



THE  
POWER  
TO KNOW.

# MDDB Report Viewer 9.2

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2008. *MDDDB Report Viewer 9.2*. Cary, NC: SAS Institute Inc.

## **MDDDB Report Viewer 9.2**

Copyright © 2008, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America.

**For a hard-copy book:** No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**For a Web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

**U.S. Government Restricted Rights Notice:** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19, Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, March 2008

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at [support.sas.com/pubs](http://support.sas.com/pubs) or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

- [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\\_ACROSS\\_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](#)

- [DISPLAY\\_ANALYSIS\\_VARS\\_Method](#)
- [DISPLAY\\_DEFAULT\\_TITLE\\_Method](#)
- [DISPLAY\\_DOWNVAR\\_CELL\\_Method](#)
- [DISPLAY\\_ERROR\\_Method](#)
- [DISPLAY\\_ONEWAY\\_Method](#)
- [DISPLAY\\_ONEWAY\\_BLOCK\\_Method](#)
- [DISPLAY\\_ONEWAY\\_HBAR\\_Method](#)
- [DISPLAY\\_ONEWAY\\_PIE\\_Method](#)
- [DISPLAY\\_ONEWAY\\_VBAR\\_Method](#)
- [DISPLAY\\_STATISTIC\\_VARS\\_Method](#)
- [DISPLAY\\_SUBSET\\_TITLE\\_Method](#)
- [DISPLAY\\_TITLE\\_Method](#)
- [DISPLAY\\_TWOWAY\\_Method](#)
- [DISPLAY\\_TWOWAY\\_BLOCK\\_Method](#)
- [DISPLAY\\_TWOWAY\\_HBAR\\_Method](#)
- [DISPLAY\\_TWOWAY\\_VBAR\\_Method](#)
- [DISPLAY\\_VALUES\\_Method](#)
- [DRILL\\_TO\\_LEVEL\\_Method](#)
- [GET\\_ANALYSIS\\_VAR\\_NAME\\_Method](#)
- [GET\\_ANALYSIS\\_VARS\\_Method](#)
- [GET\\_AVAILABLE\\_STATS\\_Method](#)
- [GET\\_DATA\\_MODEL\\_NAME\\_Method](#)
- [GET\\_DOWNVAR\\_LIST\\_Method](#)
- [GET\\_EMDDBMID\\_Method](#)
- [GET\\_GRAPH\\_VALUES\\_Method](#)
- [GET\\_MDDB\\_NAME\\_Method](#)
- [GET\\_MESSAGE\\_ID\\_Method](#)
- [GET\\_METABASE\\_NAME\\_Method](#)
- [GET\\_OUTPUT\\_FILE\\_ID\\_Method](#)
- [GET\\_RANGE\\_COLOR\\_Method](#)
- [GET\\_STATDESC\\_Method](#)
- [GET\\_SUBSET\\_FLAG\\_Method](#)
- [GET\\_USEHOLAP\\_Method](#)
- [OPEN\\_DYNAMIC\\_FILE\\_Method](#)
- [OPEN\\_FORM\\_Method](#)
- [OPEN\\_ONEWAY\\_Method](#)
- [OPEN\\_STATIC\\_FILE\\_Method](#)
- [OPEN\\_TABLE\\_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\\_BEFORE\\_CLOSE\\_BODY\\_Method](#)
- [OUTPUT\\_HTML\\_FORM\\_HEADER\\_Method](#)
- [OUTPUT\\_LAYOUT\\_BUTTON\\_Method](#)
- [OUTPUT\\_LAYOUT\\_FRAME\\_Method](#)
- [OUTPUT\\_LAYOUT\\_TOOLBAR\\_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\\_EMDDBMID\\_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 MDDDB 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 MDDDB 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
    &_amp;_service=mysevice&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:

```
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 `bgtype='color'` to control the color of the background or `bgtype='image'` to

control the background pattern displayed in the application reports. Use this option with the `bg` option, described below. A `bgtype` value is optional.

If you specify `bgtype= 'color'`, the `bg` option expects one of the named colors or a hexadecimal value for one of the colors that is supported by your Web browser. If you specify `bgtype= 'image'`, the `bg` option expects the URL of a background image file. You can specify only GIF and JPG image files for the background. If you specify `bgtype` and omit `bg`, or if you do not use either option, the background 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

```
<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:

```
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:

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

# 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:
  - o Windows users: !SASROOT\CORE\SASHELP
  - o UNIX users: !SASROOT/sashelp
  - o 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 MDDDB 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, MDDDB 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 MDDDB 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./p>

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:

```
%let _grfont=myfont;
```

By default, the MDDDB 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:

```
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:

```
%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:

```
%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 CELLSPACING=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.

ANALLBLS\_  
specifies the list of analysis variable long labels.

ANALLIST\_  
specifies the list of analysis variables and computed columns.

ANALVARS\_  
specifies the list of selected analysis variables.

ATOTAL\_  
specifies a flag that indicates whether the across totals are turned on.

CLASS\_  
contains the 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.

EMDDBMID\_  
specifies the identifier of the data model class instance.

EXPFLAG\_  
specifies a flag that indicates whether the expands are displayed.

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

## EXPVALS\_

specifies a list that contains the values of 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.\_DMDDB.HOLAP\_M.CLASS.

## HTMLFILE\_

specifies the identifier of the output file for writing HTML.

## IMGURL\_

contains the URL for the images.

## MDDB\_

specifies the name of the selected mddb.

## METABASE\_

specifies the name of the selected metabase.

## ROTFLAG\_

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

## SESSIONID\_

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

## SHOWTAB\_

specifies a flag that indicates whether to display the table, where 1=Yes, 2=No.

## STATDESC\_

specifies a list of all possible statistics labels.



## STATLIST\_

specifies a list of the available statistics from the metabase.

## STATVARS\_

specifies a list of the selected statistics.

## SUBHT\_

indicates the number of rows to display in the filter list boxes.

## SUBLOC\_

specifies the location of the filter list boxes, where 1 = right, 2 = left, 3 = top, and 4 = bottom.

## SUBSET\_BY\_

specifies the list of selected filter values.

## SUBSET\_FLAG\_

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

## SUBVARS\_

specifies the list of selected filter variables.

## SUBWID\_

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

## TBLOC\_

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

## THISSESSION\_

specifies the value for the `_THISSESSION` variable for the application server session.

## USEHOLAP\_

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

## VMDOFF\_

specifies a flag that indicates whether metadata verification checking is to be done on the data model, where any nonblank character = turn off metadata checking, and a blank = perform metadata checking.

---

## Flow of Control in the MDDB Report Viewer Class

The following figures illustrate the flow of control in the MDDB Report Viewer WEBEIS class. For more information on the methods listed in these figures, refer to the individual method descriptions.

- [Figure 1. Flow of Control for the Layout Page \(MDDBRPTS.SCL\)](#)

- [Figure 2. Flow of Control for the Dimensions Page \(LAYOUT.SCL\)](#)
  - [Figure 3. Flow of Control for the Layout Toolbar \(HEADER.SCL\)](#)
  - [Figure 4. Flow of Control for the Report Page \(SHOWRPT.SCL\)](#)
  - [Figure 5. Flow of Control for the Report Page \(OPRPT.SCL\) \(Part 1\)](#)
  - [Figure 5. Flow of Control for the Report Page \(OPRPT.SCL\) \(Part 2\)](#)
  - [Figure 5. Flow of Control for the Report Page \(OPRPT.SCL\) \(Part 3\)](#)
  - [Figure 6. Flow of Control for the Report Page Toolbar \(OPTBAR.SCL\) \(Part 1\)](#)
  - [Figure 6. Flow of Control for the Report Page Toolbar \(OPTBAR.SCL\) \(Part 2\)](#)
- 

## 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\\_ACROSS\\_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](#)

- DISPLAY ONEWAY BLOCK
- DISPLAY ONEWAY HBAR
- DISPLAY ONEWAY PIE
- DISPLAY ONEWAY VBAR
- DISPLAY STATISTIC VARS
- DISPLAY SUBSET TITLE
- DISPLAY TITLE
- DISPLAY TWOWAY
- DISPLAY TWOWAY BLOCK
- DISPLAY TWOWAY HBAR
- DISPLAY TWOWAY VBAR
- DISPLAY VALUES
- DRILL TO LEVEL
- GET ANALYSIS VAR NAME
- GET ANALYSIS VARS
- GET AVAILABLE STATS
- GET DATA MODEL NAME
- GET DOWNVAR LIST
- GET EMDDBMID
- GET GRAPH VALUES
- GET MDDB NAME
- GET MESSAGE ID
- GET METABASE NAME
- GET OUTPUT FILE ID
- GET RANGE COLOR
- GET STATDESC
- GET SUBSET FLAG
- GET USEHOLAP
- OPEN DYNAMIC FILE
- OPEN FORM
- OPEN ONEWAY
- OPEN STATIC FILE
- OPEN TABLE
- OPEN TWOWAY
- OPEN WEBOUT FOR SPDSHT
- OUTPUT ACROSS LIST
- OUTPUT ADDTL CLSVAL PARMS
- OUTPUT ADDTL RT PARMS

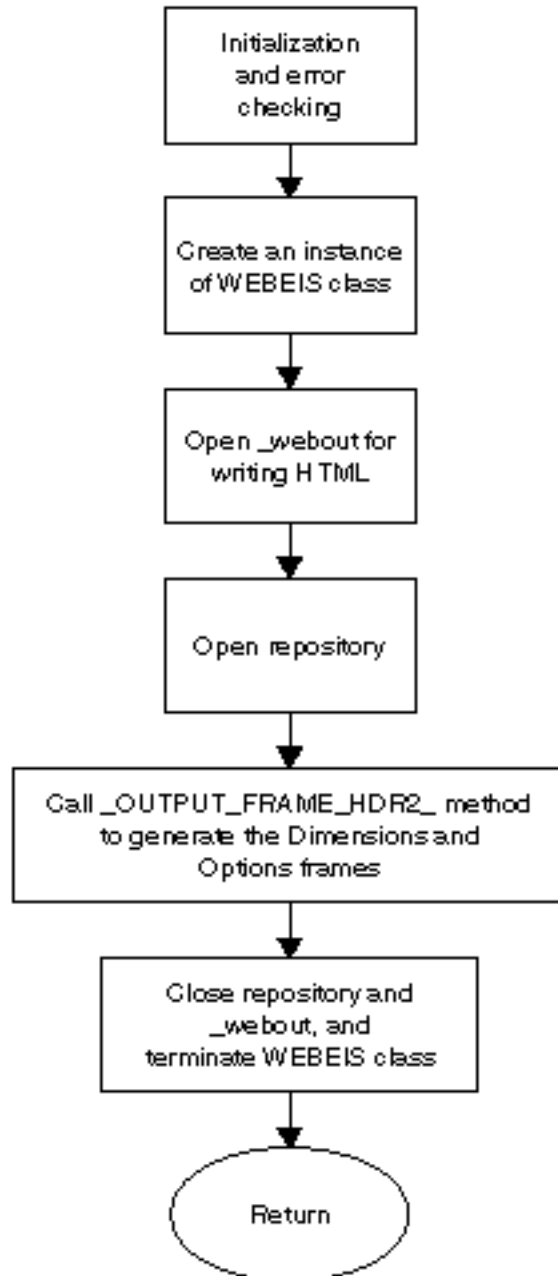
- [OUTPUT\\_ADDTOFAV\\_FUNCTION](#)
- [OUTPUT\\_ALL\\_URL\\_ITEMS](#)
- [OUTPUT\\_ANAL\\_LIST](#)
- [OUTPUT\\_ANAL\\_SELECT](#)
- [OUTPUT\\_ARROW\\_FUNCTIONS](#)
- [OUTPUT\\_BAR\\_SHAPE\\_LIST](#)
- [OUTPUT\\_BOOKMARK\\_BUTTON](#)
- [OUTPUT\\_CLICKABLE\\_GRAPH](#)
- [OUTPUT\\_BOOKMARK\\_URL](#)
- [OUTPUT\\_CLASSVAL\\_URL\\_FN](#)
- [OUTPUT\\_CONTENT\\_HEADER](#)
- [OUTPUT\\_CSV\\_CONTENT\\_HEADER](#)
- [OUTPUT\\_DIMBTN\\_URL\\_FN](#)
- [OUTPUT\\_DIMENSIONS\\_BUTTON](#)
- [OUTPUT\\_DOWN\\_LIST](#)
- [OUTPUT\\_DP\\_TITLE\\_OPTION](#)
- [OUTPUT\\_DS2HTM\\_HTML](#)
- [OUTPUT\\_DS2HTM\\_ST](#)
- [OUTPUT\\_DYNAMIC\\_HIDDEN\\_FLDS](#)
- [OUTPUT\\_EMPTY\\_CELL](#)
- [OUTPUT\\_EMPTY\\_SERVICE\\_LIST](#)
- [OUTPUT\\_GRAPH\\_DIMS\\_OPTION](#)
- [OUTPUT\\_GRAPH\\_INSTR](#)
- [OUTPUT\\_GRAPH\\_LIST](#)
- [OUTPUT\\_GRAPH\\_LOC\\_OPTION](#)
- [OUTPUT\\_GRAPH\\_OPTION](#)
- [OUTPUT\\_GRAPH\\_SOURCE\\_OPTION](#)
- [OUTPUT\\_GRAPH\\_TABLE\\_DISP](#)
- [OUTPUT\\_HDR](#)
- [OUTPUT\\_HELP\\_BUTTON](#)
- [OUTPUT\\_HIDDEN\\_FIELDS](#)
- [OUTPUT\\_HIDDEN\\_VARS](#)
- [OUTPUT\\_HTML\\_AFTER\\_BODY](#)
- [OUTPUT\\_HTML\\_BEFCLOSE\\_BODY](#)
- [OUTPUT\\_HTML\\_FORM\\_HEADER](#)
- [OUTPUT\\_LAYOUT\\_BUTTON](#)
- [OUTPUT\\_LAYOUT\\_FRAME](#)
- [OUTPUT\\_LAYOUT\\_TOOLBAR](#)

