SAS Publishing

SAS® Web Analytics 5.2

Administrator's Guide

The Power to Know.®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2005. *SAS® Web Analytics 5.2: Administrator's Guide*. Cary, NC: SAS Institute Inc.

SAS® Web Analytics 5.2: Administrator's Guide

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

1st printing, November 2005

Contents

What's New

Overview

New capabilities, functions, and options in SAS Web Analytics 5.2 improve your ability to perform the following tasks:

□ more accurately capture information by using a page tagging template

□ view additional information in the improved Path Analysis report

□ run dynamic funnel reports and scheduled funnel reports (reports that are executed routinely with the ETL process in a batch program)

□ take advantage of a more powerful and flexible Equation Editor

□ better integrate and manipulate data for dashboards, segmentation reports, and scorecards with external data sources

□ independently develop and test new summarizations by using the new %WASUMTST macro

Details

□ Page tagging, which is the ability to place JavaScript or other code on the Web to more accurately capture Web data, has become a standard requirement for Web analysis vendors. SAS Web Analytics 5.2 now supports this page tagging capability by providing a page tagging template.

□ The Path Analysis report in SAS Web Analytics 5.2 includes the following new parameters:

  □ percent of overall visitors

  □ percent drop-off

  □ percent of visitors remaining from the start of the clickstream path

□ The %WAPATHDP macro has been extensively updated, which significantly improves the performance of the Path Analysis report.

□ Reports in SAS Web Analytics 5.2 are grouped for better accessibility.

CHAPTER

1

Overview of the SAS Web Analytics 5.2 Solution

About the Documentation for System Administrators

After you have installed all of the components of the SAS Web Analytics 5.2 solution successfully, use this documentation to help you set up and configure an analysis-ready Web mart. Setup and configuration includes the following tasks:

- □ Initialize a Web mart.
- □ Register a Web mart.
- □ Edit Web mart properties in the SAS e-Data ETL Administrator.
- □ Customize the ETL processing of the detail data sets by using the SAS Web Analytics Administrator.
- □ Configure SAS summary data sets by using the SAS Web Analytics Administrator. These summary data sets are queried from the SAS Web Analytics Report Viewer to display the reports.
- □ Process the Web log data (using the Daily.sas program).
- □ Create new traffic reports.
- □ Set up the variables and the metrics for scorecards, dashboards, and segmentation reports by using the SAS Web Analytics Administrator.

What's Included in the SAS Web Analytics 5.2 Solution

The software bundle that comprises the SAS Web Analytics solution includes SAS®9 and the following applications:

- □ SAS e-Data ETL software
- □ SAS Web Analytics 5.2 software

What Is SAS e-Data ETL Software?

SAS e-Data ETL software is a data sourcing tool for clickstream data. It is used to extract large volumes of raw clickstream data, identify the data by visit, and then parse, filter, and load the data into SAS detail data sets. You use the SAS e-Data ETL software to create Web marts and to edit Web mart properties.

What Is SAS Web Analytics 5.2 Software?

SAS Web Analytics 5.2 software is a clickstream analysis and graphic reporting tool that performs the following functions:

- □ reads the detail data sets that are generated by SAS e-Data ETL software
- □ generates the summary data sets (part of the Web mart) from the detail data sets
- □ reads data into the Web mart from sources other than Web logs (optional)
- □ provides a Web-based interface for dynamically producing reports about Web site visitors and their behavior

Why Does My Organization Need the SAS Web Analytics 5.2 Solution?

Organizations that do business on the Web typically want to increase the revenue that is generated from their Web sites or to increase customer retention. However, the nature of Web server log files makes it difficult to transform the raw Web server log data into information that organizations can easily use to accomplish their key business objectives. The SAS Web Analytics solution facilitates this transformation of raw data into useful information.

The Business Objectives of Retail Industries

Specific business objectives can vary across industries. Retail organizations typically have the following key business objectives for their Web sites:

- understanding the effects of online and offline campaigns in order to improve the perceived value of the Web site for customers or to improve customer experience at their Web site
- providing insight into distinct browsing, interest, and buying patterns
- reducing marketing expenses by targeting the right customers

Business Objectives of Financial and Service Organizations

Financial and services organizations typically have the following key business objectives for their Web sites:

- reducing costs by encouraging customers to use the Web site rather than more expensive call centers or personal visits to loan officers
- increasing customer “wallet share” by taking advantage of cross-selling and up-selling of other related products

Business Objectives of Telecomunications Organizations

Telecommunications organizations typically have the following business objectives for their Web sites:

- efficiently handling the enormous volumes of Web server log data that must be processed daily
- integrating a wide variety of data sources, including wireless logs, contract data, and point-of-sales information, in order to obtain a more complete view of customers

Capabilities That Are Required to Meet Business Objectives

In order to accomplish the business objectives of your organization, regardless of its specific type, you need the following capabilities:

- the ability to read and extract Web server log data using a process that can be easily customized
- the ability to transform the data into a useful format
- the ability to load the data into an analysis-ready Web mart
- the ability to generate dynamic, customizable reports that provide the necessary information for your organization to accomplish its key objectives

The SAS Web Analytics solution provides these capabilities. The software enables a wide variety of organizations to better understand the activity on their Web sites and, in turn, helps them to accomplish their key business objectives.

Web Mart and Data Structures

SAS Business Intelligence Architecture

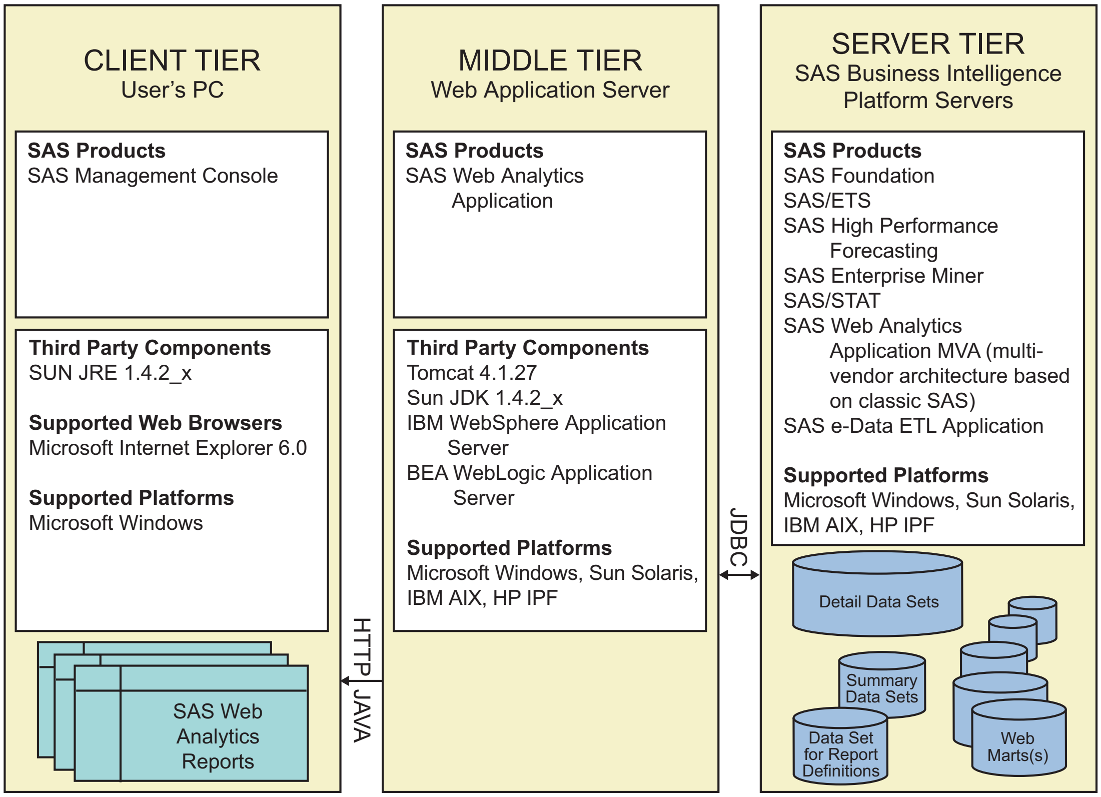
The SAS Web Analytics 5.2 solution is built upon the three-tier architecture of the SAS®9 Business Intelligence platform, which is made up of the following tiers:

Server tier — The ETL processes and the SAS programs that produce reports reside on the server tier.

Middle tier — The Java code for the SAS Web Analytics Administrator and the SAS Web Analytics Report Viewer resides on the middle tier.

Client tier — The Web browser that invokes the SAS Web Analytics Administrator and Report Viewer resides on the client tier.

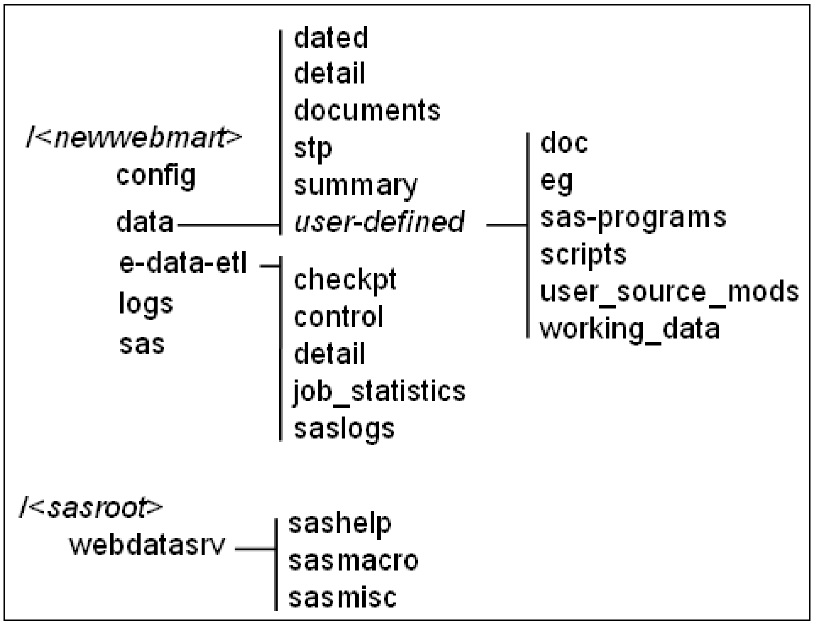
Display 1.1 The Three-Tier Architecture of SAS Web Analytics 5.2


CLIENT TIER
User's PC
SAS Products
SAS Management Console
Third Party Components
SUN JRE 1.4.2_x
Supported Web Browsers
Microsoft Internet Explorer 6.0
Supported Platforms
Microsoft Windows
SAS Web Analytics Reports
HTTP JAVA
MIDDLE TIER
Web Application Server
SAS Products
SAS Web Analytics Application
Third Party Components
Tomcat 4.1.27
Sun JDK 1.4.2_x
IBM WebSphere Application Server
BEA WebLogic Application Server
Supported Platforms
Microsoft Windows, Sun Solaris, IBM AIX, HP IPF
JDBC
SERVER TIER
SAS Business Intelligence Platform Servers
SAS Products
SAS Foundation
SAS/ETS
SAS High Performance Forecasting
SAS Enterprise Miner
SAS/STAT
SAS Web Analytics Application MVA (multi-vendor architecture based on classic SAS)
SAS e-Data ETL Application
Supported Platforms
Microsoft Windows, Sun Solaris, IBM AIX, HP IPF
Detail Data Sets
Summary Data Sets
Data Set for Report Definitions
Web Marts(s)

Directory Structure

Several directories are created when you create a new Web mart, and they are always associated with that specific Web mart. Other directories exist in your SAS root directory path. Additionally, you can create directories that might be needed for customized processing. The following diagram illustrates the directory structure for a SAS Web Analytics Web mart:

Figure 1.1 Directory Structure for a SAS Web Analytics Web Mart



The following directories are significant to the administration of SAS Web Analytics:

- **`webdatasrv`**
- **`e-data-etl`**
- **`data`**
- a user-defined directory that contains the Web logs to be processed
- a temporary working directory

Webdatasrv Directory

When you install SAS e-Data ETL software in the Windows operating environment, the **`webdatasrv`** directory (Figure 1.1 on page 5) is created in your SAS root directory (in the server tier). The **`webdatasrv`** directory contains—among other libraries—the Sashelp library, which contains the Wbetl catalog. The Sashelp.Wbetl catalog is significant because it contains the source entries for the SAS e-Data ETL processes.

Note: An example of the path to the **`webdatasrv`** directory is **`C:\Program Files\SAS\SAS 9.1\webdatasrv`**. △

The **`webdatasrv`** directory contains the following libraries:

Sashelp contains the distributed catalogs and data sets for the SAS e-Data ETL application.

Sasmacros contains the macros for the SAS e-Data ETL application.

Sasmisc contains miscellaneous SAS program files.

Note: If you are installing SAS e-Data ETL software under UNIX, then the directory that contains the miscellaneous files is the **misc** directory. △

E-data-etl Directory

When you initialize a new Web mart by using the %WAETL macro, the %WAETL macro creates a set of permanent directories for that Web mart on the server tier (a SAS Business Intelligence Platform server) in a directory location that you specify.

The **e-data-etl** directory contains the libraries that contain the data sets that control the SAS e-Data ETL software processes. An **e-data-etl** directory exists for each Web mart that is created (see Figure 1.1 on page 5).

The following table describes the libraries that are automatically created within the **e-data-etl** directory for each Web mart:

Table 1.1 Libraries within an E-data-etl Directory

Name	Description
Checkpt	Contains the contents of the detail data set from the last run of the Load process.
Control	Contains modified data sets from the Sashelp library. Examples of data sets that you might customize include Wbconfig and Wbfields.
Detail	Contains detail data sets such as Weblog_Detail_1.sas7bdat.
Job_Statistics	Contains the Job_Statistics data set for the Extract and Load process. Using the Job_Statistics data set, you can produce reports to examine the success and run time of a process.
Saslogs	Contains the SAS logs from the child jobs that are run in separate, concurrent SAS sessions (to optimize the jobs). Save these logs so that you can provide them to SAS Technical Support if you have a problem.
Summary	Contains data sets for internal use only.

Data Directory

The **data** directory for each Web mart contains the detail data sets, summary data sets, and other data sets that are stored in the following directories:

Table 1.2 Directories within the Data Directory

Name	Description
dated	The storage area for the daily session (or visit) summary data sets. The daily session summary data sets are named according to the following convention: Session_yyyymmdd. For example, the daily summary data set that includes all of the sessions on January 15, 2004, is named Session_20040115.
detail	The storage area for the daily detail data sets. The detail data sets are named according to the following convention: Detail_yyyymmdd. For example, the daily detail data set for January 15, 2004, is named Detail_20040115.
documents	The location of the metadata data sets for the SAS Web Analytics reports.

stp	The storage area for the temporary output data sets that are generated by the stored processes. The SAS Web Analytics Report Viewer looks for the output data set(s) in the **stp** directory in order to create the requested report. There are no permanently stored entries in this directory; all stored process output is deleted after it is displayed.
summary	The storage area for the summarized data sets, which are named according to the following convention: the descriptive name that identifies the type of data that has been summarized (such as Referrer_Search_Term) is followed by a string of text that identifies the time interval for the summarized data (such as _DAY, _WEEK, _MONTH, _QTR, or _YEAR). For example, Referrer_Search_Term_MONTH.sas7bdat specifies the summary data set that contains all of the referrer search terms that were summarized by month (for a specified Web mart).
user-defined directory	A directory that you have defined if you need a separate storage location for supplemental data. For example, you can use this directory to store data for a metric (such as server CPU usage) that is not created by the SAS Web Analytics ETL processes.

Temporary Working Directory

SAS e-Data ETL software creates a temporary working directory for the data sets that are created during the Extract and Load processes. You specify the name of this directory during the creation of the Web mart. See the Temporary Location step ("Creating a New Web Mart by Using the SAS e-Data ETL Administrator" on page 20) for more information.

The Directory that Contains the Web Logs

The SAS e-Data ETL Extract process looks for Web logs to process (for a selected Web mart) in the location that you specify in the SAS e-Data ETL Administrator. To either specify or change the location of the Web logs to process, see "The Web Log Location Properties" on page 35.

Creating Additional Directories for a New Web Mart

You might want to create some additional directories after defining a new Web mart. You should store these directories in the same directory path as the directories that are automatically created, within the **e-data-etl** directory. The following table lists the directories you can manually create, along with descriptions of what each directory might contain.

Table 1.3 Web Mart Directories That You Can Create

Name	Description
doc	Contains documentation that you might want to add. For example, you might list customizations that you have made.
eg	Contains site-specific Enterprise Guide projects if you are using SAS Enterprise Guide.
sas_programs	Contains the SAS programs that are used during processing. For example, you can store the SAS code that is necessary to allocate the libraries or to run the Extract and Load processes. You can use these SAS programs with your scheduling package to automate your SAS e-Data ETL processes.

Name	Description
`scripts`	Contains VBScripts or Perl scripts that are used during the automation process.
`user_source_mods`	Contains modifications to source entries in the Wbetl catalog. The Usermods library reference points to this directory.
`working_data`	Contains the Web logs that SAS e-Data ETL software needs to process. These logs need to be moved from their original storage location to this directory in order to be processed. You can use scripts to automate the moving of the Web logs.

Input for the SAS Web Analytics Solution

Types of Data Sources for the SAS Web Analytics Solution

The SAS Web Analytics solution can read the following types of data sources:

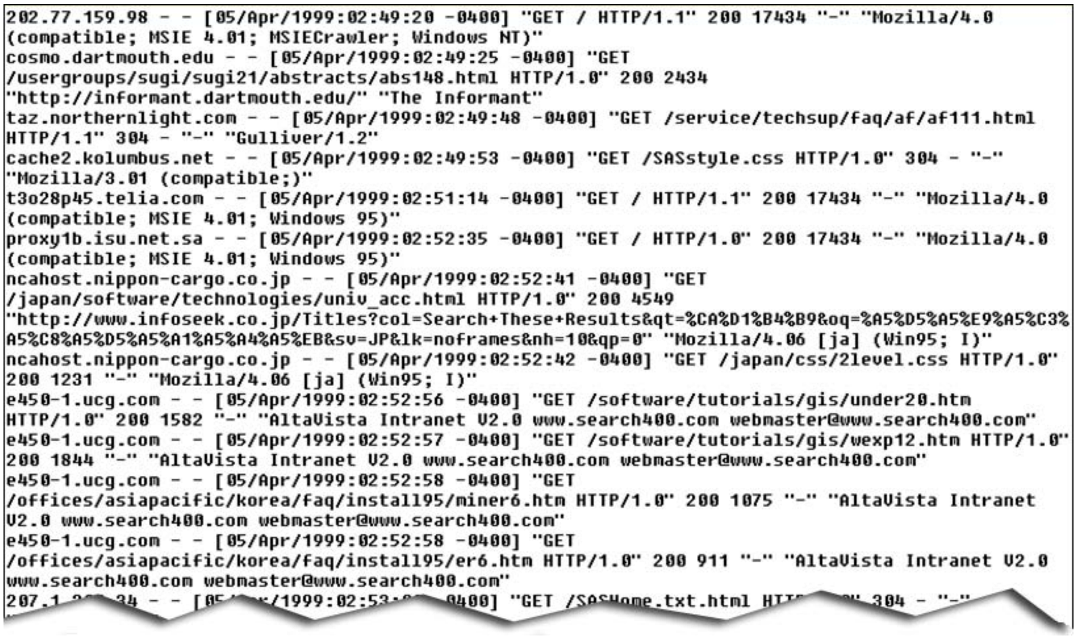
□ Web server log files (by using e-Data ETL software)

□ non-Web log sources (formatted as SAS data sets) such as demographic data about customers from your organization's sales channels

Web Server Log Files

The data in a typical log file from a Web server looks similar to the following display:

Display 1.2 Example of the Output from a Typical Web Server Log File



```
202.77.159.98 - - [05/Apr/1999:02:49:20 -0400] "GET / HTTP/1.1" 200 17434 "-" "Mozilla/4.0
(compatible; MSIE 4.01; MSIECrawler; Windows NT)"
cosmo.dartmouth.edu - - [05/Apr/1999:02:49:25 -0400] "GET
/usergroups/sugi/sugi21/abstracts/abs148.html HTTP/1.0" 200 2434
"http://informant.dartmouth.edu/" "The Informant"
taz.northernlight.com - - [05/Apr/1999:02:49:48 -0400] "GET /service/techsup/faq/af/af111.html
HTTP/1.1" 304 - "-" "Gulliver/1.2"
cache2.kolumbus.net - - [05/Apr/1999:02:49:53 -0400] "GET /SASstyle.css HTTP/1.0" 304 - "-"
"Mozilla/3.01 (compatible;)"
t3o28p45.telia.com - - [05/Apr/1999:02:51:14 -0400] "GET / HTTP/1.1" 200 17434 "-" "Mozilla/4.0
(compatible; MSIE 4.01; Windows 95)"
proxy1b.isu.net.sa - - [05/Apr/1999:02:52:35 -0400] "GET / HTTP/1.0" 200 17434 "-" "Mozilla/4.0
(compatible; MSIE 4.01; Windows 95)"
ncahost.nippon-cargo.co.jp - - [05/Apr/1999:02:52:41 -0400] "GET
/japan/software/technologies/univ_acc.html HTTP/1.0" 200 4549
"http://www.infoseek.co.jp/Titles?col=Search+These+Results&qt=%CA%D1%B4%B9&oq=%A5%D5%A5%E9%A5%C3%
A5%C8%A5%D5%A5%A1%A5%A4%A5%EB&sv=JP&lk=noframes&nh=10&qp=0" "Mozilla/4.06 [ja] (Win95; I)"
ncahost.nippon-cargo.co.jp - - [05/Apr/1999:02:52:42 -0400] "GET /japan/css/2level.css HTTP/1.0"
200 1231 "-" "Mozilla/4.06 [ja] (Win95; I)"
e450-1.ucg.com - - [05/Apr/1999:02:52:56 -0400] "GET /software/tutorials/gis/under20.htm
HTTP/1.0" 200 1582 "-" "AltaVista Intranet V2.0 www.search400.com webmaster@www.search400.com"
e450-1.ucg.com - - [05/Apr/1999:02:52:57 -0400] "GET /software/tutorials/gis/wexp12.htm HTTP/1.0"
200 1844 "-" "AltaVista Intranet V2.0 www.search400.com webmaster@www.search400.com"
e450-1.ucg.com - - [05/Apr/1999:02:52:58 -0400] "GET
/offices/asiapacific/korea/faq/install95/miner6.htm HTTP/1.0" 200 1075 "-" "AltaVista Intranet
V2.0 www.search400.com webmaster@www.search400.com"
e450-1.ucg.com - - [05/Apr/1999:02:52:58 -0400] "GET
/offices/asiapacific/korea/faq/install95/er6.htm HTTP/1.0" 200 911 "-" "AltaVista Intranet V2.0
www.search400.com webmaster@www.search400.com"
207.1    34 - - [05/  /1999:02:53:   0400] "GET /SASHome.txt.html HTT    304 - "-"
```

The SAS e-Data ETL program can recognize and read all the major Web log formats. You can also modify the SAS Web Analytics solution to read custom Web logs (see

"Overview: Custom Web Logs" on page 64). The SAS Web Analytics solution supports the following common Web log formats:

Table 1.4 Common Web Log Formats That Are Supported by the SAS Web Analytics Solution

Web Log Format	Type of Server
Common Log Format (CLF)	Apache Web servers
Extended Log Format (ELF)	Microsoft Internet Information Server (IIS) Web servers
Microsoft Proxy	Microsoft Proxy servers
Netscape/iPlanet	Netscape and iPlanet Web servers
IIS (original)	Microsoft personal Web servers

Use the SAS e-Data ETL Administrator to specify the location of your Web log file(s) (see "The Web Log Location Properties" on page 35). When you run the Daily.sas file each day, the Extract process reads any Web log files that are in the location that you have specified.

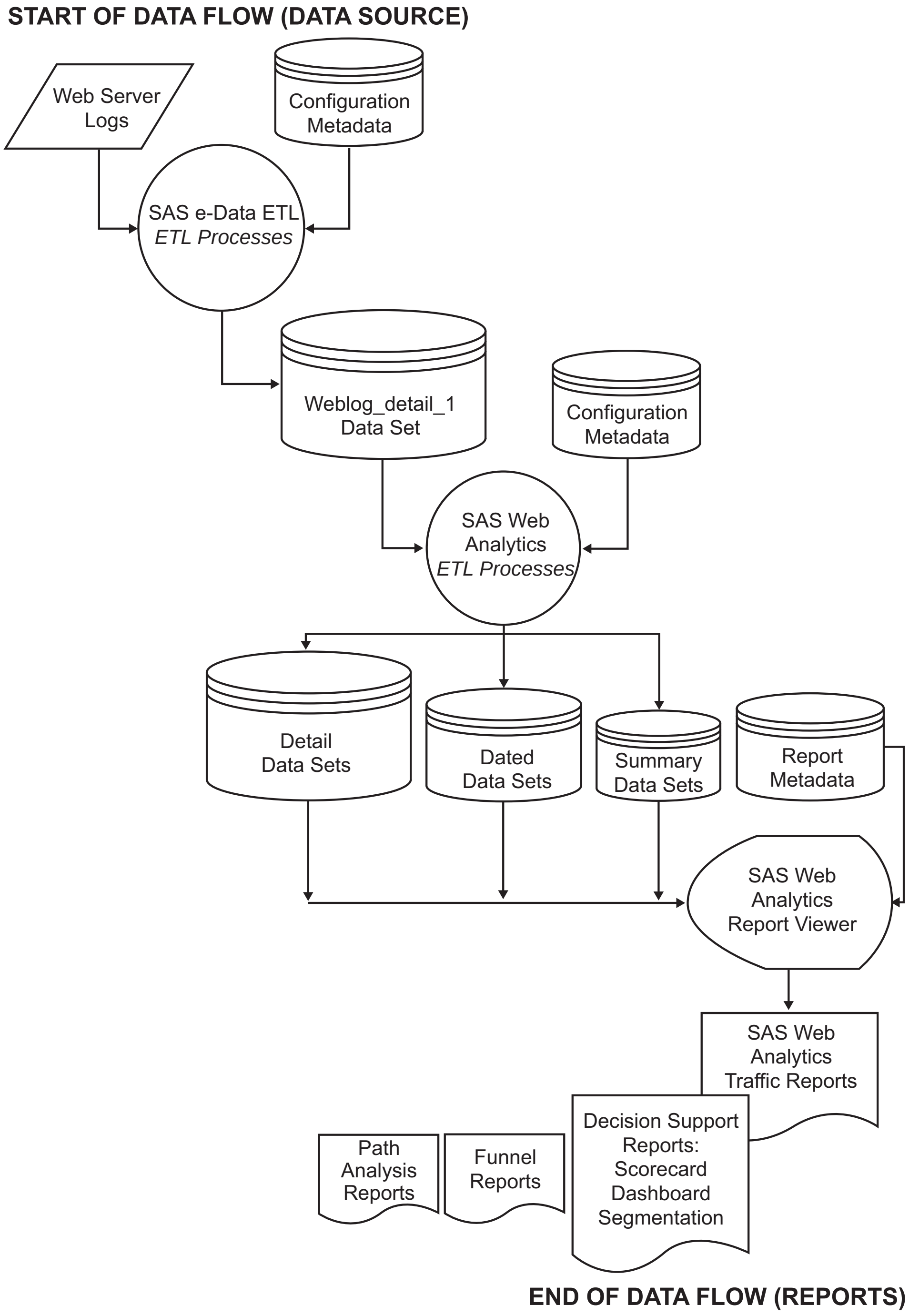
Other Data Sources

You can add a custom data source (such as a table of online product sales figures) to your Web mart if you have prepared your Web site structure and if you have customized your Web log to include an identifier that makes a link between the Web mart and the external data source. You also need to create custom SAS code that runs during the ETL processes to summarize any new variables and add them to your summarized data sets.

ETL Processes for the SAS Web Analytics Solution

The following diagram illustrates the flow of data for the SAS Web Analytics solution:

Display 1.3 Data Flow for SAS Web Analytics



The Daily.sas program invokes two macros that perform the extraction, transformation, and loading (ETL) processes—turning the raw data from Web server log files to an analysis-ready format in SAS data sets.

The Daily.sas program is generated when you create a new Web mart by using the %WAETL macro with the INITIALIZE argument for the PROGRAM parameter. The Daily.sas program runs the macros shown in the following table, which comprise the ETL portion of the SAS Web Analytics solution:

Table 1.5 Macros That the Daily.sas Program Runs

Name	Description
%EDATAETL (PROGRAM=EXTRACT)	Reads the Web server logs and creates temporary SAS data sets. The data is extracted by running multiple SAS sessions concurrently. To run only the Extract process, see "Running the Extract Process" on page 77. To customize the Extract process, see "Overview: The Modules in the Extract Process" on page 80.
%EDATAETL (PROGRAM=LOAD)	Reads the temporary data sets that are created by the Extract process. Creates detail SAS data sets, including the Weblog_Detail_1 data set. To run only the Load process, see "Running the Load Process" on page 98. To customize the Load process, see "Overview: The Modules of the Load Process" on page 99.
%WAETL (PROGRAM=WAREHOUSE)	Reads the Weblog_Detail_1 data set that is created by the Load process. The WAREHOUSE argument also creates the summary data sets that are referenced in order to present SAS Web Analytics reports to the user. For more about the WAREHOUSE argument, see "Using the %WAETL Macro to Load Data into an Existing Web Mart" on page 272.

To generate the Daily.sas program, see "Using the %WAETL Macro to Load Data into an Existing Web Mart" on page 272. To run the Daily.sas program, see "Using the Daily.sas Program to Perform the ETL Processes" on page 68.

Note: After ETL processing is finished, use the SAS Web Analytics Report Viewer to view online reports. When you select a report to view in the SAS Web Analytics Report Viewer, the information about which variables to display for the selected report is then retrieved from the summary data sets and the metadata data sets in order to create the SAS Web Analytics report. △

How the Extract Process Prepares to Read the Web Logs

From a functional perspective, the Extract process performs the following tasks before it reads each Web server log file:

- □ identifies the available Web server log files to be read and determines the type of Web server logs to be read by consulting the Control.Wbfields data set (a data set that contains metadata).
- □ builds the reader jobs (SAS programs) that will read the Web server log files. The Web server log files are divided among many reader jobs. The specific number of reader jobs is determined by the value of the Number_of_Parallel_Reader_Jobs property that you specify within the SAS e-Data ETL Administrator.

- builds the INPUT statements for each reader job. These statements specify both the Web server log type and other fields that are defined in the Control.Wbfields data set.

The Extract Process: About Temporary Data Sets

The Extract process then executes the reader jobs that read the Web server log files. Each reader job groups its data into temporary data sets. As each row of the Web log is read, the reader job performs the following actions:

- filters out the non-page items, the requests from special clients, the requests from spiders and robots, and the requests that generated error status codes
- identifies cookie information if cookie information is present
- identifies any requests for executable programs such as CGI programs
- identifies the visits and the clickstream path that each visitor followed
- generates the path analysis information for the Web site

After the Web logs have been read, the Extract process executes the analysis jobs. The analysis jobs restructure the temporary data sets according to the visitor ID and the datetime stamp.

Customizing the Extract Process

The Extract process, Sashelp.Wbetl.Extract.Source, is invoked by the %EDATAETL macro within the Daily.sas program. The Extract process includes a number of modules. The modules within the Extract process that are available for customization are called custom access modules (CAMs). You can customize the following CAMs:

- User_assignments_after_input
- User_assignments_before_output
- User_assignments_by_session_id

To edit the CAMs in the Extract process, see "Modules That You Can Modify in the Extract Process" on page 76. For a list of all of the modules (both customizable and non-customizable) that are executed by the Extract process, see "Overview: The Modules in the Extract Process" on page 80.

The Load Process

The Load process, Sashelp.Wbetl.Load.Source, is also invoked by the %EDATAETL macro within the Daily.sas program. Although the Load process includes a number of modules, the only custom access module (CAM) is the User_assignments_for_data_mining CAM.

The User_assignments_for_data_mining CAM prepares your Web log data for data mining. User_assignments_for_data_mining creates a view called Detail.Web_Mine that contains all the current Web log detail data for page views, including CGI parameters and cookies. You might customize the code in User_assignments_for_data_mining for your Web site so that data mining operations in your Web site are more effective.

The User_assignments_for_data_mining CAM is automatically executed at the end of the Load process. To edit the User_assignments_for_data_mining CAM, see "User_assignments_for_data_mining" on page 100.

Customizing the Summarizing and Reporting Processes in SAS Web Analytics

You can customize the settings that control the summarizing and reporting processes as described in the following table:

Table 1.6 Settings That You Can Customize in SAS Web Analytics

Category of Information	Description
Web Analytics Configuration	Controls processing details such as the name of the default SUMMARY Engine metadata data set and the number of days available in the _DAY summaries.
User Interface Settings	Controls the configuration and appearance of reports, such as the foreground color.
Summaries	Controls how to create and process the summary data sets.
Scorecards	Controls how to process a selected scorecard.
Dashboards	Controls how to process a selected dashboard.
Segmentations	Controls the definitions for the automatic segmentation process.

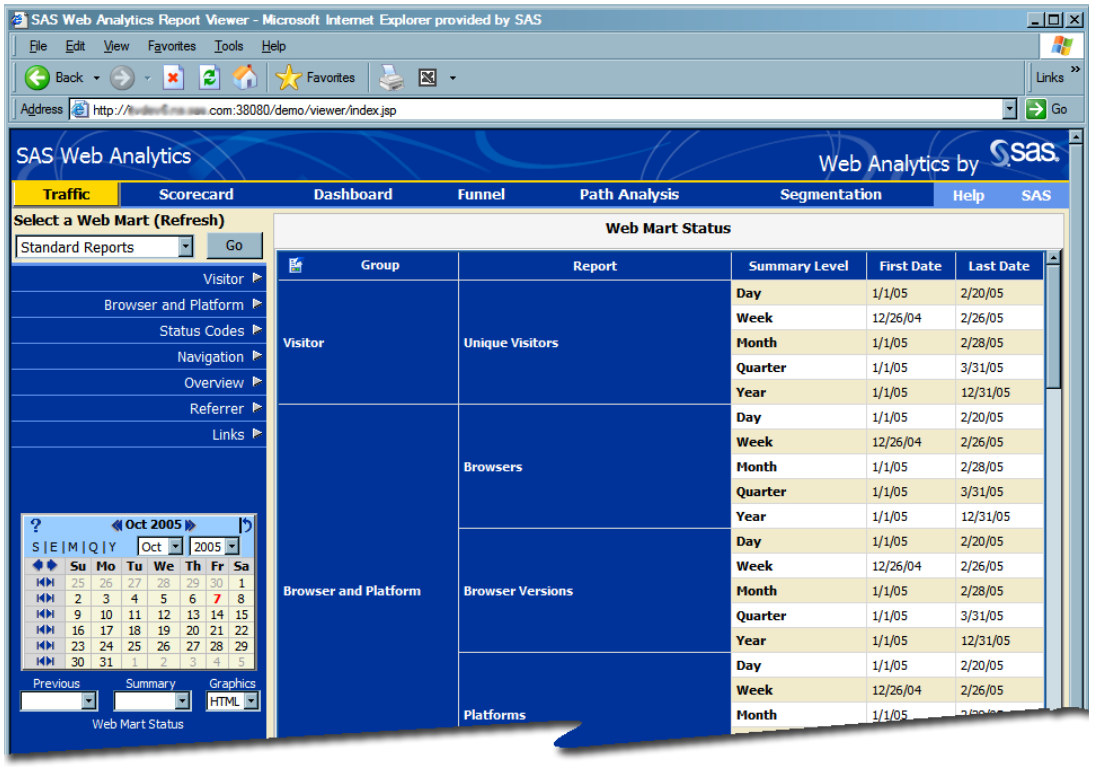
For more information about the SAS data sets that contain these settings, see "The Waadmsum Data Set" on page 291.

Output from the SAS Web Analytics Solution

The SAS Web Analytics Report Viewer

The SAS Web Analytics Report Viewer (see Display 1.4 on page 14) is a Web-based application that users can access from any supported Web browser. It enables users to view SAS Web Analytics reports from available Web mart data.

Display 1.4 The SAS Web Analytics Report Viewer in the Microsoft Internet Explorer Browser



Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Web Mart Status

Group	Report	Summary Level	First Date	Last Date
Visitor	Unique Visitors	Day	1/1/05	2/20/05
		Week	12/26/04	2/26/05
		Month	1/1/05	2/28/05
		Quarter	1/1/05	3/31/05
		Year	1/1/05	12/31/05
Browser and Platform	Browsers	Day	1/1/05	2/20/05
		Week	12/26/04	2/26/05
		Month	1/1/05	2/28/05
		Quarter	1/1/05	3/31/05
		Year	1/1/05	12/31/05
	Browser Versions	Day	1/1/05	2/20/05
		Week	12/26/04	2/26/05
		Month	1/1/05	2/28/05
		Quarter	1/1/05	3/31/05
		Year	1/1/05	12/31/05
	Platforms	Day	1/1/05	2/20/05
		Week	12/26/04	2/26/05
		Month	1/1/05	

Users can select either HTML or Java to display a report in the SAS Web Analytics Report Viewer. If they want to perform new analyses or use new data from the Web mart, then they can run the decision support reports individually (without reprocessing all of the Web log data).

For details about navigating in the SAS Web Analytics Report Viewer, click **Help** in the SAS Web Analytics Report Viewer, or see the *SAS Web Analytics: User's Guide*.

Supplied Reports

The SAS Web Analytics Report Viewer supplies a set of standard reports. You can modify existing report definitions and can create custom reports by using the SAS Web Analytics Administrator.

The standard SAS Web Analytics reports appear by category in the navigation tree at the left of the SAS Web Analytics Administrator. For detailed descriptions of each report, see the *SAS Web Analytics: User's Guide*. For more information about using the SAS Web Analytics Administrator Administrator to set up reports and report groups, see Chapter 9, "Using the SAS Web Analytics Administrator to Manage Your Web Mart(s)," on page 111.

Table 1.7 Categories of SAS Web Analytics Reports

Report Category	Function	Types of Reports
Traffic	Monitors the volume of activity and reports the status codes that were generated for your Web site.	Visitor Browser and Platform Status Codes Navigation Overview Referrer
Scorecard	Determines which variables in the input data set have a statistically significant impact on the target metric.	Scorecard: Site Metrics
Dashboard	Alerts and notifies you of changes in business direction by displaying a trend of key performance indicators.	Dashboard: Site Metrics
Funnel	Shows how many visits viewed a specific sequence of pages.	Interactive Funnel Static Funnel
Path Analysis	Shows the different paths that visitors took to get from one page to another.	Entry Path Referrer Entry Paths Interactive Path Analysis
Segmentation	Creates a set of rules (segments) that help predict which visitors will return to the site.	Repeat Visitor—Totals Repeat Visitor—Averages

SAS Web Analytics reports help you perform the following tasks:

- □ determine which pages of your Web site are the most frequently visited
- □ determine which pages of your Web site that visitors find the most or the least interesting
- □ determine which pages of your Web site are buried too deeply for visitors to find
- □ determine the pages to which repeat visitors return, and determine whether they see the special offers that you are promoting in a campaign
- □ discover why potential purchasers abandon their transactions
- □ reduce business costs

Setting Up Reports

Use the SAS Web Analytics Administrator to set up default SAS Web Analytics reports and to create new reports (see “The Traffic Page” on page 123). The following sections describe the requirements for setting up the reports that you need to customize.

The Scorecard

The scorecard requires a single data set that contains a set of metric variables that are summarized by day. The default input data set is Summary.Daily_Total_Day. The required columns for this data set are the following variables:

- Date
- a target variable (the default target variable is visit count)
- one or more input variables

In order to produce the scorecard statistics for a given date, at least thirty continuous days of data must exist in the input data set (Summary.Daily_Total_Day, by default) in the time interval before that date.

The Dashboard

The dashboard requires a single data set that contains a set of metric variables that are summarized by day. The default dashboard uses the Summary.Daily_Total_Day data set, which contains only the daily metric data from the Web log. Additional metric variables can be added to the Summary.Daily_Total_Day data set or to a new daily data set that is created for the dashboard (see Chapter 12, “Dashboard Administration,” on page 207).

The dashboard is generated by code within the %WAETL macro. After the %WAETL macro summarizes all of the detail data, the default data set called Daily_Total_Day (located in the Summary library) is used as input to the dashboard. By default, the Summary.Daily_Total_Day data set contains values for all of the traffic-type metrics and health-type metrics for the Web site.

In order to produce the dashboard statistics for a given date, at least thirty continuous days of data must exist in the input dataset (Summary.Daily_Total_Day, by default) before that date.

The Segmentation Report

Segmentation analysis requires a persistent visitor ID because it tracks visitor action over a period of time. Creating customized versions of a segmentation analysis is not recommended if the visitor ID is defined by using the following methods:

- The visitor ID consists of the user agent and the IP address.
- A nonpersistent cookie is used as the visitor ID.

Path Analysis Reports

To set up an Interactive Path Analysis report, a user defines the following parameters for a specified path of interest in the SAS Web Analytics Report Viewer:

- the dates to use in the report
- a start page and/or an end page that define the limits of the specified path
- the number of paths to display
- the maximum length of the path, which can be 1 to 7 pages
- the type of visual chart to use to display the data
- the scheme to use to display the report

The Funnel Reports

A funnel report provides a detailed description of any sequential process on a Web site, such as a sequence of Web pages that are visited. For example, you can find the point at which visitors leave a particular path of Web pages. The information is given in the form of conversion and drop-off rates. You can also use this information to discover how effectively your Web site is moving traffic to a specific target page.

Two types of funnel reports are available: Static Funnel reports and Interactive Funnel reports. Static Funnel reports are created during the ETL process, and provide statistics for the standard date ranges (the most recent one-day, seven-day, and thirty-day intervals).

Interactive Funnel reports are created by using the SAS Web Analytics Report Viewer, and provide statistics for the date range that you specify in the calendar. Interactive Funnel reports can be either pre-defined or can be created as ad hoc reports.

For details about how to run a stored process that generates a funnel report, see "Creating a Stored Process for a New Report" on page 249.

Getting Started with SAS Web Analytics 5.2: Administration Tasks

Administrative tasks for using the SAS Web Analytics solution to analyze a Web site fall into the following categories:

1 setting up a new Web mart

2 customizing and processing data

3 setting up and customizing reports

The following tables show the general administrative tasks that you might perform in order to prepare an analysis-ready Web mart.

Table 1.8 Setting Up a New Web Mart

Step	How Often	Action	Application to Use
1	Once	Initialize an empty Web mart by calling the %WAETL macro with the INITIALIZE argument for the PROGRAM parameter. The Daily.sas program is created.	SAS session
2	As needed	Edit the properties of the Extract process (such as those specifying how to handle the requests from spider or robot clients, or how to count the page requests that result in an error code) for a Web mart.	SAS e-Data ETL Administrator
3	As needed	Edit the properties of your new Web mart by customizing SAS code.	SAS session
4	Once	Register the new Web mart so that the Web mart data is recognized by the SAS Web Analytics software (and appears in the SAS Web Analytics Report Viewer).	SAS Web Analytics Administrator

Table 1.9 Customizing and ETL Processing Tasks

Step	How Often	Action	Application to Use
5	As needed (optional)	Create and define new variables for the detail data set by editing its metadata in the Config.Waconfig data set. Set up new variables for summarizations by editing the metadata in the Config.Waadmsum data set.	SAS Web Analytics Administrator
6	As needed	If new variables are created, then perform a summarization (create new summary data sets) by calling the %WASUMTST macro with the WAREHOUSE argument for the PROGRAM parameter.	SAS session
7	As needed (daily)	Copy the Web server log files from the Web server to the location that is expected by the SAS e-Data ETL Extract process.	Windows operation or a batch file
8	As needed	Delete any data sets that were created from previous executions of Daily.sas (or any of the SAS Web Analytics ETL processes).	SAS session
9	As needed (daily)	Run the Daily.sas program.	SAS session
10	As needed (daily)	Move the processed Web logs to an archive location.	Windows operation or a batch file

Table 1.10 Setting Up and Customizing Reports

Step	How Often	Action	Application to Use
11	As needed	If you have defined new variables, then edit an existing report or define a new report to add the new variables to it and, if necessary, create a new report group.	SAS Web Analytics Administrator
12	As needed	If you have defined new report groups or new reports, then verify that the new report groups and the reports appear in the SAS Web Analytics Report Viewer.	SAS Web Analytics Report Viewer
13	As desired	Define new traffic reports.[1]	SAS Web Analytics Administrator
14	As desired	If you want to create a decision support report (scorecard, dashboard, or segmentation report), then provide the required parameters and run the %WADECIDE macro.	SAS session SAS Web Analytics Administrator SAS Web Analytics Report Viewer

[1] Execution of a customized or new report depends on the availability of data for the analysis.

CHAPTER

2

Creating a New Web Mart

Overview: Creating a New Web Mart

You can create a new Web mart by using either one of the following methods:

□ Call the %WAETL macro with the INITIALIZE argument for the PROGRAM parameter in the SAS Program Editor window.

□ Use the SAS e-Data ETL Administrator. If you use the SAS e-Data ETL Administrator to create a new Web mart, then you must also run the %WAETL macro with the INITIALIZE argument for the PROGRAM parameter (described in the previous method) so that the SAS Web Analytics software can locate the Web server log files and the other components of the new Web mart.

□ Copy an existing Web Mart and then edit the copy (see "Copying a Web Mart in the SAS Web Analytics Administrator" on page 117).

After you initialize a new Web mart and before you process any Web log data for the Web mart for the first time, you must register the Web mart. To register a new Web mart, see "Registering a New Web Mart" on page 24.

For information about editing the properties of a Web mart, see Chapter 3, "Customizing the Properties of a Web Mart," on page 27.

Creating a New Web Mart by Using the %WAETL Macro

To create a new Web mart, invoke the %WAETL macro by typing the following code in a SAS session:

```
%waetl(program=initialize,
       swamart=newwebmartpath
       );
```

PROGRAM specifies the %WAETL macro function that you request. Specify the argument INITIALIZE in order to set up a new SAS Web Analytics Web mart. Although the argument must match the spelling as shown here, it is not case-sensitive.

SWAMART specifies the path to the root directory of the SAS Web Analytics Web mart that you are setting up. Because the %WAETL macro uses this information to create the directory that you have specified, the directory must not already exist.

For more about the %WAETL macro, see “The %WAETL Macro” on page 270.

The Functions of the %WAETL Macro

The %WAETL macro (with the INITIALIZE argument for the PROGRAM parameter) performs the following functions:

- Creates a new directory for the new Web mart from the argument that you specify for the SWAMART parameter. The name of the new directory is the name of your new Web mart.
- Creates the default set of subdirectories that must exist before it can load data into the new Web mart.

  Note: If you want a customized directory structure for your Web mart, then you must manually create it. △

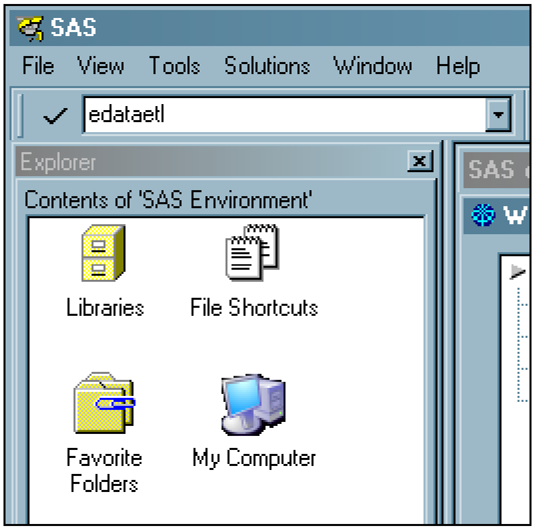
- Creates the metadata that will control the standard operations in the Web mart.
- Creates the Daily.sas program (and other programs) that will help you use the Web mart in the future. The Daily.sas program, which is created in the **sas** directory, is a program that you can use for all subsequent invocations of the %WAETL macro for the Web mart that you have just created.

Creating a New Web Mart by Using the SAS e-Data ETL Administrator

Follow these steps to create a new Web mart by using the SAS e-Data ETL Administrator:

1 To open the SAS e-Data ETL Administrator main window, type **edataetl** on the SAS command line and press ENTER.

Display 2.1 SAS Command Line

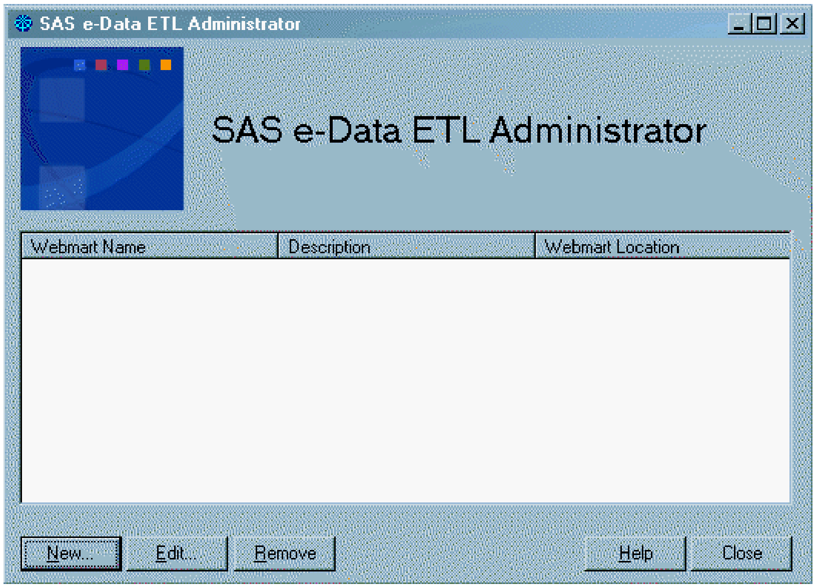


edataetl

SAS
File View Tools Solutions Window Help
Explorer
Contents of 'SAS Environment'
Libraries
File Shortcuts
Favorite Folders
My Computer
SAS

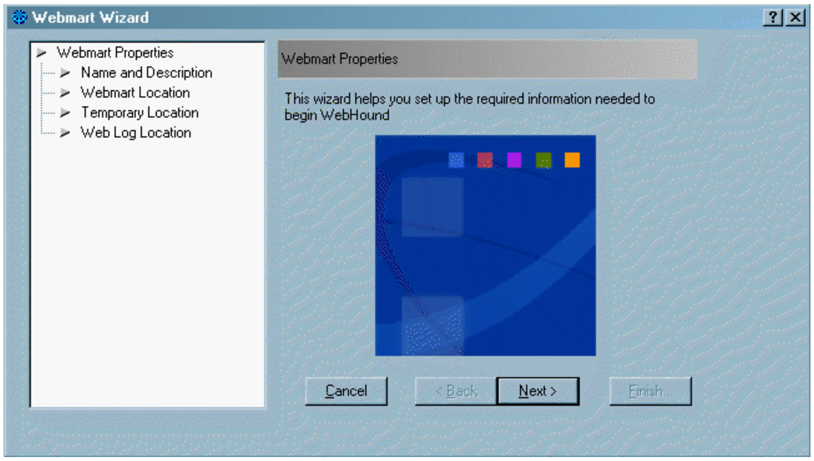
The SAS e-Data ETL Administrator main window appears.

Display 2.2 SAS e-Data ETL Administrator Main Window with No Web Marts Defined



2 To define a new Web mart, select **New** from the bottom of the SAS e-Data ETL Administrator main window. The Webmart Wizard window opens.

Display 2.3 Introductory Window of the SAS e-Data ETL Webmart Wizard

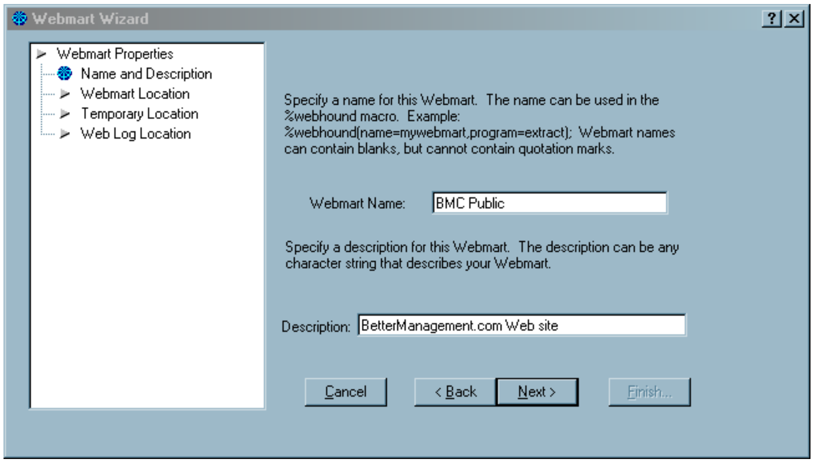


This wizard helps you set up the required information needed to begin WebHound

3 From the Webmart Wizard window, select **Next**.

4 For the **Name and Description** property, enter a name and a description for the Web mart that you are building. Select **Next** to continue defining the Web mart.

Display 2.4 Webmart Wizard: Name and Description Step



Specify a name for this Webmart. The name can be used in the %webhound macro. Example: %webhound(name=mywebmart,program=extract); Webmart names can contain blanks, but cannot contain quotation marks.

Webmart Name: BMC Public

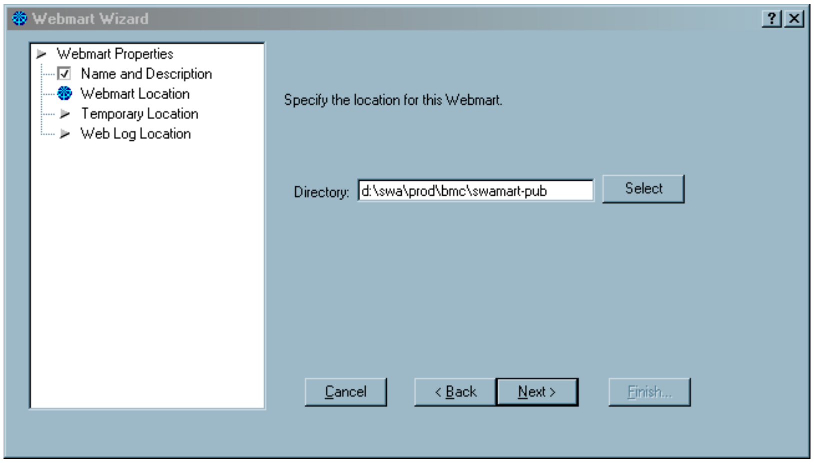
Specify a description for this Webmart. The description can be any character string that describes your Webmart.

Description: BetterManagement.com Web site

5 For the **Webmart Location** property, type the pathname of the directory that will contain the Web mart, or use the **Select** button to select an existing directory. An example directory path is **d:\swa\prod\bmc\swamart-pub**.

Note: You cannot change this directory path after exiting the Webmart Wizard. △

Display 2.5 Webmart Wizard: Webmart Location Step



Specify the location for this Webmart.

Directory: d:\swa\prod\bmc\swamart-pub

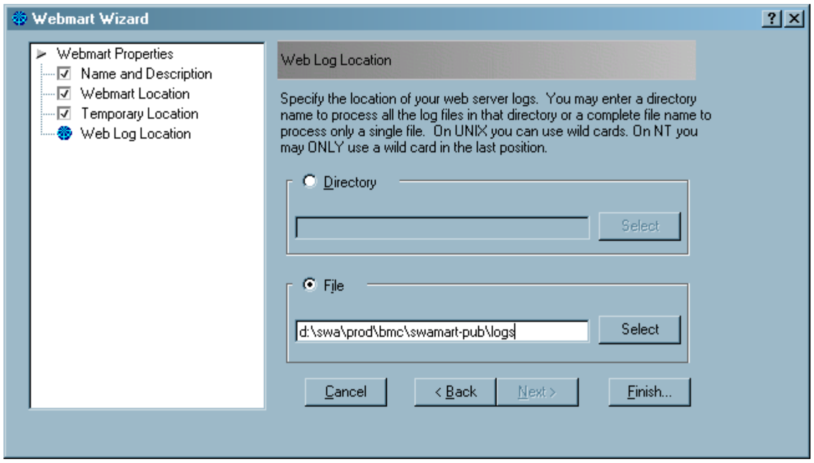
After choosing the directory for your Web mart, select **Next**.

6 For the **Temporary Location** property, the SAS e-Data ETL Administrator provides a default location for the temporary files. However, you can change this location by entering a new directory path or by using the **Select** button to select an existing directory. Be sure that the temporary file location provides enough disk space for the intermediate tables that will be created during SAS e-Data ETL processing.

After accepting the default location or entering the directory path for your temporary files, select **Next**.

7 For the **`Web Log Location`** property, type the location of the Web logs to be processed. The SAS e-Data ETL program can process a single file or all the files within a specified directory. To process all the files in a directory, select the **`Directory`** option. To process a specific file in the directory, select the **`File`** option. For either option, enter a directory (or a file) either by typing its pathname or by using the **`Select`** button to select the directory.

Display 2.6 Webmart Wizard: Web Log Location Step



Web Log Location

Specify the location of your web server logs. You may enter a directory name to process all the log files in that directory or a complete file name to process only a single file. On UNIX you can use wild cards. On NT you may ONLY use a wild card in the last position.

○ Directory

● File: d:\swa\prod\bmc\swamart-pub\logs

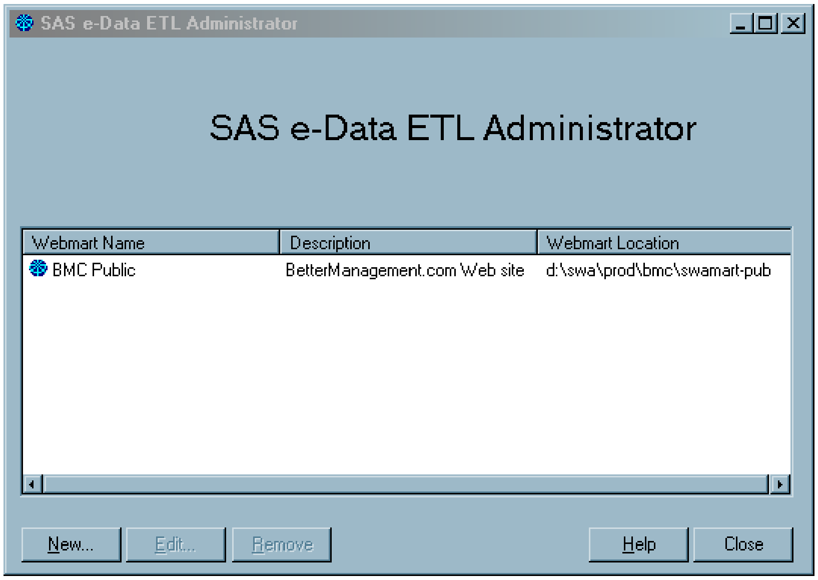
Enter the Web log location by either typing its path or by using **`Select`**. You can use the wildcard character (*) when specifying a location for Web log files in a UNIX or Windows operating environment. However, the asterisk is valid only in the last part of the directory or filename in the Windows environment.

Note: To process a single Web log file, select the **`File`** option. △

8 Select **`Finish`** to complete the creation of a new Web mart. SAS e-Data ETL software creates the directories that are required for the Web mart. Click **`OK`** to close the window and exit the Webmart Wizard.

The newly defined Web mart now appears in the SAS e-Data ETL Administrator main window.

Display 2.7 SAS e-Data ETL Administrator Main Window Displaying a Newly Defined Web Mart



Webmart Name	Description	Webmart Location
BMC Public	BetterManagement.com Web site	d:\swa\prod\bmc\swamart-pub

Note: If you create a new Web mart using the SAS e-Data Administrator, then you must also initialize the new Web mart. See “Creating a New Web Mart by Using the %WAETL Macro” on page 19 for specific instructions about initializing a new Web mart. △

Registering a New Web Mart

After you initialize a new Web mart and before you process any Web log data for the Web mart for the first time, you must register the Web mart. When you register a Web mart, the Web mart is recognized by the SAS Web Analytics software and is listed in the SAS Web Analytics Report Viewer.

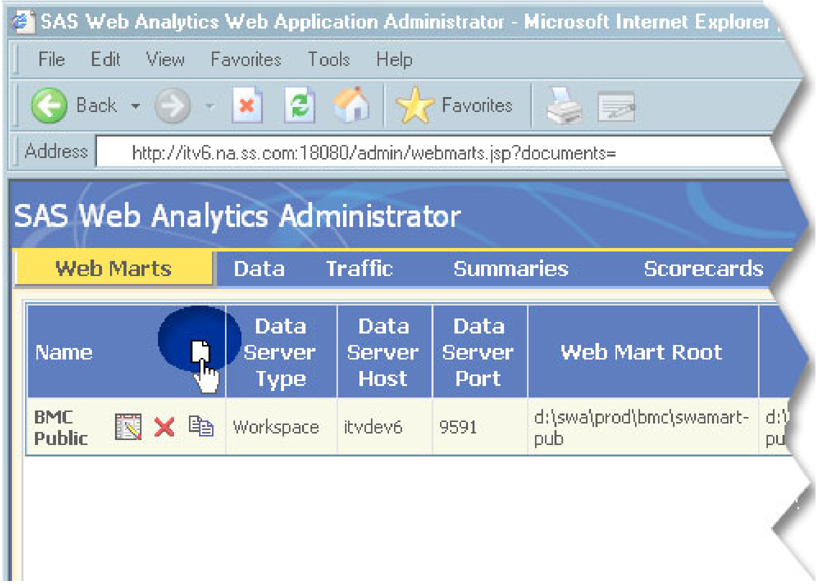
Use the SAS Web Analytics Administrator to register a new Web mart by following these steps:

1 Open the SAS Web Analytics Administrator in your browser by selecting the URL that was established during setup.

2 Select the **`Web Marts`** link at the top of the page.

3 Click 🗋 (see the following display) in the first row of the table that appears.

Display 2.8 Registering a New Web Mart

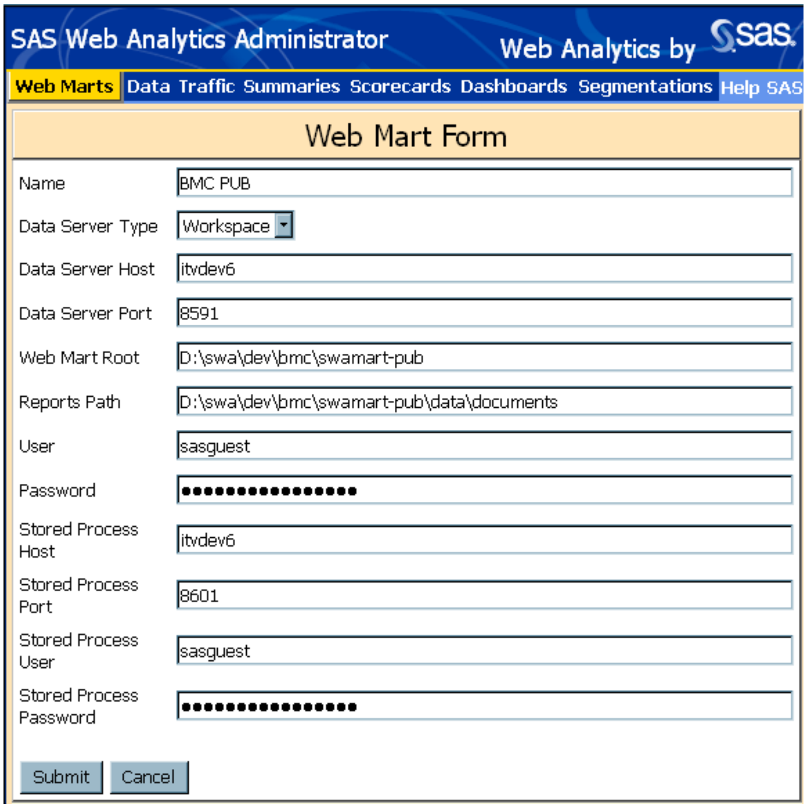

SAS Web Analytics Web Application Administrator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Favorites
Address
SAS Web Analytics Administrator
Web Marts
Data
Traffic
Summaries
Scorecards

http://itv6.na.ss.com:18080/admin/webmarts.jsp?documents=

Name		Data Server Type	Data Server Host	Data Server Port	Web Mart Root	
BMC Public		Workspace	itvdev6	9591	d:\swa\prod\bmc\swamart-pub	d:\ pu

The Web Mart Form page appears.

Display 2.9 The Web Mart Form for Registering a New Web Mart



Web Mart Form

Name	BMC PUB
Data Server Type	Workspace
Data Server Host	itvdev6
Data Server Port	8591
Web Mart Root	D:\swa\dev\bmc\swamart-pub
Reports Path	D:\swa\dev\bmc\swamart-pub\data\documents
User	sasguest
Password	●●●●●●●●●●●●●●●●●
Stored Process Host	itvdev6
Stored Process Port	8601
Stored Process User	sasguest
Stored Process Password	●●●●●●●●●●●●●●●●●

4 Consult with the authorities in your organization who are familiar with the configuration of the SAS solution on your server(s) to provide the information in the Web Mart Form page.

5 Click **`Submit`** to register the new Web mart.

CHAPTER

3

Customizing the Properties of a Web Mart

Overview: The Properties of a SAS Web Analytics Web Mart

To customize the properties of a Web mart (such as the location of the Web log(s) and the amount of data to store), use the SAS e-Data ETL Administrator. The SAS e-Data ETL Administrator is the Windows interface of the SAS e-Data ETL software.

After you select a Web mart and click **Edit** in the SAS e-Data ETL Administrator main window, the Properties window for the selected Web mart opens. In the Properties window, select any property from the tree view. Edit the components in the property frame that opens to the right of the tree view.

The following table briefly describes each of the properties of a Web mart that you can edit through its corresponding property frame in the Properties window. For detailed information about editing a property, see the corresponding subsection within this section.

Table 3.1 Property Frames of the Properties Window

Property Frame	Description
Webmart Location	Enables you to view the parent directory for the Web mart. This path is the location that you specified in the Webmart Wizard. You cannot change this directory path.
Detail Tables	Enables you to view and edit the number of Weblog_Detail, CGI_Parms, ReferrerParms, Pathing, and Cookie tables that are created and the amount of historical data that is kept for each table.
Temporary Location	Enables you to provide the location for the temporary files that are created and used during processing.
Web Log Location	Enables you to specify the location of your Web logs.
Processing	Enables you to specify what information to extract from your Web logs, define the length of a session, indicate which INFILE statement options to use with various Web log types, list the SAS session options to use during processing, and activate or deactivate DNS lookup.
Parsing	Enables you to specify how URLs are parsed, add new CGI program locations, and add new browsers or platforms that should be recognized.
Filtering	Enables you to specify what information from the Web log (such as special clients, spiders and robots, non-page requests, and bad status codes) should be included or excluded from processing.
Compressed Files	Enables you to identify whether your Web server logs files are compressed and to specify where the uncompressing executable file is located.
Execution Tuning	Enables you to modify processing parameters that affect performance.
Advanced Customizations	Enables you to copy the CAM source entries in the catalogs for SAS e-Data ETL processing (for example, the Wbetl, Wbmacros, Wbreptex, and Wbrephlp catalogs) to the Usermods library for editing.

Although the default values will work in many situations, you can use the Properties window to customize most of the properties of a Web mart for your specific needs.

Saving Your Changes

Select **OK** at any point in the process to save all changes that you have made during your editing session in the Web mart Properties window. After saving these changes, you return to the SAS e-Data ETL Administrator main window.

Canceling Your Changes

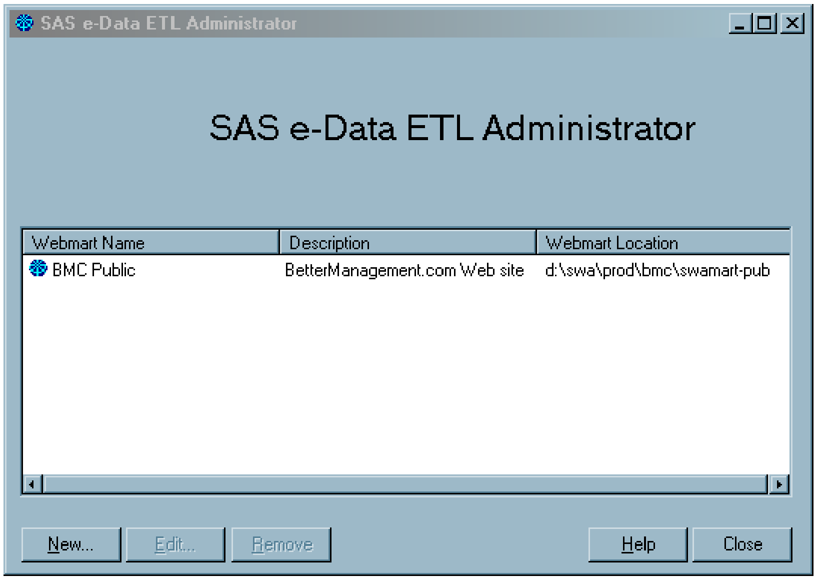
Select **Cancel** at any point to cancel all changes that you have made during your editing session in the Web mart Properties window. After canceling these changes, you return to the SAS e-Data ETL Administrator main window.

The Properties Window of the SAS e-Data ETL Administrator

Selecting a Web Mart

To select a Web mart, begin from the SAS e-Data ETL Administrator main window.

Display 3.1 SAS e-Data ETL Administrator Main Window Displaying a Web Mart Named BMC-Public



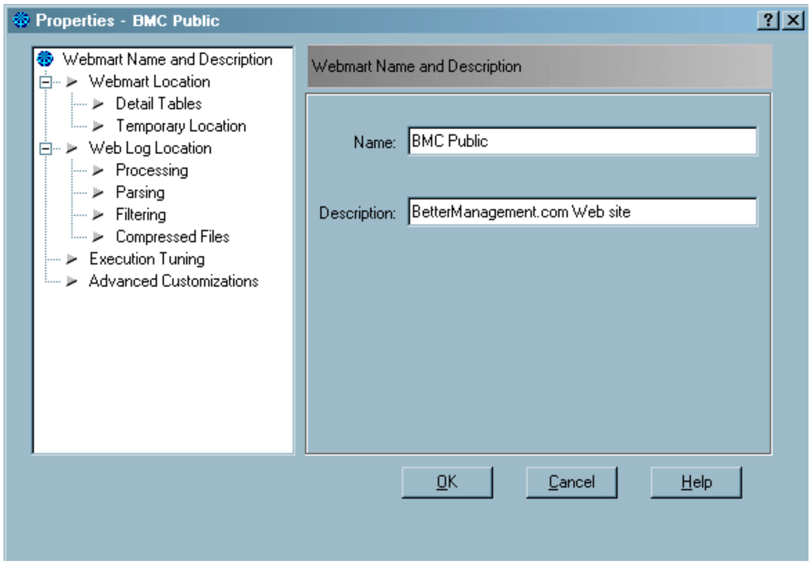
Webmart Name	Description	Webmart Location
BMC Public	BetterManagement.com Web site	d:\swa\prod\bmc\swamart-pub

In the Webmart Name column, highlight the name of the Web mart whose properties you want to customize, and click **`Edit`**.

Note: If you do not see one or more Web marts listed in the SAS e-Data ETL Administrator main window, then you have not set up a Web mart. To set up a new Web mart, click **`New`** and follow the instructions (see “Creating a New Web Mart by Using the SAS e-Data ETL Administrator” on page 20). △

The Properties - <*Web mart name*> window opens. The properties of the Web mart are displayed in the tree view. The name and description of the selected Web mart are displayed in the frame on the right.

Display 3.2 Properties Window of the BMC Public Web Mart



Name: BMC Public

Description: BetterManagement.com Web site

Customizing a Single Property

To customize a single property of a Web mart, click a property from the tree view. When the property frame opens, make the modifications that are needed. If this is the last property that you want to customize, then select **OK**.

Customizing Multiple Properties

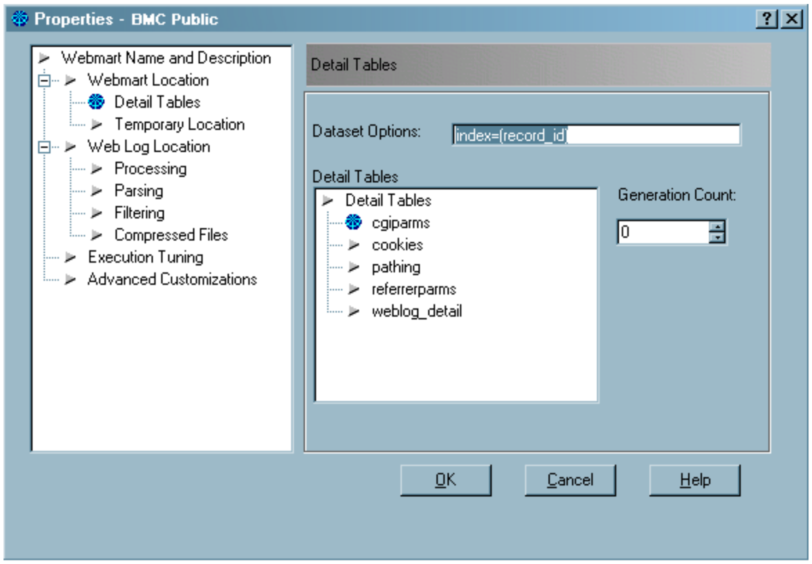
You can customize multiple properties of a Web mart. When you finish configuring a property, instead of selecting **OK**, select the next property to be customized from the tree view. When you finish making all your changes to the properties, select **OK**. The Web mart properties are saved all at once, and you return to the SAS e-Data ETL Administrator main window.

The Detail Tables Properties

The number of detail tables that are created during processing and the amount of historical data to be retained are important properties of a Web mart. These properties are specified in the detail tables (which are also called SAS detail data sets). The Detail Tables frame contains the following fields:

- a list of the detail table names
- **Generation Count**, which specifies the number of generations to keep for the highlighted detail table
- **Dataset Options** (not a required field), which enables you to specify options such as an index

Display 3.3 The Detail Tables Frame of the Properties Window



Dataset Options: index=(record_id)

Generation Count: 0

Navigating to the Detail Tables Frame

From the tree view on the left side of the Properties - <*Web mart name*> window, select **Detail Tables** under **Webmart Location**. The Detail Tables frame appears to the right of the tree view.

Data Set Options for the Detail Tables

Any data set options that the user enters are applied to the detail tables.

To increase efficiency, you can join one detail table to another detail table by using the **Dataset Options** field to set a common index, such as Record_ID, for each of the tables. To index a table by the variable Record_ID, type the following code in the **Dataset Options** field of the Detail Tables frame for that detail table:

```
index=(record_id)
```

Descriptions of the Detail Tables

The following table gives general descriptions of each of the six detail tables that are created by the SAS e-Data ETL application. The significance and default values of the modifiable fields in each table are also listed.

Table 3.2 Descriptions of the Detail Tables

Table Name	Description	Defaults for Key Fields
CGIParms	The CGIParms table contains name-value pairs that have been parsed from the query parameters in the original record of the Web server log file. The CGIParms table also contains a Record_ID field that you can use to match this information with its source URL.	The default value, 0, for **`Generation Count`** indicates that SAS e-Data ETL software will not keep any generations of CGIParms tables. The supplied default entry for **`Dataset Options`** enables the software to index this table by Record_ID.
Cookies	The Cookies table contains name-value pairs for the information that has been parsed from the Cookies field in the Web log. It also contains a Record_ID field to match this information in the CGIParms and Weblog_Detail tables.	The default value, 0, for **`Generation Count`** indicates that the SAS e-Data ETL software will not keep any tables. The supplied default entry for **`Dataset Options`** enables the software to index this table by Record_ID.
Pathing	The Pathing table contains one entry for each session that is held during a particular time period. The record for the entry contains the Session_ID and the duration of the session. Its datetime value is picked up from the date/time field of the first Web log record in the path. The Clickpath field contains a series of numbers that can be translated back to respective URLs by using the Unique_URLs table.	The default value, 3, for **`Generation Count`** indicates that SAS e-Data ETL software will keep three generations of the Pathing table after each run of the Load process.
ReferrerParms	The ReferrerParms table contains name-value pairs that have been parsed from the Referrer field in the original Web server log record. The ReferrerParms table provides you with a list of the keywords that are used by visitors at referring search engine sites. This table also contains a Record_ID field that you can use to match this information with its source URL.	The default value, 3, for **`Generation Count`** indicates that SAS e-Data ETL software will keep three generations of the ReferrerParms table after each run of the Load process. The supplied default entry for **`Dataset Options`** enables SAS e-Data ETL software to index this detail table by Record_ID.

Table Name	Description	Defaults for Key Fields
Weblog_Detail	The Weblog_Detail table contains the same information as the input Web server data, but the information is contained in SAS tables and is sorted according to the session identifier. A Weblog_Detail table is generated every time you run the SAS e-Data ETL Load process. The Weblog_Detail table also contains a Record_ID field that can be used to match this information with information in the other detail tables.	The default value, 3, for **Generation Count** indicates that SAS e-Data ETL software will keep three generations of the Weblog_Detail table after each run of the Load process.
Unique_URLs	The Unique_URLs table contains all the URLs that are referenced in the retained Pathing tables. A Unique_URLs table is generated every time you run the SAS e-Data ETL Load process.	The default value is the URL for all page views minus any parameters. For example, the default value for **http://www.orionstar.com/header.asp?loc=man=5GDKLJQLG6S92GFE00AKHBAXFJSP3UVE** is **http://www.orionstar.com**.

Calculating the Number of History Tables to Keep

For the **Generation Count** field, type the number of generations of detail tables that you want to keep. Use the following formula to determine the number of generations to keep:

```
Days of detail you want to keep * Number of times you run the Load process per day
```

This formula applies to all of the detail tables.

Example calculation: If the Load process is being run once a day, and if you want nine days of Pathing data for reporting and analysis, then the **Generation Count** is 9.

Note: Consider the amount of disk space that is available when you select the number of generations of detail tables to keep. Depending on the size of your Web log, these detail tables might require a large amount of disk space. △

The Temporary Location Properties

The Temporary Location frame lists the directory path for the temporary files that are created during processing. When you initially open this window, the location that you see is the path that is provided in the Webmart Wizard.

Navigating to the Temporary Location Frame

From the tree view in the Properties - <*Web mart name*> window, select **Temporary Location**. The Temporary Location frame appears to the right of the tree view.

Changing the Path for Temporary Files

You can change the path for temporary files by entering a new directory path or by using the **`Select`** button to select an existing directory. Be sure that the temporary location provides enough disk space for the intermediate tables that will be created during processing.

Deleting Temporary Files

To remove the temporary files that are created during the Extract and Load processes, select the **`Delete temporary files after loading web log files`** check box. If you do not select this check box, then the files in this temporary directory are overwritten.

The Web Log Location Properties

In the Web Log Location frame, you specify the location of the Web logs to be processed.

You can use the wildcard character (*) when specifying a location for Web log files in a UNIX or Windows operating environment. However, the asterisk is valid only in the last part of the directory or filename in the Windows environment.

Navigating to the Web Log Location Frame

Select **`Web Log Location`** from the tree view in the Properties - <*Web mart name*> window. The Web Log Location frame appears to the right of the tree view.

Setting the Web Log Location

SAS e-Data ETL software can process either a single file or all the files that are contained within a specified directory. To process all the files in a directory, select the **`Directory`** option. To process a specific file in the directory, select the **`File`** option. For either option, enter the directory or filename either by typing its pathname or by using the **`Select`** button to select the directory.

CAUTION:

SAS e-Data ETL software processes all files in the directory that you specify for Web Log Location, whether or not they are Web server log files. Therefore, be sure that Web server log files are the only files in the Web Log Location directory. △

Note: The **`File`** option is a good choice when you set up a Web mart for the first time because it enables you to test a small subset of your Web log data. △

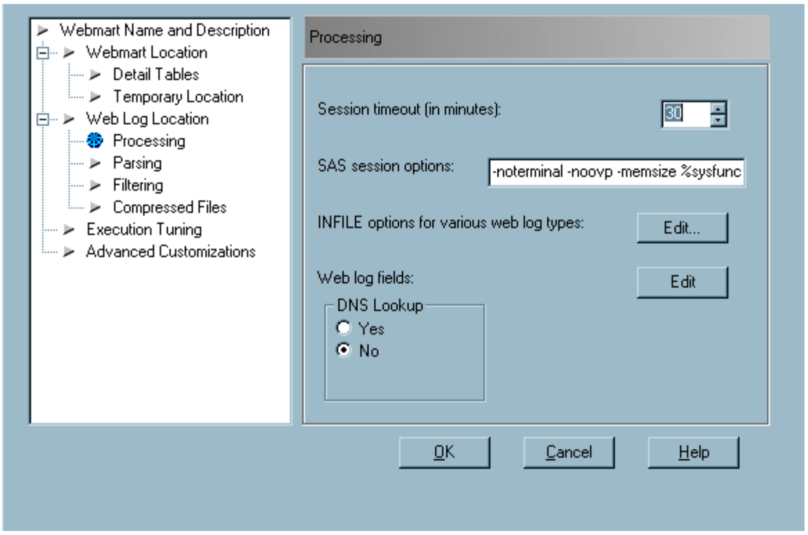
The Processing Properties

In the Processing frame, you can specify the following properties:

□ the length of a SAS e-Data ETL session's timeout limit (in minutes)

- □ the SAS system options that are used during processing
- □ the INFILE options that are used for various types of Web server log files
- □ the fields or variables to extract from the Web server log files
- □ the option of converting a numeric IP address to the equivalent domain name (DNS lookup) during the Extract process

Display 3.4 The Processing Frame of the Properties Window



Session timeout (in minutes): 30

SAS session options: -noterminal -noovp -memsize %sysfunc

INFILE options for various web log types:

Web log fields:

DNS Lookup

○ Yes

● No

Navigating to the Processing Frame

From the tree view in the Properties - <*Web mart name*> window, select **`Processing`** under **`Web Log Location`**. The Processing frame (shown in the preceding display) appears to the right of the tree view.

Session Timeout

Definition of a Session

A session (also called a *visit*) is the period of time that a visitor spends at a Web site. A session begins when a visitor initially requests a page on the Web site. The session ends either when the visitor exits the Web site or when the session timeout limit, shown in the **`Session timeout (in minutes)`** field in the Processing frame, is exceeded before the visitor makes another request. The default session timeout limit is 30 minutes. Any visitor activity that continues beyond this limit is logged as a new or unique session.

How Sessions Are Split

Web server log files are generally copied and restarted once a day. When the Web server log files are processed, it is assumed that all sessions end at the end of the time span that is covered by each Web server log file.

For example, a Web server stops writing to Tuesday's log file at 1:00 a.m. each Wednesday. If a visitor browses the site at 12:55 a.m. and continues browsing until 1:05 a.m., then the activity is processed as two sessions.

Preserving the Continuity of One Session across Multiple Servers

To be sure that SAS e-Data ETL software correctly processes a single session that is spread across multiple servers (and is thereby recorded in multiple Web server log files), process all the Web server log files in the same run of the Extract process.

SAS Session Options

This property refers to the SAS system options that are used when invoking the concurrent SAS sessions for your parallel jobs. For more information about the SAS system options that you can use, see SAS Help and Documentation.

INFILE Options for Various Web Log Types

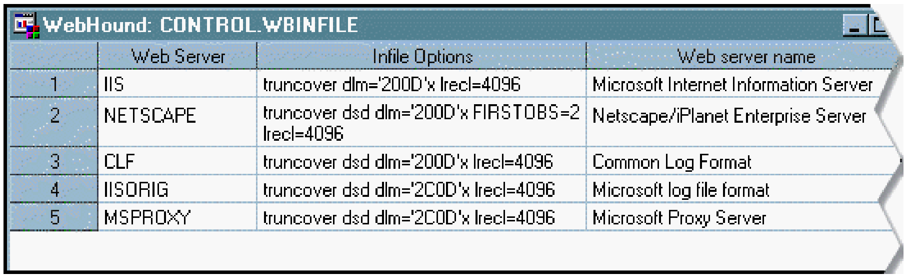
The INFILE options property enables you to specify the exact options for the SAS INFILE statement when the Web logs are read. An INFILE statement identifies an external file containing data that you want to read. It opens the file for input or, if the file is already open, makes it the current input file. This means that subsequent INPUT statements are read from this file until another file is made to be the current input file.

SAS e-Data ETL software can use one or more entries in this statement to serve as a template for reading the new Web log file type. See SAS Help and Documentation for more about the INFILE statement before making changes to the INFILE options property.

Changing the INFILE Option in the Control.Wbinfile Table

To set the INFILE options for processing Web logs, select the **`Edit`** button that is next to **`INFILE options for various web log types`** in the Processing frame. The Control.Wbinfile table opens for you to edit as shown in the following display.

Display 3.5 Control.Wbinfile Table



	Web Server	Infile Options	Web server name
1	IIS	truncover dlm='200D'x lrecl=4096	Microsoft Internet Information Server
2	NETSCAPE	truncover dsd dlm='200D'x FIRSTOBS=2 lrecl=4096	Netscape/iPlanet Enterprise Server
3	CLF	truncover dsd dlm='200D'x lrecl=4096	Common Log Format
4	IISORIG	truncover dsd dlm='2C0D'x lrecl=4096	Microsoft log file format
5	MSPROXY	truncover dsd dlm='2C0D'x lrecl=4096	Microsoft Proxy Server

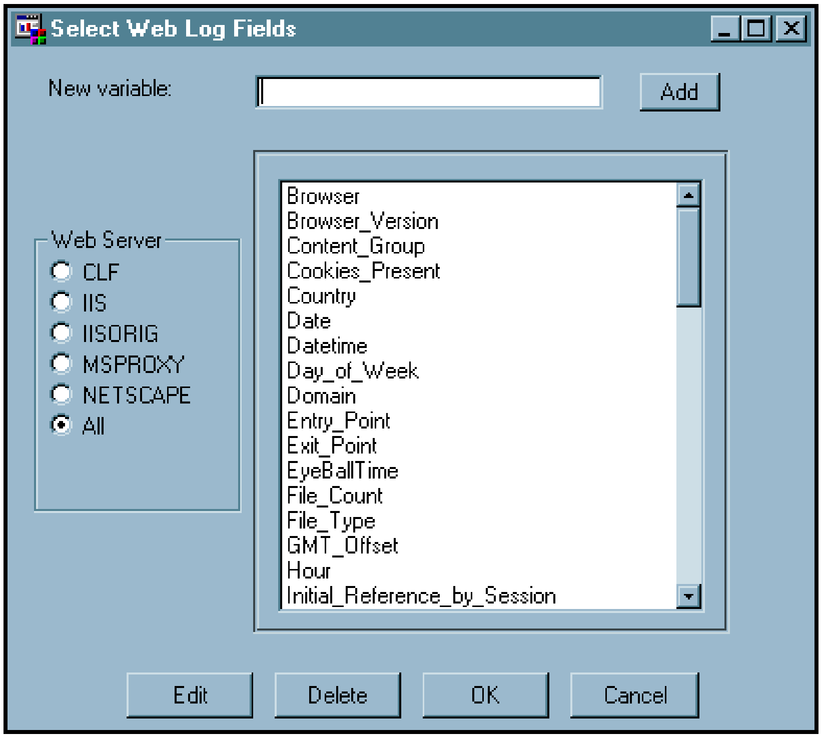
When you have finished making your modifications, close the table. In the dialog box that appears, click **`Yes`** to confirm any changes that you made to the table or click **`No`** to cancel any modifications that you made. The Processing frame re-appears.

Adding or Deleting Fields in Your Web Server Log File

To add or delete the field variables that are extracted (by default) from your Web server log files, select the **`Edit`** button that is next to **`Web log fields`** in the Processing frame.

The major types of Web server log files that are used in the industry are listed on the left side of the Select Web Log Fields window under **`Web Server`**. When you select a particular type of Web server log file, its default fields are listed in the right pane.

Display 3.6 The Select Web Log Fields Window



Select Web Log Fields

New variable:

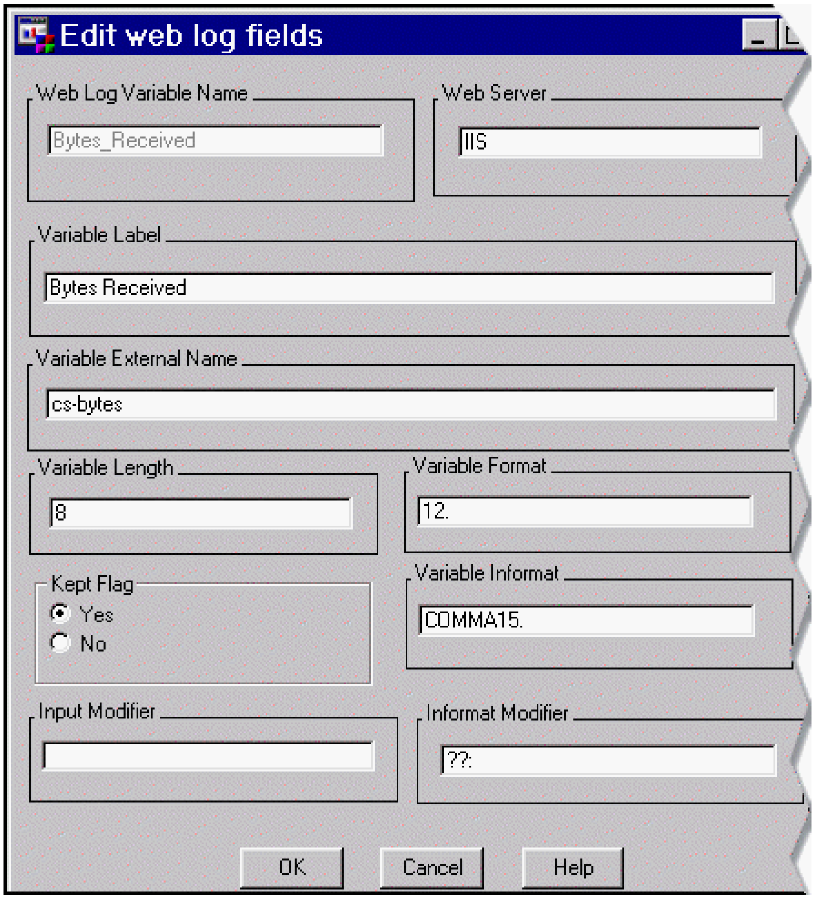
Web Server

○ CLF
○ IIS
○ IISORIG
○ MSPROXY
○ NETSCAPE
● All

Browser
Browser_Version
Content_Group
Cookies_Present
Country
Date
Datetime
Day_of_Week
Domain
Entry_Point
Exit_Point
EyeBallTime
File_Count
File_Type
GMT_Offset
Hour
Initial_Reference_by_Session

You can edit each of the fields for your type of Web server log file. After you select the type of Web server log file, highlight the field variable that you want to edit from the list. Select **`Edit`** to open the Edit Web Log Fields window, as shown in the following display.

Display 3.7 The Edit Web Log Fields Window



Web Log Variable Name: Bytes_Received

Web Server: IIS

Variable Label: Bytes Received

Variable External Name: cs-bytes

Variable Length: 8

Variable Format: 12.

Kept Flag: ● Yes ○ No

Variable Informat: COMMA15.

Input Modifier:

Informat Modifier: ??:

The following table lists the field variable parameters that you can edit in a Web server log file.

Table 3.3 Parameters of a Field Variable in a Web Server Log File

Parameter	**Description**
Web Log Variable Name	Enables you to alter the variable names in a Web log to fit the SAS syntax for variable names. Some variable names in Web server log files (those names that contain a hyphen, for example, such as cs-bytes) are not valid SAS variable names. In this case, you could use the Web Log Variable Name field to specify that the variable name cs-bytes should correspond to Bytes_Received.
Web Server	Indicates the software that handles the requests for content that are received from visitors to your Web site.
Variable Label	Indicates the text string that appears instead of the variable name in a report such as a PROC PRINT listing.
Variable External Name	Indicates the original name of the Web server log file variable.

Parameter	Description
Variable Length	Indicates the number of bytes of physical storage that each value of a variable uses in each observation of a SAS data set.
Kept Flag	Indicates whether or not this variable is stored in the data sets that are created. If you do not want to extract any data for this field variable from the Web server log file, then set **`Kept Flag`** to **`No`**.
Input Modifier	Used in an INPUT statement to position the pointer in the input buffer. You can specify a column position or a search string. The input modifier is placed before the variable name in the INPUT statement. Input modifiers can be specified for variables in the Wbfields metadata table to build the INPUT statement that is used to read a particular Web server log file. For example, the input modifier **`@10`** places the pointer in column 10. The input modifier **`@ 'S_UA'`** places the pointer after the first occurrence of the string "S_UA."
Variable Format	Used in a FORMAT statement to control how SAS displays a variable value. For example, if you want a number to appear with commas, you can use the COMMA*w.d* format, where *w* represents the total field width and *d* represents how many decimal places are displayed.
Variable Informat	Used in an INPUT statement to control how SAS creates a variable's value (that is, a reading instruction). For example, to read a number that contains commas or certain other special characters, use the COMMA*w.* format, where *w* represents the number of columns to read.
Informat Modifier	Used in an INPUT statement in conjunction with an informat to control how SAS creates a variable's value. The informat modifier is placed between the variable name and the informat. For example, use a colon (:) to indicate that SAS should use the LIST style of input to capture a value from the input, and then use the informat to follow it as described previously to create the variable's value. Informat modifiers can be specified for variables in the Wbfields metadata table to build the INPUT statement that is used to read a particular Web server log file.

About DNS Lookup

Every Web server uses Domain Name Services (DNS) to resolve language-based domain names such as **`godfrey.ibm.com`** into a 32-bit numeric Internet Protocol (IP) address written as four numbers separated by periods, such as 9.50.115.89. Each of the four numbers that form a unique IP address on the Internet must be between 0 and 255. DNS uses IP addresses to identify a specific network and a host on that network.

If you activate DNS lookup, then SAS e-Data ETL software uses a process called *reverse domain name resolution* to resolve (convert) visitor IP addresses into their corresponding user-friendly domain names. A domain name can identify one or more IP addresses. You can use domain names to determine the origin of visitors, either friendly or competitive, to your Web site.

Using the DNS Lookup Script

To avoid slower performance if you elect to resolve visitor IP addresses, use the DNS lookup script (supplied with SAS e-Data ETL software) *outside* SAS e-Data ETL processing. The DNS lookup script makes a copy of each Web server log file that contains both the IP addresses of visitors and their converted domain names. Use these copies as input to the Extract process and archive them to keep a record of the most current domain names that are associated with visitor IP addresses.

The Parsing Properties

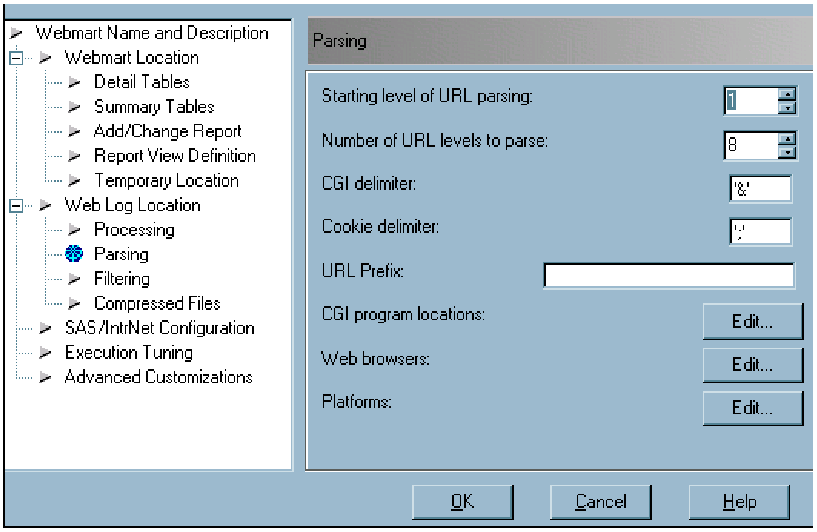
You can edit these properties for parsing your Web server log files:

- □ starting level for URL parsing
- □ number of URL levels to parse
- □ CGI delimiters to use for parsing
- □ cookie delimiters to use for parsing
- □ URL prefix
- □ location of executable programs and directories
- □ list of browsers that the SAS e-Data ETL application recognizes
- □ list of platform types that the SAS e-Data ETL application recognizes

Navigating to the Parsing Frame

From the tree view in the Properties - <*Web mart name*> window, select **`Parsing`** under **`Web Log Location`**. The Parsing frame (shown in the following display) appears to the right of the tree view.

Display 3.8 The Parsing Frame in the Properties Window



Starting level of URL parsing: 1

Number of URL levels to parse: 8

CGI delimiter: '&'

Cookie delimiter: ';'

URL Prefix:

CGI program locations:

Web browsers:

Platforms:

Starting Level of URL Parsing

The URL level equates to the logical directory structure of the Web server, which is delimited by a forward slash (/). Some Web sites might have the same values for the first few levels of all their Web pages, such as **/web/html/pages/**. In such cases, setting the starting level of URL parsing to 4 would exclude the first three common levels from processing. The default value for the starting level is 1.

Number of URL Levels to Parse

This property indicates how many directory levels to parse from a URL. For example, the URL **/web/html/pages/index.htm** has four levels. When you set the number of URL levels, remember that lengthy URLs might require additional horizontal space for display of each additional level in the output of a report. The default (and maximum) value for the number of levels to parse is 8.

CGI Delimiter

The CGI delimiter refers to the character(s) that separate the distinct name-value pairs in a URL. The ampersand (&), which is this product's default value, is most often used as the CGI delimiter. However, the international standard for URLs does allow other characters to be used as the CGI delimiter.

Note: When entering a CGI delimiter character into this field, enclose it in single quotation marks ('). △

Cookie Delimiter

The cookie delimiter refers to the character(s) that separate the distinct cookie name-value pairs when the fields that contain cookie values exist. The semicolon (;) is most often used as a delimiter. SAS e-Data ETL software uses the semicolon as its default. However, there are currently no standards for delimiters.

Note: When you type a cookie delimiter into this field, enclose it in single quotation marks ('). △

URL Prefix

The URL prefix consists of the URL scheme and authority (such as **http://www.sas.com**) that are used to prefix the requests that are logged by your Web server. Web servers that use standard Web log formats do not store the URL scheme and authority for individual requests, so you must supply this information to the SAS e-Data ETL application separately.

This value is stored in the URL_Prefix field of the Control.Wbconfig table. You can enter only one value in this field. The value that you specify in the URL_Prefix field will be prefixed to all the requests in your Web log.

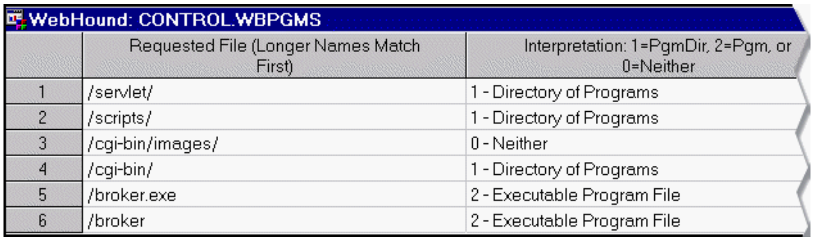
In some cases, one URL prefix might not be sufficient. You might have multiple logical Web sites that are served by one Web server. As a result, you might want to prefix some requests with **http://www.site1.com** and the remaining requests with **http://www.site2.com**.

