THE POWER TO KNOW.

SAS Decision Manager 2.1
Administrator's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *SAS® Decision Manager 2.1: Administrator's Guide*. Cary, NC: SAS Institute Inc.

SAS® Decision Manager 2.1: Administrator's Guide

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

July 2013

SAS provides a complete selection of books and electronic products to help customers use SAS® software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit **support.sas.com/bookstore** or call 1-800-727-3228.

Contents

Chapter 1 • Architecture **1**
Architecture of the SAS Intelligence Platform 1
Client Tier 2
Middle Tier 4
SAS Server Tier 6
Data Tier 11

Chapter 2 • Administrative Tasks **17**
Administrative Objectives 17
Tasks 17
Setting Up 18

Chapter 3 • Users and Groups **21**
Overview of Security Administration 21
Administering SAS Identities for Users 22
Administering Groups and Roles 23
Administering Permissions 30

Chapter 4 • Folders in SAS Management Console **33**
Overview of Folders 33
Directories for Business Rules Metadata and XML Files 34
Assets and Permissions 34
Copying and Pasting 34
Importing and Exporting SAS Packages 35
Deleting Business Rules XML Content 36
Best Practices for Exporting and Importing Objects 36
Importing and Pasting Calculated Items 37

Chapter 5 • SAS Decision Manager and SAS Decision Services **39**
Overview of SAS Decision Manager and SAS Decision Services 39
Life Cycle of a Decision Flow 39
Architecture 40
Data Libraries, Event Variables, and Data Processes 41
Making Models Available to SAS Decision Manager 43
Marking Items for Deployment 44

Chapter 6 • SAS Information Maps for Flows **45**
Identifying Data for Flow Activities 45
The Role of the Information Map in SAS Decision Manager 46
Creating a SAS Information Map for SAS Decision Manager 47
About Custom Properties 53
Understanding the Counts Metadata 75

Chapter 7 • Common Data Model: Concepts **79**
About the Common Data Model 79
How the Common Data Model Is Organized 80
About Extension (_EXT) Tables 90
About the History Table 92
About Lookup Tables 93

Chapter 8 • Implementing the Common Data Model **99**

Overview of Steps . . . 99
Creating the Table Structure of the Common Data Model . . . 100
Setting Up the History Table . . . 103
Customizing the Extension (EXT) Tables . . . 104
Setting Up the Lookup Tables . . . 106
Registering Data Sources . . . 106
Saving, Publishing, and the Common Data Model . . . 107

Chapter 9 • Designing SAS Data Sets . . . **109**
Design SAS Data Sets for Test Cases in SAS Decision Manager . . . 109
Column Names in Event Variables . . . 110

Chapter 10 • The Launcher . . . **113**
Overview of the Launcher . . . 113
Enable Logging for the Launcher . . . 113
Launcher Command Arguments . . . 115
Error Codes . . . 120

Chapter 11 • Backing Up and Restoring Data . . . **123**
The Difference between Backing Up Data and Archiving Data . . . 123
General Strategy for Backing Up Data . . . 123
Backing Up Files Before Installing a New Version . . . 125

Chapter 12 • Improving Performance . . . **127**
Activating the Bulk Load Facility . . . 127
Setting Data Set Options When a Bulk Load Facility Is Not Available . . . 129
Specifying JVM Memory Size . . . 130
Troubleshooting JVM Error Messages . . . 132

Chapter 13 • Macros for Importing and Exporting Business Rules Data . . . **133**
Introduction to the Import and Export Macros . . . 133
Dictionary . . . 134

Chapter 14 • Utilities for Administrators . . . **159**
Release Tables Utility . . . 160
Batch Export and Import Tools . . . 161
Delete Environment Settings Utility . . . 164
Load Treatments . . . 165
Scripts for Managing SAS Decision Manager Flows . . . 170

Chapter 15 • Troubleshooting . . . **203**
Overview of Troubleshooting . . . 203
Problem Definition . . . 204
Error Dialog Box . . . 204
Unexpected Behavior . . . 205
Common Sources of Error . . . 206
Summary of Logs for Troubleshooting . . . 208
Contacting SAS Technical Support . . . 211

Chapter 16 • Optional Client Tier Applications . . . **213**
Overview of Optional SAS Solutions . . . 213
Hardware Requirements for the Client Tier . . . 214
SAS Data Integration Studio . . . 214
SAS Enterprise Guide . . . 215
SAS Enterprise Miner . . . 216

Glossary **219**
Index **225**

Chapter 1

Architecture

Architecture of the SAS Intelligence Platform **1**

Client Tier **2**
SAS Decision Manager 2
SAS Information Map Studio 3

Middle Tier **4**
Overview of the Middle Tier 4
Hardware Requirements 4
SAS Content Server 5
SAS Web Application Server 5

SAS Server Tier **6**
Overview of the SAS Server Tier 6
Hardware Requirements 6
Object Spawner 6
Launcher 7
SAS Workspace Server 8
SAS Stored Process Server 9
SAS Metadata Server 10

Data Tier **11**
Overview of the Data Tier 11
Hardware Requirements 12
Components of the Data Tier 12
Registering Data for SAS Decision Manager 12
SAS Library Resources 13
Defining Library References 14
The autoexec_usermods.sas File 16

Architecture of the SAS Intelligence Platform

The SAS Intelligence Platform is designed to efficiently access large amounts of data, while simultaneously providing timely intelligence to a large number of users. The Platform uses an *n*-tier architecture that enables you to distribute functionality across computer resources, so that each type of work is performed by the resources that are most suitable for the job.

A *tier* in the SAS architecture represents a conceptual category of software components that perform similar types of computing tasks and that require similar types of resources.

Different tiers do not necessarily represent separate computers or groups of computers. More than one computer can be used for a specified tier as well.

You can modify the SAS architecture to meet the demands of your workload. For a large company, the architecture can easily consist of many computers with different operating environments. For prototyping, demonstrations, or very small enterprises, the components for all of the tiers can be installed on a single computer.

The architecture of the SAS Intelligence Platform consists of the following tiers:

Client	The client tier provides users with access to intelligence data and to functionality through web-based interfaces.
Middle	The middle tier includes SAS Content Server and SAS Web Application Server. SAS Content Server stores digital content that is created and used by SAS web applications. SAS Web Application Server provides enhanced monitoring and management, web-based administration, load balancing, and improved availability. The middle tier passes the requests for processing and analysis to the SAS servers on the SAS server tier.
SAS Server	The server tier performs SAS processing on your enterprise data. Several types of SAS servers are available to handle different workload types and processing intensities. The software distributes the processing loads among the various servers so that multiple requests for information by clients can be handled quickly. Specialized functions provided by SAS servers include centralized control and access to a central repository of metadata, and the execution of SAS jobs that are submitted by client applications. Also, a SAS Stored Process Server executes and delivers results in a multi-client environment.
Data	The data tier stores your enterprise data. All of your existing data assets can be used, whether your data is stored in relational database management systems, SAS scalable performance data server, SAS tables, or enterprise resource planning system (ERP) tables.

For more information about the SAS Intelligence Platform, see **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**, “Administration Documentation.”

Client Tier

SAS Decision Manager

Description

SAS Decision Manager is a web application that provides designers with a graphical user interface to create, modify, and deploy decision flows.

Typical User

A typical user is a flow designer.

Files

Log

Within the SAS Decision Manager user interface, use the **Log** tab on the Process node. The **Log** tab displays the SAS logs that are generated by the custom code in a stored process or that are entered as SAS code.

Requirements

There are no special requirements for SAS Decision Manager.

Administrative Notes

There are no notes of administrative interest.

SAS Information Map Studio

Description

SAS Information Map Studio is a Java application that provides a graphical interface to create and manage information maps that are required to create business contexts. An information map is a business view of the data.

Relationship to SAS Decision Manager

The IT professional must initially create one or more SAS Information Maps for anyone who creates flows, such as the flow designer. The SAS Information Map is then assigned to a business context (such as a division). The flow designer must specify a business context to be associated with each flow that is created.

A SAS Information Map for a flow presents the physical data from the database in terms that are appropriate and meaningful to a flow designer in a business environment.

Typical User

Information maps are typically created by an information systems administrator who understands the underlying data and knows how the data should be displayed to the flow analyst.

Files

Executable

The typical location of the executable file is **`C:\Program Files\SASHome\SASInformationMapStudio\4.4`**.

Log

There are no logs of administrative interest.

Requirements

There are no requirements of administrative interest

Administrative Notes

Use SAS Information Map Studio to perform the following task:

- create SAS Information Maps.
- identify data sources for flows.
- create data items for use in SAS Decision Manager.
- set the custom properties of the data items for use in SAS Decision Manager.

- specify whether a data item is visible to users in SAS Decision Manager.
- specify how the summary metadata information for a data item is displayed in SAS Decision Manager.
- organize the data items in a SAS Information Map into folders.
- set the classification for a data item to either **Category** or **Measure**.

To use SAS Information Map Studio, see the SAS Information Map Studio Help that is available within the product. For details about using SAS Information Map Studio in order to support SAS Decision Manager objectives, see Chapter 6, "SAS Information Maps for Flows," on page 45.

For an overview, take an interactive tour at **http://www.sas.com/technologies/bi/tourext/mgmtcnsl_itour_flash.html**.

For administration documentation, see *SAS Intelligence Platform: Desktop Application Administration Guide* (located under "Administration Documentation") at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Middle Tier

Overview of the Middle Tier

The SAS Decision Manager resources on the middle tier provide the analytical processing power for the applications that perform decision activities. For a description of SAS *n*-tier architecture, see "Architecture of the SAS Intelligence Platform" on page 1.

Middle-tier capabilities include enhanced monitoring and management, web-based administration, load balancing, and improved availability. Resources on the middle tier provide most of the business application logic. For example, the middle-tier components perform the following tasks:

- request information from the SAS metadata server when users log on to SAS Decision Manager.
- from the data tier, retrieve information from configured databases to display to users.
- maintain user sessions.
- create the logs.
- trigger the execution of the appropriate internal stored processes on the SAS server tier.
- return the results of the stored processes to appear in SAS Decision Manager.

Hardware Requirements

To view the latest system requirements for your system, access **http://support.sas.com/** and select **Third Party Software Reference** under **Knowledge Base**, or contact SAS Institute Technical Support. Sites in the U.S. and Canada should call (919) 677-8008. Other sites should contact their on-site SAS support personnel or the nearest SAS Institute office.

SAS Content Server

Description

The SAS Content Server is part of the SAS Web Infrastructure Platform. This server stores digital content (such as documents, reports, and images) that is created and used by SAS web applications. For example, the SAS Content Server stores report definitions, as well as images and other elements that are used in reports.

A process called content mapping ensures that report content is stored using the same folder names, folder hierarchy, and permissions that the SAS Metadata Server uses to store corresponding report metadata.

In addition, the SAS Content Server stores documents and other files that are to be displayed in the SAS Information Delivery Portal or in SAS solutions.

Relationship to SAS Decision Manager

To interact with the SAS Content Server, client applications (such as SAS Decision Manager) use Web Distributed Authoring and Versioning (WebDAV) based protocols for access, versioning, collaboration, security, and searching.

Requirements

There are no special requirements for this component.

Administrative Notes

The SAS Content Server Administration Console enables you to manage files and WebDAV folders in the SAS Content Server. To access the console, enter the following URL in your web browser and substitute the server name and port number of your SAS Content Server: **`http://<server:port>/SASContentServer/dircontents.jsp`**

Administrative users can use the browser-based SAS Management Console to create, delete, and manage permissions for folders on the SAS Content Server. Administrative users can also search the SAS Content Server by using industry-standard query syntax, including XML Path Language (XPath) and DAV Searching and Locating (DASL).

The SAS Content Server starts automatically when the web application server is started and depends on the SAS Services Application. The SAS Services Application deploys a set of services called Remote Services that are used by SAS Information Delivery Portal, the SAS Stored Process web application, and other web applications. The SAS Services Application must be started before you start your web application server.

For details about using the SAS Content Server, see *SAS Intelligence Platform: Web Application Administration Guide* at **`http://support.sas.com/documentation/onlinedoc/intellplatform/index.html`**.

SAS Web Application Server

SAS Web Application Server is an embedded middle-tier server that is based on VMware vFabric tc Server. External third-party application servers are neither required nor supported.

For more information, see *SAS Intelligence Platform: Middle-Tier Administration Guide*at **`SAS Intelligence Platform`**.

SAS Server Tier

Overview of the SAS Server Tier

SAS Decision Manager resources on the SAS server tier include several types of SAS servers that handle different workload types and different processing intensities. For a description of SAS *n*-tier architecture, see "Architecture of the SAS Intelligence Platform" on page 1.

See also *Server Soup: Understanding the SAS®9 Client/Server Recipe* at **http://support.sas.com/rnd/papers/sugi30/ServerSoup30_040505.pdf**

Hardware Requirements

To view the latest system requirements for your system, access **http://support.sas.com** and select **Install Center** under **Knowledge Base**, or contact SAS Technical Support. Sites in the U.S. and Canada should call (919) 677-8008. Other sites should contact on-site SAS support personnel or the nearest SAS office.

Object Spawner

Description

The object spawner is a process that runs on the SAS server host and listens for requests, including requests from the SAS Decision Manager application server. When a request is received, the object spawner accepts the connection and sends a job to the workspace server or to the stored process server.

Requirements

The object spawner requires the following software to be installed on the computer that is assigned to the SAS server tier:

- SAS 9 Software or later
- SAS Integration Technologies
- SAS Metadata Server

Note: SAS/SECURE is an optional component.

Files

Executable

ObjectSpawner.bat (Windows) or ObjectSpawner.sh (UNIX) is typically located in <*config-dir*> **Lev1\ObjectSpawner** on the SAS server computer.

Configuration

metaConfig.xml is located in <*config-dir*>**\Lev1\ObjectSpawner**. The configuration file contains the name of the metadata server and the login credentials for the SAS Trusted User.

Log

Objspawn.log is located in **\ObjectSpawner\Logs**. You can view the object spawner log file in order to troubleshoot connections between the client and server

computers. The object spawner log file is not created by default. To create the object spawner log file, see *System Administration Guide* (located under **Administration Documentation**) at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Administrative Notes

The object spawner must be refreshed to include any changes that are made to autoexec_usermods.sas and sasv9.cfg.

Windows

See *System Administration Guide* (located under **Administration Documentation**) at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

UNIX

See **http://support.sas.com/rnd/itech/doc9/admin_oma/sasserver/startserv/sp_suunx.html** for information about starting and stopping under UNIX..

Launcher

Description

The Launcher is a transparent Java application that is invoked by the job scheduling program in order to run a job on the SAS Application Server. A job can be one or more test cases that are generated by SAS Decision Manager. A job can also be used to promote flows from test to production. A test case job can consist of a single test case or all test cases for a particular flow.

Relationship to SAS Decision Manager

The job scheduling program can execute only Java applications or SAS programs, and cannot run directly on the SAS Application Server. Therefore, in order to access the SAS Application Server, the job scheduling program uses the Launcher.

Files

Executable

The executable for the Launcher application is sasdcmlauncher.exe. On Windows, this file is typically located in **C:\Program Files\SASHome\SASDecisionManagerLauncher\6.2**. On UNIX, it is typically located in *<!sasroot>* **/SASDecisionManagerLauncher/6.2** .

Log

To create a log file every time you execute a job flow, open sasdcmlauncher.ini. Add one or more of the following options to the beginning of the list of options:

-Dma.launcher.log=<*logfilename*>

Creates a log file with the specified name. If <*logfilename*> is left blank, the filename is created based on the current time. By default, the log file is created in the home directory of the Launcher application.

-Dma.launcher.logdir=<*subdir*>

Creates a log file in a different directory, specified relative to the home directory of the Launcher application.

-Dma.launcher.abslogdir=<*absdir*>
Creates a log file in a different directory, specified as an absolute path on the machine.

To enable command logging, open sasdcmlauncher.ini and add the one of the following options at the beginning of the list of options:

-Dma.launcher.commandlogmode=
Enables command logging in Append mode.

-Dma.launcher.commandlogmode=replace
Enables command logging in Replace mode.

Command log files are written to the same directory as general log files. Command log filenames are created based on the type of operation, the object name or ID, and the last directory name in the path.

Requirements

There are no special requirements to run the Launcher.

Administrative Notes

The Launcher is typically launched in batch mode.

SAS Workspace Server

Description

The SAS Workspace Server enables client applications to submit SAS code to a SAS session that uses an application programming interface (API). Your environment can include one or more workspace servers.

Relationship to SAS Decision Manager

When IT professionals process a flow, SAS code is generated by the computation engine on the middle tier, and is submitted to a workspace server.

Requirements

To view the latest system requirements for your environment, see *System Administration Guide* (located under **Administration Documentation**) at `http://support.sas.com/documentation/onlinedoc/intellplatform/index.html`.

Files

Executable
You can find the name and location of your workspace servers by using the SAS Server Manager in SAS Management Console.

Log
Workspace servers are not initially configured to produce log files. For troubleshooting purposes, workspace server logs can be helpful by capturing calls that are made to the database. In these situations, you can use the alternative logging configuration file (logconfig.trace.xml) that is provided in each workspace server's configuration directory. If a log is created, the default location is <*config-dir*>`\Lev1\SASApp\WorkspaceServer\Logs` .

For information about managing logs for the workspace server, see *System Administration Guide* (located under **Administration Documentation**) at `http://`

support.sas.com/documentation/onlinedoc/intellplatform/index.html.

Administrative Notes

There are no notes of administrative interest.

SAS Stored Process Server

Description

The SAS Stored Process Server executes and delivers results from SAS stored processes in a multi-client environment. SAS stored processes are SAS programs that are stored centrally and that can be executed by business users and by client programs on demand.

The SAS Stored Process Server performs work that is similar to the workspace server. For information about the difference between the two servers, see *Application Server Administration Guide* (located under **Administration Documentation**) at http://support.sas.com/documentation/onlinedoc/intellplatform/index.html.

Relationship to SAS Decision Manager

Stored processes are used in SAS Decision Manager to perform computations.

Requirements

See *Stored Process Software Requirements* at http://support.sas.com/rnd/itech/doc9/dev_guide/stprocess/requires.html.

Files

Executable

In SAS Management Console, the initial stored process server is configured as a load-balancing server named SASApp-Stored Process Server. Find the name and location of the SAS Stored Process Server by using the Server Manager in SAS Management Console.

Configuration

sasv9.cfg is located in <*config-dir*> **\Lev1\SASApp\StoredProcessServer** and calls sasv9.cfg that is located in SASApp.

Log

The stored process server log is located in <*config-dir*>**\Lev1\SASApp\StoredProcessServer\logs** . Log parameters are specified in sasv9_StorProcSrv.cfg.

For information about managing logs, see *System Administration Guide* (located under **Administration Documentation**) at http://support.sas.com/documentation/onlinedoc/intellplatform/index.html.

Administrative Notes

Each stored process server process handles multiple users, and by default each server uses multiple server processes. A load-balancing algorithm distributes client requests between the server processes.

By default, the object spawner starts the processes of the stored process server by authenticating the SAS Spawned Servers user ID, **sassrv**.

For information about load balancing, see *Application Server Administration Guide* (located under **Administration Documentation**) at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

SAS Metadata Server

Description

The SAS Metadata Server controls access to a central repository of metadata that is shared by all of the applications that run as part of SAS Decision Manager.

Relationship to SAS Decision Manager

Metadata for the following objects is generated and stored for SAS Decision Manager:

- information maps that display the physical source data as business data that is used by business users to query a data warehouse
- intermediate tables that are generated when you create flows by using SAS Decision Manager
- decision history tables
- authorization rules for objects (such as information maps, business contexts, and flows) that are used by SAS Decision Manager
- user IDs and passwords for SAS Decision Manager that are used to access a third-party server (for example, an Oracle database server)
- SAS stored processes that update and execute nodes
- reports about SAS Decision Manager that are accessible by using SAS Management Console

Requirements

There are no special requirements.

Files

Configuration

sasv9_MetadataServer.cfg is located in <*config-dir*> **\Lev1\SASMeta\MetadataServer** and points to Sasv9.cfg in <*config-dir*>**\Lev1\SASMeta**. Sasv9.cfg points to the default SAS configuration file at **C:\Program Files\SASHome\SASFoundation\9.4\sasv9.cfg**.

The metadata configuration file contains information about accessing a metadata server. The spawner uses the information that is contained in the configuration file to connect to a metadata server and to read the appropriate server definitions. To enable the spawner to connect to and read the appropriate metadata from a metadata server, you must specify the appropriate login information in the metadata configuration file.

Log

MetadataServer#d#b#y.log is located in <*config-dir*>**\Lev1\SASMeta\MetadataServer\logs**. This log file contains information about users who connect to the SAS Metadata Server . For example, this log might show the status of the SAS Metadata Server and might indicate whether a request from a client application was successful. The log entries also contain information that is useful for diagnosing server start-up problems, diagnosing authorization failures, and debugging method calls.

According to the SAS Metadata Server options that you specify at invocation, the SAS Metadata Server can write the following categories of information to the SAS log:

- the start and stop information for the SAS Metadata Server:
  - the user ID
  - the SAS long version number
  - the SAS Metadata Model version
  - the directory where the SAS Metadata Server was started
  - configuration file options
- user connection and disconnection events
- creation and deletion of a repository
- opening, closing, pausing, resuming, and refreshing events
- errors, such as these:
  - task and thread exceptions
  - memory allocation errors
  - I/O errors
  - application logic errors
  - authentication errors
  - authorization failures
- authentication events
- XML input strings and XML output strings
- traces that are invoked by the debugging options in omaconfig.xml

Administrative Notes

For detailed documentation, access the SAS Knowledge Base / Product Documentation page for SAS Metadata Server at **`http://support.sas.com/documentation/onlinedoc/metadatasrvr/index.html`**.

Data Tier

Overview of the Data Tier

The data tier is a critical structure in the SAS Decision Manager architecture. For a description of SAS *n*-tier architecture, see "Architecture of the SAS Intelligence Platform" on page 1.

Information maps point to data that resides on the data tier. For example, you might want to attribute orders to particular groups. A decision is made for each row of the input table, and an output table is created that contains the processed rows. The input tables and the output tables reside on the data tier. Decision history is also contained in the data tier.

The Web Infrastructure Platform data server shares session information across resources on the middle tier.

Note: The Web Infrastructure Platform data server is experimental for SAS Decision Manager 2.1

Hardware Requirements

To view the latest requirements for your system, access **http://support.sas.com** and select **Third-Party Software Reference** under **Knowledge Base**, or contact SAS Technical Support. Sites in the U.S. and Canada should call (919) 677-8008. Other sites should contact on-site SAS support personnel or the nearest SAS office.

To learn more about the currently supported databases, see the system requirements document that is delivered with your release of SAS Decision Manager software.

When a new release of your database management software (such as Oracle) becomes available, contact on-site SAS support personnel to verify the compatibility of the new release with SAS software.

Components of the Data Tier

Your data tier might include several sources of data for SAS Decision Manager processes. The sources of data are determined by the objectives of your organization and by company resources.

You can access almost any type of data source by using SAS LIBNAME statements.

For example, SAS can access the following native data sources:

- SAS data sets, which are analogous to relational database tables

SAS/ACCESS provides direct access to the following relational databases:

- DB2
- Microsoft SQL Server
- Oracle
- Teradata

Note: For the specific versions of each database that is supported, see your on-site SAS support personnel.

A database should be configured for use by SAS Business Rules Manager on the data tier.

For information about SAS data sets, see *Data Administration Guide* (located under **Administration Documentation**) at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

For details about SAS/ACCESS, see *Data Administration Guide* (located under **Administration Documentation**) at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Registering Data for SAS Decision Manager

What Data Sources Do I Need to Register?

The data sources that must be registered to the SAS Metadata Repository for access by SAS Decision Manager include the following tables:

- any database source for SAS Decision Manager data (such as Oracle or Teradata) that is referenced by a SAS Information Map
- the common data model tables
- exported output from a batch simulation
- (optional) libraries to contain exported data, lists, and custom details. These libraries are specified in the business context

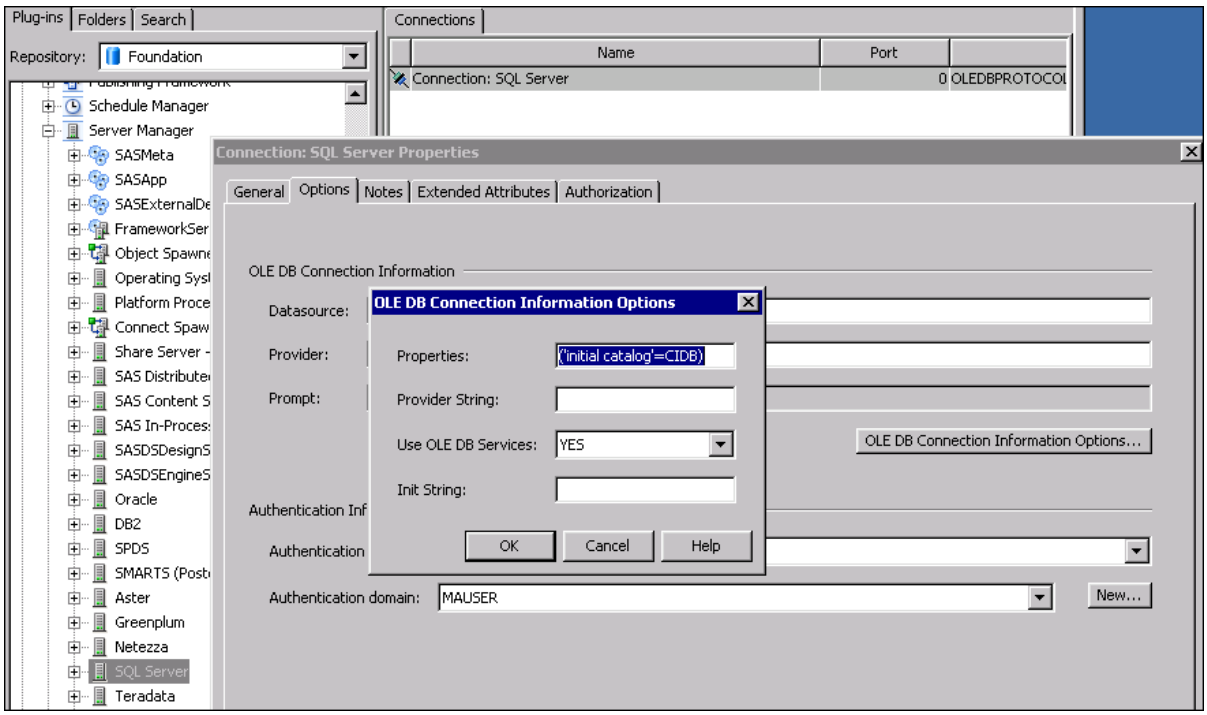
Mapping SQL Server to Multiple Databases

If a user ID in SQL Server is mapped to more than one database, specify the name of the initial database in the OLE DB Connection Information Options for the server. To specify the name of the initial database, take the following steps:

1. Select the SQL server name from the Server Manager plug-in in SAS Management Console.
2. Right-click the connection name and select **Properties**.
3. On the **Options** tab, click **OLE DB Connection Information Options**.
4. In the **Properties** field, enter the following code:

```
('initial catalog'=database-name)
```

Display 1.1 *Initial Catalog Option*



SAS Library Resources

Overview of SAS Library Resources

SAS Decision Manager requires several SAS libraries that contain SAS metadata, holding tables, and operational data. Most of the required libraries are defined when the

product is installed. The following library locations are specified on the **Settings** tab of the business context.

- Decision test data grid input library
- Decision test data grid output library
- Reporting libref

The metadata library is specified on the **Metadata** tab of the business context.

Verify that all SAS Decision Manager users, including saswbadm, have both Read and Write access to the physical location of the library.

You can specify the location of a libref in any preferred physical path. The physical path is represented in the following sections as <*plan-area*>.

Specifying the MATables Library

The MATables libref specifies the location of the holding tables that are created by various stored processes. The MATables libref also specifies the location of the holding tables for metadata that is generated by SAS Decision Manager when a flow is created. These holding tables are automatically deleted when they are no longer needed.

Here is an example of the LIBNAME statement that defines the MATables library:

LIBNAME MATables <*plan-area*> ;

This is the default location of the MATables library:<*config*>**/Lev1/Applications/SASDecisionManager6.2/data/MATables**.

Specifying the MAMisc Library

Here is an example of the LIBNAME statement that defines the MAMisc library:

LIBNAME MAMisc <*plan-area*>;

This is the location of the MAMisc library: <*config*>**/Lev1/Applications/SASDecisionManager6.2/data/MAMisc**.

The MAMisc libref specifies the location of various tables that contain information such as sequence numbers and task lists. It also specifies the location of the defined SAS formats that are used by SAS data sets.

Defining Library References

Overview

Define the required libraries for accessing SAS Decision Manager data by using the Data Library Manager in SAS Management Console. For more information, see *Connecting to Common Data Sources* in *SAS Intelligence Platform: Data Administration Guide* at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Note: Although you can specify some library references (librefs) by editing the autoexec_usermods.sas file, these libref values are used only if no values for librefs have been specified in SAS Management Console. For more information, see "The autoexec_usermods.sas File" on page 16.

Configuring a SAS SPD Server

During installation, the SAS SPD Server creates a library parameter file called libnames.parm. The libnames.parm file contains the LIBNAME domain definitions for

the data tables that are stored on the SAS SPD Server. These tables include the tables of the published flow data in the common data model.

Any other data tables (for example, MATABLES and TMP) that are stored on a SAS SPD Server must have libraries that are configured to use SPD Server data as well.

These examples define libraries called cicom (common data model), mafunc (customer data), matble (MATables), and ematmp (temp tables) during a typical SAS SPD Server installation:

```
LIBNAME=cicom pathname=/sanr52/chmeta roptions="datapath=('/sanr52/chspd')
  indexpath=('/sanr52/chindex')" DYNLOCK=YES;

LIBNAME=mafunc pathname=/sanr51/spdma2/meta roptions="datapath=
  ('/sanr51/spdma2/data/sanr52/spdma2/data')indexpath=('/sanr52/spdma2/index')";

LIBNAME=matble pathname=/sanr52/spdmatbl/meta roptions="datapath=
  ('/sanr52/spdmatbl/data')indexpath=('/sanr52/spdmatbl/index')";

LIBNAME=ematmp pathname=/sanr52/ematmp;
```

If you define any custom formats in the SAS SPD Server, then you need to define a library that references the custom formats. Here is an example:

```
LIBNAME formats SASSPDS IP=yes LIBGEN=yes schema="formats" USER="anonymous"
HOST='serveruno.unx.sas.com' Serv='5190';
```

Use the spdsserv.parm file to control other configuration options for the SAS SPD Server. Here is an example:

```
spdsserv.parm
SORTSIZE=1024M;
INDEX_SORTSIZE=256M;
GRPBYROWCACHE=128M;
BINBUFSIZE=32K;
INDEX_MAXMEMORY=256M;
WORKPATH="/sanr52/spdtmp";
COREFILE;
SEQIOBUFMIN=64K;
RANIOBUFMIN=4K;
MAXWHTHREADS=16;
MAXSEGRATIO=75;
WHERECOSTING;
RANDOMPLACEDPF;
MINPARTSIZE=1024M;
TMPDOMAIN='EMATMP';
```

Note: SAS recommends that you set the maximum number of threads (MAXWHTHREADS) to 16. This value prevents the SAS SPD Server from launching large numbers of unused concurrent threads that could increase run-time processing. For more information, see Chapter 12, “Improving Performance,” on page 127.

The autoexec_usermods.sas File

Overview of the autoexec_usermods.sas File

The autoexec_usermods.sas file specifies some options in the SAS environment for all SAS sessions that are created by the object spawner and used by SAS Decision Manager. SAS sessions include workspace server sessions and calls to the SAS Stored Process Server. This file can be edited to specify the following options:

- SAS options that specify the search order for locating library resources. For more information, see "Specifying the Search Order for Locating Library Resources" on page 16.

The library reference values that are specified in autoexec_usermods.sas are used only if the libraries have not been defined in SAS Management Console.

By default, autoexec_usermods.sas is installed in the configuration directory for the main SAS server. The configuration directory is typically located at **.../Lev1/<*SASApp*>/WorkspaceServer** , where *SASApp* is the default context server.

Note: Updating the mausrexp.sas macro with librefs for database export files is an alternative to including these librefs in the autoexec_usermods.sas file. This method has the advantage of not disrupting every spawned SAS session, which invokes the autoexec_usermods.sas. Having the librefs in the mausrexp.sas file has less performance impact if there is a problem with resolving a libref to a single database export. The mausrexp.sas file is normally found under **C:\Program Files\SASHome\SASFoundation\9.4\dcmprcssvr\sasmacro**.

Specifying the Search Order for Locating Library Resources

Several SAS system options designate the search order that is used to locate SAS library resources. You can specify the statements anywhere within autoexec_usermods.sas.

```
FMTSEARCH=(MAMisc SecondLib ThirdLib) NOFMTERR;
```

FMTSEARCH specifies the search order for the format catalog or catalogs. The previous example specifies the search to begin in the MAMisc library, then to continue in a library called SecondLib, and finally to end in a library called ThirdLib.

NOFMTERR specifies to issue a note and continue processing if the specified variable format cannot be located.

For details about these SAS system options, see *SAS Language Reference: Dictionary*.

Chapter 2

Administrative Tasks

Administrative Objectives **17**

Tasks **17**

Setting Up **18**
Introduction 18
Use a Single Install Account 18
Use Fully Qualified Server Names 18
Specify Date Format for Exported Files 19
Support Double-Byte Character Sets 19
Set Session Timeout Value 19

Administrative Objectives

To enable users such as flow designers to achieve their objectives, administrators set up and maintain the SAS Decision Manager environment and the supporting SAS Intelligence Platform environment.

For information about integrating SAS products with other applications in your enterprise, see *SAS Integration Technologies* at **http://support.sas.com/documentation/onlinedoc/inttech**.

Tasks

After you complete the setup tasks by following the instructions that are provided in the SAS Deployment Wizard User's Guide, you might be required to perform additional tasks. These tasks might include setting options for environmental components that include third-party application servers, SAS servers, and scripts to support flow activities by users.

1. Control user access to business contexts, flows, and reports. For more information, see Chapter 3, "Users and Groups," on page 21.
2. Create information maps for flow activities. For more information, see Chapter 6, "SAS Information Maps for Flows," on page 45.You can also modify the provided information map for SAS Decision Manager. See the instructions for importing the information map in Chapter 8, "Implementing the Common Data Model," on page 99.

3. Implement the common data model. See the instructions for importing the information map in Chapter 8, "Implementing the Common Data Model," on page 99.
4. Update the counts metadata as needed. For more information, see "Understanding the Counts Metadata" on page 75.
5. Optimize the SAS environment by customizing the autoexec_usermods.sas file. For more information, see "Data Tier" on page 11.
6. Define a strategy for backing up and restoring information maps and campaigns. For more information, see Chapter 11, "Backing Up and Restoring Data," on page 123.
7. Troubleshoot problems by setting logging levels. For more information, see Chapter 15, "Troubleshooting," on page 203.

Setting Up

Introduction

See your on-site SAS support personnel for installation instructions.

To view the setup instructions for the common SAS Intelligence Platform, see the *SAS Intelligence Platform: Installation and Configuration Guide* at **`http://support.sas.com/documentation/onlinedoc/intellplatform/index.html`**.

Use a Single Install Account

Problems often occur when different accounts are used for various installations and subsequent installations. To eliminate potential problems, always use the same account (such as the SAS install account that was specified on the pre-installation checklist) for all installations. This practice is particularly important when applying service packs. If the user is running Windows Terminal Services, then you should operate in install mode.

Use Fully Qualified Server Names

Specify fully qualified server machine names in the SAS Deployment Wizard of SAS Management Console in order to avoid problems that can result in connecting to the workspace and stored process servers.

Table 2.1 *Example of a Fully Qualified Server Name*

Incorrect	joesmachine
Correct (Fully Qualified)	joesmachine.unx.mybusiness.com

Edit and correct any server machine names that are not fully qualified by modifying the server's properties in the **`Server Manager`** folder in SAS Management Console.

Specify Date Format for Exported Files

Instead of using the default date format for exported files, you can set a configuration option to make the date format consistent with the current locale.

To set the date format:

1. Open SAS Management Console.
2. Select the **Plug-ins** tab.
3. Expand the **`Application Management`** folder.
4. Expand the **Configuration Manager** plug-in.
5. Right-click the **Customer Intelligence Core 6.1** plug-in and select **Properties**.
6. In the Customer Intelligence Core 6.1 Properties dialog box, select the **Advanced** tab.
7. For the ExportDateFormat property, specify **`nldate.`**. Include the period (.) as part of the date format.

Support Double-Byte Character Sets

Text in many Asian languages requires the support of double-byte character sets. In order to display this text properly in SAS Decision Manager tables, an option must be set in the SAS configuration file.

Specify the following option in the sasv9.cfg file:

```
-VALIDVARNAME ANY
```

By default, the location for sasv9.cfg is **`C:\Program Files\SASHome \SASFoundation\9.4\sasv9.cfg`**.

Set Session Timeout Value

By default, the HTTP session timeout value for SAS Decision Manager is seven days.

You can modify the timeout setting by editing the config.xml and web.xml files in the ci.sas.shell.war file. The location of the sas.ci.shell.war file is determined by the SAS Web Application Server.

Edit the number of minutes in **`session-timeout`**.

The timeout values in config.xml and web.xml must be equivalent.

Chapter 3

Users and Groups

Overview of Security Administration **21**

Administering SAS Identities for Users **22**
Overview of SAS Identities 22
Create SAS Identities 22
The Decision Manager Services User ID 22

Administering Groups and Roles **23**
About Groups 23
About Roles 24
Viewing Roles in SAS Management Console 24
About Predefined Roles for SAS Decision Manager 25
About Predefined Roles for Business Rules 27
Capabilities and Comments 28
Configuring Access to Model Projects 28
Administering Group and Role Membership 29
Modifying Roles 29

Administering Permissions **30**
About Permissions 30
Using the Access Control Template (ACT) 30
Assigning User Permissions for Business Contexts 31
Assigning User Permissions for Custom Repositories 31

Overview of Security Administration

The security model for SAS Decision Manager and the SAS Intelligence Platform provides the following features:

- single sign-on from and across disparate systems
- secure access to data and metadata
- role-based access to application features
- confidential transmission and storage of data
- logging and auditing of security events
- access control reporting

Security administration consists of the following tasks:

- administering SAS identities for your users by adding account information to the SAS Metadata Server
- administering groups of users in order to simplify the management of roles and permissions
- administering roles, which provide users with access to specific application features
- administering users' permissions to access metadata repositories, folders, and objects

For more information about security administration, see *SAS Intelligence Platform: Security Administration Guide* at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Administering SAS Identities for Users

Overview of SAS Identities

For each user, you must create an individual SAS identity on the SAS Metadata Server. The SAS identity is a copy of the ID with which the user logs on to SAS applications. Based on this identity, the system can determine who can access which application, and can audit individual actions in the metadata layer. The SAS identity consists of a name and the user ID for the user's external account. This ID can be any type of account that is known to the metadata server's host such as an LDAP, Active Directory, host, or other type of account. When entering user IDs for Windows accounts, be sure to qualify the ID (for example, **WIN\myID** or **myID@mycompany.com**).

Create SAS Identities

To create SAS identities for your users, manually enter the information for each user through the User Manager plug-in in SAS Management Console. If you have a large number of users, then you can extract user and group information from one or more enterprise identity sources. You can then use SAS bulk-load macros to create the identity metadata from the extracted information.

For more information, see *SAS Intelligence Platform: Security Administration Guide* at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

The Decision Manager Services User ID

A Decision Manager Services User ID is created during the installation and configuration of SAS Decision Manager. The user ID has access to several system-level operations. The user ID is automatically assigned to the correct roles and capabilities. Do not change the permissions for this user ID or use it for any other purpose.

Administering Groups and Roles

About Groups

Groups enable you to give multiple users membership in a role or permissions to metadata, thus simplifying security administration. You can create as many groups as are needed in order to manage your installation.

The following groups are provided in your initial installation:

SAS Users
: This group includes everyone who can access the metadata server, either directly or through a trust relationship. If a user is able to log on to a client application and has an individual SAS identity, the user is assumed to be in this group. Because this group has implicit membership, you cannot explicitly add or remove users from this group.

Public
: This group includes everyone who can access the metadata server, either directly or through a trust relationship. If a user is able to log on to a client application but does not have an individual SAS identity, the user is assumed to be in the public group. Because this group has implicit membership, you cannot explicitly add or remove users from this group.

SAS Administrators
: This is a standard group for metadata administrators. In a standard configuration, members are granted broad access and administrative capabilities, but are not unrestricted.

Decision Manager Basic Flow Designer
: In your initial installation, this group is a member of the Decision Manager: Basic Flow Design role. You can add users to this group to give them access to basic flow design functionality.

Decision Manager Advanced Flow Designer
: In your initial installation, this group is a member of the Decision Manager: Advanced Flow Design role. You can add users to this group in order to give them access to advanced flow design functionality.

Decision Manager Administrator
: In your initial installation, this group is a member of the Decision Manager: Administration role. You can add users to this group in order to enable them to administer Decision Manager applications.

Decision Manager Common Administrator
: In your initial installation, this group is a member of the Decision Manager Common: Administrator role. You can add users to this group in order to enable them to manage administration resources such as business contexts, user sessions, and environment settings.

SAS System Services
: This group enables members to export files on the **Folders** tab of SAS Management Console.

About Roles

In SAS Decision Manager applications, certain actions are available only to users or groups that have a particular role. A role is a set of capabilities that correspond to particular application features such as menu items and plug-ins. Any user or group who is a member of a role has all of that role's capabilities.

Roles can contribute to one another. A role automatically includes all of the capabilities of a role that contributes to it.

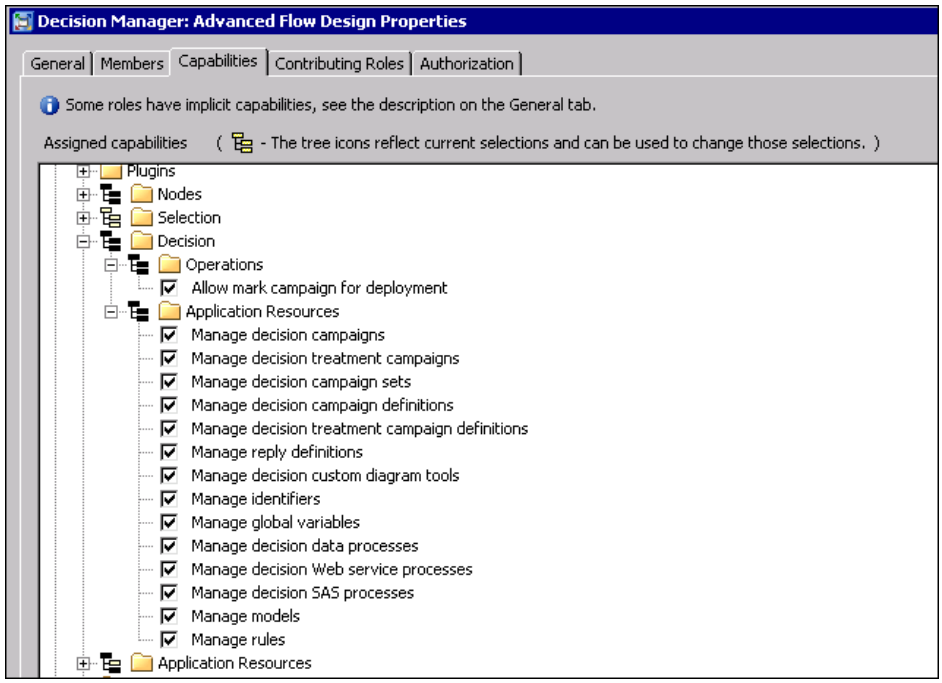
Roles differ from permissions. In general, roles do not affect access to metadata or data.

Viewing Roles in SAS Management Console

To view details about roles, open the User Manager plug-in in SAS Management Console, right-click the role, and select **Properties**. You can then view tabs that display the role's members, capabilities, and contributing roles.

For example, here is the **Capabilities** tab for the initial configuration of the Decision Manager: Advanced Flow Design role:

Display 3.1 *Capabilities Tab for the Decision Manager: Advanced Flow Design Role*



Note: The capabilities in the Selection folder have no effect on SAS Decision Manager.

The following icons provide information about the capabilities:

Means that none of the capabilities in this category have been specified for this role.

Means that some of the capabilities in this category have been specified for this role, either explicitly or through a contributing role.

Means that all of the capabilities in this category have been specified for this role, either explicitly or through a contributing role. To see details, click the plus sign (+).

- Shaded check boxes indicate capabilities that come from contributing roles.
- Some roles have implicit capabilities that are not specified on the **Capabilities** tab.

About Predefined Roles for SAS Decision Manager

Your installation includes several predefined roles for administrators and users of SAS Decision Manager. Depending on what software you have installed, you might have other predefined roles.

The following roles are provided for users and administrators:

Decision Manager: Usage
: This role provides the implicit capability to log on to SAS Decision Manager applications. This role contributes to the other SAS Decision Manager roles.

Decision Manager: Basic Flow Design
: In addition to the Decision Manager: Usage capabilities, users in this role have the implicit capability to access basic nodes to design new flows. This role contributes to the Decision Manager: Advanced Flow Design role.

Decision Manager: Advanced Flow Design
: In addition to the Decision Manager: Usage and Decision Manager: Basic Flow Design capabilities, users in this role can access advanced nodes to design new flows, write code in Process nodes, and access operations and application resources features.

Decision Manager: Administration
: In addition to the Decision Manager: Usage and Decision Manager: Administrator capabilities, users in this role can manage SAS Decision Manager administration resources, other SAS Management Console plug-ins, decision processes, and diagram tools.

Decision Manager Common: Administrator
: Users in this role can manage the categories in the Administration workspace: business contexts, user sessions, locks, and environment settings.

Note: The ability to access and update flow metadata is subject to permissions that are placed on that metadata. The SAS Decision Manager roles do not affect permissions.

The following table provides details about the capabilities in each of the predefined roles.

An asterisk (*) indicates that the capability is from a contributing role.

Table 3.1 *Roles for SAS Decision Manager*

Capability	Decision Manager: Usage Role	Decision Manager: Basic Flow Design Role	Decision Manager: Advanced Flow Design Role	Decision Manager: Administration Role	Decision Manager Common: Administrator Role

Log on to Decision Manager Applications	X	X*	X*	X*	X*
Manage flows		X	X		
Edit and delete comments		X	X		
Allow use of advanced nodes			X		
Allow writing code in Process node			X		
Manage calculated items			X		
Manage flow definitions			X		
Manage custom diagram tools			X		
Allow mark campaign for deployment			X		
Manage reply definitions			X		
Manage identifiers			X		
Manage global variables			X		
Manage data processes			X		
Manage web service processes			X		
Manage SAS processes			X		
Manage models			X		
Manage treatments			X		
Manage custom detail groups			X		

Manage response definitions			X		
Manage built-in diagram tools			X	X	
Manage events			X		
Manage channels				X	
Manage information map metadata				X	
Manage custom processes				X	
Manage business contexts				X*	X
Manage user sessions				X*	X
Manage locks				X*	X
Manage environment settings				X*	X

About Predefined Roles for Business Rules

Your installation includes several predefined roles for users of business rules.

Business Rules Manager: All Capabilities
: Users in this role can view and edit all business rules content, including vocabularies, entities, terms, lookup tables, rule sets, and rule flows.

Business Rules Manager: Designer
: Users in this role can create, edit, and delete rule sets and rule flows in the Designer workspace.

Business Rules Manager: Designer Read-Only
: Users in this role can view rule sets and rule flows in the Designer workspace.

Business Rules Manager: Definitions
: Users in this role can create, edit, and delete vocabularies, entities, terms, and lookup tables in the Definitions workspace.

Business Rules Manager: Definitions Read-Only
: Users in this role can view vocabularies, entities, terms, and lookup tables.

Capabilities and Comments

The ability to edit and delete comments is controlled by two capabilities in the **SAS Application Infrastructure** category.

Display 3.2 *Edit and Delete Comments Capabilities*

If these capabilities are not assigned to a role, the user cannot use the Comments page in SAS Decision Manager.

Configuring Access to Model Projects

The ability to view and interact with the Model Projects category is controlled by the **Access Model Projects category** capability that is located in **Model Manager JavaSvcs 12.3** ⇨ **Model Manager Plugin**. This capability must be assigned to the role that a user or group is a member of. Otherwise, the user cannot view the Model Projects category. Users or groups must also be members of a SAS Model Manager user group in order to use the functions in the Model Projects category. Such functions include importing models, creating or viewing reports, and publishing models.

Here are the configurations that are required to access and interact with the Model Projects category:

1. Add the **Decision Manager: Usage** role as a contributing role to each of the following SAS Model Manager roles:
   - Model Manager: Usage
   - Model Manager: Advanced Usage
   - Model Manager: Administration Usage
2. Enable the **Access Model Projects category** capability for the following roles:
   - Model Manager: Usage
   - Model Manager: Advanced Usage
   - Model Manager: Administration Usage

3. Add the user or group to one of the following SAS Model Manager groups:
   - Model Manager Users
   - Model Manager Advanced Users
   - Model Manager Administrator Users

Note: For more information about the SAS Model Manager roles and groups, see "Configuring Users, Groups, and Roles" in Chapter 4 of *SAS Model Manager: Administrator's Guide*.

Administering Group and Role Membership

To administer group and role membership, use the User Manager plug-in in SAS Management Console. In most cases, the best way to place a user in a role is to add the user to a group that belongs to the role. You can also add users directly to roles.

