# Contents

**Chapter 1 / Overview of the MDDB Report Viewer** ........................................ 1  
What Is the MDDB Report Viewer? ................................................................. 1  
Support for Access Control Features ......................................................... 2  
Requirements for Running the MDDB Report Viewer ............................ 2

**Chapter 2 / Setting Up the MDDB Report Viewer** ....................................... 5  
Methods for Setting Up the MDDB Report Viewer ...................................... 5  
Working with Repositories ........................................................................ 8

**Chapter 3 / Using the MDDB Report Viewer** ................................................ 13  
Tips for Using the MDDB Report Viewer ...................................................... 13

**Chapter 4 / Making Advanced Customizations to the MDDB Report Viewer** ...... 21  
MDDB Report Viewer Class .................................................................... 24  
MDDB Report Viewer Instance Variables ............................................... 24  
Flow of Control in the MDDB Report Viewer Class .................................. 27  
MDDB Report Viewer Variables .............................................................. 36  
MDDB Report Viewer Cascading Style Sheets ........................................ 42  
Dictionary ................................................................................................. 44
Overview of the MDDB Report Viewer

What Is the MDDB Report Viewer?

The MDDB Report Viewer enables users to generate and view reports and graphs of data that are stored in a multidimensional database (MDDB) without running a SAS session.

An MDDB is a specialized data storage facility that stores summarized data for fast and easy access. Users can quickly view large amounts of data as a value at any cross-section of business dimensions. A business dimension can be any vision of the data that makes sense, such as time, geography, or product. Users create and update multidimensional databases using SAS/EIS software or the MDDB procedure when the SAS OLAP Server has been licensed.

The MDDB Report Viewer enables users who do not have access to SAS software (or who do not want to invoke SAS software) to view the data in an MDDB. This capability eliminates the need to have SAS software running on all users’ machines and provides access to the MDDB reports and graphs in a web environment.

Note: The SAS OLAP Server, available in SAS 9.1 and later, enables users to develop advanced SAS Business Intelligence applications using all of our new software, including SAS Information Delivery Portal, SAS Web Report Studio, and SAS Enterprise Guide. If you are developing new OLAP applications, then consider using this new technology rather than the MDDB Report Viewer.
Support for Access Control Features

The MDDB Report Viewer enables you to perform the following tasks that are associated with the Access Control features of SAS/EIS software:

- deny access to the entire table
- drop or keep hierarchies
- drop or keep ANALYSIS/COMPUTED columns
- hide ANALYSIS columns
- drop or keep CATEGORY columns
- drop or keep hierarchy levels
- drop or keep data values and totals
- hide or show data values
- set initial drill levels
- drop or keep statistics for individual ANALYSIS/COMPUTED columns
- hide the special Total value
- define initial drill subsets.

The MDDB Report Viewer supports the following Applications Access features:

- Report Layout
- Show Detail Data.

Requirements for Running the MDDB Report Viewer

Before you begin setting up the MDDB Report Viewer, you must meet the following requirements:

- Version 9 or later of the following SAS software products must be licensed at your site:
  - Base SAS software.
  - SAS/IntrNet software. The Application Dispatcher component (consisting of the Application Broker and Application Server components) must be installed and configured.
  - SAS/GRAPH software (optional but recommended).
  - SAS/EIS software or the SAS OLAP Server must be licensed at your site.
Note: MDDB Report Viewer 9.4 works only with the V8 SAS OLAP Server, which is available with both SAS 8 (as a separate product) and SAS 9 (as part of the SAS OLAP Server).

- The MDDB that you use to generate reports must be created, registered in a repository, and stored in a location to which you have access. You can create an MDDB by using SAS/EIS software or PROC MDDB when the SAS OLAP Server has been licensed. SAS/EIS software automatically registers the MDDB in the repository. If you use PROC MDDB to create the MDDB file, you must register the MDDB in a SAS/EIS repository. See the online Help for these products for complete instructions on how to create an MDDB. The MDDB Report Viewer can use only MDDB files to create reports. It cannot use SAS data sets.

- Your web browser must support HTML pages with frames.
Chapter 1 / Overview of the MDDB Report Viewer
Setting Up the MDDB Report Viewer

Methods for Setting Up the MDDB Report Viewer

Overview of Methods for Setting Up the MDDB Report Viewer

The MDDB Report Viewer consists of three HTML pages in which users can enter information to generate reports and graphs from an MDDB. Some features of the MDDB Report Viewer pages might appear slightly different on different web browsers. If you use more than one web browser to access the MDDB Report Viewer, consider these differences when you set up and customize the tool.

You can use SAS/EIS software access control features with the MDDB Report Viewer. See "Support for Access Control Features" on page 2 to learn more about using access control.

Note: To run this release of the MDDB Report Viewer, your system administrator must have previously set up a Repository Manager for accessing metadata. For more information about this setup procedure, see "Working with Repositories" on page 8. You can also refer to the online SAS Help and Documentation for Base SAS software and SAS/EIS software for details about setting up a repository.
You can use any of three methods to set up the MDDB Report Viewer.

Method 1

Copy the sample webeis.html page for the MDDB Report Viewer. The sample webeis.html page is included in the SAS/IntrNet CGI Tools for the web server installation package and can be found in the sasweb/IntrNet9/MRV directory under your web server root document directory. Modify the webeis.html file to specify your site's repositories, services, background colors, and so on. You can specify a subclass of the WEBEIS class to customize viewer behavior. See Method 3, Step 2 for a description of the CLASS parameter.

Method 2

Use the dynamic entry into the application by entering a URL that is similar to the following in your web browser:

http://web-server-name/broker-URI?_program=sashelp.webeis.rptsel.scl
&_service=myservice&metabase=sashelp.mbeis&bgtype=color&bg=red
&class=sashelp.override.myweb.class

Here, broker-URI, BGTYPE, BG, and CLASS are as described in Method 3, Step 2. With this method, no HTML pages are created or stored.

Method 3

Run the SAS AF command to create HTML pages for your repositories and to set up the MDDB Report Viewer at your site. Follow these steps:

1. Start a SAS session.

2. To create the MDDB Report Viewer HTML file, enter the following command in the Program Editor window and submit the command to SAS for processing:

   ```
   dm "af c=sashelp.webeis.rptsel.scl metabase=my-metabase
   pathname='HTML-file' <CGI='broker-URI'>
   <title='1996 Sales Report'> <bgtype='color'> <bg=blue>
   <class='sashelp.override.myweb.class'>";
   ```

   Here

METABASE

is the name of the SAS/EIS repository in which the MDDB has been registered. A METABASE value is required. The name can contain up to 60 characters and blank spaces. If you use blank spaces or special characters in the name, you must delimit the name with single quotation marks ('). SAS recommends that you use the same or similar filenames for the METABASE and PATHNAME options so that you can easily determine the metabase with which a particular instance of the MDDB Report Viewer is associated.

---

**Note:** The term metabase is retained for backward compatibility.
PATHNAME
is the path and filename of the MDDB Report Viewer HTML file that is created by the AF command. The directory is typically located under the web server document root or in another directory served by the web server. A PATHNAME value is required. SAS recommends that you use the same or similar filenames for the METABASE and PATHNAME options so that you can easily determine the metabase with which a particular instance of the MDDB Report Viewer is associated.

CGI
is the optional URI for the Application Broker component of Application Dispatcher (for example, /cgi-bin/broker or scripts/broker.exe). If you do not specify a value for this option, you must supply a value in the HTML file after it is created.

TITLE
is the title that appears at the top of the report. A TITLE value is optional. If you do not specify a title, the title "Multidimensional Reports" is used.

Note: Avoid using a percent sign (%) in the title because this symbol might be misinterpreted.

BGTYPE
is the type of background that appears in the application reports. Specify bgtype='color' to control the color of the background or bgtype='image' to control the background pattern displayed in the application reports. Use this option with the BG option, described below. A BGTYPE value is optional. If you specify bgtype='color', the BG option expects one of the named colors or a hexadecimal value for one of the colors that is supported by your web browser. If you specify bgtype='image', the BG option expects the URL of a background image file. You can specify only GIF and JPG image files for the background. If you specify BGTYPE and omit BG, or if you do not use either option, the background is the default color, silver.

Note: When you control the background color of the MDDB Report Viewer HTML pages, you might also want to control the background color of graphs that are displayed on the HTML pages. To do this, you can use a transparent GIF image, which is an image with a transparent background in which the HTML background color is visible. In effect, you create a graph in a clear frame so that the background color of the HTML page displays through the frame. A device driver to create the transparent GIF is not supplied with SAS/GRAPH software. However, you can use the TRANSPARENCY option of the SAS/GRAPH GOPTIONS statement to create a graph with a transparent background. For more information about the TRANSPARENCY option, see the documentation for the GOPTIONS statement in the SAS/GRAPH Help and Documentation.

BG
specifies the color or image to display in the background. A BG value is optional. If you specify bgtype='color', then specify a color value for BG. If you specify bgtype='image', then specify an image value for BG. You can specify a color name or a hexadecimal value for the color value. You can specify a URL for the image file value. See the documentation for your web browser for valid color values. If you specify BG and omit BGTYPE, or if you do not use either option, the background is the default color, silver.
CLASS is the name of a subclass of the WEBEIS class. A CLASS value is optional. Add this parameter if the user has overridden any WEBEIS methods to change the viewer behavior. You can specify either a three- or four-level name. For example, the following are both valid:

sashelp.override.myweb

sashelp.override.myweb.class

3 In a text editor, open the HTML file that you created, and supply your own values in the HTML code that is preceded by a comment. These values include the following:

broker-URI
In the tag <FORM ACTION="broker-URI">, you must supply a value if you did not specify the CGI= option in the AF command that creates the HTML pages.

service-name, service-label
In the HTML lines

<br>Select service: <SELECT NAME="_service">
OPTION VALUE="service-name" SELECTED>service-label

specify the list of services that are available at your site. Provide an <OPTION> tag for each of your services. For more information about services, see _SERVICE.

debug selection list
You can modify the list of debug options for your site in the following HTML line:

Debugging level: <SELECT NAME="_debug">

4 Start the Application Server and point your web browser to the HTML file that is generated in Method 3, Step 2.

You can specify the METABASE, PATHNAME, CGI, TITLE, BCTYPE, BG, and CLASS options in any order. Run the Application Server for each repository that contains MDDBs that users access when they run their reports.

---

Working with Repositories

Overview of Working with Repositories

The Common Metadata Repository is a general-purpose metadata management facility that provides common metadata services to different SAS/EIS applications. The Common Metadata Repository enables SAS/EIS software to share metadata with other SAS products.
Complete all of the following tasks to set up the Common Metadata Repository:

1. Specify the system repository manager location. See page 9
2. Set up the system repository manager files. See page 9
3. Define the repository to the Application Server. See page 10
4. Set up the SASHELP repository. See page 10

Note: You must have Write access to the SASHELP directory to complete these tasks.

Specifying the System Repository Manager Location

Follow these steps to specify the location of the system repository manager:

1. Create a directory that is dedicated exclusively to the storage of repository manager files, for example:
   - Windows users: !SASROOT\RPOSMGR
   - UNIX users: !SASROOT/RPOSMGR
   This directory should not be used to store other SAS files.

   Note: This system repository manager path is used later in this task.

2. Enter REGEDIT at a SAS command line. From the menu bar, select Tools ⇒ Options ⇒ Registry Editor to open the Registry Editor Options window. In the Select Registry View region, select the View All check box and then click OK. From the menu bar, select File ⇒ Close to close the Registry Editor Options window.

3. Enter REGEDIT again at a SAS command line. Under the HKEY_SYSTEM_ROOT tree, expand CORE and REPOSITORY. Select the REPOSITORY_MGR node. From the menu bar, select Tools ⇒ Options ⇒ Registry Editor. Select Open HKEY_SYSTEM_ROOT for write access. Then click OK.

4. Select the Path item in the right pane. From the pop-up menu, select Modify. Enter the path from Step 1. For example, enter !SASROOT\RPOSMGR. Click OK to close the Edit String Value window. From the menu bar, select File ⇒ Close to close the Registry Editor Options window and save the changes.

Setting Up the System Repository Manager Files

Complete the following steps to set up the necessary system repository manager files. You must have Write access to SASHELP to specify the system repository manager.
1 Create a directory that is dedicated exclusively to the storage of repository manager files, for example:
   - Windows users: \SASROOT\RPOSMGR
   - UNIX users: /SASROOT/RPOSMGR

Do not store other SAS files in this directory.

2 At a SAS command line, enter REPOSMGR and then select **Setup Repository Manager**.

3 In the Repository Manager Setup window, **Library** defaults to RPOSMGR. For **Path**, specify the path from Step 1 and then select the **Write values to system registry** check box. Then click **OK**.

4 In the resulting dialog box, click **Yes** to generate the necessary repository manager files.

This completes the set up for the System Repository Manager. You can create additional repository managers (a user repository manager, for example) by repeating these steps and by using a different path.

**Note:** This step sets the default location for the repository manager for your site. Individual users can override this location by executing the previous steps.

---

### Defining the Repository to Application Dispatcher

After you set up the Repository Manager files, you must include the following statements after the PROC APPSRV statement:

```
ALLOCATE LIBRARY RPOSMGR 'rposmgr-path' ;
DATALIBS RPOSMGR;
```

### Setting Up the SASHELP Repository

Complete the following steps to set up the SASHELP repository:

1 At a SAS command line, enter REPOSMGR and then select **Repository Registration**.

2 In the Repository Registration window, select **New**.

3 In the Register Repository (New) window, enter **SASHELP** (in uppercase) in the **Repository** field. In the **Path** field, enter the full directory path where the CORE catalog is located. For example:
   - Windows users: \SASROOT\CORE\SASHELP
   - UNIX users: /SASROOT/sashelp

4 In the **Description** field, you can enter any character string (for example, SASHELP Repository). Click **OK** to close the Register Repository (New) window. Click **Close** to exit the Repository Registration window.
Note: Repositories cannot span multiple directories because the path cannot contain concatenated directories. If you have existing metabases in concatenated directories, copy the metabases to a single path that is referenced as a repository.
Using the MDDB Report Viewer

Tips for Using the MDDB Report Viewer

Using the Interface

How do I use the MDDB Report Viewer?

The MDDB Report Viewer contains four web pages in which you enter information or manipulate your report data:

Report Layout page
This page contains drop-down lists from which you select the MDDB and the style sheet to use.

Dimensions page
This page enables you to select the items that you want to include in the report.

- Click Options at the top of the page to go to the Optional Settings page, where you can specify a variety of options that control the layout of the report. In addition, you can specify whether to display a graph in the report.
- In the Columns section, define the report layout by selecting items to include from the Down and the Across list boxes.
- In the Analysis section, select one or more analysis variables from the Columns list box.
- In the Statistics section, select the variables that you want to specify statistics for from the Select Column list box (the items in the list box are the variables that you selected in the Analysis section). Then, from the Available list box, select one or more statistics by highlighting the desired statistics and then clicking the right-arrow button. To select all of the available
statistics, click the double-right-arrow button. To deselect statistics, select the statistics in the **Selected** list box and then click the appropriate left-arrow button to remove them from the list box.

- Click **View Report** to display the report.

**Optional Settings page**
This page enables you to set the report options and to specify whether to display a graph in the report.

- Click **Dimensions** at the top of the page to go to the Dimensions page, where you select the items to include in your report.
- In the **Filter Columns** list box, select category variables for subsetting your report data.
- In the **Filter Listbox Options** section, customize the size and location of the **List By** list box on the Report page.
- In the **Report** section, specify a title for the report and whether to display a table in the report.
- In the **Graph** section, specify whether to display a graph and then customize its appearance and location in the report.
- When you click **View Report**, the report is displayed.

**Report page**
This page displays the table and graph that are produced from selections made in the previous pages. You can specify new variables and select subset values to change the report.

- Click **Download to spreadsheet** to download the data in the HTML table, including the titles, as it appears on the page.
- Click **Rotate** to rotate the down and across dimensions of a report.
- Click **Dimensions** to go to the Dimensions page, where you can select the items to include in your report.
- Click **Options** to go to the Optional Settings page, where you can specify a variety of options that control the layout of the report. In addition, you can specify whether to display a graph in the report.
- Click **? Help** to view the MDDB Report Viewer documentation or a Help page that you created.
- Change report dimensions by selecting different variables from the **Down** and **Across** list boxes on the Report page. After you select the new dimensions, click **View Report** to display the new report.
- In the **Filter By** list box, select the values of the category variables by which to subset your data and then click **Apply Filter**. The report is redisplayed with the subset applied. If a graph was previously displayed, it is redisplayed with the subset applied.

**How do I select items from a selection list?**

Web browsers have different selection methods. For example, some browsers use a SHIFT-click combination and others use a mouse click only. Use the selection method that is appropriate for your browser.
How do I know whether the items that I select for a report are valid?

Because selection list items cannot be disabled, you receive a message when an item is invalid. For example, you cannot select the same item (or hierarchy containing the same item) for the Down and Across values in a report. Simply reselect the items and run the report again.

What does the Rotate button do?

Use the Rotate button to rotate the down and across dimensions of a report.

How does Download to spreadsheet work?

The Download to spreadsheet button appears on the Report page and on the detail data page (after a reach-through to detail data). On the Report page, the Download to spreadsheet button downloads the data in the HTML table, including the titles, as it appears on the page. On the detail data page, the Download to spreadsheet button downloads the detailed data that is displayed on the page. The data is written in comma-delimited format, and you can open the file in your spreadsheet program or save the file to disk for later use.

You can use the _MRVSEP global variable to specify a delimiter other than a comma. For more information, see Table 2, MDDB Report Viewer Global Variables on page 40.

Printing Reports

How can I print reports?

You can print reports using the browser. Follow the instructions for printing that are appropriate for your browser.

Can I print extremely large tables?

If you print a table that is extremely wide, you might not get the results that you want. Tables cannot be resized, so when you print a large table, some columns might be truncated.

Changing the Appearance of a Report

Can I change report dimensions from the Report page?

You can change report dimensions by selecting different variables from the Down and Across list boxes on the Report page. After you select the new dimensions, click View Report to display the new report.
To add or change analysis variables or statistics, click Dimensions to return to the Dimensions page and change your selections. Then click View Report. The report is automatically displayed with your new selections.

Can I change the colors of my report?

Colors for report values are determined by values that are set in the RANGE entry in the SAS/EIS metabase in which the MDDB is registered. To change the colors in which report values are displayed, edit the RANGE entry in the SAS/EIS metabase. To use colors that are supplied by your browser, delete the RANGE entry in the SAS/EIS metabase. The background color of the table cell is set to the color value in the RANGE entry. Make sure that the numeric text is not set to the background color so that the text is readable.

Note: Cascading style sheet (CSS) settings overwrite a RANGE setting.

---

**Viewing Your Data**

How do I drill down to additional values in a report?

To drill down to other values in a report, select a Down or Across value. The report title changes when you drill down to other levels of information.

How do I subset my report data?

On the Optional Settings page, select the category variables (in the Filter Columns list box) by which to subset, and then click View Report. When the report is displayed, select the values of the category variables (in the Filter By list box) by which to subset your data, and click Apply Filter. The report is redisplayed with the subset applied. If a graph was previously displayed, it is redisplayed with the subset applied.

How do I see the detail data?

The numbers in the table should be hyperlinked if the BASETABLE attribute is in the metadata and if the base table exists. If the numbers are not hyperlinked, reach-through is not available for the selected MDDB. Click a number, and select the variables that you want to see from the data set. Click Next, and the detail data is displayed in a table.

---

**Creating Graphs**

How do I generate a 3-D graph of the report data?

To generate a three-dimensional graph of the report data, go to the Optional Settings page (by clicking Options), and select 3D Clickable Graph in the GRAPH
section. Then select the graph type (block, vertical bar, and so on) from the **Type** drop-down list. Click **View Report** to display the report along with a graph of the first column of data in the table. You can right-click within the graphics display area to change the graph's properties or to save the graph to a file. The three-dimensional graph is produced with the Graph Applet.

**How do I generate a standard GIF graph of the report data?**

To generate a standard GIF graph of the report data, go to the Optional Settings page (by clicking **Options**), and select **Standard GIF Graph** in the **GRAPH** section. Then select the graph type (block, vertical bar, and so on) from the **Type** drop-down list. Click **View Report** to display a report along with a graph of the first column of data in the table. You can select the GRAPH icon next to any column in the report to change the statistic that is graphed.

The GIF graph works in a different manner from the three-dimensional graph. To drill down using the GIF graph, you must drill down on the table rather than the graph itself. The GIF graph is a static graph, similar to the type of graph that is produced by the GPLOT procedure.

**How do I change the font for the standard GIF graph?**

You can specify the font for the standard GIF graph from the REQUEST INIT program that is used by your application server. In the REQUEST INIT program, set the **_GRFONT** macro variable by specifying the following:

```%@ _grfont=myfont;```

By default, the MDDB Report Viewer uses the SWISSB font if a value is not specified for _GRFONT. For a complete list of available fonts, refer to **SAS/GRAPH: Reference**. For more information about the REQUEST INIT program, see the **REQUEST statement syntax**.

---

**Modifying the Default MDDB Report Viewer Settings**

**How do I specify the repository manager for the Application Dispatcher Server?**

After you set up the repository manager files, you must include the following statements after the PROC APPSRV statement:

```%@ ALLOCATE LIBRARY RPOSIMG 'rposmgr-path'; DATA LIBS RPOSIMG;```

**How do I specify a different delimiter for Download to spreadsheet?**

To use a different delimiter for **Download to spreadsheet**, set the **_MRVSEP** macro variable in the REQUEST INIT program that is used by your application server. For example, to use a semicolon (;) instead of the default comma (,) delimiter, insert the following into your REQUEST INIT program:

```%@ _mrvsep=%str(;)```
Can I create my own Help page?

By default, the Help button points to the following URL, which is located on the SAS web site:

http://support.sas.com/rnd/web/internet/mddbapp/hinttips.html

You can create your own Help page with information that is specific to your site. To do this, create the Help web page and specify the URL in the _MRVHELP macro variable in the REQUEST INIT program that is used by your application server. For example, you could insert a line similar to the following in your REQUEST INIT program:

%let _mrvhelp=http://myserver/myhelp.html;

Can I use cascading style sheets to modify the appearance of my report?

The MDDB Report Viewer, Version 8 and later, supports cascading style sheets. Style sheets provide you with an easy way to customize the viewer for your site. For more information about how to use style sheets with the MDDB Report Viewer, see “MDDB Report Viewer Cascading Style Sheets” on page 42.

Can I change the toolbar location?

You can change the toolbar location by setting a macro variable in the REQUEST INIT program. Set the _MRTBLOC variable to

%let _mrtbloc=toolbar-location-value;

In this setting, the toolbar-location-value can be one of the following values: 1=top, 2=bottom, 3=left, 4=right, and 5=no toolbar.

The default toolbar location is 1=top.

Can I display reports without the Down and Across list boxes?

You can disable the display of the Down and Across list boxes by specifying the following in your service definition in the Application Broker configuration file:

ServiceSet _MRNODIMBOXES "X"

Can I disable the sorting feature?

You can disable the sorting feature by specifying the following in your service definition in the Application Broker configuration file:

ServiceSet _MRNOSORT *X"

Can I disable the row paging feature?

You can disable the row paging feature by specifying the following in your service definition in the Application Broker configuration file:
Can I modify the settings for the number of rows to display?

By default, the options page lists ALL, 25, 50, and 100 as the number of rows to display. To modify these, specify a ServiceSet directive in the Application Broker configuration file for your service for the _MRVRNDX1, MRVRNDX2, MRVRNDX3, and MRVRNDX4 macro variables. For example, if you want the number of rows options to be ALL, 100, 200, and 500, use the ServiceSet directives in the Application Broker configuration file as follows:

```
ServiceSet _MRVRNDX1 "ALL"
ServiceSet _MRVRNDX2 "100"
ServiceSet _MRVRNDX3 "200"
ServiceSet _MRVRNDX4 "500"
```

Can I change the number of paging links that are displayed beneath the report table?

By default, five page links are displayed beneath the report. To modify this setting, use a ServiceSet directive for the _MRVNRLKS macro variable. For example, to display 10 paging links, specify

```
ServiceSet _MRVNRLKS "10"
```

How do I specify to the viewer not to use HTML frames?

To modify this setting, use a ServiceSet directive for the _MRNOFRAMES macro variable. For example, specify

```
ServiceSet _MRNOFRAMES "X"
```

The toolbar buttons on both the Layout and the Report pages are displayed at the top.

Can I change the appearance of the report table?

Use the _MRTBLPRM macro variable in a ServiceSet directive to change the appearance of the report table. For example, specify

