About MDDB Report Viewer
Support for Access Control Features
Requirements for Running the MDDB Report Viewer
Setting Up the MDDB Report Viewer
Working with Repositories
Specifying the System Repository Manager Location
Setting Up the System Repository Manager Files
Defining the Repository to Application Dispatcher
Setting Up the SASHELP Repository
Using the MDDB Report Viewer
Advanced Customizations
MDDB Report Viewer Class, Instance Variables, Flow of Control, and Methods
Flow of Control for the Layout Page
Flow of Control for the Dimensions Page
Flow of Control for the Layout Toolbar
Flow of Control for the Report Page
Flow of Control for the Report Page (Part 1)
Flow of Control for the Report Page (Part 2)
Flow of Control for the Report Page (Part 3)
Flow of Control for the Report Page Toolbar (Part 1)
Flow of Control for the Report Page Toolbar (Part 2)
_BUILD_ACROSSL_LIST_Method
_BUILD_ANALYSIS_LIST_Method
_BUILD_ANLSORTORDER_Method
_BUILD_APPLICATION_LIST_Method
_BUILD_CURRENT_SUBSETS_Method
_BUILD_DOWNL_LIST_Method
_BUILD_STATSL_LIST_Method
_BUILD_TOTAL_Method
_BUILD_URL_ONSUBMIT_Method
_BUILD_WHERE_FORMAT_STRING_Method
_CHECK_HIER_MEMBER_Method
_CLOSE_FORM_Method
_CLOSE_PAGE_Method
_CLOSE_STATIC_FORM_Method
_CREATE_STAT_ARRAYS_Method
_DISPLAY_ACROSS CELLS_Method
- _OPEN_TWOWAY_Method
- _OPEN_WEBOUT_FOR_SPDSHT_Method
- OUTPUT_ACROSS_LIST_Method
- OUTPUT_ADDTL_CLSVAL_PARMS_Method
- OUTPUT_ADDTL_RT_PARMS_Method
- OUTPUT_ADDTOFAV_FUNCTION_Method
- OUTPUT_ALL_URL_ITEMS_Method
- OUTPUT_ANAL_LIST_Method
- OUTPUT_ANAL_SELECT_Method
- OUTPUT_ARROW_FUNCTIONS_Method
- OUTPUT_BAR_SHAPE_LIST_Method
- OUTPUT_BOOKMARK_BUTTON_Method
- OUTPUT_BOOKMARK_URL_Method
- OUTPUT_CLASSVAL_URL_FN_Method
- OUTPUT_CLICKABLE_GRAPH_Method
- OUTPUT_CONTENT_HEADER_Method
- OUTPUT_CSV_CONTENT_HEADER_Method
- OUTPUT_DEBUG_LIST_Method
- OUTPUT_DEFLT_TITLE_OPTION_Method
- OUTPUT_DIMBTN_URL_FN_Method
- OUTPUT_DIMENSIONS_BUTTON_Method
- OUTPUT_DOWN_LIST_Method
- OUTPUT_DP_TITLE_OPTION_Method
- OUTPUT_DS2HTM_HTML_Method
- OUTPUT_DS2HTM_ST_Method
- OUTPUT_DYNAMIC_HIDDEN_FLDS_Method
- OUTPUT_EMPTY_CELL_Method
- OUTPUT_EMPTY_SERVICE_LIST_Method
- OUTPUT_GRAPH_DIMS_OPTION_Method
- OUTPUT_GRAPH_INSTR_Method
- OUTPUT_GRAPH_LIST_Method
- OUTPUT_GRAPH_LOC_OPTION_Method
- OUTPUT_GRAPH_OPTION_Method
- OUTPUT_GRAPH_SOURCE_OPTION_Method
- OUTPUT_GRAPH_TABLE_DISP_Method
- OUTPUT_HDR_Method
- OUTPUT_HELP_BUTTON_Method
- OUTPUT_HIDDEN_FIELDS_Method
● OUTPUT_HIDDEN_VARS_Method
● OUTPUT_HTML_AFTER_BODY_Method
● OUTPUT_HTML_BEF_CLOSE_BODY_Method
● OUTPUT_HTML_FORM_HEADER_Method
● OUTPUT_LAYOUTBUTTON_Method
● OUTPUT_LAYOUTFRAME_Method
● OUTPUT_LAYOUTTOOLBAR_Method
● OUTPUT_LOGOUT_BUTTON_Method
● OUTPUT_MAIN_TOOLBAR_FRAME_Method
● OUTPUT_MDDB_LIST_Method
● OUTPUT_NUMROWS_LINKS_Method
● OUTPUT_NUMROWS_OPTION_Method
● OUTPUT_OPTBTN_URL_FN_Method
● OUTPUT_OPTIONS_BUTTON_Method
● OUTPUT_OPTIONS_FORM_Method
● OUTPUT_REACHTHRU_LINK_Method
● OUTPUT_REACHTHRU_URL_FN_Method
● OUTPUT_REPORT_FRAME_Method
● OUTPUT_REPORT_RADIO_BTNS_Method
● OUTPUT_REPORT_TYPE_SELECT_Method
● OUTPUT_ROTATE_BUTTON_Method
● OUTPUT_ROTATE_URL_Method
● OUTPUT_SETURL_FUNCTION_Method
● OUTPUT_SPREADSHEET_BUTTON_Method
● OUTPUT_SPREADSHEET_URL_Method
● OUTPUT_STANDARD_GRAPH_Method
● OUTPUT_STAT_BOXES_Method
● OUTPUT_STATIC_HIDDEN_FLDS_Method
● OUTPUT_STAT_LIST_Method
● OUTPUT_SUBSET_DIMS_OPTION_Method
● OUTPUT_SUBSET_LOC_OPTION_Method
● OUTPUT_SUBSETS_Method
● OUTPUT_SUBSET_SELECTIONS_Method
● OUTPUT_TABLE_OPTIONS_Method
● OUTPUT_TABLE_DISP_OPTION_Method
● OUTPUT_TOOLBAR_Method
● OUTPUT_TOOLBAR_FRAME_Method
● OUTPUT_TOTALS_OPTIONS_Method
● OUTPUT_UPDATE_CLEAR_Method
● OUTPUT_URL_OPTIONS_Method
● OUTPUT_VAR_FUNCTIONS_Method
● OUTPUT_VARIABLE_SEL_FORM_Method
● OUTPUT_VARLIST_FORM_Method
● OUTPUT_VARLIST_FUNCTIONS_Method
● OUTPUT_VARLIST_HTML_Method
● OUTPUT_VIEWRPT_BUTTON_Method
● OUTPUT_VIEWRPT2_BUTTON_Method
● POST_DISPLAY_OPTIONS_Method
● PRE_DISPLAY_OPTIONS_Method
● PRINT_A_BLANK_Method
● SET_ACROSS_TOTAL_FLAG_Method
● SET_DOWN_TOTAL_FLAG_Method
● SET_DRILL_LEVELS_Method
● SET_EMDBMID_Method
● SET_EXPAND_FLAG_Method
● SET_HIERL_LIST_Method
● SET_SUBSET_BY_LIST_Method
● SET_SUBSET_FLAG_Method
● SET_SUBSETS_LIST_Method
● SHOW_GRAPH_Method
● SUBMIT_GOPTIONS_Method
● SUBMIT_GRAPH_PATTERN_Method
● SUBMIT_GRAPH_TITLE_Method
● UPDATE_STATS_LIST_Method
● MDDB Report Viewer Variables
● Customizing the MDDB Report Viewer Using Cascading Style Sheets
About 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 PROC MDDB when SAS OLAP Server software 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.

See the following topics for more information about the MDDB Report Viewer:

- Requirements
- Setting Up
- Using the MDDB Report Viewer
- Customizing the 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 SAS OLAP Server software must be licensed at your site.

  **Note:** MDDB Report Viewer 9.2 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 product).

- The MDDB that you will 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 SAS OLAP Server software 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 documentation 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 regular SAS data sets.

- Your Web browser must support HTML pages with frames.
Setting Up the MDDB Report Viewer

The MDDB Report Viewer consists of three HTML pages in which users can enter information in order 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 will use more than one 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 to learn more about using access control. For complete information on access control, see SAS OLAP Server Administrator's Guide, Release 8.1.

Note: In order 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 on this setup procedure, see Working With Repositories. You can also refer to the online SAS Help and Documentation for Base SAS software and SAS/EIS software for details on 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.rptseld.scl&_service=myservice&metabase=sashelp.mbeis&bgtype=color&bg=red&class=sashelp.override.myweb.class

where broker-URI, bgtype, bg, and class are as described in Method 3, step 2, below. With this method, no HTML pages are created and stored.

Method 3

Run the SAS AF command in order 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, type the following command in the Program Editor window and submit the command to SAS for processing:

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

   where

   **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 can contain 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 should normally be 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 will appear at the top of the report. A title value is optional. If you do not specify a title, the title "Multidimensional Reports" will be used.

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

   **bgtype**
   
   is the type of background that will appear in the application reports. Specify \text{bgtype='color'} to control the color of the background or \text{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 will be 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 will be the default color, silver.

**Note:** If you specify an invalid color value, your Web browser will map the specification to a valid value.

**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 3-level or a 4-level name. For example, the following are both valid:

```
sashelp.override.myweb
```

```
sashelp.override.myweb.class
```

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

```html
<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 optionally modify the list of debug options for your site in the following HTML line:

```html
Debugging level: <SELECT NAME="_debug">
```

3. Start the Application Server and point your Web browser to the HTML file that is generated in Method 3, step 2, above.

You can specify the metabase, pathname, CGI, title, bgtype, bg, and class options in any order. Run the Application Server for each repository that contains MDDBs that users will access when they run their reports.
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 in order to set up the Common Metadata Repository:

Note: You must have write access to the SASHELP directory to complete the following tasks.

1. Specify the system repository manager location.
2. Set up the system repository manager files.
3. Define the repository to Application Dispatcher Server.
4. Set up the SASHELP repository.
Specifying the System Repository Manager Location

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

1. Create a directory that will be dedicated exclusively to the storage of repository manager files, for example:
   - Windows users: !SASROOT\RPOSMGR
   - UNIX users: !SASROOT/RPOSMGR
   - VMS users: !SAS$ROOT:[RPOSMGR]

   This directory should not be used to store other SAS files.

   **Note:** This system repository manager path will be used later in this task.

2. Type `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 select **OK**. From the menu bar, select **File ➤ Close** to close the Registry Editor window.

3. Type `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 select **OK**.

4. Select the **Path** item in the right window. From the pop-up menu, select **Modify**. Type the path from Step 1, above; for example, type `!SASROOT\RPOSMGR`. Select **OK** to close the Edit String Value window. From the menu bar, select **File ➤ Close** to close the Registry Editor window and to save the changes.
Setting Up the System Repository Manager Files

Complete the following steps in order to set up the necessary system repository manager files. You must have write access to SASHELP in order to specify the system repository manager.

1. Create a directory that will be dedicated exclusively to the storage of repository manager files. For example:
   - Windows users: !SASROOT\RPOSMGR
   - UNIX users: !SASROOT/RPOSMGR
   - VMS users: !SAS$ROOT: [RPOSMGR]

   Do not store other SAS files in this directory.

2. At a SAS command line, type REPOSMGR and then select Setup Repository Manager.
3. In the Repository Manager Setup window, Library will default to RPOSMGR. For Path, specify the path from step 1, above, and then select the Write values to system registry check box. Then select OK.
4. In the resulting dialog box, select Yes in order to generate the necessary repository manager files.

This completes the setup for the System Repository Manager. You can create additional repository managers (a user repository manager, for example) by repeating the steps above 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:

```plaintext
ALLOCATE LIBRARY RPOSMPGR 'rposmgr-path';
DATALIBS RPOSMPGR;
```
Setting Up the SASHELP Repository

Complete the following steps in order to set up the SASHELP repository:

1. At a SAS command line, type `REPOSMGR` and then select **Repository Registration**.
2. In the Repository Registration window, select **New**.
3. In the Register Repository (New) window, type `SASHELP` (in uppercase) in the **Repository** field. In the **Path** field, type the full directory path where the CORE catalog is located. For example:
   
   - **Windows users:** `\SASROOT\CORE\SASHELP`
   - **UNIX users:** `!SASROOT/sashelp`
   - **VMS users:** `!SAS$ROOT:[HELP]`

4. In the **Description** field, you can type any character string (for example, SASHELP Repository). Select **OK** to close the Register Repository (New) window. Select **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 will be referenced as a repository.
Using the MDDB Report Viewer

Using the Interface

- How do I use the MDDB Report Viewer?
- How do I select items from a selection list?
- How will I know whether the items that I select for a report are valid?
- What does the Rotate button do?
- How does Download to spreadsheet work?

Printing Reports

- How can I print reports?
- Can I print extremely large tables?

Changing the Appearance of a Report

- Can I change report dimensions from the Report page?
- Can I change the colors of my report?

Viewing Your Data

- How do I drill down to additional values in a report?
- How do I subset my report data?
- How do I see the detail data?

Creating Graphs

- How do I generate a 3D graph of the report data?
• How do I generate a standard GIF graph of the report data?
• How do I change the font for the standard GIF graph?

Modifying the Default MDDB Report Viewer Settings

• How do I specify the repository manager for the Application Dispatcher Server?
• How do I specify a different delimiter for Download to spreadsheet?
• Can I create my own Help page?
• Can I use cascading style sheets in order to modify the appearance of my report?
• Can I change the toolbar location?
• Can I display reports without the Down and Across list boxes?
• Can I disable the sorting feature?
• Can I disable the row paging feature?
• Can I modify the settings for the number of rows to display?
• Can I change the number of paging links that are displayed beneath the report table?
• How do I specify to the viewer not to use HTML frames?
• Can I change the appearance of the report table?

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 be used.

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 be included 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 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 specify 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 respecify new variables and select subset values in order 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 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 will be redisplayed with the subset applied. If a graph was previously displayed, it will be 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 will 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 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.

**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.

**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 go back 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 will be readable.

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

**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 will be redisplayed with the subset applied. If a graph was previously displayed, it will be 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.

How do I generate a 3D 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 click the right mouse button 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. In order to drill down using the GIF graph, you must perform the drilldown 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:

```plaintext
%let _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 Software: Reference*. For more information about the REQUEST INIT program, see the [PROC APPSRV. REQUEST statement syntax](#).

**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:

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

**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:

```plaintext
%let _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/intrnet/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:

```plaintext
%let _mrvhelp=http://myserver/myhelp.html;
```
Can I use cascading style sheets to modify the appearance of my report?

MDDB Report Viewer, Version 8 and later support 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 [Customizing the MDDB Report Viewer Using Cascading Style Sheets](#).

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 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 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 broker configuration file:

```
ServiceSet _MRNOPGOP "X"
```
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 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 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 will be 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 CELLSspacing=2 BORDER=3"
```

These attributes are inserted into the <TABLE> tag for the report.
Advanced Customizations

You can customize the MDDB Report Viewer by modifying the following components:

- the MDDB Report Viewer class, instance variables, flow of control, and methods
- the MDDB Report Viewer macro variables and global variables
- the MDDB Report Viewer cascading style sheets.
MDDB Report Viewer Class, Instance Variables, Flow of Control, and Methods

- The MDDB Report Viewer class
- Instance variables
- Flow of control in the MDDB Report Viewer class
- MDDB Report Viewer methods

---

The MDDB Report Viewer Class

The MDDB Report Viewer class is a viewer that is used to display 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 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

---

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.
ACRVAR$_
    specifies the list of selected variables for the across dimension.
ALEVELS_
    specifies the list of drill-down levels for the across variables.
ANALLBL5S_
   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 3- or 4-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 to be displayed.
DTOTAL_
   specifies a flag that indicates whether the down totals are turned on.
EMDDDBMID_
   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 only the expanded rows.