To place a user in one of the predefined roles, you can add the user to one of the predefined groups:

- To add a user to the Decision Manager: Basic Flow Design role, add the user to the Decision Manager Basic Flow Designer group. The user also receives the capabilities of the contributing role, Decision Manager: Usage.
- To add a user to the Decision Manager: Advanced Flow Design role, add the user to the Decision Manager Advanced Flow Designer group. The user also receives the capabilities of the contributing roles, Decision Manager: Basic Flow Design and Decision Manager: Usage.
- To add a user to the Decision Manager: Administration role, add the user to the Decision Manager Administrator group. The user also receives the capabilities of the contributing role, Decision Manager: Usage.

Note: There is no reason to add a user directly to the Decision Manager: Usage role. This role enables a user to log on, but does not provide any other useful functionality.

A user can be added to more than one group, and a user or group can be added to more than one role. For example, suppose a user needs to perform both administration tasks and advanced application tasks. You could take one of the following actions:

- Add the user to both the Decision Manager Administrator group and the Decision Manager Advanced Flow Designer group. This method might be appropriate if only one user needs this combination of capabilities.
- Create a new group called Decision Manager: Administrator and Advanced. You could then add the new group to both the Decision Manager: Administration role and the Decision Manager: Advanced Flow Design role. This method might be appropriate if multiple users need this combination of capabilities.

Modifying Roles

The User Manager plug-in in SAS Management Console enables you to modify roles by selecting or deselecting different capabilities.

CAUTION:

No automated method can revert a role to its original set of capabilities. Instead of adjusting the capabilities of a predefined role, consider creating a new role. This advice is especially important if major changes are needed.

If you modify a role, then follow these best practices:

- Do not rename the predefined roles. Renaming the predefined roles makes it difficult for SAS Technical Support to help you resolve problems.

- Back up the metadata server before modifying roles, and keep a record of the changes that you make.

When modifying a role, you can use only the capabilities that already appear in User Manager. You cannot create new capabilities.

For more information about roles and how to modify them, see *SAS Intelligence Platform: Security Administration Guide* at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Administering Permissions

About Permissions

SAS provides a metadata-based authorization layer that supplements the protections from the host environment and other systems. Protections are cumulative across authorization layers. In order to perform a task, a user must have sufficient access in all of the applicable layers.

You can set permissions at several levels:

- Repository-level controls provide the default access controls for objects that do not have other access controls.
- Resource-level controls manage access to a specific item such as an information map, a flow, a node, a business context, or a folder. The controls can be defined individually (by using explicit settings) or in patterns (by using access control templates).
- Fine-grained controls affect access to subsets of data within a resource. You can use these controls to specify who can access either particular rows within a table or members within a cube dimension.

The effect of a selected permission setting is influenced by any related settings that have higher precedence. For example, if a flow inherits a permission from its parent folder but also has an explicit denial, then the explicit setting determines the outcome. Similarly, if a group has been granted a permission, and a user who is a member of the group has an explicit denial, then the explicit setting determines the outcome.

Permissions are set by using the following three methods:

- The access control template (ACT), which provides a set of default permissions.
- Business contexts, which are groupings of flows. When you create a business context, you specify which users can view or edit the flows in that context.
- Custom repositories, which you can use to physically separate metadata for storage or security purposes.

Each of these methods is described in more detail below.

Using the Access Control Template (ACT)

The Decision Manager template provides a set of default permissions for SAS Decision Manager resources. This template is automatically applied to all SAS Decision Manager objects, including business contexts, flows, and nodes. In its initial configuration, this template denies ReadMetadata (RM) and WriteMetadata (WM) permission to the public group.

You might want to update the ACT in these situations:

- Give one or more users broad access to flow data for the purpose of application troubleshooting or administration. To do this, you can add the user to the ACT, either permanently or temporarily, and specify the appropriate permissions.
- Access flow metadata that was created by a user who is no longer in your organization. To do this, temporarily add an administrator to the ACT so that the administrator can transfer the flow permissions to a different user.

To update the ACT, open the Access Control Templates folder in the Authorization Manager plug-in in SAS Management Console..

Assigning User Permissions for Business Contexts

Flows are grouped into user-defined business contexts. Business contexts enable you to separate flow depending on which users should have access. When you define a new business context, you specify which users and groups have permission to view or edit the flows within that business context. Then, as flows are designed and created within the business context, the software applies access control entries (ACEs). ACEs give those users and groups the appropriate permission to access the flow data and metadata.

You must add users and groups to the appropriate roles before you can give them View or Edit permission for a business context.

Be sure that the sastrust account has been granted access to all business contexts. To give SAS Trusted User access to all existing and future business contexts, add SAS Trusted User to the access control template and grant ReadMetadata permission.

If the sastrust account is not granted access to each business context, then CI Common (SASCustIntelReporting) might fail to retrieve business contexts. After you have given the sastrust account access to business contexts, restart the CICommon (SASCustIntelReporting) application.

Assigning User Permissions for Custom Repositories

If you create a custom repository in order to physically separate metadata for storage or security purposes, then you can apply permissions to specify which users do and do not have access to the repository. In the **SAS Folders** tree in SAS Management Console, select the folder that represents the repository. Then open the Properties dialog box and update the **Authorization** tab.

For more information about permissions, see *SAS Intelligence Platform: Security Administration Guide* at `http://support.sas.com/documentation/onlinedoc/intellplatform/index.html` and the user's guide for your product.

Chapter 4

Folders in SAS Management Console

Overview of Folders 33
Directories for Business Rules Metadata and XML Files 34
Assets and Permissions 34
Copying and Pasting 34
Importing and Exporting SAS Packages 35
Deleting Business Rules XML Content 36
Best Practices for Exporting and Importing Objects 36
Importing and Pasting Calculated Items 37

Overview of Folders

The **Folders** tab of SAS Management Console displays the folders that contain metadata for SAS Decision Manager objects. You can import and export SAS Packages, and copy and paste objects between folders. SAS Decision Manager does not use the `My Folder` or `Shared Data` folders.

Folder permissions are applied to the contents of the folders. If a user has permission to edit a folder, then the user can edit the contents of the folder and delete an empty folder. If a user has permission to view the folder, then the user can view the contents of the folder. The user cannot add content to the folder, and cannot delete the folder. If a user has no access permissions to the folder, then the user cannot view or add contents to the folder, and cannot delete the folder. For more information about administrative roles and setting permissions, see the online Help for SAS Management Console.

You can create folders, but you cannot rename folders that are used by SAS Decision Manager.

When you create a folder, it inherits the permissions of the parent folder. You can change the inherited permissions.

Directories for Business Rules Metadata and XML Files

SAS Decision Manager creates two directories for business rules metadata: **`Products`** and **`/System`**. These directories are on the content server.

SAS Decision Manager creates a location for published XML files, **`Sasdav/Products`**. The BusinessRuleFlow metadata object does not delete the XML documents stored in this location in order to ensure that an audit trail is maintained. For more information, see "Deleting Business Rules XML Content" on page 36.

Assets and Permissions

The following types of assets are viewable on the **Folders** tab of SAS Management Console:

- System assets are stored in the **`/System/Applications/SAS Decision Manager`** folder.
- Flow assets are stored in the folder that is assigned to the business context for the flow.

Permissions for these two types of assets are configured differently.

Any user who has permission to write to system assets can also copy, paste, export, import, and delete the assets. No additional configuration is required.

To perform operations on flow assets, the user must be a member of the Decision Manager: Usage role. Even with full Write permission, the user cannot perform any of these operations without membership in the Decision Manager: Usage role. For information about roles, see "Administering Groups and Roles" on page 23.

If you have Write permission to flow assets, you can set permissions for an additional user or modify the permissions for an existing user. Within a business context folder, right-click an asset such as a flow definition, and select **Properties**. Modify access permissions on the **Authorization** tab.

Copying and Pasting

You can copy and paste folders and their contents. Before you copy and paste objects that depend on information maps, the application must be installed and configured, and information maps, business contexts, users, and groups must be defined. For objects that do not depend on information maps, the application must be installed and configured. Use copy and paste, rather than export and import, to move an item to a new business context on the same machine. The codes that are retained depend on the settings for the business context.

To paste an item into a folder, you must have the appropriate access permissions for the folder. An administrator can temporarily add a user to the access control template. The user can be removed after the required operation has been completed.

To copy an item, right-click the item and select **Copy**. To paste an item, right-click the destination and select **Paste** or **Paste Special**.

Note: You cannot rename objects.

When you select **Paste Special**, a copy of the item is created. If an item by the same name already exists in the target destination, the item is named `Copy of` *item name*.

To use **Paste Special** to include all of the dependencies of a copied object, select the object on the **Select Objects to Copy** page of the Paste Special wizard. Select **Select All** to include all of the objects that are listed on the **Dependencies** tab. If you select individual objects on the **Dependencies** tab, select the objects under the **Copied Objects** folder, and then select their dependencies on the **Dependencies** tab.

If you are copying assets such as business contexts or flow definitions, use **Paste Special** to retain all of the dependencies. For example, copies of flows and the flows that they link to are installed with the same folder structure as the original items. However, assets that are associated with inbound flows do not retain their dependencies. Tags that are no longer in the same environment as the copied treatment are not retained. If you use **Paste Special** to paste copied items directly into the **SAS Folders** folder, the source paths are not preserved. Instead, use **Export SAS Package** and **Import SAS Package** to preserve source paths in the **SAS Folders** folder.

You cannot copy and paste model processes.

Copied items retain the permissions of the original items. You can change the permissions of individual objects, such as flow definitions. Do not change permissions on processes; there might be several flows that rely on access to a process.

Note: If you copy a project in the `SAS Folders/Shared Data/Model Manager/Publish/Projects` folder, do not paste the copied project into the same folder. Models must be unique within a folder.

When you copy and paste flows and treatments, codes and control group names are retained according to the setting for the business context. Codes and control group names are retained only when the copied object is pasted within the same business context.

Importing and Exporting SAS Packages

You can use the SAS Package Wizard to export and import collections of objects and their dependencies into subfolders in the SAS Folders tree. To export the contents of a folder, right-click the folder and select **Export SAS Package**. You can then select the objects for inclusion in the export and save a SAS package file. To import a collection of objects and their dependencies, right-click the destination and select **Import SAS Package** and select a SAS package file. A user who has View permission can export a SAS package, but Write permission is required in order to import a SAS package. A user who has Read and Write permissions to both locations should both export and import the SAS package. The best practice is to export and import all objects simultaneously, to avoid losing the dependencies between objects. Definitions can be imported only into the correct subfolder in the Configuration folder.

To import an item into a folder, you must have the appropriate access permissions for the folder. An administrator can temporarily add a user to the access control template. The user can be removed after the required operation has been completed.

Make sure that users are logged off from SAS Decision Manager before you import or export SAS Packages.

Imported items retain their original names and overwrite items of the same name that are in the target location. You can choose to import all objects or only new objects. For more information about the SAS Package Wizard, see the online Help for SAS Management Console. The publishing information for imported flows is reset, except for the following codes in the common data model:

- CAMPAIGN_CD
- COMMUNICATION_CD
- MARKETING_CELL_CD
- PACKAGE_CD

When you import a definition, the only business contexts that remain associated with that definition are those business contexts for which you have Write permission.

When you import treatments along with a flow, an information icon () indicates that you should verify that the treatments are correct when you open the imported flow.

Comments are removed from the imported object.

Note: The system that you are exporting from and the system that you are importing to must have the same encoding.

Deleting Business Rules XML Content

Before you delete any XML content from **`Sasdav/Products`**, you should do the following:

1. Back up all versions of the content. The easiest way to back up the content is to use SAS Management Console to export the BusinessRuleFlow object that refers to the content.
2. Ensure that there are not any BusinessRuleFlow objects that refer to the content.

Best Practices for Exporting and Importing Objects

To configure the destination environment, do the following:

1. Create an information map in the destination environment that is identical to the information map in the source environment.
2. Create user names in the destination environment that are identical to the user names in the source environment. Assign the same roles and capabilities to the destination user names. User names are case-sensitive.
3. Create a repository in the destination environment that has the same name as the repository in the source environment.
4. Create a business context in the destination environment that is identical to the business context in the source environment. The destination business context must have the same name as the source business context.

5. If the objects in the source environment contain dynamic lists, create data libraries in the destination environment that contain the list values.
6. You must have Write permission in order to export and import objects. An import that fails because of permission issues results in a corrupt object. The visual indicator of a corrupt object is a red exclamation point ( ). This object can be deleted only by a user with Write permission.

To export objects from the source environment, do the following:

1. Select the SAS Management Console objects, such as flow definitions, that you want to export. Right-click the objects and select **Export SAS Package**. Select **Include dependent objects when retrieving initial collection of objects** in the Export SAS Package Wizard.
2. Select the SAS Decision Manager objects, such as flows, in the folder where they are stored. Right-click the objects and select **Export SAS Package**. Select **Include dependent objects when retrieving initial collection of objects** in the Export SAS Package Wizard.
3. Right-click the destination folder and select **Import SAS Package**. Select **Include access controls** in the Import SAS Package Wizard.

The exporting and importing process retains all codes, but regenerates surrogate keys.

Importing and Pasting Calculated Items

When you import or use **Paste Special** to paste calculated items in the **Folders** tab, keep in mind the following results:

- If an item with the same name already exists in the target folder, the item is named *item-name* (*n*), where *n* is an incremented number.
- If there are two items that are identical in all but name, and an item matches these items in the target folder, both pasted or imported items are mapped to the item that exists in the target folder.
- If there are two items that are identical in all but name, and two items with different names match these items in the target folder, both pasted or imported items are mapped to the first item that is first in alphabetical order.
- If there are two items that are identical in all but name, and two items with the same names match these items in the target folder, both pasted or imported items are mapped to the items that existed in the target folder.
- If there are two items that are identical in all but name, and one target item has the same name and one target item has a different name, the imported or copied item is mapped to the target item with the same name. The second item is mapped to the item that is first in alphabetical order.

Chapter 5

SAS Decision Manager and SAS Decision Services

Overview of SAS Decision Manager and SAS Decision Services **39**

Life Cycle of a Decision Flow **39**

Architecture **40**

Data Libraries, Event Variables, and Data Processes **41**
Data Libraries 41
Event Variables 41
Data Processes 41

Making Models Available to SAS Decision Manager **43**

Marking Items for Deployment **44**

Overview of SAS Decision Manager and SAS Decision Services

SAS Decision Services provides a rich set of activities for constructing decision flows that automate real-time decisions and actions. Activities perform work actions. Activities might execute SAS programs on a SAS server, store and access information from a relational database, send web service requests to external systems, and execute scoring models. For more information about SAS Decision Services, see *SAS Decision Services: Administrator's Guide* at **http://support.sas.com/documentation/solutions/ds/index.html**. You must supply the following user name and password to view this site:

```
User Name: sas
Password: CIadmin123
```

SAS Decision Manager bases decision flows on activities that were created in the Decision Services Manager plug-in for SAS Management Console.

Life Cycle of a Decision Flow

A decision flow is developed, tested, and promoted to a production system, where it is activated. Each stage is associated with a different repository. To promote a decision flow, the administrator exports the flow from the development repository and imports it

into a test or production repository. For information about exporting and importing, see "Best Practices for Exporting and Importing Objects" on page 36.

A decision flow is developed when the user connects nodes in a SAS Decision Manager flow. When the decision flow is ready, the user marks it for deployment in SAS Decision Manager. Marking a decision flow for deployment causes the flow to be stored in a development repository.

After a decision flow has been developed and stored in the development repository, the administrator can promote the flow to a test repository to verify the results.

Finally, the administrator promotes the flow to the production repository. The flow is generated on the SAS Decision Services design-time server and published to the common data model.

The promotion process is described in detail in *SAS Decision Services Administrator's Guide*. DS2 packages, which implement SAS activity methods, should be promoted to production less frequently than flows. Promoting DS2 packages every time that a campaign is promoted can break decision flows that used older versions of activities. When you consider whether to promote a DS2 package, keep in mind the potential impact this might have on existing flows in production that use the same DS2 package code.

Note: In SAS Decision Manager, activities within a flow cannot be executed concurrently.

Architecture

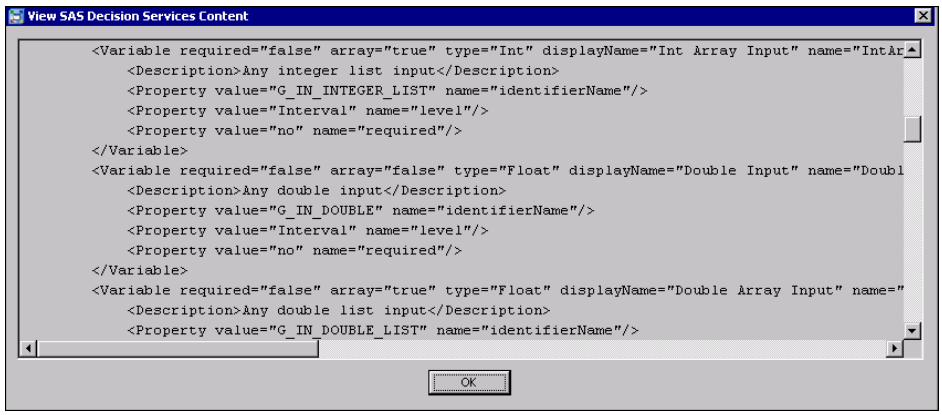
Events and global variable definitions are stored in a single location: a SAS Decision Services repository within the SAS Metadata Repository. Process and model definitions are stored in two locations: a Customer Intelligence folder and a SAS Decision Services repository within the SAS Metadata Repository. Identifiers are stored in the SAS Content Server.

Events and global variables are shared across business contexts. An event or global variable that is created in one business context is visible in all other business contexts. If you want to separate events and global variables between business contexts, then you must create a separate SAS Decision Services repository for each business context.

Data processes continue to read from the SAS library. The list of libraries is then filtered to show only those libraries that have a corresponding JDBC Library Resource of the same name in the SAS Decision Services repository.

When the SAS Decision Manager user creates an event or a global variable, the definition is stored in a SAS Decision Services design repository. You display the code by right-clicking the event or variable and selecting **View SAS Decision Services content**.

Display 5.1 *Event Definition*



Data Libraries, Event Variables, and Data Processes

Data Libraries

It is possible for a data process and an event input variable of type Data Grid to reference the same data library. There are two areas where SAS Decision Services uses the Data Library Manager in SAS Management Console. The list of data libraries that are available for selection by a data process is based on the JDBC library resources that are defined in the SAS Decision Services repository. Data libraries that are available for selection by a business context are listed by the Data Library Manager in SAS Management Console. The same library location can be referenced in both areas.

Event Variables

An event can have an input variable of type Data Grid. The location of input data grids is specified in the business context, as the **Decision test data grid input library** selection. The data grid for the event input variable is selected on the **Test** tab of a diagram in Test Mode.

An event can also have an output variable of type Data Grid. The location of output data grids is specified in the business context as the **Decision test data grid output library** selection.

Both library locations are selected from a list that is provided by the Data Library Manager in SAS Management Console.

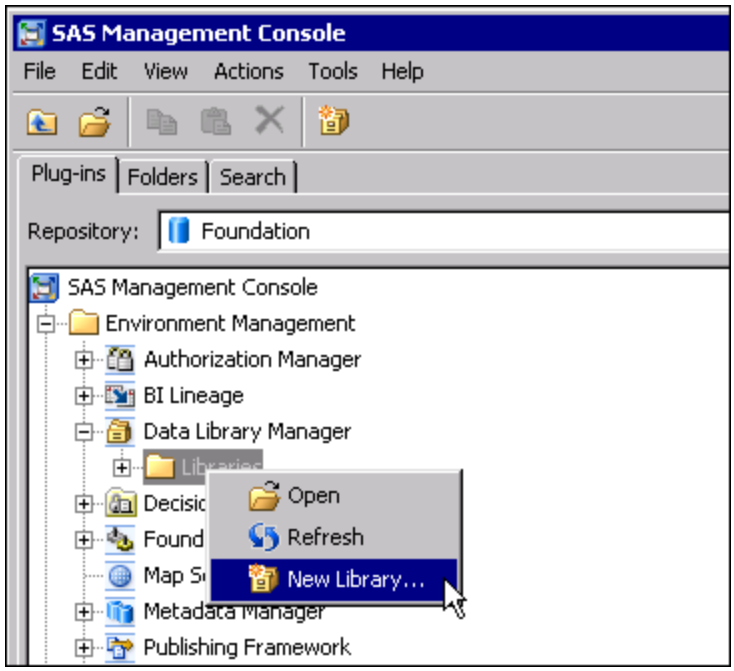
Data Processes

Data processes use a combination of data libraries and JDBC libraries. Before a data process can be used in SAS Decision Manager, you must create a data library, a JDBC Connection system resource, and a Library system resource. Both the data library and the Library system resource must have the same name.

To prepare to use a data process in SAS Decision Manager:

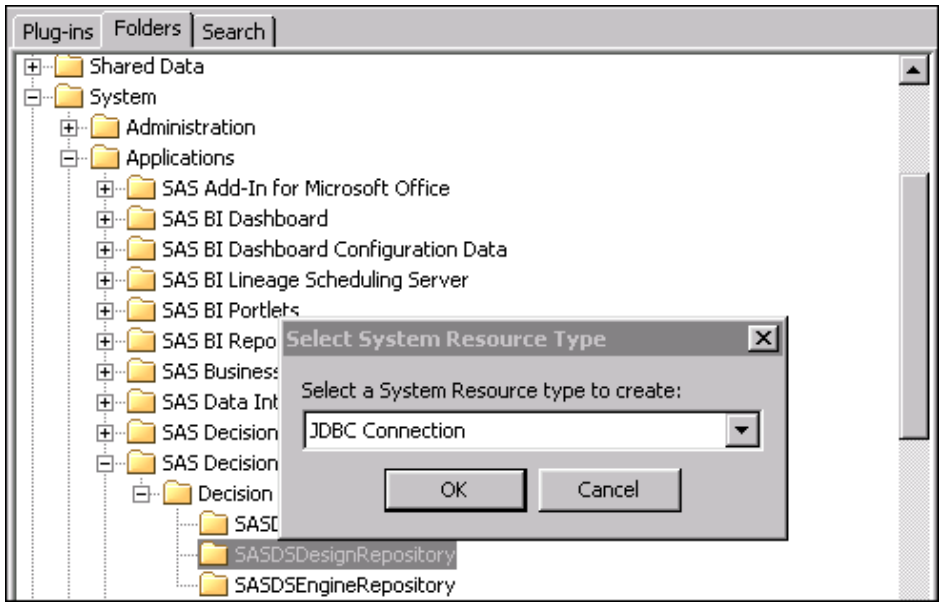
1. On the **Plug-ins** tab of SAS Management Console, create a library in the Data Library Manager.

Display 5.2 *Data Library*



2. A JDBC Connection system resource definition is required for each database vendor. On the **Folders** tab, create a JDBC Connection system resource. The location is `/System/Applications/SAS Decision Services/Decision Services 6.2/SASDSDesignRepository`.

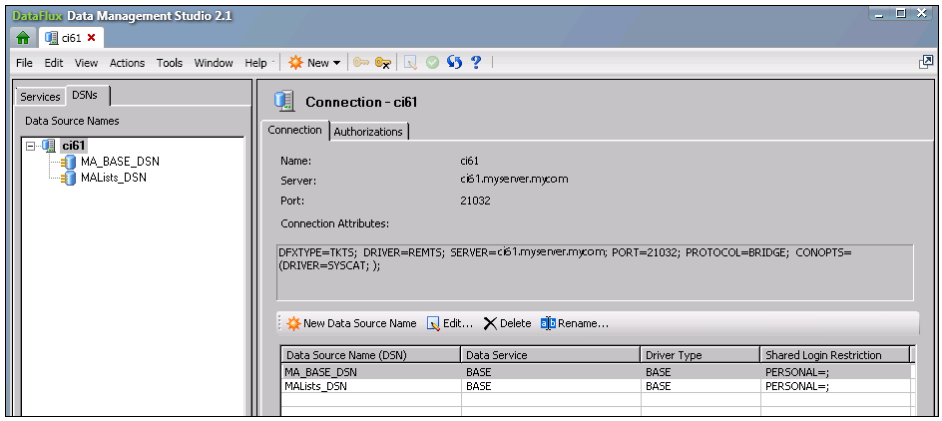
Display 5.3 *Create System Resource*



3. In the same location, create a Library system resource and link it to the JDBC Connection system resource that you just created. The Library system resource must have the same name as the data library that you created in the first step.

4. If you are using SAS data sets, create a SAS Federation Server definition in Data Management Studio. A separate Data Source Name (DSN) must be defined on the Federation Server for each database libref.

Display 5.4 *Federation Server Definition*

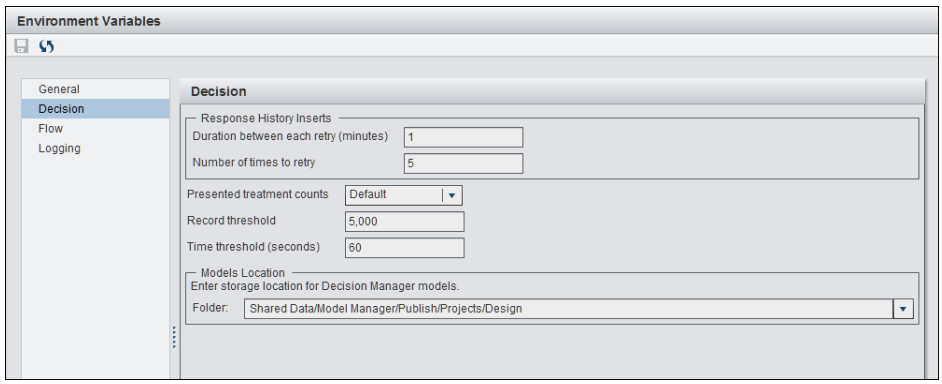


Making Models Available to SAS Decision Manager

If you have licensed SAS Model Manager, you can define a model process in SAS Decision Manager. You select the model process in the Score node.

To make a model available to SAS Decision Manager, use SAS Model Manager to publish a model to a metadata folder. Here is an example: **`SAS Folders/Shared Data/Model Manager/Publish/Projects/Design`**. The folder should be the same folder that is specified in the **Models Location** section of the Decision page in the Environment Variables category in SAS Decision Manager.

Display 5.5 *Folder for Models*



You can then select a select a Model Manager project for use in a model process.

You can publish a manually written DATA step model through SAS Model Manager. Because SAS Model Manager translates the model into DS2 using the DSTRANS procedure, there are some limitations on the SAS DATA step code that can be published. SAS code that is used in the Score node cannot contain list colon syntax or hard tabs.

Refer to the SAS Technical Support for more information on running model score code in portable environments.

Marking Items for Deployment

To mark a flow for deployment, select the **Folders** tab of SAS Management Console. In the folder that contains the SAS Decision Manager items, right-click an item and select **Mark for Deployment**.

Chapter 6

SAS Information Maps for Flows

Identifying Data for Flow Activities **45**

The Role of the Information Map in SAS Decision Manager **46**
What Is an Information Map? 46
What Does an Information Map Contain? 46

Creating a SAS Information Map for SAS Decision Manager **47**
Overview of Steps 47
Organizing Your Folders as Categories 49
Tips: Building Your Information Map 50
Best Practice: Use a Phased Approach to Create Your Information Map 52

About Custom Properties **53**
Overview 53
Required Custom Properties 53
Specify the MAMeta Library as a Custom Property 53
Add Custom Properties to an Information Map: Steps 54
Making Custom Properties Available 56
Custom Properties (Map Level) 61
Custom Properties (Folder Level) 62
Custom Properties (Data Item Level) 63
All Custom Properties for Information Maps 64
Definitions of the Levels of Data Items 73

Understanding the Counts Metadata **75**
What Is the Counts Metadata? 75
Descriptions of the Counts Metadata Tables 75
Generating the Counts Metadata 76

Identifying Data for Flow Activities

As an administrator, you find the data that will be available to SAS Decision Managero users. You then identify all of the data by creating a single information map in SAS Information Map Studio. SAS Decision Manager users analyze this data, prioritize targets based on this data, and export this data.

You can include as many physical tables as you want. Note, however, that performance is optimized when fewer tables are used.

Although some businesses need only one information map, some large enterprises create additional information maps for various business contexts. New business contexts can be created and associated with an existing information map in SAS Decision Manager.

The Role of the Information Map in SAS Decision Manager

What Is an Information Map?

An information map consists of metadata that describes a data warehouse in business terms. A SAS Decision Manager information map describes the data sources that enable SAS Decision Manager users to easily create flows. For example, instead of viewing a large collection of tables and columns, the business user views a simple list of business terms.

All information maps are stored in the SAS Metadata Repository in a folder that is designated as **`ReportStudio/Maps`**. On the **Folders** tab of SAS Management Console, SAS Decision Manager information maps are typically stored in an **`Information Maps`** subfolder. You can save an information map in any folder on the SAS server tier. For more information, see *SAS Decision Manager User's Guide*.

What Does an Information Map Contain?

An information map specifies the basic elements of a SAS Decision Manager data environment.

Display 6.1 *List of Elements*

- **Folders** (an example is label 1) reorganize information into categories that are relevant to your business. Folders in the information map appear as folders in selection dialogs in SAS Decision Manager.
- **Data items** (label 2) provide a business representation of a physical data element. Data items in the information map typically serve as selection criteria in SAS Decision Manager.
- **Calculated items** (label 3) can perform predefined calculations. Calculated data items that are created in the information map are globally available.
- **Filters** (label 4) specify business rules for dividing data into subsets.

All data items in an information map are associated with one or more subjects. This association is usually specified at the **Folder** level of the information map. The values for these subjects can be selected in SAS Decision Manager.

Note: When an information map is created, the locale for that information map is set to the current locale and cannot be changed. When you run a query that uses a data item from that information map, your results conform to the locale that was originally specified for the information map.

Creating a SAS Information Map for SAS Decision Manager

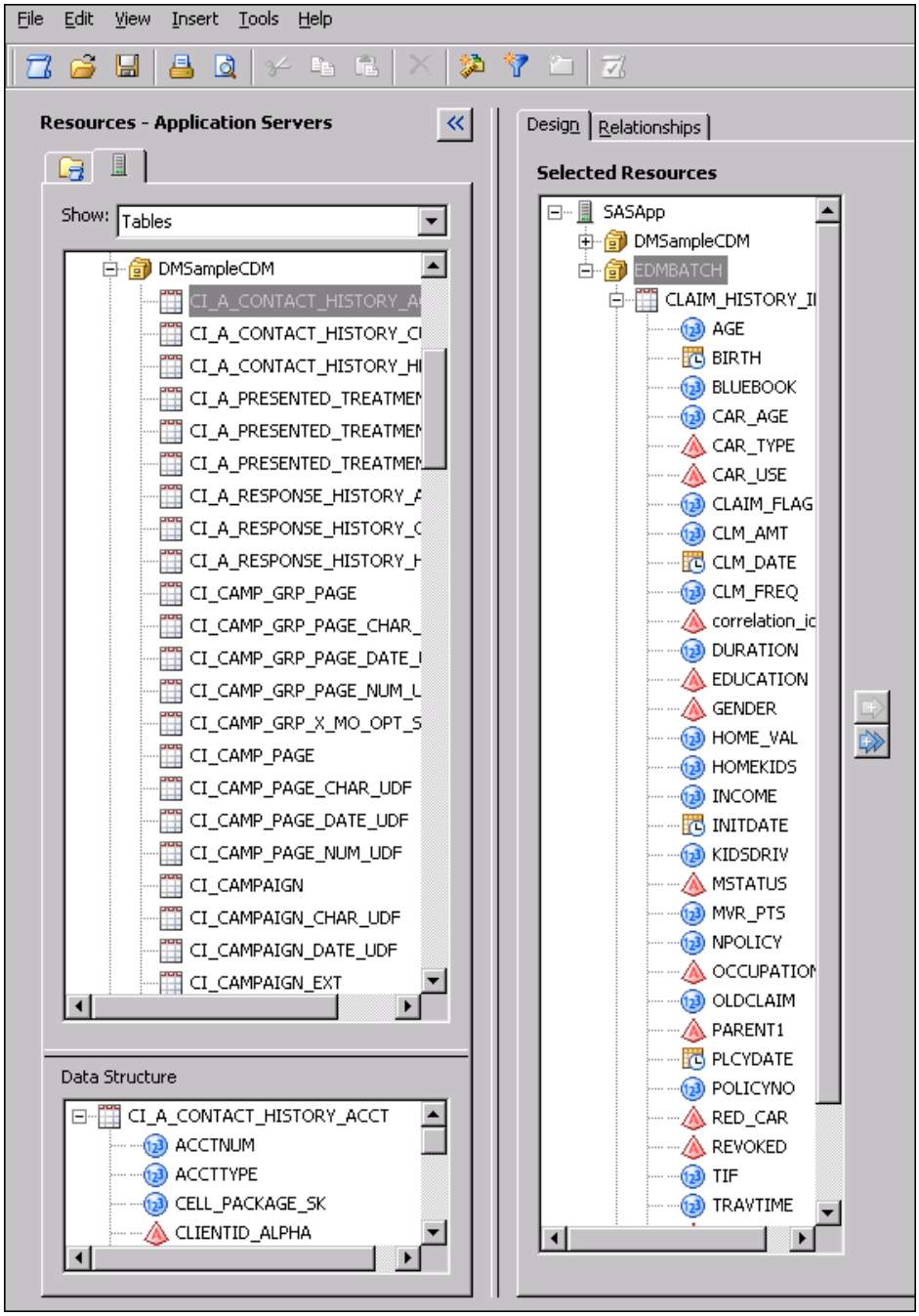
Overview of Steps

After you register the locations of the physical data, you create an information map in order to make data available to SAS Decision Manager.

To create an information map, follow these steps.

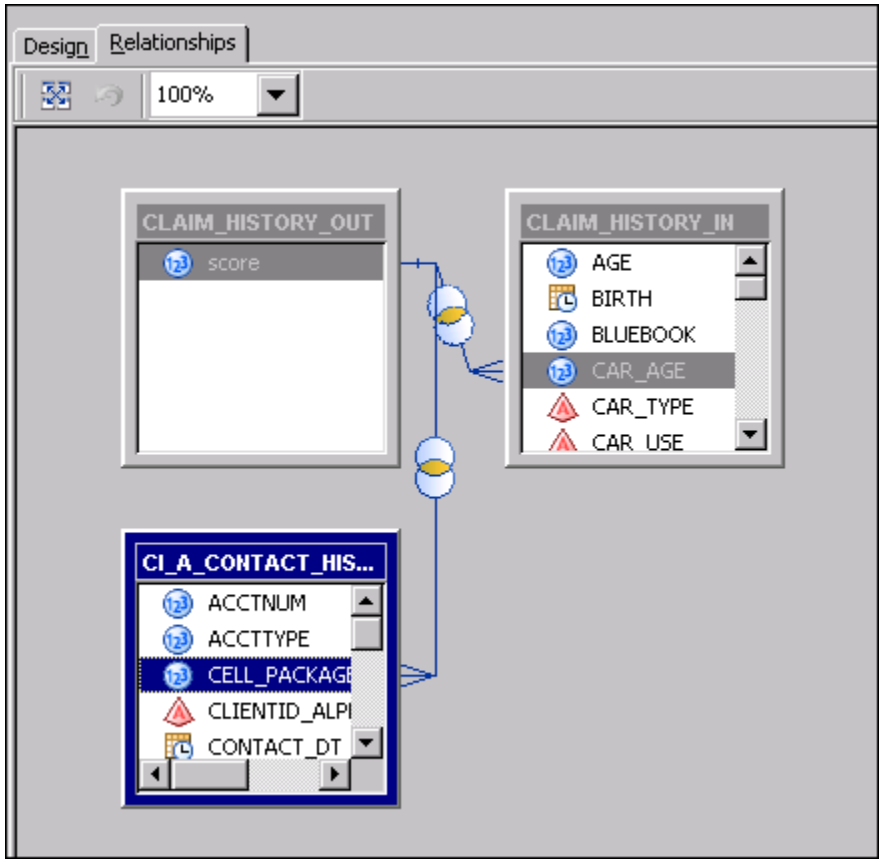
1. In SAS Information Map Studio, specify the data sources for the data items that are available to SAS Decision Manager users. For more information, see "Organizing Your Folders as Categories" on page 49.

Display 6.2 *Selected Resources*



2. Specify the table relationships on the **Relationships** tab to enable the retrieval of information from multiple tables.

Display 6.3 *Relationships*



For more information, see the online Help for SAS Information Map Studio and "Tips: Building Your Information Map" on page 50.

Note: Table relationships are established by specifying join keys and join types.

3. Specify any custom properties (extended attributes) in the information map. For more information, see "About Custom Properties" on page 53.
4. Save the information map. You can store it in any folder. For more information, see "Organizing Your Folders as Categories" on page 49.

Organizing Your Folders as Categories

In order to present the data in a business context, organize pertinent table columns as folders. Not all of the columns of a table are required to be added to the information map. For example, the attributes that are used to support the data loading processes can be hidden from users by not adding those attributes to the information map.

A folder can contain the data from one or more tables. Not all of the columns of a table have to be contained in a single folder; organize the data in order to help the end users perform their tasks.

Tips: Building Your Information Map

Make the Names of Subjects Plural

In some diagram nodes in SAS Decision Manager, the variable for <*subject*> is read by the application and displayed to the user within the labels for the check boxes. Here is an example: **Select** <*subject*> **that do not meet these criteria**.

When the value for <*subject*> is automatically substituted, the resulting text can produce grammatical errors. To avoid any automatically generated phrases such as **Select Customer that do not meet these criteria**, enter the plural of <*subject*>. For example, type **Customers** instead of **Customer**.

Adding a Relationship in a Cluttered Work Area

After many new data sources have been added, it becomes difficult to use the "select and drag" method to create a new relationship. When the **Relationship** tab becomes cluttered, use the following method to define a new relationship:

1. After you add a data source, right-click in a blank area of the entity box and select **Insert Relationship**.
2. Use the Insert Relationship dialog box to create the relationship.

Editing Entities That Are Joined with Multiple Fields

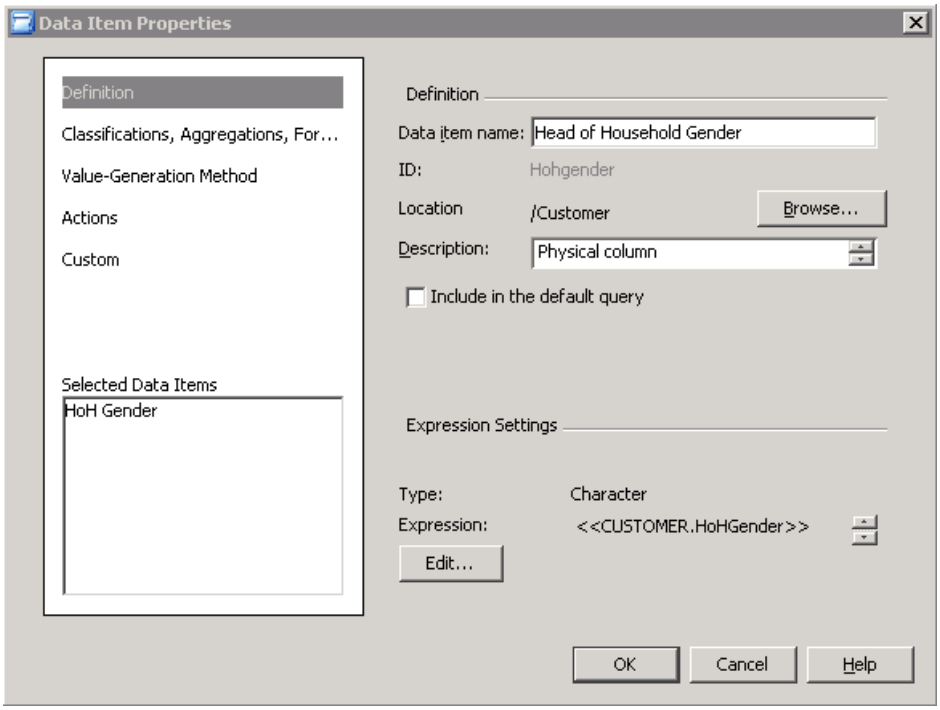
On the **Relationships** tab where entities are joined together using multiple fields, instead of dragging all of the fields at the same time, drag one field, and then edit the relationship in order to add the remaining fields.

Assign Meaningful Business Names

Replace the physical data names with more meaningful business names.

In the figure below, the column HoHGender is specified to be displayed as Head of Household Gender in SAS Decision Manager.

Display 6.4 *Meaningful Business Names*



Use the Cancel Button to Exit More Quickly

On the **Presentation** tab, if you make no changes, select **Cancel** to resume control more quickly than selecting **OK**.

Find Physical Data More Quickly

On the **Presentation** tab, when you are working with the physical data, close all of the open tables. Closing all of the open tables enables you to find the data to be mapped more quickly, especially when there are data sources that have many attributes.

Improve Performance When Generating the Counts Metadata

Eliminate the generation of the counts metadata for those data items where frequency counts are not applicable, thereby reducing the time it takes to generate metadata in general.

Example

In this example, it is not important how many customers have the last name of "Smith." Save time by setting **`Metadata=None`** for the **Customer Last Name** data item so that the counts metadata is not generated for this data item.

Alternatively, if you set **`Metadata=Values`**, then you enable users to view a list of last names; but if they try to select by **Customer Last Name**, then the counts are queried from the database instead of from the previously calculated counts metadata.

Best Practice: Use a Phased Approach to Create Your Information Map

Phase I: One Subject

A phased approach to creating an information map and generating its associated metadata can simplify troubleshooting.

Create a simple information map with one of each of the following data items:

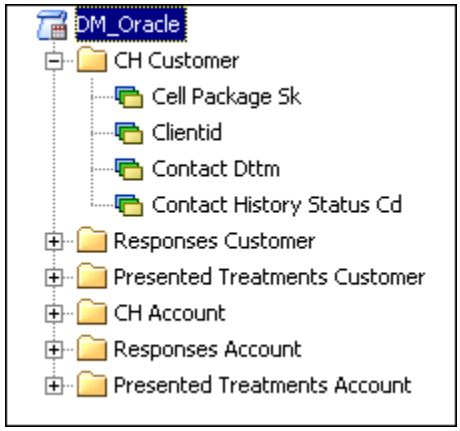
- table
- folder
- subject
- key
- other data item

Generate the counts metadata and validate it to ensure that the physical data and server infrastructure are working as expected.

Phase II: Multiple Subjects

Continue to build the simple information map by adding tables, subjects, subject keys, and folders. Add only one other data item to each folder. Generate metadata and validate it to ensure that table and subject relationships are properly specified.

Display 6.5 *Multiple Subjects*



Phase III: Fully Populated Map

Continue to build the information map incrementally by adding data items by folder, function, type, or any other logical progression until the information map is fully populated. Generate the counts metadata and validate it frequently to ensure that the physical data and server infrastructure are working as expected.

About Custom Properties

Overview

During the installation of SAS Information Map Studio, if you indicate that SAS Decision Manager will also be installed, then a file named MATemplate.txt is automatically loaded. MATemplate.txt contains a prescribed list of custom properties that are needed to create information maps for SAS Decision Manager. For more information, see "Example: Adding Custom Subjects to MATemplate.txt" on page 60.

Required Custom Properties

An information map that has two subjects requires the following custom properties:

- At the information map level:
  - Subject_ID_<*Name1*>
  - Subject_ID_<*Name2*>
  - From_Subject_ID_<*Name1*>_To_Subject_ID_<*Name2*>
  - From_Subject_ID_<*Name2*>_To_Subject_ID_<*Name1*>
  - MAMeta. For more information, see "Specify the MAMeta Library as a Custom Property" on page 53.
  - MetadataTable_Prefix_Subject_ID_<*Name1*>
  - MetadataTable_Prefix_Subject_ID_<*Name2*>
  - Subject_Default
- At the data item level:
  - Level
  - UseInSubjectID

Specify the MAMeta Library as a Custom Property

The MAMeta library must be specified as a custom property in any information map for SAS Decision Manager. For best performance, data should be stored on the SAS server rather than the data server.

Note: Do not define the MAMeta library in autoexec_usermods.sas. Otherwise, whenever the first stored process that is started by SAS Decision Manager finishes running, it closes all libraries that are defined in the autoexec_usermods.sas file, including the MAMeta library. This action prevents SAS Decision Manager from accessing your marketing metadata.

To specify the MAMeta library as a custom property, follow these steps:

1. Using SAS Information Map Studio, locate and select the appropriate information map.
2. In the Information Map Contents pane, right-click the information map name, and select **Properties**.

3. Select the **Custom** tab and locate the MAMeta label in the table of custom properties.

   If the **Custom** tab is not displayed in SAS Information Map Studio, then see "Making Custom Properties Available" on page 56.

4. Specify the appropriate path for the MAMeta information map libref in the Value column.

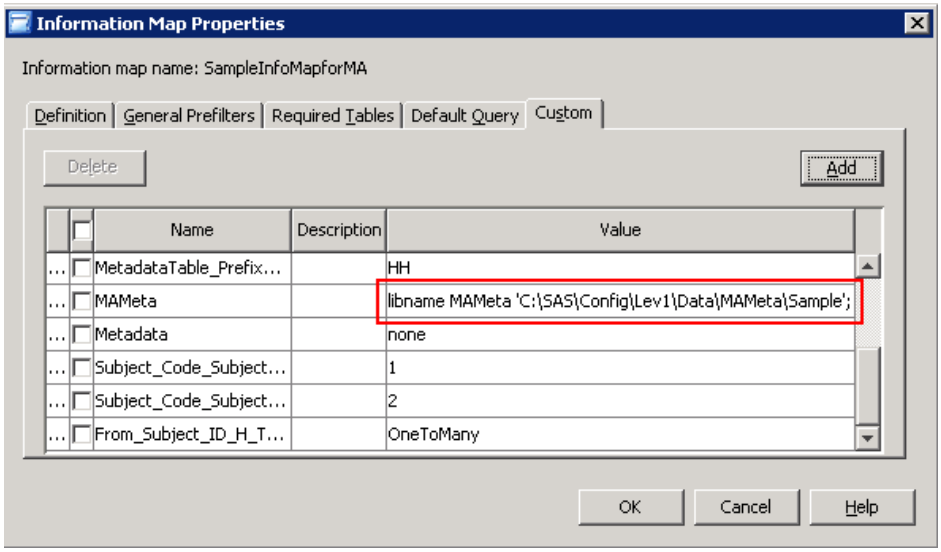
   If the libref points to a UNIX server, then follow these steps to ensure that the correct path is specified:

   a. In a DOS window, navigate to the directory on the UNIX server where the MAMeta information map is located.

   b. Type `pwd`. The full current path is returned.

   c. Copy the path and paste it into the Value column.

Display 6.6 *MAMeta Pathname*



5. Click **OK** to save and close the **Custom** tab.

6. Verify that SAS Decision Manager users (including the users of the saswbadm user account) have both READ and WRITE access to the physical location of the MAMeta library.

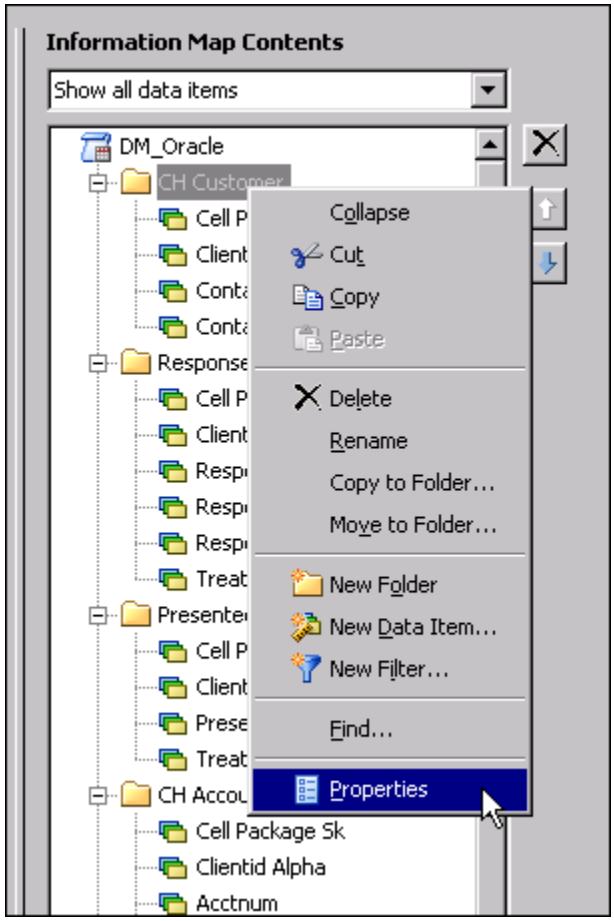
Add Custom Properties to an Information Map: Steps

To add custom properties to a SAS Information Map at the map level, folder level, or data item level, follow these steps:

1. Open the information map in SAS Information Map Studio.

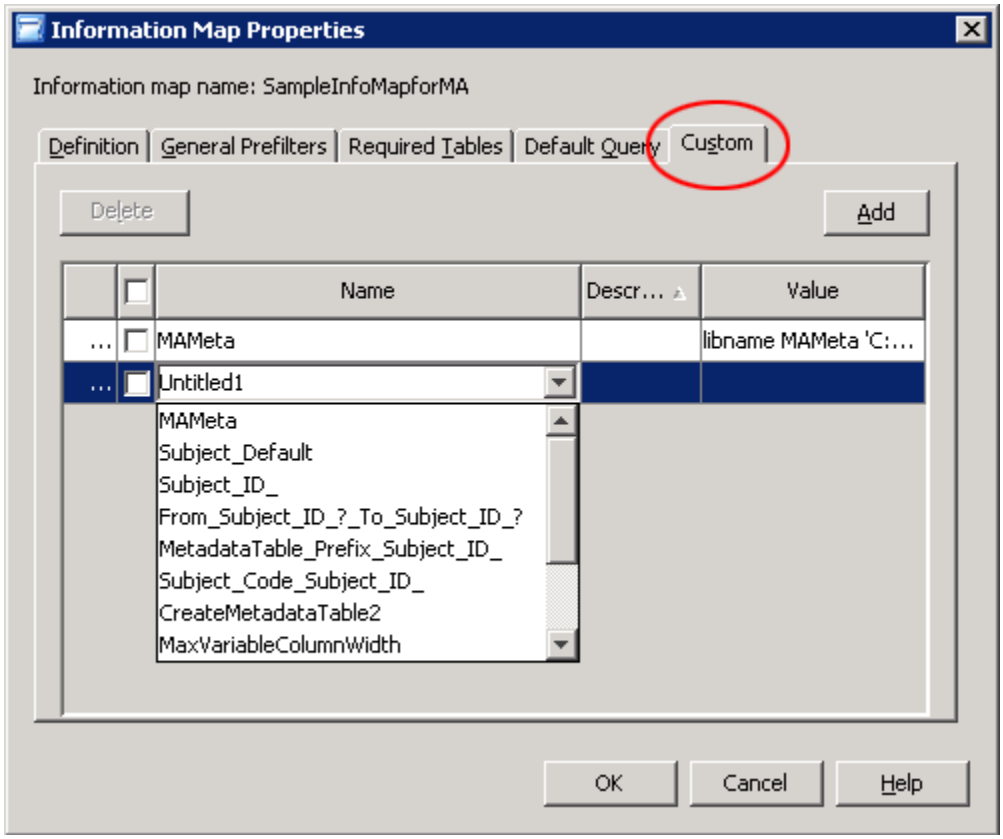
2. In the Information Map Contents pane, right-click the information map, folder, or data item, and select **Properties**.