```
ServiceSet _MRTBLPRM "CELLPADDING=4 CELLS PACING=2 BORDER=3"
```

These attributes are inserted into the <TABLE> tag for the report.
Making Advanced Customizations to the MDDB Report Viewer

MDDB Report Viewer Class .............................................. 24
MDDB Report Viewer Instance Variables ................................ 24
Flow of Control in the MDDB Report Viewer Class ......................... 27
MDDB Report Viewer Variables ........................................... 36
MDDB Report Viewer Cascading Style Sheets .............................. 42
Dictionary ........................................................................ 44
  _BUILD_ACROSS_LIST_ Method ........................................ 44
  _BUILD_ANALYSIS_LIST_ Method ...................................... 45
  _BUILD_ANLSORTORDER_ Method ...................................... 46
  _BUILD_APPLICATION_LIST_ Method ................................... 46
  _BUILD_CURRENT_SUBSETS_ Method .................................... 48
  _BUILD_DOWNLN_LIST_ Method .......................................... 48
  _BUILD_STATSL_LIST_ Method ........................................... 49
  _BUILD_TOTAL_ Method .................................................. 50
  _BUILD_URL_ONSUBMIT_ Method ....................................... 51
  _BUILD_WHERE_FORMAT_STRING_ Method .............................. 53
  CHECK_HIER_MEMBER_ Method .......................................... 54
  _CLOSE_FORM_ Method ................................................... 55
  _CLOSE_PAGE_ Method .................................................... 56
  _CLOSE_STATIC_FORM_ Method .......................................... 56
  _CREATE_STAT ARRAYS_ Method ...................................... 57
  DISPLAY ACROSS CELLS_ Method ...................................... 61
  DISPLAY_ANALYSIS_VARS_ Method ..................................... 63
  DISPLAY_DEFAULT_TITLE_ Method ..................................... 64
  DISPLAY_DOWNVAR_CELL_ Method ...................................... 65
  DISPLAY_ERROR_ Method .................................................. 68
  DISPLAY_ONEWAY_ Method .............................................. 68
  DISPLAY_ONEWAY_BLOCK_ Method ....................................... 69
  DISPLAY_ONEWAY_HBAR_ Method ....................................... 70
  DISPLAY_ONEWAY_PIE_ Method .......................................... 71
  DISPLAY_ONEWAY_VBAR_ Method ........................................ 71
  DISPLAY_STATISTIC_VARS_ Method ...................................... 72
  DISPLAY_SUBSET_TITLE_ Method ........................................ 75
  DISPLAY_TITLE_ Method ................................................... 75
  DISPLAY_TWOWAY_ Method .............................................. 76
  DISPLAY_TWOWAY_BLOCK_ Method ....................................... 77
  DISPLAY_TWOWAY_HBAR_ Method ....................................... 78
  DISPLAY_TWOWAY_VBAR_ Method ........................................ 79
_DISPLAY_VALUES_Method .................................................. 80
_DRILL_TO_LEVEL_Method ................................................. 84
_GET_ANALYSIS_VAR_NAME_Method ........................................ 84
_GET_ANALYSIS_VARS_Method ............................................... 85
_GET_AVAILABLE_STATS_Method ............................................. 85
_GET_DATA_MODEL_NAME_Method ............................................. 86
_GET_DOWNVAR_LIST_Method ............................................... 86
_GET_EMDDBMID_Method ..................................................... 87
_GET_GRAPH_VALUES_Method ............................................... 87
_GET_MDB_NAME_Method ..................................................... 87
_GET_MESSAGE_ID_Method ................................................... 90
_GET_METABASE_NAME_Method .............................................. 90
_GET_OUTPUT_FILE_ID_Method .............................................. 90
_GET_RANGE_COLOR_Method ................................................. 91
_GET_STATDESC_Method ..................................................... 91
_GET_SUBSET_FLAG_Method .................................................. 92
_GET_USEHOLAP_Method .................................................... 92
_OPEN_DYNAMIC_FILE_Method .............................................. 92
_OPEN_FORM_Method ........................................................ 93
_OPEN_ONEWAY_Method ...................................................... 93
_OPEN_STATIC_FILE_Method ................................................ 94
_OPEN_TABLE_Method ......................................................... 94
_OPEN_TWOWAY_Method ..................................................... 96
_OPEN_WEBOUT_FOR_SPDSHT_Method ..................................... 98
_OUTPUT_ACROSS_LIST_Method ............................................. 98
_OUTPUT_ADDTL_CLSVAL_PARMS_Method .................................. 99
_OUTPUT_ADDTL_RT_PARMS_Method ....................................... 99
_OUTPUT_ADDTOFAV_FUNCTION_Method .................................. 99
_OUTPUT_ALL_URL_ITEMS_Method .......................................... 100
_OUTPUT_ANAL_LIST_Method ............................................... 100
_OUTPUT_ANAL_SELECT_Method ............................................. 101
_OUTPUT_ARROW_FUNCTIONS_Method .................................... 102
_OUTPUT_BAR_SHAPE_LIST_Method ....................................... 103
_OUTPUT_BOOKMARK_BUTTON_Method ................................... 104
_OUTPUT_BOOKMARK_URL_Method ......................................... 104
_OUTPUT_CLASSVAL_URL_FN_Method ..................................... 106
_OUTPUT_CLICKABLE_GRAPH_Method ................................... 107
_OUTPUT_CONTENT_HEADER_Method .................................... 109
_OUTPUT_CSV_CONTENT_HEADER_Method ................................. 109
_OUTPUT_DEBUG_LIST_Method .............................................. 110
_OUTPUT_DEFLT_TITLE_OPTION_Method ................................. 110
_OUTPUT_DIMBTN_URL_FN_Method ........................................ 110
_OUTPUT_DIMENSIONS_BUTTON_Method ................................ 111
_OUTPUT_DOWN_LIST_Method .............................................. 112
_OUTPUT_DP>Title_OPTION_Method ....................................... 113
_OUTPUT_DS2HTM_HTML_Method .......................................... 113
_OUTPUT_DS2HTM_ST_Method ................................................ 114
_OUTPUT_DYNAMIC_HIDDEN_FLDS_Method ............................. 115
_OUTPUT_EMPTY_CELL_Method ............................................. 117
_OUTPUT_EMPTY_SERVICE_LIST_Method ................................ 117
_OUTPUT_GRAPH_DIMS_OPTION_Method .................................. 118
_OUTPUT_GRAPH_INSTR_Method ............................................ 118
_OUTPUT_GRAPH_LIST_Method ............................................. 118
_OUTPUT_GRAPH_LOC_OPTION_Method .................................. 119
MDDB Report Viewer Class

The MDDB Report Viewer class is a viewer that displays MDDB data. The class is a component of the MDDB Report Viewer, which is an application used by SAS/EIS software, SAS/IntrNet Application Dispatcher software, and SAS OLAP Server software.

The MDDB Report Viewer class enables you to specify dimensions that can be hierarchies or category variables, in addition to analysis variables. This class enables you to drill down on the hierarchy and other navigation, as well as to specify various types of graphic charts. The class writes output from the application to HTML in a web browser.

PARENT: SASHELP.FSP.OBJECT.CLASS

CLASS: SASHELP.WEBEIS.WEBEIS.CLASS

MDDB Report Viewer Instance Variables

The following instance variables are used in many of the MDDB Report Viewer methods:

ACRDRL_
  specifies the list of drill-down values for the across variables.

ACRVARS_
  specifies the list of selected variables for the across dimension.

ALEVELS_
  specifies the list of drill-down levels for the across variables.

ANALLBLS_
  specifies the list of analysis variable long labels.
ANALLIST_
specifies the list of analysis variables and computed columns.

ANALVARS_
specifies the list of selected analysis variables.

ATOTAL_
specifies a flag that indicates whether the across totals are turned on.

CLASS_
contains the three- or four-level name of the WEBEIS subclass.

CSSTURL_
contains the URL for the toolbar frame style sheet.

CSSURL_
contains the URL for the style sheet.

DEBUG_
contains the application server debug level.

DEFTITLE_
contains the value of the default title that is specified by the user.

DIMLBLS_
specifies the list of labels for the down and across dimensions.

DLEVELS_
specifies the list of drill-down levels for the down variables.

DLSEP_
contains the download-to-spreadsheet delimiter. The default value is a comma.

DMODEL_
specifies the four-level name of the data model class.

DOWNDRL_
specifies the list of drill-down values for the down variables.

DOWNL_
specifies the down variables list from the application list.

DOWNVARS_
specifies the list of selected variables for the down dimension.

DPTITLE_
specifies a flag that indicates whether the drill-path title is displayed.

DTOTAL_
specifies a flag that indicates whether the down totals are turned on.

EMDDBMID_
specifies the identifier of the data model class instance.

EXPFLAG_
specifies a flag that indicates whether the expands are displayed.

EXPLIST_
specifies a list that contains sublists for each expand. The sublists are of the form VAR='VALUE'.

EXPVALS_
specifies a list that contains the values of the expanded rows only.

EXPVAR_
specifies the name of the expanded variable.
GRFHT_
    contains the value of the graph height option.

GRFSRC_
    specifies the graph source that is selected by the user, where 1 is a 3-D clickable
graph and 2 is a standard GIF graph.

GRFWID_
    contains the value of the graph width option.

GRLOC_
    specifies the graph location that is selected by the user, where 1=bottom, 2=top,
    3=left, and 4=right.

GRPHTYPE_
    specifies the graph type selected by the user. Valid types include: BLOCK=block
    chart, HBAR=horizontal bar chart, PIE=pie chart, PLOT=plot, and VBAR=vertical
    bar chart.

GRPHVALS_
    specifies a list that contains the data points for the 3-D graph.

HIERL_
    specifies the list of metabase hierarchies.

HMODEL_
    specifies the four-level name of the HOLAP data model class. The default value
    is SASTOOL__DMDB.HOLAP_M.CLASS.

HTMLFILE_
    specifies the identifier of the output file for writing HTML.

IMGURL_
    contains the URL for the images.

MDDB_
    specifies the name of the selected MDDB.

METABASE_
    specifies the name of the selected metabase.

ROTFLAG_
    specifies a flag that indicates whether the user selected the Rotate button,
    where 1=Rotate button was selected and 2=Rotate button was not selected.

SESSIONID_
    specifies the value for the _SESSIONID variable for the application server
    session.

SHOWTAB_
    specifies a flag that indicates whether to display the table, where 1=yes, 2=no.

STATDESC_
    specifies a list of all possible statistics labels.

STATLIST_
    specifies a list of the available statistics from the metabase.

STATVARS_
    specifies a list of the selected statistics.

SUBHT_
    indicates the number of rows to display in the filter list boxes.
**SUBLOC_**
specifies the location of the filter list boxes, where 1=right, 2=left, 3=top, and 4=bottom.

**SUBSET_BY_**
specifies the list of selected filter values.

**SUBSET_FLAG_**
indicates whether filter values have been selected, where 1=filters have been selected and 0=filters have not been selected.

**SUBVARS_**
specifies the list of selected filter variables.

**SUBWID_**
contains the maximum width (in characters) of the filter list boxes.

**TBLOC_**
specifies the location of the filter toolbar, where 1=top, 2=bottom, 3=left, 4=right, and 5=do not display a toolbar.

**THISSESSION_**
specifies the value for the _THISSESSION variable for the application server session.

**USEHOLAP_**
indicates whether a HOLAP metabase registration is being used, where 1=HOLAP metabase registration is being used and 0=HOLAP metabase registration is not being used.

**VMDOFF_**
specifies a flag that indicates whether metadata verification checking is done on the data model, where any nonblank character=do not perform metadata checking and a blank=perform metadata checking.

---

**Flow of Control in the MDDB Report Viewer Class**

The following figures illustrate the flow of control in the MDDB Report Viewer WEBEIS class. For more information about the methods listed in these figures, refer to the individual method descriptions.
Figure 1. Flow of Control for the Layout Page

1. Initialization and error checking
2. Create an instance of WEBEIS class
3. Open _webout for writing HTML
4. Open repository
5. Call _OUTPUT_FRAME_HDR2_ method to generate the Dimensions and Options frames
6. Close repository and _webout, and terminate WEBEIS class
7. Return

This generates the <FRAMESET> tag for the Dimensions and Options pages, as well as the <FRAME> tags for the toolbar and layout frames.
Figure 2. Flow of Control for the Dimensions Page

Initialization and error checking

Instance WEBEIS class

Open repository

Call _GET_DOWNVAR_LIST_

Call _GET_ANALYSIS_VARS_

Call _GETAVAILABLE_STATS_

Open _webout for writing HTML

Call _GETOUTPUT_FILE_ID_

Call _OUTPUT_HDR_

Call _OUTPUT_VARIABLE_SEL_FORM_

Output all HTML hidden fields for Dimensions form

Close repository and _webout, and terminate WEBEIS class

Return
Figure 3. Flow of Control for the Layout Toolbar

Initialization and error checking

Create an instance of WEBEIS class

Open_webout for writing HTML

Call_GET_MESSAGE_ID_

Call_GET_OUTPUT_FILE_ID_

Output text as HTML content type header

Output <HTML>, <HEAD>, and <TITLE> tags

Output page title

Output </TITLE> tag

Style sheet parameter specified

Yes

Output <LINK> tag for stylesheet

No

Output <SCRIPT> tag

Call_OUTPUT_DMBTN_URL_FN_

Call_OUTPUT_QMBTN_URL_FN_

Output </SCRIPT> and </HEAD> tags

Output </BODY> tag

Output </TABLE> tag

Call_OUTPUT_LAYOUT_TOOLBAR_

Output </TABLE>, </BODY>, and </HTML> tags

Close_webout

Terminate WEBEIS class

Return
Figure 4. Flow of Control for the Report Page

Initialization and error checking

Create an instance of WEBEIS class

Open _webout for writing HTML

Call _GET_METABASE_NAME_

Call _GET_OUTPUT_FILE_ID_

Open repository

Call _OUTPUT_FRAME_HDR_

Close repository

Close _webout

Terminate WEBEIS class

Return

This generates the <FRAMESET> for the Report page, as well as the <FRAME> tags for the toolbar and report frames.
Figure 5. Flow of Control for the the Report Page (part 1)
Figure 5. Flow of Control for the the Report Page (part 2)

(Ceased)

**_IFNOFRAMES is not blank_**
**_IMRTBLOCLS NE ""_**?

Yes

- Output `<TABLE>` tag
- _Call OUTPUT_TOOLBAR_

No

- _Cell OUTPUT_HTML_AFTET_BOD YT_
- _Call OPEN_FORM_
- Output `<CENTER>` tag
- _Call OPEN_TABLE_

- Filter variable displayed on top or left?

Yes

- _Call repository GET_HIERARCHY_
- _Call SET_HIERL_LIST_

No

1

- _Call OUTPUT_SUBSET_SELECTIONS_
- SSL macro = 2 or 3

Yes

- _Call DISPLAY_DEFAULT_TITLE_
- _Call DISPLAY_TITLE_ for Down variable dropdown title
- _Call DISPLAY_TITLE_ for Across variable dropdown title
- _Call DISPLAY_SUBSET_TITLE_

No

- _Call SHOW GRAPH_
- GYL macro = 2 or 3

Yes

- _Call OUTPUT_DOWN_LIST_

No

- _Call OUTPUT_ACRROSS_LIST_
- Across variable selected?

Yes

- _Call OUTPUT_WRAPBUTTON_

No

(Ceased)
Figure 5. Flow of Control for the Report Page (part 3)
Flow of Control in the MDDB Report Viewer Class

Figure 6. Flow of Control for the Report Page Toolbar (part 1)

1. Initialization and error checking
2. Create an instance of WEBEIS class
3. Open _webout for writing HTML
   - Call _GET_MESSAGE_ID_
   - Call _GET_OUTPUT_FILE_ID_
   - Output text/HTML content type header
4. Output <HTML>, <HEAD>, and <TITLE> tags
5. Output title information
6. Output <TABLE> tag
   - Stylesheet parameter specified
     - Yes: Output <LINK> tag with stylesheet information
     - No: Output <SCRIPT> tag
5. Call _OUTPUT_VAR_FUNCTIONS_
6. Call _OUTPUT_ROTATE_FUNCTION_
7. Call _OUTPUT_CLASSVAL_URL_FN_
8. Call _OUTPUT_SETURL_FUNCTION_
9. Call _OUTPUT_ADDTOFAV_FUNCTION_

(Continued)
The MDDB Report Viewer uses macro variables that are set by users and passed into the viewer when the application executes. Table 4.1 on page 37 lists and describes the macro variables.

The MDDB Report Viewer also uses global variables that you can set in the REQUEST INIT program that is used by your application server. Table 4.2 on page 40 lists and describes these variables. For more information about the REQUEST INIT program, see PROC APPSRV, REQUEST Statement syntax.
Table 4.1  MDDB Report Viewer Macro Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDDB</td>
<td>Selected MDDB</td>
<td>Selected MDDB (for example, SASHELP.PRDMDDDB)</td>
</tr>
<tr>
<td>METABASE</td>
<td>Selected metabase</td>
<td>Selected metabase (for example, SASHELP.MBEIS)</td>
</tr>
<tr>
<td>SR</td>
<td>First row to display in the table</td>
<td></td>
</tr>
<tr>
<td>NR</td>
<td>Number of rows to display in the table</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Down</td>
<td>Hierarchies and category variables. Do not create variable names that contain an embedded percent sign (%) if the percent sign precedes the following characters: 0 through 9, a through f, and A through F. Names that contain these character combinations could be misinterpreted due to encoding and decoding issues. For example, a variable name of Product%20Line could be incorrectly interpreted as Product Line because %20 is the encoding sequence for a blank space.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Description</td>
<td>Details</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>AC</td>
<td>Across</td>
<td>Hierarchies and category variables. Do not create variable names that contain an embedded percent sign (%) if the percent sign precedes the following characters: 0 through 9, a through f, and A through F. Names that contain these character combinations could be misinterpreted due to encoding and decoding issues. For example, a variable name of Product%20Line could be incorrectly interpreted as Product Line because %20 is the encoding sequence for a blank space.</td>
</tr>
<tr>
<td>A</td>
<td>Analysis variable</td>
<td>Analysis variable. This macro variable is deprecated.</td>
</tr>
<tr>
<td>S</td>
<td>Statistic</td>
<td>Globally applied statistic. This macro variable is deprecated.</td>
</tr>
<tr>
<td>Am</td>
<td>Analysis variable</td>
<td>Analysis variable, where m designates the particular variable.</td>
</tr>
<tr>
<td>AmSn</td>
<td>Statistic</td>
<td>Statistic that is applied only to the analysis variable specified by Am. n designates the particular statistic. For example, A1S1 and A1S2 designate statistics that are applied only to the A1 analysis variable.</td>
</tr>
<tr>
<td>SV</td>
<td>Filter variables</td>
<td>Category variables to filter by</td>
</tr>
<tr>
<td>SL</td>
<td>Filter variable values</td>
<td>Values to filter by (for example, SL=COUNTRY:CANADA)</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Description</td>
<td>Details</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>EX</td>
<td>Expand values</td>
<td>For example, EX=COUNTRY=CANADA</td>
</tr>
<tr>
<td>V</td>
<td>Down dimension drill-down values</td>
<td>For example, V=YEAR=1995</td>
</tr>
<tr>
<td>VA</td>
<td>Across dimension drill-down values</td>
<td>For example, VA=PRODTYPE=FURNITURE</td>
</tr>
<tr>
<td>ST</td>
<td>Display table</td>
<td>1=yes, 2=no</td>
</tr>
<tr>
<td>DT</td>
<td>Default title</td>
<td>Max length=200</td>
</tr>
<tr>
<td>DP</td>
<td>Show drill-path in title</td>
<td>1=yes, 2=no</td>
</tr>
<tr>
<td>DC</td>
<td>Show down totals</td>
<td>1=yes</td>
</tr>
<tr>
<td>ACB</td>
<td>Show across totals</td>
<td>1=yes</td>
</tr>
<tr>
<td>GSC</td>
<td>Graph source</td>
<td>1=3-D clickable graph, 2=standard GIF graph (SAS/GRAPH software)</td>
</tr>
<tr>
<td>GL</td>
<td>Graph location</td>
<td>1=bottom, 2=top, 3=left, 4=right</td>
</tr>
<tr>
<td>GRT</td>
<td>Graph type</td>
<td>BLOCK, HBAR, PIE, PLOT, VBAR</td>
</tr>
<tr>
<td>BS</td>
<td>Graph bar shapes</td>
<td>STAR, HEXAGON, PRISM, CYLINDER</td>
</tr>
<tr>
<td>SPDSHT</td>
<td>Download to Spreadsheet flag</td>
<td></td>
</tr>
<tr>
<td>GW</td>
<td>Graph width</td>
<td>Default=600, max length=4</td>
</tr>
<tr>
<td>GH</td>
<td>Graph height</td>
<td>Default=450, max length=4</td>
</tr>
<tr>
<td>SSL</td>
<td>Filter list box location</td>
<td>1=right, 2=left, 3=top, 4=bottom</td>
</tr>
<tr>
<td>SW</td>
<td>Filter list box width</td>
<td>Default=15</td>
</tr>
<tr>
<td>SH</td>
<td>Filter list box height</td>
<td>Default=3</td>
</tr>
<tr>
<td>VIEW</td>
<td>View Report button</td>
<td>Value=view report</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Description</td>
<td>Details</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>GD</td>
<td>Graph down variable</td>
<td>Category variable for graphing</td>
</tr>
<tr>
<td>GA</td>
<td>Graph across variable</td>
<td>Across variable for filtering graph values</td>
</tr>
<tr>
<td>GG</td>
<td>Graph group variable</td>
<td>Graph group-by variable, second innermost down variable</td>
</tr>
<tr>
<td>GSG</td>
<td>Graph subgroup variable</td>
<td>Graph subgroup-by variable, third innermost down variable</td>
</tr>
<tr>
<td>SD</td>
<td>Down variable</td>
<td>Same as D, needed for Filter FORM</td>
</tr>
<tr>
<td>SAC</td>
<td>Across variable</td>
<td>Same as AC, needed for Filter FORM</td>
</tr>
<tr>
<td>CLASS</td>
<td>WEBEIS class name</td>
<td>For subclassing, default is SASHELP.WEBEIS.WEBEIS.CLASS</td>
</tr>
<tr>
<td>CSS</td>
<td>Style sheet URL</td>
<td>Applies to Variable Selection and Report pages</td>
</tr>
<tr>
<td>CSST</td>
<td>Toolbar style sheet URL</td>
<td>Applies to toolbar frame; if not specified, uses CSS value</td>
</tr>
<tr>
<td>BG</td>
<td>Background color or image</td>
<td>Color name, hexadecimal value, or image URL</td>
</tr>
<tr>
<td>BGTYPE</td>
<td>Background type</td>
<td>COLOR, IMAGE</td>
</tr>
</tbody>
</table>

*Table 4.2* MDDB Report Viewer Global Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMDOFF</td>
<td>Turn VERIFYMD checking off</td>
<td>Any nonblank character turns it off, the default is on</td>
</tr>
<tr>
<td>_GRFONT</td>
<td></td>
<td>SWISSB is the default; use SAS font names</td>
</tr>
<tr>
<td>_MRVHELP</td>
<td>URL of Help file</td>
<td>The default is the Hints and Tips page</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Description</td>
<td>Details</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>_MRTBLOC</td>
<td>Toolbar location</td>
<td>1=top, 2=bottom, 3=left, 4=right, and 5=none; the default is top</td>
</tr>
<tr>
<td>_MRVSEP</td>
<td>Download to spreadsheet delimiter</td>
<td>A comma is the default</td>
</tr>
<tr>
<td>_MRVTBSC</td>
<td>Toolbar frame scrolling</td>
<td>NO or blank indicates no scrolling; YES adds a scroll bar to frame</td>
</tr>
<tr>
<td>_MRVTBSZ</td>
<td>Toolbar size in pixels</td>
<td>A character string of the form (horizontal, vertical); the default is (50, 125)</td>
</tr>
<tr>
<td>_MRNODIMBOXES</td>
<td>Turns off the down and across list boxes and the View Report button on display</td>
<td>A nonblank value turns this off; the default is on</td>
</tr>
<tr>
<td>_MRNOFRAMES</td>
<td>Indicates whether to use HTML frames in the output</td>
<td>A nonblank value turns this off; the default is on</td>
</tr>
<tr>
<td>_MRNOVARCHECK</td>
<td>Turns off the down and across variable selection error checking</td>
<td>A nonblank value turns this off; the default is on</td>
</tr>
<tr>
<td>_MRBODYONLY</td>
<td>Nonblank</td>
<td>Outputs the HTML between the &lt;BODY&gt; and &lt;/BODY&gt; tags on the Layout and the Report pages</td>
</tr>
<tr>
<td>_MRVFRAMESET</td>
<td>Enables you to specify a custom &lt;FRAMESET&gt; tag on the Report page</td>
<td></td>
</tr>
<tr>
<td>_MRVNOPGOP</td>
<td>Nonblank</td>
<td>Turns off the paging feature</td>
</tr>
<tr>
<td>_MRVRNDX1</td>
<td>Value for the radio button that represents the first number of rows</td>
<td>The default is All</td>
</tr>
<tr>
<td>_MRVRNDX2</td>
<td>Value for the radio button that represents the second number of rows</td>
<td>The default is 25</td>
</tr>
<tr>
<td>_MRVRNDX3</td>
<td>Value for the radio button that represents the third number of rows</td>
<td>The default is 50</td>
</tr>
</tbody>
</table>
### MDDB Report Viewer Cascading Style Sheets

The MDDB Report Viewer’s cascading style sheet (CSS) properties enable you to customize the viewer output. You can use cascading style sheets to modify background colors, fonts, and the size and location of the HTML elements and to indicate whether the HTML elements are displayed. For more information about style sheet capabilities, consult your favorite HTML reference guide.

HTML elements use the CLASS parameter to surface style sheet properties. Table 4.3 on page 42 lists the CLASS definitions that are used by the MDDB Report Viewer. An example style sheet is shipped with the viewer, and you can create your own to use as well. To apply a style sheet to the viewer output, specify the CSS parameter as a hidden field on your initial HTML page. For example,

```
<INPUT TYPE="hidden" NAME="CSS" VALUE="http://myserver/mystyle.css">
```

You can also add the CSS parameter to the URL of bookmarked reports, as in the following (note the URL encoding):

