
<tr>
<td>mapIDVariable</td>
<td>ID</td>
</tr>
<tr>
<td>mapType</td>
<td>Marker (the default)</td>
</tr>
</tbody>
</table>

3. Set the following attribute values for the response information:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>responseDataSetName</td>
<td>Mylib.Ausrev</td>
</tr>
<tr>
<td>responseIDVariable</td>
<td>ID</td>
</tr>
<tr>
<td>sizeVariable</td>
<td>Volume</td>
</tr>
<tr>
<td>colorVariable</td>
<td>Revenue</td>
</tr>
<tr>
<td>labelVariable</td>
<td>Name</td>
</tr>
</tbody>
</table>
4 Return to the frame and use the popup menu to select Map Control –> Add.
5 In the frame, click on a map area to add a marker. If the response data set has data for the area, the marker will be sized, colored, and labeled accordingly. Otherwise it will be gray (the default marker color), indicating no data.

---

**Version 2: Map ID and Response ID Do Not Match**

The response data set MYLIB.AUSCITY shown below has an ID variable that does not match the ID variable in the map data set.

**Example Code 17.8 Response Data Set MYLIB.AUSCITY**

<table>
<thead>
<tr>
<th>OFFICE</th>
<th>VOLUME</th>
<th>REVENUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DARWIN</td>
<td>10</td>
<td>25000</td>
</tr>
<tr>
<td>SYDNEY</td>
<td>100</td>
<td>250000</td>
</tr>
<tr>
<td>ADELAIDE</td>
<td>25</td>
<td>100000</td>
</tr>
<tr>
<td>MELBOURNE</td>
<td>75</td>
<td>150000</td>
</tr>
</tbody>
</table>

OFFICE, which contains the city names, is the ID variable.

1 In a frame, create a map object by selecting a Map Control from the Components window.
2 Set the following attribute values for the map information:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapDataSetName</td>
<td>Maps.Austral</td>
</tr>
<tr>
<td>mapIDVariable</td>
<td>ID</td>
</tr>
<tr>
<td>mapType</td>
<td>Marker (the default)</td>
</tr>
</tbody>
</table>

3 Set the following attribute values for the response information:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>responseDataSetName</td>
<td>Mylib.Auscity</td>
</tr>
<tr>
<td>responseDVariable</td>
<td>Office</td>
</tr>
<tr>
<td>sizeVariable</td>
<td>Volume</td>
</tr>
<tr>
<td>colorVariable</td>
<td>Revenue</td>
</tr>
<tr>
<td>labelVariable</td>
<td>Office</td>
</tr>
<tr>
<td>IDQuery</td>
<td>On</td>
</tr>
</tbody>
</table>

4 Return to the frame and use the popup menu to select Map Control –> Add.
5 Click on an area to add a marker. The IDMATCH Info dialog opens. In the window, type the value of OFFICE that corresponds to the map area. For example, if the map area is Victoria, type MELBOURNE. At this point the map object assigns MELBOURNE as the value for IDMATCH; when you clicked on Victoria, it assigned the map ID value, 6, to the marker data set ID. Now the link between the map area, 6, and the response value, MELBOURNE is complete. Click OK.
The marker for Melbourne appears, sized, colored, and labeled accordingly.

Note: Because you are making the connection for the marker data set, you can assign any response value you like to the IDMATCH variable. Descriptionce the marker is created you can drag it to any area of the map and either change its IDMATCH value when the IDMATCH Info dialog opens or select OK to keep it.

---

**Using Legends**

A map object can display two types of legends, a color legend that is displayed as a color bar, and a size legend that is displayed as a wedge.

The color bar represents the values of the color variable or the values of the color RANGE entry. By default the legend displays the minimum and maximum values of the color variable. If you assign a RANGE entry, the legend displays the minimum and maximum values defined by the RANGE entry. The legend title is the color variable’s label, if it has one, or the variable name. The color bar appears on the legend for both area maps and marker maps.

The wedge-shaped size symbol shows the minimum and maximum values from the size variable. The legend title is the size variable’s label, if it has one, or the variable name. The size symbol appears on the legend for marker maps only.

You can control and manage the two legends separately. On maps that support both legend types, you can display either legend or both legends.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorLegendState</td>
<td>determines whether the color legend is displayed. Valid values are On and Off.</td>
</tr>
<tr>
<td>sizeLegendState</td>
<td>determines whether the size legend is displayed. Valid values are On and Off.</td>
</tr>
</tbody>
</table>

For a map object named map1, the following code displays both legends:

```javascript
map1.colorLegendState='on';
map1.sizeLegendState='on';
```

Both legends are interactive in the object; you can move them or change their size. As you change the legend’s size, legend information is hidden if it cannot fit within the legend. For example, if you make the legend too narrow to display the title, the title is hidden.