- [OUTPUT\\_LOGOUT\\_BUTTON](#)
- [OUTPUT\\_MAIN\\_TOOLBAR\\_FRAME](#)
- [OUTPUT\\_MDDB\\_LIST](#)
- [OUTPUT\\_NUMROWS\\_LINKS](#)
- [OUTPUT\\_NUMROWS\\_OPTION](#)
- [OUTPUT\\_OPTBTN\\_URL\\_FN](#)
- [OUTPUT\\_OPTIONS\\_BUTTON](#)
- [OUTPUT\\_OPTIONS\\_FORM](#)
- [OUTPUT\\_REACHTHRU\\_LINK](#)
- [OUTPUT\\_REACHTHRU\\_URL\\_FN](#)
- [OUTPUT\\_REPORT\\_FRAME](#)
- [OUTPUT\\_REPORT\\_RADIO\\_BTNS](#)
- [OUTPUT\\_REPORT\\_TYPE\\_SELECT](#)
- [OUTPUT\\_ROTATE\\_BUTTON](#)
- [OUTPUT\\_ROTATE\\_URL](#)
- [OUTPUT\\_SETURL\\_FUNCTION](#)
- [OUTPUT\\_SPREADSHEET\\_BUTTON](#)
- [OUTPUT\\_SPREADSHEET\\_URL](#)
- [OUTPUT\\_STANDARD\\_GRAPH](#)
- [OUTPUT\\_STAT\\_BOXES](#)
- [OUTPUT\\_STATIC\\_HIDDEN\\_FLDS](#)
- [OUTPUT\\_STAT\\_LIST](#)
- [OUTPUT\\_SUBSET\\_DIMS\\_OPTION](#)
- [OUTPUT\\_SUBSET\\_LOC\\_OPTION](#)
- [OUTPUT\\_SUBSETS](#)
- [OUTPUT\\_SUBSET\\_SELECTIONS](#)
- [OUTPUT\\_TABLE\\_OPTIONS](#)
- [OUTPUT\\_TABLE\\_DISP\\_OPTION](#)
- [OUTPUT\\_TOOLBAR](#)
- [OUTPUT\\_TOOLBAR\\_FRAME](#)
- [OUTPUT\\_TOTALS\\_OPTIONS](#)
- [OUTPUT\\_UPDATE\\_CLEAR](#)
- [OUTPUT\\_URL\\_OPTIONS](#)
- [OUTPUT\\_VAR\\_FUNCTIONS](#)
- [OUTPUT\\_VARIABLE\\_SEL\\_FORM](#)
- [OUTPUT\\_VARLIST\\_FORM](#)
- [OUTPUT\\_VARLIST\\_FUNCTIONS](#)
- [OUTPUT\\_VARLIST\\_HTML](#)

- 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\_OPTIONS
- SUBMIT\_GRAPH\_PATTERN
- SUBMIT\_GRAPH\_TITLE
- UPDATE\_STATS\_LIST

# Flow of Control for the Layout Page

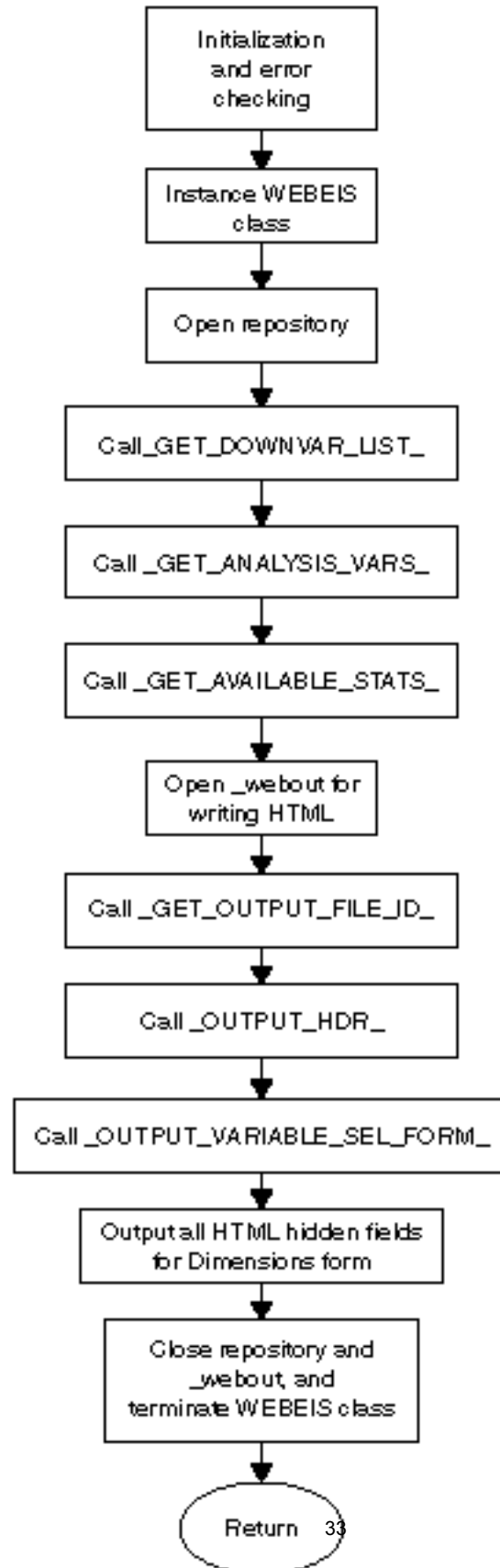
Figure 1. Flow of Control for the Layout Page