Display 6.7 *Folder Properties*



3. Select the **Custom** tab.
4. If an Untitled1 row is not already displayed, then click **Add** to create a new row. Otherwise, click in the right side of the Untitled1 cell to view the drop-down list of custom properties that are available, and select a custom property.

Display 6.8 *Custom Properties*



5. Complete any partial data items that you select. To enter text in a cell, double-click in the cell. For more information, see "Custom Properties (Map Level)" on page 61.
6. After you enter the name, description, and value for a new custom property, click within a header cell to remove the highlighting from the new row. If you click **OK** while any part of a row is highlighted, then the changes in that row are not saved.
7. Add the custom properties that are required for the map level. For more information, see "Custom Properties (Map Level)" on page 61.
8. To exit and save the new custom properties, click a heading cell or an existing row to remove the highlighting from any attribute, and then click **OK**.

Making Custom Properties Available

Overview of the MATemplate.txt File

The MATemplate.txt file enables you to specify the valid values for properties for an information map, its folders, and its data items. These are all elements that might be used in an information map that is created for SAS Decision Manager users.

When you add or edit a custom property in SAS Information Map Studio, SAS Information Map Studio displays a drop-down list of items that is read from MATemplate.txt.

MATemplate.txt is typically activated during the installation process. If you do not see the **Custom** tab in any Information Map Properties dialog box, then you still need to activate the MATemplate.txt file by making custom properties available.

Contents of the Default MATemplate.txt File

These are the contents of the default MATemplate.txt file.

```
<ExtendedAttributesTemplate>
  <InformationMap EditablePicklist="True">
    <Item Key="MAMeta" EditablePicklist="True">
      <Option Value="Libname MAMeta "/>
    </Item>
    <Item Key="Subject_Default" EditablePicklist="True">
      <Option Value="Subject_ID_"/>
    </Item>
    <Item Key="Subject_ID_"/>
    <Item Key="From_Subject_ID_?_To_Subject_ID_?" EditablePicklist="False">
      <Option Value="OneToOne"/>
      <Option Value="OneToMany"/>
      <Option Value="ManyToOne"/>
      <Option Value="ManyToMany"/>
    </Item>
    <Item Key="MetadataTable_Prefix_Subject_ID_"/>
    <Item Key="Subject_Code_Subject_ID_"/>
    <OptionalItem Key="CreateMetadataTable2" DefaultValue="False" EditablePicklist="False">
      <Option Value="True"/>
      <Option Value="False"/>
    </OptionalItem>
    <OptionalItem Key="MaxVariableColumnWidth" DefaultValue="40"/>
    <OptionalItem Key="MaxValueColumnWidth" DefaultValue="200"/>
    <OptionalItem Key="Verbose" DefaultValue="False" EditablePicklist="False">
      <Option Value="True"/>
      <Option Value="False"/>
    </OptionalItem>
    <OptionalItem Key="DiscreteSql" DefaultValue="SP" EditablePicklist="False">
      <Option Value="IQ"/>
      <Option Value="SP"/>
    </OptionalItem>
    <OptionalItem Key="Metadata" DefaultValue="Counts" EditablePicklist="False">
      <Option Value="Counts"/>
      <Option Value="Values"/>
      <Option Value="None"/>
    </OptionalItem>
    <OptionalItem key="MaxItemsForInQuery" DefaultValue="1000"/>
    <OptionalItem key="FilteredTableExport" DefaultValue="False" EditablePicklist="False">
      <Option Value="True"/>
      <Option Value="False"/>
    </OptionalItem>
    <OptionalItem key="ForbidProblemUploads" DefaultValue="False" EditablePicklist="False">
      <Option Value="True"/>
      <Option Value="False"/>
    </OptionalItem>
    <OptionalItem key="SPRefinementExport" DefaultValue="False" EditablePicklist="False">
      <Option Value="True"/>
      <Option Value="False"/>
    </OptionalItem>
    <OptionalItem key="ExtraBlankQuery" DefaultValue="False" EditablePicklist="False">
      <Option Value="True"/>
      <Option Value="False"/>
    </OptionalItem>
```

```
</InformationMap>
<Folders EditablePicklist="True">
  <Item Key="Subject_ID_">
    <Option Value="Subject_ID_"/>
  </Item>
  <OptionalItem Key="Histogram_NumBins" DefaultValue="64"/>
  <OptionalItem Key="Histogram_DisplayNumBins" DefaultValue="16"/>
  <OptionalItem Key="Metadata" DefaultValue="Counts" EditablePicklist="False">
    <Option Value="Counts"/>
    <Option Value="Values"/>
    <Option Value="None"/>
  </OptionalItem>
</Folders>
<DataItems EditablePicklist="True">
  <Item Key="Level" EditablePicklist="False">
    <Option Value="Id"/>
    <Option Value="Nominal"/>
    <Option Value="Interval"/>
    <Option Value="Binary"/>
    <Option Value="Ordinal"/>
    <Option Value="Unary"/>
  </Item>
  <OptionalItem Key="UseInReports" DefaultValue="False" EditablePicklist="False">
    <Option Value="True"/>
    <Option Value="False"/>
  </OptionalItem>
  <OptionalItem Key="UseInCluster" DefaultValue="False" EditablePicklist="False">
    <Option Value="True"/>
    <Option Value="False"/>
  </OptionalItem>
  <OptionalItem Key="UseInSubjectId" DefaultValue="Subject_ID_" EditablePicklist="True">
    <Option Value="Subject_ID_"/>
  </OptionalItem>
  <OptionalItem Key="UseInSubjectIdTop" DefaultValue="Subject_ID_" EditablePicklist="True">
    <Option Value="Subject_ID_"/>
  </OptionalItem>
  <OptionalItem Key="IsFilterItem" DefaultValue="False" EditablePicklist="False">
    <Option Value="True"/>
    <Option Value="False"/>
  </OptionalItem>
  <OptionalItem Key="Precision"/>
  <OptionalItem Key="VarUsedWithFilter"/>
  <OptionalItem Key="Histogram_NumBins" DefaultValue="16"/>
  <OptionalItem Key="Histogram_DisplayNumBins" DefaultValue="16"/>
  <OptionalItem Key="Histogram_Start"/>
  <OptionalItem Key="Histogram_Increment"/>
  <OptionalItem Key="Contact_History" DefaultValue="Subject_ID_" EditablePicklist="True">
    <Option Value="Subject_ID_"/>
  </OptionalItem>
  <OptionalItem Key="Responses" DefaultValue="Subject_ID_" EditablePicklist="True">
    <Option Value="Subject_ID_"/>
  </OptionalItem>
  <OptionalItem Key="Presented_Treatments" DefaultValue="Subject_ID_" EditablePicklist="True">
    <Option Value="Subject_ID_"/>
  </OptionalItem>
  <OptionalItem Key="NoMetadata" DefaultValue="False" EditablePicklist="False">
```

```
      <Option Value="True"/>
      <Option Value="False"/>
    </OptionalItem>
    <OptionalItem Key="AllowBlankValue" DefaultValue="False" EditablePicklist="False">
      <Option Value="True"/>
      <Option Value="False"/>
    </OptionalItem>
    <OptionalItem Key="OutputColumnName"/>
    <OptionalItem Key="Metadata" DefaultValue="Counts" EditablePicklist="False">
      <Option Value="Counts"/>
      <Option Value="Values"/>
      <Option Value="None"/>
    </OptionalItem>
    <OptionalItem Key="UseInOptimization" DefaultValue="Subject_ID_" EditablePicklist="True">
      <Option Value="Subject_ID_"/>
    </OptionalItem>
    <OptionalItem Key="Identifier"/>
    <OptionalItem Key="ExportIgnoresRefinement" DefaultValue="False" EditablePicklist="False">
      <Option Value="True"/>
      <Option Value="False"/>
    </OptionalItem>
  </DataItems>
  <FilterItems EditablePicklist="True">
  </FilterItems>
  <Relationships EditablePicklist="True">
  </Relationships>
</ExtendedAttributesTemplate>
```

Make Custom Properties Available: Steps

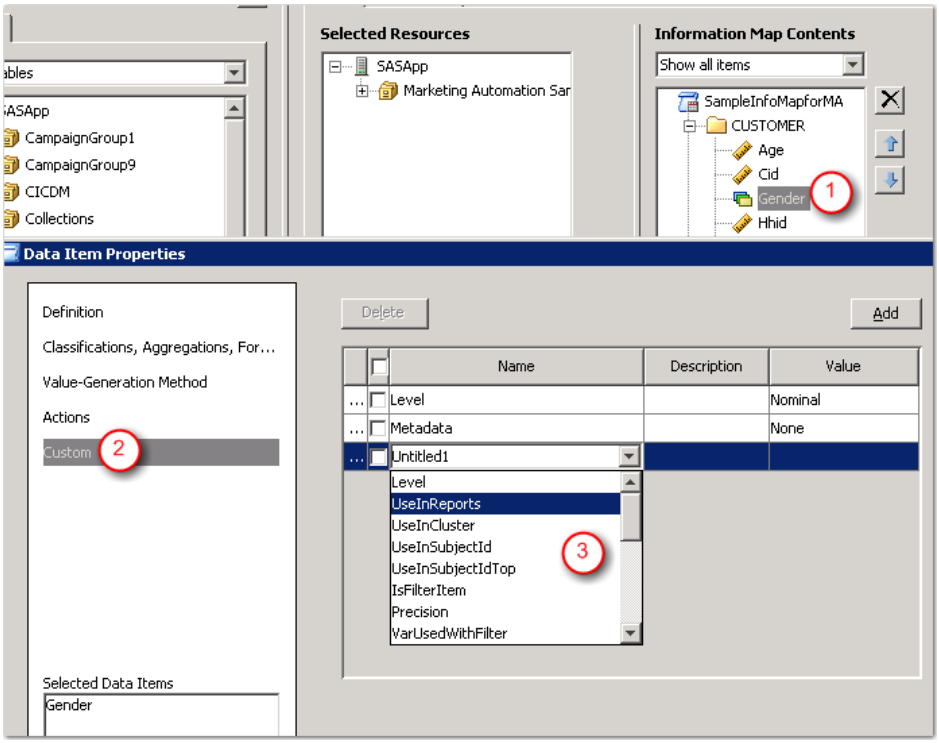
1. In SAS Information Map Studio, select **Tools** ⇨ **Options** from the menu bar.
2. On the **Advanced** tab of the Options dialog box, select the **Custom properties at start-up** check box.
3. Specify the MATemplate.txt file by browsing for a location such as this on the SAS server tier:**`C:\Program Files\SASHome\SASDecisionManager\6.2`**.
4. Click **OK** to close the Options dialog box.
5. Restart SAS Information Map Studio to apply the new setting.

Verify That Custom Properties Are Available

To verify that the custom properties for editing an information map are available in SAS Information Map Studio, follow these steps:

1. In SAS Information Map Studio, open an information map for SAS Decision Manager.
2. On the **Design** tab, in the Information Map Contents pane, double-click an information map, folder, or specific data item. **Gender** (label 1) is the selected data item.
3. In the Information Map Properties dialog box that appears, if the **Custom** option (label 2) is displayed, then MATemplate.txt is successfully activated.
4. Click in a row beneath the Name column. The drop-down list (label 3) contains items that SAS Information Map Studio reads from MATemplate.txt.

Display 6.9 *Verify Custom Properties*



Customize Subject_ID in MATemplate.txt

The following example demonstrates how you modify MATemplate.txt in order to specify the list of subjects on the **Custom** tab in SAS Information Map Studio.

Any modification that you make to MATemplate.txt will be overwritten by an upgrade of SAS Decision Manager software. Therefore, if you want to retain your modified MATemplate.txt file, then save a copy of it in a safe location in order to replace the new default template file.

Note: If you modify a custom property in MATemplate.txt, the change does not affect any previously created information maps. The change affects only the new information maps that are created for SAS Decision Manager. If you specify a new value for an existing custom property in MATemplate.txt, then the new value is displayed in the list of possible values for the custom property.

Example: Adding Custom Subjects to MATemplate.txt

This example produces a list of values that the user can select.

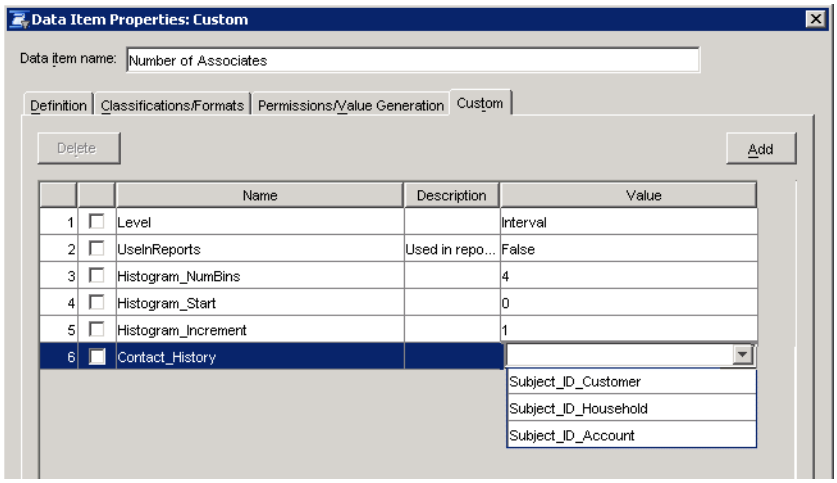
Search for instances of the following code in MATemplate.txt.

```
<Option Value="Subject_ID_">
```

Replace each instance of the code in MATemplate.txt with the following code:

```
<Option Value="Subject_ID_Customer"/>
<Option Value="Subject_ID_Household"/>
<Option Value="Subject_ID_Account"/>
```

Display 6.10 *Custom Subjects*



Verify that the list of subjects is correctly displayed. For more information, see "Verify That Custom Properties Are Available" on page 59.

Custom Properties (Map Level)

In the example values in the following table, the two subjects are Customer and Household.

The extended attributes for a SAS Decision Manager information map that are related to subject are in the following table.

Table 6.1 *Custom Properties (Map Level) That Are Related to Subject*

Label	Description	Example Value	Required?	Default Value
FilteredTableExport	affects the refinement of export behavior. If FilteredTableExport is set to True, then when a subject is specified to be exported, at least one line for every subject is exported.	True	no	-
From_Subject_ID_ *subjectname1*_ To_Subject_ID_ *subjectname2*	is defined for each relationship between the valid subjects on the source tables. An entry is necessary for each direction of the relationship. These are valid relationships: • OneToOne • OneToMany • ManyToOne • ManyToMany* * not recommended because if the cells or lists of customers are created, then any duplicates are deleted.	From_Subject_ID_Customer_To_Subject_ID_Household=ManyToOne From_Subject_ID_Household_To_Subject_ID_Customer=OneToMany	yes	-

MetadataTable_Prefix_ Subject_ID_*subjectname* (for example, MetadataTable_Prefix_ Subject_ID_Customer)	is the prefix name for the SAS data sets that contain the generated data (the counts metadata). This is also the prefix name of data items for which NoMetadata extended attribute is set to False. Only one extended attribute per subject should exist.	Customer	yes	-
Subject_Code_Subject_ID__ *subjectname*	associates a two-digit code with a subject. Only one extended attribute per subject should exist. This code is encoded as part of the ResponseTrackingCode that is a mandatory column to be passed to SAS Customer Intelligence Common Services. This code enables SAS Customer Intelligence Common Services to determine which subject table to write the contact responses and the presented treatment rows to.	Subject_Code_ Subject_ID__ Customer = 01	yes	-
Subject_Default	is the default subject extended attribute to be used throughout the information map for any data items where an associated subject extended attribute has not been explicitly set.	Subject_ID_Customer	yes	-
Subject_ID_*subjectname* (for example, Subject_ID_Customer)	is established for each subject as a unique extended attribute. Only one such extended attribute exists per subject. This *subjectname* is used in many places for other extended attribute values. The value appears in the **Subject** drop-down list in the Select Data Item dialog box in SAS Decision Manager. The actual physical data field name that makes up this subject is defined at the data item level.	Customer	yes	-

Custom Properties (Folder Level)

Table 6.2 *Custom Properties (Folder Level) That Are Related to Subject*

Label	Description	Example Value	Required?	Default Value

Subject_ID_*x* (for example, Subject_ID_1	is a shortcut that defines the subject attribute of this data item as this value for all of the data items in this folder, unless otherwise specified. Each Subject_ID_*x* label defines a valid subject for selection rules when a user selects data items in this folder. The value must correspond to a subject extended attribute name that was defined at the information map level. More than one of these extended attributes can be defined in a folder, by incrementing the number. In a a diagram node, the user chooses a category (folder) and a subject. Then the user selects the data items in the corresponding folder that has the chosen subject as a value of one of these Subject_ID_*x* extended attributes at either the folder level or data item level. If Subject_ID_1 is not set for either a given data item or parent folder, then the subject default value that is set at the Information Map level is assumed. For example, if for the data item Account Balance), Subject_ID_1 = Subject_ID_Customer and Subject_ID_2 = Subject_ID_Account, then Account Balance can be selected only if the user's chosen subject is either Customer or Account.	Subject_ID_ Customer	no	-

Custom Properties (Data Item Level)

Table 6.3 *Custom Properties (Data Item Level) That Are Related to Subject*

Label	Description	Example Value	Required?	Default Value
Contact_History	identifies the table in which contact history records are added for the specified subject. The contact history property is used by SAS Decision Manager in order to direct new records to the contact history table. The underlying physical column name or names of the data items that have UseInSubjectID equal to this value are the physical column name or names of the subject on Contact History. This attribute should be set once per subject.	Subject_ID_ Customer	yes for one data item derived from each contact history table	-
ExportIgnoresRefinement	affects the refinement of export behavior. If ExportIgnoresRefinement is set to True for a data item, then the data item ignores any refinement that has been added. This extended attribute is useful for aggregated items.	True	No	-

Presented_Treatments	specifies that, for the specified subject, the presented treatment is stored in the table that contains this data item. This attribute should be set once per subject.	Subject_ID_ Customer	yes for one data item	-
Responses	identifies the table in which the responses are stored for the specified subject. The Responses property is used by SAS Decision Manager in order to fetch the appropriate response records from the response table for the response node. Also, the underlying physical column names of the data items that have UseInSubjectID equal to this value are the physical column name or names of the subject on responses. This attribute should be set once per subject.	Subject_ID_ Customer	yes for one data item derived from each response table	-
UseInSubjectID	is set only for the data items that are derived from primary keys of a primary subject table. The value that is specified here is the Subject_ID_*x* extended attribute value that is assigned for this subject table at the information map level. For example, if this extended attribute is on data item Customer ID whose underlying physical column name is CUSTTABLE.CUST_ID, then this extended attribute defines the physical column as CUSTTABLE.CUST_ID for subject Subject_ID_Customer. If two physical fields make up a single subject, then a UseInSubjectID extended attribute exists for both of the data items that are associated with those fields. Only one (or one set) of these attributes for each subject that is represented in the information map should exist. It does not matter which folder the data item with this attribute is in; it is the physical subject column or columns that are derived for all data items in the information map that is associated with that subject. The field name and type are used to generate the holding table when the selections are made at this subject level.	Subject_ID_ Customer	yes for data items from the primary key of the subject tables	-

All Custom Properties for Information Maps

The extended attributes in an information map describe settings for the following levels:

Map
: The custom properties (for information maps) typically describe the metadata and define the subjects in an information map. Subject properties specify either a default subject or valid subjects for selection rules, and specify the relationship between

valid subjects of selection rules. Metadata properties specify the location for the metadata tables, the names for tables, and the metadata column widths.

Folder
: The custom properties for folders specify valid subjects for selection rules, and specify the details about histograms.

Data Item
: The custom properties for data items define the supported data types, the creation of primary keys and multi-keys, and the location of contact history and response tables for each subject.

Each of the following tables includes all of the extended attributes for information maps at the map level, folder level, and data item level, respectively.

Table 6.4 *Map Level: All Custom Properties*

Label	Description	Example Value	Required ?	Default Value
CIBusinessContext	This attribute is needed for Visual Selection integration with Web Report Studio. It is added to the Information Map to be used in the Web Report Studio Report. The user of the information map must be assigned the Web Report Studio Advanced Users role. The value is the name of the business context where the report export is available. Report exports are the SAS data sets or criteria that are created by Web Report Studio. Only one of these extended attributes can be set per information map.	BC_Name	no	-
Count(*)	Use this attribute to specify whether to use a select count(*) to get a count for the query. The default value uses select count(*) to obtain a row count for the query instead of using a JDBC method to obtain a row count. **Caution**This attribute is designed to be used only for performance testing, and should not be used in the production environment.	True False	no	True
CreateMetadataTable2	If this attribute is set to True, then the suffix 2 is added to each generated table name. The tables are created in the library that is defined by the MAMeta library reference.	True False	no	False

DiscreteSql	is used to specify the SQL method to use in order to generate discrete (NUM and CHAR) metadata.	DiscreteSql = [SP] **SP: use a stored process** DiscreteSql = [IQ] **IQ: use Iquery (JDBC)**	no	SP
FilteredTableExport	affects the refinement of export behavior. If FilteredTableExport is set to True, then when a subject is specified to be exported, at least one line for every subject is exported.	True	no	-
ForbidProblemUploads	identifies problematic queries and does not pass the problematic query to the database. SAS performs the query locally. If your tables are large, this setting can cause additional performance issues. This custom property is especially useful when using Netezza	True	no	False
From_Subject_ID_ *subjectname1_* To_Subject_ID_ *subjectname2*	is defined for each relationship between the valid subjects on the source tables. An entry is necessary for each direction of the relationship. These are valid relationships: • OneToOne • OneToMany • ManyToOne • ManyToMany* * not recommended for this reason: if the cells or lists of customers are created, then any duplicates are deleted.	From_Subject_ID _Customer_ To_Subject_ID_ Household= ManyToOne From_Subject_ID _Household_ To_Subject_ID_C ustomer= OneToMany	yes	-
IDListSeparator	overwrites the default separator (<>) for the IDList parameter that is passed into the stored process in the case where the data item ID also contains this pattern. This value is used by the metacount stored process.	—- ++++ ****	no	<>
MAMeta	specifies a libref for the location of the metadata tables.		yes	-

MaxItemsForInQuery	converts ITEM IN(...) queries with > N items: SELECT ITEM IN(...< N) UNION SELECT ITEM IN(...< N) Use this attribute to override the specification that is imposed by a database (such as Oracle) that limits the number of items in a list.	1000 (for Oracle)	no	0
MaxProcSummary	specifies discrete (NUM and CHAR) variable metadata. MAxProcSummary specifies the maximum number of variables to process within one execution of PROC SUMMARY. By reducing the number of variables, you can reduce memory requirements.	20	no	50
MaxValueColumnWidth	a numeric attribute that defines the width of the metadata columns Value and FormattedValue in each metadata table.	100	no	200
MaxVariableColumnWidth	a numeric attribute that defines the width of the metadata column Variable in each metadata table.	100	no	40
Metadata	Metadata = [Values | Counts | None] Values: distinct values only. Counts: default (distinct values and counts). None: equivalent to NoMetadata=True. The Values argument is for discrete variables only. The Metadata extended attribute can be propagated from upper level to lower level. For example, if you specify Metadata = x at the map level, then all of the folders inherit this setting. An exception is when a folder also has its own Metadata = y attribute that overrides the x setting.	Values Counts None	no	Counts
MetadataTable_Prefix_ Subject_ID_*subjectname* (for example, MetadataTable_Prefix_Subject_ID_Customer)	specifies the prefix name for the SAS data sets that contain the generated data and for the data items that do not have the NoMetadata extended attribute set to False. Only one extended attribute per subject should exist.	Customer	yes	-

Subject_Code_Subject_ID__ *subjectname*	associates a two-digit code with a subject. Only one extended attribute per subject should exist. For example, Subject_Code_Subject_ID_Customer=01 Subject_Code_Subject_ID_Account=02	01	yes	-
Subject_Default	is the default subject extended attribute to be used throughout the information map for any data items where an associated subject extended attribute has not been explicitly set.	Subject_ID_Customer	yes	-
Subject_ID_*subjectname* (for example, Subject_ID_Customer)	is a unique extended attribute that exists for each subject. The *subjectname* is used in many places for other extended attribute values. The value appears in the **Subject** drop-down list in the **Select Data Item** dialog box in SAS Decision Manager. The actual physical data field name or names that make up this subject are defined at the data item level.	Customer	yes	-
Verbose	If this extended attribute is set to True, then the extended attributes for the map, folder, and data items are output to the log files of the middle tier.	True False	no	False

Table 6.5 *Folder Level: All Custom Properties*

Label	Description	Example Value	Required?	Default Value
Histogram_Display-NumBins	defines the number of intervals used in the histogram in the Select Data Item dialog box in SAS Decision Manager. Unless specified at the data item level, this value applies to all Interval level data items within the folder.	8	no	If Histogram_NumBins is 64, then the default value is 16.

Histogram_NumBins	defines the number of bins to be used by the histogram stored process in order to support the histogram display of interval data items. Interval data items are specified in the Select Data Item dialog box in SAS Decision Manager. The value for this attribute is required to be an even power of 2. For example, 16 is an acceptable value because 16 is equal to 2 to the 4th power. This value applies to all of the interval level data items within the folder, unless the value is specified at the data item level.	32	no	16
Metadata	Metadata = [Values | Counts | None] Values: distinct values (discrete variables only). Counts: default (distinct values and counts). None: equivalent to NoMetadata=True. This extended attribute can be propagated from upper level to lower level. For example, if you specify Metadata = Counts at the map level, then all of the folders inherit this setting. An exception is when an attribute such as Values for a folder has been specified. In this example, the folder attribute (Values) overrides the setting of the map level (Counts).	Values Counts None	no	Counts
Subject_ID_*x* (for example, Subject_ID_1)	is a shortcut to define that for all of the data items in this folder, unless otherwise specified, the data item's subject attribute is this value. Each Subject_ID_*x* defines a valid subject for selection rules when a user selects data items in this folder. The value must correspond to a subject extended attribute name that is defined at the information map level. More than one of these extended attributes can be defined in a folder, by incrementing the number. In a diagram node, the user chooses a category (folder) and a subject. Then the user selects the corresponding folder that has the chosen subject as a value of one of these Subject_ID_*x* extended attributes at either the folder level or data item level. If Subject_ID_1 is not set for either a given data item or the parent folder, then the Subject Default that is set at the Information Map level is assumed. For example, if Subject_ID_1=Subject_ID_Customer and Subject_ID_2=Subject_ID_Account are specified for the data item Account Balance, then Account Balance can be selected only if the user's chosen subject is either Customer or Account.	Subject_ID_ Customer	no	-

Table 6.6 *Data Item Level: All Custom Properties*

Label	Description	Example Value	Required?	Default Value
AllowBlankValue	specifies that a blank value is allowed to be used as a distinct value when this attribute is set to true. Set this attribute to True for a Teradata database (or any database that is not configured to allow blank values).	True False	no	False
Contact_History	identifies the table in which contact history records are added for the specified subject. The contact history property is used by SAS Decision Manager in order to direct new records to the contact history table. The underlying physical column name or names of the data items that have UseInSubjectID equal to this value are the physical column name or names of the subject on Contact History. This attribute should be set once per subject.	Subject_ID _Customer	yes for 1 data item derived from each contact history table	-
ExportIgnoresRefinement	affects the refinement of export behavior. If ExportIgnoresRefinement is set to True for a data item, then the data item ignores any refinement that has been added. This extended attribute is useful for aggregated items.	True	No	-
Histogram_Display-NumBins	defines the number of intervals that are used in the histogram in the Select Data Item dialog box in SAS Decision Manager. Unless specified at the data item level, this value applies to all interval level data items within the folder.	8	no	If Histogram _NumBins is 64, then the default value is 16.
Histogram_Increment	specifies the increment value for the histogram stored process.	1	no	-
Histogram_NumBins	defines the number of bins to be used by the histogram stored process in order to support the histogram display of interval data items. Interval data items are specified in the Select Data Item dialog box in SAS Decision Manager. The value for this attribute is required to be an even power of 2. For example, 16 is an acceptable value because 16 is equal to 2 to the 4th power. This value applies to all of the interval level data items within the folder, unless the value is specified at the data item level.	32	no	16
Histogram_Start	specifies the start value for the histogram stored process.	1	no	-
Identifier	specifies a string that is entered by a user.	-	no	-

IsFilterItem	specifies whether the corresponding data item is a filter item that is applied in the WHERE clause.	True False	no	False
Level	specifies the level of value to be assigned to the data item. You can assign the following levels to a data item: • ID • Unary • Binary • Nominal • Interval • Ordinal Data items that are set to Level=ID are not used in SAS Decision Manager. See Table 6.7 on page 74 for the definition of each valid level of data item. See Display 6.11 on page 75 for valid combinations of Level with data type.	Nominal	yes	-
Metadata	Metadata = [Values | Counts | None] Values: distinct values only. Counts: default (distinct values and counts). None: equivalent to NoMetadata=True. Values is for discrete variables only. The Metadata extended attribute can be propagated from upper level to lower level. For example, if you specify Metadata = x at the map level, then all of the folders inherit this setting. An exception is when a folder also has its own Metadata = y attribute that overrides the x setting.	Values Counts None	no	Counts
NoMetadata	specifies whether a discrete list of values should be generated in the metadata. If a data item whose level is nominal has many distinct values (for example, 1,000) to be generated for the metadata tables, then performance degrades. If you use NoMetadata, then the user must specify values manually in order to use the data item for selections. Values are case sensitive and must be enclosed in quotation marks for data items that are specified by character table columns. To prevent poor performance for nominal data items that have too many unique values, specify NoMetadata=True.	True	no	False
OutputColumnName	specifies the column name that replaces the physical column name that is used in the internally generated SQL query when this data item is selected for processing.	Any valid column name.	no	-

Precision	specifies the precision for the interval variable	100	no	1
Presented_Treatments	specifies that the table containing this data item is where the presented treatment is stored for the specified subject. This attribute should be set once per Subject_ID.	Subject_ID _Customer	yes for 1 data item from each presented treatment table	-
Responses	identifies the table in which the responses are stored for the specified subject. The Responses property is used by SAS Decision Manager to fetch the appropriate response records from the response table. The underlying physical column name or names of the data items that have UseInSubjectID equal to this value are the physical column name or names of the subject on responses. This attribute should be set once per subject.	Subject_ID _Customer	yes for one data item derived from each response table	-
UnivariateMethod	generates metadata for a data item in a univariate table. The default is the default SAS behavior. These are the valid values: InDatabase uses the MEAN function to calculate the values Summary uses PROC SUMMARY to calculate the values AVGMinMax calculates the average of the minimum and maximum values	InDatabase	no	default SAS behavior
UseInCluster	specifies whether this data item is used in the Cluster node. These are the valid values: • True • False Nominal and ordinal data items are clustered separately from interval data items. If UseInCluster is set to True in any nominal or ordinal data items, then at least three nominal or ordinal data items with the UseInCluster attribute set to True must exist. At least three interval data items with UseInCluster set to True must exist if any interval data items have UseInCluster set to True. To improve performance, do not cluster any nominal or ordinal data items with many values. Limit the data items that you cluster to those data items that are the most meaningful to cluster.	True	no	False

UseInOptimization	specifies whether the data item is used in an optimization of the subject ID. The valid value is the subject ID.	Subject_ID _Customer	no	-
UseInReports	This custom property is not referenced in Release 6.1.			
UseInSubject- IDTop	is the same as UseInSubjectID, except that it also specifies the order of this data item to be placed ahead of the other multi-part keys.	Subject_ID _ Arrangement	no	-
UseInSubjectID	is set only for the data items that are derived from primary keys of a primary subject table. The value that is specified here is the Subject_ID_*x* extended attribute value that is assigned for this subject table at the information map level. For example, if this extended attribute is on data item Customer ID whose underlying physical column name is CUSTTABLE.CUST_ID, then this extended attribute defines that for subject Subject_ID_Customer, the physical column is CUSTTABLE.CUST_ID. If two physical fields make up a single subject, then a UseInSubjectID extended attribute would exist for both of the data items that are associated with those fields. Only one (or one set) of these attributes for each subject that is represented in the information map should exist. It does not matter which folder the data item with this attribute is in; it is the physical subject column or columns that are derived for all data items in the information map that is associated with that subject. The field name and type are used to generate the holding table when the selections are made at this subject level.	Subject_ID _ Customer	yes for data items from the primary key of the subject tables	-
VarUsedWithFilter	specifies the data item name to be used together with a filter item. Only a Numeric Interval data item is a valid entry.	Customer Model Score No	no	-
Visible	If Visible is set to False for a data item, then the unfiltered targets of a filtered data item are hidden and metadata generation is faster. These hidden items can still be referenced as target items for filtered data items.	True False	no	True

Definitions of the Levels of Data Items

You can assign a data item to one of the following levels:

Table 6.7 *Definitions of the Levels of Data Items*

Level Value	Definition	Character	Numeric	Date
ID	identifies the data, but is not used in analysis.	yes	yes	yes
Unary	has one distinct value, including missing.	yes	yes	no
Binary	has two distinct values, including missing.	yes	yes	no
Nominal	has distinct values, but has no order (such as gender or marital status), including missing.	yes	yes	no
Interval	has many values, including infinite.	no	yes	yes
Ordinal	has distinct ordered values, such as income brackets, age groups, or scores.	yes	yes	no

The following display provides the valid combinations of the Level attribute with data type. **Classification** refers to how the data item is classified in SAS Information Map Studio.

Display 6.11 *Valid Data Type and Level Combinations*

Level	Data Type		
	Character	Numeric	Date
Id	✓	✓	✓
Unary	✓	✓	×
Binary	✓	✓	×
Nominal	✓	✓	×
Interval	×	✓	✓
Ordinal	✓	✓	×

Level	Classification	
	Category	Measure
Id	✓	×
Unary	✓	×
Binary	✓	×
Nominal	✓	×
Interval	✓	✓
Ordinal	✓	×

Understanding the Counts Metadata

What Is the Counts Metadata?

The *counts metadata* is metadata that is specifically generated for SAS Decision Manager views, and is distinct from the SAS metadata in the SAS Metadata Repository. The counts metadata is typically a list of values and a count of how often each value occurs in the database. Occasionally the counts metadata consists of a list of how many values fall into each of a set number of sub-intervals. The sub-intervals depend on the type of level that is selected for a data item.

A counts metadata value is read from a library reference (libref) of an information map. A counts metadata value is displayed within the nodes of a SAS Decision Manager diagram.

Descriptions of the Counts Metadata Tables

The counts metadata is stored in a series of SAS tables stored in the location that is defined by the MAMeta extended attribute at the map level. These are the metadata tables:

subject charcounts	is a list of character data items and the associated counts by subject.
*subject*numcounts	is a list of numeric data items and the associated counts by subject.

subject histocounts	contains information that is used to generate histograms for numeric data items by subject.
subject datehistocounts	information that is used to generate histograms for the date data items by subject.
univariatestats	min, max, mean, and standard deviation for interval variables. This attribute must be updated before you update either *subject*histocounts or *subject*datehistocounts.
physicalmapping	variable and physical reference data that is based on the information map. Must be updated before updating any of the other tables in this list.

For the first four metadata tables in the previous list, one table per subject exists. Metadata is generated only for items whose subject is set at the folder level. For example, for a folder whose Subject_ID_*x* is not specified for the household subject, no metadata is generated for any data item in the folder at the household level.

Generating the Counts Metadata

Overview of Generating the Counts Metadata

You generate the counts metadata in SAS Decision Manager. For more information, see *SAS Decision Manager User's Guide*.

When Should the Counts Metadata Be Generated?

Update the metadata cache only when the system does not have any active users. Generate or update the counts metadata in any of the following circumstances:

- The underlying data changes to ensure that all the possible values are available for use.
- A new folder is added to the information map.
- A new subject is added to a folder.
- New data items are added to the information map.
- Data item attributes that require support by the metadata are changed.

Update the Counts Metadata

You can manually update the counts metadata tables by using the custom property **CreateMetadataTable2** on the information map level. When you set this custom property to **`True`**, the generated metadata tables (from the underlying information map) are renamed by appending a 2 to the table names. For example, customercharcounts.sas7bdat is renamed customercharcounts2.sas7bdat. In order to update the old metadata tables, replace the contents of the original tables with the files that are appended with a 2, and then clear the metadata cache. For more information, see "When to Enable the Clear Metadata Cache Option" on page 77.

When to Use PROC SQL to Generate Metadata

If the information map is not complex, but the underlying data is very large, then generate the metadata by using the SAS SQL procedure (PROC SQL). In the PROC SQL method, an SQL query is streamed to the SAS procedure as XML. The XMLgenerates the counts for discrete variables and the bin ranges for the histogram variables.

Accessing the Tables of the Counts Metadata

If an information map has three subjects (such as household, customer, and account), the resulting metadata is contained in these tables:

- accountcharcounts
- accountdatehistocounts
- accounthistocounts
- customercharchounts
- customerdatehistocounts
- customerhistocounts
- customernumcounts
- householddatehistocounts
- householdhistocounts
- householdnumcounts
- physicalmapping
- univariatestats

The data that is contained in these tables can be accessed like any other SAS table. For example, you can use the following SAS code to check the data item variable:

```
LIBNAME MAMETA '\\machinename\SAS\MA\Data\MAMeta';
proc sql;
      select * from MAMeta.customercharcounts
      where Variable='Gender';
quit;
```

When to Enable the Clear Metadata Cache Option

In some environments, the counts are stored in a separate set of tables. The tables are copied into the SAS Decision Manager metadata library. The SAS Decision Managere engine, however, keeps a copy of those tables in memory. When you make changes to the metadata tables, the engine does not recognize and update it. Users do not see the new values of the counts metadata in the diagrams in SAS Decision Manager.

To enable SAS Decision Manager to recognize the new metadata values, clear the cache on the **Metadata** tab in the business context. For more information, see *SAS Decision Manager User's Guide*.

Chapter 7

Common Data Model: Concepts

About the Common Data Model **79**
What Is the Common Data Model? 79
Updating the Content of the Common Data Model 80
Managing Multiple Business Contexts 80
Reporting and the Common Data Model 80

How the Common Data Model Is Organized **80**
Overview 80
Campaigns or Flows 81
Campaign Groups 84
Communications 85
Cells, Packages, and Treatments 87
Control Groups 88
History 90
SAS Marketing Optimization 90

About Extension (_EXT) Tables **90**
Purpose of _EXT Tables 90
About Populating the _EXT Tables 91

About the History Table **92**
Overview of the History Table 92
Estimating the Size of the History Table 92
The CI_CONTACT_HISTORY Table 92

About Lookup Tables **93**

About the Common Data Model

What Is the Common Data Model?

The SAS Customer Intelligence Common Data Model is a common data repository for SAS Customer Intelligence solutions, including SAS Marketing Automation and SAS Real-Time Decision Manager, and for SAS Decision Manager.

Some solutions use the common data model to store data such as campaigns that have been conducted, communications that have been run, offers that have been presented, and customer responses.

All of the business contexts for your site can share a single version of the common data model. Alternatively, you can specify a separate version of the model for each business

context. For more information, see "Managing Multiple Business Contexts" on page 80.

For descriptions of the tables and their columns in the common data model, see *SAS Customer Intelligence Common Data Model: Data Dictionary* at **http://support.sas.com/documentation/solutions/dcm/index.html**. You must supply the following user name and password to view this site:

```
User Name: sas
Password: CIadmin123
```

A diagram of the model is included at the end of the data dictionary. Use the diagram to understand the relationships among the tables that comprise the model.

Updating the Content of the Common Data Model

The common data model is typically updated when you save a campaign or flow (after the selection campaign, decision campaign, or decision flow has been published), or each time you execute a campaign. A campaign or flow is not published if the publish operation is not enabled, and if the location of the data model for the respective business context is not specified.

For every occurrence of a communication, the contact history records for that communication occurrence are first deleted and repopulated with a newer set of contact records. This update happens only if the **Update contact history** flag of the Communication node is enabled.

Managing Multiple Business Contexts

In an environment where multiple business contexts exist, you can deploy the common data model and capture data across business contexts in two ways.

- Specify a single instance of the common data model. Specify that all of the business contexts point to the same reporting libref. The **BUSINESS_CONTEXT** field in the CI_CAMPAIGN table captures the business context that is associated with the campaign or flow. In this pattern, all of the campaign or flow data across the enterprise resides in the same location.
- Create a separate instance of the common data model for each business context (recommended). In this pattern, the data of a business context resides at different locations, and each instance of the data model contains only the information for a single business context.

Reporting and the Common Data Model

When campaign or flow data is published to the common data model tables, users can view this data in reports.

How the Common Data Model Is Organized

Overview

The tables of the common data model can be organized conceptually in the following categories:

- Campaigns or flows. For more information, see “Campaigns or Flows” on page 81.
- Campaign groups. For more information, see “Campaign Groups” on page 84.
- Communications. For more information, see “Communications” on page 85.
- Cells, packages, and treatments. For more information, see “Cells, Packages, and Treatments” on page 87.
- Control groups. For more information, see “Control Groups” on page 88.
- History and responses. For more information, see “History ” on page 90.

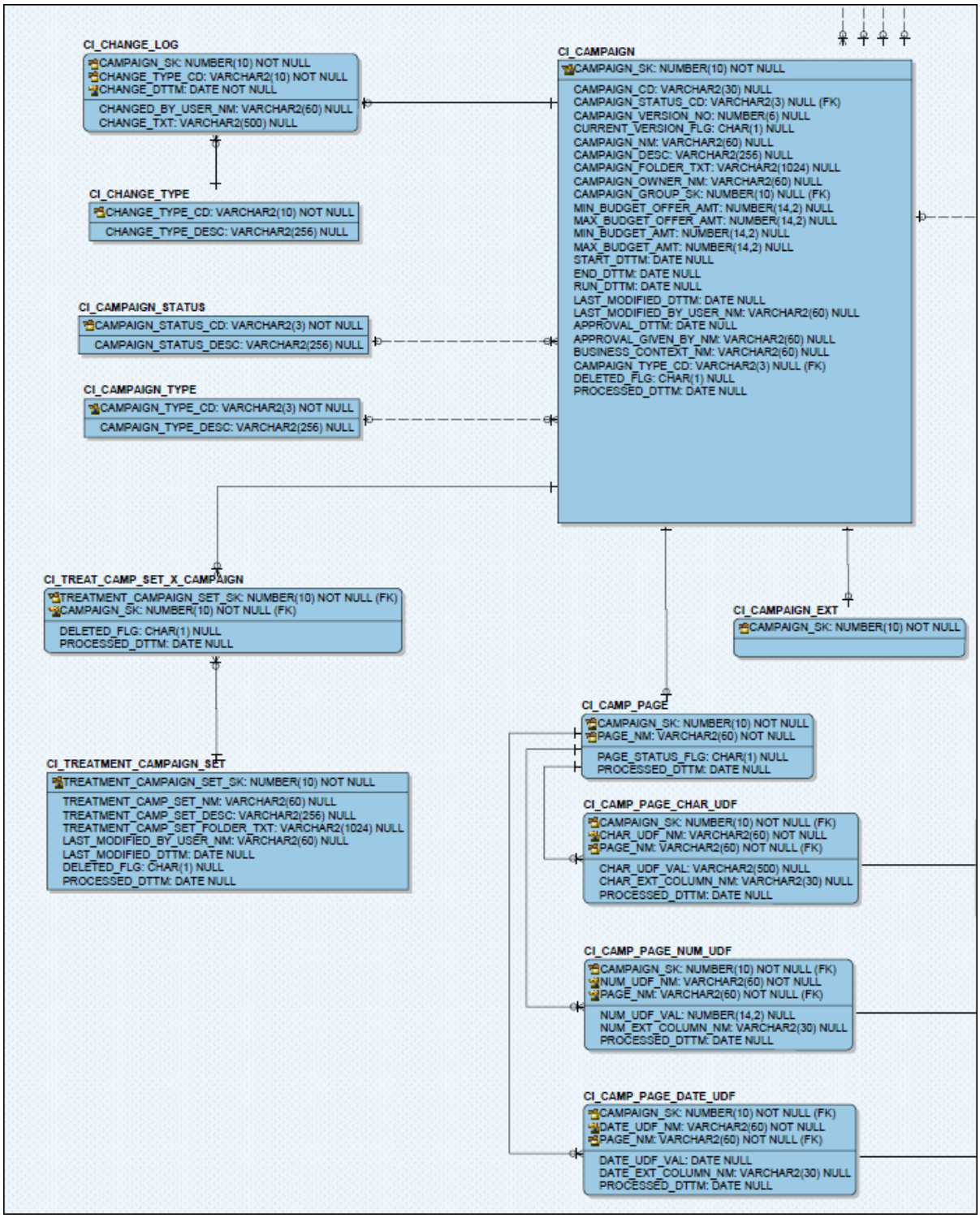
To implement these tables, see Chapter 8, “Implementing the Common Data Model,” on page 99.

Campaigns or Flows

Campaign Tables

Campaign tables contain campaign information for SAS Marketing Automation and SAS Real-Time Decision Manager, and flow information for SAS Decision Manager.

Display 7.1 *Campaign Tables*



CI_CAMPAIGN

The CI_CAMPAIGN table is the central table to which the other tables in the campaigns category connect. A campaign contains a planned set of one or more communications over one or more channels that are directed at a group of subjects such as customer, household, or individual.

CI_CAMPAIGN_EXT

The CI_CAMPAIGN_EXT table contains the metadata that links the checklist items and any customized checklist steps to a specific campaign. For more information, see "About Extension (_EXT) Tables" on page 90.

CI_CAMPAIGN_STATUS

The CI_CAMPAIGN_STATUS table is a reference table that contains the status codes and status descriptions for a campaign. This table is automatically populated when the common data model is installed. Additional user-defined status codes can be specified. For the default populated values, see "About Lookup Tables" on page 93.

CI_CAMPAIGN_TYPE

The CI_CAMPAIGN_TYPE table is a reference table that contains the types of campaigns. This table is automatically populated with the values Selection and Decision when the common data model is installed.

CI_CAMP_PAGE

The CI_CAMP_PAGE table contains the custom details for the campaign. A number of _UDF (user-defined field) tables are associated with this table.

For information about _UDF tables, see "UDF (User-Defined Field) Tables" on page 91.

CI_CHANGE_LOG

The CI_CHANGE_LOG table records the version changes to a campaign. When you create the campaign, a row in the CI_CHANGE_LOG table is populated with the initial date and time that the campaign is created. When the campaign version changes, a new row is added to the table. Each row contains the date and time of a version change. Any other changes to the campaign do not update the CI_CHANGE_LOG table.

CI_CHANGE_TYPE

The CI_CHANGE_TYPE table is referenced by the CI_CHANGE_LOG table. The CI_CHANGE_TYPE table contains the types of changes that are stored in the change log. This table is automatically populated when the common data model is installed. Currently, the only supported type of change that can be recorded for a campaign is version type.

CI_TREATMENT_CAMPAIGN_SET

The CI_TREATMENT_CAMPAIGN_SET table is a set of treatment campaigns that are applicable only in SAS Real-Time Decision Manager campaigns. After each treatment campaign has determined the eligibility of an offer, the campaign returns the information to the treatment campaign set. The treatment campaign set compares the offers and returns the best offers to the individual customers.