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 = a 3-D clickable graph, and 2 = 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.
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 on the methods listed in these figures, refer to the individual method descriptions.

- **Figure 1. Flow of Control for the Layout Page (MDDBRPTS.SCL)**
MDDB Report Viewer Methods

The MDDB Report Viewer class contains the methods listed below. Each method description contains a brief summary of the method's purpose and the syntax for the method. Some method descriptions also contain an example of how the method is used.

- **BUILD_ACROSSL_LIST**
- **BUILD_ANALYSIS_LIST**
- **BUILD_ANLSORTORDER**
- **BUILD_APPLICATION_LIST**
- **BUILD_CURRENT_SUBSETS**
- **BUILD_DOWNL_LIST**
- **BUILD_STATSL_LIST**
- **BUILD_TOTAL**
- **BUILD_URL_ONSUBMIT**
- **BUILD_WHERE_FORMAT_STRING**
- **CHECK_HIER_MEMBER**
- **CLOSE_FORM**
- **CLOSE_PAGE**
- **CLOSE_STATIC_FORM**
- **CREATE_STAT_ARRAYS**
- **DISPLAY_ACROSS_CELLS**
- **DISPLAY_ANALYSIS_VARS**
- **DISPLAY_DEFAULT_TITLE**
- **DISPLAY_DOWNVAR_CELL**
- **DISPLAY_ERROR**
- **DISPLAY_ONEWAY**
• _OUTPUT_VIEWRPT_BUTTON_
• _OUTPUT_VIEWRPT2_BUTTON_
• _POST_DISPLAY_OPTIONS_
• _PRE_DISPLAY_OPTIONS_
• _PRINT_A_BLANK_
• _SET ACROSS TOTAL FLAG_
• _SET DOWN TOTAL FLAG_
• _SET DRILL LEVELS_
• _SET EMDDBMID_
• _SET EXPAND FLAG_
• _SET HIERL LIST_
• _SET SUBSET BY LIST_
• _SET SUBSET FLAG_
• _SET SUBSETS LIST_
• _SHOW GRAPH_
• _SUBMIT GOPTIONS_
• _SUBMIT GRAPH PATTERN_
• _SUBMIT GRAPH TITLE_
• _UPDATE_STATS_LIST_
Flow of Control for the Layout Page

Figure 1. Flow of Control for the Layout Page

Initialization and error checking

Create an instance of WBEIS class

Open _webout for writing HTML

Open repository

Call OUTPUT_FRAME_HDR2_ method to generate the Dimensions and Options frames

Close repository and _webout, and terminate WBEIS class

Return

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

Figure 2. Flow of Control for the Dimensions Page

1. Initialization and error checking
2. Instance WEBEIS class
3. Open repository
4. Call GET_DOWNVAR_LIST
5. Call GET_ANALYSIS_VARS
6. Call GETAVAILABLE_STATS
7. Open _webout for writing HTML
8. Call GETOUTPUT_FILE_ID
9. Call OUTPUT_HDR
10. Call OUTPUT_VARIABLE_SELFORM
11. Output all HTML hidden fields for Dimensions form
12. Close repository and _webout, and terminate WEBEIS class
13. Return
Terminate WEBEIS class

Return
Flow of Control for the Layout Toolbar

Figure 3: Flow of Control for the Layout Toolbar

1. Initialization and error checking
2. Create an instance of WEBEIS class
3. Open_webout for writing HTML
4. Call_GET_MESSAGE_ID_
5. Call_GET_OUTPUT_FILE_ID_
6. Output <HTML> content type header
7. Output <HTML>, <HEAD>, and <TITLE> tags
8. Output page title
9. Output <TITLE> tag
10. StyleSheet parameter specified
    a. Yes: Output <LINK> tag for stylesheetinfo
    b. No: Output <SCRIPT> tag
11. Call_OUTPUT_DIMBTN_URL_ARG
12. Call_OUTPUT_OPTBTN_URL_ARG
13. Output <SCRIPT> and <HEAD> tags
14. Output <BODY> tag
15. Output <TABLE> tag
16. Call_OUTPUT_LAYOUT_TOOLBAR
17. Output <TABLE>, <BODY>, and <HTML> tags
18. Close webout
Call _OUTPUT_LAYOUT_TOOLBAR_

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

Close _webout_

Terminate WEB EIS class

Return
Flow of Control for the Report Page

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_METADATA_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.
Flow of Control for the Report Page (Part 1)

Figure 5. Flow of Control for the Report Page (part 1)
Flow of Control for the Report Page (Part 2)

Figure 5. Flow of Control for the Report Page (part 2)

Flowchart details as shown in the diagram.
Flow of Control for the Report Page (Part 3)

Figure 5. Flow of Control for the Report Page (part 3)
Flow of Control for the Report Page Toolbar (Part 1)

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

- 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/HTML content type header
- Output <HTML>, <HEAD>, and <TITLE> tags
- Output title information
- Output <TABLE> tag
- Stylesheet parameter specified?
  - Yes: Output <LINK> tag with stylesheet information
  - No: Output <SCRIPT> tag
- Call _OUTPUT_VAR_FUNCTIONS_
- Call _OUTPUT_ROTAUTE_FUNCTION_
- Call _OUTPUT_CLASSVAL_URL_RN_
Call _OUTPUT_ROTATE_FUNCTION_

Call _OUTPUT_CLASSVAL_URL_FN_

Call _OUTPUT_SET_URL_FUNCTION_

Call _OUTPUT_ADDTOFAV_FUNCTION_

(Continued)
Flow of Control for the Report Page Toolbar (Part 2)

Figure 6. Flow of Control for the Report Page Toolbar (part 2)

(Continued)

- Output `</SCRIPT>` tag
- Output `</HEAD>` tag
- Output `</BODY>` tag
- Output page background information
- Output `</TABLE>` tag
- Call `OUTPUT_TOOLBAR()`
  - Output `</TABLE>`, `</BODY>`, and `</HTML>` tags
- Close_webout
- Terminate WEBEIS class
- Return
_BUILD_ACROSS_LIST_ Method

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

This method

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

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>application-list</td>
<td>N</td>
<td>the list ID of the application list. For more information on application lists, see the online documentation for SAS/EIS software.</td>
</tr>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the variable that is selected for the across dimension (optional, and no longer used).</td>
</tr>
</tbody>
</table>

Example

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

The following sublist will be added to the application list:

Across:( PRODUCT LINE= ( HIERARCH= 'Product Line' )[1081] ) [989]
BUILD_ANALYSIS_LIST_ Method

Builds the analysis sublist on the application list

This method

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

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>application-list</td>
<td>N</td>
<td>the list ID of the application list. For more information on application lists, see the online Help for SAS/EIS software.</td>
</tr>
</tbody>
</table>

Example

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

The following sublist will be 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

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

Where...  Is Type...  And Contains...

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

Example

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

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_ACROSSL_LIST 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 on the structure of application lists, see the online Help for SAS/EIS software.

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>application-list</td>
<td>N</td>
<td>the list ID of the application list.</td>
</tr>
<tr>
<td>metabase-id</td>
<td>N</td>
<td>the ID number of the metabase.</td>
</tr>
<tr>
<td>catalog-entry</td>
<td>C</td>
<td>the catalog entry of the Report Gallery template.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the variable that is selected for the down dimension (optional). (This parameter is included for compatibility with previous releases of this application.)</td>
</tr>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the variable that is selected for the across dimension (optional). (This parameter is included for compatibility with previous releases of this application.)</td>
</tr>
</tbody>
</table>

Example

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);
_BUILD_CURRENT_SUBSETS_ Method

Updates the saved_l sublist on the application list to define the specified filters.

This method

- builds the HIERARCHIES_L sublist on the SAVED_L list if it is empty
- builds the CURRENT_SUBSETS and CURRENT_DRILLS lists on HIERARCHIES_L if it is empty
- updates the CURRENT_SUBSETS lists for each hierarchy and class variable with the current filter information.

Syntax

CALL SEND(OBJID,'_BUILD_CURRENT_SUBSETS_','application-list,metabase-id');

Where... | Is Type... | And Contains...
---|---|---
application-list | N | the list ID of the application list. For more information on application lists, see the online Help for SAS/EIS software.
metabase-id | N | the ID number of the metabase

Example

applist= makelist();
rc=fillist('CATALOG','SASHELP.EISRG.ONEWAY.EIS',applist);
mbid=instance(loadclass('SASHELP.MB.METABASE.CLASS'));
call send(webid,'_BUILD_CURRENT_SUBSETS_','applist,mbid);
BUILD_DOWNL_LIST Method

Builds the DOWNL sublist on the application list

This method

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

Syntax

CALL SEND(OBJID,'_BUILD_DOWNL_LIST_',application-list,down-variable);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>application-list</td>
<td>N</td>
<td>the list ID of the application list. For more information on application lists, see the online Help for SAS/EIS software.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the selected down variable. (This optional parameter is included for compatibility with previous releases of the MDDB Report Viewer.)</td>
</tr>
</tbody>
</table>

Example

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

The following sublist will be added to the application list:

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

Builds the STATSL_ sublist on the application list

This method

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

Syntax

CALL SEND(OBJID,'_BUILD_STATSL_LIST_','application-list');

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>application-list</td>
<td>N</td>
<td>the list ID of the application list. For more information on application lists, see the online Help for SAS/EIS software.</td>
</tr>
</tbody>
</table>

Example

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

The following sublist will be added to the application list:

statsl: ( SUM= 'SUM' )[2445]
**_BUILD_TOTAL_ Method**

Builds the TOTALS sublist on the application list in order to turn report totals on

**Syntax**

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>application-list</td>
<td>N</td>
<td>the list ID of the application list. For more information on application lists, see the online Help for SAS/EIS software.</td>
</tr>
<tr>
<td>metabase-id</td>
<td>N</td>
<td>the ID number of the metabase.</td>
</tr>
<tr>
<td>total-variable</td>
<td>C</td>
<td>the variable that is selected from the down or across dimension.</td>
</tr>
</tbody>
</table>

**Example**

```sas
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 will be added to the application list:

```
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

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

Syntax

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

Where... Is Type... And Contains...

| url     | C  | the broker component of the URL.

Sample output:

```javascript
function geturl(down, across, analysis) {
  D0=0; A0=0; AC0=0; var href=""; var stats="";
  param=new Object;
  param._SERVICE = "default";
  param._PROGRAM = "sashelp.webeis.showrpt.scl";
  param._DEBUG = "2";
  param.MDDB = "SASHELP.PRDMDDB";
  param.METABASE = "SASHELP";
  param.CSS="http%3A%2F%2Flocalhost%2Fcss%2Fmrv.css";
  param.GRT="NONE";
  param.DC="1";
  param.AC="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;
```

```javascript
```

56
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);

Where... Is Type... And Contains...

<table>
<thead>
<tr>
<th>metabase-id</th>
<th>N</th>
<th>the ID number of the metabase</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable-name</td>
<td>C</td>
<td>the name of the variable in the metabase</td>
</tr>
<tr>
<td>in-data-value</td>
<td>C</td>
<td>the unformatted data value</td>
</tr>
<tr>
<td>out-data-value</td>
<td>C</td>
<td>the string to add to the reach-through WHERE clause.</td>
</tr>
</tbody>
</table>

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 result is produced:

fmtval=put(MONTH,$MONTH.)='Jan'
**_CHECK_HIER_MEMBER_ Method**

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

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.

**Syntax**

```call send(objid,'_check_hier_member_',metabase-id,error-flag, 
    hierarchy-var,member-var,message);```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>metabase-id</td>
<td>N</td>
<td>the ID number of the metabase</td>
</tr>
<tr>
<td>error-flag</td>
<td>N</td>
<td>an error flag, where 0 = no error, and 1 = error</td>
</tr>
<tr>
<td>hierarchy-var</td>
<td>C</td>
<td>the hierarchy variable</td>
</tr>
<tr>
<td>member-var</td>
<td>C</td>
<td>the member variable</td>
</tr>
<tr>
<td>message</td>
<td>C</td>
<td>the error message that is to be displayed</td>
</tr>
</tbody>
</table>

**Example**

```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.

This method outputs

- the </FORM> tag
- the link back to the initial HTML page.

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial-url</td>
<td>C</td>
<td>the URL of the initial HTML page</td>
</tr>
<tr>
<td>service-name</td>
<td>C</td>
<td>the broker service value</td>
</tr>
<tr>
<td>metabase-name</td>
<td>C</td>
<td>the metabase name</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>an optional background type (IMAGE or COLOR)</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>an optional background value</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the HTML page title</td>
</tr>
<tr>
<td>webeis-class</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
</tbody>
</table>

Example

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:
</TD></TR>
</FORM>
</TD></TR>
CLOSE_PAGE_ Method

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

Syntax

CALL SEND(OBJID,'_CLOSE_PAGE_');

Example

The following output is produced:

</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

CALL SEND(OBJID,'_CLOSE_STATIC_FORM_');

Example

The following output is produced:

<TR><TD>  </TD></TR></TD>
<TR><TD colspan=2 align=center>  
<INPUT TYPE= "submit" VALUE= "Next"> </TABLE>  
</FORM>  
</BODY>  
</HTML>
_CREATE_STAT_ARRAYS_ Method

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

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

Syntax

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

The following output is produced:

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

//STATS
function stats(select,statbox) {
    var vararrayname="";
    var varstatsstring="";
    var allstatsstring="";
    for (i=0; i < select.options.length; i++) {
        if (select.options[i].selected) {
            vararrayname +=""; //value for array
            varstatsstring +=""; //value for string
            allstatsstring +=""; //value for allstats
        }
    }
}
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 {
    if (num>1) {
        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

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.

**Syntax**

```plaintext
CALL SEND(OBJID, '_DISPLAY_ACROSS_CELLS_', column-list, actions-list,
view-report-flag, analysis-variable, statistic-variable,
across-variable, url, argument-string, argument-string2,
initial-url, background-type, background-value, title,
webeis-class, dlflag, service);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>column-list</td>
<td>N</td>
<td>the column list from the emddb_m class.</td>
</tr>
<tr>
<td>actions-list</td>
<td>N</td>
<td>the actions sublist for drill down.</td>
</tr>
<tr>
<td>view-report-flag</td>
<td>N</td>
<td>the flag for the View Report button.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable to be graphed.</td>
</tr>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic to be graphed.</td>
</tr>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the analysis variable for graphing.</td>
</tr>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>argument-string</td>
<td>C</td>
<td>the argument string for the next query.</td>
</tr>
<tr>
<td>argument-string2</td>
<td>C</td>
<td>the argument string for the next query.</td>
</tr>
<tr>
<td>initial-url</td>
<td>C</td>
<td>the URL of the initial HTML page. (This parameter is obsolete. It is included in the method statement so that overrides are not broken.)</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the optional background type (IMAGE or COLOR).</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the optional background value.</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the HTML page title.</td>
</tr>
<tr>
<td>webeis-class</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
<tr>
<td>dlflag</td>
<td>N</td>
<td>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.</td>
</tr>
<tr>
<td>service</td>
<td>C</td>
<td>the service name.</td>
</tr>
</tbody>
</table>

**Example**

```plaintext
emddbmid= instance(loadclass('SASHELP.WEBEIS.EMDDB_M.CLASS'));
collist= makelist();
call send(emddbmid, '_GET_CLASS_COMBINATIONS_', 'COL', collist);
actions1= makelist();
```
The example produces the following output:

```html
<TR>
<TH CLASS="COLLAB" COLSPAN=1>Month</TH>
<TH CLASS="COLLAB" COLSPAN=4>Jan</TH>
<TH CLASS="COLLAB" COLSPAN=4>Feb</TH>
<TH CLASS="COLLAB" COLSPAN=4>Mar</TH>
<TH CLASS="COLLAB" COLSPAN=4>Apr</TH>
<TH CLASS="COLLAB" COLSPAN=4>May</TH>
<TH CLASS="COLLAB" COLSPAN=4>Jun</TH>
<TH CLASS="COLLAB" COLSPAN=4>Jul</TH>
<TH CLASS="COLLAB" COLSPAN=4>Aug</TH>
<TH CLASS="COLLAB" COLSPAN=4>Sep</TH>
<TH CLASS="COLLAB" COLSPAN=4>Oct</TH>
<TH CLASS="COLLAB" COLSPAN=4>Nov</TH>
<TH CLASS="COLLAB" COLSPAN=4>Dec</TH>
<TH CLASS="TCOLLAB" COLSPAN=4>TOTAL</TH>
</TR>
```
_DISPLAY_ANALYSIS_VARS_ Method

Outputs the chosen analysis variables to the report table

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>column-list</td>
<td>N</td>
<td>the column list from the <em>EMDDB_M</em> class</td>
</tr>
<tr>
<td>dlflag</td>
<td>N</td>
<td>a flag that indicates whether to download the table to a spreadsheet.</td>
</tr>
</tbody>
</table>

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 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">
ACTUAL SALES </TH>
```

<table>
<thead>
<tr>
<th>PREDICTED SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
</tr>
</tbody>
</table>
**_DISPLAY_DEFAULT_TITLE_** Method

Displays the user-specified title

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**.

