



THE
POWER
TO KNOW.

SAS[®] Model Manager 2.2

User's Guide



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2009. *SAS® Model Manager 2.2: User's Guide*. Cary, NC: SAS Institute Inc.

SAS® Model Manager 2.2: User's Guide

Copyright © 2009, SAS Institute Inc., Cary, NC, USA

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

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

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

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

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

1st electronic book, May 2009

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

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

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

Contents

<i>About This Book</i>	<i>vii</i>
<i>What's New in SAS Model Manager 2.2</i>	<i>ix</i>
<i>Recommended Reading</i>	<i>xiii</i>

PART 1 What Is SAS Model Manager? 1

Chapter 1 • Overview of SAS Model Manager	3
Managing Analytical Models Using SAS Model Manager	3
The SAS Model Manager Operational Environment	4
Model Management Process	6
Chapter 2 • Introduction to SAS Model Manager	9
Accessibility Features of SAS Model Manager	9
Layout of the SAS Model Manager Window	9
SAS Model Manager User Groups, Roles, and Tasks	18

PART 2 Working with Projects and Versions 25

Chapter 3 • Working with Data Sources	27
Overview of Data Sources	27
Project Tables	28
Creating Project Input and Output Tables	31
Creating Scoring Task Input and Output Tables	32
Creating a Test Table	33
Creating a Performance Table	34
Add a Data Source Table	35
Delete a Data Source Table	36
Chapter 4 • Organizing the Project Tree	39
Overview of the Project Tree	39
Create an Organizational Folder	40
Associate Documents with a Folder	41
Deleting an Object in the Project Tree	42
Chapter 5 • Working with Projects	43
Overview of Projects	43
Planning a Project	45
Prerequisites for Creating Projects	46
Create a Project	47
Setting the Project Champion Model Status	48
Project Properties	49
Chapter 6 • Working with Versions	53
Overview of Versions	53
Creating Life Cycle Templates	56

Create a Version	67
Version Properties	68
Working with Life Cycles	70

PART 3 Importing, Scoring, and Validating Models 77

Chapter 7 • Importing Models	79
Overview of Importing Models	79
Import Models from Metadata Repository	80
Import Package Files from SAS Enterprise Miner	81
Import SAS Code Models	83
Import PMML Models	92
Import Partial Models	93
Set Model Properties	94
Map Model Variables to Project Variables	95
User-Defined Model Templates	96
Specific Properties for a Model	103
Chapter 8 • Scoring Models	107
Overview of Scoring Tasks	107
Scoring Task Tabbed Views	109
Create Scoring Output Tables	111
Create a Scoring Task	113
Modify a Scoring Task	114
Map Scoring Task Output Variables	115
Execute a Scoring Task	115
Graph Scoring Task Results	118
Generated Scoring Task Content Files	121
Scoring Task Properties	121
Result Set Properties	122
Chapter 9 • Validating Models Using Comparison Reports	125
Overview of Model Comparison Reports	125
Model Profile Reports	127
Creating Delta Reports	129
Creating Dynamic Lift Reports	131
View Reports	134
Chapter 10 • Validating Models Using User Reports	135
Overview of User Reports	135
Ad Hoc Reports	137
User-Defined Reports	140

PART 4 Deploying and Delivering Champion Models 147

Chapter 11 • Deploying Models	149
Overview of Deploying Models	149
Champion Models	150
Freezing Models	152
The Default Version for a Project	154

Chapter 12 • Delivering Models	157
Overview of Model Delivery	157
Publishing Models	158
Exporting Models	162
Publish Teradata Scoring Functions	164

Chapter 13 • Replacing a Champion Model	177
Overview of Replacing a Champion Model	177
Select a New Default Version	178
Retire a Project	178

PART 5 Performance Monitoring 179

Chapter 14 • What Are Performance Monitoring Reports?	181
Overview of Performance Monitoring Reports	181
Types of Performance Monitoring Reports	182
Performance Index Warnings and Alerts	188
Performance Data Sets Versus Performance Data Sources	190
The Process of Monitoring Champion Models	190
Formatting Performance Monitoring Reports	192
View Reports	195

Chapter 15 • Create Reports by Defining a Performance Task	197
Overview of Creating Reports Using a Performance Task	197
Prerequisites for Running the Define Performance Task Wizard	199
Run the Define Performance Task Wizard	201

Chapter 16 • Create Reports Using Batch Programs	205
Overview of SAS Programs to Monitor Model Performance	206
Prerequisites for Running Batch Performance Reports	207
Report Output in Test and Production Modes	210
Define the Report Specifications	211
Extracting the Champion Model from a Channel	222
SAS Code to Run Performance Reports	225

PART 6 Appendixes 233

Appendix 1 • Query Utility	235
Overview of the Query Utility	235
Search for Models	236
Search by Using a UUID	238
Search Life Cycles for Tasks Assigned to Users	239

Appendix 2 • SAS Model Manager Access Macros	243
Overview of SAS Model Manager Access Macros	243
Using the SAS Model Manager Access Macros	244
Dictionary of SAS Model Manager Access Macros	249

Appendix 3 • Properties	283
General Properties	283
System Properties	284

Specific Properties for a Project	285
User-Defined Properties	287
Specific Properties for a Version	289
Specific Properties for Milestones and Tasks	290
Specific Properties for a Model	291
Scoring Task Properties	293
Result Set Properties	294
Appendix 4 • SAS Model Manager In-Database Scoring for Teradata	297
Overview of SAS Model Manager In-Database Scoring for Teradata	297
Using the Java Scoring API	298
Glossary	303
Index	309

About This Book

Audience

SAS Model Manager is designed for the following users:

- Those who are responsible for developing analytical models.
- Those who are responsible for modeling project management.
- Those who are responsible for model validation and performance testing.
- Scoring officers.
- Analysts.

You might be assigned to a specific user group or role, and that assignment determines which tasks you can perform. For more information, see [“SAS Model Manager User Groups, Roles, and Tasks”](#) on page 18.

Prerequisites

Prerequisites for Using SAS Model Manager

Here are the prerequisites for using SAS Model Manager:

- The SAS Model Manager client is installed on your computer.
- You have a user ID and password for logging on to SAS Model Manager.

Conventions Used in This Document

The following typographical conventions are used for all text in this document except for syntax:

bold

identifies an item in the SAS Model Manager window or a menu item.

italics

identifies a book title or a value that is supplied by the user.

monospace

identifies SAS code.

UPPERCASE

identifies a SAS language element, such as the SAS statements KEEP or DROP.

The following typographical conventions are used in SAS Model Manager macro syntax:

bold

identifies the name of a macro.

italic

identifies an argument that must be supplied by the user.

< >

identifies an optional macro argument.

| (vertical bar)

indicates that you can choose one value from a group. Values that are separated by the vertical bar are mutually exclusive.

UPPERCASE

indicates a keyword that can be used as a value for an argument.

What's New in SAS Model Manager 2.2

Overview

SAS Model Manager 2.2 has the following new features and enhancements:

- score models in a Teradata Enterprise Data Warehouse using SAS Scoring Accelerator for Teradata
- import multiple models simultaneously
- manage scorecard model projects
- import and store PMML models
- create life cycle and model templates using the SAS Model Manager Template Editor
- create production model monitoring reports using the Define Performance Task wizard
- create additional output formats for the New Report wizard
- attach multiple versions of a document
- set user-defined project properties used by other SAS applications

Score Models in a Teradata Enterprise Data Warehouse Using SAS Scoring Accelerator for Teradata

From project initiation to retiring a model, SAS Model Manager now supports score code projects that you run on your Teradata Enterprise Data Warehouse (EDW). SAS Model Manager enables you to publish classification and prediction models to the Teradata EDW. When you publish your champion model, the SAS Scoring Accelerator for Teradata converts and exports your champion model score code to scoring functions that you can deploy inside Teradata. A scoring application (for example, a call center application that uses the SAS Model Manager Java Scoring API) can then execute the scoring functions in the Teradata EDW.

For more information, see [“Publish Teradata Scoring Functions”](#) on page 164.

Import Multiple Models Simultaneously

Instead of importing models into a version one at a time using the SAS Model Manager client, you can import multiple models at the same time by using the %MM_RegisterByFolder macro.

For more information, see [“%MM_RegisterByFolder Macro” on page 274](#).

Manage Scorecard Model Project

You can import scorecard models for credit risk scoring into SAS Model Manager from a SAS Enterprise Miner package file or from SAS code local files for a classification model. After you import the model, all SAS Model Manager functionality is available to scorecard models.

For more information, see [“Import Package Files from SAS Enterprise Miner” on page 81](#) and [“Import SAS Code Models” on page 83](#).

Import and Store PMML Models

You can import PMML models into SAS Model Manager. All PMML model types are associated with one of the four SAS Model Manager model types: analytical, classification, prediction, or segmentation. PMML models can be stored only in SAS Model Manager. Scoring, deployment, and reporting of PMML models are not available in this release.

For more information, see [“Import PMML Models” on page 92](#).

Create Life Cycle and Model Templates Using the SAS Model Manager Template Editor

You can now create customized templates to use as the life cycle template for a version and model templates for importing SAS code models into the SAS Model Manager Template Editor.

In a custom life cycle template, you can create milestones to complete and tasks for each milestone. For each task, you can define task-related properties, such as date and owner information, and task dependencies. The template is available to use in SAS Model Manager after a SAS Model Manager administrator stores the template on the middle-tier server.

You can use the template editor to create custom model templates for SAS code models that you want to register with SAS Model Manager. Using the editor, you define model template properties and the model component files that must be included for a SAS code model. The template is available for use in SAS Model Manager after a SAS Model Manager administrator stores the template on the middle-tier server.

For more information, see [“Creating Life Cycle Templates” on page 56](#) and [“User-Defined Model Templates” on page 96](#).

Create Production Model Monitoring Reports Using the Define Performance Task Wizard

After a champion model is in production, you can use the Define Performance Task wizard to generate the SAS code that creates model monitoring reports. After the code has been generated, you can execute the code from a new **PerformanceMonitor** project node in the Project Tree.

For more information, see [“What Are Performance Monitoring Reports?” on page 181](#).

Create Additional Output Formats for the New Report Wizard

In addition to creating PDF output when you run the New Report wizard, you can now create HTML, RTF, and Excel output formats.

For more information, see [“Model Comparison Output Files” on page 126](#).

Attach Multiple Versions of a Document

You can attach multiple versions of a document to the same folder in the Project Tree without changing the name of the document. A new Show History window displays document versions.

For more information, see [“Associate Documents with a Folder” on page 41](#).

Set User-Defined Project Properties Used by Other SAS Processes

If you run your SAS Model Manager models in a SAS Customer Intelligence Real-Time Decision Management environment, then you previously had to define four user-defined properties. SAS Model Manager now defines the properties **KeyType**, **TableKey**, **Precode**, **ScoringInputTable**, **UDFName**, **UDFPrefix**, and **DBmsTable**. You must provide the appropriate values for your model.

For more information, see the documentation for SAS Customer Intelligence and [“User-Defined Properties” on page 287](#).

Recommended Reading

- *SAS Model Manager: Administrator's Guide*
- *Getting Started with SAS Enterprise Miner*
- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *SAS Macro Language: Reference*
- *SAS Management Console Help*

For a complete list of SAS publications, see the current SAS Publishing Catalog. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/pubs

* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Part 1

What Is SAS Model Manager?

Chapter 1

Overview of SAS Model Manager 3

Chapter 2

Introduction to SAS Model Manager 9

Chapter 1

Overview of SAS Model Manager

Managing Analytical Models Using SAS Model Manager	3
The SAS Model Manager Operational Environment	4
Model Management Process	6

Managing Analytical Models Using SAS Model Manager

Using SAS Model Manager, you can organize modeling projects, develop and validate candidate models, assess candidate models for champion model selection, and publish and monitor champion models in a production environment. All model development and model maintenance personnel, including data modelers, validation testers, scoring officers, and analysts can use SAS Model Manager.

SAS Model Manager in a Business Intelligence environment can meet many model development and maintenance challenges. Here are some of the services SAS Model Manager provides:

- You use a single interface, the SAS Model Manager client, to access all of your business modeling projects. The SAS Model Manager client presents projects in a tree structure, known as the Project Tree.
- All models are stored in a central, secure model repository.
- All project or model metadata is readily accessible through the SAS Model Manager client.
- You create custom milestones and tasks to meet your business requirements and to match your business processes. You use these milestones and tasks to monitor the development and deployment of models.
- Data tables that are registered in SAS Management Console can be used in SAS Model Manager.
- The models that you import into SAS Model Manager can be SAS Enterprise Miner models or they can be models that you develop using SAS code. You can create custom model templates for SAS code models so that SAS Model Manager knows exactly what files and metadata are associated with a model.
- After you import candidate models, you can use SAS Model Manager to run scoring tasks to validate models.

- SAS Model Manager has several reports that you can use to compare and assess candidate models. You can also write your own SAS reporting programs to assess candidate models and run them in SAS Model Manager.
- You can publish and score models in Teradata Enterprise Data Warehouse using the SAS Scoring Accelerator.
- After you choose a champion model, you can lock the model and its associated data for future reference or auditing by freezing the containing version.
- SAS Model Manager uses the SAS Integration Technologies Publishing Framework to publish models to a channel.
- You can monitor the performance of a champion model in a production environment either using the SAS Model Manager window or SAS Model Manager macros in a batch environment.
- SAS Model Manager provides macro programs for you to run model registration and scoring in a batch environment.
- Using a query utility, you can look for models by name or identifier, or you can look for tasks.

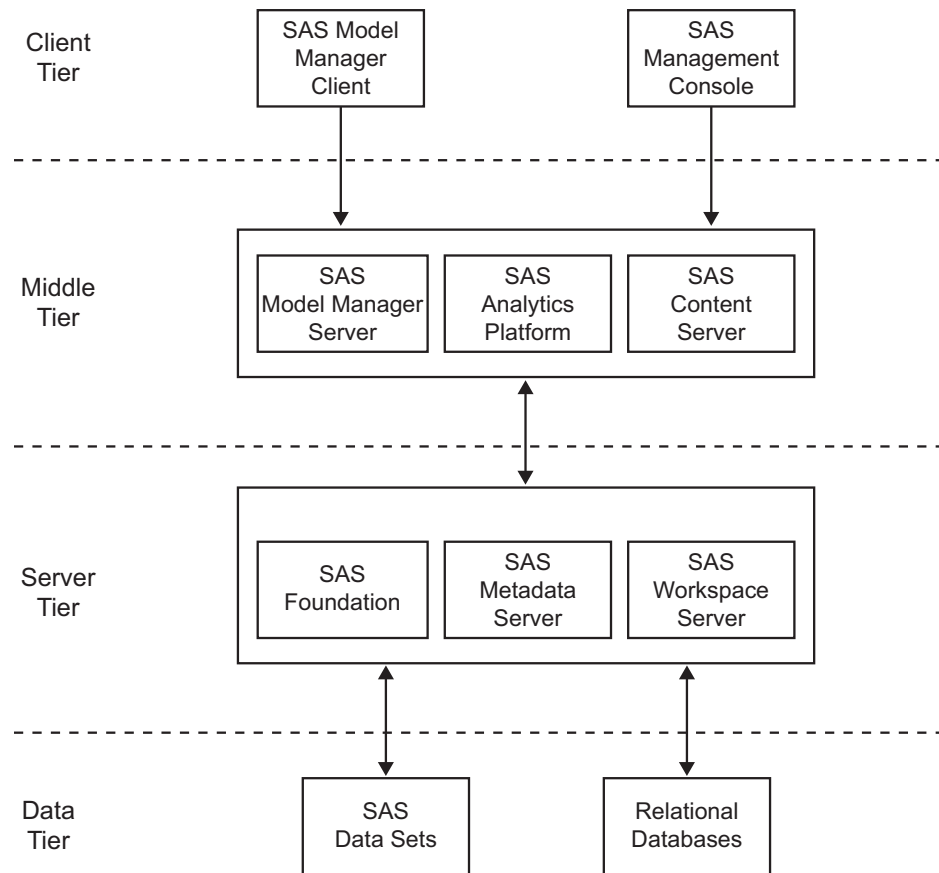
Any user who is registered in SAS Management Console can be assigned to a SAS Model Manager group, and can then work in SAS Model Manager. SAS Model Manager has three groups.

- Users in the Model Manager Administrator group ensure that all aspects of the modeling project are configured and in working order. Users in the Model Administrator group can perform all tasks within SAS Model Manager.
- Users in the Model Manager Advanced User group can perform some of the tasks that the Model Manager Administrator group can perform as well as all tasks that users in the Model Manager User group can perform.
- Users in the Model Manager User group can perform development, validation, reporting and publishing tasks with some Write access limitations.

Data source tables are an integral part of the modeling process in SAS Model Manager. SAS Model Manager requires project input, output, and scoring output prototype tables to define variables to SAS Model Manager. Data tables are used for scoring, testing, and performance monitoring. Most data source tables can be created from the train table that you used to develop a model. Performance data can be created from your operational data.

The SAS Model Manager Operational Environment

The following figure illustrates the components of a typical SAS Model Manager operational environment when the data tier does not use Teradata tables:



SAS Model Manager Client

User communication to and from SAS Model Manager is carried out using the SAS Model Manager Client. You use the SAS Model Manager client to create projects and versions, import models, connect with data sources, validate models, run modeling reports, run scoring tasks, set project status, declare the champion model, and run performance tests.

SAS Management Console

Users in the SAS Model Manager Administrator group use SAS Management Console to define the users, roles, data sources, and publishing channels for use with SAS Model Manager.

SAS Model Manager Server

The SAS Model Manager Server orchestrates the communication and movement of data between all servers and components in the SAS Model Manager operational environment.

SAS Analytics Platform

The SAS Analytics Platform provides a common application framework for analytic applications such as SAS Enterprise Miner and SAS Forecast Studio. SAS Analytics Platform is a middle-tier component.

SAS Content Server

The SAS Model Manager model repository as well as the SAS Model Manager metadata are stored in the SAS Content Server. Communication between SAS Model Manager and the SAS Content Server uses the WebDAV communication protocol.

SAS Foundation

SAS Foundation includes Base SAS and a superset of SAS software that is a collection of software which is required to support the deployment of SAS in your organization.

Data modelers can develop SAS code models using Base SAS and other analytic SAS software such as SAS/STAT software. The SAS Model Manager Server directs all score code, macro programs, and reporting and monitoring programs to be processed by SAS Foundation.

SAS Metadata Server

SAS Model Manager can store and retrieve basic metadata about models from the SAS Metadata Server. The basic model metadata in a metadata repository includes input and output metadata as well as score code. All model-related files can be stored with the model in a metadata repository.

SAS Workspace Server

The execution of SAS score code within SAS Model Manager occurs on a SAS Workspace Server.

SAS Data Sets

SAS data sets can serve as data sources in SAS Model Manager.

Relational Databases

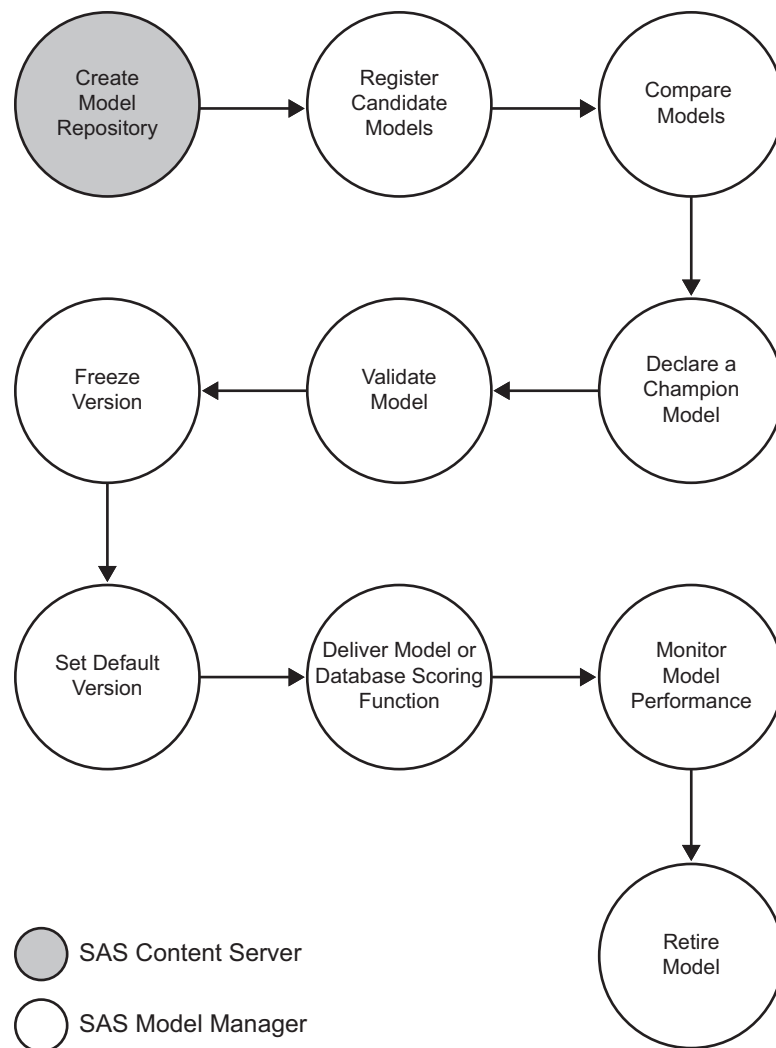
Tables in relational databases can serve as data sources in SAS Model Manager.

See Also

[“Publish Teradata Scoring Functions” on page 164](#)

Model Management Process

The following diagram illustrates the model management process that you use in SAS Model Manager:

Figure 1.1 The Model Management Process

Summary of the model management process:

- **Create Model Repository:** create a secure model repository on the SAS Content Server where SAS code, input and output files, and metadata that is associated with a model can be stored.
- **Register Candidate Models:** register input and output files, and then import and configure a model.
- **Compare Models:** perform scoring tests and create comparison reports for models by using test data sources.
- **Declare a Champion Model:** declare the model to use for testing and production phases of the life cycle.
- **Validate Model:** perform scoring tests and create validation reports for the champion model by using test data sources.
- **Freeze Version:** lock a version when the champion model in a version folder is approved for production.
- **Set Default Version:** set a default version before exporting a project champion model. A default version is the current production version for the SAS Model Manager project.

- **Deliver Model or Database Scoring Function:** publish and share the model life cycle and performance data over established reporting channels or publish a scoring function for a model to a database.
- **Monitor Model Performance:** provide comparative model performance benchmarking.
- **Retire Model:** retire a model from production.

Chapter 2

Introduction to SAS Model Manager

Accessibility Features of SAS Model Manager	9
Layout of the SAS Model Manager Window	9
Overview of the SAS Model Manager Window	9
SAS Model Manager Perspectives	11
SAS Model Manager Toolbar and Menus	15
SAS Model Manager User Groups, Roles, and Tasks	18
SAS Model Manager Groups	18
SAS Model Manager Roles	18
Setting Up SAS Model Manager	19
Setting Up Projects and Versions	20
Importing and Assessing Models	21
Deploying and Delivering Models	21
Monitor Champion Model Performance	22
General Tasks	22

Accessibility Features of SAS Model Manager

This product has not been tested for compliance with U.S. Section 508 standards. If you have questions or concerns about the accessibility of SAS products, send them to accessibility@sas.com or call SAS Technical Support.

Layout of the SAS Model Manager Window

Overview of the SAS Model Manager Window

About the SAS Model Manager Window

The SAS Model Manager user interface provides you with quick access to data, metadata, and summary information for your projects and models. The interface includes a menu bar, a toolbar, a perspective button bar, and perspectives that contain views. The menu bar enables you to perform tasks on your models and projects. The toolbar provides shortcuts to tasks that you can perform on your models and projects. Many toolbar options are also available on pop-up menus. The list of active options on the menu bar or on the toolbar varies according to your perspective and the component that is selected. Inactive options

are dimmed. For more information about the SAS Model Manager toolbar and menus, see [“SAS Model Manager Toolbar and Menus” on page 15](#).

The perspective button bar enables you to select a perspective to display. SAS Model Manager has three perspectives: the Projects perspective, the Life Cycle perspective, and the Data Sources perspective. Each perspective contains views that enable you to view information about your models and projects, and to perform specific tasks on your life cycle templates, data sources, models and projects.

Overview of Perspectives




The SAS Model Manager interface is divided into three perspectives:

- [Projects perspective](#)
- [Life Cycle perspective](#)
- [Data Sources perspective](#)

Each perspective is a work area for an aspect of model management. The perspectives contain views that enable you to access specific information and functionality to manage your projects and models. For example, the Projects perspective has three views: the Repository view, the Properties view, and the Annotations view.

The perspective button bar is located in the upper left corner of the SAS Model Manager interface. Click the perspective button to see that perspective in the SAS Model Manager interface.

Table 2.1 The Perspective Buttons

	Projects perspective button
	Life Cycle perspective button
	Data Sources perspective button


Overview of SAS Model Manager Toolbar and Menus

The SAS Model Manager toolbar provides shortcuts to SAS Model Manager tasks. You can use the toolbar to perform such tasks as creating and organizing a project, importing a model file, and selecting a champion model. For more information about the SAS Model Manager toolbar, see [“SAS Model Manager Toolbar” on page 15](#).

The SAS Model Manager menus enable you to perform general tasks such as renaming an object or accessing Help. The menus also enable you to perform tasks that are specific to SAS Model Manager such as creating and organizing a project, importing a model file, and selecting a champion model. For more information about SAS Model Manager menus, see [“SAS Model Manager Menus” on page 16](#).

SAS Model Manager Perspectives

Projects Perspective

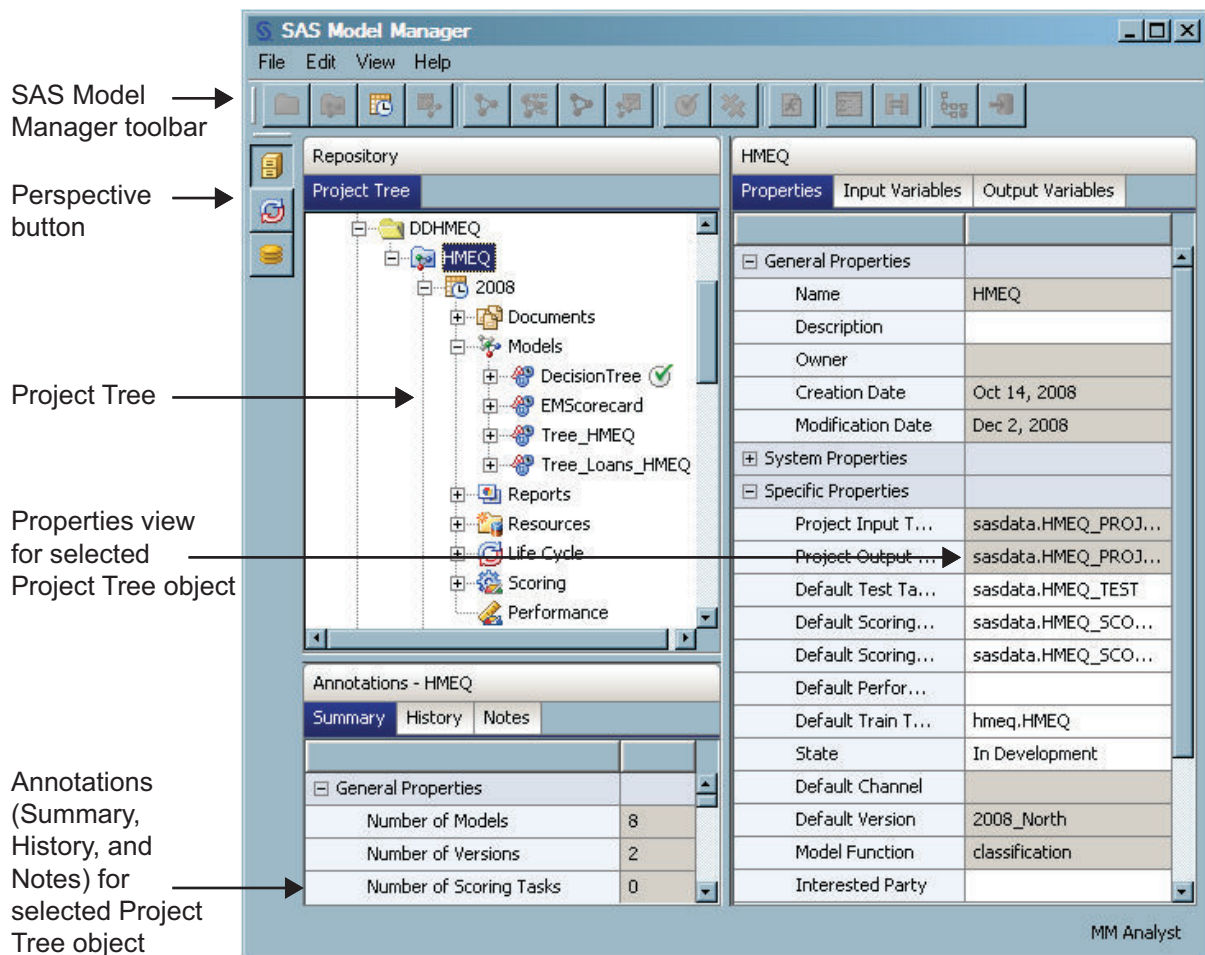
When SAS Model Manager opens, the Projects perspective is displayed. If you are working in another perspective, click the button  to display the Projects perspective.

Most of your work in SAS Model Manager is performed in the Projects perspective where you manage model projects and their components. The Projects perspective contains three major views: the Repository view, the Properties view, and the Annotations view.

The Repository view displays the Project Tree, which resembles a file utility. In the Project Tree, you can select and expand organizational folders that contain one or more project folders. Inside a project folder, you can create individual project versions. Project versions are containers that hold documents, models, modeling reports, and scoring tasks that are all associated with the same time span, such as a retail season, a fiscal quarter, or a fiscal year.

The hierarchical folder or object that is selected in the Project Tree controls the content that is displayed in the Properties view and the Annotations view. In the following example, the Project Tree displays an open organizational folder that is named DDHMEQ. The DDHMEQ folder contains a project folder that is named HMEQ. The HMEQ Project folder contains a Version folder that is named **2008**. The **2008** Version folder contains a **Models** folder that contains four models: DecisionTree, EMScorecard, Tree_HMEQ, and Tree_Loans_HMEQ.

Figure 2.1 The Projects Perspective




The Properties view displays the metadata that is associated with the selected component in the Project Tree. For example, if you select a model component in the Project Tree, the Properties view displays model-level metadata. When you select a version folder in the Project Tree, the Properties view displays the metadata that is associated with the version.

When you select an object such as an organizational folder, a project folder, or a version folder in the Project Tree, the Annotations view contains three tabs. In this example, the **Summary** tab displays a model aging report. For information about the model aging report, see [“Summary Report” on page 183](#). The **History** tab displays a time-stamped log that documents the following transactions by user ID for the selected repository component:

- Create
- Modify
- Import
- Publish
- Export
- Delete

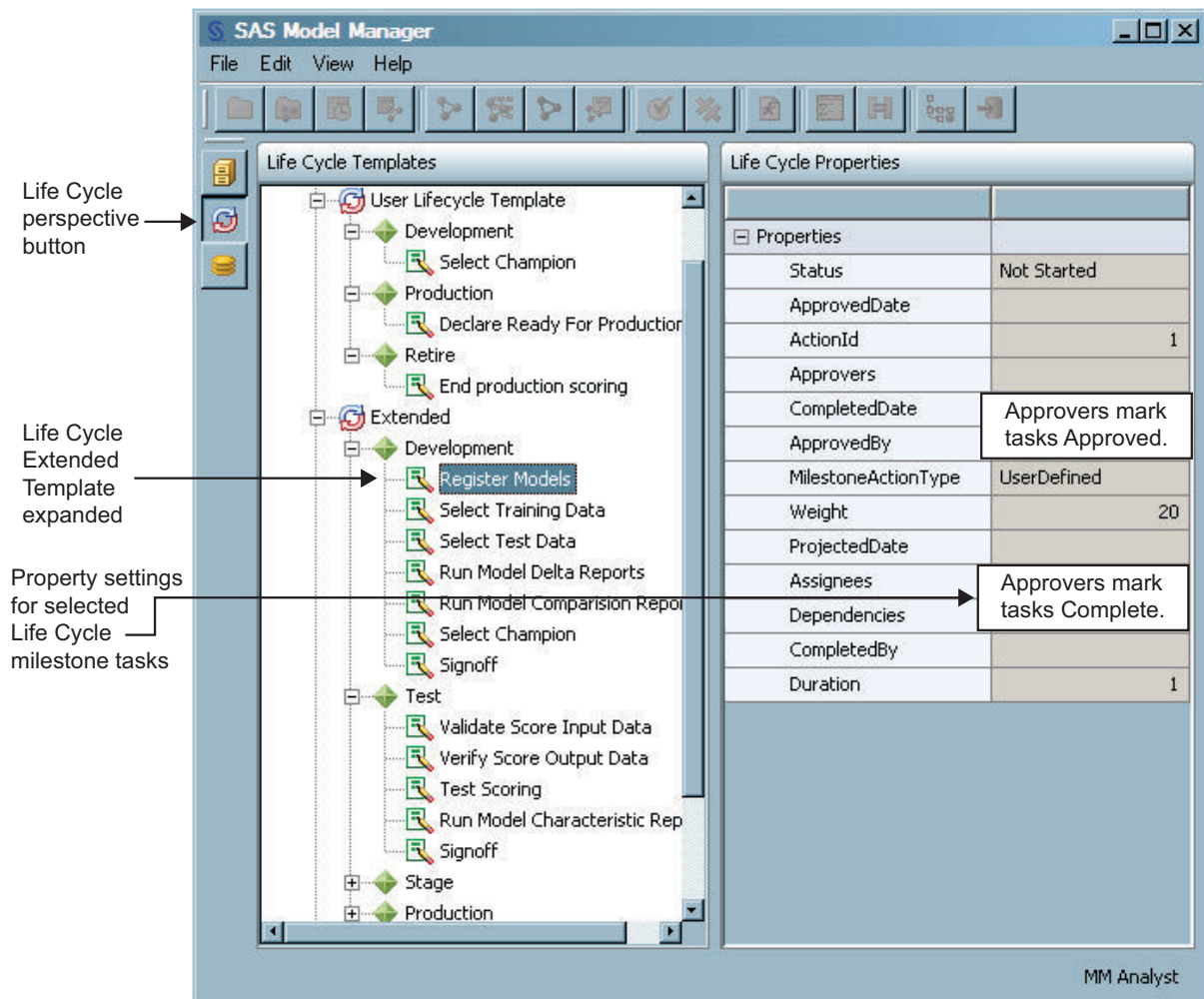
The **Notes** tab enables you to record information about the selected component that could be useful for later reference. For more information about SAS Model Manager projects, see [“Working with Projects” on page 43](#).

Life Cycle Perspective

Click the Life Cycle perspective button  to browse the model life cycle templates. Each life cycle template contains milestones that correspond to key events in the life span of modeling projects in SAS Model Manager. Example templates are included with the software so that individuals in your organization can learn about model life cycle templates. By default, the Life Cycle perspective displays three example life cycle templates: Basic, Standard, and Extended.


The Life Cycle Templates view in the following example displays a User Life Cycle Template.

Figure 2.2 The Life Cycle Perspective



For more information about Life Cycles, see [“Working with Life Cycles”](#) on page 70.