```
&CSS=http%3A//myserver/mystyle.css
```

An additional CSST parameter is provided so that you can apply a separate style sheet to the toolbar frame. If you do not specify the CSST parameter, the toolbar frame uses the value that is specified by the CSS parameter.

### Table 4.3 MDDB Report Viewer CSS Class Tags

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAINTAB</td>
<td>Main report table</td>
</tr>
<tr>
<td>ROWLAB</td>
<td>Row label cells</td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TROWLAB</td>
<td>Total row label cell</td>
</tr>
<tr>
<td>STROWLAB</td>
<td>Total row label cell for expanded row (for example, &quot;subtotals&quot;)</td>
</tr>
<tr>
<td>TROWCELL</td>
<td>Total row data cells</td>
</tr>
<tr>
<td>TDCELL</td>
<td>All other data cells</td>
</tr>
<tr>
<td>TCOLLAB</td>
<td>Total column label cell</td>
</tr>
<tr>
<td>STCOLLAB</td>
<td>Total column label cell for nested totals</td>
</tr>
<tr>
<td>TCOLCELL</td>
<td>Total column data cells</td>
</tr>
<tr>
<td>COLLAB</td>
<td>Column label cells</td>
</tr>
<tr>
<td>EMPTY</td>
<td>Empty cell in upper left corner</td>
</tr>
<tr>
<td>FILTERBOX</td>
<td>Table containing filter list boxes</td>
</tr>
<tr>
<td>DIMBOX</td>
<td>Table containing dimension selector list boxes</td>
</tr>
<tr>
<td>DIMSELBOX</td>
<td>Table containing dimension selector list boxes (Report Layout page)</td>
</tr>
<tr>
<td>ANALYBOX</td>
<td>List box for selecting analysis variable (Report Layout page)</td>
</tr>
<tr>
<td>ANALYSIS</td>
<td>Class for the &lt;DIV&gt; tag for the analysis variable list box</td>
</tr>
<tr>
<td>STATSBOX</td>
<td>List box for selecting statistic (Report Layout page)</td>
</tr>
<tr>
<td>STATS</td>
<td>Class for the &lt;DIV&gt; tag for the statistics list box</td>
</tr>
<tr>
<td>ANALYCOL</td>
<td>Analysis variable column</td>
</tr>
<tr>
<td>STATSCOL</td>
<td>Statistics column</td>
</tr>
<tr>
<td>GRAPH</td>
<td>Class for the &lt;IMG&gt; tag for standard GIF graph</td>
</tr>
<tr>
<td>GRAPHAPP</td>
<td>Class for the graph application tag</td>
</tr>
<tr>
<td>TOOLTAB</td>
<td>Class for the toolbar</td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>IMGBKMRK</td>
<td>Class for the <code>&lt;IMG&gt;</code> tag for Bookmark</td>
</tr>
<tr>
<td>IMGDIM</td>
<td>Class for the <code>&lt;IMG&gt;</code> tag for Dimensions</td>
</tr>
<tr>
<td>IMGOPT</td>
<td>Class for the <code>&lt;IMG&gt;</code> tag for Options</td>
</tr>
<tr>
<td>IMGHELP</td>
<td>Class for the <code>&lt;IMG&gt;</code> tag for Help</td>
</tr>
<tr>
<td>IMGLAY</td>
<td>Class for the <code>&lt;IMG&gt;</code> tag for Layout</td>
</tr>
<tr>
<td>IMGLOGOUT</td>
<td>Class for the <code>&lt;IMG&gt;</code> tag for Logout</td>
</tr>
<tr>
<td>IMGROTATE</td>
<td>Class for the <code>&lt;IMG&gt;</code> tag for Rotate</td>
</tr>
<tr>
<td>HEADER</td>
<td>Report Layout HTML headers</td>
</tr>
<tr>
<td>LABEL</td>
<td>Report Layout HTML labels</td>
</tr>
<tr>
<td>SELECT</td>
<td>Report Layout HTML for <code>&lt;SELECT&gt;</code> and <code>&lt;INPUT&gt;</code> tags</td>
</tr>
<tr>
<td>SSELECT</td>
<td>Class for statistics selection list boxes</td>
</tr>
<tr>
<td>SUBMIT</td>
<td>Submit (View Report) button class</td>
</tr>
</tbody>
</table>

**Dictionary**

**_BUILD_ACROSSL_LIST_ Method**

Builds the across list (variables in the across dimension) on the application list.

**Syntax**

```
CALL SEND(OBJID,'_BUILD_ACROSSL_LIST_','application-list,across-variable');
```

**Required Arguments**

**application-list**

the list ID of the application list. For more information about application lists, see the online Help for SAS/EIS software.

| Type   | Numeric   |
**across-variable**

the variable that is selected for the across dimension (optional and no longer used).

| Type   | Character |

**Details**

This method

- clears the across sublist on the application list
- adds the selected across variables to the across sublist.

**Example**

```plaintext
acrosvar='Product Line';
rc=insertc(acrvars_,acrosvar,-1);
applist=makelist();
rc=fillist('CATALOG','SASHELP.EISRG.ONEWAY.EIS',applist);
call send(webid,'_BUILD_ACROSSLIST_',applist);
```

The following sublist is added to the application list:

```
Across: (PRODUCT LINE= (HIERARCH='Product Line'))
```

---

**_BUILD_ANALYSIS_LIST_ Method**

Builds the analysis sublist on the application list.

**Syntax**

```plaintext
CALL SEND(OBJID,'_BUILD_ANALYSIS_LIST_','application-list');
```

**Required Argument**

| **application-list** |

the list ID of the application list. For more information about application lists, see the online Help for SAS/EIS software.

| Type   | Numeric |

**Details**

This method

- clears the analysis sublist on the application list
- adds the selected analysis variables to the analysis sublist.
Example

```plaintext
applist = makelist();
rc = fillist('CATALOG','SASHELP.EISRG.ONEWAY.EIS',applist);
call send(webid,'_BUILD_ANALYSIS_LIST_',applist);

The following sublist is added to the application list:

Analysis:{ ACTUAL= ()[1083] }[985]
```

_BUILD_ANLSORTORDER_ Method

Updates the ANLSORTORDER sublist on the application list that is used to specify an analysis/statistic column sort.

Syntax

```plaintext
CALL SEND(OBJID,'_BUILD_ANLSORTORDER_','application-list');
```

Required Argument

**application-list**

the list ID of the application list. For more information about application lists, see the online Help for SAS/EIS software.

Type Numeric

Example

```plaintext
applist = makelist();
rc = fillist('CATALOG','SASHELP.EISRG.ONEWAY.EIS',applist);
call send(webid,'_BUILD_ANLSORTORDER_',applist);
```

_BUILD_APPLICATION_LIST_ Method

Builds the application list for the data model.

Syntax

```plaintext
CALL SEND(OBJID,'_BUILD_APPLICATION_LIST_','application-list,metabase-id, catalog-entry,down-variable,across-variable');
```

Required Arguments

**application-list**

the list ID of the application list.
Details

This method

- copies the Report Gallery Template application list
- changes the table name on the application list to the selected MDDB
- replaces the metabase name on the application list with the selected metabase
- calls _BUILD_DOWNL_LIST to add the selected down variables to the application list
- calls _BUILD_ACROSSLIST to add the selected across variables to the application list (if necessary)
- calls _BUILD_ANALYSIS_LIST to add the selected analysis variables to the application list
- calls _BUILD_STATS_LIST to add the selected statistics to the application list
- calls _CLEAR_POPUP_ to clear the unneeded popup_l sublist on the application list
- calls _BUILD_TOTAL_ to turn report totals on for the down variables
- calls _BUILD_TOTAL_ to turn report totals on for the across variables (if necessary)

For more information about the structure of application lists, see the online Help for SAS/EIS software.

Example

```plaintext
applist= makelist();
mbid= instance(loadclass('SASHELP.MB.METABASE.CLASS'));
centry= 'SASHELP.EISRG.ONEWAY.EIS';
```
downvar = 'Geographic';
rc=insertc(downvars_, downvar, -1);
acrosvar = 'Year';
rc=insertc(acrvars_, acrosvar, -1);
call send(webid,'_BUILD_APPLICATION_LIST_',applist,mbid,centry);

_EVENTS.Outbox.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.

_DOWNL_LIST_ Method
Builds the DOWNL sublist on the application list.
Syntax
CALL SEND(OBJID,'_BUILD_DOWNL_LIST_+',application-list,down-variable);

Required Arguments

application-list
the list ID of the application list. For more information about application lists, see the online Help for SAS/EIS software.

Type Numeric

down-variable
the selected down variable. (This optional parameter is included for compatibility with previous releases of the MDDB Report Viewer.)

Type Character

Details
This method
- clears the down sublist on the application list
- adds the selected down variable to the down sublist.

Example

applist= makelist();
rc= fillist('CATALOG','SASHELP.BISRG.ONEWAY.EIS',applist);
downvar= 'Geographic';
rc= insertc(downvars_,downvar, -1);
call send(webid,'_BUILD_DOWNL_LIST_+',applist);

The following sublist is added to the application list:

downl: ( GEOGRAPHIC= ( HIERARCH= 'Geographic' )[2453] )[2367]

_BUILD_STATSL_LIST_ Method
Builds the STATSL_ sublist on the application list.

Syntax
CALL SEND(OBJID,'_BUILD_STATSL_LIST_+',application-list);
Required Argument

**application-list**

the list ID of the application list. For more information about application lists, see the online Help for SAS/EIS software.

*Type* Numeric

Details

This method
- clears the statistics sublist on the application list
- adds the selected statistics to the statistics sublist.

Example

```plaintext
applist= makelist();
rc= fillist('CATALOG','SASHELP.EISRG.ONEWAY.EIS',applist);
call send(webid,'_BUILD_STATSL_LIST_',applist);

The following sublist is added to the application list:

  statsl: { SUM= 'SUM'
              ...
    }[2445]
```

__BUILD_TOTAL__ Method

Builds the TOTALS sublist on the application list to turn report totals on.

Syntax

```plaintext
CALL SEND(OBJID,'_BUILD_TOTAL_',application-list,metabase-id,total-variable);
```

Required Arguments

**application-list**

the list ID of the application list. For more information about application lists, see the online Help for SAS/EIS software.

*Type* Numeric

**metabase-id**

the ID number of the metabase.

*Type* Numeric

**total-variable**

the variable that is selected from the down or across dimension.

*Type* Character
Example

```plaintext
applist= makelist();
rc=fillist('CATALOG', 'SASHELP.EISRG.ONEWAY.EIS', applist);
mbid=instance(loadclass('SASHELP.MB.METABASE.CLASS'));
downvar='COUNTRY';
call send(webid,'_BUILD_TOTAL_',applist,mbid,downvar);
```

The following sublist is added to the application list:

```plaintext
TOTALS: 
  DSNAME= 'SASHELP.PRDMDDB'
  MBNAME= 'SASHELP.MBEIS'
  SEL_EXCL= 'CATEGORY'
  MB_AVAIL= 1
  CUSTOM= ( COUNTRY= ( TOTALON= 1
                LABEL= 'TOTAL'
                FONT= ()[1095]
              )[1093]
            )[1063]
)[1061]
```

_BUILD_URL_ONSUBMIT_ Method

Outputs the geturl JavaScript function on the Dimensions page.

Syntax

```plaintext
CALL SEND(OBJID,'_BUILD_URL_ONSUBMIT_','url');
```

Required Argument

**url**

the Application Broker component of the URL.

Type Character

Details

This function runs when the View Report button is pressed. It builds the URL for the report request.

Example

Sample output:

```plaintext
function geturl(down,across,analysis) {
  D0=0; A0=0; AC0=0; var href="../mddbapp.hlp/"; var stats="";
  param=new Object;
  param._SERVICE = "default";
  param._PROGRAM = "sashelp.webeis.showrpt.scl";
  param._DEBUG = "2";
  param.MDDB = "SASHELP.PRDMDDB";
```
param.METABASE = "SASHHELP";
param.CSS = "http://localhost/css/mrv.css";
param.GRT = "NONE";
param.DC = "1";
param.ACB = "1";
param.ST = "1";

href = "/cgi-bin/broker.exe?";

for (name in param) { href += name + "=" + param[name] + "&" }

href2="";

for (i=0; i<down.options.length; i++) {
    if (down.options[i].selected) {
        D0=eval(D0+1);
        href2+="&D" +D0 +"=" +down.options[i].value;
        if (eval(D0)==1) {
            href2+="&D" +"=" +down.options[i].value;
        }
    }
}

href="D0=" +D0 +href2;

href2="";

for (i=0; i<across.options.length; i++) {
    if (across.options[i].selected & across.options[i].value!="") {
        AC0=eval(AC0+1);
        href2+="&AC" +AC0 +"=" +across.options[i].value;
        if (eval(AC0)==1) {
            href2+="&AC" +"=" +across.options[i].value;
        }
    }
}

href="&AC0=" +AC0 +href2;

href2="";

for (i=0; i<analysis.options.length; i++) {
    if (analysis.options[i].selected) {
        A0=eval(A0+1);
        href2+="&A" +A0 +"=" +analysis.options[i].value;
        if (eval(A0)==1) {
            href2+="&A" +"=" +analysis.options[i].value;
        }
        stats=analysis.options[i].value+"STATS";
        statsarray=eval(stats);
        if (statsarray.length==1 & statsarray[0]=="nunique") {
            href2+="&A" +A0 +"S=" +"NUNIQUE";
        }
        else if (statsarray.length==1 & statsarray[0]!="nunique") {
            href2+="&A" +A0 +"S=" +"SUM";
        }
        else {
            if (statsarray.length == 2) {
                href2+="&A" + A0 + "S=" + statsarray[1];
            }
        }
    }
}
else {
    for (j=1; j<statsarray.length; j++)
        href2+="&A" +A0 +"S" +j +"=" +statsarray[j];
    }
    }
    }
    href="&A0=" +A0 +href2;
    return href;
}

_BUILD_WHERE_FORMAT_STRING_ Method

Builds a portion of the WHERE clause that provides the reach-through to detail data, including the variable format.

Syntax

CALL SEND(OBJID,'_BUILD_WHERE_FORMAT_STRING_','metabase-id',variable-
          name,'in-data-value','out-data-value');

Required Arguments

metabase-id
    the ID number of the metabase.
    Type  Numeric

variable-name
    the name of the variable in the metabase.
    Type  Character

in-data-value
    the unformatted data value.
    Type  Character

out-data-value
    the string to add to the reach-through WHERE clause.
    Type  Character

Example

mbid=instance (loadclass('SASHELP.MB.METABASE.CLASS'));
myvar='MONTH';
myvalue='Jan';
fmtval=' ';
call send (webid,'_BUILD_WHERE_FORMAT_STRING_','mbid,myvar,myvalue,fmtval);
The following output is produced:

```plaintext
fmt.val=put(MONTH, $MONTH.) = 'Jan'
```

---

**_CHECK_HIER_MEMBER_ Method**

Checks to make sure that one dimension variable (member-variable) is not a member of the hierarchy chosen for the other dimension variable (hierarchy-variable).

---

**Syntax**

```plaintext
CALL SEND(OBJID, '_CHECK_HIER_MEMBER_', metabase-id, error-flag, hierarchy-variable, member-variable, message);
```

**Required Arguments**

- **metabase-id**
  - the ID number of the metabase.
  - Type: Numeric

- **error-flag**
  - an error flag, where 0=no error, and 1=error.
  - Type: Numeric

- **hierarchy-variable**
  - the hierarchy variable.
  - Type: Character

- **member-variable**
  - the member variable.
  - Type: Character

- **message**
  - the error message to display.
  - Type: Character

**Details**

This method ensures that the variables users select to create a report are valid. For example, specifying `DOWN=COUNTRY, ACROSS=GEOGRAPHIC` produces an error if country is a member of the geographic hierarchy.

**Example**

```plaintext
mbid=instance(loadclass('SASHELP.MB.METABASE.CLASS'));
downvar='Geographic';
```
acrosvar='COUNTRY';
call send(webid,'_CHECK_HIER_MEMBER_',mbid,varerr,downvar,acrosvar,msg);

_CLOSE_FORM_ Method

Outputs the closing variable selection form tags.

Syntax

CALL SEND(OBJID,'_CLOSE_FORM_','initial-url,service-name, metabase-name, background-type,background-value, title,webeis-class);

Required Arguments

initial-url
the URL of the initial HTML page.
Type Character

service-name
the Application Broker service value.
Type Character

metabase-name
the metabase name.
Type Character

background-type
an optional background type (IMAGE or COLOR).
Type Character

background-value
an optional background value.
Type Character

title
the HTML page title.
Type Character

webeis-class
the WEBEIS class name.
Type Character

Details

This method outputs
the <FORM> tag
the link back to the initial HTML page.

Example

```sas
mddblink= 'http://www.test.com/mddbpage.html';
service= 'default';
metabase= 'SASHELP.MBEIS';
bgtype= 'COLOR';
bg= 'YELLOW';
title= 'Third Quarter Sales Reports';
webcls= 'SASHELP.WEBCAT.MYWEB.CLASS';
call send(webid,'_CLOSE_FORM_',mddblink,service,metabase,
  bgtype,bg,title,webcls);
```

The following output is produced:

```html
</TD></TR>
</FORM>
</TD></TR>
```

__CLOSE_PAGE__ Method

Outputs the </TABLE>, </BODY>, and </HTML> tags.

Syntax

```sas
CALL SEND(OBJID,'_CLOSE_PAGE_');
```

Example

The following output is produced:

```html
</TABLE>
</BODY>
</HTML>
```

__CLOSE_STATIC_FORM_ Method

Outputs the Next button and the closing </TABLE>, </FORM>, </BODY>, and </HTML> tags for the initial HTML page.

Syntax

```sas
CALL SEND(OBJID,'_CLOSE_STATIC_FORM_');
```
Example

The following output is produced:

```
<TD>&nbsp;</TD><TD>&nbsp;</TD>
<TD colspan=2 align=center>
<INPUT TYPE= "submit" VALUE= "Next">
</TD>
</TABLE>
</FORM>
</BODY>
</HTML>
```

.CREATE_STAT_ARRAYS_ Method

Outputs the stats JavaScript function and the associated statistics JavaScript arrays on the Dimensions page.

Syntax

```
CALL SEND(OBJID,'_CREATE_STAT_ARRAYS_');
```

Details

This function updates the list of displayed available and selected statistics based on the selected analysis variable.

Example

The following output is produced:

```
var ACTUALSTATS= new Array('analysis','NMISS','N','SUM','MIN','MAX','USS','RANGE','AVG','CSS','VAR','STD','STDDER','CV','Z','PRT','LCLI','UCLI','PCTSUM','PCTN');
```
var DIFFSTATS= new Array(
    "computed",
    "MAX",
    "MIN",
    "PCTN",
    "PCTSUM",
    "SUM",
    "N"
);

var PREDICTSTATS= new Array(
    "analysis",
    "NMISS",
    "N",
    "SUM",
    "MIN",
    "MAX",
    "USS",
    "RANGE",
    "AVG",
    "CSS",
    "VAR",
    "STD",
    "STDERR",
    "CV",
    "T",
    "PRT",
    "LCLM",
    "UCLM",
    "PCTSUM",
    "PCTN"
);

var SALESRATSTATS= new Array(
    "computed",
    "MAX",
    "MIN",
    "PCTN",
    "PCTSUM",
    "SUM",
    "N"
);

var statslabellist = new Array();
statslabellist["SUM"]="Sum";
statslabellist["PCTSUM"]="Percent of Sum";
statslabellist["AVG"]="Average";
statslabellist["N"]="Total Number of Nonmissing Values";
statslabellist["PCTN"]="Percent of Total Number";
statslabellist["MIN"]="Minimum";
statslabellist["MAX"]="Maximum";
statslabellist["RANGE"]="Range";
statslabellist["NMISS"]="Total Number of Missing Values";
statslabellist["STD"]="Standard Deviation";
statslabellist["STDERR"]="Standard Error of Mean";
statslabellist["LCLM"]="Lower Confidence Limit";
statslabellist["UCLM"]="Upper Confidence Limit";
statslabellist["USS"]="Uncorrected Sum of Squares";
statslabellist["CSS"]="Corrected Sum of Squares";
statslabellist["VAR"]="Variance";
statslabellist["CV"]="Coefficient of Variation";
statslabellist["T"]="T Value";
statslabellist["PRT"]="Probability of Greater Absolute Value";
statslabellist["SUMWGT"]="Sum of Weights";
statslabellist["UWSUM"]="Unweighted Sum";
statslabellist["NUNIQUE"]="Nunique";
statslabellist["MIXED"]="*MIXED SELECTIONS";

analysisdesclist = new Array(
  "SUM",
  "PCTSUM",
  "AVG",
  "N",
  "PCTN",
  "MIN",
  "MAX",
  "RANGE",
  "NMISS",
  "STD",
  "STDERR",
  "LCLM",
  "UCLM",
  "USS",
  "CSS",
  "VAR",
  "CV",
  "T",
  "PRT",
  "SUMWGT",
  "UWSUM"
);

computeddesclist = new Array(
  "MAX",
  "MIN",
  "PCTN",
  "PCTSUM",
  "SUM",
  "N",
);

cnuniquedesclist = new Array(
  "SUM"
);

nuniquedesclist = new Array(
  "NUNIQUE"
);

var vararrayname = new Array();
num = 0;
function stats(select, statbox) {
    var vararrayname="";
    var varstatsstring="";
    var allstatsstring="";
    for (i=0; i < select.options.length; i++) {
        if (select.options[i].selected) {
            vararrayname=select.options[i].value+"STATS";
            varstatsstring=eval(vararrayname).toString();
            if (num==1) {
                varstatsstring=eval(vararrayname)[0];
                for (j=0; j < statbox.length; j++) {
                    if (statbox.options[j].text !="")
                        varstatsstring="", +statbox.options[j].value;
                }
            } else {
                allstatsarray=eval(vararrayname[0]+"desclist");
                allstatsstring=allstatsarray.toString();
                if (""!=statbox.options[j].text !="" & ""MIXED SELECTIONS" !=statbox.options[j].text & -1==varstatsstring.indexOf(statbox.options[j].value)) {
                    varstatsstring="", +statbox.options[j].value ;
                }
            }
        }
    }
    temparray=varstatsstring.split(",");
    if ("ACTUALSTATS"==vararrayname) {
        ACTUALSTATS.length=temparray.length;
        for (k=0; k < temparray.length; k++)
            ACTUALSTATS[k]=temparray[k];
    } else if ("DIFFSTATS"==vararrayname) {
        DIFFSTATS.length=temparray.length;
        for (k=0; k < temparray.length; k++)
            DIFFSTATS[k]=temparray[k];
    } else if ("PREDICTSTATS"==vararrayname) {
        PREDICTSTATS.length=temparray.length;
        for (k=0; k < temparray.length; k++)
            PREDICTSTATS[k]=temparray[k];
    } else if ("SALESRATSTATS"==vararrayname) {
        SALESRATSTATS.length=temparray.length;
        for (k=0; k < temparray.length; k++)
            SALESRATSTATS[k]=temparray[k];
    }
}
//STATS
_DISPLAY_ACROSS_CELLS_ Method
Displays the values for the across dimension.

Syntax
CALL SEND(OBJID,'_DISPLAY_ACROSS_CELLS_*,column-list,actions-list,view-report-flag,
analysis-variable,statistic-variable,across-variable,url,argument-string,argument-string-2,
initial-url,background-type,background-value,title,webeis-class,dlflag,service);

Required Arguments

**column-list**
the column list from the EMDDB_M class.

Type Numeric

**actions-list**
the actions sublist for drill down.

Type Numeric

**view-report-flag**
the flag for the View Report button.

Type Numeric

**analysis-variable**
the analysis variable to graph.

Type Character

**statistic**
the statistic to graph.

Type Character

**across-variable**
the analysis variable for graphing.

Type Character

**url**
the Application Broker component of the URL.

Type Character

**argument-string**
the argument string for the next query.

Type Character
argument-string-2
the argument string for the next query.

Type Character

initial-url
the URL of the initial HTML page. (This parameter is obsolete. It is included in
the METHOD statement so that overrides are not broken.)

Type Character

background-type
the optional background type (IMAGE or COLOR).

Type Character

background-value
the optional background value.

Type Character

title
the HTML page title.

Type Character

webeis-class
the WEBEIS class name.

Type Character

dlflag
a flag that indicates whether to download the table to a spreadsheet, where
0=output HTML tags with data values, and 1=output data values with
spreadsheet delimiters.

Type Numeric

service
the service name.

Type Character

Details
This method
- calls the _SET_ACTIVE_VALUE_ method of the EMDDB_M class
- calls the _SET_ACTION_STATUS_ method of the EMDDB_M class
- outputs the class values for the across dimension with <A> tags for drill down, if
  drill down is valid.

Example
emddbmid= instance(loadclass('SASHELP.WEBEIS.EMDDB_M.CLASS');
collist= makelist());
call send(emddbmid,'_GET_CLASS_COMBINATIONS_','COL',collist);
actionsl= makelist();
rc= insertc(actionsl,'',-1,'CL_DRILL');
vrflag= 1;
grphvar= 'Actual Sales';
grphstat= 'Sum';
grphacr= 'Month';
url= 'cgi-bin/broker?_PROGRAM=SASHELP.WEBEIS.MDDBRPTS.SCL&_SERVICE=
default&_debug=&0GRT=BLOCK';
args= '&MDDB=SASUSER.SALES&METABASE=SASUSER.NEWMB&D=COUNTRY&AC=
=MONTH&A0=2&A1=ACTUAL&A2=PREDICT';
args2= '&S0=2&S1=SUM&S2=AVG';
btype= 'COLOR';
bg= 'YELLOW';
title= '1996 Sales Reports';
webcls= 'SASHELP.WEBEIS.WEBEIS.CLASS';
dlflag=0;
service='DEFAULT';
call send(webid,'_DISPLAY_ACROSS_CELLS_',collist,actionsl,vrflag,grphvar,
grphstat,grphacr,url,args,args2,' ',btype,bg,title,webcls,dlflag,service);

The following output is produced:

```
<table>
<thead>
<tr>
<th>Month</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
<th>TOTAL</th>
</tr>
</thead>
</table>
```

(DISPLAY_ANALYSIS_VARS_ Method)

Outputs the chosen analysis variables to the report table.

Syntax

CALL SEND(OBJID,'_DISPLAY_ANALYSIS_VARS_'.column-list,dlflag);

Required Arguments

**column-list**

the column list from the _EMDDBB_M_ class.
**dlflag**
a flag that indicates whether to download the table to a spreadsheet.

**Example**
The following output is produced:

```html
<TR>
  <TH COLSPAN=2 CLASS="analycol">
  <DIV CLASS="analysis">
    <SELECT NAME="A" CLASS="ANALYBOX" onChange="submit();">
      <OPTION SELECTED VALUE=ACTUAL> Actual Sales
      <OPTION VALUE=DIFF> Sales Lag
      <OPTION VALUE=PREDICT> Predicted Sales
      <OPTION VALUE=SALESRAT> Sales Ratio
    </SELECT>
  </DIV>
  </TH>
  <TH COLSPAN=2 CLASS="analycol">
  <DIV CLASS="analysis">
    <SELECT NAME="A" CLASS="ANALYBOX" onChange="submit();">
      <OPTION VALUE=ACTUAL> Actual Sales
      <OPTION VALUE=DIFF> Sales Lag
      <OPTION SELECTED VALUE=PREDICT> Predicted Sales
      <OPTION VALUE=SALESRAT> Sales Ratio
    </SELECT>
  </DIV>
  </TH>
  <TH COLSPAN=2 CLASS="analycol">
  ACTUAL SALES
  </TH>
  <TH COLSPAN=2 CLASS="analycol">
  PREDICTED SALES
  </TH>
</TR>
```

---

**_DISPLAY_DEFAULT_TITLE_ Method**
Displays the user-specified title.

**Syntax**