**Syntax**

```call send(objectid,'_DISPLAY_DEFAULT_TITLE_','dlflag');```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlflag</td>
<td>N</td>
<td>a flag that indicates whether to download the table to a spreadsheet, where 0 = output HTML tags with data values; 1 = output data values with spreadsheet delimiters.</td>
</tr>
</tbody>
</table>

**Example**

```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

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.

**Syntax**

```
CALL SEND(OBJID,'_DISPLAY_DOWNVAR_CELL_','row-list,vrflag,
analysis-variable,statistic-variable,down-variable,
across-variable,_url,_argument-string,_argument-string2,
initial-url,service-name,url,background-type,
background-value,title,webeis-class,dlflag);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>row-list</td>
<td>C</td>
<td>the row list from the EMDDB_M class.</td>
</tr>
<tr>
<td>vrflag</td>
<td>C</td>
<td>a flag that indicates that the View Report button was pressed.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>N</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>statistic-variable</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down dimension variable that is to be graphed.</td>
</tr>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the across dimension variable that is to be graphed.</td>
</tr>
<tr>
<td>_url</td>
<td>C</td>
<td>the browser component of the URL.</td>
</tr>
<tr>
<td>_argument-string</td>
<td>C</td>
<td>the argument string for the next query.</td>
</tr>
<tr>
<td>_argument-string2</td>
<td>C</td>
<td>the argument string for the next query.</td>
</tr>
<tr>
<td>initial-url</td>
<td>C</td>
<td>the URL of the initial HTML page.</td>
</tr>
<tr>
<td>service-name</td>
<td>C</td>
<td>the broker service that is being used.</td>
</tr>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This value is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This value is optional.</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the title. This value is optional.</td>
</tr>
<tr>
<td>web-class</td>
<td>C</td>
<td>the WEBEIS class name (for subclassing).</td>
</tr>
<tr>
<td>dlflag</td>
<td>N</td>
<td>a flag that indicates whether to download the table to a spreadsheet.</td>
</tr>
</tbody>
</table>

**Example**

```
call send(emddbmid_,'_GET_CLASS_COMBINATIONS_','ROW',rowlist);
vflag=1;
grphvar='Actual+Sales';
```
grphstat='Sum';
grphdown='';
grphacr='PRODTYPE';
_url='/cgi-bin/broker?_PROGRAM=sashelp.webeis.mddbrpts.scl&_SERVICE=default
&_DEBUG=0&RPTTYPE=2&GRTYPE=BLOCK'
_args='&MDDB=PERMDATA.MAPINFO&METABASE=PERMDATA.MB612&DOWN=Geographic&ACROSS
=Product+Line&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='SASHELP.WEBEIS.WEBEIS';
dlflag=0;
call send(webid,'_DISPLAY_DOWNVAR_CELL_',rowlist,vrflag,grphvar,
    grphstat,grphdown,grphacr,_url,_args,_args2,mddblink,service,url,
    bgtype,bg,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.webeis.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=SASHELP.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');

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>error-message</td>
<td>C</td>
<td>the error message that is to be displayed.</td>
</tr>
</tbody>
</table>

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

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 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.

Syntax

CALL SEND(OBJID, '_DISPLAY_ONEWAY_', DLFLAG);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlflag</td>
<td>N</td>
<td>a flag that indicates whether to download the table to a spreadsheet.</td>
</tr>
</tbody>
</table>
_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,dsname,gif-device);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down dimension variable that is to be graphed</td>
</tr>
<tr>
<td>dsname</td>
<td>C</td>
<td>the data set name from the <em>WRITE</em> method</td>
</tr>
<tr>
<td>gif-device</td>
<td>C</td>
<td>the device driver name.</td>
</tr>
</tbody>
</table>
**_DISPLAY_ONEWAY_HBAR_ Method**

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

**Syntax**

```plaintext
CALL SEND(OBJID,'_DISPLAY_ONEWAY_HBAR_','statistic,analysis-variable,
down-variable,dsname,gif-device);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down dimension variable that is to be graphed</td>
</tr>
<tr>
<td>dsname</td>
<td>C</td>
<td>the data set name from the <em>WRITE</em> method</td>
</tr>
<tr>
<td>gif-device</td>
<td>C</td>
<td>the device driver name.</td>
</tr>
</tbody>
</table>
**_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,dsname,gif-device);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down dimension variable that is to be graphed</td>
</tr>
<tr>
<td>dsname</td>
<td>C</td>
<td>the data set name from the <em>WRITE</em> method</td>
</tr>
<tr>
<td>gif-device</td>
<td>C</td>
<td>the device driver name.</td>
</tr>
</tbody>
</table>
METHOD

_DLLPLAY_ONEWAY_VBAR_ Method

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

Syntax

CALL SEND(OBJID,'_DLLPLAY_ONEWAY_VBAR_','statistic,analysis-variable,
          down-variable,dsname, gif-device);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down dimension variable that is to be graphed</td>
</tr>
<tr>
<td>dsname</td>
<td>C</td>
<td>the data set name from the <em>WRITE</em> method</td>
</tr>
<tr>
<td>gif-device</td>
<td>C</td>
<td>the device driver name.</td>
</tr>
</tbody>
</table>
_DISPLAY_STATISTIC_VARS_ Method

Outputs the selected statistics to the report table

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.

Syntax

CALL SEND(OBJID,'_DISPLAY_STATISTIC_VARS_',column-list,analysis-variable,_url,_argument-string,_argument-string2,initial-url,URL,service,background-type,background-value,title,webcls,dlflag,rowlist);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>column-list</td>
<td>N</td>
<td>the column list from the EMDDB_M class.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>N</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>_url</td>
<td>C</td>
<td>the URL of the next query.</td>
</tr>
<tr>
<td>_argument-string</td>
<td>C</td>
<td>the argument string for the next query.</td>
</tr>
<tr>
<td>_argument-string2</td>
<td>C</td>
<td>the argument string for the next query.</td>
</tr>
<tr>
<td>initial-url</td>
<td>C</td>
<td>the URL of the initial HTML page.</td>
</tr>
<tr>
<td>URL</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>service</td>
<td>C</td>
<td>the broker service that is being used.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This value is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This value is optional.</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the HTML page title.</td>
</tr>
<tr>
<td>webcls</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
<tr>
<td>dlflag</td>
<td>N</td>
<td>a flag that indicates whether to download the table to a spreadsheet, where 0 = output HTML tags with data values; 1 = output data values with spreadsheet delimiters. This parameter is optional.</td>
</tr>
<tr>
<td>rowlist</td>
<td>N</td>
<td>the rowlist from the <em>GET_CLASS_COMBINATIONS</em> method. This parameter is optional.</td>
</tr>
</tbody>
</table>

Example

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';
mdbdlink='DYNAMIC';
The example produces the following output:

```html
<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);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlflag</td>
<td>N</td>
<td>a flag that indicates whether to download the data to a spreadsheet. This parameter is optional.</td>
</tr>
</tbody>
</table>

Example

The following output is produced:

```
<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**

```plaintext
CALL SEND(OBJID,'_DISPLAY_TITLE_','srchchar,titlemsg,varname,dlflag);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>srchchar</td>
<td>C</td>
<td>the drill string for the down variable (V) or the across variable (VA).</td>
</tr>
<tr>
<td>titlemsg</td>
<td>C</td>
<td>the name of title message, where the name can be CL_DOWN (for down) or IN_ACROSS (for across).</td>
</tr>
<tr>
<td>varname</td>
<td>C</td>
<td>the down or across variable.</td>
</tr>
<tr>
<td>dlflag</td>
<td>N</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Example**

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

The following output is produced:

```plaintext
<TABLE>
<TR><TD><STRONG>Down: Country=CANADA</STRONG><BR></TD></TR>
</TABLE>
```
DISPLAY_TWOWAY Method

Calls the methods to display the two-way report

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 _BUILD_APPLICATION_LIST_.
- calls _CHECK_HIER_MEMBER_.
- calls _SET_DRILL_LEVELS_ to drill to the current level.
- calls emddb_m _SET_APPLICATION_.
- calls _BUILD_ARGS_STRING_.
- calls _BUILD_ARGS2_STRING_.
- 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 emddb_m _GET_CLASS_COMBINATIONS_ to get the row list.
- calls emddb_m _GET_CLASS_COMBINATIONS_ 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 _OPEN_TABLE_.
- calls _OPEN_TWOWAY_.
- calls _DISPLAY_ACROSS_CELLS_.
- calls _OUTPUT_EMPTY_CELL_.
- calls _DISPLAY_ANALYSIS_VARS_.
- calls _DISPLAY_DOWNVAR_CELL_.
- calls _DISPLAY_STATISTIC_VARS_.
- calls _DISPLAY_VALUES_.

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlflag</td>
<td>N</td>
<td>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.</td>
</tr>
</tbody>
</table>
**_DISPLAY_TWOWAY_BLOCK_ Method**

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

**Syntax**

```sas
CALL SEND(OBJID,'_DISPLAY_TWOWAY_BLOCK_','statistic,analysis-variable,
down-variable,across-variable,dsname,gif-device,subset-list);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down variable that is to be graphed.</td>
</tr>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the across variable that is to be graphed.</td>
</tr>
<tr>
<td>dsname</td>
<td>C</td>
<td>the data set name from the <em>WRITE</em> method.</td>
</tr>
<tr>
<td>gif-device</td>
<td>C</td>
<td>the device driver name.</td>
</tr>
<tr>
<td>subset-list</td>
<td>N</td>
<td>the initial subset list. This parameter is optional.</td>
</tr>
</tbody>
</table>
_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, dsname, gif-device, subset-list);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down variable that is to be graphed.</td>
</tr>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the across variable that is to be graphed.</td>
</tr>
<tr>
<td>dsname</td>
<td>C</td>
<td>the data set name from the <em>WRITE</em> method.</td>
</tr>
<tr>
<td>gif-device</td>
<td>C</td>
<td>the device driver name.</td>
</tr>
<tr>
<td>subset-list</td>
<td>N</td>
<td>the subset list that is used for the initial graph. This parameter is optional.</td>
</tr>
</tbody>
</table>
__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__', stat, var, down, across, dsname, gifdev, subset-list);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>stat</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>var</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>down</td>
<td>C</td>
<td>the down variable that is to be graphed.</td>
</tr>
<tr>
<td>across</td>
<td>C</td>
<td>the across variable that is to be graphed.</td>
</tr>
<tr>
<td>dsname</td>
<td>C</td>
<td>the data set name from <strong>WRITE</strong> method.</td>
</tr>
<tr>
<td>gifdev</td>
<td>C</td>
<td>the device driver name.</td>
</tr>
<tr>
<td>subset-list</td>
<td>N</td>
<td>the subset list for the initial graph. This parameter is optional.</td>
</tr>
</tbody>
</table>
**DISPLAY_VALUES Method**

Outputs the numerical values to the report table

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 drilldown
- outputs the class value for the current row
- outputs an <A> tag if drilldown 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.

**Syntax**

CALL SEND(OBJID,'_DISPLAY_VALUES_',row-list, column-list, actions-list, metabase-id, viewreport-flag, _url, _argument-string, _argument-string2, initial-url, analysis-variable, statistic-variable, across-variable, background-type, background-value, title, webcls, dlflag);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>row-list</td>
<td>N</td>
<td>the Row list from EMDDB_M.</td>
</tr>
<tr>
<td>column-list</td>
<td>N</td>
<td>the column list from EMDDB_M.</td>
</tr>
<tr>
<td>actions-list</td>
<td>N</td>
<td>the actions sublist that determines drilldown.</td>
</tr>
<tr>
<td>metabase-id</td>
<td>N</td>
<td>the metabase ID number.</td>
</tr>
<tr>
<td>viewreport-flag</td>
<td>N</td>
<td>the View Report button flag.</td>
</tr>
<tr>
<td>_url</td>
<td>C</td>
<td>the URL of the next query.</td>
</tr>
<tr>
<td>_argument-string</td>
<td>C</td>
<td>the argument string for the next query.</td>
</tr>
<tr>
<td>_argument-string2</td>
<td>C</td>
<td>the argument string for the next query.</td>
</tr>
<tr>
<td>initial-url</td>
<td>C</td>
<td>the URL of the initial HTML page.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>statistic-variable</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>across-variable</td>
<td>N</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This value is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This value is optional.</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the HTML page title.</td>
</tr>
<tr>
<td>webcls</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
</tbody>
</table>
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.