If one URL prefix will not meet your site's needs, then you must modify the code in the User_assignments_before_output custom access module (CAM) to set the value of the URL prefix based on other information in the request. For example, your code can use the value of the Requested File field in the Web log record to determine which Web site contains the requested file. After you determine the Web site, then you can set the URL prefix.

CGI Program Locations

The Control.Wbpgms table contains the locations of executable programs and directories. Select the `Edit` button that is next to `CGI program locations` to open the Control.Wbpgms table.

Display 3.9 Control.Wbpgms Table


WebHound: CONTROL.WBPGMS

	Requested File (Longer Names Match First)	Interpretation: 1=PgmDir, 2=Pgm, or 0=Neither
1	/servlet/	1 - Directory of Programs
2	/scripts/	1 - Directory of Programs
3	/cgi-bin/images/	0 - Neither
4	/cgi-bin/	1 - Directory of Programs
5	/broker.exe	2 - Executable Program File
6	/broker	2 - Executable Program File

Identifying Executable URLs

The Control.Wbpgms table contains the interpretation assignments for the URLs. These assignments include the locations of executable programs. This table enables you to specify the interpretation assignment of each directory or file that is listed.

Interpretation Values of the Executable URLs

The interpretation values of the executable URLs and their meanings are listed in the following table.

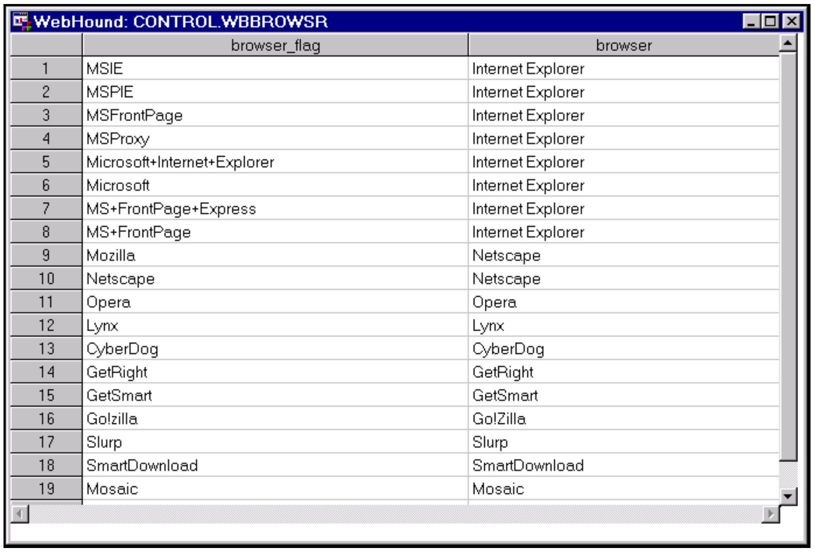
Table 3.4 Control.Wbpgms Table: Interpretation Values and Their Meanings

Interpretation Value	Meaning
1	All URLs under this directory are considered to be the locations of executable programs.
2	The specific program is considered to be an executable program.
0	The directory is parsed the same as all other URLs. Use type 0 to allow directories such as image directories to exist in a subdirectory that might otherwise contain all executable programs.

Identifying Web Browsers

The Control.Wbbrowsr table contains a list of all the browsers that are recognized by default. To open this table, select the **`Edit`** button that is next to **`Web browsers`** in the Parsing frame.

Display 3.10 Control.Wbbrowsr Table



	browser_flag	browser
1	MSIE	Internet Explorer
2	MSPIE	Internet Explorer
3	MSFrontPage	Internet Explorer
4	MSProxy	Internet Explorer
5	Microsoft+Internet+Explorer	Internet Explorer
6	Microsoft	Internet Explorer
7	MS+FrontPage+Express	Internet Explorer
8	MS+FrontPage	Internet Explorer
9	Mozilla	Netscape
10	Netscape	Netscape
11	Opera	Opera
12	Lynx	Lynx
13	CyberDog	CyberDog
14	GetRight	GetRight
15	GetSmart	GetSmart
16	Go!zilla	Go!Zilla
17	Slurp	Slurp
18	SmartDownload	SmartDownload
19	Mosaic	Mosaic

Matching the Web Log Contents with Control.Wbbrowsr

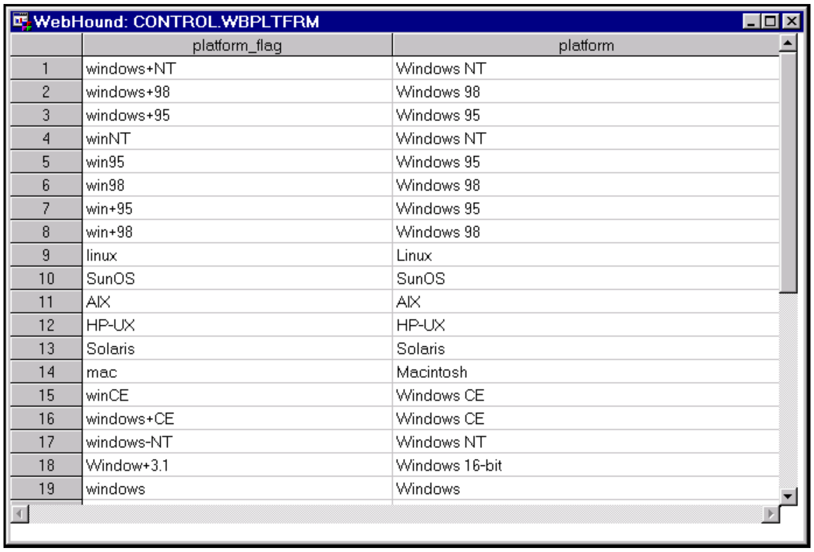
The name of the visitor's browser is captured by the User_Agent field of the Web log. During processing, the User_Agent field is compared to browser_flag strings in the Control.Wbbrowsr table.

If the User_Agent field matches one of the browser_flag strings, then the value for the corresponding browser is stored in the detail tables. You can specify additional values for browser_flag. If none of the values of browser_flag match the User_Agent field, then the value of browser is set to **`other`**.

Identifying Platforms

The Control.Wbpltfrm table contains a list of all the operating system platforms that are recognized by default. To open this table, select the **`Edit`** button that is next to **`Platforms`** in the Parsing frame.

Display 3.11 Control.Wbpltfrm Table



	platform_flag	platform
1	windows+NT	Windows NT
2	windows+98	Windows 98
3	windows+95	Windows 95
4	winNT	Windows NT
5	win95	Windows 95
6	win98	Windows 98
7	win+95	Windows 95
8	win+98	Windows 98
9	linux	Linux
10	SunOS	SunOS
11	AIX	AIX
12	HP-UX	HP-UX
13	Solaris	Solaris
14	mac	Macintosh
15	winCE	Windows CE
16	windows+CE	Windows CE
17	windows-NT	Windows NT
18	Window+3.1	Windows 16-bit
19	windows	Windows

Matching the Web Log Contents with Control.Wbpltfrm

The name of the visitor's platform is captured in the User_Agent field of the Web server log file. During processing, the text in the User_Agent field is compared to platform_flag strings in the Control.Wbpltfrm table.

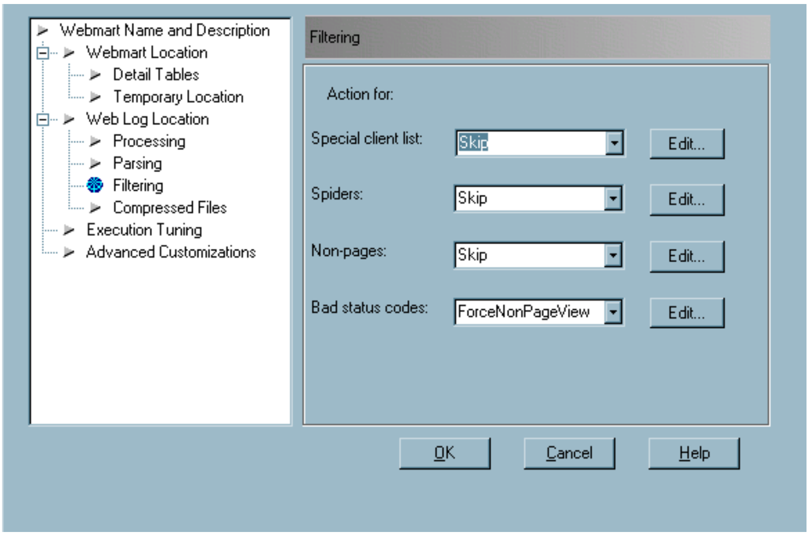
If the User_Agent field matches one of the platform_flag strings, then the corresponding platform value is stored in the detail tables. You can specify additional values for platform_flag. If none of the values of platform_flag are present in the User_Agent field, then the value of platform is set to **`other`**.

The Filtering Properties

You can set properties in the Filtering frame to determine how to handle the following requests:

- requests from clients that are designated as special, such as clients whose requests should be excluded from the final results
- requests from spider or robot clients
- images or other non-text requests
- page requests that resulted in a bad status code (indicating that an error occurred)

Display 3.12 Filtering Frame of the Properties Window



Action for:

Special client list: Skip

Spiders: Skip

Non-pages: Skip

Bad status codes: ForceNonPageView

Navigating to the Filtering Frame

From the tree view in the Properties - <*Web mart name*> window, select **`Filtering`** under **`Web Log Location`**. The Filtering frame appears to the right of the tree view.

Keep, Skip, Tally, and Force Non-Page View Actions

For each of the filtering properties (special client list, spiders, non-pages, and bad status codes), you can select a **`Keep`**, **`Skip`**, **`Tally`**, or **`ForceNonPageView`** action.

`Keep` Includes and processes all incoming records of this type from the Web log.

`Skip` Ignores all incoming records of this type during processing.

`Tally` Counts the number of this type of record, but does not perform any further analysis. SAS e-Data ETL software does not include analytical data for this type of record in the Weblog_Detail table.

`ForceNonPageView`
Does not count these requests along with legitimate page requests, but does include data for the records in the Weblog_Detail table. This option is useful, for example, when you want to use the SAS Web Analytics software to produce a report that lists all the requests that produced error messages, but you don't want to count them with the other requests.

The following table shows how SAS e-Data ETL software processes a filtering property according to the selected action.

Table 3.5 How SAS e-Data ETL Software Processes Filtering Properties According to Action

Action	Filtering Property Is Counted as a Page	Filtering Property Is Included in the Weblog_Detail Table
`Keep`	✓	✓
`Skip`		
`Tally`	✓	
`ForceNonPageView`		✓

Special Client List

The special client list property specifies how SAS e-Data ETL software should treat records that originate from any client in the special client list. This list contains host names or IP addresses that identify clients that need to have special treatment. For example, you can use this list to exclude records that originate from within a company, so that the analysts can concentrate on Web traffic that originates from customers only. By default, this table initially contains no entries.

The default action for special clients is `Skip`. During processing, SAS e-Data ETL software ignores all the records (in the Web server log data) from the special clients in this list.

Editing the Contents of the Special Client List Table

To add an IP address to the special client list table (Control.Wbspecl), select the `Edit` button that is next to `Special client list`. The Control.Wbspecl table opens. When you first open this table, it does not contain any entries. Add the host names or IP addresses that you want SAS e-Data ETL software to treat as special clients.

Using the Wildcard Character in the Special Client List Table

If you want all the client IDs from the same domain to be treated as special clients, then use a wildcard character (single leading or trailing asterisk) to match multiple client IDs. Here are the acceptable uses of the wildcard character to specify special clients:

Table 3.6 Examples of Using the Wildcard Character

Wildcard Usage	Effect
10.*	matches all client IDs that start with "10."
34.12.*	matches all client IDs that start with "34.12."
*.sas.com	matches all client IDs that end with ".sas.com"
*.org	matches all client IDs that end in ".org"

Spiders

Most search engine portals use a spider or robot program to search and catalog Web pages. Spiders and robots frequently access all the pages in a Web site while they search for content.

If the requests from spiders and robots were not segregated when SAS e-Data ETL software analyzes a Web server traffic pattern, then the statistics such as Most Frequent Visitors and the Entry and Exit Points would, at worst, be skewed or inflated. At best, the data that describes client activity would not generally represent the mindful selections of human clients.

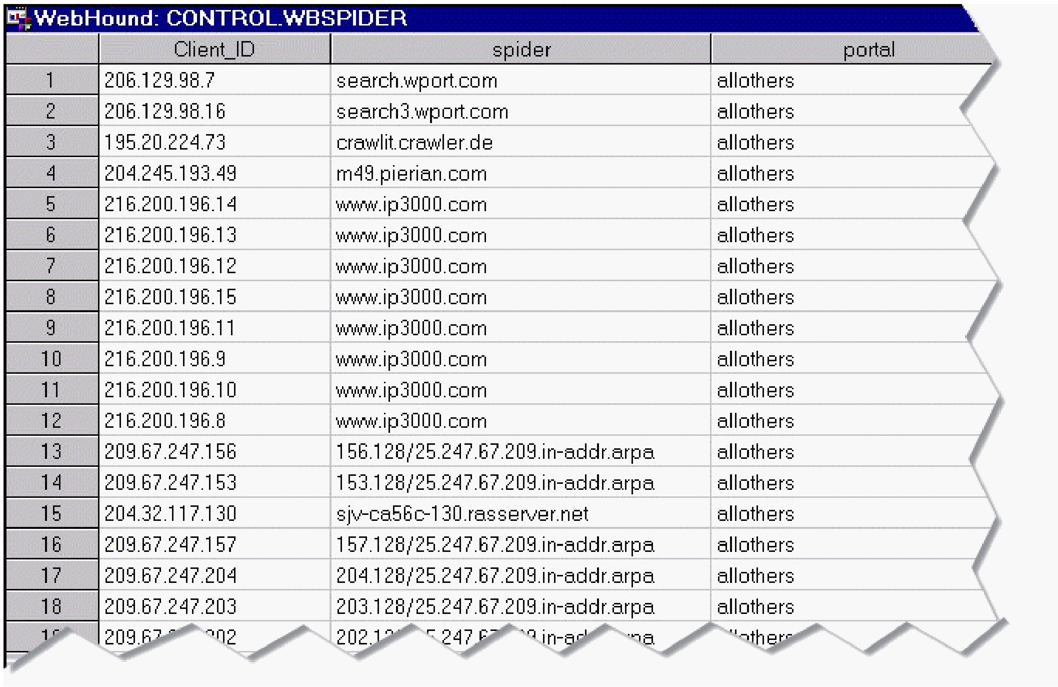
The purpose of identifying the spider clients in the Control.Wbspider table is to enable the hits from spiders or robot programs (as opposed to hits from human clients) to be excluded from analysis.

As with the special client list, the default action for spider clients is **`Skip`**. During processing, the actions of spider and robot clients that are listed in the Control.Wbspider table are ignored.

Editing the Contents of the Control.Wbspider Table

To access the Control.Wbspider table, select the **`Edit`** button that is next to **`Spiders`**. The Control.Wbspider table displays. To add a new spider client or to delete a spider or robot client from the Control.Wbspider table, highlight a row and select **`Edit`** on the tool bar. Select the appropriate action from the drop-down menu.

Display 3.13 Contents of a Sample Control.Wbspider Table



WebHound: CONTROL.WBSPIDER

	Client_ID	spider	portal
1	206.129.98.7	search.wport.com	allothers
2	206.129.98.16	search3.wport.com	allothers
3	195.20.224.73	crawlit.crawler.de	allothers
4	204.245.193.49	m49.pierian.com	allothers
5	216.200.196.14	www.ip3000.com	allothers
6	216.200.196.13	www.ip3000.com	allothers
7	216.200.196.12	www.ip3000.com	allothers
8	216.200.196.15	www.ip3000.com	allothers
9	216.200.196.11	www.ip3000.com	allothers
10	216.200.196.9	www.ip3000.com	allothers
11	216.200.196.10	www.ip3000.com	allothers
12	216.200.196.8	www.ip3000.com	allothers
13	209.67.247.156	156.128/25.247.67.209.in-addr.arpa	allothers
14	209.67.247.153	153.128/25.247.67.209.in-addr.arpa	allothers
15	204.32.117.130	sjv-ca56c-130.rasserver.net	allothers
16	209.67.247.157	157.128/25.247.67.209.in-addr.arpa	allothers
17	209.67.247.204	204.128/25.247.67.209.in-addr.arpa	allothers
18	209.67.247.203	203.128/25.247.67.209.in-addr.arpa	allothers
1	209.67. 202	202.12 5.247.6 9.in-ad pa	lothers

Identifying a New Spider

Spiders or robots might make requests to every page of your Web site. If a visitor's clickstream is unusually long, then a message is written to the SAS log warning that the request probably came from a spider or robot. You might want to add this visitor to your Control.Wbspider table.

Non-Pages

An example of a non-page type of resource is an image (GIF or JPG) file or an audio (AU or WAV) file. Each requested item, whether it is a non-page GIF file or a genuine

HTML file, is called a *hit* when Web log data is being analyzed. However, by default, only HTML files, ASP programs, CGI programs, ASPX programs, JavaHTML files, PDF files, and text- or script-based Cold Fusion files are counted as page views.

Processing non-page resources provides little useful data for analysis but requires a great deal of resources. If you set non-pages to an action other than **`Skip`**, then loading this type of data into the detail tables will greatly increase the time that it takes to process Web data and can unnecessarily increase the demand for data storage.

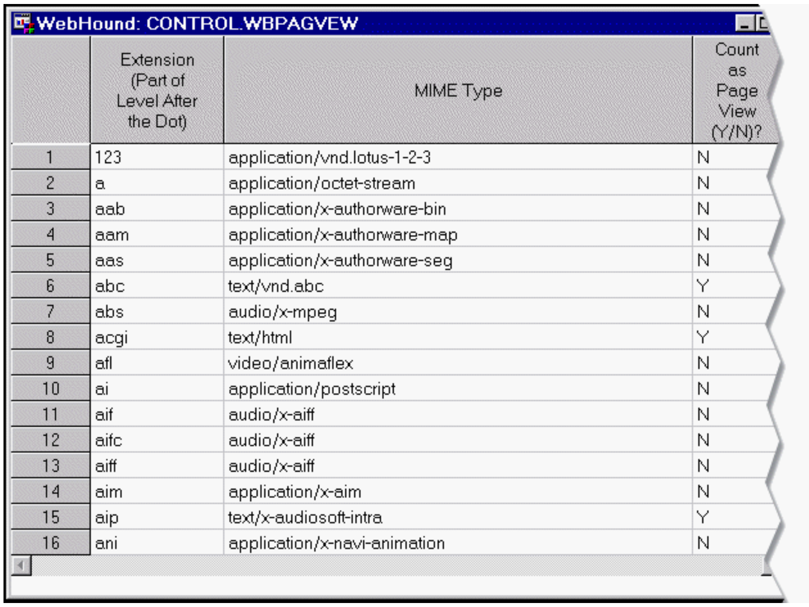
The default action **`Skip`** instructs SAS e-Data ETL software to ignore and discard all requests for non-pages during processing.

How to Count or Exclude a MIME Type as a Page View

In addition to editing the action property for non-page items, you can specify whether to count certain types of files as pages for the page view information. The Control.Wbpagvew table associates each type of file, known as a MIME type, with a file extension. You edit the Control.Wbpagvew table in order to instruct SAS e-Data ETL software to exclude particular MIME types (such as JavaScript files, Perl scripts, or other types of files) as page views.

To open the Control.Wbpagvew table, select the **`Edit`** button that is next to **`Non-pages`**. The Control.Wbpagvew table contains all MIME types that it recognizes. By entering *Y* or *N* beside the associated MIME type, you instruct SAS e-Data ETL software to count or exclude that MIME type during processing. In the following display, for example, three MIME types—ABC, ACGI, and AIP—will be counted as page views.

Display 3.14 Control.Wbpagvew Table



	Extension (Part of Level After the Dot)	MIME Type	Count as Page View (Y/N)?
1	123	application/vnd.lotus-1-2-3	N
2	a	application/octet-stream	N
3	aab	application/x-authorware-bin	N
4	aam	application/x-authorware-map	N
5	aas	application/x-authorware-seg	N
6	abc	text/vnd.abc	Y
7	abs	audio/x-mpeg	N
8	acgi	text/html	Y
9	afl	video/animaflex	N
10	ai	application/postscript	N
11	aif	audio/x-aiff	N
12	aifc	audio/x-aiff	N
13	aiff	audio/x-aiff	N
14	aim	application/x-aim	N
15	aip	text/x-audiosoft-intra	Y
16	ani	application/x-navi-animation	N

Bad Status Codes

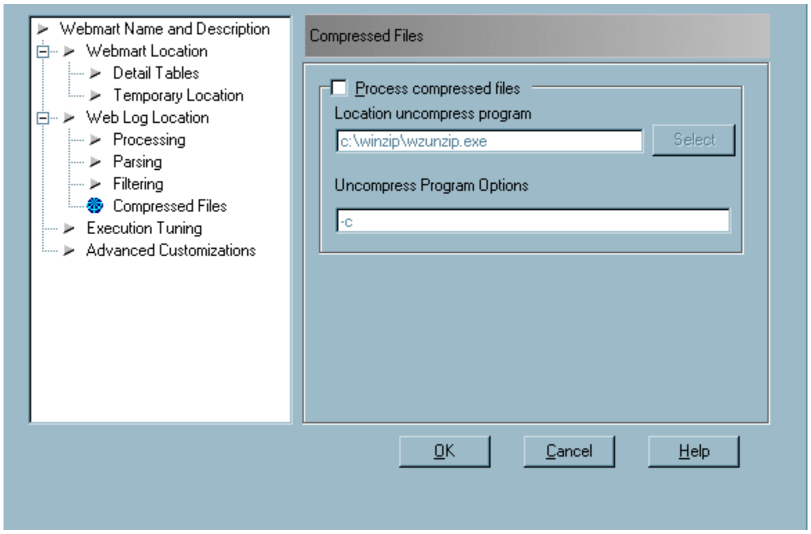
The bad status codes property enables you to customize the action that SAS e-Data ETL software takes when it encounters records that contain an error status code in the incoming data from the Web server log file. For example, status code 404 indicates that the Web server could not locate the requested resource.

If you want your detail tables to contain data about bad status code pages, then the records must be available in your table. **`ForceNonPageView`**, which is the default action, instructs the software to keep the information about the resources that produced error codes in the table, but not to count bad status codes as pages.

The Compressed Files Properties

In the Compressed Files frame of the Properties window, you specify whether your Web server log files are compressed, where the uncompressing executable file is located, and, if applicable, what uncompress command options to use.

Display 3.15 Compressed Files Frame of the Properties Window



☐ Process compressed files

Location uncompress program: c:\winzip\wzunzip.exe

Uncompress Program Options: -c

Navigating to the Compressed Files Frame

From the tree view in the Properties - <*Web mart name*> window, select **`Compressed Files`** under **`Web Log Location`**. The Compressed Files frame appears to the right of the tree view.

Processing Compressed Web Server Log Files

Selecting the **`Process compressed files`** option instructs SAS e-Data ETL software to process Web server log files that have been compressed with either a ZIP utility or the UNIX **`compress`** command. If the option is selected, then SAS e-Data ETL software calls the specified uncompress program to uncompress the Web server log files during processing, but it does not physically place the uncompressed files onto the disk of the processing system.

CAUTION:

> **Performance metrics have shown that selecting this option results in poor performance in the Windows operating environment.** For better performance, use a separate utility to uncompress your Web server log files before processing them with SAS e-Data ETL software in the Windows environment. △

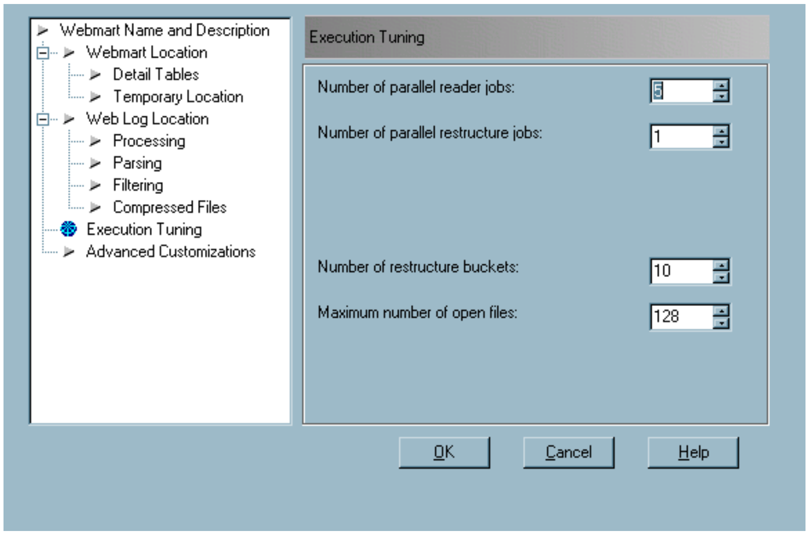
If the **`Process compressed files`** option is selected, then specify the name and location of the program that will be used to uncompress the Web server log files. Use the **Select** button to locate the appropriate program, or type the path and name of the uncompress program in the text field.

If you use the uncompress command in UNIX, then enter **-c** in the **Uncompress Program Options** field (see Display 3.15 on page 50). The **-c** option converts the compressed file to the standard output that SAS e-Data ETL software reads.

The Execution Tuning Properties

In the Execution Tuning frame, you can specify the maximum number of sessions that you want to allow for different types of SAS e-Data ETL software tasks in order to minimize the bottlenecks that are caused by intensive processes.

Display 3.16 Execution Tuning Frame of the Properties Window



Execution Tuning

Number of parallel reader jobs: 5

Number of parallel restructure jobs: 1

Number of restructure buckets: 10

Maximum number of open files: 128

Navigating to the Execution Tuning Frame

From the tree view in the Properties - <*Web mart name*> window, select **Execution Tuning**. The Execution Tuning frame appears to the right of the tree view.

Description of Number of Parallel Jobs Properties

SAS e-Data ETL software uses parallel processing during the Extract process. The number of parallel jobs that are created depends on the number you specify for **Number of parallel reader jobs** and for **Number of parallel restructure jobs**. The following table describes the default values for these properties.

Table 3.7 Description of Parallel Jobs Properties

Property Name	Default Value	Description
Number of parallel reader jobs	5	Controls how many concurrent SAS sessions are started when SAS e-Data ETL software is reading Web server log data
Number of parallel restructure jobs	1	Controls how many concurrent SAS sessions are started when SAS e-Data ETL is restructuring Web server log data

The default value for this property might not be the optimal value for your site.

Number of Restructure Buckets

The **`Number of restructure buckets`** value indicates the number of divisions that are applied to the data that is being processed. By dividing the incoming data into restructure buckets, SAS e-Data ETL software sorts the data more efficiently. The default number of restructure buckets is 10.

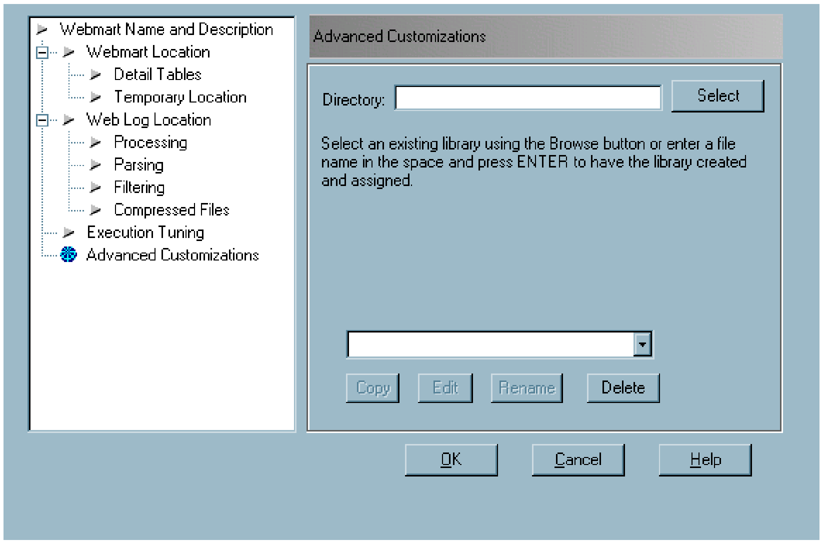
Maximum Number of Open Files

The **`Maximum number of open files`** value indicates the operating environment limit. Some operating environments limit the number of files that can be opened simultaneously; some environments base this number on the account of the user. The value of this property might need to be adjusted if the programs fail or if a program has problems with opening tables due to operating environment limits. The default value is 128.

The Advanced Customizations Frame

The Advanced Customizations frame of the Properties window enables you to select, copy, and modify source entries called custom access modules (CAMs), such as User_assignments_after_input.

Display 3.17 Advanced Customizations Frame of the Properties Window



Select an existing library using the Browse button or enter a file name in the space and press ENTER to have the library created and assigned.

Navigating to the Advanced Customizations Frame

From the tree view in the Properties - <*Web mart name*> window, select **`Advanced Customizations`**. The Customizations frame appears to the right of the tree view.

About Custom Access Modules (CAMs)

The source entries that have been created specifically to enable you to customize the SAS e-Data ETL processes are called custom access modules (CAMs). The CAMs that are supplied with the SAS e-Data ETL program are located in the Sashelp.Wbetl catalog (within your SAS 9.1 directory structure). The comments within a CAM provide instructions for modifying the CAM.

CAUTION:

Do not modify any source entry that is not a CAM. Modifying a source entry other than a CAM will produce unexpected results and is strongly discouraged. △

The code from a CAM is run at the beginning or end of the Extract process, depending on the function of the CAM.

The following list contains the source entries that you can modify:

Custom_log_format
: Executes any custom code for creating SAS formats that are used by the SAS e-Data ETL Extract process. This code will be executed after all of the other formats that are used by SAS e-Data ETL program have been created. Therefore, this code can be used to add new formats or to change the definition of existing formats.

Custom_log_input
: Builds the input statement that will read the custom log records as specified in the call to RXMATCH. It also performs any calculations that handle the formatting for a specific Web log in order to set up values that are expected by subsequent ETL processes.

Custom_log_rxfree
: Executes the call to RXFREE that deallocates the memory that is used by RXPARSE.

Custom_log_rxmatch
: Executes the call to RXMATCH that will detect the new Web log format that is specified by the pattern defined in Custom_log_rxparse.

Custom_log_rxparse
: Executes the call to RXPARSE that detects the new Web log format.

Custom_log_set_server_type
: Sets the server type as specified by the results of the call to RXMATCH in Custom_log_rxmatch.

Custom_log_test_harness
: Used to test the support of custom Web logs in an environment that is separate from the SAS e-Data ETL environment. The comment blocks located at the top of each of the custom_log_ source entries (with names that begin with **`custom_log_`**) contain example code that reads Common Log Format (CLF) log files. For your first custom log, run the code in this example as it is written, and make sure that the program runs successfully before you make any changes to the code. You can use the example data called **`clf.log`** located in SASMISC to test the example source entries.

Determine_server_and_sitename
: Executes any code that is needed to parse the name of the file being processed in order to set any variables that are not available in the data itself. For example, the sitename might be used as the name of a directory in which the file is located.

User_assignments_after_input
: Used in one of two ways: to set variables that are not available until after the Web log is read, or to populate any fields that are not recorded directly in the Web log. The statements in this module are executed immediately after the SAS INPUT statement.

User_assignments_before_input
: Used to customize certain variables before a record is written to the SAS tables. While the Extract process is running, this module is positioned to run after the URL is parsed into all relevant pieces and after all other processing of the incoming record (within a Web log) is completed.

User_assignments_by_session_id
: Used to customize certain variables after the Web log records are sorted by Session_ID and Date_Time. The User_assignments_by_session_id module is invoked during the execution of the Execute_the_analysis_ jobs module, which is immediately before each observation is output.

User_assignments_for_data_mining
: Used to transform the fields of your Web log to make them more effective for data mining functions. Statements that are added here will be executed automatically at the end of Load processing.

Visitor_id_logic
: Calculates the visitor ID from other columns in the Control.Wbconfig data set. See also “How to Set Visitor ID Values from Cookies or CGI Parameters” on page 75.

For a complete list of the source entries that are executed by the Extract process, see “Overview: The Modules in the Extract Process” on page 80. For a complete list of the source entries that are executed by the Load process, see “Overview: The Modules of the Load Process” on page 99.

How to Modify a CAM

The Advanced Customization frame enables you to select and modify CAM source entries in order to customize specific ETL processes. When you select a catalog (such as Wbetl) that contains the CAM source entry that you want to modify, a temporary copy of the catalog is created in your Work library. Any modified source entry is saved (with the same name) in a duplicate catalog in your Usermods library, and the original source entry in the Sashelp library always remains unmodified.

CAUTION:

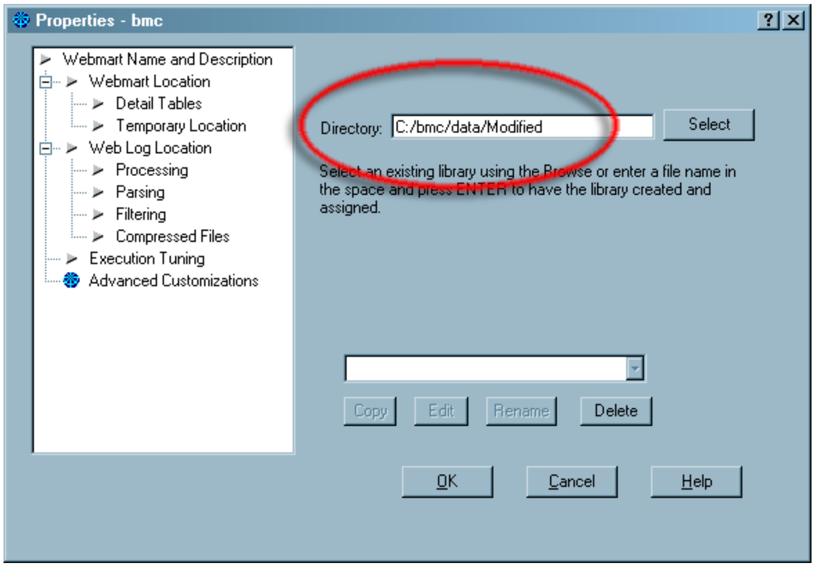
> **The original source entries in the Sashelp library should never be modified directly.** The Advanced Customization frame enables you to modify and save a copy of the source entry in the Usermods library. △

The following sections use the User_assignments_after_input CAM (one of many CAMs that are provided with the SAS e-Data ETL software) as an example to illustrate how to use the Advanced Customizations frame for modifying CAMs in the SAS e-Data ETL Administrator.

Note: You can use these steps to modify any CAM source entry by substituting the source entry name for User_assignments_after_input. △

1 Navigate to the Advanced Customizations frame of the Properties window of your Web mart.

2 In the **Directory** text box, type the path to a destination library (or the location of your modified source entries). If you have not assigned your Usermods library, then type the path, including the new name, and press ENTER. The new library is created and assigned. In the following display, for example, a new library named Modified is assigned to the Usermods library:



Directory: C:/bmc/data/Modified

Select an existing library using the Browse or enter a file name in the space and press ENTER to have the library created and assigned.

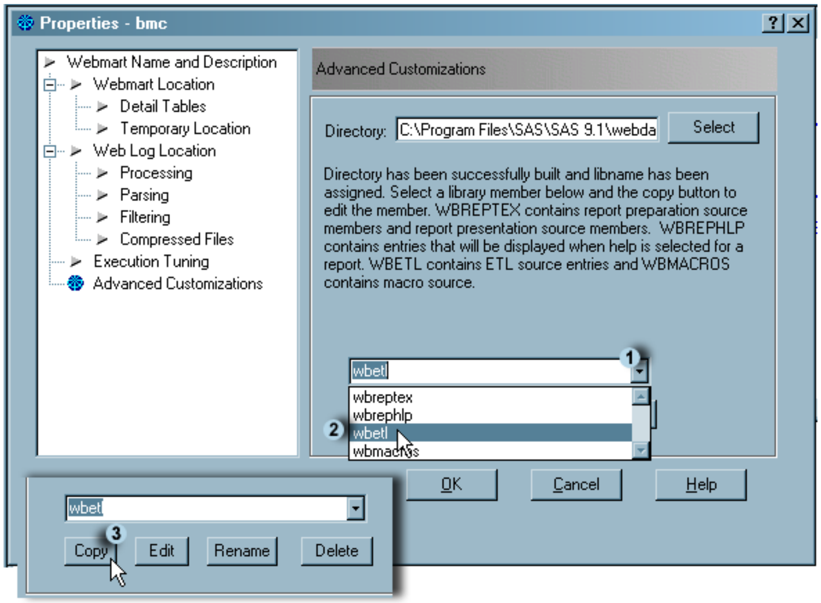
3 Click **Select**. The text below the **Select** button changes to prompt you to select the catalog that contains the source entry to copy (in order to modify the copy in a later step).

Note: All of the CAMs for SAS Web Analytics ETL processing are located in the Wbetl catalog of the Sashelp library. △

4 Click ▾ (1 in the following display) to display the drop-down list.

5 Highlight the catalog that contains the source entry that you want to copy. The User_assignments_after_input source entry is in the Wbetl catalog, so **wbetl** is selected (2 in the following display). The selected catalog name appears in the text box.

6 Click **Copy** (3 in the insert of the following display). The Select An Entry window appears.

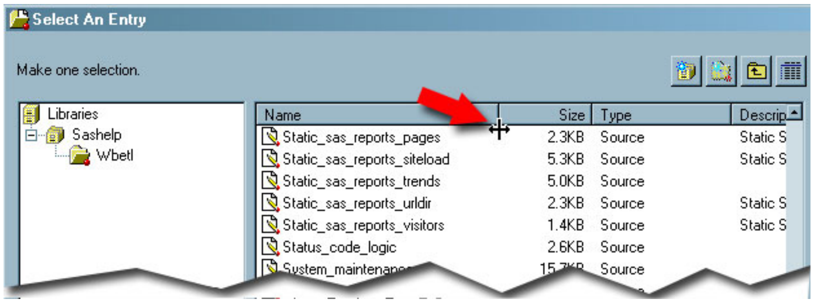


Directory: C:\Program Files\SAS\SAS 9.1\webda

Directory has been successfully built and libname has been assigned. Select a library member below and the copy button to edit the member. WBREPTEX contains report preparation source members and report presentation source members. WBREPHLP contains entries that will be displayed when help is selected for a report. WBETL contains ETL source entries and WBMACROS contains macro source.

wbetl

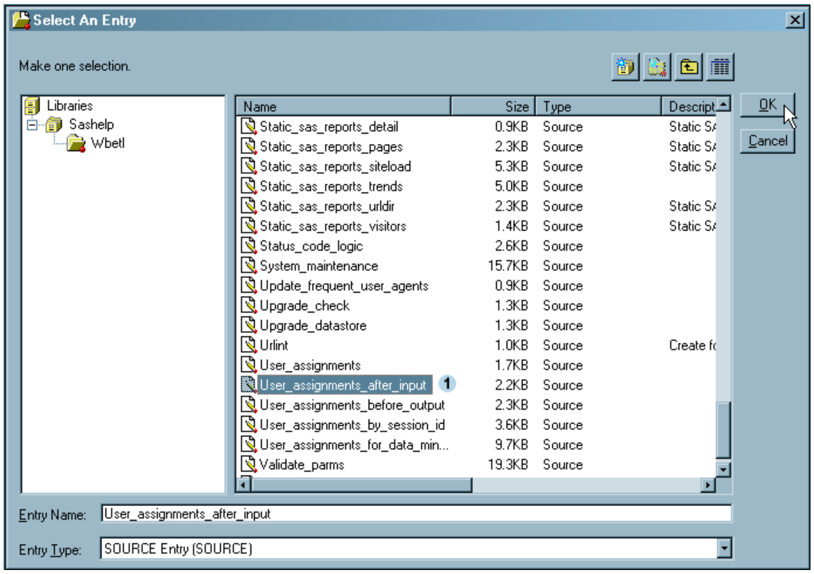
7 To widen the **Name** column, move the cursor over the line between the Name column and the Size column until the cursor changes (see the following display):



Make one selection.

Name	Size	Type	Descrip
Static_sas_reports_pages	2.3KB	Source	Static S
Static_sas_reports_siteload	5.3KB	Source	Static S
Static_sas_reports_trends	5.0KB	Source	
Static_sas_reports_urldir	2.3KB	Source	Static S
Static_sas_reports_visitors	1.4KB	Source	Static S
Status_code_logic	2.6KB	Source	
System_maintenanc	15.7KB	Source	

Drag the cursor to the right until you can identify the source entry names in the Name column. Highlight the source entry that you want to modify (in this example, User_assignments_after_input), and select **OK** (see the following display):



Make one selection.

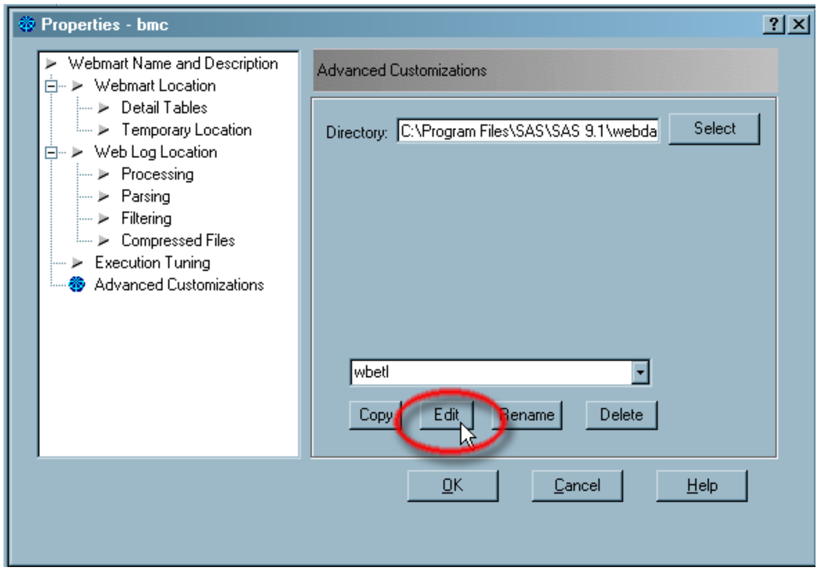
Name	Size	Type	Descript
Static_sas_reports_detail	0.9KB	Source	Static SA
Static_sas_reports_pages	2.3KB	Source	Static SA
Static_sas_reports_siteload	5.3KB	Source	Static SA
Static_sas_reports_trends	5.0KB	Source	
Static_sas_reports_urldir	2.3KB	Source	Static SA
Static_sas_reports_visitors	1.4KB	Source	Static SA
Status_code_logic	2.6KB	Source	
System_maintenance	15.7KB	Source	
Update_frequent_user_agents	0.9KB	Source	
Upgrade_check	1.3KB	Source	
Upgrade_datastore	1.3KB	Source	
Urlint	1.0KB	Source	Create fo
User_assignments	1.7KB	Source	
User_assignments_after_input	2.2KB	Source	
User_assignments_before_output	2.3KB	Source	
User_assignments_by_session_id	3.6KB	Source	
User_assignments_for_data_min...	9.7KB	Source	
Validate_parms	19.3KB	Source	

Entry Name: User_assignments_after_input

Entry Type: SOURCE Entry (SOURCE)

The CAM source entry is copied to your Usermods library, and the Advanced Customization frame re-appears.

8 Select the copy of the CAM source entry from the catalog in the Work library in order to modify the CAM source entry. Click **Edit** (see the following display):



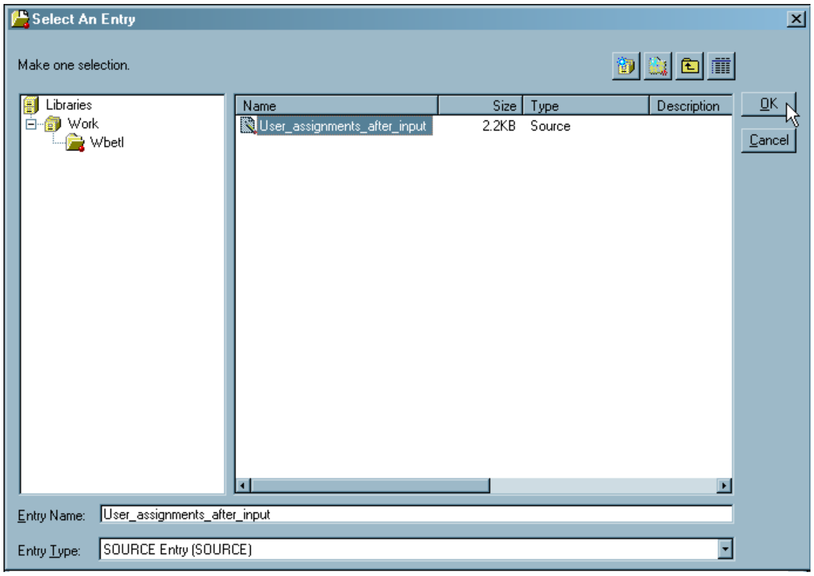
Advanced Customizations

Directory: C:\Program Files\SAS\SAS 9.1\webda

wbetl

The Select An Entry window displays all of the source entries in your Work library.

9 Select User_assignments_after_input and click **OK** (see the following display):

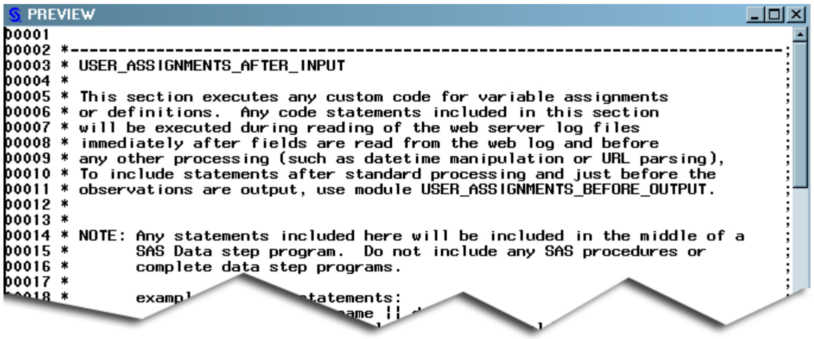


Name	Size	Type	Description
User_assignments_after_input	2.2KB	Source	

Entry Name: User_assignments_after_input

Entry Type: SOURCE Entry (SOURCE)

A SAS Program Editor window opens (see the following display):



```
*--------------------------------------------------------------------------;
* USER_ASSIGNMENTS_AFTER_INPUT
*
* This section executes any custom code for variable assignments
* or definitions.  Any code statements included in this section
* will be executed during reading of the web server log files
* immediately after fields are read from the web log and before
* any other processing (such as datetime manipulation or URL parsing).
* To include statements after standard processing and just before the
* observations are output, use module USER_ASSIGNMENTS_BEFORE_OUTPUT.
*
*
* NOTE: Any statements included here will be included in the middle of a
*       SAS Data step program.  Do not include any SAS procedures or
*       complete data step programs.
*
*       examp            tatements:
```

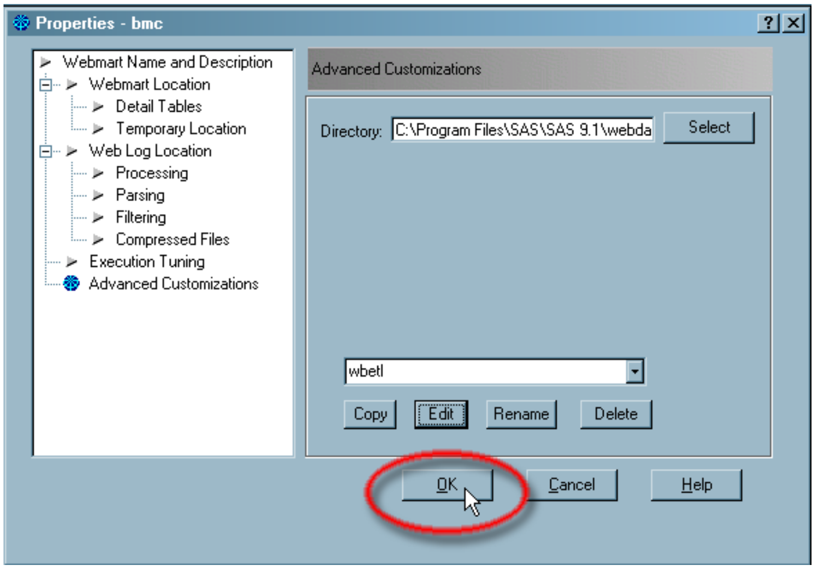
10 Enter your changes to the source entry code. The instructions for modifying a CAM are located in the comment lines of the source entry. When you are finished modifying the code, close the SAS Program Editor window. Click **Yes** in the small dialog box that appears to save your modifications.

Note: Click **Cancel** to return to the SAS Program Editor window where you can continue making modifications, or click **No** to lose any modifications that you made to the code and to return to the Advanced Customization frame. △

Your changes are temporarily saved to the Work library.

11 To modify additional source entries in the same catalog, repeat steps 7–10.

12 To save all of the modified source entries to your Usermods.Wbetl catalog, click **OK** in the Advanced Customization frame (see the following display):



Directory: C:\Program Files\SAS\SAS 9.1\webda

wbetl

Note: If you click **Cancel** in the Advanced Customizations frame, then the modifications that you made to any source entries are lost, and no modified source entries are copied to the Usermods library. However, temporary copies of any modified source entries might exist in the Work library for as long as your SAS session remains open. See the following section to delete a source entry from the Work library. △

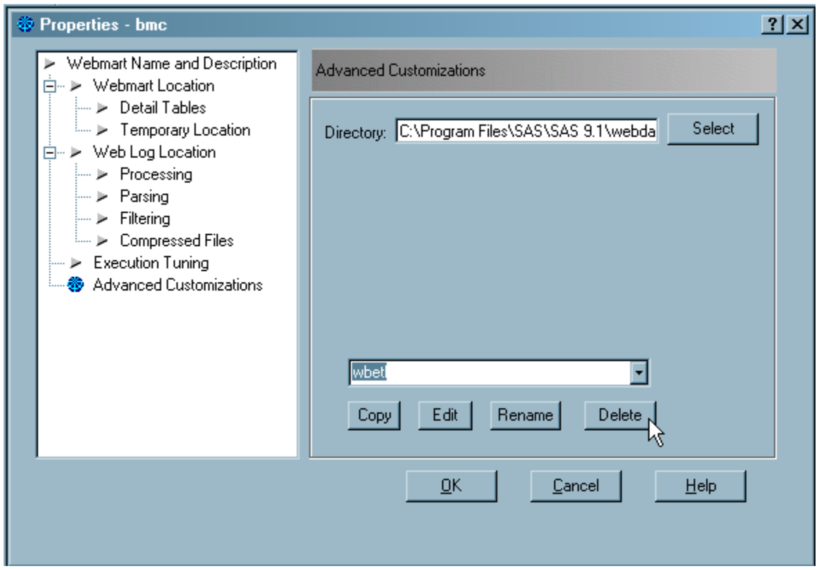
Deleting a CAM from the Work Library

Note: You can delete only the CAM source entries in your temporary Work library. △

To delete a CAM source entry:

1 In the SAS e-Data ETL Administrator, go to the Advanced Customizations frame of the Properties window of a selected Web mart.

2 Click ▾ near the bottom of the Advanced Customizations frame to display the drop-down list. Select the catalog that contains the CAM source entry that you want to delete and click **Delete** (see the following display):



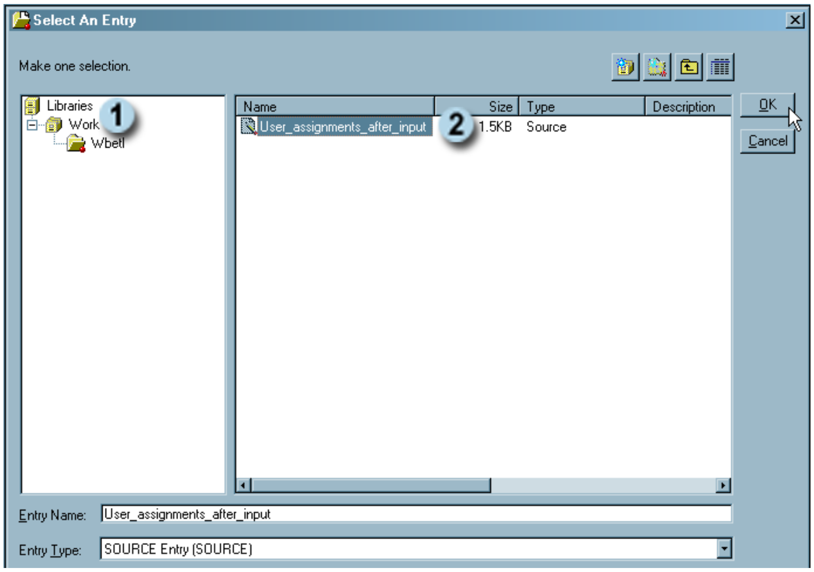
Directory: C:\Program Files\SAS\SAS 9.1\webda

wbetl

The Select an Entry window appears, showing a list of the CAM source entries in your Work library (1 in the following display).

Note: Only the source entries that you did not save to the Usermods library will appear in the temporary Work library. △

3 Select the CAM that you want to delete (2 in the following display) and click **OK**



Name	Size	Type	Description
User_assignments_after_input	1.5KB	Source	

Entry Name: User_assignments_after_input

Entry Type: SOURCE Entry (SOURCE)

SAS e-Data ETL software deletes the source entry without confirmation and returns you to the Advanced Customizations frame.

How the Customizing Process Works in SAS e-Data ETL Software

The ETL processes read metadata tables and execute source entries, both of which are located in the Sashelp library. To customize the ETL processes, you use the SAS e-Data ETL Administrator to modify the underlying metadata, instead of modifying the metadata tables directly. You can also modify the ETL processing by customizing the selected source entries called CAMs.

The Control and Usermods libraries are important for enabling you to customize the SAS e-Data ETL process. These libraries are associated with a specific Web mart. Therefore, every time you create a new Web mart, you need to create and assign the Control and Usermods libraries for the Web mart.

The Control Library for Your Web Mart

The Control library enables you to customize the metadata tables in the Sashelp library. You can use the SAS e-Data ETL Administrator to customize most of these tables. For more information about customizing a Web mart, see "Overview: The Properties of a SAS Web Analytics Web Mart" on page 28.

Description of Specific Control Tables

The SAS metadata tables that are listed in the following table are included in the Control library.

Table 3.8 Tables in the Control Library

Table Name	Contents
Wbconfig	The description and the default values for SAS e-Data ETL program parameters. Any changes that you make to the SAS e-Data ETL metadata, such as changing a default value, are reflected in this table.
Wbfields	The fields that are extracted from the Web logs and are used to create the Weblog-Detail table. To make modifications to this table, see "Adding or Deleting Fields in Your Web Server Log File" on page 38.
Wbinfile	The INFILE options for processing Web logs. In this table, you can specify the desired options for the SAS INFILE statement for each Web log. For more information, see "INFILE Options for Various Web Log Types" on page 37.
Wbpagvew	All MIME types that SAS e-Data ETL software recognizes during processing. This table can be edited to exclude particular file types. For more information, see "Non-Pages" on page 48.
Wbspecl	All the host names or IP addresses that SAS e-Data ETL software should exclude during processing. For more information, see "Special Client List" on page 47.
Wbspider	A current list of all recognized spiders and robots. To learn how to update this list, see "Spiders" on page 47.

Usermods Library

The purpose of the Usermods library is to contain the catalogs of all of your modified CAM source entries and modified SAS data sets. You should create and assign a Usermods library for each Web mart.

When you use the Advanced Customizations frame of the SAS e-Data ETL Administrator, modified CAM source entries are automatically saved to the Usermods library that you either create at that time or which already exists (in a path that you specify). During ETL processing, if modified source entries exist in the Usermods library of your Web mart, then the SAS e-Data ETL Extract and Load processes automatically execute those modified source entries instead of executing the corresponding default source entries (located in the Sashelp library). For more information, see "The Advanced Customizations Frame" on page 52.

CHAPTER 4

Setting Up ETL Processing for SAS e-Data ETL Software

Working with Web Logs

Guidelines for Managing Web Logs

Because of variations in the configuration of Web servers and the security policies that are used by corporate Web sites, you will need to customize the daily maintenance of the Web log files that are processed by the SAS Web Analytics solution.

However, some Web log management tasks are common to nearly all Web sites. Following are some programming methods that you can use to automate some of these routine but critical tasks as you track and maintain your Web logs:

Close off your Web logs.
Close off the Web server log files that are retained by each of your Web servers in order to avoid losing or duplicating any information about the use of your Web site. Perform this operation at least once a day at the same time on all servers.

Your Web server should include utility software that provides instructions for automatically closing off its Web logs at specified intervals.

Rename each Web log.
Change the name of the Web log file before you process it. For example, include the Web server name and the datetime that the Web log was closed off to create a unique Web log file name.

Resolve the client IP addresses to their host names.
If your Web server does not resolve the client IP addresses to their host names for you, look up the client IP addresses in the Web log to resolve them to their host names.

Compress the Web logs.
Compress the Web server log files by using a ZIP utility program (in Windows) or by using the `compress` command (in UNIX). Be sure to specify the corresponding file options in the SAS e-Data ETL Administrator (see “The Compressed Files Properties” on page 50).

Transfer the Web logs to the SAS Web Analytics host.
Transfer the Web server log files from the Web server host(s) to the server where SAS e-Data ETL is installed in order to analyze and archive the Web log data.

Overview: Custom Web Logs

To enable the SAS e-Data ETL processes to read custom Web logs, use the source entries (located in the Wbetl catalog) that begin with Custom_log_. These Custom_log_ source entries contain examples in their code comments that you can place in each custom Web log. Refer to these examples along with the following instructions.

How to Add Support for a Custom Web Log

1 Create a `Usermods` directory with a Wbetl catalog that contains all the Custom_log_* source entries (both the entries that you have updated and the entries that remain unchanged).

   If you want to run the example in the source entries, then copy the statements from the comment block at the top of each Custom_log_* source and paste them into a SAS Program Editor window. Remove the comment delimiters at the beginning and end of each line. You can run these examples in the test harness that is provided. A test harness is a program that replicates the SAS e-Data ETL environment. Because the test harness enables you test your code outside SAS e-Data ETL software, it is faster and simpler to use.

2 Add entries to the Control.Wbfields table for any of the new fields that you are defining. If you are running the example, then you do not need to add any fields. If you forget to add a field to Wbfields, then the test harness will work, but SAS e-Data ETL software will not keep the variable in the Web mart when it reads the custom log.

3 Add an entry for your custom log format to the Control.Wbinfile table. This entry gives your new log format a name (in all uppercase) and supplies options for the INFILE statement that is used to read the data. If you are running the example, then duplicate the observation for CLF and update the Web server name to "EXAMPLE."

4 Set the macro variables described in the following table.

Table 4.1 Macro Variables to Set in the Custom_log_test_harness Module

Macro Variable	Definition
Custom_Weblog_Filename	The directory path to the location of your Web log.
Control_Location	The directory path of your Web mart's permanent results with **/control** appended to the end. On UNIX, it is the directory for your Web mart with **/control** appended to the end. The **/control** specifies the Web mart's control directory.
Custom_Log_Name	The unique name that is used to identify your custom Web log. Web log names that the SAS e-Data ETL process recognizes are CLF, IISOrig, MSProxy, ELF, and Netscape. When naming your custom Web log, do not use any of these values. The name that you choose should be a valid SAS variable name.

5 Make any modifications to the Custom_log_format module that are needed. The Custom_log_format module contains any custom code necessary to create SAS formats that the Extract process uses. This code is executed after all of the other formats for SAS e-Data ETL processing have been created.

6 Make any needed modifications to the Custom_log_rxparse module. This module contains a call to the RXPARSE SAS function. Follow the example below. You can find this example in the comment block within the source code of the Custom_log_rxparse module.

Detect a log by using the following format. (Note that this is a single line of code. It has been broken here to fit on the page.)

```
client-ip remote_login cs-username [datetime] req-type
   cs-usr-stem sc-status sc-bytes
```

Note that the cs(Referrer), cs(User-Agent), and Cookie fields are optional.

7 Make any changes to the Custom_log_rxmatch module that are needed. To make these changes, use the following format (continuing with the example):

```
match_example = rxmatch(example, _infile_);
```

8 Make any changes to the Custom_log_input module that are needed. The code in this section builds the input statement that will read the custom log records as specified in the call to the RXMATCH function. The code also performs any calculations that are specific to the log format in order to set up values that are expected by additional SAS e-Data ETL processing. These values include the following:

Combined_URL Should contain the URL from the input record so that it will be parsed according to RFC 2396.

Query_String Should contain the query string so that it will be parsed into name-value pairs. (The query string can also be included as part of the Combined_URL. In that case it will be separated from the requested file by a question mark.)

Username Should contain the authorized user ID that a user entered if he was prompted by the Web server.

Datetime Should contain the datetime of the Web log record so that the datetime derivative variables can be calculated from it.

GMT_Offset Should contain the GMT offset of the timestamp in the Web log record, though the processing that occurs later by default does not reference this field. You also need to modify the Datetime_logic program to use the GMT offset.

9 Make any changes to the Custom_log_set_server_type module that are needed. This module sets the server type as specified in the results of the call to RXMATCH in the Custom_log_rxmatch module.

10 Change the Custom_log_rxfree module to include the call to the RXFREE call routine.

11 Run the SAS e-Data ETL Extract process and check for errors before continuing. After the Extract process runs successfully, proceed to the SAS e-Data ETL Load process.

For more information about the RXPARSE or RXMATCH functions or the RXFREE call routine, see SAS Help and Documentation.

Correcting Data Reader Errors

Data reader errors may occur during the execution of the SAS e-Data ETL Extract process. As long as the Extract process can read some parts of the Web log, it continues to run. However, if the Extract process can read no part of the Web log, then a fatal error occurs, and the process stops.

The following table lists the most common causes of data reader errors and offers possible solutions:

Table 4.2

Cause of a Data Reader Error	Solution
A blank that is not enclosed in double quotes exists in a field.	Using the SAS e-Data ETL Administrator, change the INPUT and INFORMAT modifiers so that blanks are enclosed in double quotes.
The datetime field is missing from a record.	Modify your Web log to include the datetime field (see “How to Add the Date to a Web Log” on page 67.
The process is backloading large volumes of data.	To minimize problems that might occur when you backload data, break the compressed data into smaller pieces before processing it. For example, process data for only one day within a single execution of the ETL process. In the Execution Tuning frame of the SAS e-Data ETL Administrator, you can also adjust the settings of the parallel job properties in order to minimize these errors (see “The Execution Tuning Properties” on page 51).
A field value is expressed as a name-value pair.	Change the INFORMAT modifier of the field in the Control.Wbfields table to include the name string.
The process does not recognize the format of the Web server log file.	Modify the Web log or add the Web log as a custom log so that SAS e-Data ETL software will recognize and process the file.

Skipped Records or Fields

You might discover that the total number of records that are reported in a SAS Web Analytics report are not equal to the number of records in the processed Web log. To determine the reason(s) for any discrepancies of this nature, check the following factors that affect whether the ETL process either counts or skips a record in a Web log:

Table 4.3 Factors that Cause the ETL Process to Ignore a Web Log Record

Cause	Description
Incorrect syntax	The ETL process skips a record if a line in a Web log contains incorrect syntax. For example, if no blank space exists between the keyword GET and the URL, then the record is skipped. Examine the syntax of the Web log. If necessary, make the correction to the syntax (see “Overview: Custom Web Logs” on page 64).
URL contains a GIF	If a URL contains a GIF parameter, then the ETL process interprets the record as a non-page.
How you defined spider clients	If you have set the action for spider clients to **skip**, then the ETL process will not count any records that correspond to requests that come from a recognized spider client.

How to Add the Date to a Web Log

If your Web log does not contain the date (within the datetime field), then ask a system administrator to turn on the date option so that the Web server will add a datetime field to the Web log. For accuracy in reporting, you want the date when each record was written rather than the date that comes from a file header or some other location.

If you need to work with the dates in the Web log, then add code to the first subsection, Detect Header, of the Job_build__inputs CAM (see “Job_build__inputs” on page 83) in order to use the date from the header record of the Web log as the date for the session. The Job_build__inputs CAM is used to determine the layout of the Web server log file.

Adjusting Your Local Time Zone

To adjust for your time zone in relation to Greenwich Mean Time (GMT), modify the appropriate variable in the Sashelp.Wbetl.Datetime_logic CAM. To modify a CAM, see “How to Modify a CAM” on page 55. Here is the code for the Datetime_logic CAM:

```
*==========================================================================;
* section: datetime logic ;
* ;
* descrip: this section calculates datetime and its derivatives.           ;
* ;
*==========================================================================;
* ;
%put ######################################################################### ;
%put # Datetime logic # ;
%put ######################################################################### ;

%macro Datetime_Logic;
   if datetime=. then delete ;
/*------------------------------------------------------------------------*
* Adjust datetime stamp for Daylight Savings Time and for GMT Offset
* replacing the value surfaced in the web server log file.
*
* NOTE: Most servers report local time and most GMT offsets have already
* been adjusted for daylight savings time. These calculations are
* provided as examples of the adjustments that may be necessary to
* convert the reported time to GMT, or Coordinated Universal Time.
*
*
* * Example adjustment to datetime for Daylight Savings Time;
* dstime = dhms(mdy(4,8-weekday(mdy(4,7,year(datepart(datetime)))),
* year(datepart(datetime))),3,0,0)
* < datetime <
* dhms(mdy(10,32-weekday(mdy(10,31,year(datepart(datetime)))),
* year(datepart(datetime))),2,0,0);
* if dstime then datetime = datetime - ((GMT_Offset + 1) * 3600);
* else datetime = datetime - (GMT_Offset * 3600);
*
*-------------------------------------------------------------------------*/
*-------------------------------------------------------------------------;
* Calculate derivatives of Datetime as rounded dates. ;
* ;
* Each DATETIME derivative variable (date, week, month, quarter, and year);
* must be an actual SAS date value for the PROC SUMMARY at the ;
* end of Extract to identify the data properly. That is, these variables ;
* must contain a fullyfledged SAS date value, not a shortened form ;
* like 1-4 for quarter. The other datetime-related variables like ;
* HOUR and DAY_OF_WEEK are not used to pin an observation down in time ;
* and so do not have this restriction. ;
* ;
* Note also that every WebHound table *must* contain at least one ;
* recognizable DATETIME derivative variable so that generations of the ;
* data can be "managed over time" successfully. ;
*-------------------------------------------------------------------------;
Date = datepart(datetime) ;
Week = (Date - weekday(Date)+1);
Month = mdy(month(Date),1,year(Date)) ;
Quarter = mdy(month(date) - mod(month(date)-1,3),1,year(date));
Year = mdy(1,1,year(Date));
* These variables do not need to be SAS date values;
Hour = hour(timepart(Datetime)) ;
Day_of_Week = weekday(Date) + 1; * 1 = Sunday, 2 = Monday ... ;
%mend Datetime_Logic;
%Datetime_Logic;
*===========================================================================;
* END of DATETIME LOGIC ;
*===========================================================================;
```

Using the Daily.sas Program to Perform the ETL Processes

When you run the %WAETL macro to initialize your new Web mart, the Daily.sas program (see the sample below) is created in the **sas** directory of your SAS Web Analytics Web mart. The Daily.sas program automates the ETL processes by invoking the following programs through the associated macros:

1 the Extract process (%EDATAETL macro)
2 the Load process (%EDATAETL macro)
3 the Warehouse process (%WAETL macro)

Before you can use the Daily.sas program, you need to modify it according the instructions in the header block of the program. If you have a customized directory structure, you also need to modify the locations of the custom directories in the corresponding arguments within the %WAETL macro.

```
/**
 * Use this program to process weblog data into the SAS Web Analytics warehouse.
 *
 * IMPORTANT: You must supply the location of the weblogs to process.
 *       Either
 *          A. Specify the location in the WEBLOGS macro variable below by replacing
 *              with the location of your weblogs. WEBLOGS can
 *             be the full path either for a single weblog or for a directory that
 *             contains multiple weblogs.
 *       Or
 *          B. Launch the e-Data ETL administrative GUI from an interactive SAS session
 *             using the "edataetl" command and specify the location of your weblog(s)
 *             in the appropriate field.
 *       If you use method B, then you can remove the "weblog_path=&weblogs" parameter
 *       from the %edataetl call below. If you leave the parameter in the %edataetl
 *       call, it will take precedence over the Weblog location information you have
 *       entered via the e-Data ETL Administrator GUI.
 */

%MACRO SWA_ETL;

*** TO DO: specify the location of your weblogs;
%let WEBLOGS=;

%let webmart=d:\swa\prod\bmc\swamart-pub;

%let wbrc=0;

/* Extract raw weblog data into staging datasets. */
%edataetl( data_store   =  &webmart\e-data-etl,
           weblog_path  =  &weblogs,
           program      =  extract
         );

/* Load web data into final detail dataset. */
%edataetl( data_store   =  &webmart\e-data-etl,
           program      =  load
         );

/* Load web detail data into warehouse. */
%if &wbrc = 0 %then %do;

   %waetl( swamart  =  &webmart,
           program  =  warehouse
         );
```

```
%end;

%MEND SWA_ETL;
%SWA_ETL;
```

How to Load Data into Custom Directory Locations

If you have a customized directory structure, then you need to modify the locations of the custom directories in the corresponding arguments within the %WAETL macro. The following examples offer some suggestions for setting up your invocation of the %WAETL macro.

Custom Invocation of the %WAETL Macro: Example 1

This example loads the standard parsed Web log from the standard SAS e-Data ETL application directory for the **swamart-pub** Web mart, storing the intermediate data sets in the existing **d:\swa\prod\bmc\swamart-pub\tempstore** directory.

```
%waetl(name=swamart-pub,
        temp_store=d:\swa\prod\bmc\swamart-pub\tempstore,
        program=warehouse
       );
```

Custom Invocation of the %WAETL Macro: Example 2

This example loads the customized parsed Web log called **custom_log_detail** from the customized SAS e-Data ETL application directory (**d:\custom\e_data_etl**) for the Web mart whose root is at **d:\swa\prod\bmc\swamart-pub**, and stores the intermediate data sets in the default temporary directory.

```
libname custom 'd:\custom\e-data-etl\detail';
%waetl(detail=custom.custom_log_detail,
        swamart=d:\swa\prod\bmc\swamart-pub,
        program=warehouse
       );
```

Custom Invocation of the %WAETL Macro: Example 3

This example loads the customized parsed Web log called **custom_log_detail** from the standard SAS e-Data ETL application library (Detail) for the **swamart** Web mart, storing intermediate data sets in the default temporary directory.

```
libname detail 'd:\swa\prod\bmc\swamart\e-data-etl\detail';
%waetl(name=swamart-pub,
        detail=detail.custom_log_detail,
        program=warehouse
       );
```

For more about using the %WAETL macro, see "The %WAETL Macro" on page 270.

Using the %EDATAETL Macro to Extract and Load Web Log Data

When to Use the %EDATAETL Macro

If you customize either the Extract or Load processes by editing one or more of their customer accessible modules (CAMs), then you might want to invoke these programs individually rather than as part of the Daily.sas program.

How to Run the Extract and Load Processes Individually

To run the Extract or Load processes individually, submit the %EDATAETL macro in a SAS interactive session (SAS Program Editor window). Run these processes in the following order:

1 Extract

2 Load

Execute the SAS e-Data ETL Extract and Load processes by submitting the %EDATAETL macro in the SAS Program Editor window as follows:

```
%edataetl(name=webmart-name, program=program-name);
```

For the NAME argument, type the name of your Web mart exactly as it appears in the SAS e-Data ETL Administrator main window. For the PROGRAM argument, valid arguments include EXTRACT or LOAD.

For example, to run the Extract process for the BMC Web mart (the name of the Web mart for the BetterManagement.com Web site as registered in the SAS e-Data ETL Administrator), submit the following code:

```
%edataetl (name=bmc, program=extract);
```

Rarely would you run only one program without running the other program. To run both the Extract and the Load processes, type the %EDATAETL macro twice. For example, to run both the Extract and the Load processes for the BMC Web mart, submit the following code:

```
%edataetl (name=bmc, program=extract);
%edataetl (name=bmc, program=load);
```

However, if an error existed in the Extract process, then the Load process would probably also fail. By executing only the Extract process, then you might check the SAS log for errors before executing the Load process, thereby possibly saving some time.

Allocating Libraries for a Web Mart

If you want only to allocate your libraries for a Web mart, then submit the %EDATAETL macro in the SAS Program Editor window by using the following code:

```
%edataetl (name=<webmartname>);
```

The %EDATAETL macro establishes the required libraries, copies the necessary control tables, and reads the Web mart properties for the Web mart as specified in the SAS e-Data ETL Administrator.

CHAPTER

5

The SAS e-Data ETL Extract Process

Overview of the Extract Process

The purpose of the Extract process is to read the Web server log files (Web logs) and to create SAS detail data sets. The Extract process parses the Web log data into a temporary list of detail data sets in preparation for the Load process.

As it reads each Web log, the Extract process performs the following actions:

- identifies visitor sessions and the clickstream path that each visitor followed
- discovers cookie information if it is present
- identifies requests for executable programs such as CGI parameters
- filters the Web log data, identifying non-page items, requests from special clients, requests from robots and spiders, and requests that generate bad status codes

Functions of the Extract Process

Identifying Available Web Logs

When the Extract process starts, it searches the directories that are specified in the SAS e-Data ETL Administrator and makes a list of the files in each directory. The

Extract process processes all files that it identifies as valid Web server log files in these directories and any subdirectories.

Determining Web Server Type

The Extract process then determines the type of Web server that created the Web log. The following types of Web logs are recognized by default:

- □ Common Log Format (CLF), also known as Combined Log Format or extended Common Log Format, most often associated with Apache Web servers
- □ Extended Log Format (ELF), most often associated with Microsoft Internet Information Server (IIS) Web servers
- □ Netscape/iPlanet, used by Netscape and iPlanet Web servers
- □ Microsoft Proxy, used by Microsoft Proxy servers
- □ IIS (original), used exclusively by Microsoft's personal Web Servers

You can add new or custom Web log formats to the list of formats that can be read by modifying a default definition for one of the recognized Web log formats.

Creating SAS Reader Programs

After the Web logs are located, SAS e-Data ETL software generates SAS programs that read the Web logs. User specifications (such as filtering processes that ignore page requests from spiders or identify non-page requests) that have been defined for the Web mart are included in the code.

Sorting the Data

The Extract process concurrently executes the programs that read the Web logs. Then, the data that is extracted from the Web logs is sorted and combined with the existing data that is stored in detail tables. The detail tables are located in the permanent results directory.

Determining Visitors and Sessions

Overview: The Visitor ID

The task of tracking visitors and sessions has inherent complications that affect all Web log analysis programs. If a visitor accesses a Web server through an Internet gateway, then Internet Protocol (IP) addresses can be assigned dynamically to the visitor from a range of available IP addresses. This situation can arise when the client uses an Internet service provider such as America Online (AOL) or Microsoft Network (MSN). Two individual AOL visitors, for example, might be considered as a single visitor because the AOL server repeatedly uses IP addresses from the same pool of available addresses. Proxy servers can create similar problems.

Many Web sites use cookies to better determine the identity of a visitor and the visitor's session.

In this example of a field from a Web log record, the visitor ID is a string that is set by the SHOPPERID parameter in a cookie:

```
SHOPPERID= FBBICEFBGLAGEIJDCMLDGJGL
```

In the following example, the visitor ID is a combination of the visitor's IP address and the string that describes the visitor's browser and browser version:

```
24.142.56.13Mozilla/4.61+(Win98;+I)
```

How to Set Visitor ID Values from Cookies or CGI Parameters

To determine the value for visitor ID from a cookie or CGI parameter, you must specify the cookie or the CGI parameter in the Value column for the UBIQUITOUS_IDENTIFIERS parameter (located in the Wbconfig.sas7bdat data set) by performing the following steps:

1. Using the SAS Table Editor, select and open the Wbconfig data set (Wbconfig.sas7bdat) in the Control library.
2. Scroll down to the 44th row of the Wbconfig table. The parameter called UBIQUITOUS_IDENTIFIERS is in the Parameter_Name column.
3. In the Value column of this row, enter a value that corresponds to the cookie or CGI parameter. See the Description column in the same row for a list of the options that you can use to specify the value(s) for the UBIQUITOUS_IDENTIFIERS parameter.

Note: For more about the Wbconfig data set, see "How the Customizing Process Works in SAS e-Data ETL Software" on page 60. △

It is also useful (although not required) to use the COOKIES_TO_COPY or CGIPARMS_TO_COPY parameters (also located in the Wbconfig data set) in conjunction with the UBIQUITOUS_IDENTIFIERS parameter. When you enter **`cookies_to_copy`** or **`cgiparms_to_copy`** as the value for the UBIQUITOUS_IDENTIFIERS parameter, the column name that is specified in the COOKIES_TO_COPY or CGIPARMS_TO_COPY parameter is also used as the value for the UBIQUITOUS_IDENTIFIERS parameter.

For example, if you have a cookie that you assign to the User_Site_Logon column using the COOKIES_TO_COPY parameter, and then you set **`cookies_to_copy`** as the value of the UBIQUITOUS_IDENTIFIERS parameter, then the value in the User_Site_Logon column will be prefixed with the letter "U" and will be assigned to the output column Visitor_ID in the Weblog_Detail table.

The e-Data ETL Extract process determines the visitor ID in the following order:

1. If a value for the UBIQUITOUS_IDENTIFIERS parameter exists in the Wbconfig table, then this value is used and is prefixed with the letter "U" by the Extract process.
2. If no value for the UBIQUITOUS_IDENTIFIERS parameter exists, then the standard server cookie parameter, SITESERVER ID, is used and is prefixed with the letter "U" by the Extract process. The SITESERVER ID parameter is located in the cookie parameters (in the Web log) that are prefixed with **`SITESERVER=ID=`**.
3. If the SITESERVER ID cookie parameter is not available, then the value for the Visitor_ID column is generated by using the two output fields Client_ID and User_Agent. This value is prefixed with the letter "N" by the Extract process.

How a Session Is Defined

SAS e-Data ETL software uses the following method to define a session when a more accurate ID source (cookie, authentication, or registration) is not available.

SAS e-Data ETL software counts only sessions that contain at least one page view. The Entry_Point value is a count of the number of sessions in which at least one page was viewed. When the first page is viewed, Entry_Point is set to 1. Non-page views, such as bad status codes or image files, are ignored if they precede the first page. If a session contains no page views, then neither the Entry_Point nor Exit_Point values are set.

How Session Duration Is Defined

By default, if a visitor does not make a new request for a page within 30 minutes of the last request, SAS e-Data ETL software considers the current session ended. SAS e-Data ETL software interprets any additional time that a visitor remains at a page (beyond this 30-minute timeout threshold) as part of a new session for that visitor. To modify the default timeout threshold of a session for your Web mart, see "The Processing Properties" on page 35.

Dates in a Web Log

If the date is not specified for each record within a Microsoft Web log, then the Extract process uses the date in the heading of the log for processing those records.

If a date is not present in the heading of a log that has originated from other Web servers, then the Extract process skips this log.

Modules That You Can Modify in the Extract Process

The modules of the Extract process are described in "Overview: The Modules in the Extract Process" on page 80. The modules of the Extract process that you can modify are called custom access modules (CAMs). The CAMs for the Extract process include the following modules:

- □ User_assignments_after_input
- □ User_assignments_before_output
- □ User_assignments_by_session_id

CAUTION:

Aside from the CAMs listed in this section, you should not modify, customize or override any other modules of the Extract process. △

For the general instructions on modifying any CAM, see "About Custom Access Modules (CAMs)" on page 53. To see the code in any of the following CAMs, see "Overview: The Modules in the Extract Process" on page 80.

User_assignments_after_input CAM

Edit the User_assignments_after_input module to accomplish either of the following operations:

- □ to set variables that are not available until after the Web log is read
- □ to populate any fields that are not recorded directly in the Web log

The statements in this module are executed immediately following the SAS INPUT statement.

User_assignments_before_output CAM

Edit the User_assignments_before_output module to customize selected variables before a record is written to the SAS data sets. This module runs after the URL is parsed into all relevant pieces and after all other processing of the incoming record (within a Web log) is completed.

User_assignments_by_session_ID CAM

Use the User_assignments_by_session_ID module to customize selected variables after the Web log records are sorted by Session_ID and Datetime.

Running the Extract Process

The Extract process is normally executed automatically within the Daily.sas program as part of the ETL processing of your Web log data. To run the Extract process separately from the other ETL programs, invoke the %EDATAETL macro by typing **extract** (in the SAS Program Editor window) as the argument of the PROGRAM parameter as follows:

```
%edataetl(name=mybigcompany, program=extract);
```

The following example demonstrates how to use macro variables when calling the %EDATAETL macro:

```
/* Set up macro variables */

%let weblog_path=C:\MyData\WebLogs;
%let data_store=C:\My Data\MyWebMart\&work;
%let work_area=C:\My Data\MyWebMart\&work\TEMP;
/* Initialize the Web mart */

%edataetl(
  weblog_path=&weblog_path,
  data_store=&data_store,
  work_area=&work_area
);

/* Run Extract program */

%edataetl(
  weblog_path=&weblog_path,
  data_store=&data_store,
  work_area=&work_area,
  program=extract
);

/* Run Load program */

%edataetl(
  weblog_path=&weblog_path,
  data_store=&data_store,
  work_area=&work_area,
  program=load
);
```

CHAPTER 6

Reference Notes for SAS e-Data ETL Extract Process

Overview: The Modules in the Extract Process

The section entitled "The Modules in the Extract Process" on page 80 briefly describes the source entries in the Extract process so that you can identify their general functionality. The source entries are listed in order of data flow.

CAUTION:

The only source entries in the SAS e-Data ETL Extract process that you can modify are the custom access modules (CAMs) that are listed in "About Custom Access Modules (CAMs)" on page 53. △

To modify a CAM, see "About Custom Access Modules (CAMs)" on page 53.

For descriptions of the CAMs in the Extract process and instructions for modifying them, see the following sections:

□ "User_assignments_after_input" on page 83.

□ "User_assignments_before_output" on page 88.

□ "User_assignments_by_session_ID" on page 92.

Note: For the module(s) in the Load process that you can edit, see "User_assignments_for_data_mining" on page 100. △

The section entitled "The Temporary Working Area" on page 94 describes the significance of the temporary working area and describes the outputs that are created in the temporary working area.

The Modules in the Extract Process

Create_formats

The Create_formats module defines the SAS formats that the Extract process uses. Because the data upon which the formats are built changes often (as in the case of the current list of the spiders or robots), the Extract process defines these formats at every invocation to ensure that the most recent changes are included when the logs are read.

The formats are created from SAS tables that are supplied with SAS e-Data ETL software. These tables are located in the Sashelp library, and their names begin with **`WB`**.

Custom_log_format

This module executes any custom code for creating SAS formats that are used by the SAS e-Data ETL Extract process. This code is executed after all of the other formats that are used by SAS e-Data ETL program have been created. Therefore, this code can be used to add new formats or to change the definition of existing formats.

Custom_log_input

This module builds the input statement that reads the custom log records as specified in the call to RXMATCH. It also performs any calculations that handle the formatting for a specific Web log in order to set up values that are expected by subsequent ETL processes.

Custom_log_rxfree

This module executes the call to RXFREE that deallocates the memory that is used by RXPARSE.

Custom_log_rxmatch

This module executes the call to RXMATCH that detects the new Web log format that is specified by the pattern defined in Custom_log_rxparse.

Custom_log_rxparse

This module executes the call to RXPARSE that detects the new Web log format.

Custom_log_set_server_type

This module sets the server type as specified by the results of the call to RXMATCH in Custom_log_rxmatch.

Custom_log_test_harness

This module tests the support of custom Web logs in an environment that is separate from the SAS e-Data ETL environment. The comment blocks located at the top of each of the Custom_log_ source entries (with names that begin with **`Custom_log_`**) contain example code that reads Common Log Format (CLF) log files. For your first custom log, run the code in this example as it is written, and make sure that the program runs successfully before you make any changes to the code. You can use the example data file called **`clf.log`**, which is located in Sasmisc, to test the example source entries.

Determine_available_weblogs

The Determine_available_weblogs module searches for the Web log location that the user specified during the Web mart setup in order to find all Web logs that need to be processed. If the specified location is a directory, then the module searches all files and all subdirectories within that directory for potential Web logs.

User_assignments_for_data_mining

This module transforms the fields of your Web log to make them more effective for data mining functions. Statements that are added here are executed automatically at the end of Load processing.

Determine_weblog_type

The Determine_weblog_type module examines the list of Web logs and determines the type of Web server that created these logs. This module provides useful feedback early in the module execution about the types of files that are being processed. It does not examine compressed Web log files; it only recognizes them as files to be processed.

If a Web log file is in an unknown format, then the module displays a warning in the SAS log and continues to the next Web log. The possible corrective action depends upon whether this file is in a format that SAS e-Data ETL software cannot recognize but should process anyway, or whether the unrecognized file is not actually a Web log and should be removed from the directory.

Extract_build_filenames

The Extract_build_filenames module constructs the proper SAS FILENAME statement that is required for processing each Web log file. The type of FILENAME statement that is generated might vary depending on whether or not the file is compressed and whether or not DNS resolution is specified.

Save_macro_variable_settings

The Save_macro_variable_settings module creates a file that contains all the properties of the Web mart so that these properties can be accessed by the concurrent SAS programs.

Job_build__main

The Job_build__main module (note the double underscore after *build*) builds the SAS reader programs with other modules, applying the parameters of the currently selected Web mart.

The logic modules (Special_client_logic, Spider_client_logic, Visitor_ID_logic, and Status_code_logic) apply filtering criteria in the following order:

1 special clients
2 spider clients
3 non-page views
4 bad status codes

Incoming records are divided into groups, or buckets, for more efficient processing by the system. All the concurrent SAS programs use the same algorithm for dividing the records into buckets. All of the information that pertains to a particular visitor can be merged again, even if the records that constitute the visit to the Web site are distributed among different servers or Web logs.

The Job_build__main module includes the rest of the modules in this subsection, from Job_build__inputs through Sort_buckets.

Job_build__inputs

The Job_build__inputs source entry (a double underscore exists between “build” and “inputs” in the name of this source entry) generates the SAS code that is required for reading each specific Web log. This module opens each log, determines the type of Web server that created the log, and determines the fields that are available in the Web log.

The SAS INPUT statement is created by comparing the variables in a Web log with the information in the Control.Wbfields table.

If a compressed Web log contains other Web logs, then all the files must be of the same log format and must contain the same variables. If the Web server type cannot be determined by comparing the first 30 lines of input, then the Web log is not processed, and the program skips to the next Web log.

You can modify the Job_build__inputs module in order to accommodate any Web log formats that the Extract process might encounter but would not recognize.

Job_build__data_attributes

The Job_build__data_attributes module generates the SAS ATTRIB statements based on the information that is contained in the Control.Wbfields table.

User_assignments_after_input

The User_assignments_after_input module can be used in one of two ways: to set variables that are not available until after the Web log is read, or to populate any fields that are not recorded directly in the Web log.

The statements in this module are executed immediately after the SAS INPUT statement.

For the general instructions on modifying any CAM, see “About Custom Access Modules (CAMs)” on page 53.

The default code in the User_assignments_after_input CAM follows:

```
*---------------------------------------------------------------------------;
* USER_ASSIGNMENTS_AFTER_INPUT                                              ;
*                                                                           ;
* This section executes any custom code for variable assignments            ;
* or definitions.  Any code statements included in this section             ;
* will be executed during reading of the web server log files               ;
* immediately after fields are read from the web log and before             ;
* any other processing (such as datetime manipulation or URL parsing),      ;
* To include statements after standard processing and just before the      ;
* observations are output, use module USER_ASSIGNMENTS_BEFORE_OUTPUT.      ;
*                                                                           ;
*                                                                           ;
* NOTE: Any statements included here will be included in the middle of a    ;
*       SAS Data step program.  Do not include any SAS procedures or        ;
*       complete data step programs.                                        ;
*                                                                           ;
*       example of valid statements:                                        ;
*          my_variable = username || date;                                  ;
*          if upcase(combined_url) = "/my/special/file.htm" then            ;
*             pagecnt = sum(pagecnt, 1);                                    ;
*          if client_id eq ’0.0.0.0’ then delete;                           ;
*                                                                           ;
*---------------------------------------------------------------------------;
* CHANGES:                                                                  ;
*                                                                           ;
* 20021029 cabahl Added GOMEZAGENT                                    C001  ;
* 20021114 cabahl added CONTYPE                                       C002  ;
* 20021204 cabahl consolidated the user agent exclusion               C003  ;
* 20030103 cabahl added logic to rename gif files for auto quote      C004  ;
*                 mapping                                                   ;
* 2003019 cabahl  added logic to rename gif files used for auto       C005  ;
*                 autoquote mapping 2/4/03 redesign                         ;
*===========================================================================;


   /* Make any custom definitions and assignments here */

/* C003--remove known agents that are not wanted: */
%itv_remove_agents;

if user_cookie = ’-’ then user_cookie=’’;

/* C004 - renaming gifs
PreScreen Page =  images/jspnames/preScreen.gif
PreScreen Knockout - Thank You Major Violation = images/jspnames/thankYouMajorVio.gif
PreScreen Knockout - Thank you no Driver 50 or over = images/jspnames/thankyouNoDriver50.gif
PreScreen Knockout - Thank You_rfq = images/jspnames/thankyou_rfq.gif
PreScreen Knockout - Thank You - Mass. = images/jspnames/thankyou_ma.gif
PreScreen Knockout - Thank You - NJ = images/jspnames/thankyou_nj.gif
Start (State Select) = images/jspnames/Welcome.gif
Premium Page = images/jspnames/premiumSummary.gif
*/

*combined_url = lowcase(combined_url);

if index(combined_url,’images/jspnames/’) > 0 then do;
   select(lowcase(combined_url));
     when(’images/jspnames/prescreen.gif’)          combined_url=’images/jspnames/prescreen.html’;
     when(’images/jspnames/thankyoumajorvio.gif’)   combined_url=’images/jspnames/thankyoumajorvio.html’;
     when(’images/jspnames/thankyounodriver50.gif’) combined_url=’images/jspnames/thankyounodriver50.html’;
     when(’images/jspnames/thankyou_rfq.gif’)       combined_url=’images/jspnames/thankyou_rfq.html’;
     when(’images/jspnames/thankyou_ma.gif’)        combined_url=’images/jspnames/thankyou_ma.html’;
     when(’images/jspnames/thankyou_nj.gif’)        combined_url=’images/jspnames/thankyou_nj.html’;
     when(’images/jspnames/welcome.gif’)            combined_url=’images/jspnames/welcome.html’;
     when(’images/jspnames/premiumsummary.gif’)     combined_url=’images/jspnames/premiumsummary.html’;
     otherwise;
   end;
end;

/* C005 - 2/04 redesign */
if index(lowcase(combined_url),’/sales/images/jsptracker.gif’) > 0 then combined_url=tranwrd(combined_url,’gif’,’jsp’);
if index(lowcase(combined_url),’/service/images/jsptracker.gif’) > 0 then combined_url=tranwrd(combined_url,’gif’,’jsp’);
```

```
*------------------------------------------------------------------------;
* END of USER ASSIGNMENT AFTER INPUT                                     ;
*------------------------------------------------------------------------;
```

Determine_server_and_sitename

The Determine_server_and_sitename module populates the Server field and Sitename field from information that is contained in the Web log if that log contains these fields.

The Server field indicates the name of the physical system on which the Web server is running. The Sitename field indicates the name of the virtual Web site. One server can host many Web sites. Conversely, one Web site can use many physical Web servers.

Main_variable_assignments

The Main_variable_assignments module translates the variables as they are read from the Web log into the set of known variables.

Datetime_logic

The Datetime_logic module sets the following variables:

```
Date                = datepart(datetime);
Week                = (Date-weekday(Date+1);
Month               = mdy(Month(Date),1,Year(Date));
Quarter             = mdy(Month(Date)-mod(Month(Date)-1,3),1,Year(Date));
Year                = mdy(1,1,Year(Date));
Hour                = Hour(timepart(Datetime));
Day_Of_Week         = weekday(Date) + 1;
```

Where 1=Sunday, 2=Monday, 3=Tuesday, and so on. These variables do not need to be SAS date values.

Special_client_logic

Log records from clients that are listed in the Control.Wbspecl table are processed by the Special_client_logic module, based on the setting of the Special_Client_Action property. Here are the possible values of the Special_Client_Action property:

- □ `Skip` (discard and ignore the records from this client)
- □ `Keep` (process the records from this client normally)
- □ `Tally` (count the number of records that are received from this client, but do not process them further)

Spider_client_logic

SAS e-Data ETL software automatically detects spiders and robots when they access the `robots.txt` file. If the value of the Number_Of_Auto_Spiders field in the Control.Wbspider table is greater than zero, then any Web log records that contain clients that are designated as spider or robot programs are processed by the S pider_client_logic module. This module processes the spiders and robots based on the setting of the Spider_Client_Action property.

This property is set by using the SAS e-Data ETL Administrator, and the value is saved in the Control.Wbconfig table. Clients that are identified as robots or spiders are stored in the Control.Wbspider table.

See “Special_client_logic” on page 85 for the possible values of Spider_Client_Action property.

For information about enabling SAS e-Data ETL software to recognize new spider clients, see “Spiders” on page 47.

Visitor_ID_logic

SAS e-Data ETL software tracks a visitor through a Web site by using the Visitor_ID variable. The Visitor_ID variable is set as the value of the Ubiquitous Identifier if this option is set. Otherwise, the Visitor_ID variable is set as the value of the cookie SITESERVER=ID (if this cookie is present) or as the value of the cookie ASPSESSIONID (if this cookie is present). If neither of these cookies is present in the Web log, then the value of Visitor_ID is set as the combination of the Client_ID field and the User_Agent field. For more about Visitor_ID, see “Determining Visitors and Sessions” under “Overview of the Extract Process” on page 73.

SAS e-Data ETL software does not use the Username (or authenticated user) field as the Visitor_ID, because the Username field in a Web log is often left blank. The server can determine the Username only for those URLs that have established authentication procedures. For example, if a visitor logs on to a site (realm) that requires authentication such as a user ID from the visitor, the server can record that ID label as the Username for as long as the visitor remains within that authenticated realm. If all the sessions of interest exist within the authenticated realm that is served by the Web server, then the Visitor_ID variable can be set to equal the value of the Username field.

Determine_executable_program

The Determine_executable_program module examines the current request to determine whether the request is for an executable program, such as a CGI program. The module makes this determination by checking for CGI parameters, by checking the file extension of the requested file, and by checking the URL against the list of files and directories that are specified in the Control.Wbpgms table.

To add new types of executable programs to the Control.Wbpgms table, modify the list of files and directories by following the directions under “CGI Program Locations” on page 43.

Parse_main_url

The Parse_main_url module examines the requested URL and parses it into constituent pieces, based on the standard that is provided by RFC 2396 and by the Internet Draft of the WWW Common Gateway Protocol Version 1.1.

The Parse_main_url module also contains the logic that determines whether a request should be counted as a hit or a page view. The determination is based on the information in the Control.Wbpagvew table and the file extension of the requested object. For example, if a requested file has an extension of HTM, then this request is counted as a page view. The File_Type field, which represents the MIME type of the requested object, is also set in this module.

Note: SAS e-Data ETL software counts all executable programs as page views, regardless of their file extensions. △

Parse_referrer

The Parse_referrer module examines the Referrer field and parses it into its constituent pieces, based on the standard that is provided by RFC 2396 and by the Internet Draft of the WWW Common Gateway Protocol Version 1.1.

Page_logic

The Page_logic module acts on the request as specified in the setting of the Non_Page_Views property, which you can set by using the SAS e-Data ETL Administrator. Items that are typically considered to be non-page views include graphics files. Most graphics files for Web display have a GIF or JPG extension. See "Non-Pages" on page 48 for more information about customizing a Web mart to recognize additional types of non-pages.

Here are the possible values of the Non_Page_Views property:

□ `Skip` (ignore and discard the request for the non-page object)

□ `Keep` (process the request for the non-page object normally)

□ `Tally` (count the number of non-page object requests, but do not save detailed information about the request)

Status_code_logic

The Status_code_logic module responds to the page request as specified in the settings of the Status_Code_Action property, which you can set by using the SAS e-Data ETL Administrator. This module bases its action on whether the server indicates an error when it attempts to fulfill a page request. See "Bad Status Codes" on page 49 for more information about how SAS e-Data ETL software reports the page requests that return errors.

Here are the possible values of the Status_Code_Action property:

□ `Skip` (ignore and discard the request for the page)

□ `Keep` (process the request for the page normally)

□ `Tally` (count the number of page requests, but do not save detailed information about the request)

□ `ForceNonPageView` (treat the request the same as any other request for a non-page object)

Determine_browser_and_platform

The Determine_browser_and_platform module parses the User_Agent variable into variables that represent the following information:

□ the browser program that is being used by the visitor

□ the browser version

□ the operating system on which the visitor's machine is running

Because no formatting standards currently exist for specifying a browser and its version, the Determine_browser_and_platform module consults a list of known values in order to determine the visitor's browser and platform. The list of known values is kept in the Control.Wbbrwsr and Control.Wbpltfrm tables.

Determine_organization

If the visitor's IP address was resolved into a host name by using DNS lookup, then the Determine_organization module parses the host name into organization and organization type. If the host name contains the country and state, then the visitor's country and U.S. state are also retrieved.

Determine_unique_url

The Determine_unique_url module is used to build the path or sequence of page requests as a visitor travels through a Web site. For advanced usage, you can customize this module in order to assign meaningful phrases to this sequence, such as a name for the path of a purchase activity by a customer.

User_assignments_before_output

The User_assignments_before_output module is supplied if you want to customize selected variables before a record is written to the SAS tables. While the Extract process is running, this module is positioned to run after the URL is parsed into all relevant pieces and after all other processing of the incoming record (within a Web log) is completed.