CI_TREAT_CAMP_SET_X_CAMPAIGN

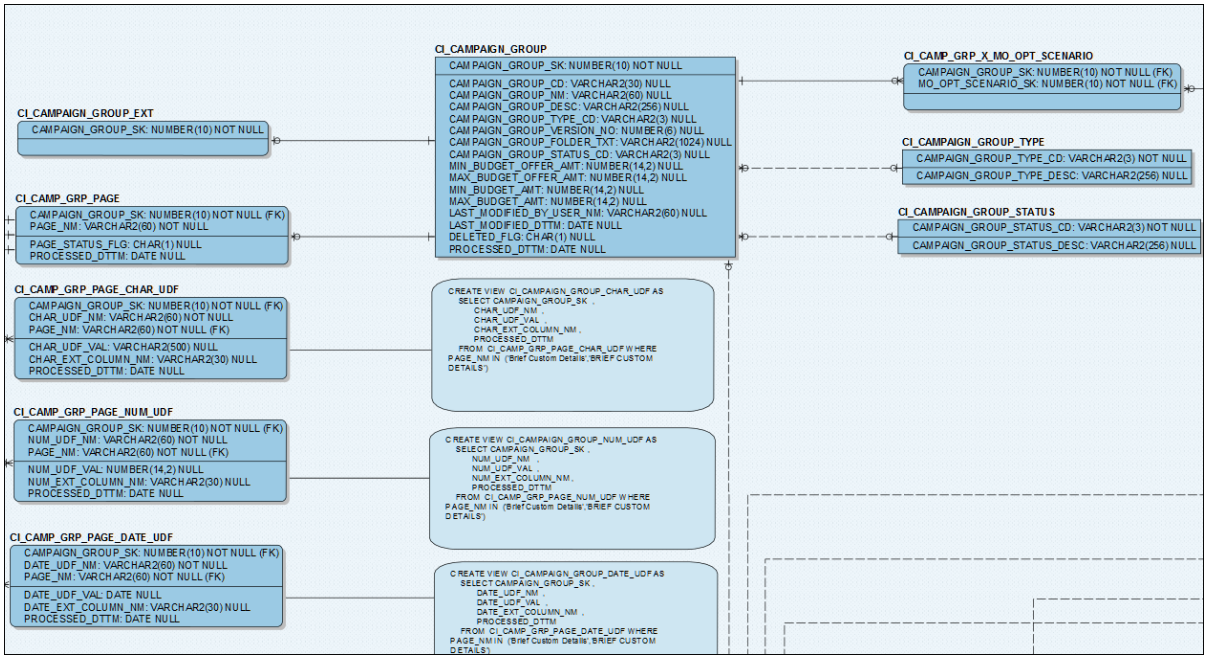
The CI_TREAT_CAMP_SET_X_CAMPAIGN table is an intersection table that models the many-to-many relationship between treatments and campaigns. When a treatment campaign is removed from the treatment campaign set, a clean publish operation removes the treatment campaign from this table.

Campaign Groups

Campaign Group Tables

Campaign group tables are used by SAS Marketing Automation.

Display 7.2 *Campaign Group Tables*



CI_CAMPAIGN_GROUP

The CI_CAMPAIGN_GROUP table is the central table to which the other tables in the campaign groups category connect. The CI_CAMPAIGN_GROUP table is associated through CAMPAIGN_GROUP_SK to the CI_CAMPAIGN table for all campaigns that are a part of the campaign group. A campaign group bundles campaigns together. Either SAS Marketing Automation or SAS Marketing Optimization can access data about campaign groups.

CI_CAMPAIGN_GROUP_EXT

The CI_CAMPAIGN_GROUP_EXT table contains additional custom details. To add a custom detail, see the *SAS Marketing Automation User's Guide*. For more information, see "About Extension (_EXT) Tables" on page 90.

CI_CAMPAIGN_GROUP_STATUS

The CI_CAMPAIGN_GROUP_STATUS table is a reference table that contains the status codes and descriptions for a campaign group. This table is prepopulated when the common data model is installed.

CI_CAMPAIGN_GROUP_TYPE

The CI_CAMPAIGN_GROUP_TYPE table is a reference table that contains the types of campaign groups. This table is prepopulated when the common data model is installed.

CI_CAMP_GRP_PAGE

The CI_CAMP_GRP_ PAGE table contains the custom details for the campaign group. A number of _UDF (user-defined field) tables are associated with this table.

For information about _UDF tables, see "UDF (User-Defined Field) Tables" on page 91.

CI_CAMP_GRP_X_MO_OPT_SCENARIO

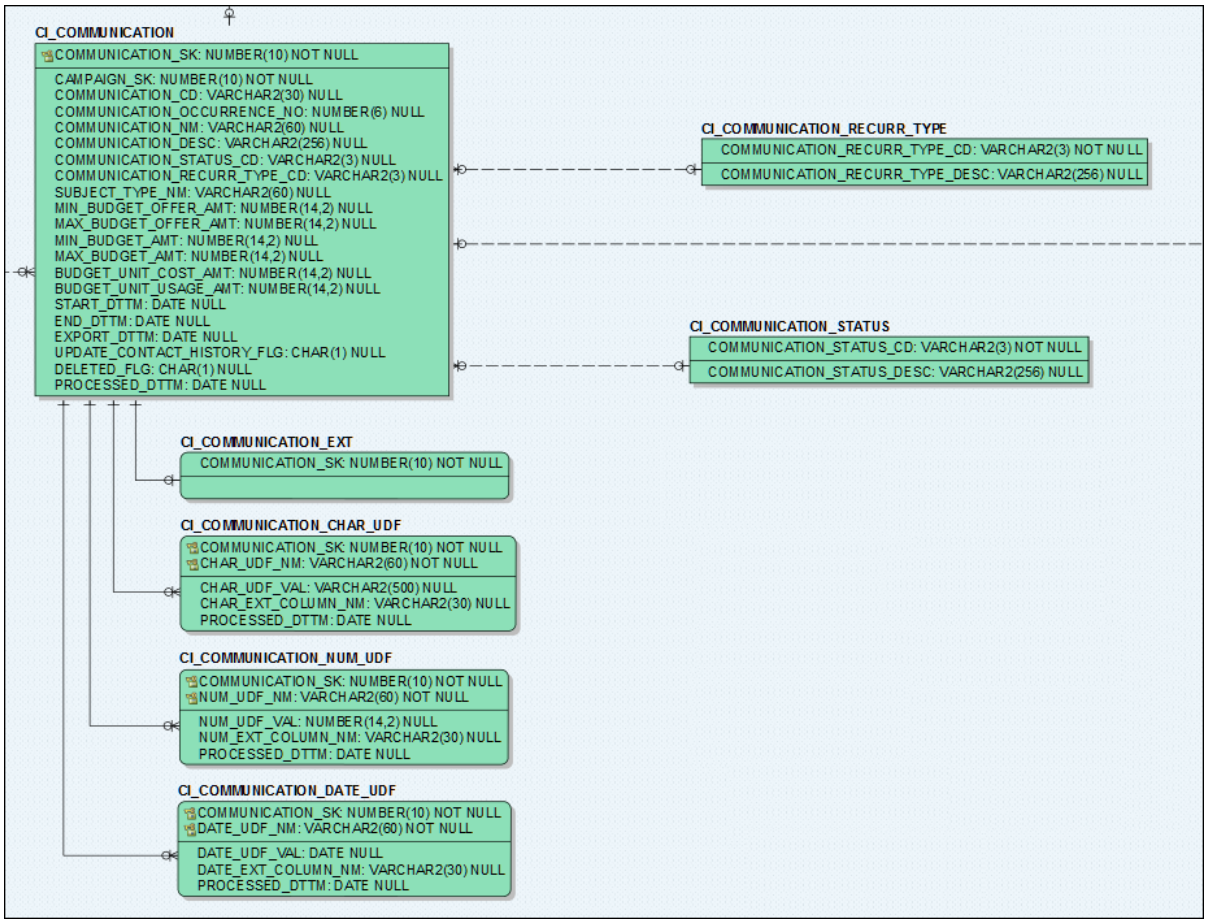
The CI_CAMP_GRP_X_MO_OPT_SCENARIO table is an intersection table that models the many-to-many relationship between campaign groups and SAS Marketing Optimization scenarios.

Communications

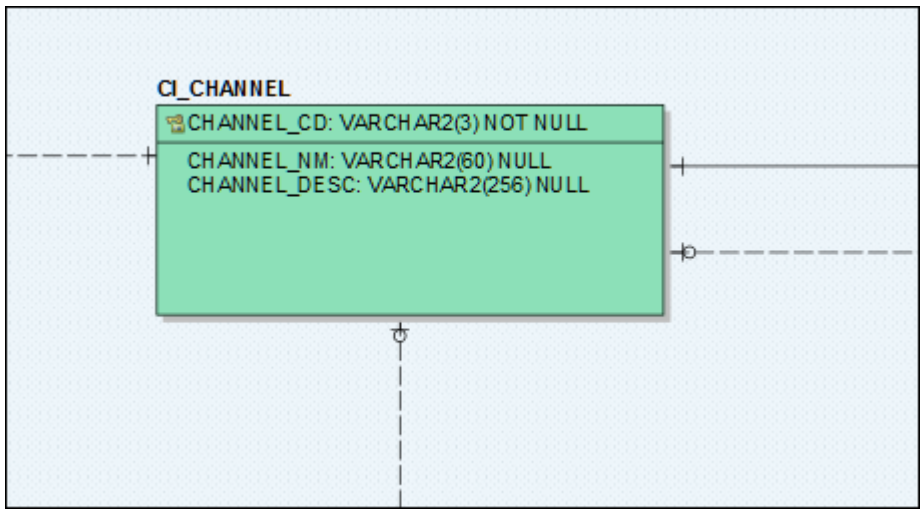
Communication Tables

Communication tables are used by SAS Marketing Automation.

Display 7.3 *CI_COMMUNICATION_ Tables*



Display 7.4 *CI_CHANNEL Table*



CI_CHANNEL

The CI_CHANNEL table is a reference table. It contains the channels, channel codes, and channel descriptions that are typically used in a campaign. A channel represents the means of communication (such as e-mail) between a company and targeted customers. This table is automatically populated with common channel codes. For more information, see "About Lookup Tables" on page 93.

CI_COMMUNICATION

The CI_COMMUNICATION table contains the general descriptive information about a communication. A communication represents the delivery of a package of treatments over a channel to particular cells. Each communication occurrence is represented by a row. A communication represents a contact through a channel and might or might not be associated with a set of treatments. A treatment represents a marketing message and its content that are delivered over a channel.

CI_COMMUNICATION_EXT

The CI_COMMUNICATION_EXT table contains additional custom details. To add a custom detail, see the *SAS Marketing Automation User's Guide*. For more information, see "About Extension (_EXT) Tables" on page 90.

CI_COMMUNICATION_RECURR_TYPE

The CI_COMMUNICATION_RECURR_TYPE table is a reference table that contains the types of communication recurrences. This table is prepopulated with the values None, Hourly, Daily, Weekly, Monthly, and Other, when the common data model is installed.

CI_COMMUNICATION_STATUS

The CI_COMMUNICATION table references the CI_COMMUNICATION_STATUS table, a lookup table that contains the status codes and status code descriptions for a communication. For the default values, see "About Lookup Tables" on page 93.

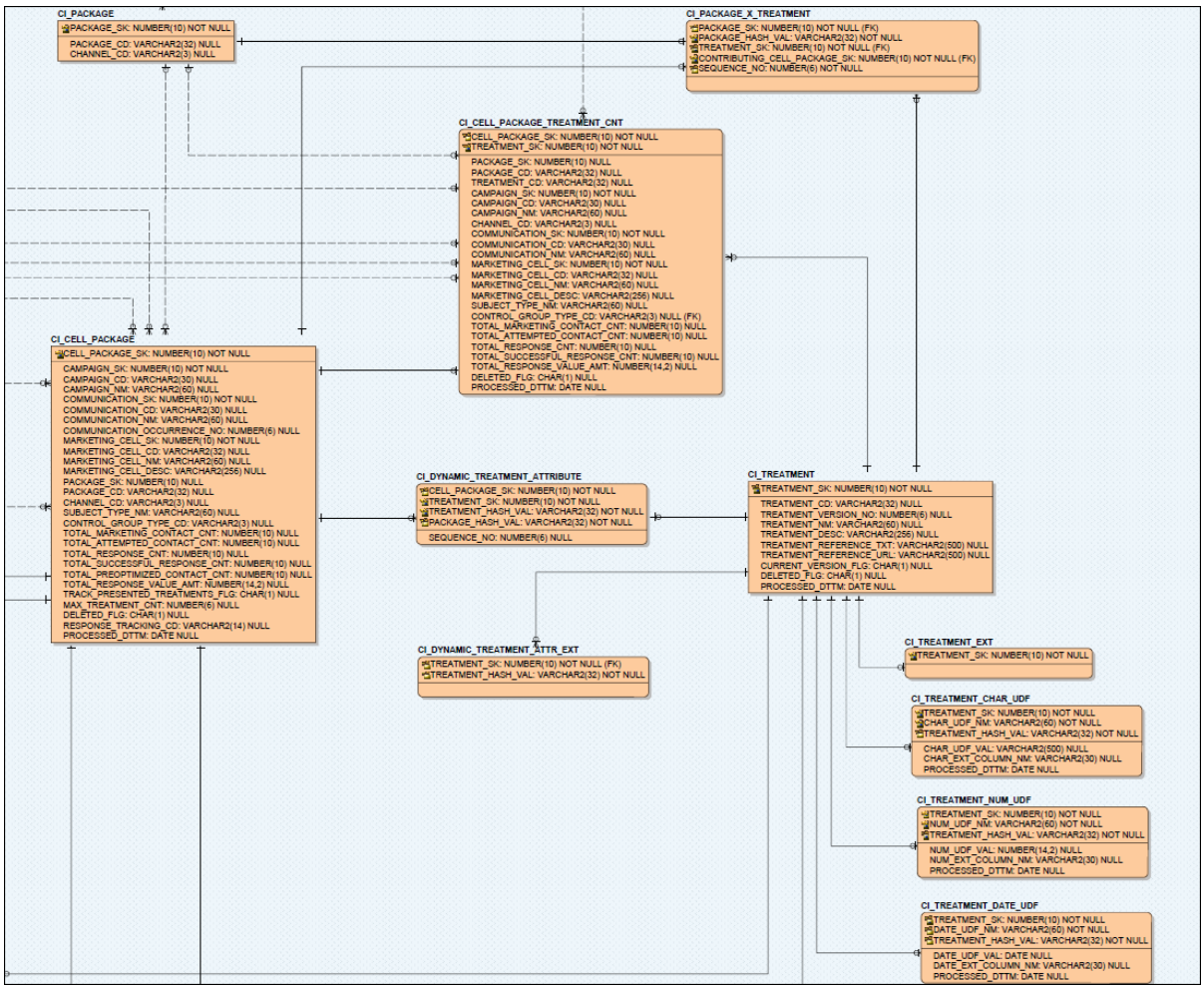
_UDF Tables

For information about _UDF tables, see "UDF (User-Defined Field) Tables" on page 91.

Cells, Packages, and Treatments

Cell, Package, and Treatment Tables

Display 7.5 *Cell, Package, and Treatment Tables*



CI_CELL_PACKAGE

The CI_CELL_PACKAGE table contains columns to associate marketing cells, communications, and treatments with a package in order to facilitate reporting. This table also includes response tracking at a cell level and summary counts for contacts and responses at a package level. The counts are updated when the campaign is executed.

CI_CELL_PACKAGE_TREATMENT_CNT

The CI_CELL_PACKAGE_TREATMENT_CNT table contains the counts and reference information about the individual treatments that are assigned to a marketing cell. This information is included if a campaign is identified by the user as a campaign for tracking treatment level presentations (as opposed to simply package level contacts). The counts are updated when the campaign is executed.

For packages that do not have treatments, the data is an exact replica of the CI_CELL_PACKAGE table and the TREATMENT_SK column from the

CI_TREATMENT table. This column is associated with a treatment code such as Not Treatment.

CI_DYNAMIC_TREATMENT_ATTRIBUTE

The CI_DYNAMIC_TREATMENT_ATTRIBUTE table associates contact history records with the dynamic treatments that were actually executed for a specified campaign. In dynamic treatments, values can change each time that a campaign is run.

CI_DYNAMIC_TREATMENT_ATTR_EXT

The CI_DYNAMIC_TREATMENT_ATTR_EXT table contains data from the CI_TREATMENT_CHAR_EXT, CI_TREATMENT_NUM_EXT, and CI_TREATMENT_DATA_EXT tables that is used for dynamic custom details. For more information, see "About Extension (_EXT) Tables" on page 90.

CI_PACKAGE

The CI_PACKAGE table contains a collection of one or more treatments. A package is used as a collection point for one or more treatments. Treatments are delivered to the channel as part of a package that is associated with a campaign.

CI_PACKAGE_X_TREATMENT

The CI_PACKAGE_X_TREATMENT table is an intersection table that models the many-to-many relationship between packages and treatments. A package is basically a group of offers, called treatments. Therefore, a package consists of several treatments. A single treatment can also be a part of multiple packages.

CI_TREATMENT

The CI_TREATMENT table contains the treatments that are associated with a campaign. A treatment represents a marketing message and its content that are delivered over a channel. Treatments can be combined into a package.

When a business user defines a treatment, that user can specify whether the treatment should be static or dynamic. Static treatments are named because their values do not change as the treatment or treatments are applied throughout the execution of a campaign.

The dynamic attribute of a treatment in a decision campaign can be specified as either a variable or a manually defined value.

CI_TREATMENT_EXT

The CI_TREATMENT_EXT table contains additional custom details.For more information, see "About Extension (_EXT) Tables" on page 90.

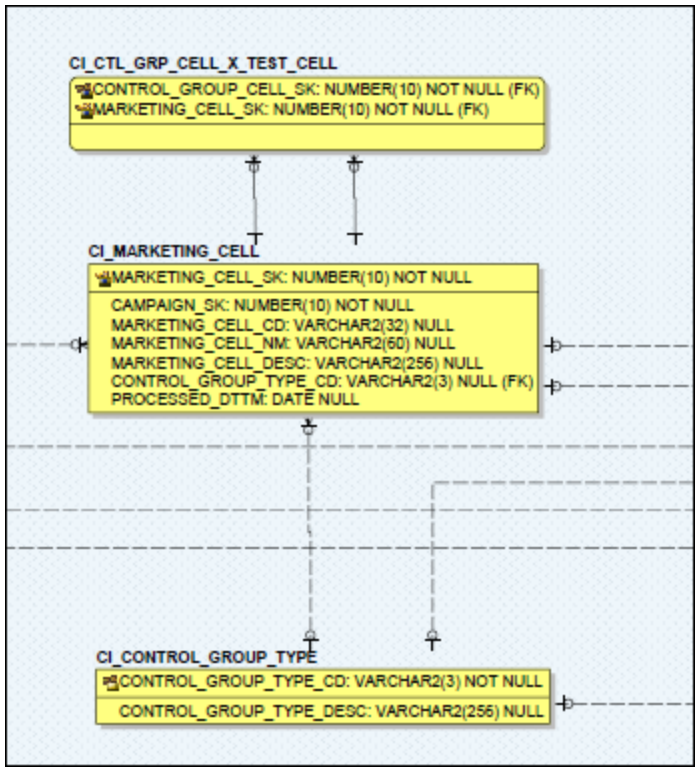
_UDF Tables

For information about _UDF tables, see "UDF (User-Defined Field) Tables" on page 91.

Control Groups

Control Group Tables

Display 7.6 *Control Group Tables*



CI_CONTROL_GROUP_TYPE

The CI_CONTROL_GROUP_TYPE table contains the types of control groups that are used for reporting. The supplied type is _ST (which stands for Standard). This table is automatically populated when the common data model is installed. You can supply additional user-defined types. For more information, see Table 7.11 on page 97.

In the common data model, these associations are established.

- For champion/challenger settings, the champion is the control group that the challengers are compared with.
- For challenger/challenger settings, each challenger is a control group for all the other challengers in a node, so that any two challengers can be compared with each other.

CI_CTL_GRP_CELL_X_TEST_CELL

The CI_CTL_GRP_CELL_X_TEST_CELL table is an intersection table that enables the mapping between control groups and the marketing cells with which the control groups are associated. The values for CONTROL_GROUP_CELL_SK and MARKETING_CELL_SK are added as rows in the table when a campaign is published. This table is used for holdout control groups as well as for champion/challenger and challenger/challenger control groups.

CI_MARKETING_CELL

The CI_MARKETING_CELL table contains a grouping of subjects that are targeted by a campaign. An example of a subject is customer, household, or individual.

History

History Table

CI_CONTACT_HISTORY

The CI_CONTACT_HISTORY table contains a contact record of the delivery of a particular package to a particular channel for a particular subject. This table references the CI_CONTACT_HISTORY_STATUS table.

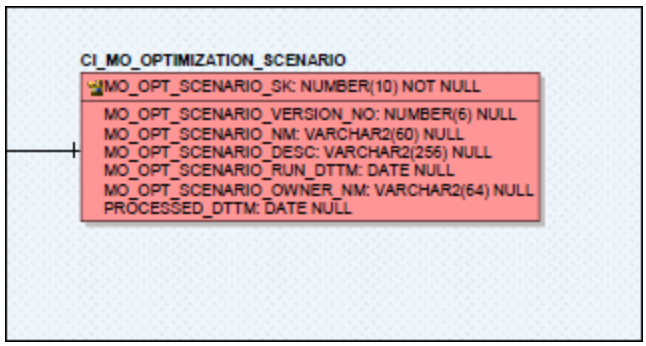
CI_CONTACT_HISTORY_STATUS

The CI_CONTACT_HISTORY_STATUS table is a lookup table that contains the status codes and status descriptions for a contact history record. For the default values, see "About Lookup Tables" on page 93.

SAS Marketing Optimization

SAS Marketing Optimization Tables

Display 7.7 *SAS Marketing Optimization Tables*



CI_MO_OPTIMIZATION_SCENARIO

The CI_MO_OPTIMIZATION_SCENARIO table contains columns that associate marketing optimization scenarios with campaign groups. This table enables associations between SAS Marketing Optimization and SAS Marketing Automation data. For information about the columns in this table, see *SAS Customer Intelligence Common Data Model: Data Dictionary* at **http://support.sas.com/documentation/solutions/ci/index.html**.

About Extension (_EXT) Tables

Purpose of _EXT Tables

Extension (_EXT) tables are typically created in order to contain any additional information that is specific to your business practices such as the metadata for campaign groups, campaigns, communications, or treatments.

About Populating the _EXT Tables

Overview

The _EXT tables are not initially structured and are initially empty. As part of the customizing process, you define the new column structures that you want to add to the _EXT tables. This is usually a one-time database administrator task.

When a campaign or communication is executed, those name/value pairs in the columns of a UDF table are transposed to the rows of the associated _EXT table. The rows of the _EXT tables are populated with data.

Note: For list type custom details, the value, rather than the display value, is used to populate the _EXT table.

The CI_DYNAMIC_TREATMENT_ATTR_EXT table is populated with the dynamic custom details from the CI_TREATMENT_*_UDF tables and the CI_TREATMENT_EXT table is populated with the static custom details from the CI_TREATMENT_*_UDF tables.

For more information, see "Customizing the Extension (EXT) Tables" on page 104.

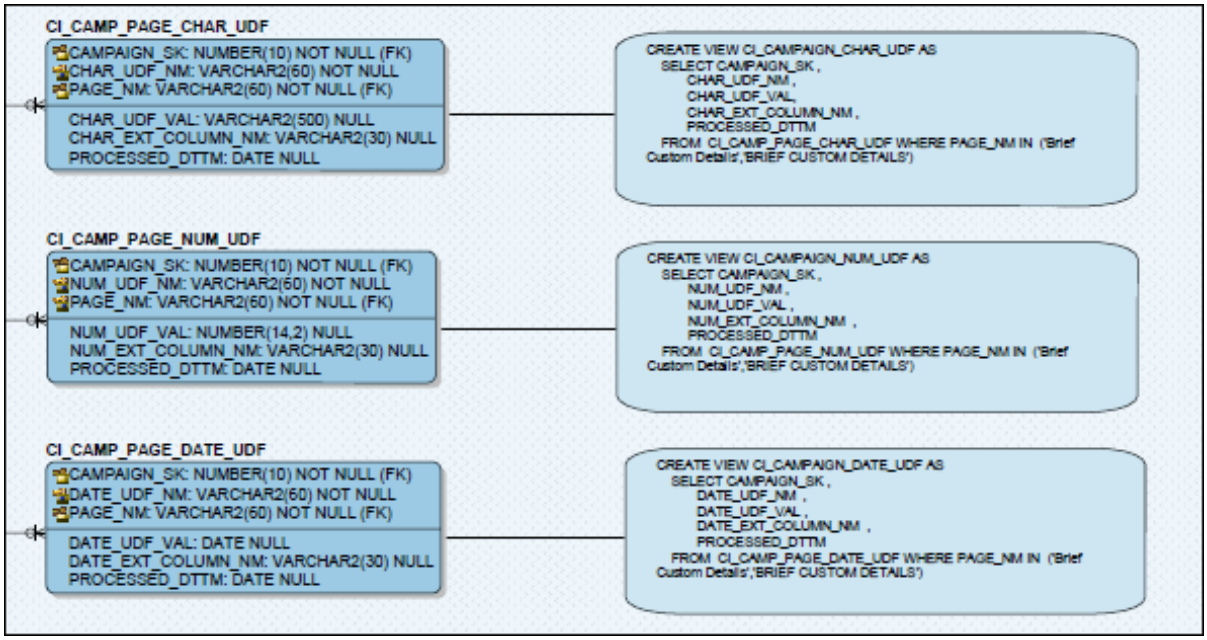
To add a custom detail, see *SAS Marketing Automation User's Guide* or *SAS Real-Time Decision Manager User's Guide*.

UDF (User-Defined Field) Tables

A UDF table exists for each data type (character, numeric, and date). When a campaign or flow is published, a row is added to the associated UDF table for each custom detail that is defined within the campaign.

For campaigns, flows, and campaign groups, views are associated with UDF tables.

Display 7.8 *Example of UDF Views*



The following views are associated with campaign, flow, and campaign group UDF tables.

Table 7.1 Views Associated with UDF Tables

Table	View
CI_CAMP_PAGE_CHAR_UDF WHERE PAGE_NM = 'Brief Custom Details'	CI_CAMPAIGN_CHAR_UDF
CI_CAMP_PAGE_NUM_UDF WHERE PAGE_NM = 'Brief Custom Details'	CI_CAMPAIGN_NUM_UDF
CI_CAMP_PAGE_DATE_UDF WHERE PAGE_NM = 'Brief Custom Details'	CI_CAMPAIGN_DATE_UDF
CI_CAMP_GRP_PAGE_CHAR_UDF WHERE PAGE_NM = 'Brief Custom Details'	CI_CAMPAIGN_GROUP_CHAR_UDF
CI_CAMP_GRP_PAGE_NUM_UDF WHERE PAGE_NM = 'Brief Custom Details'	CI_CAMPAIGN_GROUP_NUM_UDF
CI_CAMP_GRP_PAGE_DATE_UDF WHERE PAGE_NM = 'Brief Custom Details’	CI_CAMPAIGN_GROUP_DATE_UDF

About the History Table

Overview of the History Table

Decision history is recorded in the CI_CONTACT_HISTORY table. The CI_CONTACT_HISTORY table has the following requirements or properties:

- It is deployed in the same location as the common data model.
- It must be registered in the SAS Metadata Repository.
- It must be included in a SAS Information Map in order to make the decision history data available for use in SAS Decision Manager.

Estimating the Size of the History Table

To estimate maximum table size, multiply the estimated number of rows in each history table by the size of each row (in bytes). Then multiply the result by the number of years that you want to retain the history table data for each table.

Note: The CI_CONTACT_HISTORY table is the only table in the common data model that might increase in size as responses are recorded.

The CI_CONTACT_HISTORY Table

What the CI_CONTACT_HISTORY Table Contains

The CI_CONTACT_HISTORY table contains the following columns:

CELL_PACKAGE_SK
is the primary reference key that associates the decision history with a cell package.

CONTACT_DTTM
is the date and time at which the subject (such as Customer, Account, or Household) is contacted.

CONTACT_DT
represents the mm/dd/yyyy value of CONTACT_DTTM.

CONTACT_HISTORY_STATUS_CD
is the reference key code that associates the decision history with a decision history status. See the CI_CONTACT_HISTORY_STATUS table for the supplied values of the status codes.

EXTERNAL_CONTACT_INFO_ID1 and EXTERNAL_CONTACT_INFO_ID2
are columns for SAS Real-Time Decision Manager contact processing.

OPTIMIZATION_BACKFILL_FLG
is a column for use by SAS Marketing Optimization.

PACKAGE_HASH_VALUE
contains a hash value generated to supply a single lookup value for a combination of the dynamic custom details for a package.

RESPONSE_TRACKING_CD
contains the response code.

SUBJECT_ID
is the primary reference key of a unique identifier such as Subject ID (for example, Customer, Account, or Household) that associates the CI_CONTACT_HISTORY table with an external table. You need to replace the placeholder value that is assigned for this column during installation with a valid key to the appropriate external table.

Note: SAS Technical Support does not recommend adding fields to the table. If you need additional contact-related fields, then create a separate table and link the information to the history table by using a key.

Structure of the CI_CONTACT_HISTORY Table: Subject Levels

Create a separate table for each subject level for which contacts are made.

The CI_CONTACT_HISTORY table contains one row for each subject for each occurrence.

About Lookup Tables

Lookup tables contain information about campaigns in SAS Marketing Automation and SAS Real-Time Decision Manager, and flows in SAS Decision Manager. These lookup tables, also called reference tables, are prepopulated with typical industry values:

- CI_CAMPAIGN_GROUP_STATUS
- CI_CAMPAIGN_GROUP_TYPE
- CI_CAMPAIGN_STATUS
- CI_CAMPAIGN_TYPE
- CI_CHANGE_TYPE

- CI_CHANNEL
- CI_COMMUNICATION_RECURR_TYPE
- CI_COMMUNICATION_STATUS
- CI_CONTACT_HISTORY_STATUS
- CI_CONTROL_GROUP_TYPE
- CI_RESPONSE
- CI_RESPONSE_TYPE

Note: These values are supplied when the common data model is created. Here are the tables, with their prepopulated values. Your lookup tables might contain either different or additional values if they have been customized.

Table 7.2 *CI_CAMPAIGN_GROUP_STATUS Table: Prepopulated Values*

Campaign Group Status Code	Campaign Group Status Description
__1	Not Ready
__2	Ready to Optimize

Table 7.3 *CI_CAMPAIGN_GROUP_TYPE Table: Prepopulated Values*

Campaign Group Type Code	Campaign Group Type Description
OI_	Inbound Optimization Group
OO_	Outbound Optimization Group
O__	Other Optimization Group
_I_	Inbound Campaign Group
_O_	Outbound Campaign Group
___	Other

Campaigns that are migrated from a previous release retain their original campaign status codes. The codes are not converted to the values in the following table.

Table 7.4 *CI_CAMPAIGN_STATUS Table: Prepopulated Values*

Campaign Status Code	Campaign Status Description
1_1	Initiating
1_2	Initiation Complete
1_3	Designing

1_4	Design Complete
2_1	Approval Requested
2_2	Approval Denied
2_3	Approval Approved
3_1	Ready to Execute
3_2	Scheduled
3_3	Executing
3_4	Execute Complete
3_5	Marked for Deployment

Table 7.5 *CI_CAMPAIGN_TYPE Table: Prepopulated Values*

Campaign Type Code	Campaign Type Description
I	Decision
O	Selection
T	Treatment

Table 7.6 *CI_CHANGE_TYPE Table: Prepopulated Value*

Change Type Code	Change Type Description
_NV	New Version

Table 7.7 *CI_CHANNEL Table: Prepopulated Values*

Channel Code	Channel Name	Channel Description
_AG	Agent	Agent channel
_AT	ATM	ATM channel
_BT	Batch	Batch channel
_CA	Calendar	Calendar channel
_CC	Call Center	Call center channel

_CT	Catalog	Catalog channel
_EL	Electronic	Electronic channel
_EM	Email	E-mail channel
_EV	Event	Event channel
_FX	Fax	Fax channel
_GI	General insert	General insert channel
_MC	Multichannel	Multiple potential channels
_ML	Mail	Mail channel
_MP	Mobile phone	Mobile phone channel
_OP	Off the page	Off the page channel
_PA	Print Ad	Print advertisement channel
_PB	Point of Sale Branch	Point of Sale (POS) branch channel
_PG	Pager mail	Pager mail channel
_PH	Phone	Telephone channel
_PM	Phone mail	Phone mail channel
_RD	Radio	Radio channel
_SC	SAS Conversation Center	SAS Conversation Center channel
_SF	Point of Sale Sales Force	Point of Sale (POS) sales force channel
_SI	Statement Insert	Statement insert channel
_SM	Statement Message	Statement message channel
_SO	Social Media	Social Media channel
_SP	Sales Promo	Sales promotion channel
_TV	TV	Television channel
_WB	Web	Web channel

_WC	Web Campaign	SAS Web Analytics Campaign

Table 7.8 *CI_COMMUNICATION_RECURR_TYPE Table: Prepopulated Values*

Communication Recurr Type Code	Communication Recurr Type Description
__H	hourly
__D	daily
__W	weekly
__M	monthly
__N	none
__U	unknown

Table 7.9 *CI_COMMUNICATION_STATUS Table: Prepopulated Values*

Communication Status Code	Communication Status Description
_11	Executed
_30	Failed
_32	Future

Table 7.10 *CI_CONTACT_HISTORY_STATUS Table: Prepopulated Values*

Contact History Status Code	Contact History Status Description
_11	Executed
_13	Excluded Mandatory Contacts
_20	Control Group
_30	Failed

Table 7.11 *CI_CONTROL_GROUP_TYPE Table: Prepopulated Values*

Control Group Type Code	Control Group Type Description
_AU	Automatic

_CL	Challenger
_CP	Champion
_ST	Manual Manual control groups are created by selecting the **Cell represents a control group** option in the Cell node.

Table 7.12 *CI_RESPONSE Table: Prepopulated Values*

Response Code	Response Name
_EC	E-mail - Clicked Link
_EO	E-mail - Opened
_ER	E-mail - Reply

Table 7.13 *CI_RESPONSE_TYPE Table: Prepopulated Values*

Response Type Code	Response Type Description
_CV	Converted
_RS	Responded

Chapter 8

Implementing the Common Data Model

Overview of Steps **99**

Creating the Table Structure of the Common Data Model **100**
Overview of DDL Scripts 100
How to Find Your DDL Scripts for Creating the Common Data Model 101
Specify Your Database Connection Values 101

Setting Up the History Table **103**
Overview: Setting Up the History Tables 103
Update the Placeholder SUBJECT_ID Columns 103

Customizing the Extension (EXT) Tables **104**
Overview 104
Modify the Definition of an Extension Table 105

Setting Up the Lookup Tables **106**
Prepopulate the Lookup Tables 106
Resolving Translated Data 106

Registering Data Sources **106**
Updating the Libref for the Common Data Model 106

Saving, Publishing, and the Common Data Model **107**
About Saving versus Publishing 107
Full and Incremental Publish Operations 107

Overview of Steps

This section contains the steps for creating the table structure of the common data model. The common data model is typically created as part of the SAS Decision Manager installation. The common data model is used by SAS Marketing Automation, SAS Real-Time Decision Manager, and SAS Decision Manager. SAS Decision Manager and SAS Real-Time Decision Manager use the same tables in the common data model. For conceptual information about the common data model, see Chapter 7, “Common Data Model: Concepts,” on page 79.

To implement the common data model, perform the following steps:

1. Verify that the user and group authorizations are specified for the users who start the Customer Intelligence Common Web Service in order to access the metadata for the business context. For more information, see Chapter 3, “Users and Groups,” on page 21.

2. Create the table structure for common data model tables in your database. For more information, see "Creating the Table Structure of the Common Data Model" on page 100.
3. Set up the history tables. For more information, see "Setting Up the History Table" on page 103.
4. Customize the _EXT tables. To add the user-defined fields for the associated UDF tables, For more information, see "Customizing the Extension (EXT) Tables" on page 104.
5. Set up the lookup tables. For more information, see "Setting Up the Lookup Tables" on page 106.
6. Create the library definition for the common data model. For details, see **SAS Intelligence Platform: Data Administration Guide**.
7. Edit your business contexts to include the location of the metadata for the common data model tables.

Creating the Table Structure of the Common Data Model

Overview of DDL Scripts

SAS Decision Manager provides the data definition language (DDL) scripts that are contained in a program called ci_cdm_ddl_<*db*>.sas. You execute these scripts to create the column structure of the common data model tables in your SAS environment. The SAS system administrator modifies these DDL scripts to customize some of the tables.

To set up the common data model tables, open ci_cdm_ddl_<*db*>.sas (where <*db*> is the database that you are using, as specified in ."How to Find Your DDL Scripts for Creating the Common Data Model" on page 101). Then follow these steps:

- Customize the history tables. Scroll down to BEGIN HISTORY SECTION and follow the instructions for modifying the history tables. For more information, see "Setting Up the History Table" on page 103.
- Customize the user-defined fields for all UDF tables. Scroll down to BEGIN USER DEFINED SECTION and follow the instructions for modifying the UDF tables.
- Customize the extension tables:
  - CI_CAMPAIGN_EXT
  - CI_CAMP_CHECKLIST_ITEM_EXT
  - CI_CAMPAIGN_GROUP_EXT
  - CI_CAMP_GRP_CLIST_ITEM_EXT
  - CI_COMMUNICATION_EXT
  - CI_DYNAMIC_TREATMENT_ATTR_EXT
  - CI_TREATMENT_EXT

  Note: CI_CAMPAIGN_EXT columns that are meant to hold dates must use a database type that holds date and time.

Scroll down to BEGIN EXTENSION SECTION and follow the instructions for modifying the extension tables.

After customizing the ci_cdm_ddl_<*db*>.sas file, execute it via SAS to place the tables in the database. When this is complete, the ci_cdm_key_<*db*>.sas file might need to be customized to provide the needed unique key and referential integrity for the database tables. Not all supported database management systems have referential integrity or unique key capability. DB2, Oracle, and SQL Server need the ci_cdm_key_<*db*>.sas customized and executed. Scalable Performance Data Server and Teradata include any unique key or referential integrity requirements in the ci_cdm_ddl_<*db*>.sas file. Netezza does not support unique key or referential integrity.

How to Find Your DDL Scripts for Creating the Common Data Model

The following table lists the name of the program that contains the DDL script for each database type:

Database	Program Name
DB2	ci_cdm_ddl_db2.sas
Greenplum	ci_cdm_ddl_greenplum.sas
Netezza	ci_cdm_ddl_netezza.sas
Oracle	ci_cdm_ddl_oracle.sas
Scalable Performance Data server	ci_cdm_ddl_spds.sas
SQL Server (Windows only)	ci_cdm_ddl_sqlserver.sas
Teradata	ci_cdm_ddl_teradata.sas

Find the program at the following default locations:

- Under Windows: **`Program Files\SASHome\SASFoundation\9.4\dcmsvr\sasmisc`**
- Under UNIX: **`/SASHome/SASFoundation/9.4/dcmsvr/sasmisc`**

Specify Your Database Connection Values

To specify the connections between your database and the common data model tables, open ci_cdm_ddl_<*db*>.sas , where <*db*> is the database that you are using. For more information, see "How to Find Your DDL Scripts for Creating the Common Data Model" on page 101. Make the changes that apply for the applicable database type as follows:

Oracle Example

```
%let path = <Oracle TNS Entry>;   /* From tnsnames.ora     */
%let user = <User Name>;          /* Oracle User/Schema    */
```

```
%let pass = <Password>;           /* Oracle Password       */
```

For example, if the user name is scott, the password is tiger, and the Oracle TNS Entry computer is cicommon.acme.com, then your %LET statements would be similar to the following statements:

```
%let path = cicommon.acme.com;
%let user = scott;
%let pass = tiger;
```

Note: For Oracle, if the common data model is in a separate database instance from the customer's subject data (for example: customer, household, account), then the tnsnames.ora file must be updated with the new connection information about the subject data files. The tnsnames.ora file is located on the SAS computer where the Oracle client is installed.

DB2 Example

```
%let user = <User Name>;          /* Other than Default User */
%let pass = <Password>;           /* DB2 Password            */
%let dsn  = <Data Source>;        /* DB2 Data Source         */
```

Greenplum Example

```
%let user = <User Name>;          /* Other than Default User */
%let pass = <Password>;           /* Greenplum Password      */
%let db  = <Database>;            /* Greenplum Database      */
%let server = <Server>;           /* Greenplum Server        */
%let port = <Port>;               /* Greenplum Port          */
%let schema = <Schema>;           /* Greenplum Schema        */
```

Teradata Example

```
%let lib    = <Target Library>;  /* For example, MAREPORT    */
%let user   = <User Name>;       /* Teradata User            */
%let pass   = <Password>;        /* Teradata Password        */
%let server = <Server>;          /* Teradata Server or TDPID */
%let database = <Database>;      /* Teradata Database        */
```

SQL Server Example

```
%let user = <User Name>;          /* Other than Default User */
%let pwd  = <Password>;           /* SQL Server Password     */
%let dsn  = <Data Source>;        /* SQL Server Data Source  */
%let schema = <Schema>;           /* SQL Server Schema       */
%let catalog = <Catalog>;         /* SQL Server Catalog      */
```

SAS Scalable Performance Data Server Example

```
%let lib     = <Target Library>;   /* For example, CICOMSPD  */
%let domain  = '<Libname Domain>'; /* For example, CICOMSPD  */
%let server  = <serverNode.Port>;  /* Name Server Network Node, Port */
%let user    = '<User>';           /* Scalable Performance Data Server User */
%let dttmfmt = <DateTime Format>;  /* For example, NLDATM21. */
%let dtfmt   = <Date Format>;      /* For example, DATE9.    */
```

Netezza Example

```
%let lib  = <library>;          /* For example, MAREPORT   */
%let user = <user>;             /* Netezza User            */
%let pwd  = <password>;         /* Netezza Password        */
%let db   = <database>;         /* Netezza Database        */
%let server = <server>;         /* Netezza Server          */
```

Setting Up the History Table

Overview: Setting Up the History Tables

The history table must be created according to subject type within the environment:

- CI_CONTACT_HISTORY records the contacts who are identified.

The following changes are required to set up this table for subject and business context. The section of the open ci_cdm_ddl_<*db*>.sas program code for the history tables appears in BEGIN HISTORY SECTION.

Note: When you run this code in order to set up a demonstration, one change to this program is required: specify the customer-specific target source connection values in the %LET statements at the beginning of the program.

Update the Placeholder SUBJECT_ID Columns

Placeholder Columns

Each of the tables in BEGIN HISTORY SECTION by default contains the primary key column, SUBJECT_ID, that is used as a placeholder column. A history table can belong to one or more subjects, as well as to one or more business contexts. You must update SUBJECT_ID with a customer-specific subject key column or columns that typically point to an external table of custom business values.

Update a Single Subject

1. SUBJECT_ID is a placeholder column for the subjects. Replace this column with the key column or columns of the subject or subjects. For example, if the subject that is referenced is Customer and if the key column is CUSTOMER_ID that is referenced as VARCHAR(10), then replace the SUBJECT_ID column with this column:

   ```
   CUSTOMER_ID VARCHAR(10) NOT NULL
   ```

   Note: If the subject has a composite key, then the multiple columns of the composite key replace the SUBJECT_ID column.

2. Replace SUBJECT_ID column in the primary key constraint statements.

Update Multiple Subjects

1. Duplicate the history table for each subject that tracks history, and assign a subject-specific name to each table. For example, if one of the multiple subjects that are referenced is Customer, then the history table might have this name:

   - CI_CUST_CONTACT_HISTORY

2. Rename the constraint names to match the new subject-specific table names.

3. Follow the same steps in"Update a Single Subject" on page 103 for the set of tables for each subject.
4. Repeat the previous steps for each subject.

Update a Single Subject in Multiple Business Contexts

A single business context requires no change other than the modifications described above for the single or multiple subject updates. If a multiple business context setup is required, you can use one of the following techniques:

- Run this entire SAS program using a different schema for each business context. Every common data model table should be duplicated for each schema.
- For each schema, duplicate the tables in the history section, once for each subject in all business contexts, and assign an identifiable business context name to each table.

If a company is divided into two separate business contexts, duplicate and name the tables in the history section according to the corresponding business context. Here are examples:

- CI_AP_CUST_CONTACT_HISTORY
- CI_NA_CUST_CONTACT_HISTORY

Note: Table names must conform to the restrictions (such as length and special characters) of the target data source. For example, the maximum length of a table name in Oracle is 30 characters.

Update Multiple Subjects in Multiple Business Contexts

Multiple subjects within multiple business contexts can also exist. Use a naming convention to easily identify the combination of subject and business context. Here are examples:

- CI_AP_CUST_CONTACT_HISTORY
- CI_AP_HHLD_CONTACT_HISTORY
- CI_NA_CUST_CONTACT_HISTORY

Customizing the Extension (EXT) Tables

Overview

These steps are required in order to customize an extension table:

1. Modify the database table definition for the extension table. This step is performed by the system DBA.
2. Identify which custom details (UDFs) are associated with which column of the extension table. This step is performed by the administrator in SAS Management Console.

Note: User-defined fields (UDFs) in the common data model are represented as custom details in SAS Decision Manager and in SAS Management Console.

Modify the Definition of an Extension Table

Overview

The extension tables contain one column by default that is a key column that is associated with a main table in the common data model. For example, the extension table, CI_CAMPAIGN_EXT, contains the single column called CAMPAIGN_SK that is a key column in the associated table, CI_CAMPAIGN.

To create the new UDF columns for reporting, add a column to the extension table. Here is an example of SAS code in which a column called CAMPAIGN_ADDITIONAL_DESC is added to the CI_CAMPAIGN_EXT extension table in an Oracle database.

Example: Modifying the Oracle Database Table Definition of an Extension Table

```
/*===================================================================*/
/* Enter Customer Specific Target Source Connection Values - Oracle  */
/*===================================================================*/

%let path = <Oracle TNS Entry> ;  /* From tnsnames.ora      */
%let user = <User Name> ;         /* Oracle User for schema */
%let pass = <Password> ;          /* Oracle Password        */

PROC SQL NOERRORSTOP;

CONNECT TO ORACLE (USER=&USER PASS=&PASS PATH=&PATH);

EXECUTE (ALTER TABLE CI_CAMPAIGN_EXT
  ADD CAMPAIGN_ADDITIONAL_DESC VARCHAR (50)) BY ORACLE;
  DISCONNECT FROM ORACLE;

QUIT;
```

Set Up the User-Defined Fields in SAS Decision Manager

Custom details, or user-defined fields, are defined in SAS Decision Manager. In order to indicate that a specific UDF (user-defined field) should be added to the extension table, you specify the name of the column that corresponds to the custom detail. This custom detail must conform to the column-naming conventions of the target data source such as SAS, Oracle, or DB2. For example, a database might disallow special characters, or might restrict the column length.

Populating the Extension Tables

The extension tables are populated automatically when a flow is published or a campaign is executed. Using the specific UDF tables as input, the system performs simple row-to-column mapping. One column in the extension tables is populated for each row that exists in the associated UDF tables.

This example code shows the values of the specific UDF table that return a value of *East*. The value *East* is also published to the CAMPAIGN_ADDITIONAL_DESC column in the extension table.

```
CAMPAIGN_SK = 1000000001
CHAR_UDF_NM = 'Sales Unit'
```

```
CHAR_UDF_SEQ_NO = 000002
CHAR_UDF_CHECKLIST_FLG = 'N'
CHAR_UDF_VAL = 'East'
CHAR_EXT_COLUMN_NM = 'CAMPAIGN_ADDITIONAL_DESC'
PROCESSED_DTTM = 01-APR-2013 00:00:00
```

If the column does not exist in the extension table, then no error is reported. However, no data is written. You can either create flow definitions that have extension column names before the tables are modified, or modify the tables before you create the flow definitions.

Setting Up the Lookup Tables

Prepopulate the Lookup Tables

After the lookup tables for the common data model are created, you next prepopulate the lookup tables with default values. To prepopulate the lookup tables and assign start-up codes, run the ci_cdm_load_codes.sas program. Find ci_cdm_load_codes.sas at the following locations:

- Under Windows: <*!SASroot*> **\ma\sasmisc**
- Under UNIX: <*!SASroot*> **/misc/ma**

For a list of lookup tables and their prepopulated values, see "About Lookup Tables" on page 93.

Resolving Translated Data

The SASMSG function has been added to each INSERT statement in ci_cdm_ddl_<*db*>.sas to properly resolve the translated data that contains characters in Unicode escape representation. The SASHELP.ci tables that are created by SMD files contain these characters. These characters are resolved by the SASMSG function when the message text is read from the table. These characters are used in the table to enable you to store data for all languages in one SAS data set. The tables are also created in an encoding that supports ASCII characters that are used in Unicode escape representation to prevent problems with data loss during transcoding. These problems can occur if the language of the locale is incompatible with the session encoding.

Registering Data Sources

Updating the Libref for the Common Data Model

Re-register the Common Data Model

An administrator registered the common data model during the installation of SAS Decision Manager. However, if you change the location of your common data model tables, then you must re-register the common data model by updating the libref that references the common data model. For details, see **SAS Intelligence Platform: Data Administration Guide**.

Specify the Common Data Model Library Location for a Business Context

To configure a business context so that it points to the library location for the common data model, specify the data options, database upload options, and reporting options on the **Settings** tab. For more information, see *SAS Decision Manager User's Guide*.

Saving, Publishing, and the Common Data Model

About Saving versus Publishing

When users publish flow data, they populate or update the tables of the common data model.

A new flow designer saves a flow after creating it. SAS Decision Manager does not automatically publish it when it is saved, because any view of the partial data (if it were updated to the common data model) would be incorrect. The Save operation, because it does not update the common data model, also uses fewer system resources than the Publish operation. You can specify an option in SAS Decision Manager to publish a flow when you save it.

After all of the components of the flow have been completed and the common data model is ready to be updated, the designer can manually publish the flow. If the **Automatically publish flows on subsequent save** option has been set for the business context, the flow data is published every time that the flow is saved after the Publish operation. When any communication within a SAS Marketing Automation campaign is executed, the system automatically publishes the campaign. In subsequent Publish operations, the common data model is updated with any changes that were made since it was last published. The common data model is updated every time that an occurrence runs.