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

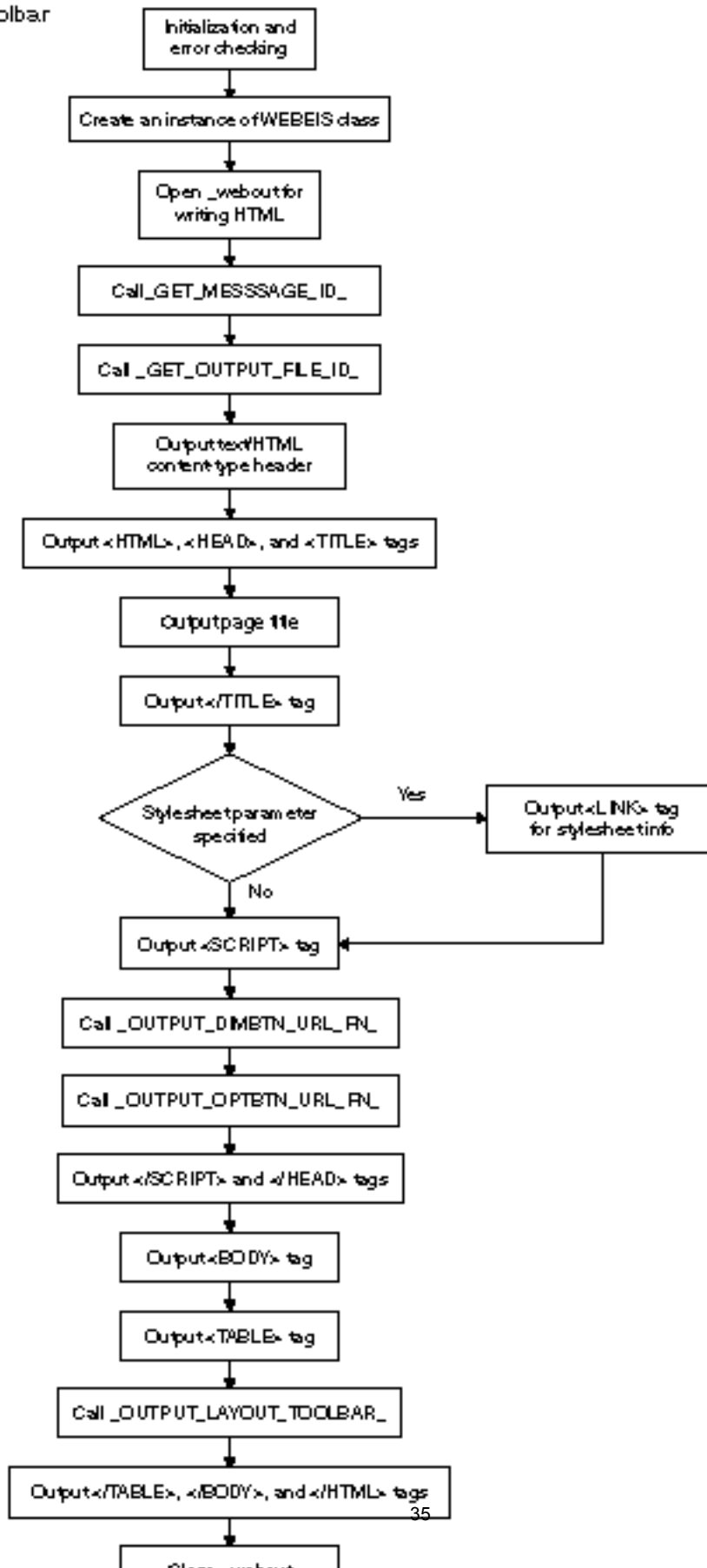


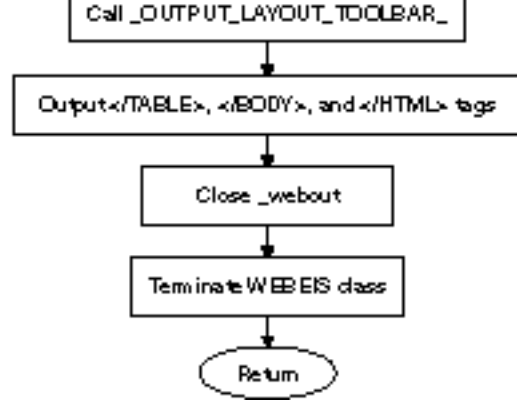




# Flow of Control for the Layout Toolbar

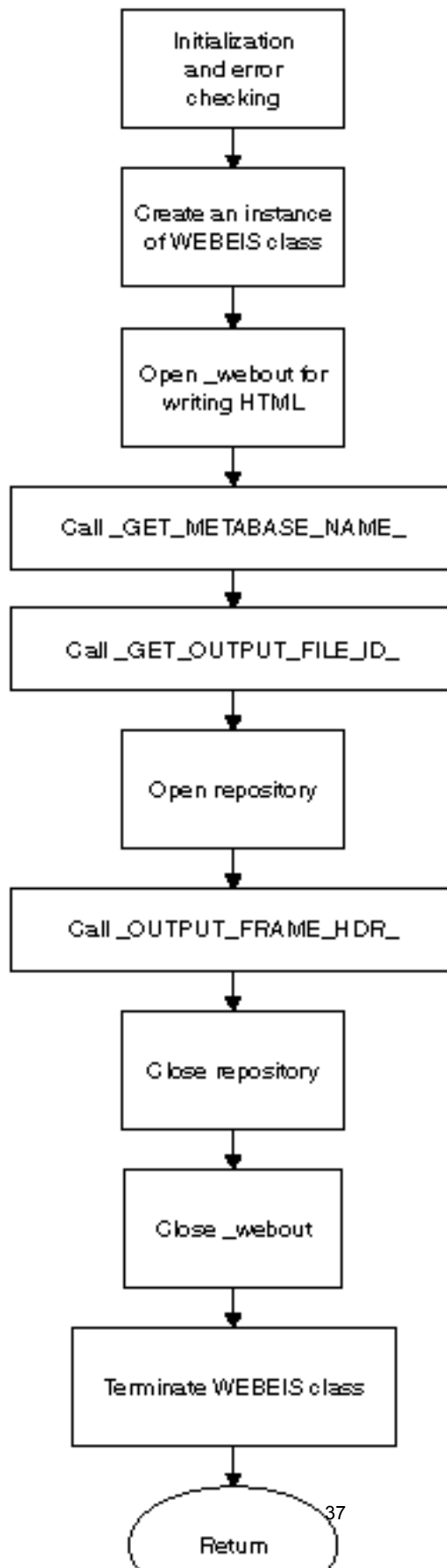
Figure 3. Flow of Control for the Layout Toolbar

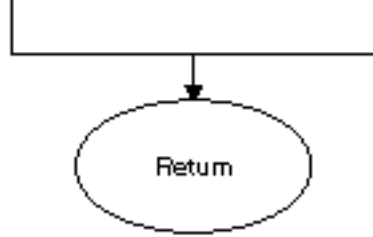




# Flow of Control for the Report Page

Figure 4. Flow of Control for the the Report Page

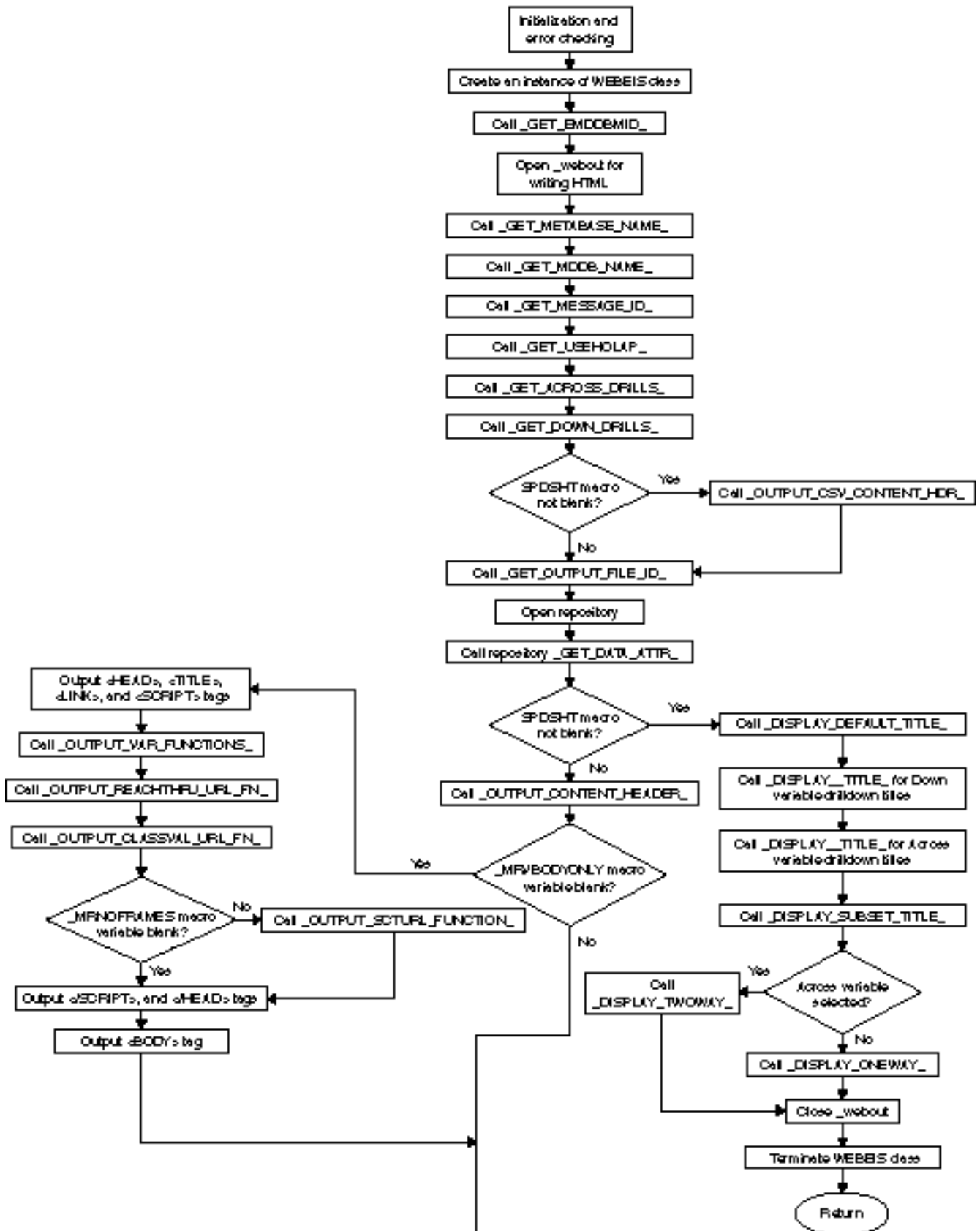




This generates the <FRAMESET> for the Report page, as well as the <FRAME> tags for the toolbar and report frames.

# Flow of Control for the Report Page (Part 1)

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

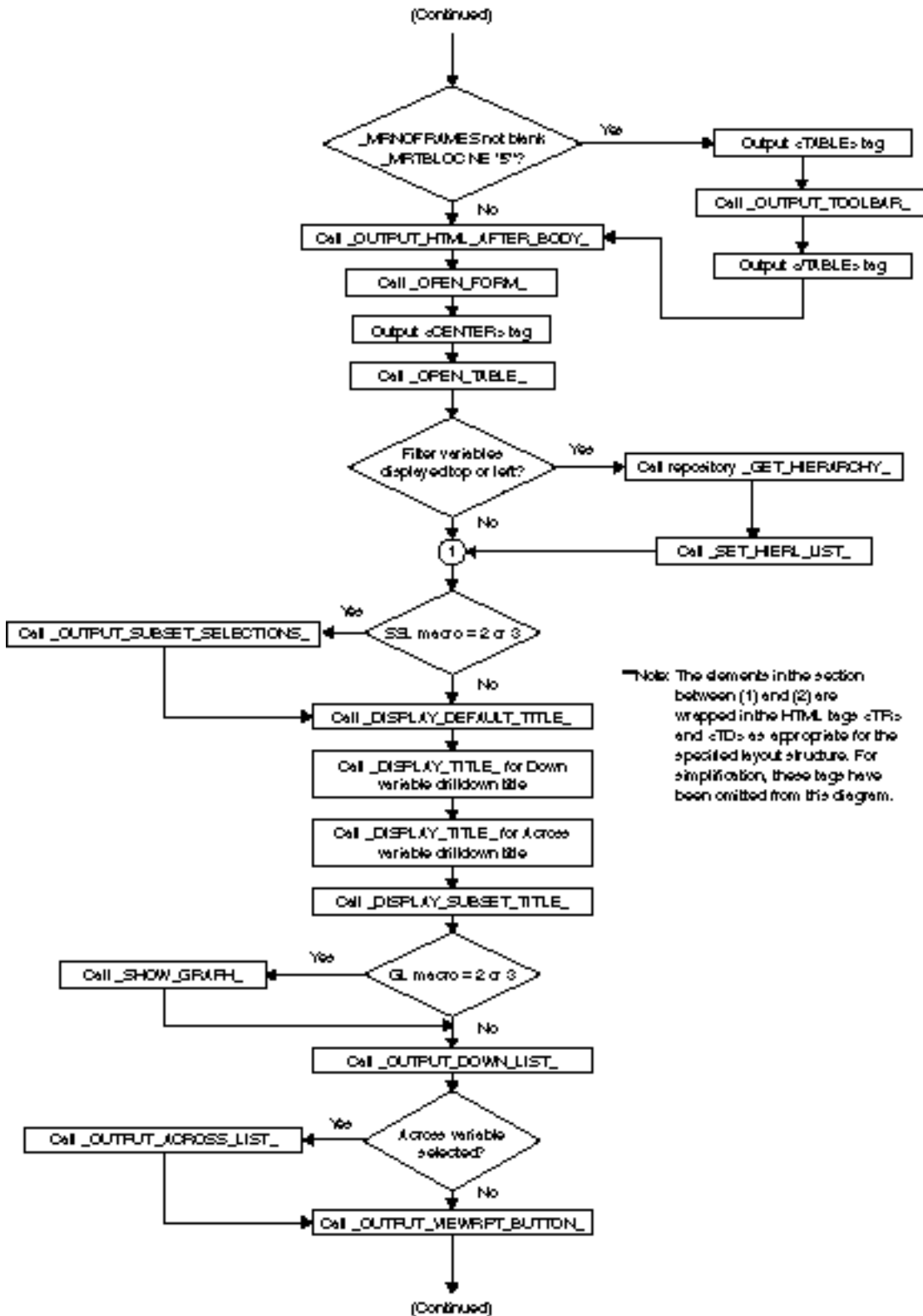


↓  
(Continued)