The default code in the User_assignments_before_output CAM follows:

```
*---------------------------------------------------------------------------;
* USER_ASSIGNMENTS_BEFORE_OUTPUT                                            ;
*                                                                           ;
* This section executes any custom code for variable assignments            ;
* or definitions.  Any code statements included in this section             ;
* will be executed during reading of the web server log files               ;
* immediately before data is written out to SAS data sets and after        ;
* any other processing (such as datetime manipulation or URL parsing),      ;
* To include statements before standard processing and just after the       ;
* data is read from the web log file, use module                            ;
* USER_ASSIGNMENTS_AFTER_INPUT.                                             ;
*                                                                           ;
*                                                                           ;
* NOTE: Any statements included here will be included in the middle of a    ;
*       SAS Data step program.  Do not include any SAS procedures or        ;
*       complete data step programs.                                        ;
*                                                                           ;
*       example of valid statements:                                        ;
*          my_variable = username || date;                                  ;
*          if upcase(requested_file) = "/my/special/file.htm" then          ;
*             pagecnt = sum(pagecnt, 1);                                    ;
*          if client_id eq ’0.0.0.0’ then delete;                           ;
*                                                                           ;
* 20020820 ccb add logic to create a spider flag                            ;
*===========================================================================;


   /* Make any custom definitions and assignments here */
 *-----------------------------------------------------------;
 * Check for the existence of name/value pair for module=    ;
 * cgi parameter.  If a value exists, add it to the          ;
 * requested_file variable.   This will enhance the page     ;
 * view reports.                                             ;
 *-----------------------------------------------------------;
 requested_file=lowcase(requested_file);
```

```
*-----------------------------------------------------------;
* Create variables for business segments.  If a session has ;
* a hit to the url level below, tag that session as being   ;
* part of the segment.  The next three variables will be    ;
* propagated throughout the session via  wbconfig field     ;
*                                                           ;
* Webhound won’t allow us [yet] to propagate numeric fields ;
* so use character values.   Each segment variable must     ;
* also be defined in wbfields.                              ;
*-----------------------------------------------------------;
url_level_1=lowcase(url_level_1);
url_level_2=lowcase(url_level_2);
url_level_3=lowcase(url_level_3);
url_level_4=lowcase(url_level_4);
url_level_5=lowcase(url_level_5);
url_level_6=lowcase(url_level_6);
url_level_7=lowcase(url_level_7);
url_level_8=lowcase(url_level_8);


*----------------------------------------------------------;
* Set spider flag - uses the spider flag and other         ;
* indicators to determine whether or not a detail record   ;
* is a spider                                              ;
* variables created -                                      ;
*   spider: 0 (not a spider)                               ;
*           1 (spider)                                     ;
*           .5 (potentially a spider)                      ;
*----------------------------------------------------------;
spider=0;
/* determine if client_id exists in WBSPIDER if so then set spider to 1 */
if input(client_id,spider.) then spider=1;

*---------------------------------------------------------;
* if client_id not in wbspider determine if other client  ;
* id or requested_file meet spider criteria               ;
*---------------------------------------------------------;
if spider = 0 then
  if
     /* check requested_file to see if robot */
     index(lowcase(requested_file),’robot’) > 0 then spider = 1;

/* check user_agent to see if contains suspicuous text */
if spider = 0 then
  if index(lowcase(user_agent),’seeker’) > 0
     or index(lowcase(user_agent),’finder’) > 0
     or index(lowcase(user_agent),’index’) > 0
     or index(lowcase(user_agent),’crawler’) > 0
     or index(lowcase(user_agent),’lwp’) > 0
     or index(lowcase(user_agent),’arach’) > 0
     or index(lowcase(user_agent),’spider’) > 0
     or index(lowcase(user_agent),’bot’) > 0
     or index(lowcase(user_agent),’borg’) > 0
     then spider=.5;
```

```
*-------------------------------------------------------------------------;
* END of USER ASSIGNMENTS BEFORE OUTPUT                                   ;
*-------------------------------------------------------------------------;
```

Output_referrer_query_parms

The Output_referrer_query_parms module creates the ReferrerParms table. This table contains the CGI parameter information from the Referrer field in the Web log. This module parses the CGI parameters into name-value pairs. Then the module writes each name-value pair to a SAS table that contains the Parameter_Name, Parameter_Value, and Record_ID fields. These fields can be used to map the name-value pair back to the original record of the Web log.

Output_referrer_pathinfo_parms

The Output_referrer_pathinfo_parms module creates the ReferrerParms table. This table contains the PATH_INFO parameter from the Referrer field in the Web log. This module parses the PATH_INFO parameters into name-value pairs. Each name-value pair is written to a SAS table that contains the Parameter_Name, Parameter_Value, and Record_ID fields. These fields can be used to map the name-value pair back to the original record of the Web log.

Output_cookie

If cookie parsing is requested, then the Output_cookie module builds the SAS statements necessary to create the tables that will contain a visitor's cookie information. The cookie information is obtained from the Web log.

The Output_cookie module parses a cookie string into its name-value pairs. Then the module writes each name-value pair to a SAS table that contains the Parameter_Name, Parameter_Value, and Record_ID fields. These fields can be used to map the name-value pair back to the original record of the Web log.

See "The Detail Tables Properties" on page 31 for information on modifying the Number_of_cookie_datasets property.

Output_query_string_parms

The Output_query_string_parms module creates the CGIParms table. This table contains CGI parameter information that is discovered in the Web log. This module parses any CGI parameter (also referred to as the query string) or PATH_INFO parameters that are contained in the Web log records into name-value pairs.

Then the module writes each name-value pair to a SAS table that contains the Parameter_Name, Parameter_Value, and Record_ID fields. These fields can be used to map the name-value pair back to the original record of the Web log.

Note: The CGI parameter information for records that have been filtered by using the **Skip** or **Tally** value (of the Number_Of_CGI_Parms_Datasets property) is not written to the SAS tables. △

Output_pathinfo_parms

The Output_pathinfo_parms module creates the CGIParms table. This table contains the PATH_INFO parameter from the URL field of the Web log. This module parses the PATH_INFO parameter into name-value pairs. Each name-value pair is written to a SAS table that contains the Parameter_Name, Parameter_Value, and Record_ID fields.

These fields can be used to map the name-value pair back to the original record of the Web log.

Job_build__output_logic

The Job_build__output_logic module contains the logic to divide the requests into separate buckets based on the IP address or host name of the visitor. The incoming records are divided into the number of groups that are specified as the number of restructure buckets in SAS e-Data ETL Administrator. See "Number of Restructure Buckets" on page 52 for information about changing the number of restructure buckets.

Note: If you choose to edit the tables directly rather than using the SAS e-Data ETL Administrator interface, the number of restructure buckets is defined in the Number_Of_Analysis_Buckets property. △

Sort_buckets

Web server data is divided into SAS tables according to the value specified as the number of restructure buckets. See "Number of Restructure Buckets" on page 52 for information about changing the number of restructure buckets. The Sort_buckets module sorts each of these tables so that they can be merged with the corresponding buckets or tables that are created by any other concurrent processes.

Note: If you choose to edit the tables directly rather than using the SAS e-Data ETL Administrator interface, the number of restructure buckets is defined in the Number_Of_Analysis_Buckets property. △

Extract_initialization

The main function of the Extract_initialization module is to initialize the parallel SAS sessions. This preparation includes starting the parallel SAS sessions and copying any necessary programs or formats, such as the macros that are used to parse a visitor's URL into its constituent pieces.

Execute_the_data_read_jobs

The Execute_the_data_read_jobs module starts the number of concurrent SAS sessions for reading the Web logs. The Number_Of_Parallel_Reader_Jobs property, which is set by using the SAS e-Data ETL Administrator, specifies the number of parallel SAS sessions (jobs) to be opened, unless there are fewer Web logs to be read than the specified number of parallel jobs.

Each session runs the generated SAS programs to read the Web logs. Each SAS session or reader job has a corresponding SAS program, Read_job*n*.sas (which is located in a directory that is used for storing temporary files), and a corresponding log file, Read_job*n*.file*x*.log (which is contained in the **`saslogs`** directory, a directory that is used for storing permanent results). The tables and the information that are particular to each job (1, 2, 3...*n*) are located in a directory named **`Read_Jobn`**, which is located in the temporary file directory (see "Overview: The Temporary Working Area" on page 94).

When each concurrent SAS session is completed, SAS logs are searched for any errors or warnings. Any condition that results in an error causes the Extract process to stop processing.

Aggregate_job_datasets

The Aggregate_job_datasets module joins all the restructure buckets together so that session information can be determined by the analysis jobs. This module also joins the Job_Statistics tables from each parallel SAS session and adds this information to the Job_Statistics table.

Execute_the_analysis_jobs

The Execute_the_analysis_jobs module operates similarly to the Execute_the_data_read_jobs module. However, in the Execute_the_analysis_jobs module, the number of concurrent SAS sessions is controlled by the Number_Of_Parallel_Analysis_Jobs property (rather than the Number_Of_Parallel_Reader_Jobs property). Unless the system that is running SAS e-Data ETL software has an extraordinary amount of memory, the value of this property will be less than the value of the Number_Of_Parallel_Reader_Jobs property.

The concurrent SAS sessions analyze the groups of SAS tables that contain the Web server information. Each SAS session that is run from this module is designated as an analysis job. Each session has a corresponding log file, **`Analysis_jobn.log`**, which is contained in the **`saslogs`** directory.

Each analysis job has its own directory by bucket. The tables are contained in in the temporary file location directories named **`Analysis_Job1`** through **`Analysis_Job10`** (see “Analysis_job*n*.sas” on page 95), assuming that the default setting for the Number_Of_Analysis_Buckets property is 10.

Note: In the SAS e-Data ETL Administrator, the Number_of_analysis_buckets property is specified as the number of restructure buckets. See “Number of Restructure Buckets” on page 52 for information about changing the number of restructure buckets that are specified in the SAS e-Data ETL Administrator. △

The User_assignments_by_session_id module is invoked during the execution of the Execute_the_analysis_ jobs module immediately before each observation is output.

Aggregate_analysis_stats

The Aggregate_analysis_stats module joins job statistics information from each analysis job.

User_assignments_by_session_ID

The User_assignments_by_session_ID module is supplied in the event that you want to customize certain variables after the Web log records are sorted by Session_ID and Datetime.

The User_assignments_by_session_id module is invoked during the execution of the Execute_the_analysis_ jobs module, which is immediately before each observation is written to the data sets.

The default code in the User_assignments_by_session_ID module follows:

```
*-------------------------------------------------------------------;
* User_Assignments_By_session_id                                    ;
*                                                                   ;
*  Set variables based on SESSION_ID value and on the fact       ;
*  that the data is sorted by SESSION_ID at this point.          ;
*  This macro is invoked during the analysis jobs of Extract     ;
*  just before each observation is output.  If pathing data sets ;
*  are being created, this logic can also access the value       ;
*  of the variables in the PATH variable as it is being          ;
*  constructed.  The observations at this point are grouped by   ;
*  Session_ID but the BY statement cannot be used, so use the    ;
*  the internal variables _lastSessionNum and _firstSessionNum   ;
*  to detect the start or end of a group of                      ;
*  observations in the same session.  For example:               ;
*                                                                ;
*    if _firstSesssionNum then do;                               ;
*       * First click of session processing goes here;           ;
*    end;                                                        ;
*                                                                ;
*    if _lastSesssionNum then do;                                ;
*       * Last click of session processing goes here;            ;
*    end;                                                        ;
*                                                                ;
*    Note that any variables that you have declared in           ;
*    WBFIELDS cannot be RETAINed in this code (they will have    ;
*    been established in earlier processing and will inherit a   ;
*    missing value in every observation), so if you              ;
*    want to retain values of one of these variables between     ;
*    successive observations, use a temporary variable name      ;
*    to retain the value and assign the WBFIELDS-defined         ;
*    name directly.  For example, to make a sequence counter     ;
*    of the clicks in a session, you might code something        ;
*    like this (SESSION_SEQUENCE has been defined in WBFIELDS    ;
*    but TEMP_SESSION_SEQUENCE has not):                         ;
*                                                                ;
*       retain temp_session_sequence 0;                          ;
*       drop   temp_session_sequence;                            ;
*       if _firstSessionNum then                                 ;
*          temp_session_sequence = 0;                            ;
*       temp_session_sequence = temp_session_sequence + 1;       ;
*       session_sequence = temp_session_sequence;                ;
*                                                                ;
*-------------+--------------------------------------------------;
* History:    |                                                  ;
* 01Jan2002   |S0127106 Update comment for reheading (df)        ;
* 19Nov2003   |Added logic to handle pdf pages appropriately     ;
*             |cabahl                                     C0001  ;
* 03Dec2003   |Added file size check                      C0002  ;
*----------------------------------------------------------------;


   /*-------------REHEADING----------------------------------------*/
   /* Code statements that detect end or beginning of session here */

   retain temp_session_sequence 0;
   drop   temp_session_sequence;
   if _firstSessionNum then
```

```
      temp_session_sequence = 0;
   temp_session_sequence = temp_session_sequence + 1;
   session_sequence = temp_session_sequence;
   /*--------------------------------------------------------------*/

   *--------------------------------------------------------------------------;
   * C001: Many of the PDF files were coming up with multiple page_counts due ;
   *       to partial loading (status code 206) or caching (status code 304-- ;
   *       not modified). Typically the first hit would have                  ;
   *       status code 200 and subsequent hits would have status code 206     ;
   *       partial load) or status code 304 (cached/not modified). However,   ;
   *       sometimes the first hit would have status code 304 or 206.         ;
   *                                                                          ;
   *       To remedy this, set page count to 0 for PDF files with status code ;
   *       304 or 206 if the previous record in the session is that same PDF  ;
   *       file.   This will remove the subsequent hit from page view reports ;
   *       but not from status code reports.                         C001     ;
   *--------------------------------------------------------------------------;
   if length(requested_file) > 4 then do;
     if compress(session_id) = compress(lag(session_id)) and
        lowcase(compress(lag(requested_file)))=lowcase(compress(requested_file)) and
        substr(lowcase(requested_file),length(requested_file)- 3)='.pdf' and
        status_code in (206,304)
     then page_count=0;
   end;

%mend user_assignments_by_session_id;
```

Check_sas_logs_for_metrics

The Check_sas_logs_for_metrics module examines the SAS log files from the concurrent SAS sessions and searches for information that is related to the amount of data that is processed during each SAS session. This information is then added to the performance information that is contained in the Job_Statistics table, which is located in the permanent results directory.

The Temporary Working Area

Overview: The Temporary Working Area

The temporary working area is populated by the Extract process and is used as input to the Load process. The temporary working area is used to store SAS tables and programs that are created and used during the execution of the Extract and the Load processes.

Read_job*n*.sas

The Read_job*n*.sas modules contain the SAS statements that are generated by the Extract process. They are used to read the Web logs. These programs are executed by the concurrent SAS sessions that have been started in the Execute_the_data_read_jobs module. The concurrent SAS sessions are started by using SAS MP/CONNECT. The concurrent SAS sessions are sometimes referred to as *child processes*. The Number_Of_Parallel_Reader_Jobs property determines the number of Read_job*n* files.

Analysis_job*n*.sas

The Analysis_job*n*.sas modules contain the SAS statements that are generated by the Extract process in the analysis phase. These programs are executed by the concurrent SAS sessions that are started in the Execute_the_analysis_jobs module (see "Execute_the_analysis_jobs" on page 92. The concurrent SAS sessions are started by using SAS MP/CONNECT. The Number_Of_Parallel_Analysis_jobs property determines the number of Analysis_Job*n* files.

CHAPTER

7

The SAS e-Data ETL Load Process

Overview of the Load Process

The Load process combines the Web server data that the Extract process reads with the existing data in the Web mart.

Note: The Load process permanently updates the Web mart. Make sure that the Web mart is backed up on a regular basis. △

When the Load process runs, it performs the following actions:

- creates new generations of the Weblog_Detail data sets and Pathing data sets
- creates the Unique_URLs data set

Note: By default, the number of generations specified for the Cookies data sets, the CGIParms data sets, and the ReferrerParms data sets is 0. Unless you change this default value, the Load process does not create generations of these data sets. △

With each invocation, the Load process updates some of the files in the `detail` directory.

Note: Save the contents of the subdirectories in the `detail` directory before each run of the Load process. If the Load process fails, then you can recover your detail information from your backup. △

Your Web mart's `checkpt` directory lists the contents of these directories from the last Load process. This information can help you determine which files were updated during a failed run of the Load process. However, you might prefer to use the backup directories to restore the entire contents of your `detail` directory.

The Modules of the Load Process

The modules of the Load process are described in "The Modules in the Load Process" on page 100. Except for the User_assignments_for_data_mining custom access module (CAM), you should not modify, customize, or override the modules in the Load process.

The User_assignments_for_data_mining CAM

The User_assignments_for_data_mining CAM is specifically created to enable you to modify its code. This module prepares your Web log data for data mining by creating a view called Detail.Web_Mine that contains all the current Web log detail data for page views. This module is executed automatically at the end of the Load process. In order to use data mining effectively, customize the code of this module to meet the needs of your organization.

Running the Load Process

The Load process is normally executed automatically within the Daily.sas program as part of the ETL processing of your Web log data. To run the Load process separately from the other ETL processes, invoke the %EDATAETL macro by typing **`load`** (in the SAS Program Editor window) as the argument of the PROGRAM parameter as follows:

```
%edataetl(
  weblog_path=C:\Weblogs,
  data_store=C:\Webmart,
  work_area=C:\Webmart\Temp,
  program=load);
```

CHAPTER 8

The Modules in the SAS e-Data ETL Load Process

Overview: The Modules of the Load Process

The section entitled "The Modules in the Load Process" on page 100 describes the operation of each module in the Load process. Except for the User_assignments_for_data_mining module, you should not modify, customize, or override the modules in the Load process. Understanding the operation of each module is not necessary for general use, but it might be valuable when you troubleshoot the SAS e-Data ETL processes.

CAUTION:

The only module in the Load process that you can modify is the User_assignments_for_data_mining custom access module (CAM). △

For a description of the User_assignments_for_data_mining CAM, see "User_assignments_for_data_mining" on page 100. To modify the User_assignments_for_data_mining CAM, see "About Custom Access Modules (CAMs)" on page 53.

Note: For the modules that you can modify in the SAS e-Data ETL Extract process, see Chapter 6, " Reference Notes for SAS e-Data ETL Extract Process," on page 79. △

The section "Contents of the Detail Library" on page 105 describes the detail tables that are created by the Load process.

The Modules in the Load Process

Checkpoint_load_init

The Checkpoint_load_init module creates a list of all the files in the Summary and Detail libraries as well as the contents of the Control library. This list is stored in the Web mart's **checkpt** directory before Load processing begins. You can use these lists to help recover your data if a system failure occurs during Load processing.

Create_generations

The Create_generations module creates new generations of the Weblog_Detail, CGIParms, Cookies, Pathing, and ReferrerParms tables in the Detail library. The generations of these tables are not bound to specific time spans. Each generation of a detail table corresponds to the amount of data that is processed during a single execution of the Load process.

Create_unique_urls

The Create_unique_urls module assigns a value to the Unique_URL property. This value is used only for the Pathing table. By default, the Requested_File URLs (with any parameters stripped off) are recorded for all page views in the Pathing table.

Clean_up_working_areas

The Clean_up_working_areas module removes all the temporary tables that are created in your Web mart's temporary working area. These tables are created during Extract processing to be used during the Load process.

User_assignments_for_data_mining

The User_assignments_for_data_mining module is executed automatically at the end of the Load processing. This module prepares your Web log data for data mining. It creates a view called Detail.Web_Mine that contains all the current Web log detail data for page views. To be effective for data mining, the code in this module needs to be customized to meet the needs of your site.

For the general instructions for modifying any CAM, see "About Custom Access Modules (CAMs)" on page 53.

The default code in the User_assignments_for_data_mining CAM follows:

```
/*======================================================================*
 * User_Assignments_For_Data_Mining
 *
 * This section does setup of the weblog data for data mining.
 * It is used to transform the fields of the input
 * web log to make them most effective for data mining.
 * Statements added here will be executed automatically at the end
 * of Load processing.
 *
 * The example coded here creates a view of all the current web log
 * detail data for page views, including CGI parms and
 * cookies and surfaces them in separate columns so they are ready
 * for easy selection via SAS Enterprise Miner.
 *
 * You will probably want to change this behavior somewhat at your site.
 * For example, you might want to use only a part of the REQUESTED_FILE
 * or you might want to exclude some or all of the CGI parms or cookies
 * (for best performance, if you do not intend to use any cookie or
 * CGI parm values, you should comment out the code here that reads
 * the CGI parm and/or cookie data set).  If the number of CGI parms
 * or cookies for any single line of web log exceeds the array size
 * declared here, update the cgiparm_count_max, cookie_count_max,
 * or referrer_parm_max values that appear below.
 *
 *---------------------------------------------------------------------------*
 *
 * History:
 *  27Aug2001 S0123412 Creation (df)
 *
 *===========================================================================*/



%Section_Header(Section=User_Assignments_For_Data_Mining) ;


* Create a view of weblog detail data with parms and cookies;

data    DETAIL.WEB_MINE
 / view=DETAIL.WEB_MINE ;



   *--------------------------------------------------------------------------;
   * This view contains all the columns in WEBLOG_DETAIL plus arrays for      ;
   * CGI parms, referrer parms, and cookies.  Change the max values           ;
   * for each of these if you have more values than will fit in the default.  ;
   *--------------------------------------------------------------------------;
   %let  cgiparm_count_max      = 32;
   %let  cookie_count_max       = 32;
   %let  referrerparm_count_max = 32;

   %put %qcmpres(%wbnote(USER_ASSIGNMENTS_FOR_DATA_MINING) A maximum of
                 &cgiparm_count_max CGI parms for each request will be used for data mining.);
   %put %qcmpres(%wbnote(USER_ASSIGNMENTS_FOR_DATA_MINING) A maximum of
                 &referrerparm_count_max referrer parms for each request will be used for data mining.);
   %put %qcmpres(%wbnote(USER_ASSIGNMENTS_FOR_DATA_MINING) A maximum of
                 &cookie_count_max cookies for each request will be used for data mining.);

   attrib cookies_read       label="Cookies read for this record";
   attrib cookies_kept       label="Cookies kept for this record";
   attrib cgiparms_read      label="CGI/PathInfo parms read for this record";
   attrib cgiparms_kept      label="CGI/PathInfo parms kept for this record";
   attrib referrerparms_read label="CGI/PathInfo parms read for this record";
   attrib referrerparms_kept label="CGI/PathInfo parms kept for this record";

   array cgiparm_name  { &cgiparm_count_max } $64;
   array cgiparm_value { &cgiparm_count_max } $1000;
   array cgiparm_source{ &cgiparm_count_max } $1000;             * Valid values are
                                                                   "Query_String" or
                                                                   "Path_Info";
   array referrerparm_name  { &referrerparm_count_max } $64;
   array referrerparm_value { &referrerparm_count_max } $1000;
   array referrerparm_source{ &referrerparm_count_max } $1000;   * Valid values are
                                                                   "Referrer_Query_String" or
                                                                   "Referrer_Path_Info";
   array cookie_name   { &cookie_count_max } $64;
   array cookie_value  { &cookie_count_max } $1000;

   goto SKIP;
      * Supply declaration to avoid type conflict in case no parms or cookies;
      parameter_name   = cgiparm_name{1};
      parameter_source = cgiparm_source{1};
      parameter_value  = cgiparm_value{1};
   SKIP:


   *------------------------------------------------------------------------;
   * Select_Table_by_Date reads all generations of WEBLOG_DETAIL              ;
   *------------------------------------------------------------------------;
   set %Select_Tables_by_Date( table              = weblog_detail
                              ,begin_date         = 1984-03-20
                              ,end_date           = 2084-03-20
                              ,in_variable_prefix = _in_detail_
                              ,where              = page_count gt 0 ) ;







   *------------------------------------------------------------------------;
   * Read CGI parms (names, values, and source) into arrays                  ;
   *------------------------------------------------------------------------;

   _iorc_ = 0;
   cgiparms_read = 0;
   do while(_iorc_ eq 0);

      * Get_Parms sets PARAMETER_NAME, PARAMETER_VALUE, PARAMETER_SOURCE, and _IORC_;
      %Get_Parms( cgiparms, in_variable_prefix=_in_detail_ );

      * If _IORC_ is zero, parms were found;
      if _iorc_ eq 0 then do;
         cgiparms_read = cgiparms_read + 1;

         * Do not overflow array;
         if cgiparms_read gt hbound(cgiparm_name) then do;
            put "WARNING:(USER_ASSIGNMENTS_FOR_DATA_MINING) Unable to hold "
                parameter_source "parm " cgiparms_read
                parameter_name +(-1) "=" parameter_value
                " for Record_ID " record_id +(-1)
                ". Increase CGIPARM_COUNT_MAX from &cgiparm_count_max..";
         end;

         else do;
            * Save the values we just read;
            cgiparm_name{cgiparms_read}   = parameter_name;
            cgiparm_value{cgiparms_read}  = parameter_value;
            cgiparm_source{cgiparms_read} = parameter_source;
         end;
      end;

   end;
   _error_ = 0;





   *-------------------------------------------------------------------------;
   * Read Referrer parms (names, values, and source) into arrays              ;
   *-------------------------------------------------------------------------;

   _iorc_ = 0;
   referrerparms_read = 0;
   do while(_iorc_ eq 0);

      * Get_Parms sets PARAMETER_NAME, PARAMETER_VALUE, PARAMETER_SOURCE, and _IORC_;
      %Get_Parms( referrerparms, in_variable_prefix=_in_detail_ );

      * If _IORC_ is zero, parms were found;
      if _iorc_ eq 0 then do;
         referrerparms_read = referrerparms_read + 1;

         * Do not overflow array;
         if referrerparms_read gt hbound(referrerparm_name) then do;
            put "WARNING:(USER_ASSIGNMENTS_FOR_DATA_MINING) Unable to hold "
                parameter_source "parm " referrerparms_read
                parameter_name +(-1) "=" parameter_value
                " for Record_ID " record_id +(-1)
                ". Increase REFERRERPARM_COUNT_MAX from &referrerparm_count_max..";
         end;

         else do;
            * Save the values we just read;
            referrerparm_name{referrerparms_read}   = parameter_name;
            referrerparm_value{referrerparms_read}  = parameter_value;
            referrerparm_source{referrerparms_read} = parameter_source;
         end;
      end;
   end;
   _error_ = 0;




   *------------------------------------------------------------------------;
   * Read cookies (names and values) into arrays                            ;
   *------------------------------------------------------------------------;

   _iorc_ = 0;
   cookies_read = 0;
   do while(_iorc_ eq 0);

      * Get_Parms sets PARAMETER_NAME, PARAMETER_VALUE, PARAMETER_SOURCE, and _IORC_;
      %Get_Parms( cookies, in_variable_prefix=_in_detail_ );

      * If _IORC_ is zero, parms were found;
      if _iorc_ eq 0 then do;
         cookies_read = cookies_read + 1;

         * Do not overflow array;
         if cookies_read gt hbound(cookie_name) then do;
            put "WARNING:(USER_ASSIGNMENTS_FOR_DATA_MINING) Unable to hold "
                parameter_source "parm " cookies_read
                parameter_name +(-1) "=" parameter_value
                " for Record_ID " record_id +(-1)
                ". Increase COOKIE_COUNT_MAX from &cookie_count_max..";
         end;

         else do;
            * Save the values we just read;
            cookie_name{cookies_read}   = parameter_name;
            cookie_value{cookies_read}  = parameter_value;
         end;
      end;

   end;
   _error_ = 0;


   *------------------------------------------------------------------------;
   * Show how many cookies and parms we were able to keep                   ;
   *------------------------------------------------------------------------;
   cgiparms_kept      = min(cgiparms_read,hbound(cgiparm_name));
   referrerparms_kept = min(referrerparms_read,hbound(referrerparm_name));
   cookies_kept       = min(cookies_read,hbound(cookie_name));


   run;


%Section_Footer(Section=User_Assignments_For_Data_Mining) ;
```

Checkpoint_load_term

The Checkpoint_load_term module records the time of completion of the Load step. This information is recorded in the Job_Statistics table, which can be used to produce reports.

Contents of the Detail Library

Overview: The Detail Library and Its Detail Tables

If the Generation_Count property for each detail table (Weblog_Detail, CGIParms, Cookies, Pathing, and ReferrerParms) is set to keep one or more generations, then each time the Load process is executed, it creates a new generation of the table. When the Detail library contains the number of each type of table that is specified its associated Generation_Count property, the oldest table is deleted at the next execution of the Load process.

For more information about setting the number of generations for a detail table, see "The Detail Tables Properties" on page 31.

The following table shows the contents of the Detail library after the Load step has successfully run. The Load process creates the Weblog_Detail, CGIParms, ReferrerParms, Cookies, and Pathing tables only when the Generation_Count property is greater than zero. The Unique_URLs table exists only if the metadata value for the number of Pathing tables is greater than zero.

Note: The default value in the Generation_Count property for the CGIParms, ReferrerParms, and Cookies tables is zero. Therefore, by default, these tables will not be generated unless you change the number of generations to keep to a non-zero value. For information about how to change the number of history tables to keep, see "Calculating the Number of History Tables to Keep" on page 34. △

Table 8.1 Contents of the Detail Library after Running the Load Process

View	Definition
Weblog_Detail	Combines Weblog_Detail_1, Weblog_Detail_2, etc.
CGIParms	Combines CGI Parms_1, CGI Parms_2, etc.
ReferrerParms	Combines ReferrerParms_1, ReferrerParms_2, etc.
Cookies	Combines Cookies_1, Cookies_2, etc.
Pathing	Combines Pathing_1, Pathing_2, etc.
Unique_URLs	Used with the Pathing table to translate numbers to URLs.
Web_Mine	Combines Weblog_Detail for page views.

Weblog_Detail_*n*

After each run, the Load process creates a Weblog_Detail_*n* table that contains one observation for each line of Web log input. Those items that are specified in the Filters frame of the SAS e-Data ETL Administrator are excluded. (See "The Filtering Properties" on page 45 for more information.)

Weblog_Detail_1 is the newest table, Weblog_Detail_2 is the next newest table, and so on. The Weblog_Detail_*n* tables are sorted by Session_ID and Date_Time. A similarly named view with no numeric suffix, Detail.Weblog_Detail, interleaves all of the existing tables together, maintaining their sorted order.

The following fields can be included in the Weblog_Detail table.

Browser
Browser_Version
Bytes_Received
Bytes_Sent
Client_ID
Combined_URL
Cookie_Jar
Cookie_Present
Content_Group
Country
Date
Date_Time
Day_Of_Week
Domain
Entry_Point
Exit_Point
Eyeball_Time
File_Count
File_Type
GMT_Offset
Hour
HTTP_Version
Initial_Reference_By_Session
Method
MIME_Type
Month
Org
Org_Type
Page_Count
Path
Path_Info
Platform
Protocol
Quarter
Query_String
Record_ID
Referrer
Referrer_Domain
Referrer_Path_Info
Referrer_Query_String
Referrer_URL_Level_1
Referrer_URL_Level_2
Referrer_URL_Level_3
Referrer_URL_Level_4
Referrer_URL_Level_5
Referrer_URL_Level_6
Referrer_URL_Level_7
Referrer_URL_Level_8
Requested_File
Scheme
Server
Session_Duration
Session_ID
Session_Start
Sitename
State
Status_Code
Time
Time_Taken
Total_Bytes
Unique_URL
URL_Fragment
URL_Level_1
URL_Level_2
URL_Level_3
URL_Level_4
URL_Level_5
URL_Level_6
URL_Level_7
URL_Level_8
User_Agent
Username
View_EyeballTime
Visitor_ID
WebLogFileName
Week

Some of these fields are not kept in the Weblog_Detail table, but are defined so that they can be added by changing their kept status from NO to YES in the Control.Wbfields table. The Control.Wbfields table is accessed from the Processing frame of the SAS e-Data ETL Administrator. To learn how to view and edit the Control.Wbfields table, see “Adding or Deleting Fields in Your Web Server Log File” on page 38.

CGIParms_*n*

The CGIParms_*n* table contains the name-value pairs that have been parsed from the query string in the Web log record. CGIParms_1 is the newest table, CGIParms_2 is the next newest table, and so on. The metadata for the table determines the number of tables that are created. The Detail.CGIParms view interleaves all the existing tables together, maintaining their sorted order.

You can use the Record_ID field to join an observation in the CGIParms_*n* table with an observation in the corresponding Weblog_Detail_*n* table. Combining these tables will give you a complete picture of the Web log record in which the parameter appeared. An example of the code that will join the CGIParms_*n* table with the Weblog_Detail_*n* table appears in the User_assignments_for_data_mining CAM.

The following fields are included in the CGIParms table:

Record_ID

Parameter

Value Indexed by Record_ID

ReferrerParms_*n*

The ReferrerParms_*n* table contains the name-value pairs that are parsed from the Referrer field in the Web log record. ReferrerParms_1 is the newest table, ReferrerParms_2 is the next newest table, and so on. The metadata for the table determines the number of tables that are created. The Detail.ReferrerParms view interleaves all the existing tables together, maintaining their sorted order.

You can use the Record_ID field to join an observation in the ReferrerParms_*n* table with an observation in the corresponding Weblog_Detail_*n* table. Combining these tables will give you a complete picture of the Web log record in which the parameter appeared. An example of the code that will join the ReferrerParms_*n* table with the Weblog_Detail_*n* table appears in the User_assignments_for_data_mining CAM.

The following fields are included in the ReferrerParms table:

Record_ID

Parameter

Value Indexed by Record_ID

Cookies_*n*

The Load process creates a Cookies_*n* table that contains Cookie name-value pairs and a record identifier. Cookies_1 is the newest table, Cookies_2 is the next newest table, and so on. The metadata for the table determines the number of tables that are created. The Detail.Cookies view interleaves all the existing tables together, maintaining their sorted order.

You can use the Record_ID field to join an observation in the Cookies_*n* table with an observation in the corresponding Weblog_Detail_*n* table. Combining these tables gives you a complete picture of the Web log record in which the cookies appeared. An example of the code that joins the Cookies_*n* table with the Weblog_Detail_*n* table appears in the User_assignments_for_data_mining CAM.

The following fields are included in the Cookies table:

Record_ID
Parameter
Value Indexed by Record_ID

Pathing_*n*

The Pathing_*n* table contains a session's path, duration, and datetime stamp. Pathing_1 is the newest table, Pathing_2 is the next newest table, and so on. These tables are sorted by Session_ID and Date_Time. The metadata for this table determines the number of tables that are created. The Detail.Pathing view interleaves all the existing tables together, maintaining their sorted order.

The value of the Path field is determined by the code in the Determine_unique_url module during the Extract process. (For more information about this module, see "Determine_unique_url" on page 88.) Each observation in the Path field contains a series of numbers that are separated by colons. These numbers can be mapped into their corresponding URL by using the ID2URL format or the $ID2URL. format. The ID2URL format is defined for the number of Pathing tables that are returned. ID2URL is in Control.Formats, which is part of the FMTSEARCH path that the %EDATAETL macro establishes. You can use a similar format, $URL2ID., to determine the ID of any URL in the Weblog_Detail table. (The variable name in that table is Requested_File.)

The following fields are included in the Pathing table:

Date_Time
Session_ID
Path
Session_Duration

$ID2URL. Format Example

In the Pathing table, the Path field consists of a string of numbers that are separated by colons. An example of an entry in the Path field is 42:95:3:101. Each of these numbers represents a page that is requested by the visitor to your Web site. You can use the $ID2URL. format to map the number (or requested page) to the full pathname of the requested page.

The following code removes the right-most number from your Path field entry and converts it to the full pathname of the requested page:

```
node_label=put(trim(left(scan(node,-1,’’:’’))),$id2url.);
```

By using this code in the entire Path string, you can resolve all these numbers to see the path that each visitor is traversing on your Web site.

Unique_URL

The Unique_URL tables are maintained by the Load step. They contain all the URLs that are referenced in the retained Pathing tables. The value of each unique URL is determined by the code in the Determine_unique_url module of the Extract process. For more information about this module, see “Determine_unique_url” on page 88.

The Unique_URL tables are used to create the formats that map the numeric ID of the URL (in the Pathing table’s Path field) to an actual URL name, and vice versa. These formats, $ID2URL. and $URL2ID., are contained in Control.Formats.

The following fields are included in the Unique_URL tables:

Requested_File

Generation

URL_Guid

CHAPTER 9

Using the SAS Web Analytics Administrator to Manage Your Web Mart(s)

Navigating in the SAS Web Analytics Administrator

The SAS Web Analytics Administrator is a Web-based interface that is designed to enable you to do the following:

- □ customize how the data is summarized into the summary data sets during ETL processing of your Web mart data
- □ customize the operation of the SAS Web Analytics Report Viewer in order to control how data appears in reports

The main menu of the Administrator consists of graphic links to the Web pages described in the following table:

Table 9.1 Main Menu Bar in the SAS Web Analytics Administrator

Page	**Action**
Web Marts	Register, edit or copy a Web mart. View the basic SAS server connections and the root of each of your Web marts.
Data	Edit the summarization and other processing parameters for Web site analysis (Web Analysis Configuration). Customize the appearance of reports (User Interface Settings).
Traffic	Add, modify, or delete libref assignments. Add, modify, or delete the data sets that are used to populate drop-down menus in reports. Add, modify, or delete the features of stored processes. Add, modify, or delete the features of report groups and individual reports.
Summaries	Customize the rules that control the summarization of the metrics into the summary data sets.
Scorecards	Create, edit, or delete a scorecard.
Dashboards	Create, edit, or delete a dashboard.
Segmentations	Create, edit, or delete a segmentation report.

The Interface of the SAS Web Analytics Administrator

The following display shows the components of the SAS Web Analytics Administrator. These components operate as follows:

❶ The main menu bar displays the links to the pages of the SAS Web Analytics Administrator.

❷ The title of the current selected page is yellow.

❸ A specific Web mart must be selected in order to view tree view items on any page of the SAS Web Analytics Administrator.

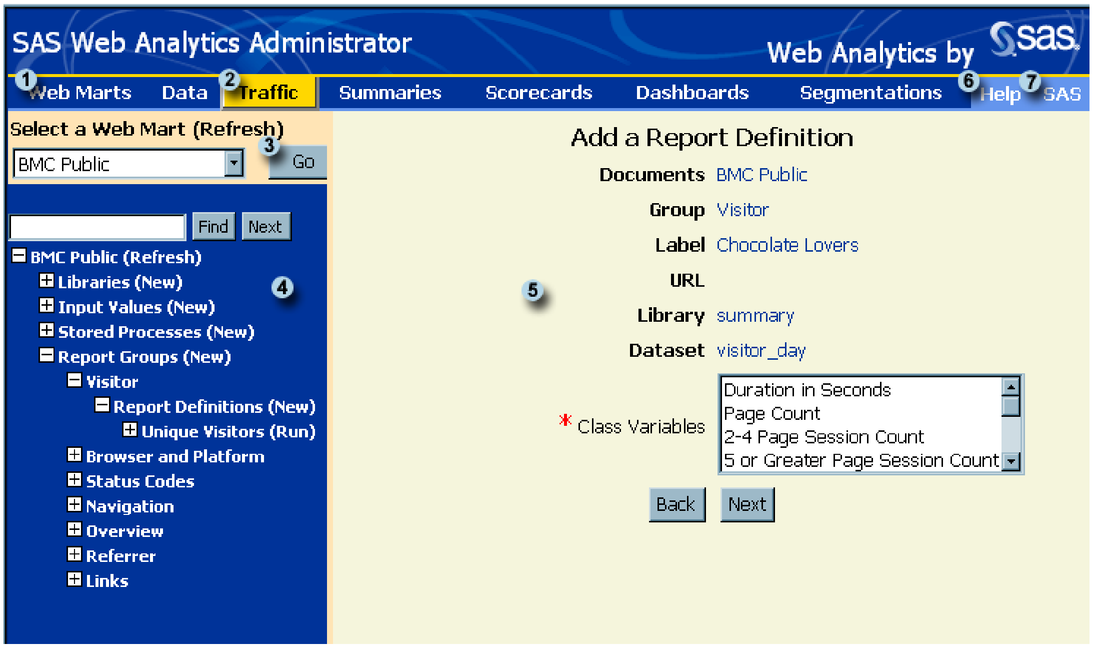
❹ The blue navigation area on the left displays a tree view of items.

❺ The right side of the page displays a form, a table, or other informational display that results from selecting an item in the tree view.

❻ **`Help`** is a link to the *SAS Web Analytics 5.2: Administrator's Guide*, a PDF document that you can view online, use for keyword searches, or print by using Adobe Reader 6.0 or a later version.

❼ **`SAS`** is a link to the SAS Home Page.

Display 9.1 Components of the SAS Web Analytics Administrator Interface



Add a Report Definition

Documents BMC Public

Group Visitor

Label Chocolate Lovers

URL

Library summary

Dataset visitor_day

* Class Variables Duration in Seconds
Page Count
2-4 Page Session Count
5 or Greater Page Session Count

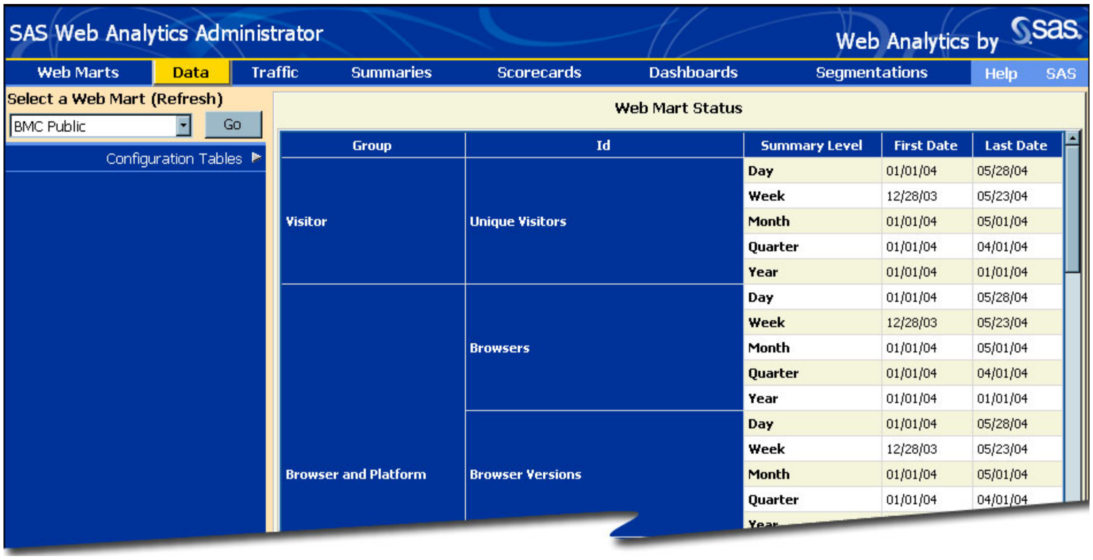
If you do not see a navigation tree, then select a Web mart from the drop-down list (item ❸ in the previous display).

Understanding Date Ranges in the Web Mart Status Table

As a convenient reference, the Web Mart Status table is initially displayed on the right of all of the pages of the SAS Web Analytics Administrator (except the Web Marts page and the Traffic pages). The Web Mart Status table appears after a Web mart is initially selected and remains until you select an item or action in the navigation tree.

In the following display, the Web Mart Status table is shown in the Data page, for example.

Display 9.2 Web Mart Status Table on the Data Page of the SAS Web Analytics Administrator



Web Mart Status

Group	Id	Summary Level	First Date	Last Date
Visitor	Unique Visitors	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
Browser and Platform	Browsers	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
	Browser Versions	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year		

The Web Mart Status table shows the date ranges that have been processed for each of the corresponding summary levels (day, week, month, quarter, and year). The first date and last date for the different summary levels are not identical, because each summary level represents a unit of time that cannot be subdivided. Each date, regardless of whether it is listed in the First Date or the Last Date column, represents the beginning date of its corresponding interval. For example, the month interval for May 2004 is labeled as 05/01/04.

Examine the first date and last date of the summary levels for the Unique Visitors report in the example Web Mart Status table. The first date of the `Day` summary level is 01/01/2004, and the last date is 05/28/2004. These dates indicate that Web mart data has been processed for the 24-hour intervals (days) starting on January 1, 2004 and ending on (and including) May 28, 2004.

The first date of the `Week` summary level for Unique Visitors is 12/28/2003, assuming that the first day of a week begins on Sunday. December 28th is the start day of the specific week that includes the first day that data was actually processed (01/01/2004). Data was processed through Friday, May 28th, 2004, but the last date of the Week summary level appears as 05/23/04 because the first day (Sunday) of that week interval occurs on 05/23/04.

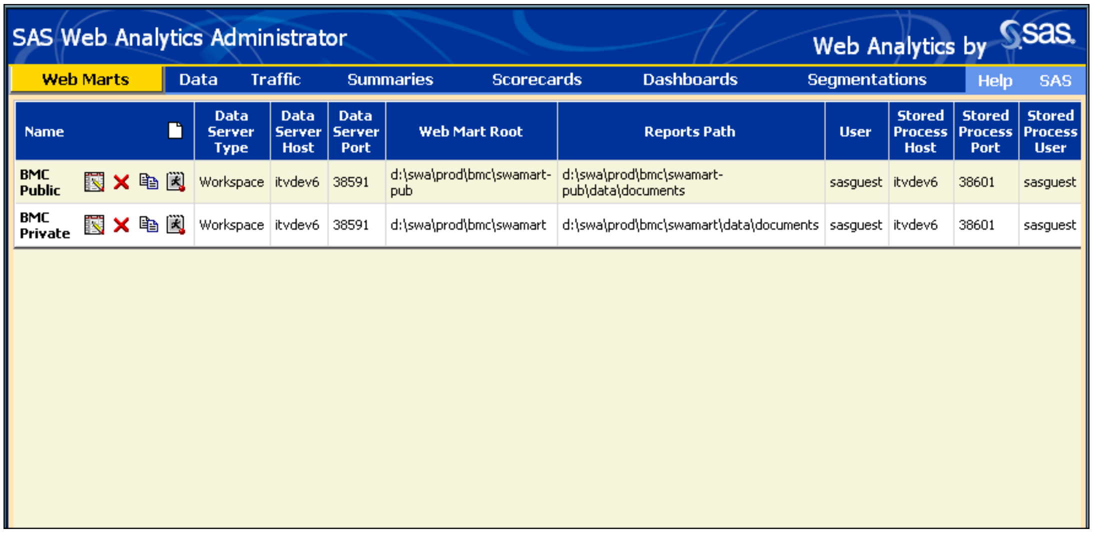
Note: The summary of the data for the last week in May 2004 is a partial summary because the Web log(s) for Saturday, May 29, 2004 have not been processed. △

The first date of the `Month` summary level is 01/01/04. The last date for the `Month` summary level is 05/01/04 because May 1, 2004 is the first day of the last month for which Web logs were processed.

The Web Mart Status table specifies the dates for the `Quarter` and the `Year` summary levels in a similar way.

The Web Marts Page

Display 9.3 The Web Marts Page of the SAS Web Analytics Administrator



Name	Data Server Type	Data Server Host	Data Server Port	Web Mart Root	Reports Path	User	Stored Process Host	Stored Process Port	Stored Process User
BMC Public	Workspace	itvdev6	38591	d:\swa\prod\bmc\swamart-pub	d:\swa\prod\bmc\swamart-pub\data\documents	sasguest	itvdev6	38601	sasguest
BMC Private	Workspace	itvdev6	38591	d:\swa\prod\bmc\swamart	d:\swa\prod\bmc\swamart\data\documents	sasguest	itvdev6	38601	sasguest

The Web Marts page of the SAS Web Analytics Administrator (see the previous display) displays system information about all of the Web marts that have been registered. You can also do the following tasks (see Display 9.4 on page 116):

❶ register a new Web mart (for details, see "Registering a New Web Mart" on page 24)

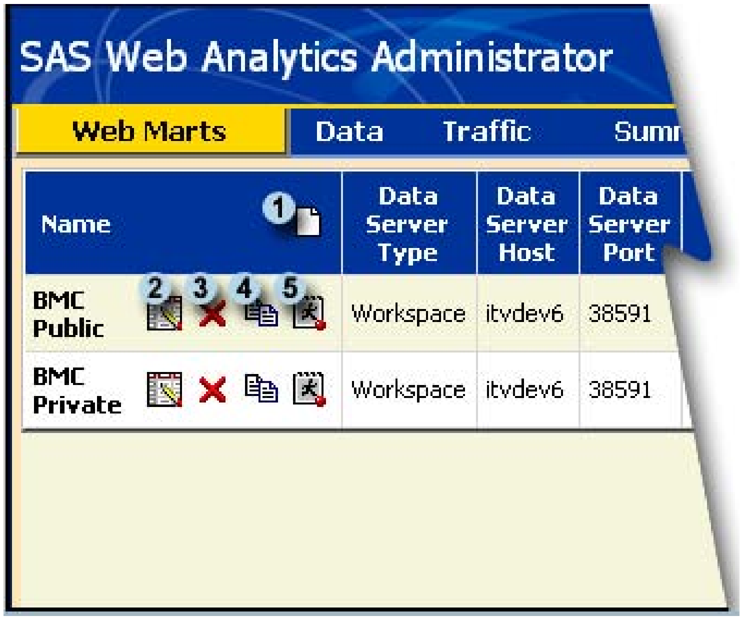
❷ edit the Web mart (that is in the same row as the icon)

❸ delete the Web mart

❹ copy the Web Mart

❺ generate the status of the Web mart

Display 9.4 Task Icons in the Web Marts Page



Name	Data Server Type	Data Server Host	Data Server Port
BMC Public	Workspace	itvdev6	38591
BMC Private	Workspace	itvdev6	38591

Editing a Web Mart in the SAS Web Analytics Administrator

The Edit a Web Mart form (Display 9.5 on page 117) enables you to edit the environmental characteristics of a Web mart.

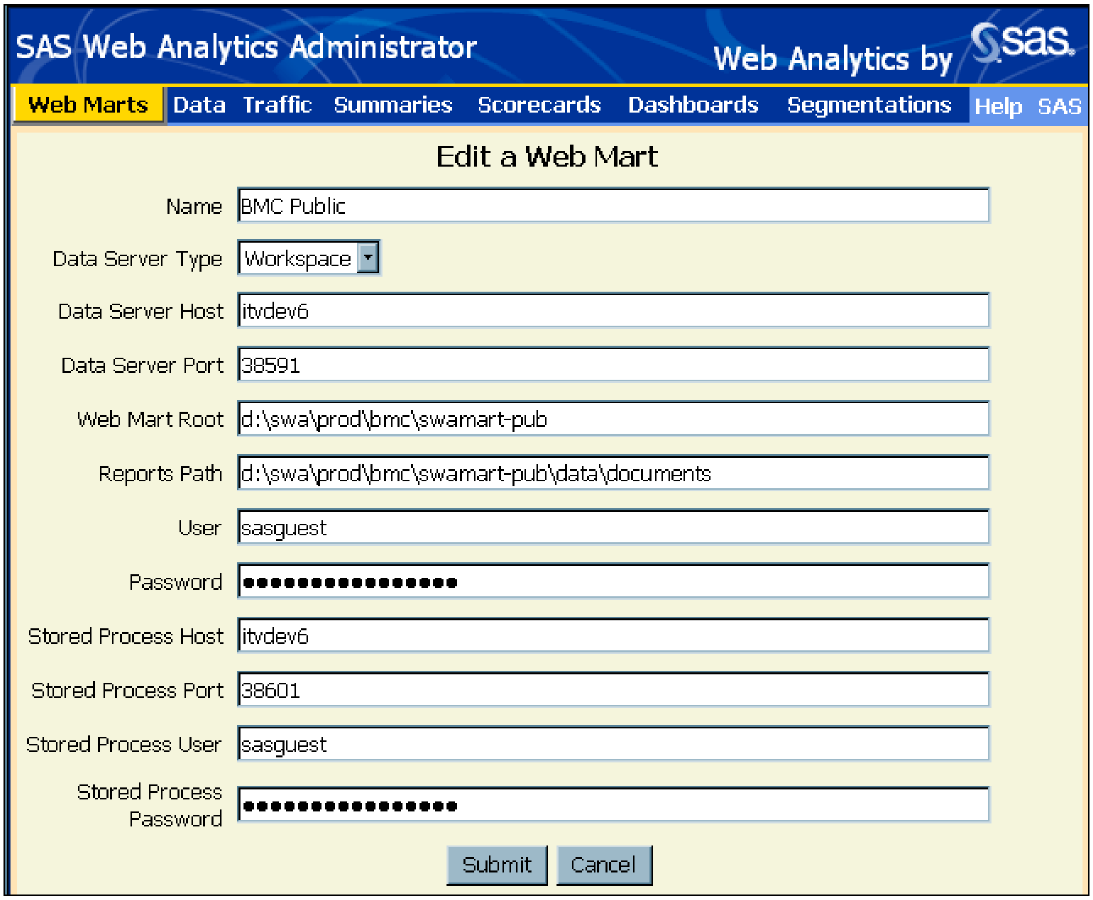
Note: To edit all of the other properties of a Web mart, use the SAS e-Data ETL Administrator (see “Overview: The Properties of a SAS Web Analytics Web Mart” on page 28). △

To edit the environmental characteristics of a Web mart, perform the following steps:

1 In the Web Marts page, click ✎ next to the Web mart you want to edit. The Edit a Web Mart form appears.

2 Enter changes to any fields as necessary. For assistance, consult with the authorities in your organization who are familiar with the configuration of the SAS solution on your server(s). When you are finished, click **`Submit`**.

Display 9.5 The Edit a Web Mart Form



Edit a Web Mart

Name	BMC Public
Data Server Type	Workspace
Data Server Host	itvdev6
Data Server Port	38591
Web Mart Root	d:\swa\prod\bmc\swamart-pub
Reports Path	d:\swa\prod\bmc\swamart-pub\data\documents
User	sasguest
Password	●●●●●●●●●●●●●●●●
Stored Process Host	itvdev6
Stored Process Port	38601
Stored Process User	sasguest
Stored Process Password	●●●●●●●●●●●●●●●●

Deleting a Web Mart in the SAS Web Analytics Administrator

To delete a Web Mart, select the Web Marts page of the SAS Web Analytics Administrator. Click ✗ in the first column next to the name of the Web mart to delete it.

Copying a Web Mart in the SAS Web Analytics Administrator

You might want to copy an existing Web mart for one of the following reasons:

- □ As an administrator, you want to enable different communities of end users to quickly access the specific SAS Web Analytics reports that relate to their work.
- □ You have a highly customized Web mart and you want to use these settings in a new Web mart.

In the first scenario, you might copy a Web mart in order to provide a custom subset of reports. You would provide this subset of reports as a separate Web mart in the SAS Web Analytics Report Viewer. In this way, the data for the customized subset of reports is available without duplicating the ETL processing of the original Web mart data.

The copy (⎘) action in the Web Marts page of the SAS Web Analytics Administrator duplicates the values for the fields of an existing Web mart, but does not create new directories.

CAUTION:

> **Before you copy a Web mart in the SAS Web Analytics Administrator, all of the destination directory structures must previously exist.** Either manually copy and paste the Web mart directory and all its subdirectories to a new location in the Windows environment, or run the %WAETL macro with the INITIALIZE argument for the PROGRAM parameter. △

To make a copy of a Web mart (assuming that a directory structure already exists for the new Web mart copy), perform the following steps:

1. In the Web Marts page, click ⧉ next to the Web mart that you want to copy. The Copy a Web Mart form appears. Except for the title, the Copy a Web Mart form is identical to the Edit a Web Mart form (Display 9.5 on page 117).
2. In the Name field, the words “Copy of” are automatically added to the old Web mart name. Modify the name as you desire.
3. Modify the other values as necessary. For assistance, consult with the authorities in your organization who are familiar with the configuration of the SAS solution on your server(s). When you are finished, click **`Submit`**.

Updating a Web Mart

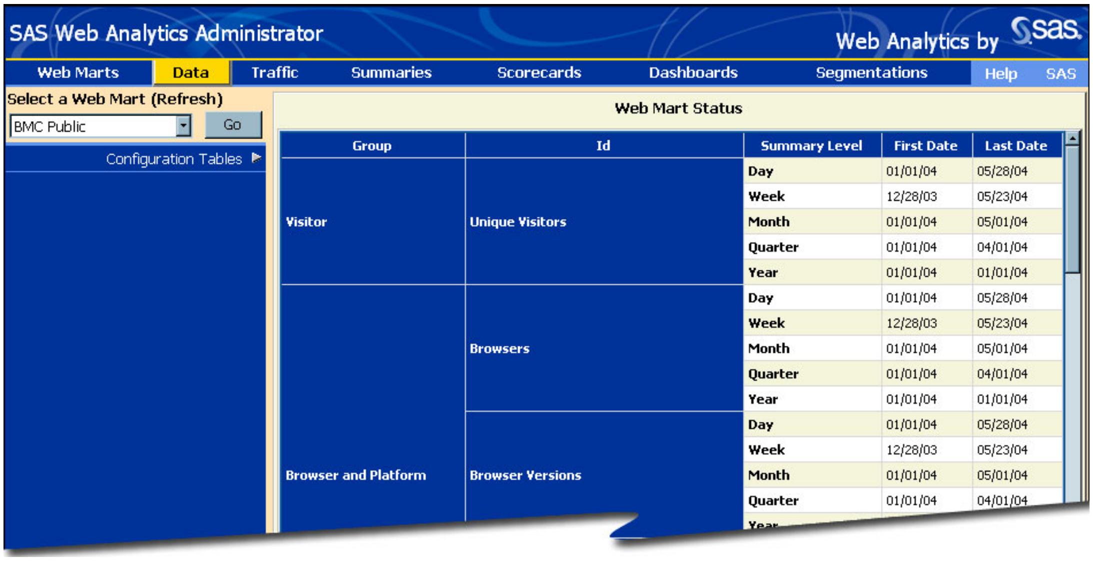
You might need to update a Web mart to create a provisional Web Mart Status table or to re-establish a Web Mart Status table that cannot be read.

When you select a Web mart, the interface (of either the SAS Web Analytics Administrator or Report Viewer) must access the Web Mart Status table. This table is routinely created during ETL processing by the Daily.sas program. However, if you want to select a Web mart in the SAS Web Analytics interface before you run the ETL processes, then the 🏃 icon enables you to create a provisional version of the Web Mart Status table for the interface to access.

To create a provisional Web Mart Status table or to re-establish a Web Mart Status table, click 🏃 next to the Web mart that you want to update. After a few moments, a message verifies that the Web Mart Status table was successfully generated. Click **`Continue`**. You are returned to the Web Marts page.

The Data Page

Display 9.6 The Data Page of the SAS Web Analytics Administrator



Select a Web Mart (Refresh)

BMC Public

Web Mart Status

Group	Id	Summary Level	First Date	Last Date
Visitor	Unique Visitors	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
Browser and Platform	Browsers	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
	Browser Versions	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year		

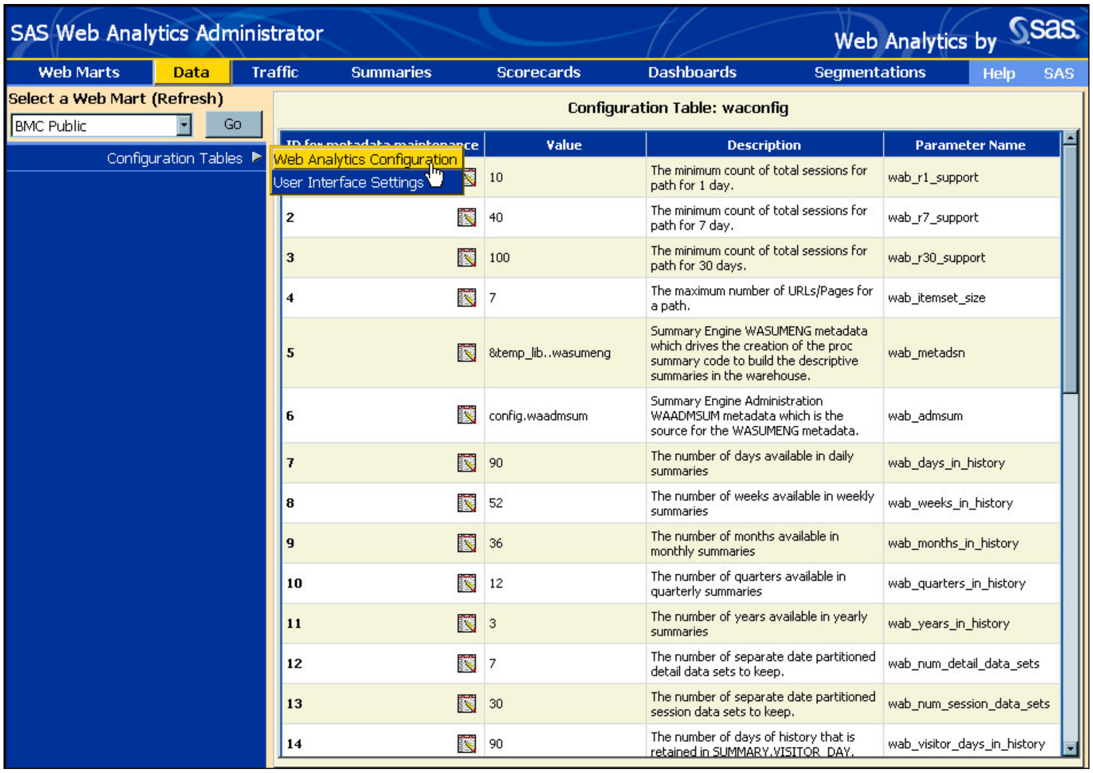
The Data page enables you to customize the parameters in the following areas:

Web Analytics Configuration
: configures the settings for summarizing the data for a selected Web mart.

User Interface Settings
: sets the graphical components for decision support reports and a parameter for the funnel report for a selected Web mart.

Customizing the Parameters that Control the Configurable Settings (Web Analytics Configuration)

Display 9.7 The Table of Parameters that Control the Configurable Settings (Waconfig Data Set)



Configuration Table: waconfig

ID for metadata maintenance	Value	Description	Parameter Name
	10	The minimum count of total sessions for path for 1 day.	wab_r1_support
2	40	The minimum count of total sessions for path for 7 day.	wab_r7_support
3	100	The minimum count of total sessions for path for 30 days.	wab_r30_support
4	7	The maximum number of URLs/Pages for a path.	wab_itemset_size
5	&temp_lib..wasumeng	Summary Engine WASUMENG metadata which drives the creation of the proc summary code to build the descriptive summaries in the warehouse.	wab_metadsn
6	config.waadmsum	Summary Engine Administration WAADMSUM metadata which is the source for the WASUMENG metadata.	wab_admsum
7	90	The number of days available in daily summaries	wab_days_in_history
8	52	The number of weeks available in weekly summaries	wab_weeks_in_history
9	36	The number of months available in monthly summaries	wab_months_in_history
10	12	The number of quarters available in quarterly summaries	wab_quarters_in_history
11	3	The number of years available in yearly summaries	wab_years_in_history
12	7	The number of separate date partitioned detail data sets to keep.	wab_num_detail_data_sets
13	30	The number of separate date partitioned session data sets to keep.	wab_num_session_data_sets
14	90	The number of days of history that is retained in SUMMARY.VISITOR_DAY.	wab_visitor_days_in_history

The parameters that control the configurable settings are located in the Waconfig data set. Explore the fields in the Waconfig data set to become familiar with the ways that you can modify these parameters. See "The Waconfig Data Set: Configuration Settings" on page 299 for how these parameters affect Web analysis and how to set these parameters.

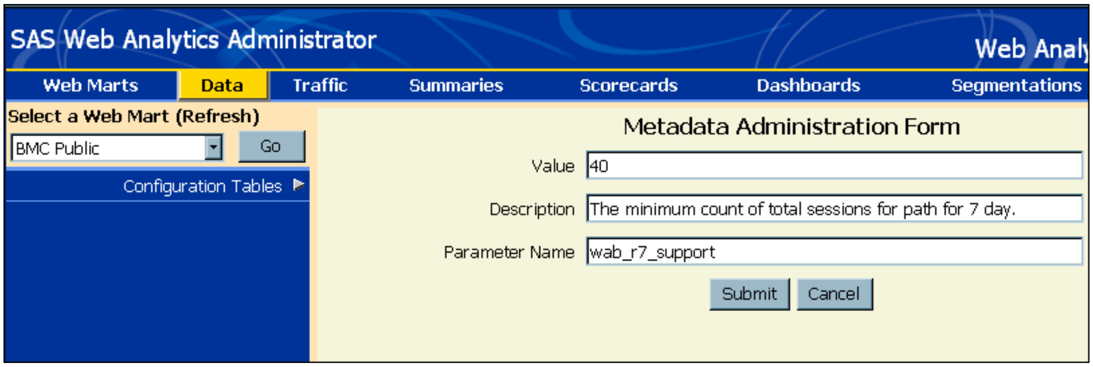
To access the Waconfig data set, perform the following steps:

1 Select **`Data`** in the main menu bar in the SAS Web Analytics Administrator. If a Web mart is not already selected, then select a Web mart and click **`Go`**. The **`Configuration Tables`** option appears in the navigation area.

2 Select **Configuration Tables ▶ Web Analytics Configuration**. The Waconfig table appears on the right.

To edit a parameter in the Waconfig table, perform the following steps:

1 Click ✎ in the row of the parameter that you want to edit. A Metadata Administration form appears on the right, similar to the following display:



BMC Public

Metadata Administration Form

Value: 40

Description: The minimum count of total sessions for path for 7 day.

Parameter Name: wab_r7_support

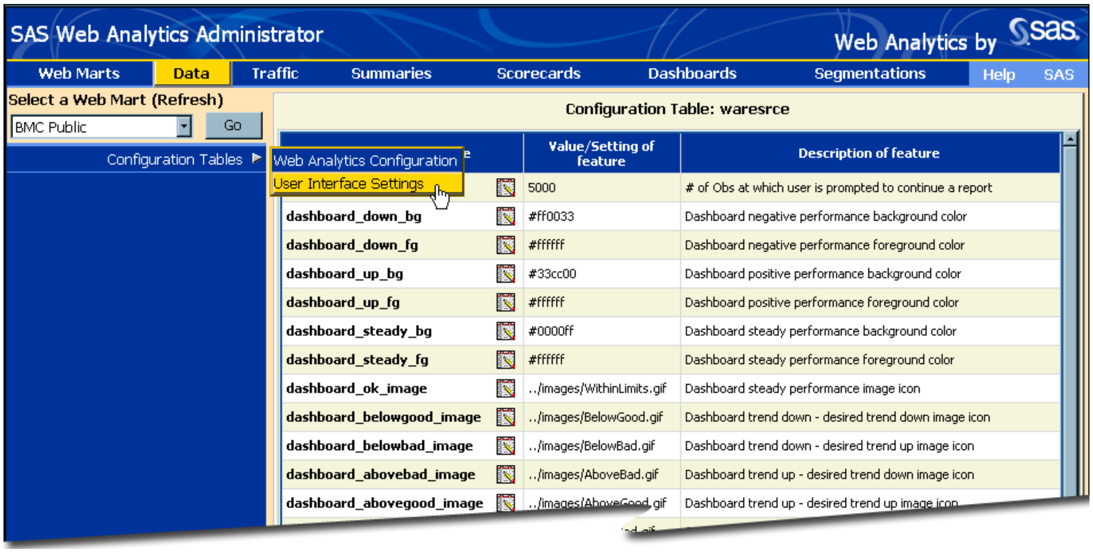
2 Modify the values for one or more fields as you wish. When you are finished, click **Submit**.

3 Click **Continue** below the confirming message that appears on the right. The table reappears.

4 Repeat these steps to edit additional parameters.

Customizing the Graphical Components of Reports (User Interface Settings)

Display 9.8 Table for Customizing User Interface Settings



BMC Public

Configuration Table: waresrce

e	Value/Setting of feature	Description of feature
	5000	# of Obs at which user is prompted to continue a report
dashboard_down_bg	#ff0033	Dashboard negative performance background color
dashboard_down_fg	#ffffff	Dashboard negative performance foreground color
dashboard_up_bg	#33cc00	Dashboard positive performance background color
dashboard_up_fg	#ffffff	Dashboard positive performance foreground color
dashboard_steady_bg	#0000ff	Dashboard steady performance background color
dashboard_steady_fg	#ffffff	Dashboard steady performance foreground color
dashboard_ok_image	../images/WithinLimits.gif	Dashboard steady performance image icon
dashboard_belowgood_image	../images/BelowGood.gif	Dashboard trend down - desired trend down image icon
dashboard_belowbad_image	../images/BelowBad.gif	Dashboard trend down - desired trend up image icon
dashboard_abovebad_image	../images/AboveBad.gif	Dashboard trend up - desired trend down image icon
dashboard_abovegood_image	../images/AboveGood.gif	Dashboard trend up - desired trend up image icon

The parameters that control the appearance of graphical components in several types of reports are located in the Waresrce configuration table (see the previous display). Explore the fields in this configuration table to become familiar with the ways that you can modify how information is displayed in some types of SAS Web Analytics reports.

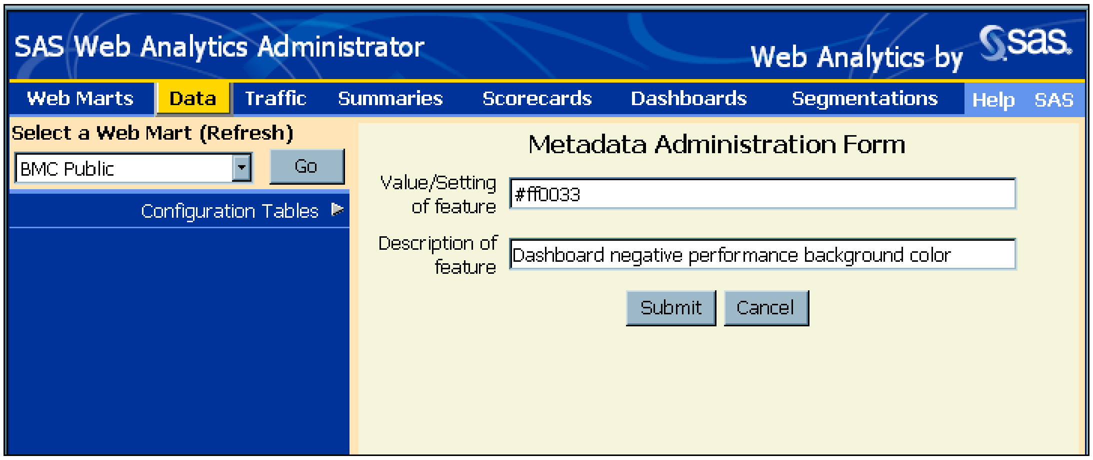
To access the Waresrce configuration table, perform the following steps:

1 Select **Data** in the main menu bar in the SAS Web Analytics Administrator. If a Web mart is not already selected, then select a Web mart and click **Go**. The **Configuration Tables** option appears in the navigation area.

2 Select **Configuration Tables ▶ User Interface Settings**. The Waresrce table appears on the right.

To edit a parameter in the table, perform the following steps:

1 Click 📝 in the row of the parameter that you want to edit. A Metadata Administration form appears on the right, similar to the following display:



Metadata Administration Form

BMC Public

Value/Setting of feature	#ff0033
Description of feature	Dashboard negative performance background color

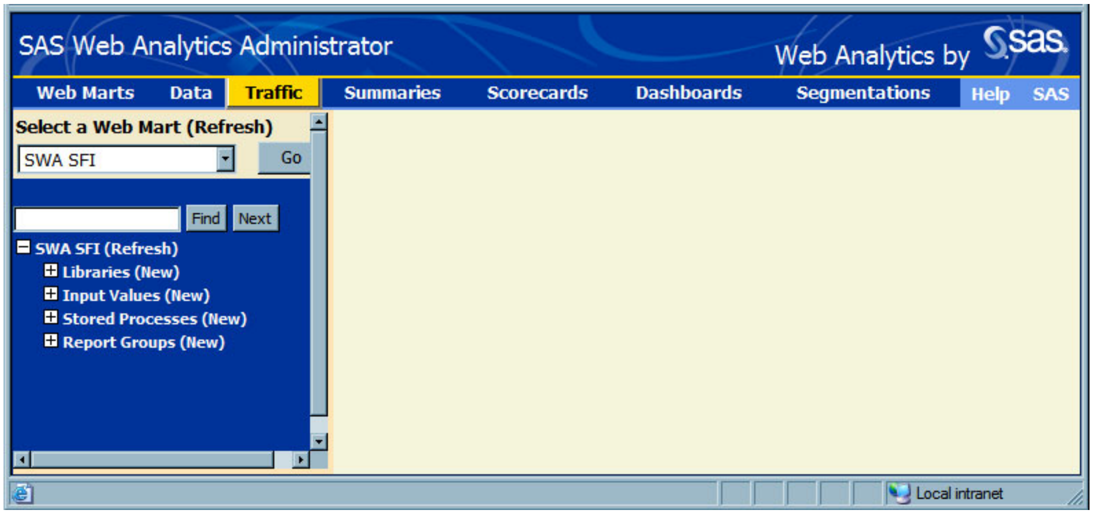
2 Modify the values for one or more fields as you wish. When you are finished, click **Submit**.

3 Click **Continue** below the confirming message that appears on the right. The table reappears.

4 Repeat these steps to edit additional parameters.

The Traffic Page

Display 9.9 The Traffic Page of the SAS Web Analytics Administrator



The Traffic page of the SAS Web Analytics Administrator enables you to do the following tasks:

1 assign, modify, copy and delete a library for SAS Web Analytics entries

2 add an input value that points to a data set of items to display in a drop-down menu for a specified report

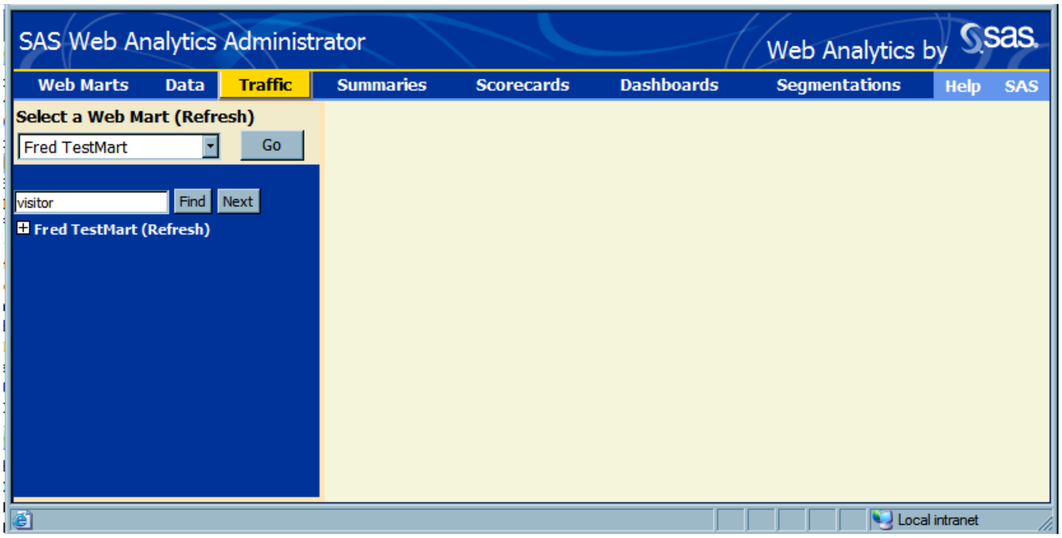
3 create, modify, or run a stored process for a funnel report

4 create, customize, or delete report groups for viewing by end users in the SAS Web Analytics Report Viewer

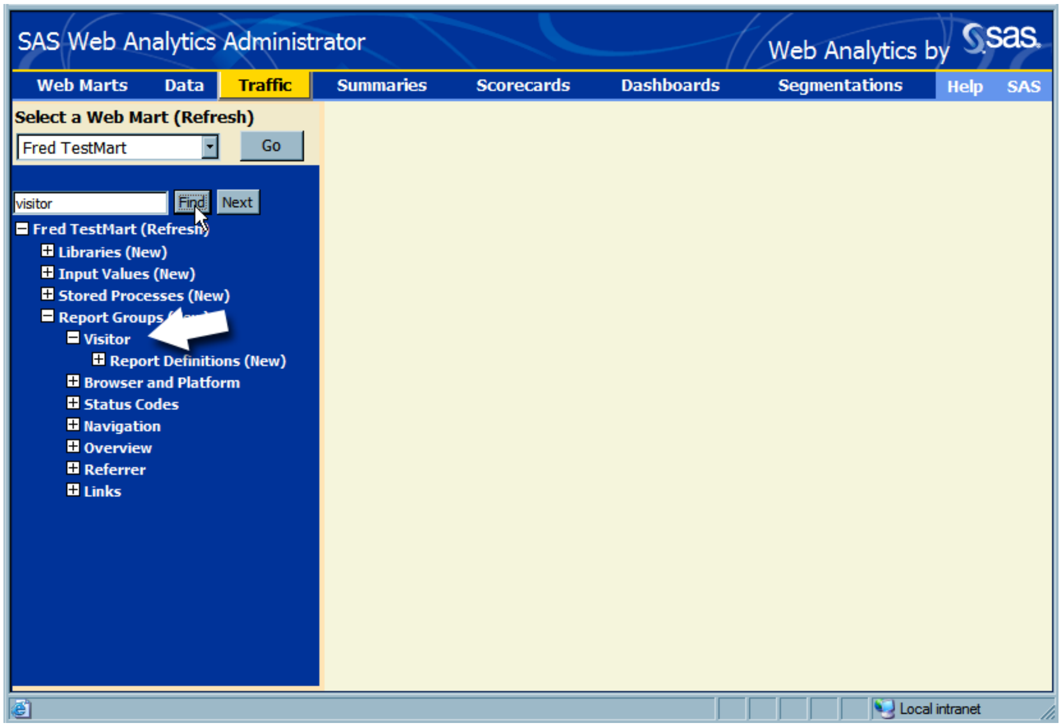
5 create, customize, or delete reports (excluding dashboards, scorecards, and segmentation reports) for viewing by end users in the SAS Web Analytics Report Viewer

Using Find and Next to Locate a Report in the Navigation Tree

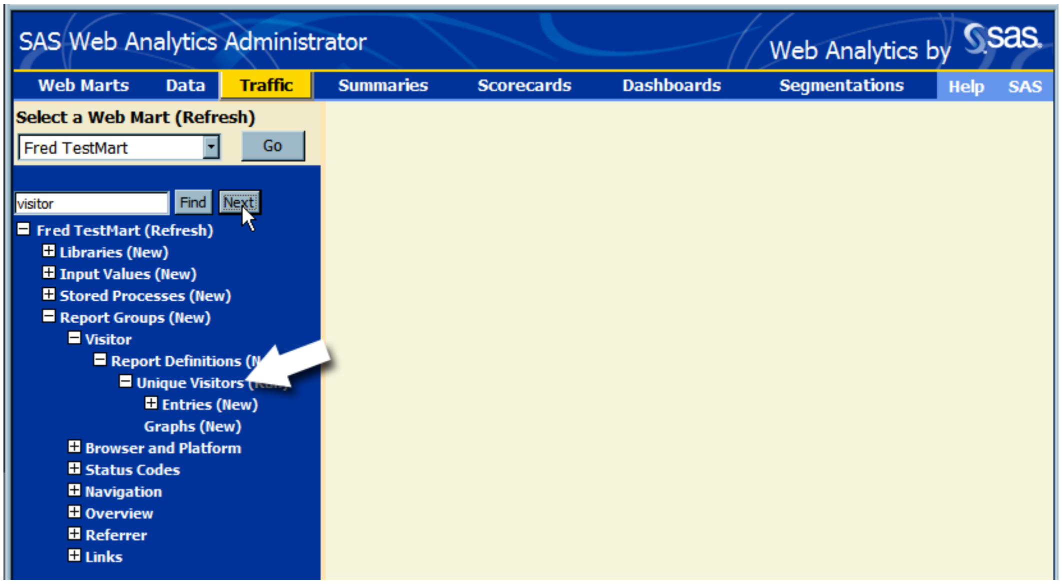
You can type a keyword in the text box next to the `Find` button in the Traffic page to expand the navigation tree to the first occurrence of the keyword in the navigation tree. For example, if you want to see where the visitor reports are located, then you can type *visitor* (see the following display):



Then click **Find**. The navigation tree expands to the first occurrence of "visitor" (see the following display):



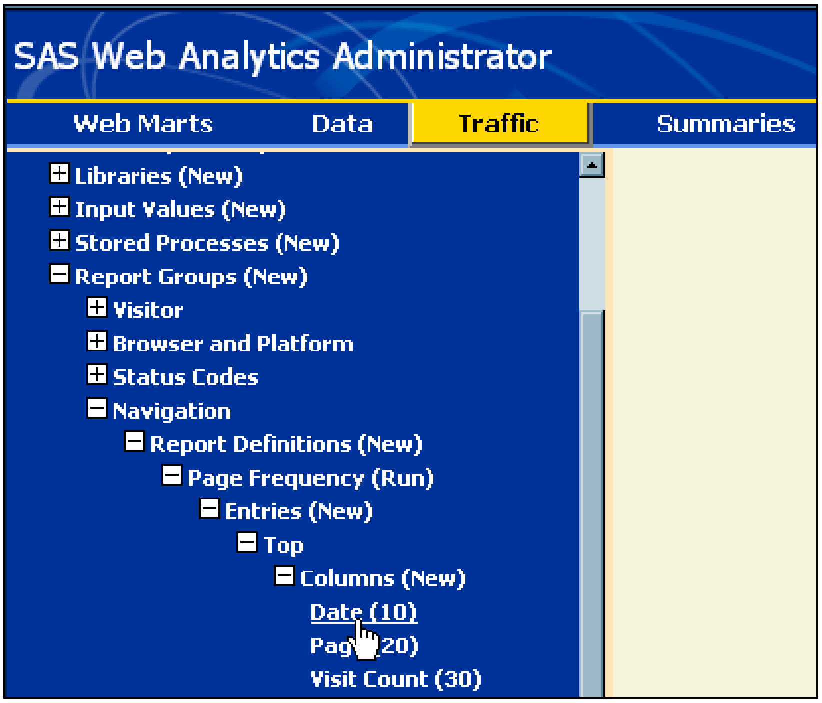
To find the next occurrence of visitor, click **Next**. The navigation tree expands further to reveal the next occurrence of visitor, which is the visitor referrer node (see the following display):



Creating an Item in the Traffic Page

To create an item (library, input value, stored process, report, or report group) in the Traffic page, perform the following general steps:

1 Expand the navigation tree by clicking + next to an item to view a component of the item. Depending on the component, you might need to expand several levels in order to expose the component you want to create (for example, see the following display):



2 Select **New** (in parentheses) next to the component that you wish to create.

3 Edit the fields in the form that appears on the right.

4 Click **Submit**, and click **Continue**.

Modifying, Copying or Deleting an Item in the Traffic Page

To modify, delete, or copy an item in the Traffic page, perform the following general steps:

1 Expand the navigation tree to view the item or the component of the item. Depending on the item, you might need to click several items in order to expose the desired component.

2 Select the item or component that you want to modify, delete, or copy by clicking it.

3 In the table that appears on the right, select **Modify**, **Delete** or **Copy**.

4 Provide the information that is required by the action. When you are finished, click **Submit**, and click **Continue**.

5 To verify any modifications that you made, select **Refresh** next to **Select a Web Mart**.

Libraries

By default, the following standard libraries are defined when a Web mart is registered:

Dated contains session data sets.

Summary contains summary data sets.

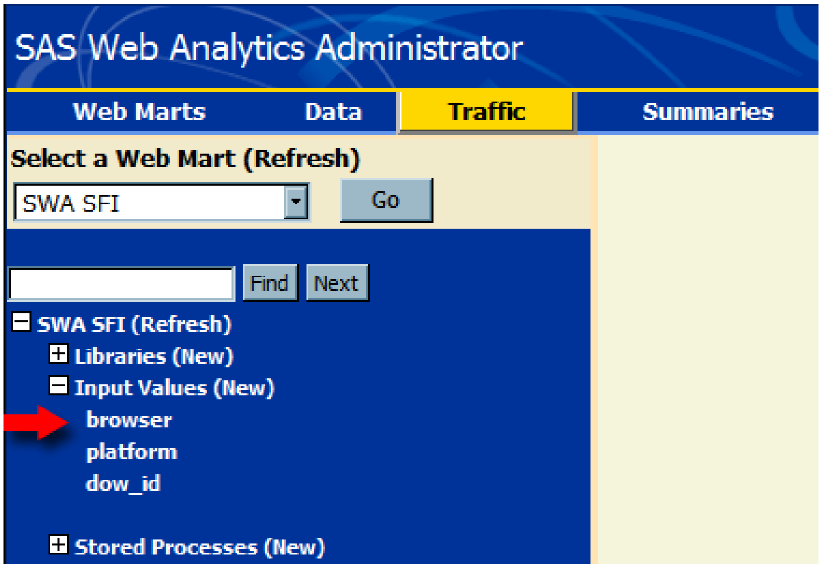
Stp contains stored processes.

Config contains metadata that is used by the SAS Web Analytics application.

Input Values

An *input value* is a pointer to a data set that contains values that populate a drop-down list in the SAS Web Analytics Report Viewer. SAS Web Analytics provides the default input values for some commonly-used categories (see the following display).

Display 9.10 The Default Categories for Input Values



For example, the dow_id category refers to a data set that contains the days of the week. A drop-down menu will be populated with the contents of that data set so a user can select what day of the week to display first in a specified report.

For how to use input values to create a customized drop-down list in a report, see "Implementing a Customized Drop-Down Menu in a Report" on page 180.

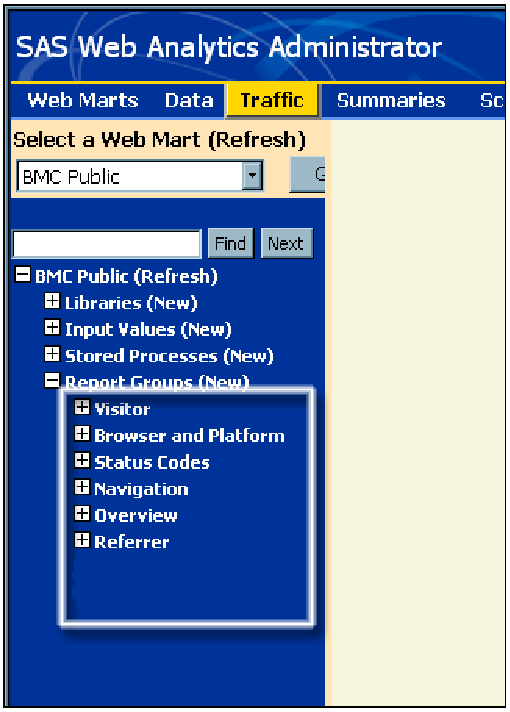
Stored Processes

See "Creating a Stored Process for a New Report" on page 249 for how to create and use a stored process in report definitions.

Report Groups

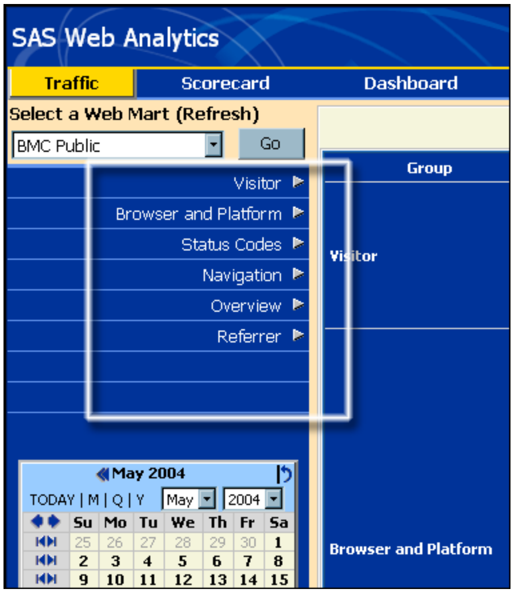
A report group is a category in the SAS Web Analytics Administrator to which you can assign related reports. The default report groups that are located on the Traffic page of the SAS Web Analytics Administrator are shown in the following display:

Display 9.11 Default Report Groups in the SAS Web Analytics Administrator



The report groups in the SAS Web Analytics Administrator correspond to the report groups that appear in the Traffic page of the the SAS Web Analytics Report Viewer. The default report groups in the SAS Web Analytics Report Viewer are shown in the following display:

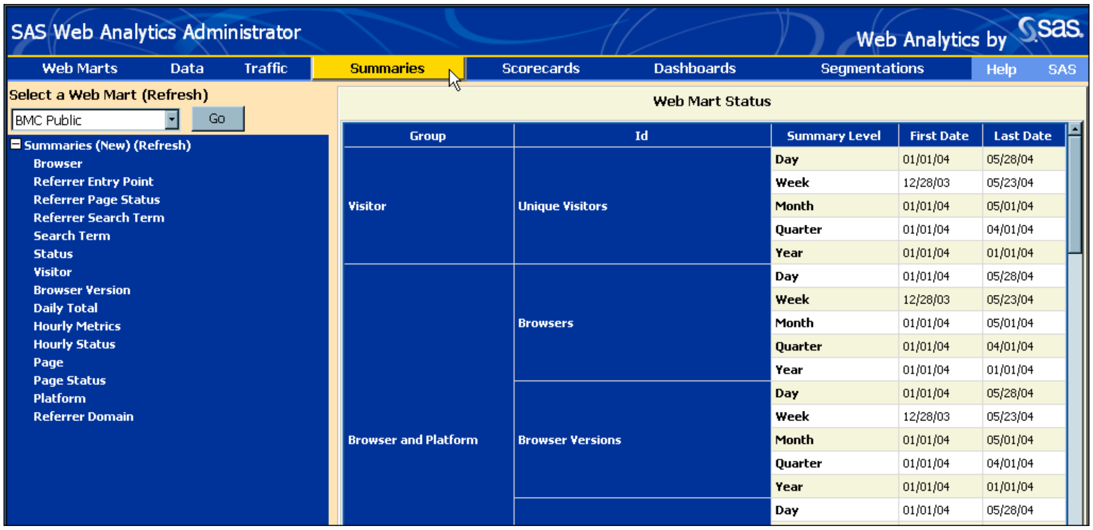
Display 9.12 Default Report Groups in the SAS Web Analytics Report Viewer



See “Working with Report Groups” on page 144.

The Summaries Page

Display 9.13 The Summaries Page of the SAS Web Analytics Administrator



Summaries (New) (Refresh)
- Browser
- Referrer Entry Point
- Referrer Page Status
- Referrer Search Term
- Search Term
- Status
- Visitor
- Browser Version
- Daily Total
- Hourly Metrics
- Hourly Status
- Page
- Page Status
- Platform
- Referrer Domain

Web Mart Status

Group	Id	Summary Level	First Date	Last Date
Visitor	Unique Visitors	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
Browser and Platform	Browsers	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
	Browser Versions	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
		Day	01/01/04	05/28/04

In the Summaries page of the SAS Web Analytics Administrator, you can create, modify, or delete data summarizations. The initial display on the right of the Summaries page is the Web Mart Status table. The Web Mart Status table displays the date ranges for which the Web log data has been processed for the selected Web mart. When you select one of the summaries in the navigation tree on the left of the Summaries page, the Summary window opens. See also "The Waadmsum Data Set" on page 291.

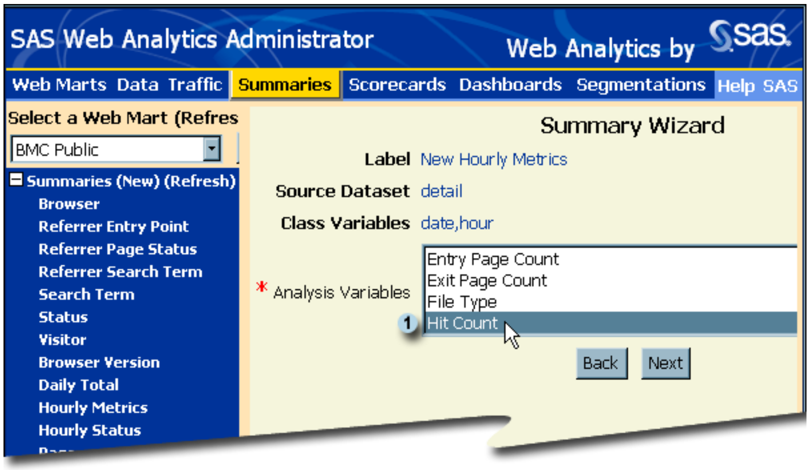
Working with Summaries

The Summary page uses the following conventions:

Required fields
Fields in a form that are labeled with a red asterisk (*) are required fields. You must supply information for a required field in order to continue a task.

Drop-down menus
Use the standard Windows keyboard conventions to select contiguous or noncontiguous items from a drop-down menu.

Variable labels and variable names
In the Summary Wizard, a drop-down menu (of the variables to select) shows the variable labels rather than the variable names. When the Summary Wizard confirms your variable choices, however, it displays the variable name(s) that correspond to the variable names in the Waadmsum data set. For example, `Hit Count` ❶ is an analysis variable label that is selected from the drop-down menu in the following display:



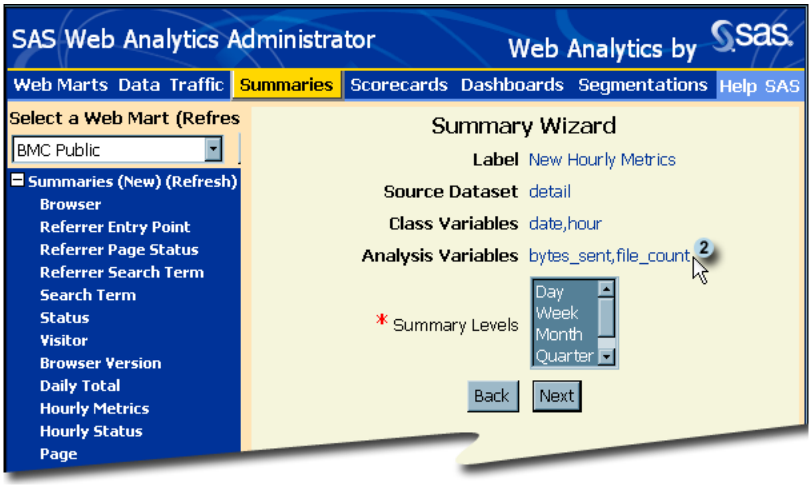
Label New Hourly Metrics

Source Dataset detail

Class Variables date,hour

* Analysis Variables Hit Count

In contrast, the corresponding variable name that is displayed in the confirming text is **file_count** ❷ in the following display:



Label New Hourly Metrics

Source Dataset detail

Class Variables date,hour

Analysis Variables bytes_sent,file_count

* Summary Levels

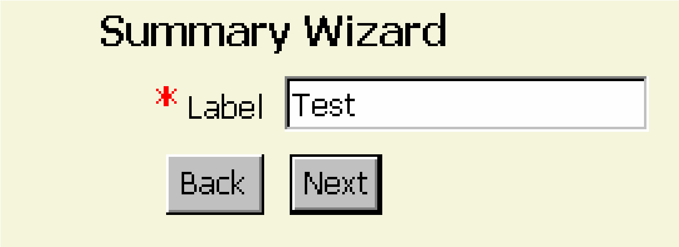
Creating a New Summarization

Before creating a new summarization family, you need to clearly understand the metrics that you want to summarize and how to implement your new summarization family. It can be useful to write a trial SUMMARY procedure step that is modeled on one of the examples presented in “General Information About SAS Web Analytics Summaries” on page 293. To create a new summarization, perform the following general steps:

1 Expand the navigation tree to view the item or the component of the item. Depending on the item, you might need to make several clicks in order to expose the desired component in the SAS Web Analytics Administrator.

2 Select **New** (in parentheses) next to **Summaries**.

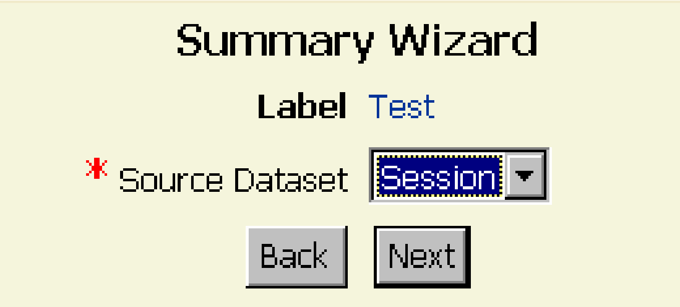
3 Enter a name for the summary in the **Label** field and click **Next**.



Summary Wizard

* Label Test

4 Select the data set (either Detail or Session) to summarize, and click **`Next`**. The Detail data set from the Web server log file contains all data from the SAS e-Data-ETL application and from other marker variables. The Session data set contains one record per session.



Summary Wizard

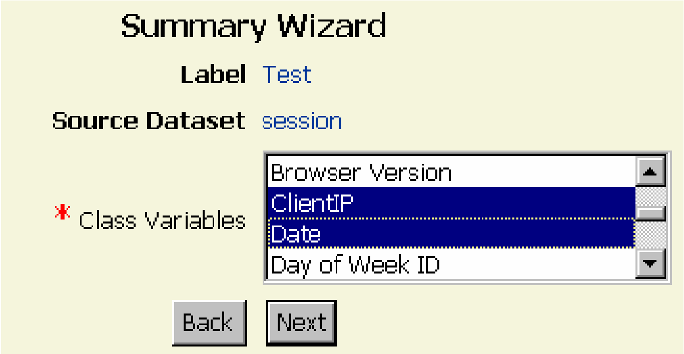
Label Test

* Source Dataset Session

5 Select the classification variable(s) for the summarization (for example, Date and ClientIP).

Note: All summarizations require the Date variable; specify additional variables according to the type of analyses that you need. △

Click **`Next`**.



Summary Wizard

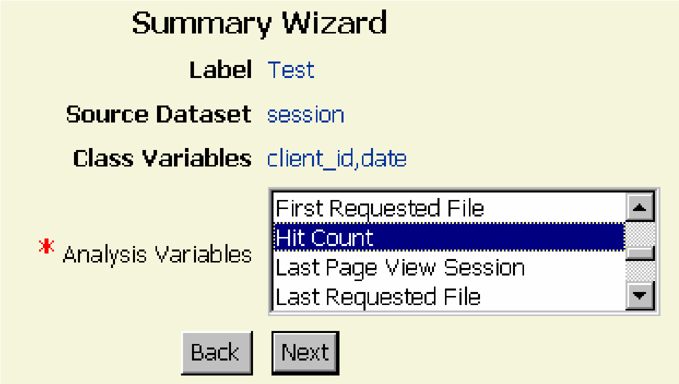
Label Test

Source Dataset session

* Class Variables

- Browser Version
- ClientIP
- Date
- Day of Week ID

6 Select the analysis variables to be summarized. The values must be numeric. Click **`Next`**.



Summary Wizard

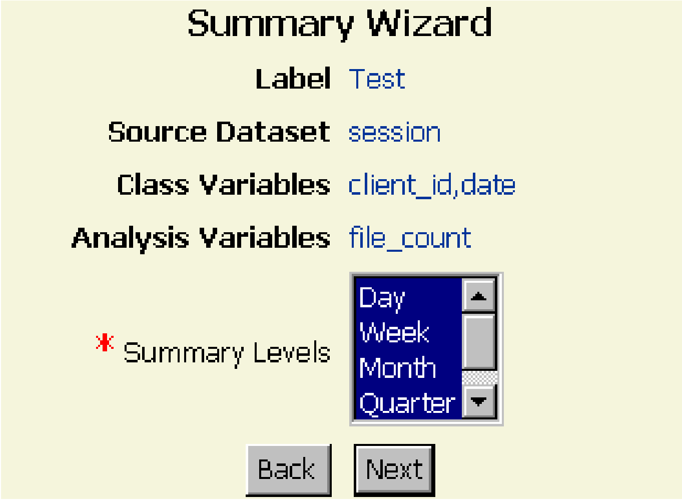
Label Test

Source Dataset session

Class Variables client_id,date

* Analysis Variables Hit Count

7 Select the summary levels. All summary levels (Day, Week, Month, Quarter, and Year) are selected by default. Click **Next**.