To move the legend, click on the middle of the legend, and then drag the legend to the desired location within the 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.

You can also move and size the object through its size and location attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorLegendXSize</td>
<td>width of the color legend, measured as a percent of the object region</td>
</tr>
<tr>
<td>colorLegendYSize</td>
<td>height of the color legend, measured as a percent of the object region</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>colorLegendXLoc</td>
<td>X location of the top left corner of the color legend, measured as a percent of the object region</td>
</tr>
<tr>
<td>colorLegendYLoc</td>
<td>Y location of the top left corner of the color legend, measured as a percent of the object region</td>
</tr>
<tr>
<td>sizeLegendXSize</td>
<td>Width of the size legend, measured as a percent of the object region</td>
</tr>
<tr>
<td>sizeLegendYSize</td>
<td>Height of the size legend, measured as a percent of the object region</td>
</tr>
<tr>
<td>sizeLegendXLoc</td>
<td>X location of the top left corner of the size legend, measured as a percent of the object region</td>
</tr>
<tr>
<td>sizeLegendYLoc</td>
<td>Y location of the top left corner of the size legend, measured as a percent of the object region</td>
</tr>
</tbody>
</table>

In SCL code, you can use the `getValue` method to determine whether a legend has been clicked on and what variable the legend represents. The method arguments are:

<table>
<thead>
<tr>
<th>Argument</th>
<th>What it returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON_LEGEND</td>
<td>0 if no legend was clicked</td>
</tr>
<tr>
<td></td>
<td>1 if the color legend was clicked</td>
</tr>
<tr>
<td></td>
<td>2 if the size legend was clicked</td>
</tr>
<tr>
<td>LEGEND_VARIABLE</td>
<td>Label of the variable represented by the clicked legend</td>
</tr>
</tbody>
</table>

For an object named `map1`, the following SCL code shows how `getValue` works. If you compile and test the code, you can click on either of the legends or on the map, and then look in the Log window to see the information returned by the put statements.

```scl
init:
    /* assume map1 is an existing marker map */
    map1.colorLegendState='on';
    map1.sizeLegendState='on';
    return;

main:
    length legvar $32;
    x = 0;
    leg = 0;
    infolist = makeList();
    map1._getValue(infolist);
    x = getnitemn(infolist, 'MAP_X', 1, 1);
    leg = getnitemn(infolist, 'ON_LEGEND', 1, 1);
    legvar = getnitemc(infolist, 'LEGEND_VARIABLE', 1, 1);
    put 'you clicked on ' x ;
    put ' A legend? ' leg;
    put ' which is for ' legvar;
    rc = dellist(infolist);
    return;
```
Attributes

Attributes specified for the Map Control class are described here. See sashelp.fsp.Widget.class"Attributes" on page 135 for inherited attributes.

Public Attributes

Dictionary

IDQuery

Returns or sets whether or not to query for ID values
Type: Character
Initial Value: (Object)
Valid Values: Off On Nomatch
Category: Misc

colorLegendState

Returns or sets whether or not to display color legend
Type: Character
Initial Value: (Object)
Valid Values: On Off
Category: Appearance

colorLegendXLoc

Returns or sets X location of top left corner of color legend in % of region units
colorLegendXSize

Returns or sets width of color legend in % of region

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Size/Location

colorLegendYLoc

Returns or sets Y location of top left corner of color legend in % of region units

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Size/Location

colorLegendYSize

Returns or sets height of color legend in % of region

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Size/Location
**colorRange**

Returns or sets the color range for data mapping

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

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

**colorVariable**

Returns or sets the variable to be used to color markers or map areas

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

**description**

Returns the short description for the object

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

**labelColor**

Returns or sets the default label color
**Type:** Character  
**Initial Value:** (Object)  
**Valid Values:**  
**Category:** Appearance

---

**labelFont**

Returns or sets the font to be used for marker or region labels

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

---

**labelSize**

Returns or sets the size of the labels in screen pct

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

---

**labelVariable**

Returns or sets the variable to be used to label markers or map areas

**Type:** Character  
**Initial Value:** (Object)  
**Valid Values:** \sashelp.classes.mapobjvareditor.sd  
**Category:** Data
**legendBackgroundColor**

Returns or sets the default Legend background color

Type: Character

Initial Value: (Object)

Valid Values:

Category: Appearance

---

**legendFont**

Returns or sets the font to be used for legend text

Type: List

Initial Value: (Object)

Valid Values:

Category: Appearance

---

**legendTextColor**

Returns or sets the default Legend Text color

Type: Character

Initial Value: (Object)

Valid Values:

Category: Appearance

---

**mapColor**

Returns or sets the default map color
Map Control

mapIDVariable

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

mapDataSetID

Returns or sets the Map DataSet ID
Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Data

mapDataSetName

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

mapIDVariable

Returns or sets the Map ID Variable
Type: Character
Initial Value: (Object)
Valid Values: \sashelp.classes.mapobjvareditor.sd
Category: Data
mapType

Returns or sets the type of map being used
Type: Character
Initial Value: (Object)
Valid Values: Marker Area
Category: Appearance

marker

Returns or sets the default marker type
Type: Character
Initial Value: (Object)
Valid Values: Bar Box Pyramid Cylinder Hexagon Circle Square Diamond Triangle Star
Category: Appearance

markerColor

Returns or sets the default marker color
Type: Character
Initial Value: (Object)
Valid Values:
Category: Appearance

markerDataSetID

Returns or sets the Marker DataSet ID
minimumMarkerSize

Returns or sets the minimum marker size in screen pct

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

maximumMarkerSize

Returns or sets the maximum marker size in screen pct

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

markerDataSetName

Returns or sets the marker dataset to be used

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

minimumMarkerSize

Returns or sets the minimum marker size in screen pct

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

Returns or sets what action occurs when user clicks on map

Type: Character
Initial Value: (Object)
Valid Values: Select Add Delete Rotate
Category: Misc

**outlineColor**

Returns or sets the default outline color

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

**responseDataSetID**

Returns or sets the Response DataSet ID

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

**responseDataSetName**

Returns or sets the response dataset to be used
responseIDVariable

Returns or sets the Response ID Variable

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

rotationAngle

Returns or sets the amount to rotate the map

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

selectMode

Returns or sets what can be clicked on

Type: Character
Initial Value: (Object)
Valid Values: BOTH MAP MARKER NONE
Category: Misc
**selectionColor**

Returns or sets the default selection color

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

**showEmptyMarkers**

Returns or sets whether markers without corresponding response data are visible

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

**sizeLegendState**

Returns or sets whether or not to display size legend

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

**sizeLegendXLoc**

Returns or sets X location of top left corner of size legend in % of region units
Map Control

sizeLegendYSize

Returns or sets height of size legend in % of region

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Size/Location

sizeLegendXSize

Returns or sets width of size legend in % of region

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Size/Location

sizeLegendYLoc

Returns or sets Y location of top left corner of size legend in % of region units

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Size/Location

sizeLegendYSize

Returns or sets height of size legend in % of region

Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Size/Location
sizeVariable

Returns or sets the variable to be used to size markers
Type: Character
Initial Value: (Object)
Valid Values: \sashelp.classes.mapobjvareditor.scl
Category: Data

textMode

Returns or sets whether text is 3D or 2D
Type: Character
Initial Value: (Object)
Valid Values: 3D 2D
Category: Appearance

tiltAngle

Returns or sets the amount to tilt the map
Type: Numeric
Initial Value: (Object)
Valid Values:
Category: Appearance

viewMode

Returns or sets projection used (PERSP makes far object appear smaller)
Map Control

_setcamColorLegendState

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

Methods

Methods specified for the Map Control class are described here. See sashelp.fsp.Widget.class “Methods” on page 145 for inherited methods.

Public Methods

_print

Print the map

Syntax

return = objectName_print();

__saveAs

Save as...

Syntax

return = objectName_saveAs();

_setcamColorLegendState

Specify whether or not to display color legend
Syntax

return = objectName_setcamColorLegendState();

_setcamColorLegendXLoc

Specify X location of top left corner of color legend in % of region units

Syntax

return = objectName_setcamColorLegendXLoc();

_setcamColorLegendXSize

Specify width of color legend in % of region

Syntax

return = objectName_setcamColorLegendXSize();

_setcamColorLegendYLoc

Specify Y location of top left corner of color legend in % of region units

Syntax

return = objectName_setcamColorLegendYLoc();
_setcamColorLegendYSize

Specify height of color legend in % of region

Syntax

return = objectName_setcamColorLegendYSize();

_setcamColorRange

Set the color range for data mapping

Syntax

return = objectName_setcamColorRange();

_setcamColorVariable

Set the variable to be used to color markers or map areas

Syntax

return = objectName_setcamColorVariable();

_setcamIDQuery

Specify whether or not to query for ID values
Syntax
return = objectName_setcamLabelColor();

_setcamLabelColor
Specify the default label color

Syntax
return = objectName_setcamLabelColor();

_setcamLabelFont
Specify the font to be used for marker or region labels

Syntax
return = objectName_setcamLabelFont();

_setcamLabelSize
Specify the size of the labels in screen pct.