# Flow of Control for the Report Page (Part 2)

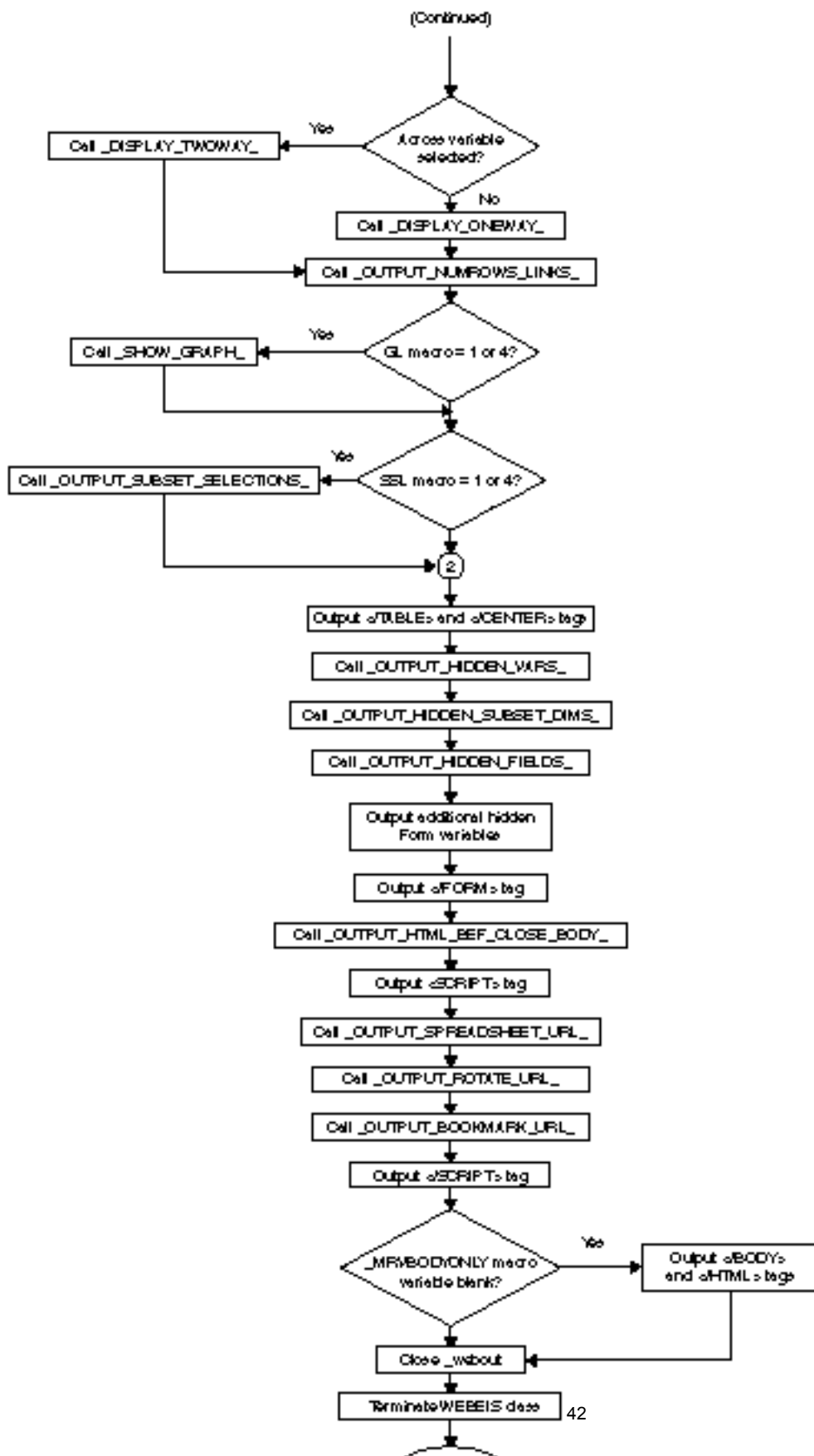
Figure 5. Flow of Control for the the Report Page (part 2)

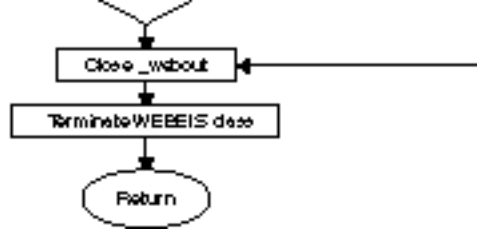




# Flow of Control for the Report Page (Part 3)

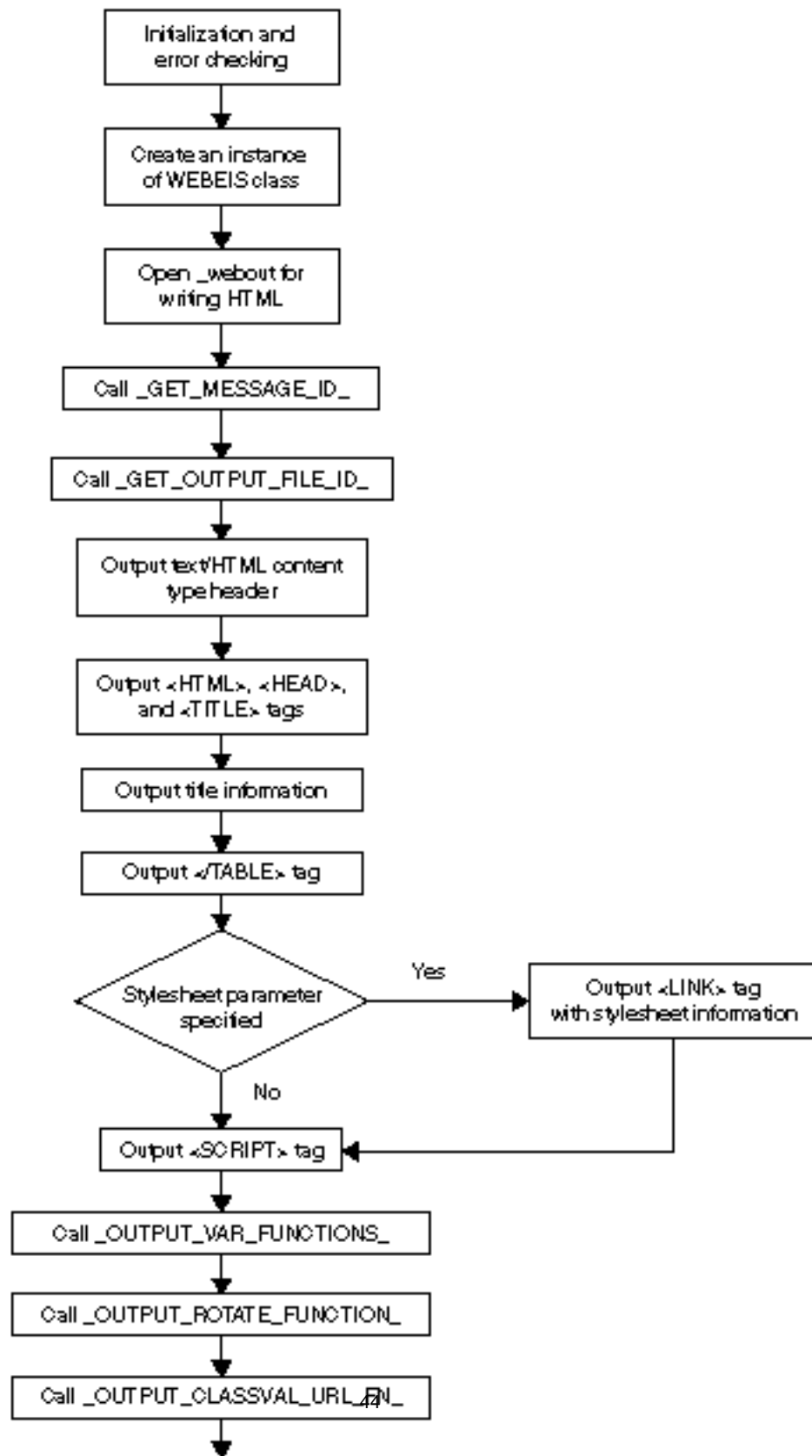
Figure 5. Flow of Control for the the Report Page (part 3)

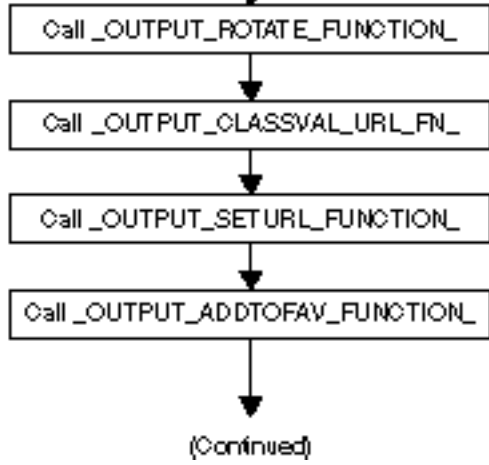




# Flow of Control for the Report Page Toolbar (Part 1)

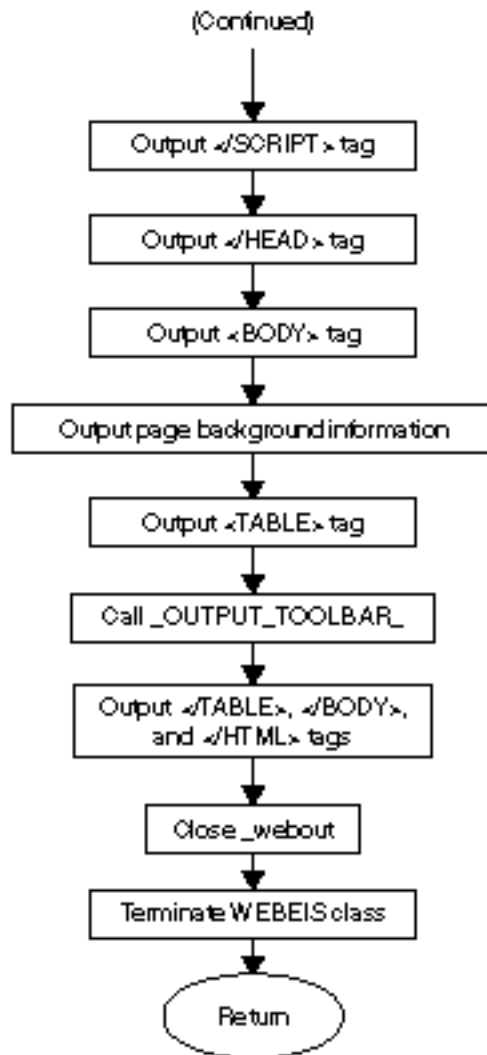
Figure 6. Flow of Control for the the Report Page Toolbar (part 1)





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

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



# **`_BUILD_ACROSSL_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_ACROSSL_LIST_', application-list, across-variable);
```

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

## **Example**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>application-list</i>	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_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>application-list</i>	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_1` 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);
```

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

## **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);
```

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

## **Example**

```
applist= makelist();  
rc=filllist('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);
```

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

## **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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>application-list</i>	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_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);
```

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

## **Example**

```
applist= makelist();  
rc=filllist('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.PRDMDDDB'  
          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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>url</i>	C	the broker component of the URL.
------------	---	----------------------------------

### **Sample output:**

```
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.ACB="1";
  param.ST="1";

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

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

href2="";

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

```

        if (eval(D0)==1) {
            href2+="&D" +"=" +down.options[i].value;
        }
    }
}
href+="D0=" +D0 +href2;

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

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

```



```
    }  
}  
href+="&A0=" +A0 +href2;  
return href;  
}
```

# **`_BUILD_WHERE_FORMAT_STRING_` Method**

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

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>metabase-id</i>	N	the ID number of the metabase
<i>variable-name</i>	C	the name of the variable in the metabase
<i>in-data-value</i>	C	the unformatted data value
<i>out-data-value</i>	C	the string to add to the reach-through WHERE clause.