Summary Wizard

Label Test

Source Dataset session

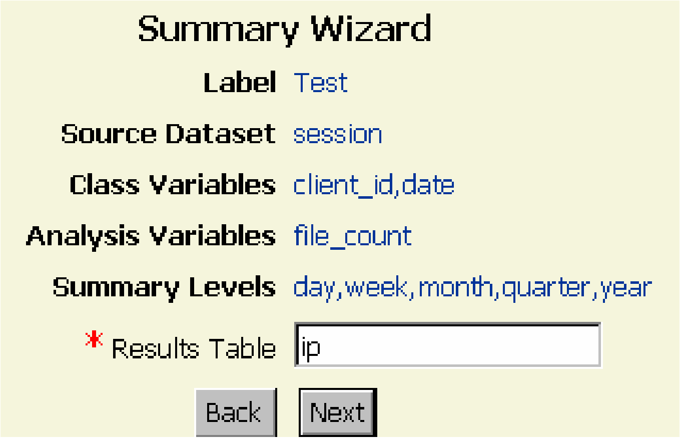
Class Variables client_id,date

Analysis Variables file_count

* Summary Levels Day, Week, Month, Quarter

8 Enter the name of new summary data set in the **Results Table** field. **Results Table** is the name of the data set with the summarization results in the Summary library.

Note: Each data set has a corresponding summary level with an identifying ending. For example, Ip_Week is the data set that contains the weekly version of the IP summarization. △



Summary Wizard

Label Test

Source Dataset session

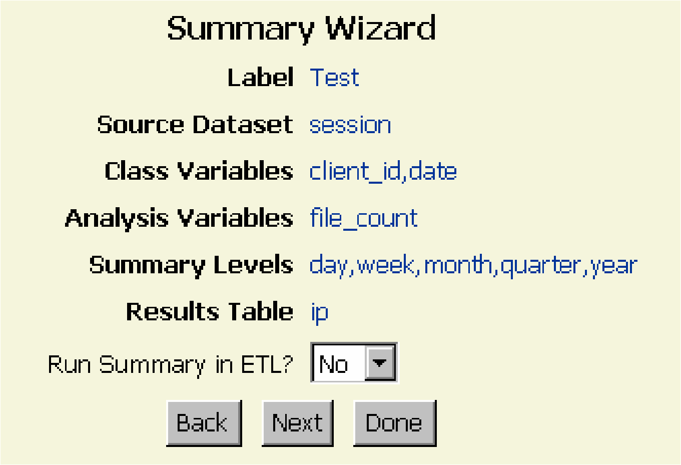
Class Variables client_id,date

Analysis Variables file_count

Summary Levels day,week,month,quarter,year

* Results Table ip

9 The **Run Summary in ETL** field specifies whether to create the newly defined summary when the %WAETL macro runs (for more information, see "Purpose of the %WADECIDE Macro" on page 262). The default value, **No**, specifies that you do not want to create this summary the next time that the %WAETL macro runs. If you are ready for the %WAETL macro to create this summary, then change the value to **Yes**.



Summary Wizard

Label Test

Source Dataset session

Class Variables client_id,date

Analysis Variables file_count

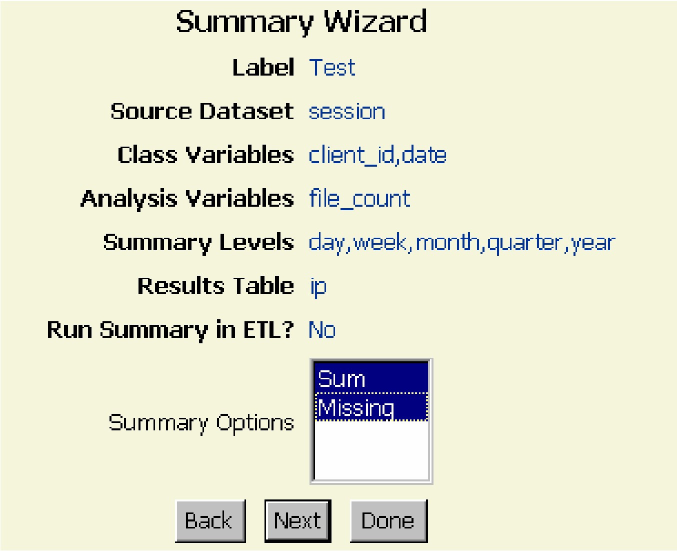
Summary Levels day,week,month,quarter,year

Results Table ip

Run Summary in ETL? No

At this point, you have specified all of the the required parameters for a summarization. You can either click **Done** and **Finish** to complete the summary definition, or click **Next** to specify values for optional parameters. For an explanation of how the optional parameters operate in a summarization, see the documentation for the MEANS procedure (PROC MEANS) in SAS OnlineDoc under Base SAS Procedures. Also, you should be thoroughly familiar with the material in "The Waadmsum Data Set" on page 291 before you use the optional summarization parameters.

10 If you click **Next**, the MISSING and SUM options are selected in the **Summary Options** list by default. The MISSING option indicates that the values for missing classification variables should be kept.



Summary Wizard

Label Test

Source Dataset session

Class Variables client_id,date

Analysis Variables file_count

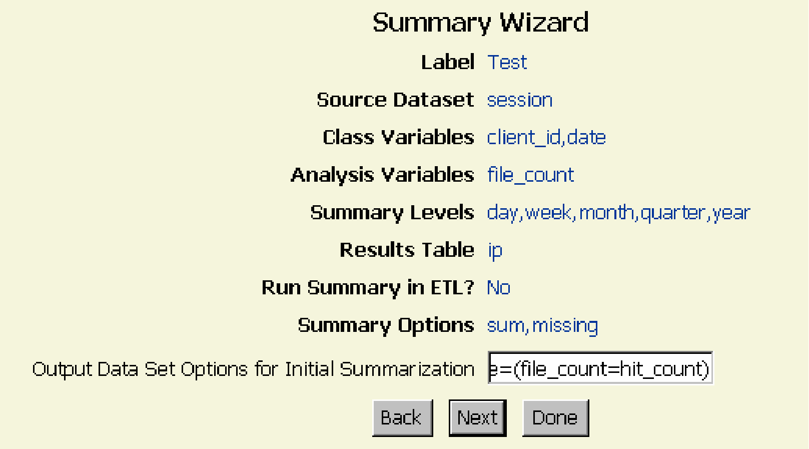
Summary Levels day,week,month,quarter,year

Results Table ip

Run Summary in ETL? No

Summary Options Sum, Missing

11 To specify a value for the **`Output Data Set Options for Initial Summarization`** field, see SAS OnlineDoc for the data set options that can be used in procedures for output data sets.



Summary Wizard

Label Test

Source Dataset session

Class Variables client_id,date

Analysis Variables file_count

Summary Levels day,week,month,quarter,year

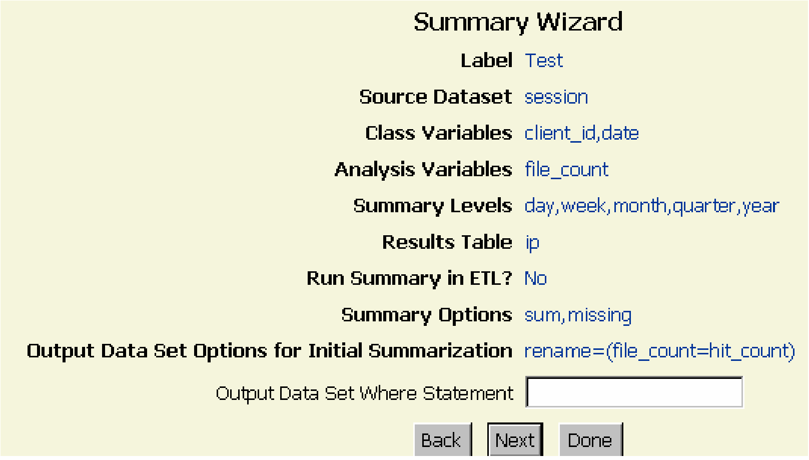
Results Table ip

Run Summary in ETL? No

Summary Options sum,missing

Output Data Set Options for Initial Summarization e=(file_count=hit_count)

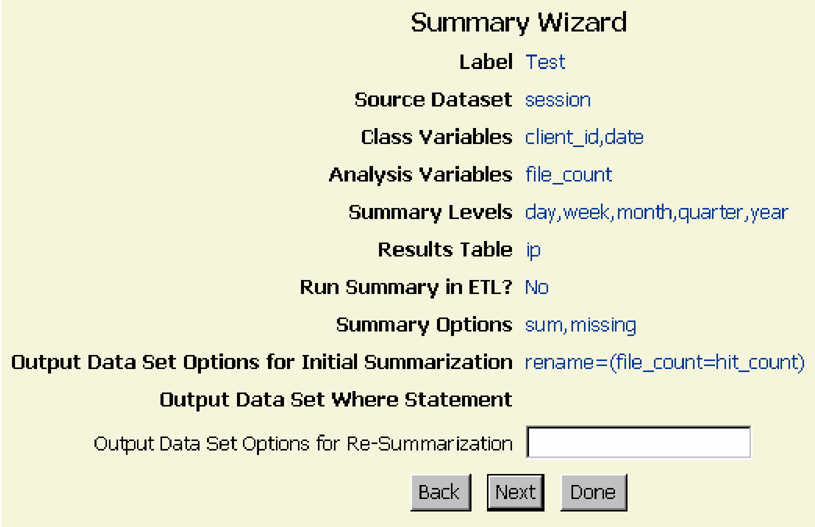
12 For the syntax to use in a WHERE statement used in the **`Output Data Set Where Statement`** field, see SAS OnlineDoc for more information.



Summary Wizard

Label	Test
Source Dataset	session
Class Variables	client_id,date
Analysis Variables	file_count
Summary Levels	day,week,month,quarter,year
Results Table	ip
Run Summary in ETL?	No
Summary Options	sum,missing
Output Data Set Options for Initial Summarization	rename=(file_count=hit_count)
Output Data Set Where Statement	

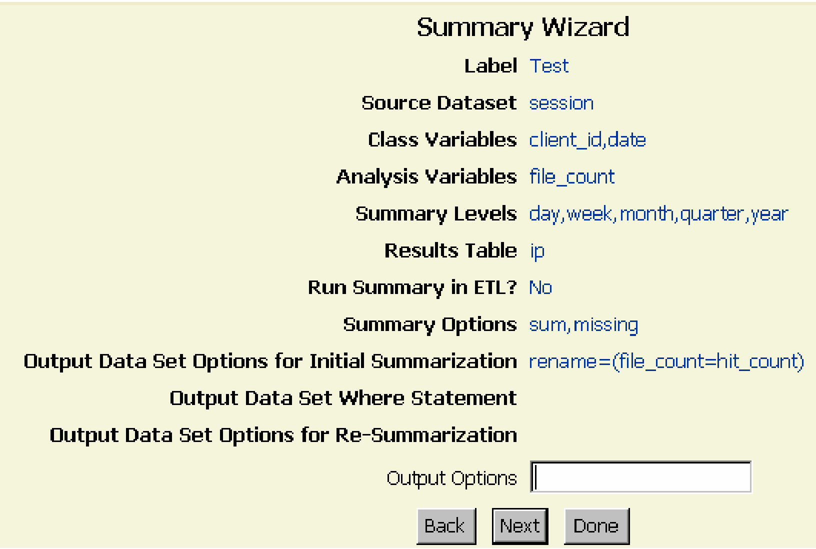
13 The syntax to use for the **`Output Data Set Options for Re-summarization`** field is similar to the entry for the **`Output Data Set Where Statement`** field. The SUMMARY engine requires two summarizations to produce the final result. The information that you provide in the **`Output Data Set Options for Initial Summarization`** field is for the initial summarization. The information that you provide in the **`Output Data Set Where Statement`** field is for the second summarization.



Summary Wizard

Label	Test
Source Dataset	session
Class Variables	client_id,date
Analysis Variables	file_count
Summary Levels	day,week,month,quarter,year
Results Table	ip
Run Summary in ETL?	No
Summary Options	sum,missing
Output Data Set Options for Initial Summarization	rename=(file_count=hit_count)
Output Data Set Where Statement	
Output Data Set Options for Re-Summarization	

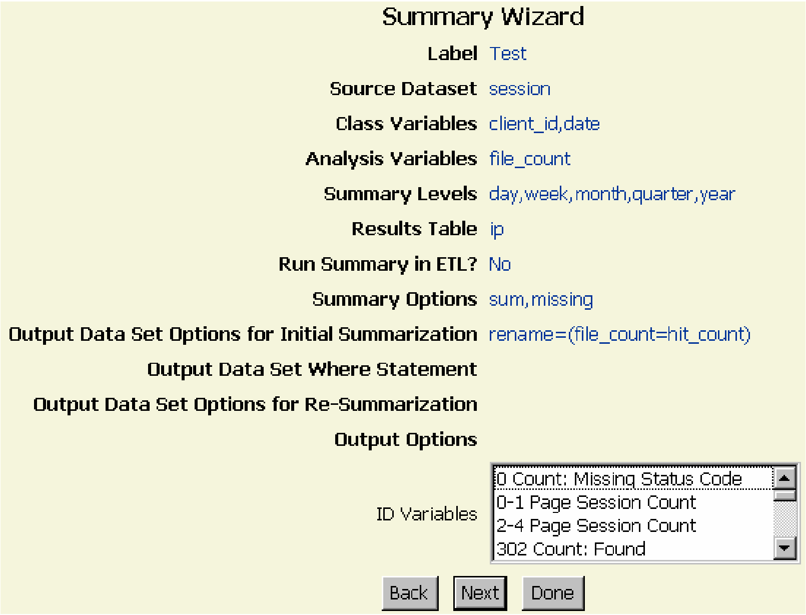
14 See SAS OnlineDoc for the options that can be used with the OUTPUT statement in the MEANS procedure for the **`Output Options`** field.



Summary Wizard

Label	Test
Source Dataset	session
Class Variables	client_id,date
Analysis Variables	file_count
Summary Levels	day,week,month,quarter,year
Results Table	ip
Run Summary in ETL?	No
Summary Options	sum,missing
Output Data Set Options for Initial Summarization	rename=(file_count=hit_count)
Output Data Set Where Statement	
Output Data Set Options for Re-Summarization	
Output Options	

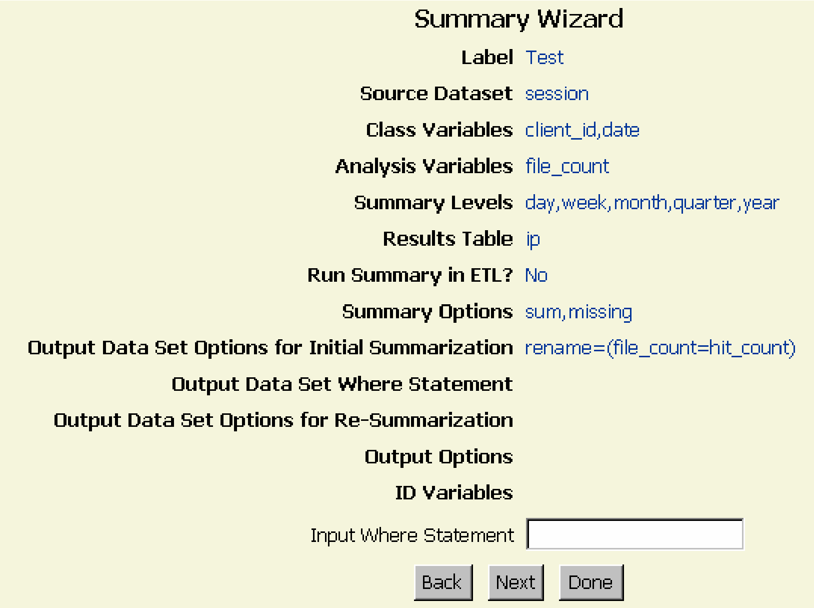
15 See SAS OnlineDoc for how to use and specify the values for the **`ID Variables`** field in the MEANS procedure.



Summary Wizard

Label	Test
Source Dataset	session
Class Variables	client_id,date
Analysis Variables	file_count
Summary Levels	day,week,month,quarter,year
Results Table	ip
Run Summary in ETL?	No
Summary Options	sum,missing
Output Data Set Options for Initial Summarization	rename=(file_count=hit_count)
Output Data Set Where Statement	
Output Data Set Options for Re-Summarization	
Output Options	
ID Variables	0 Count: Missing Status Code 0-1 Page Session Count 2-4 Page Session Count 302 Count: Found

16 The **`Input Where Statement`** field refers to the WHERE statement that will be applied to the input data set that is specified in the **`Source Data Set`** field. Click **`Next`** and click **`Done`**.



Summary Wizard

Label	Test
Source Dataset	session
Class Variables	client_id,date
Analysis Variables	file_count
Summary Levels	day,week,month,quarter,year
Results Table	ip
Run Summary in ETL?	No
Summary Options	sum,missing
Output Data Set Options for Initial Summarization	rename=(file_count=hit_count)
Output Data Set Where Statement	
Output Data Set Options for Re-Summarization	
Output Options	
ID Variables	
Input Where Statement	

Modifying or Deleting a Summarization

To modify or delete a summarization, perform the following general steps:

1 Expand the navigation tree to view the summarization.

2 Select the summarization that you want to modify or delete by clicking it.

3 In the table that appears on the right, select **Modify** or **Delete**.

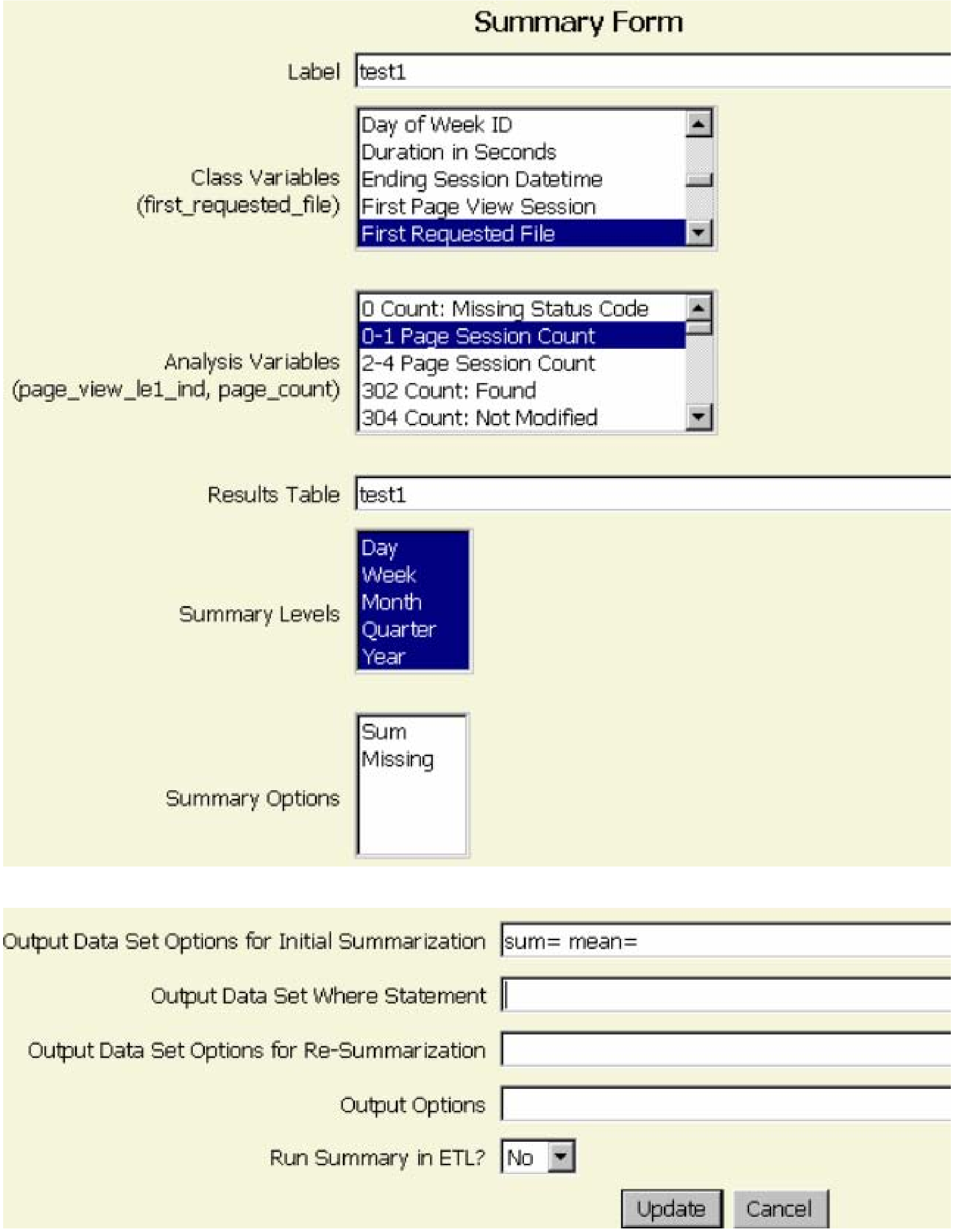
4 If you are modifying a summarization, then either select the variables you want to modify from a drop-down menu or type the new information into a text box (see Display 9.14 on page 138). When you are finished, click **Update**, and click **Continue**. If you are deleting a summarization, then select the appropriate buttons to continue.

5 To verify any modifications that you made, select **Refresh** next to **Select a Web Mart**.

The Summary Form: Specifying the Properties for Customized Summarizations

Use the Summary Form (see the following display) to specify the properties for the initial summarizations and the re-summarizations that are needed by the customized SAS Web Analytics summarizations.

Display 9.14 The Summary Form



Summary Form

Label: test1

Class Variables (first_requested_file):
- Day of Week ID
- Duration in Seconds
- Ending Session Datetime
- First Page View Session
- First Requested File

Analysis Variables (page_view_le1_ind, page_count):
- 0 Count: Missing Status Code
- 0-1 Page Session Count
- 2-4 Page Session Count
- 302 Count: Found
- 304 Count: Not Modified

Results Table: test1

Summary Levels:
- Day
- Week
- Month
- Quarter
- Year

Summary Options:
- Sum
- Missing

Output Data Set Options for Initial Summarization: sum= mean=

Output Data Set Where Statement:

Output Data Set Options for Re-Summarization:

Output Options:

Run Summary in ETL? No

The following table indicates how to specify the properties that advanced programmers can use to define customized summarizations for SAS Web Analytics reports:

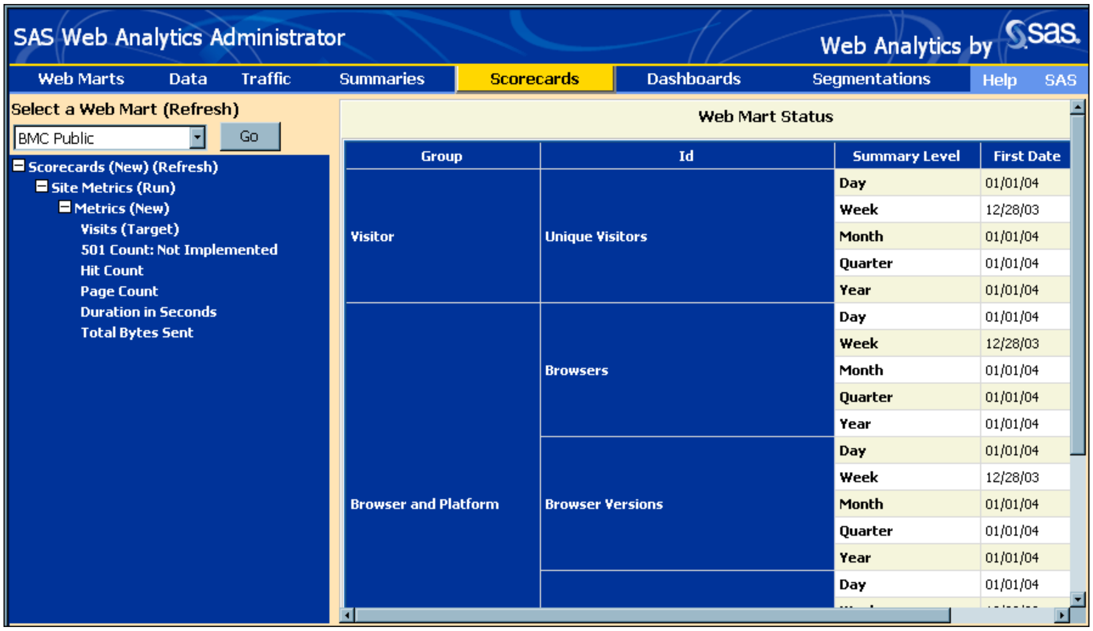
Table 9.2 Using the Fields in the Summary Form

Field	**Description**
Label	A name that identifies the summarization that you are creating or modifying.
Class Variable(s)	One or more variables by which the analysis variables are to be summarized. The previously selected class variable(s) are displayed in parentheses next to the selection box. ***CAUTION:*** **The DATE variable must be one of the classification variables or the summarization will fail.** △
Analysis Variable(s)	One or more variables to be summarized. The previously selected class analysis variable(s) will be displayed in parentheses next to the selection box.
Result Table	Name of the summary data set that will be created. The result table name can have a maximum length of 27 characters.
Summary Levels	The time interval(s) for which you want to summarize the data. To use the day, week, month, quarter, and year summary levels, the data set must contain the following SAS Date variables: □ **DATE** – daily SAS Date value □ **START_OF_WEEK** – the SAS Date value for the first day in the week □ **START_OF_MONTH** – the SAS Date value for the first day in the month □ **START_OF_QTR** – the SAS Date value for the first day in the quarter □ **START_OF_YEAR** – the SAS Date value for the first day in the year
Summary Options	The options for the SUMMARY procedure (PROC SUMMARY). MISSING and SUM and the most commonly used options. CHARTYPE is always used by default and does not need to be explicitly listed. See also the SAS documentation for the SUMMARY procedure.
Output Data Set Options for Initial Summarization	The data set options for the output data sets created by the SUMMARY procedure. See the SAS documentation for more information about data set options. See also “The Waadmsum Data Set” on page 291 for more information about initial summarizations.
Output Data Set Where Statement	A syntactically correct "where" statement for the output data set.

Field	Description
Output Data Set Options for Re-Summarization	The data set options for the output data sets created by the SUMMARY procedure. See the SAS documentation for more information about data set options. See also “The Waadmsum Data Set” on page 291 for more information about initial summarizations.
Output Options	See the online help for the SUMMARY procedure for a list of available output options.
Run Summary in ETL?	Determines whether the summary definition is run when the %WAETL macro is run. The default is no, so you can test the summary definition by using the %WASUMTST macro (see “The %WASUMTST Macro” on page 274).

The Scorecards Page

Display 9.15 The Scorecards Page of the SAS Web Analytics Administrator



Scorecards (New) (Refresh)
- Site Metrics (Run)
  - Metrics (New)
    - Visits (Target)
    - 501 Count: Not Implemented
    - Hit Count
    - Page Count
    - Duration in Seconds
    - Total Bytes Sent

Web Mart Status

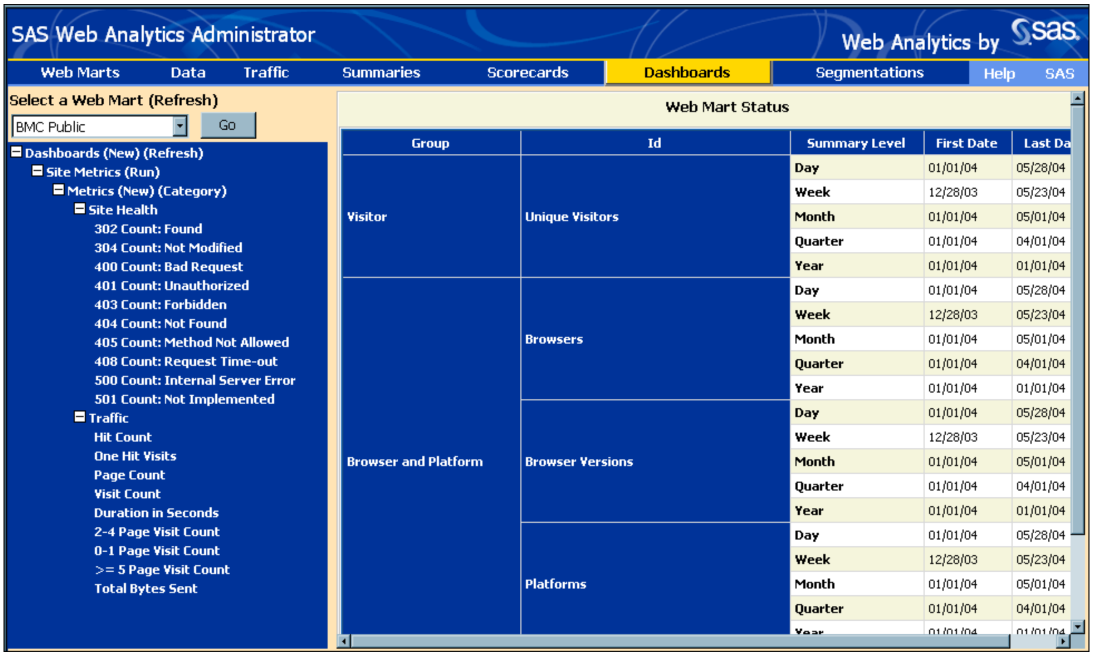
Group	Id	Summary Level	First Date
Visitor	Unique Visitors	Day	01/01/04
		Week	12/28/03
		Month	01/01/04
		Quarter	01/01/04
		Year	01/01/04
Browser and Platform	Browsers	Day	01/01/04
		Week	12/28/03
		Month	01/01/04
		Quarter	01/01/04
		Year	01/01/04
	Browser Versions	Day	01/01/04
		Week	12/28/03
		Month	01/01/04
		Quarter	01/01/04
		Year	01/01/04
		Day	01/01/04

In the Scorecards page of the SAS Web Analytics Administrator (see the previous display), you can create, modify or delete scorecards. After you select a Web mart, the Scorecards page displays the Web Mart Status Table on the right. The Web Mart Status table displays the date ranges for which the Web log data has been processed for the selected Web mart. For more about the dates in this table, see “Understanding Date Ranges in the Web Mart Status Table” on page 113.

To customize data for scorecards, see “Overview: Scorecard” on page 190.

The Dashboards Page

Display 9.16 The Dashboards Page of the SAS Web Analytics Administrator



Select a Web Mart (Refresh)

BMC Public

Web Mart Status

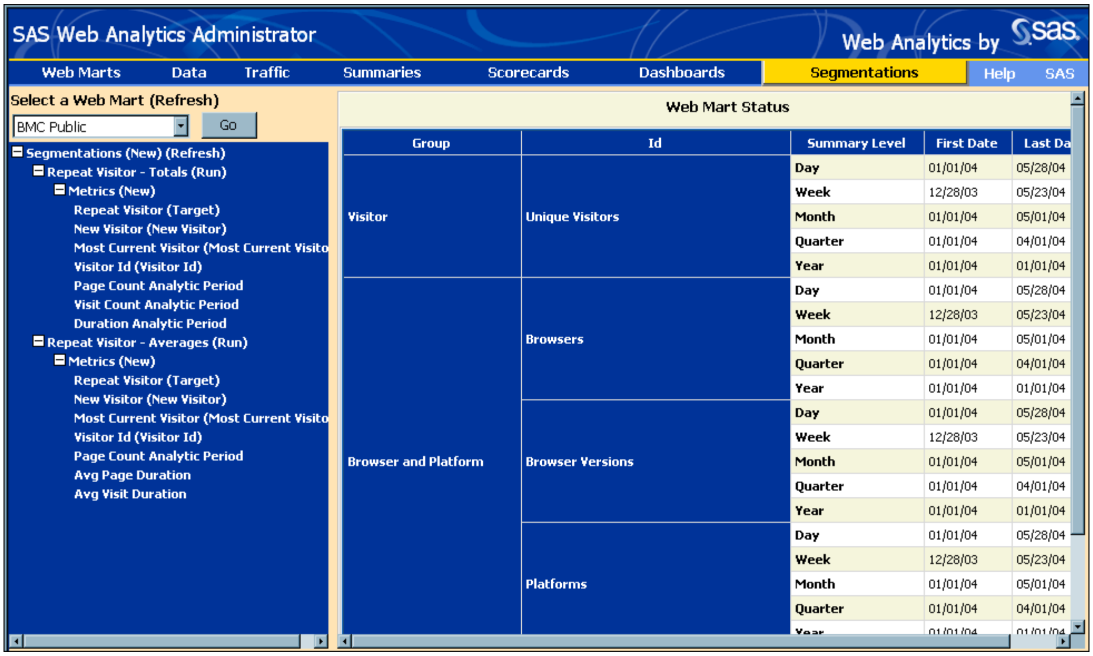
Group	Id	Summary Level	First Date	Last Da
Visitor	Unique Visitors	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
Browser and Platform	Browsers	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
	Browser Versions	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
	Platforms	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04

In the Dashboards page of the SAS Web Analytics Administrator (see the previous display), you can create, modify or delete dashboards. After you select a Web mart, the Dashboards page displays the Web Mart Status table on the right. The Web Mart Status table displays the date ranges for which the Web log data has been processed for the selected Web mart. For more about the dates in this table, see “Understanding Date Ranges in the Web Mart Status Table” on page 113.

To customize data for dashboards, see “Overview: Dashboard” on page 207.

The Segmentations Page

Display 9.17 The Segmentations Page of the SAS Web Analytics Administrator



- Segmentations (New) (Refresh)
  - Repeat Visitor - Totals (Run)
    - Metrics (New)
      - Repeat Visitor (Target)
      - New Visitor (New Visitor)
      - Most Current Visitor (Most Current Visito
      - Visitor Id (Visitor Id)
      - Page Count Analytic Period
      - Visit Count Analytic Period
      - Duration Analytic Period
  - Repeat Visitor - Averages (Run)
    - Metrics (New)
      - Repeat Visitor (Target)
      - New Visitor (New Visitor)
      - Most Current Visitor (Most Current Visito
      - Visitor Id (Visitor Id)
      - Page Count Analytic Period
      - Avg Page Duration
      - Avg Visit Duration

Web Mart Status

Group	Id	Summary Level	First Date	Last Da
Visitor	Unique Visitors	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
Browser and Platform	Browsers	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
	Browser Versions	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04
	Platforms	Day	01/01/04	05/28/04
		Week	12/28/03	05/23/04
		Month	01/01/04	05/01/04
		Quarter	01/01/04	04/01/04
		Year	01/01/04	01/01/04

In the Segmentations page of the SAS Web Analytics Administrator (see the previous display), you can create, modify or delete segmentation reports. After you select a Web mart, the Segmentations page displays the Web Mart Status table on the right. The Web Mart Status table displays the date ranges for which the Web log data has been processed for the selected Web mart. For more about the dates in this table, see “Understanding Date Ranges in the Web Mart Status Table” on page 113.

To customize data for segmentation reports, see Chapter 13, “ Setting Up a Segmentation Report,” on page 219.

CHAPTER

10

Administration of SAS Web Analytics Reports

Working with Report Groups

Create a New Report Group

For the definition of a report group, and a list of the default report groups that are provided in the SAS Web Analytics Administrator, see "Report Groups" on page 127.

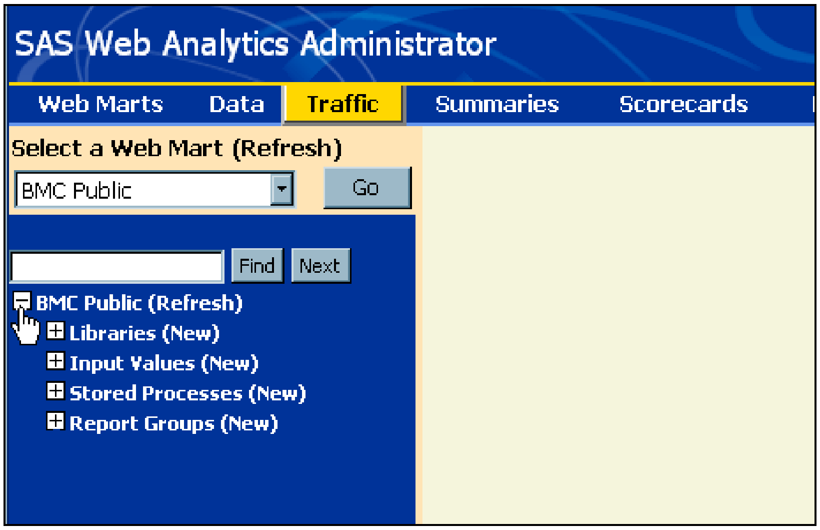
To create a new report group using the SAS Web Analytics Administrator, perform the following steps:

Note: The SAS Web Analytics Administrator is a Web-based application. To access the SAS Web Analytics Administrator in your Internet browser, select the URL address that you configured during the setup of the SAS Web Analytics solution. △

1 Select **`Traffic`** in the main menu bar of the SAS Web Analytics Administrator.

2 If no Web mart is selected, or if the Web mart that is displayed is not correct, then select a Web mart from the **`Select a Web Mart`** drop-down menu. Click **`Go`**.

3 Click the node icon (⊞) next to a Web mart name in the tree view to expand the node. Libraries, Input Values, Stored Processes, and Report Groups nodes appear below the expanded Web mart node. For example, the following display shows the expanded node of the BMC Public Web mart:



4 Click **`(New)`** next to **`Report Groups`** to create a new report group.

5 The following series of prompts for information about the new report group appear in succession. For each item, type the information in the text box and click **`Next`**. An asterisk (*) next to the prompt indicates that you must provide information for that item in order to finish the task.

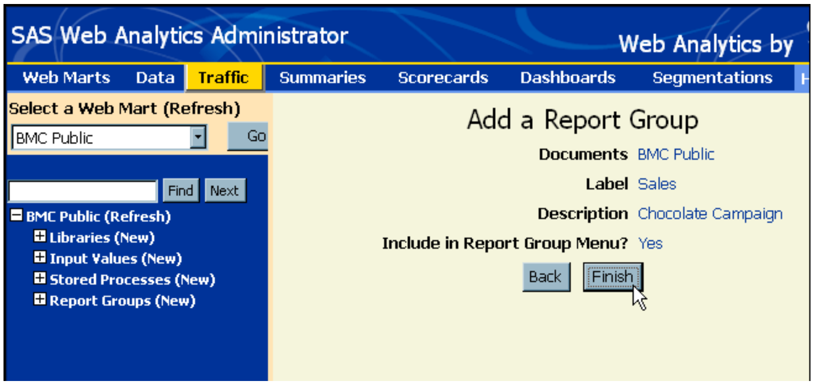
`Enter Label`: The text that you enter here appears in the navigation tree of the Traffic page in the SAS Web Analytics Report Viewer.

`Enter Description`: The text that you enter here appears in the SAS Web Analytics Report Viewer as a description of the report group.

`Include in Report Group Menu`: Leave the default choice as **`Yes`**, and click **`Next`**. The new report group appears in the tree view of the Traffic page in the

SAS Web Analytics Report Viewer. If you select **No**, then the report group (and any reports that are created under it) are hidden.

6 Click **Finish** (see the following display).



Add a Report Group

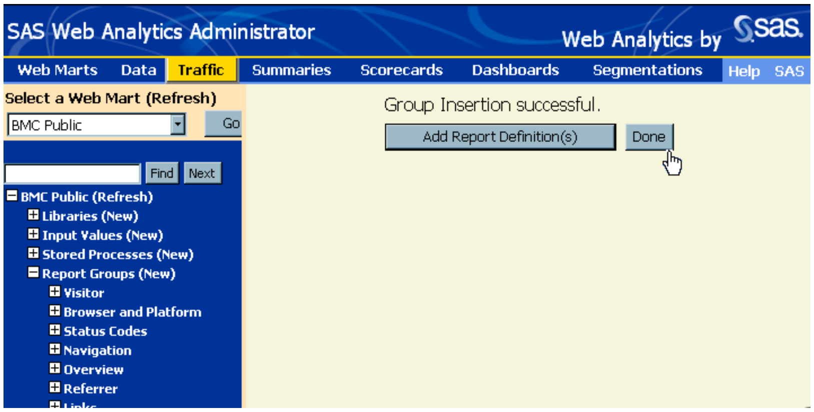
Documents BMC Public

Label Sales

Description Chocolate Campaign

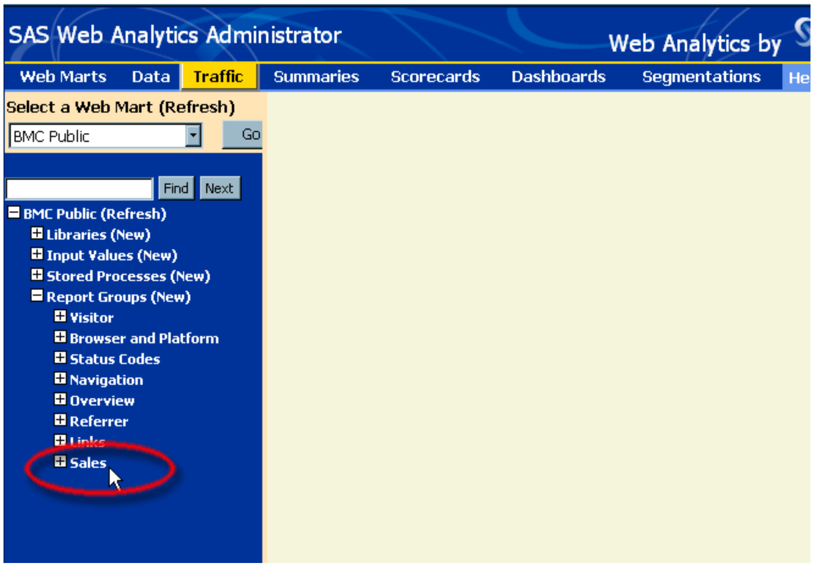
Include in Report Group Menu? Yes

7 Click **Done** (see following display).

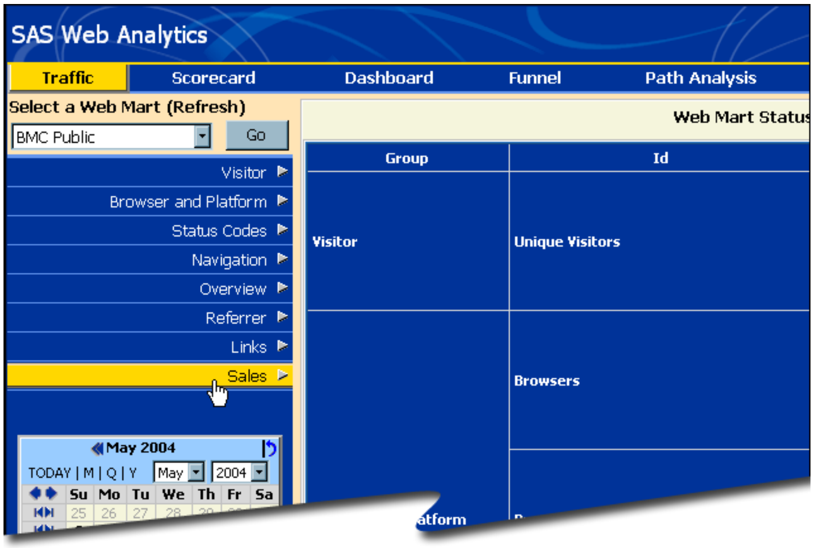


Group Insertion successful.

8 To verify the new report group (**Sales** in the following example), click either **Refresh** from the browser menu or **(Refresh)** next to the Web mart name in the tree view of the SAS Web Analytics Administrator.



If it is not specified as hidden, then the new report group also appears on the Traffic page of the SAS Web Analytics Report Viewer. The report group Sales, which was created in the previous example, is shown in the following display:

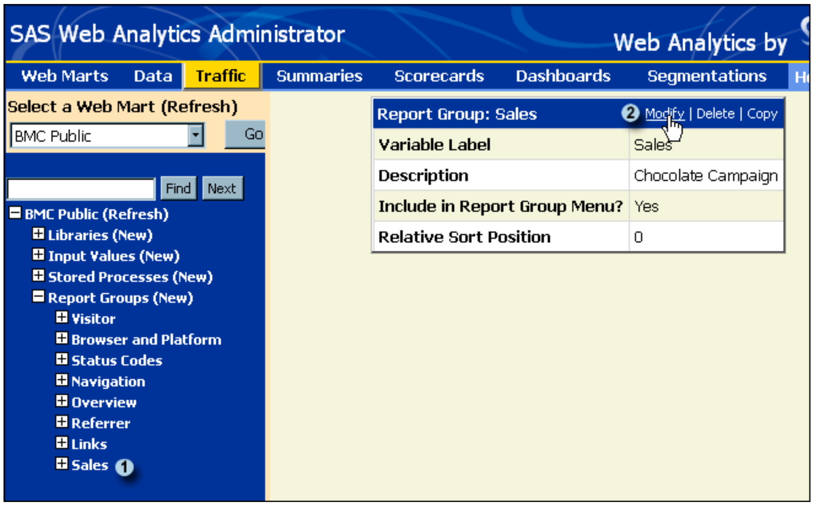


Group	Id
Visitor	Unique Visitors
	Browsers
atform	

Modify a Report Group

To modify the display properties of a report group, perform the following steps:

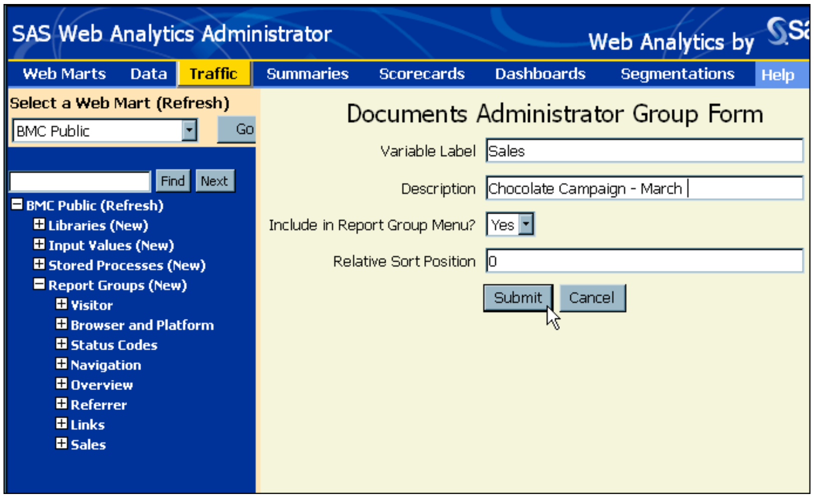
1 Verify that you are on the Traffic page of the SAS Web Analytics Administrator (the **Traffic** link in the menu bar is yellow). In the tree view on the left, select the name of the report group that you want to modify. For example, the Sales report group is selected (❶ in the following display) . A table of properties appears at the right of the page.



Report Group: Sales	
Variable Label	Sales
Description	Chocolate Campaign
Include in Report Group Menu?	Yes
Relative Sort Position	0

2 Select **`Modify`** (❷ in the previous display).

3 In the Documents Administrator Group Form, enter the changes that you want to make to the report group. Click **`Submit`** (see the following display).



Documents Administrator Group Form

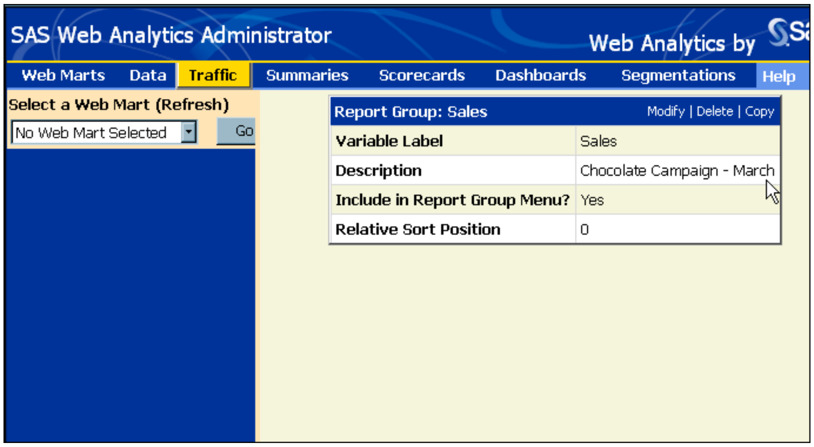
Variable Label: Sales

Description: Chocolate Campaign - March

Include in Report Group Menu? Yes

Relative Sort Position: 0

4 Below the “Status Group Update successful” message that appears, click **`Continue`**. The updated table of properties appears (see the following display).



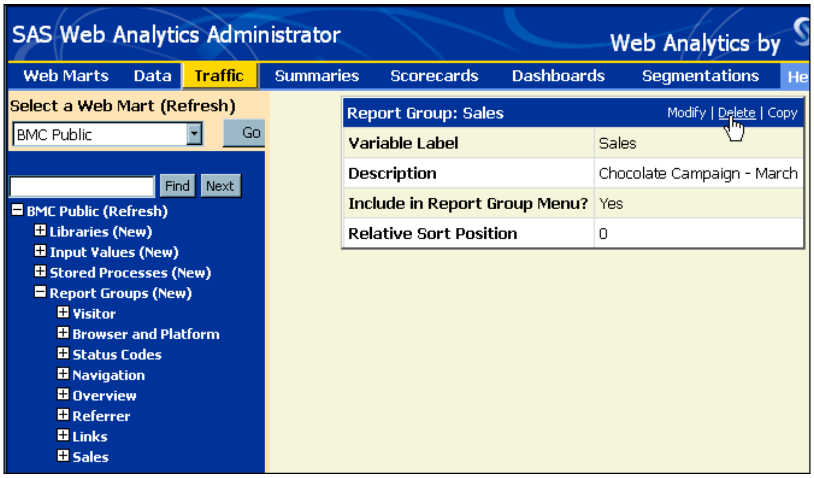
Report Group: Sales	
Variable Label	Sales
Description	Chocolate Campaign - March
Include in Report Group Menu?	Yes
Relative Sort Position	0

Delete a Report Group

To delete a report group, perform the following steps:

1 Verify that you are on the Traffic page of the Web Analytics Administrator (the **Traffic** link in the menu bar is yellow). In the tree view on the left, select the name of the report group that you want to delete. A table of properties for that report group appears at the right of the page.

2 Select **Delete** in the upper right corner of the table (see the following display).



Report Group: Sales	
Variable Label	Sales
Description	Chocolate Campaign - March
Include in Report Group Menu?	Yes
Relative Sort Position	0

3 Click **OK**. Below the "Status Group Deletion successful" message that appears, click **Continue**. The report group is removed from the tree view.

Create a New Report Definition

Before you create a new report definition, determine the report group to which it will belong. If none of the existing report groups is appropriate for the new report, then create a new report group first (see “Create a New Report Group” on page 144).

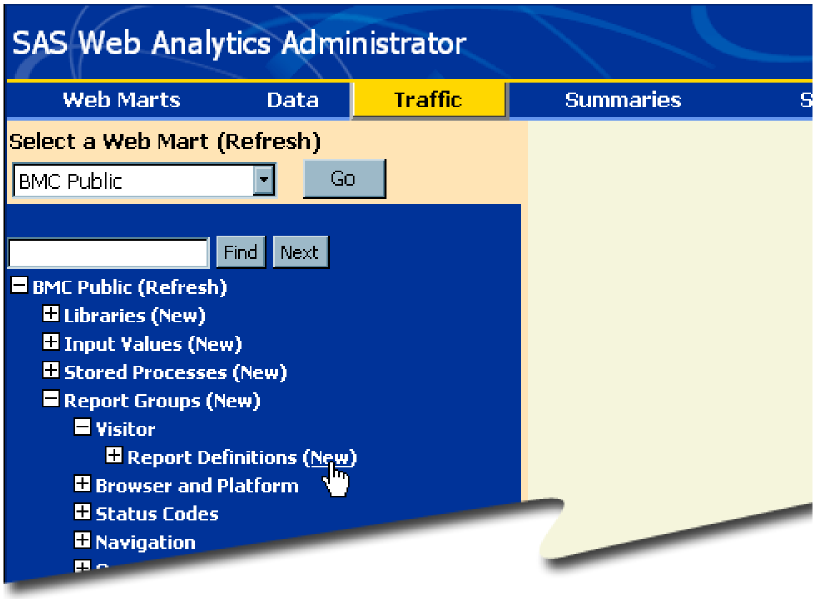
Note: If you are planning to use the data from a stored process to populate a funnel report, then skip this section and see “Creating a Stored Process for a New Report” on page 249. △

To create a new report definition, perform the following steps:

1 Verify that you are on the Traffic page of the SAS Web Analytics Administrator (the **Traffic** link in the menu bar is yellow).

2 Expand the node next to the name of the report group to which you want to add a new report definition. The tree view expands to display a new item, **Report Definitions (New)**, below the selected report group.

3 Click **(New)** next to **Report Definitions** (see the following display).



4 Several prompts for information about the report definition appear. At each prompt, type the information in the text box, and click **Next**. Your typed information is added and the next prompt appears. An asterisk (*) next to a prompt indicates that you must provide information for that item in order to finish the task. The following list describes the information that you are prompted for:

Label
The text that you enter here is displayed in the SAS Web Analytics Report Viewer as the name of the report in the tree view.

URL
The SAS Web Analytics Report Viewer can display reports or can link to Web sites (for example, http://www.sas.com).

Link to a Web site
If you want the report to be a link to a Web site, then type the URL of the Web site in the text box.

Note: If you specify a URL, then the prompts for the library, data set, class variables, and analysis variables fields are not displayed. △

Display a report
If you want the report to be a presentation of variables in a data set, then leave this text box empty. The form will request additional information about the report, as described in the fields that follow (**Library**, **Dataset**, **Class Variables**, and **Analysis Variables**).

Library
In the drop-down menu, select the library that contains the data set that will be displayed in the report. The default list contains the Dated and Summary libraries by default. If your data set is not in one of these libraries, then select the **Show All** option to view an expanded list of libraries.

Dataset
In the drop-down menu, select the data set that contains the variables to display in the report.

Class Variables
In the drop-down menu, select the classification variables to display in the report. Use the standard Windows keyboard conventions to select contiguous or non-contiguous items from the drop-down menu.

CAUTION:
The DATE variable must be one of the classification variables or the summarization will fail. △

Analysis Variables
In the drop-down menu, select the analysis variables to display in the report. Use the standard Windows keyboard conventions to select contiguous or non-contiguous items from the drop-down menu.

5 After you select all of the analysis variables, click **Finish**. Below the “Definition Insertion Successful” message that appears, click **Done**.

Note: The **Add Report Definitions** button enables you to continue creating another report definition (instead of selecting **New** in the navigation tree of the SAS Web Analytics Administrator). △

6 Add a new column (if necessary). See “Add a New Column” on page 154.

7 Add the Date filter (required) and any other filters (if necessary) for your new report (see “Creating Filters” on page 163).

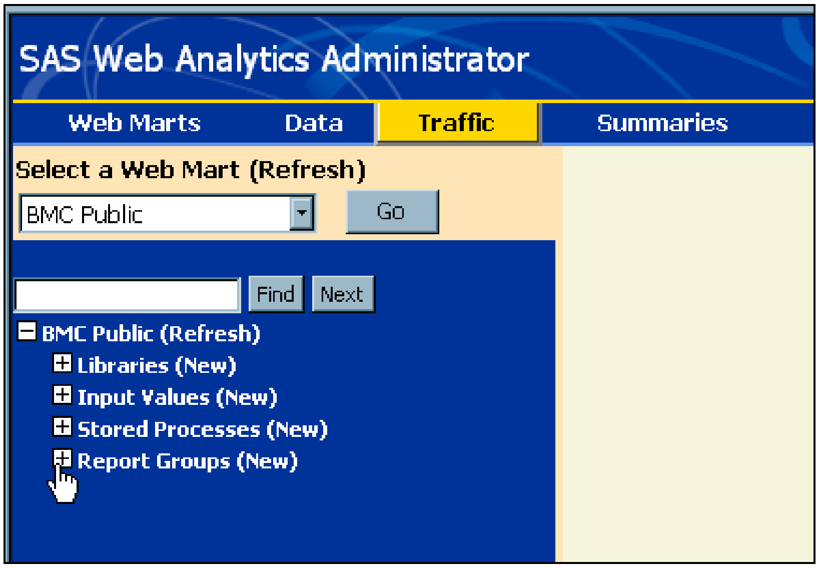
At this point, you have created the basic definition of a report. If a basic report is sufficient for your report presentation needs, then you are finished with your report definition and you only need to run the report job in order to create the report output that is viewed in a browser. However, if you want to customize a report, see “Customize a Report Definition” on page 151. If you want to create a drillable report, then you also need to specify the link(s) from the main report to one or more destination (drilled) report(s). See “Creating Drillable Reports” on page 185.

Customize a Report Definition

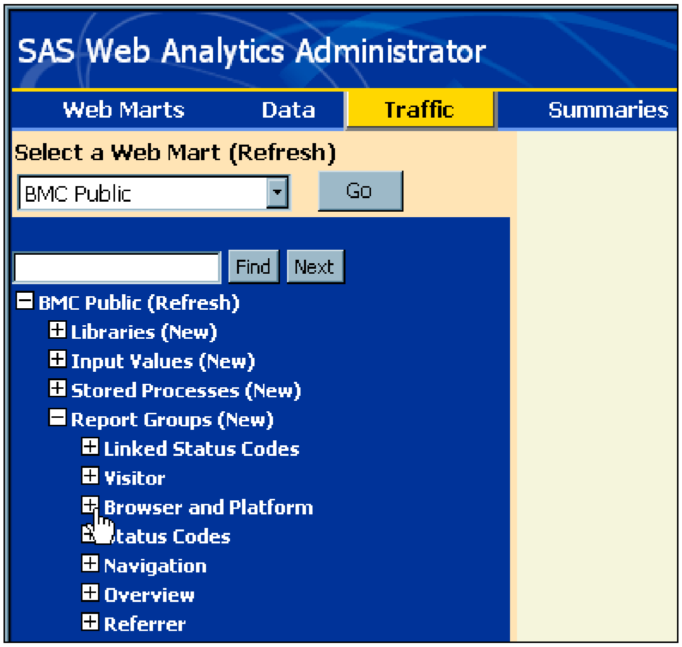
To add refinements to a report, such as a modified report label or the location of a custom Help file for the report, perform the following steps:

1 Verify that you are on the Traffic page of the SAS Web Analytics Administrator (the **`Traffic`** link of the menu bar in the Traffic page is yellow).

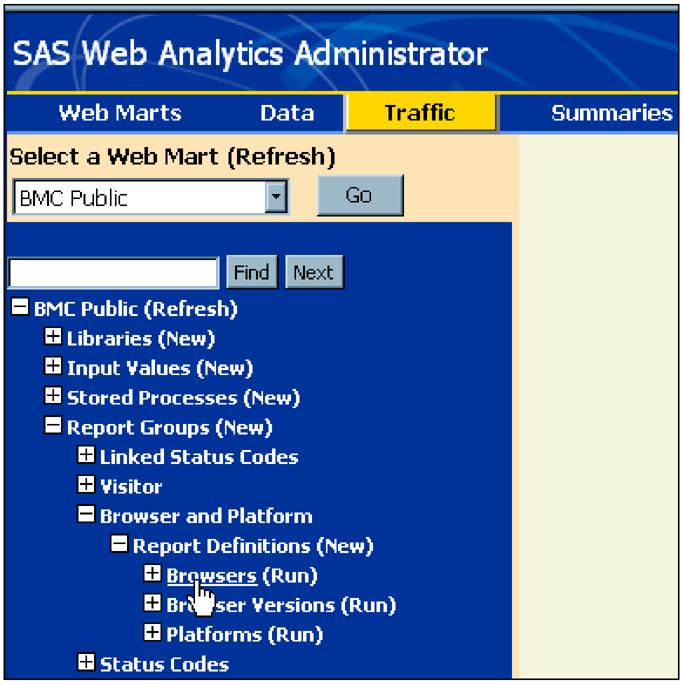
2 Select a Web mart (if necessary). If necessary, expand the Web mart node and the **`Report Groups (New)`** node.



3 Expand the report group that contains the report that you want to refine.



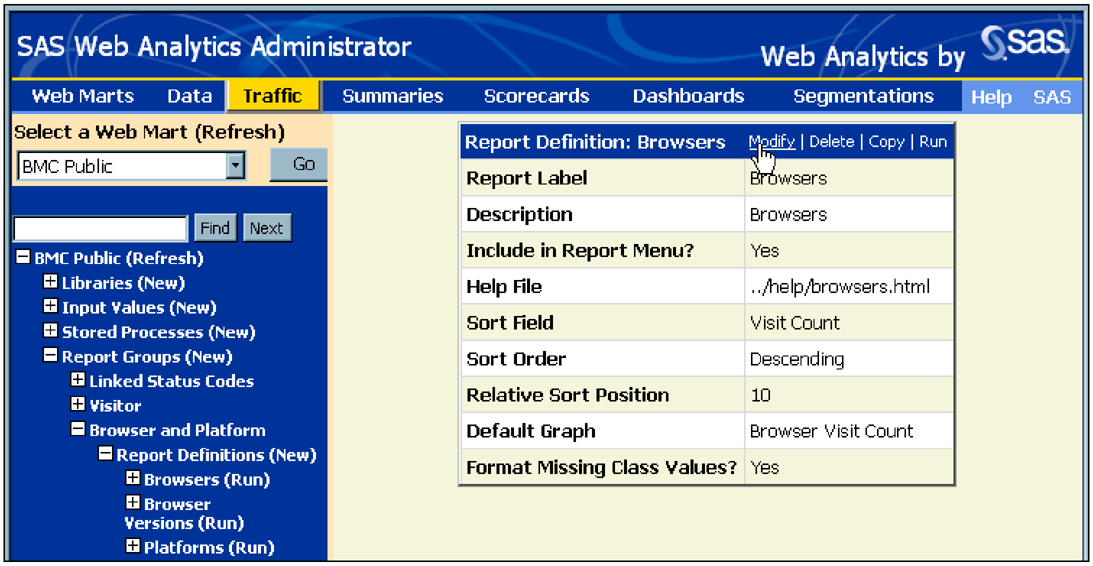
4 Expand **Report Definitions (New)** to view the report list. Select the report that you want to refine in the navigation tree.



The metadata table for the report appears on the right of the Traffic page.

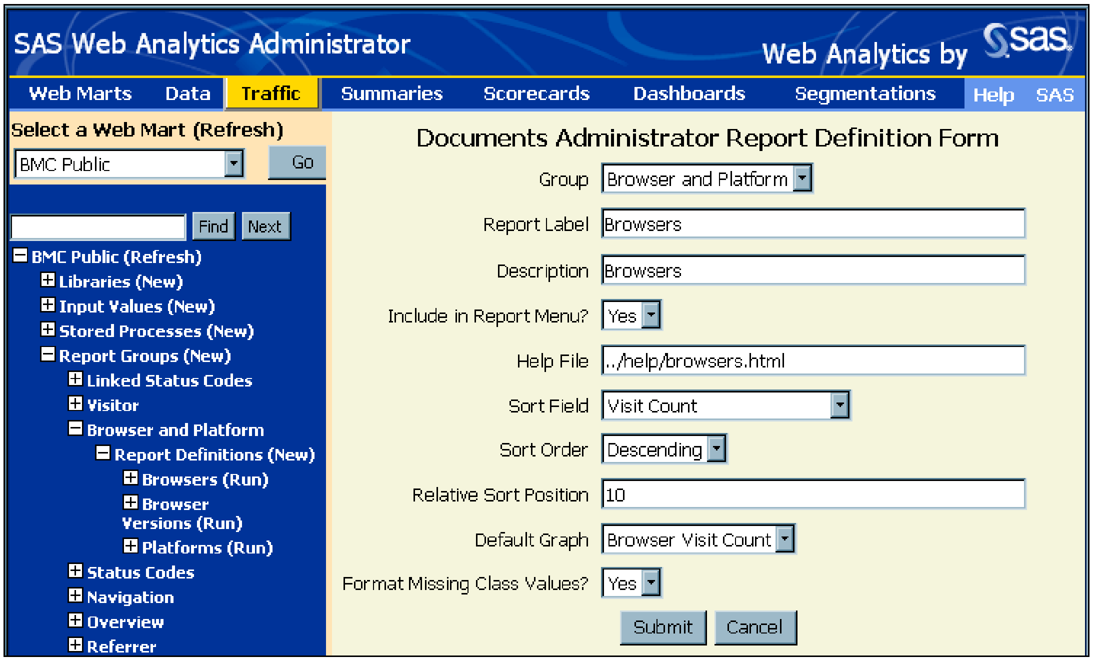
Note: The metadata table for the report is also displayed after you finish a basic report definition. △

5 Select **Modify** in the upper right corner of the table.



Report Definition: Browsers	Modify | Delete | Copy | Run
Report Label	Browsers
Description	Browsers
Include in Report Menu?	Yes
Help File	../help/browsers.html
Sort Field	Visit Count
Sort Order	Descending
Relative Sort Position	10
Default Graph	Browser Visit Count
Format Missing Class Values?	Yes

The Documents Administrator Report Definition Form appears.



Documents Administrator Report Definition Form

Group	Browser and Platform
Report Label	Browsers
Description	Browsers
Include in Report Menu?	Yes
Help File	../help/browsers.html
Sort Field	Visit Count
Sort Order	Descending
Relative Sort Position	10
Default Graph	Browser Visit Count
Format Missing Class Values?	Yes

This form enables you to access the fields in order to refine the report, as well as to modify the basic report definition.

The fields that you can change in the Documents Administrator Report Definition Form are described in the following list:

Group
is the report group to which the report belongs.

Report Label
is the text that is displayed as the name of the report in the Traffic page of the SAS Web Analytics Report Viewer (and in the navigation tree of the Traffic page of the SAS Web Analytics Administrator).

Description
is a short description of the report. By default, the description is the same as the report label.

Include in Report Menu?
determines whether the report appears in the menu of the report group to which it belongs in the Traffic page in the SAS Web Analytics Report Viewer. If you select **Yes**, then the report appears in the menu. If you select **No**, then the report does not appear in the menu.

Help File
consists of the path and filename of the online help for this report. The standard SAS Web Analytics reports include online help. For customized reports, you can provide the location of a corresponding help file (if it exists) in this field.

Sort Field
lists the variables that you selected when you defined the report. If you want the report to be sorted by a specific variable, then select the variable from the drop-down list.

Relative Sort Position
is an integer that corresponds to the desired position of the report in relation to the other reports in the report group as they appear in the SAS Web Analytics Report

Viewer. To move the position of a report higher in the list, change the value of its relative sort position to a number that is smaller than that of the other reports.

Note: For the relative sort position, any non-zero value takes precedence over 0. For example, if you set the relative sort position of an item in a list to 2, and if all of the other items in the list have a default relative sort position of 0, then the item with a relative sort position of 2 is placed first in the list. △

`Default Graph`
specifies which previously defined graph to display by default for this report.

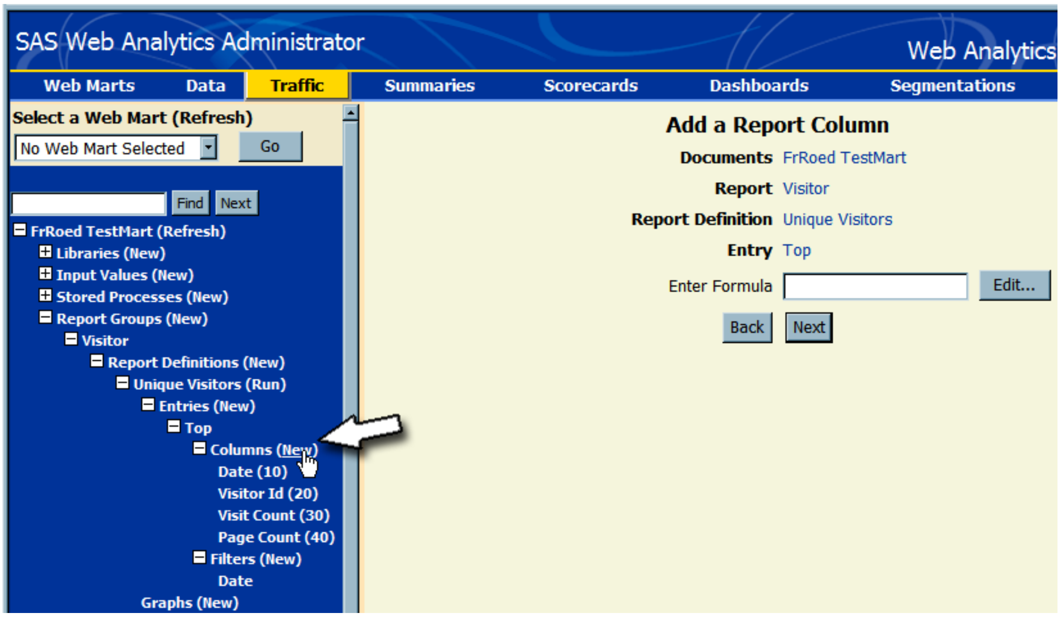
`Format Missing Class Values`
specifies how to display any missing values in the report as follows:

- □ If **`Yes`**, then the application will give placeholders to missing values.
- □ If **`No`**, then the application will display any missing values that are located within a grouping variable as blanks.

Add a New Column

To add a new column to an existing report specification, perform the following steps:

1 In the **`Traffic`** page of the SAS Web Analytics Administrator, expand the navigation tree to view the specific report definition (under a report group) that you want to add a new column to. If necessary, expand the **`Report Definitions`** node, the **`Entries`** node, and the **`Top`** node in order to view the **`Columns (New)`** node. See the following example display that shows the location of the Columns node below the Unique Visitors report specification:



Add a Report Column

Documents FrRoed TestMart
Report Visitor
Report Definition Unique Visitors
Entry Top

2 Click **`New`** next to **`Columns`**. The first page of the Add a Report Column wizard appears on the right (see the previous display).

3 To enter a formula, click **`Edit`** and follow the instructions in "Using an Equation to Calculate a Value in a Column" on page 165. Otherwise, click **`Next`**. As you proceed through the Add a Report Column wizard, provide the entries for fields that are required (indicated by an asterisk (*) in the wizard), and click **`Next`**.

Following are the descriptions of entries for the fields in the Add a Report Column wizard:

`Variable Name`
specifies the variable to be displayed in the column. Select a variable from the drop-down menu.

`Label`
is the name of the column as it will appear in the report. Either accept the default label or type a custom label for the name of the column.

`Is this a classification variable?`
specifies whether the column represents a classification variable. If the column represents a classification variable, then select **`Yes`** from the drop-down menu; otherwise, select **`No`**.

`Percentage`
specifies whether the field is expressed as a percentage. If you select **`Yes`**, then the program calculates the percentage (of the class variable) in relation to the total entries in the class. For example, the last column (%) in the following table would be included in the specified report if the **`Percentage`** field is set to **`Yes`**.

Table 10.1 Example of a Percentage Column for a Class Variable

Browser	Version	Sessions	%
Internet Explorer			
	3	5	5
	4	5	5
	5	10	10
	6	80	80
Netscape			
	5	20	20
	6	30	30
	7	50	50

`Hide this field?`
specifies whether the column appears in the report. If you select **`Yes`**, then the column is hidden in the report.

`Relative column position`
specifies a non-zero integer that corresponds to the desired position of the column in relation to the other columns in the report. For example, if you set the relative column position to *1*, then that column is the first column (starting from the left) of the report. Relative column positions 2, 3, 4, and so on are displayed in ascending numerical order next to relative column position 1.

`Enter Link`
specifies an optional link for the column. If you want the column cell to contain a link to a Web site, specify the URL of the Web site in the text box or create a link that drills to another report. Otherwise, leave this field blank. See “Creating Drillable Reports” on page 185 for an example that shows you how to create a drillable report.

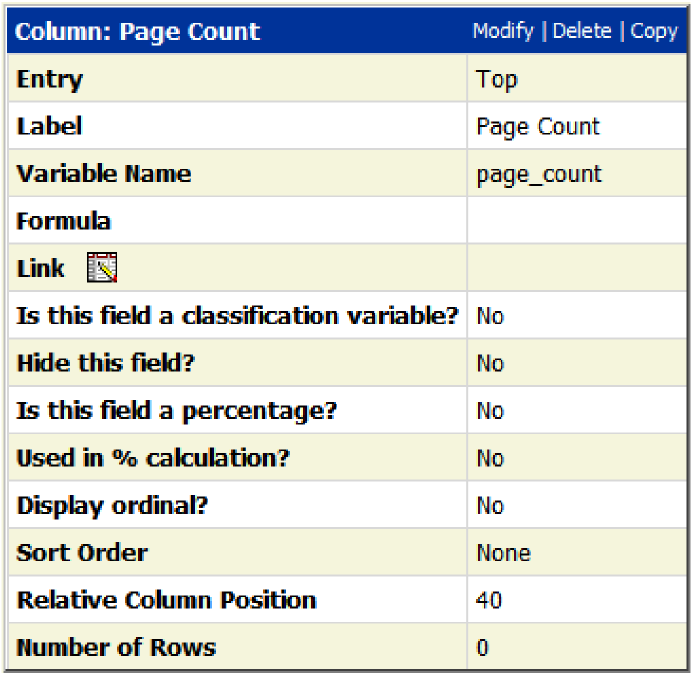
4 Click **Finish** to complete the addition of the new column to the report specification. To see the new column in the navigation tree, click **Refresh** next to the Web mart name in the navigation tree.

Modify the Properties of a Column

To modify the properties of a column in a report, perform the following steps:

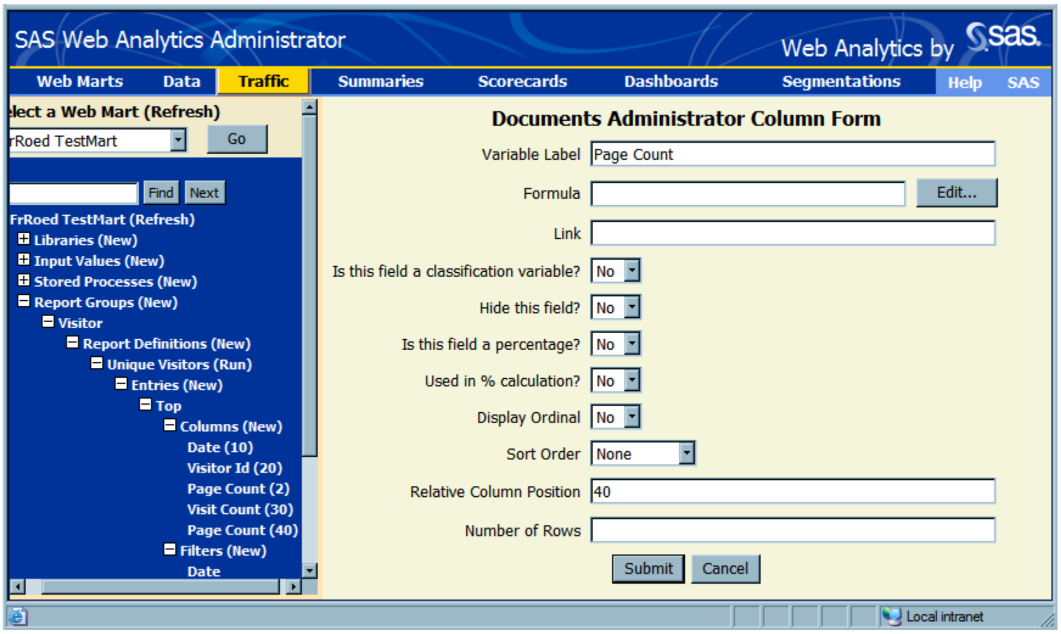
1 In the Traffic page of the SAS Web Analytics Administrator, navigate to the metadata table of a column by clicking the column name below a report in the navigation tree.

In the following example, the metadata table of a hypothetical column called Page Count is displayed:



Column: Page Count	Modify | Delete | Copy
Entry	Top
Label	Page Count
Variable Name	page_count
Formula	
Link	
Is this field a classification variable?	No
Hide this field?	No
Is this field a percentage?	No
Used in % calculation?	No
Display ordinal?	No
Sort Order	None
Relative Column Position	40
Number of Rows	0

2 Modify a property of a column by clicking **Modify** in the title banner of the metadata table. The Documents Administrator Column Form appears in the right panel of the Traffic page (see the following display).



Documents Administrator Column Form

Variable Label	Page Count
Formula	
Link	
Is this field a classification variable?	No
Hide this field?	No
Is this field a percentage?	No
Used in % calculation?	No
Display Ordinal	No
Sort Order	None
Relative Column Position	40
Number of Rows	

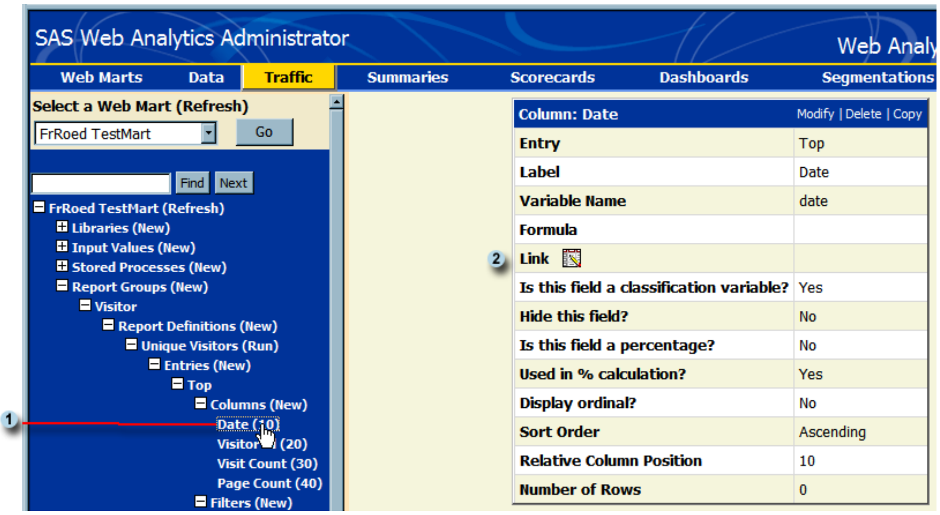
The properties of a column that you can modify are described in the following list:

Label
is the name of the column as it will appear in the report.

Formula
is the equation to use to calculate the value in the column.

Link
is used to optionally specify a link to one of the following destinations:

- □ If you want the field in the column to contain a hyperlink to a Web site, then specify the URL of the Web site in the text box.
- □ If you want the field in the column to contain a hyperlink from a drillable (main) report to another (drilled) report, then use the Set the Column Link wizard to create that link by performing the following steps (see the following display):



Column: Date	
Entry	Top
Label	Date
Variable Name	date
Formula	
Link	
Is this field a classification variable?	Yes
Hide this field?	No
Is this field a percentage?	No
Used in % calculation?	Yes
Display ordinal?	No
Sort Order	Ascending
Relative Column Position	10
Number of Rows	0

a On the Traffic page of the SAS Web Analytics Administrator, click the column name (❶) to open the properties form of the selected column.

b Click the icon next to `Link` (❷) to open the Set the Column Link wizard.

c Follow the instructions in the Set the Column Link wizard.

`Is this field a classification variable?`

specifies whether the column represents a classification variable. If the column represents a classification variable, then select **`Yes`** from the drop-down menu; otherwise, select **`No`**.

`Hide this field?`

specifies whether the column appears in the report. If you select **`Yes`**, then the column is hidden in the report.

`Is this field a percentage?`

specifies whether the field is expressed as a percentage. If you select **`Yes`**, then the program calculates the percentage (of the classification variable) in relation to the total entries in the class. For example, the last column (%) in Table 10.1 on page 155 would be included in the specified report if the **`Percentage`** field is set to **`Yes`**.

`Used in % calculation?`

specifies whether the variable is used for calculating a percentage. Select **`Yes`** if the classification variable in this column is used in order to calculate a percentage (that might be displayed in another column).

`Display ordinal?`

specifies whether values are displayed as ordinals. Select **`Yes`** if you want the report to display values in an ordinal or relative rank to each other on the left side of the graph.

`Sort Order`

`None` specifies that the values in this column do not need to be listed in any particular position relative to each other.

`Ascending` specifies that the values in this column should be displayed in ascending order, with the smallest value at the top of this column.

`Descending` specifies that the values in this column should be displayed in descending order, with the largest value at the top of the column.

`Relative Column Position`

is a non-zero integer that corresponds to the desired position of the column in relation to the other columns in the report. For example, if you set the relative column position to *1*, then that column is the first column (starting from the left) of the report. Relative column positions 2, 3, 4, and so on are displayed in ascending numerical order next to relative column position 1.

`Number of Rows`

specifies the number of rows to display in the report at the same time.

Add a Column from Another Data Set

You can add one or more columns whose values are calculated from a data set that is different from the data set used by the existing table of a report. In the following example, two new columns (Unique Visitors and Repeat Visitors) are added to a report named Site Metrics B. This report currently contains a table entry for a multi-column table named Top.

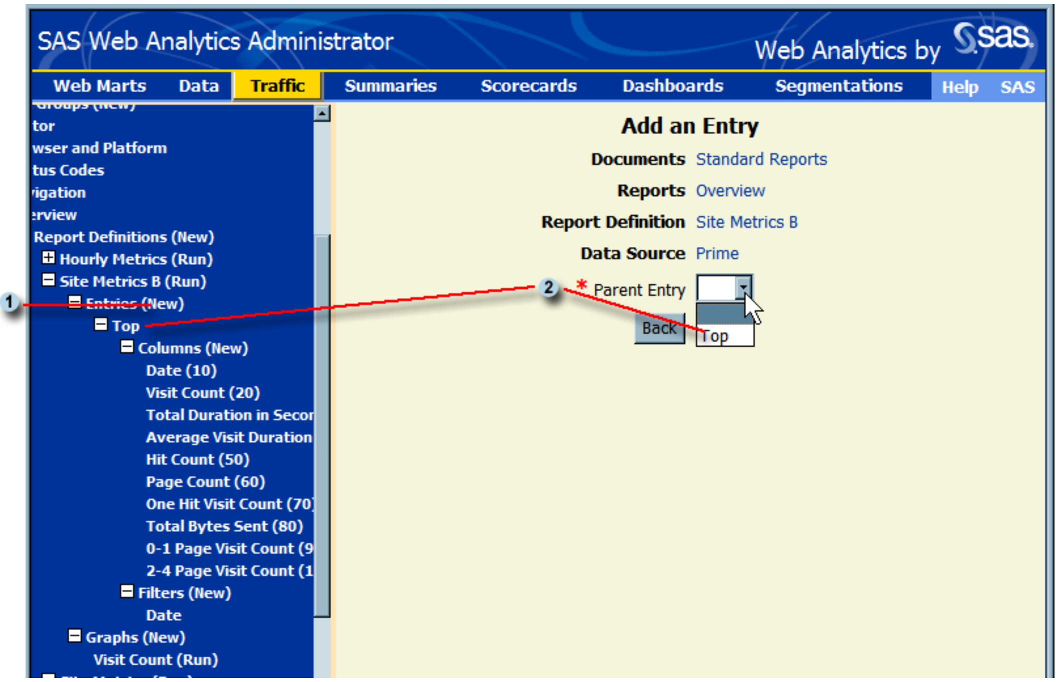
Note: An *entry* for a report definition can be defined as either a table or a stored process. △

If the analysis variables in the new columns come from a new data set, then a new entry must be created that specifies the other data set and defines the new columns. The new entry in the example will be named Child.

To add to a report table one or more columns whose values come from another data set, and to create a new entry for the report, perform the following steps, substituting the appropriate values for the values provided in the example:

1 Navigate to the Traffic page of the SAS Web Analytics Administrator. Expand the appropriate (**`Overview`**) report group , the **`Site Metrics B`** report definition in this example, and the **`Entries`** node (if necessary) so that the **`Top`** node is visible. Click **`New`** (❶) next to **`Entries`** to begin the process of creating an entry called Child for the Site Metrics B report (see the following display). The Add an Entry wizard appears in the right panel of the Traffic page.

2 In the **`Parent Entry`** field, select **`Top`** (❷) from the drop-down menu, and click **`Next`** (see the following display).



Add an Entry

Documents Standard Reports

Reports Overview

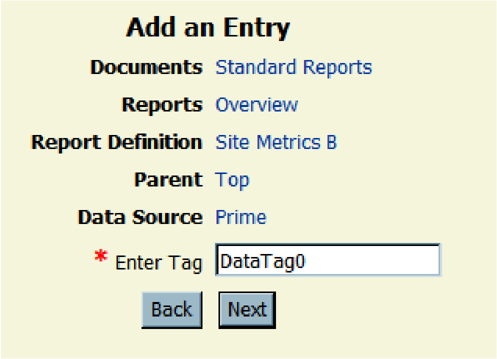
Report Definition Site Metrics B

Data Source Prime

* Parent Entry Top

The **`Top`** entry defines the existing upper table of the Site Metrics report, which is the parent of the new entry that you are creating.

3 Accept the default entry for the **`Enter Tag`** field, and click **`Next`** (see the following display). This field is used internally.



Add an Entry

Documents Standard Reports

Reports Overview

Report Definition Site Metrics B

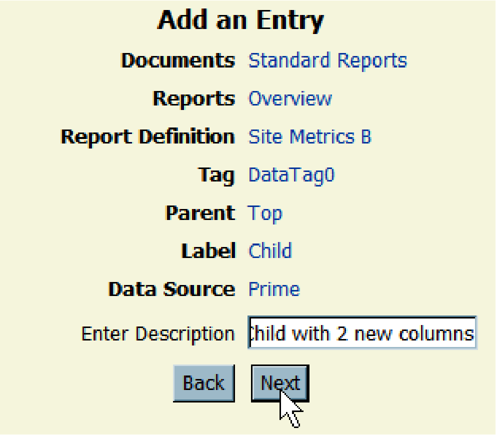
Parent Top

Data Source Prime

* Enter Tag DataTag0

4 Enter *Child* in the **Enter Label** field, and click **Next**. The label will appear as a new node (below any existing entries for the selected report definition) in the navigation tree after the Child entry is submitted.

5 In the **Enter Description** field, enter a description for the child entry, and click **Next**. In the example, *child with 2 new columns* is entered (see the following display).



Add an Entry

Documents Standard Reports

Reports Overview

Report Definition Site Metrics B

Tag DataTag0

Parent Top

Label Child

Data Source Prime

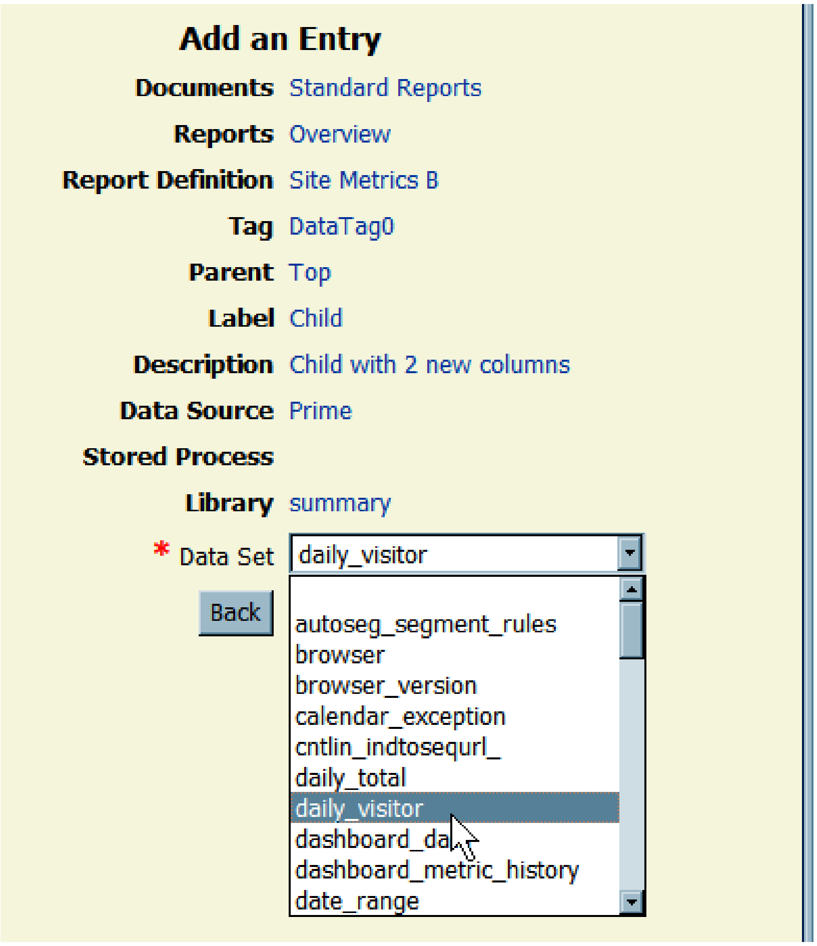
Enter Description child with 2 new columns

6 Leave the **Stored Process** field blank, and click **Next**.

7 Select **Summary** in the drop-down menu next to **Library**, and click **Next**.

Note: To see a list of all libraries that are available, click in the box next to **Show All**. △

8 In the **Data Set** field, select **daily_visitor** from the drop-down menu (see the following display), and click **Next**.



Add an Entry

Documents Standard Reports

Reports Overview

Report Definition Site Metrics B

Tag DataTag0

Parent Top

Label Child

Description Child with 2 new columns

Data Source Prime

Stored Process

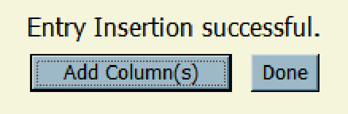
Library summary

* Data Set daily_visitor

The Daily_Visitor_Day data set appears in the **Data Set** field.

9 Click **Finish**.

10 To add the first column to the child entry, click **Add Column(s)** (see the following display).



Entry Insertion successful.

11 In the Add a Report Column wizard, leave the **Enter Formula** field blank, and click **Next**.

12 Select **unique_visitor** from the drop-down menu next to the **Variable** field.

13 Type *Unique Visitors* in the text box next to the **Enter Label** field, and click **Next**.

14 Because Unique Visitor is not a classification variable, verify the default selection of **No** and click **Next**.

15 Select **No** for the value of **Percentage**, (because in the example we do not need to calculate the percentage of unique visitors in this table) and click **Next**.

16 Select **No** to specify that the new column should not be hidden, and click **Next**.

17 Leave the **Relative Column Position** field blank, or enter an integer. The smaller the value for the relative column position, the farther to the left the column is positioned within the table.

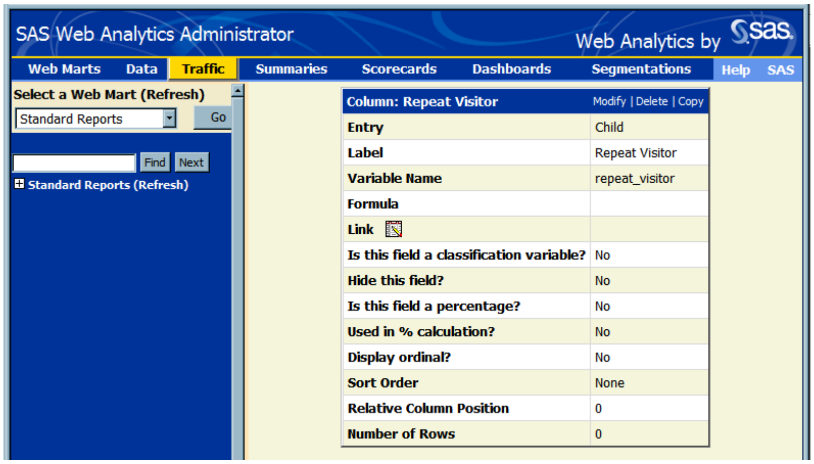
18 Leave the **Enter Link** field blank and click **Next**. Click **Finish** to complete the definition of the first column of the child entry.

19 To add another column that is named Repeat Visitors to the child entry, click **Add Another Column** and repeat the previous steps beginning at step 10, but enter the following values for the following fields:

- □ **Variable Name**: repeat_visitor
- □ **Enter Label**: Repeat Visitors

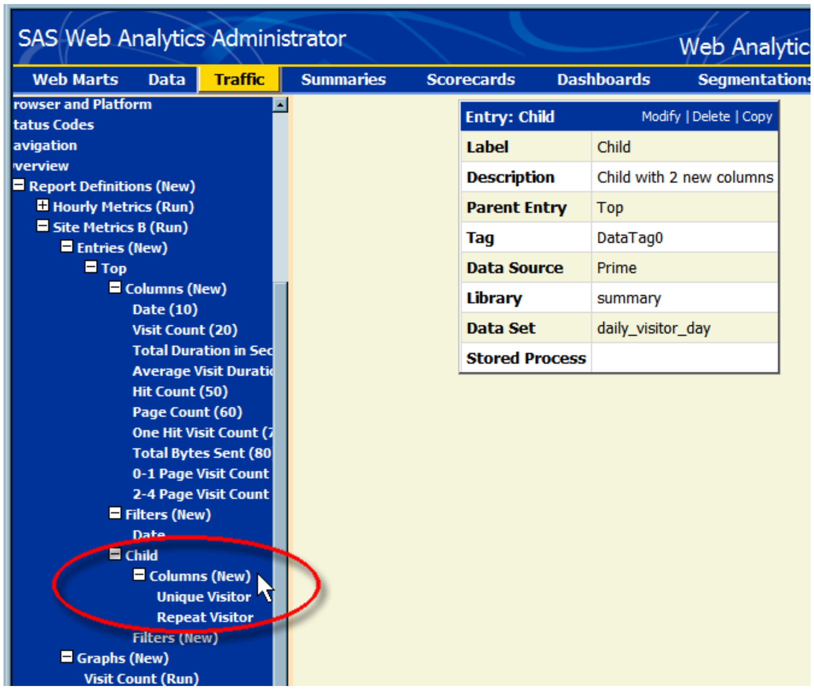
and click **Finish**.

20 This time, below the "Column Insertion successful" message, click **Done**. The Traffic page refreshes, and the metadata table of the Repeat Visitor column appears in the panel on the right (see the following display).



Column: Repeat Visitor	
Entry	Child
Label	Repeat Visitor
Variable Name	repeat_visitor
Formula	
Link	
Is this field a classification variable?	No
Hide this field?	No
Is this field a percentage?	No
Used in % calculation?	No
Display ordinal?	No
Sort Order	None
Relative Column Position	0
Number of Rows	0

21 Expand the navigation tree to verify the existence of the new node (Child) and the two new columns below the Entries node of the Site Metrics report definition (see the following display).



Entry: Child	
Label	Child
Description	Child with 2 new columns
Parent Entry	Top
Tag	DataTag0
Data Source	Prime
Library	summary
Data Set	daily_visitor_day
Stored Process	

22 Run the Site Metrics B report to view the new columns added to the table.

Creating Filters

Overview: Filters

Filters are used by the SAS Web Analytics programs to restrict the display of all of the data (which is available from the SAS data sets) to only the data that is needed for a specific report. By default, all SAS Web Analytics traffic report definitions include a Date filter, so you must also add the following filter to every new traffic report that you create:

```
date between $start,$end
```

For instructions on creating the Date filter for a new report definition, see the following section ("Create the Date Filter" on page 163).

For examples of other types of filters that you can create, see "Filter Example 1: Filter to Specify Two Browser Columns" on page 163 and "Filter Example 2: Filter That Enables the User to Select a Column to Hide" on page 164.

Create the Date Filter

A filter that enables the user to specify a range of dates must exist in the report definition for each SAS Web Analytics report. To create the filter called Date for a new report definition that you have created, perform the following steps:

1 In the navigation tree of the Traffic page of the SAS Web Analytics Administrator, under the report definition you have created, click **New** next to **Filters**. The Add a Filter wizard appears in the right panel.

2 In the **Label** field, type *Dates*. Click **Next**. Additional fields appear.

3 For the **Variable** field, select **Date** from the drop-down menu.

For the **Operator** field, select **Between** from the drop-down menu.

In the **Enter Value** field, type *$start,$end* and click **Next**.

4 Click **Finish**, and then click **Done**.

Because the SAS Web Analytics Report Viewer has a calendar that is used to resolve these variables that are specified by the Date filter for each report, the user is not directly prompted to specify the dates.

Filter Example 1: Filter to Specify Two Browser Columns

To create a filter that specifies the two browsers that you want the report to include, perform the following steps:

1 Navigate to the Traffic page of the SAS Web Analytics Administrator. In the navigation tree, under the report definition that you have created, click **New** next to **Filters**. The Add a Filter wizard appears in the right panel.

2 In the **Label** field, type a name for the filter. Click **Next**. Additional fields appear.

3 For the **Variable** field, select **Browser** from the drop-down menu.

For the **Operator** field, select **In** from the drop-down menu.

In the **Enter Value** field, type *internet explorer,netscape*. Values must be separated only by commas. Values are case sensitive and must exactly match the values within the data. Click **Next**.

4 Click **Finish** and then click **Done**.

5 Run the report. Only the Internet Explorer and the Netscape browsers are listed in the resulting report.

Filter Example 2: Filter That Enables the User to Select a Column to Hide

In this example, the filter is named Hide and requires the user to select the browser that should not be included in the Browsers report. To create the filter, perform the following steps:

1 In the navigation tree, under the appropriate report definition (in this example, the Browsers report definition is selected), click **New** next to **Filters**. The Add a Filter wizard appears in the right panel.

2 In the **Label** field, type a short description. This label is displayed as the prompting text for the filter. The user will use the label to understand the nature of the filter variable that he must select in order to view the resulting report in both the SAS Web Analytics Report Viewer and in the SAS Web Analytics Administrator. Click **Next**. Additional fields appear.

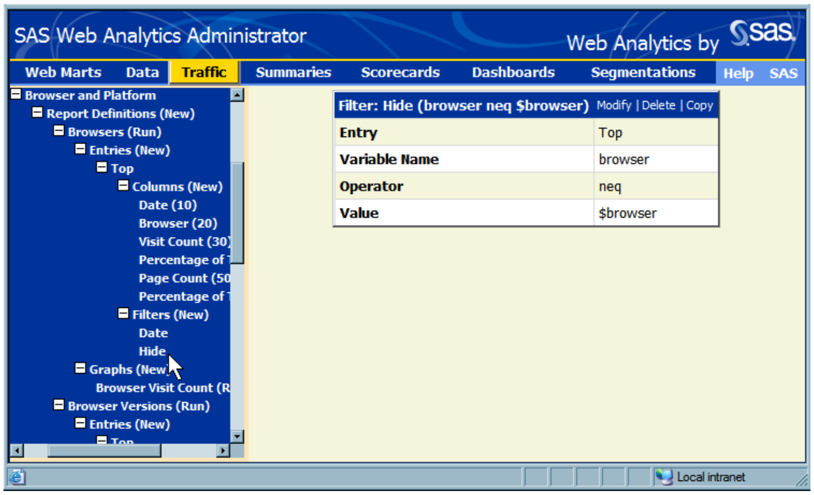
3 For the **Variable** field, select **Browser** from the drop-down menu.