Example

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='';
webcls='SASHELP.WEBEIS.WEBEIS';
dlflag=0;
call send(_self_,'_DISPLAY_VALUES_',rowlist,collist,actionsl,mbid,vrflag,
_url,_args,_args2,mddblink,grphvar,grphstat,grphacr,
bgtype,bg,title,webcls,dlflag);

The following output is produced:

<TR><TH CLASS="rowlab" NOWRAP ROWSPAN=1 COLSPAN=1>
<A href="" onClick="this.href=clsurl('V11=COUNTRY=CANADA&V10=1
&_PROGRAM=SASHELP.WEBEIS.OPFRAME.SCL')" TARGET="_top">CANADA</A>
</TH>
<TR><TH CLASS="rowlab" COLSPAN=1 ROWSPAN=1>
<A HREF="/cgi-bin/broker?_PROGRAM=SASHELP.WEBEIS.OPFRAME.SCL&_SERVICE=default
&_DEBUG=0&GRT=NONE&MDDB=SASHELP.PRDMDDB&METABASE=SASHELP.MBEIS
&D=Geographic&AC=Product%20Line&A=ACTUAL&S=SUM
&EX=1&EX=COUNTRY=CANADA&AC=1&ST=1&GL=1&GSC=1
&SSL=1&SH=3&SW=15&GH=450&GW=600&DP=1&PD=Geographic
&PAC=Product%2BLine&BGTYPE=color&BG=%23FFFFE7" TARGET="_top">"<A HREF="/myimages/images/expand.gif" BORDER=0 ALT="Expand">"</A></A></TH>
<TD CLASS="tcolcell" BGCOLOR="#008000">$97,864</TD>
<TD CLASS="tcolcell" BGCOLOR="#00FFFF">$149,126</TD>
<TD CLASS="tcolcell" BGCOLOR="#0000FF">$246,990</TD>
</TR><TR><TH CLASS="rowlab" NOWRAP ROWSPAN=1 COLSPAN=1>
<A href="" onClick="this.href=clsurl('V11=COUNTRY=GERMANY&V10=1&_PROGRAM=SASHELP.WEBEIS.OPFRAME.SCL')" TARGET="_top">GERMANY</A>
</TH>
<TD CLASS="tcolcell" BGCOLOR="#008000">$97,864</TD>
<TD CLASS="tcolcell" BGCOLOR="#00FFFF">$149,126</TD>
<TD CLASS="tcolcell" BGCOLOR="#0000FF">$246,990</TD>
</TR>
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

U.S.A.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TOTAL

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TARGET="_top">GERMANY</A>

</TH>

<TD CLASS="rowlab" COLSPAN=1 ROWSPAN=1>

<A HREF="/cgi-bin/broker?_PROGRAM=SASHELP.WEBEIS.OPFRAME.SCL&_SERVICE=default &_DEBUG=0&GRT=NONE&MDDB=SASHELP.PRDMDDB&METABASE=SASHELP.MBEIS &D=Geographic&AC=Product%20Line&A=ACTUAL&S=SUM &EX=1&EX=COUNTRY=GERMANY&DC=1&ACB=1&ST=1&GL=1&GSC=1 &SSL=1&SH=3&SW=15&GH=450&GW=600&DP=1&PD=Geographic&PAC=Product%2BLine &BCTYPE=color&BG=%23FFFFE7" TARGET="_top">

<IMG SRC="/myimages/images/expand.gif" BORDER=0 ALT="Expand"></A></TH>

<TD CLASS="tdcell" BGCOLOR=#00FFFF><A href="" onClick="this.href=rturl (''_WHERE=COUNTRY%3DGERMANY&WHERE=PRODTYPE%3DFURNITURE')" TARGET="_blank">$101,194</A></TD>

<TD CLASS="tdcell" BGCOLOR=#00FFFF><A href="" onClick="this.href=rturl (''_WHERE=COUNTRY%3DGERMANY&WHERE=PRODTYPE%3DOFFICE')" TARGET="_blank">$144,804</A></TD>

<TD CLASS="tcolcell" BGCOLOR=#0000FF><A href="" onClick="this.href=rturl (''_WHERE=COUNTRY%3DGERMANY')" TARGET="_blank">$245,998</A></TD>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

U.S.A.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TOTAL

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TARGET="_top">U.S.A.</A>

</TH>

<TH CLASS="rowlab" COLSPAN=1 ROWSPAN=1>

<A HREF="/cgi-bin/broker?_PROGRAM=SASHELP.WEBEIS.OPFRAME.SCL&_SERVICE=default &_DEBUG=0&GRT=NONE&MDDB=SASHELP.PRDMDDB&METABASE=SASHELP.MBEIS&G=Geographic&AC=Product%20Line &A=ACTUAL&S=SUM &EX=1&EX=COUNTRY=U.S.A.&DC=1&ACB=1&ST=1&GL=1&GSC=1 &SSL=1&SH=3&SW=15&GH=450&GW=600&DP=1&PD=Geographic&PAC=Product%2BLine &BCTYPE=color&BG=%23FFFFE7" TARGET="_top">

<IMG SRC="/myimages/images/expand.gif" BORDER=0 ALT="Expand"></A></TH>

<TD CLASS="tdcell" BGCOLOR=#008000><A href="" onClick="this.href=rturl (''_WHERE=COUNTRY%3DU.S.A.&WHERE=PRODTYPE%3DFURNITURE')" TARGET="_blank">$91,567</A></TD>

<TD CLASS="tdcell" BGCOLOR=#00FFFF><A href="" onClick="this.href=rturl (''_WHERE=COUNTRY%3DU.S.A.&WHERE=PRODTYPE%3DOFFICE')" TARGET="_blank">$145,782</A></TD>

<TD CLASS="tcolcell" BGCOLOR=#0000FF><A href="" onClick="this.href=rturl (''_WHERE=COUNTRY%3DU.S.A.')" TARGET="_blank">$237,349</A></TD>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TARGET="_top">U.S.A.</A>

</TH>

<TH CLASS="rowlab" COLSPAN=1 ROWSPAN=1>

<A HREF="/cgi-bin/broker?_PROGRAM=SASHELP.WEBEIS.OPFRAME.SCL&_SERVICE=default &_DEBUG=0&GRT=NONE&MDDB=SASHELP.PRDMDDB&METABASE=SASHELP.MBEIS&D=Geographic&AC=Product%20Line &A=ACTUAL&S=SUM &EX=1&EX=COUNTRY=TOTAL&DC=1&ACB=1&ST=1&GL=1&GSC=1 &SSL=1&SH=3&SW=15&GH=450&GW=600&DP=1&PD=Geographic&PAC=Product%2BLine &BCTYPE=color&BG=%23FFFFE7" TARGET="_top">

<IMG SRC="/myimages/images/expand.gif" BORDER=0 ALT="Expand"></A></TH>

<TD CLASS="tdcell" BGCOLOR=#0000FF><A href="" onClick="this.href=rturl (''_WHERE=&WHERE=PRODTYPE%3DFURNITURE')" TARGET="_blank">$290,625</A></TD>

<TD CLASS="tdcell" BGCOLOR=#0000FF><A href="" onClick="this.href=rturl (''_WHERE=&WHERE=PRODTYPE%3DOFFICE')" TARGET="_blank">$439,712</A></TD>

<TD CLASS="trowcell" BGCOLOR=#0000FF><A href="" onClick="this.href=rturl (''_WHERE=')" TARGET="_blank">$730,337</A></TD>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TARGET="_top">92</A>

</TR></TABLE><BR><BR>

92
_DRILL_TO_LEVEL_ Method

This method has been replaced by the _SET_DRILL_LEVELS_ method. See the _SET_DRILL_LEVELS_ 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',varlist,name);```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>C</td>
<td>the long label for an analysis variable</td>
</tr>
<tr>
<td>varlist</td>
<td>N</td>
<td>the list of analysis variables</td>
</tr>
<tr>
<td>name</td>
<td>C</td>
<td>the analysis variable name.</td>
</tr>
</tbody>
</table>
**_GET_ANALYSIS_VARS_ Method**

Returns the available analysis variables from the metabase and builds the labels list

This method

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

**Syntax**

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>metabase-id</td>
<td>N</td>
<td>the metabase ID number.</td>
</tr>
</tbody>
</table>

**Example**

The following output is produced:

```plaintext
anallbls_= ( 'Predicted Sales'
               'Actual Sales'
           )[563]
```
_GETAVAILABLE_STATS_Method

Gets the available statistics from the metabase

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>metabase-id</td>
<td>N</td>
<td>the metabase ID number.</td>
</tr>
</tbody>
</table>
_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);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>model-name</td>
<td>C</td>
<td>the name of the data model to use.</td>
</tr>
</tbody>
</table>
**_GET_DOWNVAR_LIST_ Method**

Builds the down variable list and the dimensions label list

This method

- calls the metabase _GET_HIERARCHY_ method
- calls the metabase _GET_VARIABLES_ method
- builds the down variable list and the dimensions label list.

**Syntax**

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>metabase-id</td>
<td>N</td>
<td>the metabase ID number.</td>
</tr>
</tbody>
</table>
_GET_EMDDBMID_ Method

Returns the ID of the data model from the EMDDBMID_ instance variable

Syntax

CALL SEND(OBJID, '_GET_EMDDBMID_', id);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>N</td>
<td>the ID of the data model.</td>
</tr>
</tbody>
</table>
_GET_GRAPH_VALUES_ Method

Gets the numeric values for the 3D clickable graph

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.

Syntax

CALL SEND(OBJID, '_GET_GRAPH_VALUES_');

Example

The GRPHVALS_ instance variable contains the following:

( ( COUNTRY='CANADA'  
  _ANLSYS_='Actual Sales'  
  _STATS_='Sum'  
  PRODTYPE='FURNITURE'  
  '97864'  
  'DOLLAR12.'  
) [1073]  
( COUNTRY='CANADA'  
  _ANLSYS_='Actual Sales'  
  _STATS_='Sum'  
  PRODTYPE='OFFICE'  
)  
)
'149126'
'DOLLAR12.'
)[227]
(COUNTRY='CANADA'
_ANLSYS_='Actual Sales'
_STATS_='Sum'
PRODTYPE='TOTAL'
'246990'
'DOLLAR12.'
)[1411]
(COUNTRY='GERMANY'
_ANLSYS_='Actual Sales'
_STATS_='Sum'
PRODTYPE='FURNITURE'
'101194'
'DOLLAR12.'
)[1631]
(COUNTRY='GERMANY'
_ANLSYS_='Actual Sales'
_STATS_='Sum'
PRODTYPE='OFFICE'
'144804'
'DOLLAR12.'
)[1711]
(COUNTRY='GERMANY'
_ANLSYS_='Actual Sales'
_STATS_='Sum'
PRODTYPE='TOTAL'
'245998'
'DOLLAR12.'
)[1715]
(COUNTRY='U.S.A.'
_ANLSYS_='Actual Sales'
_STATS_='Sum'
PRODTYPE='FURNITURE'
'91567'
'DOLLAR12.'
)[1719]
(COUNTRY='U.S.A.'
_ANLSYS_='Actual Sales'
_STATS_='Sum'
PRODTYPE='OFFICE'
'145782'
'DOLLAR12.'
)[1723]
( COUNTRY='U.S.A.'
  _ANLSYS_='Actual Sales'
  _STATS_='Sum'
  PRODTYPE='TOTAL'
  '237349'
  'DOLLAR12.'
)[1727]
( COUNTRY='TOTAL'
  _ANLSYS_='Actual Sales'
  _STATS_='Sum'
  PRODTYPE='FURNITURE'
  '290625'
  'DOLLAR12.'
)[1731]
( COUNTRY='TOTAL'
  _ANLSYS_='Actual Sales'
  _STATS_='Sum'
  PRODTYPE='OFFICE'
  '439712'
  'DOLLAR12.'
)[1735]
( COUNTRY='TOTAL'
  _ANLSYS_='Actual Sales'
  _STATS_='Sum'
  PRODTYPE='TOTAL'
  '730337'
  'DOLLAR12.'
)[1739]
)[1399]
_GET_MDDB_NAME_ Method

Returns the MDDB name from the MDDB_ instance variable

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>mddb</td>
<td>C</td>
<td>the MDDB name.</td>
</tr>
</tbody>
</table>
_GET_MESSAGE_ID_ Method

Returns the ID of the message class from the DMODEL_instance variable

Syntax

CALL SEND(OBJID,'_GET_MESSAGE_ID_',[msgid]);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>msgid</td>
<td>N</td>
<td>the ID of the message object.</td>
</tr>
</tbody>
</table>
_GET_METABASE_NAME_ Method

Returns the metabase name from the METABASE_ instance variable

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>metabase</td>
<td>C</td>
<td>the metabase</td>
</tr>
<tr>
<td></td>
<td></td>
<td>name.</td>
</tr>
</tbody>
</table>
_GET_OUTPUT_FILE_ID_ Method

Returns the output file ID from the HTMLFILE_ instance variable

Syntax

```
CALL SEND(OBJID,'_GET_OUTPUT_FILE_ID_','fileid');
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileid</td>
<td>N</td>
<td>the ID of the output file.</td>
</tr>
</tbody>
</table>
_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,num);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>C</td>
<td>the display color</td>
</tr>
<tr>
<td>range-list</td>
<td>C</td>
<td>the RANGE list</td>
</tr>
<tr>
<td>num</td>
<td>N</td>
<td>the numerical value to search for.</td>
</tr>
</tbody>
</table>
__GET_STATDESC__ Method

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

Syntax

CALL SEND(OBJID,'__GET_STATDESC__',statdesc);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>statdesc</td>
<td>N</td>
<td>the ID of the list that contains statistics descriptions.</td>
</tr>
</tbody>
</table>
**_GET_SUBSET_FLAG_ Method**

Returns the value of the SUBSET_FLAG instance variable

**Syntax**

```plaintext
CALL SEND(OBJID,'_GET_SUBSET_FLAG_',flagval);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>flagval</td>
<td>C</td>
<td>the value of the subset flag.</td>
</tr>
</tbody>
</table>
_GET_USEHOLAP_ Method

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

Syntax

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

<table>
<thead>
<tr>
<th>Where... Is Type... And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>id       N</td>
</tr>
</tbody>
</table>
_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

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

<table>
<thead>
<tr>
<th>Where</th>
<th>Is Type</th>
<th>And Contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>C</td>
<td>the URL of the next query.</td>
</tr>
<tr>
<td>form-name</td>
<td>C</td>
<td>the name of the form. This parameter is optional.</td>
</tr>
<tr>
<td>form-target</td>
<td>C</td>
<td>the target window name. This parameter is optional.</td>
</tr>
</tbody>
</table>

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

Example

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

The following output is produced:

<FORM ACTION="/SCRIPTS/BROKER" NAME="MYFORM" TARGET="MENUFORM">
__OPEN_ONEWAY__ Method

Opens the one-way report table

This method

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

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlflag</td>
<td>N</td>
<td>a flag that indicates whether to download the table to a spreadsheet.</td>
</tr>
</tbody>
</table>

Example

The following output is produced:

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

Opens a file in which static HTML will be written

Syntax

CALL SEND(OBJID,'_OPEN_STATIC_FILE_',indxfile, msgdest, rc);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>indxfile</td>
<td>C</td>
<td>the fileref of the file that is to be opened.</td>
</tr>
<tr>
<td>msgdest</td>
<td>C</td>
<td>the destination for error messages. Valid values are LOG or DIALOG.</td>
</tr>
<tr>
<td>rc</td>
<td>N</td>
<td>the return code for errors (1=error).</td>
</tr>
</tbody>
</table>
**_OPEN_TABLE_ Method**

Outputs the <TABLE> tag for the dynamic HTML pages

**Syntax**

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>brdrvalue</td>
<td>C</td>
<td>an optional parameter that specifies the table border thickness</td>
</tr>
<tr>
<td>table-width</td>
<td>C</td>
<td>an optional parameter that specifies the width of the table cells (as a percentage of the document width)</td>
</tr>
<tr>
<td>border-color-dark</td>
<td>C</td>
<td>an optional parameter that specifies a table cell border color attribute</td>
</tr>
<tr>
<td>border-color-light</td>
<td>C</td>
<td>an optional parameter that specifies a table cell border color attribute</td>
</tr>
<tr>
<td>background-color</td>
<td>C</td>
<td>an optional parameter that specifies the background color of the table</td>
</tr>
<tr>
<td>cell-padding</td>
<td>C</td>
<td>an optional parameter that specifies the spacing that is inside the table cells</td>
</tr>
<tr>
<td>cell-spacing</td>
<td>C</td>
<td>an optional parameter that specifies the spacing between the table cells</td>
</tr>
<tr>
<td>css-class</td>
<td>C</td>
<td>an optional parameter that specifies the label for a cascading style sheet tag</td>
</tr>
</tbody>
</table>