## **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);
```

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

## **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);
```

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

## **Example**

```
mddbblink= '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_', mddbblink, service, metabase,  
         bgtype, bg, title, webcls);
```

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>
```

# **\_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> </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=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**

```
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);
```

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

## **Example**

```
emddbmid= instance(loadclass('SASHELP.WEBEIS.EMDDB_M.CLASS'));
collist= makelist();
call send(emddbmid, '_GET_CLASS_COMBINATIONS_', 'COL', collist);
actions1= makelist();
```

```

rc= insertc(actions1, '', -1, 'CL_DRILL');
vrflag= 1;
grphvar= 'Actual Sales';
grphstat= 'Sum';
grphacr= 'Month';
url= 'cgi-bin/broker?_PROGRAM=SASHELP.WEBEIS.MDDBRPTS.SCL&_SERVICE=
      default&_debug=&0GRT=BLOCK';
args= '&Mddb=SASUSER.SALES&METABASE=SASUSER.NEWMB&D=COUNTRY&AC=
      =MONTH&A0=2&A1=ACTUAL&A2=PREDICT';
args2= '&S0=2&S1=SUM&S2=AVG';
bgtype= 'COLOR';
bg= 'YELLOW';
title= '1996 Sales Reports';
webcls= 'SASHELP.WEBEIS.WEBEIS.CLASS';
dlflag=0;
service='DEFAULT';
call send(webid, '_DISPLAY_ACROSS_CELLS_', collist, actions1, vrflag, grphvar,
          grphstat, grphacr, url, args, args2, ' ', bgtype, bg, title, webcls, dlflag, service);

```

The example produces the following output:

```

<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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>column-list</i>	N	the column list from the <code>_EMDDB_M_</code> class
<i>dlflag</i>	N	a flag that indicates whether to download the table to a spreadsheet.

## **Example**

The following output is produced:

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

```
<TH COLSPAN=2 CLASS="analycol">  
PREDICTED SALES </TH>  
</TR>
```

# **`_DISPLAY_DEFAULT_TITLE_` Method**

## **Displays the user-specified title**

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 *dflag*.

## **Syntax**

```
CALL SEND(OBJID, '_DISPLAY_TITLE_', dflag);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>dflag</i>	N	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.

## **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);
```

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

## **Example**

```
call send(emddbmid_, '_GET_CLASS_COMBINATIONS_', 'ROW', rowlist);
vrflag=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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>error-message</i>	C	the error message that is to be displayed.

## **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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>dlflag</i>	N	a flag that indicates whether to download the table to a spreadsheet.
---------------	---	---

# **`_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>statistic</i>	C	the statistic that is to be graphed
<i>analysis-variable</i>	C	the analysis variable that is to be graphed
<i>down-variable</i>	C	the down dimension variable that is to be graphed
<i>dsname</i>	C	the data set name from the <code>_WRITE_</code> method
<i>gif-device</i>	C	the device driver name.

# **\_DISPLAY\_ONEWAY\_HBAR\_ Method**

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

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>statistic</i>	C	the statistic that is to be graphed
<i>analysis-variable</i>	C	the analysis variable that is to be graphed
<i>down-variable</i>	C	the down dimension variable that is to be graphed
<i>dsname</i>	C	the data set name from the _WRITE_ method
<i>gif-device</i>	C	the device driver name.

# **`_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>statistic</i>	C	the statistic that is to be graphed
<i>analysis-variable</i>	C	the analysis variable that is to be graphed
<i>down-variable</i>	C	the down dimension variable that is to be graphed
<i>dsname</i>	C	the data set name from the <code>_WRITE_</code> method
<i>gif-device</i>	C	the device driver name.

# **\_DISPLAY\_ONEWAY\_VBAR\_ Method**

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

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>statistic</i>	C	the statistic that is to be graphed
<i>analysis-variable</i>	C	the analysis variable that is to be graphed
<i>down-variable</i>	C	the down dimension variable that is to be graphed
<i>dsname</i>	C	the data set name from the <code>_WRITE_</code> method
<i>gif-device</i>	C	the device driver name.



# **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>column-list</i>	N	the column list from the EMDDDB_M class.
<i>analysis-variable</i>	N	the analysis variable that is to be graphed.
<i>_url</i>	C	the URL of the next query.
<i>_argument-string</i>	C	the argument string for the next query.
<i>_argument-string2</i>	C	the argument string for the next query.
<i>initial-url</i>	C	the URL of the initial HTML page.
<i>URL</i>	C	the broker component of the URL.
<i>service</i>	C	the broker service that is being used.
<i>background-type</i>	C	the background type (IMAGE or COLOR). This value is optional.
<i>background-value</i>	C	the background value. This value is optional.
<i>title</i>	C	the HTML page title.
<i>webcls</i>	C	the WEBEIS class name.
<i>dlflag</i>	N	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.
<i>rowlist</i>	N	the rowlist from the <code>_GET_CLASS_COMBINATIONS_</code> method. This parameter is optional.

## **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';
mddbblink='DYNAMIC';
```

```

url='/cgi-bin/broker';
service='default';
bgtype='color';
bg='yellow';
title='';
webcls='SASHELP.WEBEIS.WEBEIS';
call send(webid,'_DISPLAY_STATISTIC_VARS_',collist,'',_url,_args,
         _args2,mddblink,url,service,bgtype,bg,title,webcls,dlflag,rowlist);

```

The example produces the following output:

```

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

```

# **\_DISPLAY\_SUBSET\_TITLE\_ Method**

Displays the applied subsets in a title

## **Syntax**

```
CALL SEND(OBJID, '_DISPLAY_SUBSET_TITLE_', dlflag);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>dlflag</i>	N	a flag that indicates whether to download the data to a spreadsheet. This parameter is optional.

## **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**

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

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

## **Example**

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

The following output is produced:

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

# **\_DISPLAY\_TWOWAY\_ Method**

## **Calls the methods to display the two-way report**

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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>dlflag</i>	N	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.
---------------	---	---

# **\_DISPLAY\_TWOWAY\_BLOCK\_ Method**

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

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>statistic</i>	C	the statistic that is to be graphed.
<i>analysis-variable</i>	C	the analysis variable that is to be graphed.
<i>down-variable</i>	C	the down variable that is to be graphed.
<i>across-variable</i>	C	the across variable that is to be graphed.
<i>dsname</i>	C	the data set name from the <code>_WRITE_</code> method.
<i>gif-device</i>	C	the device driver name.
<i>subset-list</i>	N	the initial subset list. This parameter is optional.

# **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>statistic</i>	C	the statistic that is to be graphed.
<i>analysis-variable</i>	C	the analysis variable that is to be graphed.
<i>down-variable</i>	C	the down variable that is to be graphed.
<i>across-variable</i>	C	the across variable that is to be graphed.
<i>dsname</i>	C	the data set name from the <code>_WRITE_</code> method.
<i>gif-device</i>	C	the device driver name.
<i>subset-list</i>	N	the subset list that is used for the initial graph. This parameter is optional.

# **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>stat</i>	C	the statistic that is to be graphed.
<i>var</i>	C	the analysis variable that is to be graphed.
<i>down</i>	C	the down variable that is to be graphed.
<i>across</i>	C	the across variable that is to be graphed.
<i>dsname</i>	C	the data set name from _WRITE_ method.
<i>gifdev</i>	C	the device driver name.
<i>subset- list</i>	N	the subset list for the initial graph. This parameter is optional.



# 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);
```

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

*dlflag* N 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);
actions1=makelist();
rc=insertc(actions1, '', -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, actions1, 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>
<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&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&BGTYPE=color&BG=%23FFFFFFE7" 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%3DCANADA& _WHERE=PRODTYPE%3DFURNITURE')" TARGET="_blank">
$97,864</A></TD>
<TD CLASS="tdcell" BGCOLOR=#00FFFF><A href="" onClick="this.href=rturl
(' _WHERE=COUNTRY%3DCANADA& _WHERE=PRODTYPE%3DOFFICE')" TARGET="_blank"> $149,126</
A></TD>
<TD CLASS="tcolcell" BGCOLOR=#0000FF><A href="" onClick="this.href=rturl
(' _WHERE=COUNTRY%3DCANADA')" TARGET="_blank"> $246,990</A></TD>
<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>
<TH CLASS="rowlab" COLSPAN=1 ROWSPAN=1>
<A HREF="/cgi-bin/broker?_PROGRAM=SASHELP.WEBEIS.OPFRAME.SCL&_SERVICE=default
&_DEBUG=0&GRT=NONE&MDDDB=SASHELP.PRDMDDDB&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
&BGTYPE=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>
<TR><TH CLASS="rowlab" NOWRAP ROWSPAN=1 COLSPAN=1>
<A href="" onClick="this.href=clsurl('V11=COUNTRY=U.S.A.&V10=1
&_PROGRAM=SASHELP.WEBEIS.OPFRAME.SCL' )" 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
&MDDDB=SASHELP.PRDMDDDB&METABASE=SASHELP.MBEIS&D=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
&BGTYPE=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>
<TR><TH CLASS="trowlab" ROWSPAN=1 COLSPAN=1>TOTAL</TH>
<TH CLASS="trowlab" COLSPAN=1 ROWSPAN=1>
<A HREF="/cgi-bin/broker?_PROGRAM=SASHELP.WEBEIS.OPFRAME.SCL&_SERVICE=default
&_DEBUG=0&GRT=NONE
&MDDDB=SASHELP.PRDMDDDB&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
&BGTYPE=color&BG=%23FFFFE7" TARGET="_top">
<IMG SRC="/myimages/images/expand.gif" BORDER=0 ALT="Expand"></A></TH>
<TD CLASS="trowcell" BGCOLOR=#0000FF><A href="" onClick="this.href=rturl
(' _WHERE=&_WHERE=PRODTYPE%3DFURNITURE' )" TARGET="_blank">      $290,625</A></TD>
<TD CLASS="trowcell" 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>
</TR></TABLE><BR><BR>

```



## **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>label</i>	C	the long label for an analysis variable
<i>varlist</i>	N	the list of analysis variables
<i>name</i>	C	the analysis variable name.

# **`_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**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>metabase-id</i>	N	the metabase ID number.

## **Example**

The following output is produced:

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

# **`_GET_AVAILABLE_STATS_` Method**

Gets the available statistics from the metabase

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>metabase-id</i>	N	the metabase ID number.



# **\_\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>model-name</i>	C	the name of the data model to use.

# **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>metabase-id</i>	N	the metabase ID number.

# **\_GET\_EMDDDBMID\_ Method**

Returns the ID of the data model from the EMDDDBMID\_ instance variable

## **Syntax**

```
CALL SEND(OBJID, '_GET_EMDDDBMID_', id);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>id</i>	N	the ID of the data model.

# **`_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>mddb</i>	C	the MDDB name.
-------------	---	-------------------

# **\_GET\_MESSAGE\_ID\_ Method**

Returns the ID of the message class from the `DMODEL_` instance variable

## **Syntax**

```
CALL SEND(OBJID, '_GET_MESSAGE_ID_', msgid);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>msgid</i>	N	the ID of the message object.
--------------	---	-------------------------------



# **`_GET_METABASE_NAME_` Method**

Returns the metabase name from the `METABASE_` instance variable

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>metabase</i> C		
-------------------	--	--

		the metabase name.
--	--	-----------------------

# **\_GET\_OUTPUT\_FILE\_ID\_ Method**

Returns the output file ID from the HTMLFILE\_ instance variable

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>fileid</i>	N	the ID of the output file.

# **`_GET_RANGE_COLOR_` Method**

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

## **Syntax**

```
CALL SEND(OBJID, '_GET_RANGE_COLOR_', color, range-list, num);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>color</i>	C	the display color
<i>range-list</i>	C	the RANGE list
<i>num</i>	N	the numerical value to search for.

# **\_GET\_STATDESC\_ Method**

Returns the ID of the statistics description list from the **STATDESC\_** instance variable

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>statdesc</i>	N	the ID of the list that contains statistics descriptions.