Note: If a flow is published, these consequences might result:

- Performance is reduced during each subsequent Save operation.
- If the flow designer saves a partially completed flow, any view of the data in the common data model reflects this partial state.

Anaylysts can view the reports of the flow data from the common data model.

Full and Incremental Publish Operations

A full Publish operation occurs when you first manually publish a flow from within SAS Decision Manager. When you first manually publish a flow, all publishable data is sent to the common data model. Subsequent requests to publish the campaign from within SAS Decision Manager result in an incremental Publish operation.

An incremental Publish operation updates only those fields in the common data model that have changed or have never been published. An incremental Publish operation occurs under these conditions:

- You request that a flow be published from SAS Decision Manager and the flow has been previously published.
- An execution of a SAS Marketing Automation campaign (or a communication within the campaign) occurs.

- A previously published flow is saved and the **Automatically publish flows on subsequent saves** option is selected for the business context.

Chapter 9

Designing SAS Data Sets

Design SAS Data Sets for Test Cases in SAS Decision Manager 109

Column Names in Event Variables 110

Design SAS Data Sets for Test Cases in SAS Decision Manager

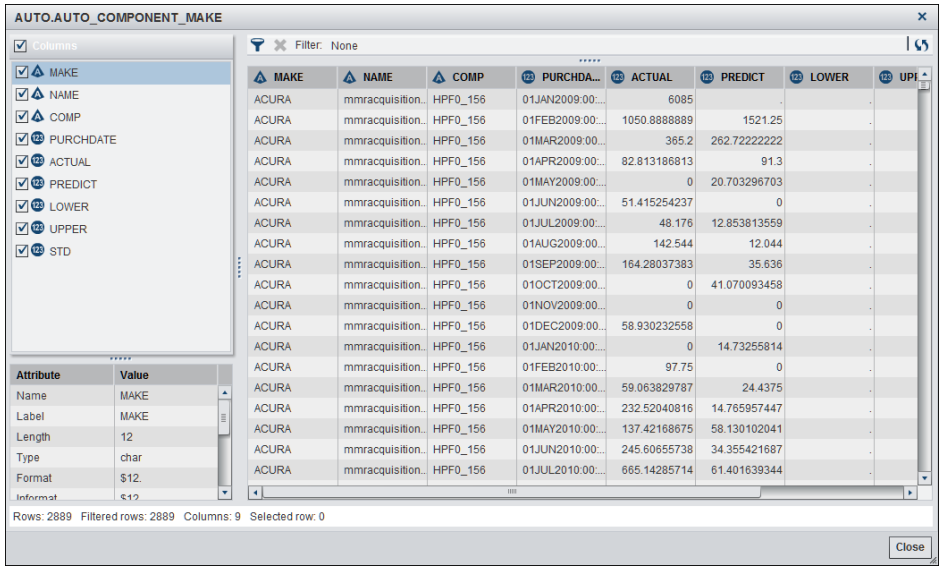
You can perform comprehensive testing of SAS Decision Manager flows by using test cases that are based on historical data from other systems or files.

To use test cases that are based on historical data, create a SAS data set that consists of test cases. The user imports the test cases in SAS Decision Manager by selecting the library name and the appropriate table.

A single row in the SAS data set corresponds to a test case. The row contains the initial values as well as the expected values.

The first column in the table must be named **`_Test_Name`**.

Display 9.1 *Test Case Table Example*



For the columns in the historical data, use the following SAS types:

String
Character type

Integer
Numeric type

Double
Numeric type

Boolean
Character type. The string **true** results in a value of **`TRUE`**. Any other string results in a value of **`FALSE`**.

Date
Numeric type with SAS DATE format.

Data grid
Character type, using either of the following:

- LIBNAME.TABLENAME
- TABLENAME, where LIBNAME is assumed to be same as the library to which the data set belongs.

String List
Character type, with each string delimited by a semicolon (;).

Integer List
Character type, with each string delimited by a semicolon (;).

Double List
Character type, with each string delimited by a semicolon (;).

Boolean List
Character type, with each string delimited by a semicolon (;).

Date List
Character type, with each string delimited by a semicolon (;). Use one of the following date formats:

- Short date format in the form *mm/dd/yy*
- Long date format (for example **January 22, 2013**).

The following rules apply when matching variable names to column names. Variables that are not found during the import process are ignored.

- Reply variable and treatment variable names begin with a double underscore (__).
- In the case of custom details and custom detail tags, use variable names for the column labels. If more than one custom detail has the same name, only one is imported.

Column Names in Event Variables

When you import event variables into SAS Real-Time Decision Manager, the column names in the imported table must match the localized event column names in the user interface. For example, in the English locale, the imported event variable table must have the following column names.

- Variable Name
- Display Name
- Description
- Type
- Identifier
- Level
- Required

Chapter 10

The Launcher

Overview of the Launcher **113**

Enable Logging for the Launcher **113**
Overview of Logging 113
Locate sasdcmlauncher.ini 114
Add Command-Line Arguments (sasdcmlauncher.ini) 114

Launcher Command Arguments **115**
Launcher Command Syntax 115
Example of a System-Generated Launcher Command 119
Examples of Generating Information Map Metadata Tables 120
Example of Listing Input Variables 120

Error Codes **120**
Troubleshooting the Most Common Launcher Errors 120
List of Error Codes 121

Overview of the Launcher

The Launcher is a small utility program (sasdcmlauncher.exe) whose purpose is to pass command-line arguments in a format that is understood by SAS Decision Manager. The Launcher can be used to generate reports of input variables and other actions. Input variables are references to request variables in a flow.

Note: You can also run the Launcher either manually or from some other third-party application. In the latter case, you identify the flow by name rather than by its identifier.

Enable Logging for the Launcher

Overview of Logging

The Launcher supports two types of logging:

General logging
: logs each execution of the Launcher. Logging information can be written to a different file for each execution or it can be appended to an existing file.

Command logging
: logs a particular execution of the Launcher. Command logging enables you to create logs for specific flows. The following considerations apply to command logging:

  - Command logging runs in Append mode by default.
  - Command logging can be used only if general logging is enabled.
  - Command log files are written to the same directory as general log files. If a directory other than the default has been specified for general logging, then command logs are also written to the specified directory.
  - Command log filenames are created based on the type of operation, the object name or ID, and the last directory name in the path.

The sasdcmlauncher.ini file is the Launcher initialization file that contains start-up and configuration parameters for the Launcher. To configure the Launcher's logging parameters, including log filenames and directories:

1. Locate the sasdcmlauncher.ini file.
2. Open sasdcmlauncher.ini. You will see Java code similar to the sample file excerpt that is displayed.
3. Add command lines that specify the logging parameters that you want to change. Add each new command argument at the end of the file.

Locate sasdcmlauncher.ini

sasdcmlauncher.ini is located by default in the SAS Decision Manager installation directory:

- UNIX: **.../SASDecisionManagerLauncher/6.2**
- Windows: <*Drive*>**:\Program Files\SASHome\SASDecisionManagerLauncher\6.2**

Add Command-Line Arguments (sasdcmlauncher.ini)

Use these Java Runtime Environment (JRE) commands within sasdcmlauncher.ini to perform the corresponding task. No spaces are allowed in the Java argument unless the path string is enclosed in quotation marks.

To direct general logging information to a log file without specifying a filename:
: **-Dma.launcher.log** (for example, **JavaArgs_7=-Dma.launcher.log**)

  A separate log file is created in the **sasdcmlauncher\6.2** directory for each Launcher call that is made. The filename is assigned as the time expressed in milliseconds (for example, 1268841617289.log).

To create a general log file with a specified filename:
: **-Dma.launcher.log=<filename>** (for example, **JavaArgs_8=-Dma.launcher.log=february.log**)

  If the file does not exist, then it is created. You must manually specify the .log extension in the filename if you want the extension to be used.

To append the general log output to an existing file:
: **-Dma.launcher.log=<filename>** (for example, **JavaArgs_9=-Dma.launcher.log=january.log**)

You must manually specify the .log extension in the filename when you create the file if you want the file extension to be used in the filename.

To direct all logging information to a directory within the home directory of the Launcher application:
-Dma.launcher.logdir="<dirname>" (for example, **JavaArgs_10=-Dma.launcher.logdir="logs"**)

The directory must exist. Enclose a path in double quotation marks.

To direct all logging information to an absolute directory:
-Dma.launcher.abslogdir="<AbsDirectory>" (for example, **JavaArgs_11=-Dma.launcher.abslogdir="\\srshq\root\logs\launcher"**)

Another example is: **JavaArgs_11=-Dma.launcher.abslogdir="C:\deptZed\logs\launcher"**

The directory must exist. In addition to the *sassrv* user who must have WRITE permission, the user who might manually start the Launcher must also have WRITE permission to the directory.

To enable command logging in Append mode:
-Dma.launcher.commandlogmode= (for example, **JavaArgs_12=-Dma.launcher.commandlogmode=**)

To enable command logging in Replace mode:
-Dma.launcher.commandlogmode=replace (for example, **JavaArgs_12=-Dma.launcher.commandlogmode=replace**)

To enable the decoding of U+ Unicode strings
JavaArgs_xx=-Dma.unicode.marker=U+

xx is the next number in the list of Java arguments.

Launcher Command Arguments

Launcher Command Syntax

The Launcher command uses the following syntax:

```
sasdcmlauncher.exe -d directive -u userid -p password
 -x business context[-n -q -v] [--] directive arguments
```

The command takes the following options:

-d camp | **file** | **imap** |**mark** | **test** | **tests**
specifies a directive

-u *userID*
specifies the ID of the user who is responsible for executing this command.

-p *password*
specifies the password for the user ID. The password can be encrypted or clear text.

–g
assigns a user ID to identify the flow.

-x
specifies the name of business context that corresponds to the flow. Spaces are allowed; the business context name is not case-sensitive.

-v
specifies informative messages in the log. If not specified, only errors are displayed.

-n
specifies that arguments are given by using the path and name rather than by identifier. The path and name must be provided in URI (Universal Resource Indicator) format unless option **-q** is specified.

-q
specifies that arguments are either quoted or have no spaces. URI translation of arguments is suppressed when this option is specified.

--
specifies the end of the options. All parameters after this are directive arguments.

directive arguments
is a list of arguments, separated by spaces. See Table 10.3 on page 118, Table 10.4 on page 119, and Table 10.5 on page 119 for the directives that are associated with each argument. These are the possible arguments:

assetType
type of asset to mark for deployment. The value is **camp**.

business context ID
uses the settings of the specified business context to regenerate metadata. Any saved settings are overridden by the directive arguments.

flowID
flow ID

path
folder path and flow name

clearCache
information map cache to be cleared after map is generated

file path
the path to the designated file

file op
the file operation. The values are **delete** or **write**.

file contents
holds the file contents to be written

genType
describes the method used. **1** is SQL. **2** is PROC SUMMARY.

imap absolute path
specifies the absolute path of the information map (for example, **/DM Assets/Information Maps/DMBusinessContext**)

For a list of arguments to *imap absolute path*, see Table 10.1 on page 117.

imap SBIP URL
specifies the SBIP URL of the information map. The SBIP URL uses the proprietary SAS Business Intelligence Platform (SBIP) protocol.

For a list of arguments to *imap SBIP URL*, see Table 10.1 on page 117.

imap URL
information map URL

report name
folder and filename of input variables report

testId
test case ID

testName
name of the test case in the flow

testResultFile
test result log file path

testSaveMode
indicates whether a flow is to be saved again after the test runs. The values are **`save`** or **`nosave`** (default).

testType
indicates whether a flow is to be tested. The value is **`camp`**.

typeArgs
space-delineated list of table-type arguments

The *flowID* argument is an identifier that is assigned by the SAS Decision Manager application server. It is not intended to be entered by the user on the command line, but it can be used if the **`-n`** option is not used.

When specified with *imap absolute path* or *imap SBIP URL*, **`imap`** takes arguments in the following table. The order is important for the **`-method`**, **`-character`**, **`-numeric`**, **`-histogram`**, **`-datehistogram`**, and **`-univariate`** options. The last entry on the command line takes precedence.

Table 10.1 *Arguments with imap absolute path and imap SBIP URL*

Argument	Parameter	Description
`-method`	**`SQL`** | **`Summary`**	use SQL or the SAS SUMMARY procedure as the query method for all tables
`-character`	**`SQL`** | **`Summary`**	use SQL or the SAS SUMMARY procedure as the query method for character variable tables
`-numeric`	**`SQL`** | **`Summary`**	use SQL or the SAS SUMMARY procedure as the query method for numeric variable tables
`-histogram`	**`SQL`** | **`Summary`**	use SQL or the SAS SUMMARY procedure as the query method for histogram tables

-datehistogram	**SQL** | **Summary**	use SQL or the SAS SUMMARY procedure as the query method for date histogram tables
-univariate	univariate options. See Table 10.2 on page 118.	override information map settings and generate univariate metadata
-mameta	*library name*	override the setting for the MAMeta library
–maxconcurrent	*whole number*	override the **Maximum number of concurrent processes per metadata generation** setting in environment variables

The following table lists the options that can be used with the **-univariate** argument.

Table 10.2 *Options for -univariate Argument*

Option	Description
Default	let the application determine the best way to generate metadata
InDatabase	force MIN/MAX/MEAN generation, even for columns that might fail with numeric overrun errors
Summary	use PROC SUMMARY to calculate the MIN/MAX/MEAN. The column is brought into a SAS session to perform the calculation.
AVGMinMax	use the formula (MIN+MAX)/2 to calculate the mean.

In the following table, the **-n** option has not been specified.

Table 10.3 *Arguments When the -n Option Has Not Been Specified*

Directive	Arguments	Description
imap	*imap absolute path* | *fimap SBIP URL* | *business context ID* **[-method] [-character]** [**-numeric**] [**-histogram**] [**-datehistogram**] [**-univariate**] [**-table**] [**-dataitem**] [**-mameta**] [**-maxconcurrent**]	generates information map metadata tables, using the specified arguments.

test	*testType flowID testID* [*testSaveMode*] [*testResultFile*]	runs a test case, saves the floe, and logs the test results
tests	*testType flowID* [*testSaveMode*] [*testResultFile*]	runs all test cases, saves the flow, and logs the test results

You can enter the *path* argument on the command line if the **-n** option has been specified. If an application server that is not the standard is used, the application server also uses this argument with the **-n** option.

In the following table, the **-n** option has been specified. All objects are identified by name or folder and name.

Table 10.4 *Arguments When the -n Option Has Been Specified*

Directive	Arguments	Description
inputvariables	*report name*	generates a list of input variables in PDF format
mark	*assetType file path*	marks an asset for deployment
test	*testType flowname test name* [*testSaveMode*] [*testResultFile*]	runs a test case, saves the flow, and logs the test results
tests	*testType flowpath* [*testSaveMode*] [*testResultFile*]	runs all test cases, saves the flow, and logs the test results

The **-n** option does not affect the **file** directive.

Table 10.5 *Arguments for the File Directive*

Directive	Arguments	Description
file	*file path* delete	deletes the file
file	*file path* write *file contents*	writes the file

Example of a System-Generated Launcher Command

Here is an example of a system-generated Launcher command:

```
c:\program files\SASHome\SASDecisionManagerLauncher\6.2\sasdcmlauncher.exe -u
sasadm -p "my pass" -x "Functional Small Teradata" -v
```

The Launcher is started by user sasadm with a non-encoded password of **my pass** in the business context **Functional Small Teradata**.

Verbose logging is specified.

Examples of Generating Information Map Metadata Tables

Here is an example of using the **imap** directive to generate information map metadata tables.

```
sasdcmlauncher.exe -d imap mybusinesscontextid
```

Information map metadata tables are generated for a business context named "mybusinesscontextid." This command has the same result as clicking **Run Now** on the **Metadata** tab of the Business Context Properties window.

Here is an example of using the **imap** directive and SQL and the SAS SUMMARY procedure to generate metadata.

```
sasdcmlauncher.exe -d imap /DM Assets/Information Maps/Mybusinesscontext
-method Summary -character SQL
```

Except for character metadata, the SUMMARY procedure is used to generate metadata in the library that is defined for the information map. Character metadata is generated by SQL and is stored in the same library.

Example of Listing Input Variables

You can generate a single list of input variables for all of the flows in a business context. You can quickly review the list to determine whether a specific variable is used. The list contains the node name or treatment where the variable is used, and the type of usage, whether variable, identifier, or calculated item.

The following command writes input variables to a file named report.pdf.

```
sasdcmlauncher.exe -u myuserid -p mypassword -x mybusinesscontext -n -q -v
-d inputvariables c:/public/report.pdf
```

In order to generate a complete report, you should have access to all of the flows and events. Otherwise, the generated report will be incomplete.

Error Codes

Troubleshooting the Most Common Launcher Errors

These are the three most common error codes that might be generated by the Launcher:

- Unexpected Error
- Execution Failed
- Unable to Log onto Application Server

Unexpected Error, **Execution Failed**
First, check for errors in the logs of the SAS Decision Manager engine (on the application server) and the SAS Remote Services server.

Next, start the Launcher from a command prompt on the computer where the Launcher is installed, and determine whether the same error or a different error occurs.

Unable to Log onto Application Server
When the Launcher cannot log on to the application server, it might be caused by the following:

- The application server (SAS Decision Manager engine) is not running.
- The password of the user has changed.

List of Error Codes

These are the error codes that can be returned by the Launcher. If a remedy or strategy for handling an error is available, then it is included in the error description.

01 Unexpected error

10 Invalid command line
Compare the generated Launcher command with the valid Launcher command-line arguments.

11 Invalid directive
The value for the directive in the **-d** option is not allowed.

12 Invalid argument scheme
The list of arguments either does not correspond to the directive or is not valid for the directive that is specified by the **-d** option.

13 Ambiguous argument scheme
The list of arguments is ambiguous for the selected directive that is specified with the **-d** option. This error occurs when a conflict exists because either the **-n** option is used to name arguments that are not provided in the correct format, or the named arguments do not correspond to the selected directive.

20 Unable to log on to application server
See "Unable to Log onto Application Server" on page 121.

21 Unable to acquire execution service
The Launcher cannot connect to the execution service within the application server. The Launcher uses an EJB (Enterprise JavaBeans) execution within the application server.

23 Unable to switch to context

24 Unable to acquire Security Manager

30 Execution failed
The Launcher command failed to execute. In UNIX operating environments, this error indicates that the user does not have permission to create and update information map metadata tables.

32 Object to execute is locked by another user
The object to be executed is already opened by another client. The Locks category in the Administration workspace of SAS Decision Manager enables you to view and release objects that are locked, such as flows.

34 Identifier resolve failure
It was not possible to resolve the specified identifier. Either the identifier no longer exists, or the specified identifier is incorrect. In extreme circumstances, this error is

displayed if the specified flow name has been incorrectly resolved from corrupted metadata.

40 Flow failed to generate correctly in running test case

41 Test case failed

42 Test case not found

45 Metadata server out of memory
The metadata server is out of memory.

46 Flow is locked
In some circumstances, you might not be able to edit an object such as a flow or treatment, even if you have Edit permission. To unlock an object so that you can edit it, select the Locks category in the Administration workspace in SAS Decision Manager. Contact the holder of the lock before you release the object.

50 Flow name not found
The specified flow name and path could not be found. Either the flow no longer exists, the specified name is incorrect, or the path is incorrect.

60 File does not exist

61 File cannot be deleted

62 File cannot be written

70 Flow cannot be marked for deployment
The flow cannot be marked for deployment because it has not been approved.

90 Problem while logging off the application server
This error occurs after the object is successfully executed by the application server, but before the Launcher is disconnected. Look for error messages in the log file of the application server.

91 Problem releasing execution service
The Launcher cannot release the execution service within the application server. The Launcher uses an execution EJB (Enterprise JavaBeans) within the application server.

301, 302 Nonzero error
If you try to run a batch job and receive this error, then the flow might be open. Close the flow and retry the batch job.

Chapter 11

Backing Up and Restoring Data

The Difference between Backing Up Data and Archiving Data **123**

General Strategy for Backing Up Data **123**
Steps for Backing Up Data 123
Synchronizing the Backups for Flow Environment 124
Backing Up the SAS Metadata Server 124
Backing Up the SAS Content Server 125
Backing Up the MATables and MAMisc Libraries 125

Backing Up Files Before Installing a New Version **125**

The Difference between Backing Up Data and Archiving Data

The primary difference between the tasks of backing up data and archiving data is in their objectives. Backups are necessary for the operational recovery of IT business systems during a disaster (for example, a flood, lightning strike, or hard drive crash). Backups are typically stored for several weeks. A daily backup cycle can protect all types of business data, including structured and unstructured data.

Archives (archived data) are created in order to comply with legislation and good corporate compliance practices. Archived data is typically kept for a number of years. To create an archive, manual processes are created to take a backup copy from its normal backup cycle and store it in a fireproof safe for long-term storage. The backup media represents a snapshot of the business at that point in time. An organization might use archived data to protect structured business data (for example, business transactions or legal contracts).

General Strategy for Backing Up Data

Steps for Backing Up Data

As part of the maintenance of your SAS Business Intelligence Platform, you should perform the following steps:

- Back up your SAS metadata repository either nightly or weekly. SAS Decision Manager makes extensive use of metadata that is stored on the SAS Metadata Server

and physical content that is stored on the SAS Content Server. Therefore, it is critical to perform regular backups of these servers.

- Back up the physical tables in the MATables and MAMisc libraries either nightly or weekly.
- Keep separate XML backup files for each of the information maps and flow. Store the XML backup files on a separate file system in order to recover data that might be lost because of a hardware malfunction such as a hard drive crash.

  If one or more objects become corrupted, then you can re-import the XML backup file for a specific information map or flow. In this way, you avoid restoring the entire SAS Metadata Server and SAS Content Server.

Synchronizing the Backups for Flow Environment

When you perform backups of the SAS Metadata Server, the SAS Content Server, and the MATables and MAMisc libraries, the backups must be synchronized in order to correctly restore the Flow environment.

In order for the backups to be synchronized, the metadata server must be paused during all of these backups. Pausing the server (and setting it to an Offline state) ensures that no Flow activity can occur between the creation of the backups.

Perform the backups in the following order:

1. Pause the SAS Metadata Server, and set it to an Offline state.
2. Back up either the SAS Content Server or the SAS Metadata Server.
3. Back up the remaining server (either the SAS Metadata Server or the SAS Content Server).
4. Back up the physical tables in the MATables and MAMisc libraries.
5. Resume the SAS Metadata Server.

For details about synchronizing your backups, as well as additional guidance for making backups, see *SAS Intelligence Platform: System Administration Guide*. The guide is available at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Backing Up the SAS Metadata Server

A correct backup of the SAS Metadata Server includes the foundation repository, all custom and project repositories, the metadata journal file, and the metadata server's configuration files. SAS provides a macro called %OMABAKUP that pauses the metadata server and creates backups of these files while minimizing disruptions in service. You can use any of the following methods to run %OMABAKUP:

- Run the sasBackup.sas program that is provided with your initial software installation. This program runs %OMABAKUP and stores the backup files in a directory on the metadata server. If necessary, you can modify the script to direct the backup to a different location.
- Use the Backup Wizard in SAS Management Console. This wizard enables you to create and run %OMABAKUP without writing code. You can specify a backup location and select additional backup options.
- Write a custom program that uses %OMABAKUP.

For details about each of these methods, see *SAS Intelligence Platform: System Administration Guide*, which is available at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Note: You can use operating system commands to back up your metadata repositories and the repository manager. If that is the case, pause the metadata server by setting it to an Offline state before performing the backup. If the metadata server is Online, or is paused and set to Administration state, then the backup files are not usable.

Backing Up the SAS Content Server

To back up the SAS Content Server, follow these steps:

1. Make sure that the SAS Metadata Server has been paused and set to an Offline state.
2. Stop either the web application server or the SAS Content Server application.
3. Use operating system commands or third-party tools to copy all of the files and subdirectories from the following path:

   SAS-configuration-directory\Lev1\AppData\SASContentServer\Repository

Backing Up the MATables and MAMisc Libraries

Be sure to back up the physical tables that are stored in the MATables and MAMisc libraries. To make the backups, use operating system commands or third-party tools.

This is the default location of the MATables library:<*config*>**/Lev1/Applications/SASDecisionManager6.2/data/MATables**.

This is the location of the MAMisc library: <*config*>**/Lev1/Applications/SASDecisionManager6.2/data/MAMisc**.

If the tables are not located in these paths in your deployment, you can determine the correct paths by opening the DecisionManager_autoexec.sas file, which is located in ***SAS-configuration-directory*\Lev1\AppData\SASContentServer\Repository**. In the file, look for statements with the following syntax: **libame MAMisc** <*path*> or **libame MATables** <*path*>.

If the directory also contains a file named autoexec_usermods.sas, be sure to look for LIBNAME statements in that file as well. These statements override statements in DecisionManager_autoexec.sas.

Backing Up Files Before Installing a New Version

Before you install a new version of SAS Decision Manager, back up the following files in the <*!sasroot*>**\SASFoundation\9.4\ma\sasmacro** directory:

- mausrexp.sas
- mausrupl.sas

Restore these files after installing the new version.

Chapter 12

Improving Performance

Activating the Bulk Load Facility **127**
How the Bulk Load Facility Improves Performance 127
Enable the Bulk Load Facility 128
Example: Enabling the Bulk Load Facility 128

Setting Data Set Options When a Bulk Load Facility Is Not Available **129**
Specify Data Set Options without Bulk Loading 129
Examples: Data Set Options without Bulk Loading 130

Specifying JVM Memory Size **130**
Overview of Specifying JVM Memory Size 130
Specify Command-line Options in the Remote Services File 130

Troubleshooting JVM Error Messages **132**

Activating the Bulk Load Facility

How the Bulk Load Facility Improves Performance

One of the fastest ways to insert large volumes of data into a relational database is to use the bulk loading capabilities of the database. When you use SAS/ACCESS, you specify the BULKLOAD data set option to use the bulk loading capabilities of database management systems (DBMS) such as Oracle. The bulk loading method significantly enhances performance, especially when database tables are indexed.

When the BULKLOAD option is enabled, SAS calls the appropriate bulk load facility for each DBMS.

- For Oracle: SQL*Loader

The bulk loading capability improves performance in the following areas:

- When you store contact history tables in a DBMS, performance might be reduced when you update a contact history table. The bulk load facility improves performance by obtaining the list of IDs from SAS. The facility passes the IDs to temporary tables in the database. The facility then runs a query or updates the contact history tables in that database.

The performance of the bulk load facility can be affected by the connectivity between SAS Decision Manager and the database server, and by the size of the cell that is being loaded. For complex flows, the disk configuration of the database server might reduce

performance. However, if only a single bulk loading process is started for one flow, then the disk configuration should not significantly affect performance.

For details about SAS/ACCESS, see the *SAS Intelligence Platform: Data Administration Guide* at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Enable the Bulk Load Facility

Specify the following options in the Options section of the **Settings** tab in the Business Context Properties window. Example entries for each database are provided in "Example: Enabling the Bulk Load Facility" on page 128.

1. **Data set options** specifies options that are used in the SAS DATA step to create the temporary table. To enable the bulk loading capabilities of the database, specify *BULKLOAD=YES*. Additional options might also be appropriate, depending on which database you are using. For recommended options, see"Example: Enabling the Bulk Load Facility" on page 128. If a bulk load facility is not available for your database, or if you choose not to use the bulk load facility, see "Setting Data Set Options When a Bulk Load Facility Is Not Available" on page 129.
2. **Schema** specifies the location for the temporary tables. The following options are valid.
   - Specify your CICOMMON schema if you want the temporary tables to be created in the same schema that stores contact history.
   - Specify another schema to use for the temporary tables.
   - Leave the field blank if you want the temporary tables to be created in the user's default schema.
   - Users must have both Read and Write access to the specified schema, including the following:
     - the ability to create and delete tables and indexes
     - the ability to delete, insert, and update records
3. **Use temporary table capability of database** specifies whether the database's native temporary table capability is to be used to create temporary tables. The check box is deselected by default. If you are enabling the bulk load facility of your database, do not select this check box. If you specify both *BULKLOAD=YES* and **Use temporary table capability of database**, processing fails.

Example: Enabling the Bulk Load Facility

Overview

The following examples provide recommended values for enabling the BULKLOAD option for your database. Enter the data set options, schema, and **Use temporary table capability of database** options on the **Options** tab of the Business Context Properties window, as described in "Enable the Bulk Load Facility" on page 128. For information about enabling the bulk load facility of other databases, visit **http://support.sas.com/documentation/index.html** and search for the keywords "bulk load facility."

*Oracle and SQL*Loader*

The bulk load facility for Oracle is called SQL*Loader. Temporary tables are stored in the specified schema.

Setting Data Set Options When a Bulk Load Facility Is Not Available

For best performance, use the bulk load facility. If a bulk load facility is not available for your database, or if you choose not to use the bulk load facility, then use the following instructions to specify data set options.

Specify Data Set Options without Bulk Loading

If a bulk load facility is not available for your database, or if you choose not to use the bulk load facility, specify the following options in the Options section of the **Settings** tab in the Business Context Properties window. Example entries are provided in "Examples: Data Set Options without Bulk Loading" on page 130.

1. **Data set options** specifies options that are used in the SAS DATA step to create the temporary table. To improve performance, specify the INSERTBUFF option.
2. **Schema** specifies the location for the temporary tables. The following options are valid.
   - If you want the temporary tables to be created in the user's default schema, leave the field blank.
   - If you are selecting the **Use temporary table capability of database** option, which uses the native temporary table capability of the database to create temporary tables, leave the **Schema** field blank. The **Schema** field is ignored if **Use temporary table capability of database** is selected.
   - Specify your CICOMMON schema if you want the temporary tables to be created in the same schema that stores contact history.
   - Specify another schema to use for the temporary tables.

   Users must have both Read and Write access to the specified schema, including the following:
   - the ability to create and delete tables and indexes
   - the ability to delete, insert, and update records
3. **Use temporary table capability of database** specifies whether the database's native temporary table capability is to be used to create temporary tables. The check box is deselected by default.

   You should select this check box only if your site does not authorize users to create any additional temporary tables. When you select the check box, remember the following.
   - The value in **Schema** is ignored, and the location of temporary tables is determined by the database.
   - Do not specify **`BULKLOAD=YES`** in the **Data set options** field. If you specify both **`BULKLOAD=YES`** and **Use temporary table capability of database**, processing fails.

Examples: Data Set Options without Bulk Loading

Overview

The following examples show data set options that are recommended when a bulk load facility is not available. Enter these values on the **Options** tab of the Business Context Properties dialog box, as described in "Setting Data Set Options When a Bulk Load Facility Is Not Available" on page 129.

Example for a Relational Database for Which a Bulk Load Facility Is Not Available

Because no schema is specified in this example, the user's default schema is used.

- Data set options: INSERTBUFF=1000
- Schema: (None)
- Use temporary table capability of database: Not selected

Example for a Relational Database When Users Are Not Authorized to Create Additional Tables

- Data set options: INSERTBUFF=1000
- Schema: (None)
- Use temporary table capability of database: Selected

Example for SPD Server

To store temporary tables in a library that has been defined on the SPD Server (for example, in the libnames.parm file), set the data set options as follows:

- Data set options: (None)
- Schema: <*schema-name*>
- Use temporary table capability of database: Selected

To use a temporary schema, the user must have permission to create a temporary folder under the CICDM domain.

Specifying JVM Memory Size

Overview of Specifying JVM Memory Size

You must increase the size of JVM memory when you use large SAS Information Maps. Information maps are user-friendly metadata definitions of physical data sources that enable your business users to query a data warehouse in order to meet specific business needs. Large information maps can contain more than 25 million rows of data, which might require significant memory for loading and processing.

Specify Command-line Options in the Remote Services File

1. Access the appropriate remote services file:

UNIX location:**Lev1/Web/Applications/RemoteServices/RemoteServices.sh**

Windows location: **Lev1\Web\Applications\RemoteServices\RemoteServices.bat**

2. Include the command-line option in the remote services file, as appropriate:

   UNIX command-line option:

```
SERVICES_OPTS="$SERVICES_OPTS -Xms1248m -Xmx1248m
 -DentityExpansionLimit=10000000"
```

   Windows command-line options:

```
set SERVICES_OPTS=%SERVICES_OPTS%
 -Xms1248m -Xmx1248m -DentityExpansionLimit=10000000
```

   Here is an explanation for each option in the command line:

   -Xms1248m -Xmx1248m
   are the initial and maximum heap sizes in megabytes. Increasing these values reduces out-of-memory errors.

   The value for the initial memory size must be equal to the value that is specified for the maximum memory capacity.

   Adjustments to memory size are constrained by the computer's actual memory size. The memory size that you specify should reflect your configuration.

   -DentityExpansionLimit
   is a system property that permits existing applications to increase the expansion limit on the total number of entity expansions that can be supported without having to recompile the code. Increasing entity expansions accommodates large information maps and reduces denials of requests for service.

   An error message is displayed if the specified size is insufficient. For example, this message reports that the value 64,000 is insufficient:

   Parser has reached the entity expansion limit "64,000" set by the Application

   To recover from this error, you must increase the expansion size.

   The recommended starting value is 10,000,000.

   Here is a sample StartRemoteServices.bat file in Windows:

```
@echo off
set JAVA_PATH=java
if not "%JAVA_HOME%" == "" (
if exist "%JAVA_HOME%\bin" (
set JAVA_PATH=%JAVA_HOME%\bin\java
)
)
set CLASSPATH=
REM Dynamically add all JARs in the WEB-INF\lib
directory to the classpath.
for %%i in (".\lib\*.jar") do call ".\cpappend.bat" %%i
set SERVICES_DIR=C:\SAS\SASConfig\Lev1\web
  \Deployments\RemoteServices
```

```
set SERVICES_OPTS=-Xms1248m -Xmx1248m -DentityExpansionLimit=10000000
set SERVICES_OPTS=%SERVICES_OPTS% -Djava.security.manager

set SERVICES_OPTS=%SERVICES_OPTS%
   -Djava.security.policy="%SERVICES_DIR%\
     sas.wik.allpermissions.sasservices.policy"
   %JAVA_PATH% %SERVICES_OPTS% com.sas.services.webapp.Bootstrap
   conf\sas_metadata_source_server.properties
```

Troubleshooting JVM Error Messages

Here are typical error messages.

- **Exception java.lang.OutOfMemoryError: requested** *size* **bytes**

  JVM cannot expand its heap size if memory is completely allocated, and if swap space is not available.

  To recover from the error, perform the following actions:

  - Increase the available swap space by allocating more of the disk for virtual memory.
  - Limit the number of applications that run simultaneously.
  - Ensure that the values for the minimum and maximum heap sizes **-Xms***n* and **-Xmx***n* command-line options are identical in the remote services file (StartRemoteServices.sh for UNIX and StartRemoteServices.bat for Windows).

  Note: Increasing the heap size to values larger than the recommended values does not fix the problem. For recommended values, see "Specify Command-line Options in the Remote Services File" on page 130.

- **Parser has reached the entity expansion limit "64,000" set by the Application**

  This error occurs when metadata is being generated or when SAS Decision Manager is retrieving metadata.

  To recover from the error, increase the entity expansion limit in the remote services file. A starting value of 10,000,000 is recommended for the **-DentityExpansionLimit** command-line option.

Chapter 13

Macros for Importing and Exporting Business Rules Data

Introduction to the Import and Export Macros **133**

Dictionary **134**
%BRM_CREATE_TEMP_TERM 134
%BRM_EXPORT_FOLDER 135
%BRM_EXPORT_LOOKUP 136
%BRM_EXPORT_RULE_FLOW 137
%BRM_EXPORT_RULESET 137
%BRM_EXPORT_VOCABULARY 138
%BRM_IMPORT_FOLDER 139
%BRM_IMPORT_LOOKUP 140
%BRM_IMPORT_MARKET_BASKETS 142
%BRM_IMPORT_RULE_FLOW 149
%BRM_IMPORT_RULESET 151
%BRM_IMPORT_VOCABULARY 154
%BRM_LOAD_VOCABULARY 156

Introduction to the Import and Export Macros

SAS Decision Manager provides the following macros for importing data into the rules database and exporting data from the rules database. These macros must be run on the server tier.

%BRM_CREATE_TEMP_TERM
: reads a CSV file or a SAS data set and produces a SAS data set named WORK.TERM that can be used as input to the %BRM_LOAD_VOCABULARY macro.

%BRM_EXPORT_FOLDER
: exports definitions of business rules folders into a CSV file.

%BRM_EXPORT_LOOKUP
: exports the contents of lookup tables into a CSV file.

%BRM_EXPORT_RULE_FLOW
: exports rule flows from the rules database into a CSV file

%BRM_EXPORT_RULESET
: exports rule sets from the rules database into a CSV file

%BRM_EXPORT_VOCABULARY
: exports vocabularies from the rules database into a CSV file

%BRM_IMPORT_MARKET_BASKET
imports association rules that are generated by SAS Enterprise Miner and creates a rule flow that deploys the association rules.

%BRM_IMPORT_FOLDER
imports the folder definitions that are in the specified CSV file into the rules database.

%BRM_IMPORT_LOOKUP
imports lookup tables from the specified CSV file into the rules database.

%BRM_IMPORT_RULE_FLOW
imports rule flows from a CSV file into the rules database

%BRM_IMPORT_RULESET
imports rule sets from a CSV file into the rules database

%BRM_IMPORT_VOCABULARY
imports vocabulary terms from a CSV file into the rules database

%BRM_LOAD_VOCABULARY
loads the vocabulary terms into the WORK.TERM data set that was created by the %BRM_CREATE_TEMP_TERM macro.

Dictionary

%BRM_CREATE_TEMP_TERM

Reads a CSV file or a SAS data set and produces a SAS data set named WORK.TERM that can be used as input to the %BRM_LOAD_VOCABULARY macro.

Restriction: This macro must be run on the server tier.

Syntax

%BRM_CREATE_TEMP_TERM (DATAFILE=*input_file*<, BRM_USER=*user_ID*>);

Required Argument

DATAFILE=*input_file*
specifies either a SAS data set name or the full pathname to a CSV file. If the input file is a CSV file, the first row of the file must contain valid SAS column names, and the remaining rows must contain column values. The column values can be numeric or character data only. You cannot specify SAS informats in the column data. The column names must be unique. For example, a simple CSV file that specifies two columns, both with numeric data, might look like the following:

```
patientID,BloodPressure
1,140
2,141
3,142
```

Optional Argument

BRM_USER=*user_ID*
specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the rules database and is displayed in the SAS Decision Manager interface.

Default User ID of the user that is running the macro

Details

This macro reads a CSV file or SAS data set that defines vocabulary terms and that creates a SAS data set named WORK.TERM. You can use the WORK.TERM data set as input to the %BRM_LOAD_VOCABULARY macro. The %BRM_LOAD_VOCABULARY macro loads the vocabulary terms into the rules database. See "%BRM_LOAD_VOCABULARY" on page 156 for more information.

The %BRM_CREATE_TEMP_TERM macro derives domain types and domain values for the vocabulary terms based on the data type of the term as described in Table 13.1.

Table 13.1 *Domain Types and Values for Input Terms*

Term Data Type	Derived Domain Type	Derived Domain Values
String	Discrete	If there are ten or fewer distinct values in the input data, all of the values are included in the list of domain values. If there are greater than ten distinct values in the input data, individual values are not listed in the domain values.
Date	Continuous	No input values are included in the list of domain values.
Datetime	Continuous	No input values are included in the list of domain values.
Boolean	Boolean	True and False
Numeric	If there are ten or fewer distinct values in the input data, the domain type is Discrete. If there are greater than ten distinct values, the domain type is Continuous.	For Discrete domain types, all of the values in the input data are included in the list of domain values. For Continuous domain types, only the minimum and maximum values are included in the list of domain values.

%BRM_EXPORT_FOLDER

Exports definitions of business rules folders into a CSV file. You can modify the CSV file and use it as input to the %BRM_IMPORT_FOLDER macro.

Restriction: This macro must be run on the server tier.

Syntax

%BRM_EXPORT_FOLDER (CSV=*output_filename*.CSV<, FOLDER_PATH=*path_name*>);

Required Argument

CSV=*output_filename*
specifies the full pathname to the CSV file where you want to export the data.

Optional Argument

FOLDER_PATH=<*path_name*>
specifies a business rules folder that you want to export. By default, %BRM_EXPORT_FOLDER exports all lookup tables. You do not need to specify the FOLDER_PATH= option unless you want to export a specific folder.

Example FOLDER_PATH=Loans/Retail/Applications

%BRM_EXPORT_LOOKUP

Exports the contents of lookup tables into a CSV file.

Restriction: This macro must be run on the server tier.

Syntax

%BRM_EXPORT_LOOKUP (CSV=*output_filename*.CSV<, FOLDER_PATH=*path-name*> <, LOOKUP='*lookup_table_1*'<, '*lookup_table_2*'>...>);

Required Argument

CSV=*output_filename*
specifies the full pathname to the CSV file where you want to export the data.

Optional Arguments

FOLDER_PATH=*path_name*
specifies a business rules folder pathname in SAS Decision Manager that you want to filter the output by. If you specify a folder pathname, then only the objects in that path are exported.

Use a forward slash to separate folder names.

Example FOLDER_PATH=Loans/Retail/Applications

LOOKUP=%STR('*lookup_table_1*'<, '*lookup_table_2*'>...)
specifies the lookup tables that you want to export. Specify the names of the lookup tables, enclosed in single quotation marks. Separate multiple names with commas.

By default, %BRM_EXPORT_LOOKUP exports all lookup tables. You do not need to specify the LOOKUP= option unless you want to export specific tables.

Tip You can filter the lookup tables that are exported by specifying the FOLDER_PATH= option.

Example lookup=%str('BadVINSTates','StateCodes')

%BRM_EXPORT_RULE_FLOW

Exports rule flows from the rules database into a CSV file.

Restriction: This macro must be run on the server tier.

Syntax

%BRM_EXPORT_RULE_FLOW (
RULEFLOWS=ALL | %STR(*rule_flow_1*<,*rule_flow_2*>...),
CSV=*output_filename*.CSV<, FOLDER_PATH=*path_name*>);

Required Arguments

CSV=*output_filename*
specifies the full pathname to the CSV file where you want to export the data.

RULEFLOWS=ALL | %STR(*rule_flow_1*<, *rule_flow_2*>...)
specifies the rule flows that you want to export. Specify ALL to export all rule flows. To export only selected rule flows, specify the identification numbers of the rule flows enclosed in quotation marks. Separate multiple identification numbers with commas.

Tip You can filter the rule flows that are exported by specifying the FOLDER_PATH= option.

Example
```
ruleflows=%str(10168,10043)
```

Optional Argument

FOLDER_PATH=*path_name*
specifies a business rules folder pathname in SAS Decision Manager that you want to filter the output by. If you specify a folder pathname, then only the objects in that path are exported. For example, if you specify RULEFLOWS=ALL and FOLDER_PATH=RetailLoans, then only the rule flows in the folder RetailLoans are exported. If you specify RULEFLOWS=%STR(10045,10572) and FOLDER_PATH=RetailLoans, but neither of the specified rule flows are in the RetailLoans folder, then no rule flows are exported.

Use a forward slash to separate folder names.

Example FOLDER_PATH=Loans/Retail/Applications

%BRM_EXPORT_RULESET

Exports rule sets from the rules database into a CSV file.

Restriction: This macro must be run on the server tier.

Syntax

%BRM_EXPORT_RULESET (RULESETS=ALL | %STR(*rule_set_1*<, *rule_set_2*>...), CSV=*output_filename*.CSV)<, FOLDER_PATH=*path_name*)>);

Required Arguments

CSV=*output_filename*
specifies the full pathname to the CSV file where you want to export the data.

RULESETS=ALL | %STR(*rule_set_1*<, *rule_set_2*>...)
specifies the rule sets that you want to export. Specify ALL to export all rule sets. To export only selected rule sets, specify the identification numbers of the rule sets enclosed in quotation marks. Separate multiple identification numbers with commas.

Tip You can filter the rule sets that are exported by specifying the FOLDER_PATH= option.

Example
```
rulesets=%str(168,43)
```

Optional Argument

FOLDER_PATH=*path_name*
specifies a business rules folder pathname in SAS Decision Manager that you want to filter the output by. If you specify a folder pathname, then only the objects in that path are exported. For example, if you specify RULESETS=ALL and FOLDER_PATH=RetailLoans, then only the rule sets in the folder RetailLoans are exported. If you specify RULESETS=%STR(10045,10572) and FOLDER_PATH=RetailLoans, but neither of the specified rule sets are in the RetailLoans folder, then no rule sets are exported.

Use a forward slash to separate folder names.

Example FOLDER_PATH=Loans/Retail/Applications

%BRM_EXPORT_VOCABULARY

Exports vocabularies from the rules database into a CSV file.

Restriction: This macro must be run on the server tier.

Syntax

%BRM_EXPORT_VOCABULARY (
VOCAB=ALL| %STR('*vocabulary_1*'<, '*vocabulary_2*'>...),
CSV=*output_filename*.CSV)<, FOLDER_PATH=*path_name*>);

Required Arguments

CSV=*output_filename*
specifies the full pathname to the CSV file where you want to export the data.

VOCAB=ALL | %STR('*vocabulary_1*'<, '*vocabulary_2*'>...)
specifies vocabularies that you want to export. Specify ALL to export all vocabularies. To export only selected vocabularies, specify the names of the

vocabularies enclosed in quotation marks. Separate multiple identification numbers with commas.

Tip You can filter the vocabularies that are exported by specifying the FOLDER_PATH= option.

Example
```
vocab=%str('LRAutoVocab','AcmeAuto')
```

Optional Argument

FOLDER_PATH=*path_name*
specifies a business rules folder pathname in SAS Decision Manager that you want to filter the output by. If you specify a folder pathname, then only the objects in that path are exported. For example, if you specify VOCAB=ALL and FOLDER_PATH=RetailLoans, then only the vocabularies in the folder RetailLoans are exported. If you specify VOCAB=%STR('loanVocab','riskVocabulary') and FOLDER_PATH=RetailLoans, but neither of the specified vocabularies are in the RetailLoans folder, then no vocabularies are exported.

Use a forward slash to separate folder names.

Example FOLDER_PATH=Loans/Retail/Applications

%BRM_IMPORT_FOLDER

Imports the folder definitions that are in the specified CSV file into the rules database.

Restrictions: This macro must be run on the server tier.

The same user can run any of import macros at the same time. However, different users cannot run the same import macro simultaneously.

Syntax

%BRM_IMPORT_FOLDER (CSV=*input_filename*.CSV, REJECT=*reject_filename*.CSV<, BRM_USER=*user_ID*>);

Required Arguments

CSV=*input_filename*
specifies the full pathname to the CSV file where you want to import the data from.

REJECT=*reject_filename*
specifies the full pathname to the CSV file where you want the macro to write any records that were not imported to the rules database. See "Using the %BRM_IMPORT_FOLDER Macro" on page 140 for more information.

Optional Argument

BRM_USER=*user_ID*
specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the rules database and is displayed in the SAS Decision Manager interface.

Default User ID of the user that is running the macro

Details

Using the %BRM_IMPORT_FOLDER Macro

The %BRM_IMPORT_FOLDER macro enables you to create new folders. You cannot update the content in existing folders with this macro. The macro uses the pathname to determine whether a folder already exists. If the pathname already exists, then the folder is rejected.

The %BRM_IMPORT_FOLDER macro runs several validation checks as it imports the folders. For example, it checks whether each folder path begins with a top-level folder and verifies that individual folder names are not longer than 100 characters. If the macro finds an invalid folder definition in the CSV file, it writes a message to the SAS log, and the folder is rejected. The macro writes the input records for the rejected folder to the CSV file that was specified in the REJECT= option.

Format of the Folder CSV Input File

Each row of the CSV input file identifies a folder. The CSV file must contain all of the columns listed in the following table, in the order listed. You must specify values for all columns, except as noted in the following table. To create a blank column in the CSV file, specify two comma separators without any content between them. For example, to import data to a folder named Applications and to specify a blank column for the folder description and default folder flag, specify the following in the CSV file:

```
Applications,,Y,,Loans/Retail
```

Table 13.2 *Format of the Folder CSV Input File*

Column	Description	Can Column Be Blank
FOLDER_NM	The name of the folder where you want to import the contents of the CSV file.	No
FOLDER_DESC	The description of the folder.	Yes
TOP_LEVEL_FOLDER_FLG	Specifies whether the folder is a top-level folder. Specify **Y** or **N**.	No
DEFAULT_FOLDER_FLG	Specifies whether the folder is the default folder. Specify **Y** or **N**.	Yes
FOLDER_PATH	The pathname to the business rules folder where you want to import the contents of the CSV file. This path must exist. Separate folder names with forward slashes.	No

%BRM_IMPORT_LOOKUP

Imports lookup tables from the specified CSV file into the rules database.

Restrictions: This macro must be run on the server tier.

The same user can run any of import macros at the same time. However, different users cannot run the same import macro simultaneously.

Syntax

%BRM_IMPORT_LOOKUP (CSV=*input_filename*.CSV,
REJECT=*reject_filename*.CSV<, *options*>);

Required Arguments

CSV=*input_filename*
specifies the full pathname to the CSV file where you want to import the data from.

REJECT=*reject_filename*
specifies the full pathname to the CSV file where you want the macro to write any records that were not imported to the rules database. See "Using the %BRM_IMPORT_LOOKUP Macro" on page 141 for more information.

Optional Arguments

BRM_USER=*user_ID*
specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the rules database and is displayed in the SAS Decision Manager interface.

Default User ID of the user that is running the macro

BYPASSLOCK=Y|N
enables you to override the lock that another user has on the importing process. See "Using the %BRM_IMPORT_LOOKUP Macro" on page 141 for more information.