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

**Example**

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

The following output is produced:

```plaintext
<TABLE BORDER=3 WIDTH=50% BORDERCOLOR=RED BORDERCOLOR=LIGHT=RED
       BGCOLOR=GRAY
       CELLPADDING=2 CELLSPACING=2 CLASS="mytable">
_OPEN_TWOWAY_ Method

Opens the two-way report table

This method

- outputs the <TABLE> tag
- calls the emddb_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 drilldown has occurred.

Syntax

CALL SEND(OBJID,'_OPEN_TWOWAY_',column-list,viewreport-flag,_url,_argument-string,_argument-string2,_argument-string3,initial-url,url,service,analysis-variable,statistic-variable,across-variable,background-type,background-value,webeis-class,dlflag);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>column-list</td>
<td>N</td>
<td>the column list from the _emddb_m.</td>
</tr>
<tr>
<td>viewreport-flag</td>
<td>N</td>
<td>the View Report button flag.</td>
</tr>
<tr>
<td>_url</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>_argument-string</td>
<td>C</td>
<td>the argument string for the next query.</td>
</tr>
<tr>
<td>_argument-string2</td>
<td>C</td>
<td>the argument string for the next query.</td>
</tr>
<tr>
<td>_argument-string3</td>
<td>C</td>
<td>the argument string for next query.</td>
</tr>
<tr>
<td>initial-url</td>
<td>C</td>
<td>the URL of the initial HTML page.</td>
</tr>
<tr>
<td>url</td>
<td>C</td>
<td>the URL for the next query.</td>
</tr>
<tr>
<td>service</td>
<td>C</td>
<td>the broker service that is being used.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>statistic-variable</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>N</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This parameter is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This parameter is optional.</td>
</tr>
<tr>
<td>webeis-class</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
<tr>
<td>dlflag</td>
<td>N</td>
<td>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.</td>
</tr>
</tbody>
</table>
_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

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.

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the previously selected across variable. This parameter is optional.</td>
</tr>
</tbody>
</table>

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
  <OPTION VALUE=MONTH>Month
  <OPTION VALUE=PRODTYPE>Product Type
  <OPTION VALUE=PRODUCT>Product
  <OPTION VALUE=QUARTER>Quarter
  <OPTION VALUE=STATE>State/Province
  <OPTION VALUE=YEAR>Year
</SELECT>
_OUTPUT_ADDTL_CLSVAL_PARMS_ Method

Adds additional URL parameters to the JavaScript function

This stub method is called from the _OUTPUT_CLASSVAL_URL_FN_ method.

Syntax

```
CALL SEND(OBJID, "_OUTPUT_ADDTL_CLSVAL_PARMS_");
```
_OUTPUT_ADDTL_RT_PARMS_ Method

Adds additional URL parameters to the reach-through links

This stub method is called from the _OUTPUT_REACHTHRU_URL_FN_ method.

Syntax

CALL SEND(OBJID, '_OUTPUT_ADDTL_RT_PARMS_');
_OUTPUT_ADDTOFAV_FUNCTION_ Method

Outputs the addtofav JavaScript function on the toolbar page

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

Syntax

CALL SEND(OBJID,'_OUTPUT_ADDTOFAV_FUNCTION_');

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);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>service-name</td>
<td>C</td>
<td>the broker service value</td>
</tr>
<tr>
<td>next-program</td>
<td>C</td>
<td>the next SCL program to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>execute.</td>
</tr>
</tbody>
</table>
**_OUTPUT_ANAL_LIST_** Method

**Outputs a label and HTML tags for a selection list**

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.

**Syntax**

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

**Example**

The following output is produced:

```html
<TR><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_',tblflag,selvar);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>tblflag</td>
<td>C</td>
<td>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.</td>
</tr>
<tr>
<td>selvar</td>
<td>C</td>
<td>the analysis variable to mark SELECTED.</td>
</tr>
</tbody>
</table>

**Example**

The following output is produced:

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

Outputs the moveall and movesel JavaScript functions on the Dimensions page

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

Syntax

CALL SEND(OBJID,'_OUTPUT_ARROW_FUNCTIONS_');

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 (from<listbox.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, vier-report-flag);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>bar-shape</td>
<td>C</td>
<td>the currently selected graph bar shape</td>
</tr>
<tr>
<td>view-report-flag</td>
<td>N</td>
<td>the View Report flag.</td>
</tr>
</tbody>
</table>

Example

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

The following output is produced:

<TD CLASS="label">Bar Shape:<br>
<SELECT NAME="BS" CLASS="select"><br>
<OPTION VALUE=Block>Block<br>
<OPTION VALUE=Cylinder>Cylinder<br>
<OPTION SELECTED VALUE=Hexagon>Hexagon<br>
<OPTION VALUE=Prism>Prism<br>
<OPTION VALUE=Star>Star<br>
_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);

Where... Is Type... And Contains...

| vrflag  | N  | the View Report button flag. |
| url     | C  | the broker component of the URL. |
| service-name | C  | the broker service value. |
| analysis-variable | C  | the analysis variable that is to be graphed. |
| statistic | C  | the statistic that is to be graphed. |
| down-variable | C  | the down variable that is to be graphed. |
| graph-type | C  | the graph type (BLOCK, HBAR, PIE, PLOT, VBAR). |
| background-type | C  | the background type (IMAGE or COLOR). This value is optional. |
| background-value | C  | the background value. This value is optional. |
| title | C  | the HTML page title. |
| webeis-class | C  | the WEBEIS class name. |

Example

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_BOOKMARK_URL_',vrflag,url,service,grphvar,grphstat,
grphdown,grphtype,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&A=ACTUAL&A1S1=SUM&BGTYPE=COLOR &BG= YELLOW&GRT=VBAR&DC=1&ACB=1&ST=1&GL=1&GSC=1&SSL=1&SH=3&S W=15&GH=450&GW=600&DP=1"
_OUTPUT_CLASSVAL_URL_FN_ Method

Outputs the CLSVAL JavaScript function on the Report page

This is a stub method.

Syntax

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

Where... | Is Type... | And Contains...
--- | --- | ---
service-name | C | the broker service value.
analysis-variable | C | the analysis variable that is to be graphed.
statistic | C | the statistic that is to be graphed.
across-variable | C | the across variable that is to be graphed.
background-type | C | the background type (IMAGE or COLOR). This value is optional.
webeis-class | C | the WEBEIS class name.
background-value | C | the background value. This value is optional.
title | C | the HTML page title.
url | C | the broker component of the URL.
vrflsg | C | the View Report bottom flag

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

Outputs the <APPLET> tag for the 3D Clickable graph

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

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>service-name</td>
<td>C</td>
<td>the broker service value.</td>
</tr>
<tr>
<td>graph-type</td>
<td>C</td>
<td>the graph type (BLOCK, HBAR, PIE, PLOT, VBAR).</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down variable that is to be graphed.</td>
</tr>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the across variable that is to be graphed.</td>
</tr>
<tr>
<td>webcls</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This value is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This value is optional.</td>
</tr>
<tr>
<td>bar-shape</td>
<td>C</td>
<td>the graph bar shape (Block, Cylinder, Hexagon, Prism, Star).</td>
</tr>
</tbody>
</table>

Example

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

The following output is produced:

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

Outputs the "text/html" content-type header

Syntax

    CALL SEND(OBJID, '_OUTPUT_CONTENT_HEADER_');
_OUTPUT_CSV_CONTENT_HEADER_

Method

Outputs the content-type header for the CSV form

Syntax

```
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 that is used to specify a default title

Syntax

    CALL SEND(OBJID,'_OUTPUT_DEFLT_TITLE_OPTION_');

Example

The following output is produced:

    <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**

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

**Syntax**

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

The following output is produced:

```javascript
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;
                            }
                            href += "&" + elements[i].name + eval(numselected) + "=" + elements[i].options[j].value;
                        }
                    }
                } 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_');

The following output is produced:

<A href="" 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

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.

Syntax

CALL SEND (OBJID, '_OUTPUT_DOWN_LIST_', down-variable, url);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the previously selected down variable. This parameter is optional.</td>
</tr>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL. This parameter is optional.</td>
</tr>
</tbody>
</table>

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 VALUE=Product+Line>Product Line (hier)
<OPTION VALUE=Time>Time (hier)
<OPTION VALUE=COUNTRY>Country
<OPTION VALUE=COUNTY>County
<OPTION VALUE=M NTH>Month
<OPTION VALUE=PRODTYPE>Product Type
<OPTION VALUE=PRODUCT>Product
<OPTION VALUE=QUARTER>Quarter
<OPTION VALUE=STATE>State/Province
<OPTION VALUE=YEAR>Year
</SELECT>
_OUTPUT_DP_TITLE_OPTION_ 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:

<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 detail data page

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset-name</td>
<td>C</td>
<td>the base table data set name</td>
</tr>
<tr>
<td>background</td>
<td>C</td>
<td>the HTML background value</td>
</tr>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL</td>
</tr>
<tr>
<td>service-name</td>
<td>C</td>
<td>the broker service value</td>
</tr>
<tr>
<td>dataset-member</td>
<td>C</td>
<td>the data set name (for example, PRDSALE)</td>
</tr>
<tr>
<td>next-program-library</td>
<td>C</td>
<td>the library for the download to spreadsheet program</td>
</tr>
<tr>
<td>next-program-catalog</td>
<td>C</td>
<td>the catalog for the download to spreadsheet program</td>
</tr>
<tr>
<td>next-program</td>
<td>C</td>
<td>the next SCL program to execute to display additional rows of data</td>
</tr>
<tr>
<td>debug-value</td>
<td>C</td>
<td>the broker debug value</td>
</tr>
<tr>
<td>where-clause</td>
<td>C</td>
<td>the WHERE clause to apply to the data.</td>
</tr>
</tbody>
</table>

Example

dataset='SASHELP.PRDSALE';
bgchar='BGCOLOR= yellow';
url='/cgi-bin/broker';
service='default';
member='PRDSALE';
pgmlib='SASHELP';
pgmcat='WEBEIS';
program='SASHELP.WEBEIS.DS2HTM.SCL';
debug='0';
where='COUNTRY= CANADA';
call send(webid,'_OUTPUT_DS2HTM_HTML_',dataset,bgchar,url,service,member,pgmlib, pgmcat, program,debug,where);
**_OUTPUT_DS2HTM_ST_ Method**

Outputs the DS2HTM statement to generate the detail data table

**Syntax**

```plaintext
CALL SEND (OBJID,'_OUTPUT_DS2HTM_ST_',dataset-name,
            variable-string,startat,number-of-rows,total-rows);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset-name</td>
<td>C</td>
<td>the base table data set name</td>
</tr>
<tr>
<td>variable-string</td>
<td>C</td>
<td>the selected variables to display, separated by spaces</td>
</tr>
<tr>
<td>startat</td>
<td>N</td>
<td>the starting row to display</td>
</tr>
<tr>
<td>number-of-rows</td>
<td>N</td>
<td>the number of rows to display</td>
</tr>
<tr>
<td>total-rows</td>
<td>N</td>
<td>the total number rows of detail data.</td>
</tr>
</tbody>
</table>

**Example**

```plaintext
dataset='SASHELP.PRDSALE';
varchar='COUNTRY ACTUAL PREDICT';
startat=1;
atitime=50;
umrows=480;
call send(webid,'_OUTPUT_DS2HTM_ST_',dataset,varchar,startat,atitime,
umrows);
```
_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);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>metabase</td>
<td>C</td>
<td>the metabase name.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background image URL or color value. This value is optional.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (COLOR or IMAGE). This value is optional.</td>
</tr>
<tr>
<td>service</td>
<td>C</td>
<td>the application server service.</td>
</tr>
<tr>
<td>debug</td>
<td>C</td>
<td>the debug level.</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the HTML page title.</td>
</tr>
<tr>
<td>webeis-class</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
</tbody>
</table>

Example

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

The following output is produced:

<INPUT TYPE="hidden" NAME="metabase" VALUE="SASHELP.MBEIS">
<INPUT TYPE="hidden" NAME="_program" VALUE="sashelp.webeis.mddbrpts.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">

148
_OUTPUT_EMPTY_CELL_ Method

Outputs an empty cell in the HTML table

Syntax

CALL SEND(OBJID, '_OUTPUT_EMPTY_CELL_', spannum, dlflag, cssclass);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>spannum</td>
<td>N</td>
<td>the number of columns to span.</td>
</tr>
<tr>
<td>dlflag</td>
<td>N</td>
<td>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.</td>
</tr>
<tr>
<td>cssclass</td>
<td>C</td>
<td>the class name for the cascading style sheet class tag. This parameter is optional.</td>
</tr>
</tbody>
</table>
_OUTPUT_EMPTY_SERVICE_LIST_ Method

Outputs an empty service list

This method outputs

- the <SELECT> tag
- an example <OPTION> tag with comments that instruct users to edit or add <OPTION> tags for their services.

Syntax

CALL SEND(OBJID,'_OUTPUT_EMPTY_SERVICE_LIST_');
_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

This method outputs

- the Change Graph Type instructions to the HTML
- the Apply submit button to the HTML.

Syntax

```call send(objid, '_OUTPUT_GRAPH_INSTR_');```
_OUTPUT_GRAPH_LIST_ Method

Outputs the list of graph types

This method outputs

- the <SELECT> tag
- an <OPTION> tag for each graph type.

Syntax

CALL SEND(OBJID,'_OUTPUT_GRAPH_LIST_','grphtype','vrflag');

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>grphtype</td>
<td>C</td>
<td>the previously selected graph type</td>
</tr>
<tr>
<td>vrflag</td>
<td>N</td>
<td>the View Report button flag, which takes the following values: 1 = View Report button click on previous action 0 = No View Report button click on previous action.</td>
</tr>
</tbody>
</table>

Example

The following output is produced:

```html
<TR><TD CLASS="label">Type</TD></TR>
<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
```
_OUTPUT_GRAPH_LOC_OPTION_ Method

Outputs a selection list for the Graph Location option

Syntax

    CALL SEND(OBJID,'_OUTPUT_GRAPH_LOC_OPTION_');

Example

The following output is produced:

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

Outputs an option tag for the Graph Type selection list

Syntax

CALL SEND(OBJID,'_OUTPUT_GRAPH_OPTION_','grtype,grmsg,groption);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>grtype</td>
<td>C</td>
<td>the previously selected graph type</td>
</tr>
<tr>
<td>grmsg</td>
<td>C</td>
<td>the mnemonic of the graph type</td>
</tr>
<tr>
<td>groption</td>
<td>C</td>
<td>message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the value for the OPTION tag</td>
</tr>
</tbody>
</table>
_OUTPUT_GRAPH_SOURCEOPTION_ Method

Outputs radio buttons for the Graph Source option

Syntax

CALL SEND(OBJID,'_OUTPUT_GRAPH_SOURCE_OPTION_');

Example

The following output is produced:

<TR><TD CLASS="label">Graph Source</TD><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</TD></TR>
_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_');

The following output is produced:

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

Outputs the opening tags for the ReportLayout page

**Syntax**

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>C</td>
<td>the Broker component of the URL.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (COLOR or IMAGE).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This parameter is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This parameter is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optional.</td>
</tr>
</tbody>
</table>

**Example**

The following output is produced:

```html
<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 CELLPACING=1 BORDER=1>
_OUTPUT_HELP_BUTTON_ Method

Outputs the Help button on the toolbar

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

Syntax

CALL SEND(OBJID,'_OUTPUT_HELP_BUTTON_');

Example

The following output is produced:

<A HREF="http://support.sas.com/rnd/web/intrnet/mddbapp/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

CALL SEND(OBJID,'__OUTPUT_HIDDEN_FIELDS__',across-variable, statistic-variable,analysis-variable,initial-url, service,bgtype,bg,title, webcls);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the across value that is to be graphed.</td>
</tr>
<tr>
<td>statistic-variable</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the variable that is to be graphed.</td>
</tr>
<tr>
<td>initial-url</td>
<td>C</td>
<td>the URL of the initial HTML page.</td>
</tr>
<tr>
<td>service</td>
<td>C</td>
<td>the Broker service.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This parameter is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This parameter is optional.</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the title for the HTML page. This parameter is optional.</td>
</tr>
<tr>
<td>webcls</td>
<td>C</td>
<td>the WEBEIS class name (for subclassing).</td>
</tr>
</tbody>
</table>

Example

The following output is produced:

```
<INPUT TYPE="hidden" NAME="_SERVICE" value="default">
<INPUT TYPE="hidden" NAME="_DEBUG" value="2">
<INPUT TYPE="hidden" NAME="MDDB" value="SASHELP.PRDMDDDB">
<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">
```
_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

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

Syntax

CALL SEND(OBJID, '_OUTPUT_HTML_AFTER_BODY_');
_OUTPUT_HTML_BEF_CLOSE_BODY_

Method

Enables users to add HTML tags to the end of the Report page

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

Syntax

CALL SEND(OBJID, '_OUTPUT_HTML_BEF_CLOSE_BODY_');
_OUTPUT_HTML_FORM_HEADER_ Method

Outputs the opening tags for the static HTML page

This method

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

Syntax

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>C</td>
<td>n optional title for the page.</td>
</tr>
<tr>
<td>cgi</td>
<td>C</td>
<td>the Broker component for the ACTION tag.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background image URL or color value. This parameter is optional.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (COLOR or IMAGE). This parameter is optional.</td>
</tr>
</tbody>
</table>
_OUTPUT_LAYOUT_BUTTON_ Method

Outputs the Layout button on the toolbar to enable users to return to the Variable Selection page

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

Syntax

    CALL SEND(OBJID,'_OUTPUT_LAYOUT_BUTTON_');

Example

The following output is produced:

    <A href="" 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);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>service-name</td>
<td>C</td>
<td>the broker service value.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This parameter is optional.</td>
</tr>
<tr>
<td>graph-type</td>
<td>C</td>
<td>the graph type (BLOCK, HBAR, PIE, PLOT, VBAR).</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This parameter is optional.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down variable that is to be graphed.</td>
</tr>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the across variable that is to be graphed.</td>
</tr>
</tbody>
</table>

Example

```
url='/cgi-bin/broker';
service='default';
graphvar='ACTUAL';
graphstat='SUM';
graphdown='COUNTRY';
graphacr='YEAR';
graphtype='VBAR';
bgtype='COLOR';
bg='YELLOW';
call send(_self_,'_OUTPUT_LAYOUT_FRAME_','url',service,bgtype,graphtype,bg,graphvar,
  graphstat,
  graphdown,graphacr);
```

The following output is produced:

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

169
_OUTPUT_LAYOUT_TOOLBAR_ Method

Outputs the Dimensions and Options buttons on the Layout toolbar page

Syntax

CALL SEND(OBJID, '_OUTPUT_LAYOUT_TOOLBAR_');

The following output is produced:

<TR>
<TD>
<A href="" 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="" 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);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>service-name</td>
<td>C</td>
<td>the broker service value.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This parameter is optional.</td>
</tr>
<tr>
<td>graph-type</td>
<td>C</td>
<td>the graph type (BLOCK, HBAR, PIE, PLOT, VBAR).</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This parameter is optional.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down variable that is to be graphed.</td>
</tr>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the across variable that is to be graphed.</td>
</tr>
</tbody>
</table>

Example

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,grptype,bg, grphvar,grphstat, grphdown,grphacr);

The following output is produced:

<FRAME NAME="header" SRC="/cgi-bin/broker?_program=sashelp.webeis.header.scl
&_service=default&_debug=0&mrvdebug=2&mddb=SASHELP.PRDMDDB
&metabase=SASHELP&D=COUNTY&AC=YEAR&A=ACTUAL
&A1S1=SUM&GRT=VBAR&BGTYPE=COLOR&BG=YELLOW
**_OUTPUT_MDB_LIST_ Method**

Outputs the list of MDDBs

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

**Syntax**

```sql
CALL SEND(OBJID,'_OUTPUT_MDB_LIST_',mddblist,mddb);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>mddblist</td>
<td>N</td>
<td>the list of MDDBs.</td>
</tr>
<tr>
<td>mddb</td>
<td>C</td>
<td>the currently selected MDDB.</td>
</tr>
</tbody>
</table>

This parameter is optional.
**_OUTPUT_NUMROWS_LINKS_ Method**

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

**Syntax**

```sql
CALL SEND(OBJID,'_OUTPUT_NUMROWS_LINKS_');
```

The following output is produced:

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

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

**Syntax**

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

**Example**

The following output is produced:

```javascript
function optbtnurl() {
    with (window.parent.main.document.mf) {
        var limit = elements.length;
        var url = " /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 sellength = elements[i].options.length;
                    var numselected = 0;
                    for (j=0; j<sellength; j++) {
                        if (elements[i].options[j].selected) {
                            numselected++;
                            if (numselected == 1) {
                                url += ";" + elements[i].name + "=" + elements[i].options[j].value;
                            }
                            url += ";" + elements[i].name + eval(numselected) + "=" + elements[i].options[j].value;
                        }
                    }
                    url += "&" + elements[i].name + eval(numselected) + "=" + elements[i].options[j].value;
                }
                if (thisvar == "A") {
                    url += ";" + elements[i].options[j].value + "=";
                    stats=elements[i].options[j].value+"STATS";
                    statsstr="window.parent.main."+stats;
                    statsarray=eval(statsstr);
                    if (statsarray.length==1 && statsarray[0]="nunique") {
                        url += "&A" + j + "S" + "=" + "NUNIQUE";
                    }
                    else if (statsarray.length==1 && statsarray[0]!="nunique") {
                        url += "&A" + j + "S" + "=" + "SUM";
                    }
                    else {
                        var anum=0;
                        for (k=1; k<statsarray.length; k++) {
                            anum=j+1;
                            url += "&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="" 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);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>_url</td>
<td>C</td>
<td>the broker component of the URL</td>
</tr>
<tr>
<td>message-id</td>
<td>N</td>
<td>the ID of the message system</td>
</tr>
<tr>
<td>graph-type</td>
<td>C</td>
<td>the graph type</td>
</tr>
<tr>
<td>bar-shape</td>
<td>C</td>
<td>the graph bar shape.</td>
</tr>
</tbody>
</table>

**Example**

The following output is produced:

```plaintext
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_,mbid,rowlist,rondx,
collist,colndx,curlist);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbid</td>
<td>N</td>
<td>the ID number of the metabase</td>
</tr>
<tr>
<td>rowlist</td>
<td>N</td>
<td>the row list from the EMDDB_M class</td>
</tr>
<tr>
<td>rondx</td>
<td>N</td>
<td>the index of the current row in the rowlist</td>
</tr>
<tr>
<td>collist</td>
<td>N</td>
<td>the column list from the EMDDB_M class</td>
</tr>
<tr>
<td>colndx</td>
<td>N</td>
<td>the index of the current column in the collist</td>
</tr>
<tr>
<td>curlist</td>
<td>N</td>
<td>the list of classes and their associated values.</td>
</tr>
</tbody>
</table>

Example

The following output is produced:

<A href="" onClick="this.href=rturl('_WHERE=COUNTRY%3D%22CANADA%22&_WHERE=PRODTYPE%3D%22FURNITURE%22')" TARGET="_blank">
_OUTPUT_REACHTHRU_URL_FN_ Method

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

Syntax

CALL SEND(OBJID,'_OUTPUT_REACHTHRU_URL_FN_,service,nextpgm,dataset,bgtype,bg,url);

Where... Is Type... And Contains...

| service  | C | the broker service that is being used. |
| nextpgm  | C | the 4-level name of the program to run in order to display the detail data. The default is SASHELP.WEBEIS.DS2HTM.SCL. |
| dataset  | C | the name of the data set that contains the detail data. |
| bgtype   | C | the background type (IMAGE, COLOR, or blank). |
| bg       | C | the background value. |
| url      | C | the broker component of the URL. |

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,bgtype,grptype,bg,grphvar, grphstat,grphdown,grphacr,debug);

Where... Is Type... And Contains...

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL</td>
</tr>
<tr>
<td>service</td>
<td>C</td>
<td>the broker service that is being used</td>
</tr>
<tr>
<td>bgtype</td>
<td>C</td>
<td>the background type (IMAGE, COLOR, or blank)</td>
</tr>
<tr>
<td>grptype</td>
<td>C</td>
<td>the selected graph type</td>
</tr>
<tr>
<td>bg</td>
<td>C</td>
<td>the background value</td>
</tr>
<tr>
<td>grphvar</td>
<td>C</td>
<td>the analysis variable that is to be graphed</td>
</tr>
<tr>
<td>grphstat</td>
<td>C</td>
<td>the statistic that is to be graphed</td>
</tr>
<tr>
<td>grphdown</td>
<td>C</td>
<td>the down dimension variable that is to be graphed</td>
</tr>
<tr>
<td>grphacr</td>
<td>C</td>
<td>the across dimension variable that is to be graphed</td>
</tr>
<tr>
<td>debug</td>
<td>C</td>
<td>the broker debug value.</td>
</tr>
</tbody>
</table>

Example

The following output is produced:

<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=% 23FFFFFF&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_',rpttype);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>rpttype</td>
<td>C</td>
<td>a previously selected report type.</td>
</tr>
</tbody>
</table>
_OUTPUT_ROTATE_BUTTON_ Method

Outputs the Rotate button for the two-dimensional report

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

Syntax

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

The following output is produced:

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>viewreport-flag</td>
<td>N</td>
<td>the View Report button flag.</td>
</tr>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>service</td>
<td>C</td>
<td>the Broker service that is being used.</td>
</tr>
<tr>
<td>initial-url</td>
<td>C</td>
<td>the URL of the initial HTML page.</td>
</tr>
<tr>
<td>across-variable</td>
<td>C</td>
<td>the across variable that is selected.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down variable that is selected.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>statistic-variable</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>down-variable</td>
<td>N</td>
<td>the down variable that is to be graphed</td>
</tr>
<tr>
<td>graph-type</td>
<td>N</td>
<td>the selected graph type.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This parameter is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This parameter is optional.</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the HTML title page.</td>
</tr>
<tr>
<td>webeis-class</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
<tr>
<td>hideflag</td>
<td>C</td>
<td>a hidden variables flag. If hideflag = 1, variables are not output. This parameter is optional.</td>
</tr>
</tbody>
</table>

Example

The following example illustrates the use of this method:

```
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';
```
The following output is produced:

```html
<A href="" 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

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

Where... Is Type... And Contains...

<table>
<thead>
<tr>
<th>vrflag</th>
<th>N</th>
<th>the View Report button flag.</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>service-name</td>
<td>C</td>
<td>the broker service value.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down variable that is to be graphed.</td>
</tr>
<tr>
<td>graph-type</td>
<td>C</td>
<td>the graph type (BLOCK, HBAR, PIE, PLOT, VBAR).</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This parameter is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This parameter is optional.</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the HTML page title.</td>
</tr>
<tr>
<td>webeis-class</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
</tbody>
</table>

Example

This example illustrates the use of the method:

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_ROTATE_URL_',vrflag,url,service,grphvar,grphstat,
grphdown,grptype,bgtype,bg,title,webcls);

The following output is produced:

rotateURL="http://mywebserver/cgi-bin/broker/.csv?_PROGRAM=SASHELP.WEBEIS.OPRPT.SCL &ROTATE=1&_SERVICE=default&_DEBUG=0&MDB=SASHELP.PRDMDDB&METABASE=SASHELP&D=COUNTRY"
_OUTPUT_SETURL_FUNCTION_ Method

Outputs the seturl JavaScript function in the toolbar page

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

Syntax

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

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);
}
```
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';
grptype='VBAR';
bgtype='COLOR';
bg='YELLOW';
title=' ';
webcls=' ';
call send (webid,'_OUTPUT_SPREADSHEET_BUTTON_',vrflag,url,service,grphvar,
grphstat,grphdown,grphtype,bgtype,webcls);
```

The following output is produced:

191
**_OUTPUT_SPREADSHEET_URL_ Method**

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

**Syntax**

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

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>vrflag</td>
<td>N</td>
<td>the View Report button flag.</td>
</tr>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>service-name</td>
<td>C</td>
<td>the broker service value.</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>statistic</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down variable that is to be graphed.</td>
</tr>
<tr>
<td>graph-type</td>
<td>C</td>
<td>the graph type (BLOCK, HBAR, PIE, PLOT, VBAR).</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (IMAGE or COLOR). This parameter is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background value. This parameter is optional.</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the HTML page title.</td>
</tr>
<tr>
<td>webeis-class</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
</tbody>
</table>

**Example**

The following example illustrates the use of this method:

```
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_SPREADSHEET_URL_',vrflag,url,service,grphvar,grphstat,
           grphdown,grphtype/bgtype,bg,title,webcls);
```

The following output is produced:

```
downloadURL="http://mywebserver/cgi-bin/broker/prdmddb.csv?_service=default&_debug=0&_program=sashelp.webeis.oprpt.scl&SPDSHT=X&mddb=SASHELP.PRDMDDB&metabase=SASHELP&D=COUNTRY&AC=YEAR&A=ACTUAL&A1S1=SUM&DC=1&ACB=1&GL=1&GSC=1&SSL=1&SH=3&SW=15"
```
_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,webcls);

Where... | Is Type... And Contains...
--- | ---
url | C the URL for the next query
service | C the broker service that is being used
graph-type | C the selected graph type
analysis-variable | C the analysis variable that is to be graphed
statistic-variable | C the statistic that is to be graphed
down-variable | C the down variable that is to be graphed
across-variable | C the analysis variable that is to be graphed
webcls | C the WEBEIS class name.

Example

The following example illustrates the use of this method:

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

The following output is produced:

<BR><BR><P>
<IMG CLASS="graph" SRC="/cgi-bin/broker?_program=sashelp.webeis.grf2way.scl
&_service=default&mddb=SASHELP.PRDMDDB&metabase=SASHELP.MBEIS&D=Geographic&AC=Product
%20Line
&A=ACTUAL&S=SUM&gtr=VBAR&gv=Actual%20Sales&gs=Sum&gd=COUNTRY&DC=1&ACB=1
&gac=PRODTYPE&GSB=PRODTYPE=TOTAL&SL=%20" ALT="Please wait."
**_OUTPUT_STAT_BOXES_ Method**

Outputs the Select Column and the Available and Selected list boxes for selecting statistics per analysis variable

**Syntax**

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

**Example**

The following output is produced:

```html
<TH ROWSPAN=2 CLASS=laylabel>
Statistics</TH>
<TD CLASS=label>
Select Column
</TD>

<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>

<TD CLASS=label>
Available
</TD>

<SELECT NAME="as" CLASS="sselect" MULTIPLE SIZE="5" ALIGN="left" onChange="change (document.mf.as); ">
</SELECT>

<TD ALIGN=CENTER CLASS=arrows>
<A href="" onClick="moveall(document.mf.as,document.mf.s); remstatanal(document.mf.as); return true">
   <IMG SRC="http://localhost/images/double_right_02g.gif" width="20" height="24" alt="Add all" BORDER=0><BR>
   <A href="" onClick="movesel(document.mf.as,document.mf.s); return true">
      <IMG SRC="http://localhost/images/right_02g.gif" width="20" height="24" alt="Add selected" BORDER=0><BR>
   </A>
   <A href="" onClick="moveall(document.mf.s,document.mf.as); remstatanal(document.mf.
      s);">
      <IMG SRC="http://localhost/images/double_left_02g.gif" width="20" height="24" alt="Remove all" BORDER=0><BR>
   </A>
</A>

<TD CLASS=label>
Selected
</TD>

<select NAME="s" CLASS="sselect" MULTIPLE SIZE="5" align="left" onChange="change
   (document.mf.s); ">
</select>
```
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');

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>metabase</td>
<td>C</td>
<td>the metabase name.</td>
</tr>
<tr>
<td>background-value</td>
<td>C</td>
<td>the background image URL or color value. This parameter is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optional.</td>
</tr>
<tr>
<td>background-type</td>
<td>C</td>
<td>the background type (COLOR or IMAGE). This parameter is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optional.</td>
</tr>
<tr>
<td>webeis-class</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
</tbody>
</table>
_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</TD>
    <DIV CLASS="stats">
        <SELECT NAME="s" CLASS="select" MULTIPLE SIZE=3 onChange="change(document.mf.s)">
            <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>
    </TR>
_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:

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

    <TR><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_SUBSETS_ Method

Outputs the list of character variables for subsetting

Syntax

    CALL SEND(OBJID,'_OUTPUT_SUBSETS_');

Example

The following output is produced:

    <TR><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_SUBSET_SELECTIONS_ Method

Outputs the subset selection lists

Syntax