# **`_GET_SUBSET_FLAG_` Method**

Returns the value of the `SUBSET_FLAG_` instance variable

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>flagval</i>	C	the value of the subset flag.
----------------	---	-------------------------------

# **\_GET\_USEHOLAP\_ Method**

Returns the value of the HOLAP flag from the USEHOLAP\_ instance variable

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>id</i>	N	the ID of the data model.

# **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>url</i>	C	the URL of the next query.
<i>form-name</i>	C	the name of the form. This parameter is optional.
<i>form-target</i>	C	the target window name. This parameter is optional.

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**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>dlflag</i>	N	a flag that indicates whether to download the table to a spreadsheet.

## **Example**

The following output is produced:

```
<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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>indxfile</i>	C	the fileref of the file that is to be opened.
<i>msgdest</i>	C	the destination for error messages. Valid values are LOG or DIALOG.
<i>rc</i>	N	the return code for errors (1=error).

# **\_OPEN\_TABLE\_ Method**

Outputs the <TABLE> tag for the dynamic HTML pages

## **Syntax**

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

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

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

## **Example**

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

The following output is produced:

```
<TABLE BORDER=3 WIDTH=50% BORDERCOLORDARK=RED BORDERCOLORLIGHT=YELLOW  
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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>column-list</i>	N	the column list from the <code>_emddb_m</code> .
<i>viewreport-flag</i>	N	the <b>View Report</b> button flag.
<i>_url</i>	C	the broker component of the URL.
<i>_argument-string</i>	C	the argument string for the next query.
<i>_argument-string2</i>	C	the argument string for the next query.
<i>_argument-string3</i>	C	the argument string for next query.
<i>initial-url</i>	C	the URL of the initial HTML page.
<i>url</i>	C	the URL for the next query.
<i>service</i>	C	the broker service that is being used.
<i>analysis-variable</i>	C	the analysis variable that is to be graphed.
<i>statistic-variable</i>	C	the statistic that is to be graphed.
<i>analysis-variable</i>	N	the analysis variable that is to be graphed.

<i>background-type</i>	C	the background type (IMAGE or COLOR). This parameter is optional.
<i>background-value</i>	C	the background value. This parameter is optional.
<i>webeis-class</i>	C	the WEBEIS class name.
<i>dlflag</i>	N	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.

# **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>across-variable</i>	C	the previously selected across variable. This parameter is optional.

## **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\_ADDDTL\_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_ADDDTL_CLSVAL_PARMS_');
```

# **\_OUTPUT\_ADDTL\_RT\_PARAMS\_ 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_PARAMS_');
```

# **\_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:

```
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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>service-name</i>	C	the broker service value
<i>next-program</i>	C	the next SCL program to execute.

# **\_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**

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

## **Example**

The following output is produced:

```
<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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>tblflag</i>	C	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.
<i>selvar</i>	C	the analysis variable to mark SELECTED.

## **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:

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

function movesel(fromlistbox,tolistbox) {
pos=0; index=0; newlength=0;
if (fromlistbox.options.length!=0) {
    pos = tolistbox.options.length;
    var listofstats = new Array();
    j = 0;
    for (i=0; i < fromlistbox.options.length; i++) {
        if (fromlistbox.options[i].selected==false && fromlistbox.options[i].value!
="MIXED" && fromlistbox.options[i].text!="") {
            listofstats[j]=fromlistbox.options[i].value;
            j++;
        }
    }

    for (j=0; j < fromlistbox.length; j++) {
        if (fromlistbox.options[j].selected && fromlistbox.options[j].text!=" " &&
fromlistbox.options[j].value!="MIXED") {
            tolistbox.options[pos]=new Option(statslabellist[fromlistbox.options[j].
value],fromlistbox.options[j].value);
            pos++;
        }
    }
}
```

```

}
remstatanal(fromlistbox);
if (num > 1) {
    j=0;
    fromlistbox.options[j]=new Option(statslabellist["MIXED"],"MIXED");
}
else
    j=-1;

for (i=0; i < listofstats.length; i++) {
    j++;
    if ( j==listofstats.length )
        break;
    else
        fromlistbox.options[j]=new Option(statslabellist[listofstats[i]],listofstats
[i]);
    }
}
stats(document.mf.sa,document.mf.s);
}

```



# **`_OUTPUT_BAR_SHAPE_LIST_ Method`**

Outputs the graph bar shape option on the Options page

## **Syntax**

```
CALL SEND(OBJID, '_OUTPUT_BAR_SHAPE_LIST_', bar-shape, view-report-flag);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>bar-shape</i>	C	the currently selected graph bar shape
<i>view-report-flag</i>	N	the <b>View Report</b> flag.

## **Example**

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

The following output is produced:

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

# **\_OUTPUT\_BOOKMARK\_BUTTON\_ Method**

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

## **Syntax**

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

# \_OUTPUT\_BOOKMARK\_URL\_ Method

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

## Syntax

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

Where...	Is Type...	And Contains...
<i>vrlflag</i>	N	the <b>View Report</b> button flag.
<i>url</i>	C	the broker component of the URL.
<i>service-name</i>	C	the broker service value.
<i>analysis-variable</i>	C	the analysis variable that is to be graphed.
<i>statistic</i>	C	the statistic that is to be graphed.
<i>down-variable</i>	C	the down variable that is to be graphed,
<i>graph-type</i>	C	the graph type (BLOCK, HBAR, PIE, PLOT, VBAR).
<i>background-type</i>	C	the background type (IMAGE or COLOR). This value is optional.
<i>background-value</i>	C	the background value. This value is optional.
<i>title</i>	C	the HTML page title.
<i>webeis-class</i>	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&MDDDB=SASHELP.PRDMDDDB&METABASE=SASHELP&D=COUNTRY&AC=YEAR
&A=ACTUAL&A1S1=SUM&BGTYPE=COLOR&BG=YELLOW&GRT=VBAR&DC=1&ACB=1&ST=1&GL=1&GSC=1&SSL=1&SH=3&SW=15
&GH=450&GW=600&DP=1"
```

# **\_OUTPUT\_CLASSVAL\_URL\_FN\_ Method**

**Outputs the CLSVAL JavaScript function on the Report page**

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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>service-name</i>	C	the broker service value.
<i>analysis-variable</i>	C	the analysis variable that is to be graphed.
<i>statistic</i>	C	the statistic that is to be graphed.
<i>across-variable</i>	C	the across variable that is to be graphed.
<i>background-type</i>	C	the background type (IMAGE or COLOR). This value is optional.
<i>webeis-class</i>	C	the WEBEIS class name.
<i>background-value</i>	C	the background value. This value is optional.
<i>title</i>	C	the HTML page title.
<i>url</i>	C	the broker component of the URL.
<i>vrflsg</i>	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
```

File</A></TD></TR>

# **\_\_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);
```

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

## **Example**

```
url='/cgi-bin/broker';
graphtype='';
service='default';
grphvar='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**

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

The following output is produced:

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

# **\_OUTPUT\_DIMENSIONS\_BUTTON\_ Method**

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

## **Syntax**

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

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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>down-variable</i>	C	the previously selected down variable. This parameter is optional.
<i>url</i>	C	the broker component of the URL. This parameter is optional.

## **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=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\_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);
```

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

## **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,
pdmcat,
program, debug, where);
```

# **\_OUTPUT\_DS2HTM\_ST\_ Method**

Outputs the DS2HTM statement to generate the detail data table

## **Syntax**

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

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

## **Example**

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

# **\_OUTPUT\_DYNAMIC\_HIDDEN\_FLDS\_ Method**

Outputs the necessary hidden fields for the initial dynamic HTML page

## **Syntax**

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

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

## **Example**

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

The following output is produced:

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

```
<INPUT TYPE="hidden" NAME="title" VALUE="1997+Sales+Reports">  
<INPUT TYPE="hidden" NAME="class" VALUE="SASHELP.WEBEIS.WEBEIS">
```

# **`_OUTPUT_EMPTY_CELL_` Method**

Outputs an empty cell in the HTML table

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>spannum</i>	N	the number of columns to span.
<i>dlflag</i>	N	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.
<i>cssclass</i>	C	the class name for the cascading style sheet class tag. This parameter is optional.

# **\_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);
```

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

## **Example**

The following output is produced:

```
<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
```

# **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>grtype</i>	C	the previously selected graph type
<i>grmsg</i>	C	the mnemonic of the graph type message
<i>groption</i>	C	the value for the OPTION tag

# **\_\_OUTPUT\_GRAPH\_SOURCE\_OPTION\_\_ Method**

Outputs radio buttons for the Graph Source option

## **Syntax**

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

## **Example**

The following output is produced:

```
<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
& ;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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>url</i>	C	the Broker component of the URL.
<i>background-type</i>	C	the background type (COLOR or IMAGE). This parameter is optional.
<i>background-value</i>	C	the background value. This parameter is optional.

## **Example**

The following output is produced:

```
<HTML><HEAD><TITLE>Mddb Report Viewer Layout</TITLE>
<script language="javascript">

function List(list) {
    for (key in list)
        if (list[key] != null) this[key]= list[key];
}

selected= new List;
selected2= new List;
function change(select) {
    if ((navigator.appName == "Netscape" &&
        navigator.appVersion.indexOf("3.0") != -1) ||
        (navigator.appName == "Microsoft Internet Explorer" &&
        navigator.appVersion.indexOf("4.0") != -1)) {
        options= new Object;
        for (i= 0; i < select.options.length; i++) {
            options[select.options[i].text]=select.options[i].value;
            selected[select.options[i].text]=
                select.options[i].selected ? select.options[i].value : null;
        }
        selected= new List(selected);
        select.options.length= 0;
        for (key in selected)
```

```

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

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

</SCRIPT>
</HEAD>
<BODY BGCOLOR=white>
<CENTER>
<TABLE CELLSPACING=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);
```

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

## **Example**

The following output is produced:

```
<INPUT TYPE="hidden" NAME="_SERVICE" value="default">  
<INPUT TYPE="hidden" NAME="_DEBUG" value="2">  
<INPUT TYPE="hidden" NAME="MDDDB" value="SASHELP.PRDMDDDB">  
<INPUT TYPE="hidden" NAME="METABASE" value="SASHELP.MBEIS">  
<INPUT TYPE="hidden" NAME="BGTYPE" value="color">  
<INPUT TYPE="hidden" NAME="BG" value="%23FFFFFFE7">  
<INPUT TYPE="hidden" NAME="GRT" value="NONE">  
<INPUT TYPE="hidden" NAME="GL" value="1">
```

```
<INPUT TYPE="hidden" NAME="GSC" value="1">  
<INPUT TYPE="hidden" NAME="SSL" value="1">  
<INPUT TYPE="hidden" NAME="ST" value="1">  
<INPUT TYPE="hidden" NAME="SH" value="3">  
<INPUT TYPE="hidden" NAME="SW" value="15">  
<INPUT TYPE="hidden" NAME="GH" value="450">  
<INPUT TYPE="hidden" NAME="GW" value="600">  
<INPUT TYPE="hidden" NAME="DC" value="1">  
<INPUT TYPE="hidden" NAME="ACB" value="1">  
<INPUT TYPE="hidden" NAME="DP" value="1">
```

# **\_OUTPUT\_HIDDEN\_VARS\_ Method**

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

## **Syntax**

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

# **\_OUTPUT\_HTML\_AFTER\_BODY\_ Method**

**Enables users to add HTML tags to the Report page**

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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>title</i>	C	n optional title for the page.
<i>cgi</i>	C	the Broker component for the ACTION tag.
<i>background-value</i>	C	the background image URL or color value. This parameter is optional.
<i>background-type</i>	C	the background type (COLOR or IMAGE). This parameter is optional.