Default N

Details

Using the %BRM_IMPORT_LOOKUP Macro

The %BRM_IMPORT_LOOKUP macro enables you to do the following tasks:

- add new lookup tables
- add new key-value pairs to existing lookup tables
- update (refresh) existing key-value pairs in existing lookup tables

The macro uses the lookup table name and path to determine whether a lookup table already exists. If the lookup table already exists, then it is updated. If the path exists but the lookup table does not exist, the lookup table is created. If the path does not exist, then the lookup table is rejected.

The %BRM_IMPORT_LOOKUP macro runs several validation checks as it imports the lookup tables. For example, the macro checks whether the LOOKUP_NM or NAME columns in the input file are empty or whether the LOOKUP_NM column specifies an invalid lookup name. All valid key-value pairs are imported. If the macro finds an invalid key-value pair in the CSV file, it writes a message to the SAS log, and the key-value pair is rejected. The macro writes the input records for the rejected key-value pairs to the CSV file that was specified in the REJECT= option.

When you run the %BRM_IMPORT_LOOKUP macro, it creates a lock table in the rules database named lock_import_lookup table. The SAS log states which user holds the lock and the time at which the lock started. This lock might remain in place after the macro has finished. If this happens, you can override the lock by specifying the BYPASSLOCK=Y option when you run the macro.

Format of the Lookup CSV Input File

Each row of the CSV input file identifies a key-value pair and the lookup table in which it belongs. The CSV file must contain all of the columns listed in the following table, in the order listed. You must specify values for all columns, except as noted in the table. To create a blank column in the CSV file, specify two comma separators without any content between them. For example, to import the key **AU** and the value **Australia** into the lookup table **Country_Codes** and to specify a blank column for the description, specify the following in the CSV file:

```
Country_Codes,,AU,Australia,Loans/Retail
```

Table 13.3 *Format of the Lookup CSV Input File*

Column	Description	Can Column Be Blank
FOLDER_PATH	The pathname to the business rules folder where you want to import the lookup table. This path must exist. Separate folder names with forward slashes.	No
LOOKUP_NM	The name of the lookup table.	No
DESCRIPTION	The description of the lookup table.	Yes
NAME	The lookup key.	No
VALUE	The lookup value.	Yes

%BRM_IMPORT_MARKET_BASKETS

Imports association rules that are generated by SAS Enterprise Miner and creates a rule flow that deploys the association rules.

Requirement: Before running this macro, you need to perform a market basket analysis using SAS Enterprise Miner and generate the data set that this macro uses as input.

Syntax

%BRM_IMPORT_MARKET_BASKETS (
RULETABLE= %STR (*association-rules-data-set*),
TRANSDATA= %STR (*transactional-data-set*),
CUSTOMER= %STR (*column-name*),
ITEM= %STR (*column-name*),
FOLDER= %STR (*top-level-folder*</*folder*...>
<, *options*>
);

Required Arguments

CUSTOMER= %STR (*column-name*)
specifies the variable name of the customer column in the transactional data set. The column name must be a valid SAS variable name.

FOLDER= %STR (*top-level-folder</folder...>*)
specifies the business rules folder where you want to load the output of the %BRM_IMPORT_MARKET_BASKETS macro.

Requirement This folder must already exist. If it does not exist, the macro does not import any data, and it writes an error to the SAS log.

ITEM= %STR (*column-name*)
specifies the variable name of the Item Purchased column in the transactional data set. The column name must be a valid SAS variable name.

RULETABLE= %STR (*association-rules-data-set*)
specifies the data set that contains the association rules that was created by SAS Enterprise Miner. For information, see SAS Enterprise Miner documentation at `http://support.sas.com/documentation/onlinedoc/miner/index.html`.

TRANSDATA= %STR (*transactional-data-set*)
specifies the transactional SAS data set that contains the customer and item variables. This data set must contain the dictionary of the original transaction data that was used for the market basket analysis. The macro retrieves only the dictionary information for the customer and item variables.

Optional Arguments

BYPASSLOCK=Y|N
enables you to override the lock that another user has on the importing process.

Default N

CLUSTER=OPTGRAPH | FASTCLUS | NONE
specifies the method that was used to group the association rules in clusters. Associations analysis can produce a very large number of rules. Clustering creates multiple rule sets by grouping together rules that have similar sets of conditions. Multiple rule sets are easier to manage and more efficient than deploying a single set of many unrelated rules.

Default OPTGRAPH

DESCRIPTION= %STR (*description*)
specifies the description for the business rules vocabulary, the rule set, and the rule flow. The maximum length is 256 bytes.

Default Business Rules for Market Basket Analysis

LOADBRM=Y|N
specifies whether to load the output of the macro into the business rules database. If you want to review the data sets that this macro produces before the data is loaded into the business rules database, specify **N**. The data sets are named brm_ruleflow, Vocabulary, RuleInitialize, ConditionAssign, and ActionAssign*n*, where*n* is the cluster number.

Default	Y
Requirement	You must have access to the business rules metadata source in order to specify `Y`.

NAME= %STR (*name*)
specifies the name of the business rules vocabulary and the prefix for the rule set and rule flow names. The name must be a valid vocabulary name.

Default	BR_for_MBA

Details

About Market Basket Analysis

Associations analysis is used to find items, such as milk and bread, that are commonly purchased together more often than random chance would explain. Rules are then derived to predict the item that will be purchased when other items are purchased. This type of analysis is frequently referred to as market basket analysis. Market basket analysis can be used to predict the items that are most likely to be added to the basket. These predictions can be expressed as rules. For example:

- If a customer buys bread, then the customer is likely to buy milk. When this prediction is defined as a business rule in SAS Decision Manager, bread is a condition term, and milk is an action term.
- If a customer buys bread and milk, then the customer is likely to buy eggs. When this prediction is defined as a business rule in SAS Decision Manager, milk and bread are condition terms, and eggs is an action term.

These business rules can be used as recommendations. They can be used to design sales promotions, structure loyalty programs, determine discount plans, plan up-selling and cross-selling strategies, and so on.

The %BRM_IMPORT_MARKET_BASKETS macro uses the market basket data generated by SAS Enterprise Miner to create rule sets and a rule flow. It imports the rule sets and rule flow into the business rules database. Using SAS Decision Manager, you can enable or disable rules and modify rule conditions or rule actions. You can combine these market basket rules with other rules concerning business strategy such as pricing, discounts, customer segments, or loyalty. You can test the rule flow by using any input data that is in the same format as data that was used to perform the original associations analysis in SAS Enterprise Miner.

SAS Enterprise Miner provides several features that enable you to perform market basket analysis and produce the input data required by this macro. For more information about associations analysis, see SAS Enterprise Miner documentation at `http://support.sas.com/documentation/onlinedoc/miner/index.html`.

Business Rules Content Created by the Macro

The %BRM_IMPORT_MARKET_BASKETS macro imports business rules derived from a market basket analysis. Specifically, this macro creates a vocabulary, rule sets, and a complex rule flow, and imports all of this content into SAS Decision Manager. This content is imported into the folder specified by the FOLDER= option.

The vocabulary created by the macro is described in "Vocabulary Created by the Macro" on page 145.

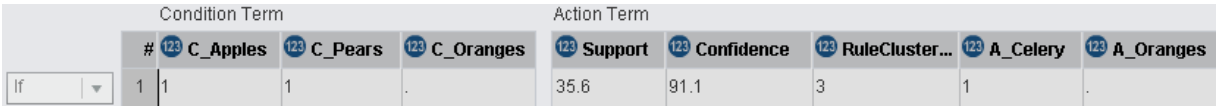
In the rule sets created by the macro, purchased products are added to the decision table as condition terms. Recommended products are added to the decision table as action

terms. The statistical measures Support and Confidence are added to the decision table as action terms. No term is added as both a condition term and an action term.

Each rule is one recommendation. For example, a logical rule can be stated as follows:

```
IF basket CONTAINS ('apples', 'pears') THEN recommendation = celery
```

The %BRM_IMPORT_MARKET_BASKETS macro might translate this logical rule into a business rule that appears in the decision table as follows:



		Condition Term			Action Term				
	#	C_Apples	C_Pears	C_Oranges	Support	Confidence	RuleCluster...	A_Celery	A_Oranges
If	1	1	1	.	35.6	91.1	3	1	.

The rule flows that %BRM_IMPORT_MARKET_BASKETS creates use the customer identifier variable as the BY-group term. Each unique customer identifier defines a BY-group. The rule flow contains the following rule sets:

- one rule set for initializing the indicator conditions and actions. This rule flow is in the Group Start section of the rule flow.
- one rule set that assigns values to indicator conditions. This rule set is in the Main section of the rule flow.
- one rule set for each cluster of rules. These rule sets are in the Group End section of the rule flow. The rules in the Group End section are the recommendations. You should examine these rules for patterns that you want to use in your business scenario. Look for actions that you might want to modify to fit your business strategy. For example, you might want to sell steak instead of eggs.

CAUTION:

You should not edit the rules in Group Start and Main sections except to change the name of an item. The rules in the Group Start and Main sections transform the transactional data so that it can be processed by the rules in the Group End section.

Vocabulary Created by the Macro

The %BRM_IMPORT_MARKET_BASKETS macro defines a vocabulary with four entities: Actions, Conditions, Data, and Measures.

The terms in each entity are listed in the following table.

Table 13.4 *Vocabulary Created by the %BRM_IMPORT_MARKET_BASKETS Macro*

Entity	Term	Exclude from input	Exclude from output	Description
Actions	A_*product_name*	Y	N	The *product_name* is the name of a product in customer transaction data. The product (item) name must be a valid SAS variable name. In the rule sets, the value of A_*product_name* is 1 if the product name is a recommendation based on the association pattern. Otherwise, the value of A_*product_name* is 0.

Entity	Term	Exclude from input	Exclude from output	Description
Conditions	C_*product_name*	Y	Y	The *product_name* is the name of a product in customer transaction data. The product (item) name must be a valid SAS variable name. In the rule sets, the value of C_*product_name* is 1 if the product name is found in the market basket and is a condition in the association pattern. Otherwise, the value of C_*product_name* is 0.
Data	*Customer*	N	N	Variable name of the Customer column in the transactional data.
Data	*Item*	N	Y	Variable name of the Item Purchased column in the transactional data.
Measures	Confidence	Y	Y	Confidence index of the association pattern. Confidence is a measure of the likelihood of the action given that the condition has occurred. This value is provided as information about the rule and is excluded from both input and output.
Measures	Support	Y	Y	Support index of the association pattern. Support is a measure of how often this pattern occurs in the source data. Users often want to find high confidence, low support rules. This value is provided as information about the rule and is excluded from both input and output.
Measures	RuleClusterIndex	Y	Y	Index of the cluster of rules found by the clustering method specified by the CLUSTER= option.

Example: Market Basket Analysis of Groceries Data

Note: This example requires SAS Enterprise Miner.

The following example generates a market basket analysis of the data in the SAMPSIO.ASSOCS data set. It generates a vocabulary, several rule sets, and a complex rule flow. It loads all of this content into the folder **`MyFolder/Grocery`**.

In the SAMPSIO.ASSOCS data set, the variable Customer is the customer identifier and is a numeric variable. The variable Product is the product identifier and is a character variable.

The following code performs the market basket analysis and generates the output data set that is used as input to the %BRM_IMPORT_MARKET_BASKETS macro.

```
proc dmdb data=sampsio.assocs dmdbcat=d;
   class customer product;
quit;

proc assoc data=sampsio.assocs dmdbcat=d out=a
   items=6 support=90;
   customer customer;
   target product;
```

```
run;

proc rulegen in=a dmdbcat=d out=rulegen
   minconf = 99;
run;
```

The %BRM_IMPORT_MARKET_BASKETS macro uses the output data set that the RULEGEN procedure created in order to generate and load a rule flow into the business rules database.

```
%BRM_IMPORT_MARKET_BASKETS (
     RuleTable = rulegen,
     TransData = sampsio.assocs,
     Customer = customer,
     Item = product,
     Name = %str(MBA_RuleCluster),
     Description = %str(MBA_RuleCluster),
     Folder = %str(MyFolder/Grocery),
     Cluster = OPTGRAPH,
     LoadBRM = Y,
     bypassLock = Y);
```

The following display shows part of the vocabulary that the %BRM_IMPORT_MARKET_BASKETS macro created. Because the NAME= option specifies MBA_RuleCluster, both the vocabulary and the rule flow that the macro created are named MBA_RuleCluster.

Display 13.1 *Vocabulary MBA_RuleCluster*

Grocery	Folder	Market Basket example using Grocery Data SAMPSIO.ASSOCS		
MBA_RuleCluster	Vocabulary	MBA Rule Cluster		
Actions	Entity	Market Basket Actions		
A_avocado	Term	Market Basket Action: A_avocado	Integer	Discrete
A_baguette	Term	Market Basket Action: A_baguette	Integer	Discrete
A_bourbon	Term	Market Basket Action: A_bourbon	Integer	Discrete
A_chicken	Term	Market Basket Action: A_chicken	Integer	Discrete
A_coke	Term	Market Basket Action: A_coke	Integer	Discrete
A_corned_b	Term	Market Basket Action: A_corned_b	Integer	Discrete
A_cracker	Term	Market Basket Action: A_cracker	Integer	Discrete
A_heineken	Term	Market Basket Action: A_heineken	Integer	Discrete
A_hering	Term	Market Basket Action: A_hering	Integer	Discrete
A_ice_crea	Term	Market Basket Action: A_ice_crea	Integer	Discrete
A_olives	Term	Market Basket Action: A_olives	Integer	Discrete
A_sardines	Term	Market Basket Action: A_sardines	Integer	Discrete
Conditions	Entity	Market Basket Conditions		
Data	Entity	Market Basket Transaction Data Variables		
CUSTOMER	Term	Market Basket Transaction Data Variable: CUSTOMER	Decimal	Discrete
PRODUCT	Term	Market Basket Transaction Data Variable: PRODUCT	String	Discrete
Measures	Entity	Market Basket Measures		
Confidence	Term	Market Basket Measure: Confidence	Decimal	Continuous
RuleClusterIndex	Term	Market Basket Measure: RuleClusterIndex	Integer	Discrete
Support	Term	Market Basket Measure: Support	Decimal	Continuous

The following display shows the structure of the MBA_RuleCluster rule flow. The rules in the Group Start and Main sections transform the transactional data so that it can be processed by the rules in the Group End section.

Display 13.2 *Rule Flow MBA_RuleCluster*

▾ Group Start (Start of each BY-group) [1]

	Name	Description
1	MBA_RuleCluster, Initialize	Market Basket Rule: MBA_RuleCluster, Step 1: Initialization

▾ Main [1]

	Name	Description
1	MBA_RuleCluster, Assign Conditions	Market Basket Rule: MBA_RuleCluster, Step 2: Assignment of Conditions

▾ Group End (End of each BY-group) [5]

	Name	Description
1	MBA_RuleCluster, Implement Rule Cluster 1	Market Basket Rule: MBA_RuleCluster, Step 3: Implementation of Rule Cluster 1
2	MBA_RuleCluster, Implement Rule Cluster 2	Market Basket Rule: MBA_RuleCluster, Step 3: Implementation of Rule Cluster 2
3	MBA_RuleCluster, Implement Rule Cluster 3	Market Basket Rule: MBA_RuleCluster, Step 3: Implementation of Rule Cluster 3
4	MBA_RuleCluster, Implement Rule Cluster 4	Market Basket Rule: MBA_RuleCluster, Step 3: Implementation of Rule Cluster 4
5	MBA_RuleCluster, Implement Rule Cluster 5	Market Basket Rule: MBA_RuleCluster, Step 3: Implementation of Rule Cluster 5

The following display shows the condition table for the first rule set in the Group End section, RuleCluster1. The rules in the Group End section are the recommendations.

Display 13.3 *Condition Table in Rule Set RuleCluster1*

Condition Term

	#	C_artichok	C_avocado	C_baguette	C_cracker	C_ham	C_heineken	C_hering	C_soda
If	1			1	1				1
If	2	1	1		1				
If	3	1			1	1			
If	4			1	1			1	1
If	5	1	1	1				1	
If	6	1	1		1	1			
If	+	1			1	1	1		

The following display shows the action table for the first rule set in the Group End section, RuleCluster1. You should examine the patterns in the rule sets in the Group End section. You might find actions that you want to modify to fit your business strategy.

Display 13.4 *Action Table in Rule Set RuleCluster1*

Action Term

Support	Confidence	RuleCluster...	A_avocado	A_heineken
12.58741258741	99.21259842519	1		1
11.18881118881	99.11504424778	1		1
9.990009990009	99.00990099009	1		1
11.48851148851	100	1		1
9.890109890109	100	1		1
9.890109890109	100	1		1
9.890109890109	99	1	1	

%BRM_IMPORT_RULE_FLOW

Imports rule flows from the specified CSV file into the rules database.

Restrictions: This macro must be run on the server tier.

The same user can run any of import macros at the same time. However, multiple users cannot run the same import macro simultaneously.

Syntax

%BRM_IMPORT_RULE_FLOW(CSV=*input_filename*.CSV, REJECT=*reject_filename*.CSV<, *options*>);

Required Arguments

CSV=*input_filename*
specifies the full pathname to the CSV file where you want to import the data from. For more information, see "Format of the Rule Flow CSV Input File" on page 150.

REJECT=*reject_filename*
specifies the full pathname to the CSV file where you want the macro to write any records that were not imported to the rules database. See "Using the %BRM_IMPORT_RULE_FLOW Macro" on page 150 for more information.

Optional Arguments

BRM_USER=*user_ID*
specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the rules database and is displayed in the SAS Decision Manager interface.

Default User ID of the user that is running the macro

BYPASSLOCK=Y|N
enables you to override the lock that another user has on the importing process. See "Using the %BRM_IMPORT_RULE_FLOW Macro" on page 150 for more information.

Default N

Details

Using the %BRM_IMPORT_RULE_FLOW Macro

The %BRM_IMPORT_RULE_FLOW macro enables you to add new rule flows and to update existing rule flows. The macro uses the rule flow name and rule flow path to determine whether a rule flow already exists. If the rule flow path and name already exist, then the rule flow is updated. If the rule flow path exists but the rule flow name does not exist, the rule flow is created. If the rule flow path does not exist, then the rule flow is rejected.

The %BRM_IMPORT_RULE_FLOW macro runs several validation checks as it imports the rule flows. For example, it checks whether a rule set is referenced in a given rule flow more than once and whether section codes are correct. If the macro finds a validation error in a rule flow, it writes a message to the SAS log, and the rule flow is rejected. The macro writes the input records for the rejected rule flow to the CSV file that was specified in the REJECT= option.

When you run the %BRM_IMPORT_RULE_FLOW macro, it creates a lock table in the rules database named lock_import_rule_flow. The SAS log states which user holds the lock and the time at which the lock started. This lock might remain in place after the macro has finished. If this happens, you can override the lock by specifying the BYPASSLOCK=Y option when you run the macro.

Format of the Rule Flow CSV Input File

Each row of the CSV input file identifies a rule set and a rule flow, and each row provides the information about how that rule set fits into the rule flow. The CSV file must contain all of the columns that are listed in the following table, in the order listed. You must specify values for all columns, except as noted in the table. To create a blank column in the CSV file, specify two comma separators without any content between them. For example, to add a rule set to the main section of the rule flow named assignRisk and to specify a blank column for the rule flow description, specify the following in the CSV file:

```
assignRisk,,main
```

Table 13.5 *Format of the Rule Flow CSV Input File*

Column	Description	Can Column Be Blank
RULE_FLOW_SK	The identification number of the rule flow.	Yes
RULE_FLOW_NM	The name of the rule flow where you want to add the rule set that is specified in RULE_SET_NM.	No
RULE_FLOW_SHORT_DESC	The description of the rule flow.	Yes

Column	Description	Can Column Be Blank
RULE_SET_SECTION_CODE	The section of the rule flow to which the rule set that is specified in RULE_SET_NM belongs. Specify **`init`**, **`groupstart`**, **`main`**, **`groupend`**, or **`final`**. The codes **`groupstart`** and **`groupend`** are valid only if you also specify at least one term for BY_TERM.See “Simple Rule Flows, Complex Rule Flows, and BY Groups” in Chapter 9 of *SAS Decision Manager: User's Guide* for more information.	No
RULE_FLOW_PATH	The pathname to the business rules folder for the rule flow. This path must exist. Separate folder names with forward slashes.	No
RULE_SET_NM	The name of the rule set to be added to the rule flow. A rule set can be added to the same rule flow only once.	No
RULE_SET_PATH	The pathname to the business rules folder to the rule set that is specified by RULE_SET_NM. The rule set must exist at the specified location. Separate folder names with forward slashes.	No
VOCAB_NM	The name of the vocabulary that the rule set uses. All rule sets in the same rule flow must use the same vocabulary.	No
BY_TERM	The list of BY-group terms that the rule set uses. Separate multiple BY-group terms with commas. The BY-group terms must be the same for all rule sets that are in the same rule flow. All of the BY-group terms must belong to the same vocabulary. See “Simple Rule Flows, Complex Rule Flows, and BY Groups” in Chapter 9 of *SAS Decision Manager: User's Guide* for more information.	Yes
ORDER	The order number for the rule set that is in the rule flow. Order numbers must start with 1 and be continuous through the entire rule flow. Do not restart order numbers at section boundaries.	No

%BRM_IMPORT_RULESET

Imports rule sets from the specified CSV file into the rules database.

Restrictions: This macro must be run on the server tier.

The same user can run any of import macros at the same time. However, multiple users cannot run the same import macro simultaneously.

Syntax

%BRM_IMPORT_RULESET (CSV=*input_filename*.CSV, REJECT=*reject_filename*.CSV<, *options*>);

Required Arguments

CSV=*input_filename*
specifies the full pathname to the CSV file where you want to import the data from. For more information, see "Format of Rule Set CSV Input File" on page 152.

REJECT=*reject_filename*
specifies the full pathname to the CSV file where you want the macro to write any records that were not imported to the rules database. See "Using the %BRM_IMPORT_RULESET Macro" on page 152 for more information.

Optional Arguments

BRM_USER=*user_ID*
specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the rules database and is displayed in the SAS Decision Manager interface.

Default User ID of the user that is running the macro

BYPASSLOCK=Y|N
enables you to override the lock that another user has on the importing process. See "Using the %BRM_IMPORT_RULESET Macro" on page 152 for more information.

Default N

Details

Using the %BRM_IMPORT_RULESET Macro

The %BRM_IMPORT_RULESET macro enables you to add new rule sets and to update existing rule sets. The macro uses the rule set name and rule set path to determine whether a rule set already exists. If the rule set path and name already exist, then the rule set is updated. If the rule set path exists but the rule set name does not exist, the rule set is created. If the rule set path does not exist, then the rule set is rejected.

The %BRM_IMPORT_RULESET macro runs several validation checks as it imports the rule sets. For example, it verifies that the expressions are valid, ensures that the first rule in each rule set uses the IF operator, and verifies that the specified vocabularies exist. If the macro finds a validation error in a rule set, it writes a message to the SAS log, and the rule set is rejected. The macro writes the input records for the rejected rule set and the reason that the record was rejected to the CSV file that was specified in the REJECT= option.

When you run the %BRM_IMPORT_RULESET macro, it creates a lock table in the rules database named lock_import_rule_set. The SAS log states which user holds the lock and the time at which the lock started. This lock might remain in place after the macro has finished. If this happens, you can override the lock by specifying the BYPASSLOCK=Y option when you run the macro.

Format of Rule Set CSV Input File

Each row of the CSV input file specifies a rule, rule set, term, and an expression for that term. The row also specifies whether the expression is a condition expression or an action expression. Each row of the input file can specify only one condition expression or one action expression for a given rule. The CSV file must contain all of the columns that are listed in the following table, in the order listed. You must specify values for all

columns, except as noted in the table. To create a blank column in the CSV file, specify two comma separators without any content between them. For example, to add a rule to the rule set named assignRisk that uses the loanVocab vocabulary and to specify a blank column for the rule set description, specify the following in the CSV file:

```
assignRisk,,loanVocab
```

Table 13.6 *Format of the Rule Set CSV Input File*

Column	Description	Can Column Be Blank
RULE_SET_SK	The identification number of the rule set.	Yes
RULE_SET_NM	The name of the rule set where you want to add the rule that is specified in RULE_NM.	No
RULE_SET_DESC	The description of the rule set.	Yes
VOCAB_NM	The name of the vocabulary that the rule set uses. All rules in the same rule set must use the same vocabulary.	No
RULE_SET_PATH	The pathname to the business rules folder for the rule set. This path must exist. Separate folder names with forward slashes.	No
RULE_NM	The name of the rule to be added to the rule set.	No
RULE_DESC	The description of the rule.	Yes
RULE_SEQ_NO	The order number for the rule that is in the rule set. Order numbers in a rule set start with 1.	No
CONDITIONAL_NM	The operator for the rule. Specify **`if`**, **`elseif`**, or **`or`**. The first rule in a rule set must use the **`if`** operator.For information about these operators, see “Controlling Which Conditions Are Evaluated” in Chapter 8 of *SAS Decision Manager: User's Guide*.	No
LHS_TERM	The term for the expression specified in the EXPRESSION column. Terms that are specified in the LHS_TERM column are the terms that SAS Decision Manager displays at the top or left side of the decision table. These terms appear in the column headings of the decision table when you are viewing the decision table in the horizontal format. They appear in the row headings of the decision table when you are viewing the decision table in the vertical format.	No

Column	Description	Can Column Be Blank
EXPRESSION	A single condition or action expression for the term specified in the LHS_TERM column. This expression is the expression that you would enter into a cell in the decision table. See "Defining New Rules in the Rule Set" in Chapter 8 of *SAS Decision Manager: User's Guide* for more information about expressions.	Yes
EXPRESSION_ORDER	The order number of the rule's condition or action expressions. A rule's condition and action expressions are numbered beginning with 1. For example, a rule might have two condition expressions that are numbered 1 and 2, and it might have three action expressions that are numbered 1, 2, and 3.	No
EXPRESSION_TYPE	The type of expression. Specify **`condition`** or **`action`**.	No

%BRM_IMPORT_VOCABULARY

Imports vocabulary terms from the specified CSV file into the rules database.

Restrictions: This macro must be run on the server tier.

The same user can run any of import macros at the same time. However, the same import macro cannot be run simultaneously by different users.

Syntax

%BRM_IMPORT_VOCABULARY (CSV=*input_filename*.CSV, REJECT=*reject_filename*.CSV<, *options*>);

Required Arguments

CSV=*input_filename*
specifies the full pathname to the CSV file where you want to import the data from. For more information, see "Format of the Vocabulary CSV Input File" on page 155.

REJECT=*reject_filename*
specifies the full pathname to the CSV file where you want the macro to write any records that were not imported to the rules database. See "Using the %BRM_IMPORT_VOCABULARY Macro" on page 155 for more information.

Optional Arguments

BRM_USER=*user_ID*
specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the rules database and is displayed in the SAS Decision Manager interface.

Default User ID of the user that is running the macro

BYPASSLOCK=Y|N
enables you to override the lock that another user has on the importing process. See "Using the %BRM_IMPORT_VOCABULARY Macro" on page 155 for more information.

Default N

Details

Using the %BRM_IMPORT_VOCABULARY Macro

The %BRM_IMPORT_VOCABULARY macro enables you to add new vocabulary terms. You cannot use the macro to update existing terms.

The %BRM_IMPORT_VOCABULARY macro runs several validation checks as it imports the vocabulary terms. For example, it verifies that term, entity, and vocabulary names are valid, and ensures that a term is not duplicated in a vocabulary. If the macro finds a validation error, it writes a message to the SAS log, and the term is rejected. The macro writes the input records for the rejected term to the CSV file that was specified in the REJECT= option.

When you run the %BRM_IMPORT_VOCABULARY macro, it creates a lock table in the rules database named lock_import_vocabulary. The SAS log states which user holds the lock and the time at which the lock started. It is possible for this lock to remain in place after the macro has finished. If this happens, you can override the lock by specifying the BYPASSLOCK=Y option when you run the macro.

Format of the Vocabulary CSV Input File

Each row of the CSV input file defines a term, including the term data type, domain type, and the entity and vocabulary that contains the term. The CSV file must contain all of the columns listed in the following table, in the order listed. You must specify values for all columns, except as noted in the table. To create a blank column in the CSV file, specify two comma separators without any content between them. For example, to add a term to the entity named Customer in the vocabulary named loanVocab and to specify a blank column for the vocabulary description, specify the following in the CSV file:

```
loanVocab,,Customer
```

Table 13.7 *Format of the Vocabulary CSV Input File*

Column	Description	Can Column Be Blank
VOCAB_NM	The name of the vocabulary where you want to add entity and term specified by VOCAB_ENTITY_NM and VOCAB_TERM_NM.	No
VOCAB_SHORT_ DESC	The description of the vocabulary.	Yes
VOCAB_ENTITY_NM	The name of the entity that the term in the VOCAB_TERM_NM column belongs to.	No
VOCAB_ENTITY_ SHORT_DESC	The description of the entity.	Yes

Column	Description	Can Column Be Blank
VOCAB_TERM_NM	The name of the term.	No
VOCAB_TERM_ SHORT_DESC	The description of the term.	Yes
VOCAB_TERM_DATA _TYPE_TXT	The data type of the term. Specify **`string`**, **`decimal`**, **`integer`**, **`boolean`**, **`date`**, or **`datetime`**.	No
VOCAB_TERM_ DOMAIN_TYPE_TXT	The domain type for the term. Specify **`discrete`**, **`continuous`**, or **`boolean`**. A domain value is discrete if it is just an individual value such as **`5.3`** or **`18JUL2012:10:25:00`**. A domain value is continuous if it specifies a range such as **`>5`** or **`<18JUL2012:10:25:00`**. Terms that are assigned the data type **`string`** can have discrete domain values only. Boolean terms can have Boolean domain values only.	No
VOCAB_TERM_ DOMAIN_TXT	The set of expected values for a term. Separate individual domain values with a semi-colon (;). See "Specify Domain Values" in Chapter 5 of *SAS Decision Manager: User's Guide* for more information.	Yes
VOCAB_TERM_ INPUT_EXCLUDE_ FLG	Specifies whether the term must be mapped to a column in an input data set. Specify **`Y`** or **`N`**.	No
VOCAB_TERM_ OUTPUT_EXCLUDE_ FLG	Specifies whether to exclude the term from the output data sets created by rule flows. Specify **`Y`** or **`N`**.	No
FOLDER_PATH	The pathname to the business rules folder for the rule flow. This path must exist. Separate folder names with forward slashes.	No

%BRM_LOAD_VOCABULARY

Loads the vocabulary terms in the WORK.TERM data set that was created by the %BRM_CREATE_TEMP_TERM macro.

Syntax

%BRM_LOAD_VOCABULARY (FOLDER_PATH=*path*,
VOCAB_NM=*vocabulary-name*,
VOCAB_ENTITY_NM=*entity-name*<, *options*>);

Required Arguments

FOLDER_PATH=*path-name*
specifies the pathname to the business rules folder where you want to import the vocabulary terms. Separate folder names with forward slashes. This path must exist. If the path does not exist, the macro terminates and writes an error message to the SAS log.

Example FOLDER_PATH=Loans/Retail/Applications

VOCAB_NM=*vocabulary-name*
specifies the name of the vocabulary to which the terms in the WORK.TERM file will be added. The vocabulary must not exist. If it already exists, the macro terminates and writes an error message to the SAS log.

VOCAB_ENTITY_NM=*entity-name*
specifies the name of the entity to which the terms in the WORK.TERM file will be added. This entity must not exist. If it already exists, the macro terminates and writes an error message to the SAS log.

Optional Arguments

BRM_USER=*user_ID*
specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the rules database and is displayed in the SAS Decision Manager interface.

Default User ID of the user that is running the macro

BYPASSLOCK=Y|N
enables you to override the lock that another user has on the importing process.

Default N

Details

When you run the %BRM_LOAD_VOCABULARY macro, it creates a lock table in the rules database named lock_import_vocabulary. The SAS log states which user holds the lock and the time at which the lock started. It is possible for this lock to remain in place after the macro has finished. If this happens, you can override the lock by specifying the BYPASSLOCK=Y option when you run the macro.

Chapter 14

Utilities for Administrators

Release Tables Utility **160**
Overview of the Release Tables Utility 160
How to Use the Release Tables Utility 160
Release Tables Utility Examples 160

Batch Export and Import Tools **161**
Overview of the Batch Export and Import Tools 161
How to Use the Batch Export Tool 161
How to Use the Batch Import Tool 163
Batch Export and Import Tool Examples 164

Delete Environment Settings Utility **164**
Overview of the Delete Environment Settings Utility 164
How to Use the Delete Environment Settings Utility 164
Delete Environment Settings Utility Examples 165

Load Treatments **165**
Overview of the Load Treatments Utility 165
How to Use the Load Treatments Utility 166
Treatment Tables 166
Treatment Custom Detail Tables 167
Treatments to Delete Table 169
List Values Tables 170
Load Treatment Utility Example 170

Scripts for Managing SAS Decision Manager Flows **170**
Overview of Scripts 170
Vars Script 170
Run Script 174
Export Script 178
Import Script 181
Deploy Script 185
Test Script 188
Promote Script 191
Activate and Deactivate Scripts 194
Master Script 198
Log Files 201

Release Tables Utility

Overview of the Release Tables Utility

The Release Tables utility releases some of the disk space that is occupied by out-of-date tables in the MATables library. Use this utility to delete the tables belonging to flows that are no longer needed. The affected flows are reset and their counts are cleared.

In order to use the Release Tables utility, you must have Write or Update permission for all of the flows whose counts should be cleared.

How to Use the Release Tables Utility

The Release Tables utility is contained in the sasciutils.exe file. The typical location of this file is **`C:\Program Files\SASHome\SASDecisionManagerUtilities\6.2`**. From the directory that contains the sasciutils.exe file, execute the command with the following syntax:

sasciutils_console.exe *-argument1 -argument2*

The command takes the following arguments:

-releasetables
specifies the Release Tables utility.

-userid
is the user ID of the account that is used to execute the utility.

-password
is the password for the user ID.

-utilitylogfile
specifies the location to which error messages and success messages are logged. If you specify a directory path, all directories in the path must exist before you execute the utility.

-deldate
is the date to be used for determining whether a flow is reset. Tables that were last modified before this date will be deleted.

-bcname
specifies the name of the single business context to be processed. If this argument is not used, all business contexts are processed.

-listonly
directs the utility not to delete the tables or reset the counts for the flows. All logging is performed. This argument is useful for determining which flows would be affected by the utility.

Release Tables Utility Examples

The following example produces a log file that lists the flows that would be affected in the NorthWestBanking business context if the Release Tables utility were executed. The log file will list flows with **last modified** dates that are prior to April 28, 2013.

```
"C:\Program Files\SASHome\SASDecisionManagerUtilities\6.2\sasciutils_console.exe"
```

```
-releasetables -userid MyUserID -password MyPassword
-utilitylogfile c:/temp/tableCleanup.log -deldate "Apr 28, 2013"
-listonly -bcname NorthWestBanking
```

The following example clears the counts and releases the tables for all flow with a **last modified** date that is prior to June 1, 2013. The user has Update permission for all business contexts.

From the **C:\Program Files\SASHome\SASDecisionManagerUtilities \6.2** directory, execute the following command:

```
sasciutils_console.exe -releaseTables -userid MyUserId
-password MyPassword -utilitylogfile c:/temp/cleanup.log
-deldate "Jun 01, 2013"
```

Batch Export and Import Tools

Overview of the Batch Export and Import Tools

The batch export and import tools enable you to perform promotions from an operating system command line or from a batch script. You can use these tools to export and import flows and other objects between folders.

For information about the batch export and import tools, see the "Using the Batch Export and Import Tools" chapter in *SAS Intelligence Platform: System Administration Guide* at **SAS Intelligence Platform Product Documentation**.

How to Use the Batch Export Tool

The following code is an example of a Windows script that calls the batch export tools to export an object:

```
@ECHO OFF

SET SASHOME=C:\Program Files\<sas-config-dir>
SET SASFOLDER=/<path-to-object>
SET PROFILE=<metadata-server-connection-profile>

SET CURRENT=%CD%

IF %1.==. GOTO INVALID
IF %2.==. GOTO INVALID

CD %SASHOME%\SASPlatformObjectFramework\9.3

ExportPackage -profile "%PROFILE%" -subprop -package
"%USERPROFILE%\%~2.spk" -log "%USERPROFILE%\export_%~2.log"
-objects "%SASFOLDER%%~2(%~1)" %~3 %~4 %~5 %~6 %~7 %~8 %~9
GOTO END

:INVALID
GOTO END

:END
CD %CURRENT%
```

```
PAUSE
```

Modify the code and place it in a text file named ExportCIObject.cmd. From the directory that contains this file, execute the command with the following syntax:

```
ExportCIObject object-type object-name [-includeDep, -options]
```

The command takes the following arguments:

object-type
is the public object type. For more information, see Table 14.1 on page 162.

object-name
is the name of the object.

-includeDep
includes dependencies.

-options
are the other batch export tool options.

After you have exported the objects, modify the substitution properties file. For more information, see *SAS Intelligence Platform: System Administration Guide* at **SAS Intelligence Platform Product Documentation**.

Table 14.1 *Customer Intelligence Public Object Types*

Public Object Type	Name	Description
CIBusinessContext	Customer Intelligence business context	An environment with associated resources that is used to partition flow and treatment artifacts from one another when running on the same server
CICalculatedItem (not visible in SAS Management Console **Folder** view)	Customer Intelligence calculated item	A calculated item
CICampaign	Campaign	A planned set of one or more replies that are directed at the single recipient
CICampaignDefinition_Decision	Decision campaign definition	A template that specifies information about the underlying structure of a decision flow
CICellNode (not visible in SAS Management Console **Folder** view)	Campaign cell node	A node that represents a group or a subgroup that is targeted by a flow
CICustomDetailsGroup	Custom detail group	A collection of custom details that can be reused in more than one flow, treatment, or reply
CICustomDetailsTag	Custom detail tag	A user-friendly name for a custom detail
CIDecisionEvent	Campaign event	An event that is available to a decision flow
CIDecisionGlobalVariable	Campaign global variable	A global variable available to a decision flow

CIDecisionProcess	Campaign process	A process that is available to a decision flow
CIDiagram	Campaign diagram	A collection of nodes that make up a diagram
CIDocument (not visible in SAS Management Console **Folder** view)	Campaign document	A collection of reports that summarize the information in flow
CIEnvironmentData	Customer Intelligence environment definition	The definition of a SAS Decision Manager environment
CIIdentifier	Campaign variable identifier	A description of a variable
CIMetadataGenerationDefinition	Customer-Intelligence-generated metadata definition	The format of SAS-Decision-Manager-generated metadata
CINodeDefinition (not visible in SAS Management Console **Folder** view)	Campaign node definition	The definition of a node in a diagram
CIReplyDefinition	Campaign reply definition	A template that specifies information about a reply
CIReportExport	Customer Intelligence report export	A report link
CIResponseDefinition	Campaign response definition	A template that defines information about a response
CISurrogateKey	Customer Intelligence surrogate key definition	A map of SAS Decision Manager surrogate keys
CITreatment	Campaign treatment	A type of marketing communication. A treatment includes the format, creative content, and offer

How to Use the Batch Import Tool

The following code is an example of a Windows script that calls the batch import tools to import the objects in a package:

```
@ECHO OFF

SET SASHOME=C:\Program Files\<sas-config-dir>
SET SASFOLDER=<destination-pathname-of-imported-object>
SET PROFILE=metadata-server-connection-profile

SET CURRENT=%CD%

IF %1.==. GOTO INVALID

CD %SASHOME%\SASPlatformObjectFramework\9.3

ImportPackage -subprop "%USERPROFILE%\%~1.subprop" -newOnly -includeACL
```

```
-profile "%PROFILE%" -package "%USERPROFILE%\%~1.spk"
-log "%USERPROFILE%\import_%~1.log"
-target "%SASFOLDER%" %~2 %~3 %~4 %~5 %~6 %~7 %~8 %~9
GOTO END

:INVALID
GOTO END

:END
CD %CURRENT%
PAUSE
```

Modify the code and place it in a text file named ImportCIObject.cmd. From the directory that contains this file, execute the command with the following syntax:

```
ImportCIObject package-name  [-options]
```

The command takes the following arguments:

package-name
is the name of the package that contains the exported objects.

-options
are the batch import tool options.

Batch Export and Import Tool Examples

The following example exports an object that is named **my flow**. Because the name contains spaces, it is enclosed in quotation marks. The public object type is CICampaign. Dependencies are included in the export.

```
ExportCIObject CICampaign "my flow" -includeDep
```

The following example imports a package that is named **my flow**. Because the name contains spaces, it is enclosed in quotation marks. The **.spk** package extension is not included in the name.

```
ImportCIObject "my flow"
```

Delete Environment Settings Utility

Overview of the Delete Environment Settings Utility

In some circumstances, the SAS Decision Manager environment in the Folder view of SAS Management Console is corrupted and you cannot log on to SAS Decision Manager. You can resolve the problem by using the Delete Environment Settings utility to delete the Environment Settings object. The environment will be re-created the next time you log on.

How to Use the Delete Environment Settings Utility

The Delete Environment Settings utility is contained in the sasciutils.exe file on Windows and in the sasciutils.sh file on UNIX. To write the results to the console rather than to the log, use sasciutils_console.exe or sasciutils_console.sh.

On Windows, the sasciutils.exe files is typically installed in **C:\Program Files \SASHome\SASDecisionManagerUtilities\6.2**. From the Windows directory that contains the sasciutils.exe file, execute the command with the following syntax:

sasciutils -useraction DeleteEnvironmentSettings *-argument1 -argument2*

On UNIX, the sasciutils.sh file is typically installed in **./SASHome/ SASDecisionManagerUtilities/6.2**. From the UNIX directory that contains the sasciutils.sh file, execute the command with the following syntax:

sasciutils.sh -useraction DeleteEnvironmentSettings *-argument1 -argument2*

The command takes the following arguments:

-useraction DeleteEnvironmentSettings
specifies the Delete Environment Settings utility.

-userid
is the user ID of the account that is used to execute the utility.

-password
is the password for the user ID.

-server
is the host name for the SAS Metadata Repository.

-delete
deletes the environment setting.

Delete Environment Settings Utility Examples

The following example searches for an Environment Settings object on Windows and displays the results to the console:

```
sasciutils_console -useraction DeleteEnvironmentSettings
 -userid myuserid -password mypassword -server myhostname
```

The following example deletes an Environment Settings object on UNIX and writes the results to the log:

```
sasciutils.sh -useraction DeleteEnvironmentSettings
 -userid myuserid -password mypassword -server myhostname -delete
```

Load Treatments

Overview of the Load Treatments Utility

The Load Treatments utility loads internal SAS Decision Manager treatments from external tables. The utility reads treatments and their custom details from SAS data sets. This utility does not update treatments within flows.

If an error is returned for a particular treatment, that treatment is skipped. The utility continues to load the rest of the treatments.

You can run the Load Treatments utility more than once on the same tables. The utility updates existing treatments that have the same name and folder. If there any differences from the previous version of a treatment, the treatment version number is incremented.

The treatment name and folder name form the unique ID of a treatment. You cannot use the Load Treatments utility to rename a treatment or to move a treatment to a new folder. Instead, delete the treatment and run the Load Treatments utility to load the treatment with the changed name or folder.

If there are multiple entries with the same name or folder in the treatment table, the last entry overwrites the first entry. The version number might be incremented.

If columns in the input tables are wider than the associated column in the common data model, the values are truncated in the common data model.

How to Use the Load Treatments Utility

The Load Treatments utility is contained in the sasciutils.exe file on Windows and in the sasciutils.sh file on UNIX. To write the results to the console rather than to the log, use sasciutils_console.exe or sasciutils_console.sh.

On Windows, the sasciutils.exe files is typically installed in **`C:\Program Files \SASHome\SASDecisionManagerUtilities\6.2`**. From the Windows directory that contains the sasciutils.exe file, execute the command with the following syntax:

sasciutils -loadtreatments *-argument1 -argument2*

On UNIX, the sasciutils.sh file is typically installed in **`./SASHome/ SASDecisionManagerUtilities/6.2`**. From the UNIX directory that contains the sasciutils.sh file, execute the command with the following syntax:

sasciutils.sh -loadtreatments *-argument1 -argument2*

The command takes the following arguments:

-loadtreatments
specifies the Load Treatments utility

-userid
is the user ID of the account that is used to execute the utility.

-password
is the password for the user ID.

-bcname
is the name of the business context.

-libname
is the LIBNAME of the tables that contain the treatments.

-treatmenttablename
is the name of the treatment table. The treatment table, custom detail table, and treatments to delete table must be in the same library.

-customdetailtablename
is the name of the custom detail table. The treatment table, custom detail table, and treatments to delete table must be in the same library.

-treatmentstodeletetablename
deletes the specified treatment table. The treatment table, custom detail table, and treatments to delete table must be in the same library.

Treatment Tables

Treatment tables should contain the following fields.

Table 14.2 *Treatment Table Fields*

Field	Description
NAME	The name of the treatment. The name must be 60 characters or fewer in length. It cannot contain any front slashes (/), backslashes (\), or control characters.
FOLDER	The location relative to the business context root data folder. The specified folder must already exist. Otherwise, an error is returned for that treatment.
DESCRIPTION	The treatment description
CODE	The treatment code
ASSET_LINK_LABEL	The display value of the treatment reference
ASSET_LINK_URL	The URL of the treatment reference

Treatment Custom Detail Tables

Treatment custom detail tables should contain the following fields. For a particular treatment, the order of custom details in the table is the order of custom details in the loaded treatment. Multiple custom details with the same name, whether they have the same type, cannot be in the same treatment.

Table 14.3 *Treatment Custom Detail Table Fields*

Field	Description
TREATMENT_NAME	The name of the treatment. The name must be 60 characters or fewer in length. It cannot contain any front slashes (/), backslashes (\), or control characters.
TREATMENT_FOLDER	The location of the treatment relative to the business context root data folder. The specified folder must already exist. Otherwise, an error is returned for that treatment.
CUSTOM_DETAIL_NAME	The name of the treatment custom detail. The custom detail name must be unique within a treatment.
TYPE	The default values are defined in UtilResources.properties. See below for more information.

STRING_VALUE	This value is used as a default or initial value if the type is **Text** or **List**. If type is **List**, and multiple values are allowed, then this value is a list of strings that is separated by the list delimiter character that is specified in SAS Decision Manager. If the type is List, the value in the STRING_VALUE column must be in the specified list table. Otherwise, an error is returned.
DOUBLE_VALUE	The value is used as a default or initial value if the type is **Numeric** or **Check box**. If type is **Check box**, then zero is **false** and nonzero is **true**. An error is returned if the DOUBLE_VALUE column is not numeric.
DATE_VALUE	This value is used as the default or initial date value if the type is **Date**. An error is returned if the DATE_VALUE column is not numeric.
LINK_LABEL	This value is used as the default or initial display value if the type is **Link.**
LINK_URL	This value is used as the default or initial URL value if type is **Link**.
REQUIRED_FLAG	Specifies whether a value is value required for this custom detail. **Y** specifies that the value is required. Otherwise, the value is not required. If the custom detail is static, the REQUIRED_FLAG is set to **Y**.
COLUMN_NAME	The column name used for this custom detail in the UDF extension tables in the common data model. The values in the COLUMN_NAME field of the custom detail have the same restrictions as when the treatment is created in SAS Decision Manager. The length of a name must be between 1 and 60 characters. Names cannot include leading or trailing blank spaces, front slash (/) characters, backslash (\) characters, or any control characters.
TAG_NAME	Specifies whether a custom detail tag name is associated with this custom detail. **Y** specifies that a tag name is associated. Otherwise, a tag name is not associated.

STATIC_FLAG	Specifies whether a custom detail is static or dynamic. If the custom detail is dynamic, the value of the custom detail can be overridden. **`Y`** specifies that the custom detail is static. Otherwise, the custom detail is dynamic. An error is returned if a UDF is static and has no default value. If the custom detail is static, the REQUIRED_FLAG is set to **`Y`**.
ALLOW_MULTIPLE_LIST_VALUES_FLAG	If the type is **`List`**, this flag determines whether multiple items can be selected as the value of this custom detail. **`Y`** specifies that multiple items can be selected. Otherwise, multiple items cannot be selected.
DYNAMIC_LIST_FLAG	If the type is **`List`**, this flag determines whether the available values for the list are read from a table each time that the value is selected or only read in once.
LIST_LIB_NAME, LIST_TABLE_NAME, LIST_VALUE_COLUMN, LIST_DISPLAY_VALUE_COLUMN	This field is used if the type is **`List`**. This field specifies where to read the values for the list. The values in the LIST_VALUE_COLUMN must be unique.

The values for the **Type** field in the Treatment Custom Detail table are defined in **`CIUtilities/Source/Java/com/sas/analytics/crm/util/client/UtilResources.properties`**. The default values are as follows.

```
Utilities.LoadTreatments.numericType.txt = Numeric
Utilities.LoadTreatments.dateType.txt = Date
Utilities.LoadTreatments.textType.txt = Text
Utilities.LoadTreatments.checkboxType.txt = Check box
Utilities.LoadTreatments.listType.txt = List
Utilities.LoadTreatments.linkType.txt = Link
```

Type values are case-sensitive.

Treatments to Delete Table

The Treatments to Delete table should contain the following fields.

Field	**Description**
NAME	The name of the treatment. The name must be 60 characters or fewer in length. It cannot contain any front slashes (/), backslashes (\), or control characters.

FOLDER	The location of the treatment relative to the business context root data folder. The specified folder must already exist. Otherwise, an error is returned for that treatment.

List Values Tables

In the list values tables, the name of the list value column must match the value of LIST_VALUE_COLUMN in the custom detail table. The name of the display value column must match the value of LIST_DISPLAY_VALUE_COLUMN in the custom detail table.