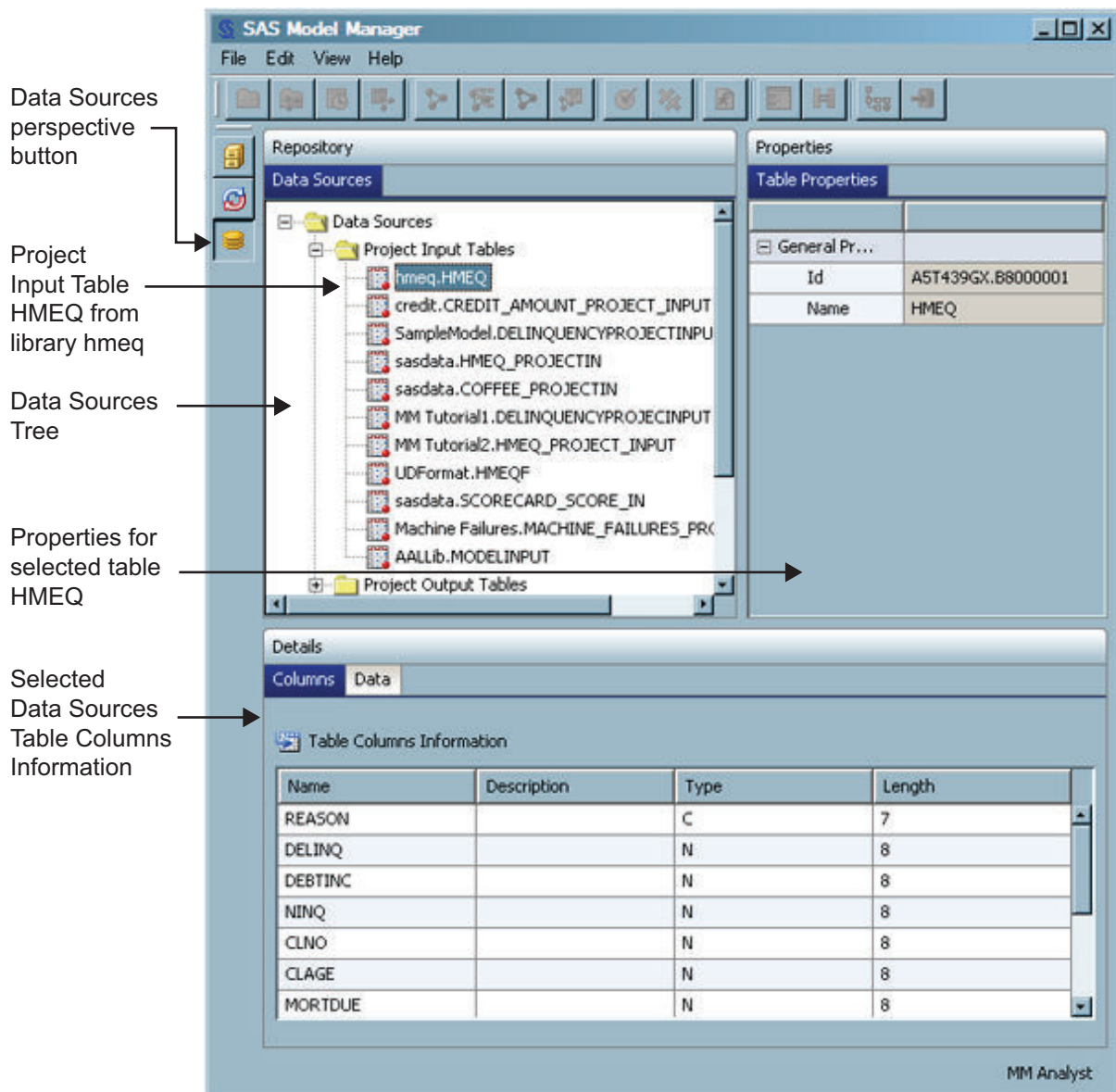
Data Sources Perspective

Click the Data Sources perspective button  to add data sources to SAS Model Manager. You can populate the SAS Model Manager Data Sources from libraries that are defined through SAS Management Console.

In the following examples, the Repository view lists the data tables that are available to SAS Model Manager projects. The Properties view displays a list of the metadata for the

selected data table. The Details view displays information about the contents of the selected prototype or data table.

Figure 2.3 The Data Sources Perspective



SAS Model Manager organizes the tables in the Data Sources into seven groups:

- Project Input Tables
- Project Output Tables
- Scoring Task Input Tables
- Scoring Task Output Tables
- Train Tables
- Test Tables
- Performance Tables

For more information about Data Sources, see [“Working with Data Sources” on page 27](#).

SAS Model Manager Toolbar and Menus

SAS Model Manager Toolbar

The buttons on the SAS Model Manager toolbar are shortcuts to SAS Model Manager tasks. You can also perform these tasks by accessing the main menu or pop-up menu. The list of active tasks varies according to the component that you select. Inactive tasks are hidden. Tooltips appear when you rest the pointer over an icon on the toolbar. Click the icon to select a task.

The following example displays a SAS Model Manager toolbar that has all of the buttons enabled. The individual buttons in the toolbar are enabled only when the proper usage context exists. When you select a component in the SAS Model Manager user interface, buttons that are not applicable are dimmed and are not available for use.



- 1 **New Folder** creates an organizational folder under the selected folder in the Project Tree. For more information, see [“Create an Organizational Folder” on page 40](#).
- 2 **New Project** creates a mining project folder underneath an organizational folder. Project folders normally contain one or more version folders. For more information, see [“Create a Project” on page 47](#).
- 3 **New Version** creates a version folder. A version folder contains the models and their related files that are typically associated with a chronological period, such as a fiscal year or quarter. You can create version folders only under a project folder. For more information, see [“Create a Version” on page 67](#).
- 4 **New Scoring Task** creates a scoring task in the Scoring folder that you have selected in the Project Tree. To enable the New Scoring Task button and menus, select the Scoring folder in your project version. For more information, see [“Create a Scoring Task” on page 113](#).
- 5 **Metadata Repository** imports a SAS Enterprise Miner model from the SAS Metadata Repository into the **Models** folder that is selected in the Project Tree. For more information, see [“Import Models from Metadata Repository” on page 80](#).
- 6 **SAS Enterprise Miner Package File** imports a SAS Enterprise Miner package file (SPK) from the user's client machine to the **Models** folder that is selected in the Project Tree. For more information, see [“Import Package Files from SAS Enterprise Miner” on page 81](#).
- 7 **Local Files** imports SAS code models that were not developed in SAS Enterprise Miner (such as LOGISTIC procedure models) into SAS Model Manager. For more information, see [“Import SAS Code Models” on page 83](#).
- 8 **PMML Model File** imports a PMML model. For more information, see [“Import PMML Models” on page 92](#).
- 9 **Set Champion Model/Default Version** selects the champion model or the default version. When you highlight a model in the Project Tree, click the Set Champion Model/Default Version button to promote the model to champion status over all other models in the version folder that contains the models. When you highlight a version folder in the Project Tree, click the Set Champion Model/Default Version button to promote that

version folder to the default version within the project for model scoring and life cycle management. For more information, see [“Deploying Models” on page 149](#).

- 10 **Clear Champion Model/Default Version** deselects the champion model, the default version, or both. When you highlight a default version in the Project Tree, click the Clear Champion Model/Default Version button to clear the default status of that version. When you highlight a champion model in the Project Tree, click the Clear Champion Model/Default Version button to clear that model's status as the champion model. For information, see [“Deploying Models” on page 149](#).
- 11 **Execute** performs the scoring task that is highlighted in the Project Tree. You can browse the scored data result set, generate plots of the scored data, and review SAS output from the scoring run. For more information, see [“Execute a Scoring Task” on page 115](#).
- 12 **Create Output Table** creates a new output table structure for one or more SAS Model Manager scoring tasks. For more information, see [“Create Scoring Output Tables” on page 111](#).
- 13 **Quick Mapping Check** enables you to compare the input data source variable names that were submitted to a scoring task with the names of the required input variables in the model. If the variables of the scoring input table are an incomplete subset of the model's required variables, then the score results might be statistically invalid. The variable data type is not validated. For more information, see [“Overview of Scoring Tasks” on page 107](#).
- 14 **Advanced View** opens a SAS Enterprise Miner Package Viewer window that displays the contents of the SAS Enterprise Miner SPK file, if one was created with the model that is highlighted in the Project Tree.

When you register a model in a metadata repository for SAS Enterprise Miner, a SAS Enterprise Miner Package file is registered if a Web folder to store the package file was previously defined using SAS Management Console. For more information, see the section on model deployment in the Help for SAS Enterprise Miner 6.1.
- 15 **Export Model into SAS Metadata Repository** exports the model that is highlighted in the Project Tree to a SAS Metadata Repository. For more information, see [“Exporting Models” on page 162](#).

SAS Model Manager Menus

The SAS Model Manager main menu varies in content depending on whether you select the Projects perspective, the Life Cycle perspective, or the Data Sources perspective. If a menu item is not applicable in the selected perspective, then the item appears dimmed and is not available for selection. Here is a list of all available menu items:

File

- **New Folder** creates a new organizational folder. For more information, see [“Create an Organizational Folder” on page 40](#).
- **New Project** creates a new modeling project folder. For more information, see [“Create a Project” on page 47](#).
- **New Version** creates a new version folder within a Project. For more information, see [“Create a Version” on page 67](#).
- **New Scoring Task** creates a new scoring task within a **Scoring** folder. For more information, see [“Create a Scoring Task” on page 113](#).
- **Manage Templates** opens the SAS Model Manager Template Editor, which enables you to create or edit life cycle templates and model templates. For more information,

see [“Creating Life Cycle Templates” on page 56](#) and [“User-Defined Model Templates” on page 96](#).

- **Import From** specifies the method that you want to use to import models into a version's **Models** folder. For more information, see [“Importing Models” on page 79](#).
 - **Metadata Repository** opens a SAS Folder View window that you can use to select SAS Enterprise Miner models and then import them into a **Models** folder. For more information, see [“Import Models from Metadata Repository” on page 80](#).
 - **SAS Enterprise Miner Package File** opens a local file browser window that you can use to import a model of a SAS Enterprise Miner Package File (SPK). For more information, see [“Import Package Files from SAS Enterprise Miner” on page 81](#).
 - **Local Files** opens a local file browser window that you can use to import SAS code models that were not developed in SAS Enterprise Miner, such as LOGISTIC procedure models. For more information, see [“Import SAS Code Models” on page 83](#).
 - **PMML Model File** opens a local file browser window that you can use to import PMML models. For more information, see [“Import PMML Models” on page 92](#).
- **Exit** ends the SAS Model Manager session and closes the window.

Edit

- **Copy** creates a copy of the selected model object.
- **Paste** places the model object that is in a copy buffer in a location that you select in the Project Tree.
- **Rename** enables you to enter a new name for the selected folder or file.

View

- **Projects** displays the Projects perspective. For more information, see [“Projects Perspective” on page 11](#).
- **Life Cycles** displays the Life Cycles perspective. For more information, see [“Life Cycle Perspective” on page 13](#).
- **Data Sources** displays the Data Sources perspective. For more information, see [“Data Sources Perspective” on page 13](#).
- **Toolbar** toggles the SAS Model Manager toolbar on and off. For more information, see [“SAS Model Manager Toolbar” on page 15](#).

Help

- **Help** opens the table of contents of the SAS Model Manager Help.
- **About** displays information about the version of SAS Model Manager that you are using.

SAS Model Manager User Groups, Roles, and Tasks

SAS Model Manager Groups

When you work in SAS Model Manager, the SAS Model Manager administrator assigns your user ID to one of three SAS Model Manager groups, Model Manager Administrators, Model Manager Advanced Users, and Model Manager Users. Groups can perform certain tasks within SAS Model Manager. For example, users in the Model Manager Administrator group are the only users who can freeze a version.

Users in the Model Manager Administrator group can perform any task with SAS Model Manager. The Model Manager Advanced User and Model Manager User groups are more restrictive. Use the tables in the subsequent sections for a list of SAS Model Manager tasks and the groups whose users can perform the task.

The SAS Model Manager administrator can create custom groups for your organization as well and assign SAS Model Manager roles to those groups. Contact your SAS Model Manager administrator to find out your group and roles.

The following table lists the abbreviations for groups that are used in the task tables below:

SAS Model Manager Group	Abbreviation
Model Manager Administrator Users	MM Admin
Model Manager Advanced Users	MM Adv User
Model Manager Users	MM User

SAS Model Manager has two other groups, Model Manager Example Life Cycle Assignees and Model Manager Example Life Cycle Approvers. These groups are used in the life cycle templates that are provided by SAS Model Manager for example purposes only. The life cycle templates and groups supplied by SAS should not be modified.

SAS Model Manager Roles

The SAS Model Manager roles enable specific users or groups to be assigned to complete specific tasks within SAS Model Manager. In most cases, roles are assigned to groups. Three of the roles are general and correlate to the groups that are supplied by SAS Model Manager. Roles that are associated with the life cycle enable users and groups to be assigned to complete tasks or approve that tasks are complete.

The following table describes the roles and lists the role abbreviations that are used in the list of tasks:

Role	Description	Abbreviation
Model Manager: Administration Usage	A user who can perform all SAS Model Manager tasks. This role is assigned to the group Model Manager Administrators.	:Admin
Model Manager: Advanced Usage	A user who can perform all SAS Model Manager tasks except for tasks that can be performed only by a SAS Model Manager administrator. This role is assigned to the group Model Manager Advanced User.	:Adv
Model Manager: Usage	A SAS Model Manager general user. The general user can perform all tasks except for advanced user tasks and administrator tasks. This role is assigned to the group Model Manager Users.	:User
Model Manager: Life Cycle Participant Usage	A user or group whose role is displayed in the Life Cycle Template Editor Participant List selection list.	:LC Participant
Model Manager: Life Cycle Assignee Usage	A user or group who can be assigned to complete a life cycle task.	:LC Assignee
Model Manager: Life Cycle Approval Usage	A user or group who can approve the completion of a life cycle task.	:LC Approver

Setting Up SAS Model Manager

Use the following table to determine the users who can complete the tasks to set up SAS Model Manager:

Task	Group	Topic
Create SAS Model Manager Users in SAS Management Console	SAS Administrator	See <i>SAS Model Manager: Administrator's Guide</i>
Create data libraries in SAS Management Console	MM Adv User, MM Admin, SAS Administrator	See <i>SAS Model Manager: Administrator's Guide</i>

Task	Group	Topic
Create channel location folders on a SAS server	MM Admin	See <i>SAS Model Manager: Administrator's Guide</i>
Create SAS Model Manager channels in SAS Management Console	SAS Administrator	See <i>SAS Model Manager: Administrator's Guide</i>
Define channel subscribers in SAS Management Console	SAS Administrator	See <i>SAS Model Manager: Administrator's Guide</i>
Create project tables	MM User, MM Adv User, MM Admin	“Working with Data Sources” on page 27
Register project tables in SAS Management Console	MM Adv User, MM Admin	See <i>SAS Model Manager: Administrator's Guide</i>
Configures the SAS Content Server for SAS Model Manager	MM Admin	See <i>SAS Intelligence Platform: Web Application Administration Guide</i>

Setting Up Projects and Versions

Use the following table to determine the users who can complete the tasks to set up projects and versions in SAS Model Manager:

Task	Group	Topic
Add or delete data sources	MM User, MM Adv User, MM Admin	“Add a Data Source Table” on page 35 “Delete a Data Source Table” on page 36
Create organizational folders	MM Adv User, MM Admin	“Create an Organizational Folder” on page 40
Create projects	MM Adv User, MM Admin	“Create a Project” on page 47
Create versions	MM Adv User, MM Admin	“Create a Version” on page 67
Delete a node in the Project Tree	MM Adv User, MM Admin	“Deleting an Object in the Project Tree” on page 42
Create life cycle templates	MM User, MM Adv User, MM Admin	“Creating Life Cycle Templates” on page 56
Store life cycle templates on the middle-tier server	any user with Write access to the directory	<i>SAS Model Manager: Administrator's Guide</i>

Importing and Assessing Models

Use the following table to determine the users who can complete the tasks to import and assess models:

Task	Group	Topic
Create model templates	MM Adv User, MM Admin	“User-Defined Model Templates” on page 96
Import models	MM Adv User, MM Admin	“Importing Models” on page 79
Configure model properties	MM Adv User, MM Admin	“Set Model Properties” on page 94
Map model variables to project variables	MM Adv User, MM Admin	“Map Model Variables to Project Variables” on page 95
Run model comparison reports	MM Adv User, MM Admin	“Validating Models Using Comparison Reports” on page 125
Create user reports	MM Adv User, MM Admin	“Validating Models Using User Reports” on page 135
Create scoring task output tables	MM Adv User, MM Admin	“Create Scoring Output Tables” on page 111
Create and run scoring tasks	MM Adv User, MM Admin	“Scoring Models” on page 107

Deploying and Delivering Models

Use the following table to determine the users who can complete the tasks to deploy and deliver models:

Task	Group	Topic
Select a champion model	MM Adv User, MM Admin	“Champion Models” on page 150
Validate the champion model by running a scoring task using test data and reviewing the scoring task output	MM Adv User, MM Admin	“Validating Models Using Comparison Reports” on page 125 “Validating Models Using User Reports” on page 135
Set the default version	MM Adv User, MM Admin	“The Default Version for a Project” on page 154

Task	Group	Topic
Freeze or unfreeze versions	MM Admin	“Freezing Models” on page 152
Publish a project, version, or model	MM Adv User, MM Admin	“Publishing Models” on page 158
Extract a model	any user who has the appropriate access rights to the SAS Metadata Repository	“Extract a Published Model” on page 161
Export a model	MM Adv User, MM Admin	“Exporting Models” on page 162
Publish a Teradata scoring function	MM Adv User, MM Admin	“Publish Teradata Scoring Functions” on page 164

Monitor Champion Model Performance

Use the following table to determine the users who can complete the tasks to create and run the reports to monitor the champion model performance:

Task	Group	Topic
Set project properties	MM Adv User, MM Admin	“Project Properties” on page 49
Run the Define Performance Task wizard	MM Adv User, MM Admin	“Create Reports by Defining a Performance Task” on page 197
Execute the performance monitoring SAS programs from the PerformanceMonitor node.	MM Adv User, MM Admin	“Run the Define Performance Task Wizard” on page 201
Run performance monitoring batch jobs	in Test mode: MM User, MM Adv User, MM Admin in Production mode: MM Adv, MM Admin	“Create Reports Using Batch Programs ” on page 205
View monitoring reports and charts	MM User, MM Adv User, MM Admin	“View Reports” on page 195

General Tasks

Use the following table to determine the users who can complete these general tasks:

Task	Group or Role	Topic
Update life cycle status	<p>For a version life cycle, any SAS Model Manager user or group that is assigned to the role :LC Assignee.</p> <p>If no user or group is assigned to the role :LC Assignee, then any user or group that is assigned to the role :LC Participant can update the life cycle status.</p>	“Update Milestone Status” on page 73
Approve a life cycle task	<p>For a version life cycle, any SAS Model Manager user or group that is assigned to the role :LC Approver.</p> <p>If no user or group is assigned to the role :LC Approver, then any user or group that is assigned to the role :LC Participant can approve a life cycle task.</p>	“Update Milestone Status” on page 73
Use the Query utility	MM User, MM Adv User, MM Admin	“Query Utility” on page 235
Set the status of a project champion model	MM Adv User, MM Admin	“Setting the Project Champion Model Status” on page 48
Replacing a champion model	MM Adv, MM Admin	“Overview of Replacing a Champion Model” on page 177

Part 2

Working with Projects and Versions

<i>Chapter 3</i>	
Working with Data Sources	27
<i>Chapter 4</i>	
Organizing the Project Tree	39
<i>Chapter 5</i>	
Working with Projects	43
<i>Chapter 6</i>	
Working with Versions	53

Chapter 3

Working with Data Sources

Overview of Data Sources	27
Project Tables	28
Project Input Tables	28
Project Output Tables	28
Scoring Task Input Tables	29
Scoring Task Output Tables	29
Train Tables	30
Test Tables	30
Performance Tables	30
Creating Project Input and Output Tables	31
Create a Project Input Table	31
Create a Project Output Table	32
Creating Scoring Task Input and Output Tables	32
About Scoring Task Input and Output Tables	32
Create a Scoring Task Input Table	33
Create a Scoring Task Output Table	33
Creating a Test Table	33
Creating a Performance Table	34
About Performance Tables	34
Naming a Performance Table for Use with the Define Performance Task Wizard	34
Create a Performance Table	35
Add a Data Source Table	35
Delete a Data Source Table	36

Overview of Data Sources

Data sources are a collection of prototype tables and data tables that reside in the SAS Metadata Repository and are used by SAS Model Manager. You can view all registered data sources in SAS Model Manager by clicking the Data Sources perspective button



Prototype tables define the input and output variables that SAS Model Manager uses to define or score a model. SAS Model Manager does not use any data that might be contained

in a prototype table other than the variable definitions. Data tables contain the data that you use to train or validate models, test models, and create reports that monitor the performance of a champion model in production.

The following tables are prototype tables :

- Project Input Tables
- Project Output Tables
- Scoring Task Output Tables

The project input table and the project output table must be registered as data sources before you create a project. The scoring task input table and the scoring task output table must be registered as data sources before you create a scoring task.

You use the following data tables to train models, test models, and run production model monitoring reports:

- Scoring Task Input Tables
- Train Tables
- Test Tables
- Performance Tables

After you create a table, it must be registered using SAS Management Console. After it is registered, you can add it to SAS Model Manager as a data source. For instructions on registering data sources, see *SAS Model Manager: Administrator's Guide*.

Project Tables

Project Input Tables

A project input table is a SAS data set that must contain the champion model input variables and their attributes. It is a prototype table that is used to define the project input variables and the variable attributes such as data type and length. A project can have numerous candidate models that use different predictor variables as input. Because the project input table must contain the champion model input variables, project input table becomes a superset of all of the potential input variables that any candidate model in the project, might use.

A project input table can have one or more observations. Data that is in a project input table is not used by SAS Model Manager.

Before you can create a project, you create a project input table, register the table using the SAS Management Console, and add it as a SAS Model Manager data source. In SAS Model Manager, you can view project input tables in the Data Source perspective **Project Input Tables** folder.

See Also

[“Creating Project Input and Output Tables” on page 31](#)

Project Output Tables

A project output table is a SAS data set that defines project output variables and variable attributes. The project output table contains a subset of the potential output variables that

any model in the project might create. It is a prototype table that is used to define the project output variables and the variable attributes, such as data type and length.

A project output table can have one or more observations. Data in a project output table is not used by SAS Model Manager.

Before you can create a project, you create a project output table, register the table using the SAS Management Console, and add it as a SAS Model Manager data source. In SAS Model Manager, you can view project output tables in the Data Sources perspective **Project Output Tables** folder.

See Also

[“Creating Project Input and Output Tables” on page 31](#)

Scoring Task Input Tables

A scoring task input table is a SAS data set that contains the input data that is used in a scoring task.

Before you can create a scoring task, you create a scoring task input table, register the table in the SAS Metadata Repository, and add it as a SAS Model Manager data source. In SAS Model Manager, you can view scoring task input tables in the Data Sources perspective **Scoring Task Input Tables** folder.

See Also

[“Creating Scoring Task Input and Output Tables” on page 32](#)

Scoring Task Output Tables

A scoring task output table is used by a scoring task to define the variables for the scoring results table. As input, it is a prototype table that is used to define the scoring task output variables and the variable attributes, such as data type and length. Data that is in a scoring task output table is not used by SAS Model Manager.

Depending on the mode in which a scoring task is run, it might also be an output table. A SAS Model Manager scoring task can run in test mode, which is the default mode, or it can run in production mode. In both test mode and production mode, a scoring task output table is used by the scoring task to define the structure of the scoring results table. When the task runs, it creates a scoring results table. In test mode, the scoring results table is stored in the SAS Model Manager model repository, and you can view the table under the scoring task node in the version **Score** folder. The scoring task output table is not updated in test mode. In production mode, the contents of the scoring task output table are replaced by the contents of scoring results table. The scoring results table is not stored in the SAS Model Manager model repository.

Before you can create a scoring task, the scoring task output table must be added to the **Scoring Task Output Tables** folder in the Data Sources perspective. To add the table to SAS Model Manager, follow one of these processes:

- Add the table manually by creating the table, registering the table using SAS Management Console, and add the table in the **Scoring Task Output Tables** folder in the Data Sources perspective.
- Use the SAS Model Manager Create Output Table window. When you use the Create Output Table window, SAS Model Manager creates the table, registers the table using SAS Management Console, and adds the table to the **Scoring Task Output Tables** folder in the Data Sources perspective.

In SAS Model Manager, you can view scoring task output tables in the Data Sources perspective **Scoring Task Output Tables** folder.

See Also

[“Creating Scoring Task Input and Output Tables” on page 32](#)

Train Tables

A train table is a table that is used to build predictive models. Whether your models are created using SAS Enterprise Miner or you created SAS code models, you used a train table to create your model. SAS Model Manager uses this same train table. Ensure that your train table is registered in SAS Management Console and added as a SAS Model Manager data source.

Typically, you specify a train table as a version level property. By defining the train table at the version level, the table can be used to build all models that are defined in the version's **Models** folder.

In SAS Model Manager, train tables are used for information only with one exception. SAS Model Manager uses train tables to validate scoring results immediately after you create a Teradata scoring function and if the **Validate scoring results** box is checked when you publish Teradata scoring functions.

For information about registering train tables using SAS Management Console, see *SAS Model Manager: Administrator's Guide*.

Test Tables

A test table is used to create dynamic lift charts that can be used to identify the champion model. Test tables are typically a subset of a train table and are identical in table structure to the corresponding train table. Update test tables by creating a new subset of the corresponding train table.

Test tables must be registered in the SAS Metadata Repository and added as a SAS Model Manager data source in order to be viewed in SAS Model Manager. You can view test tables in the Data Sources perspective **Test Tables** folder.

After a test table is added to SAS Model Manager, you can specify the table in the **Default Test Table** field in the project properties.

For information about registering tables using SAS Management Console, see the *SAS Model Manager: Administrator's Guide*.

See Also

[“Creating a Test Table” on page 33](#)

Performance Tables

A performance table is a SAS data set that is used as the input table for each SAS Model Manager performance task. A performance task monitors a champion model's performance by comparing the observed target variable values with predicted target variable values. A performance table is a sampling of operational data that is taken at a single point in time. Each time that you run a performance task, you use a new performance table by taking a new sampling of the operational data. For example, a champion model is deployed to a production environment for the first time in March 2009. You might want to take a new

sampling of the underlying data in June 2009, September 2009, and January 2010. These new tables are the performance tables in the context of SAS Model Manager.

Performance tables must be registered using SAS Management Console and added as a SAS Model Manager data source in order to be viewed in SAS Model Manager. You can view performance tables in the Data Sources perspective, in the **Performance Tables** folder. After a performance table is added as a SAS Model Manager data source, you can specify the table in the **Default Performance Table** field in the project properties. The default performance table value at the project level is the default value for the **Performance data source** field in the Define Performance Task wizard.

Note: If you run the SAS Model Manager report macros outside of SAS Model Manager to monitor a champion model's performance, the macros do not access the performance tables in SAS Model Manager to create model performance monitoring reports.

See Also

[“Creating a Performance Table” on page 34](#)

See Also

- [“Add a Data Source Table” on page 35](#)
- [“Delete a Data Source Table” on page 36](#)

Creating Project Input and Output Tables

Create a Project Input Table

You can create a project input table either from the train table that you used to develop your model, or you can define the project variables in a DATA step. The project input table must include the input variables that are used by the champion model. Therefore, if you have several candidate models for your project, make sure that all candidate model input variables are included in the project input table. If you create the project input table from the train table, be sure to exclude the target variable from the project input table.

One method you can use to create the project input table from the train table is to use the SET statement to specify the train table and the DROP or KEEP statements to specify the variables from the train table that you want in the project input table. You can drop the target variable or keep all variables except the target variable.

This DATA step creates the project input table from the train table and drops the target variable BAD:

```
data hmeqtabl.invars;
  set hmeqtabl.training (obs=1);
  drop bad;
run;
```

This DATA step creates the project input table from the train table and keeps all variables except for the target variable BAD:

```
data hmeqtabl.invars;
  set hmeqtabl.training (obs=1);
  keep mortdue reason delinq debinc yoj value ninq job clno derog clag loan;
run;
```

You can also create the project input table using the LENGTH statement to specify the variables and their type and length. You could also specify the LABEL, FORMAT, or INFORMAT statements, or the ATTRIB statement to specify additional variable attributes. The following DATA step uses the LENGTH statement to specify the project input variables in the table:

```
data hmeqtabl.invars;
  length mortdue 8 reason $7 delinq 8
        debinc 8 yoj 8 value 8
        ning 8 job $7 clno 8 derog 8
        clag 8 loan 8;
run;
```

See Also

- *SAS 9.2 Language Reference: Dictionary*
- *SAS 9.2 Language Reference: Concepts*

Create a Project Output Table

You can create a project output table either from the train table that you used to develop your model, or you can define the project variables in a DATA step. The project output table includes only output variables that are created or modified by the champion model. Therefore, if you have several candidate models for your project, you must make sure that all project output variables are mapped to the champion model output variables.

To create the project output table using the training table, use the SET statement to specify the training table and the KEEP statement to specify the variables from the training table that you want in the project output table. The following DATA step creates the project output table hmeqtabl.outvars:

```
data hmeqtabl.outvars;
  set hmeqtabl.training (obs=1);
  %include "c:\temp\score.sas";
  keep score;
run;
```

The following DATA step creates the same project input table using the LENGTH statement to specify the output variable and its type and variable length:

```
data hmeqtabl.invars;
  length score 8;
run;
```

Creating Scoring Task Input and Output Tables

About Scoring Task Input and Output Tables

The scoring task input table is a data table whose input is used by the scoring task to score a single model. The scoring task input table must contain the variables and input data for the variables that the model requires. Typically, a scoring table is identical to its corresponding train table except that the target variables in the train table are not included in the scoring table.

A scoring task output table is a prototype table that defines one or more input and output variables whose values contain the scoring results. You can create a scoring output table using the Create Output Table function directly from the model in the Project Tree or you can write and execute a DATA step. If you create the table using the Project Tree, SAS Model Manager registers the table in the SAS Metadata Repository and adds the table as a SAS Model Manager data source. After SAS Model Manager creates the table, the table can be selected as the output table for subsequent scoring tasks. If you create the scoring output table using a DATA step, you must register the table using SAS Management Console and add the table as a SAS Model Manager Scoring Task Output Table data source.

Create a Scoring Task Input Table

This DATA step creates a scoring task input table from customer data, keeping 500 rows from the train table:

```
data hmeqtabl.scorein;
    set hmeqtabl.customer (obs=500);
    keep mortdue reason delinq debinc yoj value ninq job clno derog clage loan;
run;
```

Create a Scoring Task Output Table

You can create a scoring output table using the Create Output Table window that you open from the Project Tree. The Create Output Table window provides a graphical interface for you to select the variables that you want to include in your scoring output table. SAS Model Manager registers the table in the SAS Metadata Repository and adds the table as a SAS Model Manager data source. For information, see [“Create Scoring Output Tables” on page 111](#).

You can also create a scoring task output table using a DATA step to keep or drop variables from the train table.

The input variables that you might want to keep in the output data set are key variables for the table. Key variables contain unique information that distinguishes one row from another. An example would be a customer ID number.

This DATA step keeps the input variable CLNO, the client number, which is the key variable, and the output variable SCORE:

```
data hmeqtabl.scoreout;
    length clno 8 score 8;
run;
```

Creating a Test Table

The test table is used during model validation by the Dynamic Lift report. You can create a test table by taking a sampling of rows from the original train table, updated train table, or any model validation table that is set aside at model training time. This DATA step randomly selects approximately 25% of the train table to create the test table:

```
data hmeqtabl.test;
    set hmeqtabl.train;
    if ranuni(1234) < 0.25;
run;
```

See Also

[“Creating Dynamic Lift Reports” on page 131](#)

Creating a Performance Table

About Performance Tables

A performance table requires the same variable structure as a train table. That is, it requires the champion model input variables and a target variable. The value of the target variable must be a binary value, such as Yes or No.

You create a performance table by taking a sampling of data from an operational data mart. Make sure that your sampling of data includes the target or response variables. The data that you sample must be prepared by using your extract, transform, and load business processes. When this step is complete, you can then use that data to create your performance table.

As part of the planning phase, you can determine how often you want to sample operational data to monitor the champion model performance. Ensure that the operational data that you sample and prepare represents the period that you want to monitor. For example, to monitor a model that determines whether a home equity loan could be bad, you might want to monitor the model every six months. To do this, you would have two performance tables a year. The first table might represent the data from January through June, and the second table might represent the data from July through December.

Here is another example. You might want to monitor the performance of a champion model that predicts the delinquency of credit card holders. In this case, you might want to monitor the champion model more frequently, possibly monthly. You would need to prepare a performance table for each month in order to monitor this champion model.

In addition to planning how often you sample the operational data, you can also plan how much data to sample and how to sample the data. Examples in this section show you two methods of sampling data and naming the performance tables. You can examine the sampling methods to determine which might be best for your organization.

Naming a Performance Table for Use with the Define Performance Task Wizard

The Define Performance Task wizard is a graphical interface to assist you in creating a performance task to monitor the champion model performance. When you run the Define Performance Task wizard, you specify a performance table that has been registered using SAS Management Console and added as a SAS Model Manager data source. When you create a performance table, you can name the performance table in a way that best suits your business process:

- Each time that you create a performance table you create a unique name for the table and store the table in any location that is accessible to SAS Model Manager. In this case, you register each table in SAS Management Console and add each table as a SAS Model Manager performance data source. Whenever you run the Define Performance Task wizard for this project, you would specify the appropriate performance data source.
- The first time that you create a performance table, you register the table using SAS Management Console and add the table as a Performance data source in SAS Model

Manager. For each subsequent performance table that you create, you reuse the performance table name and store the table in the location where it is registered in SAS Management Console. This step makes the table available to SAS Model Manager. In this case, you would register the performance table only once in SAS Management Console and add the table only once in SAS Model Manager. In the Define Performance Task wizard, you would use the same performance data source name each time that you run the wizard for that project. Although the performance data source name remains the same, the data in the table is specific for the time period.

For more information, see [“Determine How to Use the Performance Data Sets”](#) on page 198.

Create a Performance Table

You can use the following DATA steps as examples to create your performance tables.

This DATA step creates a performance table using 5,000 sequential observations from the operational data:

```
data hmeqtabl.perform;
    set hmeqop.JulDec (firstobs=12001 obs=17000);
run;
```

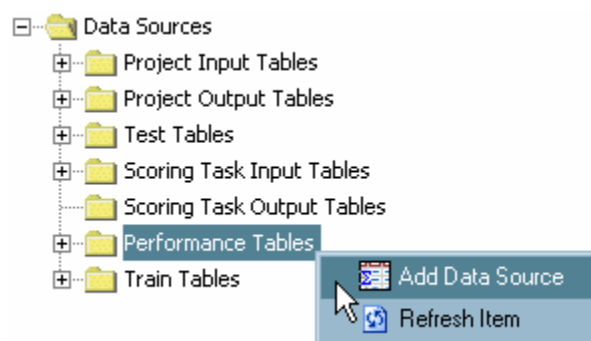
This DATA step creates a performance table from operational data for the last six months of the year. The IF statement creates a random sampling of approximately 10% of the operational data:

```
data hmeqtabl.perform;
    set hmeqop.JulDec;
    if ranuni(1234) < 0.1;
run;
```

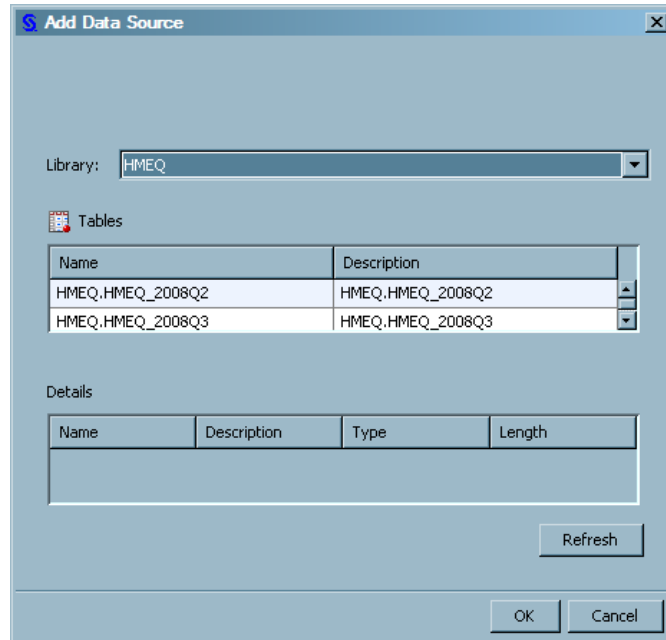
Add a Data Source Table

To add a data source table, follow these steps:

1. Click the **Data Sources** perspective button.



2. Right-click the type of data source that you want to add and select **Add Data Source**. The Add Data Source window opens.

Display 3.1 Add Data Source


3. Click **Refresh** to make sure that the Add Data Source window lists new tables that have been added to the **Data Libraries Manager** folder in SAS Management Console.
4. In the **Library** list box, select the SAS library that contains the table that you want to add.
5. In the **Tables** section, select the table that you want to add. The **Details** section lists the variables in that table.
6. Click **OK**.

See Also

- [“Project Tables” on page 28](#)
- [“Delete a Data Source Table” on page 36](#)

Delete a Data Source Table

To delete a data source table, follow these steps:

1. Select the Data Sources perspective button .
2. Expand the data source folder that contains the table that you want to delete.
3. Right-click the table and select **Delete Item**. SAS Model Manager deletes the table.

A table remains available to be added again as a SAS Model Manager data source table as long as it is registered in SAS Management Console.