For the **Operator** field, select **Not Equal To** from the drop-down menu.

In the **Enter Value** field, type *$browser*. This populates the drop-down menu with the names of the browsers. Click **Next**.

Note: A variable is indicated by a leading dollar sign ($). If a variable has the same name as an input value, then the user can choose from the list of the input values when he needs to resolve a variable. △

4 Click **Finish** and then click **Done**. The name of the new filter (which is Hide in this example) appears below the **Filters** node of your report definition in the navigation tree (see the following display).

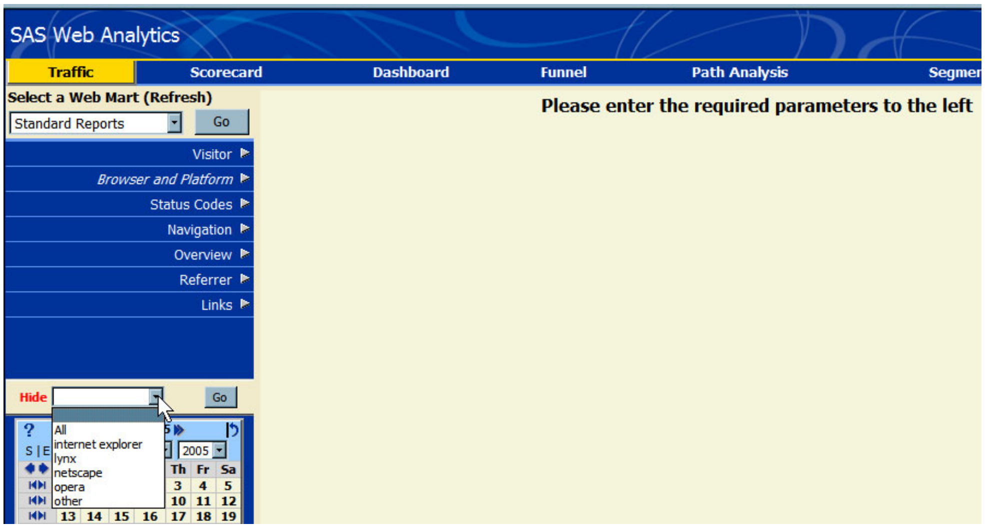


Filter: Hide (browser neq $browser)	Modify | Delete | Copy
Entry	Top
Variable Name	browser
Operator	neq
Value	$browser

5 Run the report from either the SAS Web Analytics Administrator or the SAS Web Analytics Report Viewer.

Note: If multiple filters exist, then all filters are applied to the report. △

Display 10.1 A Filter Named Hide Prompts You to Select a Browser to Hide (in the Browsers Report)



Please enter the required parameters to the left

Using an Equation to Calculate a Value in a Column

Overview: The Equation Editor

The equation editor (see Display 10.2 on page 166) in the SAS Web Analytics Administrator enables you to either specify a numeric equation or use functions that can do any of the following tasks (if appropriate) to calculate a value for a new column:

- □ return true/false indicators
- □ parse character variables
- □ group pages (or visitors) based on specific criteria
- □ specify a calculation to be performed on two separate existing columns within a selected data set

Access the Equation Editor

To open the equation editor, perform the following steps:

1 Select a report definition from the navigation tree on the Traffic page of the SAS Web Analytics Administrator.

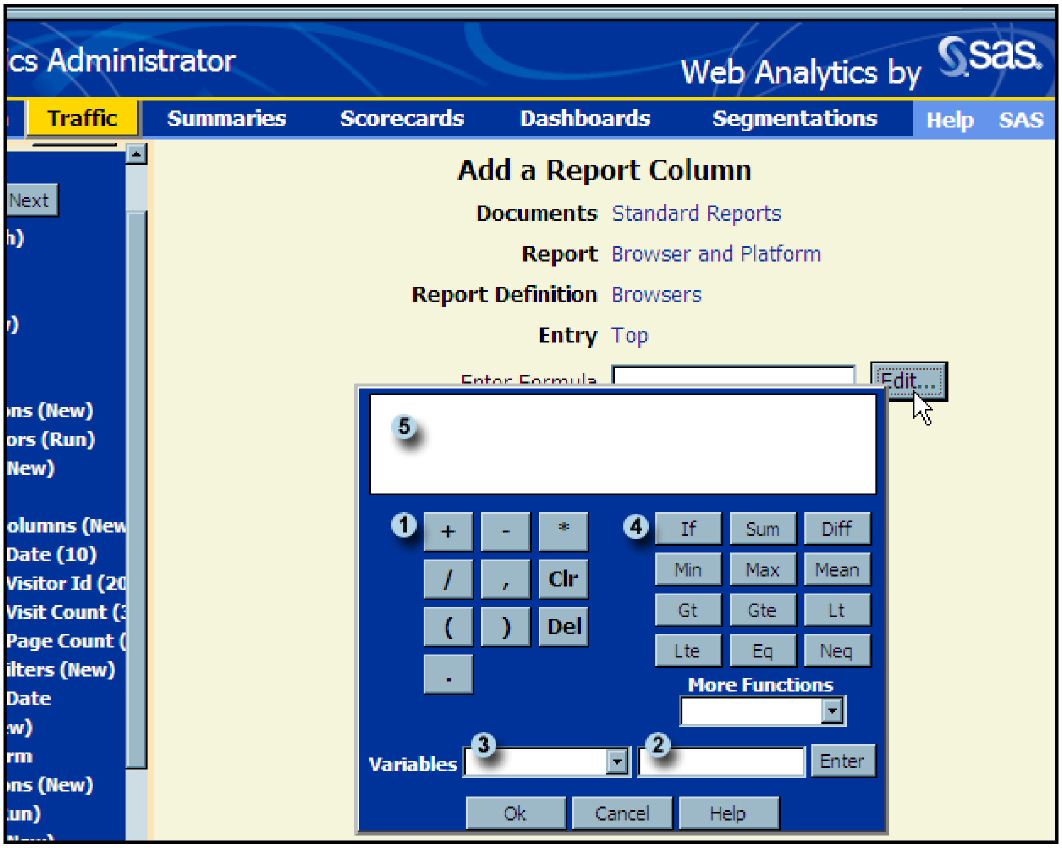
2 Click **`New`** next to **`Columns`**.

3 In the Add a Report Column wizard that appears, click **`Edit`** next to the **`Enter Formula`** text box. The equation editor appears (see Display 10.2 on page 166).

Buttons and Functionality of the Equation Editor

Refer to the display and the legend that follows for a description of the interface of the equation editor.

Display 10.2 The Functional Areas of the Equation Editor



Add a Report Column

Documents Standard Reports

Report Browser and Platform

Report Definition Browsers

Entry Top

Enter Formula

Legend for Display 10.2 on page 166

❶ Arithmetic operators, other equation operators, and the following special keys are included in this group of buttons:

□ **`Clr`** – clears the equation display box (❺)

□ **`Del`** – deletes a single selected item within the equation display box

❷ This text box is used to enter numbers and other equation values. Click **`Enter`** to add the character(s) that you have typed in the text box to the equation.

❸ The **`Variables`** drop-down menu contains all selected variables from the data set that was specified to use within the report. Although you can type a variable name in the text box, you should use the **`Variables`** drop-down menu instead in order to avoid syntax errors when you add any variable names to the equation.

❹ This group of buttons represents the most commonly-used functions. The **`More Functions`** drop-down menu contains additional functions. The functions can be grouped into the following areas:

□ general comparators

□ numeric functions

□ character functions

For a complete list of available functions, see "Available Functions in the Equation Editor" on page 172.

❺ The items in the equation appear in this equation display box as you build the equation by using the buttons and text box in the equation editor.

Ok – closes the equation editor and writes the equation in the **Edit Formula** text box of the Add a Report Column page.

Cancel – closes the equation editor without saving any of the changes that you made.

Help – opens the online Help for the equation editor.

Create an Equation

In general, to create an equation in the equation editor, use the equation editor keys in the equation editor interface instead of using the keyboard. Build the parts of the equation by inserting each part (operator, function, variable, or character string) at the insertion point as follows:

1 Click a function or operator button.

2 Either select a variable or enter a character (a number or a text string) in the text box (❷ in the previous display). The selected part(s) appear in the equation display box (❺ in the previous display).

3 Verify the new location of the insertion point in the equation display box. If necessary, relocate the insertion point by clicking the space in the equation (in the equation display box) where you want to insert the next part of the equation.

Note: Be sure that you specify a blank space as your *insertion point* instead of highlighting an existing character (that is not a comma or a closing parenthesis). Any highlighted characters (except a comma or a closing parenthesis) are replaced by the next character or string that is entered. △

A box appears in the blank space where you clicked that indicates the insertion point for the next part.

4 Select the next appropriate variable or other part(s) of the equation. The new part is inserted into the equation at the insertion point, and the insertion point is automatically moved. Again, check the location of the new insertion point, re-locating it if necessary, and continue to add each new part into the equation.

5 When you are finished, click **Ok** to close the equation editor. The equation is displayed in the **Enter Formula** text box of the Add a Report Column wizard.

To delete a character, variable, or operator in an equation that you are creating, select the item in the equation display box by clicking it, and then click **Del** in the equation editor interface.

Example: Creating a Simple Equation

This procedure demonstrates how you would use the equation editor to create the following equation:

```
(page_count/session_count)*100
```

1 Click the **(** button. An opening parenthesis appears in the equation display box.

2 Click the **Variables** drop-down menu to expand the list. Select **Page Count** from the list. **{page_count}** appears in the equation display box.

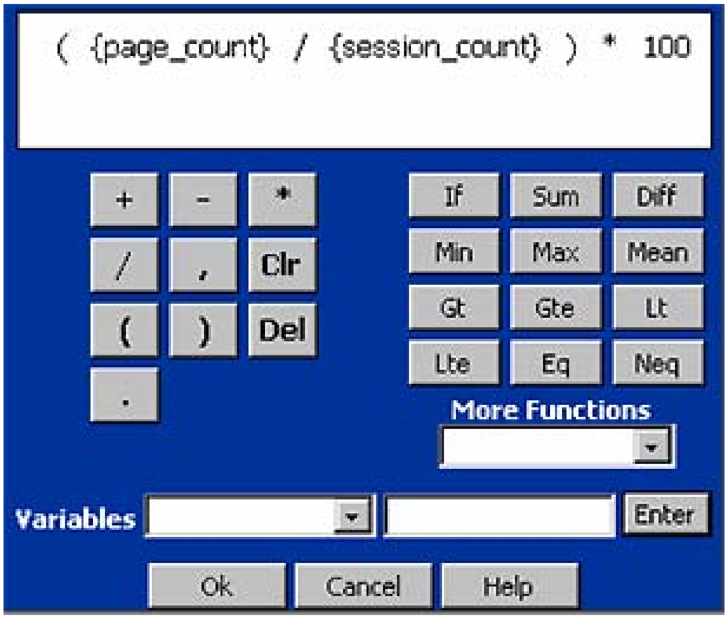
3 Click the **/** button. A forward slash (/) appears in the equation display box.

4 Click the **Variables** drop-down menu and select **Session Count** from the list. **{session_count}** appears in the equation display box.

5 Click the * button. An asterisk (*) appears in the equation display box.

6 In the text entry box, type *100* and click **Enter**. The completed sample equation should look like the equation in the following display:

Display 10.3 A Sample Equation in the Equation Display Box of the Equation Editor



({page_count} / {session_count}) * 100

7 Click **Ok** to close the equation editor and to return to the Add a Report Column wizard.

Using Functions in the Equation Editor

How Functions Are Used

In addition to creating equations (in order to calculate a value in a column), you can also use the equation editor to build functions. Functions can be used to create flags, summarize columns, or parse the character columns for classification information.

The following stepwise examples illustrate how to build numeric and character functions.

Example 1: Using the SUM Function to Calculate Total Errors

In this example, you will use the SUM function to calculate total errors. The results of the function are displayed in a new column of the report when the report is run.

At this point, you have selected a report, specified a new column for the report, and opened the equation editor (see “Access the Equation Editor” on page 165). Follow these steps to continue with the example:

1 In the equation editor, click the **SUM** button. The empty syntax string for the SUM function appears in the equation display box.

Note: The syntax for the SUM function is

```
sum(expr0,expr1,...,exprN)
```

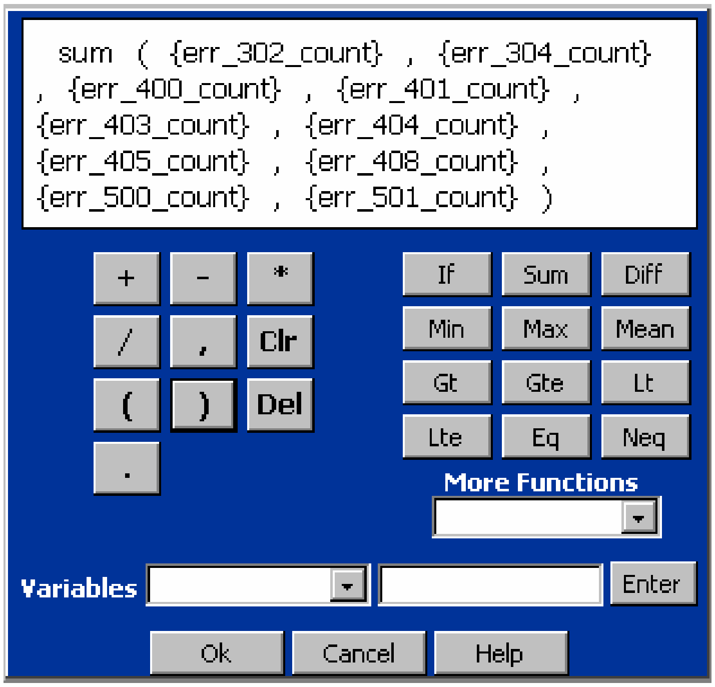
where expr = a variable. △

2 Click the **Variables** drop-down menu and select **ERROR 302** from the list. **{err_302_count}** appears next to the opening parenthesis in the equation display box. The insertion point moves to the comma in the equation display box. Leave the insertion point at the comma. (The next entry will be placed after any comma when it is highlighted as an insertion point.)

3 To add the next variable, select it from the list. The variable appears in the equation display box, and the insertion point moves to the closing parenthesis. In this example, eight variables are added to the function. After the first two variables are added, you must add a comma between each variable to separate them. To do this you must re-position the insertion point to the space between the last variable and the closing parenthesis by clicking on the space. A box indicates the insertion point. Then, continue by selecting the next variable.

Note: When a character is selected (highlighted) in the equation display box and a new character (except an opening parenthesis or a comma) is added, the selected character is replaced. △

The final equation will look like the example equation in the following display:



```
sum ( {err_302_count} , {err_304_count}
, {err_400_count} , {err_401_count} ,
{err_403_count} , {err_404_count} ,
{err_405_count} , {err_408_count} ,
{err_500_count} , {err_501_count} )
```

4 Click **Ok** to close the equation editor.

Example 2: Using the PROPERTY Function to Select the KEYWORDS Parameter Value from the Query String

In this example, you want to define a new column that lists the keywords that visitors used in order to search a Web site, and the number of bytes that were sent for each keyword search. Each keyword is a classification variable for which the number of bytes sent is summarized. The results of the calculation are displayed in a new column of the report when the report is run.

To define the new Keywords column, you will use the PROPERTY function to select the parameter value for KEYWORDS from the query string.

At this point, you have selected a report, specified a new column for the report, and opened the equation editor (see "Access the Equation Editor" on page 165). Follow these steps to continue with the example:

1 Select PROPERTY from the **More Functions** drop-down menu. The empty syntax string for the PROPERTY function appears in the equation display box.

Note: The syntax for the PROPERTY function is

```
property(expr,outdelim,indelim,name)
```

where

expr = a variable

outdelim = a delimiter that determines a name-value pair. In the query string, the delimiter is an ampersand (&).

indelim = a delimiter that divides the name-value pair into both the name and the value. In this example, the delimiter is the equals sign (=).

name = the value of the name in the name-value pair. In this example, the value for the name (CGI parameter) is *keywords*.

△

2 Click the **Variables** drop-down menu and select **QUERY_STRING**. **{query_string}** appears to the right of the opening parenthesis, and the cursor moves to the first comma in the equation display box.

3 In the text box, type an ampersand (&) and click **Enter** to add the ampersand on the right of the first comma in the equation display box. The cursor automatically moves to the second comma.

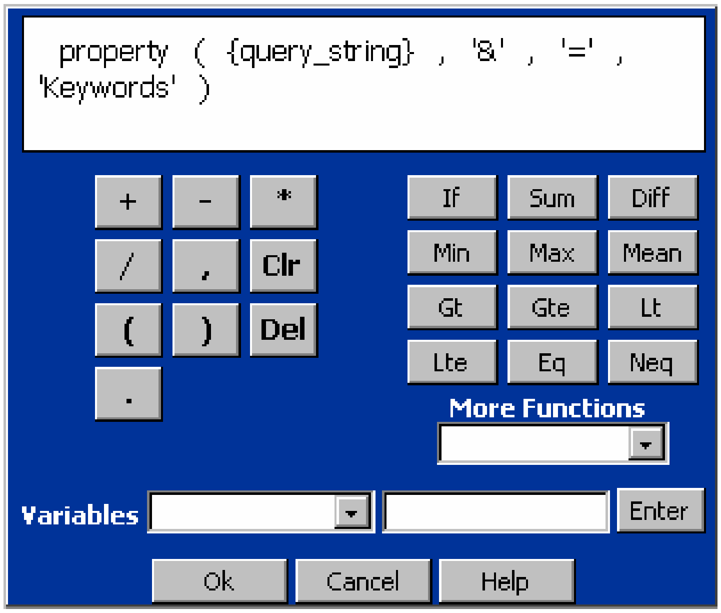
4 In the text box, type an equals sign (=) and click **Enter** to add an equals sign on the right of the second comma in the equation display box. The cursor automatically moves to the third comma.

5 In the text box, type *Keywords*.

Note: Be sure to verify the spelling and capitalization of any values for the options (such as “Keywords” in this example) because function syntax is case sensitive. △

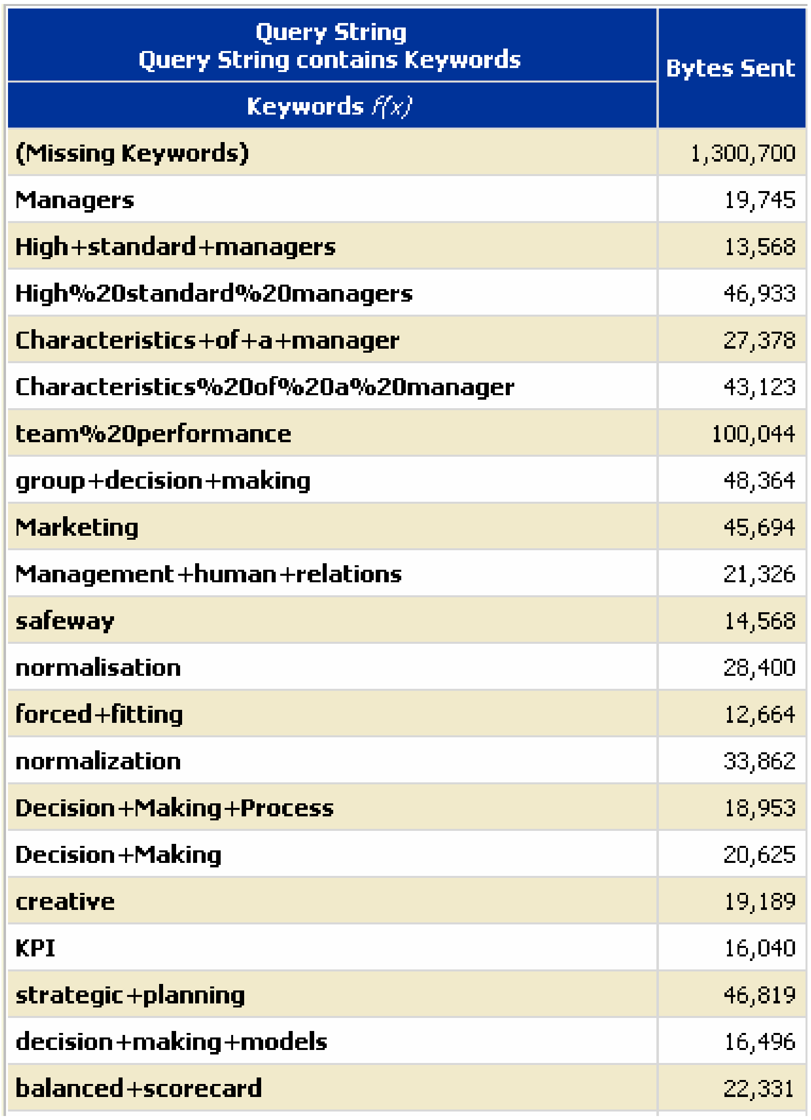
6 If necessary, delete any extra commas between the opening and closing parentheses of the function.

The final equation in this example will look like the equation in the following display:



property ({query_string} , '&' , '=' ,
'Keywords')

The resulting report (see the following example) contains the **Keywords** column as specified by the PROPERTY function. The values in the **Bytes Sent** column are specified by the report definition.



Query String Query String contains Keywords	Bytes Sent
Keywords *f(x)*	
(Missing Keywords)	1,300,700
Managers	19,745
High+standard+managers	13,568
High%20standard%20managers	46,933
Characteristics+of+a+manager	27,378
Characteristics%20of%20a%20manager	43,123
team%20performance	100,044
group+decision+making	48,364
Marketing	45,694
Management+human+relations	21,326
safeway	14,568
normalisation	28,400
forced+fitting	12,664
normalization	33,862
Decision+Making+Process	18,953
Decision+Making	20,625
creative	19,189
KPI	16,040
strategic+planning	46,819
decision+making+models	16,496
balanced+scorecard	22,331

Available Functions in the Equation Editor

The following table describes all of the functions that can be used in the equation editor to create a new column:

Note: Expr = expression. △

Table 10.2 Functions That are Available in the Equation Editor

Function	**Type**	**Syntax**
		Result
General Comparators		
And	Boolean	**`and(expr0,expr1)`** If both expr0 and expr1 are true, then the returned value is true. Otherwise, the returned value is false.
Equals	Boolean	**`equals(expr0,expr1)`** The returned value is true if expr0 is equal to expr1. Otherwise, the returned value is false. *Note:* The type of the comparison is specified by the type of expression that expr0 describes.△
False	Boolean	**`false(expr)`** The returned value is true if the Boolean value of the expr is false. Otherwise, the returned value is false.
If	Variable	**`if(expr0,expr1,expr2)`** If the Boolean value of expr0 is true, then the returned value is expr1. Otherwise, the returned value is expr2.
Nequals	Boolean	**`nequals(expr0,expr1)`** The returned value is true if expr0 is not equal to expr1 (the type of the comparison depends on the type of expr0). Otherwise, the returned value is false.
Not	Boolean	**`not(expr)`** The returned value is true if the Boolean value of the expr is false. Otherwise, the returned value is false.
True	Boolean	**`true(expr)`** The returned value is true if the Boolean value of the expr is true. Otherwise, the returned value is false.
Numeric Functions		
Between	Boolean	**`between(expr0,expr1,expr2)`** If expr0 is greater than expr1 and less than expr2, then the returned value is true. Otherwise, the returned value is false.
Ceiling	Integer	**`ceiling(expr)`** Returns the integer value that is above the numeric value of expr.
Diff	Float	**`diff(expr0,expr1)`** The returned value is expr0 – expr1.
Floor	Integer	**`floor(expr)`** Returns the integer value that is below the numeric value of expr.

Function	Type	Syntax Result
Gt	Boolean	**`gt(expr0,expr1)`** The returned value is true if expr0 is greater than expr1. Otherwise, the returned value is false. *Note:* The type of the comparison is specified by the type of expression that expr0 describes.△
Gte	Boolean	**`gte(expr0,expr1)`** The returned value is true if expr0 is greater than or equal to expr1. Otherwise, the returned value is false. *Note:* The type of the comparison is specified by the type of expression that expr0 describes.△
Lt	Boolean	**`lt(expr0,expr1)`** The returned value is true if expr0 is less than expr1. Otherwise, the returned value is false. *Note:* The type of the comparison is specified by the type of expression that expr0 describes.△
Lte	Boolean	**`lte(expr0,expr1)`** The returned value is true if expr0 is less than or equal to expr1. Otherwise, the returned value is false. *Note:* The type of the comparison is specified by the type of expression that expr0 describes.△
Max	Float	**`max(expr0,expr1,...,exprN)`** The returned value is the largest numeric value of all of the expressions.
Mean	Float	**`mean(expr0,expr1,...,exprN)`** The returned value is the mean value of all expressions.
Min	Float	**`min(expr0,expr1,...,exprN)`** The returned value is the smallest numeric value of all expressions.
Mod	Float	**`mod(expr0,expr1)`** The returned value is the remainder of expr0 / expr1.
Number	Float	**`number(expr)`** The returned value is the numeric value of expr.
Round	Integer	**`round(expr)`** The returned value is the rounded numeric value of expr.

Function	**Type**	**Syntax** **Result**
Sum	Float	**`sum(expr0,expr1,...,exprN)`** The returned value is the sum of the numeric values of all expressions.
Character Functions		
Boolean	Boolean	**`boolean(expr)`** If the expression is a string, then the returned value is true if the string has length greater than 0. If the expression is numeric, then the returned value is true if the value is not equal to 0. If the expression is Boolean, the returned value is true if the value is true. Otherwise, the returned value is false.
Concat	String	**`concat(expr0,expr1,...,exprn)`** The returned value is a string that is the concatenation of the individual string values of the expressions.
ConcatDelim	String	**`concatdelim(dlm,expr0,expr1,...,exprn)`** The returned value is a string that is the concatenation of the string values of the expressions separated by the delimiter.
Contains	Boolean	**`contains(expr0,expr1)`** If the string value of expr0 contains the string value of expr1, then the returned value is true. Otherwise, the returned value is false.
Index	Variable	**`index(expr0,expr1,expr2,...,exprn)`** The returned value is the expression at the index of expr0.
IndexOf	Integer	**`indexof(expr0,expr1)`** The returned value is the index of the string value of expr1 within the string value of expr0. If the string value of either expression is not found, then the returned value is -1 .
Interval	Variable	**`interval(expr,b0,out0,b1,out1,...,bN,outN,above)`** Compares the numeric value of the expression to a range boundary (values(bN)) and returns the expression (outN) according to where the numeric value lies within the range.
LastIndexOf	Integer	**`lastindexof(expr0,expr1)`** The returned value is the last index of the string value of expr1 within the string value of expr0. If the string value of either expression is not found, then the returned value is -1 .
LastToken	String	**`lasttoken(expr0,expr1,index)`** Tokenizes expr0 by expr1 and then returns the token at the index from the last token. For example, LastToken(www.sas.com,.,12589) means that the expression "www.sas.com" that is tokenized by "." is "www", "sas", "com".

Function	Type	Syntax Result
Matches	Boolean	`matches(expr0,expr1)` If expr0 matches expr1 (using regular expression matching), then the returned value is true. Otherwise, the returned value is false.
NormalizeSpace	String	`normalizespace(expr)` The returned value is a string that is created by replacing any extra white-space characters in the expression with a single space.
Outstr	String	`outstr(expr,s0,out0,s1,out1,...,sN,outN,default)` If the expression contains a string or one of a group of strings(sN), then the returned value is the associated value (outN). If no matches exist, then the returned value is the default.
Properties	String	`properties(expr,outdelim,indelim,delim,name1,...,nameN)` Creates a set of name-value pairs from the expression by using the outer delimiter (outdelim) and the inner delimiter (indelim), and then extracts the values for each name (1 through N).
Property	String	`property(expr,outdelim,indelim,name)` Creates a set of name-value pairs from the expression by using the outer delimiter (outdelim) and the inner delimiter (indelim), and then extracts the value for the name.
Replace	String	`replace(expr0,expr1,expr2)` The returned value is the string created by performing a regular expression replacement of the first occurence of expr1 with expr2 in expr0.
ReplaceAll	String	`replaceall(expr0,expr1,expr2)` The returned value is the string that is created by performing a regular expression replacement of all occurences of expr1 with expr2 in expr0.
StartsWith	Boolean	`startswith(expr0,expr1)` The returned value is true if the string value of expr0 starts with the string value of expr1. Otherwise, the returned valus is false.
String	String	`string(expr)` The returned value is the string value of the expression.
StringLength	Integer	`stringlength(expr)` The returned value is the length of the string of the expression.
Strip	String	`strip(expr)` The returned value is the string that is created by removing any leading and trailing white space(s) from the expression.

Function	Type	Syntax Result
Substring	String	**`substring(expr,index,[length])`** The returned value is the string that is created by taking the substring of the string value expresion at the index. If a length is specified, then the substring will be that length; otherwise, the returned string will contain all of the characters to the end of the string.
SubstringAfter	String	**`substringafter(expr0,expr1,[index])`** The returned value is the string that is created by taking the content of the string value of the expression after an occurence of expr1. If the index is specified, then the content after the occurence identified by the index is returned.
SubstringBefore	String	**`substringafter(expr0,expr1,[index])`** The returned value is the string that is created by taking the content of the string value of the expression after an occurence of expr1. If an index is specified, then the content after the occurence identified by the index is returned.
SubstringBefore	String	**`substringbefore(expr0,expr1,[index])`** The returned value is the string that is created by taking the content of the string value of the expression before an occurence of expr1. If an index is specified, then the content before the occurence identified by the index is returned.
Token	String	**`token(expr0,expr1,index)`** Tokenizes expr0 by expr1, then returns the token at index from the first token.
ToLower	String	**`tolower(expr)`** The returned value is the string value of the expression converted to all lowercased letters.
ToUpper	String	**`toupper(expr)`** The returned value is the string value of the expression converted to all uppercased letters.

Examples of Functions That You Can Create

The following table contains examples of the functions from Table 10.2 on page 173:

Table 10.3 Examples of Functions

Function	Example(s)
And	and(2,3) = true and(2,3,gt(9,7),0) = false
Between	between(21,3,sum(20,5)) = true between(21,3,sum(10,5)) = false

Function	Example(s)
Boolean	boolean(sum(5,5)) = true boolean(sum(5,-5)) = false boolean(contains(‘www.sas.com’, ‘sas’) = true
ceiling	ceiling(4*4.1) = 17
Concat	concat(‘Name: ’, ‘Joe, Age: ’,floor(sum(25,10)),‘.’) = Name: Joe, Age: 35.
ConcatDelim	concatdelim(‘.’,‘www’,‘sas’,‘com’) = www.sas.com
Contains	contains(‘www.google.com’,‘sas’) = true contains(‘www.google.com’,‘gaggle’) = false
Diff	diff(sum(8,7),22) = -7.0
Equals	equals(sum(10,3),13) = true equals(‘internet explorer’,‘internet explorer’) = true equals(‘internet explorer’,‘netscape’) = false
False	false(gt(5,4)) = false false(gt(4,5)) = true
Floor	floor(3.5 * 7) = 24
Gt	gt(sum(10,4),13) = true gt(‘internet explorer’,‘internet explorer’) = false gt(‘internet explorer’,‘netscape’) = false
Gte	gte(sum(10,3),13) = true gte(‘internet explorer’,‘internet explorer’) = true gte(‘internet explorer’,‘netscape’) = false

If	if(contains(‘microsoft browser’,‘microsoft’),‘internet explorer’,‘mozilla’)
	if(gte(70,80),‘Good Score’,‘Bad Score’)

Index	index(4,‘jan’,‘feb’,‘mar’,‘apr’,‘may’,‘jun’,‘jul’,‘aug’,‘sep’,‘oct’,‘nov’,‘dec’) = may
IndexOf	indexof(‘www.google.com’,‘google’) = 4
Interval	interval(sum(33,22),0,‘Terrible’,33,‘Bottom Third’,66,‘Middle Third’,‘Great’) = Middle Third interval(sum(33,46),0,‘Terrible’,33,‘Bottom Third’,66,‘Middle Third’,‘Great’) = Great
LastIndexOf	lastindexof(‘http://www.sas.com/products/webanalytics’,‘/’) = 27
LastToken	lasttoken(‘http://www.sas.com/products/webanalytics’,‘/’,0) = webanalytics lasttoken(‘http://www.sas.com/products/webanalytics’,‘/’,1) = products lasttoken(‘http://www.sas.com/products/webanalytics/overview.html’,‘.’,0) = html
Lt	lt(sum(10,4),13) = false lt(‘internet explorer’,‘internet explorer’) = true lt(‘internet explorer’,‘netscape’) = true
Lte	lte(sum(10,3),13) = true lte(‘internet explorer’,‘internet explorer’) = true lte(‘internet explorer’,‘netscape’) = true

Function	Example(s)
Matches	matches(‘.*\.html’,‘overview.html’) = true
	matches(‘.*\.html’,‘entrypage.jsp’) = false
Max	max(33,44.2,sum(1,3,2,12),-33.21) = 44.2
Mean	mean(33,44.2,sum(1,3,2,12),-33.21) = 15.4975
Min	min(33,44.2,sum(1,3,2,12),-33.21) = -33.21
Mod	mod(10,3) = 1.0
Nequals	nequals(sum(10,3),13) = false
	nequals(‘internet explorer’,‘internet explorer’) = false
	nequals(‘internet explorer’,‘netscape’) = true
NormalizeSpace	normalizespace(‘Sentence with many spaces’) = Sentence with many spaces
Not	not(sum(10,10)) = false
	not(diff(10,10)) = true
Number	number(‘453’) = 453.0
Outstr	outstr(‘government spending’,‘govern’,‘Government’,‘Other’) = Government
	outstr(‘spending’,(‘govern’,‘spend’),‘Government or Spending’,‘Other’) = Government or Spending
	outstr(‘john’,(‘govern’,‘spend’),‘Government or Spending’,(‘john’,‘jack’,‘bob’),‘Names’,‘Other’) = Names
	outstr(‘web analytics’,(‘govern’,‘spend’),‘Government or Spending’,(‘john’,‘jack’,‘bob’),‘Names’,‘Other’) = Other
Properties	properties(‘first=John,last=Doe’,‘,’,‘=’,‘:’,‘first’,‘last’) = John:Doe , search query string for example
Property	property(‘first:John;last:Doe;occupation:plumber’,‘;’,‘:’,‘occupation’) = plumber search query string for example
Replace	replace(‘http://www.google.com,google.jsp,google.asp’,‘google.*’, ‘google.net’) = http://www.google.net
ReplaceAll	replaceall(‘google.com,google.jsp,google.asp’, ‘google’,‘gaggle’) = gaggle.com,gaggle.jsp,gaggle.asp
Round	round(34.56) = 35
	round(34.46) = 34
StartsWith	startswith(‘www.sas.com’,‘www.’) = true
	startswith(‘www.sas.com’,‘sww.’) = false
String	string(33) = 33.0
StringLength	stringlength(‘SAS Web Analytics’) = 17
Strip	strip(‘ SAS Web Analytics ’)= SAS Web Analytics
Substring	substring(‘www.bettermanagement.com’,4,6) = better
SubstringAfter	substringafter(‘www.bettermanagement.com’,‘.’) = bettermanagement.com
	substringafter(‘www.bettermanagement.com’,‘.’,2) = com

Function	Example(s)
SubstringBefore	substringbefore(‘www.bettermanagement.com’,‘.’) = www substringbefore(‘www.bettermanagement.com’,‘.’,2) = www.bettermanagement
Sum	sum(33,22,55.4,34,min(0,4,-9)) = 135.4
Token	token(‘http://www.sas.com/products/webanalytics’,‘/’,0) = http: token(‘http://www.sas.com/products/webanalytics’,‘/’,1) = www.sas.com token(‘http://www.sas.com/products/webanalytics/overview.html’,‘.’,0) = http://www
ToLower	tolower(‘Http://Www.Sas.Com/Products/Webanalytics’) = http://www.sas.com/products/webanalytics
ToUpper	toupper(‘Http://Www.Sas.Com/Products/Webanalytics’) = HTTP://WWW.SAS.COM/PRODUCTS/WEBANALYTICS
True	true(gt(5,4)) = true true(gt(4,5)) = false

Implementing a Customized Drop-Down Menu in a Report

Overview: Creating a Customized Drop-Down Menu

It is often convenient for report users to select from a list of values rather than to type a value in a text box. For any appropriate report, you can implement a drop-down menu that contains a list of custom values for users to select from.

To create a drop-down list of values for a report, perform the following steps:

1 Either create or identify a SAS data set that contains the following components:

   □ the list of values to be included in the drop-down list

   □ a variable name for this list of values

   Note: You do not need to create a new data set if you can identify an existing data set that contains the list of values to be included in the drop-down list. For example, the Summary.Wadtsort data set that is provided with the SAS Web Analytics program contains a list of the days of the week. △

2 Define an input value by specifying the variable name that identifies the column in the data set that contains the values for the drop-down list.

3 Add a filter to the report that uses this input value (see “Creating Filters” on page 163).

Creating a SAS Data Set with Values for a Drop-Down Menu

In the following example, the PROC SORT step sorts the Summary.Catalog_Summaries_Day data set by Catalog_Name, and writes the sorted rows to the Summary.Unique_Catalogs data set. The NODUPKEY option causes one row per unique value of Catalog_Name to be written to the output data set.

```
proc sort data=summary.catalog_summaries_day
      out=summary.unique_catalogs(keep=catalog_name)
          nodupkey;
      by catalog_name;
   run;
```

Substitute the name of your output data set for Summary.Unique_Catalogs, and substitute the Catalog_Name variable with the variable that you want to sort by.

Defining an Input Value

After the data set that contains the values for the drop-down list has been identified or created, the input value can be defined. In this step, you specify the values that you want to display in the drop-down list by using a variable from a specified data set (for example, the newly created data set in the previous section).

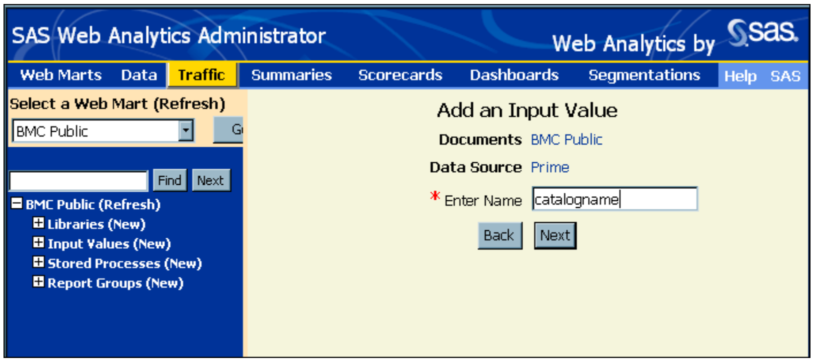
To define the input value to be the variable that represents the values in the drop-down list, perform the following steps:

1 Verify that you are on the Traffic page of the SAS Web Analytics Administrator (the Traffic link on the menu bar is yellow).

2 Select a Web mart (if necessary). If necessary, expand the Web mart node.

3 Next to **Input Values**, select **New**. The Add an Input Value wizard opens. At each prompt, type the information in the text box (if required) as described below, and click **Next**. An asterisk (*) next to a field indicates that you must provide information for that field in order to continue.

Name
is a label that identifies the list of values that you want to display in the drop-down menu. This text string does not have to match the variable name of the variable in the data set that contains the values (for example, the sample data set in "Creating a SAS Data Set with Values for a Drop-Down Menu" on page 180).



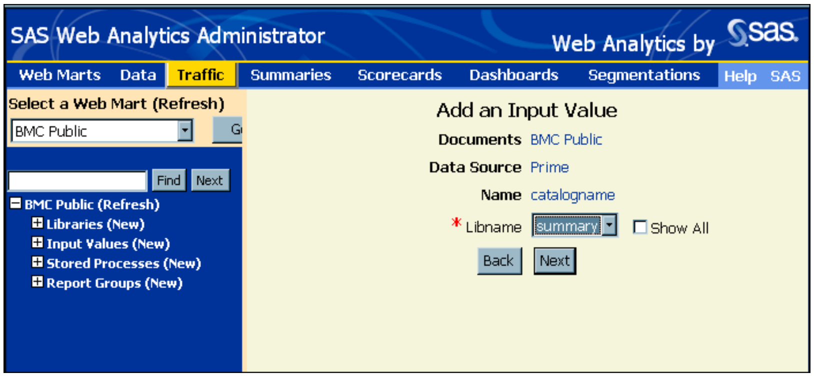
Add an Input Value

Documents BMC Public

Data Source Prime

* Enter Name catalogname

Libname
is the library that contains the data set from which the values in the drop-down list are derived. In the example (see the following display), the Summary library contains the Unique_Catalogs data set; therefore, Summary is selected from the drop-down list.



Add an Input Value

Documents BMC Public

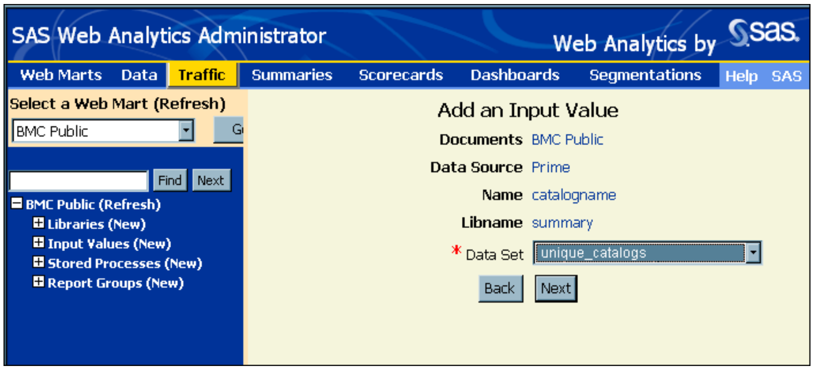
Data Source Prime

Name catalogname

* Libname summary □ Show All

Data Set

is the data set that contains the values to display in the drop-down menu. In the example, Unique_Catalogs is the data set that was created for this purpose (see the following display).



Add an Input Value

Documents BMC Public

Data Source Prime

Name catalogname

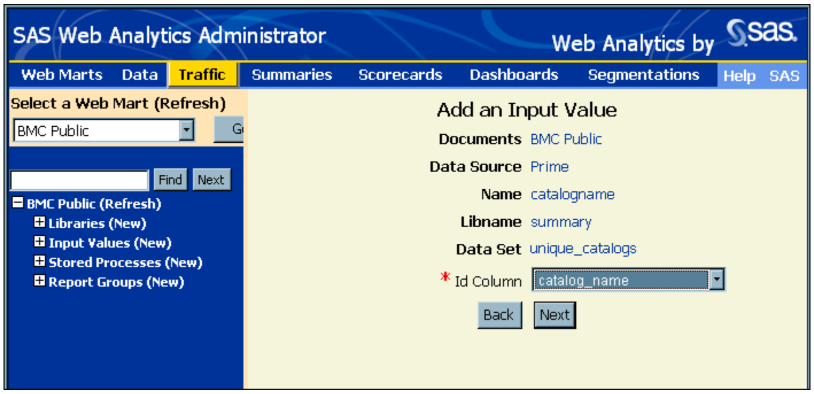
Libname summary

* Data Set unique_catalogs

Id Column

is the variable name that represents the column (in the data set that you specified in the **`Dataset`** field) that contains the list of values to display in the drop-down menu. In the example, the Unique_Catalogs data set contains only one column, Catalog_Name (see the following display).

Display 10.4 A Value for the ID Column Is Selected in the Add an Input Value Wizard



Add an Input Value

Documents BMC Public

Data Source Prime

Name catalogname

Libname summary

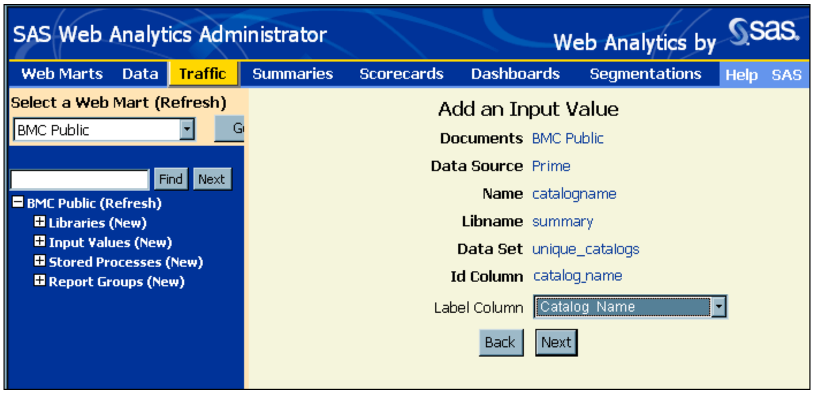
Data Set unique_catalogs

* Id Column catalog_name

Label Column

is the variable name that represents a column (in the data set that you specified in the **Dataset** field) that contains the descriptive strings or labels for the corresponding values in the ID column of the same data set.

Display 10.5 A Value for the Label Column Is Entered in the Add an Input Value Wizard



Add an Input Value

Documents BMC Public

Data Source Prime

Name catalogname

Libname summary

Data Set unique_catalogs

Id Column catalog_name

Label Column Catalog Name

For example, in the Wadtsort data set that is used by the SAS Web Analytics application, the ID column contains the following values: 1, 2, 3, 4, 5, 6, 7. The Label column contains the following descriptive labels that correspond to the previous values: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.

If you specify the variable name of the Label column of the Wadtsort data set, then the descriptions in the Label column will be displayed in the drop-down list for the benefit of the user: the drop-down list will display Sunday, Monday, Tuesday, and so on instead of 1, 2, 3, and so on.

If you do not specify a Label column, then the values are displayed in the drop-down list exactly as they appear in the ID column of the specified data set.

Define a Graph in a Report

You can define one or more graphs to be displayed in a report, but only one graph is displayed at a time.

You can also select a graph style and a graph scheme. The graph *style* affects the border design and the background of the Report Viewer panel around the graph. The graph *scheme* affects the colors of the text, label, legend, and lines inside the graph.

After you run the report to view the graph in the SAS Web Analytics Administrator, you can change the graph type, style, scheme, height, and width, and re-run the report to interactively view the various design combinations.

You can define the following types of graphs:

□ vertical bar chart

□ horizontal bar chart

□ line chart

- □ radar chart
- □ pie chart
- □ scatter chart

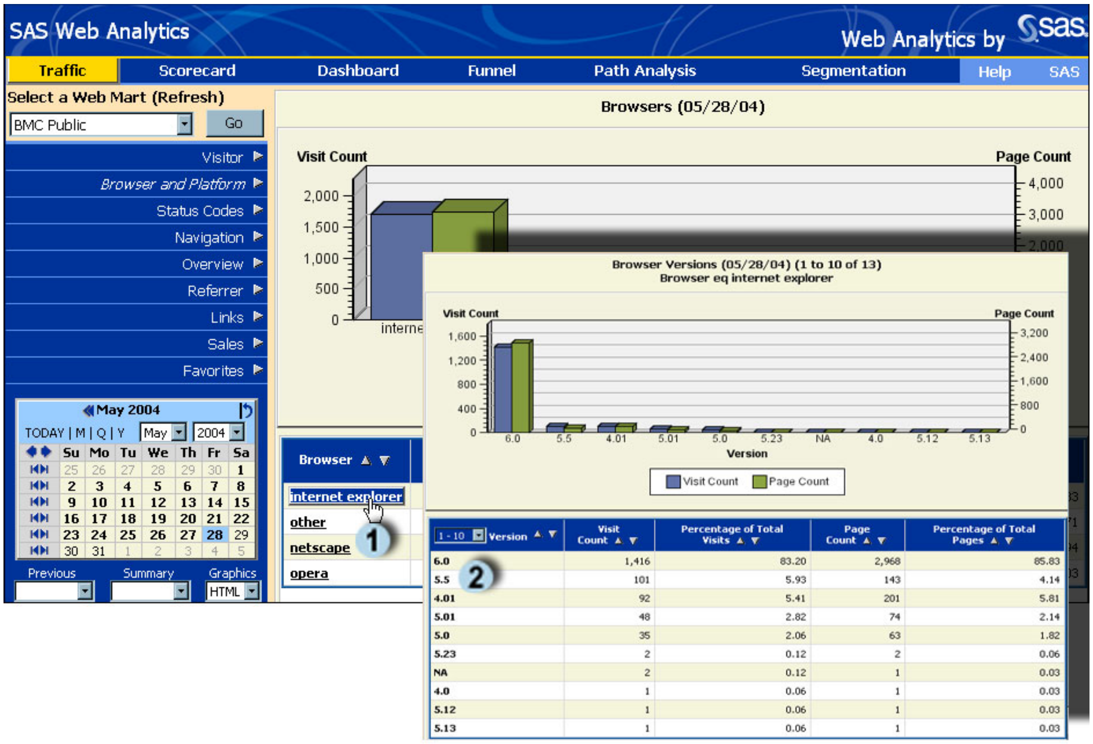
To define a graph for a report definition, perform the following steps:

1. Navigate to the Traffic page of the SAS Web Analytics Administrator. In the navigation tree, expand the node of the report definition to which you want to add a graph.
2. Click **`New`** next to **`Graphs`** below the selected report definition. The Add a Graph wizard appears in the right panel of the Traffic page.
3. Enter a name for the graph in the **`Enter Label`** field. Click **`Next`**.
4. Enter a description for the graph in the **`Enter Description`** field. Click **`Next`**.
5. Select the graph type from the drop-down menu and click **`Next`**.
6. Select the response variable from the available variable(s) by clicking the check box beside the variable name, and click **`Next`**.
7. Change the graph style and the graph scheme if you desire, and click **`Next`**. After you run the report, you will be able to view and interactively edit the graph style and scheme again.
8. You can optionally specify the heighth and width of the graph, or you can use the default size. After you run the report and view it, you will be able to edit the width and height of the graph.
9. Click **`Next`**, **`Finish`**, and **`Done`** to complete the definition of the graph.
10. Click **`Run`** to view the graph.
11. You can adjust the appearance of the graph by changing one or more parameters, and then view the results by clicking **`Run`**.
12. To add the graph to the report, double-click the report definition in the navigation tree of the Traffic page. The metadata table appears on the right.
13. Select **`Modify`** in the tool bar of the metadata table. The Documents Administrator Report Definition form appears.
14. Select the graph that you defined in the **`Default Graph`** drop-down menu, and click **`Submit`**.
15. Users can view and edit the graph in the SAS Web Analytics Report Viewer. See also the SAS Web Analytics User's Guide for more information about viewing graphs in the SAS Web Analytics Report Viewer.

Creating Drillable Reports

Overview: Drillable Reports

Display 10.6 Example Drillable Report in the SAS Web Analytics Report Viewer



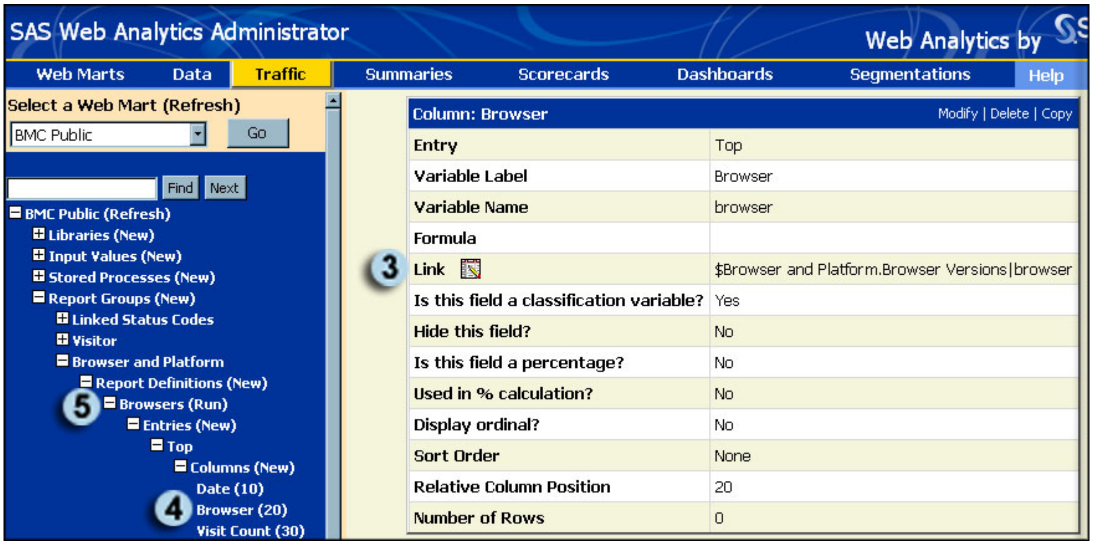
1 - 10 Version	Visit Count	Percentage of Total Visits	Page Count	Percentage of Total Pages
6.0	1,416	83.20	2,968	85.83
5.5	101	5.93	143	4.14
4.01	92	5.41	201	5.81
5.01	48	2.82	74	2.14
5.0	35	2.06	63	1.82
5.23	2	0.12	2	0.06
NA	2	0.12	1	0.03
4.0	1	0.06	1	0.03
5.12	1	0.06	1	0.03
5.13	1	0.06	1	0.03

A drillable report enables you to click on higher-level, summary information in order to view the detail data from which the summary data is derived.

For example, in the Browsers main report (previous display), a user can select the Internet Explorer link (❶) to view the breakdown of the versions of Internet Explorer in the Browser Versions drilled report (❷).

The attribute of a report that changes a column into a drill mechanism is the **`Link`** field (see the following display). You can specify values for the **`Link`** field (❸) in the metadata table of the Browser column (❹) of the Browsers report (❺). The value for the **`Link`** field contains the information that is needed for the drill action to occur between the main report and the drilled report.

Display 10.7 The Link Field of a Column: The Drill Mechanism



Column: Browser

Entry	Top
Variable Label	Browser
Variable Name	browser
Formula	
Link	$Browser and Platform.Browser Versions|browser
Is this field a classification variable?	Yes
Hide this field?	No
Is this field a percentage?	No
Used in % calculation?	No
Display ordinal?	No
Sort Order	None
Relative Column Position	20
Number of Rows	0

When you create a drillable report, perform the following general steps in this order to ensure that the link of a main report never points to a drilled report that does not exist:

1 Create or modify the drilled report (see “Create a New Report Definition” on page 149).

2 Create the required Date filter (see “Creating Filters” on page 163).

3 If necessary, create a filter for the drilled report so that the drilled report only displays the rows whose values match the selected cell from the main report. For example, if the user selects the Internet Explorer cell from the Browsers report, then the resulting drilled report definition (see Display 10.6 on page 185) should include a filter that instructs the report to display the versions of only the Internet Explorer browser, but not the versions of any other browsers.

4 Create or modify the main report. Create the link(s) from the main report to the drilled report(s) (see “Create a Link” on page 187).

5 In the report definition of the main report, set the **`Include in Report Menu?`** field to **`Yes`** so users can select the main report from the list in the SAS Web Analytics Report Viewer. If you want users to access the drilled report only from the main report, then in the report definition of the drilled report, set the **`Include in Report Menu?`** field to **`No`**.

Create a Link

To specify a new link that enables the user to access a drilled report from a main (drillable) report, select the column of the main report that you want to use as the link to the drilled report.

For example (see Display 10.7 on page 186), to specify a new link from the Browser column of the Browsers report (the main report) to the Browser Versions report (the drilled report), first select the browser column (❹) of the Browsers report (❺) on the Traffic page of the SAS Web Analytics Administrator. The metadata table of the Browser column appears in the right panel of the Traffic Page.

Follow these steps to access and use the Set the Column Link wizard, substituting your choices (where appropriate) in the example:

Note: Before the links (between reports) can be created, the drilled reports and the appropriate filters for displaying the drilled reports must exist. △

1 Click the icon next to **`Link`** (❸) in the metadata table for the selected column. The Set the Column Link wizard appears in the right panel of the Traffic page.

2 Select the report group of the drilled (destination) report from the **`Group`** drop-down menu. In the example, the Browser and Platform report group is selected because the Browser Versions report is in the Browser and Platform report group. Click **`Next`**.

3 Select the drilled report name (the report that you want to link to) from the **`Report`** drop-down menu. In the example, the Browser Versions report is selected. Click **`Next`**.

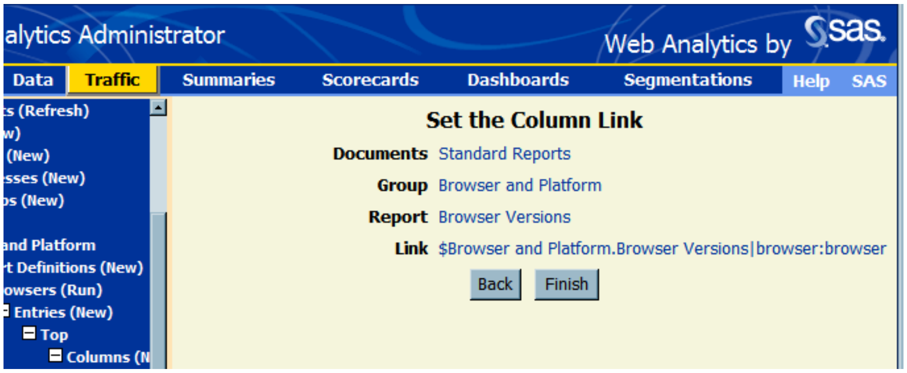
4 For the **`Parameter Field`**, select the column (of the main report) that will contain the link. Select the variable name for the parameter from the **`Parameter Variable`** drop-down menu. In the report definition for the Browser Version report, you are specifying a filter called Browser (the **`Parameter Field`**) with a variable called $browser (the **`Parameter Variable`**). Click **`Next`**.

5 Because you are specifying only one drill level to the Browser Versions report, click **`Next`**.

Note: If you are defining a multiple level drill report, then for each additional link to a sublevel report that you want to add, specify both a column label and a variable. The additional column label segment and variable segment is added to the entry in the Link field. △

A summary of the metadata that you provided about the new link appears (see the following display). Click **`Finish`** to save the changes. The link will be created when the reports are next run in the SAS Web Analytics Administrator or in the SAS Web Analytics Report Viewer.

Display 10.8 The Set the Column Link Wizard: Summary of the Metadata for a New Link



Set the Column Link

Documents Standard Reports

Group Browser and Platform

Report Browser Versions

Link $Browser and Platform.Browser Versions|browser:browser

The following table describes how the value for the **`Link`** property in this example is formatted:

Table 10.4 How to Format the Value for the Link Property for a Drillable Report

`$Browser and Platform.Browser Versions|browser`		
Part	**Description**	**Separator**
$Browser and Platform	The name of the report group that contains the drilled report. The report group is followed by the separator.	. (a period)
Browser Versions	The name of the drilled report. It is followed by the separator.	| (a vertical slash or pipe)
browser	The name of the field that connects the main report and the drilled reports.	

For an example of how to define the **`Link`** property in a *multiple-level drill report*, look at the Visit Referrer Domains report in the Traffic page of the SAS Web Analytics Report Viewer, and then note how the **`Link`** property is specified in the properties form of the **`Referrer Domain`** column.

CHAPTER 11

Scorecard Administration

Overview: Scorecard

The scorecard is a decision support report that enables the user to view the performance and the forecast values of the key performance indicators (KPIs) that are related to traffic on the Web site. The default KPI is the total session count for the current day that is being viewed. The scorecard performs the following tasks:

□ determines the variables in the input data set that have a statistically significant impact on the target variable

□ lists these variables in the order of how each variable affects the target variable

The scorecard requires a single data set that contains a set of metric variables that are summarized by day. The required columns for this data set are the Date variable, a target variable, and one or more input variables. Target variables are the numeric metrics that measure the success of a site. Input variables are the numeric metrics that are the potential drivers of the target metric. The default scorecard uses the Summary.Daily_Total_Day data set and uses Session Count as the target.

The scorecard is generated by code within the %WAETL macro. After the %WAETL macro summarizes all of the detail data, the default data set called Daily_Total_Day (located in the Summary library) is used as input to the scorecard report. By default, the Daily_Total_Day data set contains values for all of the traffic-type metrics and the metrics about the health of the Web site.

Note: Examples of traffic-related metrics are Session Count, Page Count, Duration, and Bytes Sent. Examples of metrics about the health of a Web site (that are contained in the Summary.Daily_Total_Day data set) are the status code variables such as ERROR_302_CNT and ERROR_402_CNT. △

Data Requirements for a Scorecard

Data Set Structure Requirements

Summary.Daily_Total_Day is the default data set that is used for scorecard analyses. The data set contains a single record for each date. To add metrics for a custom scorecard, you can do one of the following:

□ Create a new data set that contains summarized metrics by day with a Date column that contains the SAS date.

□ Add the new metrics to Summary.Daily_Total_Day.

Note: If you have more than one record per day, then the scorecard analysis will not run, and an error message appears in the log. △

Data Required for an Analysis to Run

In order to produce the scorecard statistics for a given date, at least 30 continuous days of data must exist in the input data set (Summary.Daily_Total_Day, by default) in the time interval before that date. If a day is missing, then the data for the missing day is estimated by using data from the same day of the week. For example, if Web log processing begins on January 1, then the SAS Web Analytics Report Viewer shows the following message in the scorecard for any day earlier than January 30:

```
NOT ENOUGH PREVIOUS DAYS OF DATA
```

All of the other numeric columns are set to zero for these days as well.

Defining a New Scorecard

Selecting Metrics for a Scorecard

To define a new scorecard, you select the a target metric and one or more input metrics for the scorecard. The following definitions describe the difference between a target metric and input metrics:

Target
: the business metric (variable) that is the performance indicator of the Web site. Examples of a target metric include the following:

- □ hits to particular content on the website
- □ number of sales or dollars from sales
- □ number of users who complete an online task
- □ users who unsubscribe
- □ single-hit sessions

CAUTION:
Never use a date variable as a target. △

Input
: metrics that have the potential to affect the target.

Guidelines for Choosing the Input Metrics for a Scorecard

Use the following guidelines in choosing the input metrics for a scorecard:

- □ The input metrics should make business sense for affecting the target metric.
- □ The input metrics should be measures that can be controlled. For example, the number of files that are not found might be controlled using site maintenance to find and correct broken links. Referrals from a particular Web site might be increased with banner ads or sponsored links.
- □ Each input metric should measure a unique activity and represent distinct information about the target. For example, if the target is *total book sales*, then do not include both the *percent book orders of total orders* and the *total book orders* as input metrics.

CAUTION:
Never use a date variable as an input metric. △

Identifying the Data Set for Storing the Target and Input Metrics

If the input metrics and the target metrics are not stored in the Summary.Daily_Total_Day data set, then you can either create a new data set of daily summaries, or you can add the target metrics and the input metrics to the

Summary.Daily_Total_Day data set. You can obtain data from Web logs or from external data sources such as customer relationship management databases.

Specifying Information to Enter in a Scorecard

Target and Input Metrics

After you verify that the data set for the target and input metrics exists, you define a set of metrics that are used to create the scorecard report. To create a new scorecard, the information that you need to provide includes the following:

□ a target metric

□ input metrics that have the potential to affect the target metric

□ business direction

□ measurement range

The Business Direction of a Metric in a Scorecard

The business direction indicates to the scorecard which direction is desirable for a given metric. For example, you might want the values for Session Count to increase over time. Therefore, you might set the Positive Business Direction for Session Count to **`up`**. For another metric such as error count (whose values you want to decrease over time), you might set the Positive Business Direction for Error 404 Count to **`down`**. The Positive Business Direction value for each metric is stored in the report definition for the scorecard.

The Measurement Range of a Metric in a Scorecard

The measurement range enables you to specify a valid range of values for an input variable. All values that lie outside this range are displayed as **`missing`**. You can specify the following measurement ranges:

□ number

□ positive number

□ negative number

□ percentage

□ negative percentage

□ positive percentage

□ proportion

□ negative proportion

□ positive proportion

Running a Custom Scorecard

One of the features that you specify when you define a custom scorecard is whether you want the scorecard to run automatically within the %WAETL macro. Initially, while you develop and test a custom scorecard, you might want to specify that it *not* run automatically within the %WAETL macro. After you fully develop and test the scorecard, you can change the specification so that the custom scorecard runs automatically within the %WAETL macro. If, however, the custom scorecard uses special data sets that are not available to the %WAETL macro, then even after you fully develop and test the custom scorecard, you would continue to specify that it not run automatically within the %WAETL macro.

Create a New Scorecard

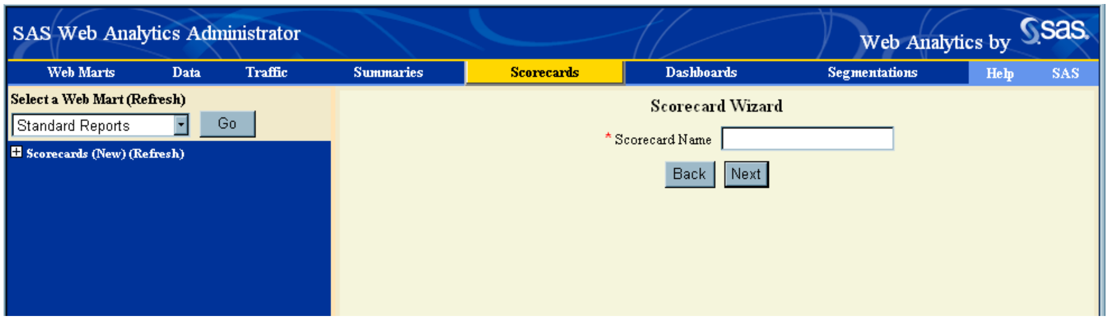
To create a new scorecard, perform the following steps:

1 Open the SAS Web Analytics Administrator and select the **Scorecards** link from the banner.

2 Select a Web mart (if it is not already selected) and click **Go**.

3 Click **New** next to **Scorecards** in the navigation tree at the left.

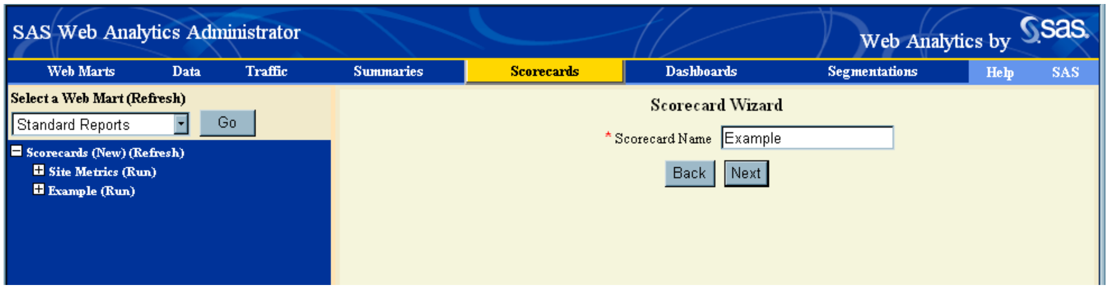
The scorecard wizard appears.



Select a Web Mart: Standard Reports

* Scorecard Name:

4 Enter a name for the new scorecard in the **Scorecard Name** field. The name can be a maximum of 25 characters and can include spaces. The name is the value that is used to identify the scorecard to the SAS Web Analytics application.

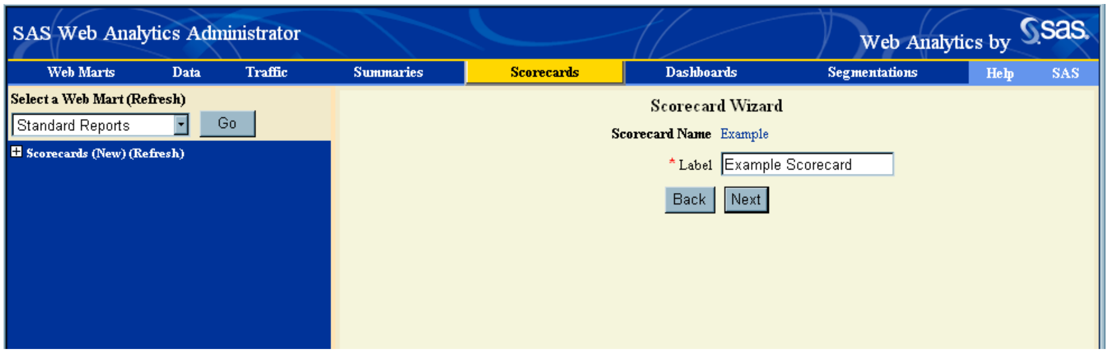


Select a Web Mart: Standard Reports

* Scorecard Name: Example

5 Click **Next**.

6 Enter a label for the new scorecard. The label is the text that will appear below **Scorecards** in the navigation tree.



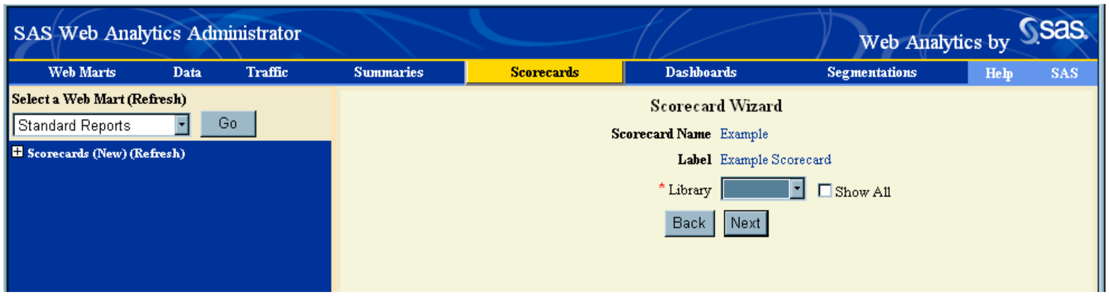
Scorecard Wizard

Scorecard Name Example

* Label Example Scorecard

7 Click **Next**.

8 From the **Library** drop-down menu, select a library that contains the data set that will be used for the scorecard. The default menu contains the Dated and Summary libraries. If your data set is not in one of these libraries, then select **Show All** to view an expanded list of libraries. After you select a library, click **Next**.



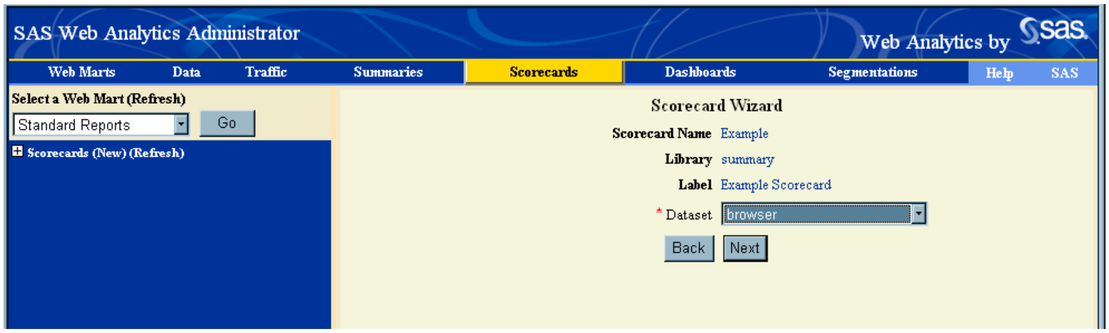
Scorecard Wizard

Scorecard Name Example

Label Example Scorecard

* Library ☐ Show All

9 Select a data set from the **Dataset** drop-down menu and then click **Next**.



Scorecard Wizard

Scorecard Name Example

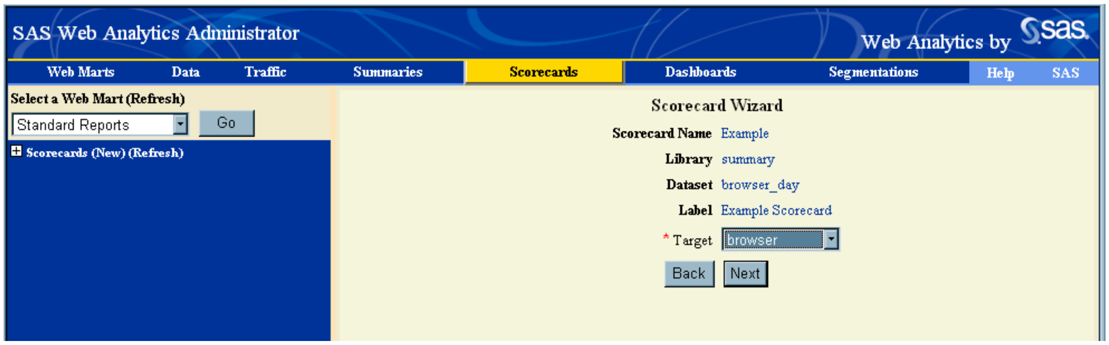
Library summary

Label Example Scorecard

* Dataset browser

10 Select a target metric from the **Target** drop-down menu and then click **Next**.

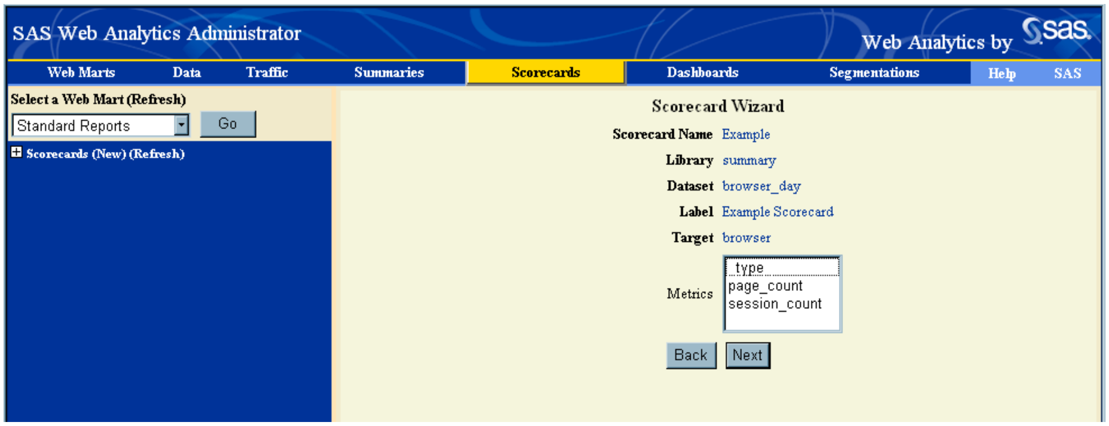
Note: You can select only one variable to be your target variable. △



Scorecard Wizard

Scorecard Name Example
Library summary
Dataset browser_day
Label Example Scorecard
* Target browser

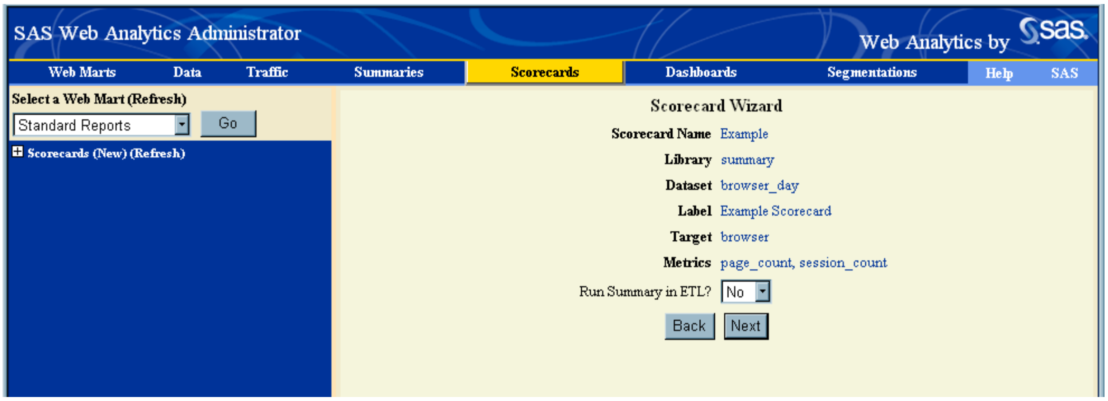
11 Select one or more input metrics from the **Metrics** list and then click **Next**.



Scorecard Wizard

Scorecard Name Example
Library summary
Dataset browser_day
Label Example Scorecard
Target browser
Metrics _type
page_count
session_count

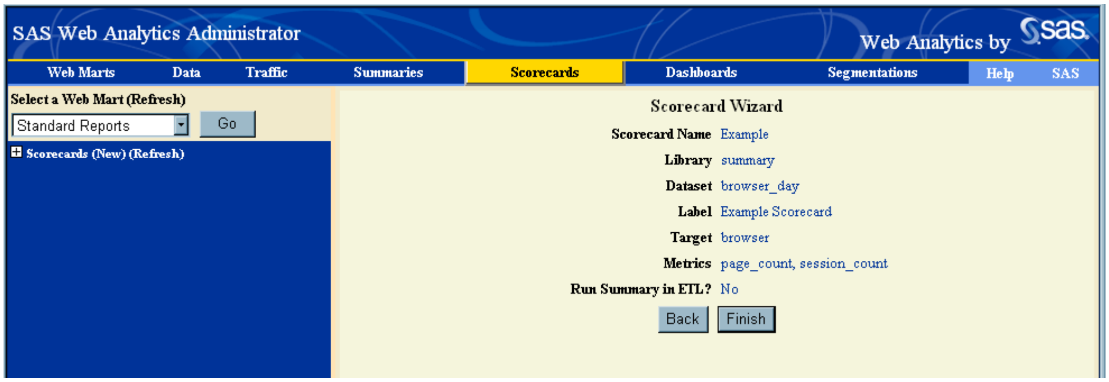
12 In the **Run Summary in ETL?** field, select **Yes** or **No** to indicate whether you want the scorecard to be automatically created the next time that the %WAETL macro runs. Click **Next**.



Scorecard Wizard

Scorecard Name Example
Library summary
Dataset browser_day
Label Example Scorecard
Target browser
Metrics page_count, session_count
Run Summary in ETL? No

13 Click **Finish** in order to create an entry within the Config.Wascrcrd data set for the new scorecard and to update the navigation tree.



Scorecard Wizard

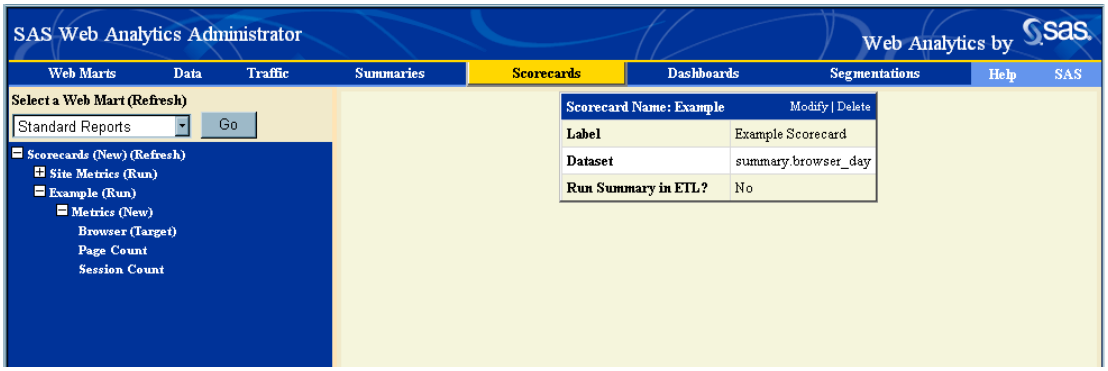
Scorecard Name	Example
Library	summary
Dataset	browser_day
Label	Example Scorecard
Target	browser
Metrics	page_count, session_count
Run Summary in ETL?	No

A message displays that indicates whether the update was successful.

14 Click **`Continue`**.

15 Click **+** to expand the **`Scorecards`** node.

16 Click **+** next to the new scorecard name in the navigation tree. The expanded node displays the input metrics and the target metric that you set up during the creation of the new scorecard. A metadata table appears on the right.



Scorecard Name: Example	
Label	Example Scorecard
Dataset	summary.browser_day
Run Summary in ETL?	No

If you answered **`Yes`** to the question about running the scorecard in ETL, then the new scorecard is generated the next time that the %WAETL macro runs. To view the scorecard before the %WAETL macro normally runs, configure and run the %WADECIDE macro. See "The %WADECIDE Macro" on page 262.

Adding, Modifying, and Deleting Metrics in a Scorecard

Add a New Metric

To add an input metric to the scorecard, or to specify a different target metric for a scorecard, select **`New`** next to **`Metrics`** below the scorecard you want to change. In the Scorecard New Metrics Wizard that appears, provide the values for each requested field. Click **`Next`** after each entry.

Modify a Metric

To modify a metric in the scorecard, perform the following steps:

1 Select the **`Scorecards`** link from the banner of the SAS Web Analytics Administrator.

2 In the navigation tree, expand the node of the scorecard to view its metrics.

3 Select a metric. Its metadata table appears on the right.

4 In the upper right corner of the metadata table, select **`Modify`** to modify the metric.

Delete a Metric

To delete a metric for a scorecard, perform the following steps:

1 Select the **`Scorecards`** link from the banner of the SAS Web Analytics Administrator.

2 In the navigation tree, expand the node of the scorecard to view its metrics.

3 Select a metric. Its metadata table appears on the right.

4 In the upper right corner of the metadata table, select **`Delete`** to delete the metric.

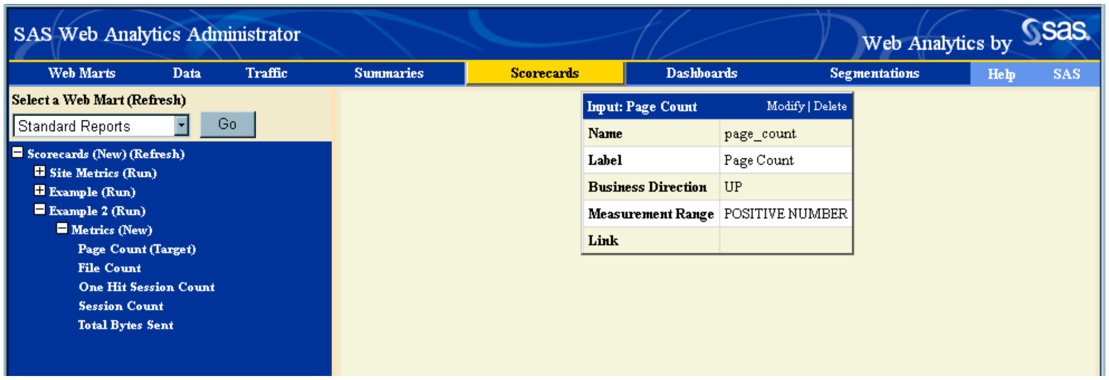
Change an Input Metric to a Target Metric

If you want to change an input metric to a target metric, you must delete the existing target metric, delete the input metric that you want to change to a target metric, and then add the input metric as the target metric. To accomplish this task, perform the following steps:

1 Select the **`Scorecards`** link from the banner in the SAS Web Analytics Administrator (the **`Scorecards`** link is yellow when selected).

2 In the navigation tree, click **+** to expand the scorecard name that contains the metric that you want to change.

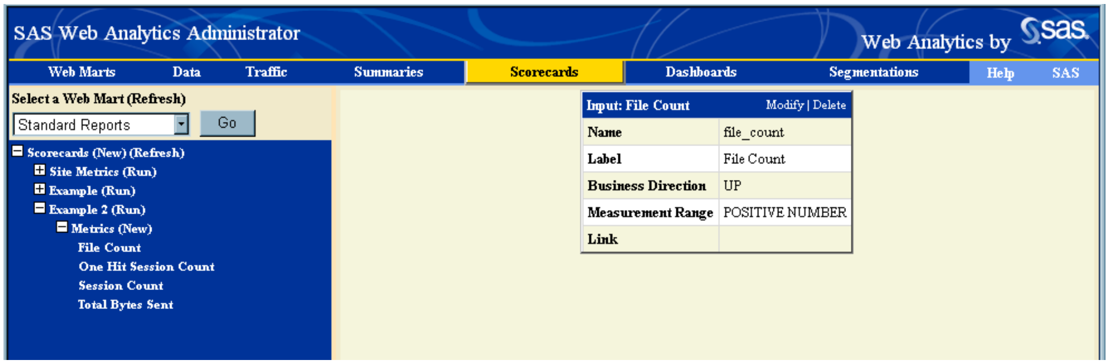
3 Click the target metric in the navigation tree to display the metadata table for the metric.



Input: Page Count	Modify | Delete
Name	page_count
Label	Page Count
Business Direction	UP
Measurement Range	POSITIVE NUMBER
Link	

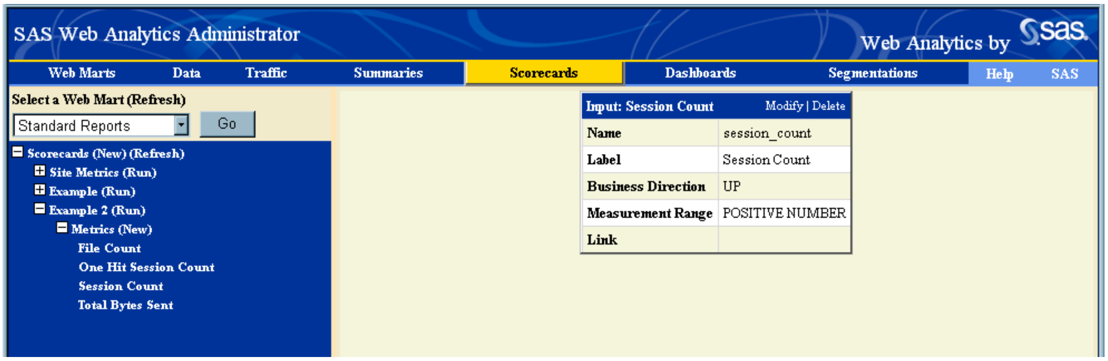
4 Select **`Delete`** in the upper right corner of the table to delete the target metric.

5 Click **`OK`** in the dialog box, and then click **`Continue`**. The target metric is removed from the navigation tree.



Input: File Count	
Name	file_count
Label	File Count
Business Direction	UP
Measurement Range	POSITIVE NUMBER
Link	

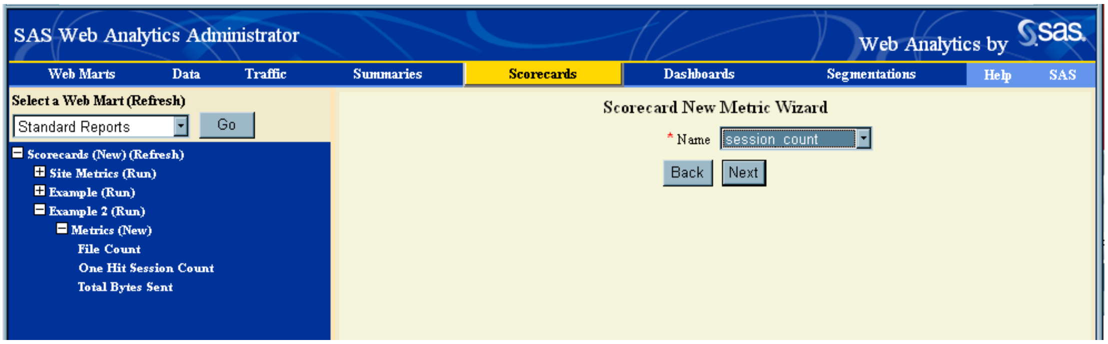
6 Click the input metric that you want to change to the new target metric.



Input: Session Count	
Name	session_count
Label	Session Count
Business Direction	UP
Measurement Range	POSITIVE NUMBER
Link	

7 Select **Delete** in the upper right corner of the table to delete the input metric.

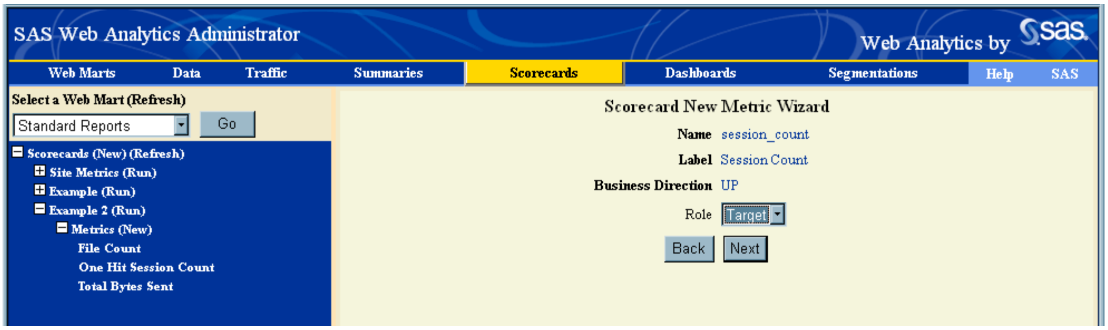
8 In the navigation tree, select **New** next to **Metrics** below the scorecard. You will re-create the metric that you deleted, but this time the former input metric will be a target metric.



Scorecard New Metric Wizard

* Name session_count

9 In the Scorecard New Metric Wizard that appears, provide the information for each of the prompts. Click **Next** after each entry. From the **Role** drop-down menu, be sure to select **Target**.



Scorecard New Metric Wizard

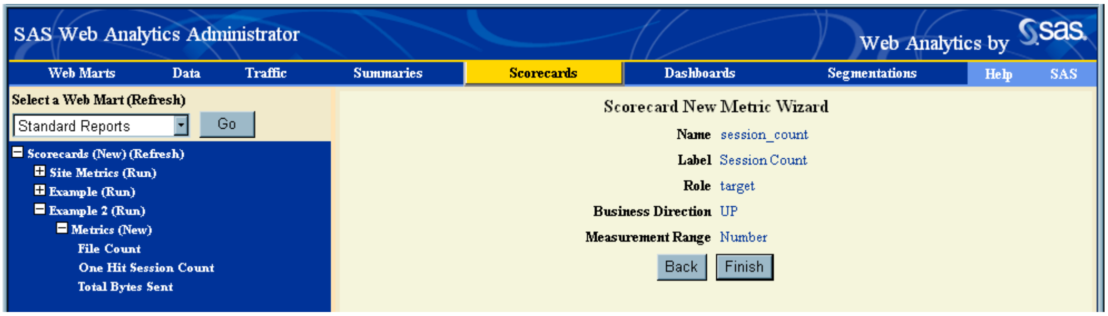
Name session_count

Label Session Count

Business Direction UP

Role Target

10 After you select a value for the **Measurement Range**, click **Next** and then click **Finish**.



Scorecard New Metric Wizard

Name session_count

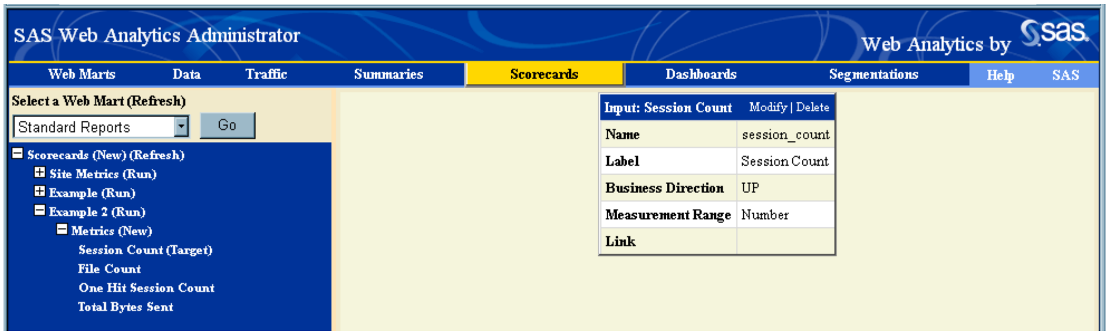
Label Session Count

Role target

Business Direction UP

Measurement Range Number

11 Click **Continue**. The metadata table for the newly created metric appears on the right of the Scorecards page.



Input: Session Count	Modify | Delete
Name	session_count
Label	Session Count
Business Direction	UP
Measurement Range	Number
Link	

12 To view the new metric in the navigation tree, expand the **Scorecards** node and the scorecard node of interest, if necessary.

Note: In order for a scorecard definition to be successfully processed, the definition must have one and only one target metric. △

Add Variables That Are Not in the Web Log to a Scorecard

Occasionally you might want to add variables to the summary data sets after the %WAETL macro has processed the Web log data. To add non-Web log variables to the summary data sets, perform the following steps:

1 Before you create the scorecard and dashboard definitions, create a new summary data set (in the Summary library) that will combine the data (for example, daily sales totals) from a non-Web log data source with data from a Web log.

The following code illustrates how you might create this new data set:

```
data summary.scorecard_day;
   merge summary.daily_total_day
         summary.daily_sales;
   by date;
   /* Be sure that non-Web log variables have labels. */
   label var1='Var1 label'
         var2='Var2 label'
         ...
         varX='VarX label';
run;
```

2 Create the scorecard definitions by using the SAS Web Analytics Administrator.

3 In the batch program Daily.sas, add the following code for each scorecard after the %WAETL invocation:

```
/* Scorecard example */
data summary.scorecard_day;
   merge summary.daily_total_day
         summary.daily_sales end=eof;
   by date;
   /*
   *  Put custom code here to handle missing data.
   */
   /* Get the first and last date in the scorecard data set. */
   if _n_ then
      call symput('first_date', "'"||put(date,date9.)||"'d");
   else if eof then
      call symput('last_date', "'"||put(date,date9.)||"'d");

   /* Be sure that non-Web log variables have labels. */
   label var1='Var1 label'
         var2='Var2 label'
         ...
         varX='VarX label';
run;

%wadecide(program=scorecard,
          date_1=&first_date,
          date_2=&last_date,
          swamart=&webmart\swamart,
          indsn=summary.scorecard_day,
          outlib=devlib,
          use_first_two_weeks=no,
          definition_to_run=scorecard_name
         );
```

Note: Custom metrics that are added to the Daily_Total_Day data set must have labels that are associated with the new variables. Labels are required for a customized data set. △

See also "General Information About SAS Web Analytics Summaries" on page 293.

Delete a Scorecard Definition

To delete a scorecard definition, perform the following steps.

1. Select the **Scorecards** link from the banner in the SAS Web Analytics Administrator (the **Scorecards** link is yellow when selected).
2. In the navigation tree, click the scorecard name for the scorecard definition that you want to delete.
3. In the metadata table that appears, click **Delete** in the upper right corner.

CAUTION:

Deleting a scorecard will "orphan" any analyses that currently exist. The Delete operation removes the scorecard definition from the scorecard metadata tables. △

Testing the Scorecard Output

The %WADECIDE Macro

To view the new scorecard, run the %WADECIDE macro within a SAS interactive session by using the example that follows as a guide. For more information about the %WADECIDE macro, see Appendix 1, "Macro Reference for the SAS Web Analytics 5.2 Solution," on page 261.

The following block of code can be used to test a scorecard:

```
libname devlib 'c:\xxx';
%wadecide(program=scorecard,
          date_1='1jun2004'd,
          date_2='30jun2004'd,
          swamart=c:\xxx\swamart,
          temp_store= c:\xxx\swamart\temp,
          indsn=devlib.testdata,
          outlib=devlib,
          use_first_two_weeks=no,
          definition_to_run=Example Scorecard
         );
proc print data=devlib.scorecard_data
              (where=('1jun2004'd le date le '30jun2004'd));
run;
proc print data=devlib.scorecard_metric_history
              (where=('1jun2004'd le date le '30jun2004'd));
run;
```

Explanation of the Example Code for the Scorecard

Here is the explanation of the preceding block of code:

- □ Use data that ranges from June 1, 2004 to June 30, 2004.
- □ Specify that the root directory of the SAS Web Analytics Web mart is at **c:\xxx\swamart**.

- □ Store the intermediate results in the **`c:\xxx\swamart\temp`** directory (the location of the Worklib library).
- □ Use the Devlib.Testdata data set as the data source.

  Note: The Devlib library has been defined separately from the %WADECIDE macro. △
- □ Store the scorecard report in the Devlib library.

  Note: When you provide an argument for the OUTLIB parameter, the output will not be stored in the default Summary library and you will not be able to see the results using the SAS Web Analytics Report Viewer. To display the data, two PROC PRINT steps have been added after the invocation of the %WADECIDE macro. △
- □ Exclude the first 14 days (first two weeks) of data.
- □ Create the scorecard with the name *Example Scorecard*.
- □ Display the scorecard report in the Output Window using PROC PRINT.
- □ Display the historical values for each metric plus seven days of forecasted values in the Output Window using PROC PRINT.

For more information about the arguments for the %WADECIDE macro, see Appendix 1.

The Table Layout of the Scorecard Report

View a Scorecard

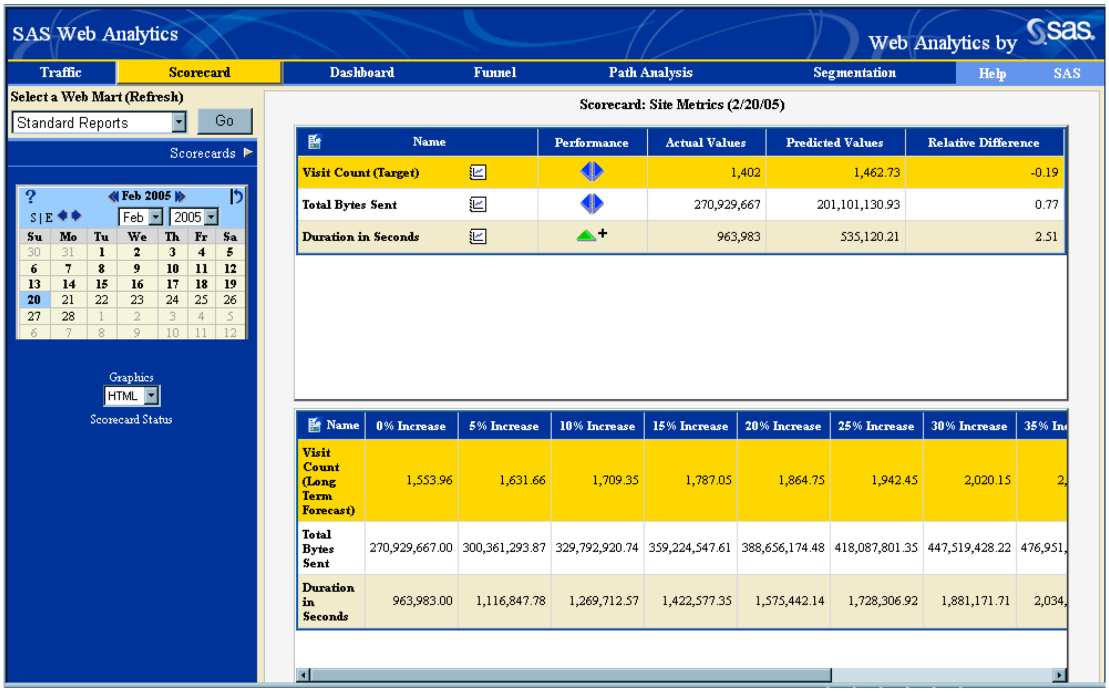
To view a scorecard, perform the following steps:

1. Select the **`Scorecard`** link in the SAS Web Analytics Report Viewer banner.
2. If a Web mart is not selected, then select a Web mart from the **`Select a Web Mart`** drop-down menu (located at the upper left of the SAS Web Analytics Report Viewer interface).