Load Treatment Utility Example

The following example loads the treatments from a CI_TREATMENT_IMPORT table that is in a library named TREATMENTLIBNAME. The CI_TREATMENT_CUST_DTLS table is also loaded. The results are printed to load_treatments.out.

```
sasciutils -loadtreatments -userid myuserid -password
mypassword -bcname mybusinesscontext -libname TREATMENTLIBNAME
-treatmenttablename CI_TREATMENT_IMPORT -customdetailtablename
CI_TREATMENT_CUST_DTLS > load_treatments.out
```

Scripts for Managing SAS Decision Manager Flows

Overview of Scripts

The scripts in this section can be used to automate SAS Decision Manager flow operations. The Linux scripts are for the Bash shell only. The scripts might not work in other command shells in the Linux or UNIX environment.

Note: Because these scripts will be overwritten by later releases, make a copy of the scripts before you edit them.

Vars Script

Overview of the Vars Script

The **vars** script contains all of the environment variables that are needed by the rest of the scripts. This script is called by the **run** script. For more information, see "Run Script" on page 174.

Windows Environment

Copy and paste the following code into a text file named vars.cmd. Edit the code to make sure that the pathnames and settings are correct for your environment.

```
@Echo off
```

```
Rem /*****************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                         */
Rem /*             All Rights Reserved.                              */
Rem /*  This software is protected by copyright laws and             */
Rem /*               international treaties                          */
Rem /*                                                               */
Rem /*            U.S. Government Restricted Rights                  */
Rem /* Use, duplication, or disclosure of this software and related  */
Rem /*  documentation by the U.S. government is subject to the       */
Rem /*   Agreement with SAS Institute and the restrictions set forth */
Rem /*  in FAR 52.227-19, Commercial Computer Software -             */
Rem /*              Restricted Rights ( June 1987 )                  */
Rem /*                                                               */
Rem /*****************************************************************/

Rem /*****************************************************************/
Rem /*                       Do not edit as                          */
Rem /*    this script will be overwritten by subsequent releases.    */
Rem /*                                                               */
Rem /*****************************************************************/


Rem ----------- Variables used by the scripts ---------------

Rem Enter the user information
Set USERNAME=[user name]
Set USERID=[user id]
Set PASSWORD=[password]

Rem These could be different to above as administrator rights are needed for malauncher
Set MALAUNCHER_USER=[user id]
Set MALAUNCHER_PWD=[passsword]

Rem Ensure values match your environment and set accordingly
Set SAS_HOME=C:\Program Files\SASHOME
Set SAS_FRAMEWORK=%SAS_HOME%\SASPlatformObjectFramework\9.3
Set HOST=[sas tier]
Set PORT=8561
Set EXPORT_FROM_REPOSITORY=/System/Applications/SAS Decision Services/Decision Services 5.6/
SASDSDesignRepository
Set IMPORT_TO_REPOSITORY=/System/Applications/SAS Decision Services/Decision Services 5.6/
SASDSEngineRepository
Set MALAUNCHER_PATH="%SAS_HOME%/SASDecisionManagerLauncher/6.2"
Set MALAUNCHER_BC_NAME=[Business Context Name]
Set BATCH_ACTIVATOR_PATH=%SAS_HOME%/SASDecisionServicesServerConfiguration/5.6

Rem Change these settings if you wish the log files and packages to be located elsewhere
Set WORKING_FOLDER=%USERPROFILE%
SET LOGSTAMP=%date:~-4,4%%date:~-10,2%%date:~-7,2%-%time:~0,2%%time:~3,2%%time:~6,2%
Set LOG_FOLDER=%WORKING_FOLDER%\logs
Set PACKAGES_FOLDER=%WORKING_FOLDER%\packages
Set ERRORCODE=-987
Set CURRENT=%CD%

Rem Do not change
```

```
Set NON_SYSTEM=CICampaign_Decision,CICampaignGroup,CICellNode,CICommunicationNode,
CICustomDetailsGroup,CICustomDetailsTag,CIDiagram,CIDocument,CITreatment,CITreatmentCampaignSet,
CIDecisionEvent,CIDecisionProcess
Set SYSTEM_DESIGN_REPOSITORY=System/Applications/SAS Decision Services/Decision Services 5.6/
SASDSDesignRepository
Set SYSTEM_RTDM_VARIABLE=%SYSTEM_DESIGN_REPOSITORY%
Set SYSTEM_RTDM_EVENTS=%SYSTEM_DESIGN_REPOSITORY%
Set SYSTEM_RTDM_ACTIVITY=%SYSTEM_DESIGN_REPOSITORY%
Set SYSTEM_RTDM_RESOURCE=%SYSTEM_DESIGN_REPOSITORY%
```

Vars Script Parameters in the Windows Environment

In the Windows environment, the **vars** script contains these parameters:

EXPORT_FROM_REPOSITORY
specifies the Decision Services repository path that you want to promote from.

IMPORT_TO_REPOSITORY
specifies the Decision Services repository path that you want to promote to.

WORKING_FOLDER
is set to the user's workspace area, **%USERPROFILE%**. Edit this setting if you want to use a different working folder.

NON_SYSTEM
lists the objects that are saved in non-system folders.

SYSTEM_*XXXXX*
specifies the paths for the objects that are saved in system folders.

MALAUNCHER_USER
is the user ID that is authorized to execute the launcher. Edit this setting to be the user ID for your site.

MALAUNCHER_PWD
is the password for the **MALAUNCHER_USER** userid. Edit this setting to be the correct password.

Linux Environment

Copy and paste the following code into a text file named vars.sh. Edit the code to make sure that the paths and settings are correct for your environment.

```
#!/bin/bash
# /*****************************************************************/
# /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
# /*                  USA.                                         */
# /*             All Rights Reserved.                              */
# /*  This software is protected by copyright laws and             */
# /*               international treaties                          */
# /*                                                               */
# /*            U.S. Government Restricted Rights                  */
# /* Use, duplication, or disclosure of this software and related  */
# /*  documentation by the U.S. government is subject to the       */
# /*   Agreement with SAS Institute and the restrictions set forth */
# /*  in FAR 52.227-19, Commercial Computer Software -             */
# /*              Restricted Rights ( June 1987 )                  */
# /*                                                               */
# /*****************************************************************/

# /*****************************************************************/
```

```
# /*                      Do not edit as                          */
# /*    this script will be overwritten by subsequent releases.    */
# /*                                                              */
# /****************************************************************/

set +v

# ----------- Variables used by the scripts ---------------
USERNAME=[user name]
USERID=[user id]
PASSWORD=[password]

# These could be different to above as administrator rights are needed for malauncher
MALAUNCHER_USER=[user id]
MALAUNCHER_PWD=[password]

# Ensure values match your environment and set accordingly
SAS_HOME="/data/SASHome"
SAS_FRAMEWORK="$SAS_HOME/SASPlatformObjectFramework/9.3"
HOST=[sas tier]
PORT=8561
EXPORT_FROM_REPOSITORY="/System/Applications/SAS Decision Services/
Decision Services 5.6/SASDSDesignRepository"
IMPORT_TO_REPOSITORY="/System/Applications/SAS Decision Services/
Decision Services 5.6/SASDSEngineRepository"
MALAUNCHER_PATH="$SAS_HOME/SASDecisionManagerLauncher/6.2"
MALAUNCHER_BC_NAME=[Business Context Name]
BATCH_ACTIVATOR_PATH="$SAS_HOME/SASDecisionServicesServerConfiguration/5.6"

# Change these settings if you wish the log files and packages to be located elsewhere
WORKING_FOLDER=[working folder location]
LOGSTAMP=$(date +%Y%m%d-%H%M)
LOG_FOLDER="$WORKING_FOLDER/logs"
PACKAGES_FOLDER="$WORKING_FOLDER/packages"
TMPDIR="."
ERRORON="yes"

# Do not change
NON_SYSTEM="CICampaign_Decision,CICampaignGroup,CICellNode,CICommunicationNode,
CICustomDetailsGroup,CICustomDetailsTag,CIDiagram,CIDocument,CITreatment,
CITreatmentCampaignSet,CIDecisionEvent,CIDecisionProcess"
SYSTEM_DESIGN_REPOSITORY="System/Applications/SAS Decision Services/
Decision Services 5.6/SASDSDesignRepository"
SYSTEM_RTDM_VARIABLE=$SYSTEM_DESIGN_REPOSITORY
SYSTEM_RTDM_EVENTS=$SYSTEM_DESIGN_REPOSITORY
SYSTEM_RTDM_ACTIVITY=$SYSTEM_DESIGN_REPOSITORY
SYSTEM_RTDM_RESOURCE=$SYSTEM_DESIGN_REPOSITORY
```

Vars Script Parameters in the Linux Environment

In the Linux environment, the Vars script contains these parameters:

EXPORT_FROM_REPOSITORY
specifies the Decision Services repository path that you want to promote from.

IMPORT_TO_REPOSITORY
specifies the Decision Services repository path that you want to promote to.

WORKING_FOLDER
Edit this setting to be the correct working folder.

NON_SYSTEM
lists the objects that are saved in non-system folders.

SYSTEM_*XXXXX*
specifies the paths for the objects that are saved in system folders.

MALAUNCHER_USER
is the user ID that is authorized to execute the launcher. Edit this setting to be the user ID for your site.

MALAUNCHER_PWD
is the password for the **MALAUNCHER_USER** user ID. Edit this setting to be the correct password.

Run Script

Overview of the Run Script

The **run** script executes the other scripts. This script can be called from the master script or directly from the command line.

Windows Environment

In the Windows environment, the **run** script uses the following syntax:

```
run scriptname listfile.txt packagefile
```

scriptname
is the name of the script to execute.

listfile.txt
is the file that contains the list of objects that are managed by the script.

packagefile
is the name of the package file. This parameter is used by the **export** script.

You can execute multiple scripts with a single **run** script.

In the following example, the RUN command executes the **import** script. The imported objects are listed in a file named importlist.txt.

```
run import importlist.txt
```

In the following example, the RUN command executes the **import**, **deploy**, and **test** scripts.

```
run import importlist.txt deploy deploylist.txt test testlist.txt
```

The objects are listed in files named importlist.txt, deploylist.txt, and testlist.txt.

Copy and paste the following code into a text file named run.cmd.

```
@Echo off
Rem /*****************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                         */
Rem /*             All Rights Reserved.                              */
Rem /*  This software is protected by copyright laws and             */
Rem /*               international treaties                          */
Rem /*                                                               */
```

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

Rem /*****************************************************************/
Rem /*                       Do not edit as                          */
Rem /*    this script will be overwritten by subsequent releases.    */
Rem /*                                                               */
Rem /*****************************************************************/



Rem Do not change. This is the script that calls the function scripts.

Rem Set up the variables needed by all scripts
call vars.cmd

Rem Create the log and packages folder if not already existing
if not exist %LOG_FOLDER% md %LOG_FOLDER%
if not exist %PACKAGES_FOLDER% md %PACKAGES_FOLDER%

Rem now loop around the command line parameters
Set ok=no
:MAINLOOP

If "%~1"=="" goto :finish
If "%~1"=="/?" goto :usage

If /I "%~1"=="export" goto :EXPORT
If /I "%~1"=="activate" goto :ACTIVATE
If /I "%~1"=="deactivate" goto :DEACTIVATE
Set ok=yes
Call %1.cmd %2
If %ERRORLEVEL% equ %ERRORCODE% goto :ERROR
Shift
Shift
Goto :MAINLOOP

:DEACTIVATE
:ACTIVATE
Set ok=yes
Call %1.cmd
If %ERRORLEVEL% equ %ERRORCODE% goto :ERROR
Shift
Goto :mainloop


:EXPORT
If "%~2"=="" goto :usage
If "%~3"=="" goto :usage
Set ok=yes
```

```
Call %1.cmd %1 %2 %3
If %ERRORLEVEL% equ %ERRORCODE% goto :ERROR
Shift
Shift
Shift
Goto :MAINLOOP

:FINISH
If /I %ok%==no goto :usage
Exit /B 0

:ERROR
Exit /B %ERRORCODE%

:usage
Echo Usage:
Echo   run [command] [parameter] [command] [parameter] [command [parameter] ... ...
Echo   [command] is one of export,import,deploy,test,promote,activate,deactivate
Echo   e.g.
Echo   run export exportList.txt exportPackage
Echo   run import importList.txt
Echo   run deploy deployList.txt
Echo   run test testList.txt
Echo   run promote
Echo   run activate
Echo   run deactivate
Echo   More than one command per run can be issued: e.g. run export exportList.txt
Echo exportPackage import importList.txt deploy deployList.txt test testList.txt Echo promote activate
```

Linux Environment

In the Linux environment, the **run** script uses the following syntax:

```
./run.sh scriptname listfile.txt packagefile
```

scriptname
is the name of the script.

listfile.txt
is a text file that contains the list of objects that are managed by the script.

packagefile
is a text file that contains the list of objects in a package. This parameter is used by the **export** script.

You can execute multiple scripts with a single **run** script.

In the following example, the RUN command executes the **import** script. The imported objects are listed in a file named importlist.txt.

```
./run.sh import importlist.txt
```

In the following example, the RUN command executes the **import**, **deploy**, and **test** scripts.

```
./run.sh import importlist.txt deploy deploylist.txt test testlist.txt
```

The objects are listed in files named importlist.txt, deploylist.txt, and testlist.txt.

Copy and paste the following code into a text file named run.sh.

```
#!/bin/bash
# /*****************************************************************/
# /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
# /*                  USA.                                         */
# /*             All Rights Reserved.                              */
# /*  This software is protected by copyright laws and             */
# /*               international treaties                          */
# /*                                                               */
# /*            U.S. Government Restricted Rights                  */
# /* Use, duplication, or disclosure of this software and related  */
# /*  documentation by the U.S. government is subject to the       */
# /*   Agreement with SAS Institute and the restrictions set forth */
# /*  in FAR 52.227-19, Commercial Computer Software -             */
# /*              Restricted Rights ( June 1987 )                  */
# /*                                                               */
# /*****************************************************************/

# /*****************************************************************/
# /*                       Do not edit as                          */
# /*    this script will be overwritten by subsequent releases.    */
# /*                                                               */
# /*****************************************************************/

# Do not change. This is the script that calls the function scripts.

function usage()
{
    echo "Usage:"
    echo   "run [command] [parameter] [command] [parameter] [command [parameter] ... ..."
    echo   "[command] is one of export,import,deploy,test,promote,activate,deactivate"
    echo   "e.g."
    echo   "run export exportList.txt exportPackage"
    echo   "run import importList.txt"
    echo   "run deploy deployList.txt"
    echo   "run test testList.txt"
    echo   "run promote"
    echo   "run activate"
    echo   "run deactivate"
    echo   "More than one command per run can be issued: e.g. run export exportList.txt
echo exportPackage import importList.txt deploy deployList.txt test testList.txt
echo promote activate"
    exit -1
}

# $1 command (export, import etc), $2 parameter, $3 parameter
# Set up the variables needed by all scripts
source ./vars.sh

# Create the log and packages folder if not already existing
if [ ! -d "$LOG_FOLDER" ];  then mkdir "$LOG_FOLDER"; fi
if [ ! -d "$PACKAGES_FOLDER" ]; then mkdir "$PACKAGES_FOLDER"; fi

if [ "$#" == "0" ]; then
    usage
fi
```

```
# now loop around the command line parameters
while (($#)); do
    if [ "$1" == "/?" ]; then
        usage
    else
        if [ "$1" == "export" ]; then
            source ./$1.sh "$2" "$3"
            shift
            shift
            shift
        else
            if [ "$1" == "activate" ] || [ "$1" == "deactivate" ] || [ "$1" == "promote" ];
then
                source ./$1.sh
                shift
            else
                source ./$1.sh "$2"
                shift
                shift
            fi
        fi
    fi
done

return
```

Export Script

Overview of the Export Script

The **export** script exports a list of objects into a package file.

Windows Environment

In the Windows environment, the **export** script uses the following syntax:

```
run export listfile.txt packagefile
```

listfile.txt
is the file that contains the list of objects that you want to export. The dependencies for each object will be included by the ExportPackage utility.

The list file contains the full path to each object and the object type. Here is an example:

```
/assets/flows/flow1,CICampaign_Decision
/assets/flows/flow2,CICampaign_Decision
/assets/flows/flow3,CICampaign_Decision
```

The flows and treatments are placed in a single package file. If you want to place the objects in multiple package files, use a separate **export** command and a separate list file for each package.

packagefile
is the name of the package file. Do not append .spk to the filename. The package file will be placed in the **packages** subfolder. within the working folder that is specified by vars.cmd.

Copy and paste the following code into a text file named export.cmd.

```
@Echo off
Rem /*****************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                         */
Rem /*             All Rights Reserved.                              */
Rem /*  This software is protected by copyright laws and             */
Rem /*               international treaties                          */
Rem /*                                                               */
Rem /*            U.S. Government Restricted Rights                  */
Rem /* Use, duplication, or disclosure of this software and related  */
Rem /*  documentation by the U.S. government is subject to the       */
Rem /*   Agreement with SAS Institute and the restrictions set forth */
Rem /*  in FAR 52.227-19, Commercial Computer Software -             */
Rem /*              Restricted Rights ( June 1987 )                  */
Rem /*                                                               */
Rem /*****************************************************************/

Rem /*****************************************************************/
Rem /*                       Do not edit as                          */
Rem /*    this script will be overwritten by subsequent releases.    */
Rem /*                                                               */
Rem /*****************************************************************/


Echo "*********** Starting Export... **********"

Rem Do not change. The code below reads in from a text file the object path and the object type.
Rem The name of the text file to read in and the name of the package file are passed in as
Rem parameters from the run script.
Set CURRENT=%CD%
Set RETURNCODE=0
Set objectList=
Setlocal ENABLEDELAYEDEXPANSION

Rem For each line in the text file, get the object path and type and construct the list used by the
Rem export package
For /F "usebackq skip=1 tokens=1,2,3,* delims==," %%X in ("%~2") do call :SUB "%%X" "%%Y" "%%Z"
If not %ERRORLEVEL% equ 0 goto :ERROR
Cd %SAS_FRAMEWORK%
ExportPackage -includeDep -host %HOST% -port %PORT% -password %PASSWORD% -user %USERID% -subprop
-package "%PACKAGES_FOLDER%\%~3.spk" -log "%LOG_FOLDER%\export_%~3_%LOGSTAMP%.log"
-objects %objectList%
If not %ERRORLEVEL% equ 0 goto :ERROR
Cd %CURRENT%
Echo "*********** Finished Export ***********"
Exit /B %RETURNCODE%

:SUB
Set objectList=!objectList! "%~1(%~2)"
Goto :eof

:ERROR
Echo An error has occurred in the export script
Cd %CURRENT%
Set RETURNCODE=%ERRORCODE%
Exit /B %ERRORCODE%
```

```
ENDLOCAL
```

Linux Environment

In the Linux environment, the **export** script uses the following syntax:

```
./run.sh export listfile.txt packagefile
```

listfile.txt
is the text file that contains the list of objects that you want to export. The dependencies for each object will be included by the ExportPackage utility.

The list file contains the full path to each object, and the object type. Here is an example:

```
/assets/flows/flow1,CICampaign_Decision
/assets/flows/flow2,CICampaign_Decision
/assets/flows/flow3,CICampaign_Decision
```

The flows and treatments are placed in a single package file. If you want to place the objects in multiple package files, use a separate **export** command and a separate list file for each package.

packagefile
is the name of the package file. Do not append .spk to the filename. The package file is placed in a subfolder named **packages**.

Copy and paste the following code into a text file named export.sh.

```
#!/bin/bash

function error() {
    if [ "$ERRORON" == "yes" ]; then
        echo Error: $1
        echo An export package error has occurred
        exit $1
    fi
}

set +x
echo Starting Export...
pushd .

filelines=`cat $1`
old_IFS=$IFS
IFS=$'\n'
skip="yes"
for line in $filelines ;
do
    pathToken=${line%,*}
    typeToken=${line#*,}
    if [ "$skip" == "no" ]; then
        objectList+="$pathToken($typeToken) "
    else
        skip="no"
    fi
done
IFS=$old_IFS
```

```
cd $SAS_FRAMEWORK
./ExportPackage -includeDep -host $HOST -port $PORT -password $PASSWORD
-user $USERID -subprop -package "$PACKAGES_FOLDER/$2.spk"
-log "$LOG_FOLDER/export_$2_$LOGSTAMP.log" -objects $objectList
if [ $? -ne 0 ]; then error $?; fi

popd
echo Finished Export
exit 0
```

Import Script

Overview of the Import Script

The **import** script imports a list of objects from package files.

Windows Environment

In the Windows environment, the **import** script uses the following syntax:

```
run import listfile.txt
```

listfile.txt contains a list of packages, object types, and the target folder. Here is an example:

```
myFlowPackage,CICampaign_Decision,/importedAssets/flows
myTreatmentPackage,CITreatment,/importedAssets/treatments
```

CAUTION:

The target folder must exist. If the target folder is different from the source folder, you might need to edit the file that is specified by the **subprop** parameter for some packages. View the log file to determine what actions you should take. For more information, see "Log Files" on page 201.

Copy and paste the following code into a file named import.cmd:

```
Echo off
Rem /*****************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                         */
Rem /*             All Rights Reserved.                              */
Rem /*  This software is protected by copyright laws and             */
Rem /*               international treaties                          */
Rem /*                                                               */
Rem /*            U.S. Government Restricted Rights                  */
Rem /* Use, duplication, or disclosure of this software and related  */
Rem /*  documentation by the U.S. government is subject to the       */
Rem /*   Agreement with SAS Institute and the restrictions set forth */
Rem /*  in FAR 52.227-19, Commercial Computer Software -             */
Rem /*              Restricted Rights ( June 1987 )                  */
Rem /*                                                               */
Rem /*****************************************************************/

Rem /*****************************************************************/
Rem /*                       Do not edit as                          */
Rem /*    this script will be overwritten by subsequent releases.    */
```

```
Rem /*                                                                  */
Rem /*********************************************************************/


Echo "*********** Starting Import... ***********"

Rem Do not change. The code below reads in from a text file the package name, the object type,
Rem and the destination path
Rem The name of the text file are passed in as parameters from the run script.
Set CURRENT=%CD%
Setlocal ENABLEDELAYEDEXPANSION
Set RETURNCODE=0
Rem For each line in the text file, get the package name,type, destination path and perform an
Rem import package
For /F "usebackq skip=1 tokens=1,2,3,* delims==," %%X in (%1) do call :SUB "%%X" "%%Y" "%%Z"
Echo "*********** Finished Import ***********"
Exit /B %RETURNCODE%

:SUB
Cd %SAS_FRAMEWORK%
If /I not %~2 equ CICampaign_Decision goto :NOTCAMPAIGN
Rem If it is a campaign we are importing we need to import objects seperately from the package as
Rem some reside in different places

ImportPackage -types "RTDMVariable" -newOnly -includeACL -preservePaths -host %HOST% -port %PORT%
-user %USERID% -password %PASSWORD% -subprop "%PACKAGES_FOLDER%\%~1.subprop"
-package "%PACKAGES_FOLDER%\%~1.spk" -log "%LOG_FOLDER%\import_%~1_%LOGSTAMP%.log"
-target "%SYSTEM_RTDM_VARIABLE%"
If not %ERRORLEVEL% equ 0 goto :ERROR
ImportPackage -types "RTDMResource" -newOnly -includeACL -preservePaths -host %HOST% -port %PORT%
-user %USERID% -password %PASSWORD% -subprop "%PACKAGES_FOLDER%\%~1.subprop"
-package "%PACKAGES_FOLDER%\%~1.spk" -log "%LOG_FOLDER%\import_%~1_%LOGSTAMP%.log"
-target "%SYSTEM_RTDM_RESOURCE%"
If not %ERRORLEVEL% equ 0 goto :ERROR
ImportPackage -types "RTDMActivity" -newOnly -includeACL -preservePaths -host %HOST% -port %PORT%
-user %USERID% -password %PASSWORD% -subprop "%PACKAGES_FOLDER%\%~1.subprop"
-package "%PACKAGES_FOLDER%\%~1.spk" -log "%LOG_FOLDER%\import_%~1_%LOGSTAMP%.log"
-target "%SYSTEM_RTDM_ACTIVITY%"
If not %ERRORLEVEL% equ 0 goto :ERROR
ImportPackage -types "RTDMEVENT" -newOnly -includeACL -preservePaths -host %HOST% -port %PORT%
-user %USERID% -password %PASSWORD% -subprop "%PACKAGES_FOLDER%\%~1.subprop"
-package "%PACKAGES_FOLDER%\%~1.spk" -log "%LOG_FOLDER%\import_%~1_%LOGSTAMP%.log"
-target "%SYSTEM_RTDM_EVENTS%"
If not %ERRORLEVEL% equ 0 goto :ERROR
ImportPackage -types "%NON_SYSTEM%" -newOnly -includeACL -preservePaths -host %HOST% -port %PORT%
-user %USERID% -password %PASSWORD% -subprop "%PACKAGES_FOLDER%\%~1.subprop"
-package "%PACKAGES_FOLDER%\%~1.spk" -log "%LOG_FOLDER%\import_%~1_%LOGSTAMP%.log"  -target "%~3"
If not %ERRORLEVEL% equ 0 goto :ERROR
Cd %CURRENT%
Goto :eof

:NOTCAMPAIGN
ImportPackage -types "%2" -newOnly -includeACL -preservePaths -host %HOST% -port %PORT%
-user %USERID% -password %PASSWORD% -subprop "%PACKAGES_FOLDER%\%~1.subprop"
-package "%PACKAGES_FOLDER%\%~1.spk" -log "%LOG_FOLDER%\import_%~1_%LOGSTAMP%.log"
-target "%~3"
```

```
If not %ERRORLEVEL% equ 0 goto :ERROR
Cd %CURRENT%
Goto :eof

:ERROR
Echo An error has occurred in the import script
Cd %CURRENT%
Set RETURNCODE=%ERRORCODE%
Exit /B %ERRORCODE%
```

Linux Environment

In the Linux environment, the **`import`** script uses the following syntax:

```
./run.sh import listfile.txt
```

listfile.txt contains a list of packages, object types, and the target folder. Here is an example:

```
myFlowPackage,CICampaign_Decision,/importedAssets/flows
myTreatmentPackage,CITreatment,/importedAssets/treatments
```

CAUTION:

The target folder must exist. If the target folder is different from the source folder, you might need to edit the file that is specified by the **`subprop`** parameter for some packages. View the log file to determine what actions you should take. For more information, see "Log Files" on page 201.

Copy and paste the following code into a file named import.sh:

```
#!/bin/bash
# /*****************************************************************/
# /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
# /*                  USA.                                         */
# /*             All Rights Reserved.                              */
# /*  This software is protected by copyright laws and             */
# /*               international treaties                          */
# /*                                                               */
# /*            U.S. Government Restricted Rights                  */
# /* Use, duplication, or disclosure of this software and related  */
# /*  documentation by the U.S. government is subject to the       */
# /*   Agreement with SAS Institute and the restrictions set forth */
# /*  in FAR 52.227-19, Commercial Computer Software -             */
# /*              Restricted Rights ( June 1987 )                  */
# /*                                                               */
# /*****************************************************************/

# /*****************************************************************/
# /*                       Do not edit as                          */
# /*    this script will be overwritten by subsequent releases.    */
# /*                                                               */
# /*****************************************************************/

# Do not change. The code below reads in from a text file the package name, the
# object type, and the destination path
# The name of the text file are passed in as parameters from the run script.

function error() {
    if [ "$ERRORON" == "yes" ]; then
```

```
        echo Error: $1
        echo An import package error has occurred
        exit -1
    fi
}

set +x
echo "*********** Starting Import... ***********"
pushd .
# Package name, object type, target
filelines=`cat $1`
old_IFS=$IFS
IFS=$'\n'
skip="yes"
cd $SAS_FRAMEWORK
for line in $filelines ;
do
    token=${line%,*}
    packageToken=${token%,*}
    typeToken=${token#*,}
    token=${line#*,}
    pathToken=${token#*,}
    if [ "$skip" == "no" ]; then
        if [ "$typeToken" == "CICampaign_Decision" ]; then
            echo Importing campaign and system resources
            ./ImportPackage -newOnly -includeACL -preservePaths -host $HOST -port $PORT
-user $USERID -password $PASSWORD -types "RTDMVariable"
-subprop "$PACKAGES_FOLDER/$packageToken.subprop"
-package $PACKAGES_FOLDER/$packageToken.spk
-log $LOG_FOLDER/import_$packageToken_$LOGSTAMP.log
-target "$SYSTEM_RTDM_VARIABLE"
            if [ $? -ne 0 ]; then error $?; fi
            ./ImportPackage -newOnly -includeACL -preservePaths -host $HOST -port $PORT
-user $USERID -password $PASSWORD -types "RTDMResource"
-subprop "$PACKAGES_FOLDER/$packageToken.subprop"
-package $PACKAGES_FOLDER/$packageToken.spk
-log $LOG_FOLDER/import_$packageToken_$LOGSTAMP.log
-target "$SYSTEM_RTDM_RESOURCE"
            if [ $? -ne 0 ]; then error $?; fi
            ./ImportPackage -newOnly -includeACL -preservePaths -host $HOST -port $PORT
-user $USERID -password $PASSWORD -types "RTDMActivity"
-subprop "$PACKAGES_FOLDER/$packageToken.subprop"
-package $PACKAGES_FOLDER/$packageToken.spk
-log $LOG_FOLDER/import_$packageToken_$LOGSTAMP.log
-target "$SYSTEM_RTDM_ACTIVITY"
            if [ $? -ne 0 ]; then error $?; fi
            ./ImportPackage -newOnly -includeACL -preservePaths -host $HOST -port $PORT
-user $USERID -password $PASSWORD -types "RTDMEVENT"
-subprop "$PACKAGES_FOLDER/$packageToken.subprop"
-package $PACKAGES_FOLDER/$packageToken.spk
-log $LOG_FOLDER/import_$packageToken_$LOGSTAMP.log  -target "$SYSTEM_RTDM_EVENTS"
            if [ $? -ne 0 ]; then error $?; fi
            ./ImportPackage -newOnly -includeACL -preservePaths -host $HOST
-port $PORT -user $USERID -password $PASSWORD -types $NON_SYSTEM
-subprop "$PACKAGES_FOLDER/$packageToken.subprop"
-package $PACKAGES_FOLDER/$packageToken.spk
```

```
-log $LOG_FOLDER/import_$packageToken_$LOGSTAMP.log  -target \"$pathToken\"
            if [ $? -ne 0 ]; then error $?; fi

        else
            echo Importing $typeToken
            ./ImportPackage -newOnly -includeACL -preservePaths -host $HOST -port $PORT
-user $USERID -password $PASSWORD -types $typeToken
-subprop "$PACKAGES_FOLDER/$packageToken.subprop"
-package $PACKAGES_FOLDER/$packageToken.spk
-log $LOG_FOLDER/import_$packageToken_$LOGSTAMP.log  -target \"$pathToken\"
            if [ $? -ne 0 ]; then error $?; fi
        fi
    else
        skip="no"
    fi
done
IFS=$old_IFS
popd
echo "*********** Finished Import ***********"
return
```

Deploy Script

Overview of the Deploy Script

The **deploy** script deploys a list of flows.

Windows Environment

In the Windows environment, the **deploy** script uses the following syntax:

```
run deploy listfile.txt
```

listfile.txt contains the path from the business context root and the type of object. Each line should contain the flow path and name, followed by **camp**, if the object is a flow, as in the following example:

```
\assets\flows\flow1,camp
\assets\flows\flow2,camp
```

The path separator is a backslash (\).

After the objects are deployed successfully, a file named deployedFlowList is created. This file contains a list of the names of the deployed flows. The deployedFlowList file can be input to the **promote** command or the **activate** command. The deployedFlowList file is placed in the user's **%TEMP%** folder. Edit the vars.cmd file if you want to place the deployedFlowList file in a different location.

Copy and paste the following code into a file named deploy.cmd.

```
@Echo off
Rem /*************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                     */
Rem /*             All Rights Reserved.                          */
Rem /*  This software is protected by copyright laws and         */
Rem /*               international treaties                      */
Rem /*                                                           */
Rem /*           U.S. Government Restricted Rights               */
```

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

Rem /*****************************************************************/
Rem /*                       Do not edit as                          */
Rem /*    this script will be overwritten by subsequent releases.    */
Rem /*                                                               */
Rem /*****************************************************************/


Echo "*********** Starting Deployment ***********"

Rem Do not change. The code below reads in from a text file the object path (campaign or set),
Rem and the type of object it is ( camp or set - malauncher terms ).
Rem The name of the text file is passed in as a parameter from the run script

Set CURRENT=%CD%
Set RETURNCODE=0
Setlocal ENABLEDELAYEDEXPANSION

Rem if the malauncher deployed flow list file exists, delete it first
If exist %TEMP%\deployedFlowList del /Q %TEMP%\deployedFlowList

Rem for each line in the text file, get the object path and type and call malauncher to mark it
Rem for deployment
For /F "usebackq skip=1 tokens=1,2,* delims==," %%X in (%1) do call :SUB %%Y "%%X"
CD %CURRENT%
Echo "*********** End Deployment ***********"
Exit /B %RETURNCODE%

:SUB
Cd %MALAUNCHER_PATH%
sasdcmlauncher -u %MALAUNCHER_USER% -p %MALAUNCHER_PWD% -n -v -q -x %MALAUNCHER_BC_NAME%
-d mark %1 "%~2"
If not %ERRORLEVEL% equ 0 goto :ERROR
Cd %CURRENT%
Goto :eof

:ERROR
Echo An error has occurred in the deploy script
Cd %CURRENT%
Set RETURNCODE=%ERRORCODE%
Exit /B %ERRORCODE%

Endlocal
```

Linux Environment

In the Linux environment, the **`deploy`** script uses the following syntax:

```
./run.sh deploy listfile.txt
```

listfile.txt contains the path from the business context root and the type of object. Each line should contain the path and name, followed by **camp**, as in the following example:

```
\assets\flows\flow1,camp
\assets\flows\flow2,camp
```

The path separator is a backslash (\).

After the objects are deployed successfully, a file named deployedFlowList is created. This file contains a list of the names of the deployed flows. The deployedFlowList file can be input to the **promote** command or the **activate** command. The deployedFlowList file is copied to the **temp** folder.

Copy and paste the following code into a text file named deploy.sh:

```
#!/bin/bash
# /*****************************************************************/
# /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
# /*                  USA.                                         */
# /*             All Rights Reserved.                              */
# /*  This software is protected by copyright laws and             */
# /*               international treaties                          */
# /*                                                               */
# /*            U.S. Government Restricted Rights                  */
# /* Use, duplication, or disclosure of this software and related  */
# /*  documentation by the U.S. government is subject to the       */
# /*   Agreement with SAS Institute and the restrictions set forth */
# /*  in FAR 52.227-19, Commercial Computer Software -             */
# /*              Restricted Rights ( June 1987 )                  */
# /*                                                               */
# /*****************************************************************/

# /*****************************************************************/
# /*                       Do not edit as                          */
# /*    this script will be overwritten by subsequent releases.    */
# /*                                                               */
# /*****************************************************************/

# Do not change. The code below reads in from a text file the object path (campaign or set),
# and the type of object it is ( camp or set - malauncher terms ).
# The name of the text file is passed in as a parameter from the run script


function error() {
    if [ "$ERRORON" == "yes" ]; then
        echo Error: $1
        echo An deploy error has occurred
        exit -1
    fi
}

set +x
echo "*********** Starting Deployment ***********"
pushd .

filelines=`cat $1`
old_IFS=$IFS
IFS=$'\n'
```

```
skip="yes"
echo ------
echo $MALAUNCHER_PATH
cd $MALAUNCHER_PATH
rm deployedFlowList

for line in $filelines ;
do
    pathToken=${line%,*}
    typeToken=${line#*,}
    if [ "$skip" == "no" ]; then
        ./sasdcmlauncher -u $MALAUNCHER_USER -p $MALAUNCHER_PWD -n -v -q -x $MALAUNCHER_BC_NAME
-d mark $typeToken $pathToken > $LOG_FOLDER/malauncher_deploy_$LOGSTAMP.log
        if [ $? -ne 0 ]; then error $?; fi
        else
        skip="no"
    fi
done
IFS=$old_IFS

echo "*********** End Deployment ***********"
popd
return
```

Test Script

Overview of the Test Script

The **test** script runs test cases against a list of flows.

Windows Environment

In the Windows environment, the **test** script uses the following syntax:

```
run test listfile.txt
```

listfile.txt contains the list of flows and test cases to run. The list of flows includes a partial path from the business context root, followed either by the test case name, or an asterisk (*) for all test cases. The following list is an example.

```
assets\flow1,Test Case 1
assets\flow2,*
assets\flow3,Test Case 3
```

Copy and paste the following code into a file named test.cmd:

```
@Echo off
Rem /*****************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                         */
Rem /*             All Rights Reserved.                              */
Rem /*  This software is protected by copyright laws and             */
Rem /*               international treaties                          */
Rem /*                                                               */
Rem /*            U.S. Government Restricted Rights                  */
Rem /* Use, duplication, or disclosure of this software and related  */
Rem /*  documentation by the U.S. government is subject to the       */
Rem /*   Agreement with SAS Institute and the restrictions set forth */
```

```
Rem /*  in FAR 52.227-19, Commercial Computer Software -            */
Rem /*              Restricted Rights ( June 1987 )                 */
Rem /*                                                              */
Rem /****************************************************************/

Rem /****************************************************************/
Rem /*                       Do not edit as                         */
Rem /*    this script will be overwritten by subsequent releases.   */
Rem /*                                                              */
Rem /****************************************************************/


Echo "*********** Running Test Cases... ***********"

Rem Do not change. The code below reads in from a text file the campaign path, and the test case
Rem name to run, or * to run all test cases
Rem The name of the text file is passed in as a parameter from the run script

Set CURRENT=%CD%
SET RETURNCODE=0
Setlocal ENABLEDELAYEDEXPANSION
Rem For each row in the text file, get the campaign name and test case name parameter, and then
Rem call malauncher
For /F "usebackq skip=1 tokens=1,2,3,* delims==," %%X in (%1) do call :SUB "%%X" "%%Y"
Cd %CURRENT%
Echo "*********** Finished running Test Cases ***********"
Exit /B %RETURNCODE%


:SUB
Cd %MALAUNCHER_PATH%
Rem Either run a specific test case or all test cases in the campaign
If /I "%~2"=="*" (sasdcmlauncher -u %MALAUNCHER_USER% -p %MALAUNCHER_PWD% -n -v -q
-x %MALAUNCHER_BC_NAME% -d tests camp "%~1" ) else (sasdcmlauncher -u %MALAUNCHER_USER%
-p %MALAUNCHER_PWD% -n -v -q -x %MALAUNCHER_BC_NAME% -d test camp "%~1" "%~2" )
If not %ERRORLEVEL% equ 0 goto :ERROR
Cd %CURRENT%
Goto :eof

:ERROR
Echo An error has occurred in the run tests script
Cd %CURRENT%
SET RETURNCODE=%ERRORCODE%
Exit /B %ERRORCODE%

Endlocal
```

Linux Environment

In the Linux environment, the **`test`** script uses the following syntax:

```
./run.sh test listfile.txt
```

listfile.txt contains the list of flows and test cases to run. The list of flows includes a partial path from the business context root, followed either by the test case name, or an asterisk (*) for all test cases. The following list is an example.

```
assets\flow1,Test Case 1
```

```
assets\flow2,*
assets\flow3,Test Case 3
```

Copy and paste the following code into a text file named test.sh:

```
#!/bin/bash
# /*******************************************************************/
# /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
# /*                  USA.                                           */
# /*             All Rights Reserved.                                */
# /*  This software is protected by copyright laws and               */
# /*               international treaties                            */
# /*                                                                 */
# /*            U.S. Government Restricted Rights                    */
# /* Use, duplication, or disclosure of this software and related  */
# /*  documentation by the U.S. government is subject to the       */
# /*   Agreement with SAS Institute and the restrictions set forth */
# /*  in FAR 52.227-19, Commercial Computer Software -             */
# /*              Restricted Rights ( June 1987 )                  */
# /*                                                                 */
# /*******************************************************************/

# /*******************************************************************/
# /*                       Do not edit as                            */
# /*    this script will be overwritten by subsequent releases.    */
# /*                                                                 */
# /*******************************************************************/

# Do not change. The code below reads in from a text file the campaign path, and the test case
# name to run, or * to run all test cases
# The name of the text file is passed in as a parameter from the run script

function error() {
    if [ "$ERRORON" == "yes" ]; then
        echo Error: $1
        echo An deploy error has occurred
        exit -1
    fi
}

set +x
echo "*********** Running Test Cases... ***********"
pushd .

filelines=`cat $1`
old_IFS=$IFS
IFS=$'\n'
skip="yes"
echo ------
echo $MALAUNCHER_PATH
cd $MALAUNCHER_PATH

for line in $filelines ;
do
    pathToken=${line%,*}
    testNameToken=${line#*,}
    if [ "$skip" == "no" ]; then
```

```
        if [ "$testNameToken" == "*" ]; then
            ./sasdcmlauncher -u $MALAUNCHER_USER -p $MALAUNCHER_PWD -n -v -q -x $MALAUNCHER_BC_NAME
-d tests camp $pathToken > malauncher_testcases_$LOGSTAMP.log
        else
            ./sasdcmlauncher -u $MALAUNCHER_USER -p $MALAUNCHER_PWD -n -v -q -x $MALAUNCHER_BC_NAME
-d test camp $pathToken $testNameToken > malauncher_testcases_$LOGSTAMP.log
        fi
        if [ $? -ne 0 ]; then error $?; fi
        else
        skip="no"
    fi
done
IFS=$old_IFS

popd
echo "*********** Finished running Test Cases ***********"
return
```

Promote Script

Overview of the Promote Script

The **promote** script moves decision flows from one design server repository to another repository, (for example, from the design repository to the production repository).

Windows Environment

In the Windows environment, the **promote** script uses the following syntax:

```
run promote
```

The names of the repositories are specified in the vars.cmd file.

Copy and paste the following code into a file named promote.cmd:

```
@Echo off
Rem /*****************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                         */
Rem /*             All Rights Reserved.                              */
Rem /*  This software is protected by copyright laws and             */
Rem /*               international treaties                          */
Rem /*                                                               */
Rem /*            U.S. Government Restricted Rights                  */
Rem /* Use, duplication, or disclosure of this software and related  */
Rem /*  documentation by the U.S. government is subject to the       */
Rem /*   Agreement with SAS Institute and the restrictions set forth */
Rem /*  in FAR 52.227-19, Commercial Computer Software -             */
Rem /*              Restricted Rights ( June 1987 )                  */
Rem /*                                                               */
Rem /*****************************************************************/

Rem /*****************************************************************/
Rem /*                       Do not edit as                          */
Rem /*    this script will be overwritten by subsequent releases.    */
Rem /*                                                               */
Rem /*****************************************************************/
```

```
Echo "*********** Starting Promotion ***********"

Rem Do not change. The code below reads in from the malauncher generated deployedFlowList text
Rem file the id of the deployed flow

Set CURRENT=%CD%
Set RETURNCODE=0

Setlocal ENABLEDELAYEDEXPANSION

if not exist "%TEMP%\deployedFlowList" goto :NO_FILE
copy "%TEMP%\deployedFlowList" "%TEMP%\flowList.txt"

Rem For each row in the file, get the id and export the flow from the design repository and then
Rem import back into the production repository
For /F "usebackq tokens=1,2,* delims==," %%X in (%TEMP%\flowList.txt) do call :SUB %%X
Cd %CURRENT%
Echo "*********** End Promotion ***********"
Exit /B %RETURNCODE%

:SUB
Cd %SAS_FRAMEWORK%
Set OBJTYPE=RTDMFlow
ExportPackage -includeDep -subprop -host %HOST% -port %PORT% -password %PASSWORD% -user %USERID%
-subprop -package "%PACKAGES_FOLDER%\%~1.spk" -log "%LOG_FOLDER%\promote_export_%~1_%LOGSTAMP%.log"
 -objects "%EXPORT_FROM_REPOSITORY%/%~1(%OBJTYPE%)"
If not %ERRORLEVEL% equ 0 goto :ERROR
ImportPackage -host %HOST% -port %PORT% -password %PASSWORD% -user %USERID%
-subprop "%PACKAGES_FOLDER%\%~1.subprop"  -package "%PACKAGES_FOLDER%\%~1.spk"
-log "%LOG_FOLDER%\promote_import_%~1_%LOGSTAMP%.log"  -target "%IMPORT_TO_REPOSITORY%"
If not %ERRORLEVEL% equ 0 goto :ERROR
Cd %CURRENT%
Goto :eof

:NO_FILE
Echo Cannot promote flows: File "deployedFlowList" cannot be found. This file contains the list of
Echo deployed flows and is generated by malauncher when a campaign is marked for deployment using
Echo malauncher.

:ERROR
Echo An error has occurred in the promotion script
Cd %CURRENT%
Set RETURNCODE=%ERRORCODE%
Exit /B %ERRORCODE%
endlocal
```

Linux Environment

In the Linux environment, the **`promote`** script uses the following syntax:

```
./run.sh promote
```

The names of the repositories are specified in the vars.sh file.

Copy and paste the following code into a file named promote.sh:

```
#!/bin/bash
# /*****************************************************************/
# /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
# /*                  USA.                                         */
# /*             All Rights Reserved.                              */
# /*  This software is protected by copyright laws and             */
# /*               international treaties                          */
# /*                                                               */
# /*            U.S. Government Restricted Rights                  */
# /* Use, duplication, or disclosure of this software and related  */
# /*  documentation by the U.S. government is subject to the       */
# /*   Agreement with SAS Institute and the restrictions set forth */
# /*  in FAR 52.227-19, Commercial Computer Software -             */
# /*              Restricted Rights ( June 1987 )                  */
# /*                                                               */
# /*****************************************************************/

# /*****************************************************************/
# /*                        Do not edit as                         */
# /*    this script will be overwritten by subsequent releases.    */
# /*                                                               */
# /*****************************************************************/

# Do not change. The code below reads in from the malauncher generated
# deployedFlowList text file the id of the deployed flow

function error() {
    if [ "$ERRORON" == "yes" ]; then
        echo Error: $1
        echo A deploy error has occurred
        exit -1
    fi
}

pushd .
echo "*********** Starting Promotion ***********"
if [ -f $MALAUNCHER_PATH/deployedFlowList ]; then

    cd $SAS_FRAMEWORK
    filelines=`cat $MALAUNCHER_PATH/deployedFlowList`
    old_IFS=$IFS
    IFS=$'\n'
    for line in $filelines ;
    do
        id=${line%,*}
        ExportPackage -includeDep -subprop -host $HOST -port $PORT -password $PASSWORD
-user $USERID -package "$PACKAGES_FOLDER/$id.spk"
-log "$LOG_FOLDER/promote_export_$id_$LOGSTAMP.log"
-objects "$EXPORT_FROM_REPOSITORY/$id(RTDMFlow)"
        ImportPackage -subprop "$PACKAGES_FOLDER/$id.subprop" -host $HOST -port $PORT
-password $PASSWORD -user $USERID -package "$PACKAGES_FOLDER/$id.spk"
-log "$LOG_FOLDER/promote_import_$id_$LOGSTAMP.log"  -target "$IMPORT_TO_REPOSITORY"
    done
    IFS=$old_IFS
popd
```

```
else
    echo Cannot promote flows: File "deployedFlowList" cannot be found. This file contains
echo the list of deployed flows and is generated by malauncher when a campaign is
echo marked for deployment using malauncher.
    popd
    exit -1
fi

echo "*********** End Promotion ***********"
return
```

Activate and Deactivate Scripts

Overview of Activate and Deactivate Scripts

The **activate** and **deactivate** scripts activate and deactivate SAS Decision Services flows in the production repository.

Windows Environment

In the Windows environment, the **activate** and **deactivate** scripts use the following syntax:

```
run activate
run deactivate
```

There are no parameters. This script uses the deployedFlowList file that is generated by the **deploy** script.