See Also

- [“Project Tables” on page 28](#)
- [“Add a Data Source Table” on page 35](#)

Chapter 4

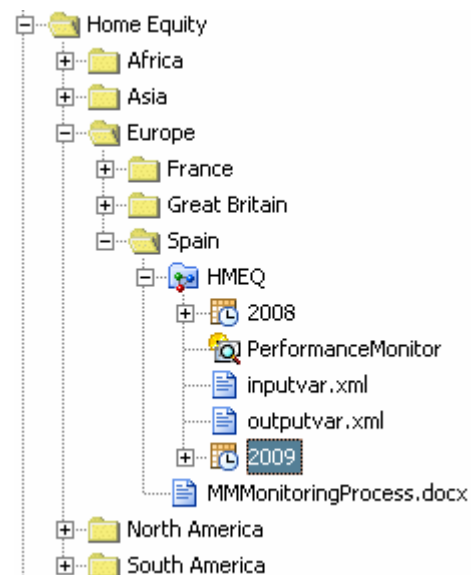
Organizing the Project Tree

Overview of the Project Tree	39
Create an Organizational Folder	40
Associate Documents with a Folder	41
About Associating Documents	41
Attach a Document to a Folder	41
Show Document Versions and View or Save Documents	41
Deleting an Object in the Project Tree	42

Overview of the Project Tree

The SAS Model Manager repository organizes projects by using folders in the Project Tree. The root node of the tree, **MMRoot**, represents the repository. From the root node, you add organizational folders to organize your modeling projects. Create as many organizational folder levels as you need. Then, you can create projects or attach associated documents to a folder.

The following example shows how folders can be organized for a global business:



The **Home Equity** folder contains subfolders for five continents. In the **Europe** folder, the **Spain** folder contains the project **HMEQ** and the attached document **MMMonitoringProcess.docx**. The **HMEQ** project has two versions, **2008** and **2009**.

To learn more about the tasks that you can perform to organize the Project Tree, see the following topics:

- “Create an Organizational Folder” on page 40
- “Create a Project” on page 47
- “Associate Documents with a Folder” on page 41
- “Create a Version” on page 67

Here are some other common tasks that you can perform on an organizational folder, a project folder, or a version folder:

- Publish models. For more information, see “Publishing Models” on page 158.
- Search for models or tasks that are assigned to SAS Model Manager users. For more information, see “Query Utility” on page 235.

Create an Organizational Folder

To create a folder, follow these steps:

1. Right-click the **MMRoot** node or a folder and select **New Folder**. The New Folder window opens.

General Properties	
Name *	DDHMEQ
Description	Home Equity

2. In the **Name** field, type a folder name. This is a required field.
3. (Optional) In the **Description** field, type a folder description.
4. Click **OK**. SAS Model Manager adds the folder to the Project Tree.

Associate Documents with a Folder

About Associating Documents

You associate documents that are stored externally to SAS Model Manager and that are related to your modeling project by attaching them to an organizational folder, a **Document** folder, or a **Resources** folder. For example, you might attach a project plan so that you can reference the plan while you are working within SAS Model Manager.

You cannot edit an attached document in SAS Model Manager. You can attach updated files of the same name to the same folder. To make changes to a document, make the changes to the file on your computer and then reattach the document in the Project Tree. Each time that you reattach a document, SAS Model Manager creates a new version of the document. All historical files are saved.

The Show History window displays the versions of a document that have been attached to a folder in the Project Tree. The **Create Date** column in the Show History window is the date that the document was attached to the folder. From the Show History window you can view the document or save a version of a document.

Attach a Document to a Folder

To attach a document to a folder, follow these steps:

1. Right-click the folder and select **Attach**.



The Open dialog box opens.

2. In the Open dialog box, select the document that you want to attach to the folder.
3. Click **OK**. SAS Model Manager attaches the document to the folder.

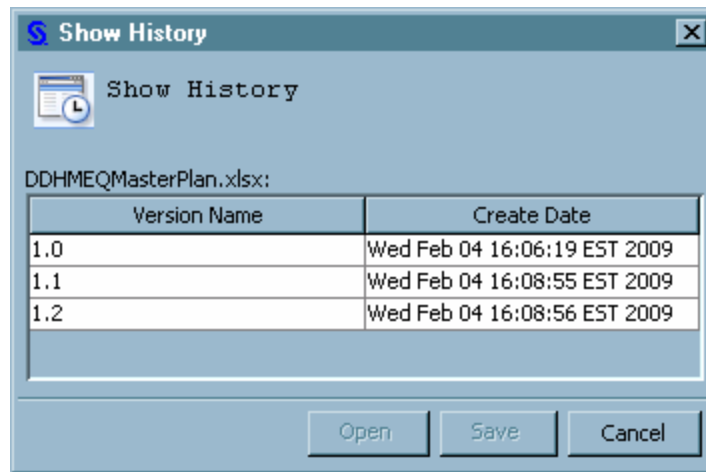
Here is an example of a document that is attached to a folder:



Show Document Versions and View or Save Documents

To show all of a document's versions, view a document, or save a document follow these steps:

1. Right-click the document and select **Show History**. The Show History window opens. The Show History window displays all of a document's versions.



2. To view one of the document versions, select the version and then click **Open**. If the application that created the document is installed on your computer, the document opens.
3. To save a document version, select the version and click **Save**. Complete the location and filename information in the Save dialog box and click **Save**.

Deleting an Object in the Project Tree

Only certain objects in the project tree can be deleted. You can delete an organizational folder, a project, a version that is not frozen, and objects under a version component. For example, you can delete an individual model, but you cannot delete the **Models** folder under a version.

To delete an object, follow these steps:

1. Right-click the object and select **Delete Item**.

Note: When a version is frozen, **Delete Item** is disabled for the frozen version as well as all models in the **Models** folder.
2. Click the **Yes** button in the Delete Item message to confirm the deletion of the object.

Chapter 5

Working with Projects

Overview of Projects	43
What Is a SAS Model Manager Project?	43
How a Project Folder Is Organized	44
Project Folder Tasks	44
Planning a Project	45
Prerequisites for Creating Projects	46
Create a Project	47
Setting the Project Champion Model Status	48
About the Champion Model Status	48
Setting the Champion Model Status	48
Project Properties	49
About Project Properties	49
Project-Specific Properties	49

Overview of Projects

What Is a SAS Model Manager Project?

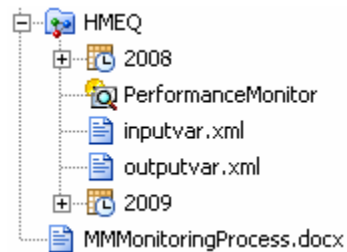
A SAS Model Manager project consists of the models, reports, documents, scoring tasks, and other resources that you use to determine a champion model. For example, a banking project might include models, data, and reports that are used to determine the champion model for a home equity scoring application. The home equity scoring application predicts whether a bank customer is an acceptable risk for granting a home equity loan.

Project work is organized in one or more time-based intervals that are called versions. Each version contains the documents, models, reports, resources, scoring tasks, and performance information for a time interval. Each version can have its own life cycle definition for tracking the progress of a project. For more information, see [“Working with Versions” on page 53](#).

Project models can be imported from SAS Enterprise Miner or you can create your own models.

How a Project Folder Is Organized

The SAS Model Manager project is a folder  that contains versions, the project input and output XML files, a **PerformanceMonitor** folder and optional attached documents. Here is the Project Tree view of a project:



In this Project Tree, the project name is HMEQ.

Here is a description of project folder components:

versions

are time-phased containers for your project. It contain life cycle information, candidate model files, model comparison reports, resource files, scoring tasks, and model performance reports. This Project Tree has two versions, 2008 and 2009.

PerformanceMonitor

is used to execute the SAS code that creates the performance monitoring report data sets.

inputvar.xml

is an XML file that contains information about the project input variables and their attributes. This file is based on the Project Input Table and is created when SAS Model Manager creates the project.

outputvar.xml

is an XML file that contains information about the project output variables and their attributes. This file is based on the Project Output Table and is created when SAS Model Manager creates the project.

document files

are any files that you attach to the project folder. You can view only documents that you attach. You cannot update them. In this Project Tree, the document file is MMMonitoringProcess.docx.

Project Folder Tasks

When you right-click the project folder, SAS Model Manager provides the following project tasks:

New Version

creates a new time-phased version to contain life cycle information, candidate model files, model comparison reports, resource files, scoring tasks, and model performance reports. For more information, see [“Create a Version” on page 67](#).

Publish

publishes models to defined channels and notifies subscribers of the publication channel when the models are delivered. For more information, see [“Publish a Model to a Channel ” on page 159](#)

Export Project Champion Model

exports the champion model metadata to the SAS Metadata Repository in order for the champion model to run in a test or production scoring environment. For more information, see [“Export a Model” on page 162](#).

Publish Teradata Scoring Function

transforms DATA step score code of a predictive model to Teradata user-defined function (UDF) executable files and writes the UDF files to the Teradata Enterprise Data Warehouse. For more information, see [“How to Publish a Teradata Scoring Function” on page 171](#).

Define Performance Task

starts a wizard that generates the SAS code to monitor the performance of a champion model. For more information, see [“Overview of Creating Reports Using a Performance Task” on page 197](#).

Query

searches for models and tasks that are assigned to SAS Model Manager users. For more information, see [“Overview of the Query Utility” on page 235](#).

Planning a Project

Before you begin a project, you must plan your project resources. Here is a list of project resources that must be available for a modeling project:

- After you know which users are assigned to a project, a SAS Model Manager administrator must ensure that the user is assigned to the appropriate SAS Model Manager user group and role. For more information, see [“SAS Model Manager User Groups, Roles, and Tasks” on page 18](#) and the *SAS Model Manager: Administrator's Guide*.
- How do you want to structure your project in the Project Tree? A project is a subfolder of an organizational folder. The Project Tree enables multiple levels of organizational folders so that you can customize how you structure the Project Tree. For example, your Project Tree could be similar to your business departmental hierarchy or it could list individual project names. For more information, see [“Organizing the Project Tree” on page 39](#).
- What models do you want to use in the project? If the models were created using SAS Enterprise Miner, all model components are available to SAS Model Manager when you import the model. If your model is a SAS code model, you must ensure that you have imported all of the model component files. For more information, see [“Import SAS Code Models” on page 83](#).
- Before you can create a project, the project input table and the project output table must be added as a SAS Model Manager data source. For more information, see [“Creating Project Input and Output Tables” on page 31](#).
- Before you create a project version, you must plan your milestones and the tasks for each milestone. When you have that information, you then create a life cycle template. The life cycle template enables you to assign users to complete projects and to monitor the progress of your project. For more information, see [“Creating Life Cycle Templates” on page 56](#).
- You might have project documents that you would like to access from SAS Model Manager. SAS Model Manager enables you to attach documents to organizational folders or to a version Documents folder in the Project Tree. You can view these

documents in SAS Model Manager only. For more information, see [“Associate Documents with a Folder” on page 41](#).

- SAS Model Manager provides several reports that you can use to help you to assess candidate models. You can review the types of reports that are available and plan for which reports you want to use. Your plans might also include a custom report that you can run in SAS Model Manager. For more information, see [“Validating Models Using Comparison Reports” on page 125](#) and [“Validating Models Using User Reports” on page 135](#).
- When you export a project to the SAS Metadata Repository, you must specify a folder to export the project to. You might need to create a folder in the SAS Metadata Repository, if one does not already exist. For more information, see [“Exporting Models” on page 162](#).
- After your champion model is in a production environment, you can monitor the performance of the model in SAS Model Manager using your organization's operational data. If you use SAS Model Manager to define and execute performance tasks, you must first prepare performance tables using the operational data and add them as a SAS Model Manager performance data source. For more information, see [“Creating a Performance Table” on page 34](#).
- When you run performance monitoring reports, you can set up performance index alert and warning conditions to notify users if conditions exceed the indexes. For more information, see [“Performance Index Warnings and Alerts” on page 188](#).

Prerequisites for Creating Projects

Projects can be created only by SAS Model Manager administrators and SAS Model Manager advanced users. Ensure that users who create projects are assigned to the group **Model Manager Administrator Users** or **Model Manager Advanced Users** in SAS Management Console.

All modeling projects require three project properties. You must know the property values before you create a project in SAS Model Manager:

- the name of the model function that the models perform, such as classification, prediction, or segmentation
- the name of the project input table, which is a SAS data set that defines all possible input variables that can be used by project models
- the name of the project output table, which is a SAS data set that defines all possible output variables that must be created by all champion models within a SAS Model Manager project

SAS Model Manager has several model function types:

- Analytical
- Prediction
- Classification
- Segmentation

To determine the model function type for your project, compare your model to the descriptions in the table [Types of Model Functions on page 51](#).

The project input table and the project output table must be created, registered in SAS Management Console, and added as a SAS Model Manager data source before you can

create a project. In order for project input and output tables to be accessible to SAS Model Manager, you must prepare the tables beforehand. See the following documents for details:

- For instructions about creating project input and output tables, see the topic [“Creating Project Input and Output Tables”](#) on page 31.
- The *SAS Model Manager: Administrator's Guide* has instructions on registering project input and output tables in SAS Management Console.
- For instructions about adding project input and output tables as SAS Model Manager data sources, see [“Add a Data Source Table”](#) on page 35..

Create a Project

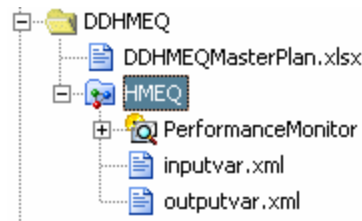
To create a project, follow these steps:

1. Right-click the organizational folder in the Project Tree and select **New** ⇒ **New Project**. The New Project window opens.

New Project	
Enter a project name and description.	
<div>General Properties</div> <div> <div>Name *</div> <div>HMEQ</div> </div> <div> <div>Description</div> <div></div> </div>	
<div>Project Properties</div> <div> <div>Model Function *</div> <div>classification</div> </div> <div> <div>Project Input Table *</div> <div>hmeq.HMEQ</div> </div> <div> <div>Project Output Table *</div> <div>hmeq.HMEQ</div> </div>	
<div>OK</div> <div>Cancel</div>	

2. Enter a name and a description for the project that you are creating. The **Name** field is required.
3. Complete the **Project Properties** section of the **New Project** window. For more information, see [“Specific Properties for a Project”](#) on page 285.
 - a. Click the **Model Function** list box and select the function type for the model.
 - b. Click the **Project Input Table** list box and select a project input table.
 - c. Click the **Project Output Table** list box and select a project output table.
4. Click **OK**. SAS Model Manager creates a new project folder in the Project Tree.

Here is an example of a project in the Project Tree:



When SAS Model Manager creates a project folder, it creates a **PerformanceMonitor** node. It uses the project input and output variable tables to create the two XML files.

- You use the **PerformanceMonitor** node to execute the SAS programs that create performance monitoring reports.
- The input and output XML files are used as project input and output metadata and are published or exported as part of the model if the model is published or exported from the project folder.

Setting the Project Champion Model Status

About the Champion Model Status

You can set the project **State** property to indicate the status of the champion model for the project. Valid values are:

In Development

Indicates that the project has started but a champion model is not yet in production.

Active

Indicates that a champion model for this project is in production.

Inactive

Indicates that the champion model is temporarily suspended from production.

Retired

Indicates that the champion model for this project is no longer in production.

Setting the Champion Model Status

To set the champion model status, follow these steps:

1. Select the project. The project properties appear.
2. Click the State property and select the state of the champion model.

State	In Development
Default Channel	In Development
Default Version	Active
Model Function	Inactive
	Retired

Project Properties

About Project Properties

Project properties contain the project metadata. Project metadata includes information such as the name of the project, the project owner, the project identifier, the name and path of the SAS Model Manager repository, and tables and variables that are used by project processes.

Project properties are organized into several types:

- **General Properties**
- **System Properties**
- **Specific Properties**
- **User-Defined Properties**

The **General Properties** and **System Properties** are system-defined properties that you cannot modify, except for the description of the folder. **Specific Properties** contain information about tables that are used by the project as well as various input and output variables and values that are used in scoring the models in test and production environments. You can add your own project properties under **User-Defined Properties**. The property-value pair is metadata for the project. The background color of a property distinguishes whether you can modify a property value. You only modify the fields that are white. When you click in a field, you either enter a value or select a value from the list box.

Project-Specific Properties

Here is a list of the specific properties for a project.

Property Name	Description
Project Input Table	Specifies the input variables that can be used by all models within a SAS Model Manager project.
Project Output Table	Specifies the list of output variables that must be created by all models within a SAS Model Manager project. If a model output variable is not the same as the project output variable, you can select the model in the Project Tree and map the model output variable to the project output variable.
Default Test Table	Specifies a default SAS data set that can be used to create model assessment reports such as dynamic lift charts.

Property Name	Description
Default Scoring Task Input Table	Specifies a default SAS data set that is used as the input data table for all scoring tasks within the SAS Model Manager project. If you specify a value for the Default Scoring Task Input Table property, the value is used as the default input table in the New Scoring Task window.
Default Scoring Task Output Table	Specifies a default SAS data set that defines the variables to keep in the scoring results table and the scoring task output table. If you specify a value of the Default Scoring Task Output Table property, the value is used as the default output table in the New Scoring Task window.
Default Performance Table	Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager project. The value of the Default Performance Table property is used as the default value for the Performance data source field in the Define Performance Task wizard if a default performance table is not specified for a version or for the model.
Default Train Table	The train table is optional and is used only as information. However, when a value is specified for a model's Default Train Table property, it is used to validate Teradata scoring functions when a user publishes the associated project champion model to a Teradata database. The value of the Default Train Table property is used to validate Teradata scoring functions only if a default train table is not specified for a version or for the model.
State	Specifies the current state of the project: In Development specifies the time period from the project start to the time where the champion model is in a production environment. Active specifies the time period where the champion model is in a production environment. Inactive specifies the time period when a project is temporarily suspended from the production environment. Retired specifies that the champion model for this project is no longer in production.
Default Channel	Specifies the channel that is used, by default, to publish a project. The default channel appears in the Channel box of the Channel Usage window.

Property Name	Description
Default Version	Specifies the version that contains the champion model in a production environment.
Model Function	Specifies the type of output that your predictive model project generates. The Model Function property that you specify affects the model templates that SAS Model Manager provides when you are ready to import models into one of your project's version folders. Once declared, the Model Function property for a project cannot be changed. Ensure that the types of models that you are going to use in the project fit within the selected model function type. For more information about the types of model functions, see Types of Model Functions on page 51 .
Interested Party	Specifies any person or group that has an interest in the project. For example, an interested party would be the business department or the business analyst whose request led to the creation of a SAS Model Manager project.
Training Target Variable	Specifies the name of the target variable that was used to train the model.
Target Event Value	The target variable value that defines the desired target variable event.
Class Target Values	For class, nominal, or interval targets, the set of possible outcome classes, separated by commas. For example, binary class target values might be 1, 0 or Yes, No . Nominal class target values might be Low, Medium, High . These values are for information only.
Output Event Probability Variable	The output event probability variable name, when the Model Function property is set to Classification.
Output Segmentation Variable	The output segmentation variable name, when the Model Function property is set to Segmentation.

Table 5.1 Types of Model Functions

Model Function	Description	Example
Analytical	Function for any model that is not Prediction, Classification, or Segmentation.	None

Model Function	Description	Example
Prediction	Function for models that have interval targets with continuous values.	The score output of a prediction model could estimate the weight of a person. The output of a model would be P_Weight.
Classification	Function for models that have target variables that contain binary, categorical or ordinal values.	DEFAULT_RISK = {Low, Med, High}
Segmentation	Function for segmentation or clustering models.	Clustering models
Any	Specify Any when you import a SAS code model and you want a choice of the model template to use in the Local Files window. When you specify Any , SAS Model Manager lists the available model templates in the Choose a model template list in the Local Files window.	None

Chapter 6

Working with Versions

Overview of Versions	53
What Is a SAS Model Manager Version?	53
How a Version Folder Is Organized	54
Version Folder Tasks	55
Creating Life Cycle Templates	56
Overview of Creating Life Cycle Templates	56
The Middle-Tier Server User-Templates Directory	56
The SAS Model Manager Template Editor Window	57
Life Cycle Template Participants	58
Create a New Life Cycle Template	60
Modify a Life Cycle Template	62
Create a New Version of a Life Cycle Template	63
Life Cycle Template Properties	64
Create a Version	67
Version Properties	68
About Version Properties	68
Version-Specific Properties	68
Working with Life Cycles	70
Overview of a Life Cycle	70
Life Cycle Tasks	71
Life Cycle Properties	73

Overview of Versions

What Is a SAS Model Manager Version?

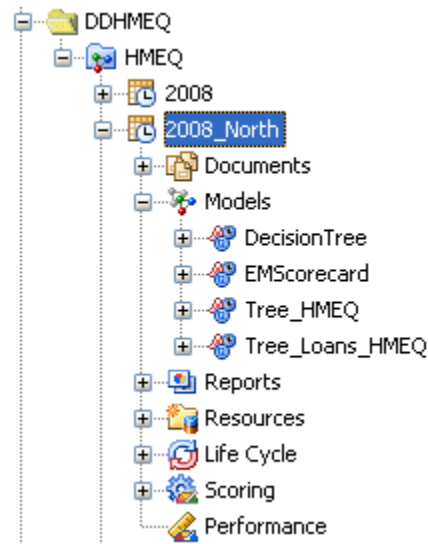
After a project is created, you create a version folder to import your models, score your models, run reports, and monitor the life cycle of these models. A version is often the time-phased container for your SAS Model Manager projects. The time interval for a project cycle is specified when you create the version, and it might represent a calendar year, a retail season, or a fiscal quarter. A SAS Model Manager project can contain multiple versions. A version contains all of the candidate model resources that you need to determine a champion model as well as all champion model resources. For example, you might develop models for a scoring program that determines whether a customer is eligible for a home equity loan. The version folder could be named 2009. The version contains all of the models, scoring tasks, and reports that are used to determine the champion model. After

you select the champion model, the subsequent tasks in a milestone are based on the champion model.

To import a model, you must have at least one version in the project.

How a Version Folder Is Organized

The SAS Model Manager version folder contains life cycle information, auxiliary version documents, candidate model files, model comparison reports, resource files, scoring tasks, and model performance reports. A typical version folder for a project might contain the following:



The SAS Model Manager life cycle template that is associated with a version determines the milestones and tasks that you complete to develop and implement the scoring model.

SAS Model Manager provides the following functionality for a version:

Documents

contains presentations, rosters, documentation, schedules, and other digital information that is related to the version that users can easily access.

Life Cycle

contains the milestone phases and tasks that your organization uses to monitor the modeling process. A time-phased road map, called a life cycle template, specifies the milestone tasks that are required to implement model life cycle activities. Typical milestone phases for the life cycle of a version are Development, Test, Stage, Production, and Retire. SAS Model Manager provides example life cycle templates that you can use as a model to create your own templates. Templates that are provided by SAS Model Manager cannot be modified. SAS Model Manager administrators and advanced users can use the SAS Model Manager Template Editor to customize life cycle templates. You must create a life cycle template for your version before you create the version.

Models

contains the imported candidate models and a champion model after it is selected. Each model contains the files that SAS Model Manager uses to run model reports and scoring tasks.

Reports

contains generated reports. Each report contains the report results, the SAS program that created the report, and a SAS log.

Resources

contains data files that SAS Model Manager creates and uses to monitor the performance of the champion model. The folder can also contain user-defined formats that the version models require and version resource files that are not in the **Documents** folder.

Scoring

contains scoring task definitions and files that are generated when model scoring tasks are completed, such as output data and statistics.

Performance

displays model performance monitoring charts when you select the **Performance** node. The data sets that create the charts are stored in the **Resources** folder under each version folder.

Version Folder Tasks

When you right-click the version folder, SAS Model Manager provides the following functionality for a version:

Set Default Version

selects the version as containing the champion model for the project. You typically set a version as the default version for the project when the champion model is ready to be deployed to a production environment. When you export the project champion model from the project folder, SAS Model Manager exports the champion model in the default version. For more information, see [“The Default Version for a Project” on page 154](#).

Clear Default Version

deselects the version as the default version. For more information, see [“Clear a Default Version” on page 155](#).

Freeze Version

disables modifications of some version models properties and files. You typically freeze a version after you declare a champion model and set the project default version in preparation for deploying the champion model to a production environment. Freezing a version restricts the activities that you do with the folder. After a version folder is frozen, you cannot import additional models or change the champion model. You can continue to add files to the **Documents**, **Reports**, **Resources**, and **Scoring** folders. For more information, see [“Freezing Models” on page 152](#).

Unfreeze Version

enables modifications of version model properties and files. For more information, see [“Unfreeze a Version” on page 153](#).

Publish

publish models to defined channels and notifies subscribers of the publication channel when the models are delivered. For more information, see, [“Publishing Models” on page 158](#).

Query

search for model and tasks that are assigned to SAS Model Manager users. For more information, see [“Query Utility” on page 235](#).

See Also

- “Overview of Importing Models” on page 79
- “Exporting Models” on page 162

Creating Life Cycle Templates

Overview of Creating Life Cycle Templates

A life cycle template is an XML file that defines the milestones and tasks that must be completed in order to place a champion model in a production environment, and to monitor and retire that model. You determine the milestones and tasks for a version in a version planning phase. For each task, you can define dependent tasks and assign users to complete or approve the tasks. By assigning a weight to each task in a milestone, you can track the progress of completing a milestone.

You create a life cycle template for a version before you create a version. A life cycle template is typically shared by multiple versions. When you create a version, the life cycle template that you want to use must be available from the Life Cycle perspective. Templates that appear in the Life Cycle perspective are the life cycle templates that are stored in the middle-tier server user-templates directory.

SAS supplies four life cycle templates that you can use to create a template: Basic, Standard, Extended, and User Lifecycle templates. These life cycle templates are provided as examples only and are not intended for use by any organization. They cannot be modified.

To create a life cycle template, you can use the SAS Model Manager Template Editor or you can create a life cycle template XML file using a text editor.

The SAS Model Manager Template Editor uses standard windowing techniques to access pop-up menus and selection lists. The template editor automatically generates milestones and task identification numbers. The editor provides a list of SAS Model Manager users, also known as participants, for you to choose for task assignments. When you save your template, SAS Model Manager saves the template as an XML file using the required XML element structure.

If you create a life cycle template using an XML file, you can copy any life cycle template from the user-templates directory and modify the file with any text editor. When you modify an XML template file, you specify the milestone and task properties as XML element and element attributes. SAS Model Manager does not generate participant identification numbers or participant lists. You must specify them explicitly in the XML file.

The Middle-Tier Server User-Templates Directory

All user templates, life cycle templates, and model templates must be stored in the middle-tier server directory `\sasconfigdir\Lev#\AnalyticsPlatform\apps\ModelManager\ext`. `sasconfigdir` is the middle-tier server name and SAS configuration directory. `Lev#` is a level number that is determined during installation. The following path is an example of a path to the middle-tier user-templates directory:

```
\\myserver\SASConfiguration\Lev1\AnalyticsPlatform\apps
\ModelManager\ext
```

This directory is examined each time SAS Model Manager starts. SAS Model Manager displays in the Life Cycle perspective all life cycle templates that are in the user-templates directory.

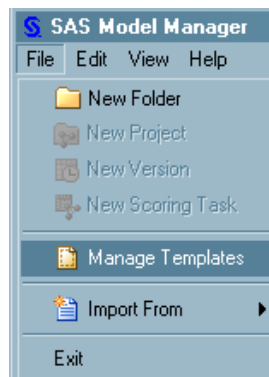
Before you create a new life cycle template by using the Life Cycle Template Editor or by modifying an XML file, you can copy any life cycle template from this directory to your local computer. Modify the template on your computer. The template can then be stored in the user-templates directory.

Only two groups can deploy templates to the user-templates directory: users in the Model Manager Administrator Users group and users in the Model Manager Advanced Users group who have Write access to the middle-tier server where SAS Model Manager is installed. For information about deploying templates, see the *SAS Model Manager: Administrator's Guide*.

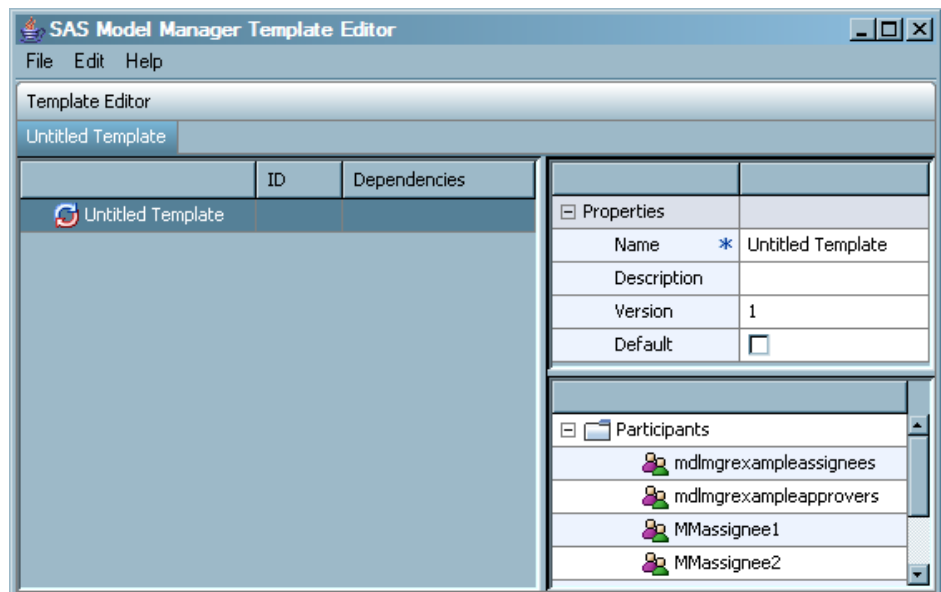
The SAS Model Manager Template Editor Window

To open the SAS Model Manager Template Editor window, follow these steps:

1. From the SAS Model Manager window, select **File** ⇒ **Manage Templates**.



2. To open a new template, in the template editor window select **File** ⇒ **New Life Cycle Template**. An empty template opens.



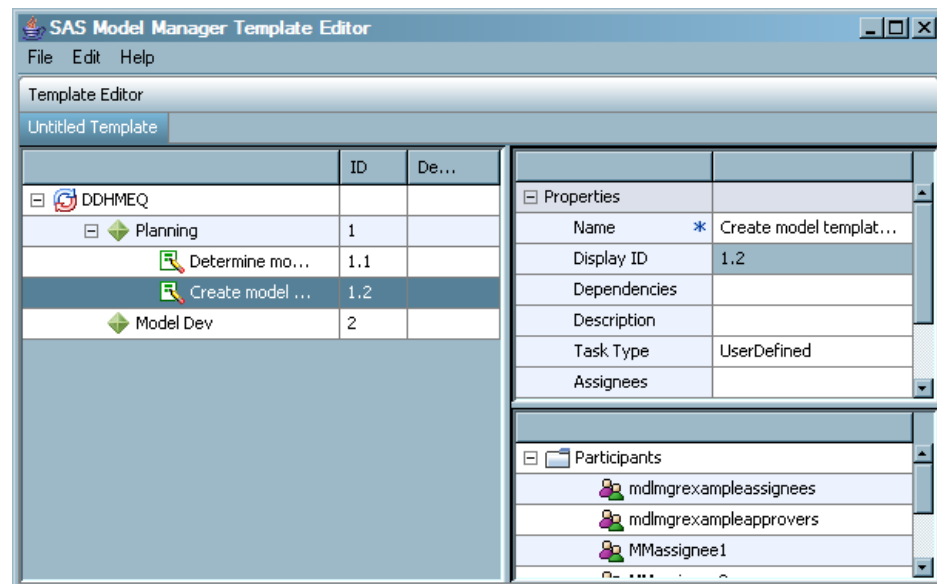
To open an existing template, select **File** ⇒ **Open**. Select the path and file and click **Open**.

When you open the SAS Model Manager Template Editor window to access a new or existing life cycle template, the editor displays three panes:

- the left pane that displays the life cycle template milestones and tasks
- the upper right pane that displays the properties for the life cycle template or the selected milestone or task
- the lower right pane that displays the list of participants. Participants are SAS Model Manager users and groups.

When you open a new template, the tab at the top of the left pane is titled **Untitled Template**. The tab name changes to the template name when you save the template. The template name appears in the tab and as the root node in the life cycle template tree. The life cycle template tree has three nodes:

- The root node is the name of the template.
- Milestone nodes appear under the root node.
- Task nodes appear under milestone nodes.



When you select a node in the tree, the properties for that node appear in the upper right pane. Required properties are indicated by a blue star. * For a description of life cycle template properties, see [“Life Cycle Template Properties”](#) on page 64.

The lower right pane displays the life cycle participants, who are users and user groups who can be assigned to a task or who can be assigned to mark a task complete.

Life Cycle Template Participants

Participant Roles

The following roles are used to determine who can be assigned to complete a task or who can mark a task complete:

- **Model Manager: Usage** is assigned to all SAS Model Manager users and groups.
- **Model Manager: Life Cycle Participant Usage** is assigned to SAS Model Manager users and groups whose user ID or group ID appears in the Life Cycle Template Editor **Participant** list. Only users and groups that are assigned this role for a life cycle, and

are in the Participant list can be assigned to the roles Model Manager: Life Cycle Assignee Usage and Model Manager: Life Cycle Approval Usage for the life cycle.

- **Model Manager: Life Cycle Assignee Usage** is assigned to users and groups to complete a task. Users who are assigned this role can be assigned to update the task **Status** field to **Not Started**, **Started**, and **Completed**.
- **Model Manager: Life Cycle Approval Usage** is assigned to users and groups who can mark a task complete. Users who are assigned this role can be assigned to update the task **Status** field to **Approved**.

When you open the template editor, the users and groups that are assigned life cycle roles appear in the **Participants** list. You cannot add or delete users and groups from the Participants list. A best practice is to ensure that all users and groups have the appropriate life cycle roles assigned to them before you create a life cycle template in the template editor.

Selecting Life Cycle Participants

When you open the SAS Model Manager Life Cycle Template Editor, the **Participants** list displays the SAS Model Manager users and groups that have been assigned the role **Model Manager: Life Cycle Participant**. Only users and groups in this list can be assigned to complete a task or approve a task.

You designate a user or group to complete a task in the **Assignee** template property. You designate a user or group to approve a task in the **Approver** template property. When you click the ellipsis button for the **Assignee** or **Approver** properties, the Select Participants window displays the users and groups that can be assigned to those tasks.

The participants that you select in the Select Participants window determine the users that appear in the task properties **To Be Completed By** and **To Be Approved By**. These users appear in the task properties after the life cycle template has been deployed and specified as a version life cycle.

- If any user or group that you select in the Select Participants window is assigned the role **Model Manager: Life Cycle Assignee Usage**, then only the selected users and groups that have that role appear as values for the **To Be Completed By** task property. Users that you select do not appear in the **To Be Completed By** task property if they are not assigned that role.
- If no user or group is assigned the role **Model Manager: Life Cycle Assignee Usage**, then all template participants appear as values for the **To Be Completed By** task property.
- If any user or group that you select in the Select Participants window is assigned the role **Model Manager: Life Cycle Approver Usage**, then only the selected users and groups that have that role appear as values for the **To Be Approved By** task property. Users that you select do not appear in the **To Be Completed By** task property if they are not assigned that role.
- If no user or group is assigned the role **Model Manager: Life Cycle Approver Usage**, then all template participants appear as values for the **To Be Approved By** task property.
- If the **Assignee** or **Approver** template properties are not assigned to any user or group, then all template participants appear as values for their respective version life cycle task properties **To Be Completed By** or **To Be Approved By**.

If you select a group to complete a task or approve a task, any or all members of the group can be responsible for completing the task or marking that the task is complete. The group members have the authorization to update the task **Status** field. However, only one member needs to set the corresponding milestone task to **Completed** or **Approved**.

Note: The SAS Model Manager administrator has permission to set any life cycle property value.

Using Groups as Assignee and Approval Participants

After a version is created, you cannot modify the life cycle definition for that version. This means that you cannot create new milestones and tasks or remove existing milestones or tasks. You cannot add or remove users or groups from the **Participants** list. However, the value of the task fields **To Be Completed By** and **To Be Approved By** can be changed to specify another user or group that is listed in the selection list for those task fields. These fields can be modified only by a SAS Model Manager administrator or by the current user that is assigned to complete or approve the task. If a group is specified, then any member of the group can modify fields.

A best practice is to assign the value of **To Be Completed By** and **To Be Approved By** to a group instead of to a user. If there is a chance that those responsibilities could be assigned to other users, you can make changes if you assign a group to those responsibilities instead of assigning an individual user. Specifying a group for the assignee and approval responsibilities is preferred because of the flexibility you then have to add or remove users in a group.