```c
CALL SEND(OBJID,'_DISPLAY_TITLE_','dlflag');
```
Required Argument

**dlflag**
- a flag that indicates whether to download the table to a spreadsheet, where 
  0=output HTML tags with data values and 1=output data values with 
  spreadsheet delimiters.
  
  **Type** Numeric

Details

This method
- gets the default title value from the DT macro variable
- outputs the title in HTML format or in comma-separated format, depending on 
  the value of dlflag.

Example

```plaintext
dflag=0;
call send(webid,'_DISPLAY_DEFAULT_TITLE_','dflag');

The following output is produced:

<H2>1998 Sales Reports </H2>
```

__DISPLAY_DOWNVAR_CELL__ Method

Displays the down dimension.

Syntax

```plaintext
CALL SEND(OBJID,'_DISPLAY_DOWNVAR_CELL_','row-list,vrflag,analysis-
variable,
statistic-variable,down-variable, across-variable,_url,_argument-string,_argument-
string-2,
initial-url,service-name,url,background-type, background-value,title,webesis-
class,dlflag);
```

Required Arguments

**row-list**
- the row list from the EMDBB_M class.
  
  **Type** Character

**vrflag**
- a flag that indicates that the View Report button was pressed.
  
  **Type** Character
**analysis-variable**
the analysis variable to graph.

Type Numeric

**statistic-variable**
the statistic to graph.

Type Character

**down-variable**
the down dimension variable to graph.

Type Character

**across-variable**
The across dimension variable to graph.

Type Character

**url**
the web browser component of the URL.

Type Character

**argument-string**
the argument string for the next query.

Type Character

**argument-string-2**
the argument string for the next query.

Type Character

**initial-url**
the URL of the initial HTML page.

Type Character

**service-name**
the Application Broker service.

Type Character

**url**
the Application Broker component of the URL.

Type Character

**background-type**
the background type (IMAGE or COLOR). This value is optional.

Type Character

**background-value**
the background value. This value is optional.

Type Character
**title**
the title. This value is optional.

**web-class**
the WEBEIS class name (for subclassing).

**dlflag**
a flag that indicates whether to download the table to a spreadsheet.

Details
If the user has drilled down, this method displays the down dimension cell with an up arrow. This method
- calls _GET_CLASS_LABEL_ of the data model to get the cell label
- outputs the labeled cell with an arrow (if necessary) for drilling up.

Example

call send(emddbmid_,'_GET_CLASS_COMBINATIONS_','ROW',rowlist);
vrlflag=1;
grphvar='Actual+Sales';
grphstat='Sum';
grphdown='';
grphacr='PRODTYPE';
_url='/cgi-bin/broker?_PROGRAM=sashelp.webesi.mddbrpts.scl&_SERVICE=default &_DEBUG=0&RPTTYPE=2&GRTYPE=BLOCK' 
_args='&MDDB=PERMDATA.MAPINFO&METABASE=PERMDATA.MB612&DOWN=Geographic&ACROSS =Product+Lines&A=ACTUAL' 
_args2 = '&S=SUM&V1=COUNTRY=U.S.A.' 
mddblink='http://myserver.com/test.html'; 
service='default'; 
url='/cgi-bin/broker'; 
bgtype=''; 
bg=''; 
title=''; 
webcls='SASHHELP.WEBEIS.WEBEIS'; 
dlflag=0; 
call send(webid,'_DISPLAY_DOWNVAR_CELL_','',rowlist,vrflag,grphvar, 
grphstat,grphdown,grphacr,_,url,_,args,_,args2,mddblink,service,url, 
bgtype,_,title,webcls,dlflag);

This example produces the following output:

```
<TR><TH CLASS="rowlab">State/Province</TH><TH CLASS=collab"><A 
HREF="/cgi-bin/broker?_PROGRAM=sashelp.webesi.mddbrpts.scl&_SERVICE=default 
&_DEBUG=0&RPTTYPE=2&GRTYPE=BLOCK&MDDB=PERMDATA.MAPINFO&METABASE=PERMDATA.MB612 
&DOWN=Geographic&ACROSS=Product+Line&A=ACTUAL&S=SUM&GVAR=Actual+Sales&GSTAT=Sum 
&GACR=PRODTYPE&GLINK=1&DRUP=1&_MDLINK=http://myserver.com/test.html 
&CLASS=SASHHELP.WEBEIS.WEBEIS* TARGET="_TOP"><IMG 
SRC="/myimages/up.gif BORDER=0 ALT="UP"></A></TH>
```
DISPLAY_ERROR_ Method
Displays an error message on dynamic pages.

Syntax
CALL SEND(OBJID,'_DISPLAY_ERROR_',error-message);

Required Argument
error-message
the error message to display.
Type Character

Example
The following output is produced:

<HTML>
<BODY BGCOLOR=SILVER> <CENTER>
<BR> <BR> <BR>
<H1>Analysis Variable Required </H1>
</BODY>
</HTML>

DISPLAY_ONEWAY_ Method
Calls methods to produce one-way tabular reports.

Syntax
CALL SEND(OBJID,'_DISPLAY_ONEWAY_','dlflag');

Required Argument
dlflag
a flag that indicates whether to download the table to a spreadsheet.
Type Numeric

Details
This method
- checks for selected down and analysis variables and statistics
- calls the _OPEN_ method of the metabase
- calls the _GET_HIERARCHY_ method of the metabase to get the list of hierarchies
- calls the _BUILD_APPLICATION_LIST_ method
- calls the _BUILD_ARGS_STRING_ method
- calls the _BUILD_ARGS2_STRING_ method
- calls the _GET_VARIABLES_ method of the metabase class to get the list of analysis variables
- calls the _SET_DRILL_LEVELS_ method, if necessary, to drill down to the current level
- calls the _SET_APPLICATION_ method of the data model
- calls the _EXPAND_VALUE_ method of the data model for all expanded variables to request the expanded data values
- calls the _GET_CLASS_COMBINATIONS_ method of the data model to get the row list
- calls the _GET_CLASS_COMBINATIONS_ method of the data model to get the column list
- calls the _OUTPUT_DOWN_LIST_ method to output the list of down variables and outputs the HTML tags to format the selection list
- calls the _OPEN_ONEWAY_ method
- calls the _DISPLAY_ANALYSIS_VARS_ method
- calls the _DISPLAY_DOWNVAR_CELL_ method
- calls the _DISPLAY_STATISTIC_VARS_ method
- calls the _DISPLAY_VALUES_ method.

_DISPLAY_ONEWAY_BLOCK_ Method

Submits the PROC GCHART statements to produce the one-way block chart.

Syntax

CALL SEND(OBJID,’_DISPLAY_ONEWAY_BLOCK_’,statistic,analysis-variable,down-variable,data-set-name.gif-device);

Required Arguments

**statistic**
the statistic to graph.

  Type Character

**analysis-variable**
the analysis variable to graph.
**_DISPLAY_ONEWAY_HBAR_ Method**

Submits the PROC GCHART statement to produce the one-way horizontal bar chart.

**Syntax**

```
CALL SEND(OBJID,'_DISPLAY_ONEWAY_HBAR_','statistic,analysis-variable,
down-variable,data-set-name,gif-device');
```

**Required Arguments**

- **statistic**
  - the statistic to graph.
  - Type: Character

- **analysis-variable**
  - the analysis variable to graph.
  - Type: Character

- **down-variable**
  - the down dimension variable to graph.
  - Type: Character

- **data-set-name**
  - the data set name from the _WRITE_ method.
  - Type: Character

- **gif-device**
  - the device driver name.
  - Type: Character
_DISPLAY_ONEWAY_PIE_ Method

Submits the PROC GCHART statement to produce the one-way pie chart.

Syntax

CALL SEND(OBJID,'_DISPLAY_ONEWAY_PIE_','statistic,analysis-variable,
down-variable,data-set-name.gif-device);

Required Arguments

statistic
the statistic to graph.
Type Character

analysis-variable
the analysis variable to graph.
Type Character

down-variable
the down dimension variable to graph.
Type Character

data-set-name
the data set name from the _WRITE_ method.
Type Character

gif-device
the device driver name.
Type Character

_DISPLAY_ONEWAY_VBAR_ Method

Submits the PROC GCHART statement to produce the one-way vertical bar chart.

Syntax

CALL SEND(OBJID,'_DISPLAY_ONEWAY_VBAR_','statistic,analysis-variable,
down-variable,data-set-name.gif-device);
Required Arguments

**statistic**
the statistic to graph.
Type Character

**analysis-variable**
the analysis variable to graph.
Type Character

**down-variable**
the down dimension variable to graph.
Type Character

**data-set-name**
the data set name from the _WRITE_ method.
Type Character

**gif-device**
the device driver name.
Type Character

__DISPLAY_STATISTIC_VARS__ Method
Outputs the selected statistics to the report table.

Syntax

CALL SEND(OBJID,'_DISPLAY_STATISTIC_VARS_','column-list,analysis-variable,_url,
_argument-string,_argument-string-2,initial-url,URL,service,background-type,
background-value,title,webeis-class,dlflag,row-list);

Required Arguments

**column-list**
the column list from the EMDBB_M class.
Type Numeric

**analysis-variable**
the analysis variable to graph.
Type Numeric

**_url**
the URL of the next query.
_DISPLAY_STATISTIC_VARS_ Method

Type Character

_argument-string
the argument string for the next query.

_Type_ Character

_argument-string-2
the argument string for the next query.

_Type_ Character

_initial-url
the URL of the initial HTML page.

_Type_ Character

_initial-url
the URL of the initial HTML page.

_Type_ Character

_URL
the Application Broker component of the URL.

_Type_ Character

_service
the Application Broker service.

_Type_ Character

_service
the Application Broker service.

_Type_ Character

_background-type
the background type (IMAGE or COLOR). This value is optional.

_Type_ Character

_background-value
the background value. This value is optional.

_Type_ Character

_title
the HTML page title.

_Type_ Character

_title
the HTML page title.

_Type_ Character

_webesis-class
the WEBEIS class name.

_Type_ Character

_webesis-class
the WEBEIS class name.

_Type_ Character

_dlflag
a flag that indicates whether to download the table to a spreadsheet, where
0=output HTML tags with data values and 1=output data values with
spreadsheet delimiters. This parameter is optional.

_Type_ Numeric

_dlflag
a flag that indicates whether to download the table to a spreadsheet, where
0=output HTML tags with data values and 1=output data values with
spreadsheet delimiters. This parameter is optional.

_Type_ Numeric

_row-list
the rowlist from the _GET_CLASS_COMBINATIONS_ method. This parameter
is optional.

_type_ Character

_row-list
the rowlist from the _GET_CLASS_COMBINATIONS_ method. This parameter
is optional.
Details

This method outputs

- a `<TH>` tag for each statistic in the column list
- a selection list of statistics on the first occurrence of each selected statistic
- an `<A>` tag followed by an `<IMAGE>` tag for each statistic if the standard GIF graph is displayed.

Example

```sas
call send(emddbmid_,'_GET_CLASS_COMBINATIONS_','COL',collist);
call send(emddbmid_,'_GET_CLASS_COMBINATIONS_','ROW',rowlist);
_url='/cgi-bin/scripts?_PROGRAM=SASHELP.WEBEIS.MDDBRPTS.SCL&_SERVICE=default
&_DEBUG=0&RPTTYPE=1&GRTYPE=BLOCK';
_args='&MDDB=PERMDATA.MAPINFO&METABASE=PERMDATA.MB612&DOWN=Geographic&A=ACTUAL';
_args2='&S0=2&S1=SUM&S2=PCTSUM';
mddblink='DYNAMIC';
_url='/cgi-bin/broker';
service='default';
bgtype='color';
bg='yellow';
title='';
webcls='SASHELP.WEBEIS.WEBEIS';
call send(webid,'_DISPLAY_STATISTIC_VARS_',collist,'',_url,_args,
_args2,mddblink,url,service,bgtype,bg,title,webcls,dlflag,rowlist);
```

The example produces the following output:

```
<TH CLASS="statscol" VALIGN=BOTTOM><DIV CLASS="stats">
<select NAME="s" CLASS="statsbox" onChange="submit();">
<option VALUE="SUM" SELECTED>Sum
<option VALUE="PCTSUM">% of Sum
<option VALUE="AVG">Average
<option VALUE="N">Total Count
<option VALUE="PCTN">% of Total 
<option VALUE="MIN">Minimum
<option VALUE="MAX">Maximum
<option VALUE="RANGE">Range
</select>
</DIV>
</TH>

<TH CLASS="statscol" VALIGN=BOTTOM><DIV CLASS="stats">
<select NAME="s" CLASS="statsbox" onChange="submit();">
<option VALUE="SUM">Sum
<option VALUE="PCTSUM" SELECTED> % of Sum
<option VALUE="AVG">Average
<option VALUE="N">Total Count
<option VALUE="PCTN">% of Total 
<option VALUE="MIN">Minimum
<option VALUE="MAX">Maximum
<option VALUE="RANGE">Range
</select>
</DIV>
</TH>
```
**DISPLAY_SUBSET_TITLE** Method

Displays the applied subsets in a title.

**Syntax**

CALL SEND(OBJID,'DISPLAY_SUBSET_TITLE',dlflag);

**Required Argument**

*dlflag*

a flag that indicates whether to download the data to a spreadsheet. This parameter is optional.

**Example**

The following output is produced:

```html
<TABLE><TR><TD><STRONG>Filter by: Country=Canada,
Germany Month=Jan,Apr,May</STRONG></TD></TR></TABLE>
```

**DISPLAY_TITLE** Method

 Displays the drill titles above the tabular report.

**Syntax**

CALL SEND(OBJID,'DISPLAY_TITLE',srchchar,title-message,variable-name,dlflag);

**Required Arguments**

*srchchar*

the drill string for the down variable (V) or the across variable (VA).

**title-message**

the name of the title message, where the name can be CL_DOWN (for down) or IN_ACROSS (for across).

**Type**

Character
variable-name
the down or across variable.

Type Character

dflag
a flag that indicates whether to download the table to a spreadsheet, where
0=output HTML tags with data values and 1=output data values with
spreadsheet delimiters. This parameter is optional.

Type Numeric

Example
dflag=0;
downvar='Geographic';
call send(webid,'_DISPLAY_TITLE_','V','CL_DOWN',downvar,dflag);

The following output is produced:

<TABLE><TR><TD><STRONG>Down: Country=CANADA</STRONG><BR></TD></TR></TABLE>

_DISPLAY_TWOWAY_ Method
Calls the methods to display the two-way report.

Syntax
CALL SEND(OBJID,'_DISPLAY_TWOWAY_','dflag');

Required Argument
dflag
a flag that indicates whether to download the table to a spreadsheet where
0=output HTML tags with data values and 1=output data values with
spreadsheet delimiters.

Type Numeric

Details
This method
- checks for the required variables for a two-way report.
- calls the metabase _GET_HIERARCHY_ method to get a list of hierarchies.
- calls the _BUILD_APPLICATION_LIST_ method.
- calls the _CHECK_HIER_MEMBER_ method.
- calls the _SET_DRILL_LEVELS_ method to drill to the current level.
- calls the EMDDB_M class _SET_APPLICATION_ method.
- calls the _BUILD_ARGS_STRING_ method.
- calls the _BUILD_ARGS2_STRING_ method.
- calls the metabase _GET_VARIABLES_ method to get a list of analysis variables.
- calls the _EXPAND_VALUE_ data model method for all expanded variables.
- calls the EMDDB_M class _GET_CLASS_COMBINATIONS_ method to get the row list.
- calls the EMDDB_M class _GET_CLASS_COMBINATIONS_ method to get the column list.
- calls the _OUTPUT_DOWN_LIST_, _OUTPUT_ACROSS_LIST_, and _OUTPUT_VIEWRPT_BUTTON_ methods to place down and across selection lists and the View Report button above the report. This method also outputs the HTML tags to format these elements on the page.
- calls the _OPEN_TABLE_ method.
- calls the _OPEN_TWOWAY_ method.
- calls the _DISPLAY_ACROSS_CELLS_ method.
- calls the _OUTPUT_EMPTY_CELL_ method.
- calls the _DISPLAY_ANALYSIS_VARS_ method.
- calls the _DISPLAY_DOWNVAR_CELL_ method.
- calls the _DISPLAY_STATISTIC_VARS_ method.
- calls the _DISPLAY_VALUES_ method.

_DISPLAY_TWOWAY_BLOCK_ Method

Submits the SAS/GRAPH PROC GCHART statements to produce the two-way block chart.

Syntax

CALL SEND(OBJID,'_DISPLAY_TWOWAY_BLOCK_ ',statistic,analysis-variable, down-variable,across-variable,data-set-name,gif-device,subset-list);

Required Arguments

**statistic**
the statistic to graph.

  Type  Character

**analysis-variable**
the analysis variable to graph.

  Type  Character
**down-variable**
the down variable to graph.
Type Character

**across-variable**
the across variable to graph.
Type Character

**data-set-name**
the data set name from the _WRITE_ method.
Type Character

**gif-device**
the device driver name.
Type Character

**subset-list**
the initial subset list. This parameter is optional.
Type Numeric

---

**_DISPLAY_TWOWAY_HBAR_ Method**
Submits the SAS/GRAPH PROC GCHART statements to produce the two-way horizontal bar chart.

**Syntax**

CALL SEND(OBJID,'_DISPLAY_TWOWAY_HBAR_','statistic,analysis-variable,down-variable,across-variable,data-set-name,gif-device,subset-list);

**Required Arguments**

**statistic**
the statistic to graph.
Type Character

**analysis-variable**
the analysis variable to graph.
Type Character

**down-variable**
the down variable to graph.
Type Character

**across-variable**
the across variable to graph.
_DISPLAY_TWOWAY_VBAR_ Method

Submits the SAS/GRAPH PROC GCHART statements to produce the two-way vertical bar chart.

Syntax

CALL SEND(OBJID, '_DISPLAY_TWOWAY_VBAR_', statistic, analysis-variable, down-variable, across-variable, data-set-name, gif-device, subset-list);

Required Arguments

statistic
the statistic to graph.

Type Character

analysis-variable
the analysis variable to graph.

Type Character

down-variable
the down variable to graph.

Type Character

across-variable
the across variable to graph.

Type Character

data-set-name
the data set name from the _WRITE_ method.

Type Character

data-set-name
the data set name from the _WRITE_ method.

Type Character

gif-device
the device driver name.

Type Character

subset-list
the subset list for the initial graph. This parameter is optional.

Type Numeric

gif-device
the device driver name.

Type: Character

subset-list
the subset list for the initial graph. This parameter is optional.

Type: Numeric

_DISPLAY_VALUES_ Method
Outputs the numerical values to the report table.

Syntax
CALL SEND(OBJID,'_DISPLAY_VALUES_','row-list','column-list','actions-list','metabase-id','view-report-flag','url','argument-string','argument-string-2','initial-url','analysis-variable','statistic-variable','across-variable','background-type','background-value','title','webeis-class','dlflag');

Required Arguments

row-list
the row list from EMDDB_M.

Type: Numeric

column-list
the column list from EMDDB_M.

Type: Numeric

actions-list
the actions sublist that determines drill-down.

Type: Numeric

metabase-id
the metabase ID number.

Type: Numeric

view-report-flag
the View Report button flag.

Type: Numeric
_url
the URL of the next query.
Type Character

_argument-string
the argument string for the next query.
Type Character

_argument-string-2
the argument string for the next query.
Type Character

initial-url
the URL of the initial HTML page.
Type Character

analysis-variable
the analysis variable to graph.
Type Character

statistic-variable
the statistic to graph.
Type Character

across-variable
the analysis variable to graph.
Type Numeric

background-type
the background type (IMAGE or COLOR). This value is optional.
Type Character

background-value
the background value. This value is optional.
Type Character

title
the HTML page title.
Type Character

webeis-class
the WEBEIS class name.
Type Character

dlflag
a flag that indicates whether to download the table to a spreadsheet, where
0=output HTML tags with data values and 1=output data values with
spreadsheet delimiters.
Details

This method:
- calls the _GET_DATA_ATTR_ method of the METABASE class to get the base table name for reach-through
- calls the _GET_EXPANDABLE_CLASS_ method of the data model to get the expand variable
- calls the EMDDB_M _SET_ACTIVE_VALUE_ method
- calls the EMDDB_M _SET_ACTION_STATUS_ method to validate drill-down
- outputs the class value for the current row
- outputs an <A> tag if drill-down is valid
- outputs the expand link if the expand is valid
- outputs the collapse link if the collapse is valid
- calls the EMDDB_M _GET_VALUES_ method to get the numerical value of the current statistic/analysis pair
- calls the _GET_ANALYSIS_VAR_NAME_ method
- calls the metabase _GET_VAR_ATTR_ method to get the variable attributes
- calls the _GET_RANGE_COLOR_ method if a range is applied
- calls the EMDDB_M _GET_CLASS_FORMAT_ method
- outputs the numerical value to a table cell
- calls the _OUTPUT_REACHTHRU_LINK_ method if the reach-through to detail is valid
- outputs the closing HTML table tag.

Example

```plaintext
rowlist=makelist();
call send(emddbmid_,'_GET_CLASS_COMBINATIONS_','ROW',rowlist);
collist=makelist();
call send(emddbmid_,'_GET_CLASS_COMBINATIONS_','COL',collist);
actionsl=makelist();
rc=insertc(actionsl,'',-1,'CL_DRILL');
mbid=instance(loadclass('SASHELP.MB.METABASE'));
vrflag=1;
_url='/cgi-bin/broker?_PROGRAM=sashelp.webeis.mddbrpts.scl&_SERVICE=default &_DEBUG=0&RPTTYPE=2&GRTYPE=BLOCK';
_args='&MDDB=SASHELP.PRDMDDB&METABASE=SASHELP.MBEIS&DOWN=Geographic&ACROSS =Product+Line&A=ACTUAL,'
_args2='&S=SUM';
grphvar='';
grphstat='';
grphacr='PRODTYPE';
bgtype='color';
bg='yellow';
title='';
```

82 Chapter 4 / Making Advanced Customizations to the MDDB Report Viewer
The following output is produced:

```
<table>
<thead>
<tr>
<th></th>
<th>Country: CANADA</th>
<th>GERMANY</th>
<th>U.S.A.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Line</td>
<td>$97,864</td>
<td>$101,194</td>
<td>$191,194</td>
</tr>
<tr>
<td>Product Line</td>
<td>$149,126</td>
<td>$144,804</td>
<td>$194,899</td>
</tr>
<tr>
<td>Product Line</td>
<td>$246,990</td>
<td>$245,998</td>
<td>$214,794</td>
</tr>
</tbody>
</table>
```

_DRILL_TO_LEVEL_ Method

Sets the drill-down values.

Details

This method has been replaced by the _SET_DRILL_LEVELS_ method. See the _SET_DRILL_LEVELS_ on page 170 method description for more information.

_GET_ANALYSIS_VAR_NAME_ Method

Returns the name of the analysis variable that is identified by the label.

Syntax

CALL SEND(OBJID,'_GET_ANALYSIS_VAR_NAME_','label,variable-list,name');

Required Arguments

*label*
the long label for an analysis variable.
variable-list
  the list of analysis variables.

name
  the analysis variable name.

Syntax

CALL SEND(OBJID,'_GET_ANALYSIS_VARS_','metabase-id');

Required Argument

metabase-id
  the metabase ID number.

Details

This method

- calls the Metabase _GET_VARIABLES_ method
- builds the list of analysis variable labels that is identified by the ANALLBLS_instance variable.

Example

The following output is produced:

anallbls_=( 'Predicted Sales'
             'Actual Sales'
       )[563]

_GET_AVAILABLE_STATS_ Method

Gets the available statistics from the metabase.
Syntax
CALL SEND(OBJID,'_GETAVAILABLE_STATS_','metabase-id');

Required Argument

metabase-id
the metabase ID number.
Type Numeric

_GET_DATA_MODEL_NAME_ Method
Returns the data model name from the DMODEL_ instance variable.

Syntax
CALL SEND(OBJID,'_GET_DATA_MODEL_NAME_','model-name');

Required Argument

model-name
the name of the data model to use.
Type Character

_GET_DOWNVAR_LIST_ Method
Builds the down variable list and the dimensions label list.

Syntax
CALL SEND(OBJID,'_GET_DOWNVAR_LIST_','metabase-id');

Required Argument

metabase-id
the metabase ID number.
Type Numeric

Details
This method
- calls the metabase _GET_HIERARCHY_ method
calls the metabase \_GET\_VARIABLES\_ method
builds the down variable list and the dimensions label list.

\_GET\_EMDDBMID\_ Method
Returns the ID of the data model from the EMDDBMID\_ instance variable.

Syntax
CALL SEND(OBJID,'\_GET\_EMDDBMID\_','id');

Required Argument
\textit{id} 
the ID of the data model.

Type
\textit{Numeric}

\_GET\_GRAPH\_VALUES\_ Method
Gets the numeric values for the 3-D clickable graph.

Syntax
CALL SEND(OBJID,'\_GET\_GRAPH\_VALUES\_');

Details
The values are stored in the GRPHVALS\_ instance variable. Thus, the graph can be displayed with or without the report. This method:

- calls \_BUILD\_APPLICATION\_LIST\_ to build the application list
- calls \_SET\_DRILL\_LEVELS\_ to set the drill-down subsets
- calls \_SET\_APPLICATION\_ of the data model to get the initial data table
- calls \_SET\_ACTIVE\_VALUE\_ and \_EXPAND\_VALUE\_ of the data model for each of the expanded variables (if necessary)
- calls \_GET\_CLASS\_COMBINATIONS\_ of the data model to get the row class values
- calls \_GET\_CLASS\_COMBINATIONS\_ of the data model to get the column class values
- calls \_GET\_VALUES\_ of the data model for each crossing from the row and column lists
- calls \_GET\_CLASS\_FORMAT\_ for the analysis variable to get its format
adds the class values, the numerical data, and the format to the GRPHVALS_list.

Example

The GRPHVALS_instance variable contains the following:

```plaintext
\[
\begin{align*}
&\{ \text{COUNTRY='CANADA' } \\
&\text{_ANLSYS_='Actual Sales' } \\
&\text{_STATS_='Sum' } \\
&\text{PRODTYPE='FURNITURE' } \\
&'97864' \\
&'DOLLAR12.' \}\[1073] \\
&\{ \text{COUNTRY='CANADA' } \\
&\text{_ANLSYS_='Actual Sales' } \\
&\text{_STATS_='Sum' } \\
&\text{PRODTYPE='OFFICE' } \\
&'149126' \\
&'DOLLAR12.' \}\[227] \\
&\{ \text{COUNTRY='CANADA' } \\
&\text{_ANLSYS_='Actual Sales' } \\
&\text{_STATS_='Sum' } \\
&\text{PRODTYPE='TOTAL' } \\
&'246990' \\
&'DOLLAR12.' \}\[1411] \\
&\{ \text{COUNTRY='GERMANY' } \\
&\text{_ANLSYS_='Actual Sales' } \\
&\text{_STATS_='Sum' } \\
&\text{PRODTYPE='FURNITURE' } \\
&'101194' \\
&'DOLLAR12.' \}\[1631] \\
&\{ \text{COUNTRY='GERMANY' } \\
&\text{_ANLSYS_='Actual Sales' } \\
&\text{_STATS_='Sum' } \\
&\text{PRODTYPE='OFFICE' } \\
&'144804' \\
&'DOLLAR12.' \}\[1711] \\
&\{ \text{COUNTRY='GERMANY' } \\
&\text{_ANLSYS_='Actual Sales' } \\
&\text{_STATS_='Sum' } \\
&\text{PRODTYPE='TOTAL' } \\
&'245998' \\
&'DOLLAR12.' \}\[1715] \\
&\{ \text{COUNTRY='U.S.A.' } \\
&\text{_ANLSYS_='Actual Sales' } \\
&\text{_STATS_='Sum' } \\
&\text{PRODTYPE='FURNITURE' } \\
&'91567' \\
&'DOLLAR12.' \}\[1719]
\end{align*}
\]
```
_GET_MDDB_NAME_ Method

Returns the MDDB name from the MDDB_ instance variable.

Syntax

CALL SEND(OBJID,'_GET_MDDB_NAME_','mddb');

Required Argument

mddb

the MDDB name.

Type Character
_GET_MESSAGE_ID_ Method
Returns the ID of the message class from the DMODEL_ instance variable.

Syntax
CALL SEND(OBJID,'_GET_MESSAGE_ID_','message-id');

Required Argument
message-id
the ID of the message object.

Type Numeric

_GET_METABASE_NAME_ Method
Returns the metabase name from the METABASE_ instance variable.

Syntax
CALL SEND(OBJID,'_GET_METABASE_NAME_','metabase');

Required Argument
metabase
the metabase name.

Type Character

_GET_OUTPUT_FILE_ID_ Method
Returns the output file ID from the HTMLFILE_ instance variable.

Syntax
CALL SEND(OBJID,'_GET_OUTPUT_FILE_ID_','file-id');

Required Argument
file-id
the ID of the output file.
_GET_RANGE_COLOR_ Method

Returns the display color that is defined in the RANGE entry for a numeric value.

Syntax
CALL SEND(OBJID,'_GET_RANGE_COLOR_',color,range-list,numerical-value);

Required Arguments

color
the display color.
Type Character

range-list
the RANGE list.
Type Character

numerical-value
the numerical value to search for.
Type Numeric

(GET_STATDESC_ Method

Returns the ID of the statistics description list from the STATDESC_ instance variable.

Syntax
CALL SEND(OBJID,'_GET_STATDESC_',statdesc);

Required Argument

statdesc
the ID of the list that contains statistics descriptions.
Type Numeric
_GET_SUBSET_FLAG_ Method

Returns the value of the SUBSET_FLAG_ instance variable.

Syntax

CALL SEND(OBJID,'_GET_SUBSET_FLAG_','flag-value');

Required Argument

flag-value

the value of the subset flag.

Type Character

_GET_USEHOLAP_ Method

Returns the value of the HOLAP flag from the USEHOLAP_ instance variable.

Syntax

CALL SEND(OBJID,'_GET_USEHOLAP_','useholap');

Required Argument

id

the ID of the data model.

Type Numeric

_OPEN_DYNAMIC_FILE_ Method

Opens the _WEBOUT file for dynamic writing.

Syntax

CALL SEND(OBJID,'_OPEN_DYNAMIC_FILE_');
**_OPEN_FORM_ Method**

Outputs the `<FORM>` tag for the dynamic HTML pages.

**Syntax**

```plaintext
CALL SEND(OBJID,'_OPEN_FORM_','url,form-name,form-target');
```

**Required Arguments**

- **url**
  - the URL of the next query.
  - Type: Character

- **form-name**
  - the name of the form. This parameter is optional.
  - Type: Character

- **form-target**
  - the target window name. This parameter is optional.
  - Type: Character

**Details**

For further explanation of the `<FORM>` tag, refer to your favorite HTML reference documentation.

**Example**

```plaintext
CALL SEND(WEBID,'_OPEN_FORM_','/SCRIPTS/BROKER','MYFORM','MENUFORM');
```

The following output is produced:

```plaintext
<FORM ACTION="/SCRIPTS/BROKER" NAME="MYFORM" TARGET="MENUFORM">
```

---

**_OPEN_ONEWAY_ Method**

Opens the one-way report table.

**Syntax**

```plaintext
CALL SEND(OBJID,'_OPEN_ONEWAY_','dlflag');
```
Required Argument

**dlflag**
a flag that indicates whether to download the table to a spreadsheet.

Type  **Numeric**

Details

This method

- outputs the `<TABLE>` tag for the report
- outputs the empty cell in the upper left corner of the report.

Example

The following output is produced:

```html
<TABLE CLASS="MAINTAB" BORDER=1>
  <TR> <TH COLSPAN=2 CLASS="COLLAB" > </TH>
```

_OPEN_STATIC_FILE_ Method

Opens a file in which static HTML is written.

Syntax

CALL SEND(OBJID,'_OPEN_STATIC_FILE_','index-file,message-destination,rc');

Required Arguments

**index-file**
the fileref of the file to open.

Type  **Character**

**message-destination**
the destination for error messages. Valid values are LOG or DIALOG.

Type  **Character**

**rc**
the return code for errors (1=error).

Type  **Numeric**

_OPEN_TABLE_ Method

Outputs the `<TABLE>` tag for the dynamic HTML pages.
Syntax

CALL SEND(OBJID,'_OPEN_TABLE_','border-value,table-width, border-color-dark, border-color-light,background-color, cell-padding, cell-spacing css-class');

Required Arguments

**border-value**
- an optional parameter that specifies the table border thickness.
  
  Type Character

**table-width**
- an optional parameter that specifies the width of the table cells (as a percentage of the document width).
  
  Type Character

**border-color-dark**
- an optional parameter that specifies a table cell border color attribute.
  
  Type Character

**border-color-light**
- an optional parameter that specifies a table cell border color attribute.
  
  Type Character

**background-color**
- an optional parameter that specifies the background color of the table.
  
  Type Character

**cell-padding**
- an optional parameter that specifies the spacing inside the table cells.
  
  Type Character

**cell-spacing**
- an optional parameter that specifies the spacing between the table cells.
  
  Type Character

**css-class**
- an optional parameter that specifies the label for a cascading style sheet tag.
  
  Type Character

Details

For more information about the `<TABLE>` tag, refer to your favorite HTML reference documentation.
Example

CALL SEND (webid, '_OPEN_TABLE_', '3', '50', 'RED', 'YELLOW', 'GRAY', '2', 'mytable');

The following output is produced:

```html
<TABLE BORDER=3 WIDTH=50% BORDERCOLORDARK=RED BORDERCOLORLIGHT=YELLOW BGCOLOR=GRAY
     CELLPADDING=2 CELLPADDING=2 CLASS="mytable">
```

__OPEN_TWOWAY__ Method

Opens the two-way report table.

Syntax

```plaintext
CALL SEND(OBJID,'_OPEN_TWOWAY_','column-list',view-report-flag,
          _url,_argument-string,
          _argument-string-2,_argument-string-3,_initial-url,url,service,
          statistic-variable,across-variable,background-type,background-value,webeis-
          class,dlflag);
```

Required Arguments

- **column-list**
  - the column list from the __EMDBB_M class.
  - Type: Numeric

- **view-report-flag**
  - the View Report button flag.
  - Type: Numeric

- **url**
  - the Application Broker component of the URL.
  - Type: Character

- **_argument-string**
  - the argument string for the next query.
  - Type: Character

- **_argument-string-2**
  - the argument string for the next query.
  - Type: Character

- **_argument-string-3**
  - the argument string for the next query.
  - Type: Character

- **_initial-url**
  - the URL of the initial HTML page.
url
the URL for the next query.

Type Character

service
the Application Broker service being used.

Type Character

analysis-variable
the analysis variable to graph.

Type Character

statistic-variable
the statistic to graph.

Type Character

analysis-variable
the analysis variable to graph.

Type Numeric

background-type
the background type (IMAGE or COLOR). This parameter is optional.

Type Character

background-value
the background value. This parameter is optional.

Type Character

webeis-class
the WEBEIS class name.

Type Character

dlflag
a flag that indicates whether to download the table to a spreadsheet, where
0=output HTML tags with data values and 1=output data values with
spreadsheet delimiters.

Type Numeric

Details
This method
- outputs the <TABLE> tag
- calls the EMDDBB_M class _GET_CLASS_LABEL_ method to get the label of the
  across variable
- outputs the across variable label cell
_outputs the arrow <IMAGE> tag if drill-down has occurred.

___OPEN_WEBOUT_FOR_SPDSHT___ Method
Opens the _WEBOUT file in output mode for the spreadsheet.

Syntax
CALL SEND(OBJID,'_OPEN_WEBOUT_FOR_SPDSHT_');

___OUTPUT_ACROSS_LIST___ Method
Outputs a label and HTML tags for a selection list.

Syntax
CALL SEND(OBJID,'_OUTPUT_ACROSS_LIST_','across-variable');

Required Argument

across-variable
the previously selected across variable. This parameter is optional.

Type Character

Details
This method outputs
- the across label for the selection list
- a <SELECT> tag for the variable list
- an <OPTION> tag for each available variable
- the closing </SELECT> tag.

Example
The following output is produced:

Across:<BR>
<SELECT NAME="ac" SIZE=3 MULTIPLE onChange="change(document.mF.ac)">
<option VALUE=" ">
<option SELECTED VALUE=Product+Line>Product Line (hier)
<option VALUE=Geographic>Geographic (hier)
<option VALUE=Time>Time (hier)
<option VALUE=COUNTRY>Country
<option VALUE=COUNTY>County
_OUTPUT_ADDTL_CLSVAL_PARMS_ Method

Adds additional URL parameters to the JavaScript function.

Syntax
CALL SEND(OBJID,"_OUTPUT_ADDTL_CLSVAL_PARMS_");

Details
This stub method is called from the _OUTPUT_CLASSVAL_URL_FN_ method.

_OUTPUT_ADDTL_RT_PARMS_ Method

Adds additional URL parameters to the reach-through links.

Syntax
CALL SEND(OBJID,'_OUTPUT_ADDTL_RT_PARMS_');

Details
This stub method is called from the _OUTPUT_REACHTHRU_URL_FN_ method.

_OUTPUT_ADDTOFAV_FUNCTION_ Method

Outputs the addtofav JavaScript function on the Toolbar page.

Syntax
CALL SEND(OBJID,'_OUTPUT_ADDTOFAV_FUNCTION_');
Details

When a user clicks the **Bookmark** button, the `addtofav` function saves the URL in the browser's bookmark list.

Example

The following output is produced:

```javascript
function addtofav(varName){
    LinkName=window.document.title;
    with (window.parent.table_window) {
        linkUrl=eval(varName);
    }
    window.external.AddFavorite(linkUrl,LinkName);
}
```

_OUTPUT_ALL_URL_ITEMS_ Method

Outputs the parameters for the `getUrl` JavaScript function that builds the URL for the report request.

Syntax

```
CALL SEND(OBJID,'_OUTPUT_ALL_URL_ITEMS_','service-name,next-program');
```

Required Arguments

- **service-name**
  - the Application Broker service value.
  - Type: Character

- **next-program**
  - the next SCL program to execute.
  - Type: Character

_OUTPUT_ANAL_LIST_ Method

Outputs a label and HTML tags for a selection list.

Syntax

```
CALL SEND(OBJID,'_OUTPUT_ANAL_LIST_');
```
Details

This method outputs
- the analysis label for the selection list
- a <SELECT> tag for the variable list
- an <OPTION> tag for each available variable
- the closing </SELECT> tag.

Example

The following output is produced:

```html
<TD CLASS="label"> Analysis:<DIV CLASS="analysis">
<SELECT NAME="A" MULTIPLE SIZE=3>
  <OPTION SELECTED VALUE=ACTUAL>Actual Sales
  <OPTION VALUE=PREDICT>Predicted Sales
</SELECT>
</DIV>
</TD>
</TR>
```

_OUTPUT_ANAL_SELECT_ Method

Outputs the <SELECT> tag and OPTIONS for the Analysis variable list box.

Syntax

CALL SEND(OBJID,'_OUTPUT_ANAL_SELECT_','table-flag,selected-variable');

Required Arguments

- **table-flag**
  - a flag that indicates whether the list is in a table, where 1=the output is in the table and 0=the output is not in the table.
  - Type: Character

- **selected-variable**
  - the analysis variable to mark SELECTED.
  - Type: Character

Example

The following output is produced:

```html
<DIV CLASS="analysis">
<SELECT NAME="A" MULTIPLE SIZE=3>
  <OPTION SELECTED VALUE=ACTUAL>Actual Sales
  <OPTION SELECTED VALUE=DIFF>Sales Lag
</SELECT>
</DIV>
```
_OUTPUT_ARROW_FUNCTIONS_ Method

Outputs the moveall and movesel JavaScript functions on the Dimensions page.

Syntax

CALL SEND(OBJID,'_OUTPUT_ARROW_FUNCTIONS_');

Details

The moveall and movesel functions update the available and selected statistics list boxes as the user makes statistic selections for the report display.

Example

The following output is produced:

```javascript
function moveall(fromlistbox,tolistbox) {
    pos=0;
    if (fromlistbox.options.length!=0) {
        pos=tolistbox.options.length;
        for (i=0; ifromlistbox.options.length; i++) {
            if (fromlistbox.options[i].value!="" && fromlistbox.options[i].value!="MIXED") {
                tolistbox.options[pos]=new Option(statslabellist[fromlistbox.options[i].value],
                fromlistbox.options[i].value);
                pos++;
            }
        }
    }
    fromlistbox.options.length=0;
    stats(document.mf.sa,document.mf.s);
}

function movesel(fromlistbox,tolistbox) {
    pos=0; index=0; newlength=0;
    if (fromlistbox.options.length!=0) {
        pos = tolistbox.options.length;
        var listofstats = new Array();
        j = 0;
        for (i=0; i < fromlistbox.options.length; i++) {
            if (fromlistbox.options[i].selected==false && fromlistbox.options[i].value!="MIXED"
            && fromlistbox.options[i].text!="") {
                listofstats[j]=fromlistbox.options[i].value;
                j++;
            }
        }
    }
```
for (j=0; j < fromlistbox.length; j++) {
    if (fromlistbox.options[j].selected && fromlistbox.options[j].text != "" &&
        fromlistbox.options[j].value != "MIXED") {
        tolistbox.options[pos] = new Option(statslabellist[fromlistbox.options[j].value],
                                            fromlistbox.options[j].value);
        pos++;
    }
}
remstatanal(fromlistbox);
if (num > 1) {
    j=0;
    fromlistbox.options[j] = new Option(statslabellist["MIXED"], "MIXED");
} else
    j=-1;
for (i=0; i < listofstats.length; i++) {
    j++;
    if (j==listofstats.length)
        break;
    else
        fromlistbox.options[j] = new Option(statslabellist[listofstats[i]],
                                            listofstats[i]);
}
stats(document.mf.sa, document.mf.s);

_OUTPUT_BAR SHAPE LIST_ Method

Outputs the graph bar shape option on the Options page.

Syntax

CALL SEND(OBJID,'_OUTPUT_BAR SHAPE LIST_','bar-shape,view-report-flag');

Required Arguments

**bar-shape**

the currently selected graph bar shape.

Type  Character

**view-report-flag**

the View Report flag.

Type  Numeric
Example

barshape='HEXAGON';
vrflag=1;
call send(webid,'_OUTPUT_BAR_SHAPE_LIST_','barshape',vrflag);

The following output is produced:

<TD CLASS="label">Bar Shape:
<Select NAME="BS" CLASS="select">
<Option VALUE=Block>Block
<Option VALUE=Cylinder>Cylinder
<Option SELECTED VALUE=Hexagon>Hexagon
<Option VALUE=Prism>Prism
<Option VALUE=Star>Star

_OUTPUT_BOOKMARK_BUTTON_ Method

Outputs the Bookmark button on the toolbar when Access Control is enabled.

Syntax

CALL SEND(OBJID,'_OUTPUT_BOOKMARK_BUTTON_');

_OUTPUT_BOOKMARK_URL_ Method

Outputs the bookmarkURL JavaScript string on the Report page for the Bookmark button URL.

Syntax

CALL SEND(OBJID,'_OUTPUT_BOOKMARK_URL_','vrflag',url,service-name,analysis-variable,statistic,down-variable,graph-type,background-type,background-value,title,webeis-class);

Required Arguments

vrflag
the View Report button flag.
Type Numeric

url
the Application Broker component of the URL.
Type Character

service-name
the Application Broker service value.
<table>
<thead>
<tr>
<th>Type</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>analysis-variable</strong></td>
<td>the analysis variable to graph.</td>
</tr>
<tr>
<td><strong>statistic</strong></td>
<td>the statistic to graph.</td>
</tr>
<tr>
<td><strong>down-variable</strong></td>
<td>the down variable to graph.</td>
</tr>
<tr>
<td><strong>graph-type</strong></td>
<td>the graph type (BLOCK, HBAR, PIE, PLOT, or VBAR).</td>
</tr>
<tr>
<td><strong>background-type</strong></td>
<td>the background type (IMAGE or COLOR). This value is optional.</td>
</tr>
<tr>
<td><strong>background-value</strong></td>
<td>the background value. This value is optional.</td>
</tr>
<tr>
<td><strong>title</strong></td>
<td>the HTML page title.</td>
</tr>
<tr>
<td><strong>webeis-class</strong></td>
<td>the WEBEIS class name.</td>
</tr>
</tbody>
</table>

**Example**

```plaintext
vrflag=1;
url='/cgi-bin/broker';
service='default';
grphvar='ACTUAL';
grphstat='SUM';
grphdown='COUNTRY';
grptype='VBAR';
bgtype='COLOR';
bg='yellow';
title='1995 Sales Report';
webcls='SASHELP.WEBEIS.WEBEIS';
call send(_self_,'_OUTPUT_BOOKMARK_URL_',vrflag,url,service,grphvar,grphstat,
grphdown,grptype,bgtype,bg,title,webcls);
```
The following output is produced:

```
bookmarkURL = "http://mywebserver/cgi-bin/broker/.csv?_PROGRAM=SASHELP.WEBEIS.SHOWRPT.SCL &_SERVICE=default&_DEBUG=0&MDDB=SASHELP.PRDMDDB&METABASE=SASHELP&D=COUNTRY &AC=YEAR&A=ACTUAL&A1S1=SUM&BGTYPE=COLOR&BG=YELLOW&GRT=VBAR&DC=1&ACB=1 &ST=1&GL=1&GSC=1&SSL=1&SH=3&SW=15&GH=450&GW=600&DP=1"
```

**_OUTPUT_CLASSVAL_URL_FN_ Method**

Outputs the CLSVAL JavaScript function on the Report page.

**Syntax**

CALL SEND(OBJID,'_OUTPUT_CLASSVAL_URL_FN_', service-name, analysis-variable, statistic, across-variable, by-type, webeis-class, by-value, URL, title, vrflag);

**Required Arguments**

- **service-name**
  - the Application Broker service value.
  
  Type: Character

- **analysis-variable**
  - the analysis variable to graph.

  Type: Character

- **statistic**
  - the statistic to graph.

  Type: Character

- **across-variable**
  - the across variable to graph.

  Type: Character

- **background-type**
  - the background type (IMAGE or COLOR). This value is optional.

  Type: Character

- **webeis-class**
  - the WEBEIS class name.

  Type: Character

- **background-value**
  - the background value. This value is optional.

  Type: Character
title

the HTML page title.

Type Character

title

the Application Broker component of the URL.

Type Character

vrflag

the View Report bottom flag.

Type Character

Details

This is a stub method.

Example

service= 'default';
grphvar='ACTUAL';
grphstat='SUM'
across='TEAR';
bgtype= 'COLOR';
bg= 'YELLOW';
title= '1995 Sales Report';
webcls= 'SASHELP.WEBCAT.MYWEB.CLASS';
url='/cgi-bin/broker';
vrflag=1;
call send(webid,'_OUTPUT_CLASSVAL_URL_FN_',service,grphvar, grphstat,across,bytype,
webcls,by,url,title,vrflag)'

The following output is produced:

</TD></TR>
</FORM>
</TD></TR>
<TD><HR><A HREF="http://www.test.com/mddbpage.html">Select New File</A></TD>

_method

_OUTPUT_CLICKABLE_GRAPH_ Method

Outputs the <APPLET> tag for the 3-D clickable graph.

Syntax

CALL SEND(OBJID,'_OUTPUT_CLICKABLE_GRAPH_',url, service-name,graph-type,
analysis-variable, statistic,down-variable, across-variable, webeis-class,
background-type,
background-value, bar-shape);

Required Arguments

url
the Application Broker component of the URL.
Type Character

service-name
the Application Broker service value.
Type Character

graph-type
the graph type (BLOCK, HBAR, PIE, PLOT, or VBAR).
Type Character

analysis-variable
the analysis variable to graph.
Type Character

statistic
the statistic to graph.
Type Character

down-variable
the down variable to graph.
Type Character

across-variable
the across variable to graph.
Type Character

webeis-class
the WEBEIS class name.
Type Character

background-type
the background type (IMAGE or COLOR). This value is optional.
Type Character

background-value
the background value. This value is optional.
Type Character

bar-shape
the graph bar shape (Block, Cylinder, Hexagon, Prism, or Star).
Type Character
Details

In addition, the method outputs the Drive Applet Javascript function that initializes this graph.

Example

```javascript
url='/cgi-bin/broker';
graphtype=' ';
service= 'default';
grphvar='ACTUAL';
grphstat='SUM';
down='COUNTRY';
across='YEAR';
bgtype= 'COLOR';
bcol= 'YELLOW';
title= '1995 Sales Report';
webcls= 'SASHELP.WEBCAT.MYWEB.CLASS';
barshape='Star';
call send(webid, barshape='Star', '_OUTPUT_CLICKABLE_GRAPH_', url, service,
    graphtype, grphvar, grphstat, down, across, webcls, bgtype,
    by, barshape);
```

The following output is produced:

```html
</TD></TR>
</FORM>
</TD></TR>
<TD><TR><TD><HR><A HREF="http://www.test.com/mddbpage.html">Select New File</A></TD>
```

_OUTPUT_CONTENT_HEADER_ Method

Outputs the "text/html" content-type header.

**Syntax**

```javascript
CALL SEND(OBJID,'_OUTPUT_CONTENT_HEADER_');
```

_OUTPUT_CSV_CONTENT_HEADER_ Method

Outputs the content-type header for the CSV form.

**Syntax**

```javascript
CALL SEND(OBJID,'_OUTPUT_CSV_CONTENT_HEADER_');
```
_OUTPUT_DEBUG_LIST_ Method
Outputs a default debug value selection list.

Syntax
CALL SEND(OBJID,'_OUTPUT_DEBUG_LIST_');

_OUTPUT_DEFLT_TITLE_OPTION_ Method
Outputs a text input field to specify a default title.

Syntax
CALL SEND(OBJID,'_OUTPUT_DEFLT_TITLE_OPTION_');

Example
The following output is produced:

```html
<TR>
<TD CLASS="label">1998 Sales Report:</TD>
<TD><INPUT NAME="DT" CLASS="SELECT" TYPE=TEXT SIZE=30 MAXLENGTH=200></TD>
</TR>
```

_OUTPUT_DIMBTN_URL_FN_ Method
Outputs the dimbtnurl JavaScript function in the Dimensions and Options toolbar page.

Syntax
CALL SEND(OBJID,'_OUTPUT_DIMBTN_URL_FN_','.url');

Details
The dimbtnurl function is called when the Dimensions button is pressed.

Example
The following output is produced:
function dimbtnurl() {
    with (window.parent.main.document.options) {
        var limit = elements.length;
        href = "~/cgi-bin/broker?_PROGRAM=SASHELP.WEBEIS.LAYOUT.SCL";
        for (i=0; i<limit; i++) {
            if (elements[i].value != ")") {
                if (elements[i].name == "_PROGRAM" || elements[i].name == "VIEW")
                    continue;
                var thisvar=elements[i].name.toUpperCase();
                if (thisvar == "SV") {
                    var sellength = elements[i].options.length;
                    var numselected = 0;
                    for (j=0; j<sellength; j++) {
                        if (elements[i].options[j].selected) {
                            numselected++;
                            if (numselected == 1) {
                                href += ";" + elements[i].name + "=" +
                                    elements[i].options[j].value;
                            } else {
                                href += "&" + elements[i].name + eval(numselected) + "=" +
                                    elements[i].options[j].value;
                            }
                        }
                    }
                    if (numselected > 0) {
                        href += "&" + elements[i].name + "0=" + eval(numselected);
                    }
                } else {
                    href += ";" + elements[i].name + "=" + elements[i].value;
                }
            }
        }
        return href;
    }
}

_OUTPUT_DIMENSIONS_BUTTON_ Method

Outputs the <A> and <IMAGE> tags for the Dimensions button on the Layout toolbar page.

.Syntax

CALL SEND(OBJID,'_OUTPUT_DIMENSIONS_BUTTON_');

.example

The following output is produced:

<A href="../mddbapp.hlp/" onClick="this.href=dimbtnurl();" TARGET="main">
<img CLASS="imglay" SRC="http://mywebserver/images/btn_dim.gif" ALT="Dimensions" BORDER=0"></A>
_OUTPUT_DOWN_LIST_ Method

Outputs a label and HTML tags for a selection list.

**Syntax**

```call send(objid,'_output_down_list_','down-variable,url');```

**Required Arguments**

- `down-variable` the previously selected down variable. This parameter is optional.
  - **Type**: Character

- `url` the Application Broker component of the URL. This parameter is optional.
  - **Type**: Character

**Details**

This method outputs
- the Down label for the selection list
- a `<SELECT>` tag for the variable list
- an `<OPTION>` tag for each available variable
- the closing </SELECT> tag.