   Note: The title SAS Web Analytics (without the words Report Viewer) appears in the title bar of the SAS Web Analytics Report Viewer. In contrast, the SAS Web Analytics Administrator interface always includes the word Administrator in the title bar. △
3. Select a date from the calendar on the left.

The scorecard report has two tables. The upper table (Scorecard Table) contains the scorecard data for the selected day. This table lists the name of the scorecard and the date for which it was processed. This date is the date that you selected in the calendar. The lower table (Goal-Seeking Table) contains the goal-seeking data.

Display 11.1 Example of a Scorecard in the SAS Web Analytics Report Viewer



Scorecard: Site Metrics (2/20/05)

Name		Performance	Actual Values	Predicted Values	Relative Difference
Visit Count (Target)		◆	1,402	1,462.73	-0.19
Total Bytes Sent		◆	270,929,667	201,101,130.93	0.77
Duration in Seconds		▲+	963,983	535,120.21	2.51

Name	0% Increase	5% Increase	10% Increase	15% Increase	20% Increase	25% Increase	30% Increase	35% In
Visit Count (Long Term Forecast)	1,553.96	1,631.66	1,709.35	1,787.05	1,864.75	1,942.45	2,020.15	2,
Total Bytes Sent	270,929,667.00	300,361,293.87	329,792,920.74	359,224,547.61	388,656,174.48	418,087,801.35	447,519,428.22	476,951,
Duration in Seconds	963,983.00	1,116,847.78	1,269,712.57	1,422,577.35	1,575,442.14	1,728,306.92	1,881,171.71	2,034,

The Variables of the Site Metrics Scorecard Table

Name

The variable Name is the name of the key performance indicator (KPI) that is used to measure Web site activity. The first KPI that is listed in the column is the target metric. The target metric is the key performance indicator that you have determined to be most important to gauging the activity on the site. The remaining KPIs in the column are sorted in descending order based on their relative significance in predicting the performance of the target KPI. Each metric links to a forecast plot. Click the metric to view the forecast plot.

Performance

The symbol in the Performance column is determined by where the actual value of the metric falls within the 95% predicted confidence limits. In addition, the business direction of the metric is used to determine the symbol. The following table lists the values for the Performance symbols:

Table 11.1 The Performance Symbols in a Scorecard

Symbol	Description
◆	Indicates that the metric is performing at a steady business state. The actual value for this metric falls within the 95% confidence level for the predicted value.
▲+	Indicates that the metric value is increasing, and this trend matches the desired business direction. The actual value is above the 95% confidence level for the predicted value.
▲−	Indicates that the metric value is increasing, and this trend does not match the desired business direction. The actual value is above the 95% confidence level for the predicted value.
▼+	Indicates that the metric value is decreasing, and this trend matches the desired business direction. The actual value is below the 95% confidence level for the predicted value.
▼−	Indicates that the metric value is decreasing, and this trend does not match the desired business direction. The actual value is below the 95% confidence level for the predicted value.

Note: The plus (+) and minus (-) signs next to the arrows are another way of indicating the direction of the metric. The signs have no additional meaning. Either a minus sign or a red color indicates a trend in the Negative Business Direction. Either a plus sign or a green color indicates a trend in the Positive Business Direction. △

Actual Values

The value in the Actual Values column is the numeric percentage, proportion, count, or amount for the associated KPI. The Actual Values column contains the value for each metric (such as Visit Count) for the date that is specified in the report.

Predicted Values

The value in the Predicted Values column is the predicted value of the associated KPI. The predicted value is derived from the history data, taking into account any historic trends. The Predicted Values column contains the value that was predicted for the day the report was run, given the values for that metric in the past. Predicted values are determined by performing a variety of forecasting methods on each metric. Some of the exponential smoothing methods include the following routines:

- single
- double
- linear
- damped trend
- seasonal
- Winters multiplicative
- Winters additive

The forecast models are created by using all of the historical data that has been collected (actual values), but the current daily actual values are not used. The forecast model for the current day is computed from the selected model, and the corresponding prediction intervals and standard errors for the forecast estimates are reported in the graph.

Relative Difference

The Relative Difference column contains the value that is calculated using the following formula:

```
((actual value - predicted value)/standard deviation)/2
```

This formula gives a scaled magnitude measure of the difference between the actual value and the predicted value for each metric in the scorecard. You use this value to get an idea of how common (or how abnormal) the actual value for the metric is today, as opposed to a typical value for the metric in general.

Goal-Seeking Table (Site Metrics)

Use the Goal-Seeking Table in the scorecard in order to determine how the increases or decreases in certain KPIs will affect the value of the long-term forecast of the target KPI. The long-term forecast of the target KPI is always listed first in the Name column. The Goal-Seeking Table shows the increases in the long-term forecast of the target KPI from 0% to 50% in increments of 5%.

How Special Cases Are Handled

Predicting the Target

For a scorecard, you identify the target metric that you want to measure, and then identify the input metrics that you think will have a correlation with the target metric. Depending on whether a correlation exists, one of following conditions occurs:

- If the scorecard analysis determines that a metric predicts the target metric, then the input metric displays in the scorecard.
- If the scorecard analysis determines that a metric does not predict the target, then the input metric is not displayed in the table, because it is not statistically significant.

How Missing Values Are Handled

If a metric or target variable is missing for a day, then the value for the missing day is extrapolated by using the value from the missing metric or target variable from the same weekday in the previous week.

CHAPTER

12

Dashboard Administration

Overview: Dashboard

A dashboard describes how a particular metric performs as specified by its recent trend and how that trend relates to its desired business direction. The dashboard displays the values for each site's metric, assigns a performance level to that metric as specified by its desired business direction, and displays a historical average, minimum, and maximum for that metric for a given date. The dashboard also provides a plot of the historical values (for the past 7 days) for a metric by using an overlaid trend line.

The dashboard requires a single data set that contains a set of metric variables that are summarized by day. The default dashboard uses the Summary.Daily_Total_Day data set, which contains only the daily metric data from the Web log. Additional metric variables can be added to the Summary.Daily_Total_Day data set or to a new daily data set that is created for the dashboard.

The dashboard is generated by code within the %WAETL macro. After the %WAETL macro summarizes all of the detail data, the default data set called Daily_Total_Day (located in the Summary library) is used as input to the dashboard. By default, the Summary.Daily_Total_Day data set contains values for all of the traffic-type metrics and health-type metrics for the Web site.

Note: Examples of traffic-type metrics are Session Count, Page Count, Duration, and Bytes Sent. Examples of metrics about the health of a Web site (that are contained in the Summary.Daily_total_day data set) are the error code variables such as ERROR_302_CNT and ERROR_402_CNT. △

Data Requirements for a Dashboard

Data Set Structure Requirements

Summary.Daily_Total_Day is the default data set that is used for dashboard analyses. The data set contains a single record for each date. To add metrics for a custom dashboard, you can do one of the following:

□ Create a new data set that contains summarized metrics by day, with a Date column that contains the SAS date.

□ Add the new metrics to Summary.Daily_Total_Day.

Note: If more than one record per day exists, then the dashboard analysis will not run and an error message appears in the log. △

Data Required for an Analysis to Run

In order to produce the dashboard statistics for a given date, at least thirty continuous days of data must exist in the input data set (Summary.Daily_Total_Day, by default) in the time interval before that date. If data for a day is missing, then the data for the missing day is estimated by using data from the same day of the week. For example, if Web log processing begins on January 1, then the SAS Web Analytics Report Viewer shows the following message in the dashboard for any day earlier than January 30:

```
NOT ENOUGH PREVIOUS DAYS OF DATA
```

All of the other numeric columns are set to zero for these days as well.

Specifying the Information for a Metric in a Dashboard

For each metric, you need to define the following fields:

Label
is the text that is displayed as the name of the report in the Traffic page of the SAS Web Analytics Report Viewer (and in the navigation tree of the Traffic page of the SAS Web Analytics Administrator).

Category
is a specification under which you can group together (within a dashboard) several metrics that have a common feature.

Positive Business Direction
is a specification that tells the dashboard which direction is desirable for a given metric. For example, you might want the values for Session Count to increase over time. Therefore, you might set the Positive Business Direction for Session Count to **up**. For another metric such as error count, whose values you want to decrease over time, you might set the Positive Business Direction for Error 404 Count to **down**. The Positive Business Direction value for each metric is stored in the report definition for the dashboard.

Link
can be used to set a hyperlink from a row to another table. To provide the destination information for the hyperlink, use the following syntax:

```
{url}|variable1,variable2,...,variablen
```

To specify a URL to be a link to another SAS Web Analytics report, use the following syntax:

```
$(Report Group).(Report Name)
```

See also “Create a Link” on page 187.

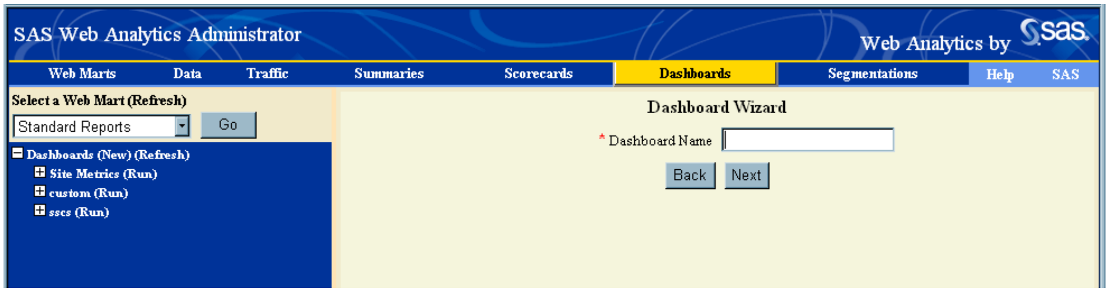
Create a New Dashboard

To create a new dashboard, perform the following steps:

1 Open the SAS Web Analytics Administrator. Select the **Dashboards** link from the banner.

2 Select a Web mart (if it is not already selected) and click **Go**.

3 Click **New** next to **Dashboards** in the navigation tree at the left. The Dashboard Wizard appears.

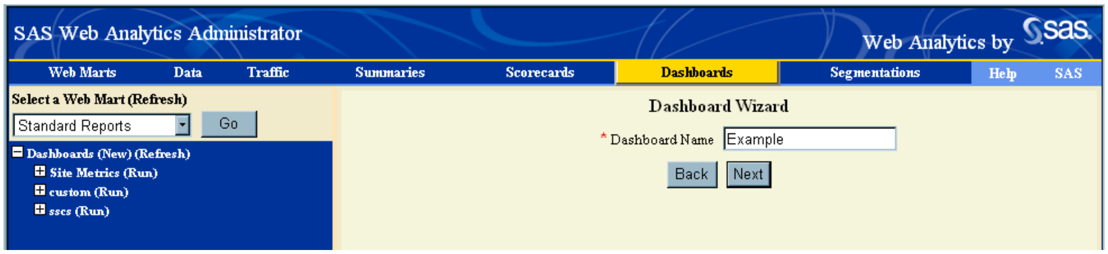


Standard Reports

Dashboard Wizard

* Dashboard Name

4 Enter a name for the new dashboard in the **Dashboard Name** field. The name is the value that is used to identify the dashboard to the SAS Web Analytics application. The name can be a maximum of 25 characters and can include spaces.



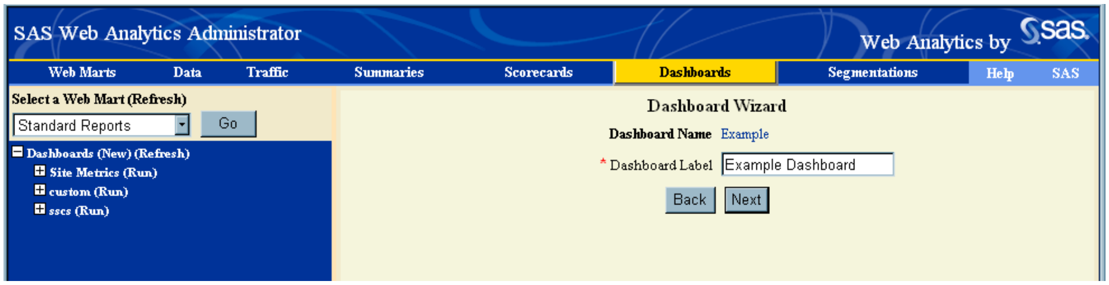
Standard Reports

Dashboard Wizard

* Dashboard Name Example

5 Click **Next**.

6 Enter the label for the new dashboard in the **Dashboard Label** field. The label is the text that appears as the name of the dashboard that users will select in the Dashboards page of the SAS Web Analytics Report Viewer (and in the navigation tree of the Dashboards page of the SAS Web Analytics Administrator).



Standard Reports

Dashboard Wizard

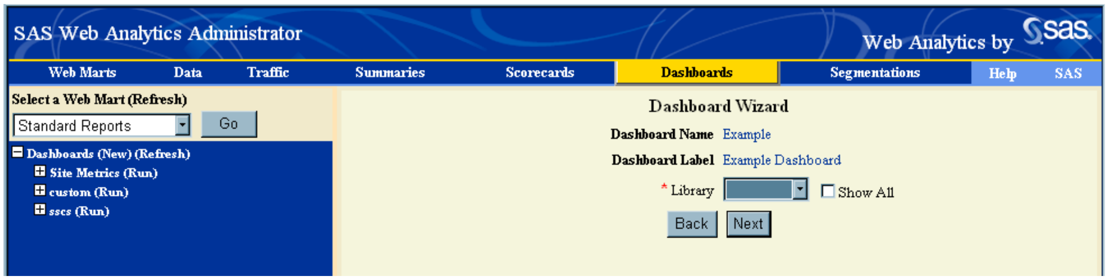
Dashboard Name Example

* Dashboard Label Example Dashboard

7 Click **Next**.

8 From the **Library** drop-down menu, select a library that contains the data set that will be used for the dashboard. The default menu contains the Dated and

Summary libraries by default. If your data set is not in one of these libraries, then select **Show All** to view an expanded list of libraries. After you select a library, click **Next**.



Dashboard Wizard

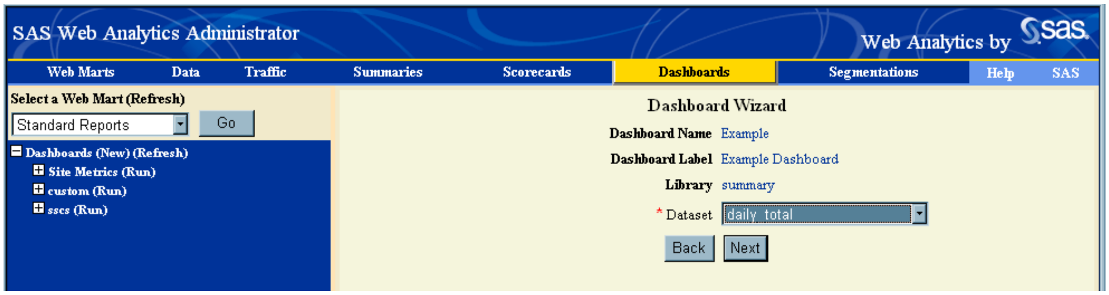
Dashboard Name Example

Dashboard Label Example Dashboard

* Library □ Show All

Select a Web Mart (Refresh): Standard Reports

9 Select a data set from the **Dataset** drop-down menu and then click **Next**.



Dashboard Wizard

Dashboard Name Example

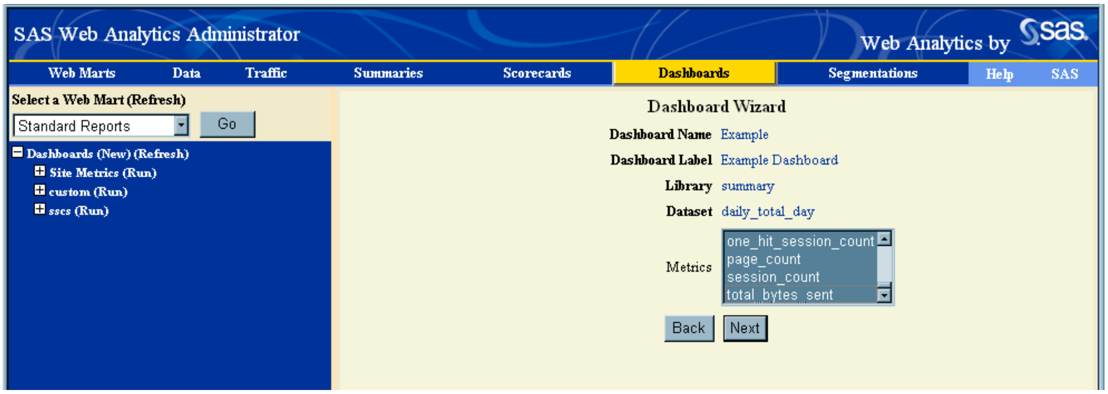
Dashboard Label Example Dashboard

Library summary

* Dataset daily_total

Select a Web Mart (Refresh): Standard Reports

10 Select one or more input metrics from the **Metrics** list. Use the standard Windows keyboard conventions to select contiguous or non-contiguous items from the list. After selecting metrics, click **Next**.



Dashboard Wizard

Dashboard Name Example

Dashboard Label Example Dashboard

Library summary

Dataset daily_total_day

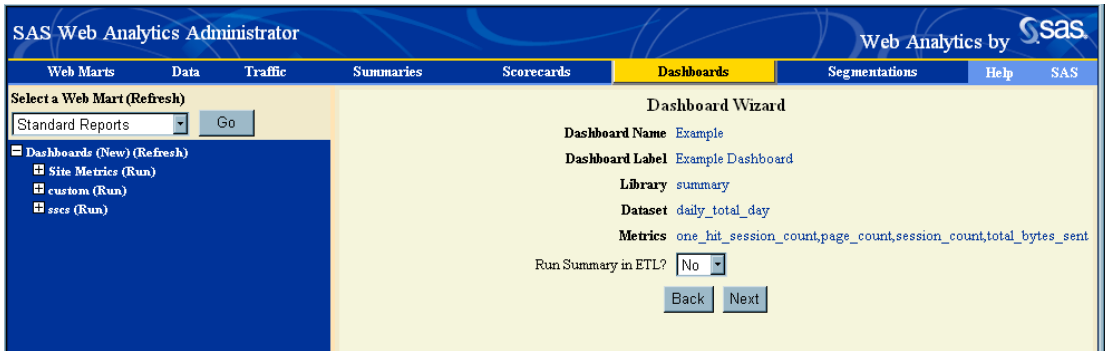
Metrics one_hit_session_count, page_count, session_count, total_bytes_sent

Select a Web Mart (Refresh): Standard Reports

CAUTION:

Never use a date variable as a metric. △

11 In the **Run Summary in ETL?** field, select **Yes** or **No** to indicate whether you want the scorecard to be automatically created the next time that the %WAETL macro runs. Click **Next**.



Dashboard Wizard

Dashboard Name: Example

Dashboard Label: Example Dashboard

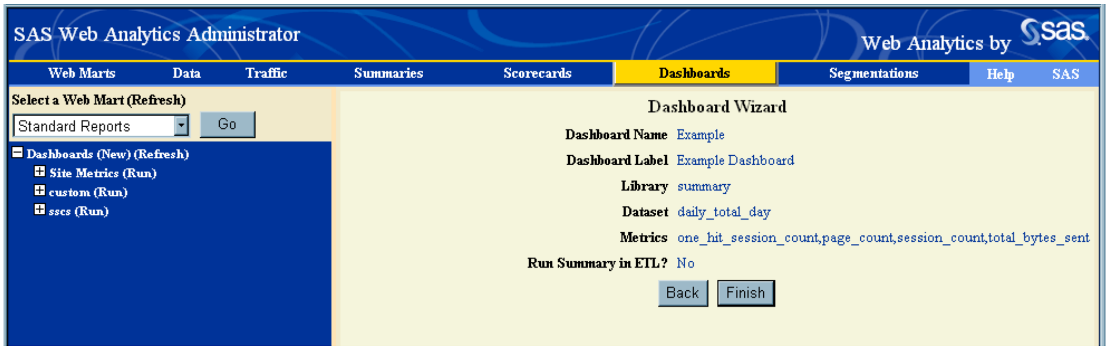
Library: summary

Dataset: daily_total_day

Metrics: one_hit_session_count,page_count,session_count,total_bytes_sent

Run Summary in ETL? No

12 Click **Finish** to complete the dashboard definition and create an entry in the Config.Wadshbrd data set.



Dashboard Wizard

Dashboard Name: Example

Dashboard Label: Example Dashboard

Library: summary

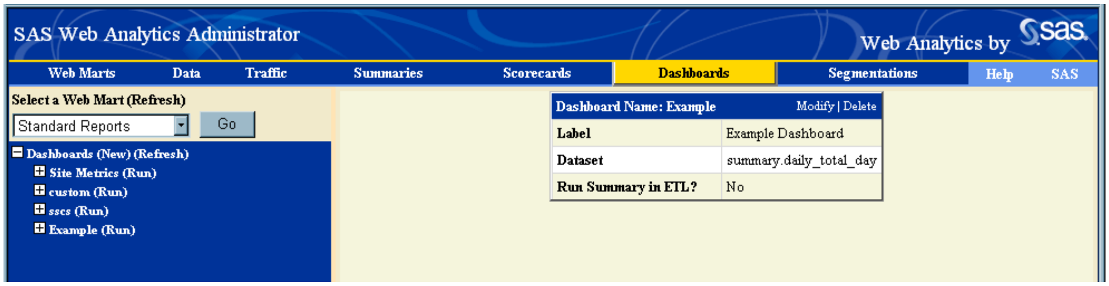
Dataset: daily_total_day

Metrics: one_hit_session_count,page_count,session_count,total_bytes_sent

Run Summary in ETL? No

A message displays that indicates whether the entry was updated successfully.

13 Click **Continue**. The navigation tree is updated with the new dashboard.



Dashboard Name: Example	Modify | Delete
Label	Example Dashboard
Dataset	summary.daily_total_day
Run Summary in ETL?	No

Adding, Modifying, and Deleting Metrics in the Dashboard

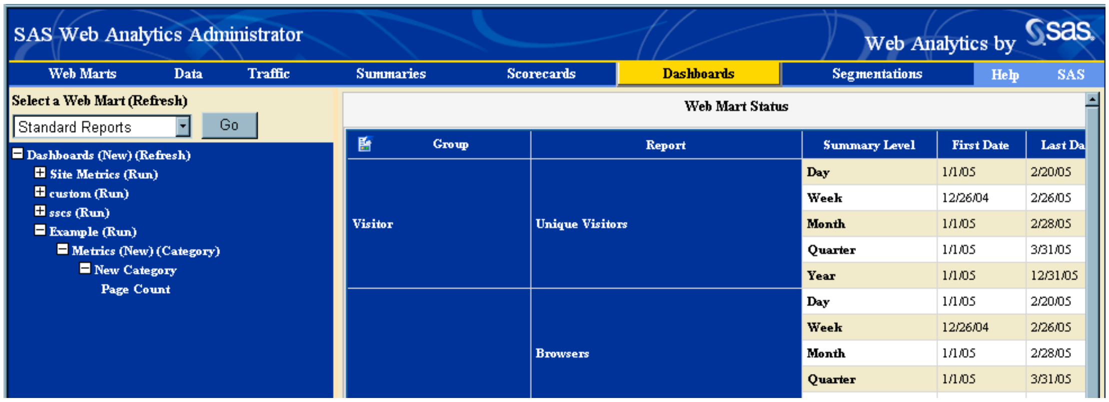
Add a New Metric

To add a new metric to a dashboard, perform the following steps:

1 Open the SAS Web Analytics Administrator. Select the **Dashboards** link from the banner.

2 Select a Web mart (if it is not already selected) and click **Go**.

3 Click **+** next to the dashboard you want to modify to expand the node in the navigation tree.

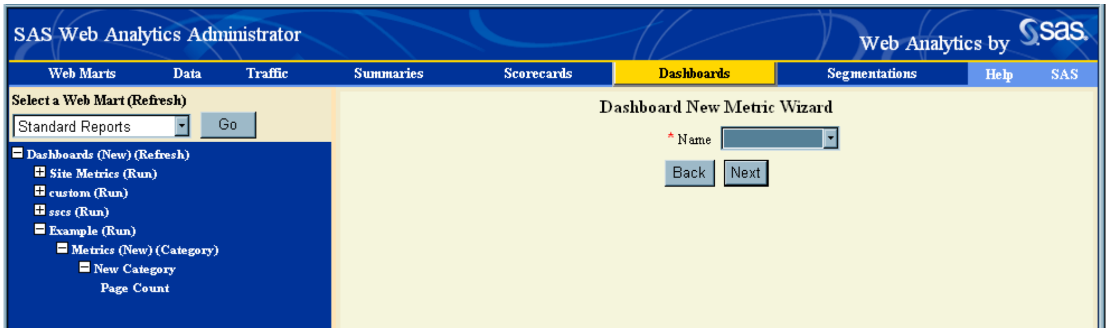


Web Mart Status

Group	Report	Summary Level	First Date	Last Da
Visitor	Unique Visitors	Day	1/1/05	2/20/05
		Week	12/26/04	2/26/05
		Month	1/1/05	2/28/05
		Quarter	1/1/05	3/31/05
		Year	1/1/05	12/31/05
	Browsers	Day	1/1/05	2/20/05
		Week	12/26/04	2/26/05
		Month	1/1/05	2/28/05
		Quarter	1/1/05	3/31/05

The node **Metrics (New) (Category)** appears below the dashboard.

4 Select **New** next to **Metrics** in the navigation page. The Dashboard New Metric Wizard appears.

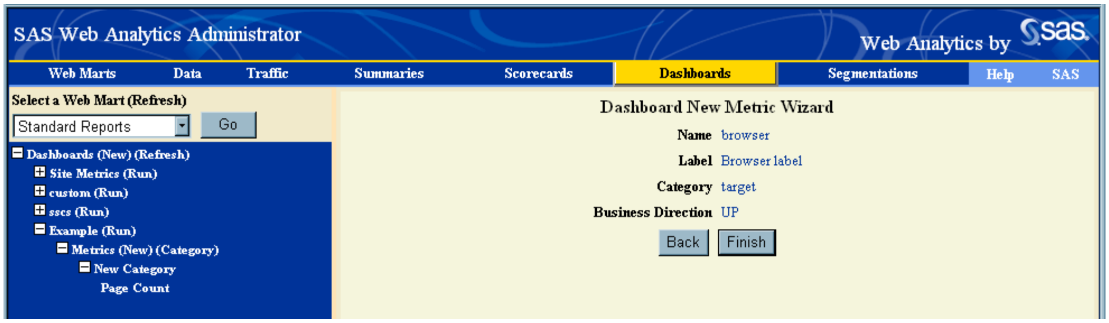


Dashboard New Metric Wizard

* Name

Provide the values for the fields that are required (see "Specifying the Information for a Metric in a Dashboard" on page 209).

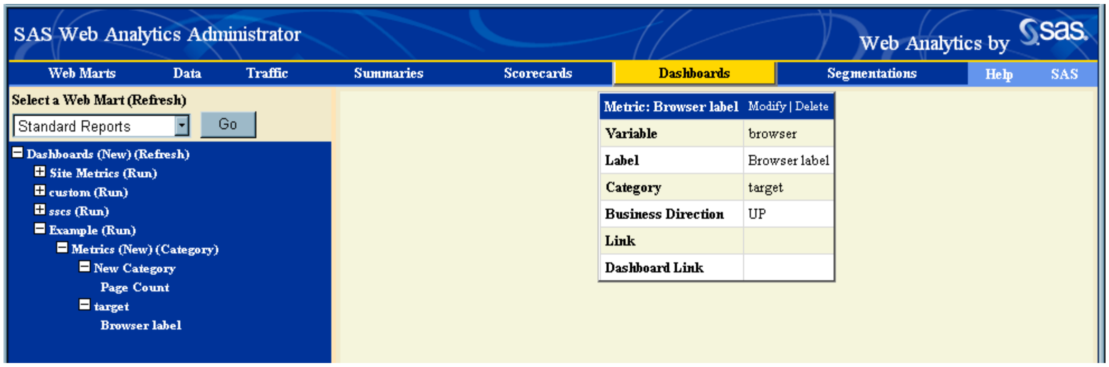
5 After you finish entering information in the fields, click **Finish**.



Dashboard New Metric Wizard

Name browser
Label Browser label
Category target
Business Direction UP

6 Click **Continue** to view the metric in the navigation tree.



Metric: Browser label	Modify | Delete
Variable	browser
Label	Browser label
Category	target
Business Direction	UP
Link	
Dashboard Link	

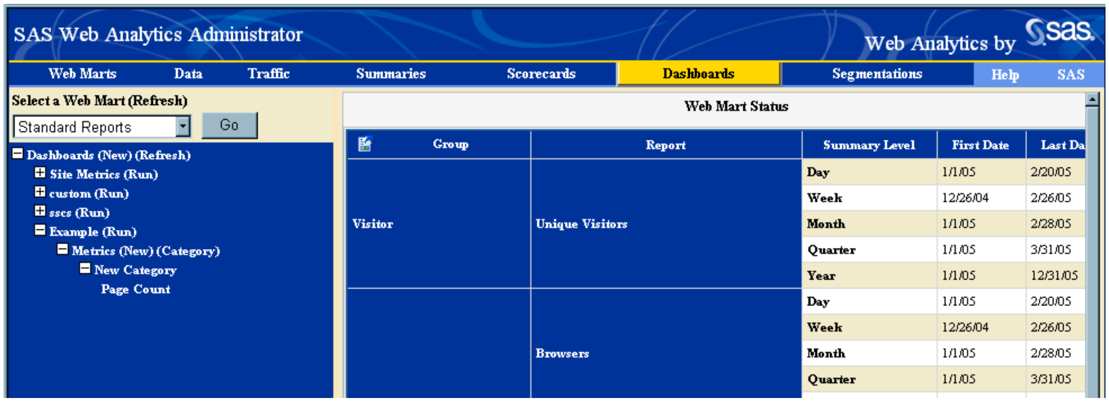
Modify a Metric

After you create a new dashboard, you can modify the attributes for each of the metrics that you specified during the creation of the dashboard by following these steps:

1 Open the SAS Web Analytics Administrator. Select the **Dashboards** link from the banner.

2 Select a Web mart (if it is not already selected) and click **Go**.

3 Click **+** next to the dashboard you want to modify to expand the node in the navigation tree.



Web Mart Status

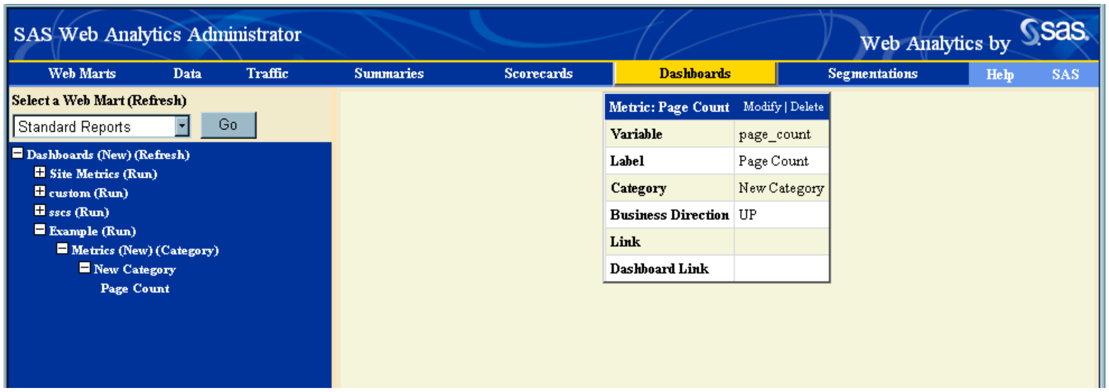
Group	Report	Summary Level	First Date	Last Da
Visitor	Unique Visitors	Day	1/1/05	2/20/05
		Week	12/26/04	2/26/05
		Month	1/1/05	2/28/05
		Quarter	1/1/05	3/31/05
		Year	1/1/05	12/31/05
	Browsers	Day	1/1/05	2/20/05
		Week	12/26/04	2/26/05
		Month	1/1/05	2/28/05
		Quarter	1/1/05	3/31/05

The node **Metrics (New) (Category)** appears below the dashboard name. If you entered the metrics during the creation of the dashboard, then you will also see those metrics listed below the **Metrics (New) (Category)** node.

Note: If no metrics have been specified for the dashboard, then see “Add a New Metric” on page 212 to add new metrics to a dashboard. △

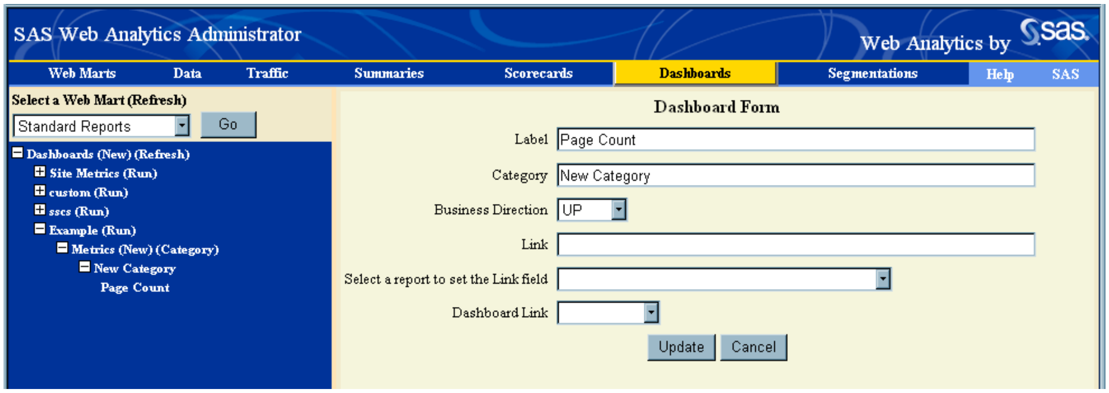
See also “Specifying the Information for a Metric in a Dashboard” on page 209.

4 Select a metric from the list. The metadata table for the metric appears on the right.



Metric: Page Count	Modify | Delete
Variable	page_count
Label	Page Count
Category	New Category
Business Direction	UP
Link	
Dashboard Link	

5 Select **`Modify`** in the upper right corner of the metadata table. The Dashboard Form appears on the right.



Dashboard Form

Label: Page Count

Category: New Category

Business Direction: UP

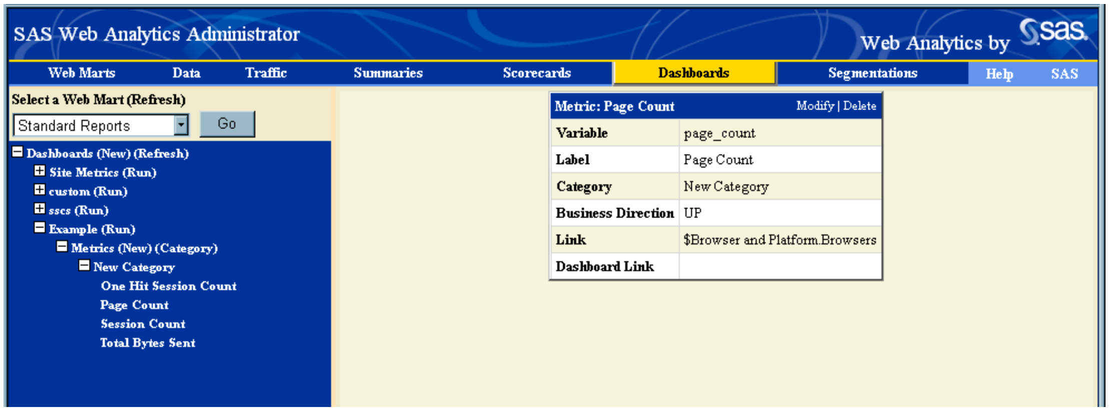
Link:

Select a report to set the Link field:

Dashboard Link:

6 Modify the values for each requested field as needed. The fields are described in "Specifying the Information for a Metric in a Dashboard" on page 209.

7 When you finish, click **`Update`**, and then click **`Continue`**. The updated metadata table for the metric reappears on the right.



Metric: Page Count	Modify | Delete
Variable	page_count
Label	Page Count
Category	New Category
Business Direction	UP
Link	$Browser and Platform.Browsers
Dashboard Link	

Delete a Metric

To delete a metric for a dashboard, perform the following steps:

1 Select the **Dashboards** link from the banner of the SAS Web Analytics Administrator.

2 In the navigation tree, expand the node of the dashboard to view its metrics.

3 Select a metric. Its metadata table appears on the right.

4 In the upper right corner of the metadata table, select **Delete** to delete the metric.

Add Variables That Are Not in the Web Log to a Dashboard

Occasionally you might want to add variables to the summary data sets after the %WAETL macro has processed the Web log data. To add non-Web log variables to the summary data sets, perform the following steps:

1 Before you create the scorecard and dashboard definitions, create a new summary data set (in the Summary library) that will combine the data (for example, daily sales totals) from a non-Web log data source with data from a Web log.

The following code illustrates how you might create this new data set:

```
data summary.dashboard_day;
   merge summary.daily_total_day
         summary.daily_sales;
   by date;
   /* Be sure that non-Web log variables have labels. */
   label var1='Var1 label'
         var2='Var2 label'
         ...
         varX='VarX label';
run;
```

2 Create the dashboard definitions by using the SAS Web Analytics Administrator.

3 In the batch program Daily.sas, add the following code for each dashboard after the %WAETL invocation:

```
/* Dashboard example */
data summary.dashboard_day;
   merge summary.daily_total_day
         summary.daily_sales end=eof;
   by date;
   /*
   *  Put custom code here to handle missing data.
   */
   /* Get the first and last date in the dashboard data set. */
   if _n_ then
      call symput('first_date', "'"||put(date,date9.)||"'d");
   else if eof then
      call symput('last_date', "'"||put(date,date9.)||"'d");
   /* Be sure that non-Web log variables have labels. */
   label var1='Var1 label'
         var2='Var2 label'
         ...
         varX='VarX label';
run;

%wadecide(program=dashboard,
          date_1=&first_date,
          date_2=&last_date,
          swamart=&webmart\swamart,
          indsn=summary.dashboard_day,
          outlib=devlib,
          use_first_two_weeks=no,
          definition_to_run=dashboard_name
         );
```

Note: Custom metrics that are added to the Daily_Total_Day data set must have labels that are associated with the new variables. Labels are required for a customized data set. △

See also "General Information About SAS Web Analytics Summaries" on page 293.

Delete a Dashboard Definition

To delete a dashboard definition, perform the following steps:

1 Select the **Dashboards** link from the banner in the SAS Web Analytics Administrator.

2 In the navigation tree, click the dashboard name for the dashboard definition that you want to delete.

3 In the metadata table that appears, click **Delete** in the upper right corner.

Testing the Dashboard Output

The %WADECIDE Macro

To view the new dashboard, run the %WADECIDE macro within a SAS interactive session.

The following block of code can be used to test a dashboard:

```
libname devlib 'c:\xxx';
%wadecide(program=dashboard,
          date_1='1jun2004'd,
          date_2='30jun2004'd,
          swamart=c:\xxx\swamart,
          temp_store= c:\xxx\swamart\temp,
          indsn=devlib.testdata,
          outlib=devlib,
          definition_to_run=Example Dashboard
         );
proc print data=devlib.dashboard_data
              (where=('1jun2004'd le date le '30jun2004'd));
run;
proc print data=devlib.dashboard_metric_history
              (where=('1jun2004'd le date le '30jun2004'd));
run;
```

Explanation of the Example Code for the Dashboard

Here is the explanation of the preceding block of code:

□ Use data that ranges from June 1, 2004 to June 30, 2004.

□ Specify that the root directory of the SAS Web Analytics Web mart is at **c:\xxx\swamart**.

□ Store intermediate results in the **c:\xxx\swamart\temp** directory (the location of the Worklib library).

□ Use the Devlib.Testdata data set as the data source.

Note: The Devlib library has been defined separately from the %WADECIDE macro. △

□ Store the dashboard report in the Devlib library.

Note: When you provide an argument for the OUTLIB parameter, the output will not be stored in the default Summary library and you will not be able to see the results using the SAS Web Analytics Report Viewer. To display the data, two PROC PRINT steps have been added after the invocation of the %WADECIDE macro. △

□ Create the dashboard with the name *Example Dashboard*.

□ Display the dashboard report in the Output Window using PROC PRINT.

□ Display the historical values for each metric in the Output Window using PROC PRINT.

For more information about the arguments for the %WADECIDE macro, see “The %WADECIDE Macro” on page 262.

CHAPTER 13

Setting Up a Segmentation Report

Data Requirements for the Default Segmentation Report

The default segmentation report uses the data set called Unique_Visitor_Segmentation, which contains visitor-level information by day. Additional data can be added to this data set or to a new data set that contains the variables that are listed in the example scenario.

Note: For the default segmentation report analysis, the target variable must have no more than four levels. △

Creating a Segmentation Report

Selecting Metrics for a Segmentation Report

Before you begin to create a new segmentation report, you need to identify a target metric, one or more input metrics, and a measure level for the segmentation:

Table 13.1 Available Metrics in a Segmentation Report

Metric	Description	
Target	Specifies a variable that is used to determine site performance. For example, a target can consist of a variable that indicates whether a visitor is a repeat visitor, or whether a visitor has made a purchase. Only one target is permitted.	
Input	Specifies a set of variables that have the potential to affect the target.	
Measurement level	Specifies the type of value that is contained within the input or target variable:	
	Nominal	Specifies that the value is an alphanumeric string such as 0 or 1, or red, white and blue.
	Ordinal	Specifies that the value is an integer or ordered character (A,B,C, and so on).
	Interval	Specifies that the value is a range of numbers, which can include fractions.

Identifying Variable Roles

Variables in a segmentation report can have the following roles:

Table 13.2 Roles of Variables in a Segmentation Report

Role	Description
Target	The target variable is required and must have a value of 0 or 1. □ 0 indicates that the visitor performed the action only in the explanatory (historical) period. See “Defining the Explanatory (Historical) and Response Time Periods” on page 235 for a full explanation of explanatory and response periods. □ 1 indicates that the visitor performed the action in both the explanatory and the response periods.
Input	The input variables are required. These are predictive variables that are summarized by visitor only for the explanatory time period. All analysis variables must end with _anl. If they do not, then the second report table will not be created. The second report table displays the proportion of visitors within each segment that is defined by the segmentation analysis.
Visitor ID	The visitor ID variable is required. This variable identifies a unique user, visitor, or customer. The analysis data set can contain only one record per visitor.
New visitor indicator	The new visitor indicator variable is optional. This variable indicates that the visitor first entered the site during the response period. The new visitor indicator has one of the following values: □ 0 indicates that the visitor is not new. □ 1 indicates that the visitor is new.
Most current visitor indicator	The most current visitor indicator variable is optional. This variable indicates whether a visitor entered the site for the first time on the maximum date in the detail data set. The most current visitor indicator has one of the following values: □ 0 indicates that the visitor did not come to the site for the first time on the maximum date. □ 1 indicates that the visitor came to the site for the first time on the maximum date.

The Segmentation Wizard

You use the Segmentation Wizard to define a set of inputs that are used by decision tree analysis. This analysis creates a set of *segments* that can be used to define the visitors to the Web site.

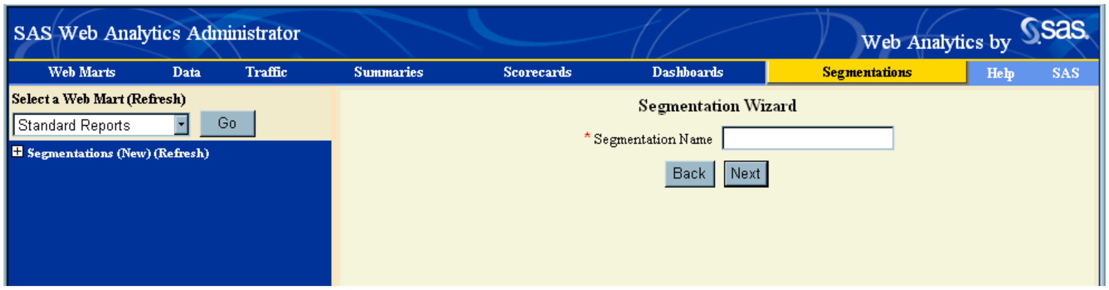
Create a Segmentation Report

To create a segmentation report, perform the following steps:

1 If a Web mart is not already selected, select a Web mart from the pull-down menu in the upper left corner of the SAS Web Analytics Administrator, and click **Go**.

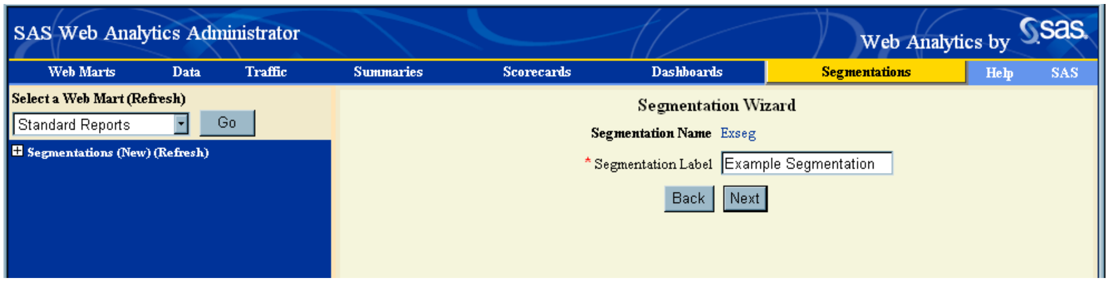
2 Select the **Segmentations** link in the banner so that it is highlighted.

3 Click **`New`** next to **`Segmentations`** in the navigation tree on the left. The Segmentation Wizard displays on the right.



4 Enter the segmentation name in the **`Segmentation Name`** field and then click **`Next`**.

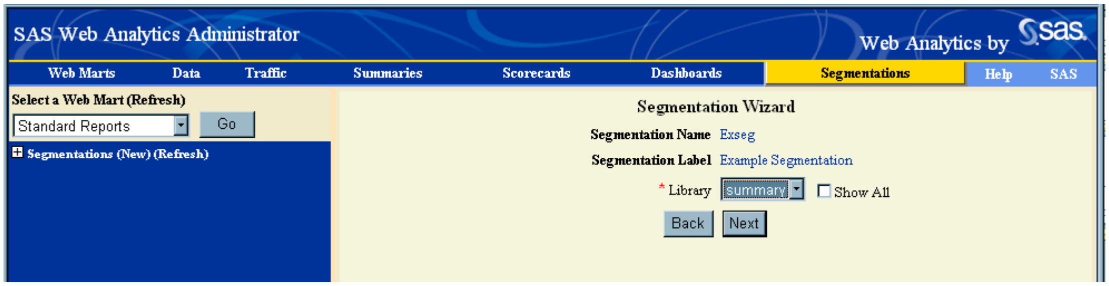
5 Enter the segmentation label in the **`Segmentation Label`** field and then click **`Next`**. The segmentation name and the segmentation label can each be up to 25 characters long. The segmentation name can be the same as the segmentation label, but they are used differently. SAS Web Analytics software uses the segmentation name to identify a segmentation, and uses the segmentation label as the text in the navigation tree of the SAS Web Analytics Administrator and the SAS Web Analytics Report Viewer.



Segmentation Name Exseg

* Segmentation Label Example Segmentation

6 Select a library from the **`Library`** drop-down menu. The default list contains the Dated and the Summary libraries. If your data set is not in one of these libraries, then select **`Show All`** to view an expanded list of libraries. After you select a library, click **`Next`**.

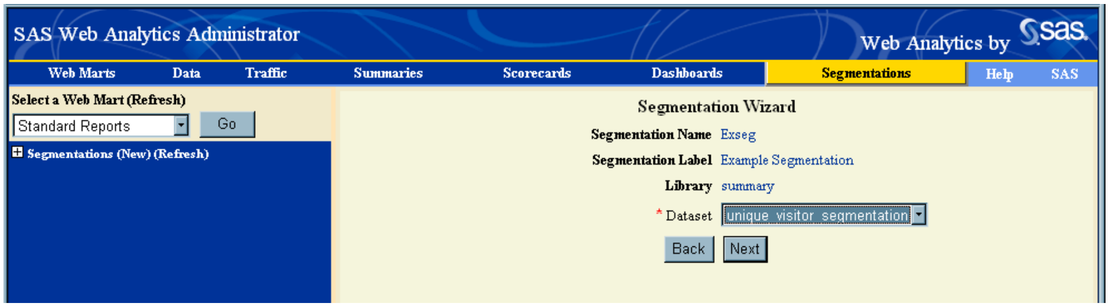


Segmentation Name Exseg

Segmentation Label Example Segmentation

* Library summary □ Show All

7 Select a data set from the **`Dataset`** drop-down menu and then click **`Next`**.



Segmentation Wizard

Segmentation Name Exseg

Segmentation Label Example Segmentation

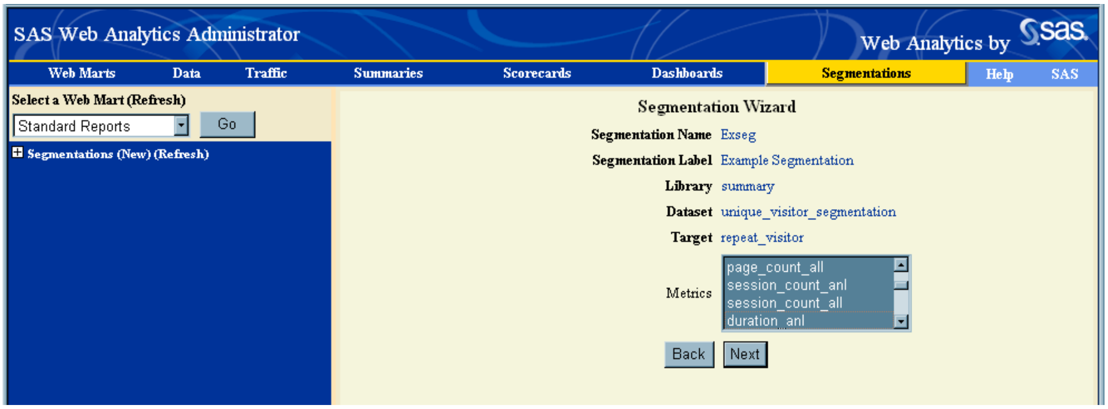
Library summary

* Dataset unique_visitor_segmentation

8 Select a target variable from the **`Target`** drop-down menu and then click **`Next`**.

Note: You are required to select a target variable, and you can select only one. △

9 Select one or more input metrics from the **`Metrics`** list and then click **`Next`**.



Segmentation Wizard

Segmentation Name Exseg

Segmentation Label Example Segmentation

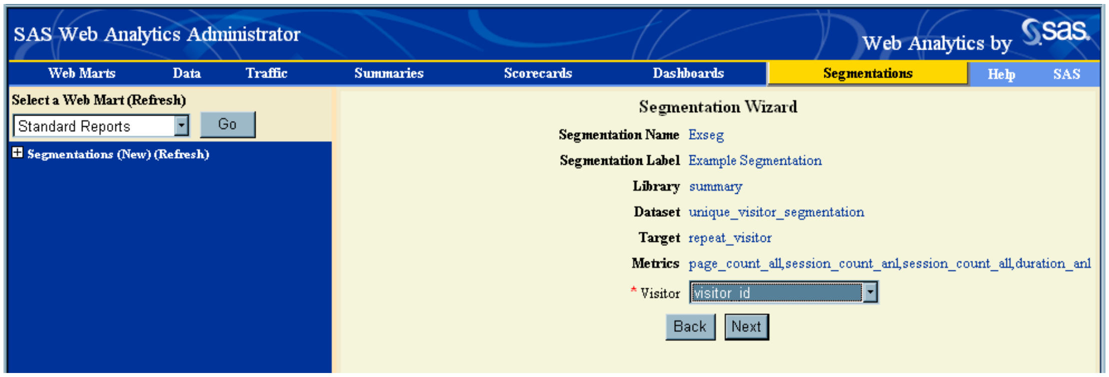
Library summary

Dataset unique_visitor_segmentation

Target repeat_visitor

Metrics page_count_all, session_count_anl, session_count_all, duration_anl

10 Select the visitor identification variable from the **`Visitor`** drop-down menu and then click **`Next`**.



Segmentation Wizard

Segmentation Name Exseg

Segmentation Label Example Segmentation

Library summary

Dataset unique_visitor_segmentation

Target repeat_visitor

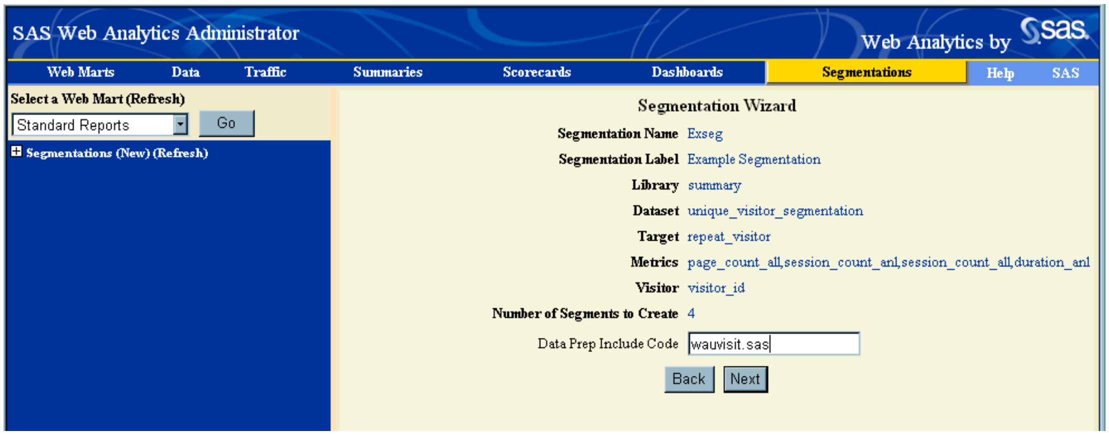
Metrics page_count_all,session_count_anl,session_count_all,duration_anl

* Visitor visitor_id

11 Enter an integer value for the number of segments to create in the **`Number of Segments to Create`** field and then click **`Next`**.

The default value is 4. Unless you have compelling reasons for selecting another value, use the default.

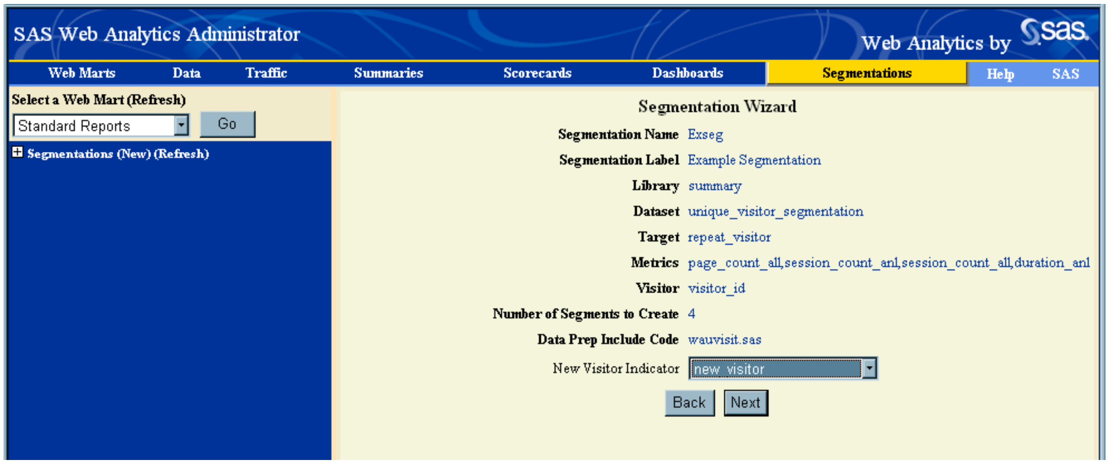
12 Enter the name of the program that contains the customized code in the **`Data Prep Include Code`** field. (See "Requirements for a Segmentation Analysis Data Set" on page 235 for more information about creating an analysis data set.) The program that is identified in this field needs to exist in the SAS subdirectory of the Web mart before you can run the segmentation. Click **`Next`**.



Segmentation Wizard

Segmentation Name: Exseg
Segmentation Label: Example Segmentation
Library: summary
Dataset: unique_visitor_segmentation
Target: repeat_visitor
Metrics: page_count_all,session_count_anl,session_count_all,duration_anl
Visitor: visitor_id
Number of Segments to Create: 4
Data Prep Include Code: wauvisit.sas

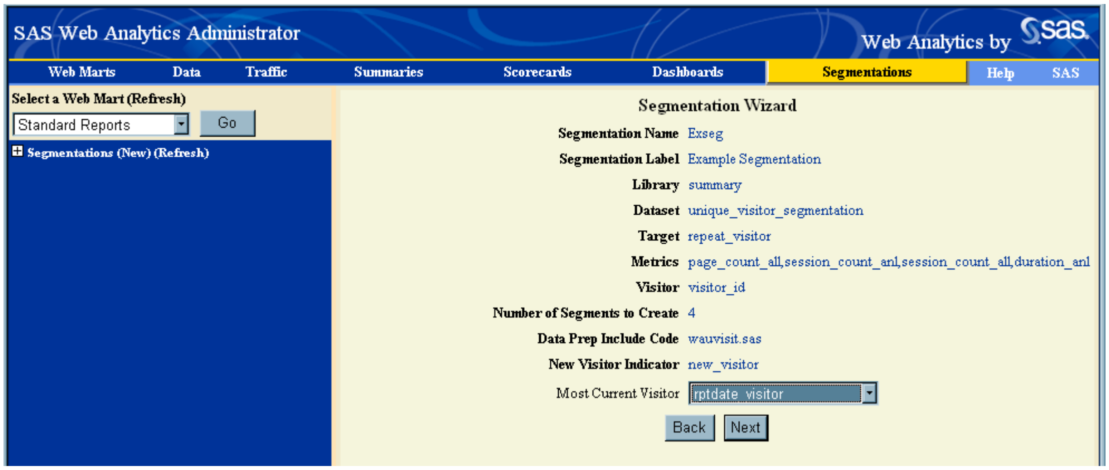
13 If you are using an optional new visitor indicator, select the indicator from the **`New Visitor Indicator`** drop-down menu and then click **`Next`**.



Segmentation Wizard

Segmentation Name: Exseg
Segmentation Label: Example Segmentation
Library: summary
Dataset: unique_visitor_segmentation
Target: repeat_visitor
Metrics: page_count_all,session_count_anl,session_count_all,duration_anl
Visitor: visitor_id
Number of Segments to Create: 4
Data Prep Include Code: wauvisit.sas
New Visitor Indicator: new_visitor

14 If you are using an optional most current visitor variable, select the variable from the **`Most Current Visitor`** drop-down menu and then click **`Next`**.

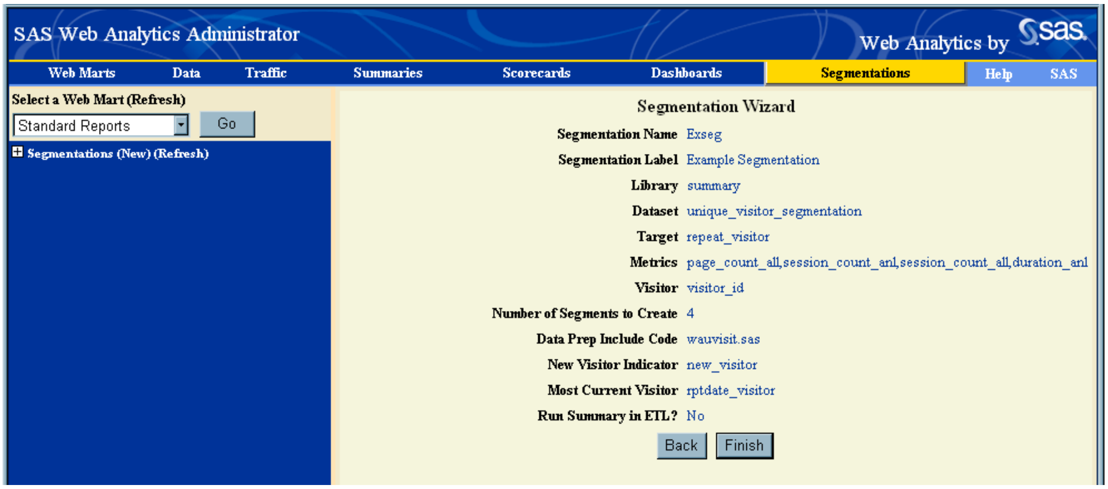


Segmentation Wizard

Segmentation Name: Exseg
Segmentation Label: Example Segmentation
Library: summary
Dataset: unique_visitor_segmentation
Target: repeat_visitor
Metrics: page_count_all,session_count_anl,session_count_all,duration_anl
Visitor: visitor_id
Number of Segments to Create: 4
Data Prep Include Code: wauvisit.sas
New Visitor Indicator: new_visitor
Most Current Visitor: rptdate_visitor

15 In the **`Run Summary in ETL?`** field, select **`Yes`** or **`No`** to indicate whether you want the segmentation to be automatically created the next time that the %WAETL macro runs. Click **`Next`**.

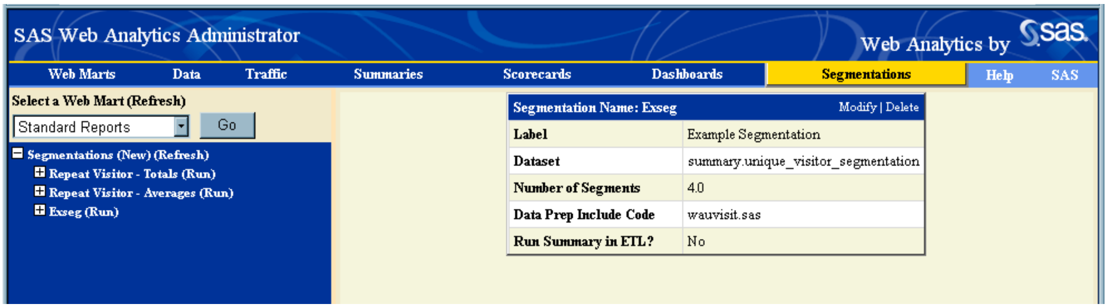
16 Click **`Finish`**.



Segmentation Wizard

Segmentation Name	Exseg
Segmentation Label	Example Segmentation
Library	summary
Dataset	unique_visitor_segmentation
Target	repeat_visitor
Metrics	page_count_all,session_count_anl,session_count_all,duration_anl
Visitor	visitor_id
Number of Segments to Create	4
Data Prep Include Code	wauvisit.sas
New Visitor Indicator	new_visitor
Most Current Visitor	rptdate_visitor
Run Summary in ETL?	No

17 Click **`Continue`**. A metadata table for the new segmentation appears on the right.


SAS Web Analytics Administrator
Web Analytics by SAS
Web Marts
Data
Traffic
Summaries
Scorecards
Dashboards
Segmentations
Help
SAS
Select a Web Mart (Refresh)
Standard Reports
Go
Segmentations (New) (Refresh)
Repeat Visitor - Totals (Run)
Repeat Visitor - Averages (Run)
Exseg (Run)
Modify | Delete

Segmentation Name: Exseg	
Label	Example Segmentation
Dataset	summary.unique_visitor_segmentation
Number of Segments	4.0
Data Prep Include Code	wauvisit.sas
Run Summary in ETL?	No

18 Click **+** next to the new segmentation name in the navigation tree to view the variables in the segmentation you created.

19 Execute the %WADECIDE macro to test the new segmentation report (see “Testing a Segmentation Report” on page 233).

Adding, Modifying and Deleting Metrics in a Segmentation

Add a Metric

You can add a new input metric to a segmentation that you have created. To add a metric, perform the following steps:

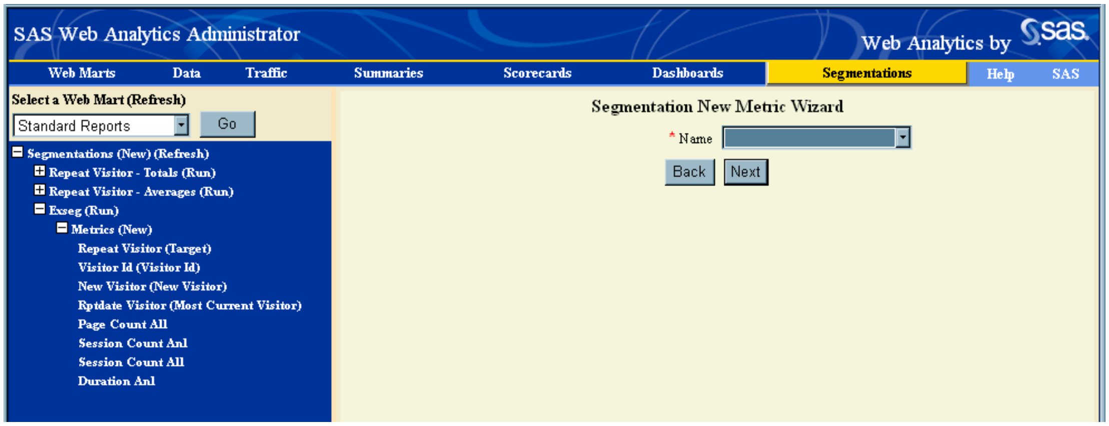
1 Open the SAS Web Analytics Administrator. Select the **`Segmentations`** link from the banner.

2 Select a Web mart (if one is not already selected) and click **`Go`**.

3 Click **+** next to a segmentation to expand the node in the navigation tree. The metrics that you specified during segmentation creation are displayed.

The node **Metrics (New)** appears below the segmentation name.

4 To add a new metric, click **(New)**. The Segmentation New Metric Wizard appears.

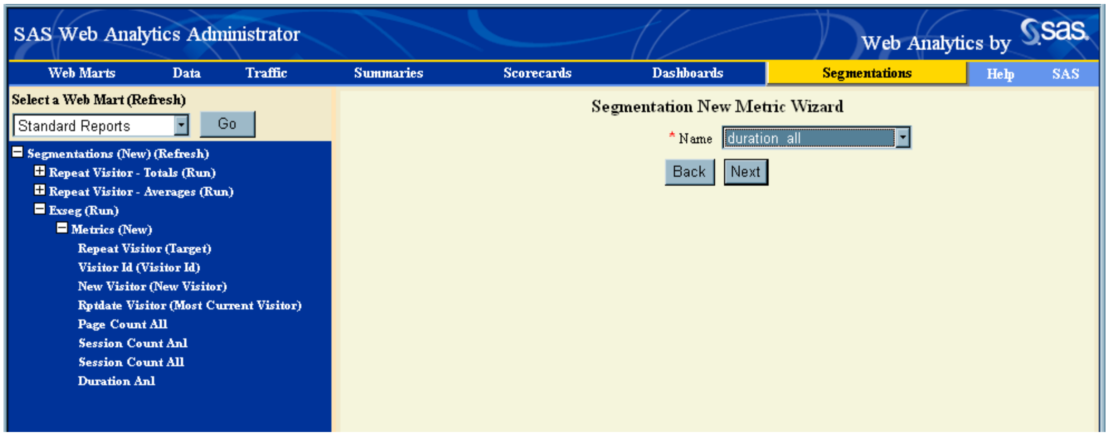


Segmentation New Metric Wizard

Select a Web Mart (Refresh): Standard Reports

* Name

5 Select a variable from the **Name** list and then click **Next**.



Segmentation New Metric Wizard

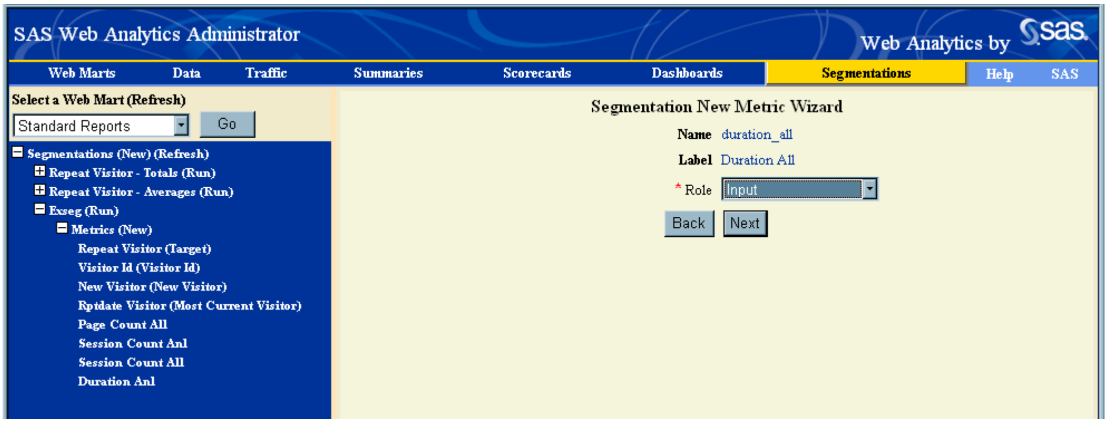
Select a Web Mart (Refresh): Standard Reports

* Name: duration_all

6 Enter a label in the **Label** field and then click **Next**.

7 Select a role for the new variable from the **Role** drop-down menu.

A role identifies how a variable is used in the segmentation analysis. For a description of all roles that are available, see "Identifying Variable Roles" on page 220. After you select a role, click **Next**.



Segmentation New Metric Wizard

Select a Web Mart (Refresh): Standard Reports

Name: duration_all

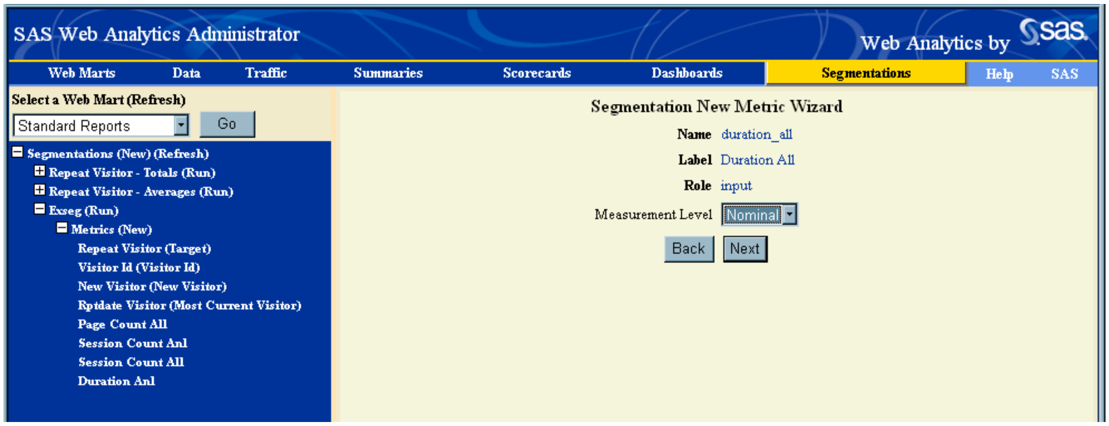
Label: Duration All

* Role: Input

8 From the **`Measurement Level`** drop-down menu, select the type of measurement level that is associated with the variable.

For a description of the measurement levels that are available, see "Selecting Metrics for a Segmentation Report" on page 220.

Click **`Next`** and then click **`Finish`**.



Segmentation New Metric Wizard

Name duration_all

Label Duration All

Role input

Measurement Level Nominal

9 Click **`Continue`** to view the new metric in the navigation tree. A metadata table appears on the right.

Modifying the Metrics in a Segmentation

Modify a Visitor-Related Metric

For visitor-related metrics, such as Repeat Visitor, New Visitor, or Visitor Id, you can modify only the label attribute. To modify a visitor-related metric, perform the following steps:

1 Open the SAS Web Analytics Administrator. Select the **`Segmentations`** link from the banner.

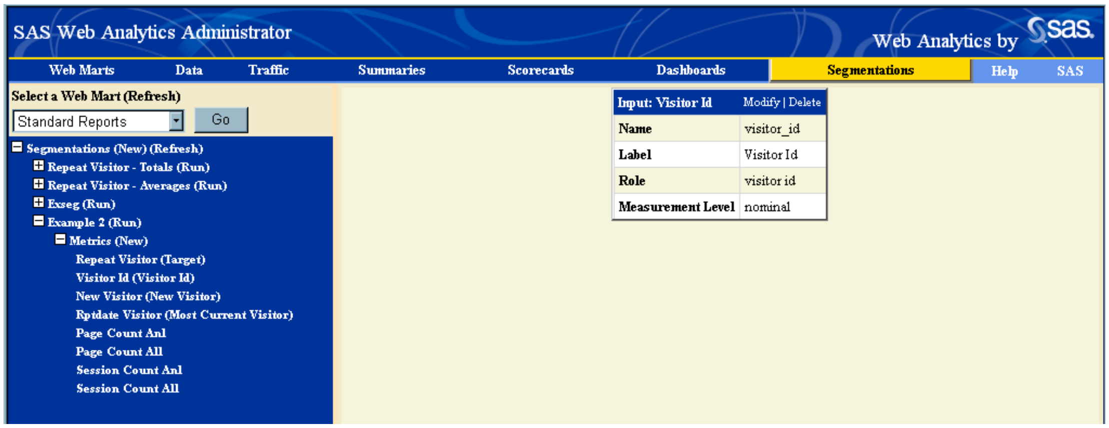
2 Select a Web mart (if one is not already selected) and click **`Go`**.

3 Click **+** next to a segmentation to expand the node in the navigation tree. The metrics that you specified during segmentation creation are displayed.

The node **`Metrics (New)`** appears below the segmentation name.

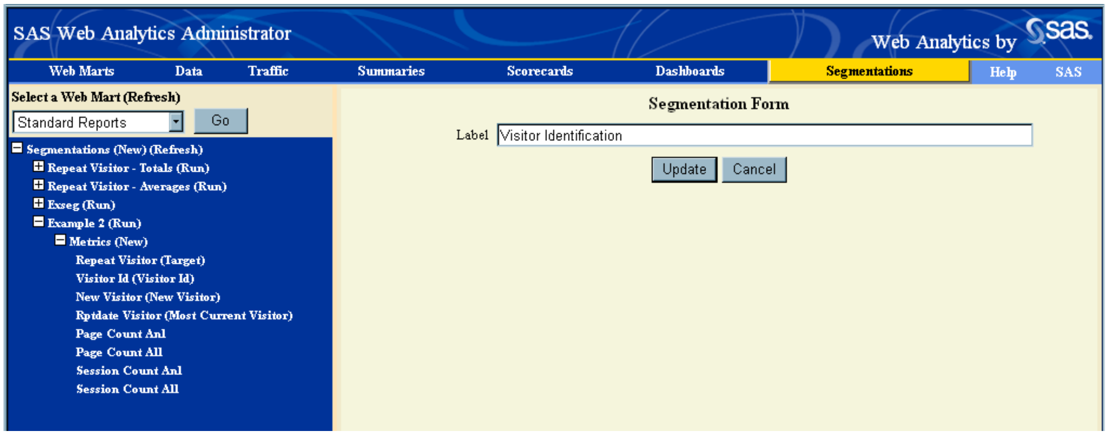
4 From the navigation tree, select the visitor-related metric that you want to modify. The metadata table appears.

5 Select **`Modify`** in the upper right section of the metadata table.



Input: Visitor Id	Modify | Delete
Name	visitor_id
Label	Visitor Id
Role	visitor id
Measurement Level	nominal

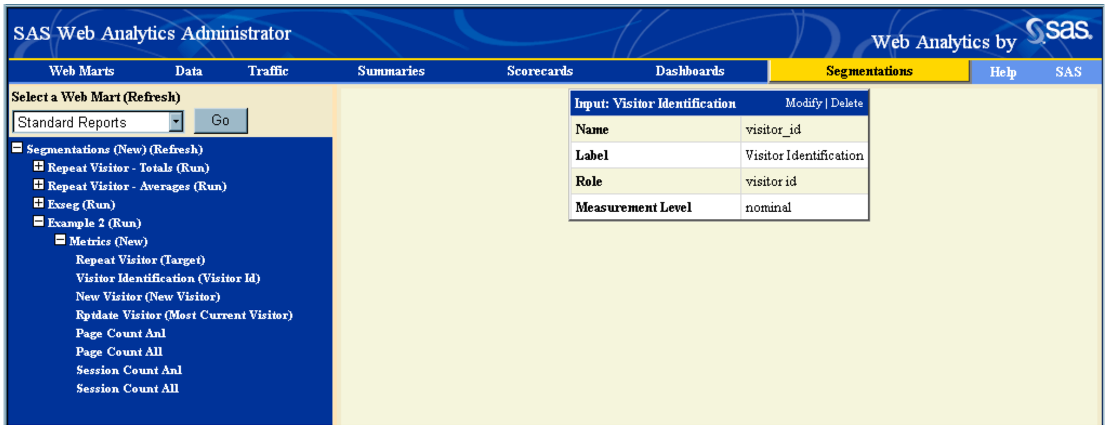
6 Enter the new label in the `Label` field and select `Update`.



Segmentation Form

Label: Visitor Identification

7 Select `Continue`. A metadata table appears that shows the name of the new label.


SAS Web Analytics Administrator
Web Analytics by sas
Web Marts
Data
Traffic
Summaries
Scorecards
Dashboards
Segmentations
Help
SAS
Select a Web Mart (Refresh)
Standard Reports
Go
Segmentations (New) (Refresh)
Repeat Visitor - Totals (Run)
Repeat Visitor - Averages (Run)
Exseg (Run)
Example 2 (Run)
Metrics (New)
Repeat Visitor (Target)
Visitor Identification (Visitor Id)
New Visitor (New Visitor)
Rptdate Visitor (Most Current Visitor)
Page Count Anl
Page Count All
Session Count Anl
Session Count All

Input: Visitor Identification	Modify | Delete
Name	visitor_id
Label	Visitor Identification
Role	visitor id
Measurement Level	nominal

Modify a Non-Visitor-Related Metric

You can modify the label and measurement level of any non-visitor-related metric in a segmentation. For information about how to change the role attribute, see “Change an Input Metric to a Target Metric” on page 230. To modify a metric, perform the following steps:

1 Open the SAS Web Analytics Administrator. Select the **`Segmentations`** link from the banner.

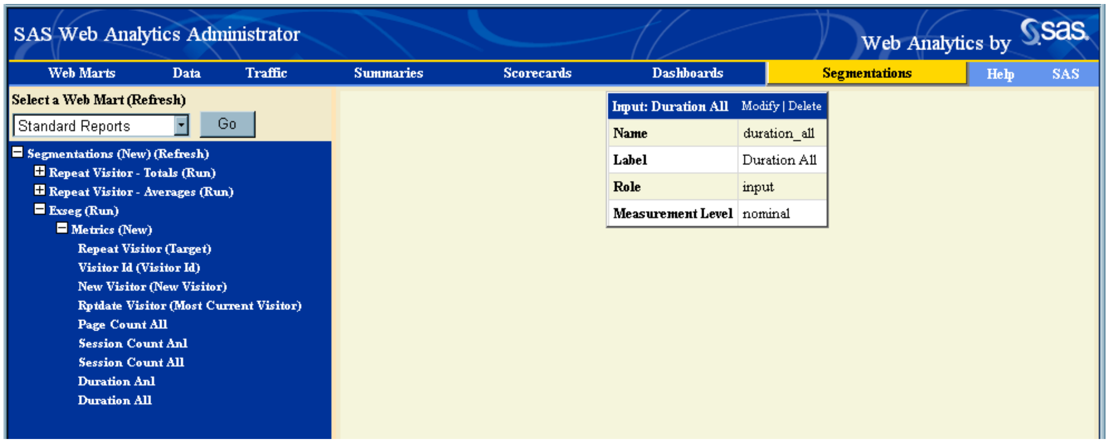
2 Select a Web mart (if one is not already selected) and click **`Go`**.

3 Click **+** next to the segmentation you of interest to expand the node in the navigation tree. The metrics that you specified during segmentation creation are displayed.

The node **`Metrics (New)`** appears below the segmentation name.

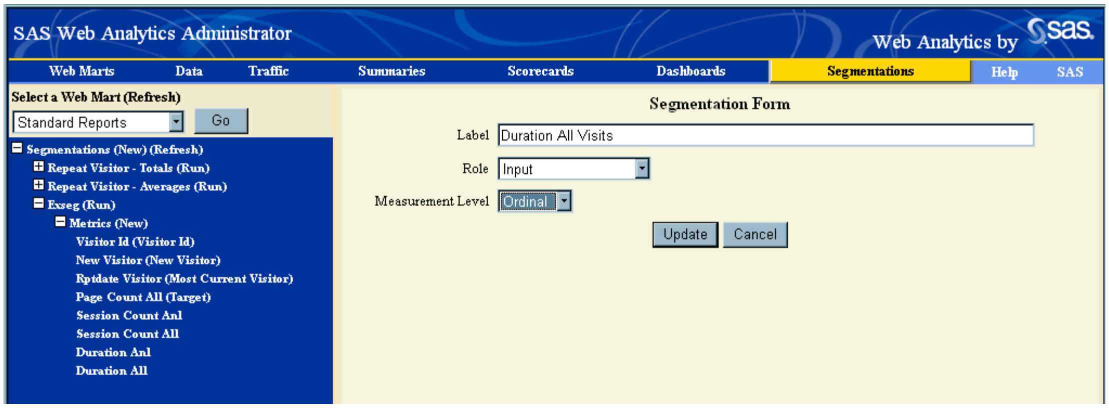
4 From the navigation tree, select the non-visitor-related metric that you want to change.

The metadata table for that metric appears.



Input: Duration All	
Name	duration_all
Label	Duration All
Role	input
Measurement Level	nominal

5 In the right corner of the table, click **`Modify`**. The Segmentation Form appears. In the **`Label`** field, you can add a new label. From the **`Role`** drop-down menu, you can select a new role for your metric. From the **`Measurement Level`** drop-down menu, you can change the measurement level of the non-visitor-related metric that you chose.



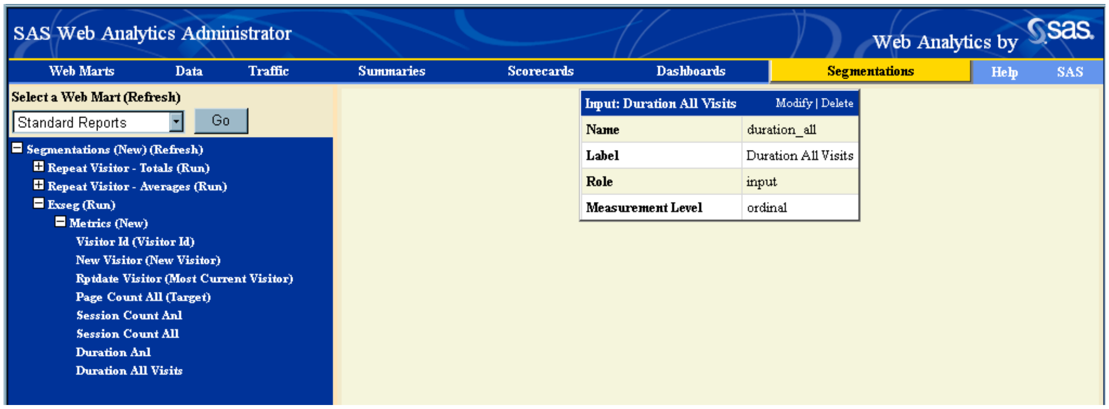
Segmentation Form

Label: Duration All Visits

Role: Input

Measurement Level: Ordinal

6 Click **`Update`** to change the properties of the metric, and then click **`Continue`**. The metadata table that appears shows the new variable attributes.



Input: Duration All Visits	
Name	duration_all
Label	Duration All Visits
Role	input
Measurement Level	ordinal

Delete a Metric

To delete an existing metric from a segmentation, perform the following steps:

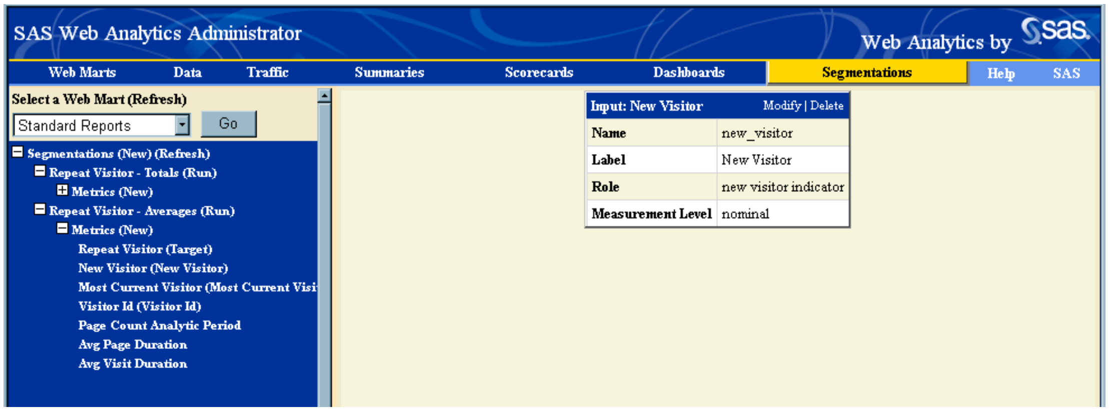
1 Select the **`Segmentations`** link from the banner in the SAS Web Analytics Administrator.

2 Select a Web mart (if one is not already selected) and click **`Go`**.

3 Click **+** next to the segmentation that has the metric you want to delete to expand the node in the navigation tree.

4 Click a metric in the navigation tree. The metadata table for the selected metric appears.

5 In the right corner of the table, click **`Delete`**.


SAS Web Analytics Administrator
Web Analytics by SAS
Web Marts
Data
Traffic
Summaries
Scorecards
Dashboards
Segmentations
Help
SAS
Select a Web Mart (Refresh)
Standard Reports
Go
Segmentations (New) (Refresh)
Repeat Visitor - Totals (Run)
Metrics (New)
Repeat Visitor - Averages (Run)
Metrics (New)
Repeat Visitor (Target)
New Visitor (New Visitor)
Most Current Visitor (Most Current Visi
Visitor Id (Visitor Id)
Page Count Analytic Period
Avg Page Duration
Avg Visit Duration
Modify | Delete

Input: New Visitor	
Name	new_visitor
Label	New Visitor
Role	new visitor indicator
Measurement Level	nominal

Change an Input Metric to a Target Metric

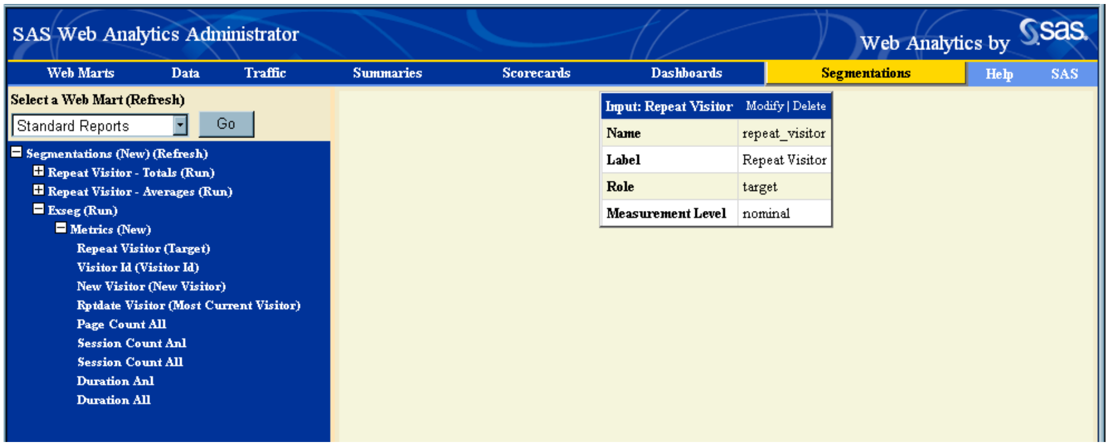
If you want to change an input metric to a target metric, you must delete the existing target metric, delete the input metric that you want to change to a target metric, and then add the input metric as the target metric. To accomplish this task, perform the following steps:

1 Select the **`Segmentations`** link from the banner in the SAS Web Analytics Administrator.

2 Select a Web mart (if one is not already selected) and click **`Go`**.

3 In the navigation tree, click **+** to expand the segmentation that contains the metric that you want to change.

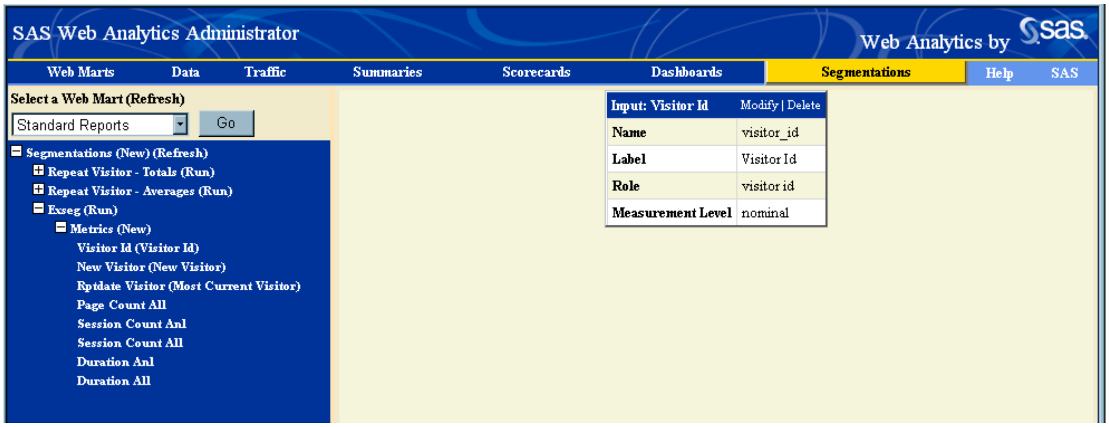
4 Click the target metric in the navigation tree to display the metadata table for the metric.



Input: Repeat Visitor	Modify | Delete
Name	repeat_visitor
Label	Repeat Visitor
Role	target
Measurement Level	nominal

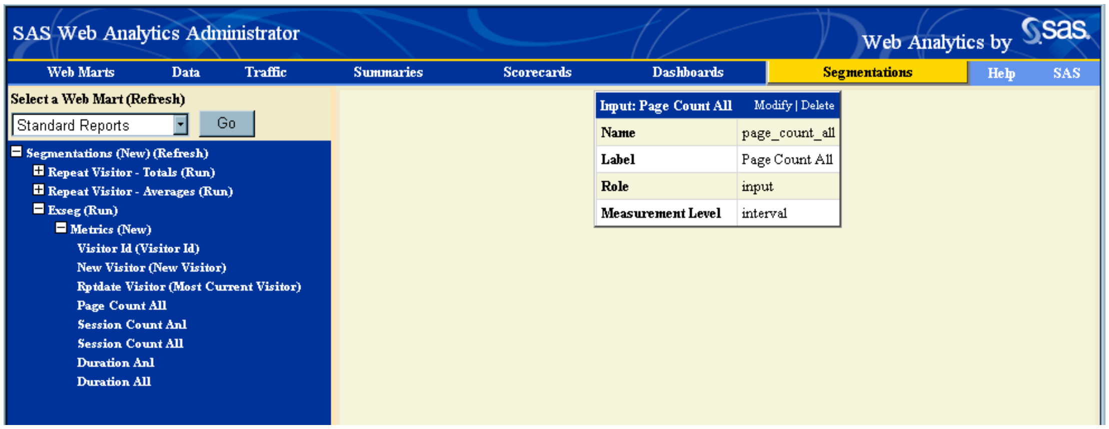
5 Select **Delete** in the upper right corner of the table to delete the target metric.

6 Click **OK** in the dialog box, and then click **Continue**. The target metric is removed from the navigation tree.



Input: Visitor Id	Modify | Delete
Name	visitor_id
Label	Visitor Id
Role	visitor id
Measurement Level	nominal

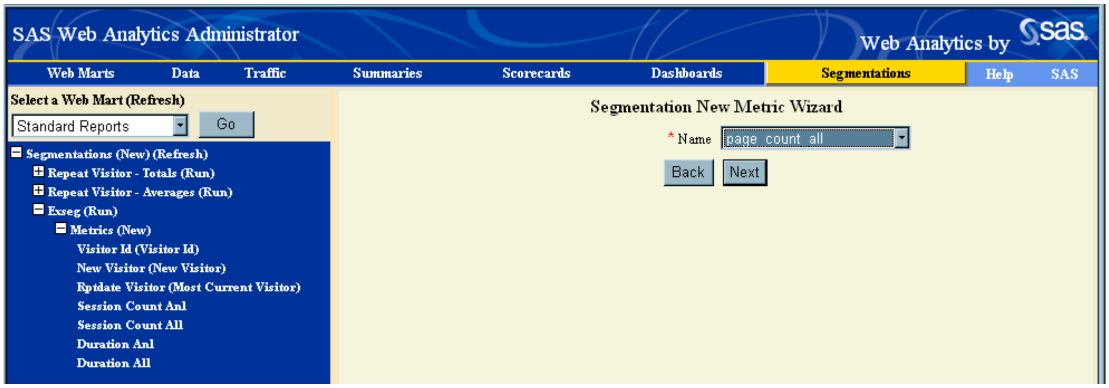
7 Click the input metric that you want to change to the new target metric. The metadata table for the selected metric appears.



Input: Page Count All	Modify | Delete
Name	page_count_all
Label	Page Count All
Role	input
Measurement Level	interval

8 Select **Delete** in the upper right corner of the table to delete the input metric.

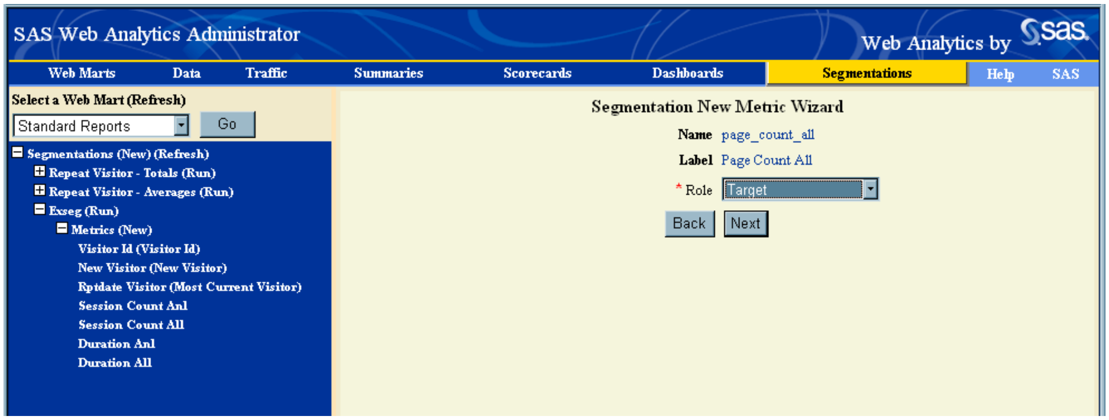
9 In the navigation tree, select **New** next to **Metrics** below the segmentation. You will re-create the metric that you deleted, but this time the former input metric will be a target metric.



Segmentation New Metric Wizard

* Name page_count_all

10 In the Segmentation New Metric Wizard that appears, provide the information for each of the prompts. Click **Next** after each entry. From the **Role** drop-down menu, be sure to select **Target**.



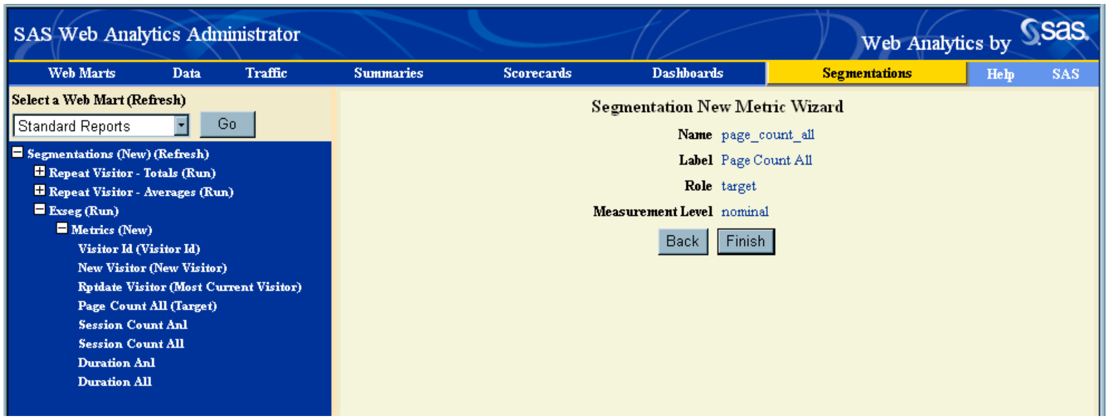
Segmentation New Metric Wizard

Name page_count_all

Label Page Count All

* Role Target

11 After you select a value for the Measurement Range, click **Next** and then click **Finish**.



Segmentation New Metric Wizard

Name page_count_all

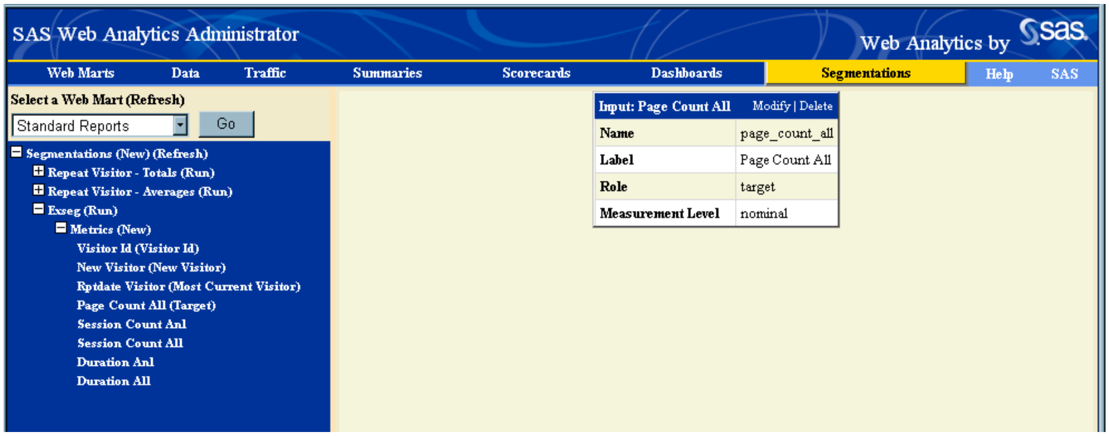
Label Page Count All

Role target

Measurement Level nominal

12 Click **Continue**. The metadata table for the newly created metric appears on the right of the page. In the navigation tree, the new metric appears as the target metric.