When you specify an individual user, only that user has the authorization to update the task **Status** field. If you specify a group, any member of the group can update the task **Status** field. The user ID of the group member who changed the status appears in the **Completed By** or **Approved By** fields.

Users can be added to or deleted from a group using SAS Management Console and no changes are needed in the life cycle template if a group is specified as an **Assignee** or **Approver**. For example, if a user leaves your organization and that user was the only assignee, then that user's SAS Model Manager user ID cannot be deleted from the system until the champion model is retired. If your organization hires a new analyst, you can add that analyst to a group that has the role of **Model Manager: Life Cycle Participant Usage** and **Model Manager: Life Cycle Assignee Usage**. That user can then complete a task and update the task **Status** field without having to create a new version and a life cycle template that includes that individual user.

When you assign a group to be an **Assignee** or an **Approver**, all users and groups in that group have the authority to change the task status. Therefore, ensure that the users and groups that are defined in the group are those users and groups that you intend to be an **Assignee** or **Approver**.

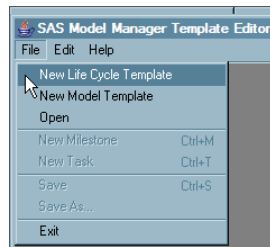
SAS Model Manager provides two groups, **Model Manager Example Life Cycle Assignee Users** and **Model Manager Example Life Cycle Approver Users**. Use these groups only as an example of how to configure a group in SAS Management Console for **Assignees** and **Approver** groups. Do not include them in your template.

Create a New Life Cycle Template

To create a new life cycle template, follow these steps:

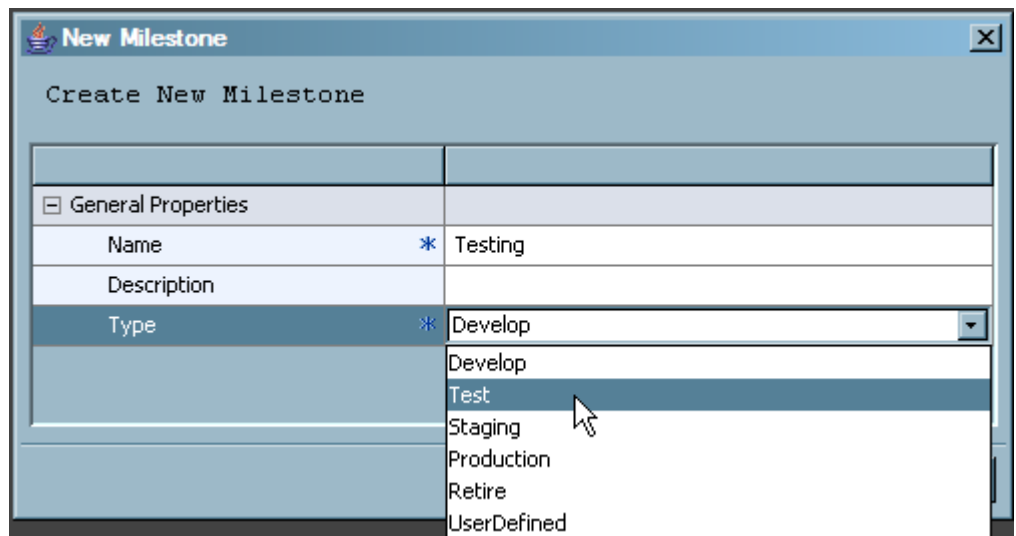
Note: You can view sample life cycle templates in the Life Cycle perspective.

1. From the SAS Model Manager Template Editor window, select **File** ⇒ **New Life Cycle Template**.



The Template Editor opens a template that has the name **Untitled Template**. Life cycle template, milestone, and task properties that display an asterisk (*) require a value for the property.

2. Name the template and save it. Type a name in the **Name** field and select **File** ⇒ **Save As**. In the Save dialog box, select the folder to save the template to. In the **File name** field, type the life cycle template name and click **Save**. This saves the template as an XML file.
3. Using a text editor, open the life cycle template XML file that you saved. Remove the individual participants who you do not want to appear in the **Participants** list. The participants are enclosed in <Participant> </Participant> tags. Be sure to remove the **mdlmgrexampleassignees** and **mdlmgrexampleapprovers** participants. If you do not remove these example groups, the **To Be Completed By** and the **To Be Approved By** version life cycle task properties cannot display only a list of participants.
Save the file.
4. In the SAS Model Manager Template Editor, select **File** ⇒ **Open**. In the Open dialog box, select the template and click **Open**.
5. Assign values to the life cycle properties **Description**, **Version**, and **Default**. For more information, see “[Template Properties](#)” on page 64.
6. Create milestones for the life cycle. For each new milestone, right-click the template name and select **New Milestone**. The New Milestone window opens.



Complete these fields:

- a. Enter a **Name** and a **Description** for the new milestone. The **Name** field is required.
- b. Click the **Type** field, select a milestone type, and then click **OK**. For more information, see “[Milestone Properties](#)” on page 64. The milestone is added to the template and assigned a **Display ID** value.

7. For each milestone, define the tasks for the milestone. Right-click the milestone and select **New Task**. The New Task window opens.

Create New Task	
<input type="checkbox"/> General Properties	
Name *	Determine reports needed
Description	
Type *	UserDefined

OK Cancel

Complete these fields:

- a. Enter a **Name** and **Description** for the task. The **Name** field is required.
 - b. Click on the **Type** field and select a task type. Click **OK**. For more information, see [“Task Properties” on page 65](#).
8. For each task, complete the task properties. For more information, see [“Task Properties” on page 65](#).
 9. To change the position of the milestone in the life cycle tree or to change the position of the task in the milestone tree, right-click the milestone name or the task name and select **Move Up** or **Move Down**. A task or milestone can be moved up or down only if, after the move is complete, no tasks are dependent on later tasks in the tree structure.
 10. To delete a milestone or a task, right-click the milestone or task and select **Delete**. When you delete a task, dependencies on that task are deleted.
- Note:* You can cut or copy milestones and tasks, and paste them as new milestones or tasks. To paste a milestone, right-click the template name and select **Paste**. To paste a task, right-click a milestone name and select **Paste**. When you cut or copy a milestone or task, dependencies on that task or milestone are deleted.
11. To save the template, select **File** ⇒ **Save As**. Select a directory and a filename for the template.
 12. Click **Save**.

Note: The template can now be copied to the middle-tier server by a user who has Write access to the SAS Model Manager user templates directory. The server administrator can then complete the deployment of the templates. For more information about the deployment of user templates, see the *SAS Model Manager: Administrator's Guide*.

Modify a Life Cycle Template

User life cycle templates are stored on the middle-tier server in the `\sasconfigdir\Lev#\AnalyticsPlatform\apps\ModelManager\ext` directory. If you have Write access to this directory, you can open and modify the templates. If you do not have Write access to this directory, copy the life cycle templates to a local directory. User life cycle templates are the only life cycle templates that can be modified by the SAS Model Manager Template Editor or saved to the user-templates directory on the middle-tier server.

To modify a life cycle template, follow these steps:

1. From the SAS Model Manager Template Editor window, select **File** ⇒ **Open**.
2. Select a life cycle template, and then click **Open**. The template properties appear on the right.
3. To modify life cycle properties, select the property and make changes to the property value. For more information, see [“Template Properties” on page 64](#).
4. Create or modify a milestone:
 - To create a new milestone, right-click the template name and select **New Milestone**. Complete the milestone properties.
 - To modify milestone properties, select the property and make changes to the property value.

For more information, see [“Milestone Properties” on page 64](#).

5. Create or modify a task:
 - To create a new task, right-click a milestone and select **New Task**. Complete the task properties. If a property has an ellipsis button (...), click the button to make changes to the property value.
 - To modify a task property, select the property and make changes to the property value. If a property has an ellipsis button (...), click the button to make changes to the property value.

For more information, see [“Task Properties” on page 65](#).

6. To delete a milestone or a task, right-click the milestone or task and select **Delete**. If you delete a task, dependencies on that task are deleted.
7. To save the template, do one of the following:
 - a. Select **File** ⇒ **Save** to save the changes to the existing template.
 - b. Select **File** ⇒ **Save As**, and then select a directory and filename for the template.

The template can now be copied to the middle-tier server by a user who has Write access to the SAS Model Manager user templates directory. The server administrator can then complete the deployment of the templates. For more information about the deployment of user templates, see the *SAS Model Manager: Administrator's Guide*.

Create a New Version of a Life Cycle Template

You can create a new version of a user life cycle template without changing the template name. To do this, you save a copy of the template using a new filename. You then modify the original file and archive the file with the new name. For example, if your life cycle template is named LChmeq.xml, you could save a copy of the file and name it LChmeqArchive.xml. You would then modify the LChmeq.xml template.

After a template is saved to the user-templates directory, existing versions that use this template are not updated. SAS Model Manager checks for new versions each time it starts. If a new life cycle version is detected, SAS Model Manager uses the updated life cycle template for new versions that specify that template.

User life cycle templates are stored on the middle-tier server in the `\sasconfigdir\Lev#\AnalyticsPlatform\apps\ModelManager\ext` directory. If you have Write access to this directory, you can open and modify the templates in this directory. If you do not have Write access to this directory, copy the life cycle templates to a local directory.

Note: User life cycle templates are the only life cycle templates that can be modified by SAS Model Manager Template Editor.

To create a new version of a life cycle template, follow these steps:

1. Open the template.
2. To save the template, select **File** ⇒ **Save As**. In the Save dialog box, select a location on your computer. Enter a filename. Click **Save**.

Note: The filename can be the same or different from the template file that you are modifying.

3. In the Template Editor, select **File** ⇒ **Open**. In the Open dialog box, select the location and the new template file. Click **Open**.
4. Select the life cycle name on the left pane. Click the **Version** property and enter a new version number.

Note: A best practice is to increment the template version by 1.

5. Make any changes to the life cycle template and select **File** ⇒ **Save**.

Note: The template can now be copied to the user-templates directory by a user who has Write access to the directory. For more information, see the *SAS Model Manager: Administrator's Guide*.

Life Cycle Template Properties

Template Properties

Here is a list of the life cycle template properties.

Property Name	Description
Name	Identifies the name of the life cycle template. This property is required.
Description	Specifies user-defined information about the life cycle template.
Version	<p>Specifies a life cycle version number. A version number is an integer number. Each time that you create a new version of a template, increment the version number by 1. The version number for each life cycle template is unique to that template. This property is required.</p> <p>SAS Model Manager checks for new versions each time it starts. If a new life cycle version is detected, SAS Model Manager uses the updated life cycle template for new versions that specify that template. Any subsequent reference of the template uses the newest version of the template.</p>
Default	<p>Specifies whether the life cycle template is the default template that is used when you create a new version in a project. Only one life cycle template in the middle-tier server, user-template directory can be the default template. Valid values are TRUE and FALSE. In the template editor, check the box to set the value to TRUE.</p> <p>This property is required.</p>

Milestone Properties

Here is a list of the milestone properties for the life cycle template.

Property Name	Description
Name	Identifies the name of the milestone. This property is required.
Display ID	Displays a system-supplied milestone identifier that is an integer greater than 0. A milestone identifier is based on the order in which it appears in the life cycle definition. For example, the first milestone in the life cycle template has an identifier of 1. The second milestone has an identifier of 2.
Description	Specifies user-defined information about the milestone.
Milestone Phase or Type	<p>Specifies the phase for the milestone. The milestone phase is for information only. The value that you select does not affect life cycle processing by SAS Model Manager. Here is a list of valid milestone phases:</p> <p>Develop specifies that the milestone has development tasks such as registering models and ensuring that a version has all of the required resources for validating candidate models.</p> <p>Test specifies that the milestone has testing tasks such as validating a model's input and output variable data structure and creating reports to compare the scores of candidate models.</p> <p>Staging specifies that the milestone has staging tasks such as exporting a champion model to a SAS metadata repository, publishing a model to a channel, and publishing In-Database scoring functions to a database.</p> <p>Production specifies that the milestone has production tasks such as scoring a champion model in a production environment, and monitoring a champion model's performance.</p> <p>Retire specifies that the milestone has retirement tasks such as removing a model from a production environment.</p> <p>UserDefined specifies a custom milestone for your organization, such as indicating that a champion model is in compliance with government regulations or industry process standards.</p>

Task Properties

Here is a list of the task properties for the life cycle template.

Property Name	Description
Name	Identifies the name of the task. This property is required.

Property Name	Description
Display ID	Displays a system-supplied task identifier in the form milestone#.task# (for example 1.1) that identifies the milestone that the task is a part of as well as the task. Each milestone and task identifier is based on the order in which it appears in the life cycle definition. For example, the first milestone in the life cycle template has an identifier of 1. The second milestone has an identifier of 2. The identifier for the first task in milestone 1 is 1.1. The second task in milestone 1 has an identifier of 1.2.
Dependencies	Identifies a DisplayID for a task that must be completed before this task can be completed.
Description	Displays user-defined information about the task.
Task Type	<p>Specifies a type for the task. Here is a list of valid task types:</p> <p>UserDefined identifies the task as a custom task for your organization. A user-defined task represents a step in your organization's model life cycle that you would like to track using SAS Model Manager. SAS Model Manager does not perform any tests or verify that any project or version tasks have been performed for any user-defined tasks.</p> <p>Signoff specifies that all of the milestone tasks are complete and have been approved.</p> <p>DeclareProduction specifies that the champion model is ready to be exported to the production environment.</p> <p>SetChampion specifies that the task is to determine a champion model. Before this task can be marked complete, a champion model must be set for the containing version.</p> <p>RetireChampion specifies that the champion model is retired.</p>
Assignees	<p>Specifies a user or group name from the Participants list. The specified user or any member of the specified group is assigned to complete the task. The specified user or group members are the only users who are authorized to set the task Status field to Not Started, Started, or Completed.</p> <p>Assignees can be unassigned. If this field is unassigned, the following rules apply:</p> <ul style="list-style-type: none"> • Updates to the task status are not required. • Only those users and groups that are in the life cycle Participants list can modify the task status.

Property Name	Description
Approvers	<p>Specifies a user or a group from the Participants list. The specified user or any member of the specified group is assigned to complete the task. The specified user or any member of the group is authorized to set the task Status field to Approved.</p> <p>Approvers can be unassigned. If this field is unassigned, the following rules apply:</p> <ul style="list-style-type: none"> The task approval status is not required to be updated. Only those users and groups that are on the life cycle's Participants list can modify the task approval status.
Weight	<p>Specifies a percentage either as an integer or a fractional number that indicates the relative work effort that is required by the task to complete the milestone. SAS Model Manager uses weight values to calculate the percentage that is complete for a milestone. The weight appears as a property for a version's Life Cycle folder. If you use the Weight property, the weight values for all tasks in a milestone should add up to 100. When weights for a milestone do not add up to 100, SAS Model Manager performs a weight proportion adjustment so that the sum of those weights within a milestone adds up to 100.</p> <p>Note: user-defined weights are not explicitly adjusted. Weights remain as they were entered and are not adjusted.</p>
Duration	<p>Specifies a number that indicates the amount of time that is allocated to complete the task. The default duration unit is the number of days.</p>

Create a Version

To create a new version, follow these steps:

1. Right-click the project folder in the Project Tree and select **New** ⇒ **New Version** from the pop-up menu. The New Version window opens.

New Version	
Enter a version name and description.	
<input type="checkbox"/> General Properties	
Name *	2010
Description	
<input type="checkbox"/> Version Properties	
Life Cycle Template	User Lifecycle Template
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

2. Specify a name and an optional description for the new version. Use a name that is unique among versions in this project.
3. Select the life cycle template that you will use to monitor the milestone phases and tasks.

Note: SAS Model Manager administrators and advanced users can use the SAS Model Manager Template Editor to customize life cycle templates. Use the Life Cycle Templates perspective to view the contents of a template.

4. Review the selections and click **OK**.
5. Examine the properties of the version folder. The value for **Creation Date** is today's date. The value for **State** is **Under Development**.

Note: SAS Model Manager automatically annotates the version's history and notes.

See Also

- [“Overview of Versions” on page 53](#)
- [“Creating Life Cycle Templates” on page 56](#)

Version Properties

About Version Properties

Version properties are metadata that describe the version attributes. Version metadata includes information such as the name of the version, the owner, unique identifiers, the name and path of the SAS Model Manager repository, and the tables that SAS Model Manager uses to score the models.

Version properties are organized into several types:

- **General Properties**
- **System Properties**
- **Specific Properties**
- **User-Defined Properties**

You cannot modify the **General Properties** or **System Properties** except to specify a description for the folder. The **Specific Properties** contain information about tables that the version uses and properties to monitor the life cycle of the version. You use **User-Defined Properties** to add your own version properties. The background color of a property determines whether you can modify a property value. You modify only the fields that are white. When you click in a field, you either enter a value or select a value from the list box.

Version-Specific Properties

Here is a list of the specific properties for a version.

Property Name	Description
Default Scoring Task Input Table	Specifies a default SAS data set that is used as the input data table for all scoring tasks within the SAS Model Manager version. If you specify a value for the Default Scoring Task Input Table property, the value is used as the default input table in the New Scoring Task window if a default scoring task input table is not specified for the model.
Default Scoring Task Output Table	Specifies a default SAS data set that defines the variables to keep in the scoring results table of the scoring task. If you specify a value for the Default Scoring Task Output Table property, the value is used as the default output table in the New Scoring Task window if a default scoring task output table is not specified for the model.
Default Performance Table	<p>Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager version.</p> <p>The value of the Default Performance Table property is used as the default value for the Performance data source field in the Define Performance Task wizard if a default performance table is not specified for the model.</p>
Default Train Table	<p>The train table is optional and is used only as information. However, when a value is specified for a model's Default Train Table property, SAS Model Manager does the following:</p> <ul style="list-style-type: none"> • uses default train table to validate Teradata scoring functions when a user publishes the associated project champion model to a Teradata database • checks the Validate scoring results box in the Publish Teradata Scoring Function window. <p>The value of the Default Train Table property is used to validate Teradata scoring functions only if a default train table is not specified for the model.</p>
State	Specifies the current status of the version.
Champion Model ID	Specifies the UUID for the champion model in the version, if one exists.
Frozen Date	Specifies the date that the version was frozen.

Property Name	Description
Production Date	Specifies the date that the status of the Production milestone task in the version's life cycle was changed from Started to Complete .
Retired Date	Specifies the date that the status of the Retire milestone task in the version's life cycle was changed from Started to Complete .

Working with Life Cycles

Overview of a Life Cycle

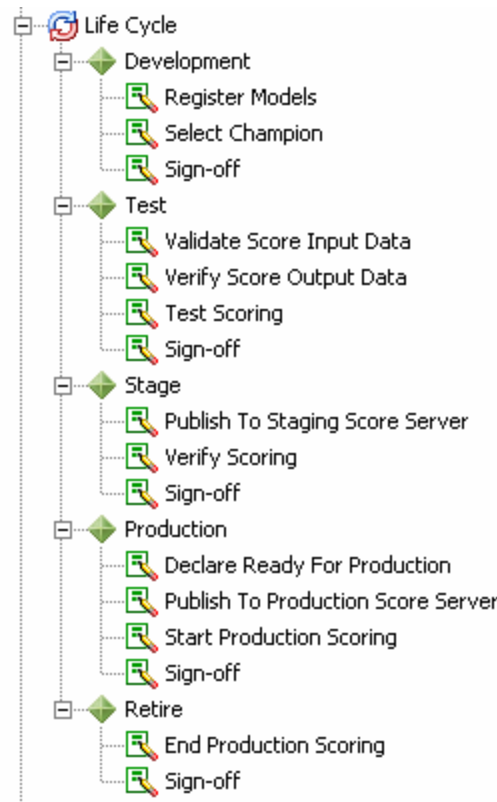
What Is a Life Cycle?

A SAS Model Manager life cycle defines the milestones and tasks that your organization uses to monitor the progress of a modeling project. The life cycle template controls the milestone tasks and the sequence of activities that are required to implement and deploy scoring models. SAS Model Manager provides example life cycle templates that you can use to create your own life cycle templates that are based on your business requirements.

The milestones in life cycle template track the progress of developing, implementing, and retiring your scoring models. Authorized users indicate when milestone tasks are started, completed, or approved. The properties of a life cycle template determine who is authorized to update the status of a milestone task. Precedence rules for successive milestones ensure that life cycle tasks are completed in the correct order. SAS Model Manager automatically records the dates, times, and users that are associated with individual life cycle milestones.

How Life Cycle Milestones Are Organized

The **Life Cycle** node contains the milestone phases and tasks for a modeling project. SAS Model Manager applies life cycle milestones to each version. Typical life cycle milestones for a modeling process might include the following:



The life cycle for a version always starts with the first milestone. A milestone is completed after all of its component tasks are completed. Milestones are normally completed sequentially, but the ordering sequence is defined at the task level. A task might be configured to depend on one or more other tasks. If a task has dependent tasks, then you cannot change the status for a task to **Completed** until all dependent tasks are also completed. Task dependencies control the milestone sequences.

Note: You can start another task when it depends on the preceding task even if that the preceding task is not yet completed.

Life Cycle Tasks

About Life Cycle Tasks

Life cycle templates define the milestones that you use to track the modeling and deployment processes for a project version. Each milestone consists of one or more life cycle tasks. Milestones for the simple life cycle template might include **Development**, **Test**, **Production**, and **Retire**. The milestone tasks for this template describe the sequential steps to develop, assess, deploy, and retire scoring models that are based on time requirements and model performance. As a practical minimum, the life cycle template that you use should include at least two milestones: Development and Production.

You select the life cycle template when you create a version. Authorized users update the life cycle to indicate whether milestone tasks have been started, completed, or approved. The configuration of a life cycle template determines who can update the status of a life cycle milestone. Precedence rules among successive milestones ensure that milestone tasks are not completed out of sequence. SAS Model Manager documents the dates, times, and individuals who are associated with individual life cycle tasks and milestone status changes.


These are some tasks that you might perform to monitor the life cycle of a model:

- View life cycle templates in the Life Cycle perspective. For more information, see [“View Life Cycle Templates” on page 72](#).
- Update the status for a milestone. For more information, see [“Update Milestone Status” on page 73](#).
- Search for a task. For more information, see [“Search Life Cycles for Tasks Assigned to Users” on page 239](#).

For information about users and groups who can update the **Status** property of a task, see [“Participant Roles” on page 58](#).

View Life Cycle Templates

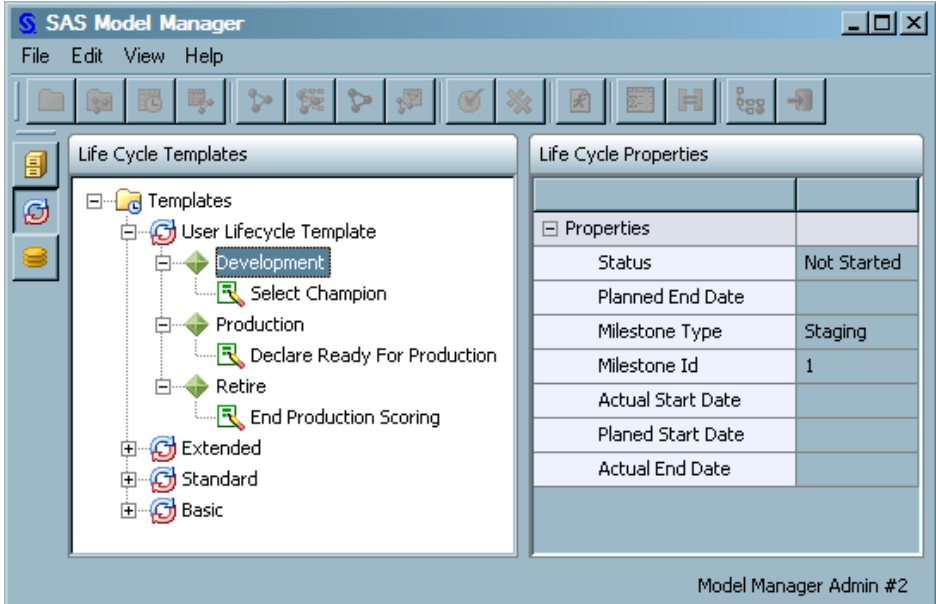
To view the life cycle templates that are available in SAS Model Manager, follow these steps:

1. Click the Life Cycle perspective button  in the SAS Model Manager window.
2. Expand the **Templates** folder to displays the available life cycle templates. The **Templates** folder contains your custom the life cycle templates and the SAS Model Manager example life cycle templates. The following example templates are provided by SAS Model Manger:

- **Basic and User Lifecycle Template**
- **Standard**
- **Extended**

These life cycle templates are example templates that you can use as a model to create life cycle templates that meet your organizations needs. Life cycle templates other than the SAS Model Manager example templates are customized templates that have been specially created by SAS Model Manager administrators and advanced users.

3. Expand the life cycle template node to explore the structure of the milestones. Examine the milestone requirements in the **Life Cycle Properties** pane. The approximate sequential ordering of the milestone phases is determined by the **MilestoneId** property. At the task level, sequential ordering is determined by the **ActionId** property.



The screenshot shows the SAS Model Manager application window. The **Life Cycle Templates** pane on the left displays a tree structure under the **Templates** folder. The **User Lifecycle Template** is expanded, showing a sequence of milestones: **Development** (selected), **Select Champion**, **Production**, **Declare Ready For Production**, **Retire**, and **End Production Scoring**. Below these are three other templates: **Extended**, **Standard**, and **Basic**.

The **Life Cycle Properties** pane on the right displays a table of properties for the selected **Development** milestone.

Life Cycle Properties	
Properties	
Status	Not Started
Planned End Date	
Milestone Type	Staging
Milestone Id	1
Actual Start Date	
Planned Start Date	
Actual End Date	

Model Manager Admin #2

4. Expand the milestone phase to view the milestone tasks. Examine the task requirements in the **Life Cycle Properties** pane.
 - **ActionId** determines the order for completing milestones and tasks.
 - **Dependencies** determines whether the task depends on one or more other tasks. If a task has dependent tasks, then you cannot change the status for a task to **Completed** until all dependent tasks are also completed. Task dependencies control milestone sequences
 - **Weight** specifies the percentage of work effort the task is assigned to complete the milestone. The sum of the weighted values for a milestone does not have to equal 100.

For information about users and groups who can update the Status property of a task, see [“Participant Roles” on page 58](#).

Update Milestone Status

To modify the status for a life cycle milestone, follow these steps:

1. Right-click the version **Life Cycle** node in the Project Tree and select **Expand All Items**.
2. Select the task that you want to update under the milestone phase. For example, if your template is modeled after the Standard or Extended life cycle template, then you can monitor the status of registering models under the **Development** milestone.

Note: Milestones are normally completed sequentially, but the ordering sequence is defined at the task level. If a task has dependent tasks, then you cannot change the status for a task to **Completed** until all dependent tasks are also completed. Task dependencies control the milestone sequence. Date requirements are benchmarks for the start and completion of life cycle milestone and tasks.

3. On the **Properties** tab, select a value for **Status** that indicates the progress of completing this milestone. Possible values are **Not Started**, **Started**, **Completed**, or **Approved**.

Note: You must be authorized to set properties for a milestone. Task properties in the life cycle template determine which users or user groups are responsible for completing and approving a milestone task.

4. Select the **Life Cycle** node to examine its properties. The value for **Modification Date** is today's date. Under **Life Cycle Properties**, the bar charts display the percentage of completed tasks for each milestone.

Life Cycle Properties

About Life Cycle Properties

Life cycle properties are metadata that describe the life cycle milestones and user roles. Life cycle metadata includes information such as the name of the milestone phase or task, the owner, unique identifiers, the name and path of the SAS Model Manager repository, and the status of milestones.

Milestone and task properties are organized into several types:

- [General Properties](#)
- [System Properties](#)
- [Specific Properties](#)

- **User-Defined Properties**

You cannot modify the **General Properties** or **System Properties** except to specify a description for the folder. The milestone **Specific Properties** contain information about start and end dates. The task **Specific Properties** contain information about status, dates, and process participants. You use **User-Defined Properties** to add your own life cycle properties. The background color of a property signifies whether you can modify a property value. You modify only the fields that are white. When you click in a field, you either enter a value or select a value from the list box.

Specific Properties for Milestones and Tasks

Here is a list of the milestone properties.

Property Name	Description
Actual Start Date	Specifies the actual date that the first task for the milestone is started. This property is Read-only.
Actual End Date	Specifies the actual date when all tasks for the milestone are finished. This property is read-only. SAS Model Manager assigns the value when the status of every milestone task is set Completed .
Planned Start Date	Specifies the expected date to start the first task for milestone.
Planned End Date	Specifies the expected date to complete all tasks for the milestone.

Here is a list of task properties:

Property Name	Description
Status	Specifies the status of task. Possible values are Not Started , Started , Completed , or Approved .
Completed Date	Specifies the date when the task is finished. This property is read-only. SAS Model Manager assigns the value when the status of the milestone task was changed to Completed .
Completed By	Specifies the name of the user who completed the task. This property is Read-only.
Approved Date	Specifies the date when completion of the task is approved. This property is Read-only.
Approved By	Specifies the name of the user who approved completion of the task. This property is Read-only.

Property Name	Description
Planned Completion Date	Specifies the expected date to complete the task.
To Be Completed By	Specifies the user who is responsible for completing the task.
To Be Approved By	Specifies the user who can approve that the task is completed.

Part 3

Importing, Scoring, and Validating Models

<i>Chapter 7</i>	
Importing Models	79
<i>Chapter 8</i>	
Scoring Models	107
<i>Chapter 9</i>	
Validating Models Using Comparison Reports	125
<i>Chapter 10</i>	
Validating Models Using User Reports	135

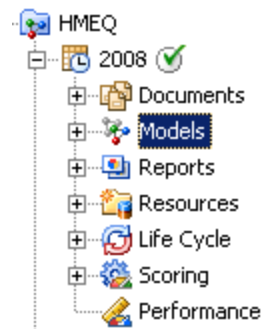
Chapter 7

Importing Models

Overview of Importing Models	79
Import Models from Metadata Repository	80
Import Package Files from SAS Enterprise Miner	81
What Is a SAS Enterprise Miner Package File?	81
Create Package Files	81
Import Package Files	82
Import SAS Code Models	83
Overview of Importing SAS Code Models	83
Model Templates	84
Model Template Component Files	86
Importing a SAS Code Model	91
Import PMML Models	92
Import Partial Models	93
Set Model Properties	94
Map Model Variables to Project Variables	95
User-Defined Model Templates	96
Creating a New Model Template	96
Model Template Properties	101
Specific Properties for a Model	103

Overview of Importing Models

After you create a project and version, the next step is to import models into SAS Model Manager. Your version has a folder hierarchy that contains all of a version's components. The version components include folders for the version Life Cycle, Documents, Models, Reports, Resources, Scoring, and Performance files.



The **Models** folder is the container for all of the models in your project version. After model evaluation, one of the candidate models will become the champion model. However, the first step is to import the candidate models into your project's version **Models** folder.

SAS Model Manager provides many methods of importing your SAS models into your project version:

Note: Scorecard models can be imported using the SAS Code Models and SAS Enterprise Miner Package File import methods.

- “Import Models from Metadata Repository” on page 80
- “Import Package Files from SAS Enterprise Miner” on page 81
- “Import SAS Code Models” on page 83
- “Import Partial Models” on page 93
- “Import PMML Models” on page 92

SAS Model Manager also provides SAS macros so that you can use SAS code to import or register SAS models into your project version. For more information, see “[SAS Model Manager Access Macros](#)” on page 243.

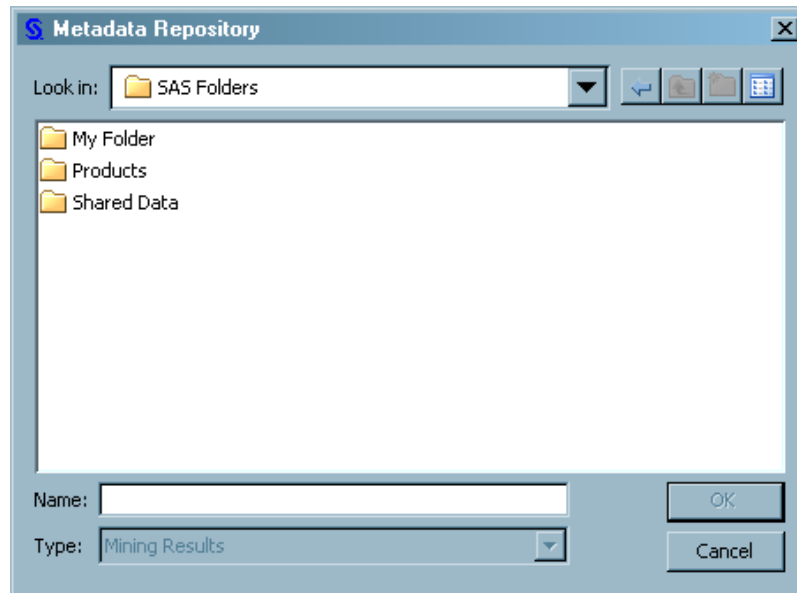
Import Models from Metadata Repository

If your SAS Enterprise Miner 5.1 (or higher) model files are registered in your SAS Metadata Repository, you can import them into SAS Model Manager from the repository.

Note: Before you import a model into your project's version, verify that the model type matches the **Model Function** property setting on the Project Properties panel. For more information about model functions, see “[Specific Properties for a Project](#)” on page 285.

To import a SAS Enterprise Miner model from a SAS Metadata Repository, follow these steps:

1. In the Project Tree, navigate to the project's version.
MMRoot ⇒ *<organizational folder>* ⇒ *<project folder>* ⇒ *<version folder>*
2. Right-click the **Models** folder and select **Import From** ⇒ **Metadata Repository**. The Metadata Repository window is displayed.



3. Navigate to the location of the folder that contains the SAS Enterprise Miner model.
4. Select a model from the folder.
Note: You can import only one model at a time in the Metadata Repository window.
5. Click **OK**. After the SAS Model Manager processes the model import request, the new model appears in the **Models** folder of your project's version.

You can select the model in the tree, then view the tabs in the properties panel on the right. You can view your newly imported model's data structures in the **Model Input**, **Model Output**, and **SAS Code** tabs.

See Also

- [“Import Package Files from SAS Enterprise Miner” on page 81](#)
- [“Import Partial Models” on page 93](#)
- [“Set Model Properties” on page 94](#)
- [“Map Model Variables to Project Variables” on page 95](#)

Import Package Files from SAS Enterprise Miner

What Is a SAS Enterprise Miner Package File?

SAS Enterprise Miner package files, or SPK files, contain complete model information. A user can import SAS Enterprise Miner models even if they are not registered in a SAS Metadata Repository.

Create Package Files

To create an SPK file for an existing SAS Enterprise Miner model, follow these steps:

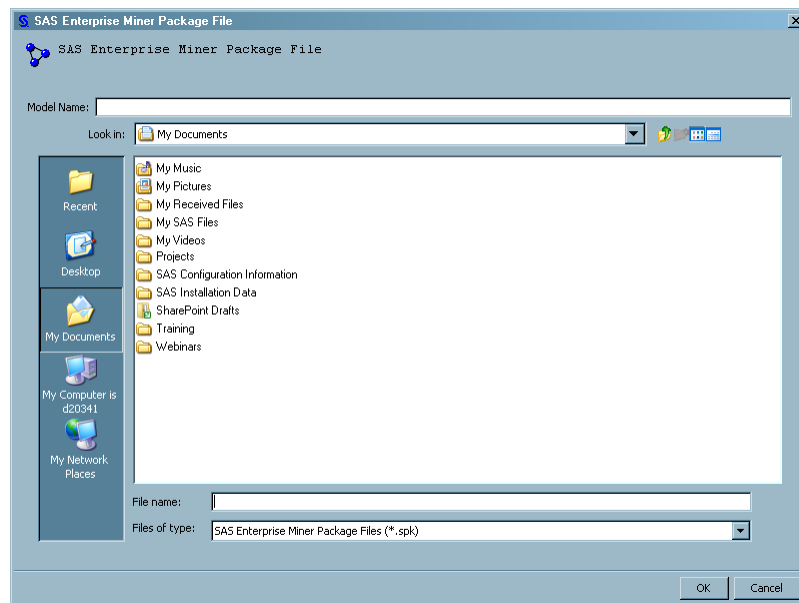
1. Open the SAS Enterprise Miner diagram that contains the model, and then run the model.
2. After the model run is complete, you can right-click the model in the SAS Enterprise Miner Diagram Workspace, and select **Create Model Package**. The new SPK filename appears under the Model Packages folder in your SAS Enterprise Miner Project Navigator.
3. Right-click the filename and select **Save As** to copy the SPK file from the SAS Enterprise Miner server to your computer.
4. Specify a destination folder on your computer, such as, **C:\MMDa**, and save the file to your workstation folder.