Syntax
return = objectName_setcamLabelSize();
_setcamLabelVariable

Set the variable to be used to label markers or map areas

Syntax

return = objectName_setcamLabelVariable();

_setcamLegendBackgroundColor

Specify the default Legend background color

Syntax

return = objectName_setcamLegendBackgroundColor();

_setcamLegendFont

Specify the font to be used for legend text

Syntax

return = objectName_setcamLegendFont();

_setcamLegendTextColor

Specify the default Legend Text color
Syntax
return = objectName_setcamLegendTextColor();

_setcamMapColor

Specify the default map color

Syntax
return = objectName_setcamMapColor();

_setcamMapDataSetID

Specify the Map Dataset ID

Syntax
return = objectName_setcamMapDataSetID();

_setcamMapDataSetName

Set the map dataset to be used

Syntax
return = objectName_setcamMapDataSetName();
_setcamMapIDVariable

Specify the Map ID Variable

Syntax

\[ \text{return} = \text{objectName\_setcamMapIDVariable( );} \]

_setcamMapType

Set the type of marker to use

Syntax

\[ \text{return} = \text{objectName\_setcamMapType( );} \]

_setcamMarker

Specify the default marker type

Syntax

\[ \text{return} = \text{objectName\_setcamMarker( );} \]

_setcamMarkerColor

Specify the default marker color
**_setcamMarkerDataSetID**

Specify the Marker DataSet ID

**Syntax**

```
return = objectName_setcamMarkerDataSetID();
```

**_setcamMarkerDataSetName**

Set the marker dataset to be used

**Syntax**

```
return = objectName_setcamMarkerDataSetName();
```

**_setcamMaximumMarkerSize**

Set the maximum marker size in screen pct

**Syntax**

```
return = objectName_setcamMaximumMarkerSize();
```
_setcamMinimumMarkerSize

Set the minimum marker size in screen pct

Syntax

return = objectName_setcamMinimumMarkerSize();

_setcamMode

Specify what action occurs when user clicks on map

Syntax

return = objectName_setcamMode();

_setcamOutlineColor

Specify the default outline color

Syntax

return = objectName_setcamOutlineColor();

_setcamResponseDataSetID

Specify the Response DataSet ID
```
return = objectName_setcamResponseDataSetID( );
```

**_setcamResponseDataSetName**

Set the response dataset to be used

```
return = objectName_setcamResponseDataSetName( );
```

**_setcamResponseIDVariable**

Specify the Response ID Variable

```
return = objectName_setcamResponseIDVariable( );
```

**_setcamRotationAngle**

Specify the amount to rotate the map

```
return = objectName_setcamRotationAngle( );
```
**_setcamSelectMode**

Specify what can be clicked on

**Syntax**

```c
return = objectName_setcamSelectMode();
```

**_setcamSelectionColor**

Specify the default selection color

**Syntax**

```c
return = objectName_setcamSelectionColor();
```

**_setcamShowEmptyMarkers**

Specify whether markers without corresponding response data are visible

**Syntax**

```c
return = objectName_setcamShowEmptyMarkers();
```

**_setcamSizeLegendState**

Specify whether or not to display size legend
_setcamSizeLegendXLoc

Specify X location of top left corner of size legend in % of region units

Syntax
return = objectName_setcamSizeLegendState( );

_setcamSizeLegendXSize

Specify width of size legend in % of region

Syntax
return = objectName_setcamSizeLegendXSize( );

_setcamSizeLegendYLoc

Specify Y location of top left corner of size legend in % of region units

Syntax
return = objectName_setcamSizeLegendYLoc( );
_setcamSizeLegendYSize

Specify height of size legend in % of region

Syntax
return = objectName_setcamSizeLegendYSize();

_setcamSizeVariable

Set the variable to be used to size markers

Syntax
return = objectName_setcamSizeVariable();

_setcamTextMode

2D vs 3D text

Syntax
return = objectName_setcamTextMode();

_setcamTiltAngle

Specify the amount to tilt the map
_setcamTiltAngle

projection used (PERSP makes far object appear smaller)

Syntax

return = objectName_setcamTiltAngle( );

_setcamViewMode

projection used (PERSP makes far object appear smaller)

Syntax

return = objectName_setcamViewMode( );

Events

Events specified for the Map Control class are described here.

System Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Enabled</th>
<th>Description</th>
<th>Handler Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT2</td>
<td>Yes</td>
<td>Occurs in response to user action</td>
<td>Not specified</td>
</tr>
</tbody>
</table>