**Example**

The following output is produced:

```
Down: <br>
<select name="d" size=3 multiple onchange="change(document.mF.d)">
  <option selected value="Geographic">Geographic (hier)</option>
  <option value="Product+Line">Product Line (hier)</option>
  <option value="Time">Time (hier)</option>
  <option value="COUNTRY">Country</option>
  <option value="COUNTY">County</option>
  <option value="MONTH">Month</option>
  <option value="PRODTYPE">Product Type</option>
  <option value="PRODUCT">Product</option>
  <option value="QUARTER">Quarter</option>
  <option value="STATE">State/Province</option>
  <option value="YEAR">Year</option>
</select>
```
_OUTPUT_DP_TITLEOPTION_ Method

Outputs radio buttons for the Show Drillpath option in the Table list box.

Syntax

CALL SEND(OBJID,'_OUTPUT_DP_TITLE_OPTION_');

Example

The following output is produced:

```html
<TR>
  <TD CLASS='Label'>Show Drillpath</TD>
  <TD>
    <INPUT NAME="DP" CLASS="select" TYPE=RADIO VALUE="1" CHECKED>Yes
    <INPUT NAME="DP" CLASS="select" TYPE=RADIO VALUE="2" CHECKED>No
  </TD>
</TR>
```

_OUTPUT_DS2HTM_HTML_ Method

Outputs the HTML for the reach-through to the detail data page.

Syntax

CALL SEND(OBJID,'_OUTPUT_DS2HTM_HTML_','data-set-name', 'background','url', 'service-name', 'data-set-member', 'next-program-library', 'next-program-catalog', 'next-program', 'debug-value', 'where-clause');

Required Arguments

**data-set-name**
the base table data set name.

  Type  Character

**background**
the HTML background value.

  Type  Character

**url**
the Application Broker component of the URL.

  Type  Character
service-name
the Application Broker service value.
Type Character

data-set-member
the data set name (for example, PRDSALE).
Type Character

next-program-library
the library for the download to spreadsheet program.
Type Character

next-program-catalog
the catalog for the download to spreadsheet program.
Type Character

next-program
the next SCL program to execute to display additional rows of data.
Type Character

display-value
the Application Broker debug value.
Type Character

where-clause
the WHERE clause to apply to the data.
Type Character

Example

dataset='SASHELP.PRDSALE';
bgchar='BGCOLOR=YELLOW';
url='/cgi-bin/broker';
service='default';
member='PRDSALE';
pgmlib='SASHELP';
pdmcat='WEBEIS';
program='SASHELP.WEBEIS.DS2HTM.SCL';
display='0';
where='COUNTRY=CANADA';
call send(webid,'_OUTPUT_DS2HTM_HTML_',dataset,bgchar,url,service,member,pgmlib,pdmcat,
program,debug,where);

_OUTPUT_DS2HTM_ST_ Method
Outputs the DS2HTM statement to generate the detail data table.
Syntax

CALL SEND (OBJID, '_OUTPUT_DS2HTM_ST_ ', data-set-name, variable-string, start-at,
number-of-rows, total-rows);

Required Arguments

data-set-name
the base table data set name.
Type Character

variable-string
the selected variables to display, separated by spaces.
Type Character

start-at
the starting row to display.
Type Numeric

number-of-rows
the number of rows to display.
Type Numeric

total-rows
the total number of rows of detail data.
Type Numeric

Example

dataset='SASHELP.PRDSALE';
varchar='COUNTRY ACTUAL PREDICT';
startat=1;
atatime=50;
numrows=480;
call send(webid,'_OUTPUT_DS2HTM_ST_ ',dataset,varchar,startat,atatime,numrows);

_OUTPUT_DYNAMIC_HIDDEN_FLDS_ Method

Outputs the necessary hidden fields for the initial dynamic HTML page.

Syntax

CALL SEND (OBJID, '_OUTPUT_DYNAMIC_HIDDEN_FLDS_ ', metabase,
background-value,
background-type, service, debug, title, webeis-class);
Required Arguments

**metabase**
the metabase name.
Type Character

**background-value**
the background image URL or color value. This value is optional.
Type Character

**background-type**
the background type (COLOR or IMAGE). This value is optional.
Type Character

**service**
the application server service.
Type Character

**debug**
the debug level.
Type Character

**title**
the HTML page title.
Type Character

**webeis-class**
the WEBEIS class name.
Type Character

Example

```
metabase='SASHELP.MBEIS';
bgtype='color';
bg='yellow';
service='default';
ddebug=0;
title='1997+Sales+Reports';
webcls='SASHELP.WEBEIS.WEBEIS';
call send(webid,'_OUTPUT_DYNAMIC_HIDDEN_FLDS_',metabase,bgtype,bg,
service,debug,title,webcls);
```

The following output is produced:

```
<INPUT TYPE="hidden" NAME="metabase" VALUE="SASHELP.MBEIS">
<INPUT TYPE="hidden" NAME="_program" VALUE="sashelp.webeis.mdbbrpts.scl">
<INPUT TYPE="hidden" NAME="bgtype" VALUE="color">
<INPUT TYPE="hidden" NAME="bg" VALUE="yellow">
<INPUT TYPE="hidden" NAME="service" VALUE="default">
<INPUT TYPE="hidden" NAME="debug" VALUE="0">
<INPUT TYPE="hidden" NAME="title" VALUE="1997+Sales+Reports">
```
**_OUTPUT_EMPTY_CELL_ Method**

Outputs an empty cell in the HTML table.

**Syntax**

```
CALL SEND(OBJID,'_OUTPUT_EMPTY_CELL_',span-number,dlflag(css-class));
```

**Required Arguments**

- **span-number**
  - the number of columns to span.
  - *Type: Numeric*

- **dlflag**
  - a flag that indicates whether to download the table to a spreadsheet where 0 output HTML tags with data values and 1 output data values with spreadsheet delimiters.
  - *Type: Numeric*

- **css-class**
  - the class name for the cascading style sheet class tag. This parameter is optional.
  - *Type: Character*

**_OUTPUT_EMPTY_SERVICE_LIST_ Method**

Outputs an empty service list.

**Syntax**

```
CALL SEND(OBJID,'_OUTPUT_EMPTY_SERVICE_LIST_');
```

**Details**

This method outputs

- the `<SELECT>` tag
- an example `<OPTION>` tag with comments that instruct users to edit or add `<OPTION>` tags for their services.
_OUTPUT_GRAPH_DIMS_OPTION_ Method
Outputs text fields for specifying the graph's width and height.

Syntax
CALL SEND(OBJID,'_OUTPUT_GRAPH_DIMS_OPTION_');

Example
The following output is produced:

```
<TR><TD CLASS="label">Width</TD><TD><INPUT TYPE=text NAME="gw" CLASS="select" SIZE=4 MAXLENGTH=4 VALUE="600"></TD></TR>
<TR><TD CLASS="label">Height</TD><TD><INPUT TYPE=text NAME="gh" CLASS="select" SIZE=4 MAXLENGTH=4 VALUE="450"></TD></TR>
```

_OUTPUT_GRAPH_INSTR_ Method
Outputs the Change Graph Type instructions and the Apply button.

Syntax
CALL SEND(OBJID,'_OUTPUT_GRAPH_INSTR_');

Details
This method outputs
- the Change Graph Type instructions to the HTML
- the Apply button to the HTML.

_OUTPUT_GRAPH_LIST_ Method
Outputs the list of graph types.

Syntax
CALL SEND(OBJID,'_OUTPUT_GRAPH_LIST_','graph-type,vrflag');
Required Arguments

**graph-type**
the previously selected graph type.

- **Type**: Character

**vrflag**
the View Report button flag, which takes the following values:

1. **View Report** button click on previous action.
2. **No View Report** button click on previous action.

- **Type**: Numeric

Details

This method outputs
- the `<SELECT>` tag
- an `<OPTION>` tag for each graph type.

Example

The following output is produced:

```html
<TD CLASS="label">Type</TD>
<TD><SELECT NAME="grt" CLASS="select">
  <OPTION SELECTED VALUE=NONE>None
  <OPTION VALUE=VBAR>Vertical bar
  <OPTION VALUE=BLOCK>Block
  <OPTION VALUE=HBAR>Horizontal bar
  <OPTION VALUE=PIE>Pie
  <OPTION VALUE=PLOT>Plot
</SELECT></TD>
```

---

**_OUTPUT_GRAPH_LOC_OPTION_ Method**

Outputs a selection list for the Graph Location option.

**Syntax**

```plaintext
CALL SEND(OBJID,'_OUTPUT_GRAPH_LOC_OPTION_');
```

**Example**

The following output is produced:

```html
<TD CLASS="label">Location</TD>
<TD><SELECT NAME="gl" CLASS="select">
  <OPTION VALUE="1" SELECTED>Bottom
  <OPTION VALUE="2">Top
  <OPTION VALUE="3">Left
</SELECT></TD>
```
_OUTPUT_GRAPH_OPTION_ Method
Outputs an OPTION tag for the Graph Type selection list.

Syntax
CALL SEND(OBJID,'_OUTPUT_GRAPH_OPTION_','graph-type',graph-type-message,option-value);

Required Arguments

*graph-type*  
the previously selected graph type.  

  Type  Character

*graph-type-message*  
the mnemonic of the graph type message.  

  Type  Character

*option-value*  
the value for the <OPTION> tag.  

  Type  Character

.OUTPUT_GRAPH_SOURCE_OPTION_ Method
Outputs radio buttons for the Graph Source option.

Syntax
CALL SEND(OBJID,'_OUTPUT_GRAPH_SOURCE_OPTION_');

Example
The following output is produced:

```html
<TD CLASS="label">Graph Source</TD>

<INPUT NAME="GSC" CLASS="select" TYPE=RADIO VALUE="1" CHECKED>3D Clickable Graph
<INPUT NAME="GSC" CLASS="select" TYPE=RADIO VALUE="2">Standard GIF Graph
```
_OUTPUT_GRAPH_TABLE_DISP_ Method

Outputs the check boxes on the Options page for the Display Table and Display Graph options.

Syntax

CALL SEND(OBJID,'_OUTPUT_GRAPH_TABLE_DISP_');

Example

The following output is produced:

<TD CLASS="label" COLSPAN="2"><INPUT NAME="ST" TYPE=CHECKBOX VALUE="1" CHECKED>Display Table & <INPUT NAME="SG" TYPE=CHECKBOX VALUE="1">Display Graph
</TD>

_OUTPUT_HDR_ Method

Outputs the opening tags for the Report Layout page.

Syntax

CALL SEND(OBJID,'_OUTPUT_HDR_','url,background-type,background-value');

Required Arguments

url

the Application Broker component of the URL.

Type Character

background-type

the background type (COLOR or IMAGE). This parameter is optional.

Type Character

background-value

the background value. This parameter is optional.

Type Character

Example

The following output is produced:

<HTML><HEAD><TITLE>MDDB Report Viewer Layout</TITLE>
<script language="javascript">
function List(list) {
    for (key in list)
        if (list[key] != null) this[key] = list[key];
}

selected = new List;
selected2 = new List;

function change(select) {
    if ((navigator.appName == "Netscape" &&
        navigator.appVersion.indexOf("3.0") != -1) ||
        (navigator.appName == "Microsoft Internet Explorer" &&
        navigator.appVersion.indexOf("4.0") != -1)) {
        options = new Object;
        for (i = 0; i < select.options.length; i++) {
            options[select.options[i].text] = select.options[i].value;
            selected[select.options[i].text] =
                select.options[i].selected ? select.options[i].value : null;
        }
    }

    selected = new List(selected);
    select.options.length = 0;
    for (key in selected)
        select.options[select.options.length] =
            new Option(key, selected[key], false, true);
    for (key in options)
        if (selected[key] == null)
            select.options[select.options.length] =
                new Option(key, options[key]);
}

function update() {
    str = "";
    for (key in selected)
        str = str + key + ",";
    if (str.length)
        document.form.order.value = str.substring(0, str.length - 1);
}

</SCRIPT>
</HEAD>
<BODY BGCOLOR=white>
<CENTER>
<TABLE CELSPACING=1 BORDER=1>

_OUTPUT_HELP_BUTTON_ Method

Outputs the Help button on the toolbar.

Syntax

CALL SEND(OBJID,'_OUTPUT_HELP_BUTTON_');
Details

This method outputs the HTML tags for the Help button hypertext link and the Help button image.

Example

The following output is produced:

```html
<A HREF="http://support.sas.com/rnd/web/internet/mddapp/hinttips.html" TARGET="_blank">
<IMG CLASS="imghelp" SRC="/my_images/btn_hlp.gif" ALT="Help" BORDER=0></A>
```

_OUTPUT_HIDDEN_FIELDS_ Method

Outputs the HTML hidden fields on the tabular report that are necessary for processing the next user action.

Syntax

```plaintext
CALL SEND(OBJID,'_OUTPUT_HIDDEN_FIELDS_','across-variable, statistic-variable, analysis-variable, initial-url, service, background-type, background-value, title, webeis-class);
```

Required Arguments

- **across-variable**
  - the across value to graph.
  - Type: Character

- **statistic-variable**
  - the statistic to graph.
  - Type: Character

- **analysis-variable**
  - the variable to graph.
  - Type: Character

- **initial-url**
  - the URL of the initial HTML page.
  - Type: Character

- **service**
  - the Application Broker service.
  - Type: Character

- **background-type**
  - the background type (IMAGE or COLOR). This parameter is optional.
  - Type: Character
**background-value**

the background value. This parameter is optional.

**title**

the title for the HTML page. This parameter is optional.

**webesis-class**

the WEBEIS class name (for subclassing).

### Example

The following output is produced:

```html
<INPUT TYPE="hidden" NAME="_SERVICE" value="default">
<INPUT TYPE="hidden" NAME="_DEBUG" value="2">
<INPUT TYPE="hidden" NAME="MDDB" value="SASHELP.PRDMDDB">
<INPUT TYPE="hidden" NAME="METABASE" value="SASHELP.MBEIS">
<INPUT TYPE="hidden" NAME="BGTYPE" value="color">
<INPUT TYPE="hidden" NAME="BG" value="%23FFFFE7">
<INPUT TYPE="hidden" NAME="GRT" value="NONE">
<INPUT TYPE="hidden" NAME="GL" value="1">
<INPUT TYPE="hidden" NAME="GSC" value="1">
<INPUT TYPE="hidden" NAME="SSL" value="1">
<INPUT TYPE="hidden" NAME="ST" value="1">
<INPUT TYPE="hidden" NAME="SH" value="3">
<INPUT TYPE="hidden" NAME="SW" value="15">
<INPUT TYPE="hidden" NAME="GH" value="450">
<INPUT TYPE="hidden" NAME="GW" value="600">
<INPUT TYPE="hidden" NAME="DC" value="1">
<INPUT TYPE="hidden" NAME="ACB" value="1">
<INPUT TYPE="hidden" NAME="DP" value="1">
```

**_OUTPUT_HIDDEN_VARS_ Method**

Outputs the filter variables, analysis variables, and statistics as HTML hidden fields for the filter form.

**Syntax**

```
CALL SEND(OBJID,'_OUTPUT_HIDDEN_VARS_');
```
_OUTPUT_HTML_AFTER_BODY_ Method
Enables users to add HTML tags to the Report page.

Syntax
CALL SEND(OBJID,'_OUTPUT_HTML_AFTER_BODY_');

Details
This stub method is called after the <BODY> tag is output for the Report page.

_OUTPUT_HTML_BEF_CLOSE_BODY_ Method
Enables users to add HTML tags to the end of the Report page.

Syntax
CALL SEND(OBJID,'_OUTPUT_HTML_BEF_CLOSE_BODY_');

Details
This stub method is called before the </BODY> tag is output for the Report page.

_OUTPUT_HTML_FORM_HEADER_ Method
Outputs the opening tags for the static HTML page.

Syntax
CALL SEND(OBJID,'_OUTPUT_HTML_FORM_HEADER_','title,cgi,background-value,background-type');

Required Arguments

*title*
an optional title for the page.

*Type* Character

*cgi*
the Application Broker component for the <ACTION> tag.
Type  Character

**background-value**
the background image URL or color value. This parameter is optional.

Type  Character

**background-type**
the background type (COLOR or IMAGE). This parameter is optional.

Type  Character

Details
This method
- outputs the opening HTML page tags
- outputs the `<BODY>` tag with the appropriate background parameters
- outputs a title
- outputs the `<FORM>` tag.

_**OUTPUT_LAYOUT_BUTTON_** Method
Outputs the **Layout** button on the toolbar to enable users to return to the Variable Selection page.

Syntax
CALL SEND(OBJID,'_OUTPUT_LAYOUT_BUTTON_');

Details
This method outputs the HTML tags for the **Layout** button hypertext link and the **Layout** button image.

Example
The following output is produced:

```html
<A href="../mddbapp.hlp/"
   onClick="this.href=clsurl('_PROGRAM=SASHELP.WEBEIS.MDDBRPTS.SCL')" TARGET="_parent">
   <IMG CLASS="imglay" SRC="/my_images/btn_lay.gif" ALT="Layout" BORDER=0></A>
```

_**OUTPUT_LAYOUT_FRAME_** Method
Outputs the `<FRAME>` tag for the Dimensions page.
Syntax

CALL SEND(OBJID,'_OUTPUT_LAYOUT_FRAME_','url','service-name','background-type','graph-type','background-value','analysis-variable','statistic','down-variable','across-variable');

Required Arguments

- **url**
  - the Application Broker component of the URL.
  - Type: Character

- **service-name**
  - the Application Broker service value.
  - Type: Character

- **background-type**
  - the background type (IMAGE or COLOR). This parameter is optional.
  - Type: Character

- **graph-type**
  - the graph type (BLOCK, HBAR, PIE, PLOT, or VBAR).
  - Type: Character

- **background-value**
  - the background value. This parameter is optional.
  - Type: Character

- **analysis-variable**
  - the analysis variable to graph.
  - Type: Character

- **statistic**
  - the statistic to graph.
  - Type: Character

- **down-variable**
  - the down variable to graph.
  - Type: Character

- **across-variable**
  - the across variable to graph.
  - Type: Character

Example

```plaintext
url='/cgi-bin/broker';
service='default';
```
The following output is produced:

```html
<FRAME NAME="main"
   SRC="/cgi-bin/broker?_program=sashelp.webeis.layout.scl&_service=default
   &_debug=0&mrvdebug=2&mddb=SASHELP.PRDMDDB&metabase=SASHELP&DC=COUNTRY&AC=YEAR&A=ACTUAL&A1S1=SUM&GRT=VBAR&BGTYPE=COLOR&BG=YELLOW&GV=ACTUAL&GS=SUM&GD=COUNTRY&GA=YEAR&DC=1&ACB=1">
```

### _OUTPUT_LAYOUT_TOOLBAR_ Method

Outputs the **Dimensions** and **Options** buttons on the Layout toolbar page.

#### Syntax

```javascript
CALL SEND(OBJID,'_OUTPUT_LAYOUT_TOOLBAR_');
```

#### Example

The following output is produced:

```html
<TR>
   <TD>
      <A href="../mddbapp.hlp/" onClick="this.href=dimbtnurl();" TARGET="main">
         <IMG CLASS="imglay" SRC="http://mywebserver/images/btn_lay.gif" ALT="Dimensions" BORDER=0></A>
      </TD>
   <TD>
      <A href="../mddbapp.hlp/" onClick="this.href=optbtnurl();" TARGET="main">
         <IMG CLASS="imglay" SRC="http://mywebserver/images/btn_lay.gif" ALT="Options" BORDER=0></A>
      </TD>
</TR>
```

### _OUTPUT_LOGOUT_BUTTON_ Method

Outputs the **Logout** button on the toolbar when access control is enabled.
Syntax
CALL SEND(OBJID,'_OUTPUT_LOGOUT_BUTTON_');

_OUTPUT_MAIN_TOOLBAR_FRAME_ Method
Outputs the <FRAME> tag for the toolbar on the Dimensions and Options page.

Syntax
CALL SEND(OBJID,'_OUTPUT_MAIN_TOOLBAR_FRAME_','url',service-name, background-type, graph-type, background-value, analysis-variable, statistic, down-variable, across-variable);

Required Arguments

url
the Application Broker component of the URL.
Type Character

service-name
the Application Broker service value.
Type Character

background-type
the background type (IMAGE or COLOR). This parameter is optional.
Type Character

graph-type
the graph type (BLOCK, HBAR, PIE, PLOT, or VBAR).
Type Character

background-value
the background value. This parameter is optional.
Type Character

analysis-variable
the analysis variable to graph.
Type Character

statistic
the statistic to graph.
Type Character

down-variable
the down variable to graph.
across-variable

the across variable to graph.

Example

```plaintext
url='/cgi-bin/broker';
service='default';
grphvar='ACTUAL';
grphstat='SUM';
grphdown='COUNTRY'
;grphacr='YEAR';
grptype='VBAR';
bgtype='COLOR';
bg='YELLOW';
call send(_self_,'_OUTPUT_MAIN_TOOLBAR_FRAME_',url,service,bgtype,grphtype,bg,grphvar,grphstat,grphdown,grphacr);
```

The following output is produced:

```plaintext
<FRAINAME="header" SRC="/cgi-bin/broker?_program=sashelp.webeis.header.scl
    &service=default&_debug=0&mrvdebug=2&mddb=SASHELP.PRDMDDB
    &metabase=SASHELP&D=COUNTRY&AC=YEAR&A=ACTUAL
    &A1S1=SUM&GRT=VBAR&BGTYPE=COLOR&BG=YELLOW
    &GV=ACTUAL&GS=SUM&GD=COUNTRY&GA=YEAR&DC=1
    &ACB=1" SCROLLING="NO">
```

_OUTPUT_MDDDB_LIST_ Method

Outputs the list of MDDBs.

Syntax

```plaintext
CALL SEND(OBJID,'_OUTPUT_MDDDB_LIST_','mddb-list','mddb');
```

Required Arguments

- **mddb-list**
  
  the list of MDDBs.

  Type  Numeric

- **mddb**
  
  the currently selected MDDB. This parameter is optional.

  Type  Character
Details
This method outputs the `<SELECT>` and `<OPTION>` tags for selecting an MDDB.

_OUTPUT_NUMROWS_LINKS_Method
Outputs the hypertext links beneath a report that enable paging through selected rows in the report.

Syntax
CALL SEND(OBJID,'_OUTPUT_NUMROWS_LINKS_');

Example
The following output is produced:

```
p.1
    <A href="/mddbapp.hlp/" onClick="this.href=clsurl
        ('_PROGRAM=SASHELP.WEBBEIS.OPRPT.SCL&SR=26&NR=25');"
        onMouseOver="window.status='Display Rows 26-50'; return true" TARGET="_self">2</A>
    <A href="/mddbapp.hlp/" onClick="this.href=clsurl
        ('_PROGRAM=SASHELP.WEBBEIS.OPRPT.SCL&SR=51&NR=25');"
        onMouseOver="window.status='Display Rows 51-75'; return true" TARGET="_self">3</A>
    <A href="/mddbapp.hlp/" onClick="this.href=clsurl
        ('_PROGRAM=SASHELP.WEBBEIS.OPRPT.SCL&SR=76&NR=25');"
        onMouseOver="window.status='Display Rows 76-100'; return true" TARGET="_self">4</A>
```

_OUTPUT_NUMROWS_OPTION_Method
Outputs the radio buttons to select the number of rows in the report table to display.

Syntax
CALL SEND(OBJID,'_OUTPUT_NUMROWS_OPTION_');

Example
The following output is produced:

```
    <TR>
        <TD CLASS="label">Number of Rows</TD>
        <TD>
            <INPUT NAME="NR" CLASS="select" TYPE=RADIO VALUE="ALL" CHECKED>ALL
            <INPUT NAME="NR" CLASS="select" TYPE=RADIO VALUE="1">1
            <INPUT NAME="NR" CLASS="select" TYPE=RADIO VALUE="2">2
            <INPUT NAME="NR" CLASS="select" TYPE=RADIO VALUE="3">3
        </TD>
    </TR>
```
**_OUTPUT_OPTBTN_URL_FN_ Method**

Outputs the optbtnurl JavaScript function in the Dimensions and Options toolbar page.

**Syntax**

```javascript
CALL SEND(OBJID,'_OUTPUT_OPTBTN_URL_FN_',url);
```

**Details**

The optbtnurl function is called when the Options button is pressed.