Import Package Files

Note: Before you import a model into your project's version, verify that the model type matches the **Model Function** property setting on the Project Properties panel. For more information about model functions, see [“Specific Properties for a Project” on page 285](#).

To import package files into SAS Model Manager, follow these steps.

1. In the Project Tree, navigate to the project's version.
MMRoot ⇒ **<organizational folder>** ⇒ **<project folder>** ⇒ **<version folder>**
2. Right-click the **Models** folder and select **Import From** ⇒ **SAS Enterprise Miner Package File**.



3. Enter a text value in the **Model Name** field. The value of the **Model Name** field appears as the model name in the Project Tree.
4. Navigate to the location of the SAS Enterprise Miner Package file (SPK) and select the file.
5. Click **OK**. After the SAS Model Manager processes the model import request, the new model appears in the **Models** folder of your project's version.

6. Repeat steps 2 through 5 to import additional model package files from your client workstation folder.

See Also

- [“Import Partial Models” on page 93](#)
- [“Set Model Properties” on page 94](#)
- [“Map Model Variables to Project Variables” on page 95](#)

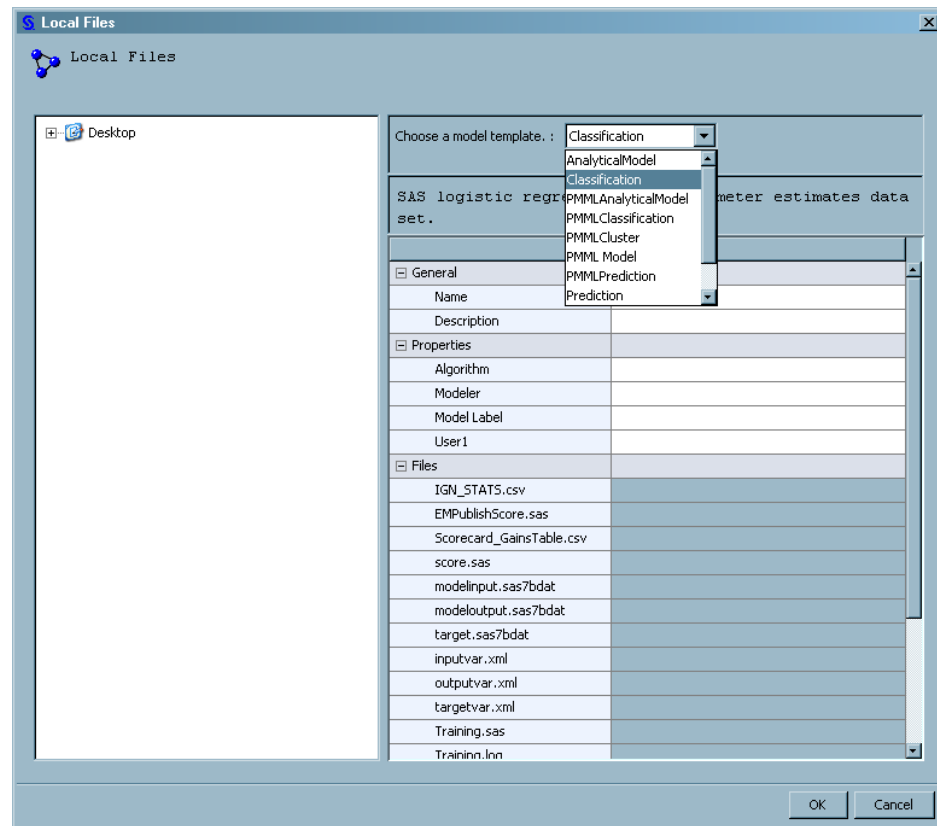
Import SAS Code Models

Overview of Importing SAS Code Models

You use the Local Files method to import models that you create using SAS code, but that were not created in or exported from SAS Enterprise Miner. An example of such a model might be a SAS LOGISTIC procedure model.

To use the Local Files method, you must prepare model component files. Model component files provide the metadata that is used to process a model in SAS Model Manager. The model component files that you prepare are dependent upon the project's model function. You can find the model function in the project property **Model Function**. The SAS Model Manager model functions are analytical, classification, prediction, and segmentation. For a list of component files by model function, see [“Model Template Component Files” on page 86](#). If you do not have all of the component files when you import the model, you can create them and add them later using the SAS Model Manager Partial Import utility. For more information, see [“Import Partial Models” on page 93](#).

After you have your model component files, you use the Local Files window to import the component files. In the Local Files window, you select a model template and assign values to the template model properties and model files. The following display shows a partial list of model templates that you can select in the Local Files window as well as the properties and files for the Classification model template:

Display 7.1 The Local Files Window

After you select your model template you complete the property values in the **General** and **Properties** section, as well as enter your component filenames in the **Files** section.

SAS code models, at a minimum, require a score code component file, and other component files to define the model input and output variables in SAS tables. Prediction and classification models also require a component file to define target variables.

Model Templates

What Is a Model Template?

Models that you import into SAS Model Manager are associated with a specific model template. A model template defines the properties and the component files that define a type of model. SAS Model Manager processes four types of models: analytical, classification, prediction, and segmentation. You can create your own model template if your model requires files other than those named in the SAS Model Manager templates.

A model template is an XML file that has three sections. The **General** section names and describes the model template. The **Properties** section provides properties to name the model algorithm, the modeler, and a model label. The **Files** section contains the component files that can be used in the template for that model function type.

You associate your component file with the appropriate model template component file. You do this by adding your component filename as a value for the model template filename. For example, most templates have a `modelinput.sas7bdat` component file. In the Local Files window, you enter the name of your SAS data set that defines the model input variables in the value field for **modelinput.sas7bdat**. Here is the model template component filename and value for the `modelinput.sas7bdat` component file from the template:

modelinput.sas7bdat

myModelInput.sas7bdat

Your component file filenames do not need to be the same name as the filenames in the model template.

For information about component files for the different model types, see [“Model Template Component Files” on page 86](#).

SAS Model Manager Model Templates

SAS Model Manager provides model templates for analytical, classification, prediction, and segmentation models.

Model Type	Description
Analytical	The Analytical model template is the most generic SAS Model Manager template that is designed for models whose model function does not fall in the prediction, classification, and segmentation category.
Classification	You use the Classification model template if your model is a prediction model that has a categorical, ordinal, or binary target, or if your model is a LOGISTIC procedure regression model. Examples of classification models are models that might classify a loan applicant as Approved or Not Approved, or models that might assess a potential customer's risk of default as Low, Medium, or High.
Prediction	The Prediction model template is used for predictive models. Predictive models declare in advance the outcome of an interval target. A model that assigns a numeric credit score to an applicant is an example of a prediction model.
Segmentation	The Segmentation model template is used for segmentation or cluster models that are written in SAS code. Segmentation models are unsupervised models that have no target variable. The object of a segmentation or cluster model is to identify and form segments, or clusters, of individuals or observations that share some affinity for an attribute of interest. The output from a segmentation model is a set of cluster IDs.

If you do not have the required component files that are in the model template, you can add them later, using the SAS Model Manager feature, [“Import Partial Models” on page 93](#).

User Model Templates

Model templates provide users with a way to define metadata about their own model. Most users do not need to write model templates because SAS Model Manager delivers a list of model templates that handle Enterprise Miner models as well as analytical, prediction, classification, and segmentation models. Users can write their own model templates if the model templates that are provided by SAS Model Manager do not satisfy their requirements.

Users can create model templates by using the SAS Model Manager Template Editor. For more information, see [“Creating a New Model Template” on page 96](#).

You can view the User model template files (as well as all of the other SAS Model Manager template files) in your SAS Model Manager installation.

User model template file location:

```
\\sasconfigdir\Lev#\AnalyticsPlatform\apps\ModelManager\ext
```

SAS Model Manager standard template files location:

```
\sasinstalldir\SASModelManagerApplicationProgrammingInterface  
\2.2\Static\conf\modelTemplates
```

In the example locations, *sasconfigdir* is a wildcard placeholder for the path to your configured SAS directory. For Windows, the default value for *sasconfigdir* is \SAS\SAS Configuration\Config, and Lev# is a value that is selected by a user in the range of Lev0 through Lev9. The default value is Lev1. For UNIX systems the default home folder for the SAS Model Manager software is */usr/local/SAS*

sasinstalldir is a wildcard placeholder for the directory where SAS Model Manager is installed. For Windows, the default value for *sasinstalldir* is \SAS. For UNIX, the default home folder for SAS Model Manager is */usr/local/SAS*.

Model Template Component Files

Here is a list of the component files that are associated with the SAS Model Manager model templates:

Filename	Analytical	Classification	Prediction	Segmentation
IGN_STATS.csv	—	✓	—	—
EMPublishScore.sas	—	✓	—	—
Scorecard_GainsTable.csv	—	✓	—	—
score.sas	✓	✓	✓	✓
modelinput.sas7bdat	✓	✓	✓	✓
modeloutput.sas7bdat	✓	✓	✓	✓
target.sas7bdat	—	✓	✓	—
inputvar.xml	✓	✓	✓	✓
outputvar.xml	✓	✓	✓	✓
targetvar.xml	—	✓	✓	—
training.sas	✓	✓	✓	✓

Filename	Analytical	Classification	Prediction	Segmentation
trainin.log	✓	✓	✓	✓
training.lst	✓	✓	✓	✓
outest.sas7bdat	✓	✓	✓	—
outmodel.sas7bdat	✓	✓	✓	—
output.spk	✓	✓	✓	✓
format.sas7bcat	✓	✓	✓	✓
dataprep.sas	✓	✓	✓	✓

IGN_STATS.csv

The value of IGN_STAT.csv is the name of a file whose values are separated by commas, and whose values are bin definitions for input variables. This is a component file that is generated by SAS Enterprise Miner for a scorecard model and is not needed for SAS code models.

EMPublishScore.sas

The value of EMPublishScore.sas is the name of a SAS code file that is used to change input variables into bins and is a component of a SAS Enterprise Miner scorecard model. This file is needed to define a performance task. This file is not needed for SAS code models.

Scorecard_GainsTable.csv

This file includes the bin score definitions and is not used in reporting by SAS Model Manager. The file's content can be viewed by users.

score.sas

The value of score.sas is the name of a filename for the SAS score code for the model.

modelinput.sas7bdat

The value of modelinput.sas7bdat is the name of a sample data set that is used to create an inputvar.xml file for the model if one does not exist. When no inputvar.xml file exists for the model, SAS Model Manager creates the inputvar.xml file using the variable name and attributes in the modelinput.sas7bdat file. Observation values are not used. Therefore, the sample data set can have no observations or it can have any number of observations. If an inputvar.xml is specified in the model template, modelinput.sas7bdat is ignored.

When you import a SAS code model, the data set that you used to test your score code can be used as the value for the modelinput.sas7bdat file.

modeloutput.sas7bdat

The value of modeloutput.sas7bdat is the name of a sample data set that is used to create an outputvar.xml file for the model if one does not exist. When no outputvar.xml file exists for the model, SAS Model Manager creates the outputvar.xml file using the variable name and attributes in the modeloutput.sas7bdat file. Observation values are

not used. Therefore, the sample data set can have no observations or it can have any number of observations. If an `outputvar.xml` is specified in the model template, `modeloutput.sas7bdat` is ignored.

You can create a `modeloutput.sas7bdat` file by running the `score.sas` file against the `modelinput.sas7bdat` file.

target.sas7bdat

The value of `target.sas7bdat` is the name of a sample data set that is used to create a `targetvar.xml` file for the model if one does not exist. When no `targetvar.xml` file exists for the model, SAS Model Manager creates the `targetvar.xml` file using the variable name and attributes in the `target.sas7bdat` file. Data set values are not used. Therefore, the sample data set can have no observations or it can have any number of observations. If a `targetvar.xml` file is specified in the model template, `target.sas7bdat` is ignored.

You can create a `target.sas7bdat` file by creating a data set that keeps only the target variables that are taken from the training data set, as in this example:

```
data mydir.target;
    set mydir.myModelTraining (obs=1)
    keep P_BAD;
run;
```

inputvar.xml

The value of `inputvar.xml` is the name of an XML file that defines the model input variables. When your model template includes a file for `modelinput.sas7bdat`, SAS Model Manager creates the model `inputvar.xml` file. Otherwise, you must create the XML file.

The following XML file is a sample `inputvar.xml` file that has one variable, `CLAGE`. You can use this model to create an `inputvar.xml` file that contains a `VARIABLE` element for each model input variable.

```
<?xml version="1.0" encoding="utf-8"?>
<TABLE>
  <VARIABLE>
    <NAME>CLAGE</NAME>
    <TYPE>N</TYPE>
    <LENGTH>8</LENGTH>
    <LABEL Missing=""/>
    <FORMAT Missing=""/>
    <LEVEL>INTERVAL</LEVEL>
    <ROLE>INPUT</ROLE>
  </VARIABLE>
</TABLE>
```

NAME

specifies the variable name.

TYPE

specifies the variable type. Valid values are N for numeric variables and C for character variables

LENGTH

specifies the length of the variable.

LABEL Missing=""

specifies the character to use for missing values. The default character is a blank space.

FORMAT Missing=""

specifies a SAS format to format the variable.

LEVEL

specify either NOMINAL, ORDINAL, INTERVAL, or BINARY

ROLE

specify INPUT for input variables.

outputvar.xml

The value of outputvar.xml is the name of an XML file that defines the model output variables. When your model template includes a file for modeloutput.sas7bdat, SAS Model Manager creates the model outputvar.xml file. Otherwise, you must create the XML file.

The following XML file is a sample outputvar.xml file that has one variable, I_BAD. You can use this model to create an outputvar.xml file that contains a VARIABLE element for each model output variable.

```
<?xml version="1.0" encoding="utf-8"?>
<TABLE>
  <VARIABLE>
    <NAME>I_BAD</NAME>
    <TYPE>C</TYPE>
    <LENGTH>12</LENGTH>
    <LABEL>Into: BAD</LABEL>
    <FORMAT Missing=" ">
    <LEVEL>NOMINAL</LEVEL>
    <ROLE>CLASSIFICATION</ROLE>
  </VARIABLE>
</TABLE>
```

NAME

specifies the variable name.

TYPE

specifies the variable type. Valid values are N for numeric variables and C for character variables

LENGTH

specifies the length of the variable.

LABEL Missing=""

specifies a label for the output variable.

FORMAT Missing=""

specifies a SAS format to format the variable.

LEVEL

specify either NOMINAL, ORDINAL, INTERVAL, or BINARY

ROLE

specify the type of model output. Valid values are CLASSIFICATION, PREDICT, SEGMENT, and ASSESS

targetvar.xml

The value of targetvar.xml is the name of an XML file that defines the model target variables. When your model template includes a file for target.sas7bdat, SAS Model Manager creates the targetvar.xml file. Otherwise, you must create the XML file.

The following XML file is a sample targetvar.xml file that has one variable, I_BAD. You can use this model to create an outputvar.xml file that contains a VARIABLE element for each model output variable.

```
<?xml version="1.0" encoding="utf-8"?>
<TABLE>
```

```

<VARIABLE>
    <NAME>BAD</NAME>
    <TYPE>N</TYPE>
    <LENGTH>8</LENGTH>
    <LABEL>Missing="" />
    <FORMAT Missing="" />
    <LEVEL>BINARY</LEVEL>
    <ROLE>TARGET</ROLE>
</VARIABLE>
</TABLE>

```

NAME

specifies the variable name.

TYPE

specifies the variable type. Valid values are N for numeric variables and C for character variables

LENGTH

specifies the length of the variable.

LABEL Missing=""

specifies a label for the target variable.

FORMAT Missing=""

specifies a SAS format to format the variable.

LEVEL

specify either NOMINAL, ORDINAL, INTERVAL, or BINARY

ROLE

specify TARGET.

training.sas

This file is the optional SAS code that was used to train the model that you are importing. If at some time, SAS Model Manager reporting utilities detect a shift in the distribution of model input data values or a drift in the model's predictive capabilities, the training.sas code can be used to retrain the model on the newer data. If it is not available at import time, the training.sas code can be added at a later point using the SAS Model Manager Partial Import utility

training.log

This file is the optional log file that was produced when the model you are importing was trained. The information in the optional SAS training log can be helpful if the model must be retrained in the future.

training.lst

This file is the optional text output that is produced when the training.sas code is run. The information in the optional SAS training.lst table can be helpful if the model must be retrained in the future.

outest.sas7bdat

This data set contains output estimate parameters that are produced by a few SAS procedures, including the LOGISTIC procedure.

outmodel.sas7bdat

This data set contains output data that is produced by a few SAS procedures, including the LOGISTIC procedure and the ARBORETUM procedure. It contains complete information for later scoring by the same SAS procedure using the SCORE statement.

output.spk

This file is the SAS package file that contains the SPK collection of model component files.

format.sas7bcat

This file is the optional SAS formats catalog file that contains the user-defined formats for their training data. If the model that you are importing does not use a user-defined format, then you do not need to import a format.sas7bcat catalog file.

dataprep.sas

This file contains optional SAS code that is intended to be executed before each run of score code.

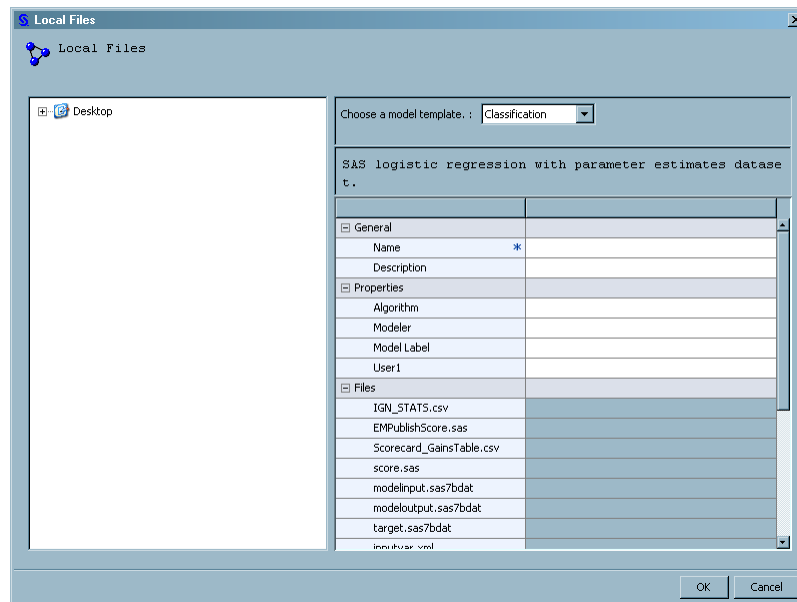
Importing a SAS Code Model

To import a SAS code model, follow these steps:

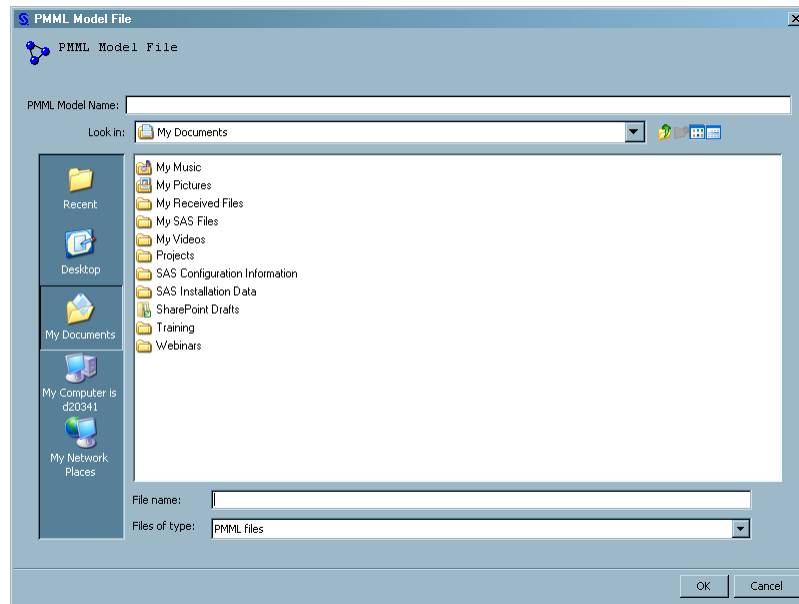
1. Copy your SAS code model and all of the associated metadata files to a location on your local workstation.
2. In the Project Tree, navigate to the project's version.

MMRoot ⇒ **<organizational folder>** ⇒ **<project folder>** ⇒ **<version folder>**

3. Right-click the **Models** folder and select **Import From** ⇒ **Local Files**.



4. Use the file utility icons to navigate to the folder on your computer that contains the component files for your model.
5. Select a template from the **Choose a model template** list. For more information about the type of model templates, see [“Model Templates” on page 84](#).
6. Enter a text value in the model **Name** field.
7. Complete the template fields. Drag the files from the left of the window to the corresponding file property on the right.



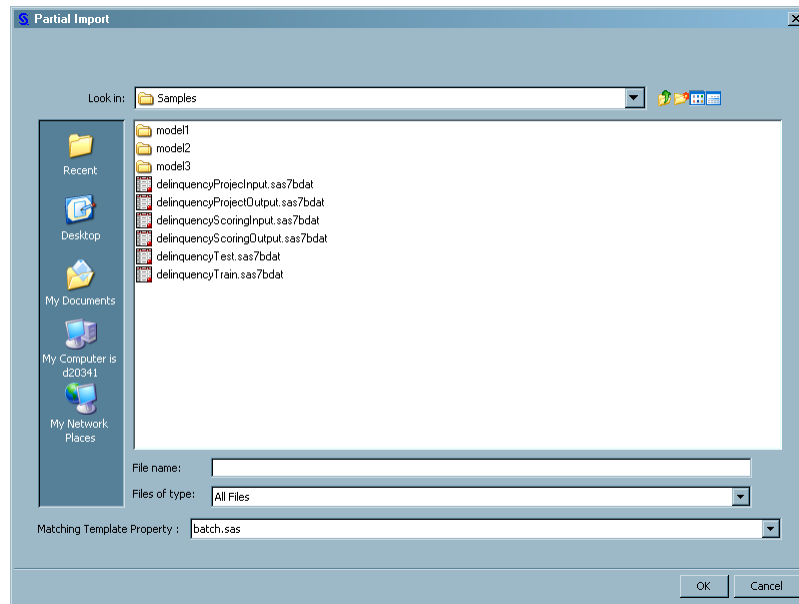
3. Enter a text value in the **PMML Model Name** field.
4. Navigate to the location of the PMML file and select the file.
5. Click **OK**. After the SAS Model Manager processes the model import request, the new model appears in the **Models** folder of your project's version.
6. Repeat steps 2 through 5 to import additional PMML files from your folder.

Import Partial Models

Suppose that you want to import a model, but you lack some of the model component files that are needed to complete a model import into SAS Model Manager. The Partial Model Import utility enables you to add files later that were not available when the model was originally imported.

To add a new file to your incomplete model in SAS Model Manager, follow these steps:

1. In the Project Tree, navigate to the project's version.
MMRoot ⇒ *<organizational folder>* ⇒ *<project folder>* ⇒ *<version folder>* ⇒ **Models**
2. Right-click the *<model name>* and select **Partial Import**. The Partial File Import window is displayed.



3. Select a file type from the **Matching Template Property** list.
4. Use the navigation icons to the locate the file to be imported and select the file.
5. Click **OK**.

You can also use the Partial Model Import utility to overwrite model component files that you have updated externally. If you use the Partial Import utility to import a model component file that already exists by the same name, the newer model component file will overwrite the older model component file.

See Also

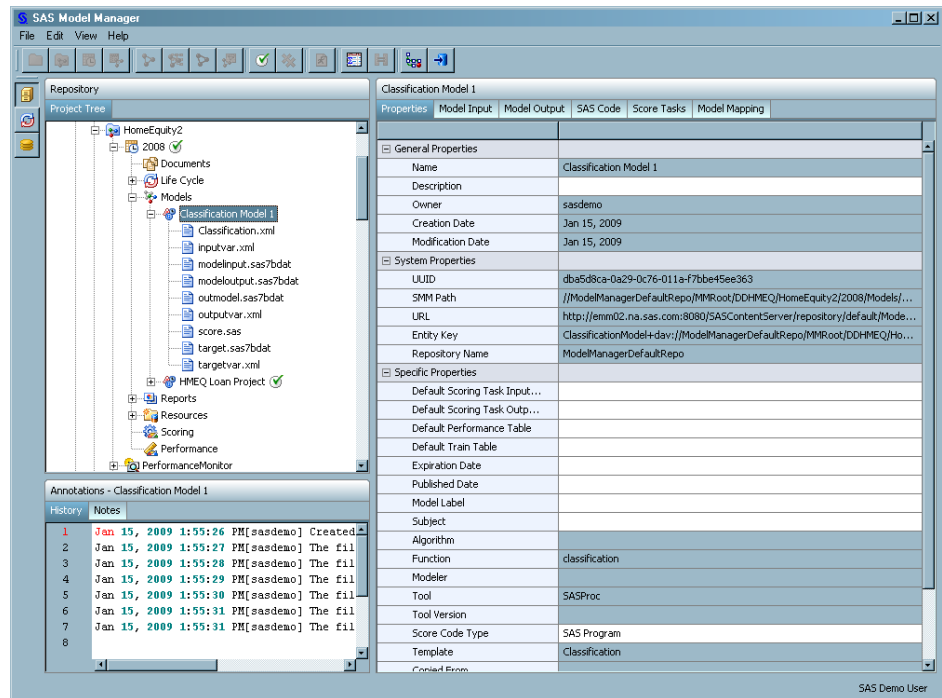
- [“Import SAS Code Models” on page 83](#)
- [“Set Model Properties” on page 94](#)
- [“Map Model Variables to Project Variables” on page 95](#)

Set Model Properties

After you import a model into SAS Model Manager you must specify additional property values for your imported model.

To set the model properties, follow these steps:

1. Select the model in the Project Tree.
2. View the **Properties** tab in the panel on the right.



3. Enter a model description in the General Properties section for your model, if you did not do so when the model was imported. This is the only property that you can edit in the General Properties section is the **Description**. For more information, see [“General Properties” on page 283](#).
4. The System Properties section is a Read-only section that is created after a model has been imported. The System Properties for models do not require any configuration after the model is imported into SAS Model Manager. To view a model's system properties, click the + icon to the left of **System Properties** to expand the section. For more information, see [“System Properties” on page 284](#).
5. Enter specific properties for a model. Some property values are automatically populated. Properties that appear in gray cannot be modified. For editable properties, click the white field, and then enter or select a value. For more information, see [“Specific Properties for a Model” on page 103](#).

See Also

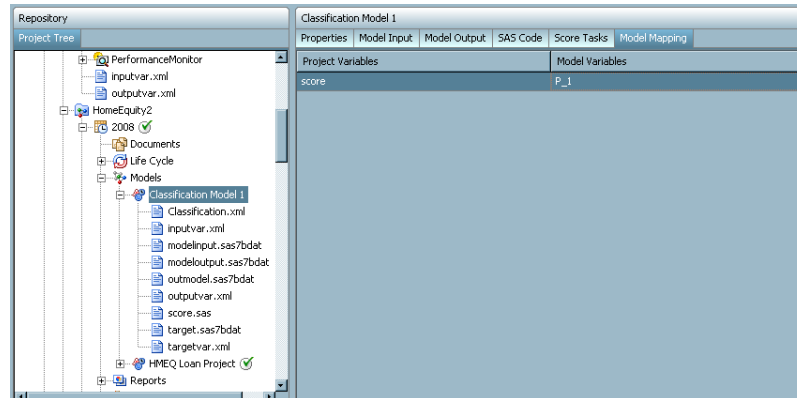
- [“Import Partial Models” on page 93](#)
- [“Map Model Variables to Project Variables” on page 95](#)

Map Model Variables to Project Variables

After a model has been imported and the remaining model properties are set on the **Properties** tab, you must map the model output variables to the project output variables. If the model output variable names already match the names of the project output variables names, mapping is unnecessary. For more information about project input and output tables, see [“Project Tables” on page 28](#).

To map the model variables to the project variables, follow these steps:

1. Select the model in your SAS Model Manager Project Tree.
2. Select the **Model Mapping** tab on the right.



3. Click the model variable field in the **Model Variables** column that is displayed next to the project variable that you want to map.
4. Select a model output variable from the list.
5. Repeat steps 3 and 4 for each model variable that requires mapping.

See Also

- [“Set Model Properties” on page 94](#)
- [“Overview of Importing Models” on page 79](#)

User-Defined Model Templates

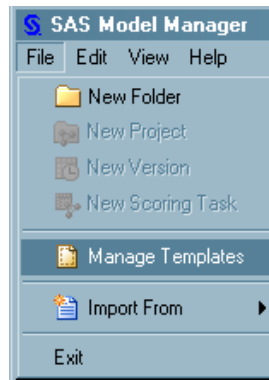
Creating a New Model Template

Overview of Model Templates

When you import a SAS code model, you must define the component files to be used in the model and specify the properties for the model. SAS Model Manager provides model templates that you can use as an example to create your own model template. You use the SAS Model Manager Template Editor to define model component files and to specify system and user properties for your model template. The model templates that are included with SAS Model Manager cannot be modified. For a list of the component files that must be created for the different model types, see [“Model Template Component Files” on page 86](#). For a list of properties, see [“Model Template Properties” on page 101](#).

Note: Only advanced users and administrators who have Write access to the middle-tier server where SAS Model Manager is installed can deploy templates after they have been created by a user. Several sample user template XML files are included with the SAS Model Manager installation package and can be made available by the server administrator and then used as a starting point for creating your own model template.

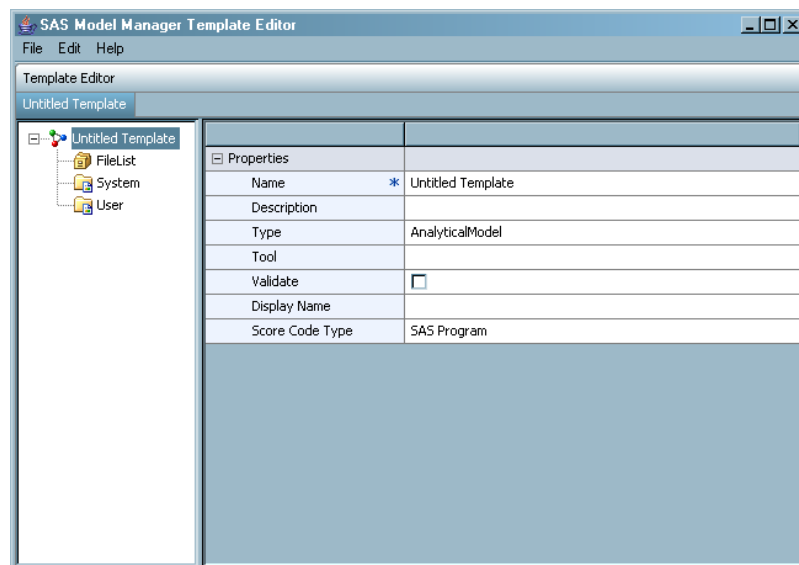
To open the Template Editor from the SAS Model Manager menu, select **File** ⇒ **Manage Templates**.



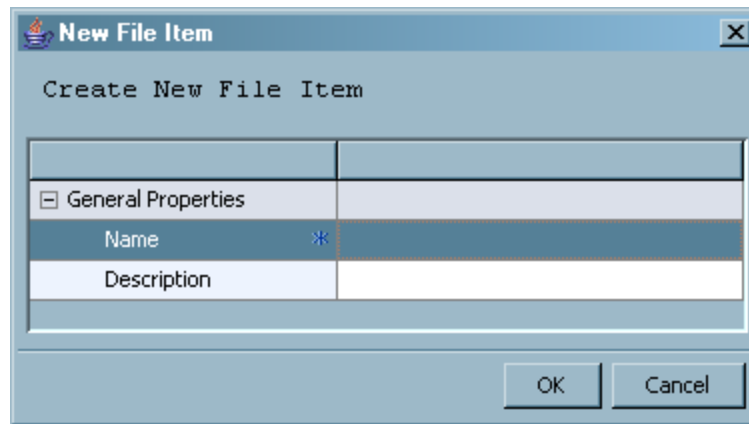
Create a New Model Template

To create a new model template, follow these steps:

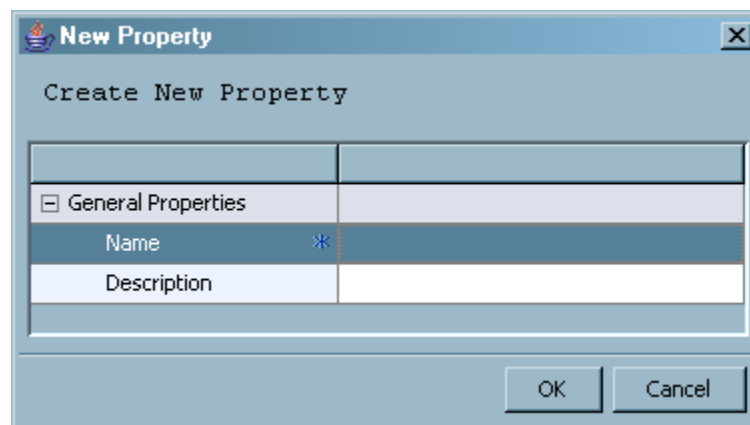
1. From the SAS Model Manager Template Editor window, select **File** ⇒ **New Model Template**. The Template Editor opens a template that has the name **Untitled Template**. Properties that display an asterisk (*) require a value for the property.



2. Assign values to the model **Properties** on the right. The **Name** field is required. Replace **Untitled Template** with the template name. For more information, see [“Model Template Properties”](#) on page 101.
3. Create a list of component files for the model. For each new filename, right-click the **FileList** folder and select **New File Item**. The New File Item window opens.



4. Enter a **Name** and **Description** for the file, and then click **OK**. For more information, see [“Model Template Component Files” on page 86](#).
5. After all required files have been created, select each filename and assign values for the properties on the right. For more information, see [“FileList Properties” on page 101](#).
6. To create a new property for the template, follow these steps:
 - a. Right-click the **System** or **User** folder and select **New Property**. The New Property window opens.



- b. Enter a **Name** and **Description** for the new property, and then click **OK**.
 - c. After all required properties have been created, select each property name and enter the property field values on the right. For more information, see [“System and User Properties” on page 102](#).
7. To save the template, select **File** ⇒ **Save As**. Then select a directory and filename for the template.
8. Click **Save**.

The template can now be copied to the middle-tier server by a user who has Write access to the SAS Model Manager user templates directory. The server administrator can then complete the deployment of the templates. For more information about the deployment of user templates, see the *SAS Model Manager: Administrator's Guide*.

Modify a Model Template

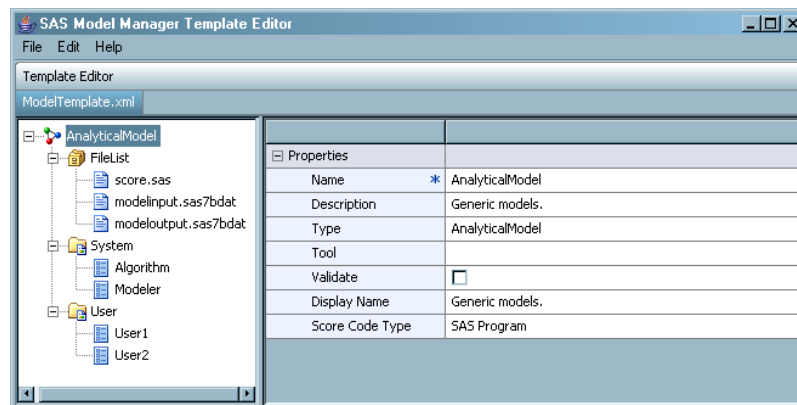
User model templates are stored on the middle-tier server in the `\sasconfigdir\Lev#\AnalyticsPlatform\apps\ModelManager\ext` directory. If you have Write

access to this directory, you can open and modify the templates in this directory. If you do not have Write access to this directory, copy the model templates to a local directory.

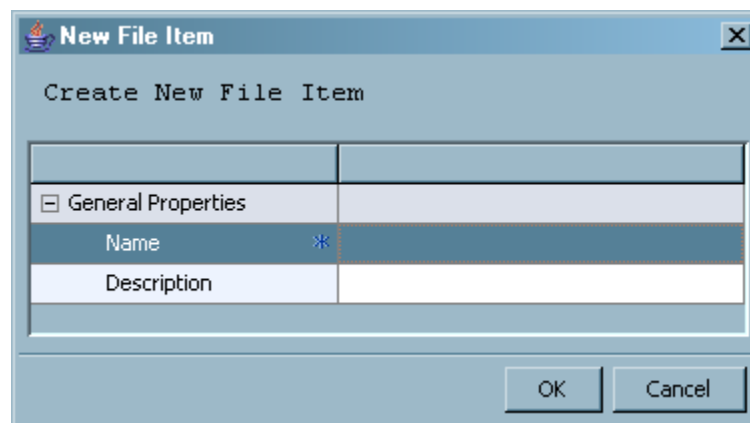
Note: Only user templates can be modified with the use of the SAS Model Manager Template Editor.