```call send(objid,'_output_subset_selections_','subloc');
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>subloc</td>
<td>C</td>
<td>the list box</td>
</tr>
</tbody>
</table>

Example

The following output is produced:

```<font size=1>
<div class="filterbox">
<table>
<tr><td colspan=4 class="header">Filter by</td></tr>
<tr>
<td class="label" nowrap>Country:</td></tr>
<tr><td>
<select name="sl" class="select" size=3 multiple>
<option value="." selected>
<option value="country:canada">canada
<option value="country:germany">germany
<option value="country:u.s.a.">u.s.a.
</select></td></tr>
<tr><td class="label" nowrap>Division:</td></tr>
<tr><td>
<select name="sl" class="select" size=3 multiple>
<option value="." selected>
<option value="division:education">education
<option value="division:consumer">consumer
</select></td></tr>
<tr><td class="label" nowrap>Month:</td></tr>
<tr><td>
<select name="sl" class="select" size=3 multiple>
<option value="." selected>
<option value="month:jan">jan
<option value="month:feb">feb
</select></td></tr>
</table>
</div>
```

204
<OPTION VALUE="MONTH:Mar">Mar
<OPTION VALUE="MONTH:Apr">Apr
<OPTION VALUE="MONTH:May">May
<OPTION VALUE="MONTH:Jun">Jun
<OPTION VALUE="MONTH:Jul">Jul
<OPTION VALUE="MONTH:Aug">Aug
<OPTION VALUE="MONTH:Sep">Sep
<OPTION VALUE="MONTH:Oct">Oct
<OPTION VALUE="MONTH:Nov">Nov
<OPTION VALUE="MONTH:Dec">Dec
</SELECT></TD></TR>

<input type="submit" name="appsub" class="submit" value="Apply Filter">
</TD></TR><TABLE>
</DIV>
</FONT>
**_OUTPUT_TABLE_OPTIONS_ Method**

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

**Syntax**

```sql
CALL SEND(OBJID,'_OUTPUT_TABLE_OPTIONS_');
```

The following output is produced:

```html
<TD CLASS="label" COLSPAN="2"><INPUT TYPE="checkbox" NAME="dc" VALUE="1">Row Totals
<INPUT TYPE="checkbox" NAME="acb" VALUE="1">Column Totals</P>
<INPUT NAME="DP" TYPE=CHECKBOX VALUE="1" CHECKED>Drillpaths</TD>
```
_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:

    <TR><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_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,grphvar,grphstat,
    grphdown,grphtype,bgtype,bg,
    title,webcls,tbloc);```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>vrflag</td>
<td>N</td>
<td>a flag indicating that the <strong>View Report</strong> button was pressed.</td>
</tr>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>service</td>
<td>C</td>
<td>the broker service that is being used.</td>
</tr>
<tr>
<td>grphvar</td>
<td>C</td>
<td>the analysis variable that is to be graphed.</td>
</tr>
<tr>
<td>grphstat</td>
<td>C</td>
<td>the statistic that is to be graphed.</td>
</tr>
<tr>
<td>grphdown</td>
<td>C</td>
<td>the down dimension variable that is to be graphed.</td>
</tr>
<tr>
<td>grphtype</td>
<td>C</td>
<td>the selected graph type.</td>
</tr>
<tr>
<td>bgtype</td>
<td>C</td>
<td>the background type (IMAGE, COLOR, or blank).</td>
</tr>
<tr>
<td>bg</td>
<td>C</td>
<td>the background value.</td>
</tr>
<tr>
<td>title</td>
<td>C</td>
<td>the title. This value is optional.</td>
</tr>
<tr>
<td>webcls</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
<tr>
<td>tbloc</td>
<td>C</td>
<td>the toolbar location, where 1=top, 2=bottom, 3=left, 4=right, and 5=none.</td>
</tr>
</tbody>
</table>

**Example**

The following output is produced:

```<tr>
<td>
<A HREF="/cgi-bin/broker/prdmddbcsv?_service=default&_debug=0&_program=sashelp.
webeis.oprtp.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&DP=1&_SAVEAS=prdmddbcsv" TARGET="_self"><img CLASS="imgdown"
SRC="/my_images(btn_xls.gif" ALT="Download to Spreadsheet" BORDER=0</A>
</td>
<td>
<A href="" 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>
</td>
<td>
<A href="" 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>
```

208
_ANGLE_OUTPUT_TOOLBAR_FRAME_ Method

Outputs the <FRAME> tag for the toolbar frame

Syntax

CALL SEND(OBJID,'_OUTPUT_TOOLBAR_FRAME_',url,service,bgtype,grphtype,
   bg,grphvar,grphstat,grphdown,grphacr);

Where... Is Type... And Contains...
url C the broker component of the URL
service C the broker service that is being used
bgtype C the background type (IMAGE, COLOR, or blank)
grphtype C the selected graph type
bg C the background value
grphvar C the analysis variable that is to be graphed
grphstat C the statistic that is to be graphed
grphdown C the down dimension variable that is to be graphed
grphacr C the across dimension variable that is to be graphed.

Example

The following output is produced:

<FRAME NAME="toolbar_window" SRC="/cgi-bin/broker?_program=sashelp.webeis.optbar.scl
&_service=default&_debug=0&mddb=SASHELP.PRDMDDB
&metabase=SASHELP.MBEIS&D=Geographic&AC=Product%2520Line
&A=ACTUAL&S=SUM&GRT=VBAR
&GG=AC&BTYPE=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" SCROLLING="NO"/>
_OUTPUT_TOTALS_OPTIONS_ Method

Outputs check boxes for the Show Totals option for the Down and Across variables

Syntax

CALL SEND(OBJID,'_OUTPUT_TOTALS_OPTIONS_');

Example

The following output is produced:

<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>
_OUTPUT_UPDATE_CLEAR_ Method

Outputs the addstatanal and remstatanal JavaScript functions on the Dimensions page

The addstatanal and remstatanal functions update the list of selected analysis variables as the user makes selections for the report.

Syntax

    CALL SEND(OBJID,'_OUTPUT_UPDATE_CLEAR_');

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_' ,noexp);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>noexp</td>
<td>C</td>
<td>an instruction not to output expand the information. A nonblank means do not output.</td>
</tr>
</tbody>
</table>
_OUTPUT_VAR_FUNCTIONS_ Method

Outputs JavaScript functions for ordering variable selections

Syntax

CALL SEND(OBJID,'_OUTPUT_VAR_FUNCTIONS_');

Example

The following output is produced:

```javascript
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, msgid, vrflag, grphtype);

Where... Is Type... And Contains...

url C the broker component of the URL
msgid N the ID number of the message system
vrflag N a flag indicating that the View Report button was pressed
grphtype C the selected graph type.

Example

The following output is produced:

```html
<FORM ACTION="/cgi-bin/broker" NAME="mf">
  <TR>
    <TD VALIGN=TOP>
      <TABLE>
        <TR>
          <TD CLASS=header>
            Dimensions</TD>
        </TR>
        <TR CLASS="dimselbox">
          <TD CLASS=label>
            Down: <BR>
            <SELECT NAME="d" CLASS="select" SIZE=3 MULTIPLE onChange="change(document.mf.d)"
              SELECTED VALUE=Geographic>Geographic (hier)
              <OPTION VALUE=Product%2520Line>Product Line (hier)
              <OPTION VALUE=Time>Time (hier)
              <OPTION VALUE=COUNTRY>Country
              <OPTION VALUE=DIVISION>Division
              <OPTION VALUE=MONT34H>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>
      </TABLE>
      Across: <BR>
    </TD>
  </TR>
</FORM>
```
<table>
<thead>
<tr>
<th>Analysis</th>
<th>Statistics</th>
<th>Filter Columns:</th>
</tr>
</thead>
</table>
| | <div class="analysis">
| <select name="a" class="select" multiple size=3 onChange="change(document.mf.a)" value="actual">Actual Sales</select> <br>
| <select name="s" class="select" multiple size=3 onChange="change(document.mf.s)" value="sum">Sum</select> <br>
| Filter Columns: | <select name="sv" class="select" multiple size=3>
| <option value="" selected="">"</option> <br>
| <option value="country">Country</option> <br>
| <option value="division">Division</option> <br>
| <option value="month">Month</option> <br>
| <option value="prodtype">Product type</option> <br>
| <option value="product">Product</option> <br>
| <option value="quarter">Quarter</option> <br>
| <option value="region">Region</option> <br>
| <option value="year">Year</option> <br>
| </div> | <div class="stats">
| <select name="s" class="select" multiple size=3 onChange="change(document.mf.s)" value="sum">Sum</select> <br>
| <select name="s" class="select" multiple size=3 onChange="change(document.mf.s)" value="pctsum">% of Sum</select> <br>
| <select name="s" class="select" multiple size=3 onChange="change(document.mf.s)" value="avg">Average</select> <br>
| <select name="s" class="select" multiple size=3 onChange="change(document.mf.s)" value="n">Total Count</select> <br>
| <select name="s" class="select" multiple size=3 onChange="change(document.mf.s)" value="pctn">% of Total #</select> <br>
| <select name="s" class="select" multiple size=3 onChange="change(document.mf.s)" value="min">Minimum</select> <br>
| <select name="s" class="select" multiple size=3 onChange="change(document.mf.s)" value="max">Maximum</select> <br>
| <select name="s" class="select" multiple size=3 onChange="change(document.mf.s)" value="range">Range</select> <br>
| </div> |
<TD ALIGN=CENTER VALIGN=TOP>
<TABLE>
<tr>
<td COLSPAN=2 CLASS=header>
Table</td>
</tr>
<tr><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>
<tr><td CLASS="label">Default Title</td>
<td><input NAME="DT" CLASS="select" TYPE=TEXT SIZE=30 MAXLENGTH=200></td>
</tr>
<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">No</td>
</tr>
<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>
<tr><td COLSPAN=2 CLASS=header>
Graph</td>
</tr>
<tr><td CLASS="label">Graph Source</td>
<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</td>
</tr>
<tr><td CLASS="label">Location</td>
<td><select NAME="gl" CLASS="select">
<option VALUE="1" SELECTED>Bottom
<option VALUE="2">Top
<option VALUE="3">Left
<option VALUE="4">Right
</select></td>
</tr>
<tr><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>
</tr>
<tr><td CLASS="label">Width</td><td><input TYPE=text NAME="gw" CLASS="select" SIZE=4 MAXLENGTH=4 VALUE="600"></td>
</tr>
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Height</strong></td>
<td>&lt;input type=&quot;text&quot; name=&quot;gh&quot; class=&quot;select&quot; size=&quot;4&quot; maxlength=&quot;4&quot; value=&quot;450&quot;&gt;</td>
</tr>
</tbody>
</table>
| **Location** | <select name="ssl" class="select">  
  <option value="1" selected>Right</option>  
  <option value="2">Left</option>  
  <option value="3">Top</option>  
  <option value="4">Bottom</option>  
</select> |
| **Width** | <input type="text" name="sw" class="select" size="3" maxlength="3" value="15"> |
| **Height** | <input type="text" name="sh" class="select" size="3" maxlength="3" value="3"> |
OUTPUT_VARLIST_FORM Method

Outputs the HTML for the reach-through to detail variable selection page

Syntax

CALL SEND(OBJID,'_OUTPUT_VARLIST_FORM_','dataset-name',url,
        htmlfile-id,message-id,dataset-id,service-name,
        debug-value,next-program,background-type,
        background-value);

Where... Is Type... And Contains...

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset-name</td>
<td>N</td>
<td>the base table data set name.</td>
</tr>
<tr>
<td>url</td>
<td>N</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>htmlfile-id</td>
<td>N</td>
<td>the ID for the _webout file.</td>
</tr>
<tr>
<td>message-id</td>
<td>N</td>
<td>the ID of the message system.</td>
</tr>
<tr>
<td>dataset-id</td>
<td>N</td>
<td>the ID for the base table data set.</td>
</tr>
<tr>
<td>service-name</td>
<td>N</td>
<td>the broker service value.</td>
</tr>
<tr>
<td>debug-value</td>
<td>N</td>
<td>the application server debug level.</td>
</tr>
<tr>
<td>next-program</td>
<td>N</td>
<td>the next SCL program to execute when the form is completed.</td>
</tr>
<tr>
<td>background-type</td>
<td>N</td>
<td>the background type (IMAGE or COLOR). This parameter is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>N</td>
<td>the background value. This parameter is optional.</td>
</tr>
</tbody>
</table>

Example

The following output is produced:

dataset='SASHELP.PRDSALE';
url='/cgi-bin/broker';
htmlfile=fopen('_WEBOUT','A');
msgid=instance(loadclass('sashelp.fsp.astmsg.class'),1);
dsidi=open(dataset);
service='default';
ddebug='0';
nextpgm='SASHELP.WEBEIS.DS2HTM.SCL';
bgtype='COLOR';
bg='yellow';
call send(webid,'_OUTPUT_VARLIST_FORM_','dataset',url,htmlfile,msgid,dsid,service,
        debug,nextpgm,bgtype,bg);
_OUTPUT_VARLIST_FUNCTIONS_ Method

Outputs the var_order, resetfields, and pickall JavaScript functions on the reach-through variable selection page

Syntax

CALL SEND(OBJID,'_OUTPUT_VARLIST_FUNCTIONS_ ',dataset-id,htmlfile-id);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset-id</td>
<td>N</td>
<td>the base table data set identifier</td>
</tr>
<tr>
<td>htmlfile-id</td>
<td>N</td>
<td>the identifier for the _webout file.</td>
</tr>
</tbody>
</table>

Example

```javascript
htmlfile=fopen('_WEBOUT','A');
dsid=open('SASHELP.PRDSALE');
call send(webid,'_OUTPUT_VARLIST_FUNCTIONS_ ',dsid,htmlfile);

The following output is produced:

labels = new Array("placeholder","Actual Sales","Predicted Sales","Country","Region","Division","Product type","Product","Quarter","Year","Month");
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 detail variable selection page

**Syntax**

```
CALL SEND(OBJID,'_OUTPUT_VARLIST_HTML_',
dataset-id,htmlfile-id,
message-id,dataset-name,url,service-name,
debug-value,next-program,background-type,
background-value);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset-id</td>
<td>N</td>
<td>the ID for the base table data set.</td>
</tr>
<tr>
<td>htmlfile-id</td>
<td>N</td>
<td>the ID for the _webout file.</td>
</tr>
<tr>
<td>message-id</td>
<td>N</td>
<td>the ID of the message system.</td>
</tr>
<tr>
<td>dataset-name</td>
<td>N</td>
<td>the base table data set name</td>
</tr>
<tr>
<td>url</td>
<td>N</td>
<td>the broker component of the URL.</td>
</tr>
<tr>
<td>service-name</td>
<td>N</td>
<td>the broker service value.</td>
</tr>
<tr>
<td>debug-value</td>
<td>N</td>
<td>the application server debug level.</td>
</tr>
<tr>
<td>next-program</td>
<td>N</td>
<td>the next SCL program to execute when the form is completed.</td>
</tr>
<tr>
<td>background-type</td>
<td>N</td>
<td>the background type (IMAGE or COLOR). This parameter is optional.</td>
</tr>
<tr>
<td>background-value</td>
<td>N</td>
<td>the background value. This parameter is optional.</td>
</tr>
</tbody>
</table>

**Example**

```plaintext
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 submit 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_');

The following output is produced:

    <A href="" 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>

225
_POST_DISPLAY_OPTIONS_ Method

Specifies additional options on the Layout page

This stub method is called after all of the display options are called. The method is useful for adding additional options to the Layout page.