**Example**

The following output is produced:

```javascript
function optbtnurl() {
  with (window.parent.main.document.mf) {
    var limit = elements.length;
    href = "http://cgi-bin/broker?_PROGRAM=SASHELP.WEBEIS.OPTIONS.SCL";
    for (i=0; i<limit; i++) {
      if (elements[i].value != "") {
        if (elements[i].name == "_PROGRAM")
          continue;
        var thisvar=elements[i].name.toUpperCase();
        if (thisvar == "D" || thisvar == "AC" || thisvar == "A") {
          var selength = elements[i].options.length;
          var numselected = 0;
          for (j=0; j<selength; j++) {
            if (elements[i].options[j].selected) {
              numselected++;
              if (numselected == 1) {
                href += "&" + elements[i].name + "=" + elements[i].options[j].value;
              }
            }
          }
        } else {
          href += "&" + elements[i].name + eval(numselected) + "=" + elements[i].options[j].value;
        }
        if (thisvar == "A") {
          var href2="";
          stats=elements[i].options[j].value+"STATS";
          statsstr="window.parent.main."+stats;
          statsarray=eval(statsstr);
          if (statsarray.length==1 && statsarray[0]!="nunique") {
            href2="&" + statsarray[0] + "NUNIQUE";
          } else if (statsarray.length==1 && statsarray[0]!="nunique") {
            href2="&" + statsarray[0] + "SUM";
          } else {
            var j=0;
            for (j=0; j<selength; j++) {
              if (elements[i].options[j].selected) {
                href2="&" + elements[i].name + "=" + elements[i].options[j].value;
              }
            }
            href2="&" + statsarray[0] + "NUNIQUE";
          }
        }
    }
  }
}
```
var anum=0;
for (k=1; k&ltstatsarray.length k++) {
  anum=j+1;
  href2+="&A" +anum +"S" +k +"=" +statsarray[k];
}
var numstats = statsarray.length-1;
if (numstats > 1) {
  href2+="&A" + anum + "S0=" + numstats;
}
href += href2;
}
if (numselected > 0) {
  href += "&" + elements[i].name + "0=" + eval(numselected);
}
else {
  href += "&" + elements[i].name + "=" + elements[i].value;
}
return href;

_OUTPUT_OPTIONS_BUTTON_ Method

Outputs the <A> and <IMAGE> tags for the Options button on the Layout toolbar page.

Syntax

CALL SEND(OBJID,'_OUTPUT_OPTIONS_BUTTON_');

Example

The following output is produced:

<A href="../mddbapp.hlp/" onClick="this.href=optbtnurl();" TARGET="main"> <IMG CLASS="imglay" SRC="http://mywebserver/images/btn_opt.gif" ALT="Options" BORDER=0></A>

_OUTPUT_OPTIONS_FORM_ Method

Outputs the HTML <FORM> tag for the Options page.
Syntax

CALL SEND(OBJID,'_OUTPUT_OPTIONS_FORM_','_url',message-id,graph-type,bar-shape);

Required Arguments

_url
the Application Broker component of the URL.
Type Character

message-id
the ID of the message system.
Type Numeric

graph-type
the graph type.
Type Character

bar-shape
the graph bar shape.
Type Character

Example

The following output is produced:

url='/cgi-bin/broker';
msgid=instance(loadclass('sashelp.fsp.astmsg.class'),1);
grptype='VBAR';
barshape='HEXAGON';
call send(webid,'_OUTPUT_OPTIONS_FORM_','_url',msgid,grphtype,barshape);

_OUTPUT_REACHTHRU_LINK_ Method

Outputs the hypertext link for the numeric data in the report to enable reach-through to the detail data.

Syntax

CALL SEND(OBJID,'_OUTPUT_REACHTHRU_LINK_','metabase-id,row-list,row-index,
column-list,column-index,cur-list);

Required Arguments

metabase-id
the ID number of the metabase.
**_OUTPUT_REACHTHRU_URL_FN_ Method**

Outputs the RTURL Javascript function that builds the reach-through to detail URLs.

**Syntax**

```verbatim
CALL SEND(OBJID,'_OUTPUT_REACHTHRU_URL_FN_,service,next-program,data-set,
background-type,background-value,url);
```

**Required Arguments**

**service**
the Application Broker service.

Type Character
**next-program**
the four-level name of the program to run to display the detail data. The default is
SASHELP.WEBEIS.DS2HTM.SCL.

**Type** Character

**data-set**
the name of the data set that contains the detail data.

**Type** Character

**background-type**
the background type (IMAGE, COLOR, or blank).

**Type** Character

**background-value**
the background value.

**Type** Character

**url**
the Application Broker component of the URL.

**Type** Character

**Example**
The following output is produced:

```javascript
function rturl(str) {
  param=new Object;
  param._PROGRAM = "SASHELP.WEBEIS.VARLIST.SCL";
  param._SERVICE = "default";
  param._DEBUG = "2";
  param.MDDB = "SASHELP.PRDMDDB";
  param.METABASE = "SASHELP.MBEIS";
  param.D = "Geographic";
  param.AC = "Product%20Line";
  param.V10="0";
  param.VA10="0";
  param.A = "ACTUAL";
  param.S = "SUM";
  param.NEXTPGM = "SASHELP.WEBEIS.DS2HTM.SCL";
  param.DATASET = "SASHELP.PRDSALE";
  param.BGTYPE = "color";
  param.BG = "%23FFFFE7";
  href = "cgi-bin/broker?";
  for (name in param) { href += name + "=" + param[name] + "&" }
  if (str) {href += str}
  return href;
}
```
_OUTPUT_REPORT_FRAME_ Method

Outputs the <FRAME> tag to create the frame in which the report is displayed.

**Syntax**

CALL SEND(OBJID,’_OUTPUT_REPORT_FRAME_’,url,service,background-type,graph-type,background-value,graph-variable, graph-statistic, graph-down,graph-across,debug);

**Required Arguments**

*url*  
the Application Broker component of the URL.  
Type Character

*service*  
the Application Broker service.  
Type Character

*background-type*  
the background type (IMAGE, COLOR, or blank).  
Type Character

*graph-type*  
the selected graph type.  
Type Character

*background-value*  
the background value.  
Type Character

*graph-variable*  
the analysis variable to graph.  
Type Character

*graph-statistic*  
the statistic to graph.  
Type Character

*graph-down*  
the down dimension variable to graph.  
Type Character
**graph-across**
the across dimension variable to graph.

Type  Character

**debug**
the Application Broker debug value.

Type  Character

Example
The following output is produced:

```xml
<FRAME NAME="table_window" SRC="/cgi-bin/broker?_program=sashelp.webeis.oprpt.scl
&_service=default&_debug=2&VIEW=View+Report&mddb=SASHELP.PRDMDDB&metabase
=SASHELP.MBEIS&D=Geographic&AC=Product%2520Line&A=ACTUAL&S=SUM&GRT=VBAR
&BGTYPE=color&BG=%23FFFFE7&DC=1&ACB=1&ST=1&GL=1&GSC=2&SSL=1&SH=3&SW=15
&GH=450&GW=600&DP=1">
```

---

**_OUTPUT_REPORT_RADIO_BTNS_ Method**

Outputs the Report Selection radio buttons.

**Syntax**

CALL SEND(OBJID,'_OUTPUT_REPORT_RADIO_BTNS_');

---

**_OUTPUT_REPORT_TYPE_SELECT_ Method**

Outputs the Report type selection list.

**Syntax**

CALL SEND(OBJID,'_OUTPUT_REPORT_TYPE_SELECT_','report-type');

**Required Argument**

`report-type`

a previously selected report type.

Type  Character
_OUTPUT_ROTATE_BUTTON_ Method

Outputs the Rotate button for the two-dimensional report.

Syntax

CALL SEND(OBJID,'_OUTPUT_ROTATE_BUTTON_','view-report-flag,url,
  service,initial-url,
  across-variable,down-variable, analysis-variable,statistic-variable,down-variable,
  graph-type,background-type,background-value,title, webeis-class,hide-flag);

Required Arguments

view-report-flag
  the View Report button flag.
  Type  Numeric

url
  the Application Broker component of the URL.
  Type  Character

service
  the Application Broker service.
  Type  Character

initial-url
  the URL of the initial HTML page.
  Type  Character

across-variable
  the across variable that is selected.
  Type  Character

down-variable
  the down variable that is selected.
  Type  Character

analysis-variable
  the analysis variable to graph.
  Type  Character

statistic-variable
  the statistic to graph.
  Type  Character
**down-variable**
the down variable to graph.

*Type* Numeric

**graph-type**
the selected graph type.

*Type* Numeric

**background-type**
the background type (IMAGE or COLOR). This parameter is optional.

*Type* Character

**background-value**
the background value. This parameter is optional.

*Type* Character

**title**
the HTML title page.

*Type* Character

**webeis-class**
the WEBEIS class name.

*Type* Character

**hide-flag**
a hidden variables flag. If hideflag = 1, variables are not output. This parameter is optional.

*Type* Character

Details
This method outputs an HTML form that contains hidden fields that are necessary to process the rotate request and output the **Rotate** submit button.

Example
The following example illustrates the use of this method:

```bash
vrflag=1;
_url='/cgi-bin/broker?PROGRAM=sashelp.webeis.mddbrpts.scl&_SERVICE=default &_DEBUG=0&RPTTYPE=2&GRTYPE=BLOCK';
service='default';
mddblink='DYNAMIC';
across='Geographic';
down='Product+Line';
avar='ACTUAL';
stat='SUM';
grphdown='';
grtype='BLOCK';
bgtype='color';
```
The following output is produced:

```html
<A href="../mddbapp.hlp/" onClick="this.href=clsurl('ROTATE=1
&_PROGRAM=SASHELP.WEBEIS.SHOWRPT.SCL')" TARGET="_parent">
  <IMG CLASS="imgrotate" SRC="/my_images/btn_rot.gif" ALT="Rotate"
BORDER=0>
</A>
```

## _OUTPUT_ROTATE_URL_ Method

Outputs the rotateURL JavaScript string on the Report page for the Rotate button URL.

### Syntax

```plaintext
CALL SEND(OBJID,'_OUTPUT_ROTATE_URL_','-vrflag,url,service,
    mddblink,across,down,avar,stat,grphdown,grtype,bgtype,
    bg,title,webcls,hideflag);
```

### Required Arguments

- **vrflag**
  - the View Report button flag.
  - **Type**: Numeric

- **url**
  - the Application Broker component of the URL.
  - **Type**: Character

- **service-name**
  - the Application Broker service value.
  - **Type**: Character

- **analysis-variable**
  - the analysis variable to graph.
  - **Type**: Character

- **statistic**
  - the statistic to graph.
  - **Type**: Character
**down-variable**
the down variable to graph.

Type Character

**graph-type**
the graph type (BLOCK, HBAR, PIE, PLOT, or VBAR).

Type Character

**background-type**
the background type (IMAGE or COLOR). This parameter is optional.

Type Character

**background-value**
the background value. This parameter is optional.

Type Character

**title**
the HTML page title.

Type Character

**webeis-class**
the WEBEIS class name.

Type Character

### Example
This example illustrates the use of the method:

```plaintext
down-variable
the down variable to graph.

Type Character

down-variable
the down variable to graph.

Type Character
```

```plaintext
graph-type
the graph type (BLOCK, HBAR, PIE, PLOT, or VBAR).

Type Character

graph-type
the graph type (BLOCK, HBAR, PIE, PLOT, or VBAR).

Type Character
```

```plaintext
background-type
the background type (IMAGE or COLOR). This parameter is optional.

Type Character

background-type
the background type (IMAGE or COLOR). This parameter is optional.

Type Character
```

```plaintext
background-value
the background value. This parameter is optional.

Type Character

background-value
the background value. This parameter is optional.

Type Character
```

```plaintext
title
the HTML page title.

Type Character

title
the HTML page title.

Type Character
```

```plaintext
webeis-class
the WEBEIS class name.

Type Character

webeis-class
the WEBEIS class name.

Type Character
```

### Example
This example illustrates the use of the method:

```plaintext
vrflag=1;
url='/cgi-bin/broker';
service='default';
grphvar='ACTUAL';
grphstat='SUM';
grphdown='COUNTRY';
grphtype='VBAR';
bgtype='COLOR';
bg='yellow';
title='1995 Sales Report';
webcls='SASHELP.WEBEIS.WEBEIS';
call send(_self_,'_OUTPUT_ROTATE_URL_',vrflag,url,service,grphvar,grphstat,
grphdown,grphtype,bgtype,bg,title,webcls);
```

The following output is produced:

```plaintext
rotateURL="http://mywebserver/cgi-bin/broker/.csv?_PROGRAM=SASHELP.WEBEIS.OPRPT.SCL
&ROTATE=1&_SERVICE=default&_DEBUG=0&MDDB=SASHELP.PRDMDDB&METABASE=SASHELP&D
=COUNTY&Ac=YEARS&A=ACTUAL&A1S1=SUM&GRT=VBAR&DC=1&ACB=1&ST=1&GL=1&GSC=1
&SSL=1&SH=1&SW=15&GW=450&DH=150&DP=1"
```
_OUTPUT_SETURL_FUNCTION_ Method

Outputs the seturl JavaScript function in the toolbar page.

Syntax

CALL SEND(OBJID,'_OUTPUT_SETURL_FUNCTION_');

Details

This function is called when either the Rotate button or the Download to Spreadsheet button is pressed.

Example

The following output is produced:

```javascript
function setURL(varName) {
    newURL='';
    with (window.parent.frames[1]) {
        newURL=eval(varName);
    }
    if (varName == 'downloadURL')
        document.location=newURL;
    else if (varName == 'rotateURL')
        window.parent.frames[1].document.location=newURL;
    }
function addtofav(varName){
    LinkName=window.document.title;
    with (window.parent.table_window) {
        linkUrl=eval(varName);
    }
    window.external.AddFavorite(linkUrl,LinkName);
}
```

_OUTPUT_SPREADSHEET_BUTTON_ Method

Outputs the Download to Spreadsheet button as an image.

Syntax

CALL SEND(OBJID,'_OUTPUT_SPREADSHEET_BUTTON_','vrflag,url,
    service,graph-variable,
    graph-statistic,graph-down,graph-type,background-type,background-
    value,title,webgis-class');
Required Arguments

**vrflag**
a flag indicating that the *View Report* button was pressed.

*Type* Numeric

**url**
the Application Broker path.

*Type* Character

**service**
the Application Broker service.

*Type* Character

**graph-variable**
the analysis variable to graph.

*Type* Character

**graph-statistic**
the statistic to graph.

*Type* Character

**graph-down**
the down dimension variable to graph.

*Type* Character

**graph-type**
the graph type.

*Type* Character

**background-type**
the background type (IMAGE or COLOR). This parameter is optional.

*Type* Character

**background-value**
the background value. This parameter is optional.

*Type* Character

**title**
the title. This parameter is optional.

*Type* Character

**webeis-class**
the WEBEIS class name (for subclassing).

*Type* Character
Example

The following example code illustrates the use of this method:

```plaintext
vrflag=1;
url='/cgi-bin/broker';
service='default';
grphvar='ACTUAL';
grphstat='SUM';
grphdown='COUNTRY';
grphtype='VBAR';
bgtype='COLOR';
bg='YELLOW';
title=' '; 
webcls=' ';
call send (webid,'_OUTPUT_SPREADSHEET_BUTTON_',vrflag,url,service,grphvar, 
grphstat,grphdown,grphtype,bgtype,bg,title,webcls);
```

The following output is produced:

```plaintext
<A HREF="/cgi-test-bin/broker/prdmdb.csv?_service=default&_debug=0 
&_program=sashelp.webeis.oprpt.scl&SPDSHT=X&mddb=SASHELP.PRDMDDB&metabase 
=SASHELP.MBEIS&D=Geographic&AC=Product%20Line&A=ACTUAL&S=SUM&ST=1&GL=1 
&DC=1&ACB=1&DIF=1&_SAVEAS=prdmdb.csv" TARGET="_self"><IMG CLASS="imgdown" 
SRC="/my_images/btn_xls.gif"ALT="Download to Spreadsheet" BORDER=0"></A>
```

_OUTPUT_SPREADSHEET_URL_ Method

Outputs the URL for the Download to Spreadsheet button as a JavaScript text string on the Report page.

Syntax

```plaintext
CALL SEND(OBJID,'_OUTPUT_SPREADSHEET_URL_',vrflag,url,service-name, 
analysis-variable,statistic,down-variable,graph-type,background-type,background-value,title, 
webcls-class);
```

Required Arguments

- **vrflag**
  - the View Report button flag.
  - Type: Numeric

- **url**
  - the Application Broker component of the URL.
  - Type: Character

- **service-name**
  - the Application Broker service value.
  - Type: Character
**analysis-variable**
the analysis variable to graph.

*Type* Character

**statistic**
the statistic to graph.

*Type* Character

**down-variable**
the down variable to graph.

*Type* Character

**graph-type**
the graph type (BLOCK, HBAR, PIE, PLOT, or VBAR).

*Type* Character

**background-type**
the background type (IMAGE or COLOR). This parameter is optional.

*Type* Character

**background-value**
the background value. This parameter is optional.

*Type* Character

**title**
the HTML page title.

*Type* Character

**webeis-class**
the WEBEIS class name.

*Type* Character

---

**Example**
The following example illustrates the use of this method:

```plaintext
vrflag=1;
url="/cgi-bin/broker";
service='default';
grphvar='ACTUAL';
grphstat='SUM';
grphdown='COUNTRY';
grptype='VBAR';
bgtype='COLOR';
bg='yellow';
title='1995 Sales Report';
webcls='SASHELP.WEBEIS.WEBEIS';
call send(_self_,'_OUTPUT_SPREADSHEET_URL_'.vrflag,url,service,grphvar,grphstat,
grphdown,grptype,bgtype,bg,title,webcls);
```
The following output is produced:

downloadURL="http://mywebserver/cgi-bin/broker/prdmddb.csv?_service=default&_debug=0
&_program=sashelp.webis.oprpt.scl&SPDSHT=X&mddb=SASHELP.PRDMDDB&metabase=SASHELP
&D=COUNTRY&AC=YEAR&A=ACTUAL&A1S1=SUM&DC=1&ACB=1&ST=1&GL=1&GSC=1&SSL=1&SH=3
&SW=15&GH=450&GW=600&DP=1&NR=ALL&BS=Star&_SAVEAS=prdmddb.csv"

.OUTPUT_STANDARD_GRAPH_. Method

Outputs the URL that drives the standard GIF Graph request.

Syntax

CALL SEND(OBJID,'_OUTPUT_STANDARD GRAPH_','url,service,graph-type,analysis-variable,statistic-variable,down-variable,across-variable,webesis-class);

Required Arguments

url
  the URL for the next query.
  Type Character

service
  the Application Broker service.
  Type Character

graph-type
  the selected graph type.
  Type Character

analysis-variable
  the analysis variable to graph.
  Type Character

statistic-variable
  the statistic to graph.
  Type Character

down-variable
  the down variable to graph.
  Type Character

across-variable
  the analysis variable to graph.
  Type Character
**webeis-class**

the WEBEIS class name.

<table>
<thead>
<tr>
<th>Type</th>
<th>Character</th>
</tr>
</thead>
</table>

**Example**

The following example illustrates the use of this method:

```plaintext
url='/cgi-bin/broker';
service='default';
grphtype='VBAR';
grphvar='ACTUAL';
grphstat='SUM';
grphdown='COUNTRY';
grphacr='PRODTYPE';
webcls=' ';
call send (webid,'_OUTPUT_STANDARD_GRAPH_',url,service,
grphtype,grphvar,grphstat,grphdown,grphacr,
webcls);
```

The following output is produced:

```plaintext
<br><br><p>
<img CLASS="graph" SRC="/cgi-bin/broker?_program=sashelp.webeis.grf2way.scl
&_service=default&mdbd=SASHELP.PRDMDDB&metabase=SASHELP.MBEIS&D=Geographic
&AC=Product%20Line&A=ACTUAL&S=SUM&grt=VBAR&gv=Actual%20Sales&gs=Sum&gd=COUNTRY&DC=1&ACB=1&gac=PRODTYPE&GSB=PRODTYPE=TOTAL&SL=%20" 
ALT="Please wait." ALIGN=CENTER WIDTH=600 HEIGHT=450 BGCOLOR=SILVER></p>
```

---

**_OUTPUT_STAT_BOXES_ Method**

Outputs the Select Column and the Available and Selected list boxes for selecting statistics based on the analysis variable.

**Syntax**

```plaintext
CALL SEND(OBJID,'_OUTPUT_STAT_BOXES_');
```

**Example**

The following output is produced:

```plaintext
<th rowspan=2 class=laylabel>
Statistics</th>
<td class=label>
Select Column
</td>
<div class="stats">
<select name="sa" class="sselect" multiple size="5" 
align="left"onChange="change(document.mf.sa); updatestatslist(document.mf.sa); ">
<option value="ACTUAL">Actual Sales</option>
</select>
</div>
```
_OUTPUT_STAT_LIST_ Method

Outputs a list of available statistics.

**Syntax**

```
CALL SEND(OBJID,'_OUTPUT_STAT_LIST_');
```

**Example**

The following example illustrates the use of the method:

```
<TR><TD CLASS="label">Statistics
<DIV CLASS="stats">
<!-- Select the available statistics -->
</DIV>
</TD></TR>
```
_OUTPUT_STATIC_HIDDEN_FLDS_ Method

Outputs the necessary hidden fields for the initial static HTML page.

Syntax

CALL SEND(OBJID,'_OUTPUT_STATIC_HIDDEN_FLDS_',metabase,background-type,background-value,webeis-class);

Required Arguments

metabase
the metabase name.

Type Character

background-value
the background image URL or color value. This parameter is optional.

Type Character

background-type
the background type (COLOR or IMAGE). This parameter is optional.

Type Character

webeis-class
the WEBEIS class name.

Type Character

_OUTPUT_SUBSET_DIMS_OPTION_ Method

Outputs text input fields for the width and height of the subset list box.

Syntax

CALL SEND(OBJID,'_OUTPUT_SUBSET_DIMS_OPTION_');
Example

The following output is produced:

```html
<TD CLASS="label">Width</TD><TD><INPUT TYPE=text NAME="sw" CLASS="select" SIZE=3 MAXLENGTH=3 VALUE="15"></TD></TR>
<TD CLASS="label">Height</TD><TD><INPUT TYPE=text NAME="sh" CLASS="select" SIZE=3 MAXLENGTH=3 VALUE="3"></TD></TR>
```

_Output_SUBSET_LOC_OPTION_ Method

Outputs a selection list for the Location option in the Filter Listboxes list.

Syntax

```
CALL SEND(OBJID,'_OUTPUT_SUBSET_LOC_OPTION_');
```

Example

The following output is produced:

```html
<TD CLASS="label">Location</TD>
<TD><SELECT NAME="ssl" CLASS="select"><OPTION VALUE="1" SELECTED>Right
 <OPTION VALUE="2">Left
 <OPTION VALUE="3">Top
 <OPTION VALUE="4">Bottom
 </SELECT></TD></TR>
```

_Output_SUBSET_SELECTIONS_ Method

Outputs the subset selection lists.

Syntax

```
CALL SEND(OBJID,'_OUTPUT_SUBSET_SELECTIONS_','subset-location');
```

Required Argument

**subset-location**

the list box location.

Type Character

Example

The following output is produced:

```html
<FONT SIZE=1>
```
_OUTPUT_SUBSETS_ Method

Outputs the list of character variables for subsetting.

Syntax

CALL SEND(OBJID,'_OUTPUT_SUBSETS_');
Example

The following output is produced:

```html
<TD CLASS="label" ALIGN=LEFT>Filter Columns: <BR>
(SELECT NAME="SV" CLASS="select" MULTIPLE SIZE=3>
  <OPTION VALUE="" SELECTED>
  <OPTION VALUE="COUNTRY">Country
  <OPTION VALUE="DIVISION">Division
  <OPTION VALUE="MONTH">Month
  <OPTION VALUE="PRODTYPE">Product type
  <OPTION VALUE="PRODUCT">Product
  <OPTION VALUE="QUARTER">Quarter
  <OPTION VALUE="REGION">Region
  <OPTION VALUE="YEAR">Year
</SELECT></TD></TR>
```

_OUTPUT_TABLE_DISP_OPTION_ Method

Outputs radio buttons for the Display Table option.

Syntax

CALL SEND(OBJID,'_OUTPUT_TABLE_DISP_OPTION_');

Example

The following output is produced:

```html
<TD CLASS="label">Display Table</TD>
<TD>
  <INPUT NAME="ST" CLASS="select" TYPE=RADIO VALUE="1" CHECKED>Yes
  <INPUT NAME="ST" CLASS="select" TYPE=RADIO VALUE="2">No
</TD>
</TR>
```

_OUTPUT_TABLE_OPTIONS_ Method

Outputs the check boxes on the Options page for the Row Totals, Column Totals, and Drillpaths options.

Syntax

CALL SEND(OBJID,'_OUTPUT_TABLE_OPTIONS_');

Example

The following output is produced:
_OUTPUT_TOOLBAR_ Method

Outputs the <FRAME> tag to create the frame in which the report is displayed.