To modify a model template, follow these steps:

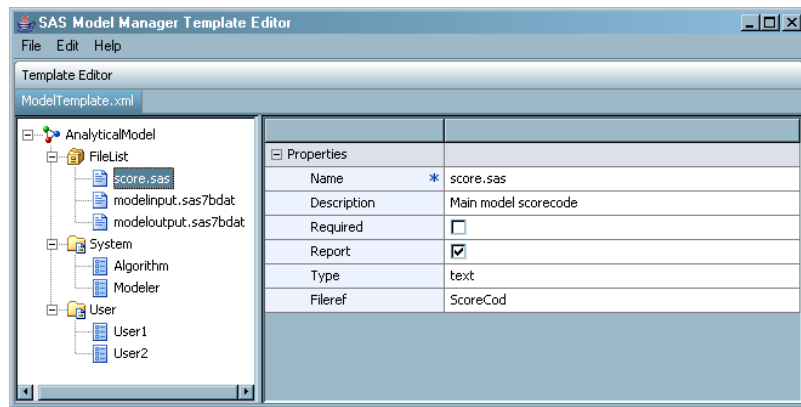
1. From the SAS Model Manager Template Editor window, select **File** ⇒ **Open**.
2. Select a model template, and then click **Open**. The template properties appear on the right.



3. To modify model template properties, select the property and make changes to the property value. For more information, see [“Template Properties” on page 101](#).
4. Create or modify file properties:
 - To create a new file property, right-click the **FileList** folder and select **New File Item**. Complete the properties, and then click **OK**.



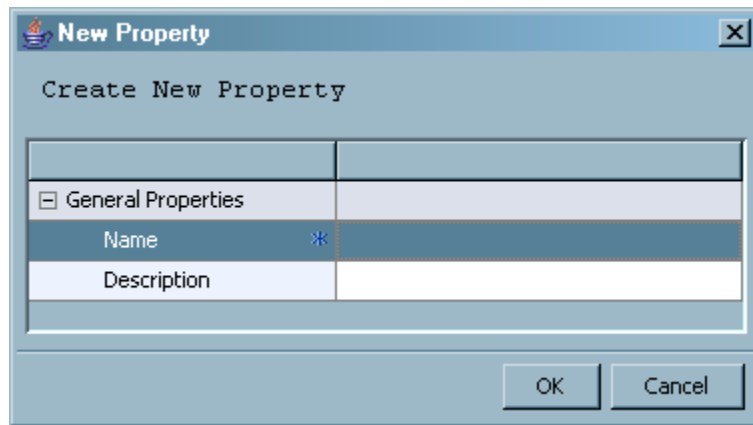
- To modify file properties, select the filename and make changes to the property values.



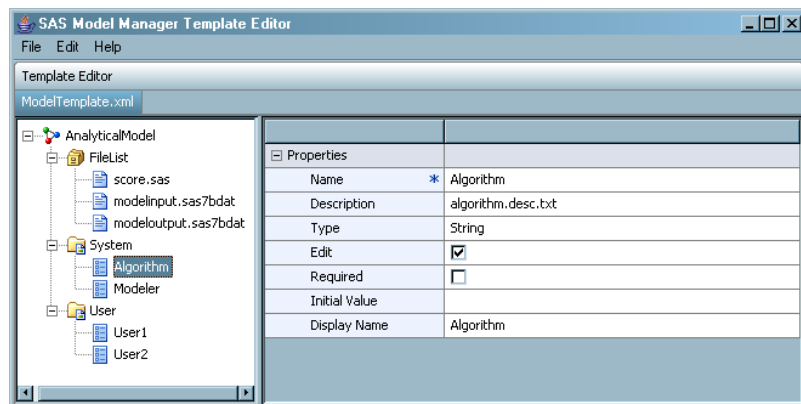
For more information, see [“FileList Properties”](#) on page 101.

5. Create or modify system and user properties:

- To create a new property, right-click the **System** or **User** folder and select **New Property**. Complete the properties, and then click **OK**.



- To modify system or user properties, select the property and make changes to the property values on the right.



For more information, see [“System and User Properties”](#) on page 102.

- To delete a file, system, or user property, right-click the property and select **Delete**.
- To save the template, do one of the following:
 - Select **File** ⇒ **Save** to save the changes to the existing template.
 - Select **File** ⇒ **Save As**, and then select a directory and filename for the template.

Note: The template can now be copied to the middle-tier server by a user who has Write access to the SAS Model Manager user templates directory. The server administrator can then complete the deployment of the templates. For more information about the deployment of user templates, see the *SAS Model Manager: Administrator's Guide*.

Model Template Properties

Template Properties

Here is a list of the general properties that define the model template.

Property Name	Description
Name	Identifies the name of the template. This property is required. The following characters cannot be used in the template name: @, \, /, *, %, #, &, \$, (,), !, ?, <, >, ^, +, \.
Description	Specifies user-defined information about the template.
Type	<p>Specifies the type of the model. SAS Model Manager supports the following model types:</p> <p>AnalyticalModel specifies the type of model that is associated with the Analytical model function.</p> <p>ClassificationModel specifies the type of model that is associated with the Classification model function.</p> <p>PredictionModel specifies the type of model that is associated with the Prediction model function.</p> <p>ClusteringModel specifies the type of model that is associated with the Segmentation model function.</p> <p>For more information about the model function types, see “SAS Model Manager Model Templates” on page 85.</p>
Tool	Specifies a text value that describes which tool is used to produce this type of model.
Validate	Indicates that SAS Model Manager verifies that all of the required files are present when users try to import a model into SAS Model Manager. If validation fails, the model will not be successfully imported.
Display Name	Specifies a text value that is displayed as the name of the model template.
Score Code Type	Specifies whether the imported model score code runs by using a DATA Step fragment, SAS Program code, or PMML .

FileList Properties

Here is a list of the FileList properties that specify the files that are contained in a model.

Property Name	Definition
Name	Identifies the name of the file. This property is required.
Description	Specifies user-defined information about the file.
Required	When it is selected, indicates that the file is a required component file of the model that must be imported.
Report	When it is selected, indicates that the file is to be included in a SAS package file when a model is published to a channel.
Type	Specifies a file whose type is text or binary.
Fileref	Specifies an eight-character (or fewer) SAS file reference to refer to this file in score.sas code. The fileref is assigned by SAS Model Manager when a SAS job is submitted.

Note: All user-defined models have three files.

- score.sas is the model's score code.
- modelinput.sas7bdat is a SAS data set whose variables are used by the model score code. The contents of the data set is not used by SAS Model Manager.
- modeloutput is a resulting data set when a user runs score.sas against modelinput.sas7bdat. The data set provides output variables that the model creates after a scoring task is executed. The contents of the data set is not used by SAS Model Manager.

System and User Properties

Here is a list of the system-defined and user-defined properties for a model template. Users can set these properties when they import a model.

Property Name	Description
Name	Identifies the name of the property. This is a required field.
Description	Specifies user-defined information about the property.
Type	Specifies a property whose type is String or Date.
Edit	Indicates that the property can be modified when importing a model or after the model is imported to SAS Model Manager.
Required	Indicates that the property is required.
Initial Value	Specifies a text string for the initial value for the property.
Display Name	Specifies a text value that is displayed as the name of the property.

Specific Properties for a Model

Here is a list of specific properties for a model that identify the fundamental model data structures and some of the critical model life cycle dates. Where applicable, project-based or version-based data structures automatically populate properties for model-based data structures.

Property Name	Description
Default Scoring Task Input Table	Specifies a default SAS data set that is used as the input data table for all of scoring tasks within the SAS Model Manager project. The model's Default Scoring Task Input Table property inherits the property value from the associated version or project, if one is specified.
Default Scoring Task Output Table	Specifies a default SAS data set that defines the variables to keep in the scoring results table and the scoring task output table. The model's Default Scoring Task Output Table property inherits the property value from the associated version or project, if one is specified.
Default Performance Table	Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager project. A model's Default Performance Table property inherits the property value from the associated version or project, if one is specified. If you do not specify a performance table, some of the SAS Model Manager Model Monitoring reports might not be enabled.
Default Train Table	The train table is optional and is used only as information. However, when a value is specified for a model's Default Train Table property, it is used to validate Teradata scoring functions when a user publishes the associated project champion model to a Teradata database.
Expiration Date	Specifies a date property by which the selected model is obsolete or needs to be updated or replaced. This property is for informational purposes and is not associated with any computational action by SAS Model Manager. This property is optional.

Property Name	Description
Model Label	Specifies a text string that is used as a label for the selected model in the model assessment charts that SAS Model Manager creates. If no value is provided for the Model Label property, SAS Model Manager uses the text string that is specified for the Model Name property. The Model Label property can be useful if the Model Name property that is specified is too long for use in plots. This property is optional.
Subject	Specifies a text string that is used to provide an additional description for a model, such as a promotional or campaign code. This property is for informational purposes and is not associated with any computational action by SAS Model Manager. This property is optional.
Algorithm	Specifies the computational algorithm that is used for the selected model. This property cannot be modified.
Function	Specifies the SAS Model Manager function class that was chosen when the SAS Model Manager associated project was created. The Function property specifies the type of output that models in the predictive model project generate. For more information, see “Overview of Importing Models” on page 79 .
Modeler	Specifies the Modeler ID or, when Modeler ID is missing, specifies the user ID of the individual that created the model which is stored in the SPK file for SAS Enterprise Miner models. Otherwise, the modeler can be specified during model import for local files into SAS Model Manager.
Tool	Specifies whether the imported model came from SAS Enterprise Miner or from other modeling tools.
Tool Version	Specifies the version number of the tool that is specified in the Tool property.
Score Code Type	Specifies whether the imported model score code is a DATA step fragment, ready-to-run SAS code, or a PMML file. Valid values are Data Step, SAS Program, and PMML.
Template	Specifies the SAS Model Manager model template that was used to import the model and to create pointers to its component files and metadata.
Copied From	Specifies where the original model is if this model is copied from another model in the SAS Model Manager repository.

Property Name	Description
Target Variable	Specifies the name of the target variable for a classification or prediction model. This property can be ignored for segmentation, cluster, and other models that do not use target variables. For example, if a model predicts when GENDER=M, then the target variable value is GENDER .
Target Event Value	Specifies a value for the target event that the model attempts to predict. This property is used only when a value is specified for the Target Variable property. For example, if a model predicts when GENDER=M, then the target event value is M .

Chapter 8

Scoring Models

Overview of Scoring Tasks	107
Scoring Task Tabbed Views	109
Create Scoring Output Tables	111
What Is a Scoring Output Table?	111
How to Create a Scoring Output Table	111
Create a Scoring Task	113
Modify a Scoring Task	114
Map Scoring Task Output Variables	115
Execute a Scoring Task	115
Graph Scoring Task Results	118
Generated Scoring Task Content Files	121
Scoring Task Properties	121
Result Set Properties	122

Overview of Scoring Tasks

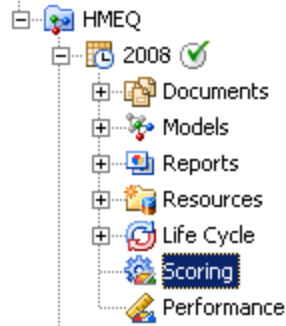
The purpose of a scoring task within SAS Model Manager is to run the score code of a model and produce scoring results that you can use for scoring accuracy and performance analysis. The scoring task uses data from a scoring task input table to generate the scoring task output table. The types of score code for a model that can be imported are a DATA step fragment and ready to run SAS code.

If your environment has its own means of executing the score code, then your use of the SAS Model Manager scoring tasks are mostly limited to testing the score code. Otherwise, you can use the SAS Model Manager scoring tasks to both test your score code and execute it in a production environment. Scoring results for a model in a test environment are stored on the SAS Content Server. Scoring results for a model in a production environment are written to the database or to the SAS library to which the scoring task output table is registered. In Windows, the scoring task output table in a SAS library must have Modify, Read & Execute, Read, and Write security permissions. For more information, see [“Scoring Task Output Tables” on page 29](#).

CAUTION:


Executing a scoring task in production mode overwrites the scoring task output table, which might result in a loss of data.

You create a new scoring task in the **Scoring** folder of your version. The following is an example of a **Scoring** folder under the version "2008".



These are the tasks that you perform as part of the Scoring Task workflow:

- Before creating a scoring task, you must create and register scoring task input and output tables. For more information, see [“Create Scoring Output Tables” on page 111](#) and [“Project Tables” on page 28](#).
- To create a new scoring task for a model you use the New Scoring Task window. When a new scoring task is successfully created, the new scoring task folder is selected under the **Scoring** folder. The scoring task tabbed view displays the various views of the scoring task information. For more information, see [“Create a Scoring Task” on page 113](#) and [“Scoring Task Tabbed Views” on page 109](#).
- Before you execute the scoring task it is recommended that you verify the scoring task output variable mappings on the Output Table view. For more information, see [“Map Scoring Task Output Variables” on page 115](#).
- After the scoring task output variables are mapped to the model output variables, it is recommended that you verify the model input variables against the scoring task input table columns. A convenient way to validate the scoring task input table is to use the

Quick Mapping Check tool . You can then execute the scoring task. For more information, see [“Execute a Scoring Task” on page 115](#).

- After the successful execution of the scoring task you can generate a number of graphical views that represent the contents of the output table. For more information, see [“Graph Scoring Task Results” on page 118](#).

See Also

- [“Create Scoring Output Tables” on page 111](#)
- [“Create a Scoring Task” on page 113](#)
- [“Modify a Scoring Task” on page 114](#)
- [“Map Scoring Task Output Variables” on page 115](#)
- [“Execute a Scoring Task” on page 115](#)
- [“Graph Scoring Task Results” on page 118](#)

Scoring Task Tabbed Views

Associated with each scoring task are tabbed views which provide you with a complete picture of a scoring task from its creation up through graphing the scoring results. The following is a list of these views:

Tabbed View	Description
Properties View	<p>This view contains four groupings of properties, some of which have fields that can be modified. Click Save, if you update any of the properties in this view. The groupings are as follows:</p> <ul style="list-style-type: none"> • “General Properties” on page 283 • “System Properties” on page 284 • “Scoring Task Properties” on page 121 • “Result Set Properties” on page 122
Model Input Variables View	This view shows the model input variables and attributes associated with the scoring task.
Input Table View	This view shows the input variables and attributes in the scoring task input table. For each input variable the model lists in the Model Input Variables View there must be a matching input variable in the input table. The input table can contain additional variables.
Model Output Variables View	This view shows the model output variables and attributes associated with the scoring task.
Output Table View	<p>This view shows the output variables and attributes in the scoring task output table. The value of the Map to Model Variable field must be the Model Output Variable that corresponds to the scoring task output variable in that row.</p> <p>When this view is displayed for the first time, SAS Model Manager attempts to discern the proper mapping and fills in the initial mapping values for you. If these are not correct or are incomplete, make the appropriate corrections and then click Save at the bottom right of the view.</p> <p>Note a Permission Denied message is displayed if a user accesses this view before the creation of the scoring task. The reason for this is because the first time this view is displayed, SAS Model Manager attempts to out the mappings it discerned. Since users who are only in the SAS Model Manager Users group do not have Write access, they cannot perform that task.</p>

Tabbed View	Description
Pre-code View	<p>This view contains the code that SAS Model Manager generates and places before the model's score code. The SAS Model Manager generated code is enclosed in comment tags that start and end the generated code. The generated code cannot be changed.</p> <p>After the generated code, you can append code that you want to have executed before the score code. You can use any variables, library references, or file references specified in the generated code section.</p>
Post-code View	<p>This view contains the code that SAS Model Manager generates and places after the model's score code. The SAS Model Manager generated code is enclosed in comment tags that start and end the generated code. The generated code cannot be changed.</p> <p>After the generated code, you can append code that you want to have executed after the score code. You can use any variables, library references, or file references specified in the generated code section.</p>
SAS Code View	<p>This view shows you the model's score code that is executed. This is the code that appears between the pre-code and the post-code. The SAS code cannot be modified from the Scoring Task tabbed view.</p> <p>To edit SAS code, follow these steps:</p> <ul style="list-style-type: none"> • Select the model in the Project Tree and click on the SAS Code tabbed view. • Select the Edit check box in the right corner of that view in order to make changes to the code. • Click Commit, to save your changes. • Refresh the scoring task's views to see the modified SAS code in this view. <p>Note that this view shows you the same code as the Model's SAS Code view. To see the code that was actually executed for a scoring task, expand the scoring task's folder and select the taskCode.sas file. If you copy this code into a SAS editor window and enter values for the user name and password you can run the code in SAS.</p>

Tabbed View	Description
Results View	<p>This view shows three separate views depending on which button at the bottom you click. Here are descriptions of each view:</p> <ul style="list-style-type: none"> • The Result Set is the view of the result data set. • The Log is the view of the log from the last execution of this scoring task. This is the default view. • The Output Table is the view of the listing file from the last execution of this scoring task. <p>Both the Log and the Output views shown here are from the files taskCode.log and taskCode.lst that can be found in the scoring task's folder.</p> <p>What is shown in the Result Set view depends on the value of the Scoring Task Type property. If the value of this property is Test then the results are read from the .sas7bdat file that is specified in the Output Table property. This file can be found in the scoring task folder. If the value of this property is Production, then the results are read from the location of the data source known to the SAS Metadata Repository. For more information, see “Create Scoring Output Tables” on page 111.</p>
Graph View	<p>This view displays graphs of the scoring task output that you create by clicking the Graph Wizard button. For more information, see “Graph Scoring Task Results” on page 118.</p>

See Also

[“Modify a Scoring Task” on page 114](#)

Create Scoring Output Tables

What Is a Scoring Output Table?

A scoring output table contains one or more output variables whose values contain the scoring results. You can create a scoring output table using the Create Output Table function directly from the model. SAS Model Manager saves the table to a SAS library that you specify. After SAS Model Manager creates the table, the table can be selected as the output table for subsequent scoring tasks. You map scoring output table variables to model output variables.

How to Create a Scoring Output Table

To create a scoring output table, follow these steps:

1. In the Project Tree, navigate to the **Models** folder.

MMRoot ⇒ <organizational folder> ⇒ <project folder> ⇒ <version folder> ⇒
Models

- Right-click the <model name> and select **Create Output Table** from the pop-up menu.

- Enter a name for the output table that is unique to the SAS library. The names in the **Library** selection list are the SAS libraries that are defined in the SAS Management Console repository under the Data Library Manager folder.
- Select a SAS library name from the **Library** list.
- Select the check box in the **Keep** column for the Input Table column names and the Generated Output variables that you want to include in the output table.
- Select the **Add Model ID** check box to add the **ModelID** variable to the output table.
- Select the **Use Project Mappings** check box to use the Project's Output Variable names.

Note: Generated Output Variable names are used when the new scoring output table is created if the model and project output variables mappings are not specified.

- Add the columns to the output table using one of these methods:
 - Click **Add Columns** to add the individual column information from each row that you selected in the Input Table and Generated Outputs table.
 - Click **Add All** to add all the columns in the Input Table and all the variables in the Generated Outputs table.

Note: To clear your output table selections, click **Remove All**.

- Click **OK**. The output table is created and the window is closed. You can view the output table in the Scoring Task Output Tables folder in the Data Sources Tree.

See Also

- “Overview of Scoring Tasks” on page 107

- “Project Tables” on page 28

Create a Scoring Task

To create a new scoring task, follow these steps:

Note: SAS Model Manager does not support scoring of PMML models.

1. Right-click the **Scoring** folder, then select **New** ⇒ **New Scoring Task** from the pop-up menu.

2. Enter the **Name** and **Description** of the scoring task.
3. Select a model from the **Select Model** list.

Note: When a model is selected the values in the **Select Input Table** field and **Select Output Table** fields change accordingly.

4. Select an input table from the **Select Input Table** list.
5. Select **Test** or **Production** for the Scoring Task Type.

Note: A best practice is to select **Test** before beginning all scoring tasks. Later, when you are satisfied with the results of running the scoring task and you are ready for it to be put into production you can change the type to **Production**. When you run in production mode in the Windows environment, a scoring task output table in a SAS library must have Modify, Read & Execute, Read, and Write security permissions.

6. Select the output table from the **Select Output Table** list.

7. Select the workspace server that will run the scoring task from the **Select Workspace Server** list.
8. Click **OK**.

Note: Four of the fields cannot be modified once the scoring task has been created. To change the following fields you must create a new scoring task.

- Name
- Model
- Input Table
- Output Table

See Also

- “Modify a Scoring Task” on page 114
- “Execute a Scoring Task” on page 115
- “Generated Scoring Task Content Files” on page 121
- “Scoring Task Properties” on page 121

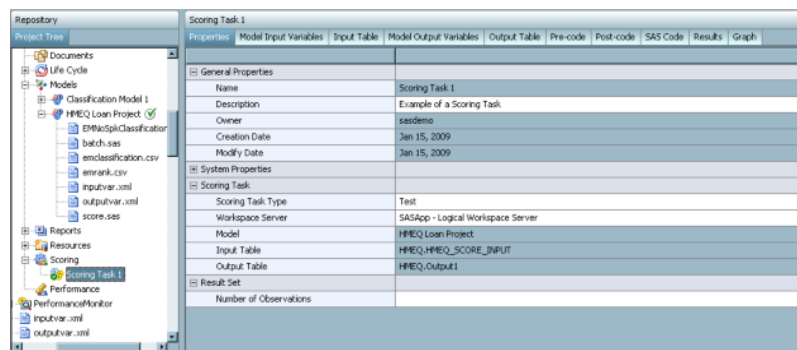
Modify a Scoring Task

The following are the only four tabbed views for a scoring task where the **Save** button exists. For more information, see “Scoring Task Tabbed Views” on page 109.

- **Properties**
- **Output Table**
- **Pre-code**
- **Post-code**

To modify a scoring task, follow these steps:

1. Expand the **Scoring** folder and select your scoring task.
2. Select the tab for which you want to modify the information. The default is the **Properties** tab.



3. Make the desired changes to the tab information.
4. Click **Save** to store the changes before proceeding to the next tabbed view.

See Also

- [“Create a Scoring Task” on page 113](#)
- [“Execute a Scoring Task” on page 115](#)
- [“Generated Scoring Task Content Files” on page 121](#)
- [“Scoring Task Tabbed Views” on page 109](#)

Map Scoring Task Output Variables

To map scoring task output variables to model output variables, follow these steps:

1. Expand the **Scoring** folder and select your scoring task.
2. Select the **Output Table** tab.

Output Table Variable Name	Type	Length	Map To Model Variable Name
score	N	8	score
I_bad	C	1	
P_0	N	8	

REASON
DELINQ
DEBTINC
NINQ
CLNO
CLAGE
MORTDUE

Save

3. For each **Output Table Variable Name** select a model output variable from the associated **Map to Model Variable Name** field.
4. Click **Save**.


See Also

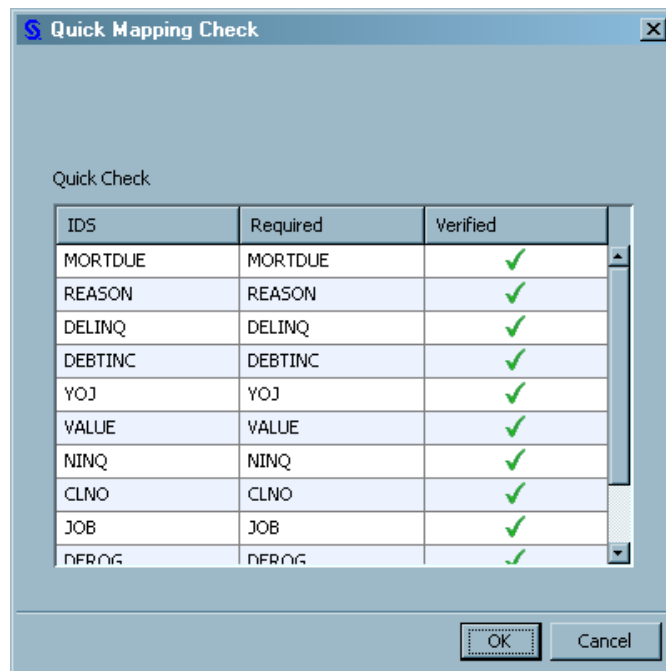
- [“Create Scoring Output Tables” on page 111](#)
- [“Create a Scoring Task” on page 113](#)
- [“Modify a Scoring Task” on page 114](#)
- [“Execute a Scoring Task” on page 115](#)

Execute a Scoring Task


To execute a scoring task, follow these steps:

1. Verify that you have mapped the model output variables to the scoring task output variables. For more information, see [“Map Scoring Task Output Variables” on page 115](#).

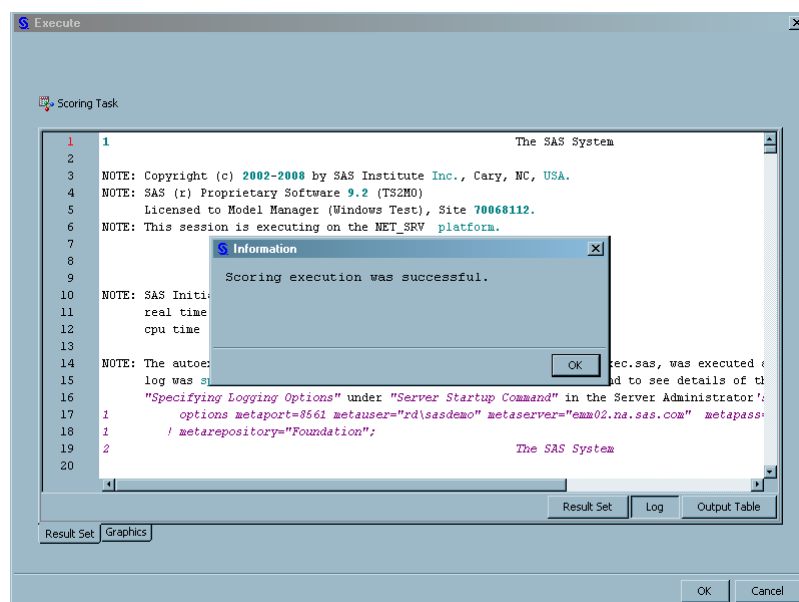
2. Select the scoring task name and click  in the toolbar to validate the input variables.



The table in the Quick Mapping Check window compares the column names from the Scoring Input Table, that is the Input Data Source, against the model's input variable names.

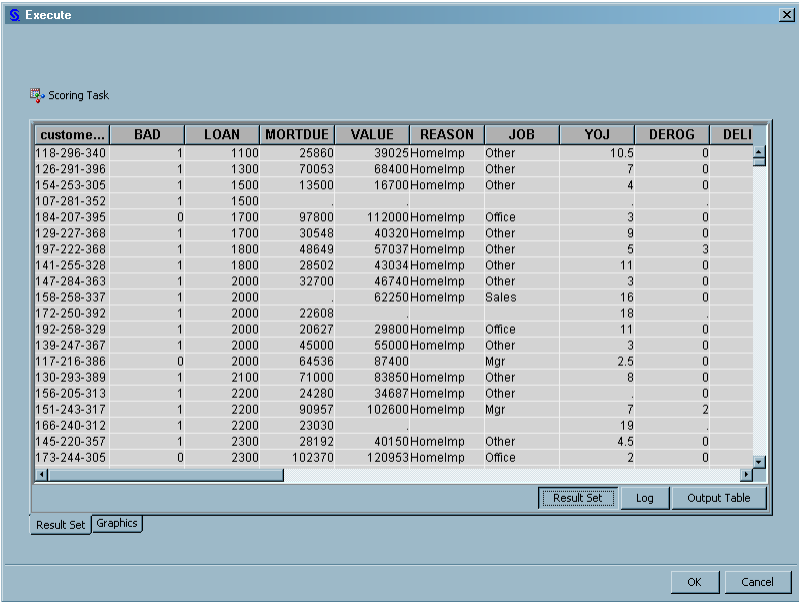
A green check mark  appears in the Verified column if the variable structures match; otherwise a red X appears if the input scoring table does not contain a variable that is used in the model. If one or more variables are not verified in the map, the integrity of the data in the generate Scoring Output Table is suspect.

3. Right-click the **Scoring Task** folder, and then select **Execute** from the pop-up menu. The Execute window displays and the task is executed.



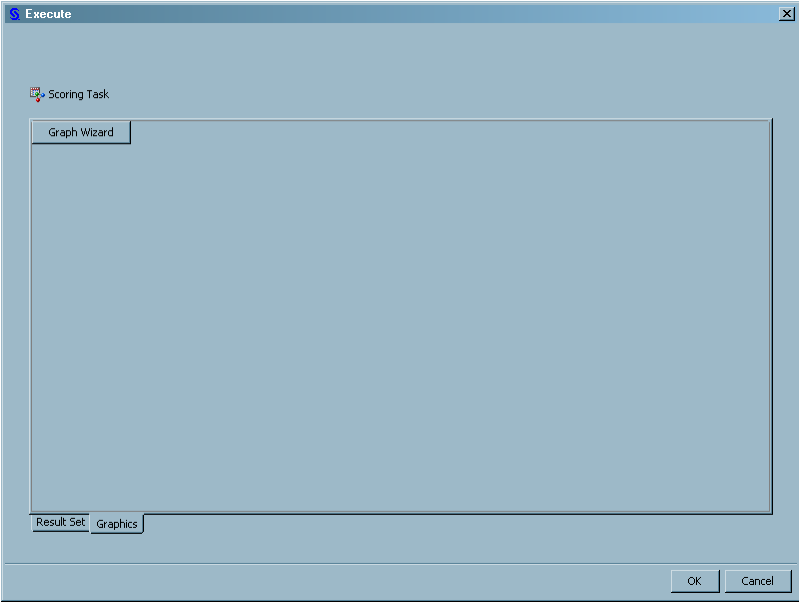
4. After the task has been completed a success or failure message is displayed, click **OK** and then review the log for error messages.

5. Click the **Result Set** tab to view the Scoring Task Results.



custome...	BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DEROG	DELI
119-296-340	1	1100	25860	39025	Homelmp	Other	10.5	0	
126-291-396	1	1300	70053	68400	Homelmp	Other	7	0	
154-253-305	1	1500	13500	16700	Homelmp	Other	4	0	
107-281-352	1	1500							
184-207-395	0	1700	97800	112000	Homelmp	Office	3	0	
129-227-368	1	1700	30548	40320	Homelmp	Other	9	0	
197-222-368	1	1800	48649	57037	Homelmp	Other	5	3	
141-255-328	1	1800	28502	43034	Homelmp	Other	11	0	
147-284-363	1	2000	32700	46740	Homelmp	Other	3	0	
158-258-337	1	2000		62250	Homelmp	Sales	16	0	
172-250-392	1	2000	22608				18		
192-258-329	1	2000	20627	29800	Homelmp	Office	11	0	
139-247-367	1	2000	45000	55000	Homelmp	Other	3	0	
117-216-386	0	2000	84536	87400		Mgr	2.5	0	
130-293-389	1	2100	71000	83850	Homelmp	Other	8	0	
156-205-313	1	2200	24280	34687	Homelmp	Other		0	
151-243-317	1	2200	90857	102600	Homelmp	Mgr	7	2	
166-240-312	1	2200	23030				19		
145-220-357	1	2300	28192	40150	Homelmp	Other	4.5	0	
173-244-305	0	2300	102370	120953	Homelmp	Office	2	0	

6. Click the **Graphics** tab to graph the results. For more information, see “[Graph Scoring Task Results](#)” on page 118.



7. Click **OK** to exit and save the result set.

Note: For a description of the content files that are created when a scoring task is executed, see “[Generated Scoring Task Content Files](#)” on page 121.

See Also

- “[Create a Scoring Task](#)” on page 113
- “[Modify a Scoring Task](#)” on page 114
- “[Generated Scoring Task Content Files](#)” on page 121

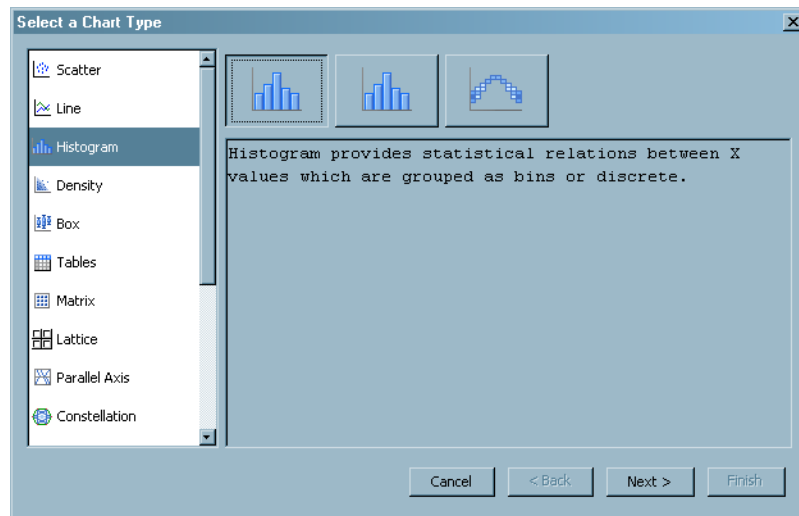
- “Graph Scoring Task Results” on page 118

Graph Scoring Task Results

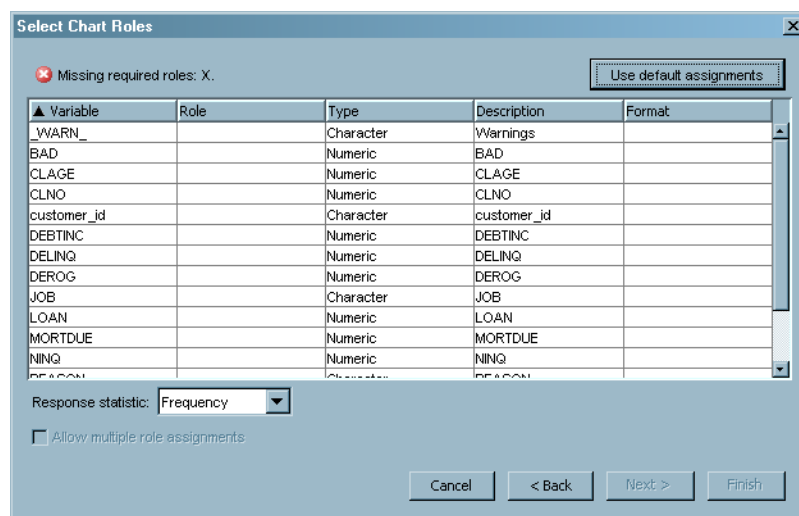
After the scoring task successfully executes, you can use the graphics wizard to plot your tailored charts.

To graph scoring task results, follow these steps:

1. Select the **Graph** tabbed view of the scoring task . The Graphics Wizard window opens.
2. Click **Graph Wizard**. The Select a Chart Type window opens.



3. Select a chart type from list box on the left, and then select the chart display button on the right. A description of how the results are graphed for a chart displays.
4. Click **Next**. The Select Chart Roles window opens.



5. For each variable, select the row and then select a value from the **Role** list.

Select Chart Roles

Use default assignments

Variable	Role	Type	Description	Format
EMP_PROBABILITY		Numeric	Probability of classm...	
F_BAD		Numeric	F_BAD	
I_BAD		Character	Intro: BAD	
JOB		Character	JOB	
LOAN		Numeric	LOAN	
MORTDUE		Numeric	MORTDUE	
NINQ		Numeric	NINQ	
P_BAD0		Numeric	Predicted: BAD=0	
P_BAD1	X	Numeric	Predicted: BAD=1	
REASON		Character	REASON	
U_BAD		Numeric	Unnormalized Intro: BAD	
VALUE		Numeric	VALUE	
YOU		Numeric	YOU	

Response statistic: Frequency

☐ Allow multiple role assignments

Cancel < Back Next > Finish

Note: You can also click **Use default assignments** to assign variables to the required roles.

6. Click **Next**.
7. Follow the Graphics Wizard to define the options such as data parameters, color, title, legend, and so forth.