Syntax

```
CALL SEND(OBJID,'_POST_DISPLAY_OPTIONS_','<parmlist>);
```

Where... Is Type... And Contains...

| parmlist  | N | an optional list for passing in information to the method. |
_PRE_DISPLAY_OPTIONS_ Method

Specifies additional options on the Layout page

This stub method is called before any of the display options are called. The method is useful for adding additional options to the Layout page.

Syntax

```sql
CALL SEND(OBJID,'_PRE_DISPLAY_OPTIONS_','<parmlist>);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>parmlist</td>
<td>N</td>
<td>an optional list for passing in information to the method.</td>
</tr>
</tbody>
</table>
_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 in order to activate across totals

Syntax

CALL SEND(OBJID,'_SET_ACROSS_TOTAL_FLAG_',
ttlflag);

Where... Is Type... And Contains...

| ttlflag | C     | 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 in order to activate down totals

**Syntax**

```plaintext
CALL SEND(OBJID, '_SET_DOWN_TOTAL_FLAG_',
ttlflag);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ttlflag</td>
<td>C</td>
<td>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.</td>
</tr>
</tbody>
</table>
_SET_DRILL_LEVELS_ Method

Updates the SAVED_L sublist on the application list to set the drilldown values

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 HIERARCHIES_L if it is empty
- updates the CURRENT_DRILLS sublist for each hierarchy with the current drilldown information
- sets the CURRENT_LEVEL value for each hierarchy on HIERARCHIES_L.

Syntax

CALL SEND(OBJID,'_SET_DRILL_LEVELS_','application-list');

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>application-list</td>
<td>N</td>
<td>the list ID of the application list. For more information on application lists, see the online Help for SAS/EIS software.</td>
</tr>
</tbody>
</table>

Example

applist= makelist();
rc=fillist('CATALOG','SASHELP.EISRG.ONEWAY.EIS',applist);
call send(webid,'_SET_DRILL_LEVELS_',applist);
_SET_EMDDBMID_ Method

Sets the EMDDBMID_ instance variable

Syntax

    CALL SEND(OBJID,'_SET_EMDDBMID_',id);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>N</td>
<td>the ID of the data model.</td>
</tr>
</tbody>
</table>
SET_EXPAND_FLAG_ Method

Sets the expflag_ instance variable that indicates whether values can be expanded.

Syntax

CALL SEND(OBJID,'_SET_EXPAND_FLAG_',rowlist, actions1);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>rowlist</td>
<td>N</td>
<td>the rowlist from the GET_CLASS_COMBINATIONS method</td>
</tr>
<tr>
<td>actions1</td>
<td>N</td>
<td>the actions1 list from the data model.</td>
</tr>
</tbody>
</table>
_SET_HIERL_LIST_ Method

Sets the hierl_instance variable

Syntax

CALL SEND(OBJID,'_SET_HIERL_LIST_','listid');

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>listid</td>
<td>N</td>
<td>the list ID of the target list to be copied.</td>
</tr>
</tbody>
</table>
_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_',flagval);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>flagval</td>
<td>C</td>
<td>the value of the subset flag.</td>
</tr>
</tbody>
</table>
_SET_SUBSETS_LIST_ Method

Defines the subsets to be used

This method sets and fills the subvars_ instance variable and adds the subvars_ list to the data model _self_ list for applying the filters.

Syntax

```
CALL SEND(OBJID, '_SET_SUBSETS_LIST_', varnum);
```

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>varnum</td>
<td>N</td>
<td>the number of selected subset values.</td>
</tr>
</tbody>
</table>
SHOW_GRAPH Method

Sets the graphing variables and calls a graphing method

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.

Syntax

CALL SEND(OBJID,'_DISPLAY_GRAPH_',url,service,_argument-string,
_argument-string2,graph-type,analysis-variable,
statistic-variable,down-variable,across-variable,
webcls);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>C</td>
<td>the broker component of the URL</td>
</tr>
<tr>
<td>service</td>
<td>C</td>
<td>the broker service that is being used</td>
</tr>
<tr>
<td>_argument-string</td>
<td>C</td>
<td>the argument string for the next query</td>
</tr>
<tr>
<td>_argument-string2</td>
<td>C</td>
<td>the argument string for the next query</td>
</tr>
<tr>
<td>graph-type</td>
<td>C</td>
<td>the selected graph type</td>
</tr>
<tr>
<td>analysis-variable</td>
<td>C</td>
<td>the analysis variable that is to be graphed</td>
</tr>
<tr>
<td>statistic-variable</td>
<td>C</td>
<td>the statistic that is to be graphed</td>
</tr>
<tr>
<td>down-variable</td>
<td>C</td>
<td>the down variable that is to be graphed</td>
</tr>
<tr>
<td>across-variable</td>
<td>N</td>
<td>the across variable that is to be graphed</td>
</tr>
<tr>
<td>webcls</td>
<td>C</td>
<td>the WEBEIS class name.</td>
</tr>
</tbody>
</table>
SUBMIT GOPTIONS Method

Submits the SAS/GRAPH GOPTIONS statement for the standard GIF graph

Syntax

CALL SEND(OBJID,'_SUBMIT.GOPTIONS_',gifdev);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>gifdev</td>
<td>C</td>
<td>the name of the device driver to be used.</td>
</tr>
</tbody>
</table>
_SUBMIT_GRAPH_PATTERN_ Method

Submits the SAS/GRAPH PATTERN statements for the standard GIF graphs

Syntax

```sas
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_', stat, var);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is Type...</th>
<th>And Contains...</th>
</tr>
</thead>
<tbody>
<tr>
<td>stat</td>
<td>C</td>
<td>the statistic used in the graph</td>
</tr>
<tr>
<td>var</td>
<td>C</td>
<td>the analysis variable used in the graph.</td>
</tr>
</tbody>
</table>
UPDATE_STATS_LIST Method

Outputs the updatestatslist JavaScript function on the Dimensions page

The updatestatslist function modifies the list of available and selected statistics as the user makes statistic selections for the Report display.

Syntax

CALL SEND(OBJID,'_UPDATE_STATS_LIST_');

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(analysisarray[0])==-1 ) {
        analysistype=analysisarray[0] +"," +analysistype;
      }
    }
  }
  if (analysistype.substr(eval(analysistype.lastIndexOf","+1), 1)=="")) {
    analysistype=analysistype.slice(0,eval(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++) {
      if ( i==0 ) {
        arrayname = eval(arrayoftypes[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]);
      } else {
        for (j=0; j < arrayname.length; j++) {
          document.mf.as.options[j] = new Option(statslabellist[arrayname[j]],
          arrayname[j]);
        }
      }
    }
  }
}
```
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++;
        }
    }
}
MDDB Report Viewer Variables

The MDDB Report Viewer uses macro variables that are set by users and passed into the viewer when the application executes. Table 1 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 2 lists and describes these variables. For more information on the REQUEST INIT program, see PROC APPSRV, REQUEST Statement syntax.

Table 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.PRDMDDB)</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.</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>AC</th>
<th>Across</th>
<th>Hierarchies and category variables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>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>
</tbody>
</table>

<p>| A    | Analysis variable | Analysis variable. This macro variable is deprecated. |
| S    | Statistic         | Globally-applied statistic. This macro variable is deprecated. |
| Am   | Analysis variable | Analysis variable, where m designates the particular variable. |
| AmSn | Statistic         | 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. |
| SV   | Filter variables  | Category variables to filter by |
| SL   | Filter variable values | Values to filter by (for example, SL=COUNTRY: CANADA) |
| EX   | Expand values     | For example, EX=COUNTRY=CANADA |
| V    | Down dimension drill-down values | For example, V=YEAR=1995 |
| VA   | Across dimension drill-down values | For example, VA=PRODTYPE=FURNITURE |
| ST   | Display table     | 1=yes, 2=no |
| DT   | Default title     | Max length=200 |
| DP   | Show drill-path in title | 1=yes, 2=no |
| DC   | Show down totals  | 1=yes |
| ACB  | Show across totals | 1=yes |</p>
<table>
<thead>
<tr>
<th></th>
<th>Graph source</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GSC</td>
<td>1=3D Clickable Graph, 2=Standard GIF Graph(SAS/GRAPH)</td>
<td></td>
</tr>
<tr>
<td>GL</td>
<td>Graph location</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1=bottom, 2=top, 3=left, 4=right</td>
<td></td>
</tr>
<tr>
<td>GRT</td>
<td>Graph type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLOCK, HBAR, PIE, PLOT, VBAR</td>
<td></td>
</tr>
<tr>
<td>BS</td>
<td>Graph bar shapes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STAR, HEXAGON, PRISM, CYLINDER</td>
<td></td>
</tr>
<tr>
<td>SPDSHT</td>
<td>Download to Spreadsheet flag</td>
<td></td>
</tr>
<tr>
<td>GW</td>
<td>Graph width</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default=600, max length=4</td>
<td></td>
</tr>
<tr>
<td>GH</td>
<td>Graph height</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default=450, max length=4</td>
<td></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></td>
</tr>
<tr>
<td></td>
<td>Default=15</td>
<td></td>
</tr>
<tr>
<td>SH</td>
<td>Filter list box height</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default=3</td>
<td></td>
</tr>
<tr>
<td>VIEW</td>
<td>View Report button</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value=View Report</td>
<td></td>
</tr>
<tr>
<td>GD</td>
<td>Graph down variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Category variable for graphing</td>
<td></td>
</tr>
<tr>
<td>GA</td>
<td>Graph across variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Across variable for filtering graph values</td>
<td></td>
</tr>
<tr>
<td>GG</td>
<td>Graph group variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Graph group-by variable, second innermost down variable</td>
<td></td>
</tr>
<tr>
<td>GSG</td>
<td>Graph subgroup variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Graph subgroup-by variable, third innermost down variable</td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>Down variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Same as D, needed for Filter FORM</td>
<td></td>
</tr>
<tr>
<td>SAC</td>
<td>Across variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Same as AC, needed for Filter FORM</td>
<td></td>
</tr>
<tr>
<td>CLASS</td>
<td>WEBEIS class name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For subclassing, default is SASHELP.WEBEIS.WEBEIS.CLASS</td>
<td></td>
</tr>
<tr>
<td>CSS</td>
<td>Style sheet URL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Applies to variable selection and report pages</td>
<td></td>
</tr>
<tr>
<td>CSST</td>
<td>Toolbar style sheet URL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Applies to toolbar frame; if not specified, uses CSS value</td>
<td></td>
</tr>
<tr>
<td>BG</td>
<td>Background color or image</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Color name or hex value, or image URL</td>
<td></td>
</tr>
<tr>
<td>BGTYPE</td>
<td>Background type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COLOR, IMAGE</td>
<td></td>
</tr>
</tbody>
</table>

**Table 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: <a href="http://support.sas.com/rnd/web/internet/mddbapp/hinttips.html">http://support.sas.com/rnd/web/internet/mddbapp/hinttips.html</a></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 scrollbar to frame</td>
</tr>
<tr>
<td>_MRVTBSZ</td>
<td>Toolbar size in pixels</td>
<td>A character string of the form (horiz, vert); 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>Symbol</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------</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>
<tr>
<td>_MRVRNDX4</td>
<td>Value for the radio button that represents the fourth number of rows</td>
<td>The default is 100</td>
</tr>
<tr>
<td>_MRVNRLKS</td>
<td>The minimum number of paging lines to display beneath the report table</td>
<td>The default is 5</td>
</tr>
<tr>
<td>_MRNOSORT</td>
<td>Turns off the sorting feature</td>
<td>A nonblank value turns this off; the default is on</td>
</tr>
<tr>
<td>_MRTBLPRM</td>
<td>Sets report &lt;TABLE&gt; tag parameters</td>
<td>For example, &quot;CELLPADDING=4 CELLPACING=2 BORDER=3&quot;</td>
</tr>
</tbody>
</table>
Customizing the MDDB Report Viewer Using Cascading Style Sheets

The MDDB Report Viewer uses cascading style sheet (CSS) properties to enable you to customize the viewer output. You can use cascading style sheets to modify background colors, fonts, the size and location of the HTML elements, and to indicate whether the HTML elements are displayed or not. For more information on style sheet capabilities, consult your favorite HTML reference guide.

HTML elements use the CLASS parameter to surface style sheet properties. Table 1 lists the CLASS definitions that are used by the MDDB Report Viewer. An example style sheet is shipped with the viewer software, 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,

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

or 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 optionally 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 1. 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>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-hand corner</td>
</tr>
<tr>
<td>FILTERBOX</td>
<td>Table containing filter list boxes</td>
</tr>
<tr>
<td><strong>DIMBOX</strong></td>
<td>Table containing dimension selector list boxes</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DIMSELBOX</strong></td>
<td>Table containing dimension selector list boxes (Report Layout page)</td>
</tr>
<tr>
<td><strong>ANALYBOX</strong></td>
<td>List box for selecting analysis variable (Report Layout page)</td>
</tr>
<tr>
<td><strong>ANALYSIS</strong></td>
<td>Class for the &lt;DIV&gt; tag for the analysis variable list box</td>
</tr>
<tr>
<td><strong>STATSBOX</strong></td>
<td>List box for selecting statistic (Report Layout page)</td>
</tr>
<tr>
<td><strong>STATS</strong></td>
<td>Class for the &lt;DIV&gt; tag for the statistics list box</td>
</tr>
<tr>
<td><strong>ANALYCOL</strong></td>
<td>Analysis variable column</td>
</tr>
<tr>
<td><strong>STATSCOL</strong></td>
<td>Statistics column</td>
</tr>
<tr>
<td><strong>GRAPH</strong></td>
<td>Class for the IMG tag for standard GIF graph</td>
</tr>
<tr>
<td><strong>GRAPHAPP</strong></td>
<td>Class for the graph application tag</td>
</tr>
<tr>
<td><strong>TOOLTAB</strong></td>
<td>Class for toolbar</td>
</tr>
<tr>
<td><strong>IMGBKMRK</strong></td>
<td>Class for the IMG tag for Bookmark</td>
</tr>
<tr>
<td><strong>IMGDIM</strong></td>
<td>Class for the IMG tag for Dimensions</td>
</tr>
<tr>
<td><strong>IMGOPT</strong></td>
<td>Class for the IMG tag for Options</td>
</tr>
<tr>
<td><strong>IMGHELP</strong></td>
<td>Class for the IMG tag for Help</td>
</tr>
<tr>
<td><strong>IMGLAY</strong></td>
<td>Class for the IMG tag for Layout</td>
</tr>
<tr>
<td><strong>IMGLOGOUT</strong></td>
<td>Class for the IMG tag for Logout</td>
</tr>
<tr>
<td><strong>IMGROTATE</strong></td>
<td>Class for the IMG tag for Rotate</td>
</tr>
<tr>
<td><strong>HEADER</strong></td>
<td>Report Layout HTML headers</td>
</tr>
<tr>
<td><strong>LABEL</strong></td>
<td>Report Layout HTML labels</td>
</tr>
<tr>
<td><strong>SELECT</strong></td>
<td>Report Layout HTML for SELECT and INPUT tags</td>
</tr>
<tr>
<td><strong>SSELECT</strong></td>
<td>Class for statistics selection list boxes</td>
</tr>
<tr>
<td><strong>SUBMIT</strong></td>
<td>Submit (View Report) button class</td>
</tr>
</tbody>
</table>