Note: In the table, the new metric retains the Input label because this label matches the label of the variable in the data set. △



Standard Reports

Input: Page Count All	
Name	page_count_all
Label	Page Count All
Role	target
Measurement Level	nominal

Delete a Segmentation Definition

To delete an existing segmentation definition, click the segment in the navigation tree and then click **Delete** in the upper right corner of the table that appears. Deleting a segmentation definition removes all information about that definition.

CAUTION:

Deleting a segmentation definition will "orphan" any analyses that are created with the deleted definition. △

Testing a Segmentation Report

The Mechanism for Testing a Segmentation Report

The %WADECIDE macro is the mechanism for testing a new segmentation report. The %WADECIDE macro enables you to create an analytic segmentation report without running the entire ETL process (%WAETL macro).

The following block of code is an example of using the %WADECIDE macro to test a segmentation report:

```
%wadecide(program=segmentation,
          date_1='30jun2004'd,
          swamart=c:\xxx\swamart,
          temp_store= c:\xxx\swamart\temp,
          definition_to_run=Exseg
         );
proc print data=summary.autoseg_segment_rules
```

```
                (where=(segmentation_name eq 'Exseg'));
run;
```

Explanation of a Code Block That Tests a Segmentation Report

The previous block of code is an example of how you can use the %WADECIDE macro to test a segmentation report. An explanation of the lines of code follows:

- □ Create the Exseg segmentation report. (The label for the Exseg segmentation report is Example Segment.)
- □ Use the data for June 30, 2004.
- □ The root of the SAS Web Analytics Web mart is at **`C:\xxx\swamart`**.
- □ Store the intermediate results in the **`C:\xxx\swamart\temp`** directory (the location of the Worklib library).
- □ Display the Exseg segmentation report in the output window using PROC PRINT.

See “The %WADECIDE Macro” on page 262 for more information about the arguments and the uses of the %WADECIDE macro.

Creating a Custom Segmentation Analysis

Overview of Segmentation Analysis

Segmentation analysis can be described as visitor categorization or classification using decision tree analytical techniques. The analysis defines a set of homogenous groups or segments (characteristics) for visitors, users, or customers that have performed a specific action. Segmentation analysis uses PROC ARBORETUM, a data mining procedure, to produce the analytical segments.

Note: Segmentation analysis requires a persistent visitor ID because the analysis tracks visitor action over time. Custom versions of this analysis are *not* recommended if the following conditions are true:

- □ The visitor ID for a Web site is user agent and IP address.
- □ A non-persistent cookie is used as the visitor ID.

△

The analytical segmentation reports that are implemented by default within SAS Web Analytics are examples and should be used to understand how to create segmentation analyses that are based on your business needs.

The steps for creating a custom segmentation analysis are as follows:

1. Create an analysis data set.

   Note: This data set, created with custom code, is used within the segmentation analysis. You can either run this code separately or you can run this code with the %WAETL or the %WADECIDE macros. △

2. Register a new segmentation analysis in the administrator and enable it to run during the %WAETL process.
3. Check the results of the %WADECIDE process.

Requirements for a Segmentation Analysis Data Set

Defining the Explanatory (Historical) and Response Time Periods

The explanatory (historical) and response time periods are used to determine whether a visitor can be classified as a target responder. The time periods are also used to create summary variables that are based on the historic time period that will be used in the actual segmentation analysis.

The explanatory (historical) time period begins with the minimum date and extends to the first day of the response period. For example, if the detail data has 30 days of data, and the "new" period is seven days, then the analytic period extends from day 1 to day 23.

The response time period is the number of days prior to the maximum date within the detail data. The number of days should be a value between 7 and no more than half of the number of days within the detail data set. For example, if the detail data set has 30 days of data, then the "new" period can be between 7 and 15 days.

Recommendations for Data in the Analysis Data Set

The segmentation analysis works best with a minimum of 30 days of data. The default segmentation reports are not produced until 30 days of data is accumulated and the mature detail data set by default contains 90 days of data.

The following is a general set of recommendations:

- □ minimum number of total days - 30 days
- □ minimum number of days for response period - 7 days
- □ maximum number of days for response period - 50% of total days

Selection of Detail Data in the Analysis Data Set

The analysis data set must be constructed from a data source that contains the following variables:

Table 13.3 Detail Data in the Analysis Data Set

Variable	Description
Visitor ID	Identifies a unique visitor. Depending on the data that you use, the visitor could be identified by visitor_id (created by e-Data ETL) or customer_id (if you use sales data).
Date	Defines the historical and response time periods. The variable is required for segmentation reports.

Variable	Description
Predictive variables	Specify metric variables that are used within the analysis to create segments that will predict a target variable. The following is a list of variables that you could use to predict a target: □ total sessions □ total pages □ average pages per session □ average duration per page □ entered site through e-mail campaign link
Target	Identifies an action that a visitor performs and that occurs within the historical and response time periods. The following is a list of variables that you could use as target variables: Repeat visitor — Identifies the visitor that visits the site within the historical and response time periods. Repeat buyer — Identifies the visitor that makes a purchase within the historical and response time periods.

Required Variables for the Analysis Data Set

The following variables are required for the analysis data set:

Table 13.4 Required Variables for the Analysis Data Set

Variable	Description
Visitor ID	Identifies a unique visitor, user, or customer. The analysis data set can contain only one record per visitor.
Target	Must have a value of 0 or 1: □ 0 - indicates that the visitor performed the action in the explanatory time period only. □ 1 - indicates that the visitor performed the action in both the explanatory (historical) and the response time periods.
Metric variables	Specify predictive variables that are summarized by the visitor in the explanatory (historical) time period only. All analysis variables must end in _anl. If they do not, then the second report table, which displays the proportion of visitors within each segment that is defined by the segmentation analysis, is not created.

Optional Variables for the Analysis Data Set

Optional variables display the proportion of visitors within each segment that is defined by the segmentation analysis. If these variables are not created, then this information is not displayed in the report. The following variables are optional for the analysis data set:

Table 13.5 Optional Variables for the Analysis Data Set

Variable	Description
New visitor indicator	Indicates that the visitor first entered the site during the response period. The new visitor indicator must have a value of 0 or 1: □ 0 - indicates that the visitor is not new. □ 1 - indicates that the visitor is new.
Most current visitor indicator	Indicates that a visitor entered the site for the first time on the maximum date in the detail data set. The most current visitor indicator must have a value of 0 or 1: □ 0 - indicates that the visitor did not come to the site for the first time on the maximum date. □ 1 - indicates that the visitor came to the site for the first time on the maximum date.
Totals	Specifies the following requirements: Summarization — Specifies analysis variables that are summarized by visitor over the entire date range within the detail data. Naming — Specifies that names must be identical to the analysis variables, replacing _anl with _all.

Create the Analysis Data Set

Write the custom data preparation code to create the analysis data set. You must create the analysis data set before you can define a new segmentation analysis.

Defining a New Segmentation Analysis in the Administrator

Segmentation Wizard Checklist

The segmentation wizard checklist consists of a list of required and optional parameters that you can use with your segmentation report. This information is requested by the Segmentation data entry wizard and assumes that the custom analysis data has been created. The following sections list the required and optional parameters.

Required Parameters

The following parameters are required when you use the Segmentation data entry wizard:

Table 13.6 Required Parameters for the Segmentation Data Entry Wizard

Parameter	Description
`Segmentation Name`	Specifies a unique name for this analysis. The name identifies the analysis when you run the %WADECIDE macro.
`Segmentation Label`	Specifies the label that is used in the SAS Web Analytics Report Viewer to identify the analysis.
`Dataset`	Specifies the input data set that is used within the segmentation analysis. You must create the data set before you start the definition process. *Note:* Create an empty version of the data set that contains all variables with the correct format in the appropriate library. To do this, you must provide the library name and the data set name. △
`Target`	Specifies the name of the target variable. The wizard correctly defines the measurement level for the target variable. No intervention is needed.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td><strong><code>Metrics</code></strong></td><td>Specifies predictive variables. The measurement level of each variable must be verified after you complete your entries in the Segmentation data entry wizard. The wizard assigns a measurement level to each metric that is based on whether it is numeric or character. By default, all numeric variables are assigned an interval measurement level, and all character variables are assigned a nominal measurement level. If the measurement level is not correct, then you must change the level to the appropriate type for the specific metric variable. You do this after completing the steps in the wizard.
The following measurement levels are available:
<table>
<tr><td>Interval</td><td>Use this level with numeric variables that contain the complete range of a given metric.</td></tr>
<tr><td>Nominal</td><td>Use this level with either numeric or character variables that contain a set of discrete levels.</td></tr>
<tr><td>Ordinal</td><td>Use this level with variables that contain a set of discrete levels that must be in a specific order. An ordinal variable can be either numeric or character, but must be coded as 1 to <em>n</em>.</td></tr>
</table></td></tr>
<tr><td><strong><code>Visitor</code></strong></td><td>Identifies a unique visitor, user, or customer. If the visitor ID is updated outside of the wizard, then you must assign the role to be visitor and the measurement level to be nominal.</td></tr>
<tr><td><strong><code>Number of Segments to Create</code></strong></td><td>Specifies the number of segments you want to create. The default value is 4, but you can set this number to any value greater than 0. Note that the segmentation analysis creates as many segments as are significant for the data.</td></tr>
</table>

Optional Parameters

The following parameters are optional entries when you use the Segmentation data entry wizard:

Table 13.7 Optional Parameters for the Segmentation Data Entry Wizard

Parameter	Description
`Data Prep Include Code`	Specifies custom code that can be included to create the input data set. If you use non-Web log data, then the custom code can be run separately before the %WAETL macro runs. You must enter the physical location and the name of the custom code that you want to run to create the input data set (for example, **`c:\customcode\web_analytics\segment_data_prep.sas`**). The following notes apply: □ The filename of the file that contains your custom code must have a .sas extension (for example, custom_code.sas). □ The recommended storage location is the SAS subdirectory within the Web mart. If you use the SAS subdirectory, then only the name of the file is required. See " Directory Structure" on page 4 for more information.
`New Visitor Indicator`	Specifies the name of the new visitor indicator variable. If you add the indicator outside of the wizard, then you must assign the role to be new visitor indicator, and the measurement level to be nominal.
`Most Current Visitor`	Specifies the name of the most current visitor indicator variable. If you add the indicator outside of the wizard, then you must assign the role to be most current visitor, and the measurement level to be nominal.

Preparation for Debugging the New Segmentation Definition

Before you begin to debug a segmentation definition in SAS Web Analytics, you must thoroughly test and validate your code.

Also, be sure that your code follows the naming conventions for the analysis data set. See "Required Variables for the Analysis Data Set" on page 236 and "Optional Variables for the Analysis Data Set" on page 236 for information about naming conventions.

Global Parameters That Affect Segmentation Analysis

Selecting Parameters for PROC ARBORETUM

The segmentation analysis uses PROC ARBORETUM to perform decision tree analysis. You can use the parameters that are listed in the tables in this section to fine-tune PROC ARBORETUM performance. For more information about any of the parameters, see PROC ARBORETUM documentation.

Note: Changing the parameters affects all segmentation analyses that are performed. It is recommended that the settings not be changed without testing by using the %WADECIDE macro first on all segmentation analyses that are defined. △

The following table lists the size of the training and the validation data sets. PROC ARBORETUM uses the training data set to create a decision tree, and then uses the validation data set to determine which nodes to keep. The segmentation analysis divides the input data set into training and validation data sets based on the parameters that are listed in the following table:

Table 13.8 Input Data Parameters for the Training and Validation Data Sets

Parameter	Description	Default	Min	Max
WAB_AUTOSEG_TRAIN_SAMP_PCT	Specifies the proportion of total visitor IDs that are used to build the automatic segmentation model.	0.6	0.5	0.7
WAB_AUTOSEG_VALID_SAMP_PCT	Specifies the proportion of total visitor IDs that are used to identify or validate the automatic segmentation model.	0.4	0.3	0.5

The following table lists the parameters that directly affect PROC ARBORETUM. It is recommended that the default values be kept:

Table 13.9 Parameters That Directly Affect PROC ARBORETUM

Parameter	Description	Default
WAB_AUTOSEG_ALPHA	Specifies a threshold p-value for the significance level of a candidate-splitting rule. This is applicable only for targets that have an interval measurement level.	0.20
WAB_AUTOSEG_MINWORTH	Specifies a threshold value for the logworth of a candidate-splitting rule. This is applicable only for targets that have nominal or ordinal measurement levels.	0.0
WAB_AUTOSEG_MAXDEPTH	Specifies the number of splitting rules that are needed to define the node. The root node has depth 0. The children of the root node have depth 1, and so on.	10
WAB_AUTOSEG_LEAFSIZE	Specifies the smallest number of training observations that a new branch might have.	

Parameter: WAB_AUTOSEG_ALPHA

This parameter determines cutoff points for interval-measured input variables. The decision tree uses cutoff points for determining when to issue splitting rules. For interval-measured input variables, the strength of measurement with the target value must be lower than this parameter setting in order to issue a split rule in the resulting tree. If there are many input variables that are very influential on the target variable, then you might consider setting this parameter higher than the default to avoid over-training a decision tree. Note that creating decision trees is an interactive procedure. You should perform iterative testing and inspection of the results while resetting this parameter.

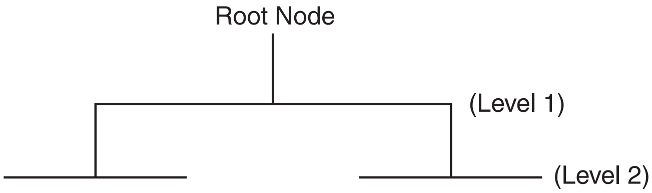
Parameter: WAB_AUTOSEG_MINWORTH

This parameter determines the worth of a non-interval-measured input variable. The algorithm considers the strength of the input with the target variable and assigns a worth statistic. In order to be considered as a splitting rule, the strength of the input variable with the target variable must exceed this parameter setting. If there are many input variables that are very influential on the target variable, then you might consider setting this parameter higher than the default to avoid over-training a decision tree. Note that creating decision trees is an interactive procedure. You should perform iterative testing and inspection of the results while resetting this parameter.

Parameter: WAB_AUTOSEG_MAXDEPTH

This parameter identifies the maximum depth that the decision tree will grow to. For example, if maximum depth is set to 2, then the resulting tree can have at most two levels off of the root node. Typically, a user will want a higher level of maximum depth if there are many explanatory variables that are believed to have a high influence on the target variable. The maximum depth decision tree has two levels, as shown in the following figure.

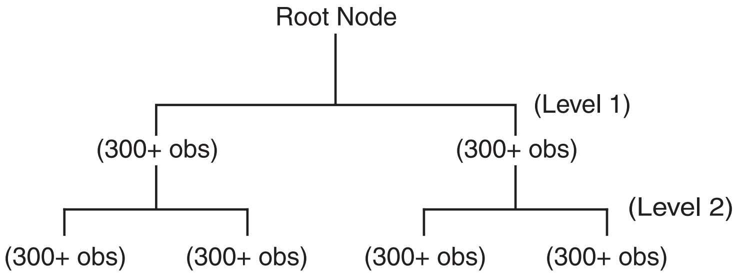
Figure 13.1 The Levels in the Maximum Depth Decision Tree



Parameter: WAB_AUTOSEG_LEAFSIZE

This parameter identifies the minimum number of observations that are needed to form any leaf on the decision tree. The parameter is usually set as a function of the input data size. A larger amount of data requires a greater leafsize to avoid over-training the decision tree. Setting leafsize too high on smaller amounts of data can result in no splitting rules. The following figure shows the relationship between leaf size and the number of observations in a data set.

Figure 13.2 Relationship Between Leaf Size and Number of Observations



Example of a Segmentation Report

Example of Data Preparation Include Code

The code that is shown in this section creates the data set that is used within the default segmentation analyses. The default segmentation analyses were created to be examples of how to create data for a segmentation analysis. The code in this section illustrates the types of logic that are required to create target and metric variables for the analysis.

The following criteria was used to create the analytical data that was used within the default segmentation analyses.

- detail data = Summary.Visitor_Day
- name of output data set = Summary.Unique_Visitor_Segmentation
- minimum number of days = 30 days
- response period definition = 15 days. If the detail data contains 30 days, then the response time period begins on day 16 and continues to the end of the 30 days. The historical time period ranges from day 1 to day 15.
- data set variables:
  - visitor_id
  - target = repeat_visitor. A repeat visitor is defined as a visitor that viewed pages in the Web site within the historical and response time periods. The value for this variable is 0 or 1.
  - metric variables
    - total sessions, session_count
    - total pages, page_count
    - total duration (minutes), duration
  - calculated metric variables
    - average pages per session, page_count_avg
    - average page duration, page_duration_avg
    - average session duration, session_duration_avg
  - analysis variable summarization time periods
    - historical - designated by the _anl suffix on metric variable names
    - total - designated by the _all suffix on metric variable names
  - indicators
    - new visitor - new_visitor. A new visitor is defined as a visitor that viewed pages within the Web site during the response time period only.
    - most current visitor indicator - rptdate_visitor. A most current visitor indicator identifies visitors that viewed pages within the Web site for the first time on the maximum date within the detail data set.

The following variables are added to the data set for debugging purposes:

last_date specifies the date of the visitor's last visit.

new_interval indicates the number of days that are specified by &new_interval.

report_date specifies &report_date.

create_date specifies the datetime stamp that identifies when the data set was created.

The following code creates the Summary.Unique_Visitor_Segmentation data set. Note that the &autoseg_report_date macro variable value is passed from the segmentation analysis module to WAUVISIT.SAS.

Code Sample for a Segmentation Report

```
%*************************************************************************;
%* Compile utility macros.                                                ;
%*************************************************************************;
%util;
%local blank;
%let blank=;

%*************************************************************************;
%*    Make sure that the data set exists.                                 ;
%*************************************************************************;
%if %sysfunc(exist(&indsn)) = 0 %then %do;
   %put %unquote(&wab_error) The &indsn data set does not exist.;
   %put %unquote(&wab_error) UNIQUE_VISITOR_SEGMENTATION cannot be created.;
   %let blank=Y;
   %goto ERREXIT;
%end;

%*************************************************************************;
%*    Make sure that &indsn can be opened.                                ;
%*************************************************************************;
%locktest(dset=&indsn);
%if &lock_sw ne 0 %then %do;
   %put %unquote(&wab_error) There was a failure opening &INDSN .;
   %put %unquote(&wab_error) UNIQUE_VISITOR_SEGMENTATION cannot be created.;
   %let blank=Y;
   %goto ERREXIT;
%end;

%*************************************************************************;
%*    Make sure that &indsn has observations.                             ;
%*************************************************************************;
%if %get_observation_count(indsn=&INDSN)=0  %then %do;
   %put %unquote(&wab_error) &INDSN contains 0 observations.;
   %put %unquote(&wab_error) UNIQUE_VISITOR_SEGMENTATION cannot be created.;
   %let blank=Y;
   %goto ERREXIT;
%end;

%*************************************************************************;
%*    Sort and subset &indsn by report_date.                              ;
%*************************************************************************;
proc sort data=&indsn
            (keep=visitor_id
                  date
                  duration
                  session_count
                  page_count)
          out =visitor_day;
   where date le &report_date ;
   by visitor_id
      date;
run;

%if &syserr gt 4 %then %do;
   %put %unquote(&wab_error) PROC SORT of &INDSN has failed.;
   %put %unquote(&wab_error) UNIQUE_VISITOR_SEGMENTATION cannot be created.;
   %let blank=Y;
   %goto ERREXIT;
%end;

%**************************************************************************;
%*    make sure that subset &indsn has observations.                       ;
%**************************************************************************;
%if %get_observation_count(indsn=visitor_day)=0  %then %do;
   %put %unquote(&wab_error) &INDSN contains 0 observations.;
   %put %unquote(&wab_error) UNIQUE_VISITOR_SEGMENTATION cannot be created.;
   %let blank=Y;
   %goto ERREXIT;
%end;

%**************************************************************************;
%*    make sure that subset &indsn has at least 30 days worth of data      ;
%**************************************************************************;
proc summary data=visitor_day nway;
  class date;
  output out=visitor_date_check;
run;

%if %get_observation_count(indsn=visitor_date_check) < &min_days  %then %do;
   %put %unquote(&wab_error) subset &INDSN contains less than &min_days days.;
   %put %unquote(&wab_error) UNIQUE_VISITOR_SEGMENTATION cannot be created.;
   %let blank=Y;
   %goto ERREXIT;
%end;

data summary.unique_visitor_segmentation;

   attrib visitor_id            label=’Visitor id’
          repeat_visitor        label=’Repeat Visitor occurs in both
                                       analysis and after period (1-yes)’
          new_visitor           label=’Visitor entering site for first
                                       time after analysis period (1-yes)’
          rptdate_visitor       label=’Visitor enters site for the first
                                       time on report date (1-yes)’
          page_count_anl        label=’Page Counts’
          page_count_all        label=’Total Page Counts’
          session_count_anl     label=’Sessions’
          session_count_all     label=’Total Sessions’
```

```
       duration_anl          label=’Duration (min)’
       duration_all          label=’Total Duration (min)’
       page_duration_avg_anl label=’Page Duration (min) Average’
       page_duration_avg_all label=’Average Page Duration (min)’
       page_count_avg_anl    label=’Page Count per Session’
       page_count_avg_all    label=’Average Page Count per session’
       session_duration_avg_anl label=’Session Duration (min) Average’
       session_duration_avg_all label=’Average Session Duration (min)’
       create_date           format=datetime21.  label=’Creation date
                                                        of data set’
      ;
format report_date first_date last_date date9.;

set visitor_day;
by visitor_id date;

retain visitor_after visitor_during rptdate_visitor
       repeat_visitor new_visitor page_duration_avg_anl
       page_duration_avg_all page_count_avg_anl page_count_avg_all
       first_date
       ;
array vars{15} visitor_after visitor_during new_visitor rptdate_visitor
              page_count_anl duration_anl session_count_anl  repeat_visitor
              page_count_all duration_all session_count_all
              page_duration_avg_anl  page_duration_avg_all
              page_count_avg_anl     page_count_avg_all
;
/* set all vars to zero when start a visitor */
if first.visitor_id then do;
  do i=1 to dim(vars);
    vars{i}=0;
  end;
  first_date=date;
end;

/* Determine if visit occured after analysis period */
if date > (&report_date - &new_interval) then do;
   visitor_after=1;
end;
/* Visit occurred during analysis period */
else do;
   visitor_during=1;
   duration_anl+duration;
   page_count_anl+page_count;
   session_count_anl+session_count;
end;

/* calculate total pages and page duration */
page_count_all+page_count;
duration_all+duration;
session_count_all+session_count;

/* define report date new visitor */
if first.visitor_id and last.visitor_id and date = &report_date then
```

```
      rptdate_visitor=1;

   if last.visitor_id then do;
      duration_anl=duration_anl/60;
      duration_all=duration_all/60;
      /* all visitors */
      if page_count_all ne 0 then do;
        page_count_avg_all=page_count_all/session_count_all;
        page_duration_avg_all=duration_all/page_count_all;
      end;
      if duration_all ne 0 then
        session_duration_avg_all=duration_all/session_count_all;
      /* analysis period metrics */
      if page_count_anl ne 0 then do;
        page_count_avg_anl=page_count_anl/session_count_anl;
        page_duration_avg_anl=duration_anl/page_count_anl;
      end;
      if duration_anl ne 0 then
        session_duration_avg_anl=duration_anl/session_count_anl;


      /* define repeat visitor */
      if visitor_during=1 and visitor_after=1 then repeat_visitor=1;
      /* define new visitor */
      if visitor_during=0 then new_visitor=1;
      last_date=date;
      new_interval=&new_interval;
      report_date=&report_date;
      create_date=datetime();
      output;
   end;

   keep visitor_id repeat_visitor new_visitor rptdate_visitor
        page_count_anl duration_anl session_count_anl
        page_count_all duration_all session_count_all
        page_duration_avg_anl  page_duration_avg_all
        page_count_avg_anl     page_count_avg_all new_interval
        session_duration_avg_all
        first_date last_date   report_date  session_duration_avg_anl
        create_date
   ;
run;

%*************************************************************************;
%* Exit point when fatal errors encountered.  Create empty data set.      ;
%*************************************************************************;
%ERREXIT:
  %if &blank=Y %then %do;
    data summary.unique_visitor_segmentation;

       attrib visitor_id            format=$96. label='Visitor id'
              repeat_visitor        format=8. label='Repeat Visitor occurs
                                                 in both analysis and
                                                 after period (1-yes)'
```

```
                new_visitor           format=8. label='Visitor entering site
                                                       for first time after
                                                       analysis period (1-yes)'
                rptdate_visitor       format=8. label='Visitor enters site for
                                                       the first time on
                                                       report date (1-yes)'
                page_count_anl        format=8. label='Page Counts'
                page_count_all        format=8. label='Total Page Counts'
                session_count_anl     format=8. label='Sessions'
                session_count_all     format=8. label='Total Sessions'
                duration_anl          format=8. label='Duration (min)'
                duration_all          format=8. label='Total Duration (min)'
                page_duration_avg_anl format=8. label='Page Duration (min) Average'
                page_duration_avg_all format=8. label='Average Page Duration (min)'
                page_count_avg_anl    format=8. label='Page Count per Session'
                page_count_avg_all    format=8. label='Average Page Count per
                                                       session'
                session_duration_avg_anl format=8. label='Session Duration (min)
                                                          Average'
                session_duration_avg_all format=8. label='Average Session
                                                          Duration (min)'
                create_date           format=datetime21.  label='Creation date
                                                                 of data set'
               ;
      run;
    %end;
%mend;
%wauvisit;


%macro wauvisit (indsn=summary.visitor_day
                 ,new_interval=15
                 ,report_date=&autoseg_report_date
                 ,min_days=30
                 );
```

CHAPTER 14

Using Stored Processes

Creating a Stored Process for a New Report

How the SAS Web Analytics Report Viewer Interacts with a Stored Process

A stored process is a SAS program that is stored on a server and can be executed as requested by client applications. Stored processes can be run interactively by the SAS Web Analytics Report Viewer. Each stored process receives a set of input macro variables from the SAS Web Analytics Report Viewer. Each stored process specifies one or more output macro variables (within the stored process) that are used by the SAS Web Analytics Report Viewer to display a report.

The SAS Web Analytics Report Viewer creates the following macro variables and passes them to the stored process:

INFO
is a Boolean value that is passed twice to the stored process. The first time that the INFO parameter is passed, the INFO parameter has a value of **`True`**. The **`True`** value in the first pass instructs the stored process to create a table that contains zero observations with all of the necessary columns, formatting, and labels for each column. The second time that the INFO parameter is passed, the INFO parameter has a value of **`False`** that instructs the stored process to populate another table.

START
contains the start date that is selected by the user in the calendar on the Traffic tab.

END
contains the end date that is selected by the user in the calendar on the Traffic tab.

LIBREFS
contains a comma-delimited list of SAS libnames (library names) that need to be defined within the stored process.

LIBPATHS
: is a comma-delimited list of locations for the corresponding LIBNAMES in the LIBREF list. The stored process uses the LIBREFS and LIBPATHS parameters to create the necessary libraries by using LIBNAME statements.

Note: The LIBPATHS and LIBREFS parameters pass information only for previously defined libraries. Therefore, verify that all libraries which are needed by the stored process that is being created are defined under **Traffic ▶ Libraries** for your webmart. See "The Traffic Page" on page 123 for more information. △

Macro variables (optional)
: are additional variables that are passed by the SAS Web Analytics Report Viewer when a filter is defined within the report definition that utilizes a stored process. Examples of additional macro variables are REQUESTED_FILE, REFERRER_DOMAIN, and STATUS_CODE.

The output of the stored process must be a single table that is displayed by the SAS Web Analytics Report Viewer through a report definition. The table must be stored in the stored process library. The stored process must pass back the OUTPUT_DATASET parameter to the SAS Web Analytics Report Viewer. The OUTPUT_DATASET parameter must contain the name of the table that the stored process will create. The table name should be specified in the following format: STP.*XXXXXX_UniqueID*, where *XXXXXX* is a string of 1 to 6 characters, and *UniqueID* is a unique ID number that is used to prevent multiple users from colliding if they access the stored process concurrently.

Writing the Code for a Stored Process

Preparation Guidelines: Writing the Code for a Stored Process

Use the following steps as guidelines as you prepare to write a stored process for a SAS Web Analytics report:

1. Determine the data sets that are used within the stored process.
2. Verify that the SAS data set libraries that the stored process will use exist in the `Libraries` node on the Traffic page.
3. Determine the path(s) to all other data sources (including the libraries that access relational database tables, computer files, and so on) within the code of the stored process.
4. Determine the final appearance of the final data set, including parameters such as the required column variables, labels, and formatting specifications.
5. Determine the error-checking code that you might need to include in the stored process program. When a stored process returns an error—for example, when a data source does not exist—the SAS Web Analytics Report Viewer displays the log file that describes the error. Add the error-checking code so that a user who encounters an error during the execution of the stored process does not have to view the red text of the error in the log file that is displayed in the SAS Web Analytics Report Viewer.

Modify the STP_Template.sas Stored Process Template

STP_Template.sas is a stored process that is provided in your Web mart's `sas` directory that can be used as a template to help you create your customized stored process. The STP_Template.sas stored process contains all of the logic needed to handle the macro variables that are passed by the SAS Web Analytics Report Viewer. The stored process template also includes the code to handle the OUTPUT_DATASET macro variable that needs to be passed back to the SAS Web Analytics Report Viewer.

To modify the STP_Template.sas stored process, open the STP_Template.sas stored process in a SAS session. After you make any modifications, be sure to save the copy of your customized stored process in the **sas** directory of your Web mart under its own name.

Note: The default stored process programs, including STP_Template.sas, should not be replaced or renamed. The default stored process programs run the interactive path analysis and funnel reports. △

In the following discussion, each step displays a section of the STP_Template.sas code, specifies the modifications (if any) that you should make to that section, and explains the purpose and use of the code in the section:

Note: In the following code, the comment blocks refer to macro variables that are defined either by the report viewer or within the macro itself. △

1 Specify a unique name for your customized macro by using the following parameter and argument:

```
%macro name_of_macro;
```

2 Add the report-defined macro variables in the following section of the template as directed in the template:

```
/*----------------------------------------------------------------------------*/
/* Stored Process Template */
/* */
/* */
/* Dependencies: */
/* The following macros need to be available: */
/* -- none -- */
/* */
/* The following macro variables need to be defined and assigned values */
/* by the User Interface/Middleware */
/* &START: start date in YYYY-MM-DD format */
/* &SEND: start date in YYYY-MM-DD format
/* &LIBREFS: a comma-delimited list of the SAS libnames that need to */
/* be defined by the process. */
/* &LIBPATHS: a comma-delimited list of the locations for the */
/* corresponding libnames in the LIBREFS list. */
/* &INFO: true or false, if true then need to create a template */
/* version of the table */
/* Add additional parameters here - example, page or search term */
/* */
/* Save Location: need to save completed program in &SWAMART/sas */
/*----------------------------------------------------------------------------*/
```

3 Do not make changes to the following section:

```
/*----------------------------------------------------------------------------*
* DO NOT CHANGE THE FOLLOWING SECTION - REQUIRED SECTION I
*----------------------------------------------------------------------------*/
%global output_dataset;
options mtrace mprint symbolgen;

%* parse the LIBREFS and LIBPATHS strings and assign necessary librefs ;
```

```
data _null_;
  length tmprefs $ 1024
         tmppaths $ 4096;

  tmprefs="&librefs";
  tmppaths="&libpaths";

  %* make sure that both lists have the same number of elements ;
  num_ref_commas=countc(tmprefs,",");
  num_path_commas=countc(tmppaths,",");

if num_ref_commas eq num_path_commas then do;
   do i=1 to (1+num_ref_commas);
      libname=scan(tmprefs,i,",");
      libpath=strip(scan(tmppaths,i,","));
      rc = filename("fileref",libpath);
      did = dopen("fileref");
      if did then do;
         rc=dclose(did);
         rc = libname(strip(libname), libpath);
         call symput("_wab_lib_error_","0");
      end;
      else do;
         call symput("_wab_lib_error_","2");
         call symput("_wab_lib_msg_"
                     ,"The physical location "||strip(libpath)
                     ||" either does not exist or cannot be "
                     ||"opened. The library "||strip(libname)
                     ||" must be respecified with a valid location.");
         end;
      end;
  end;
  else do;
    call symput("_wab_lib_error_","3");
    call symput("_wab_lib_msg_","You have different numbers of "
                              ||"library references than physical paths."
                              ||"Please respecify your libraries.");
  end;
run;

options noquotelenmax; /* avoid warning message about unbalanced quotes */
data _null_;
  start=compress("&start");
  if start in ("","$start") then call symput("startchk","N");
  else call symput("startchk","Y");
run;
```

4 You do not need to modify the following section. The START and END parameters are defined when the stored process is registered and are passed by the SAS Web Analytics Report Viewer in YYYY-MM-DD format. The following section of code converts the dates to SAS dates and determines all dates that are specified by the DATE1-DATEd macro variables. The section also creates the macro variable D, which is the total number of dates between the start and end date.

```
%if &startchk = Y %then %do;
    data _null_;
     start="&start";
     end="&end";
     start_date=input(start,yymmdd10.);
     end_date=input(end,yymmdd10.);

     i = 0;
     do date = start_date to end_date;
       i=i+1;
       call symput(compress("date"||put(i,3.)),compress(put(date,yymmdd10.),"- "));
       if date = end_date then call symput("d",compress(put(i,best.)));
     end;
    run;
  %end;

  /*-----------------------------------------------------------------*
  *  The stored process handler with SWA will first read a
  *  "template" of a STP data set and then actually create the data.
  *  When the STP is used to create a report only
  *-----------------------------------------------------------------*/
  data _null_;
    if strip(lowcase(compress("&info")))="true" then
      call symput("info","TRUE");
    else
      call symput("info","FALSE");
  run;

  /*-------------------------------------------------------------------------------*
  * END REQUIRED SECTION - I
  *--------------------------------------------------------------------------------*/
```

5 Modify the following section according to the instructions within the comments in the code.

The following section of code performs a global replacement of XXXXXX to a unique ID string that is appended to the data set names. Unique IDs prevent users who concurrently access a stored process from interfering with each other.

```
/*-------------------------------------------------------------------------------*
  * CREATE TABLE TEMPLATE: REQUIRED SECTION - II
  * Need to create a 0 observation data set that contains all of the variables in
  * the final data set.  Need to include formatting and labels - these will be
  * used when a report is created from a STP.
  * !!! Any place XXXXXX occurs this means a STP specific name is required !!!!!
  *--------------------------------------------------------------------------------*/
  %if &info = TRUE %then %do;
      data _null_;
      %* create a unique id for the dsn that will contain the report template;
      call symput("_XXXXXX_data_uid_"
                 ,compress(
                           translate(   strip(put(datetime(),15.))
                                     || "_"
                                     || strip(put(ranuni(-1),13.11))
                                    ," "
                                    ,"."
```

```
                                )
                      )
             );
run;
```

6 Follow the instructions within the comments in the following section for the naming conventions to use. In the FORMAT statement, be sure to list the variables in the correct order. Be sure to specify the variable labels. The code replaces ZZZZZZ with a unique prefix.

```
/*-------------------------------------------------------------------*
    * ZZZZZZ - - prefix appropriate for the data set between 1 and 6
    * characters the unique id will take up the rest of the 32 chars
    *----------------------------------------------------------------*/
    data stp.ZZZZZZ_&_XXXXXX_data_uid_;
      format /* Fill in with columns created in STP */
      ;
      label  /* Fill in column labels */
      ;
    run;

    %let outdsn=stp.ZZZZZZ_&_XXXXXX_data_uid_;

    %goto done;
  %end;
  /*---------------------------------------------------------------*
  * END TABLE TEMPLATE:  REQUIRED SECTION - II
  *---------------------------------------------------------------*/

  /*---------------------------------------------------------------*
  * CREATE TABLE: REQUIRED SECTION - III
  * !!Any place XXXXXX or ZZZZZZ occurs this means a STP specific
  *  name is required!!
  *----------------------------------------------------------------*
   data _null_;
     %* create a unique id for the dsn that will contain the report template  ;
     %* replace XXXXXX with text appropriate for the STP                    ;
     call symput("_XXXXXX_data_uid_"
                ,compress(
                          translate(   strip(put(datetime(),15.))
                                    || "_"
                                    || strip(put(ranuni(-1),13.11))
                                   ," "
                                   ,"."
                                   )
                         )
               );
   run;
/*------------------------------------------------------------------*
   * Put custom code here - all data sets created MUST be stored in
   * WORK.  Data sets should be named as follows
   * ZZZZZZ - prefix appropriate for the data set between 1 and 6
   * characters the unique id will take up the rest of the 32 chars
   * Data set names MUST have unique identifier or multiple users can
   * tromp on each other.
```

```
*---------------------------------------------------------------------*/

/*---------------------------------------------------------------------*
* Start data check logic: Put logic here to determine if data exists
* for dates selected if not then need to create output data set with
* zero observations. This section is required because the calendar
* will allow selection of dates with no data
*---------------------------------------------------------------------*/
```

7 After the section of the template that begins with the comment, “Start data check logic” (see the previous code comment), place code to check for the existence of data within the selected start and end dates, and to perform any other data checks that are appropriate. If no data exists, then include the code to create the output data set with zero observations. The following examples show code that you might use.

Example 1 For detail or session data, check for data by using code similar to the following code:

```
proc sql;
   create table check_dsn_&_XXXXXX_data_uid_ as
      select memname,
             nobs
      from dictionary.tables
         where libname eq 'DATED' and
               upcase(memname) in (
                  %do i = 1 %to &d
                     "SESSION_&&date&i"
                  %end;
                                    )
   ;
quit;

%if %get_observation_count(indsn=check_dsn_&_XXXXXX_data_uid_) ne &d %then %do;
   /* create obs output data set */
%end;
```

Example 2 For a SUMMARY data set, check for data by using code that is similar to the following code:

```
Data check_data_&_XXXXXX_data_uid_;
  Set summary.&input_dsn ;
    Where date between &date1 and &&date&n
Run;
```

Example 3 If necessary, modify the following code segment to determine whether the Check_Data data set has any observations. This code uses a utility macro (%GET_OBSERVATION_COUNT) that is available within the SAS Web Analytics application.

```
%if %get_observation_count(indsn= check_data_&_XXXXXX_data_uid_) eq 0 %then %do;
        /* create obs output data set */
   %end;
```

8 Place the following comments after the code segments that you modified in the previous step.

```
/*---------------------------------------------------------------------*
 * End data check logic:
```

```
 *------------------------------------------------------------------*/

/*------------------------------------------------------------------*
 * Start report logic
 *------------------------------------------------------------------*/
```

9 Create the code that checks for situations that would cause errors, in order to prevent the user from having to view a log (displayed in the SAS Web Analytics Report Viewer) if the stored process program terminates with errors. Some typical scenarios for this type of error-checking include the cases in which an intermediate data set has zero observations or a business rule is not met.

10 Follow the instructions in the following section.

Note: Be sure to uncomment the last comment and specify the new name of your customized stored process. △

The following section contains the main body of code that creates the output data set:

```
/*------------------------------------------------------------------*
 * End report logic
 *------------------------------------------------------------------*/

/*------------------------------------------------------------------*
 * Final data set creation --- make sure that all variables have
 * appropriate formats and labels.
 *------------------------------------------------------------------*/
 data stp.ZZZZZZ_&_XXXXXX_data_uid_;
   format   /* Fill in with columns created in STP */
   ;
   set work.ZZZZZZ_&_XXXXXX_data_uid_
   ;
   label    /* Fill in column labels */
   ;
 run;

 %let outdsn=stp.ZZZZZZ_&_XXXXXX_data_uid_;

 /*------------------------------------------------------------------*
 * End Final data set creation
 *------------------------------------------------------------------*/

 /*------------------------------------------------------------------*
 * Start data set clean-up
 *------------------------------------------------------------------*/
 %DONE:
  %if %upcase(&info) ne TRUE %then %do;

      proc datasets library=work nolist;
        delete

        /* Need to clean up work data sets */

        ;
      quit;
```

```
        proc datasets library=STP nolist;
          delete


          /* Need to clean up STP data set */


          ;
        quit;

  /*------------------------------------------------------------------*
   * End data set clean-up
   *------------------------------------------------------------------*/

    %end;


  %put *****OUTPUT PARAMETERS*****;
  %put stp:output_dataset=&outdsn
  %let     output_dataset=&outdsn

%mend;
/* Put macro call here and uncomment.
%name_of_macro;
*/
```

11 Save the new stored process in the **sas** directory of your Web mart. The stored process must have the file extension .sas.

Add a Created Stored Process

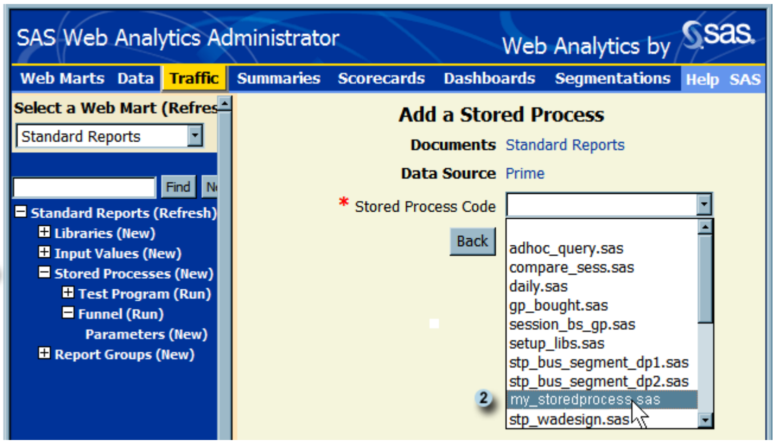
After you have have written (or identified) a stored process that you want to use to define a new report, you next add the new stored process in the SAS Web Analytics Administrator. To add the stored process (which you have placed in the **sas** directory of your Web mart) for a report definition, perform the following steps:

1 In the navigation tree of the Traffic page of the SAS Web Analytics Administrator, click **New** next to **Stored Processes** (❶ in the following display). The Add a Stored Process wizard appears in the right panel.

2 Select your stored process name from the drop-down menu (❷ in the following display) for the **Stored Process Code**.

Note: If you do not see your customized stored process in the drop-down menu, then make sure that the stored process file is located in the **sas** directory of your Web mart. If you have verified that the stored process is currently located in the **sas** directory of your Web mart, then make sure that the stored process filename has a .sas extension. △

Display 14.1 Selecting Your Customized Stored Process in the Add a Stored Process Wizard



Documents Standard Reports

Data Source Prime

* Stored Process Code

3 In the **`Enter Label`** field, type a name for the report that will be created from the output of the stored process. The text that you enter here appears as a report name below **`Stored Processes`** in the navigation tree of the Traffic page of the SAS Web Analytics Administrator.

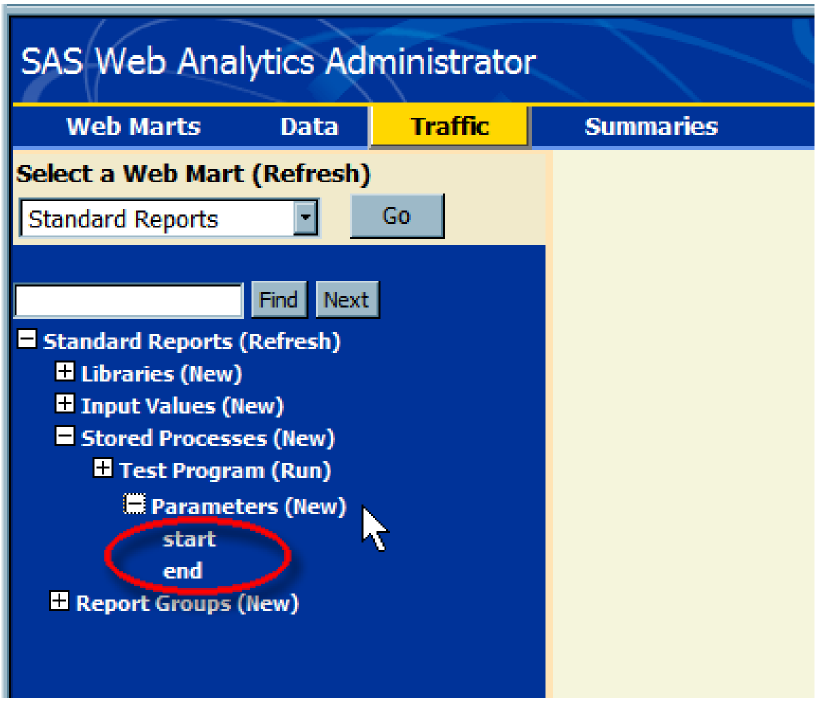
After the stored process is run successfully, the label is displayed as a report name on the Traffic page in the SAS Web Analytics Report Viewer.

4 Add the required $START and $END parameters in order to pass the starting and ending dates from the calendar (in the SAS Web Analytics Report Viewer) to the stored process.

To add the $START parameter to the stored process definition, perform the following steps:

- **a** Expand the node of the new stored process in the SAS Web Analytics Administrator.
- **b** Click **`New`** next to **`Parameters`**. The Add a Parameter wizard appears in the right panel.
- **c** Type **`start`** in the **`Enter Name`** field. Click **`Next`**.
- **d** In the **`Enter Value`** field, type **`$start`**. Click **`Next`**. Click **`Finish`**.
- **e** Click **`Add Another Parameter`** to add the $END parameter.
- **f** Repeat the previous steps for adding the $START parameter, except type **`end`** in the **`Enter Name`** field, and type **`$end`** in the **`Enter Value`** field. Click **`Finish`**, and click **`Done`**.
- **g** Refresh the Traffic page to view the new parameters for the stored process. The following display shows how the the $START and $END parameters are listed in the SAS Web Analytics Administrator for a stored process.

Display 14.2 The Location of the Parameters for a Stored Process



5 You can also create optional parameters to send information to the stored process. An optional parameter can have either an absolute value or can enable a user to enter a value.

Specify a new parameter for a stored process by following the previous steps for adding the $START parameter, with the following variation:

□ In the **`Enter Name`** field, type a name for the new input macro variable.

□ In the **`Enter Value`** field, type either a value or the name of the input macro variable, preceded by a $.

APPENDIX 1

Macro Reference for the SAS Web Analytics 5.2 Solution

Overview: Macros for SAS Web Analytics 5.2

This chapter is a reference for the major macros that are used in SAS Web Analytics 5.2. Arguments are not case-sensitive, unless specified as such.

The SAS Web Analytics solution provides the following macros:

- □ "The %WADECIDE Macro" on page 262
- □ "The %WAETL Macro" on page 270
- □ "The %WASUMTST Macro" on page 274
- □ "The %WAPATHDP Macro" on page 280

The %WADECIDE Macro

Purpose of the %WADECIDE Macro

The %WADECIDE macro creates decision support reports (scorecard, dashboard, and segmentation reports) without requiring you to execute the entire SAS Web Analytics ETL process (by using the %WAETL macro).

Aside from the obvious savings in time by avoiding the execution of the %WAETL macro when you create decision support reports, the %WADECIDE macro enables you to specify a separate location (such as a test library) for storing provisional scorecards and dashboards instead of in the summary directory, where the %WAETL macro stores scorecards and dashboards by default. If you isolate the test versions from the production versions of the reports, then you can identify and manage your reports more efficiently.

You can use the %WADECIDE macro to create decision support reports in order to use information from a customized data source. For example, you might set up a data management program that merges supplemental customer information from an external data source with the existing SAS Web Analytics data sets. After the %WAETL macro has finished, you then execute this data management program in your ETL process (such as Daily.sas). Then, you execute the %WADECIDE macro to create a decision support report that uses this special customer data.

The %WADECIDE macro enables you to populate a newly defined decision support report with the results of previous ETL processes. For example, if you define a scorecard, but you want to display the results as though the new scorecard had existed for a while, then you would invoke the %WADECIDE macro.

The %WADECIDE Macro: Syntax

Note: If an argument is enclosed in <>, then the argument is optional. △

%WADECIDE(
  PROGRAM=*DASHBOARD* | *SCORECARD* | *SEGMENTATION,*
  DATE_1=*SASdatevalue,*
  <**DATE_2=***SASdatevalue,*>
  SWAMART=*path-to-mart,*
  <**TEMP_STORE=***path-to-tempstore,*>
  <**STP_SW=***YES* | *NO,*>
  <**INDSN=***dataset-name,*>
  <**OUTLIB=***libref-name,*>
  <**USE_FIRST_TWO_WEEKS=***YES* | *NO,*>
  <**DEFINITION_TO_RUN=***decision-support-report-name(s)*>
  );

The %WADECIDE Macro: Details

PROGRAM=*DASHBOARD* | *SCORECARD* | *SEGMENTATION*

specifies the type of decision support report that you want to create. You create only one type of decision support report with each invocation of the %WADECIDE macro. If you want to create more than one type of report, then you will need to additionally invoke the %WADECIDE macro for each type of report that you want to create.

DATE_1=*SASdatevalue*

specifies the first date in your request, expressed as a SAS date constant (for example, `'01jan2004'd`).

If PROGRAM=DASHBOARD or PROGRAM=SCORECARD, then DATE_1 is the first date in the date range for the report.

If PROGRAM=SEGMENTATION, then DATE_1 is the date for the report.

If you execute the %WADECIDE macro immediately after running the %WAETL macro, then you can use the &_WAB_MIN_PROCESS_DATE_ global macro variable that %WAETL has created to designate the first date in your request.

DATE_2=*SASdatevalue*

specifies the second date in your request, expressed as a SAS date constant (for example, `'01jan2004'd`). For information about how to format SAS date constants, see SAS Help and Documentation. If PROGRAM=DASHBOARD or PROGRAM=SCORECARD, then DATE_2 is the last date in the date range for the report. If you do not specify DATE_2, then DATE_1 is used as the default value for this argument.

If PROGRAM=SEGMENTATION, then DATE_2 is ignored if it has been specified. If you execute the %WADECIDE macro immediately after running the %WAETL macro, then you can use the &_WAB_MAX_PROCESS_DATE_ global macro variable that the %WAETL macro has created to designate the last date in your request.

SWAMART=*path-to-mart*

specifies the path to the root directory for the designated SAS Web Analytics Web mart. The %WADECIDE macro uses this information to locate the data resources that are necessary to create the decision support report(s) that you requested. For example, if your SAS Web Analytics Web mart is located at `C:\xxx\swamart` and its subdirectories, then the root directory is `C:\xxx\swamart`.

TEMP_STORE=*path-to-tempstore*

specifies the path to a directory that will act as your Worklib library, where the %WADECIDE macro will store the intermediate data sets that it creates during the process of building a decision support report. You will need to create this directory so that the path exists before you run the %WADECIDE macro (it is not automatically created for you). Specify an argument to the TEMP_STORE parameter if you want to access these intermediate data sets after the %WADECIDE macro has finished.

If you do not specify this argument, then the %WADECIDE macro creates its own temporary Worklib library, which exists only as long as the SAS session from which you invoke the %WADECIDE macro.

STP_SW=*YES | NO*

specifies whether the %WADECIDE macro is being invoked from a stored process. If you write a customized stored process that invokes the %WADECIDE macro, then specify YES. Otherwise, you can omit the STP_SW parameter altogether in your invocations of the %WADECIDE macro, because NO is the default argument for the STP_SW parameter.

INDSN=*dataset-name*

specifies the LIBNAME.MEMNAME for the data set that is used to create the decision support report.

If you specified PROGRAM=DASHBOARD or PROGRAM=SCORECARD, then the %WADECIDE macro uses this information.

If you specified PROGRAM=SEGMENTATION, then the %WADECIDE macro ignores this information.

If you do not specify an argument for the INDSN parameter, and if PROGRAM=DASHBOARD, then the %WADECIDE macro uses the LIBNAME.MEMNAME that is stored in the INDSN field in the Config.Wadshbrd data set.

If you do not specify an argument for the INDSN parameter and if PROGRAM=SCORECARD, then the %WADECIDE macro uses the LIBNAME.MEMNAME that is stored in the INDSN field in the Config.Wascrcrd data set.

OUTLIB=*libref-name*

specifies the libref (library reference name) where you want the %WADECIDE macro to store its decision support report.

If you specified PROGRAM=DASHBOARD or PROGRAM=SCORECARD, then the %WADECIDE macro uses this information.

If you specified PROGRAM=SEGMENTATION, then the %WADECIDE macro ignores this information.

If you do not specify an argument for the OUTLIB parameter, then the %WADECIDE macro stores the decision support report in the Summary library.

Note: Do not specify an argument for the OUTLIB parameter if you are running a scorecard or dashboard for a day or a range of days that you want to access in the SAS Web Analytics Report Viewer. △

To store the scorecard or dashboard in a location that is different from the default location (for example, to store separate test versions of scorecards or dashboards) then specify an alternate library using this argument.

Note: You will need to create the alternate library before you run the %WADECIDE macro (the library is not automatically created for you). △

USE_FIRST_TWO_WEEKS=*YES | NO*

specifies whether to use the first 14 days of data to create the decision support report.

If you specified PROGRAM=SCORECARD, then the %WADECIDE macro uses this information.

If you have specified PROGRAM=DASHBOARD or PROGRAM=SEGMENTATION, then the %WADECIDE macro ignores this information.

Code this information as specified by the type of data in your scorecard. Specify USE_FIRST_TWO_WEEKS=NO if the target variable for your scorecard has measures that need some accumulated history (for example, repeat visitors). You can create a model with a better fit by excluding the data for the first 14 days.

The default argument is YES, so if you do not specify an argument for the USE_FIRST_TWO_WEEKS parameter, then the %WADECIDE macro includes the data for the first 14 days.

DEFINITION_TO_RUN=*decision-support-report-name(s)*

specifies how many of the different definitions of the selected report type that you want to run.

The following values are valid:

Blank
: The %WADECIDE macro creates all the decision support reports for the designated PROGRAM=*report type*.

ALL
: The %WADECIDE macro creates all the decision support reports for the designated PROGRAM=*report type* (same as blank).

ETL
: the %WADECIDE macro creates all the decision support reports for the designated PROGRAM=*report type* whose ETL_RUN_SW is set to **1**.

_NON_ETL_
: The %WADECIDE macro creates all the decision support reports for the designated PROGRAM=*report type* whose ETL_RUN_SW is set to **0**.

%BQUOTE
: specified as %BQUOTE(*comma-delimited list of report names*). The %WADECIDE macro creates each decision support report for the designated PROGRAM=*report type* whose name is in the comma-delimited list. Because commas separate the parameters in a macro invocation, you need to embed your list in the %BQUOTE function so that the reports in your list are not mistaken for additional parameters in the %WADECIDE macro instead of being recognized as elements in the argument for the DEFINITION_TO_RUN parameter.

If you specify only one report, then you do not need to use the %BQUOTE function.

Note: The label(s) that you supply in this list must exactly match, in both case and spelling, the label(s) that are associated with the corresponding decision support report in the metadata. △

For example, if you have defined 10 different experimental scorecards in the metadata, then you might designate the scorecards to be created through the non-ETL processing until you obtain the desired output. To avoid executing each of these 10 scorecards separately, use the DEFINITION_TO_RUN parameter to specify a value of _NON_ETL_ in order to instruct the %WADECIDE macro to run all the non-ETL scorecard definitions.

The following tables summarize how to specify the argument(s) for each type of decision report when you plan to run the %WADECIDE macro:

Table A1.1 How to Specify Arguments in the %WADECIDE Macro for a Dashboard Report

Parameter	**Argument for a Dashboard Report**
PROGRAM	DASHBOARD
DATE_1	*'ddMMMyy'd*
DATE_2	*'ddMMMyy'd*
SWAMART	*C:\path-to-mart*
TEMP_STORE	*C:\path-to-tempstore*
STP_SW	Not applicable
INDSN	Summary.Daily_Total_Day
OUTLIB	*libref-name*
USE_FIRST_TWO_WEEKS	YES
DEFINITION_TO_RUN	%BQUOTE(*Dashboard Name1, Dashboard Name2*) *NOTE:* The value(s) must exactly match, in both case and spelling, the label(s) that are associated with the corresponding dashboard(s) in the metadata. △

Table A1.2 How to Specify Arguments in the %WADECIDE Macro for a Scorecard Report

Parameter	**Argument for a Scorecard Report**
PROGRAM	SCORECARD
DATE_1	*'ddMMMyy'd*
DATE_2	*'ddMMMyy'd*
SWAMART	*C:\path-to-mart*
TEMP_STORE	*C:\path-to-tempstore*
STP_SW	NO
INDSN	Summary.Daily_Total_Day
OUTLIB	*libref-name*
USE_FIRST_TWO_WEEKS	Either YES or NO
DEFINITION_TO_RUN	*Scorecard Name* *NOTE:* The value(s) must exactly match, in both case and spelling, the label(s) that are associated with the corresponding scorecard(s) in the metadata. △

Table A1.3 How to Specify Arguments in the %WADECIDE Macro for a Segmentation Report

Parameter	**Argument for a Segmentation Report**
PROGRAM	SEGMENTATION
DATE_1	*'ddMMMyy'd*
DATE_2	Not applicable
SWAMART	*C:\path-to-mart*

Parameter	Argument for a Segmentation Report
TEMP_STORE	*C:\path-to-tempstore*
STP_SW	NO
INDSN	Not applicable
OUTLIB	Not applicable
USE_FIRST_TWO_WEEKS	Not applicable
DEFINITION_TO_RUN	*Segmentation Name* *NOTE:* The value(s) must exactly match, in both case and spelling, the label(s) that are associated with the corresponding segmentation report(s) in the metadata. △

Identifying Errors in the Log

The %WADECIDE macro checks for errors on the arguments that you pass to it and writes an error message in the log if it detects an error. The format of an error message follows:

```
ERROR:(WADECIDE) hh:mm:ss [description of error]
```

The name of the macro in which the error occurred is specified in the parentheses. If the text in the parentheses is not WADECIDE, then the error did not occur in the %WADECIDE macro, but in the macro that creates the decision support report.

The time when the %WADECIDE macro encountered the error is designated by *hh:mm:ss*.

The text between the brackets indicates the type of problem that the %WADECIDE macro encountered.

Selecting Dates for Viewing Decision Support Reports

The SAS Web Analytics Report Viewer enables you to view the decision support reports that have been created. Select a date in the left panel of the SAS Web Analytics Report Viewer to view the results of reports for that date. The %WADECIDE macro automatically updates the list of selectable dates for the decision support reports.

The %WADECIDE MACRO: Examples

Example 1

The following code creates the Segmentation report for 5/28/2004 for the SAS Web Analytics Web mart whose root is located in **C:\xxx\swamart**:

```
%wadecide(
      program=segmentation,
      date_1='28may2004'd,
      swamart=C:\xxx\swamart
      );
```

Example 2

The following code creates the Temp Dev Dashboard Report for the date interval 5/1/2004 through 5/31/2004 for the SAS Web Analytics Web mart whose root is located in **`C:\xxx\swamart`**. The intermediate data sets will be stored in the Worklib library that is located in the **`C:\xxx\swamart\TempStore`** directory.

```
%wadecide(
      program=dashboard,
      date_1='01may2004'd,
      date_2='31may2004'd,
      swamart=C:\xxx\swamart,
      temp_store= C:\xxx\swamart\TempStore,
      definition_to_run=Temp Dev
      );
```

Example 3

The following code creates all scorecard reports for 6/1/2004 for the SAS Web Analytics Web mart whose root is located in **`C:\xxx\swamart`**. Devlib.Testdata is the data source. The code instructs the macro to create the reports in the Devlib library. The report excludes the data from the first 14 days.

```
%wadecide(
      program=scorecard,
      date_1='01jun2004'd,
      swamart=C:\xxx\swamart,
      indsn=devlib.testdata,
      outlib=devlib,
      use_first_two_weeks=no
      );
```

Example 4

This example demonstrates a way to create customized decision support reports after the %WAETL step in a Daily.sas program by using the %WADECIDE macro. The example code creates all non-ETL scorecard reports for the range of dates just processed by the %WAETL macro for the SAS Web Analytics Web mart whose root is at **`C:\xxx\swamart`**. The reports exclude the data from the first 14 days.

```
%wadecide(
      program=scorecard,
      date_1=&_wab_min_process_date_,
      date_2=&_wab_max_process_date_,
      swamart=C:\xxx\swamart,
      use_first_two_weeks=no,
      definition_to_run=_NON_ETL_
      );
```

The %WAETL Macro

Purpose of the %WAETL Macro

The %WAETL macro is the central component for the SAS Web Analytics application. The %WAETL macro has the following primary functions:

1 to initialize a new Web mart

2 to load and manage the data for an existing Web mart

Each of these functions has its own invocation of the %WAETL macro.

The %WAETL Macro: Syntax

The %WAETL macro was designed to be flexible so that you can invoke it in several ways to achieve the same results. You can use different sets of arguments to provide the macro with the information that it needs. The following is the syntax for the %WAETL macro:

Note: If an argument is enclosed in <>, then the argument is optional. △

%WAETL(
  <**NAME=***Web-mart-name,*>
  <**DATA_STORE=***path-to-datastore,*>
  <**TEMP_STORE=***path-to-tempstore,*>
  <**DETAIL=***path-to-libname.memname,*>
  SWAMART=*path-to-mart,*
  <**WAPATHDP_SW=***Y|N,*>
  PROGRAM=*program-name*
  );

The % WAETL Macro: Details

NAME=*Web-mart-name*

specifies the name of your SAS Web Analytics Web mart (this argument does not need to be part of your invocation if you provide the SWAMART= argument). If you provide this argument, it must match the spelling of the Name field in an observation of both Sasuser.Wainit and Sasuser.Wbinit, but is not case-sensitive.

Note: Do not use this argument if you are invoking the %WAETL macro with PROGRAM=INITIALIZE. △

DATA_STORE=*path-to-datastore*

specifies the path to the root directory of the SAS e-Data ETL application. If you are using a customized directory structure (i.e., your SAS e-Data ETL directories are outside your SAS Web Analytics Web mart directory structure), this argument is the mechanism for letting the %WAETL macro know where these other SAS e-Data ETL directories are located. If you are using the default directory structure for your SAS Web Analytics Web mart, you do not need to include this argument in your invocation. If you provide this argument, then it can be a quoted string or an unquoted string; both `'C:\xxx\data_store'` and `C:\xxx\data_store` are valid forms for this argument.

TEMP_STORE=*path-to-tempstore*
specifies the path to a directory that will act as your Worklib library, where the %WAETL macro will store the intermediate data sets that it creates. This path must exist prior to running the %WAETL macro because the path will not be created for you. If you want to have access to these data sets after the SAS session in which you invoke the %WAETL macro has finished, then you should provide information for this argument. If you do not specify this argument, then the %WAETL macro creates its own Worklib library, which will exist only as long as the SAS session in which you invoke the %WAETL macro. If you provide this argument, it must be an unquoted string (for example, **`C:\xxx\temp_store`** is a valid form, but **`'C:\xxx\temp_store'`** is not a valid form for this argument).

DETAIL=*path-to-libname.memname*
specifies the SAS LIBNAME.MEMNAME for the parsed SAS e-Data ETL data set. If you are using the defaults for your SAS Web Analytics Web mart, then you do not need to include this argument in your invocation. If you use this argument, then before you invoke the %WAETL macro, you must identify the LIBNAME with either a LIBNAME statement or some other technique. See SAS Help and Documentation for a comprehensive list of how you can identify the LIBNAME.

SWAMART=*path-to-mart*
specifies the path to the root directory for the designated SAS Web Analytics Web mart (this argument does not need to be part of your invocation if you provide the NAME= argument). This information is needed so that the %WAETL macro can locate the appropriate data resources (for example, if your SAS Web Analytics Web mart is located at **`C:\xxx\swamart`** and its subdirectories, then the root directory is **`C:\xxx\swamart`**). If you provide this argument, it can be a quoted or an unquoted string (for example, both **`'C:\xxx\swamart'`** and **`C:\xxx\swamart`** are valid forms for this argument).

WAPATHDP_SW=*Y* | *N*
specifies whether to invoke the %WAPATHDP macro (which runs the PATH procedure) from within the %WAETL macro. As a SAS Web Analytics administrator, you might decide that you do not want to execute the steps of the PATH procedure within the %WAETL macro because the PATH procedure can cause problems for large Web sites.

To skip the execution of the %WAPATHDP macro, set the value to N.

To instruct the %WAETL macro to run the %WAPATHDP macro, set the value to Y.

Note: If you omit the WAPATHDP_SW parameter, then the default value Y is used. △

PROGRAM=*program-name*
specifies the function (either INITIALIZE or WAREHOUSE) that you want the %WAETL macro to perform. The argument must match the spelling of one of these two words, but is not case-sensitive.

Using the %WAETL Macro to Initialize a New Web Mart

To initialize a new Web mart, use the following options to invoke the %WAETL macro:

```
%waetl(program=initialize,
       swamart=webmartname
       );
```

PROGRAM=INITIALIZE
specifies the %WAETL macro function for setting up a new SAS Web Analytics Web mart. The argument must match the spelling, but is not case-sensitive.

SWAMART=*webmartname*
specifies the path to the root directory of the SAS Web Analytics Web mart that you are setting up. Because the %WAETL macro uses this information to create the directory that you have specified, the directory must not already exist. After creating this directory, the %WAETL macro creates the default set of subdirectories that need to be in place before it can load data into the Web mart.

Note: If you want your Web mart to have a customized directory structure, then you need to create it manually. The %WAETL macro cannot create a customized directory structure for you. △

The %WAETL macro also creates the metadata that will control the standard operations in the Web mart. Finally, it creates some program stubs that will help you use the Web mart in the future. One of these stubs, Daily.sas, which is created in the **sas** subdirectory, is a program you can use for all subsequent invocations of the %WAETL macro for the Web mart you have just created.

Using the %WAETL Macro to Load Data into an Existing Web Mart

Using the Daily.sas Program

Once you have initialized a new Web mart, you can load data into it either by using the Daily.sas program or by coding your own invocation of the %WAETL macro. The Daily.sas program (see the default code below) is created in the **sas** subdirectory of your SAS Web Analytics Web mart when you run the %WAETL macro to initialize it. Before you can use the Daily.sas program, you will need to modify it according the instructions in the program's header block. If you have a customized directory structure, you will also need to modify the arguments in the program's invocation of the %WAETL macro.

```
/**
 * Use this program to process weblog data into the SAS Web Analytics warehouse.
 *
 * IMPORTANT: You must supply the location of the weblogs to process.
 *       Either
 *          A. Specify the location in the WEBLOGS macro variable below by replacing
 *              with the location of your weblogs. WEBLOGS can
 *             be the full path either for a single weblog or for a directory that
 *             contains multiple weblogs.
 *       Or
 *          B. Launch the e-Data ETL administrative GUI from an interactive SAS session
 *             using the "edataetl" command and specify the location of your weblog(s)
 *             in the appropriate field.
 *       If you use method B, then you must remove the "weblog_path=&weblogs" parameter
 *       from the %edataetl call below. If you leave the parameter
 *       in the edataetl call, it will take precedence over the
 *       Weblog location information you have entered via the e-Data
 *       ETL Administrator GUI.
 */

%MACRO SWA_ETL;
```

```
*** TO DO: specify the location of your weblogs;
%let WEBLOGS=;

%let webmart=C:\xxx\swamart;

%let wbrc=0;

/* Extract raw weblog data into staging datasets. */
%edataetl( data_store   =  &webmart\e-data-etl,
           weblog_path  =  &weblogs,
           program      =  extract
         );

/* Load web data into final detail dataset. */
%edataetl( data_store   =  &webmart\e-data-etl,
           program      =  load
         );

/* Load web detail data into warehouse. */
%if &wbrc = 0 %then %do;

   %waetl( swamart  =  &webmart,
           program  =  warehouse
         );

%end;

%MEND SWA_ETL;
%SWA_ETL;
```

Coding a Customized Invocation of the %WAETL Macro

Overview: Your Customized Invocation of the %WAETL Macro

If the Daily.sas program is inadequate for loading data into your SAS Web Analytics Web mart, then you will need to code your own invocation of the %WAETL macro. If you find yourself in this situation, it will be because you have performed extensive customizations for your Web mart. The exact nature of your customizations will determine how you will need to set up your invocation of the %WAETL macro. The following examples offer some suggestions for setting up your invocation of the %WAETL macro.

Custom Invocation of the %WAETL Macro: Example 1

Load the standard parsed Web log from the standard SAS e-Data ETL application directory for the **swamart** Web mart, storing the intermediate data sets in the existing **C:\xxx\swamart\tempstore** directory.

```
%waetl(name=swamart,
       temp_store=C:\xxx\swamart\tempstore,
       program=warehouse
      );
```

Custom Invocation of the %WAETL Macro: Example 2

Load the customized parsed Web log called **custom_log_detail** from the customized SAS e-Data ETL application directory (**C:\custom\e_data_etl**) for the Web mart whose root is at **C:\xxx\swamart**, storing intermediate data sets in the default temporary directory.

```
libname custom 'C:\custom\e-data-etl\detail';
%waetl(detail=custom.custom_log_detail,
       swamart=C:\xxx\swamart,
       program=warehouse
      );
```

Custom Invocation of the %WAETL Macro: Example 3

Load the customized parsed Web log called **custom_log_detail** from the standard SAS e-Data ETL application library (Detail) for the **swamart** Web mart, storing intermediate data sets in the default temporary directory.

```
libname detail 'C:\xxx\swamart\e-data-etl\detail';
%waetl(name=swamart,
       detail=detail.custom_log_detail,
       program=warehouse
      );
```

The %WASUMTST Macro

Purpose of the %WASUMTST Macro

The %WASUMTST (WebAnalyticsSUMmarizationTeST) macro creates summarizations without requiring you to execute the entire SAS Web Analytics ETL process (by using the %WAETL macro).

SAS Web Analytics summarizations are created by the %WAETL macro, which uses metadata in the Config.Waadmsum data set to drive the SUMMARY engine. This process works very well for the standard set of summarizations, but the process of defining the metadata for a new summarization can be complicated and usually requires several attempts to achieve the desired results. If each attempt requires a separate execution of the %WAETL macro, then you might need several days to successfully define a new summarization.

By giving you the ability to use the SUMMARY engine outside of %WAETL macro, the %WASUMTST macro enables you to develop and test new summarizations more quickly. After you have finished developing and testing a new summarization, you can change its designation so that it becomes an ETL summarization and is subsequently created by the %WAETL macro.

You might use the %WASUMTST macro to perform a non-ETL summarization in order to use information from a customized data source. For example, you might set up a data management program that merges supplemental customer information from an external data source with the existing SAS Web Analytics data sets. After the %WAETL macro has finished, you would execute this data management program in your ETL process (such as Daily.sas). Then, you would execute the %WASUMTST macro to create summarizations that use this supplemental customer information. In this situation, even though your non-ETL summarizations are external to the SAS Web Analytics ETL processes that are performed by the %WAETL macro, they would still be part of your overall ETL processing.

Distinguishing Between ETL and Non-ETL Summarizations

The non-ETL summarizations, like the ETL summarizations, are stored in the Summary library. How they are stored depends on whether you have indicated in the RUN_TYPE parameter that you are either running a test or updating production:

- □ When you run a test, the non-ETL summarizations replace their Summary library counterparts.
- □ When you update production, the non-ETL summarizations are appended to their Summary library counterparts.

Using the %WASUMTST macro enables you to run a new summarization as a non-ETL summarization as often as you wish without losing earlier summarization results.

CAUTION:

Remember to exercise caution when you specify a new summarization as a non-ETL summarization, so that you do not inadvertently overwrite any existing summarizations. △

The %WASUMTST Macro: Syntax

Note: If an argument is enclosed in <>, then the argument is optional. △

%WASUMTST(
SWAMART=*path-to-mart,*
RUN_TYPE=*PROD* | *TEST,*
<**DATE_RANGE=***ALL* | *LAST* | *ETL,*>,
<**DATASRCE=***BOTH* | *DETAIL* | *SESSION,*>
<**SMMYSRCE=***dataset-name,*>
<**TEMP_STORE=***path-to-tempstore,*>
<**SUMMARIZATION_TO_RUN=***summary-label(s)*>
);

The %WASUMTST Macro: Details

SWAMART=*path-to-mart*

specifies the path to the root directory for the designated SAS Web Analytics Web mart. The %WASUMTST macro uses this information to locate the data resources that are needed to create the new summarization(s) that you have requested. For example, if your SAS Web Analytics Web mart is located at `C:\xxx\swamart` and its subdirectories, then the root directory is `C:\xxx\swamart`.

RUN_TYPE=*PROD* | *TEST*

indicates whether you are running production or test non-ETL summarizations as follows:

- □ Production non-ETL summarizations are appended to their Summary library counterparts (and re-summarized if they have any overlapping dates). Use the PROD argument when you are updating existing summarizations in the Summary library.

- □ Test non-ETL summarizations replace their Summary library counterparts. Use the TEST argument when you are developing new summarizations.

CAUTION:
Remember to be careful when you designate RUN_TYPE=TEST so that you do not replace an existing SUMMARY summarization by mistake. △

DATE_RANGE=*ALL | LAST | ETL*

indicates whether you want the non-ETL summarizations to use the data from ALL the available dates, the data from only the LAST available date, or the data from the last ETL processing. If the value for DATE_RANGE is set to ALL or LAST, then the dates are the dates that are available in the Dated and Wadetail libraries. If the value for DATE_RANGE is set to ETL, then the data is used from the dates that are still in the Worklib library from the most recent execution of the %WAETL macro.

CAUTION:
If you use DATE_RANGE=ETL, then the %WASUMTST macro will need to run immediately after the %WAETL macro in the same SAS session. △

%WASUMTST macro might not be able to use the data for the dates that you have indicated that you want. This situation can arise when you use the DATASRCE parameter (either explicitly or with its default value) to specify that you want to create non-ETL summarizations from both the Detail and Session data sets. In order for the SUMMARY engine to function, the Session and Detail data sets that it uses must have the same range of dates. This means that the dates whose data are summarized might be different than the dates that you intended to request when you specified ALL or LAST for the DATE_RANGE parameter.

For example, if you have specified DATE_RANGE=ALL and if the Dated library has Session_yyyymmdd data sets for the time interval of 01APR2005 through 20APR2005, but the Wadetail library has Detail_yyyymmdd data sets for the time interval of 14APR2005 through 20APR2005, then the %WASUMTST macro will use only the dates that are common to the two libraries (14APR2005 through 20APR2005), even though more dates are available in the Dated library.

If all of the non-ETL summarizations have a single type of source data (either Session data or Detail data), or if you have specified only one of the data sources (either Session or Detail) by using the DATASRCE parameter, then the selection of date ranges is not limited to the dates that are common to both libraries.

Note: DATE_RANGE=ETL is designed to be used with RUN_TYPE=PROD in order to accommodate non-ETL summarizations that are still part of your ETL processing, but that need to be created outside the %WAETL macro because they require customized data sources. If you specify DATE_RANGE=ETL, then the Worklib library from an execution of the %WAETL macro needs to still be defined and must contain either the Session_yyyymmdd or Detail_yyyymmdd data set, or both data sets. △

DATASRCE=*BOTH | DETAIL | SESSION*

indicates the type of data source that the non-ETL summarizations will use. The default setting is BOTH, which will be appropriate for most situations. However, you will need to specify either DETAIL or SESSION when the following situations occur:

- □ You have non-ETL summarizations that involve both Detail and Session data sets.
- □ You have different date ranges in the data sets of the Dated and Wadetail libraries.

- □ You need to control the dates that are used in the non-ETL summarizations.

Specify DETAIL if you want to create only the non-ETL summarizations that use Detail_yyyymmdd data sets in the Wadetail library. Specify SESSION if you want to create only the non-ETL summarizations that use Session_yyyymmdd data sets in the Dated library.

SMMYSRCE= *dataset-name*
indicates the data set that contains the non-ETL summarization metadata. Usually, the named data set will be the default data set, Config.Waadmsum (the summary administration metadata data set). However, if (for any reason) you wish to have your non-ETL summarizations stored in a different metadata data set, then you will need to identify that data set to the %WASUMTST macro with an argument to the SMMYSRCE parameter.

TEMP_STORE=*path-to-tempstore*
specifies the path to a directory that will be your Worklib library, which the %WASUMTST macro uses for work space. This path must exist before you run the %WASUMTST macro; it is not created for you. Specify this argument if you want to be able to access the contents of this work space after the %WASUMTST macro has finished. If you do not specify this argument, then the %WASUMTST macro creates its own temporary library called Worklib, which exists only as long as the SAS session from which you invoke the %WASUMTST macro.

SUMMARIZATION_TO_RUN=*summary-label(s)*
specifies the non-ETL summaries that you want to create. The following values are valid:

Blank	The %WASUMTST macro creates all of the non-ETL summaries.
ALL	The %WASUMTST macro creates all the non-ETL summaries (same as *Blank*).
%BQUOTE	specified as %BQUOTE*(comma-delimited list of summarization labels)*. The %WASUMTST macro creates each non-ETL summarization whose label is in the comma-delimited list. Because commas separate the parameters in a macro invocation, you need to embed your list in the %BQUOTE function so that the summarizations in your list are not mistaken for additional parameters in the %WASUMTST macro instead of being recognized as elements in the argument for the SUMMARIZATION_TO_RUN parameter. If you specify only one summarization, then you do not need to use the %BQUOTE function. *Note:* The label(s) that you supply in this list must exactly match, in both case and spelling, the label(s) that are associated with the corresponding summarization in the metadata. △

Identifying Errors in the Log

The %WASUMTST macro checks for errors on the arguments that you specify and writes an error message in the log if it detects an error. The format of an error message follows:

```
ERROR: (WASUMTST) hh:mm:ss [description of error]
```

The name of the macro in which the error occurred is specified in the parentheses. If the text in the parentheses is not WASUMTST, then the error did not occur in the %WASUMTST macro, but in the macro that creates the summarization.

The time when the %WASUMTST macro encountered the error is designated by *hh:mm:ss*.

The text of the message indicates the type of problem that the %WASUMTST macro detected.

The %WASUMTST Macro: Examples

Example 1

This example code creates all the non-ETL summarizations for the SAS Web Analytics Web mart whose root is at **`C:\xxx\swamart`**, and uses the default work space. The code creates non-ETL summarizations that are derived from both the Detail and Session data sources (default) and uses the last available dates that are common to both the Dated and Wadetail libraries of the Web mart.

```
%wasumtst(
      swamart=C:\xxx\swamart,
      date_range=last
      );
```

Example 2

The following code creates all the non-ETL summarizations for the SAS Web Analytics Web mart whose root is at **`C:\xxx\swamart`**, and uses the default work space. The code instructs the %WASUMTST macro to create non-ETL summarizations that are derived from the Detail data source and uses all the available dates in the Wadetail library of the Web mart.

```
%wasumtst(
      swamart=C:\xxx\swamart,
      date_range=all,
      datasrce=DETAIL
      );
```

Example 3

The following code creates the non-ETL summarizations for the SAS Web Analytics Web mart whose root is at **`C:\xxx\swamart`**, and uses the **`C:\xxx\swamart\TempStore`** directory for work space. The code instructs the %WASUMTST macro to create non-ETL summarizations that are derived from both Detail and Session data sources and uses all available dates that are common to the Dated and Wadetail libraries of the Web mart. The metadata for the non-ETL summarizations are stored in the Config.Waadmsum_Non_ETL data set.

```
%wasumtst(
      swamart=C:\xxx\swamart,
      date_range=all,
      datasrce=both,
      smmysrce=config.waadmsum_non_etl,
      temp_store= C:\xxx\swamart\TempStore
      );
```

Example 4

The following code demonstrates a way to use the %WASUMTST macro to create a non-ETL summarization that requires customized data sources (the %WASUMST macro would be executed after the %WAETL step in a Daily.sas program). This code creates the non-ETL summarization whose metadata label is “Customer Purchases” for the SAS Web Analytics Web mart whose root is at **C:\xxx\swamart**, and uses the default work space. The code instructs the macro to use the last available dates that are in the Worklib library of the Web mart from the %WAETL macro that just finished running in its SAS session, and appends its results to the existing tables in the Summary library.

```
%wasumtst(
      swamart=C:\xxx\swamart,
      run_type=prod,
      date_range=etl,
      summarization_to_run=Customer Purchases
      );
```

Creating New Reports By Using the %WASUMTST Macro

The following table shows the steps to take if you use the %WASUMTST macro to create new reports:

Table A1.4 Steps to Create New Reports by Using the %WASUMTST Macro

Step	When to Perform	Action
1	Day 1	Edit the %EDATAETL macro and/or the %WACONFIG macro as necessary to create new variable(s). If no new variables are needed for the new report, then you can skip this step.
2	Night 1	Execute %EDATAETL macro and %WAETL macro (in the Daily.sas program).
3	Day 2	Define the new summary or summaries in the SAS Web Analytics Administrator. Verify that the ETL_RUN_SW variable has the value of 0 in the new summary or summaries.
4		Execute the %WASUMTST macro to create the newly defined summary or summaries.
5		Define the new report(s) as specified by the newly defined summary or summaries. If the report does not display as you desire, then repeat the steps for day 2.
6		In the %WAADMSUM macro, change the the value of the ETL_RUN_SW variable to 1 so that the newly defined summary or summaries will be automatically created during the SAS Web Analytics ETL processing.

The %WAPATHDP Macro

Purpose of the %WAPATHDP Macro

The %WAPATHDP (WebAnalyticsPATHDataProcess) macro creates the path analyses for SAS Web Analytics by executing PROC PATH a series of times. It is a standard feature of the ETL process (the %WAETL macro), but can also be executed outside of the %WAETL macro.

During the routine processing of the data for SAS Web Analytics reports, the %WAPATHDP macro runs within the %WAETL macro. However, if the data for the path analyses cannot be processed because the system resources that are allocated to the task are not sufficient, then the %WAETL macro might fail. If a failure due to insufficient resources occurs, then you might want to run the %WAPATHDP macro outside of the %WAETL macro to enable you to complete the path analyses without rescheduling the entire ETL operation.

The %WAPATHDP macro also enables you to override the default values for the PATH procedure, which can affect the system resources that the PATH procedure requires.

Note: The default values for the parameters for the PATH procedure are stored in the Waconfig data set. See "The Waconfig Data Set: Configuration Settings" on page 299 for more information about how to permanently reset the values. △

The %WAPATHDP Macro: Syntax

%WAPATHDP(

SWAMART=*path-to-mart,*
<**RETCODE=***macro-variable,*>
<**TEMP_STORE=***path-to-tempstore,*>
<**WAETL_SW=***N* | *Y,*>
<**SUPPORT_R1=***value,*>
<**SUPPORT_R7=***value,*>
<**SUPPORT_R30=***value*>
);

The %WAPATHDP Macro: Details

SWAMART=*path-to-mart*

specifies the path to the root directory for the designated SAS Web Analytics Web mart. The %WAPATHDP macro uses this information to locate the data resources that are needed to create the new summarization(s) that you have requested. For example, if your SAS Web Analytics Web mart is located at `C:\xxx\swamart` and its subdirectories, then the root directory is `C:\xxx\swamart`.

RETCODE=*macro-variable*

assigns a value to macro variable that indicates whether the %WAPATHDP macro completed successfully. You do not need to specify this parameter when you invoke the %WAPATHDP macro from outside %WAETL. If you do specify the RETCODE parameter, then *macro variable* needs to exist. When the %WAPATHDP macro is either finished or terminated during its execution, *macro-variable* will be set by the %WAPATHDP macro to one of the following values:

0	The %WAPATHDP macro completed successfully.
1	The %WAPATHDP macro failed to complete.

TEMP_STORE=*path-to-tempstore*

specifies the path to a directory that will be your Worklib library, which the %WAPATHDP macro uses for work space. This path must exist before you run the %WAPATHDP macro; it is not created for you. Specify this argument if you want to be able to access the contents of this work space after the %WAPATHDP macro has finished. If you do not specify this argument, then the %WAPATHDP macro creates its own temporary library called Worklib, which exists only as long as the SAS session from which you invoke the %WAPATHDP macro.

WAETL_SW=*N* | *Y*

The WAETL_SW parameter automatically answers the question: "Is the %WAPATHDP macro being run inside the %WAETL macro?" This information is needed because the %WAETL macro already provides settings that the %WAPATHDP macro needs in order to function properly. When you run the %WAPTHDP macro outside the %WAETL macro, the %WAPATHDP macro needs to provide these settings for itself.

You do not need to specify the WAETL_SW parameter because its default setting already indicates that it is not being invoked from within the %WAETL macro.

N (default) indicates that the %WAPATHDP macro is not being invoked inside the %WAETL macro, so the %WAPATHDP needs to provide its own settings.

Y indicates that the %WAPATHDP macro is being executed inside the %WAETL macro and the settings that the %WAPATHDP macro requires have been provided.

SUPPORT_R1=*value*

indicates the minimum number of visits that must contain a path in order for the path to be counted in the daily tallies. For example, if the value of the SUPPORT_R1 parameter is set to 10, and a particular path occurs only 5 times in the most recent day's ETL-processed data, then that path will not be listed nor will its occurences be counted in the reports. This cutoff value makes it possible for SAS Web Analytics to focus only on the significant paths that occur with meaningful regularity.

The default value for this parameter is provided by the WAB_R1_SUPPORT parameter in the Waconfig data set (see "The Waconfig Data Set: Configuration Settings" on page 299). The SUPPORT_R1 parameter exists so that you can assign a value to it that is meaningful for your Web mart. If the default value is too large, some low-frequency but significant paths might be disregarded; if the default value is too small, then the PATH procedure might be required to process so many paths that the system resources might be exhausted.

You can instruct the %WAPATHDP macro to use a different value from the default value in the Waconfig metadata by temporarily assigning that different value with the SUPPORT_R1 parameter.

Note: To make a permanent change to the default value for the SUPPORT_R1 parameter, you will need to change the value in the Waconfig data set (see "The Waconfig Data Set: Configuration Settings" on page 299 for more information). △

SUPPORT_R7=*value*

indicates the number of visits that must contain a path in order for the path to be counted in the 7-day tallies. See the preceding description of the SUPPORT_R1 parameter for more information about the use of the SUPPORT_R7 parameter.

SUPPORT_R30=*value*

indicates the number of visits that must contain a path in order for it to be part of the 30-day tallies. See the preceding description of the SUPPORT_R1 parameter for more information about the use of the SUPPORT_R30 parameter.

Identifying Errors in the Log

The %WAPATHDP macro checks for errors on the arguments that you specify and writes an error message in the log if it detects an error. The format of an error message follows:

```
ERROR: (WAPATHDP) hh:mm:ss [description of error]
```

The name of the macro in which the error occurred is specified in the parentheses. If the text in the parentheses is not WAPATHDP, then the error did not occur in the %WAPATHDP macro, but in the module that is specified in the parentheses.

The time when the %WAPATHDP macro encountered the error is designated by *hh:mm:ss*.

The text of the message indicates the type of problem that the %WAPATHDP macro detected.

The %WAPATHDP Macro: Examples

Example 1

The following example code creates the path analyses for the SAS Web Analytics Web mart whose root is at **C:\xxx\swamart**, and uses the default work space. The default value for the SUPPORT_R1 parameter is replaced with the value of 3, but the default values for the SUPPORT_R7 and SUPPORT_R30 parameters remain in effect:

```
%WAPATHDP(
      swamart=C:\xxx\swamart,
      support_r1=3
      );
```

Example 2

The following code creates the path analyses for the SAS Web Analytics Web mart whose root is at **`C:\xxx\swamart`**, and uses the **`C:\xxx\swamart\TempStore`** directory for work space. The default value for the SUPPORT_R7 parameter is replaced with the value of 5. The default value for the SUPPORT_R30 parameter is replaced with the value of 10. The default value for the SUPPORT_R1 parameter is used.

```
%WAPATHDP(
      swamart=C:\xxx\swamart,
      temp_store= C:\xxx\swamart\TempStore,
      support_r7=5,
      support_r30=10
      );
```

APPENDIX 2

List of Standard SAS Web Analytics Reports

List of Standard Traffic Reports

The following table lists and describes each standard traffic report that is provided with SAS Web Analytics. Some of the reports are available only as drill-down reports from other reports. See the to identify the access path to individual reports.

Table A2.1 Standard Traffic Reports

Type	Report	Description
Visitor	Unique Visitors	Identifies the visitors who have the most activity on your Web site. The report lists each unique visitor whose visit count is greater than one.
Browser and Platform	Browsers	Displays a distribution of the different Web browsers that are used by visitors who navigate the Web site.
	Browser Versions	Displays a distribution of the different Web browser versions that are used by visitors who navigate the Web site.
	Platforms	Displays a distribution of the different platforms (operating systems) that are used by visitors who navigate the Web site.
Status Codes	Status Codes	Provides information about the frequency that a code was returned by the server to the visitor's browser to report the outcome of a request.
	Hourly Status Codes	Provides information about the frequency of status codes categorized by hour of day.
	Error Status Codes	Provides information about the frequency of errors that occur when visitors attempt to access your Web site.

Type	Report	Description
	Error Status Code Pages	Provides information about individual pages that generated errors.
	Error Status Code Page Referrers	Provides information about the frequency of status codes for each page that was requested.
Navigation	Page Frequency	Displays a distribution of the pages that were requested. This report enables you to identify the pages and family of pages that are most often viewed.
	Entry Pages	Displays the most common requests that visitors make to enter the Web site.
	Referrer by Entry Page	Displays a list of entry pages for each referrer.
	Entry Pages by Referrer	Displays the pages and visit information that are associated with the domains.
	Exit Pages	Provides a list of all Web pages from which the current site was exited.
Overview	Hourly Metrics	Displays daily basic traffic statistics for the site on an hourly basis.
	Site Metrics	Displays basic traffic statistics for the site.
	Day of Week Metrics	Displays daily basic traffic statistics for the site by day of week.
	Site Metrics by Day of Week	Displays statistics for the site by a particular day of the week.
Referrer	Visit Referrer Domains	Displays a distribution of referrer domains, which enables you to determine the origin of visitors.
	Entry Pages by Referrer	Displays the pages and visit information that are associated with the domains.
	Referrer by Entry Page	Displays a list of entry pages for each referrer.
	Search Terms	Displays a distribution of search terms that are used by visitors to find your Web site.
	Referrer by Search Terms	Displays the domains from which visitors searched for specific items of information.
	Search Terms by Referrer	Displays a distribution of search terms that are used by visitors within referrer domains to find your Web site.
	Like Search Terms	Displays a list of search terms that are similar to a term that you specify.
	Referrer by Search Terms	Displays the domains from which visitors searched for specific items of information.

List of Standard Non-Traffic Reports

The following table lists and describes each standard non-traffic report.

Table A2.2 Standard Non-Traffic Reports

Category	Report	Description
Scorecard	Site Metrics	Enables you to view the performance and the forecast values of the key performance indicators (KPIs) that are related to traffic on the site that is being analyzed.
Dashboard	Site Metrics	Displays the values for each site's metrics, assigns a performance level to that metric based on its desired business direction, and displays historical information for that metric for a given date.
Funnel	Interactive Funnel	Provides a detailed description of any sequential process on a Web site, such as a sequence of Web pages that are visited.
	Static Funnel	Provides a detailed description of any sequential process on a Web site, such as a sequence of Web pages that are visited. A Static Funnel report is created during the ETL process.
Path Analysis	Entry Paths	Displays a list of the most popular points of entry into the Web site.
	Referrer Entry Path	Displays a list of referrers that sent the most traffic to your Web site.
	Interactive Path Analysis	Displays navigational information about the order in which pages or URIs occur in visits whose visitors navigate the Web site. The report traces a path from any starting page or to any ending page, or between any starting and ending pages.
Segmentation	Repeat Visitors - Totals	Displays a list of repeat visitors (totals) to the Web site, based on a set of rules which help predict which visitors will return.
	Repeat Visitors - Averages	Displays a list of repeat visitors (averages) to the Web site, based on a set of rules which help predict which visitors will return.

Alphabetical List of Standard SAS Web Analytics Reports

The following table lists the standard SAS Web Analytics reports. The first column lists the name of the report, and the second column contains the path by which you access the report. The access path always begins with a selection from the banner of the SAS Web Analytics page. Some of the reports are available only as drill-down reports from other reports. You access the drill-down reports by clicking a link in the first column of the report table. If there is no link in the first column of the table, then no drill-down report is available.

Alphabetical List of Standard SAS Web Analytics Reports

Report	Access path to reports
Browser Versions	Traffic ▶ Browser and Platform ▶ Browsers ▶ Browser Versions
Browsers	Traffic ▶ Browser and Platform ▶ Browsers
Day of Week Metrics	Traffic ▶ Overview ▶ Day of Week Metrics
Entry Pages	Traffic ▶ Navigation ▶ Entry Pages
Entry Pages by Referrer (Navigation menu)	Traffic ▶ Navigation ▶ Entry Pages ▶ Referrer by Entry Page ▶ Entry Pages by Referrer
Entry Pages by Referrer (Referrer menu)	Traffic ▶ Referrer ▶ Visit Referrer Domains ▶ Entry Pages by Referrer
Error Status Code Page Referrers	Traffic ▶ Status Codes ▶ Error Status Codes ▶ Error Status Code Pages ▶ Error Status Code Page Referrers
Error Status Code Pages	Traffic ▶ Status Codes ▶ Error Status Codes ▶ Error Status Code Pages
Error Status Codes	Traffic ▶ Status Codes ▶ Error Status Codes
Exit Pages	Traffic ▶ Navigation ▶ Exit Pages
Hourly Metrics	Traffic ▶ Overview ▶ Hourly Metrics
Hourly Status Codes	Traffic ▶ Status Codes ▶ Status Codes ▶ Hourly Status Codes
Interactive Funnel	Funnel ▶ Interactive Funnel
Interactive Path Analysis	Path Analysis ▶ Path Analysis Reports ▶ Interactive Path Analysis
Like Search Terms	Traffic ▶ Referrer ▶ Like Search Terms
Page Frequency	Traffic ▶ Navigation ▶ Page Frequency
Platforms	Traffic ▶ Browser and Platform ▶ Platforms
Referrer by Entry Page (Navigation menu)	Traffic ▶ Navigation ▶ Entry Pages ▶ Referrer by Entry Page
Referrer by Entry Page (Referrer menu)	Traffic ▶ Referrer ▶ Visit Referrer Domains ▶ Entry Pages by Referrer ▶ Referrer by Entry Page
Referrer by Search Terms	Traffic ▶ Referrer ▶ Search Terms ▶ Referrer by Search Terms
Repeat Visitor - Averages	Segmentation ▶ Segmentations ▶ Repeat Visitors - Averages
Repeat Visitor - Totals	Segmentation ▶ Segmentations ▶ Repeat Visitors - Totals

Report	Access path to reports
Search Terms	Traffic ▶ Referrer ▶ Search Terms
Site Metrics (Dashboard)	Dashboard ▶ Site Metrics
Site Metrics (Scorecard)	Scorecard ▶ Site Metrics
Site Metrics (Traffic)	Traffic ▶ Overview ▶ Site Metrics
Site Metrics by Day of Week	Traffic ▶ Overview ▶ Day of Week Metrics ▶ Site Metrics by Day of Week
Static Funnel*	Funnel
Status Codes	Traffic ▶ Status Codes ▶ Status Codes
Unique Visitors	Traffic ▶ Visitors ▶ Unique Visitors
Visit Referrer Domains	Traffic ▶ Referrer ▶ Visit Referrer Domains

* Static Funnel reports consist of individual reports that are created during the ETL process.

APPENDIX

3

The SUMMARY Engine

The Waadmsum Data Set

Overview: The Waadmsum Data Set

Each time that it runs, the %WAETL macro generates and executes a series of SUMMARY procedure steps to transform the detail data sets (created by the SAS e-Data ETL processes) and the session data sets (derived by %WAETL macro from the detail data sets) into the final summarized form (a SAS summary data set). The SUMMARY engine module within the %WAETL macro uses the information in the Waadmsum data set to build the SUMMARY procedure steps.

The supplied Waadmsum data set has 15 rows, one row for each of the 15 summarization families. Each summarization family has one member for each of its summary levels: day, week, month, quarter, and year. For example, the Hourly Metrics family has the following SAS summary data sets as members:

□ Hourly_Metrics_Day

□ Hourly_Metrics_Week

□ Hourly_Metrics_Month

□ Hourly_Metrics_Qtr

□ Hourly_Metrics_Year

The SUMMARY engine first expands the Waadmsum data set in order to create a separate definition for all of the individual members of each summarization family. Using this expanded data, the SUMMARY engine then creates a SAS summary data set for each member of every summarization family.

Variables of the Waadmsum Data Set

The variables of the Waadmsum metadata data set contain all the information that the SUMMARY engine needs in order to generate the SUMMARY procedure steps that are executed from within the %WAETL macro.

Note: For more information about SUMMARY procedure, see SAS Help and Documentation. △

The following table lists the variables in the Waadmsum data set:

Table A3.1 The Variables in the Waadmsum Data Set

Variable	Description
Label	Is the name of a summarization family as it appears in the interface of the SAS Web Analytics Administrator.
Source_Table	Is the type of source data set (either session or detail) that provides the information for the summarization family.
Proc_Summary_Options	Are the options that are part of the SUMMARY procedure statement. The most commonly used option in the Waadmsum metadata is SUM MISSING (for more information about SUMMARY procedure options, see SAS Help and Documentation).
Input_Where	Is the WHERE statement that filters the Source_Table data set. The syntax for this field is: **(where=<*subsetting condition*>)** The entire WHERE statement must be contained within parentheses. If no filtering is needed, then leave this field blank.
Summary_Level_List	Is a comma-delimited list of the summary levels for the summarization family members. Five possible summary levels are possible: Day, Week, Month, Qtr (quarter), and Year. Each entry in this list becomes the suffix for a corresponding SAS summary data set.
Class_Var_List	Is a comma-delimited list of class variables. *Note:* The Date variable must be one of the class variables. △ For more information about class variables, see SAS Help and Documentation.
Id_Var_List	Is a comma-delimited list of ID variables. For more information about variables, see SAS Help and Documentation.
Id_Var_Summary_Level_List	Is a comma-delimited list of the summary levels that apply for the Id_Var_List entries. The five possible summary levels are: Day, Week, Month, Qtr (quarter), and Year.
Analysis_Var_List	Is a comma-delimited list of analysis variables. For more information about analysis variables, see SAS Help and Documentation. If no analysis variables are needed, then leave this field blank.