The following are examples of the options that you can define:

Data Where Clause

Column name: Predicted: BAD=1 (P_BAD1) Operator: Greater than or equal... Value: .10

AND Column name: Predicted: BAD=1 (P_BAD1) Operator: Less than or equal... Value: .30

Add Apply Custom Delete Reset

Cancel < Back Next > Finish

Chart Titles

Title: Example Chart

Footnote:

Legend Label:

X Axis Label: Event Posterior Probability

Y Axis Label:

Cancel < Back Next > Finish

Chart Legends

☐ Legend

☐ Top

☐ Bottom

☐ Left

☐ Right

Axes

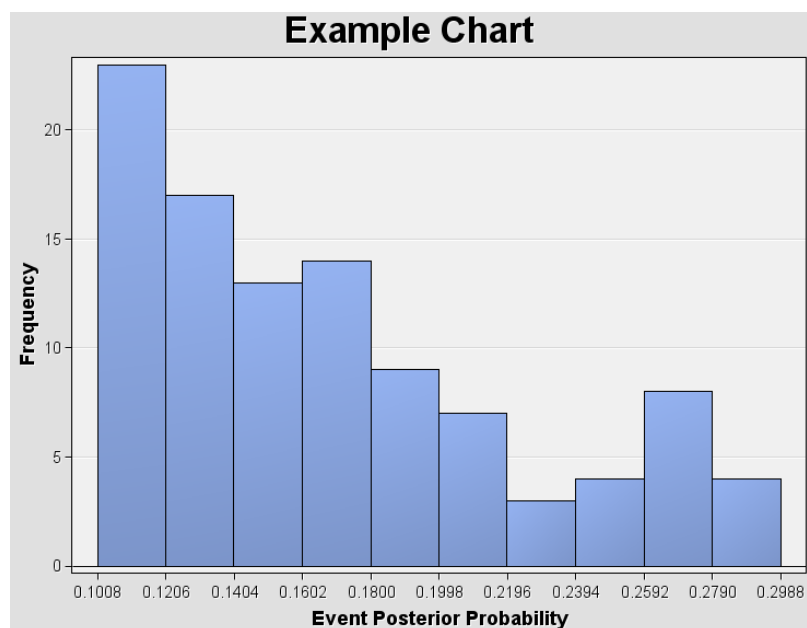
☒ Horizontal

☒ Vertical

☐ Depth

Cancel < Back Next > Finish

8. Click **Finish** to display the graph.



See Also

[“Execute a Scoring Task” on page 115](#)

Generated Scoring Task Content Files

At various points in the life cycle of a scoring task the SAS Model Manager user can create any or all of the content files described below. After the files are created they are written to the scoring task folder.

The conditions under which they are created and a description of their content follows:

Filename	Description
taskCode.sas	This file is created the first time you execute the scoring task. This file is updated each time you execute a scoring task. It contains the code which was last sent to the Workspace Server for execution.
taskCode.log	This file is the SAS log for the scoring task code that is executed on the Workspace Server. The SAS log file is in sync with the taskCode.sas file.
taskCode.lst	This file is the SAS listing file and is created only if the score code executes code that produces a listing file.
preScoreCode.sas	This file is created only if you add code after the generated code section in the Pre-code view. For more information, see “Scoring Task Tabbed Views” on page 109 .
postScoreCode.sas	This file is created only if you add code after the generated code section in the Post-code view. For more information, see “Scoring Task Tabbed Views” on page 109 .
<Output Table Name>.sas7bdat	This file is created whenever the scoring task executes and the scoring task property Scoring Task Type is set to Test. The contents are not the most recent scoring output results if the type of the scoring task was changed from Test to Production, and the scoring task is executed.

See Also

- [“Execute a Scoring Task” on page 115](#)
- [“Scoring Task Tabbed Views” on page 109](#)

Scoring Task Properties

Here is a list of the scoring task properties that provide information specific to the scoring task.

Property Name	Description
Scoring Task Type	Specifies a value of Test or Production for the type of scoring task.
Workspace Server	Specifies the name of the Workspace Server in which SAS Model Manager is connected. This value is taken from the SAS Metadata Repository.
Model Name	Specifies the name of the model whose score code is to be executed on the Workspace Server. This value is set when the scoring task is created and cannot be modified.
Input Table	Specifies the name of the input table (data source) to be used in scoring. This value is set when the scoring task is created and cannot be modified.
Output Table	Specifies the name of the output table to be used in scoring. This value is set when the scoring task is created. If the Scoring Task Type is Test then this property identifies the name of the output file (<i>output_filename.sas7bdat</i>) that is created by the SAS Workspace Server when the score code is executed. Upon creation the output file is placed in the scoring task's folder. If the Scoring Task Type is Production , then this identifies the output table where the results of the scoring are written.

See Also

- [“Create a Scoring Task” on page 113](#)
- [“Modify a Scoring Task” on page 114](#)
- [“Execute a Scoring Task” on page 115](#)

Result Set Properties

Here is a list of result set properties that provide information specific to the scoring task.

Property Name	Description
Number of Observations	This is an editable field whose value is used only when the Scoring Task Type is set to Test . It specifies how many observations are to be read from the scoring task input table. This enables you to limit the amount of records written to the scoring task output table on the SAS Content Server in order to reduce operation costs. If a value is not specified, the default value of 1000 rows is used for the number of observations.

See Also

- [“Create a Scoring Task” on page 113](#)
- [“Modify a Scoring Task” on page 114](#)

Chapter 9

Validating Models Using Comparison Reports

Overview of Model Comparison Reports	125
What Is a Model Comparison Report ?	125
Model Comparison Input File	126
Model Comparison Output Files	126
Model Profile Reports	127
About Model Profile Reports	127
Create a Model Profile Report	127
Creating Delta Reports	129
About Delta Reports	129
Create a Delta Report	129
Creating Dynamic Lift Reports	131
About Dynamic Lift Reports	131
Verify Project and Model Property Settings	131
Create a Dynamic Lift Report	132
View Reports	134

Overview of Model Comparison Reports

What Is a Model Comparison Report ?

The SAS Model Manager model comparison reports are tools that you can use to evaluate and compare the candidate models in a version or across versions to help you select and approve the champion model that moves to production status. The SAS Model Manager model comparison reports are analytical tools that project managers, statisticians, and analysts can use to assess the structure, performance, and resiliency of candidate models.

The reports present information about a number of attributes that can impact model performance. Together, the reports provide qualified information that can serve as the analytical basis for choosing and monitoring a champion model.

Here is a description of the comparison reports:

Model Profile Report

For a single model, this report displays the profile data associated with input, output, and target variables. Profile data includes the variable name, type, length, label, SAS format, measurement level and role.

Delta Report

This report compares the profile data for two models and notes the differences.

Dynamic Lift Report

The Dynamic Lift report provides visual summaries of the performance of one or more models for predicting a binary outcome variable.

You create Model Comparison reports using the **New Report Wizard** that you start from a version's **Reports** node.

Model Comparison Input File

SAS Model Manager uses a test table as the input table for the Dynamic Lift report. Before you run the New Report Wizard to create a Dynamic Lift report, make sure that a test table has been added to the **Test** folder in the Data Sources perspective and that the test table is specified in the project property **Default Test Table**.

See Also

- [“Specific Properties for a Project” on page 285](#)
- [“Creating a Test Table” on page 33](#)
- [“Add a Data Source Table” on page 35](#)

Model Comparison Output Files

The New Reports Wizard stores the model comparison output files in a report node under the **Reports** folder. The name of the report node is the value of the **Name** field that you specified in the New Report Wizard **Report Properties**.

Each time you run the New Report Wizard, the wizard creates four files:

- the report in either HTML, PDF, RTF, or Excel format
- taskCode.log
- taskCode.lst
- taskCode.sas

CAUTION:

The wizard overwrites the output files if output files of the same name already exist.

Report File	Description
<i>report-name.html</i>	This file is the report output in HTML format.
<i>report-name.pdf</i>	This file is the report output in PDF format.
<i>report-name.rtf</i>	This file is the report output in RTF format.
<i>report-name.xls</i>	This file is the report output in Excel format.
taskCode.log	This file is the log file that contains messages from running the SAS code to create the report.

Report File	Description
taskCode.lst	This file is the report output in the SAS listing output if the ODS Listing destination was set; otherwise, an empty file.
taskCode.sas	This file is the SAS code that is used to create the report.

After you create a report, you view the report from the **Reports** folder.

Model Profile Reports

About Model Profile Reports

A Model Profile report displays the profile data that is associated with the model input variables, output variables, and target variables. The report creates three tables, one each for the model input, output, and target variables.

Here is a description of the model profile data:

Profile Data	Description
Name	the name of the variable
Type	the data type of the variable: character (C) or numeric (N)
Length	the length of the variable
Label	a label associated with the variable
Format	the SAS format associated with formatting the variable
Level	the measurement level: nominal, ordinal, interval, or binary
Role	the type of variable: input, output, or target

The reports are created using these auxiliary model files:

- inputvar.xml
- outputvar.xml
- targetvar.xml

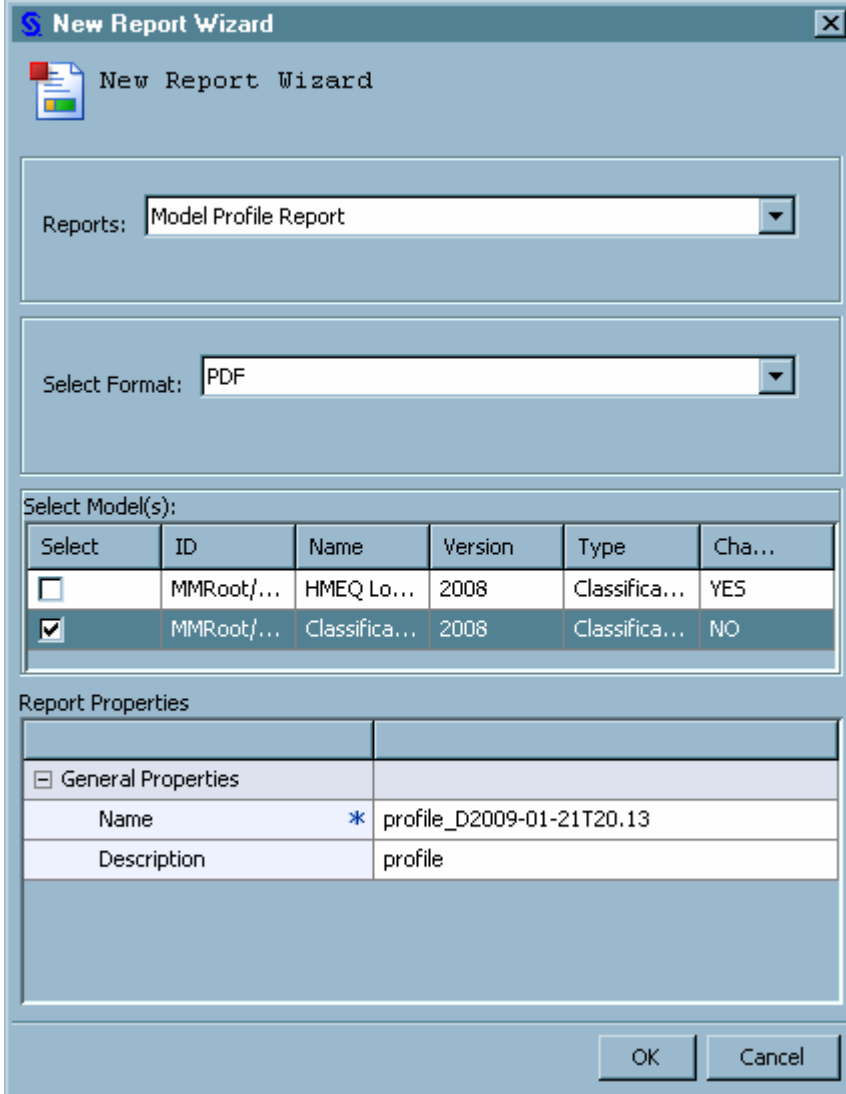
These are the tasks that you perform for Model Profile reports:

- [“Create a Model Profile Report” on page 127](#)
- [“View Reports” on page 134](#)

Create a Model Profile Report

To create a Model Profile report, follow these steps:

1. Expand the version folder .
2. Right-click the **Reports** node and select **Reports** ⇒ **New Report Wizard**. The New Report Wizard opens.



The **New Report Wizard** dialog box is shown. It has a title bar with a close button. Below the title bar is a section with a document icon and the text "New Report Wizard".

There are two dropdown menus:

- Reports:** Set to "Model Profile Report".
- Select Format:** Set to "PDF".

Below these is a section titled "Select Model(s):" containing a table:

Select	ID	Name	Version	Type	Cha...
<input type="checkbox"/>	MMRoot/...	HMEQ Lo...	2008	Classifica...	YES
<input checked="" type="checkbox"/>	MMRoot/...	Classifica...	2008	Classifica...	NO

Below the table is a section titled "Report Properties" containing a table:

Report Properties	
<input type="checkbox"/> General Properties	
Name	* profile_D2009-01-21T20.13
Description	profile

At the bottom right are **OK** and **Cancel** buttons.

3. Select **Model Profile Report** from the **Reports** list box.
4. In the **Select Format** list box, select the type of output that you want to create. The default is **PDF**. Other options are **HTML**, **RTF**, and **Excel**.
5. In the **Select** column of the **Select Model(s)** table, check the box for the model that you want to include in the report. This report requires only one model.
6. In the **Report Properties** table, complete the **Name** and **Description** properties if you do not want to use the default values. The default value for the **Name** property uses the form `profile_DdateTtime`. The following characters cannot be used in the name: `@`, `\`, `/`, `*`, `%`, `#`, `&`, `$`, `(`, `)`, `!`, `?`, `<`, `>`, `^`, `+`, `\`.
7. Click **OK**. A message confirms that the report was created successfully.

See Also

[“View Reports” on page 134](#)

Creating Delta Reports

About Delta Reports

A Delta report compares the input, output, and target variable attributes for each of the variables that are used to score two candidate models. Delta reports display the differences in the variables of competing candidate models. The report output is a table that groups the variables by the variable name. For each variable, the reports lists the attribute value for each model and whether the attribute value is the same or different than the other attribute values.

Here is a description of each of the columns in the output of a Delta report:

Column	Description
Role	Specifies the function that a variable performs in determining a score code.
Name	Specifies the name of the variable that is being compared.
Variable Attribute	Specifies the name of the variable attribute that is being compared.
<i>Model Name-1</i>	Contains the value of the attribute for the first model.
<i>Model Name-2</i>	Contains the value of the attribute for the second model.
Difference	Specifies an X if the value of the variable attribute is different than the value of the variable attributes in the other model. If the value of the variable attribute is the same, this column is blank.

These are the tasks that you perform for Delta Reports:

- [“Create a Delta Report” on page 129](#)
- [“View Reports” on page 134](#)

Create a Delta Report

To create a Delta report, follow these steps:

1. Expand the version folder .
2. Right-click the **Reports** node and select **Reports** ⇒ **New Report Wizard**. The New Report Wizard opens.

New Report Wizard

Reports: Delta Report

Select Format: PDF

Select Model(s):

Select	ID	Name	Ver...	Type	Ch...
<input checked="" type="checkbox"/>	MMRoo...	HMEQ L...	2008	Classific...	YES
<input checked="" type="checkbox"/>	MMRoo...	Classific...	2008	Classific...	NO

Report Properties

General Properties	
Name	* delta_D2009-01-22T07.15
Description	delta

OK Cancel

3. Select **Delta Report** from the **Reports** list box.
4. In the **Select Format** list box, select the type of output that you want to create. The default is **PDF**. Other options are **HTML**, **RTF**, and **Excel**.
5. In the **Select** column of the **Select Model(s)** table, select the check boxes for the two models that you want to include in the report.
6. In the **Report Properties** box, complete the **Name** and **Description** properties if you do not want to use the default values. The default value for the **Name** property uses the form `delta_DdateTtime`. The following characters cannot be used in the name: @, \, /, *, %, #, &, \$, (,), !, ?, <, >, ^, +, \.
7. Click **OK**. A message confirms that the report was created successfully.

See Also

[“View Reports” on page 134](#)

Creating Dynamic Lift Reports

About Dynamic Lift Reports

The Dynamic Lift report enables you to view a model's lift at a given point in time or to compare the lift performance of several models on one chart. The Dynamic Lift report creates the following charts:

- Lift
- Cumulative Lift
- Percent Response
- Cumulative Percent Response
- Captured Response
- Cumulative Captured Response

A Dynamic Lift report can be created only for classification models with a binary target.

The charts that are created for a Dynamic Lift report are also created in the Monitoring report, which creates multiple types of model comparison reports.

For information about the tasks that you perform for Dynamic Lift reports, see the following topics:

- [“Verify Project and Model Property Settings” on page 131](#)
- [“Create a Dynamic Lift Report” on page 132](#)
- [“View Reports” on page 134](#)

Verify Project and Model Property Settings

Verify Project Properties

Select the project name and verify that the following project properties are set:

Default Test Table

Specifies a test table that is listed in the **Test Tables** data source. The test table must contain the target variable, as well as values for the variables that are defined by the project input variables.

Training Target Variable

Specifies the name of the target variable that was used to train the model.

Target Event Value

Specifies the value for the desired target variable event or state. For example, if a model predicts when RESPONSE=YES, then the target event value is **YES**.

Output Event Probability Variable

Specifies the name of the output event probability variable.

Verify Model Properties

For each model in the Dynamic Lift report, click on the model name and verify the specified properties on the following tabs:

Model Mapping

Click the **Model Mapping** tab and verify that the model variables are mapped to the project variables. If the variable names are the same, you do not need to map the variables. If they are not mapped, for each project variable, click the **Model Variables** property and select a variable name.

Properties

Property	Description
Target Variable	Specifies the name of the target variable. For example, if a model predicts when RESPONSE=YES, then the target variable is RESPONSE .
Target Event Value	Specifies a value for the target event that the model attempts to predict. For example, if a model predicts when RESPONSE=YES, then the target event value is YES .
Score Code Type	Specifies whether the score code runs using a DATA step fragment or SAS code that is not a DATA step fragment.

Note: Dynamic lift reports are not applicable to models whose **Score Code Type** property has a value of PMML.

Create a Dynamic Lift Report

After ensuring that the appropriate project and model properties have been set, follow these instructions to create a Dynamic Lift report:

1. Expand the version folder .
2. Right-click the **Reports** node and select **Reports** ⇒ **New Report Wizard**. The New Report Wizard opens.

New Report Wizard

Reports: Delta Report

Select Format: Dynamic Lift Report

Select Model(s):

Select	ID	Name	Ver...	Type	Cha...
<input checked="" type="checkbox"/>	MMRoot...	HMEQ ...	2007	Classific...	NO
<input checked="" type="checkbox"/>	MMRoot...	HMEQ L...	2007	Predicti...	YES

Report Properties

☒ General Properties

Name	*	delta_D2009-01-22T07.24
Description		delta

OK Cancel

3. Select **Dynamic Lift Report** from the **Reports** list box.
4. In the **Select Format** list box, select the type of output that you want to create. The default is **PDF**. Other options are **HTML**, **RTF**, and **Excel**.
5. In the **Select** column of the **Select Model(s)** table, check the boxes for the models that you want to include in the report.
6. In the **Report Properties** box, complete the **Name** and **Description** properties if you do not want to use the default values. The default value for the **Name** property uses the form `dynamicLift_DdateTtime`. The following characters cannot be used in the name: `@, \, /, *, %, #, &, $, (,), !, ?, <, >, ^, +, \`.
7. Click **OK**. A message confirms that the report was created successfully.
8. If errors occurred, ensure that the prerequisite properties have been set correctly or correct the reported model configuration error.

See Also

- [“Verify Project and Model Property Settings” on page 131](#)
- [“View Reports” on page 134](#)

View Reports

To view a report, follow these steps:

1. Expand the version folder and the **Reports** folder.
2. Right-click the report name and select **Reports** ⇒ **View Report**. The report opens.

Chapter 10

Validating Models Using User Reports

Overview of User Reports	135
Ad Hoc Reports and User-Defined Reports	135
Comparison of Ad Hoc and User-Defined Reports	136
Output Created by User Reports	136
Ad Hoc Reports	137
Overview	137
Create an Ad Hoc Report	137
Example Ad Hoc Report	138
User-Defined Reports	140
Overview of User-Defined Reports	140
Create a User-Defined Report	140
The Report Template	141
Defining Macro Variables For a User-Defined Report	143
Run a User-Defined Report	143
Example User-Defined Report	143

Overview of User Reports

Ad Hoc Reports and User-Defined Reports

User reports are SAS programs that you create and import to SAS Model Manager so that you can tailor reports to meet your business requirements. The ad hoc report enables you to develop, test, and run your report from within SAS Model Manager. The user-defined report can be developed either within or external to SAS Model Manager. It requires a SAS program and the associated auxiliary files to be installed in a directory available to SAS Model Manager and is run using the New Report Wizard.

Using ad hoc reports, you modify and submit your code from the SAS Editor within the Create Ad Hoc Reports window. Ad Hoc reports are defined and can be run only under the version where it was created.

A user-defined report is a report that is available for reporting on all models in SAS Model Manager. The user-defined report requires three files to be installed in your server's file structure:

- a SAS program to create the report
- a report template XML file that specifies the report requirements, such as report name and the number of required models to run the report

- a SAS program file that lists the SAS Model Manager global macro variables and macros that are used in your report.

All user-defined report files are typically installed by a SAS Model Manager administrator or advanced user to a server directory that is accessible to SAS Model Manager. For instructions on creating installing a user-defined report see *SAS Model Manager: Administrator's Guide*.

The ad hoc report can be used to develop, test, and debug user-defined reports. When your ad hoc report is ready for a production environment, you can create the report template XML file and the macro file, and install the three files to the User-defined report file structure.

Comparison of Ad Hoc and User-Defined Reports

Report Difference	Ad Hoc Report	User-Defined Report
Version	Ad hoc reports are defined and can be run only under the version where it was created.	User-defined reports can be run under any project version.
Report template	The ad hoc report does not require a template.	User-defined reports require a template to define the report parameters.
Report results	Each time an ad hoc report is run, the existing report files are overwritten.	Each time a user-defined report is run, a new report folder is created under the Reports node.
Location of SAS files used to generate the report	The ad hoc report SAS program is stored in the report folder for the version where it was created.	The user-defined report SAS files are located in a server directory accessible to SAS Model Manager.

Output Created by User Reports

The first time that you create a report, SAS Model Manager creates a node for the report under the **Reports** node.

Each time you run the New Report Wizard, the wizard creates these files:

- the report in either HTML, PDF, RTF, or Excel format
- `smm_userCode.sas`
- `taskCode.log`
- `taskCode.lst`
- `taskCode.sas`

CAUTION:

The wizard overwrites the output files if output file of the same name already exist.

Here is a description of the ad hoc report output files:

Report File	Description
<i>report-name.html</i>	This file is the report output in HTML format.
<i>report-name.pdf</i>	This file is the report output in PDF format.
<i>report-name.rtf</i>	This file is the report output in RTF format.
<i>report-name.xls</i>	This file is the report output in Excel format.
smm_userCode.sas	This file contains the SAS program report code that was submitted in the Create Ad Hoc window.
taskCode.log	This file is the log file that contains messages from running the SAS code to create the report.
taskCode.lst	This file is the report output in the SAS listing output if the ODS LISTING destination was set; otherwise, an empty file.
taskCode.sas	This file is the SAS code that is used to create the report. The file contains the user-defined report code as well as code generated by SAS Model Manager to create the report.

You can see the contents of these files by selecting them in the Project Tree. You can also see the taskCode.sas file and the taskCode.log files by selecting the report name. SAS Model Manager displays tabs for these files to the right of the Properties tab.

Ad Hoc Reports

Overview

To create an ad hoc report, you must first write a SAS report program. When the report code is ready, you run the Create Ad Hoc Report wizard and copy your program to the **SAS Editor** tab. You then submit your program. Unlike the user-defined report, the ad hoc report does not require auxiliary files to be saved in the middle-tier server user templates directory.


To create your report output in either HTML, PDF, RTF, or Excel, you modify your report with code provided by SAS that enables you specify the report output format. The code that you need to add to your program is included in the steps to create an ad hoc program.

If you find an error in your report code, you must delete the report in the Project Tree, fix your code in your source file, and run the Create Ad Hoc Report wizard again.

Create an Ad Hoc Report

To create an ad hoc report, you must first create a SAS program. Test your program in SAS before you run your program as a SAS Model Manager ad hoc report. You copy the SAS program into the **SAS Editor** of the **Create Ad Hoc Report** window.

To create an ad hoc report, follow these steps:

1. Expand the version folder .
 2. Right-click the **Reports** node and select **Reports** ⇒ **Create Ad Hoc Report**. The Create Ad Hoc Report window opens.
 3. In the **Select Model for Ad Hoc Report** table, select any number of models for the report.
 4. Either add the following code to your report program or copy the code to the SAS Editor. This code defines the report output format, such as HTML or PDF:


```
Filename mmreport catalog "sashelp.modelmgr.reportexportmacros.source";
%include mmreport;
%MM_ExportReportsBegin(reportFormat=report-format, fileName=report-name);
...
add-your-ad-hoc-code-here
...
%MM_ExportReportsEnd(reportFormat=report-format);
```

Replace *report-format* in the %MM_ExportReportsBegin macro with one of the following values: HTML, PDF, RTF, or Excel.

Replace *report-name* with the name of your ad hoc report. The following characters cannot be used in *report-name*: @, \, /, *, %, #, &, \$, (,), !, ?, <, >, ^, +, \.
 5. Copy your SAS program to the SAS editor. Make sure that your report program is enclosed by the SAS code that defines the report output format. You can select **Macro Variables** tab to view a list of the Model Manager macro variables that can be accessed by your program.
 6. Type the report name and a description for the report in the **Report Properties** table.
 7. Click **OK**. SAS Model Manager runs the report and creates a node under the **Reports** node using the name you provided in the **Report Properties** table. The new node contains the output files that were created by running the report.
- A message box confirms that the report either completed successfully or failed. If the report failed, click **Details** to review the errors in the SAS log.

Example Ad Hoc Report

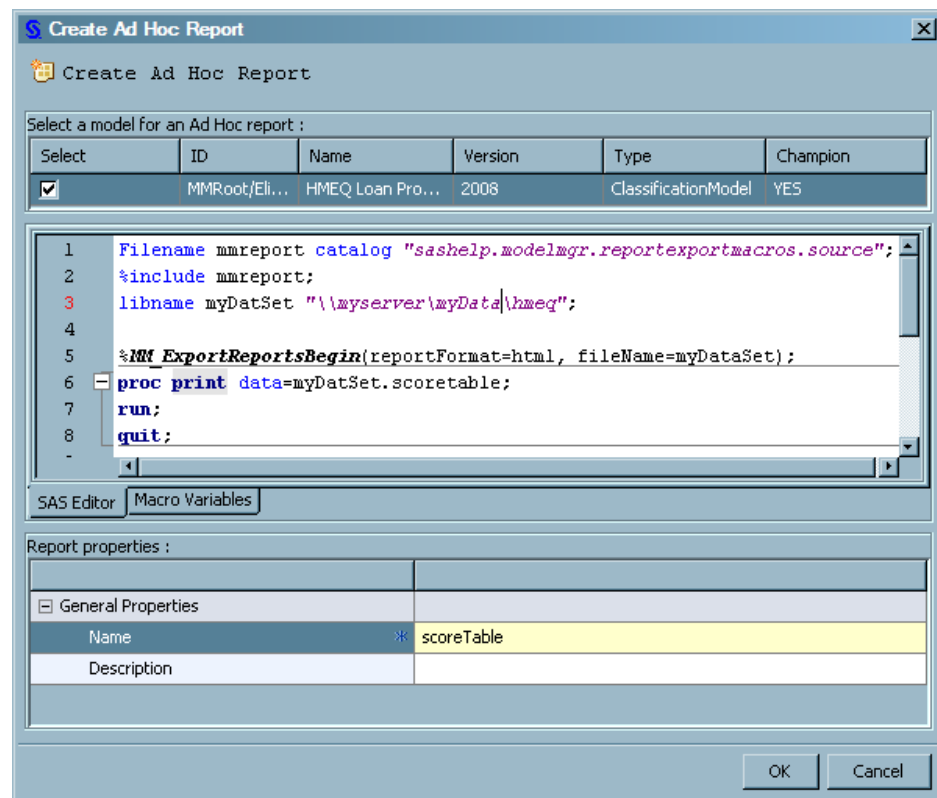
The following example code lists the score results in an HTML output format:

```
Filename mmreport catalog "sashelp.modelmgr.reportexportmacros.source";
%include mmreport;
libname myDataSet "\\myserver\myDataSets\hmeq";

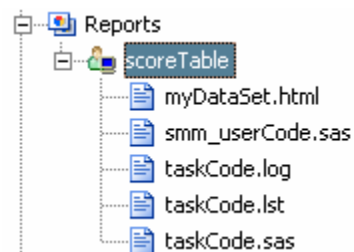
%MM_ExportReportsBegin(reportFormat=html, fileName=myDataSet);
proc print data=myDataSet.scoretable;
run;
quit;

%MM_ExportReportsEnd(reportFormat=html);
```

When you include this program in the Create Ad Hoc window, it looks like this:



After you click OK, SAS Model Manager creates the report and places the report output under the Reports folder in the Project Tree:



The following HTML output displays rows of 5000 rows of output:

Obs	_WARN_	CustID	Score
1			0.71574
2			0.54839
3			0.71574
4			0.71574
5			0.71574
6			0.06196

User-Defined Reports

Overview of User-Defined Reports

User-defined reports require three files in the middle-tier server user templates directory \server-name\SAS Configuration\Config\Lev#\AnalyticsPlatform\apps\ModelManager\ext:

- the SAS program that creates the report
- a report template XML file that specifies the report requirements, such as report name and the number of required models to run the report
- a SAS program file that lists the SAS Model Manager global macro variables that are used in your report.

If these three files are in the directory when SAS Model Manager starts, the user-defined report is included as a report in the New Reports Wizard **Reports** list.

The files can be copied only to the middle-tier server by a user who has Write access to the SAS Model Manager user templates directory. The server administrator can then complete the deployment of the templates. For more information about the deployment of user templates, see *SAS Model Manager: Administrator's Guide*.

Inherent in the New Report Wizard are controls to specify the type of output that the report create, such as HTML or PDF. You can modify your report to include the SAS code so that the New Reports Wizard offers the report output controls for your report.

Create a User-Defined Report

To create your user-defined report, complete the following tasks:

1. Write your SAS program to create a report. Include in your program the code modifications to support output formats.

To format the output for a user-Defined report, wrap the following SAS code around your report code in order to use the **Select Formats** list box in the New Report Wizard. The **Select Formats** list box enables you to select a report output format of HTML, PDF, RTF, or Excel .

Replace *report-name* with the name of your user-defined report.

```
Filename mmreport catalog "sashelp.modelmgr.reportexportmacros.source";
%include mmreport;
%MM_ExportReportsBegin(fileName=report-name);
...
your-user-defined-code
...
%MM_ExportReportsEnd;
```

2. Create a SAS program that lists the macro variables that are used in your report.
3. Copy to a file on your local computer the file UserReportTemplate.xml from the user template directory \server-name\SAS Configuration\Config\Lev#\AnalyticsPlatform\apps\ModelManager\ext. Rename the template. Modify the template for your report. For more information, see [“The Report Template” on page 141](#).

4. Have your SAS Model Manager administrator store the two SAS programs and the XML report template file in the user template directory. For more information, see *SAS Model Manager: Administrator's Guide*.
5. In order for SAS Model Manager to process these files, the SAS Analytics Platform server must be restarted by your system administrator or your SAS Model Manager administrator.

The Report Template

You create a report template XML definition file to describe your user-defined report. After you create the report template, the template can be deployed by a user with Write access to the directory \\server-name\SAS Configuration\Config\Lev1\AnalyticsPlatform\apps\ModelManager\ext.