# **\_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);
```

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

## **Example**

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

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.
PRDMDDB&metabase=SASHELP&D=COUNTRY&AC=YEAR&A=ACTUAL&A1S1=SUM
&GRT=VBAR&BGTYPE=COLOR&BG=YELLOW&GV=ACTUAL&GS=SUM&GD=COUNTRY&GA=YEAR&DC=1&ACB=1 ">
```



# **\_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);
```

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

## **Example**

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

The following output is produced:

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

&GV=ACTUAL&GS=SUM&GD=COUNTRY&GA=YEAR&DC=1  
&ACB=1" SCROLLING="NO">

# **\_OUTPUT\_MDDB\_LIST\_ Method**

## **Outputs the list of MDDBs**

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

## **Syntax**

```
CALL SEND(OBJID, '_OUTPUT_MDDB_LIST_', mddblist, mddb);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>mddblist</i>	N	the list of MDDBs.
<i>mddb</i>	C	the currently selected MDDB. 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**

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

The following output is produced:

```
p.  
1  
<A href="" onClick="this.href=clsurl  
( '_PROGRAM=SASHELP.WEBEIS.OPERPT.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.OPERPT.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.OPERPT.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**

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

## **Example**

The following output is produced:

```
function optbtnurl() {
  with (window.parent.main.document.mf) {
    var limit = elements.length;
    href = "/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) {
                href += "&" + elements[i].name + "=" + elements[i].options
[j].value;
              }
              href += "&" + elements[i].name + eval(numselected) + "=" +
elements[i].options[j].value;
            }
            if (thisvar == "A") {
              var href2="";
              stats=elements[i].options[j].value+"STATS";
              statsstr="window.parent.main."+stats;
              statsarray=eval(statsstr);
              if (statsarray.length==1 && statsarray[0]=="nunique") {
                href2+="%A" +j + "S" + "=" + "NUNIQUE";
              }
              else if (statsarray.length==1 && statsarray[0]!="nunique") {
                href2+="%A" +j + "S" + "=" + "SUM";
              }
            }
            else {
              var anum=0;
              for (k=1; k<statsarray.length; k++) {
                anum=j+1;
                href2+="%A" +anum + "S" +k + "=" +statsarray[k];
              }
            }
          }
        }
      }
    }
  }
}
```



```

        var numstats = statsarray.length-1;
        if (numstats > 1) {
            href2+="&A" + anum + "S0=" + numstats;
        }
    }
    href += href2;
}
}
}
if (numselected > 0) {
    href += "&" + elements[i].name + "0=" + eval(numselected);
}
}
else {
    href += "&" + elements[i].name + "=" + elements[i].value;
}
}
}
}
}
return href;
}
}

```

# **\_OUTPUT\_OPTIONS\_BUTTON\_ Method**

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

## **Syntax**

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

## **Example**

The following output is produced:

```
<A href="" 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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>_url</i>	C	the broker component of the URL
<i>message-id</i>	N	the ID of the message system
<i>graph-type</i>	C	the graph type
<i>bar-shape</i>	C	the graph bar shape.

## **Example**

The following output is produced:

```
url='/cgi-bin/broker';  
msgid=instance(loadclass('sashelp.fsp.astmsg.class'),1);  
grphtype='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, rowndx,  
          collist, colndx, curlist);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>mbid</i>	N	the ID number of the metabase
<i>rowlist</i>	N	the row list from the EMDDB_M class
<i>rowndx</i>	N	the index of the current row in the rowlist
<i>collist</i>	N	the column list from the EMDDB_M class
<i>colndx</i>	N	the index of the current column in the collist
<i>curlist</i>	N	the list of classes and their associated values.

## **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);
```

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

## **Example**

The following output is produced:

```
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 = "%23FFFE7";  
  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, grphtype, bg, grphvar, grphstat, grphdown, grphacr, debug);
```

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

## **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=%
23FFFE7&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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>rpttype</i>	C	a previously selected report type.
----------------	---	------------------------------------



# **\_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:

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

### **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';
```

```
avar='ACTUAL';
stat='SUM';
grphdown='';
grtype='BLOCK';
bgtype='color';
bg='yellow';
title='';
webcls='SASHELP.WEBEIS.WEBEIS';
hideflag='1';
call send(webid,'_OUTPUT_ROTATE_BUTTON_',vrflag,_url,service,
          mddbblink,across,down,avar,stat,grphdown,grtype,bgtype,
          bg,title,webcls,hideflag);
```

The following output is produced:

```
<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_', vrlflag, url, service-name,
analysis-variable, statistic, down-variable, graph-type,
background-type, background-value, title, webeis-class);
```

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

## **Example**

This example illustrates the use of the method:

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

&AC=YEAR&A=ACTUAL&A1S1=SUM&GRT=VBAR&DC=1&ACB=1&ST=1&GL=1&GSC=1&SSL=1&SH=3&SW=15  
&GH=450&GW=600&DP=1 "

# **\_OUTPUT\_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:

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

# **\_OUTPUT\_SPREADSHEET\_BUTTON\_ Method**

Outputs the Download to Spreadsheet button as an image

## **Syntax**

```
CALL SEND(OBJID, '_OUTPUT_SPREADSHEET_BUTTON_', vrflag, url,  
         service, grphvar, grphstat, grphdown, grphtype,  
         bgtype, bg, title, webcls);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>vrflag</i>	N	a flag indicating that the <b>View Report</b> button was pressed.
<i>url</i>	C	the broker path.
<i>service</i>	C	the broker service that is being used.
<i>grphvar</i>	C	the analysis variable that is to be graphed.
<i>grphstat</i>	C	the statistic that is to be graphed.
<i>grphdown</i>	C	the down dimension variable that is to be graphed.
<i>grphtype</i>	C	the graph type.
<i>background-type</i>	C	the background type (IMAGE or COLOR). This parameter is optional.
<i>background-value</i>	C	the background value. This parameter is optional.
<i>title</i>	C	the title. This parameter is optional.
<i>webcls</i>	C	the WEBEIS class name (for subclassing).

## **Example**

The following example code 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='';  
webcls='';  
call send (webid, '_OUTPUT_SPREADSHEET_BUTTON_', vrflag, url, service, grphvar,  
          grphstat, grphdown, grphtype, bgtype, bg, title, webcls);
```

The following output is produced:

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

# \_OUTPUT\_SPREADSHEET\_URL\_ Method

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

## Syntax

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

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

## 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.PRDMDDDB&metabase=SASHELP  
&D=COUNTRY&AC=YEAR&A=ACTUAL&A1S1=SUM&DC=1&ACB=1&ST=1&GL=1&GSC=1&SSL=1&SH=3&SW=15
```



&GH=450&GW=600&DP=1&NR=ALL&BS=Star&\_SAVEAS=prdmddb.csv"

# \_OUTPUT\_STANDARD\_GRAPH\_ Method

Outputs the URL that drives the standard GIF Graph request

## Syntax

```
CALL SEND(OBJID, '_OUTPUT_STANDARD_GRAPH_', url, service,
          graph-type, analysis-variable, statistic-variable,
          down-variable, across-variable, webcls);
```

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

## Example

The following example illustrates the use of this method:

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

The following output is produced:

```
<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&grt=VBAR&gv=Actual%20Sales&gs=Sum&gd=COUNTRY&DC=1&ACB=1
&gac=PRODTYPE&GSB=PRODTYPE=TOTAL&SL=%20" ALT="Please wait."
```

ALIGN=CENTER WIDTH=600 HEIGHT=450 BGCOLOR=SILVER></P>

# \_OUTPUT\_STAT\_BOXES\_ Method

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

## Syntax

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

## Example

The following output is produced:

```
<TH ROWSPAN=2 CLASS=laylabel>
Statistics</TH>
<TD CLASS=label>
Select Column
<DIV CLASS="stats">
<SELECT NAME="sa" CLASS="sselect" MULTIPLE SIZE="5" ALIGN="left" onChange="change
(document.mf.sa); updatestatslist(document.mf.sa); ">
<OPTION VALUE=ACTUAL>Actual Sales</OPTION>
</SELECT>
</DIV>
</TD>
<TD CLASS=label>
Available
<DIV CLASS="stats">
<SELECT NAME="as" CLASS="sselect" MULTIPLE SIZE="5" ALIGN="left" onChange="change
(document.mf.as); "></SELECT>
</DIV>
</TD>
<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 href="" onClick="movesel(document.mf.s,document.mf.as);
return true"><IMG SRC="http://localhost/images/left_02g.gif"
width="20" height="24" alt="Remove selected" BORDER=0><BR>
<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>
</TD>
<TD CLASS=label>
Selected<DIV CLASS="stats">
<select NAME="s" CLASS="sselect" MULTIPLE SIZE="5" align="left" onChange="change
(document.mf.s); ">
</SELECT></DIV>
```

</TD>

# **`_OUTPUT_STATIC_HIDDEN_FLDS_` Method**

Outputs the necessary hidden fields for the initial static HTML page

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>metabase</i>	C	the metabase name.
<i>background-value</i>	C	the background image URL or color value. This parameter is optional.
<i>background-type</i>	C	the background type (COLOR or IMAGE). This parameter is optional.
<i>webeis-class</i>	C	the WEBEIS class name.

# **\_OUTPUT\_STAT\_LIST\_ Method**

**Outputs a list of available statistics**

## **Syntax**

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

## **Example**

The following example illustrates the use of the method:

```
<TR><TD CLASS="label">Statistics  
<DIV CLASS="stats">  
<SELECT 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>  
</TD>  
</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);
```

Where...	Is Type...	And Contains...
<i>subloc</i>	C	the list box location.

## Example

The following output is produced:

```
<FONT SIZE=1>  
<DIV CLASS="filterbox">  
<TABLE>  
<TR><TD COLSPAN=4 CLASS="header">Filter by</TD></TR>  
<TR>  
<TR><TD CLASS="label" NOWRAP>Country:<BR></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:<BR></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:<BR></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
```

```
<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>
<R><D><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**

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

The following output is produced:

```
<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);
```

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

## **Example**

The following output is produced:

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

```
</TD>
<TD>
<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>
</TD>
</TR>
```



# **\_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...**

<i>url</i>	C	the broker component of the URL
<i>service</i>	C	the broker service that is being used
<i>bgtype</i>	C	the background type (IMAGE, COLOR, or blank)
<i>grphtype</i>	C	the selected graph type
<i>bg</i>	C	the background value
<i>grphvar</i>	C	the analysis variable that is to be graphed
<i>grphstat</i>	C	the statistic that is to be graphed
<i>grphdown</i>	C	the down dimension variable that is to be graphed
<i>grphacr</i>	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.PRDMDDDB
&metabase=SASHELP.MBEIS&D=Geographic&AC=Product%2520Line
&A=ACTUAL&S=SUM&GRT=VBAR
&GG=AC&BGTYPE=color&BG=%23FFFE7&DC=1
&ACB=1&ST=1&GL=1&GSC=2&SSL=1&SH=3
&SW=15&GH=450&GW=600&DP=1" SCROLLING="NO"&gt;
```

# **\_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:

```
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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>noexp</i>	C	an instruction not to output expand the information. A nonblank means do not output.

# **\_OUTPUT\_VAR\_FUNCTIONS\_ Method**

**Outputs JavaScript functions for ordering variable selections**

## **Syntax**

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

## **Example**

The following output is produced:

```
function List(list) {
    for (key in list)
        if (list[key] != null) this[key]= list[key];
}

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

function update() {
    str= "";
    for (key in selected)
        str= str + key + ",";
}
```

```
if (str.length)
  document.form.order.value= str.substring(0, str.length - 1);
}
```

# \_OUTPUT\_VARIABLE\_SEL\_FORM\_ Method

Outputs the HTML table elements to arrange the Variable Selection page and calls the methods that output the variable and options HTML elements

## Syntax