Variable	Description
Output_Dsn_Options_0	Are the SAS data set options that you want to apply to the Results_Table data set for the re-summarizing SUMMARY procedure step of the summarization family. The most commonly used options are DROP, KEEP, and RENAME. For more information, see SAS Help and Documentation. If no data set options are needed, then leave this field blank. The WHERE data set option is coded separately in a different field and applies to both the initial and the re-summarizing steps.
Summary_Outdsn_Where	Is the WHERE data set option that filters the Results_Table data set. If you do not need to provide a filter, then leave this field blank. The SUMMARY engine creates a **`Where=(_type_ eq 'mask')`** statement for every Results_Table data set in order to make sure that only the summaries specified in Class_Var_List are written to Results_Table. If you need additional filtering, then define it in this field by using the following syntax: **`Where=(<`*`subsetting condition`*`>)`** The entire WHERE statement is not contained within parentheses.
Output_Options	Contains the options for the SUMMARY procedure output statement. The most commonly used option is SUM, which makes sure that the Analysis_Var_List entries that are summed have the same variable name in Results_Table as the source variables in Source_Table. For more information, see SAS Help and Documentation.
Results_Table	Is the prefix that all members in a summarization family share (for example, Hourly_Metrics). The SUMMARY engine combines Results_Table with a suffix derived from an entry in Summary_Level_List to create the name for each SAS summary data set. ***CAUTION:*** **The Results_Table name cannot be more than 22 characters long.** △

General Information About SAS Web Analytics Summaries

One metadata record in the Waadmsum data set exists for each summarization family.

Note: A summarization family such as Hourly Metrics consists of one member for each summary level: day, week, month, quarter, and year. △

It is easier to maintain one metadata record per family instead of one record per family member.

The record of a summarization family in the Waadmsum data set might not have information in all of the variables. Only the variables that are necessary to define the family summaries have values. Any variables that are not needed contain blanks.

Each traffic report in the SAS Web Analytics Report Viewer displays the information from a SAS summary data set that corresponds to one family member. The following SUMMARY procedure steps are required to create each SAS summary data set:

- □ the initial step that summarizes the Source_Table information
- □ the re-summarization step that applies the results from the initial step to the information that already exists in the SAS summary data set

How the SUMMARY Engine Performs an Initial Summarization

In the following example, the SUMMARY engine uses the specific record from the Waadmsum data set to create the SAS summary data sets for the Hourly_Metrics summarization family.

Table A3.2 Variables in the Waadmsum Data Set for the Hourly_Metrics Summarization Family

Variable in the Waadmsum Data Set	**Value**
Label	Hourly Metrics
Source_Table	detail
Proc_Summary_Options	sum missing
Input_Where	*blank*
Summary_Level_List	day, week, month, qtr, year
Class_Var_List	date, hour *Note:* The Date variable must be one of the class variables.△
Id_Var_List	*blank*
Id_Var_Summary_Level_List	*blank*
Analysis_Var_List	entry_point, file_count, page_count, bytes_sent
Output_Dsn_Options_0	*blank*
Output_Dsn_Options_1	rename=(entry_point=session_count)
Output_Dsn_Where_Stmt	*blank*
Output_Options	sum=
Results_Table	hourly_metrics

The SUMMARY engine builds the following SUMMARY procedure step for the initial summarization of the _Day member of the Hourly Metrics family from the metadata listed in the previous table:

```
                    ❶                              ❷
proc summary data=&temp_lib..detail_20040115 sum missing chartype;
  ❸ class date
        hour
        ;
  ❹ types date*hour
        ;
  ❺ var entry_point
       file_count
       page_count
       bytes_sent
      ;       ❻
  output out=&temp_lib..hourly_metrics_day_X(
     ❽ where=(_type_='11')
```

```
        ❼ rename=(entry_point=session_count)
        ❼ keep= date hour entry_point file_count
                        page_count bytes_sent _type_)
    ❾ sum=;
run;
```

The following table explains the features of the initial step of the SUMMARY procedure:

Table A3.3 The Initial Step of the SUMMARY Procedure

Symbol	Description
❶	The SUMMARY engine expands Source_Table so that it has both LIBNAME (&Temp_Lib) and MEMNAME (Detail_20040115) components. The SUMMARY engine derives the yyyymmdd suffix for the MEMNAME component from the dates in the SAS e-Data ETL source file (in this example, **'20040115'** indicates the date January 15, 2004).
❷	The SUMMARY engine places the Proc_Summary_Options value (without any delimiting commas, and, if necessary, with the CHARTYPE specified) into the SUMMARY procedure statement.
❸	The SUMMARY engine converts the comma-delimited entries in Class_Var_List to a variable list for the SUMMARY procedure CLASS statement. *Note:* The Date variable must be one of the class variables.△
❹	The SUMMARY engine uses the entries in Class_Var_List to create the SUMMARY procedure TYPES statement.
❺	The SUMMARY engine converts the comma-delimited entries in Analysis_Var_List to items in the variable list for the SUMMARY procedure VAR statement.
❻	The SUMMARY engine expands Results_Table so that it has both LIBNAME (&Temp_Lib) and MEMNAME (Hourly_Metrics_Day_X) components. The SUMMARY engine derives the _Day suffix for the MEMNAME component from an entry in Summary_Level_List. The '_X' suffix forces the summarization to have a unique DSN even if several dates exist in the Web log that was parsed by the SAS e-Data ETL application.
❼	The SUMMARY engine places the Output_Dsn_Options_1 value verbatim into the OUTPUT statement. This SUMMARY procedure step is for an initial summarization of the SAS e-Data ETL data.
❼	The SUMMARY engine supplements the Output_Dsn_Options_1 value with additional output data set options that are applied to the data set that is specified in the OUT= option for the OUTPUT statement.
❽	The SUMMARY engine generates a WHERE statement to control which observations are written to the results data set. If the value for Summary_Outdsn_Where is not blank, then its filter is logically appended (with AND) to this generated WHERE statement.
❾	The SUMMARY engine places the Output_Options value verbatim into the OUTPUT statement. The SUM= option indicates that the summed results in Results_Table are stored in variables that have the same name as the source variable. For example, the sum of all the File_Count variables is stored in a variable called File_Count.

How the SUMMARY Engine Performs a Re-summarization

The following code is the SUMMARY procedure step that the SUMMARY engine builds for the re-summarization of the _Day members of the Hourly Metrics family. The code is built from the Waadmsum metadata listed above. The re-summarization step has components that are similar to the initial summarization step.

```
                  ❶
proc summary data=&temp_lib..hourly_metrics_day sum missing chartype;
  class date
         hour
        ;
  types date*hour
        ;
❷ var session_count
       file_count
       page_count
       bytes_sent
      ;        ❸
   output out=summary.hourly_metrics_day
         where=(_type_=’11’)
❹ keep= date hour session_count file_count page_count bytes_sent _type_)
         sum=;
run;
```

The following table discusses the differences between the re-summarization and the initial summarization steps:

Table A3.4 The Re-summarization Step of the SUMMARY Procedure Step

Symbol	Description
❶	The SUMMARY engine uses the data set specified by the OUTPUT OUT= option from the initial summarization to identify the input data set for the re-summarization.
❷	The SUMMARY engine changes the VAR statement from the initial summarization to reflect the variable renaming that was performed during that re-summarization step. Because the Entry_Point variable was renamed to Session_Count, the Entry_Point variable needs to be referred to as Session_Count in the re-summarization step.
❸	The SUMMARY engine expands Results_Table so that it has both LIBNAME (Summary) and MEMNAME (Hourly_Metrics_Day) components. The Summary LIBREF makes sure that the results are written to the permanent library of the Web mart. The SUMMARY engine derives the _Day suffix for the MEMNAME component from an entry in Summary_Level_List and identifies the output data set as the SAS summary data set for the _Day member of the Hourly Metrics family.
❹	The SUMMARY engine supplements the Output_Dsn_Options_0 value with additional output data set options that are necessary for the output data set to be created correctly. Because the Output_Dsn_Options_0 of this family is blank, the only text is the SUMMARY engine supplement. If the output data set for the re-summarization requires any options that the SUMMARY engine cannot derive exclusively, then the required options must be supplied by the text in Output_Dsn_Options_0.

Modifying the Waadmsum Metadata Data Set

To add a new summary or to change the way that an existing summary is performed, edit the metadata in the Waadmsum data set. To access the Waadmsum data set, perform the following steps:

1. Select a Web mart (if necessary) in the SAS Web Analytics Administrator.
2. Select **Data ▶ Configuration Tables ▶ Summary Engine Metadata**.
3. Modify the data as necessary to control the summarizations.

To customize the default SAS Web Analytics summary data sets, you will need to change the Waadmsum metadata that control the summarizations. For details on how to modify or delete a summarization family, see “Modifying or Deleting a Summarization” on page 137.

CAUTION:

Make a backup copy of the Config.Waadmsum data set before you make any changes. △

The SUMMARY engine performs validity checks on the Waadmsum metadata before it generates any SUMMARY procedure steps. If the changes that you make corrupt the integrity of the Waadmsum metadata, then both the SUMMARY engine and the %WAETL macro will fail to run. If you back up your Waadmsum metadata, then you can restore the Waadmsum data set and continue to run the %WAETL macro while you research the problem with your changes to the Waadmsum metadata. If you do not have a backup, then you will not be able to run the %WAETL macro until you have resolved the problem.

APPENDIX

4

The Waconfig Data Set: Configuration Settings

The Waconfig Data Set: Configuration Settings

Overview: The Waconfig Data Set

The purpose of the Waconfig data set is to list the parameters that you can manipulate in order to control the %WAETL macro. To control the statistical processing that is performed by the macro, it is easier to change the metadata entry in the Waconfig data set than it is to change the statements within the %WAETL macro itself.

The Waconfig data set contains the following columns:

Parameter lists the names of parameters that correspond to macro variables.

Value contains the values for the parameters.

Description identifies the specifications in the %WAETL macro to which the values in the Value column correspond.

The %WAETL macro converts each record in the Waconfig data set to a macro variable.

Example: Using a Parameter in the Waconfig Data Set to Modify the %WAETL Macro

In the Waconfig data set, the WEB_DAYS_IN_HISTORY parameter is initially set up to retain 90 days of summaries. You can change the number of days that are retained by modifying the value for this parameter. For example, you can set the value for WEB_DAYS_IN_HISTORY to 60 in the Waconfig data set. Because the %WAETL macro reads the Waconfig data set as the source for its variables, the %WAETL macro then uses 60 as the number of days of summaries to keep.

The Parameters in the Waconfig Data Set

The following table lists the parameters of the Waconfig data set by category:

Table A4.1 Parameters in the Waconfig Data Set

Category	Description of Category	Parameter
Pathing	Specifies the parameters that are used to create the Summary.Pathing data set.	WAB_R1_SUPPORT WAB_R7_SUPPORT WAB_R30_SUPPORT WAB_ITEMSET_SIZE WAB_MAX_PATH_PAGES
Data Aging Thresholds	Specifies the parameters that control how often the historical data is aged and therefore how often it is routinely removed from the Web mart.	WAB_DAYS_IN_HISTORY WAB_WEEKS_IN_HISTORY WAB_MONTHS_IN_HISTORY WAB_QUARTERS_IN_HISTORY WAB_YEARS_IN_HISTORY WAB_NUM_DETAIL_DATA_SETS WAB_NUM_SESSION_DATA_SETS WAB_VISITOR_DAYS_IN_HISTORY WAB_PATHING_DAYS_IN_HISTORY
Decision Support Reports	Specifies the parameters that control the Decision Support report data sets.	WAB_AUTOSEG_TRAIN_SAMP_PCT WAB_AUTOSEG_VALID_SAMP_PCT WAB_AUTOSEG_NUM_SEGMENTS WAB_AUTOSEG_ALPHA WAB_AUTOSEG_MINWORTH WAB_AUTOSEG_MAXDEPTH WAB_AUTOSEG_LEAFSIZE WAB_DAYS_IN_SCORECARD WAB_DAYS_IN_DASHBOARD
Session Summary Control	Specifies how session-related summaries are assigned.	WAB_INIT_BY_SESS_VARS WAB_FIRST_SESS_VARS WAB_LAST_SESS_VARS WAB_SUM_SESS_VARS
Summary Engine	Specifies which data set is the metadata source for the %WASUMENG macro.	WAB_METADSN

Listed below in alphabetical order are all the default values for the parameters in the Waconfig data set as it is shipped with SAS Web Analytics. For the parameters with blanks in the Default Value column, the default value is null.

Table A4.2 Parameters in the Waconfig Data Set

Parameter	**Default Value**	**Description**
WAB_ADMSUM	config.waadmsum	The SUMMARY engine administration WAADMSUM metadata, which is the source for the WASUMENG metadata.
WAB_AUTOSEG_ALPHA	0.20	For automatic segmentation, a threshold *p*-value for the significance level of a candidate-splitting rule. This value applies only for targets that have an interval measurement level.
WAB_AUTOSEG_LEAFSIZE	~	For automatic segmentation, the smallest number of training observations that a new branch can have.
WAB_AUTOSEG_MAXDEPTH	10	For automatic segmentation, the number of splitting rules that are needed to define the node. The root node has a depth of 0. The children of the root node have a depth of 1, and so on.
WAB_AUTOSEG_MINWORTH	0.0	For automatic segmentation, a threshold value for the logworth of a candidate-splitting rule. This value applies only for targets that have a nominal or an ordinal measurement level.
WAB_AUTOSEG_NUM_SEGMENTS	4	The number of segments that are generated by the autosegmentation model.
WAB_AUTOSEG_TRAIN_SAMP_PCT	0.6	The percentage (expressed as a decimal number) that determines whether an observation gets assigned to the Training data set or to the Validation data set.
WAB_AUTOSEG_VALID_SAMP_PCT	0.4	The proportion of total visitor IDs that are used to identify or to validate the automatic segmentation model.
WAB_DAYS_IN_AUTOSEG	365	The number of days that are available for segmentation analyses.

Parameter	Default Value	Description
WAB_DAYS_IN_DASHBOARD	365	The number of days that are available for dashboard reports.
WAB_DAYS_IN_HISTORY	185	The number of days that are available for daily summary reports.
WAB_DAYS_IN_SCORECARD	365	The number of days that are available for scorecard reports.
WAB_FIRST_SESS_VARS		A comma-delimited list of session variables that have values for the First.Session_Id variable. There is only one variable, Status_Code, listed by default.
WAB_INIT_BY_SESS_VARS	Status_Code, Requested_File	A comma-delimited list of session variables that have values for the Initialize_By_Session variable.
WAB_INTERACTIVE_SUPPORT	3	The minimum number of visits per day for a path of requested files to be output in interactive pathing.
WAB_ITEMSET_SIZE	7	The maximum number of URLs per page that are allowed in a path.
WAB_LAST_SESS_VARS		A comma-delimited list of session variables.
WAB_MAX_PATH_PAGES	500	The maximum page count for a visit to be counted in the PROC PATH analyses.
WAB_METADSN	&temp_lib.wasumeng	The summary metadata set that causes creation of the PROC SUMMARY code. This code is used to build the descriptive summary in the data warehouse.
WAB_MONTHS_IN_HISTORY	36	The number of months that are available in the monthly summary tables.
WAB_NUM_DETAIL_DATA_SETS	7	The number of separate date-partitioned detail data sets that are kept in the Detail library.
WAB_NUM_SESSION_DATA_SETS	30	The number of separate date-partitioned detail data sets that are kept in the Dated library.
WAB_PATHING_DAYS_IN_HISTORY	30	The number of days of history that are kept in the Summary.Pathing data set.
WAB_QUARTERS_IN_HISTORY	12	The number of quarter years that are available in the quarterly summary tables.

Parameter	Default Value	Description
WAB_R1_SUPPORT	5	The minimum number of total sessions that correspond to a unique path over an interval of one day.
WAB_R30_SUPPORT	20	The minimum number of total sessions that correspond to a unique path over an interval of 30 days.
WAB_R7_SUPPORT	10	The minimum number of total sessions that correspond to a unique path over an interval of 7 days.
WAB_SUM_SESS_VARS		A comma-delimited list of session variables with the sum of all Session_Id values.
WAB_VISITOR_DAYS_IN_HISTORY	90	The number of days of history that are kept in the Summary.Visitor_Day data set.
WAB_WEEKS_IN_HISTORY	52	The number of weeks that are available in the weekly summary tables.
WAB_YEARS_IN_HISTORY	3	The number of years that are available in the yearly summary tables.
WEEK_START	Sunday	The first day of the week.

Changing Values in the Waconfig Data Set

Overview: Changing Values in the Waconfig Data Set

If the default values for the parameters in the Waconfig data set do not meet your needs, then you can change the values for these parameters by using the SAS Web Analytics Administrator. The following procedure shows you how to change any values in the Waconfig data set.

CAUTION:

Make a backup copy of the Config.Waconfig data set before you make any changes to it. If the changes that you made do not work properly, then you will be able to restore your original settings by using the backup copy. △

Edit the Waconfig Data Set

To edit the Waconfig data set, open the SAS Web Analytics Administrator interface, and perform the following steps:

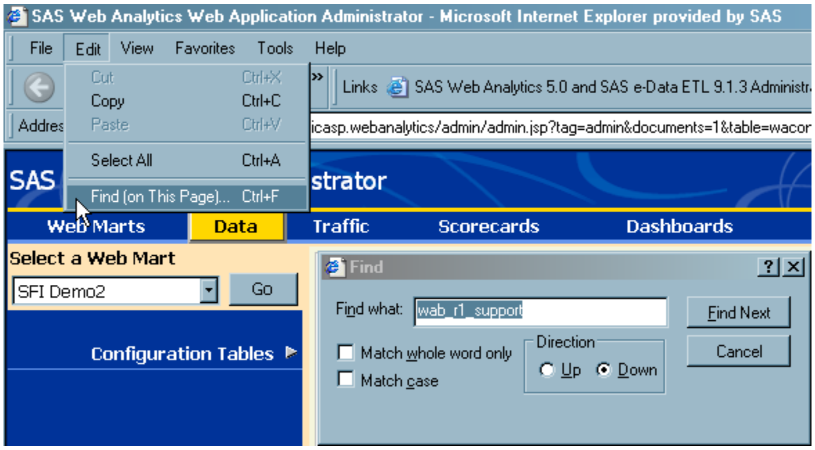
1 If a Web mart is not already selected, select a Web mart from the drop-down menu in the upper left corner of the SAS Web Analytics Administrator, and click **Go**.

2 Select the **Data** link from the menu bar so that **Data** is highlighted in yellow.

3 Move the mouse over **Configuration Tables** (located in the navigation area on the left of the Web page), and then select **Web Analytics Configuration** in the expanded navigation tree. The Waconfig data set is displayed.

4 Find the row whose description matches the summarization that you want to change. To use the browser's Find function, select **Edit**, and then **Find (on This Page)**. In the **Find** dialog box, type the name of the parameter in order to quickly locate the row you want to change in the Waconfig data set.

Display A4.1 Using the Browser's Find Function to Locate a Row in Waconfig



icasp.webanalytics/admin/admin.jsp?tag=admin&documents=1&table=wacor

SFI Demo2

Find what: wab_r1_support

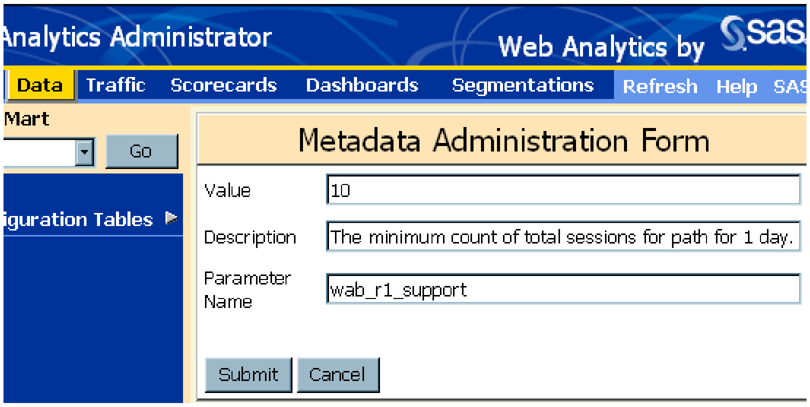
□ Match whole word only

□ Match case

○ Up ● Down

5 Click in the **ID for Maintenance** column of the row you want to change. The Metadata Administration Form (shown in the following display) enables you to edit any values for that row.

Display A4.2 The Metadata Administration Form



Metadata Administration Form

Value	10
Description	The minimum count of total sessions for path for 1 day.
Parameter Name	wab_r1_support

6 Type the new value in the **Value** text box. Click **Submit**, and then click **Continue**.

7 To find and change any additional summarization parameters in the Waconfig data set, repeat steps 4, 5 and 6 above.

Changing the Parameters in the Session Summary Control Category

The parameters in the Session Summary Control category enable you to customize the way that the %WAETL macro creates the analysis variables. The parameter WAB_INIT_BY_SESS_VARS affects the way in which the %WAETL macro creates the daily detail data sets.

The following parameters affect the way in which the %WAETL macro creates the daily session summary data sets:

□ WAB_FIRST_SESS_VARS

□ WAB_LAST_SESS_VARS

□ WAB_SUM_SESS_VARS

The WAB_INIT_BY_SESS_VARS Parameter for Controlling Daily Detail Data Sets

The initial value for the WAB_INIT_BY_SESS_VARS parameter is `status_code`. Setting `status_code` as the initial value for the WAB_INIT_BY_SESS_VARS parameter means that when the %WAETL macro creates the daily detail data sets, the first time that a status code value for a session is encountered, an indicator variable (Init_By_Sess_A) is set to `1`. For all of the subsequent times that this particular status code is encountered for that session, the variable Init_By_Sess_A is set to `0`. In this manner, the indicator variable Init_By_Sess_A enables you to count how many sessions have experienced each status code value.

The following table illustrates how the value for the variable Init_By_Sess_A changes when a sequence of values for each specific status code is encountered:

Table A4.3 How the Indicator Variable Init_By_Sess_A Is Set When a Status Code Is Encountered

Sequence in a Session	Status_Code	Init_By_Sess_A
1	200	1
2	200	0
3	401	1
4	302	1
5	200	0
6	500	1
7	401	0
8	302	0

If you want to supplement the SAS Web Analytics reports with a similar tally by session, you need to add the appropriate session variable name to the Value field of the WAB_INIT_BY_SESS_VARS parameter in the Waconfig data set.

Note: The Value field requires a comma-delimited variable list; therefore, you must insert a comma before each variable name that you add to the list. △

The %WAETL macro automatically creates an Init_By_Sess_*X* indicator variable to represent each variable in the list of values for the WAB_INIT_BY_SESS_VARS parameter, where *X* is a letter suffix that identifies the position of the source variable in the list. For example, Init_By_Sess_A is the indicator variable for status_code, the first variable in the list; Init_By_Sess_B is for the second variable in the list; Init_By_Sess_C is for the third variable; and so on.

If you add a variable to the Value field for the WAB_INIT_BY_SESS_VARS parameter, then you must make changes to the metadata in the Wasumeng summary table so that the SUMMARY engine will include the new variable in the summarizations that are performed by the %WAETL macro. For more information about %WAETL, see Appendix 1, "Macro Reference for the SAS Web Analytics 5.2 Solution," on page 261. Because the Init_By_Sess_A variable has already been set up as the indicator variable for the parameter value **status_code**, you might want to model the updates that you make to the Wasumeng summary table by using an existing summary that includes the variable Init_By_Sess_A in its ANALYSIS_VARS parameter.

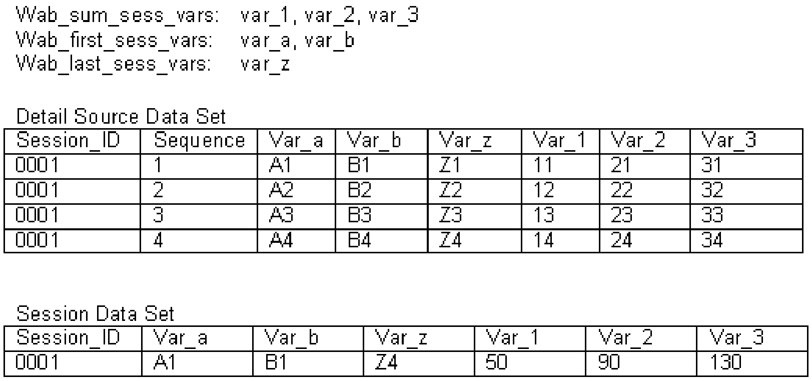
The Parameters for Customizing Session Summary Data Sets

Each session data set has one observation per Session_ID field and is created when the %WAETL macro summarizes the Weblog_Detail detail data set for a selected date by the Session_ID field. You can customize the variables in the session data sets by using any of the following methods (singly or in combination) :

- Calculate the sum of the values of the WAB_SUM_SESS_VARS parameter for the Session_ID field.
- Use the first (entry) value of the WAB_FIRST_SESS_VARS parameter for the Session_ID field.
- Use the last (exit) value of the WAB_LAST_SESS_VARS parameter for the Session_ID field.

The value of each of these parameters is a comma-delimited list of variable names. To customize a variable by using one of these three methods, add a variable name to the list in the Value column of the corresponding parameter.

Although the SAS Web Analytics software does not use this feature by default, the following example shows how this customization would work by using these variables in a hypothetical Weblog_Detail detail data set:



Wab_sum_sess_vars:	var_1, var_2, var_3
Wab_first_sess_vars:	var_a, var_b
Wab_last_sess_vars:	var_z

Detail Source Data Set

Session_ID	Sequence	Var_a	Var_b	Var_z	Var_1	Var_2	Var_3
0001	1	A1	B1	Z1	11	21	31
0001	2	A2	B2	Z2	12	22	32
0001	3	A3	B3	Z3	13	23	33
0001	4	A4	B4	Z4	14	24	34

Session Data Set

Session_ID	Var_a	Var_b	Var_z	Var_1	Var_2	Var_3
0001	A1	B1	Z4	50	90	130

APPENDIX 5

The Summary Data Sets for SAS Web Analytics

Browser_<Summary Level> Data Set

Library: Summary

Data set name: Browser_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

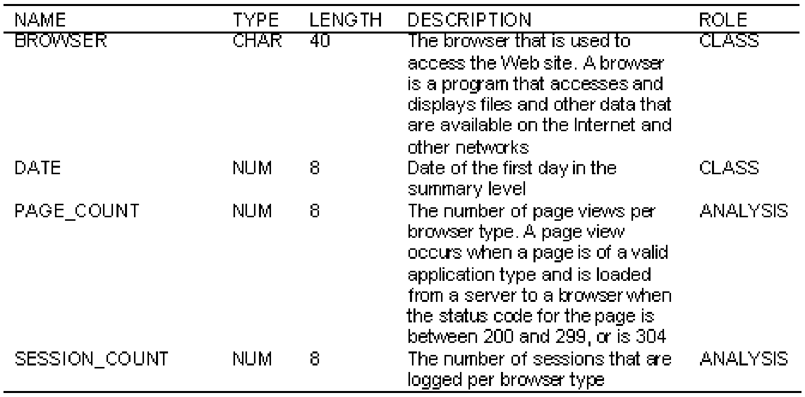
_Day
_Week
_Month
_Qtr
_Year

Description: This data set contains distribution data on the different Web browsers used by visitors navigating the Web site.

Default reports using this data (Group-Name; Report-Name): Platform; Browsers

Processes that use this data set as input: None

Display 19.1 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
BROWSER	CHAR	40	The browser that is used to access the Web site. A browser is a program that accesses and displays files and other data that are available on the Internet and other networks	CLASS
DATE	NUM	8	Date of the first day in the summary level	CLASS
PAGE_COUNT	NUM	8	The number of page views per browser type. A page view occurs when a page is of a valid application type and is loaded from a server to a browser when the status code for the page is between 200 and 299, or is 304	ANALYSIS
SESSION_COUNT	NUM	8	The number of sessions that are logged per browser type	ANALYSIS

Browser_Version_<Summary Level> Data Set

Library: Summary

Data set name: Browser_Version_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

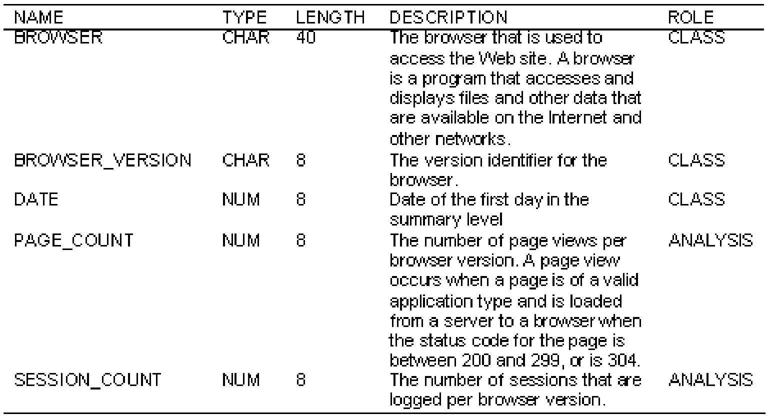
_Day
_Week
_Month
_Qtr
_Year

Description: This data set contains distribution data on the different Web browser versions that are used by visitors navigating the Web site.

Default reports using this data (Group-Name; Report-Name): Platform; Browser Version

Processes that use this data set as input: None

Display 19.2 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
BROWSER	CHAR	40	The browser that is used to access the Web site. A browser is a program that accesses and displays files and other data that are available on the Internet and other networks.	CLASS
BROWSER_VERSION	CHAR	8	The version identifier for the browser.	CLASS
DATE	NUM	8	Date of the first day in the summary level	CLASS
PAGE_COUNT	NUM	8	The number of page views per browser version. A page view occurs when a page is of a valid application type and is loaded from a server to a browser when the status code for the page is between 200 and 299, or is 304.	ANALYSIS
SESSION_COUNT	NUM	8	The number of sessions that are logged per browser version.	ANALYSIS

Daily_Total_<Summary Level> Data Set

Library: Summary

Data set name: Daily_Total_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

_Day

_Week

_Month

_Qtr

_Year

Description: This data set contains Web site traffic metric values that are summarized for the corresponding summary level.

Default reports using this data (Group-Name; Report-Name):

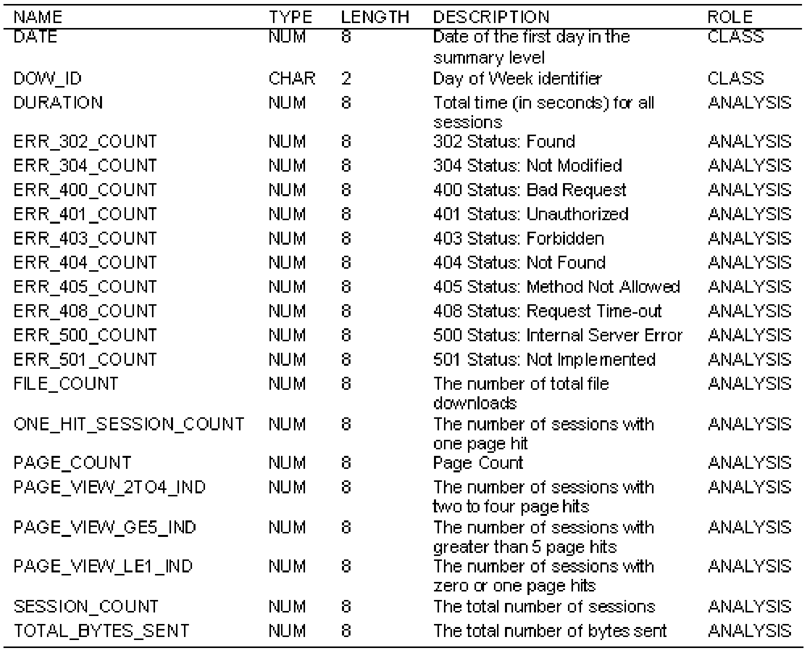
Status Codes; Status Codes

Overview; Site Metrics

Overview; Weekday Metrics

Processes that use this data set as input: Scorecard, dashboard, and segmentation

Display 19.3 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date of the first day in the summary level	CLASS
DOW_ID	CHAR	2	Day of Week identifier	CLASS
DURATION	NUM	8	Total time (in seconds) for all sessions	ANALYSIS
ERR_302_COUNT	NUM	8	302 Status: Found	ANALYSIS
ERR_304_COUNT	NUM	8	304 Status: Not Modified	ANALYSIS
ERR_400_COUNT	NUM	8	400 Status: Bad Request	ANALYSIS
ERR_401_COUNT	NUM	8	401 Status: Unauthorized	ANALYSIS
ERR_403_COUNT	NUM	8	403 Status: Forbidden	ANALYSIS
ERR_404_COUNT	NUM	8	404 Status: Not Found	ANALYSIS
ERR_405_COUNT	NUM	8	405 Status: Method Not Allowed	ANALYSIS
ERR_408_COUNT	NUM	8	408 Status: Request Time-out	ANALYSIS
ERR_500_COUNT	NUM	8	500 Status: Internal Server Error	ANALYSIS
ERR_501_COUNT	NUM	8	501 Status: Not Implemented	ANALYSIS
FILE_COUNT	NUM	8	The number of total file downloads	ANALYSIS
ONE_HIT_SESSION_COUNT	NUM	8	The number of sessions with one page hit	ANALYSIS
PAGE_COUNT	NUM	8	Page Count	ANALYSIS
PAGE_VIEW_2TO4_IND	NUM	8	The number of sessions with two to four page hits	ANALYSIS
PAGE_VIEW_GE5_IND	NUM	8	The number of sessions with greater than 5 page hits	ANALYSIS
PAGE_VIEW_LE1_IND	NUM	8	The number of sessions with zero or one page hits	ANALYSIS
SESSION_COUNT	NUM	8	The total number of sessions	ANALYSIS
TOTAL_BYTES_SENT	NUM	8	The total number of bytes sent	ANALYSIS

Each analysis variable is summed for the corresponding summary level of the data set.

Dashboard_Data Data Set

Library: Summary

Data set name: Dashboard_Data

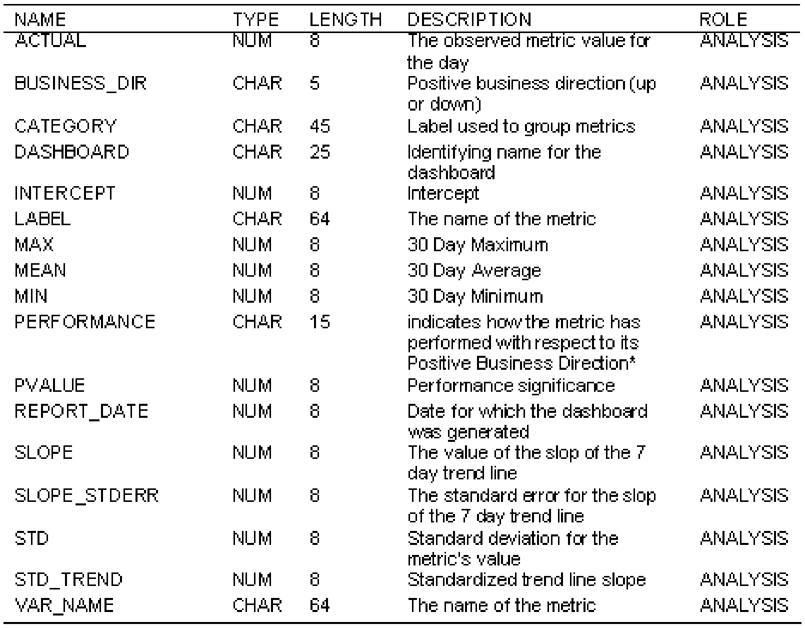
Summary levels: This data set is a repository for the dashboard report data. All defined dashboards are generated each day using the Daily_Total_Day data set. Therefore, the values that are represented in the Dashboard data set are values with the _Day summary level.

Description: This data set contains the values resulting from the dashboard process. The values for each day's dashboard are appended to this data set. The combination of a Report_Date value and the Dashboard name is used to look up a particular dashboard's data for a particular day. Values for this data set are not generated until after 30 days of Web log processing. See also: Dashboard_Metric_History

Default reports using this data (Group-Name; Report-Name): Dashboard; Dashboard

Processes that use this data set as input: None

Display 19.4 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
ACTUAL	NUM	8	The observed metric value for the day	ANALYSIS
BUSINESS_DIR	CHAR	5	Positive business direction (up or down)	ANALYSIS
CATEGORY	CHAR	45	Label used to group metrics	ANALYSIS
DASHBOARD	CHAR	25	Identifying name for the dashboard	ANALYSIS
INTERCEPT	NUM	8	Intercept	ANALYSIS
LABEL	CHAR	64	The name of the metric	ANALYSIS
MAX	NUM	8	30 Day Maximum	ANALYSIS
MEAN	NUM	8	30 Day Average	ANALYSIS
MIN	NUM	8	30 Day Minimum	ANALYSIS
PERFORMANCE	CHAR	15	indicates how the metric has performed with respect to its Positive Business Direction*	ANALYSIS
PVALUE	NUM	8	Performance significance	ANALYSIS
REPORT_DATE	NUM	8	Date for which the dashboard was generated	ANALYSIS
SLOPE	NUM	8	The value of the slop of the 7 day trend line	ANALYSIS
SLOPE_STDERR	NUM	8	The standard error for the slop of the 7 day trend line	ANALYSIS
STD	NUM	8	Standard deviation for the metric's value	ANALYSIS
STD_TREND	NUM	8	Standardized trend line slope	ANALYSIS
VAR_NAME	CHAR	64	The name of the metric	ANALYSIS

*The performance variable will have the value 1 through 5

1 means that the metric's trend is up, and the trend matches the positive business direction.

2 means that the metric's trend is up, and the trend does not match the positive business direction.

3 means that the metric's trend is level (neutral).

4 means that the metric's trend is down, and the trend matches the positive business direction.

5 means that the metric's trend is down, and the trend does not match the positive business direction.

Dashboard_Metric_History Data Set

Library: Summary

Data set name: Dashboard_Metric_History

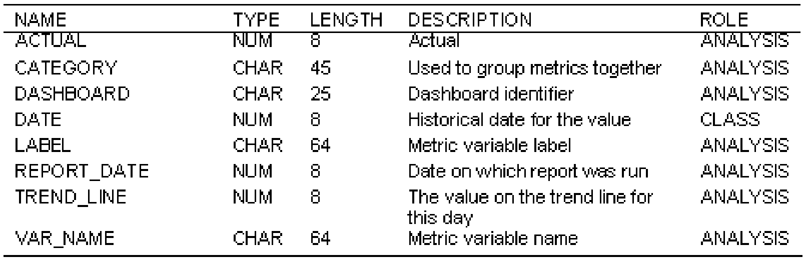
Summary levels: This data set is a repository for the dashboard report data. All defined dashboards are generated each day using the Daily_Total_Day data set. So the values represented in the dashboard data set are values with the _Day summary level.

Description: This data set contains the historical values for each metric in each defined dashboard. The values for each day's dashboard are appended to this data set. The combination of a Report_Date value and the Dashboard name is used to look up a particular dashboard's historical data. Values for this data are not generated until after 30 days of Web log processing. See also: Dashboard_Data

Default reports using this data (Group-Name; Report-Name): Dashboard; Dashboard

Processes that use this data set as input: None

Display 19.5 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
ACTUAL	NUM	8	Actual	ANALYSIS
CATEGORY	CHAR	45	Used to group metrics together	ANALYSIS
DASHBOARD	CHAR	25	Dashboard identifier	ANALYSIS
DATE	NUM	8	Historical date for the value	CLASS
LABEL	CHAR	64	Metric variable label	ANALYSIS
REPORT_DATE	NUM	8	Date on which report was run	ANALYSIS
TREND_LINE	NUM	8	The value on the trend line for this day	ANALYSIS
VAR_NAME	CHAR	64	Metric variable name	ANALYSIS

Hourly_Metrics_<Summary Level> Data Set

Library: Summary

Data set name: Hourly_Metrics_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

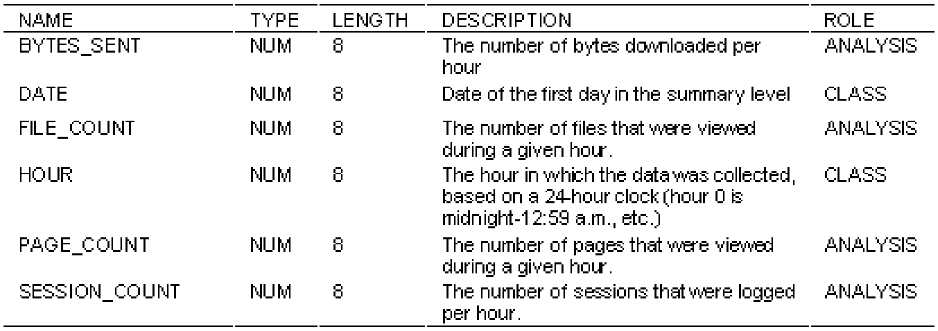
_Day
_Week
_Month
_Qtr
_Year

Description: This data set contains Web traffic metrics per hour for a Web site.

Default reports using this data (Group-Name; Report-Name): Overview; Hourly Metrics

Processes that use this data set as input: None

Display 19.6 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
BYTES_SENT	NUM	8	The number of bytes downloaded per hour	ANALYSIS
DATE	NUM	8	Date of the first day in the summary level	CLASS
FILE_COUNT	NUM	8	The number of files that were viewed during a given hour.	ANALYSIS
HOUR	NUM	8	The hour in which the data was collected, based on a 24-hour clock (hour 0 is midnight-12:59 a.m., etc.)	CLASS
PAGE_COUNT	NUM	8	The number of pages that were viewed during a given hour.	ANALYSIS
SESSION_COUNT	NUM	8	The number of sessions that were logged per hour.	ANALYSIS

Hourly_Status_<Summary Level> Data Set

Library: Summary

Data set name: Hourly_Status_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

_Day

_Week

_Month

_Qtr

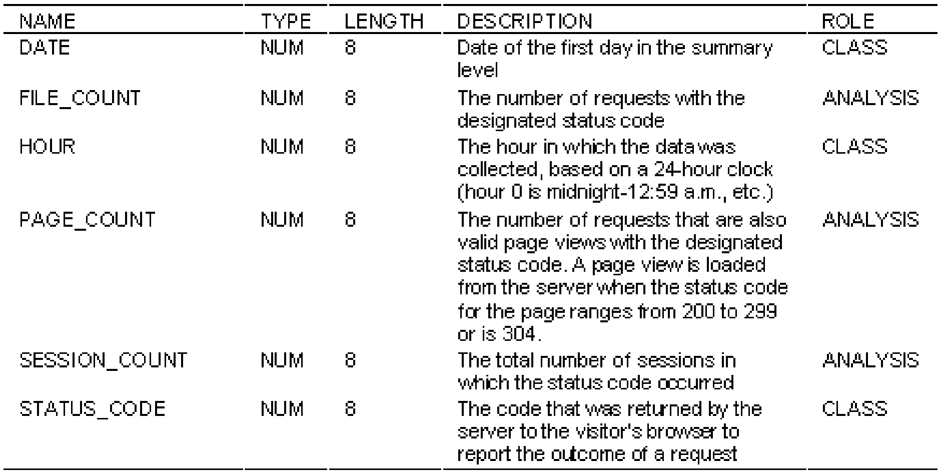
_Year

Description: This data set contains the information about all of the status codes (which contain the results of the browser's requests to the Web server) that were returned by the Web site server for each hour in the specified time interval.

Default reports using this data (Group-Name; Report-Name): Status Codes; Hourly Status Codes

Processes that use this data set as input: None

Display 19.7 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date of the first day in the summary level	CLASS
FILE_COUNT	NUM	8	The number of requests with the designated status code	ANALYSIS
HOUR	NUM	8	The hour in which the data was collected, based on a 24-hour clock (hour 0 is midnight-12:59 a.m., etc.)	CLASS
PAGE_COUNT	NUM	8	The number of requests that are also valid page views with the designated status code. A page view is loaded from the server when the status code for the page ranges from 200 to 299 or is 304.	ANALYSIS
SESSION_COUNT	NUM	8	The total number of sessions in which the status code occurred	ANALYSIS
STATUS_CODE	NUM	8	The code that was returned by the server to the visitor's browser to report the outcome of a request	CLASS

Page_Dates Data Set

Library: Summary

Data set name: Page_Dates

Summary levels: Not Applicable

Description: This data set is a utility data set that is used for the Interactive Funnel report. It contains the variable Dates, which is used to populate a drop-down menu in the SAS Web Analytics Report Viewer.

Default reports using this data (Group-Name; Report-Name): None

Processes that use this data set as input: None

Note: A stored process creates the data set for the Funnel report. △

Page_<Summary Level> Data Set

Library: Summary

Data set name: Page_ <Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

_Day

_Week

_Month

_Qtr

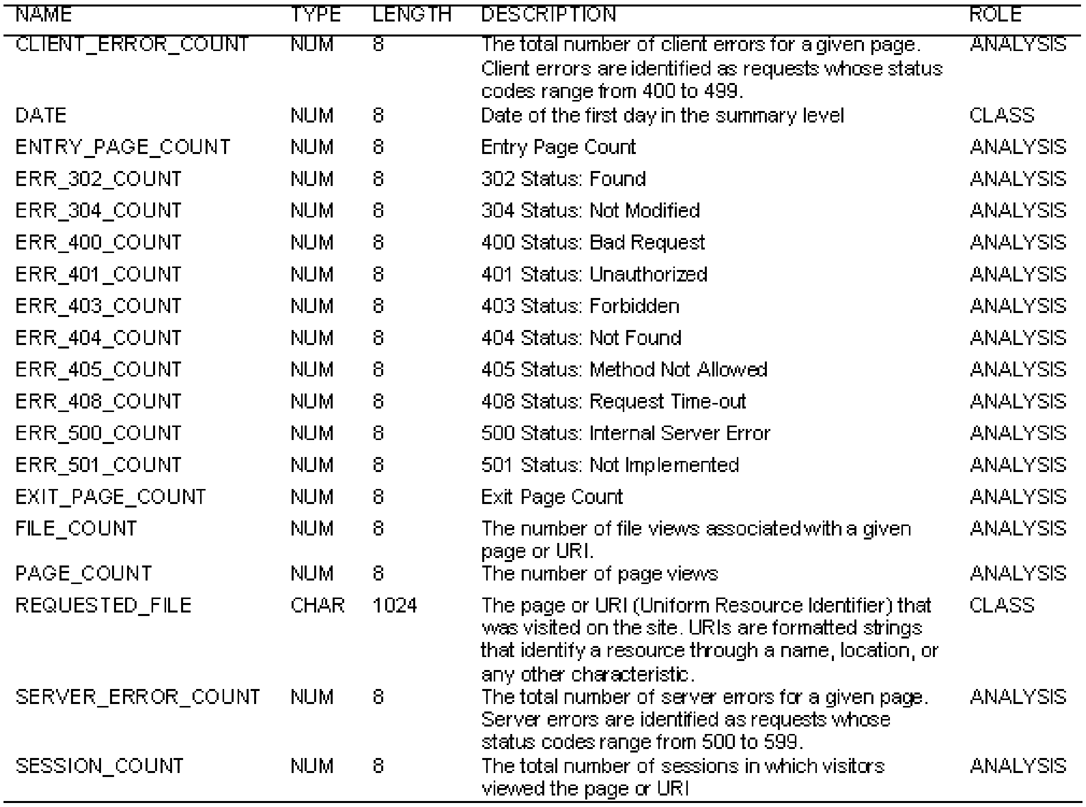
_Year

Description: This data set contains a distribution of the pages that were requested. This data set enables you to identify the pages and families of pages that are most often viewed. You can also view session and error counts and percentages for the pages. This information can be used to define business segments and to modify or optimize paths.

Default reports using this data (Group-Name; Report-Name): Navigation; Page Frequency

Processes that use this data set as input: None

Display 19.8 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
CLIENT_ERROR_COUNT	NUM	8	The total number of client errors for a given page. Client errors are identified as requests whose status codes range from 400 to 499.	ANALYSIS
DATE	NUM	8	Date of the first day in the summary level	CLASS
ENTRY_PAGE_COUNT	NUM	8	Entry Page Count	ANALYSIS
ERR_302_COUNT	NUM	8	302 Status: Found	ANALYSIS
ERR_304_COUNT	NUM	8	304 Status: Not Modified	ANALYSIS
ERR_400_COUNT	NUM	8	400 Status: Bad Request	ANALYSIS
ERR_401_COUNT	NUM	8	401 Status: Unauthorized	ANALYSIS
ERR_403_COUNT	NUM	8	403 Status: Forbidden	ANALYSIS
ERR_404_COUNT	NUM	8	404 Status: Not Found	ANALYSIS
ERR_405_COUNT	NUM	8	405 Status: Method Not Allowed	ANALYSIS
ERR_408_COUNT	NUM	8	408 Status: Request Time-out	ANALYSIS
ERR_500_COUNT	NUM	8	500 Status: Internal Server Error	ANALYSIS
ERR_501_COUNT	NUM	8	501 Status: Not Implemented	ANALYSIS
EXIT_PAGE_COUNT	NUM	8	Exit Page Count	ANALYSIS
FILE_COUNT	NUM	8	The number of file views associated with a given page or URI.	ANALYSIS
PAGE_COUNT	NUM	8	The number of page views	ANALYSIS
REQUESTED_FILE	CHAR	1024	The page or URI (Uniform Resource Identifier) that was visited on the site. URIs are formatted strings that identify a resource through a name, location, or any other characteristic.	CLASS
SERVER_ERROR_COUNT	NUM	8	The total number of server errors for a given page. Server errors are identified as requests whose status codes range from 500 to 599.	ANALYSIS
SESSION_COUNT	NUM	8	The total number of sessions in which visitors viewed the page or URI	ANALYSIS

Page_Status_<Summary Level> Data Set

Library: Summary

Data set name: Page_Status_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

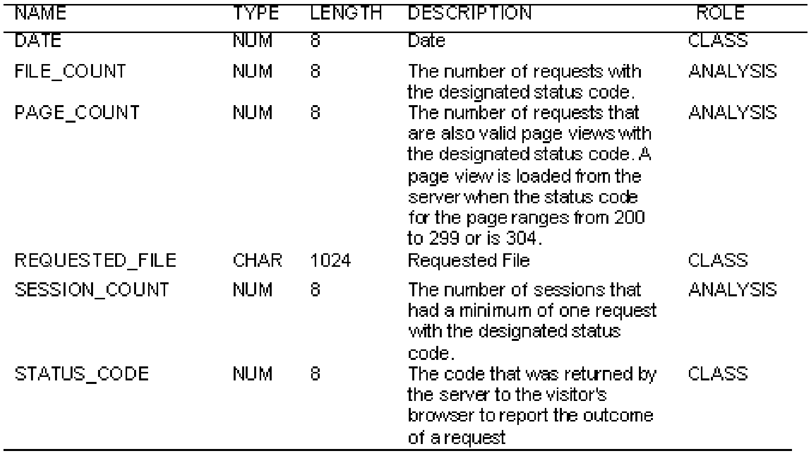
_Day
_Week
_Month
_Qtr
_Year

Description: This data set contains information about the frequency of status codes by the page that was requested on the site. Status codes that range from 400 to 499 indicate the errors that were committed by the visitor's browser. Status codes that range from 500 to 599 indicate the errors that occurred on the Web site server.

Default reports using this data (Group-Name; Report-Name): Status Codes; Page Error Counts

Processes that use this data set as input: None

Display 19.9 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date	CLASS
FILE_COUNT	NUM	8	The number of requests with the designated status code.	ANALYSIS
PAGE_COUNT	NUM	8	The number of requests that are also valid page views with the designated status code. A page view is loaded from the server when the status code for the page ranges from 200 to 299 or is 304.	ANALYSIS
REQUESTED_FILE	CHAR	1024	Requested File	CLASS
SESSION_COUNT	NUM	8	The number of sessions that had a minimum of one request with the designated status code.	ANALYSIS
STATUS_CODE	NUM	8	The code that was returned by the server to the visitor's browser to report the outcome of a request	CLASS

Pathing Data Set

Library: Summary

Data set name: Pathing

Summary levels: None

Description: The Pathing data set is used as input to the path analysis module in the SAS Web Analytics software. The data contains every request that was made by every session for a time interval of up to 30 days. The Requested_File field is used for the Entry Paths reports. The Sequenced_Requested_File field is used for the Entry Paths and the Referrer Entry Paths reports.

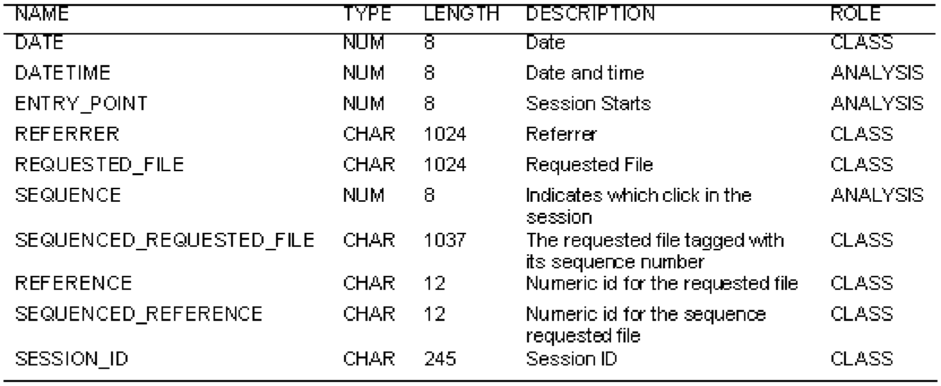
Default reports using this data (Group-Name; Report-Name):

Path Analysis; Entry Paths

Path Analysis; Referrer Entry Paths

Processes that use this data set as input: Pathing, funnel processes

Display 19.10 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date	CLASS
DATETIME	NUM	8	Date and time	ANALYSIS
ENTRY_POINT	NUM	8	Session Starts	ANALYSIS
REFERRER	CHAR	1024	Referrer	CLASS
REQUESTED_FILE	CHAR	1024	Requested File	CLASS
SEQUENCE	NUM	8	Indicates which click in the session	ANALYSIS
SEQUENCED_REQUESTED_FILE	CHAR	1037	The requested file tagged with its sequence number	CLASS
REFERENCE	CHAR	12	Numeric id for the requested file	CLASS
SEQUENCED_REFERENCE	CHAR	12	Numeric id for the sequence requested file	CLASS
SESSION_ID	CHAR	245	Session ID	CLASS

Paths_<Summary Level> Data Set

Library: Summary

Data set name: Paths_<Summary Level>

Summary levels:

R1 – the most current date

R7 – the most current 7 days

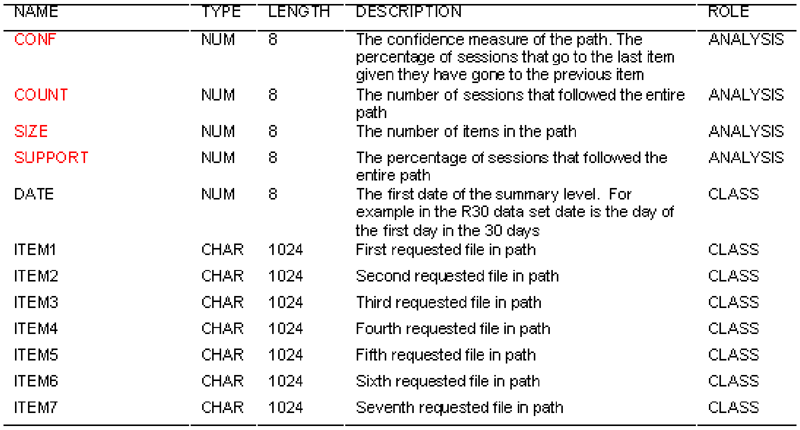
R30 – the most current 30 days

Description: These data sets represent the rolling summaries that generate standard pathing reports.

Default reports using this data (Group-Name; Report-Name): Navigation; Paths

Processes that use this data set as input: None

Display 19.11 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
CONF	NUM	8	The confidence measure of the path. The percentage of sessions that go to the last item given they have gone to the previous item	ANALYSIS
COUNT	NUM	8	The number of sessions that followed the entire path	ANALYSIS
SIZE	NUM	8	The number of items in the path	ANALYSIS
SUPPORT	NUM	8	The percentage of sessions that followed the entire path	ANALYSIS
DATE	NUM	8	The first date of the summary level. For example in the R30 data set date is the day of the first day in the 30 days	CLASS
ITEM1	CHAR	1024	First requested file in path	CLASS
ITEM2	CHAR	1024	Second requested file in path	CLASS
ITEM3	CHAR	1024	Third requested file in path	CLASS
ITEM4	CHAR	1024	Fourth requested file in path	CLASS
ITEM5	CHAR	1024	Fifth requested file in path	CLASS
ITEM6	CHAR	1024	Sixth requested file in path	CLASS
ITEM7	CHAR	1024	Seventh requested file in path	CLASS

Platform_<Summary Level> Data Set

Library: Summary

Data set name: Platform_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

_Day

_Week

_Month

_Qtr

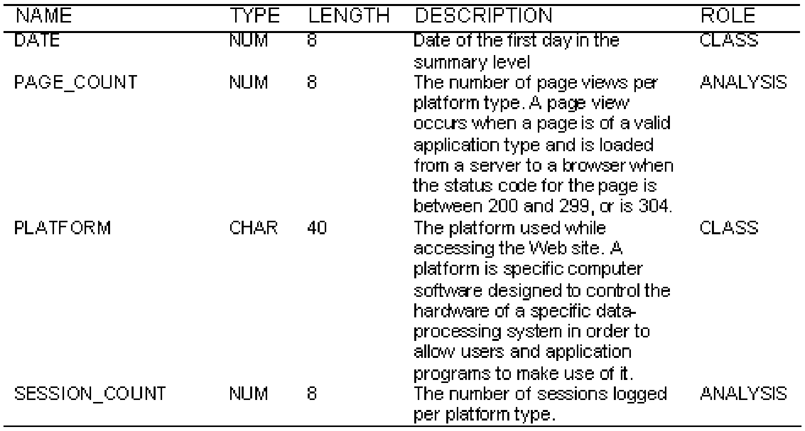
_Year

Description: The Platform_<Summary Level> data set contains a distribution of the different platforms (operating systems) that are used by visitors who navigate the Web site. Your Web site will display differently over different platforms.

Default reports using this data (Group-Name; Report-Name): Platform; Platforms

Processes that use this data set as input: None

Display 19.12 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date of the first day in the summary level	CLASS
PAGE_COUNT	NUM	8	The number of page views per platform type. A page view occurs when a page is of a valid application type and is loaded from a server to a browser when the status code for the page is between 200 and 299, or is 304.	ANALYSIS
PLATFORM	CHAR	40	The platform used while accessing the Web site. A platform is specific computer software designed to control the hardware of a specific data-processing system in order to allow users and application programs to make use of it.	CLASS
SESSION_COUNT	NUM	8	The number of sessions logged per platform type.	ANALYSIS

Referrer_Domain_<Summary Level> Data Set

Library: Summary

Data set name: Referrer_Domain_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

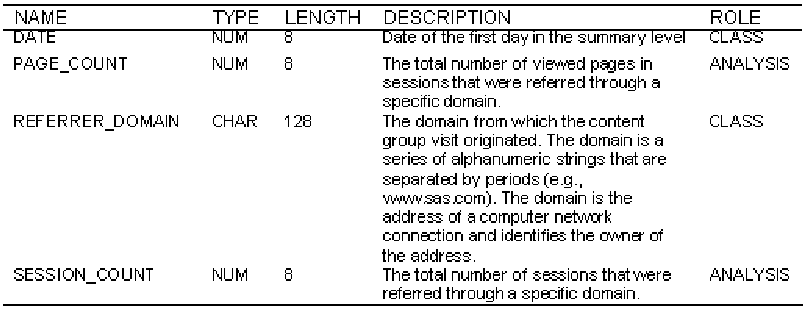
_Day
_Week
_Month
_Qtr
_Year

Description: This data set contains a distribution of referrer domains, which enables you to determine the origin of visitors. This report provides valuable information that can help you evaluate affiliate campaign strategies and assess incoming referral traffic patterns.

Default reports using this data (Group-Name; Report-Name): Referrer; Session Referrer Domains

Processes that use this data set as input: None

Display 19.13 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date of the first day in the summary level	CLASS
PAGE_COUNT	NUM	8	The total number of viewed pages in sessions that were referred through a specific domain.	ANALYSIS
REFERRER_DOMAIN	CHAR	128	The domain from which the content group visit originated. The domain is a series of alphanumeric strings that are separated by periods (e.g., www.sas.com). The domain is the address of a computer network connection and identifies the owner of the address.	CLASS
SESSION_COUNT	NUM	8	The total number of sessions that were referred through a specific domain.	ANALYSIS

Referrer_Entry_Point_<Summary Level> Data Set

Library: Summary

Data set name: Referrer_Entry_Point_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

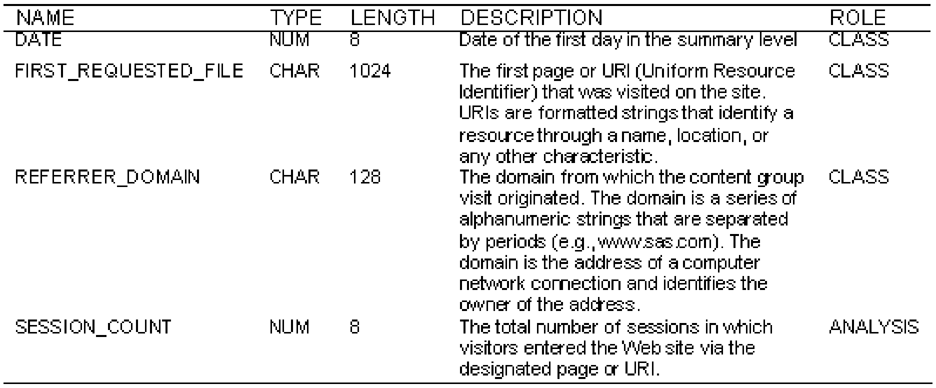
_Day
_Week
_Month
_Qtr
_Year

Description: This data set contains the most common requests that visitors make in order to enter the Web site. This data set enables you to determine which pages are most frequently visited first in a session.

Default reports using this data (Group-Name; Report-Name): Referrer; Referrer Entry Pages

Processes that use this data set as input: None

Display 19.14 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date of the first day in the summary level	CLASS
FIRST_REQUESTED_FILE	CHAR	1024	The first page or URI (Uniform Resource Identifier) that was visited on the site. URIs are formatted strings that identify a resource through a name, location, or any other characteristic.	CLASS
REFERRER_DOMAIN	CHAR	128	The domain from which the content group visit originated. The domain is a series of alphanumeric strings that are separated by periods (e.g., www.sas.com). The domain is the address of a computer network connection and identifies the owner of the address.	CLASS
SESSION_COUNT	NUM	8	The total number of sessions in which visitors entered the Web site via the designated page or URI.	ANALYSIS

Referrer_Page_Status_<Summary Level> Data Set

Library: Summary

Data set name: Referrer_Page_Status_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

_Day

_Week

_Month

_Qtr

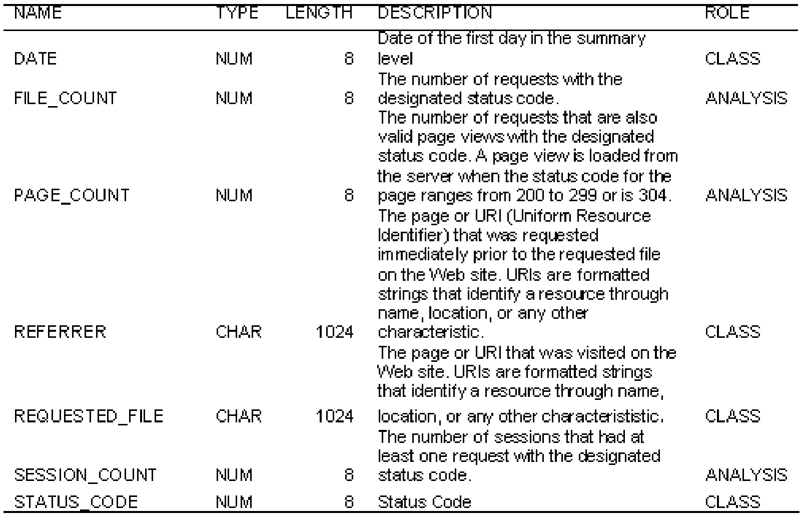
_Year

Description: The Referrer_Page_Status_<Summary Level> data set contains the distribution of the status codes that were returned when visitors requested a page on the Web site. The status codes are tallied by requested page within referring page in order to provide a comprehensive picture of how successfully that the referred visitors navigated to your Web site.

Default reports using this data (Group-Name; Report-Name): Status Codes; Referrer Page Error Codes

Processes that use this data set as input: None

Display 19.15 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date of the first day in the summary level	CLASS
FILE_COUNT	NUM	8	The number of requests with the designated status code.	ANALYSIS
PAGE_COUNT	NUM	8	The number of requests that are also valid page views with the designated status code. A page view is loaded from the server when the status code for the page ranges from 200 to 299 or is 304.	ANALYSIS
REFERRER	CHAR	1024	The page or URI (Uniform Resource Identifier) that was requested immediately prior to the requested file on the Web site. URIs are formatted strings that identify a resource through name, location, or any other characteristic.	CLASS
REQUESTED_FILE	CHAR	1024	The page or URI that was visited on the Web site. URIs are formatted strings that identify a resource through name, location, or any other characterististic.	CLASS
SESSION_COUNT	NUM	8	The number of sessions that had at least one request with the designated status code.	ANALYSIS
STATUS_CODE	NUM	8	Status Code	CLASS

Referrer_Search_Term_<Summary Level> Data Set

Library: Summary

Data set name: Referrer_Search_Term_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

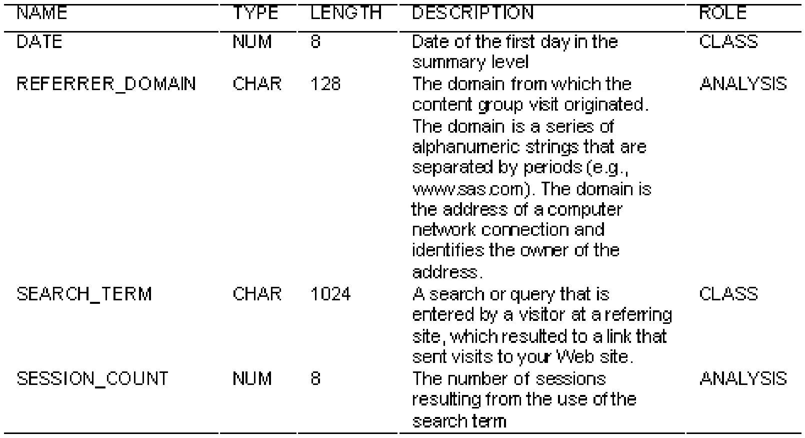
_Day
_Week
_Month
_Qtr
_Year

Description: This data set contains a distribution of search terms that are used by visitors to find your Web site. This report shows the search terms that are used to reach the Web site through a search engine from a referring party. This information can be useful when determining which keywords to specify for search engines in order to direct new traffic to your Web site.

Default reports using this data (Group-Name; Report-Name): Referrer; Search Terms

Processes that use this data set as input: None

Display 19.16 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date of the first day in the summary level	CLASS
REFERRER_DOMAIN	CHAR	128	The domain from which the content group visit originated. The domain is a series of alphanumeric strings that are separated by periods (e.g., www.sas.com). The domain is the address of a computer network connection and identifies the owner of the address.	ANALYSIS
SEARCH_TERM	CHAR	1024	A search or query that is entered by a visitor at a referring site, which resulted to a link that sent visits to your Web site.	CLASS
SESSION_COUNT	NUM	8	The number of sessions resulting from the use of the search term	ANALYSIS

Scorecard_Data Data Set

Library: Summary

Data set name: Scorecard_Data

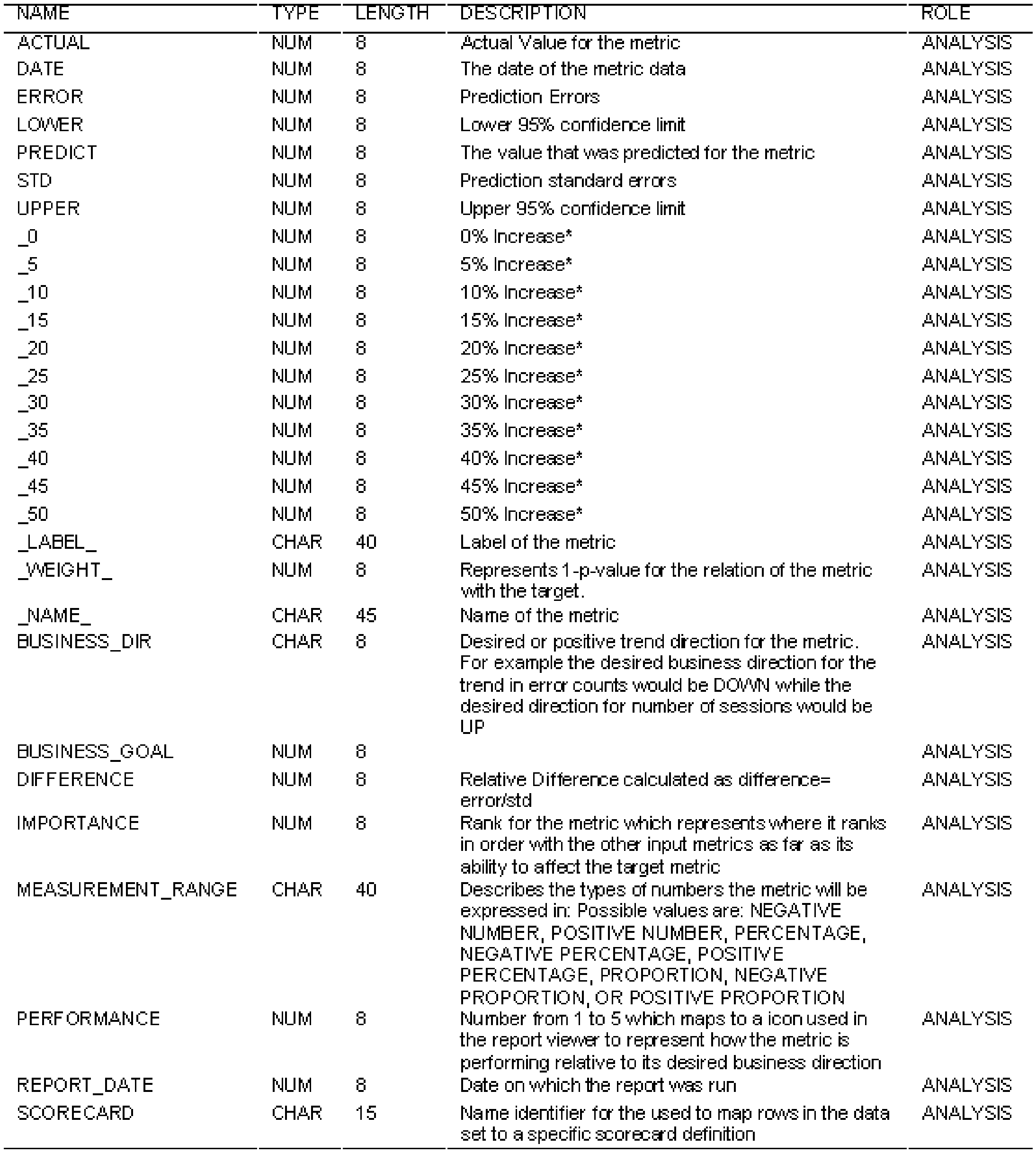
Summary levels: This data set is a repository for the scorecard report data. All defined scorecards are generated each day by using the Daily_Total_Day data set. So the values represented in the scorecard data set are values with the _Day summary level.

Description: This data set contains the values that result from the scorecard process. The values for each day's scorecard are appended to this data set. The combination of a Report_Date value and the Scorecard name is used to look up a scorecard's data for a day. Values for this data set are not generated until after 30 days of Web log processing. See also: Scorecard_Metric_History

Default reports using this data (Group-Name; Report-Name): Scorecard; Scorecard

Processes that use this data set as input: None

Display 19.17 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
ACTUAL	NUM	8	Actual Value for the metric	ANALYSIS
DATE	NUM	8	The date of the metric data	ANALYSIS
ERROR	NUM	8	Prediction Errors	ANALYSIS
LOWER	NUM	8	Lower 95% confidence limit	ANALYSIS
PREDICT	NUM	8	The value that was predicted for the metric	ANALYSIS
STD	NUM	8	Prediction standard errors	ANALYSIS
UPPER	NUM	8	Upper 95% confidence limit	ANALYSIS
_0	NUM	8	0% Increase*	ANALYSIS
_5	NUM	8	5% Increase*	ANALYSIS
_10	NUM	8	10% Increase*	ANALYSIS
_15	NUM	8	15% Increase*	ANALYSIS
_20	NUM	8	20% Increase*	ANALYSIS
_25	NUM	8	25% Increase*	ANALYSIS
_30	NUM	8	30% Increase*	ANALYSIS
_35	NUM	8	35% Increase*	ANALYSIS
_40	NUM	8	40% Increase*	ANALYSIS
_45	NUM	8	45% Increase*	ANALYSIS
_50	NUM	8	50% Increase*	ANALYSIS
LABEL	CHAR	40	Label of the metric	ANALYSIS
WEIGHT	NUM	8	Represents 1-p-value for the relation of the metric with the target.	ANALYSIS
NAME	CHAR	45	Name of the metric	ANALYSIS
BUSINESS_DIR	CHAR	8	Desired or positive trend direction for the metric. For example the desired business direction for the trend in error counts would be DOWN while the desired direction for number of sessions would be UP	ANALYSIS
BUSINESS_GOAL	NUM	8		ANALYSIS
DIFFERENCE	NUM	8	Relative Difference calculated as difference= error/std	ANALYSIS
IMPORTANCE	NUM	8	Rank for the metric which represents where it ranks in order with the other input metrics as far as its ability to affect the target metric	ANALYSIS
MEASUREMENT_RANGE	CHAR	40	Describes the types of numbers the metric will be expressed in: Possible values are: NEGATIVE NUMBER, POSITIVE NUMBER, PERCENTAGE, NEGATIVE PERCENTAGE, POSITIVE PERCENTAGE, PROPORTION, NEGATIVE PROPORTION, OR POSITIVE PROPORTION	ANALYSIS
PERFORMANCE	NUM	8	Number from 1 to 5 which maps to a icon used in the report viewer to represent how the metric is performing relative to its desired business direction	ANALYSIS
REPORT_DATE	NUM	8	Date on which the report was run	ANALYSIS
SCORECARD	CHAR	15	Name identifier for the used to map rows in the data set to a specific scorecard definition	ANALYSIS

*This value is used in the goal-seeking table. This number represents the value that the input metric must reach in order to cause the corresponding percentage change in the target metric.

Scorecard_Metric_History Data Set

Library: Summary

Data set name: Scorecard_Metric_History

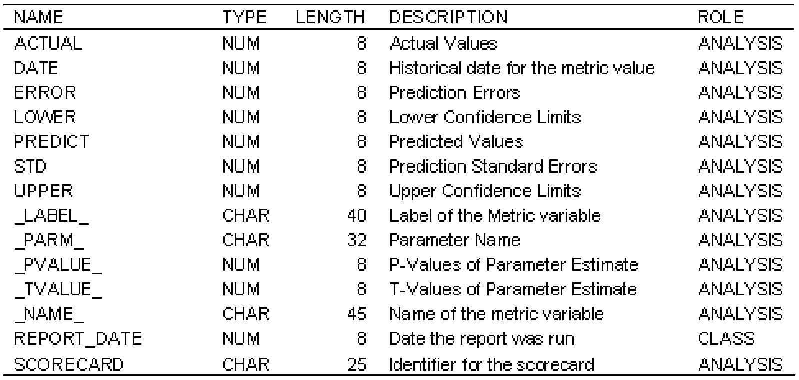
Summary levels: This data set is a repository for the scorecard report data. All defined scorecards are generated each day using the Daily_Total_Day data set. So the values represented in the scorecard data set are values with the _Day summary level.

Description: This data set contains the historical values for each metric in each defined scorecard. The values for each day's scorecard are appended to this data set. The combination of a Report_Date value and the Scorecard name is used to look up a particular scorecard's historical data. Values for this data set are not generated until after 30 days of Web log processing. See also: Scorecard_Data

Default reports using this data (Group-Name; Report-Name): Scorecard; Scorecard

Processes that use this data set as input: None

Display 19.18 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
ACTUAL	NUM	8	Actual Values	ANALYSIS
DATE	NUM	8	Historical date for the metric value	ANALYSIS
ERROR	NUM	8	Prediction Errors	ANALYSIS
LOWER	NUM	8	Lower Confidence Limits	ANALYSIS
PREDICT	NUM	8	Predicted Values	ANALYSIS
STD	NUM	8	Prediction Standard Errors	ANALYSIS
UPPER	NUM	8	Upper Confidence Limits	ANALYSIS
LABEL	CHAR	40	Label of the Metric variable	ANALYSIS
PARM	CHAR	32	Parameter Name	ANALYSIS
PVALUE	NUM	8	P-Values of Parameter Estimate	ANALYSIS
TVALUE	NUM	8	T-Values of Parameter Estimate	ANALYSIS
NAME	CHAR	45	Name of the metric variable	ANALYSIS
REPORT_DATE	NUM	8	Date the report was run	CLASS
SCORECARD	CHAR	25	Identifier for the scorecard	ANALYSIS

Search_Term_<Summary Level> Data Set

Library: Summary

Data set name: Search_Term_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

_Day

_Week

_Month

_Qtr

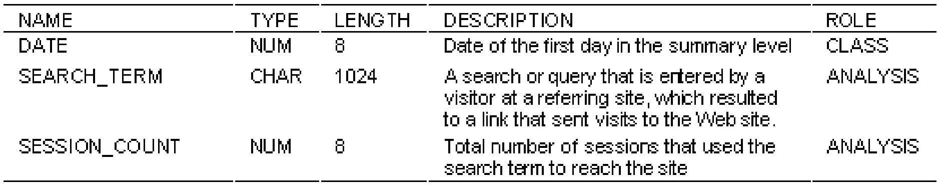
_Year

Description: This data set contains the number of sessions that resulted in the use of a search term for the corresponding summary level.

Default reports using this data (Group-Name; Report-Name): Referrer; Search Terms

Processes that use this data set as input: None

Display 19.19 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date of the first day in the summary level	CLASS
SEARCH_TERM	CHAR	1024	A search or query that is entered by a visitor at a referring site, which resulted to a link that sent visits to the Web site.	ANALYSIS
SESSION_COUNT	NUM	8	Total number of sessions that used the search term to reach the site	ANALYSIS

Status_<Summary Level> Data Set

Library: Summary

Data set name: Status_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

_Day

_Week

_Month

_Qtr

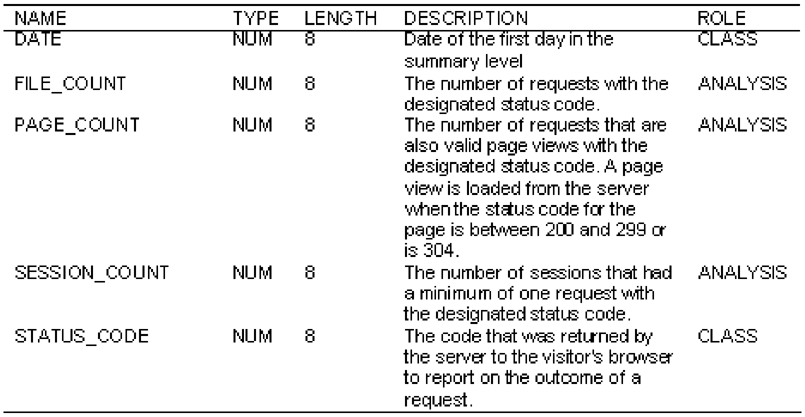
_Year

Description: This data set contains information about the frequency of status codes. Status codes that range from 400 to 499 indicate the errors that were caused by the visitor's browser. Status codes that range from 500 to 599 indicate the errors that occurred on the Web site's server.

Default reports using this data (Group-Name; Report-Name): Status Codes; Status Codes

Processes that use this data set as input: None

Display 19.20 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date of the first day in the summary level	CLASS
FILE_COUNT	NUM	8	The number of requests with the designated status code.	ANALYSIS
PAGE_COUNT	NUM	8	The number of requests that are also valid page views with the designated status code. A page view is loaded from the server when the status code for the page is between 200 and 299 or is 304.	ANALYSIS
SESSION_COUNT	NUM	8	The number of sessions that had a minimum of one request with the designated status code.	ANALYSIS
STATUS_CODE	NUM	8	The code that was returned by the server to the visitor's browser to report on the outcome of a request.	CLASS

Top_Entry_Paths_<Summary Level> Data Set

Library: Summary

Data set name: Top_Entry_Paths_<Summary Level>

Summary levels:

R1 – the most current date

R7 – the most current 7 days

R30 – the most current 30 days

Description: This data set contains a list of the most popular points of entry into the Web site. This report shows the detailed navigation patterns that visitors follow after they reach the Web site. Understanding these navigation patterns can help you interpret the user experience when visiting your Web site by providing the following navigation patterns:

□ the subject areas that attract interest

□ the high site exit paths (paths that lead to site abandonment)

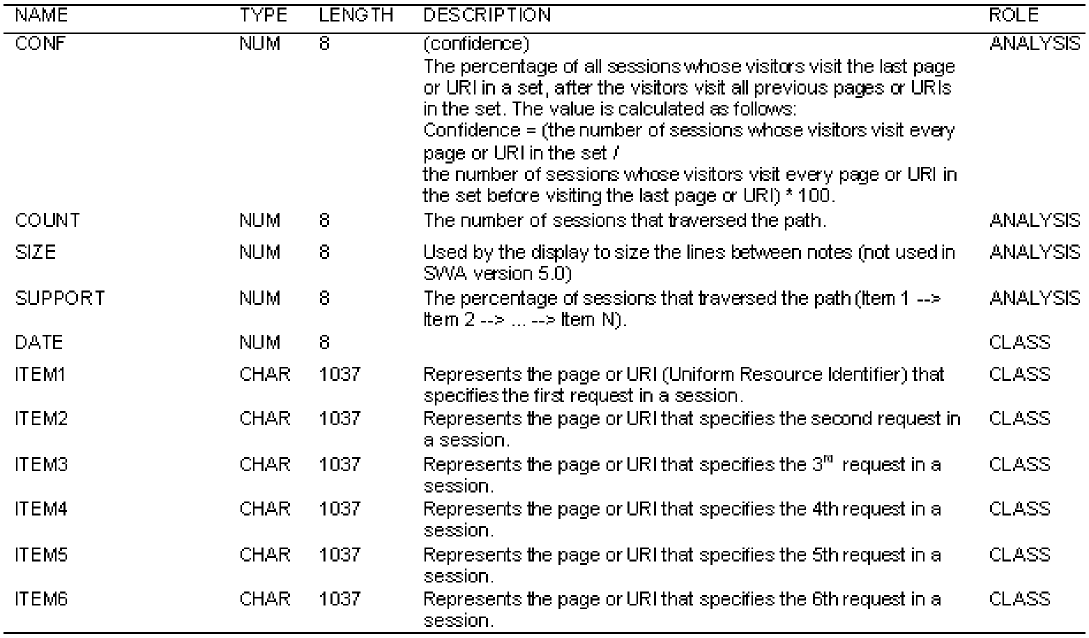
□ the paths to the significant return on investment (ROI) events

□ the most popular links from any entry point

Default reports using this data (Group-Name; Report-Name): Navigation; Entry Paths

Processes that use this data set as input: None

Display 19.21 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
CONF	NUM	8	(confidence) The percentage of all sessions whose visitors visit the last page or URI in a set, after the visitors visit all previous pages or URIs in the set. The value is calculated as follows: Confidence = (the number of sessions whose visitors visit every page or URI in the set / the number of sessions whose visitors visit every page or URI in the set before visiting the last page or URI) * 100.	ANALYSIS
COUNT	NUM	8	The number of sessions that traversed the path.	ANALYSIS
SIZE	NUM	8	Used by the display to size the lines between notes (not used in SWA version 5.0)	ANALYSIS
SUPPORT	NUM	8	The percentage of sessions that traversed the path (Item 1 --> Item 2 --> ... --> Item N).	ANALYSIS
DATE	NUM	8		CLASS
ITEM1	CHAR	1037	Represents the page or URI (Uniform Resource Identifier) that specifies the first request in a session.	CLASS
ITEM2	CHAR	1037	Represents the page or URI that specifies the second request in a session.	CLASS
ITEM3	CHAR	1037	Represents the page or URI that specifies the 3rd request in a session.	CLASS
ITEM4	CHAR	1037	Represents the page or URI that specifies the 4th request in a session.	CLASS
ITEM5	CHAR	1037	Represents the page or URI that specifies the 5th request in a session.	CLASS
ITEM6	CHAR	1037	Represents the page or URI that specifies the 6th request in a session.	CLASS

The default number of items that can exist in a path for the SAS Web Analytics application is 7. The minimum number of sessions that are allowed for 1 day is 10 sessions; for 7 days is 40 sessions, and for 30 days is 100 sessions.

Top_Referrer_Paths_<Summary Level> Data Set

Library: Summary
Data set name: Top_Referrer_Paths_<Summary Level>
Summary levels:

R1 – the most current date

R7 – the most current 7 days

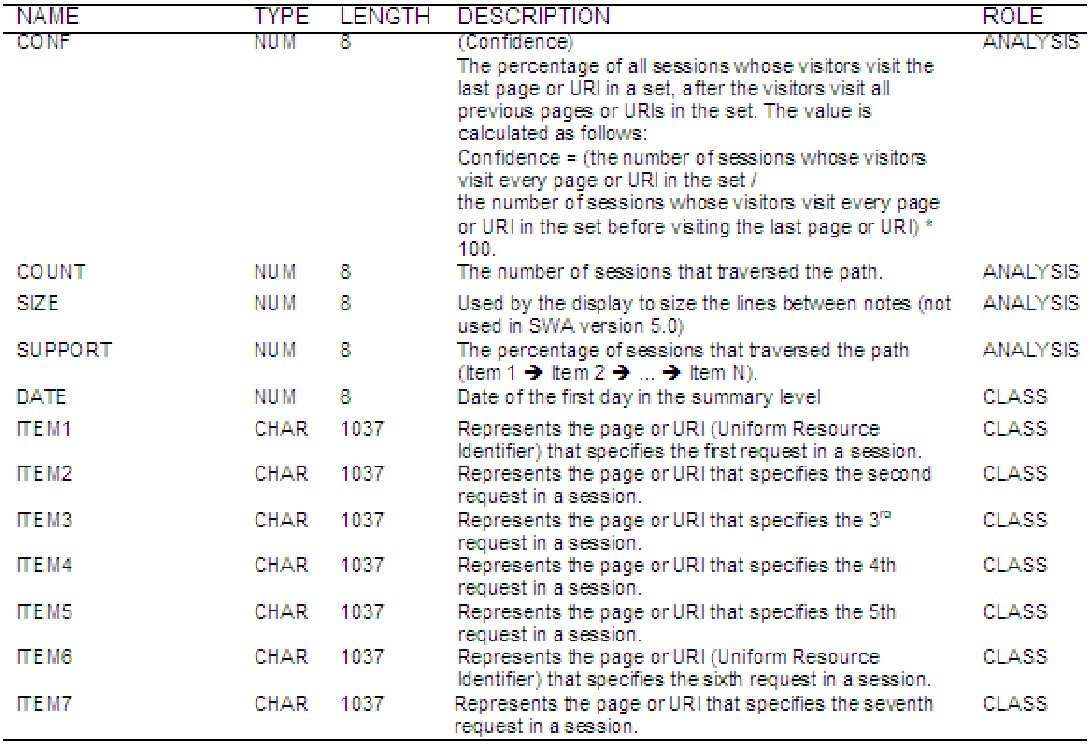
R30 – the most current 30 days

Description: This data set contains a list of the referrers that sent the most traffic to your Web site. This data set is designed to show the detailed navigation patterns that visitors follow after they reach the site from different referring agents. Understanding these navigation patterns can help you to interpret the user experience from each referrer by providing the following information:

- the areas of your Web site that attract the most interest
- the high site exit paths (paths that lead to site abandonment)
- the paths that result in significant ROI (return on investment) events
- the areas of your Web site to which affiliate programs are sending visitors
- which affiliate programs are performing well or performing poorly
- which banner advertisements are performing well or performing poorly

Default reports using this data (Group-Name; Report-Name): Navigation; Referrer Paths
Processes that use this data set as input: None

Display 19.22 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
CONF	NUM	8	(Confidence) The percentage of all sessions whose visitors visit the last page or URI in a set, after the visitors visit all previous pages or URIs in the set. The value is calculated as follows: Confidence = (the number of sessions whose visitors visit every page or URI in the set / the number of sessions whose visitors visit every page or URI in the set before visiting the last page or URI) * 100.	ANALYSIS
COUNT	NUM	8	The number of sessions that traversed the path.	ANALYSIS
SIZE	NUM	8	Used by the display to size the lines between notes (not used in SWA version 5.0)	ANALYSIS
SUPPORT	NUM	8	The percentage of sessions that traversed the path (Item 1 ➔ Item 2 ➔ ... ➔ Item N).	ANALYSIS
DATE	NUM	8	Date of the first day in the summary level	CLASS
ITEM1	CHAR	1037	Represents the page or URI (Uniform Resource Identifier) that specifies the first request in a session.	CLASS
ITEM2	CHAR	1037	Represents the page or URI that specifies the second request in a session.	CLASS
ITEM3	CHAR	1037	Represents the page or URI that specifies the 3rd request in a session.	CLASS
ITEM4	CHAR	1037	Represents the page or URI that specifies the 4th request in a session.	CLASS
ITEM5	CHAR	1037	Represents the page or URI that specifies the 5th request in a session.	CLASS
ITEM6	CHAR	1037	Represents the page or URI (Uniform Resource Identifier) that specifies the sixth request in a session.	CLASS
ITEM7	CHAR	1037	Represents the page or URI that specifies the seventh request in a session.	CLASS

The default number of items that can exist in a path for the SAS Web Analytics application is 7. The minimum number of sessions that are allowed for 1 day is 10 sessions, for 7 days is 40 sessions, and for 30 days is 100 sessions.

Visitor_<Summary Level> Data Set

Library: Summary

Data set name: Visitor_<Summary Level>

Summary levels: Each analysis variable is summed for the summary level of the corresponding data set:

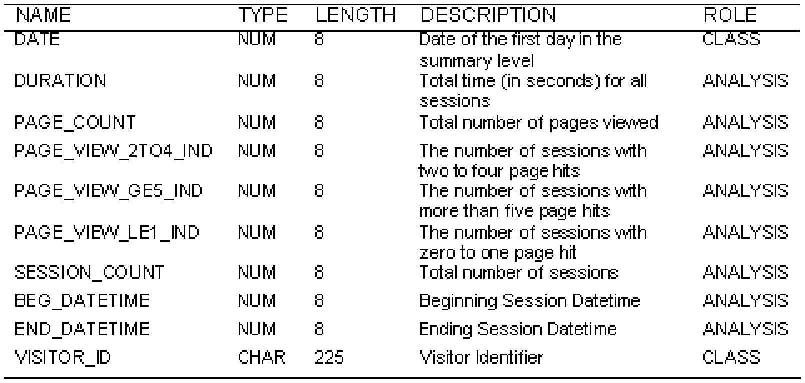
_Day
_Week
_Month
_Qtr
_Year

Description: This data set contains data which identifies the visitors that have the highest activity on your site.

Default reports using this data (Group-Name; Report-Name): Visitor; Unique Visitor

Processes that use this data set as input: None

Display 19.23 Variable Descriptions



NAME	TYPE	LENGTH	DESCRIPTION	ROLE
DATE	NUM	8	Date of the first day in the summary level	CLASS
DURATION	NUM	8	Total time (in seconds) for all sessions	ANALYSIS
PAGE_COUNT	NUM	8	Total number of pages viewed	ANALYSIS
PAGE_VIEW_2TO4_IND	NUM	8	The number of sessions with two to four page hits	ANALYSIS
PAGE_VIEW_GE5_IND	NUM	8	The number of sessions with more than five page hits	ANALYSIS
PAGE_VIEW_LE1_IND	NUM	8	The number of sessions with zero to one page hit	ANALYSIS
SESSION_COUNT	NUM	8	Total number of sessions	ANALYSIS
BEG_DATETIME	NUM	8	Beginning Session Datetime	ANALYSIS
END_DATETIME	NUM	8	Ending Session Datetime	ANALYSIS
VISITOR_ID	CHAR	225	Visitor Identifier	CLASS

Glossary

active session
a session that is still in progress for either of the following reasons: 1) the visitor is currently requesting resources such as pages, pictures, or files at the Web site, or 2) the visitor has not made any requests for a while, but the session timeout (usually 30 minutes) has not been reached.

authenticated realm
a group of Web pages that are accessible only to users who have authenticated themselves in some way. Users typically authenticate themselves by entering a user ID and a password.

bytes received
the number of bytes that a Web server has received from a particular client browser. Most Web server log files do not record bytes received. See also bytes sent.

bytes sent
the total number of bytes that a server has delivered in response to a request. Because of retransmissions and network problems, bytes sent can sometimes be larger than the size in bytes of the resource or file that was received. Bytes sent is sometimes referred to as bytes transferred.

bytes transferred
See bytes sent.

CAM (custom access module)
a SOURCE entry containing SAS code that users can change in order to change how data in a Web server log file is processed by either the Extract program or the Load program.

clickstream analysis
the analysis and interpretation of the actions of Web site visitors. These actions are recorded in the Web log as a chain of time-ordered related events, such as a trail of mouse clicks that a visitor leaves. The purpose of clickstream analysis is to understand and predict the actions of visitors as well as the paths that visitors take through a site. This analysis typically involves data-mining techniques such as identifying sequences and associations.

clickstream reporting
the process of summarizing the actions that are recorded in a Web server log file into various classes, dimensions, or buckets. The summarization is based on the visitors'

URLs, the amount of time spent on each page, and elements of the domain names. This reporting describes demographic information about the visitor population, the site activity rates, and the relative demand for various areas of the site, such as ad banners or links on a page.

content group
a theme or topic that is shared by a group of Web pages. Content groups can be specific products or services, or they can be actions such as buying and selling.

content type
a value that tells a client's Web browser how to interpret and display a transferred object such as an image file, a sound file, or a video file. For example, GIF, JPG, TIFF, MIDI, and WAV are content types. See also MIME (Multipurpose Internet Mail Extensions).

dashboard
a report that shows at a glance the trends for Key Performance Indicators (KPIs) of Web site activity. The dashboard compares KPIs for a specific day with the 30-day average, minimum, and maximum. See also Key Performance Indicator (KPI).

domain
a database of users that has been set up by an administrator by using a specific authentication provider such as LDAP or the host operating system. The domain name should be unique within your enterprise. For example, you should not have a Windows domain and a Unix domain that are both named "SALES". See also authentication domain.

entry page
the first page that a visitor views when entering a Web site.

entry point
the first page that an Internet visitor views when visiting a Web site. In SAS Web Analytics, the entry point page marks the start of a session. See also exit point.

exit page
the last page that a visitor views before leaving a Web site.

exit point
the last page that a visitor views before leaving a Web site. In SAS Web Analytics, the exit point marks the end of a session.

file count
the total number of files that a particular Web site visitor downloads during a session. See also hit, page request.

file hit
See hit.

funnel report
a report that provides a detailed description of any sequential process on a Web site, such as a sequence of Web pages that are visited. For example, a funnel report can be used to determine the page from which users leave a particular sequence of Web pages. The report can also be used to determine how many visitors visit a group of pages in a specific sequence.

hit
the result of a successful request (sent to a Web server) for a resource such as an HTML page, a GIF file, or an executable file. Each hit generates an entry in a Web server log file. By contrast, a page request (a particular type of hit) does not include the objects on the page. Requests for an HTML file and a GIF file are both

considered to be hits, but only the request for the HTML file is typically considered to be a page request. See also page request.

organization
the company, institution, or other collective group with which a Web site visitor is affiliated. The organization is determined by converting the client computer's numeric IP address to the domain name, which usually contains either the company name or a recognizable abbreviation.

organization type
the last segment of a domain name, which identifies the type of organization with which a Web site visitor is affiliated. For example, the organization type .COM indicates a commercial business; .GOV indicates a government organization; and .EDU indicates an educational institution. The organization type can be determined by converting the numeric IP address of the visitor's computer to the domain name. See top-level domain.

page count
the total number of pages identified in a Web server log file. The page count does not include objects on a Web page, such as GIF files or audio files. Page count and page views are synonyms. See also file count, hit.

page request
an attempt to access a Web page. Each page request generates an entry in a log file. Unlike a hit, a page request does not include the objects on the page, such as GIF files and audio files. A hit includes all objects on the page as well as the page itself. See also visit, hit.

page view
See page request.

referrer ID
the URL of the Web page that a visitor clicked on in order to visit the current page.

report definition
a specification that is used for generating a report. A report definition includes information such as the table and level, the names of the variables, the report style, and other attributes.

request
an attempt to access a Web page or a resource on a Web server. A request can be either a page request or a hit. See also page request, hit.

segment
a group of Web site visitors with one or more common attributes that have been identified by a rule. Segments are created by using a type of predictive model called a decision tree. The decision tree uses a set of independent variables to determine whether a visitor will return to the Web site at some time in the future.

session
a period of activity that starts when a visitor first accesses a particular Web site and that ends when the visitor has not performed any actions at that Web site within a specified time interval (usually 30 minutes). A session ID is associated with each session, and the activity that occurs during the session is recorded in a Web server log file.

session ID
a unique number that is assigned to a Web site visitor and which is used to track the visitor's path and the time of entry and exit.

session start

the time of a visitor's request for an entry point page of your Web site. The session start time is recorded in the Web server log file. See also session, entry point.

status code

in a Web server log file, a three-digit code that the server issues to describe the success or failure of a visitor's request for a file from a Web site. A status code between 200 and 299 indicates that the request was successful. A status code of 400 or greater indicates a bad request, an unauthorized request, a page not found, or some other type of failure.

stored process

a SAS program that is stored on a server and which can be executed as requested by client applications. There are two types of stored processes: IOM Direct Interface Stored Processes and SAS Stored Processes.

ubiquitous identifier

a variable whose value remains constant for all of a visitor's clicks in a Web site during a visit. One or more ubiquitous identifiers are specified in the Wbconfig data set of a Web mart and are used to track the clickstream activity of a unique Web site visitor.

unique visitors

the number of individuals who visit your Web site within a specified reporting period (hour, day, week, or month, depending on the report that you select). A unique visitor can have more than one session during the reporting period. See also session.

URL (Uniform Resource Locator)

a character string that is used by a Web browser or other software application to access or identify a resource on the Internet or on an intranet. The resource could be a Web page, an electronic image file, an audio file, a JavaServer page, or any other type of electronic object. The full form of a URL specifies which communications protocol to use for accessing the resource, as well as the directory path and filename of the resource.

visit

an instance of a person using a Web browser to access one or more files on a Web site.

Web mart

a shortened form of the term Web data mart, which refers to a generic data mart that contains Web log information.

Index

Your Turn

If you have comments or suggestions about *SAS Web Analytics 5.2: Administrator's Guide*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **`yourturn@sas.com`**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **`suggest@sas.com`**