Copy and paste the following code into a text file that is named activate.cmd:

```
@Echo off
Rem /*****************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                         */
Rem /*             All Rights Reserved.                              */
Rem /*  This software is protected by copyright laws and             */
Rem /*               international treaties                          */
Rem /*                                                               */
Rem /*            U.S. Government Restricted Rights                  */
Rem /* Use, duplication, or disclosure of this software and related  */
Rem /*  documentation by the U.S. government is subject to the       */
Rem /*   Agreement with SAS Institute and the restrictions set forth */
Rem /*  in FAR 52.227-19, Commercial Computer Software -             */
Rem /*              Restricted Rights ( June 1987 )                  */
Rem /*                                                               */
Rem /*****************************************************************/

Rem /*****************************************************************/
Rem /*                       Do not edit as                          */
Rem /*    this script will be overwritten by subsequent releases.    */
Rem /*                                                               */
Rem /*****************************************************************/


call batchactivator.cmd a
```

Copy and paste the following code into a text file that is named deactivate.cmd:

```
@Echo off
Rem /*****************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                         */
Rem /*             All Rights Reserved.                              */
Rem /*  This software is protected by copyright laws and             */
Rem /*               international treaties                          */
Rem /*                                                               */
Rem /*            U.S. Government Restricted Rights                  */
Rem /* Use, duplication, or disclosure of this software and related  */
Rem /*  documentation by the U.S. government is subject to the       */
Rem /*   Agreement with SAS Institute and the restrictions set forth */
Rem /*  in FAR 52.227-19, Commercial Computer Software -             */
Rem /*              Restricted Rights ( June 1987 )                  */
Rem /*                                                               */
Rem /*****************************************************************/

Rem /*****************************************************************/
Rem /*                       Do not edit as                          */
Rem /*    this script will be overwritten by subsequent releases.    */
Rem /*                                                               */
Rem /*****************************************************************/


call batchactivator.cmd d
```

Copy and paste the following code into a text file that is named batchactivator.cmd:

```
@Echo off
Rem /*****************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                         */
Rem /*             All Rights Reserved.                              */
Rem /*  This software is protected by copyright laws and             */
Rem /*               international treaties                          */
Rem /*                                                               */
Rem /*            U.S. Government Restricted Rights                  */
Rem /* Use, duplication, or disclosure of this software and related  */
Rem /*  documentation by the U.S. government is subject to the       */
Rem /*   Agreement with SAS Institute and the restrictions set forth */
Rem /*  in FAR 52.227-19, Commercial Computer Software -             */
Rem /*              Restricted Rights ( June 1987 )                  */
Rem /*                                                               */
Rem /*****************************************************************/

Rem /*****************************************************************/
Rem /*                       Do not edit as                          */
Rem /*    this script will be overwritten by subsequent releases.    */
Rem /*                                                               */
Rem /*****************************************************************/


Echo "*********** BatchActivator ***********"
Rem Do not change. The code below reads passes the malauncher generated deployedFlowList text file
Rem into the batch activator utility
```

```
Set CURRENT=%CD%
Set RETURNCODE=0
If not exist "%TEMP%\deployedFlowList" goto :NO_DEPLOYED_FILE
Copy "%TEMP%\deployedFlowList" "%TEMP%\flowList.txt"

Cd %BATCH_ACTIVATOR_PATH%
BatchActivator -%1 -host %HOST% -port %PORT% -password %PASSWORD% -user %USERID% -domain DefaultAuth
 -log "%LOG_FOLDER%\%1_batchactivator_%LOGSTAMP%.log" -f "%TEMP%\flowList.txt"

If not %ERRORLEVEL% equ 0 goto :ERROR
Cd %CURRENT%
Exit /B %RETURNCODE%

:ERROR
Echo An error has occurred in the batch activator script
Cd %CURRENT%
Set RETURNCODE=%ERRORCODE%
Exit /B %ERRORCODE%

:NO_DEPLOYED_FILE
Echo File "deployedFlowList" cannot be found in current folder. This file contains the list of
deployed flows and is generated by malauncher when a campaign is marked for deployment using
malauncher.
Cd %CURRENT%
Set RETURNCODE=%ERRORCODE%
Exit /B %ERRORCODE%

Endlocal
```

Linux Environment

In the Linux environment, the **`activate`** and **`deactivate`** scripts use the following syntax:

```
./run.sh activatee
./run.sh deactivate
```

There are no parameters. This script uses the deployedFlowList file that is generated by the **`deploy`** script.

Copy and paste the following code into a text file that is named activate.sh:

```
#!/bash/bin
# /*****************************************************************/
# /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
# /*                  USA.                                         */
# /*             All Rights Reserved.                              */
# /*  This software is protected by copyright laws and             */
# /*               international treaties                          */
# /*                                                               */
# /*            U.S. Government Restricted Rights                  */
# /* Use, duplication, or disclosure of this software and related  */
# /*  documentation by the U.S. government is subject to the       */
# /*   Agreement with SAS Institute and the restrictions set forth */
# /*  in FAR 52.227-19, Commercial Computer Software -             */
# /*              Restricted Rights ( June 1987 )                  */
# /*                                                               */
```

```
# /*****************************************************************/

# /*****************************************************************/
# /*                       Do not edit as                          */
# /*    this script will be overwritten by subsequent releases.    */
# /*                                                               */
# /*****************************************************************/

set -x
source ./batchactivator.sh a
```

Copy and paste the following code into a text file that is named deactivate.sh:

```
#!/bin/bash
# /*****************************************************************/
# /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
# /*                  USA.                                         */
# /*             All Rights Reserved.                              */
# /*  This software is protected by copyright laws and             */
# /*               international treaties                          */
# /*                                                               */
# /*            U.S. Government Restricted Rights                  */
# /* Use, duplication, or disclosure of this software and related  */
# /*  documentation by the U.S. government is subject to the       */
# /*   Agreement with SAS Institute and the restrictions set forth */
# /*  in FAR 52.227-19, Commercial Computer Software -             */
# /*              Restricted Rights ( June 1987 )                  */
# /*                                                               */
# /*****************************************************************/

# /*****************************************************************/
# /*                       Do not edit as                          */
# /*    this script will be overwritten by subsequent releases.    */
# /*                                                               */
# /*****************************************************************/

set -x
source ./batchactivator.sh d
```

Copy and paste the following code into a text file that is named batchactivator.sh:

```
#!/bin/bash
# /*****************************************************************/
# /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
# /*                  USA.                                         */
# /*             All Rights Reserved.                              */
# /*  This software is protected by copyright laws and             */
# /*               international treaties                          */
# /*                                                               */
# /*            U.S. Government Restricted Rights                  */
# /* Use, duplication, or disclosure of this software and related  */
# /*  documentation by the U.S. government is subject to the       */
# /*   Agreement with SAS Institute and the restrictions set forth */
# /*  in FAR 52.227-19, Commercial Computer Software -             */
# /*              Restricted Rights ( June 1987 )                  */
# /*                                                               */
# /*****************************************************************/
```

```
# /*****************************************************************/
# /*                       Do not edit as                          */
# /*    this script will be overwritten by subsequent releases.    */
# /*                                                               */
# /*****************************************************************/


#Do not change. The code below reads passes the malauncher generated
# deployedFlowList text file into the batch activator utility

function error() {
    if [ "$ERRORON" == "yes" ]; then
        echo Error: $1
        echo An batchactivator package error has occurred
        exit -1
    fi
}

set -x
echo "*********** BatchActivator ***********"
pushd .

if [ -f $MALAUNCHER_PATH/deployedFlowList ]; then
    cp $MALAUNCHER_PATH/deployedFlowList $MALAUNCHER_PATH/flowList.txt
    cd "$BATCH_ACTIVATOR_PATH"
    BatchActivator -$1 -host $HOST -port $PORT -password $PASSWORD -user $USERID
-domain DefaultAuth -log $LOG_FOLDER/$1_batchactivator_$LOGSTAMP.log
-f $MALAUNCHER_PATH/flowList.txt

    if [ $? -ne 0 ]; then error $?; fi
else
    echo File "deployedFlowList" cannot be found in $MALAUNCHER_PATH.
echo This file contains the list of deployed
echo flows and is generated by malauncher when a campaign is marked for
echo deployment using malauncher.
    popd
    exit -1
fi
popd
echo "*********** End BatchActivator ***********"
return
```

Master Script

Overview of the Master Script

The master script calls the individual scripts.

Windows Environment

In the Windows environment, the master script uses the following syntax:

```
run master.cmd
```

master.cmd is the name that you provide for the master script. Each command script within the master script is called by the **run** script:

```
call run command parameters
```

The following code is an example master script. Copy and paste the code into a text file whose name ends in .cmd. Remove the REM comments from the commands that you want to execute, and edit the parameters.

```
@Echo off
Rem /*****************************************************************/
Rem /* Copyright (c) 2007 - 2013 by SAS Institute Inc., Cary, NC,    */
Rem /*                  USA.                                         */
Rem /*             All Rights Reserved.                              */
Rem /*  This software is protected by copyright laws and             */
Rem /*               international treaties                          */
Rem /*                                                               */
Rem /*            U.S. Government Restricted Rights                  */
Rem /* Use, duplication, or disclosure of this software and related  */
Rem /*  documentation by the U.S. government is subject to the       */
Rem /*   Agreement with SAS Institute and the restrictions set forth */
Rem /*  in FAR 52.227-19, Commercial Computer Software -             */
Rem /*              Restricted Rights ( June 1987 )                  */
Rem /*                                                               */
Rem /*****************************************************************/

Rem /*****************************************************************/
Rem /*                       Do not edit as                          */
Rem /*    this script will be overwritten by subsequent releases.    */
Rem /*                                                               */
Rem /*****************************************************************/

Echo ***********************************************************************************
Echo This is an example master (umbrella) script, that calls the discrete function scripts.
Echo Amend this script to call the functions you wish to perform in one step.
Echo Should one of the scripts fail, then the master script stops.
Echo Pass /nopromote not to promote
Echo ***********************************************************************************

Echo In this script the default template parameter files are used. You will need to enter the correct
Echo path and object information in these parameter files ( See each one for usage ).
Setlocal ENABLEDELAYEDEXPANSION

Rem First export the assets into a package file, i.e. export campaigns, treatments etc.
Rem run is the main calling script. ( run.cmd or run.sh )
Rem export is the discrete function script ( export.cmd or export.sh )
Rem exportFile.txt lists the objects and types to export. Any valid text file will do.
Rem myPackage is the name of the package file to export the objects into. Omit the .spk
Call run export exportList.txt myPackage
If %ERRORLEVEL% equ %ERRORCODE% goto :ERROR

echo Campaign has been exported into myPackage.spk
echo Press a key to start import,test,mark,promote and activate steps
pause


Rem Import the objects from the package(s)
Rem run is the main calling script. ( run.cmd or run.sh )
Rem import is the discrete function script ( import.cmd or import.sh )
Rem importFile.txt lists the objects and packages to import from. Any valid text file will do.
```

```
Call run import importList.txt
If %ERRORLEVEL% equ %ERRORCODE% goto :ERROR

Rem Run test case(s) in the campaign(s)
Rem run is the main calling script. ( run.cmd or run.sh )
Rem test is the discrete function script ( test.cmd or test.sh )
Rem testFile.txt lists the campaign(s) and test(s) to run. Any valid text file will do.
Call run test testList.txt
If %ERRORLEVEL% equ %ERRORCODE% goto :ERROR

Rem Mark campaign(s) or treatment campaign set(s) with Mark for deployment status
Rem run is the main calling script. ( run.cmd or run.sh )
Rem deploy is the discrete function script ( deploy.cmd or deploy.sh )
Rem testFile.txt lists the campaign(s) and set(s) to mark. Any valid text file will do.
Rem Note: A file called deployedFlowList is produced after this step, which promote, activate and
Rem deactivate uses.
Call run deploy deployList.txt
If %ERRORLEVEL% equ %ERRORCODE% goto :ERROR

if "%~1"=="/nopromote" goto :FINISH
Rem Promote the RTDM flow(s) that is in deployedFlowList ( generated my malauncher by the deploy
Rem script above).
Rem run is the main calling script. ( run.cmd or run.sh )
Rem promote is the discrete function script ( promote.cmd or promote.sh )
call run promote
If %ERRORLEVEL% equ %ERRORCODE% goto :ERROR

Rem Activate the RTDM flow(s) listed in deployFlowList
Rem run is the main calling script. ( run.cmd or run.sh )
Rem activate is the discrete function script ( activate.cmd or activate.sh -
Rem calls batchActivator.cmd or batchActivateor.sh )
Call run activate
If %ERRORLEVEL% equ %ERRORCODE% goto :ERROR
Exit /B 0

:ERROR
:FINISH
Echo Master script stopped
Exit /B %ERRORCODE%
```

Linux Environment

In the Linux environment, the master script uses the following syntax:

```
./run.sh master.sh
```

master.sh is the name that you provide for the master script. Each command script within the master script is called by the **run** script:

```
./run.sh command parameters
```

The following code is an example master script. Copy and paste the code into a text file whose name ends in .sh. Remove the # comments from the commands that you want to execute, and edit the parameters.

```
# ./run.sh export exportList.txt myPackage
# ./run.sh import importList.txt
# ./run.sh deploy deployList.txt
```

```
# ./run.sh test testList.txt
# ./run.sh promote
# ./run.sh activate
```

Log Files

Most logs from the scripts are placed in a subfolder that is named **logs**. The location is specified in the vars.cmd file in the Windows environment and the vars.sh file in the Linux environment. A date and time stamp is appended to each log file. One log file is created for each command.

The **deploy** and **test** scripts use launcher commands. By default, the log files for these scripts are in the location that is specified for the launcher. For more information, see "Enable Logging for the Launcher" on page 113. If the user who issues the command does not have permission to create the log file in the default location for the launcher, the log text is redirected to the console window and saved in **~/USERS/*USERID*/logs**.

Chapter 15

Troubleshooting

Overview of Troubleshooting **203**

Problem Definition **204**

Error Dialog Box **204**
Troubleshoot the Error Dialog Box 204
Next Steps 205

Unexpected Behavior **205**

Common Sources of Error **206**
Locked Asset 206
Other Locked Objects 206
Library Reference That Was Not Correctly Defined 206
Missing Table in MATABLES 207
Upstream Linked Diagrams That Are Open, Incomplete, or Corrupt 207
Flow Designers Close SAS Decision Manager in the Middle of a Process That They Have Initiated 207
Stored Process Server Error in SASDecMgrCore6.2.log 208

Summary of Logs for Troubleshooting **208**
Log Types 208
Other Logs and Files 210
Set Logging Level 211

Contacting SAS Technical Support **211**

Overview of Troubleshooting

This chapter presents a systematic approach to resolving problems in SAS Decision Manager

The problems that an administrator might need to troubleshoot typically result in an error message in SAS Decision Manager.

Begin by collecting basic information. A clear definition of the problem often makes it easier to find a solution.

Problem Definition

A good problem definition includes the following information:

1. What is the exact sequence of steps that produced the problem?
2. Can the problem be reliably reproduced, or is it intermittent?
3. Has the same sequence of steps been followed before without producing the problem? If so, what might have changed?
   a. information map
   b. data format, volume, and refresh period
   c. inclusion of nulls or spaces in the data
   d. time zone change
   e. amount of space or memory available in work or export directories
   f. permissions and authorizations within SAS products, within the database, or on an operating system directory
   g. library definitions and options within the SAS Management Console Data Library Manager plug-in
   h. operating system environment variables or settings, such as locale
4. Does this problem occur for all users? If not, can this user's problem be replicated on another user's computer?
5. What is the environment in which the problem occurs?
   a. operating system of the compute tier, middle tier, and client tier
   b. database type and version for the underlying data warehouse (for example, Oracle, SQL Server)
   c. version and hot fix level of the SAS Decision Manager products

When you have collected this information, you are ready to take the following troubleshooting steps.

Error Dialog Box

Troubleshoot the Error Dialog Box

The next category of problem to troubleshoot is the Error dialog box. When you encounter an error, take the following steps:

1. Click **Details** in the Error dialog box to view the full error message.
2. Copy the full error message by clicking on the dialog box, and then select Ctrl+A and Ctrl+C.
3. Open a text editor and paste this error message onto the page. This enables you to view the full text of the error. It also enables you to send the full text of the error to SAS Technical Support, if necessary.

4. Record the time of the error so that you can identify any associated error messages in the logs.
5. Use the error message and the problem description (from the "Problem Description" section above) to search for known issues in the SAS Notes at **`http://support.sas.com/notes/index.html`**. You might need to try different combinations of keywords or different sections of the error message.
6. If the error message in the dialog box does not help you find the source of the problem, SASDecMgrCore6.2.log is the next place to look. By default, this log is written to <*SASPlanName*>**`/Lev1/Web/Logs`** on the middle tier machine.
   a. Open this log and search for the word ERROR. The timestamps for the log messages help identify which messages might be associated with the error dialog box.
   b. If you do not find any relevant error messages in the log, then search for the word WARN.
7. If SASDecMgrCore6.2.log does not provide the information that you need in order to identify and correct the error, there are more logs and logging options that are described in "Summary of Logs for Troubleshooting" on page 208. See also "Common Sources of Error" on page 206.

Next Steps

After following these steps, it still might not be clear what is causing the problem. The next step is to search for an applicable SAS Note at **`http://support.sas.com/notes/index.html`**, using different combinations of keywords or error messages that you have encountered.

If the solution to the problem remains unclear, use the problem definition guideline in "Problem Definition" on page 204 to send a description of the problem to SAS Technical Support. Include information such as the following:

- The time at which the logging levels were increased and the problem was re-created
- The steps used to re-create the problem
- The text of the error in the error dialog box
- The most recent SASDecMgrCore6.2.log
- The Stored Process Server logs
- The web application server logs
- The SAS Content Server log

Unexpected Behavior

If you experience unexpected behavior, create a problem description by following the process in "Problem Definition" on page 204, and take the following steps:

1. Record the time of the error so that you can identify any associated error messages in the logs.
2. Open the most recent SASDecMgrCore6.2.log. By default, this log is written to <*SASPlanName*>**`/Lev1/Web/Logs`** on the middle tier machine.

a. Open this log and search for the word ERROR. The timestamps for the log messages help identify which messages might be associated with behavior that you are experiencing.

b. If you do not find any relevant error messages in the log, then search for the word WARN.

c. If you do not find any ERROR or WARN messages, find the section of the log that contains timestamps from around the time at which the unexpected behavior occurred.

3. Another log that might help in debugging unexpected behavior in a campaign is the SAS log available from within SAS Decision Manager. It is often particularly useful when you are troubleshooting issues with Process nodes. To view this log, select the **Log** tab on the Process node.

4. If the next step in troubleshooting is not obvious from the text in the logs, use the errors, warnings, and keywords from your problem description and logs to search for known issues in the SAS Notes at **http://support.sas.com/notes/index.html**. You might need to try different combinations of key words or different sections of the error message.

5. If you need further information in order to identify and correct the problem, see the logs and logging options that are described in the "Summary of Logs for Troubleshooting" on page 208. See also "Common Sources of Error" on page 206.

Common Sources of Error

Often, the error message in SASDecMgrCore6.2.log gives you enough information to find the source of the problem and correct it. Here are some of the most common errors that you might find as well as information about how to address them.

Locked Asset

To address this error, release the locked object in the Locks category in the Administration workspace of SAS Decision Manager. For more information, see *SAS Decision Manager User's Guide*.

Other Locked Objects

When more than one process attempts to use a particular database table, SAS data set, or other resource, the contention can cause a variety of problems. One of the most common problems is a process or flow that does not appear to ever complete. This is most common when you use Process nodes or Custom nodes. The short-term solution is to stop all intentional SAS Decision Manager and SAS Business Intelligence processes, and then check to see whether SAS processes are still running. If there appear to be processes that are not responding, kill them. Restart the compute tier. The long-term solution is to ensure that flows that contain Process nodes are run in a way that avoids contention.

Library Reference That Was Not Correctly Defined

To address this error:

1. Log on to SAS Management Console as an administrator.
2. Correct the reference by going to the Data Library Manager plug-in, right-clicking on the library in question, and selecting **Properties**.

The library definitions that are used by SAS Decision Manager can be viewed in SASDecMgrCore6.2.log. Often a careful examination of the LIBNAME statements in this log can help you pinpoint problems with reading or writing data.

Missing Table in MATABLES

To address this error:

1. Make sure that all users have closed the problematic flow.
2. If the error continues to occur, see "Upstream Linked Diagrams That Are Open, Incomplete, or Corrupt" on page 207.

Upstream Linked Diagrams That Are Open, Incomplete, or Corrupt

A flow can fail if any of its upstream linked diagrams are open or inaccessible. SASDecMgrCore6.2.log can contain a number of different errors that are associated with this problem, including a missing table in the MATABLES library. Problems with contention for a particular table in MATABLES can also exist if multiple flows attempt to use the same linked upstream diagram at precisely the same time. If you suspect that this might be causing your problem, take the following steps:

1. Clear the counts for the problem flow and for all flows that are upstream from it.
2. Check that the upstream flow can be deployed successfully.
3. Stagger deployment of all downstream flows so that their use of the upstream diagram does not cause contention.

Flow Designers Close SAS Decision Manager in the Middle of a Process That They Have Initiated

This error can cause processes not to respond on the middle tier or the compute tier. These orphaned processes might continue to consume resources. This can slow performance or even cause out-of-memory type errors. To remedy this error:

1. Close all open instances of SAS Decision Manager.
2. Close all SAS Business Intelligence applications.
3. Shut down the compute tier (object spawner, workspace servers, stored process servers).
4. Check for remaining SAS processes that should not be running. Kill any processes that are not responding.
5. Restart the compute tier.
6. Restart the middle tier (Remote Services, web application server).

Stored Process Server Error in SASDecMgrCore6.2.log

Look for additional details in the Stored Process Server logs. These files are located on your SAS Stored Process Server machine at <*SASPlanName*>**/Lev1/SASApp/StoredProcessServer/logs/**. Check the three most recent SAS Stored Process Server log files. There is generally one log for each SAS Stored Process Server process.

Summary of Logs for Troubleshooting

Log Types

You can often gather more detailed error information by increasing the logging level, replicating the problem, and then reviewing the recent additions to the logs. SASDecMgrCore6.2.log is often the first log for which you should try increasing the logging level. If the SASDecMgrCore6.2.log messages indicate that the problem is actually with a stored process, turn on additional logging for the SAS Stored Process Server instead.

SASDecMgrStudio6.2.log is the log for the SAS Decision Manager web client.

When you run SAS Decision Manager in a production environment, logging is generally kept to a minimum in order to maximize performance. Increase the logging level only when you are troubleshooting. When you finish troubleshooting, return the server environment to the previous level of logging to avoid degrading performance. Here are the tables that display information about the various types of logs.

Table 15.1 *Core Log*

Filename	SASDecMgrCore6.2.log
Location	<*SASPlanName*> **/Lev1/Web/Logs**
Log Contents	Java errors, some stored process information, LIBNAME statements that were executed. This is the primary log for SAS Decision Manager.
How to Increase the Logging Level	See SAS Note 37978 at **http://support.sas.com/kb/37/978.html**. Note that the Remote Services application might need to be restarted in order for the increased logging setting to take effect.

Table 15.2 *SAS Stored Process Server*

Filename	SASStoredProcessServer_yyyy-mm-dd_pid.log

Location	<*SASPlanName*> **/Lev1/Web/Logs** or <*SASPlanName*>**/Lev1/StoredProcessServer/Logs**
Log Contents	Code and results from the nodes that are executed by the SAS Stored Process Server.
How to Increase the Logging Level	See SAS Note 34114 at **http://support.sas.com/kb/34/114.html**.

Table 15.3 *Object Spawner*

Filename	ObjectSpawner_yyyy-mm-dd_pid.log
Location	<*SASPlanName*> **\Lev1\ObjectSpawner\logs** or <*SASPlanName*>**\Lev1\Logs**
Log Contents	Start-up information for the stored process servers and pooled workspace servers. Execution permission errors.
How to Increase the Logging Level	See SAS Note 34114 at **http://support.sas.com/kb/34/114.html**

Table 15.4 *SAS Workspace Server*

Filename	WorkspaceServer_yyyy-mm-dd_pid.log
Location	<*SASPlanName*> **\Lev1\SASApp\PooledWorkspaceServer\Logs** or <*SASPlanName*>**\Lev1\Logs**
Log Contents	Queries that are created by nodes. Because the size of this log increases quickly, enable debug logging for this log only when troubleshooting a specific problem.
How to Increase the Logging Level	See SAS Note 34567 at **http://support.sas.com/kb/34/567.html**.

Table 15.5 *SAS Content Server*

Filename	SASContentServer.log
Location	<*SAS-configuration-dir*> **/Lev1/Web/Logs**
Log Contents	Login issues; permission issues.
How to Increase the Logging Level	Not applicable.

Table 15.6 *SAS Business Rules Manager*

Filename	SASBusinessRulesManagerWeb2.1.log is the log file SASBusinessRulesManagerWeb-log4j.xml is the log configuration file
Location	*<SAS-configuration-dir>* **/Web/Logs/SASServer7_1/SASBusinessRulesManagerWeb2.1.log** *<SAS-configuration-dir>* **/Web/Common/LogConfig/SASBusinessRulesManagerWeb2.1.log**
Log Contents	Business Rules issues.
How to Increase the Logging Level	SAS Business Rules Manager uses log4j to perform logging. You should not change the categories or root logger in the configuration file unless you are instructed to do so by SAS Technical Support. You can change the logging priority levels if needed. For more information, see" Logging Priority Levels" in *SAS Business Rules Manager: Administrator's Guide*. By default, SAS Business Rules Manager writes the log files to SASBusinessRulesManagerWeb2.1.log. You can change the location of this log file in the configuration file. For more information, see *SAS Intelligence Platform: Middle-Tier Administration Guide* at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Other Logs and Files

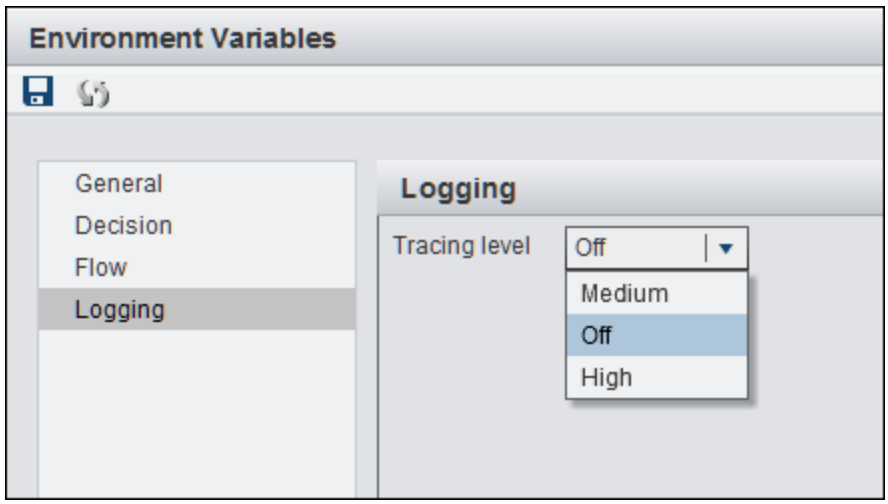
If you find that you need more information than is available in the logs above, the following might provide more details:

- SAS Metadata Server log
- event log from the Windows operating system
- database logs
- installation and configuration logs. See SAS Note 35426 at **http://support.sas.com/kb/35/426.html**
- the hosts file. The hosts file on Windows can be found in either **C:\WINNT\system32\drivers\etc** or **C:\Windows\system32\drivers\etc**
- internal only - the deployment registry can help with tracking SAS products. See **http://tsdsrv05.unx.sas.com:7777/iw/docs/sasnotes/fusion/34/276.html**.

Set Logging Level

The trace level for logging is set on the Environment Variables page in SAS Decision Manager.

Display 15.1 *Logging Page*



Select a tracing level to set the amount of detail in log messages.

Off
suppresses all logs except those logs that have a setting of DEBUGLEVEL_ALWAYS.

Medium
produces logs with a typical level of detail.

High
produces logs with a very high level of detail.

This option controls the size of the log that is generated for any SAS code that is run during a SAS Decision Manager session.

Contacting SAS Technical Support

If you are still unable to find and correct the source of the problem, contact SAS Technical Support. Include the following information:

- Problem description. (see "Problem Definition" on page 204).
- The SASDecMgrCore6.2.log that contains messages from the time the problem occurred.
- All SAS Stored Process Server logs that contain messages from the time the problem occurred.
- The web application server log or logs from the time the problem occurred.
- The SAS Content Server log from the time the problem occurred.

You can open a technical support track by e-mailing the problem description and attachments to support@sas.com. Files that are too big to e-mail can be sent by FTP, using the instructions that are available at **http://support.sas.com/kb/20/941.html**. Alternatively, you can start a technical support track by using the web form at **http://support.sas.com/ctx/supportform/createForm**.

Chapter 16

Optional Client Tier Applications

Overview of Optional SAS Solutions **213**

Hardware Requirements for the Client Tier **214**

SAS Data Integration Studio **214**
Description 214
Relationship to SAS Decision Manager 214
Typical User 214
Files 214
Requirements 214
Administrative Notes 215

SAS Enterprise Guide **215**
Description 215
Relationship to SAS Decision Manager 215
Typical User 215
Files 216
Requirements 216
Administrative Notes 216

SAS Enterprise Miner **216**
Description 216
Relationship to SAS Decision Manager 216
Typical User 216
Files 216
Requirements 217
Administrative Notes 217

Overview of Optional SAS Solutions

This section contains a summary of optional SAS solutions on the client tier that might be included in your installation. Each summary shows the relationship of the SAS solution to SAS Decision Manager. See the individual solution documentation for details.

For a description of SAS *n*-tier architecture, see "Architecture of the SAS Intelligence Platform" on page 1.

Hardware Requirements for the Client Tier

To view the latest system requirements for your environment, access **http://support.sas.com**. Select **Knowledge Base** ⇨ **System Requirements**.

For the latest requirements for Java Runtime Environments for SAS Client Applications, access **http://support.sas.com/resources/thirdpartysupport**.

Sites in the U.S. and Canada should call (919) 677-8008. Other locations should contact their on-site SAS support personnel or the nearest SAS office.

SAS Data Integration Studio

Description

SAS Data Integration Studio is a SAS application that enables you to integrate data from any data source, clean it, merge it, transform it, and place it into a target destination. Database managers use SAS Data Integration Studio to create SAS metadata about the structure of data. This metadata enables you to trace which data moved, when the data moved, how data was changed, and what business rules have been applied to data in SAS repositories. SAS metadata in SAS Data Integration Studio is different from counts metadata that is displayed in SAS Decision Manager diagrams. See the Administrative Notes in this section.

Relationship to SAS Decision Manager

SAS Data Integration Studio might be used to clean customer data for automated processes where bad data quality can lead to immediate and very high unwarranted costs. For example, you might want to avoid either sending customer mailings to the wrong address or multiple mailings to the same customer whose name has been entered in a database several times with different spellings.

Typical User

A typical user is an IT analyst.

Files

Executable
: The typical location of the executable file is **C:\Program Files\SASHome\SASETLStudio\9.3\etlstudio.exe**..

Log
: There are no logs of administrative interest for SAS Data Integration Studio.

Requirements

There are no special requirements needed to run SAS Data Integration Studio.

Administrative Notes

The counts metadata that you might see displayed in a node of a diagram is different from SAS metadata. The counts metadata is obtained from a specific group of data sets. These data sets contain the frequency information about the observations that are selected by users as they develop flows in SAS Decision Manager. This count value shows users how many observations are included in the selected range of values.

In SAS Data Integration Studio, a job is a collection of SAS tasks that specify processes that create output. Each job generates or retrieves SAS code that reads sources and creates targets in physical storage.

Use SAS Data Integration Studio to perform the following tasks:

- register a SAS library and a server for each database.
- create SAS metadata objects that define sources, targets, and the transformations that connect them. These SAS metadata objects are saved to a metadata repository.
- create a SAS metadata profile for administrators.
- register user identities.
- if applicable, configure SAS/CONNECT to access data on a machine that is remote to the default SAS Application Server.

See also SAS Data Integration Studio documentation at **http://support.sas.com/documentation/onlinedoc/etls/index.html**.

For an overview, take an interactive tour at **http://www.sas.com/technologies/dw/tour/itour_flash.html**. For administration documentation, see *Desktop Application Administration Guide* (located under **Administration Documentation**) at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

SAS Enterprise Guide

Description

SAS Enterprise Guide is a project-oriented Windows application that is designed to enable quick access to the analytic power of SAS software for statisticians, business analysts, and SAS programmers.

Relationship to SAS Decision Manager

SAS Enterprise Guide enables professionals to create SAS stored processes (SAS code) and to store the SAS code in a repository that is available to the SAS Stored Process Server. Stored processes are used to populate reports with specific data and to perform analytics. The stored process code can be made available within SAS Decision Manager as part of a Process node.

Typical User

A typical user is an IT analyst.

Files

Executable
The default location of the SAS Enterprise Guide application is **C:\ProgramFiles\SASHome\Enterprise Guide**<*version-number*>.

Log
There are no logs of administrative interest for SAS Enterprise Guide.

Requirements

A web browser is required in order to view any HTML reports that you create in SAS Enterprise Guide.

Administrative Notes

For an overview, take an interactive tour at **http://www.sas.com/technologies/bi/query_reporting/guide/tour/itour_flash.html**. For more information about Enterprise Guide, see **http://support.sas.com/documentation/onlinedoc/guide/**.

SAS Enterprise Miner

Description

SAS Enterprise Miner is a client application that provides a graphical user interface for quantitative analysts to mine large quantities of data in order to build accurate and predictive models. SAS Enterprise Miner enables the quantitative analyst to create and manage data mining process flows. A process flow is a sequence of steps that examine, transform, and process data in order to create models. These models predict complex behaviors of economic interest.

Relationship to SAS Decision Manager

Predictive models are used to target the appropriate market such as customer lifetime value, expected response rates, and so on. Models are also reviewed and refined at the conclusion of the process.

Typical User

A typical user is a flow analyst or modeler.

Files

Executable
The typical location of em.exe is **C:\Program Files\SASHome\SASAPCore\apps\EnterpriseMiner\bin**.

Log

There are no logs of administrative interest for SAS Enterprise Miner.

Requirements

There are no special requirements needed to run SAS Enterprise Miner.

Administrative Notes

For an overview, see: **http://www.sas.com/technologies/analytics/datamining/miner/**. For details about SAS Enterprise Miner, see the SAS Enterprise Miner Help that is available within the product. For product documentation, access **http://support.sas.com/documentation/onlinedoc/miner/index.html**. For administration documentation, see *Desktop Application Administration Guide* (located under **Administration Documentation**) at **http://support.sas.com/documentation/onlinedoc/intellplatform/index.html**.

Glossary

activity
See task

activity status
See task status

business context
a designation that identifies the information that a user can access. Data access is restricted to only the information that is required for a specific business need. A user can have access to more than one business context.

candidate model
a predictive model that evaluates a model's predictive power as compared with the champion model's predictive power.

challenger model
a model that is compared and assessed against a champion model for the purpose of replacing the champion model in a production scoring environment.

champion model
the best predictive model that is chosen from a pool of candidate models in a data mining environment.

champion/challenger control group
a type of control group. Members of a champion/challenger control group receive either a champion communication or a challenger communication that is intended to outperform the champion.

channel
a virtual communication path for distributing information. In SAS, a channel is identified with a particular topic. Using the features of the Publishing Framework, authorized users or applications can publish digital content to the channel, and authorized users and applications can subscribe to the channel in order to receive the content.

classification model
a predictive model that has a categorical, ordinal, or binary target.

control group
a group that is used to evaluate the effectiveness of a communication.

data object
an object that holds the business data that is required to execute workflow tasks.

data set
See SAS data set

data source
a table, view, or file from which you will extract information. Sources can be in any format that SAS can access, on any supported hardware platform. The metadata for a source is typically an input to a job.

DATA step
in a SAS program, a group of statements that begins with a DATA statement and that ends with either a RUN statement, another DATA statement, a PROC statement, or the end of the job. The DATA step enables you to read raw data or other SAS data sets and to create SAS data sets.

DATA step fragment
a block of SAS code that does not begin with a DATA statement. In SAS Model Manager, all SAS Enterprise Miner models use DATA step fragments in their score code.

delta report
a report that compares the input and output variable attributes for each of the variables that are used to score two candidate models.

diagram
a general term for a collection of nodes that make up a process.

downstream node
See successor node

holdout control group
a type of control group. Members of a holdout control group do not receive a communication. After a communication has been sent, the actions of the holdout control group can be compared with the actions of groups that received a communication.

identity
See metadata identity

input variable
a variable that is used in a data mining process to predict the value of one or more target variables.

instance
See workflow instance

library reference
See libref

libref
a SAS name that is associated with the location of a SAS library. For example, in the name MYLIB.MYFILE, MYLIB is the libref, and MYFILE is a file in the SAS library.

listen port
in a network, a communications endpoint at which a server listens for requests for service from the client application.

macro variable
a variable that is part of the SAS macro programming language. The value of a macro variable is a string that remains constant until you change it. Macro variables are sometimes referred to as symbolic variables.

metadata
descriptive data about data that is stored and managed in a database, in order to facilitate access to captured and archived data for further use.

metadata identity
a metadata object that represents an individual user or a group of users in a SAS metadata environment. Each individual and group that accesses secured resources on a SAS Metadata Server should have a unique metadata identity within that server.

milestone
a collection of tasks that complete a significant event. The significant event can occur either in the process of selecting a champion model, or in the process of monitoring a champion model that is in a production environment.

model function
the type of statistical model, such as classification, prediction, or segmentation.

model scoring
the process of applying a model to new data in order to compute outputs.

node
a graphical region of a diagram that contains information about a process flow operation. A node consists of a graphical component (icon) as well as a properties window.

output variable
in a data mining process, a variable that is computed from the input variables as a prediction of the value of a target variable.

package
See SAS package file

package file
See SAS package file

participant
a user, group, or role that is assigned to a task. These users, groups, and roles are defined in SAS metadata and are mapped to standard roles for the workflow.

policy
a workflow element that associates event-driven logic with a task or subflow. Policies are usually triggered automatically by an event such as a status change or a timer event.

predecessor node
a node that precedes another node in a diagram. A predecessor node is also called an upstream node.

prediction model
a model that predicts the outcome of an interval target.

project
a collection of models, SAS programs, data tables, scoring tasks, life cycle data, and reporting documents.

project tree
a hierarchical structure made up of folders and nodes that are related to a single folder or node one level above it and to zero, one, or more folders or nodes one level below it.

publication channel
an information repository that has been established using the SAS Publishing Framework and that can be used to publish information to users and applications.

publish
to deliver electronic information to one or more destinations. These destinations can include e-mail addresses, message queues, publication channels and subscribers, WebDAV-compliant servers, and archive locations.

Publishing Framework
a component of SAS Integration Technologies that enables both users and applications to publish SAS files (including data sets, catalogs, and database views), and other digital content to a variety of destinations. The Publishing Framework also provides tools that enable both users and applications to receive and process published information.

response
the reaction that an individual has to a campaign, such as requesting a quote, making an inquiry, opening an e-mail message, or buying the product.

SAS Content Server
a server that stores digital content (such as documents, reports, and images) that is created and used by SAS client applications. To interact with the server, clients use WebDAV-based protocols for access, versioning, collaboration, security, and searching.

SAS data set
a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views.

SAS Metadata Repository
a container for metadata that is managed by the SAS Metadata Server.

SAS Metadata Server
a multi-user server that enables users to read metadata from or write metadata to one or more SAS Metadata Repositories.

SAS package file
a container for data that has been generated or collected for delivery to consumers by the SAS Publishing Framework. Packages can contain SAS files, binary files, HTML files, URLs, text files, viewer files, and metadata.

SAS publication channel
See publication channel

SAS variable
a column in a SAS data set or in a SAS data view. The data values for each variable describe a single characteristic for all observations (rows).

scoring
See model scoring

scoring function
a user-defined function that is created by the SAS Scoring Accelerator from a scoring model and that is deployed inside the database.

scoring task
a workflow that executes a model's score code.

segmentation model
a model that identifies and forms segments, or clusters, of individual observations that are associated with an attribute of interest.

source
See data source

SPK
See SAS package file

subscriber
a recipient of information that is published to a SAS publication channel.

successor node
a node that follows another node. A successor node is also called a downstream node.

swimlane
a workflow diagram element that enables you to group tasks that are assigned to the same participant.

task
a workflow element that associates executable logic with an event such as a status change or timer event.

task status
the outcome of a task in a workflow. The status of a task (for example, Started, Canceled, Accepted) is typically used to trigger the next task.

upstream node
See predecessor node

variable
See SAS variable

version folder
a folder in the Project Tree that typically represents a time phase and that contains models, scoring tasks, life cycle data, reports, documents, resources, and model performance output.

WebDAV server
an HTTP server that supports the collaborative authoring of documents that are located on the server. The server supports the locking of documents, so that multiple authors cannot make changes to a document at the same time. It also associates metadata with documents in order to facilitate searching. The SAS business intelligence applications use this type of server primarily as a report repository. Common WebDAV servers include the Apache HTTP Server (with its WebDAV modules enabled), Xythos Software's WebFile Server, and Microsoft Corporation's Internet Information Server (IIS).

workflow
a series of tasks, together with the participants and the logic that is required to execute the tasks. A workflow includes policies, status values, and data objects.

workflow definition
a workflow template that has been uploaded to the server and activated. Workflow definitions are used by the SAS Workflow Engine to create new workflow instances.

workflow instance
a workflow that is running in the SAS Workflow Engine. After a workflow template is uploaded to the server and activated, client applications can use the template to create and run a new copy of the workflow definition. Each new copy is a workflow instance.

workflow template
a model of a workflow that has been saved to an XML file.

Index

A

access control template
  See ACT
access permissions 30
  capabilities 24
  groups 23
  roles 24
  SAS Decision Manager assets 34
  SAS identities 22
  User Manager plug-in 29
accounts
  capabilities 24
  permissions 30
  roles 24
  SAS identities 22
  User Manager plug-in 29
ACT 30
activating flows 194
architecture
  SAS Intelligence Platform 1
archiving data 123
ASCII characters
  and translated data 106
autoexec_usermods.sas file
  and object spawner 7
  configuring 16
  specifying search order 16

B

backing up data 123
binary data item level 71
bulk load facility
  activating 127
  and Oracle 129
  enabling in business context 128
business contexts
  and bulk load facility 128
  and common data model 80
  and information maps 3
  data grid library 41
  data set options 128, 129
  library location for common data model 107
  permissions 31
  updating subjects 103

C

calculated items
  and locale 47
  importing 37
  in information maps 46
  pasting 37
campaign groups
  in common data model 84
campaigns
  in common data model 81
capabilities 24
categories
  in information maps 49
cells
  in common data model 87
client tier 1, 213
  hardware requirements 214
comments
  capabilities 28
common data model 79
  and DB2 102
  and Greenplum 102
  and multiple business contexts 80
  and Netezza 102
  and Oracle 101
  and SAS SPD Server 102
  and SQL Server 102
  and Teradata 102
  campaign group tables 84
  campaign tables 81
  cell tables 87
  changing location 106
  communication tables 85
  configuring SAS library resources 15

control group tables 88
creating table structure 100
custom details 105
deploying 99
extension tables 90, 104
history table 90, 92
history tables 103
lookup tables 93
package tables 87
prepopulated values 93
publishing 107
SAS Marketing Optimization tables 90
saving 107
treatment tables 87
updating 80
user-defined tables 91
communications
in common data model 85
connection
accepting a 6
contact history tables
in common data model 90
setting up 103
control groups
in common data model 88
copying and pasting
SAS Management Console folders 34
counts metadata 215
and information maps 75
clearing 77
definition of 75
generating 76, 77
tables 77
updating 76
custom details
in common data model 105
custom formats
configuring SAS library resources 15
custom properties
and information maps 53
and MATemplate.txt file 56
data item level 63, 64
folder level 62, 64
MAMeta library 53
map level 61, 64
required 53
custom repositories
permissions 31
customer data
configuring SAS library resources 15

D

data
accessing 12, 215
and information maps 46
archiving 123
backing up 123
data grids
data libraries 41
data items
and locale 47
in information maps 46
levels 73
data processes
data libraries 41
data sources
and data grids 41
multiple 13
registering 12
data tier 1, 11
components 12
hardware requirements 12
database logs 210
database servers 215
date format 19
DB2 12
and common data model 102
DBCS
See double-byte character sets
deactivating flows 194
decision history tables
in common data model 92
Decision Manager Services User ID 22
Delete Environment Settings utility 164
deploying flows 44, 185
double-byte character sets
in SAS Decision Manager tables 19

E

errors
JVM 132
Release Tables utility 160
SAS Metadata Server 10
This item cannot be saved because it is no longer locked. 164
troubleshooting 203
event variables
column name format 110
data libraries 41
exported files
date format 19
exporting
best practices 36
flows 161
SAS packages 35, 178
exports
troubleshooting 9

F

filters
  in information maps 46
flow activities
  identifying data for 45
flows
  activating and deactivating 194
  backing up 124
  deploying 44, 185
  exporting 161
  importing 161
  listing input variables 120
  promoting 191
  unexpected behavior 205
folders
  importing and exporting objects between 161
  in information maps 46
  organizing as categories 49

G

Greenplum
  and common data model 102
groups 23
  permissions 30
  User Manager plug-in 29

H

hardware requirements
  client tier 214
  middle tier 4
history
  in common data model 90, 92
history tables 103

I

ID data item level 71
importing
  best practices 36
  flows 161
  SAS packages 35, 181
information maps
  adding relationships 50
  and business contexts 3
  and data tier 11
  and locale 47
  backing up 124
  canceling Presentation tab 51
  creating 47, 52
  custom properties 53, 61, 62, 63, 64
  data item levels 71, 73
  definition of 46
  editing names 50
  editing relationships 50
  finding physical data 51
  folders 49
  identifying data for 45
  improving performance 51
  naming subjects 50
  SAS Information Map Studio 3
input variables 120
install account
  setup instructions 18
interval data item level 71

J

job
  executing a 6
  in SAS Data Integration Studio 215
JVM
  errors 132
  memory size 130

L

Launcher
  errors 120
  logs 113
levels
  data item 71
libraries
  See SAS library resources
library references (librefs) 13
  defining 14
Limit nodes
  troubleshooting 9
load balancing
  and stored process server 9
  documentation 10
Load Treatments utility 165
locale
  and session encoding 106
  information maps 47
logging on
  problems 164
logs 208
  core 208
  database 210
  installation and configuration 210
  object spawner 209
  SAS Business Rules Manager 210
  SAS Content Server 209
  SAS Metadata Server 210
  SAS Stored Process Server 208
  SAS Workspace Server 209
  trace level 211
  Windows event 210
lookup tables

in common data model 93
prepopulating 106

M

MAMeta library 53
MAMisc library 14
backing up 124, 125
marking flows for deployment 44
MATables library 14
backing up 124, 125
configuring SAS library resources 15
releasing tables 160
MATemplate.txt file 56
metadata
access to 23
and information maps 46
backing up 123
metadata cache 77
metadata objects 215
metadata repository 10, 215
metadata server 10
configuration file 10
middle tier 1, 4
components of 5
hardware requirements 4
models 216
making available to SAS Decision Manager 28, 43

N

Netezza
and common data model 102
nominal data item level 71

O

object spawner 6
logs 209
object types
See public object types
Oracle 12, 67
and bulk load facility 129
and common data model 101
extension tables 105
table name restriction 104
ordinal data item level 71

P

packages
in common data model 87
performance
activating bulk load facility 127
business context options 128, 129
configuring SAS SPD Server 15
data set options 128, 129
INSERTBUFF option 129
JVM memory size 130
permissions 30
ACT 30
business context 31
capabilities 24
custom repositories 31
groups 23
registering user identities 215
roles 24
SAS Decision Manager assets 34
SAS identities 22
SAS Management Console folders 33
User Manager plug-in 29
predictive models 216
prepopulated values
in common data model 93
PROC SQL
and generating counts metadata 77
promoting flows 191
public object types 161

R

reference tables
in common data model 93
prepopulating 106
registering data 12
relationships
adding to information map 50
editing in information map 50
Release Tables utility 160
errors 160
remote access 215
reports
input variables 120
request
from client application 10
listening for 6
roles 24
modifying 29
permissions 30
User Manager plug-in 29

S

SAS application tier
See SAS server tier
SAS Business Rules Manager
logs 210
SAS code
client applications submitting 8
SAS Content Server 5
backing up 125

logs 209
SAS Data Integration Studio 214
SAS Decision Manager 2
and SAS Decision Services 39
assets 34
backing up 125
logs 208
marking flows for deployment 44
models 43
session timeout value 19
test cases 109
unexpected behavior 205
SAS Decision Services
activating and deactivating flows 194
and SAS Decision Manager 39
SAS Enterprise Guide 215
SAS Enterprise Miner 216
SAS identities 22
SAS Information Map Studio 3
SAS Intelligence Platform
architecture 1
security administration 22
SAS library resources 13
common data model 15
custom formats 15
customer data 15
MAMeta library 53
MAMisc library 14
MATables library 14, 15
registering 215
search order 16
temp tables 15
SAS Management Console
copying and pasting folders 34
Folders tab 33
unexpected behavior 205
SAS Marketing Optimization
common data model tables 90
SAS metadata profile 215
SAS Metadata Server
backing up 124
errors 10
log 210
SAS packages
importing and exporting 35, 163, 178
SAS server tier 1, 6
hardware requirements 6
Launcher 7
object spawner 6
SAS Content Server 5
SAS Metadata Server 10
SAS Stored Process Server 9
SAS Workspace Server 8
SAS SPD Server
and common data model 102
configuring 14
SAS Stored Process Server 9
logs 208
SAS Technical Support 211
SAS Web Application Server 5
SAS Workspace Server 8
logs 209
SAS/ACCESS 12
BULKLOAD option 127
SAS/CONNECT 215
SASMSG function 106
sasv9.cfg
and double-byte character sets 19
and object spawner 7
scheduling 7
security 21
capabilities 24
groups 23
permissions 30
registering user identities 215
roles 24
SAS identities 22
SAS Intelligence Platform 22
User Manager plug-in 29
server
naming during setup 18
problems connecting to 18
registering for each database 215
SPD 14
stored process 9
WebDAV 5
workspace 8
session encoding
and locale 106
session timeout value 19
setup instructions
date format 19
install account 18
SAS Intelligence Platform 18
server name 18
session timeout value 19
Split nodes
troubleshooting 9
SQL Server 12
and common data model 102
and multiple databases 13
SQL Server - OLE DB connection
and multiple databases 13
stored process 215
stored process server 9
subjects
naming in information map 50
system options
in autoexec_usermods.sas file 16

T

tables
  double-byte character sets 19
technical support
  See SAS Technical Support
temp tables
  configuring SAS library resources 15
Teradata 12
  and common data model 102
test cases
  from historical data 109
  SAS data sets 109
tier
  client 1, 213
  data 1, 11
  definition of 1
  middle 1, 4
  SAS server 1, 6
timeout value 19
treatments
  in common data model 87
  Load Treatments utility 165
troubleshooting 203

U

unary data item level 71
Unicode
  and translated data 106
user connections 10
user IDs
  permissions 30
  SAS identities 22
  User Manager plug-in 29
User Manager plug-in 29
utilities
  Delete Environment Settings 164
  export and import 161
  Load Treatments 165
  Release Tables 160

W

warning messages
  troubleshooting 205
web application server 5
web file server 5
Web-based Distributed Authoring and Versioning
  See WebDAV
WebDAV 5
workspace server 8