```
CALL SEND(OBJID, '_OUTPUT_VARIABLE_SEL_FORM_', url, msgid, vrflag,
          grphtype);
```

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

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

## Example

The following output is produced:

```
<FORM ACTION="/cgi-bin/broker" NAME="mf">
<TR>
<TD VALIGN=TOP>
<TABLE>
<TR>
<TD CLASS=header>
Dimensions</TD>
</TR>
<TR CLASS="dimsselbox">
<TD CLASS=label>
Down: <BR>
<SELECT NAME="d" CLASS="select" SIZE=3 MULTIPLE onChange="change(document.mf.d)">
<OPTION 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=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>
<TR CLASS="dimsselbox">
<TD CLASS=label>
Across: <BR>
```

```

<SELECT NAME="ac" CLASS="select" SIZE=3 MULTIPLE onChange="change(document.mf.ac)">
<OPTION SELECTED VALUE="">
<OPTION 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=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>
<TR><TD CLASS="label">Analysis
<DIV CLASS="analysis">
<SELECT NAME="a" CLASS="select" MULTIPLE SIZE=3 onChange="change(document.mf.a)">
<OPTION SELECTED VALUE=ACTUAL>Actual Sales
<OPTION VALUE=DIFF>Sales Lag
<OPTION VALUE=LPERDAY>LPERDAY
<OPTION VALUE=PREDICT>Predicted Sales
<OPTION VALUE=SALESRAT>Sales Ratio
</SELECT>
</DIV>
</TD>
</TR>
<TR><TD CLASS="label">Statistics
<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>
</TD>
</TR>
<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>
</TABLE>

```



```

</TD>
<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>

```

```

<TR><TD CLASS="label">Height</TD><TD><INPUT TYPE=text NAME="gh" CLASS="select"
SIZE=4 MAXLENGTH=4 VALUE="450"></TD></TR>
<TR>
<TD COLSPAN=2 CLASS=header>
Filter Listboxes</TD>
</TR>
<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>
<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>
</TD>
</TR>
</TABLE>
<TR>
<TD ALIGN=CENTER COLSPAN=2>
<INPUT TYPE="submit" NAME="view" CLASS="submit" VALUE="View Report">
</TR>
</TD>

```

# **`__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);
```

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

## **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);
dsid=open(dataset);
service='default';
debug='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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>dataset-id</i>	N	the base table data set identifier
<i>htmlfile-id</i>	N	the identifier for the _webout file.

## **Example**

```
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);
```

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

## **Example**

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

# **\_OUTPUT\_VIEWRPT\_BUTTON\_ Method**

**Outputs the View Report 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>
```



# **\_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>);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>parmlist</i>	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**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>parmlist</i>	N	an optional list for passing in information to the method.

# **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>ttlflag</i>	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**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>ttlflag</i>	C	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.
----------------	---	--

# **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>application-list</i>	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, '_SET_DRILL_LEVELS_', applist);
```

# **\_SET\_EMDDDBMID\_ Method**

Sets the EMDDDBMID\_ instance variable

## **Syntax**

```
CALL SEND(OBJID, '_SET_EMDDDBMID_', id);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>id</i>	N	the ID of the data model.
-----------	---	---------------------------

# **`_SET_EXPAND_FLAG_` Method**

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

## **Syntax**

```
CALL SEND(OBJID, '_SET_EXPAND_FLAG_', rowlist,  
actionsl);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>rowlist</i>	N	the <code>rowlist</code> from the <code>GET_CLASS_COMBINATIONS</code> method
<i>actionsl</i>	N	the <code>actionsl</code> list from the data model.



# **\_SET\_HIERL\_LIST\_ Method**

Sets the hierl\_ instance variable

## **Syntax**

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

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>listid</i>	N	the list ID of the target list to be copied.
---------------	---	--

# **\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>flagval</i>	C	the value of the subset flag.
----------------	---	-------------------------------

# **\_\_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);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>varnum</i>	N	the number of selected subset values.
---------------	---	---------------------------------------

# **\_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-sring2, graph-type, analysis-variable,
          statistic-variable, down-variable, across-variable,
          webcls);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>url</i>	C	the broker component of the URL
<i>service</i>	C	the broker service that is being used
<i>_argument-string</i>	C	the argument string for the next query
<i>_argument-sring2</i>	C	the argument string for the next query
<i>graph-type</i>	C	the selected graph type
<i>analysis-variable</i>	C	the analysis variable that is to be graphed
<i>statistic-variable</i>	C	the statistic that is to be graphed
<i>down-variable</i>	C	the down variable that is to be graphed
<i>across-variable</i>	N	the across variable that is to be graphed
<i>webcls</i>	C	the WEBEIS class name.

# **\_SUBMIT\_GOPTIONS\_ Method**

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

## **Syntax**

```
CALL SEND(OBJID, '_SUBMIT_GOPTIONS_', gifdev);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
-----------------	-------------------	------------------------

<i>gifdev</i>	C	the name of the device driver to be used.
---------------	---	---

# **\_SUBMIT\_GRAPH\_PATTERN\_ Method**

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

## **Syntax**

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

# **\_SUBMIT\_GRAPH\_TITLE\_ Method**

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

## **Syntax**

```
CALL SEND(OBJID, '_SUBMIT_GRAPH_TITLE_', stat, var);
```

<b>Where...</b>	<b>Is Type...</b>	<b>And Contains...</b>
<i>stat</i>	C	the statistic used in the graph
<i>var</i>	C	the analysis variable used in the graph.



# 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:

```
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,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]);
        }
    }
}
```

```

    }
}
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**

Variable Name	Description	Details
MDDB	Selected MDDB	Selected MDDB (for example, SASHELP.PRDMDDB)
METABASE	Selected metabase	Selected metabase (for example, SASHELP.MBEIS)
SR	First row to display in the table	
NR	Number of rows to display in the table	
D	Down	<p>Hierarchies and category variables.</p> <p>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.</p> <p>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.</p>

AC	Across	<p>Hierarchies and category variables</p> <p>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.</p> <p>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.</p>
A	Analysis variable	Analysis variable. This macro variable is deprecated.
S	Statistic	Globally-applied statistic. This macro variable is deprecated.
<i>Am</i>	Analysis variable	Analysis variable, where <i>m</i> designates the particular variable.
<i>AmSn</i>	Statistic	Statistic that is applied only to the analysis variable specified by <i>Am</i> . <i>n</i> 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

GSC	Graph source	1=3D Clickable Graph, 2=Standard GIF Graph(SAS/GRAPH)
GL	Graph location	1=bottom, 2=top, 3=left, 4=right
GRT	Graph type	BLOCK, HBAR, PIE, PLOT, VBAR
BS	Graph bar shapes	STAR, HEXAGON, PRISM, CYLINDER
SPDSHT	Download to Spreadsheet flag	
GW	Graph width	Default=600, max length=4
GH	Graph height	Default=450, max length=4
SSL	Filter list box location	1=right, 2=left, 3=top, 4=bottom
SW	Filter list box width	Default=15
SH	Filter list box height	Default=3
VIEW	View Report button	Value=View Report
GD	Graph down variable	Category variable for graphing
GA	Graph across variable	Across variable for filtering graph values
GG	Graph group variable	Graph group-by variable, second innermost down variable
GSG	Graph subgroup variable	Graph subgroup-by variable, third innermost down variable
SD	Down variable	Same as D, needed for Filter FORM
SAC	Across variable	Same as AC, needed for Filter FORM
CLASS	WEBEIS class name	For subclassing, default is SASHELP.WEBEIS.WEBEIS.CLASS
CSS	Style sheet URL	Applies to variable selection and report pages
CSST	Toolbar style sheet URL	Applies to toolbar frame; if not specified, uses CSS value
BG	Background color or image	Color name or hex value, or image URL
BGTYPE	Background type	COLOR, IMAGE

**Table 2. MDDDB Report Viewer Global Variables**

Variable Name	Description	Details
VMDOFF	Turn VERIFYMD checking off	Any nonblank character turns it off; the default is on
_GRFONT		SWISSB is the default; use SAS font names
_MRVHELP	URL of help file	The default is the Hints and Tips page: <a href="http://support.sas.com/rnd/web/intrnet/mddbapp/hinttips.html">http://support.sas.com/rnd/web/intrnet/mddbapp/hinttips.html</a>
_MRTBLOC	Toolbar location	1=top, 2=bottom, 3=left, 4=right, and 5=none; the default is top
_MRVSEP	Download to spreadsheet delimiter	A comma is the default
_MRVTBSC	Toolbar frame scrolling	NO or blank indicates no scrolling, YES adds a scrollbar to frame
_MRVTBSZ	Toolbar size in pixels	A character string of the form (horiz, vert); the default is (50, 125)
_MRNODIMBOXES	Turns off the down and across list boxes and the View Report button on display	A nonblank value turns this off; the default is on
_MRNOFRAMES	Indicates whether to use HTML frames in the output	A nonblank value turns this off; the default is on
_MRNOVARCHECK	Turns off the down and across variable selection error checking	A nonblank value turns this off; the default is on
_MRBODYONLY	Nonblank	Outputs the HTML between the <BODY> and </BODY> tags on the Layout and the Report pages.
_MRVFRAMESET	Enables you to specify a custom <FRAMESET> tag on the Report page	
_MRVNOPGOP	Nonblank	Turns off the paging feature
_MRVRNDX1	Value for the radio button that represents the first number of rows	The default is All

_MRVRNDX2	Value for the radio button that represents the second number of rows	The default is 25
_MRVRNDX3	Value for the radio button that represents the third number of rows	The default is 50
_MRVRNDX4	Value for the radio button that represents the fourth number of rows	The default is 100
_MRVNRLKS	The minimum number of paging lines to display beneath the report table	The default is 5
_MRNOSORT	Turns off the sorting feature	A nonblank value turns this off; the default is on
_MRTBLPRM	Sets report <TABLE> tag parameters	For example, "CELLPADDING=4 CELLSPACING=2 BORDER=3"

# 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,

```
<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**

Class Name	Description
MAINTAB	Main report table
ROWLAB	Row label cells
TROWLAB	Total row label cell
STROWLAB	Total row label cell for expanded row (for example, "subtotals")
TROWCELL	Total row data cells
TDCELL	All other data cells
TCOLLAB	Total column label cell
STCOLLAB	Total column label cell for nested totals
TCOLCELL	Total column data cells
COLLAB	Column label cells
EMPTY	Empty cell in upper left-hand corner
FILTERBOX	Table containing filter list boxes



DIMBOX	Table containing dimension selector list boxes
DIMSELBOX	Table containing dimension selector list boxes (Report Layout page)
ANALYBOX	List box for selecting analysis variable (Report Layout page)
ANALYSIS	Class for the <DIV> tag for the analysis variable list box
STATSBOX	List box for selecting statistic (Report Layout page)
STATS	Class for the <DIV> tag for the statistics list box
ANALYCOL	Analysis variable column
STATSCOL	Statistics column
GRAPH	Class for the IMG tag for standard GIF graph
GRAPHAPP	Class for the graph application tag
TOOLTAB	Class for toolbar
IMGBKMRK	Class for the IMG tag for Bookmark
IMGDIM	Class for the IMG tag for Dimensions
IMGOPT	Class for the IMG tag for Options
IMGHELP	Class for the IMG tag for Help
IMGLAY	Class for the IMG tag for Layout
IMGLOGOUT	Class for the IMG tag for Logout
IMGROTATE	Class for the IMG tag for Rotate
HEADER	Report Layout HTML headers
LABEL	Report Layout HTML labels
SELECT	Report Layout HTML for SELECT and INPUT tags
SSELECT	Class for statistics selection list boxes
SUBMIT	Submit (View Report) button class