Syntax

CALL SEND(OBJID,'_OUTPUT_TOOLBAR_',vrflag,url,service,graph-variable,graph-statistic,graph-down,graph-type,background-type,background-value,title,webelis-class,toolbar-location);

Required Arguments

vrflag
   a flag indicating that the View Report button was pressed.
   Type   Numeric

url
   the Application Broker component of the URL.
   Type   Character

service
   the Application Broker service.
   Type   Character

graph-variable
   the analysis variable to graph.
   Type   Character

graph-statistic
   the statistic to graph.
   Type   Character

graph-down
   the down dimension variable to graph.
   Type   Character

graph-type
   the selected graph type.
   Type   Character

background-type
   the background type (IMAGE, COLOR, or blank).
Type Character

**background-value**
the background value.

Type Character

**title**
the title. This value is optional.

Type Character

**webeis-class**
the WEBEIS class name.

Type Character

**toolbar-location**
the toolbar location, where 1=top, 2=bottom, 3=left, 4=right, and 5=none.

Type Character

Example
The following output is produced:

```html
<TR>
<TD>
<A HREF="/cgi-bin/broker/prdmddb.csv?_service=default&_debug=0&_program
 =SASHHELP.WEBEIS.OPRPT.SCL&DSHT=X&mddb=SASHELP.PRDMDDB&metabase=SASHELP.MBEIS
 &D=Geographic&AC=Product%20Line&A=ACTUAL&S=SUM&ST=1&GSL=1&DCA=1&DFA=1
 &_SAVEAS=prdmddb.csv" TARGET="_self"><IMG CLASS="imgdown"
 SRC="/my_images/btn_xls.gif" ALT="Download to Spreadsheet" BORDER=0></A>
</TD>
<TD>
<A href="../mddbapp.hlp/" onClick="this.href=clsurl('_PROGRAM
 =SASHHELP.WEBEIS.SHOWRPT.SCL')" TARGET="_parent"><IMG CLASS="imgrotate"
 SRC="/my_images/btn_rot.gif" ALT="Rotate" BORDER=0></A>
</TD>
<TD>
<A href="../mddbapp.hlp/" onClick="this.href=clsurl('_PROGRAM
 =SASHHELP.WEBEIS.MDDBRPTS.SCL')" TARGET="_parent"><IMG CLASS="imglay"
 SRC="/my_images/btn_lay.gif" ALT="Layout" BORDER=0></A>
</TD>
<TD>
<A HREF="http://support.sas.com/rnd/web/internet/mddbapp/hinttips.html"
 TARGET="_blank"><IMG CLASS="imghelp" SRC="/my_images/btn_hlp.gif" ALT="Help"
 BORDER=0></A>
</TD>
</TR>
```

**OUTPUT_TOOLBAR_FRAME** Method

Outputs the FRAME tag for the toolbar frame.
Syntax

CALL SEND(OBJID,'_OUTPUT_TOOLBAR_FRAME_','url','service','background-type','graph-type','background-value','graph-variable','graph-statistic','graph-down','graph-across');

Required Arguments

url
the Application Broker component of the URL.
Type Character

service
the Application Broker service.
Type Character

background-type
the background type (IMAGE, COLOR, or blank).
Type Character

graph-type
the selected graph type.
Type Character

background-value
the background value.
Type Character

graph-variable
the analysis variable to graph.
Type Character

graph-statistic
the statistic to graph.
Type Character

graph-down
the down dimension variable to graph.
Type Character

graph-across
the across dimension variable to graph.
Type Character
Example

The following output is produced:

```html
<TABLE>
  <TR>
    <TD CLASS="label">Show Totals</TD>
    <TD><INPUT TYPE="checkbox" NAME="dc" CLASS="select" VALUE="1" CHECKED>Down
    <INPUT TYPE="checkbox" NAME="acb" CLASS="select" VALUE="1" CHECKED>Across</TD>
  </TR>
</TABLE>
```

_OUTPUT_UPDATE_CLEAR_ Method

Outputs the addstatanal and remstatanal JavaScript functions on the Dimensions page.

Syntax

```javascript
CALL SEND(OBJID,'_OUTPUT_UPDATE_CLEAR_');
```

Details

The addstatanal and remstatanal functions update the list of selected analysis variables as the user makes selections for the report.

Example

The following output is produced:

```javascript
function addstatanal(select,analysisbox) {
    select.length=0;
    for (i=0; i < analysisbox.length; i++){
```
if (analysisbox.options[i].selected) {
    select.options[i] = new Option(analysisbox.options[i].text,
        analysisbox.options[i].value);
} }
}

function remstatanal(listbox) {
    if ( listbox.options.length > 0 ){
        listbox.options.length=0;
    }
    return false;
}

_OUTPUT_URL_OPTIONS_ Method
Outputs the viewer options, filter variables and selections, and expand information for a viewer URL.

Syntax
CALL SEND(OBJID,’_OUTPUT_URL_OPTIONS_’,’no-expand’);

Required Argument

no-expand
an instruction not to output or expand the information. A nonblank means do not output.

Type Character

_OUTPUT_VAR_FUNCTIONS_ Method
Outputs JavaScript functions for ordering variable selections.

Syntax
CALL SEND(OBJID,’_OUTPUT_VAR_FUNCTIONS_’);

Example
The following output is produced:

function List(list) {
    for (key in list) {
        if (list[key] !== null) this[key] = list[key];
    }
}
function change(select) {
    if ((navigator.appName == "Netscape" &&
        navigator.appVersion.indexOf("3.0") != -1) ||
    (navigator.appName == "Microsoft Internet Explorer" &&
        navigator.appVersion.indexOf("4.0") != -1)) {
        selected= new List;
        options= new Object;
        for (i= 0; i < select.options.length; i++) {
            options[select.options[i].text]=select.options[i].value;
            selected[select.options[i].text]=
                select.options[i].selected ? select.options[i].value : null;
        }
        selected= new List(selected);
        select.options.length= 0;
        for (key in selected)
            select.options[select.options.length]=
                new Option(key, selected[key], false, true);
        for (key in options)
            if (selected[key] == null)
                select.options[select.options.length]=
                    new Option(key, options[key]);
    }
}

function update() {
    str= "";
    for (key in selected)
        str= str + key + ",";
    if (str.length)
        document.form.order.value= str.substring(0, str.length - 1);
}

_OUTPUT_VARIABLE_SEL_FORM_ Method

Outputs the HTML table elements to arrange the Variable Selection page and calls the methods that output the variable and options HTML elements.

Syntax

CALL SEND(OBJID,'_OUTPUT_VARIABLE_SEL_FORM_'.url,message-id,vrflag, graph-type);

Required Arguments

url
the Application Broker component of the URL.

    Type    Character

message-id
the ID number of the message system.
<table>
<thead>
<tr>
<th>Type</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>vrflag</td>
<td>a flag indicating that the <strong>View Report</strong> button was pressed.</td>
</tr>
<tr>
<td>Type</td>
<td>Numeric</td>
</tr>
<tr>
<td>graph-type</td>
<td>the selected graph type.</td>
</tr>
<tr>
<td>Type</td>
<td>Character</td>
</tr>
</tbody>
</table>

**Example**

The following output is produced:

```html
<form action="/cgi-bin/broker" name="mf">
<tr>
<td align=top>dimensions</td>
<td><table>
<tr><td class=header>Dimensions</td></tr>
<tr class="dimselbox">
<td class=label>Down:</td>
<select name="d" class="select" size=3 multiple onchange="change(document.mf.d)">
<option selected value="Geographic">Geographic (hier)</option>
<option value="Product%2520Line">Product Line (hier)</option>
<option value="Time">Time (hier)</option>
<option value="COUNTRY">Country</option>
<option value="DIVISION">Division</option>
<option value="MONTH">Month</option>
<option value="PRODTYPE">Product type</option>
<option value="PRODUCT">Product</option>
<option value="QUARTER">Quarter</option>
<option value="REGION">Region</option>
<option value="YEAR">Year</option>
</select>
</tr>
<tr class="dimselbox">
<td class=label>Across:</td>
<select name="ac" class="select" size=3 multiple onchange="change(document.mf.ac)">
<option selected value=""/>
<option value="Geographic">Geographic (hier)</option>
<option value="Product%2520Line">Product Line (hier)</option>
<option value="Time">Time (hier)</option>
<option value="COUNTRY">Country</option>
<option value="DIVISION">Division</option>
<option value="MONTH">Month</option>
<option value="PRODTYPE">Product type</option>
<option value="PRODUCT">Product</option>
<option value="QUARTER">Quarter</option>
<option value="REGION">Region</option>
<option value="YEAR">Year</option>
</select>
</tr>
</table></td>
</tr>
</form>
```
<table>
<thead>
<tr>
<th>Analysis</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Sales</td>
<td>Sum</td>
</tr>
<tr>
<td>Sales Lag</td>
<td>% of Sum</td>
</tr>
<tr>
<td>LPERDAY</td>
<td>Average</td>
</tr>
<tr>
<td>Predicted Sales</td>
<td>Total Count</td>
</tr>
<tr>
<td>Sales Ratio</td>
<td>Minimum</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
</tr>
<tr>
<td></td>
<td>Range</td>
</tr>
</tbody>
</table>

Filter Columns:
- Country
- Division
- Month
- Product type
- Product
- Quarter
- Region
- Year

Display Table:
- Yes
- No
<table>
<thead>
<tr>
<th>Default Title</th>
<th>&lt;input name=&quot;DT&quot; class=&quot;select&quot; type=TEXT size=30 maxlength=200&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Drillpath</td>
<td>&lt;input name=&quot;DP&quot; class=&quot;select&quot; type=RADIO value=&quot;1&quot; checked&gt; Yes &lt;input name=&quot;DP&quot; class=&quot;select&quot; type=RADIO value=&quot;2&quot;&gt; No</td>
</tr>
<tr>
<td>Show Totals</td>
<td>&lt;input type=&quot;checkbox&quot; name=&quot;dc&quot; class=&quot;select&quot; value=&quot;1&quot; checked&gt; Down &lt;input type=&quot;checkbox&quot; name=&quot;acb&quot; class=&quot;select&quot; value=&quot;1&quot; checked&gt; Across</td>
</tr>
<tr>
<td>Graph Source</td>
<td>&lt;input name=&quot;GSC&quot; class=&quot;select&quot; type=RADIO value=&quot;1&quot; checked&gt; 3D Clickable Graph &lt;input name=&quot;GSC&quot; class=&quot;select&quot; type=RADIO value=&quot;2&quot;&gt; Standard GIF Graph</td>
</tr>
<tr>
<td>Location</td>
<td>&lt;select name=&quot;gl&quot; class=&quot;select&quot;&gt; &lt;option value=&quot;1&quot; selected&gt; Bottom &lt;option value=&quot;2&quot;&gt; Top &lt;option value=&quot;3&quot;&gt; Left &lt;option value=&quot;4&quot;&gt; Right &lt;/select&gt;</td>
</tr>
<tr>
<td>Type</td>
<td>&lt;select name=&quot;grt&quot; class=&quot;select&quot;&gt; &lt;option selected value=NONE&gt; None &lt;option value=VBAR&gt; Vertical bar &lt;option value=BLOCK&gt; Block &lt;option value=HBAR&gt; Horizontal bar &lt;option value=PIE&gt; Pie &lt;option value=PLOT&gt; Plot &lt;/select&gt;</td>
</tr>
<tr>
<td>Width</td>
<td>&lt;input type=text name=&quot;gw&quot; class=&quot;select&quot; size=4 maxlength=4 value=&quot;600&quot;&gt;</td>
</tr>
<tr>
<td>Height</td>
<td>&lt;input type=text name=&quot;gh&quot; class=&quot;select&quot; size=4 maxlength=4 value=&quot;450&quot;&gt;</td>
</tr>
<tr>
<td>Location</td>
<td>&lt;select name=&quot;ssl&quot; class=&quot;select&quot;&gt; &lt;option value=&quot;1&quot; selected&gt; Right &lt;option value=&quot;2&quot;&gt; Left &lt;option value=&quot;3&quot;&gt; Top &lt;option value=&quot;4&quot;&gt; Bottom &lt;/select&gt;</td>
</tr>
</tbody>
</table>

Chapter 4 / Making Advanced Customizations to the MDDB Report Viewer
_OUTPUT_VARLIST_FORM_ Method

Outputs the HTML for the reach-through to the detail variable selection page.

Syntax

CALL SEND(OBJID,'_OUTPUT_VARLIST_FORM_','data-set-name',url, html-file-id, message-id, data-set-id, service-name, debug-value, next-program, background-type, background-value);

Required Arguments

**data-set-name**
the base table data set name.

Type Numeric

**url**
the Application Broker component of the URL.

Type Numeric

**html-file-id**
the ID for the _webout file.

Type Numeric

**message-id**
the ID of the message system.

Type Numeric

**data-set-id**
the ID for the base table data set.

Type Numeric
**service-name**
the Application Broker service value.

**Type** Numeric

**debug-value**
the application server debug level.

**Type** Numeric

**next-program**
the next SCL program to execute when the form is completed.

**Type** Numeric

**background-type**
the background type (IMAGE or COLOR). This parameter is optional.

**Type** Numeric

**background-value**
the background value. This parameter is optional.

**Type** Numeric

---

**Example**

The following output is produced:

```plaintext
dataset='SASHELP.PRDSALE';
url='/cgi-bin/broker';
htmlfile=fopen('_WEBOUT','A');
msgid=instance(loadclass('sashelp.fsp.astmsg.class'),1);
dsid=open(dataset);
service='default';
dbg='0';
nextpgm='SASHELP.WEBEIS.DS2HTM.SCL';
bctype='COLOR';
bg='yellow';
call send(webid,'_OUTPUT_VARLIST_FORM_',dataset,url,htmlfile,msgid,dsid,service,
dbg,nextpgm,bctype,bg);
```

---

**_OUTPUT_VARLIST_FUNCTIONS_ Method**

Outputs the var_order, resetfields, and pickall JavaScript functions on the reach-through variable selection page.

**Syntax**

```plaintext
CALL SEND(OBJID,'_OUTPUT_VARLIST_FUNCTIONS_','data-set-id',html-file-id);
```
Required Arguments

**data-set-id**
the base table data set identifier.

*Type* Numeric

**html-file-id**
the identifier for the _webout_ file.

*Type* Numeric

Example

```javascript
htmlfile=fopen('_WEBOUT','A');
dsid=open('SASHELP.PRDSALE');
call send(webid,'_OUTPUT_VARLIST_FUNCTIONS_',dsid,htmlfile);
```

The following output is produced:

```javascript
varorder = new Array();
varlabel = new Array();
varorder.num = 0;
if (navigator.appName == 'Netscape') document.forms[0].reset();
function var_order(fieldnum,labeltext)
{ if (document.forms[0].elements[fieldnum].checked)
{ varorder[varorder.num] = document.forms[0].elements[fieldnum].value;
 varlabel[varorder.num] = labels[fieldnum];
 varorder.num++
 }
else
{ for(i = 0; i < varorder.num; i++)
 { if (varorder[i] == document.forms[0].elements[fieldnum].value)
 { for(j = i; j < varorder.num; j++)
 { varorder[j] = varorder[j+1];
 varlabel[j] = varlabel[j+1];
 }
 }
 }
 varorder.num--;
 }
resetfields(labeltext);
}
function resetfields(labeltext)
{ document.forms[0].elements[labeltext].value = ' ';
```
document.forms[0].elements[0].value = ' ';
if (varorder.num > 0)
{
    document.forms[0].elements[labeltext].value = varlabel[0];
    document.forms[0].elements[0].value = varorder[0];
}
for(i = 1; i < varorder.num; i++)
{
    document.forms[0].elements[labeltext].value =
    document.forms[0].elements[labeltext].value + '\r\n'+ varlabel[i];
    document.forms[0].elements[0].value =
    document.forms[0].elements[0].value + ' ' + varorder[i];
}
}
function pickall(num)
{
    for (i = 1; i <= num ; i++)
    {
        if (document.forms[0].elements[i].checked == false)
        {
            varlabel[varorder.num] = labels[i];
            varorder[varorder.num] = document.forms[0].elements[i].value;
            document.forms[0].elements[i].checked = true;
            varorder.num++;
        }
    }
    resetfields(num+1);
}

_OUTPUT_VARLIST_HTML_ Method

Outputs the HTML for the reach-through to the detail variable selection page.

Syntax

CALL SEND(OBJID,'_OUTPUT_VARLIST_HTML_','data-set-id,html-file-id,
message-id,
data-set-name,url,service-name, debug-value,next-program,background-type,
background-value);

Required Arguments

data-set-id
    the ID for the base table data set.
    Type  Numeric

html-file-id
    the ID for the _webout file.
    Type  Numeric

message-id
    the ID of the message system.
    Type  Numeric
**data-set-name**
the base table data set name.
Type Numeric

**url**
the Application Broker component of the URL.
Type Numeric

**service-name**
the Application Broker service value.
Type Numeric

**debug-value**
the application server debug level.
Type Numeric

**next-program**
the next SCL program to execute when the form is completed.
Type Numeric

**background-type**
the background type (IMAGE or COLOR). This parameter is optional.
Type Numeric

**background-value**
the background value. This parameter is optional.
Type Numeric

Example

dataset='SASHELP.PRDSALE';
dsid=open(dataset);
htmlfile=fopen('_WEBOUT','A');
msgid=instance(loadclass('sashelp.fsp.astmsg.class'),1);
url='/cgi-bin/broker';
service='default';
debug='0';
nextpgm='SASHELP.WEBEIS.DS2HTM.SCL';
bgtype='COLOR';
bg='yellow';
call send(webid,'_OUTPUT_VARLIST_HTML_',dsid,htmlfile,msgid,dataset,url,service,
        debug,nextpgm,bgtype,bg);

_OUTPUT_VIEWRPT_BUTTON_ Method

Outputs the View Report button.
Syntax

CALL SEND(OBJID,'_OUTPUT_VIEWRPT_BUTTON_');

Example

The following output is produced:

```<INPUT TYPE="submit" NAME="view" CLASS="submit" VALUE="View Report">```

__OUTPUT_VIEWRPT2_BUTTON__ Method

Outputs the View Report button on the Dimensions page.

Syntax

CALL SEND(OBJID,'_OUTPUT_VIEWRPT2_BUTTON_');

Example

The following output is produced:

```<A href="/mdbapp.hlp/" onClick="this.href=geturl(document.mf.d,document.mf.ac,document.mf.a)"TARGET="_parent">
<IMG SRC="view-report.gif" width="29" height="24"></A>```

_POST_DISPLAY_OPTIONS_ Method

Specifies additional options on the Layout page.

Syntax

CALL SEND(OBJID,'_POST_DISPLAY_OPTIONS_','<parameter-list>');

Optional Argument

`parameter-list`

An optional list for passing in information to the method.

Type  Numeric

Details

This stub method is called after all of the display options are called. It is useful for adding additional options to the Layout page.
_PRE_DISPLAY_OPTIONS_ Method
Specifies additional options on the Layout page.

Syntax
CALL SEND(OBJID,'_PRE_DISPLAY_OPTIONS_','<parameter-list>);

Optional Argument

*parameter-list*  
an optional list for passing in information to the method.

Type  Numeric

Details
This stub method is called before any of the display options are called. It is useful for adding additional options to the Layout page.

_PRINT_A_BLANK_ Method
Prints the character code to fill an empty cell.

Syntax
CALL SEND(OBJID,'_PRINT_A_BLANK_');

_SET ACROSS TOTAL FLAG_ Method
Sets the atotal_ instance variable to activate across totals.

Syntax
CALL SEND(OBJID,'_SET_ACROSS_TOTAL_FLAG_','total-flag');

Required Argument

*total-flag*  
a value that indicates whether to set a flag for the totals in the across dimension, where X=set the flag on and blank=do not set the flag.
**_SET_DOWN_TOTAL_FLAG_ Method**

Sets the dtotal_instance variable to activate down totals.

**Syntax**

```
CALL SEND(OBJID,'_SET_DOWN_TOTAL_FLAG_','total-flag');
```

**Required Argument**

**total-flag**

A value that indicates whether to set a flag for the totals in the down dimension, where X=set the flag on and blank=do not set the flag.

**Type** Character

---

**_SET_DRILL_LEVELS_ Method**

Updates the SAVED_L sublist on the application list to set the drill-down values.

**Syntax**

```
CALL SEND(OBJID,'_SET_DRILL_LEVELS_','application-list');
```

**Required Argument**

**application-list**

The list ID of the application list. For more information about application lists, see the online Help for SAS/EIS software.

**Type** Numeric

**Details**

This method

- builds the HIERARCHIES_L and SAVED_L sublists on the application list if the list is empty
- builds the CURRENT_DRILLS sublist on the HIERARCHIES_L sublist if it is empty
- updates the CURRENT_DRILLS sublist for each hierarchy with the current drill-down information
- sets the CURRENT_LEVEL value for each hierarchy on the HIERARCHIES_L sublist.

**Example**

```plaintext
applist= makelist();
rc=fillist('CATALOG','SASHELP.EISRG.ONEWAY.EIS',applist);
call send(webid,'_SET_DRILL_LEVELS_',applist);
```

__-_SET_EMDBBMID_ _Method__

Sets the EMDBBMID_ instance variable.

**Syntax**

```plaintext
CALL SEND(OBJID,'_SET_EMDBBMID_','id');
```

**Required Argument**

- **id**
  - the ID of the data model.
  - **Type**: Numeric

__-_SET_EXPAND_FLAG_ _Method__

Sets the expflag_ instance variable that indicates whether values can be expanded.

**Syntax**

```plaintext
CALL SEND(OBJID,'_SET_EXPAND_FLAG_','row-list,actionsl');
```

**Required Arguments**

- **row-list**
  - the rowlist from the GET_CLASS_COMBINATIONS method.
  - **Type**: Numeric

- **actionsl**
  - the actionsl list from the data model.
  - **Type**: Numeric
_SET_HIERL_LIST_ Method
Sets the hierl_ instance variable.

Syntax
CALL SEND(OBJID,'_SET_HIERL_LIST_','list-id');

Required Argument
list-id
the list ID of the target list to copy.

Type Numeric

_SET_SUBSET_BY_LIST_ Method
Builds the subset_by_ list from the filter value selections.

Syntax
CALL SEND(OBJID,'_SET_SUBSET_BY_LIST_');

Example
The following illustrates an example of a subset_by_ list:

subset_by_ ( COUNTRY = {'CANADA'
    }
    DIVISION = {'EDUCATION'
    }
    MONTH = {'Jan'
        'Feb'
    }
}

_SET_SUBSET_FLAG_ Method
Sets the value of the SUBSET_FLAG_ instance variable.
Syntax
CALL SEND(OBJID,'_SET_SUBSET_FLAG_',flag-value);

Required Argument

flag-value
the value of the subset flag.
Type: Character

_SET_SUBSETS_LIST_ Method
Defines the subsets to be used.

Syntax
CALL SEND(OBJID,'_SET_SUBSETS_LIST_',variable-number);

Required Argument

variable-number
the number of selected subset values.
Type: Numeric

Details
This method sets and fills the subvars_ instance variable and adds the subvars_list to the _self_list data model for applying the filters.

_SHOW_GRAPH_ Method
Sets the graphing variables and calls a graphing method.

Syntax
CALL SEND(OBJID,'_DISPLAY_GRAPH_',url,service,_argument-string,_argument-string-2,
  graph-type,analysis-variable,statistic-variable,down-variable,across-variable,webeis-class);

Required Arguments

url
the Application Broker component of the URL.
service
the Application Broker service.

_argument-string
the argument string for the next query.

_argument-string-2
the argument string for the next query.

graph-type
the selected graph type.

analysis-variable
the analysis variable to graph.

statistic-variable
the statistic to graph.

down-variable
the down variable to graph.

across-variable
the across variable to graph.

webeis-class
the WEBEIS class name.

Details
This method sets the default graphing variables if their values have not been specified and calls the appropriate graphing method (_OUTPUT_STANDARD_GRAPH_ or _OUTPUT_CLICKABLE_GRAPH_) for the selected graph source.
_SUBMIT.GOPTIONS_ Method
Submits the SAS/GRAPH GOPTIONS statement for the standard GIF graph.

Syntax
CALL SEND(OBJID,'_SUBMIT.GOPTIONS_.gif-device');

Required Argument

gif-device
    the name of the device driver to use.

Type    Character

_SUBMIT_GRAPH_PATTERN_ Method
Submits the SAS/GRAPH PATTERN statements for the standard GIF graphs.

Syntax
CALL SEND(OBJID,'_SUBMIT_GRAPH_PATTERN_');

_SUBMIT_GRAPH_TITLE_ Method
Submits the SAS/GRAPH TITLE statement for the standard GIF graph.

Syntax
CALL SEND(OBJID,'_SUBMIT_GRAPH_TITLE_.statistic,analysis-variable');

Required Arguments

statistic
    the statistic used in the graph.

Type    Character

analysis-variable
    the analysis variable used in the graph.

Type    Character
_UPDATE_STATS_LIST_ Method

Outputs the updatestatslist JavaScript function on the Dimensions page.

Syntax

CALL SEND(OBJID,'_UPDATE_STATS_LIST_');

Details

The updatestatslist function modifies the list of available and selected statistics as the user makes statistic selections for the report display.

Example

The following output is produced:

```javascript
function updatestatslist(select) {
    pos = 0;
    num = 0;
    newlength = 0;
    var arrayname = "";
    var analysistype = "";
    var arrayofstats = "";
    for (i=0; i < select.options.length; i++) {
        if (select.options[i].selected) {
            num=num+1;
            arrayname = select.options[i].value+"STATS";
            analysisarray=eval(arrayname);
            if (analysistype.indexOf(analisysarray[0])==-1 ) {
                analysistype=analysisarray[0] +"," +analysistype;
            }
        }
    }
    if (analysistype.substr(eval(analysistype.lastIndexOf","+1), 1)==="") {
        analysistype=analysistype.slice(0,analysistype.lastIndexOf","));
    }
    arrayoftypes = analysistype.split",";
    arrayoftypes.sort();
    document.mf.as.options.length=0;
    document.mf.s.options.length=0;
    if (num > 1) {
        for (i=0; i < arrayoftypes.length; i++) {
            for ( i=0 ) {
                arrayname = eval(arrayofstats[0]+"desclist");
                pos = arrayname.length;
                for ( j=0; j < arrayname.length; j++) {
                    document.mf.as.options[j] = new Option(statslabellist[arrayname[j]],
                    arrayname[j]);
                }
            }
        }
    }
```
else if (arrayoftypes[i]=="nunique") {
    arrayname = eval(arrayoftypes[i] +"desclist");
    document.mf.as.options[pos] = new Option(statslabellist[arrayname[0]],
        arrayname[0]);
}
}
document.mf.s.options[0] = new Option("*MIXED SELECTIONS", "MIXED");
}
else if (num==1) {
    k=0;
    arrayofstats=eval(arrayoftypes[0] +"desclist");
    for (i=0; i<select.options.length; i++) {
        if (select.options[i].selected) {
            arrayname = eval(select.options[i].value+"STATS");
            for (j=1; j < arrayname.length; j++) {
                document.mf.s.options[j-1] = new Option(statslabellist[arrayname[j]],
                    arrayname[j]);
            }
        }
    }
    for (i=0; i < arrayofstats.length; i++) {
        var repeat="false";
        for (j=1; j < arrayname.length; j++) {
            if (arrayofstats[i]==arrayname[j]) {
                repeat="true";
                break;
            }
        }
        if (repeat=="false" && arrayofstats[i]!="") {
            document.mf.as.options[k] = new Option(statslabellist[arrayofstats[i]],
                arrayofstats[i]);
            k++;
        }
    }
}