Here is the report template XML definition:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ReportTemplate
  name="report-name"
  type="UserDefinedReport"
  displayName="display-name"
  description="model-description"
>
  <Report>
    <Data datasetName="input-data-set-name"/>
    <Models expectedModelType="model-type"
      requiredNumberOfModels="1"
      level="level">
    </Models>
    <SourceCode>
      <PreCode filename="pre-code-filename.sas"/>
      <Code filename="score-code-filename.sas"/>
    </SourceCode>
    <Output format="output-format" filename="output-name"/>
  </Report>
  <Parameters>
    <Parameter name="parameter-name" value="parameter-value" />
  </Parameters>
</ReportTemplate>
```

<ReportTemplate> Element Arguments

name="report-name"

specifies the name of the report. The following characters cannot be used in *report-name*: @, \, /, *, %, #, &, \$, (,), !, ?, <, >, ^, +, \.

displayName="display-name"

specifies the name of the report that is displayed in the **Reports** list of the New Reports Wizard window.

description="model-description"

specifies a description of the report that displays at the bottom of the New Reports Wizard when the report is selected in the window.

<Report> Element Arguments

<Data datasetName="*input-data-set-name*"/>

specifies the name of a data source data set that is used for input to the report. The data set must be in the form *libref.filename*. You can use the following global macro variables as a value for *input-data-set-name* as long as the value of the macro variable is in the form of *libref.filename*:

- &_MM_InputLib
- &_MM_OutputLib
- &_MM_PerformanceLib
- &_MM_TestLib
- &_MM_TrainLib

<Models

expectedModelType="*model-type*"

requiredNumberOfModels="*number-of-models*"

level="*level*">

</Models>

specifies information about the model.

expectedModelType="*model-type*"

specifies the model type. If your model is not a classification, prediction, or segmentation model, use the value ANALYTICAL.

Valid values: ANALYTICAL, CLASSIFICATION, PREDICTION, SEGMENTATION

requiredNumberOfModels="*number-of-models*"

specifies the number of models that are processed in this report.

level="folder"

specifies where the report is to obtain a list of models. If folder is VERSION, the report creates a list of models in the version. If folder is PROJECT, the report creates a list of models from all versions in the project.

Valid values: VERSION, PROJECT

<SourceCode>

<PreCode filename="*pre-code-filename.sas*"/>

<Code filename="*report-code-filename.sas*"/>

</SourceCode>

specifies the files that are used to execute the report.

<PreCode filename="*pre-code-filename.sas*"/>

specifies the name of the SAS program that contains macro variable definitions.

<Code filename="*report-code-filename.sas*"/>

specifies the name of the SAS program that creates the report.

<Output format="*output-format*" filename="*output-report-name*"/>

specifies the output format arguments:

format="*output-format*"

specified the format of the report output.

Valid values: HTML, PDF, RTF, or Excel

filename="*output-report-name*"

specifies the name of the output report.

<Parameter> Element Argument

<Parameter name="parameter-name" value="parameter-value" />

This element is not used. It is reserved for future use.

Defining Macro Variables For a User-Defined Report

You create a SAS program to contain macro variables that must be defined to run your report. If you do not have macro variables in your report, create a SAS program file with a comment in it. This file is required to create a user-defined report. After you create this program, do the following so that the report can access the program:

1. Save the program in the directory \\server-name\SAS Configuration\Config\Lev1\AnalyticsPlatform\apps\ModelManager\ext\SASCode. If you are not authorized to write to this directory, the SAS Model Manager administrator must save this file in the directory for you.
2. In the XML file that defines the report, add the name of the program as the value of the filename= argument in the <PreCode> element. For example,


```
<PreCode filename="myMacroDefs.sas"/>
```

Here is an example program to define macro variables:

```
%let _MM_RepositoryId=SMMModelRepository;
%let _MM_User=miller;
%let _MM_Password=Rumpillstillskin3;
```

Run a User-Defined Report

To run a user-defined report, follow these steps:

1. Expand the version folder .
2. Right-click the **Reports** node and select **Reports** ⇒ **New Report Wizard**. The New Report Wizard window opens.
3. From the **Reports** list box, select a user-defined report.
4. In the **Select Model(s)** box, select the appropriate model or models for the report.
5. In the **Select Format** list box, select the type of output that you want to create. The default is **PDF**. Other options are **HTML**, **RTF**, and **Excel**.
6. In the **Report Properties** box, enter a report name or use the default report name. The default report name is in the form *report-name_DdateTtime*.
7. Click **OK**. The report runs, a report folder is created, and the output is stored in the report folder. The name of the report folder is the report name that you specified in the **Report Properties** box.

Example User-Defined Report

Overview of the Example User-Defined Report

The example user-defined report categorizes scoring values into score ranges and then graphs the results. The program name is Score Range Report. The following files are required to create this report:

- The SAS report program is the file ScoreRange.sas
- The SAS program file that contains macro variables is ScoreRangeMacro.sas
- The report template XML file is ScoreRangeTemplate.xml

The SAS Report Program

Here is the SAS code for a user-defined report to categorize score codes:

```
filename mmreport catalog "sashelp.modelmgr.reportexportmacros.source";
%include mmreport;

%MM_ExportReportsBegin(fileName=scoreRange);

options NOMprint;
%let _MM_PosteriorVar=P_1;

proc format;
  value score
    low - 400 = '400 and Below'
    401 - 450 = '401 - 450'
    451 - 500 = '451 - 500'
    501 - 550 = '501 - 550'
    551 - 600 = '551 - 600'
    601 - 650 = '601 - 650'
    651 - 700 = '651 - 700'
    701 - 750 = '701 - 750'
    751 - 800 = '751 - 800'
    801 - high= '801 and Above';
run;
quit;

%Macro scoreRange();

  %if &_MM_ScoreCodeType = %str(SAS Program) %then
    %do;
      %let _MM_OutputDS=work.scoreresult;
      %inc &_MM_Score;
    %end;
  %else
    %do;
      data work.scoreresult;
        set &_MM_InputDS;
        %inc &_MM_Score;
      run;
    %end;

  data work.scoreresult2;
    set work.scoreresult;
    keep score;
    if &_MM_PosteriorVar = . then delete;
    score = int (((1-&_MM_PosteriorVar) * 480) + 350 + 0.5);
  run;

proc freq data=work.scoreresult2;
```

```

table score/out=scoresummary;
format score score.;
title 'Credit Score Range';
quit;

proc gchart data=work.scoresummary;
  hbar score / sumvar=count discrete;
  title 'Credit Score Range';
run;
quit;
%Mend scoreRange;

/* Reporting section */

ods listing close;

%getModelInfo(0);
%scoreRange();
%closeLibsAndFiles();

%MM_ExportReportEnd;

```

The SAS Program File for Macro Variables

The file ScoreRangeMacro.sas contains only a comment in it as macro variables are not used in the report code:

```
/* ScoreRangeMacro.sas empty file */
```

The Report Template XML File

Here is the report template XML file for the user-defined Score Range report:

```

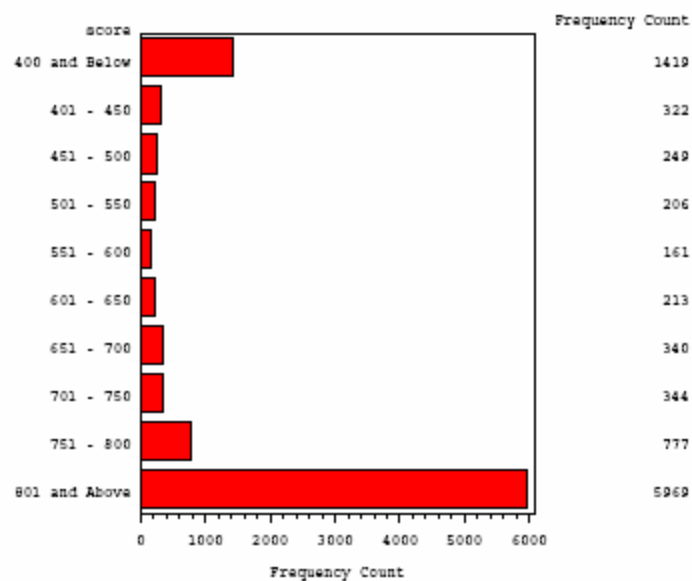
<?xml version="1.0" encoding="UTF-8" ?>
<ReportTemplate
  name="Score Range Report"
  type="UserDefinedReport"
  displayName="Score Range Report"
  description="Score Range Report"
>
  <Report>
    <Data datasetName="" />
    <Models expectedModelType="ANALYTICAL"
      requiredNumberOfModels="1" level="VERSION">
    </Models>
    <SourceCode>
      <PreCode filename="ScoreRangeMacro.sas" />
      <Code filename="ScoreRange.sas" />
    </SourceCode>
    <Output format="PDF" filename="ScoreRange" />
  </Report>
  <Parameters>
  </Parameters>
</ReportTemplate>

```

The Score Range Report Output

The following Credit Score Range graph is one of the output pages in the PDF report output:

Credit Score Range



Part 4

Deploying and Delivering Champion Models

<i>Chapter 11</i>	
Deploying Models	149
<i>Chapter 12</i>	
Delivering Models	157
<i>Chapter 13</i>	
Replacing a Champion Model	177

Chapter 11

Deploying Models

Overview of Deploying Models	149
Champion Models	150
About Champion Models	150
Requirements For a Champion Model	150
Set a Champion Model	151
Clear a Champion Model	151
Freezing Models	152
About Freezing Models	152
Freeze a Version	153
Unfreeze a Version	153
The Default Version for a Project	154
About the Default Version	154
Set a Default Version	154
Clear a Default Version	155

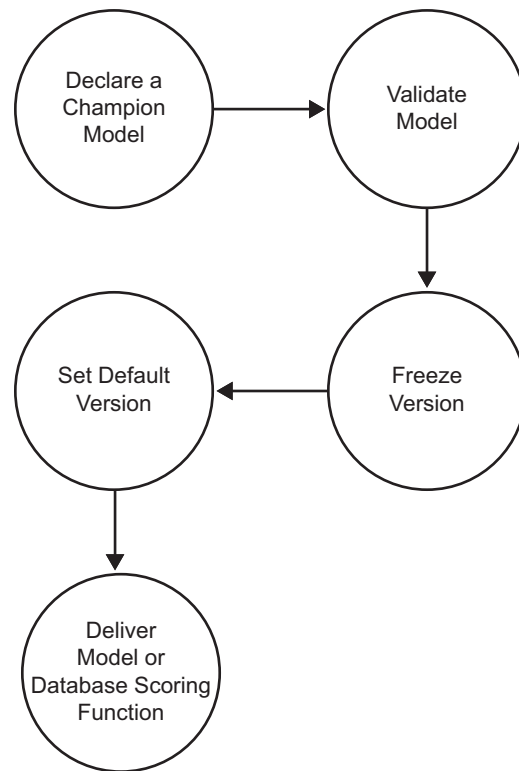
Overview of Deploying Models

The goal of a modeling project is to identify a champion model that a scoring application uses to predict an outcome. SAS Model Manager provides tools to evaluate candidate models, declare champion models, and inform your scoring officer that a predictive model is ready for validation or production.

To deploy a model from SAS Model Manager, you might use the following scenario:

1. Identify the model that outperforms other candidate models and declare this model to be a champion model.
2. Freeze the model version to prevent changes to the model.
3. Test and validate the model before you declare the model ready for production.
4. Set the model version as the default version and update your life cycle milestones.
5. Publish or export the model so that you can deploy the champion model to a production environment.

The following figure illustrates activities that might occur before a model is deployed.

Figure 11.1 Deploying Analytical Models

Champion Models

About Champion Models

To identify the champion model, you can evaluate the structure, performance, and resiliency of candidate models. You select the champion model from the models in a version. When a champion model is ready for production scoring, you select the version that contains this champion model as the default version for a project. Then the champion model in the default version becomes the default champion model for the project. When you export the project champion model, SAS Model Manager deploys the default champion model.

Note: SAS Model Manager cannot set a PMML model as the champion model.

These are the tasks that you perform to use champion models:

- [“Set a Champion Model” on page 151](#)
- [“Clear a Champion Model” on page 151](#)

Requirements For a Champion Model

Before you identify a model as the champion, perform the following tasks:

- Create a version for your project, and register at least one model.

- Verify that the model is active. If the model expiration date has passed, then you cannot set the model as a champion model.

Note: An authorized user can reset the expiration date to a later date so that you can set the champion model.

- Complete the required life cycle milestones that precede the milestone task of setting the champion model under a version.



You might use the following criteria to identify a champion model:

- model comparison reports that validate and assess the candidate models
- business decision rules, such as using a decision tree model because of difficulty interpreting results from a neural network model even when the neural network model outperforms the decision tree model
- regulatory requirements, such as when the champion model should exclude certain specific attributes (age or race).

You can register a challenger model in SAS Model Manager specifically for the purpose of comparison with the champion model. For example, your champion model for a production environment might omit restricted attributes during operational scoring because of regulatory requirements. You can use a challenger model that includes the restricted attributes in the development environment to evaluate its prediction power against the prediction of the champion model. Then you can determine the amount of predictive power that is lost due to the regulatory requirements.

Set a Champion Model

To set a champion model, follow these steps:


1. Expand the **Models** folder under the version folder .
2. Right-click the model that you want to use as the champion model and select **Set Champion Model** from the pop-up menu. A dialog box opens.
3. Click **Yes** to confirm.
4. Verify that the  icon appears beside the champion model.
5. Select the version folder to examine its properties. The value for **Modification Date** is today's date. The value for the **Champion Model Id** is the champion model's UUID.

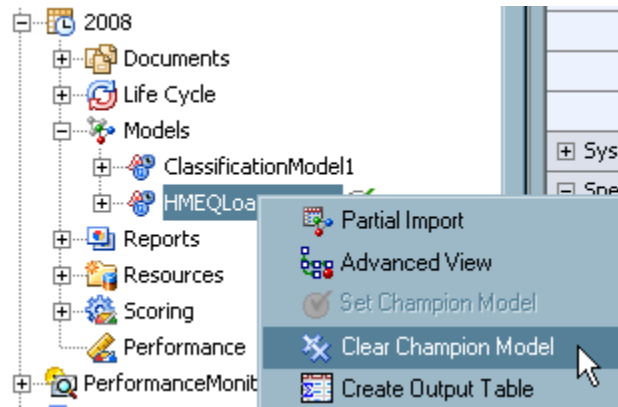
Note: To document the reasons or assumptions for your selection of the champion model, use the version **Notes** tab. SAS Model Manager automatically annotates the History tab.

6. Update the specific properties for the appropriate milestone task in the **Life Cycle** node. Specify that **Status** for selecting the champion model is completed.

Clear a Champion Model

To clear a champion model, follow these steps:

1. Expand the **Models** folder under the version folder .
2. Right-click the champion model and select **Clear Champion Model** from the pop-up menu. A message box opens.



3. Click **Yes** to confirm. If the champion model is located in the default version, then SAS Model Manager also clears the default version.

Note: If the version is frozen, then you cannot clear the champion model unless you are a SAS Model Manager administrator.

4. Select the version folder to examine its properties. Verify that the value for **Modification Date** is today's date. The value for the **Champion Model Id** has been cleared.
5. Update the specific properties for the appropriate milestone task in the **Life Cycle** node. Change **Status** for champion model to a value that is not completed, such as **Started**.

Freezing Models

About Freezing Models

SAS Model Manager administrators can freeze a project version to prevent users from modifying some properties and files for the version's models. A version is frozen when the champion model in a version folder is approved for production or is pending approval. After a version is frozen, the **Models** folder is locked so that SAS Model Manager advanced users cannot perform the following tasks:

- add or delete models
- modify version or model properties
- add, delete, or modify model objects
- change the champion model

SAS Model Manager administrators remain authorized to perform these activities. If the champion model is not deployed to an operational environment, then a SAS Model Manager administrator can unfreeze a frozen version so that users can change the models. SAS Model Manager advanced users can still modify the **Documents**, **Reports**, **Resources**, and **Scoring** folders after a version is frozen.

When the champion model has been used in production scoring and you must change the contents of a frozen default version, unfreeze the default version. However, use caution modifying the version content. If the model UUID and revision number for the score code in production scoring environments is always recorded, then you can modify a version even after the version is deployed to production environment.

These are the tasks that you perform to control access to project version:

- “Freeze a Version” on page 153
- “Unfreeze a Version” on page 153


If you attempt to delete a project that contains a frozen version, SAS Model Manager prompts you to verify that you want to delete the project.

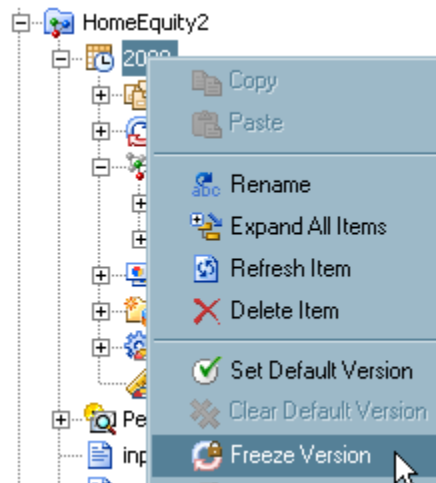
Freeze a Version


Before a version is frozen, you must complete and sign off on the required life cycle milestones that precede this activity. After the version is frozen, users cannot modify any properties for the models in the version folder. Projects cannot be deleted if a version is frozen.

Note: You must be a SAS Model Manager administrator to freeze and unfreeze a version.

To freeze a version, follow these steps:

1. Right-click the version folder .
2. Select **Freeze Version** from the pop-up menu.




3. Verify that the  icon appears beside the version folder.
4. Select the version folder to examine its properties. The value for **Frozen Date** is today's date.

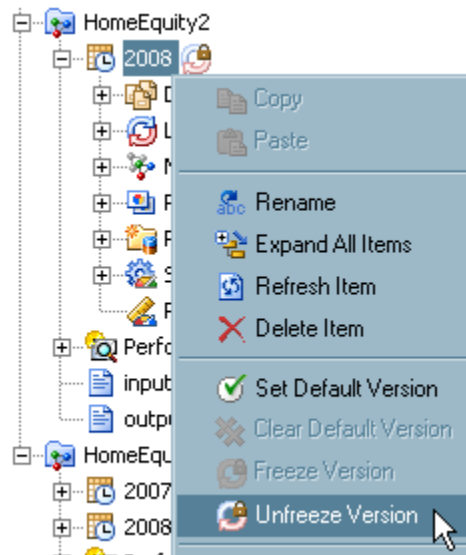
Note: To document the reasons or assumptions for freezing the version, use the version **Notes** tab. SAS Model Manager automatically annotates the History tab.


Unfreeze a Version

If changes to a model are required after the version is frozen, then a SAS Model Manager administrator has the ability to unfreeze the version.

To unfreeze a version, follow these steps:

1. Right-click the version folder  that is frozen.
2. Select **Unfreeze Version** from the pop-up menu.



3. Verify that the  icon does not appear beside the version folder.
4. Select the version folder to examine its properties. The value for **Frozen Date** is missing.

Note: Changes that are made to a model after a version is frozen can invalidate the results of life cycle milestones that you completed to deploy the model.

The Default Version for a Project

About the Default Version

A default version is assigned before you can export a project champion model. After the default champion model for the project is identified, you set the version with this champion model as the default version for the project. Then the champion model in the default version is assigned as the default champion model for the project. When the project champion model is exported, SAS Model Manager deploys the default champion model.

Selecting a new default version automatically disables the previous default version. This guarantees that the score code for a default version of the project is unique for a deployed model. You do not have to modify the application if the champion model changes since the application uses the project's input and output data.

These are the tasks that you perform to assign a default version for a project.


- “Set a Default Version” on page 154
- “Clear a Default Version” on page 155

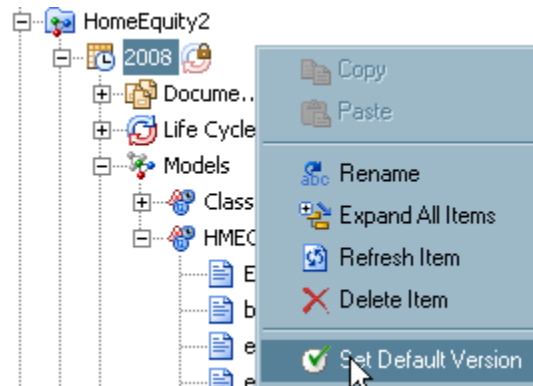
See Also


“Exporting Models” on page 162

Set a Default Version

To set the default version for a project, follow these steps:



1. Right-click the version folder .
2. Select **Set Default Version** from the pop-up menu.



3. Click **Yes** to confirm.
 4. Verify that the  icon appears beside the version folder.
 5. Select the project folder to examine its properties. The value for **Modification Date** is today's date. The value for the **Default Version** is the name of the version folder.
- Note:* To document the reasons or assumptions for your selection of the default version, use the project **Notes** tab. SAS Model Manager automatically annotates the History tab.
6. Update the specific properties for the appropriate milestone task in the **Life Cycle** node.

Clear a Default Version

To clear a default version, follow these steps:

1. Right-click the default version folder .
2. Select **Clear Default Version** from the pop-up menu.
3. Click **Yes** to confirm.
4. Verify that the  icon does not appear beside the version folder.
5. Select the project folder to examine its properties. The value for **Modification Date** is today's date. The value for the **Default Version** has been cleared.
6. Update the specific properties for the appropriate milestone task in the **Life Cycle** node.

Chapter 12

Delivering Models

Overview of Model Delivery	157
Publishing Models	158
About Publishing Models	158
Publish a Model to a Channel	159
Extract a Published Model	161
Exporting Models	162
About Exporting Models	162
Export a Model	162
Export a Project Champion Model	163
Verify the Model Export	164
Publish Teradata Scoring Functions	164
What Is a Teradata Scoring Function?	164
Teradata Scoring Function Process Flow	166
Prerequisites for Publishing a Teradata Scoring Function	168
Publish Teradata Scoring Function Field Descriptions	168
How to Publish a Teradata Scoring Function	171
Log Messages	174
Scoring Function Metadata Tables	175

Overview of Model Delivery

SAS Model Manager provides a comprehensive publishing environment for model delivery that supports sharing of life cycle and performance data. SAS Model Manager publishes models to different channels and exports champion models to a SAS Metadata Repository. SAS Model Manager can also publish classification and prediction models to the Teradata Enterprise Data Warehouse. Application software, such as SAS Data Integration Studio or SAS Enterprise Guide, enables you to access models through the SAS Metadata Server and to submit on-demand and batch scoring jobs.

SAS Model Manager publishes models to defined publication channels. Authorized users that subscribe to the channel can choose to receive e-mail notifications when updated models are ready to deploy to testing or production scoring servers and are published to a publication channel. Then you can extract and validate the scoring logic, deploy champion models to a production environment, and monitor the performance of your models.

Publishing Models

About Publishing Models

SAS Model Manager uses SAS Publishing Framework to publish models to defined channels and notifies subscribers of the publication channel when the models are delivered. You can publish models from the organization, project, version, or model folder in the Project Tree.

Note: SAS Model Manager cannot publish PMML models.

SAS Model Manager creates a SAS Package file (SPK) for the model in a publication channel. Users that subscribe to the channel can choose to receive e-mail that includes the SAS package as an attachment. To automate model deployment, the publication channel must publish models as archive (binary .spk) files to a persistent store location. The archive persistent store is specified as a physical file location, an FTP server, or an HTTP server.

The **Report** attribute for a file element in a model template indicates whether SAS Model Manager includes a file in the SAS package. You use the SAS Package Reader or a file archiver and compression utility, such as WinZip, to view the contents of the SPK file. SAS Model Manager provides SAS macro programs to extract published models and deploy the models on testing and production scoring servers.

By default, the SAS package with the published model includes the following files:

Filename	Description
inputvar.xml	input variables for the model
outputvar.xml	output variables for the model
targetvar.xml	the target variable for the model
score.sas	SAS code to generate the model
smmpostcode.sas	SAS code to map model variables to project variables
sas001.ref	URL address of the inputvar.xml
sas002.ref	URL address of the outputvar.xml
sas003.ref	URL address of the code.sas file

The SAS Package might contain additional files, depending on the number of file elements in the model template that have a **Report** attribute.

Note: The ref file contains the URL address for a folder location in the Project Tree, such as `http://MMServer:8080/SASContentServer/repository/default/ModelManager/MMRoot/organizational oolder/project/version/Models/model_name/code.sas`.

These are the tasks that you perform to use a published model:

- “Publish a Model to a Channel ” on page 159
- “Extract a Published Model” on page 161

Publish a Model to a Channel

To publish a model to a channel, follow these steps.

1. Right-click project, version, or models folder that contains the model that you want to publish and select **Publish** from the pop-up menu. The Publish using SAS Channel window opens.

Note: To publish a model, a project must contain a version folder with at least one model.

Channel: MMChannel

Description:

Subject: Model Manager

Subscribers:

Select Entries to Publish:

Select	ID	Name	Ver...	Type	Ch...
<input type="radio"/>	MMRoo...	Classific...	2008	Classific...	NO
<input type="radio"/>	MMRoo...	HMEQL...	2008	Classific...	YES

Back Next Finish Cancel

2. Select a publication channel from the **Channel** list box.

Note: The channel values for **Description**, **Subject**, and **Subscribers** are defined in the SAS Metadata Repository with SAS Management Console.

3. Select the model to publish in the **Select Entries to Publish** table. SAS Model Manager lists all of the models in the version folder. To view the entire folder name, expand the **ID** column heading.
4. Click **Next**. The next window opens.

Publish using SAS Channel

Publish using SAS Channel

Message Subject:

Notes:

Add User-Defined Property:

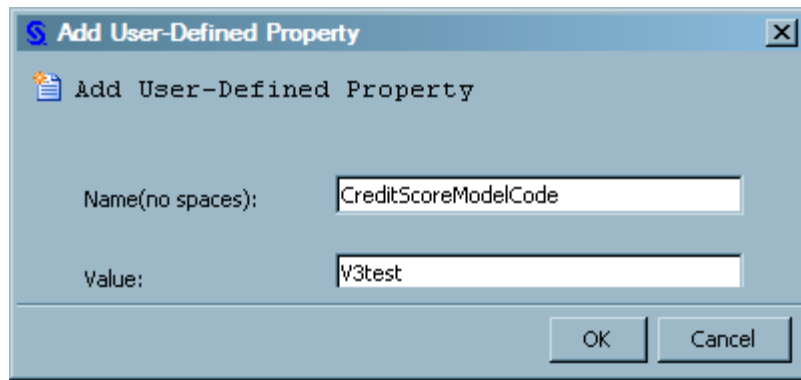
Name	Value
<input type="text"/>	<input type="text"/>

Back Next **Finish** Cancel

5. (Optional) Specify a subject line for the e-mail message in the **Message Subject** box. By default, SAS Model Manager uses the value that is defined in the publication channel. If you omit the subject line then the name of the published model is used.
6. (Optional) Use the **Notes** box to include information about the model that might be useful to other users involved with the project.
7. (Optional) Create user-defined properties that you can use to filter the notifications sent to subscribers of the publication channel. SAS Model Manager embeds user-defined properties in the SAS package file.

To create a user-defined property, complete these steps:

- a. Right-click in the **Add User-Defined Property** table and select **Add User-Defined Property**.
- b. Specify the property name and its value in the Add User-Define Property window. For example, specify **CreditScoreModelCode** for the property name and **v3test** for its value.



c. Click **OK**.

Note: You can also add user-defined properties to the model properties in the Project Tree. Then the property is already defined each time you publish the model.

8. Click **Finish**. A window opens that provides information about whether SAS Model Manager successfully published the model. Click **Details** to display a log of the publication process and any messages.
9. Click **OK**.
10. Update the specific properties for the appropriate milestone task in the **Life Cycle** node.
11. Select the project. Right-click in the Properties view and select **Add User-Defined Property**. In the **Name** field, type in a property name that depicts the channel name that was used to export the project, such as `PublishedToChannel`. In the **Value** field, enter the channel that was used to export the project. If you extract a model using the `%MM_GetModels()` macro to create performance reports, you must know the channel that was used to publish the model.

Extract a Published Model

When you publish a model, a SAS package is sent to the publication channel. The SAS package contains the model input, output, SAS code, and its properties. You can submit a SAS DATA step program that calls the SAS Publish API (Application Programming Interface) to extract and deploy the model to a testing or scoring server. SAS Model Manager also provides a SAS macro program, called `%MM_GetModels`, that extracts the SAS code and metadata to score the model. Typically, extracted files are placed on a local drive of the scoring server that is used to deploy the published model.

SAS Model Manager uses the extracted files to generate reports that monitor and evaluate model performance. Changes in model performance might indicate a need to adjust the model or identify a new champion model. You can create the reports either by using the Define Performance Task Wizard from the Project Tree or by submitting SAS macro programs. The `%MM_GetModels` macro creates the data tables that are required to run the performance reports. The macro also creates and manages the data sets that provide metadata to track the current champion models for each project, the extracted models from channels, and the archived models. For information, see [“Extracting the Champion Model from a Channel” on page 222](#).

Exporting Models

About Exporting Models

SAS Model Manager exports a model by creating a MiningResult object in a SAS Metadata Repository. You can use the model information in the MiningResult object to set up a scoring environment. A scoring application might use SAS Data Integration Studio or SAS Enterprise Guide to access the metadata and run a batch job or stored process that executes the score code. SAS Real-Time Decision Manager can also read the metadata and use it in that process environment. If you export a project champion model, then, with proper configuration, the scoring application always uses the most current the champion model.

Note: SAS Model Manager cannot export PMML models and models whose **Score Code Type** model property is set to **SAS Program**.

SAS Model Manager uses the SAS Folder view to export the model to any folder that is accessible to the user. These folders include all folders in the SAS Foundation repository and folders in custom repositories that are created in SAS Management Console to reflect the structure of your business organization.

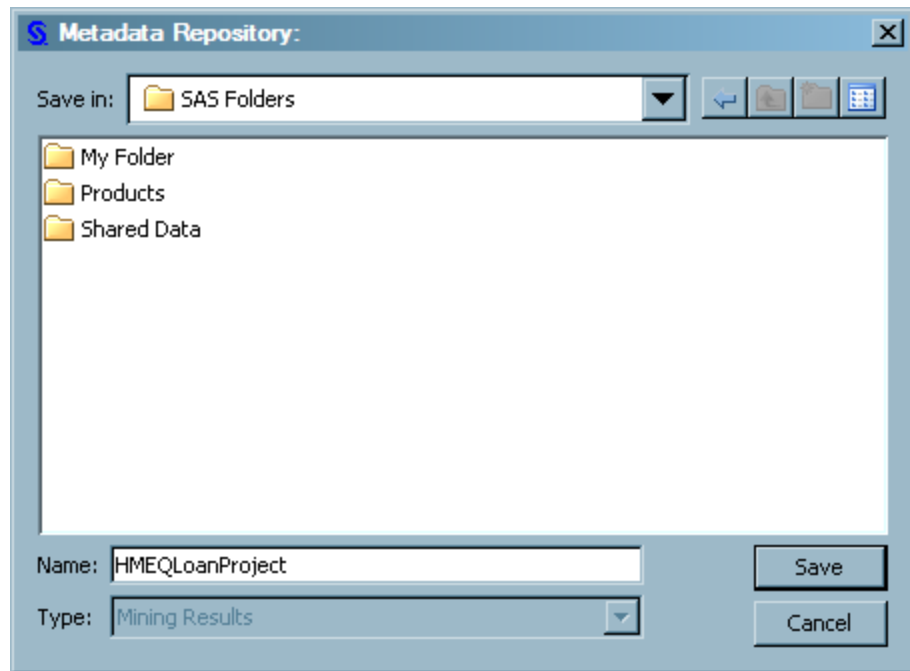
These are the tasks that you perform to export models:

- [“Export a Model” on page 162](#)
- [“Export a Project Champion Model” on page 163](#)
- [“Verify the Model Export” on page 164](#)

Export a Model

To export a model, follow these steps:

1. Right-click the model that you want to export and select **Export Model** ⇒ **Export Model into SAS Metadata Repository** from the pop-up menu. The Metadata Repository window opens. SAS Model Manager displays metadata folders.



2. Select the folder to where you want to export the model.
3. Enter the name for the model and click **Save**.
4. Click **Save**. If a MiningResult object is in the repository that has the same name, then you are asked whether to overwrite the metadata for this stored object.

Note: Do not overwrite an existing MiningResult object unless you are certain that the model is from the same project in SAS Model Manager.

Note: If the score code for the model changes, then export the model again to ensure that your score application uses the current scoring code.

Export a Project Champion Model

To export the champion model for a project, you must have already assigned the default version. SAS Model Manager examines the project and always exports the champion model in the default version. When the default version for a project changes and you export the model again at the project level, the scoring application automatically uses the latest score code.

Note: SAS Model Manager cannot export PMML models and models whose **Score Code Type** model property is set to **SAS Program**.

To export the champion model for a project, follow these steps:

1. Verify that project you want to export has a default version assigned. Select the project folder to examine its properties. The **Default Version** property contains the name of the default version.
2. Right-click the project name and select **Export Project Champion Model** from the pop-up menu. The Metadata Repository window opens. SAS Model Manager displays the metadata folders.
3. Select a folder to where to you want to export the model.
4. Click **OK**. If a MiningResult object is in the repository that has the same name, then you are asked whether to overwrite the metadata for this stored object.

Note: Do not overwrite an existing MiningResult object unless you are certain that the project is from the same project in SAS Model Manager.

Verify the Model Export

To verify that SAS Model Manager successfully created the MiningResult object in the metadata repository for an exported model, use SAS Management Console. To view the contents of the exported model or project, you can use SAS Data Integration Studio. You can also use SAS Management Console to export the MiningResult object to a SAS package.

To view a MiningResult object in the metadata repository, follow these steps:

1. In SAS Management Console, open a SAS Model Manager metadata profile that connects to the SAS Metadata Server.
2. From the SAS Management Console **Folders** tab, expand the folders to where you exported the model. When you select the folder, the right pane lists the MiningResult objects for the exported models.
3. Right-click the MiningResult object that has the name of exported model or project and select **Properties** from the pop-up menu. The Properties window opens.
4. Examine the **Keywords** box on the **General** tab to verify that the MiningResult object contains the UUID of the exported project or model.

Note: You can use the UUID to conduct filtered searches and query the exported models. For more information, see [Query Folders, Projects, and Versions on page 235](#).

5. Examine the metadata on the **Advanced** tab to determine when the MiningResult object was created or most recently updated.
6. Click **OK**.

Publish Teradata Scoring Functions

What Is a Teradata Scoring Function?

The **Publish Teradata Scoring Function** of SAS Model Manager enables you to publish classification and prediction model types to the Teradata Enterprise Data Warehouse (EDW). When you publish a Teradata scoring function for a project, SAS Model Manager exports the project's champion model to the SAS Metadata Repository and the SAS Scoring Accelerator creates scoring functions under the default version. The scoring functions can be deployed inside Teradata based on the project's champion model score code. The scoring function is then validated automatically against a default train table to ensure that the scoring results are correct. A scoring application (for example, a call center application that calls the SAS Model Manager Java Scoring API) can then execute the scoring functions in the Teradata EDW. The scoring functions extend the Teradata SQL language and can be used on SQL statements like other Teradata functions.

The Scoring Function metadata tables are populated with information about the project and pointers to the Teradata scoring function, which enables users to review descriptions and definitions of the exported model. The Audit logs track the history of the model's usage and any changes that are made to the scoring project.

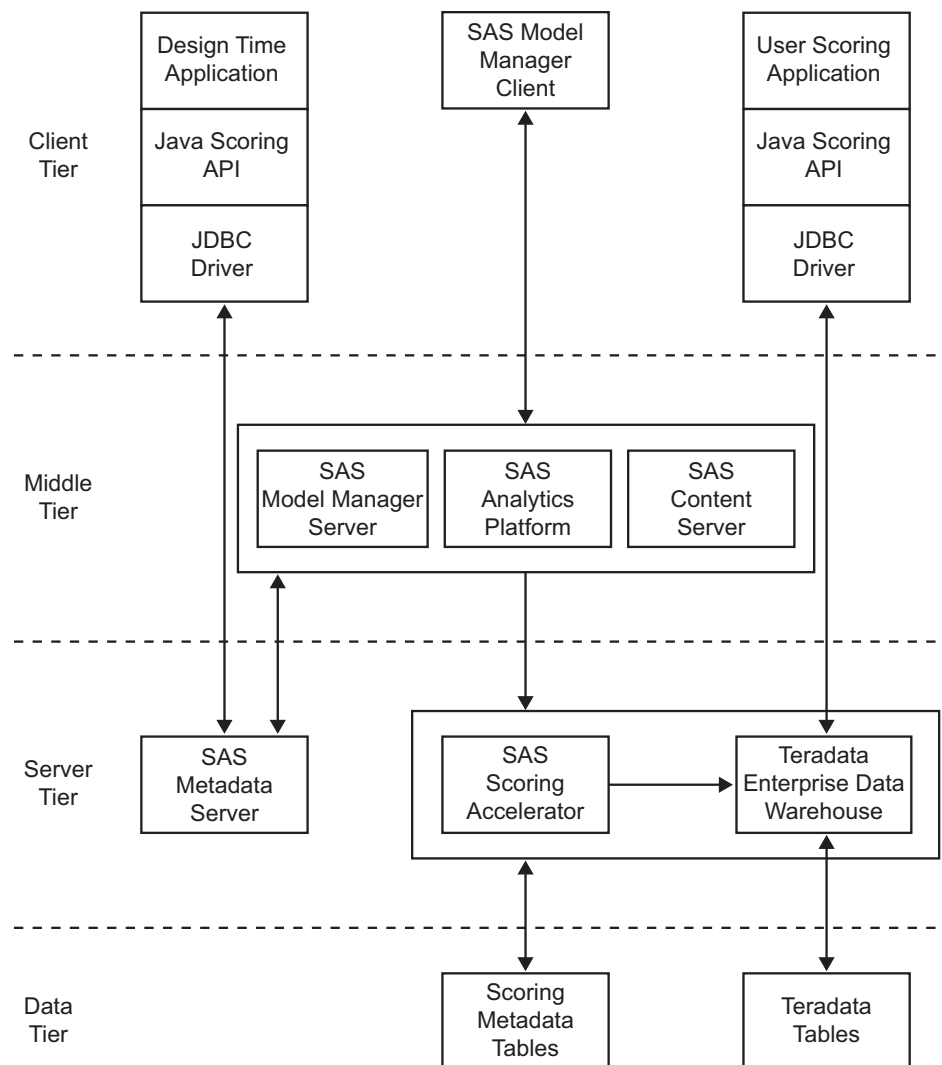
In addition to creating a scoring function, the Publish Teradata Scoring Function also creates a MiningResult metadata object which is stored in the SAS Metadata Repository. A typical use of a MiningResult object in this context is to serve input and output metadata queries made by scoring applications at the design time of application development.

For more information about the SAS Scoring Accelerator, see the *SAS Scoring Accelerator for Teradata User's Guide*.

Note: SAS Model Manager cannot publish Teradata scoring functions for PMML models. A model is supported when the score code is a SAS DATA step fragment and created by SAS Enterprise Miner. If the DATA step score code is not created by SAS Enterprise Miner, it is recommended that you select **Validate scoring results** when publishing a Teradata scoring function.

Here is a diagram that represents the relationship between SAS Model Manager 2.2 and the SAS Model Manager In-Database Scoring for Teradata.

Figure 12.1 The Relationship between SAS Model Manager 2.2 and the SAS Model Manager In-Database Scoring for Teradata



Here are descriptions of the figure's components.

SAS Model Manager Client

The SAS Model Manager Client handles communication to and from SAS Model Manager. You use the SAS Model Manager client to create projects and versions, import models, connect with data sources, validate models, run modeling reports, run scoring tasks, set project status, declare the champion model, and run performance tests.

Design Time Application

This is an application that is used to set parameters. It is called by the Java Scoring API.

User Scoring Application

This is a custom scoring application that is used to set parameters. It is called by the Java Scoring API.

Java Scoring API

The Java Scoring API is used by SAS Model Manager to produce scoring results.

JDBC Driver

The JDBC driver enables a user to compile and run the Java Scoring API to produce scoring results.

SAS Model Manager Server

The SAS Model Manager Server orchestrates the communication and movement of data between all servers and components in the SAS Model Manager operational environment.

SAS Analytics Platform

The SAS Analytic Platform provides a common application framework for analytic applications such as SAS Enterprise Miner and SAS Forecast Studio. Models that are developed using the SAS Analytics Platform can be exported to the SAS Metadata Repository or a local computer, and imported to SAS Model Manager.

SAS Content Server

The SAS Model Manager model repository and SAS Model Manager window tree configuration data and metadata are stored in the SAS Content Server. Communication between SAS Model Manager and the SAS Content Server uses the WebDAV communication protocol.

SAS Metadata Server

SAS Model Manager retrieves metadata about models from the SAS Metadata Server.

SAS Scoring Accelerator

The SAS Scoring Accelerator creates scoring functions that can be deployed inside Teradata based on the project's champion model score code.

Teradata Enterprise Data Warehouse

These relational databases serve as output data sources for SAS Model Manager.

Scoring Metadata Tables

These tables contain metadata that serves as data sources in SAS Model Manager.

Teradata Tables

These tables in relational databases serve as data sources for a scoring application.

Teradata Scoring Function Process Flow

This is an example of the process flow to publish a Teradata scoring function. For more information, see [“How to Publish a Teradata Scoring Function” on page 171](#).

1. From SAS Model Manager users select the Publish Teradata Scoring Function for the project that contains the champion model that they want to publish to Teradata. For more information, see [“How to Publish a Teradata Scoring Function” on page 171](#).

2. After the user completes all the required information about the Publish Teradata Scoring Function, SAS Model Manager establishes a JDBC connection to the Teradata database using the credentials that were entered. The user-defined part of the scoring function name is validated against the target Teradata database. If the user-defined part of the function name is not unique, an error message is displayed.
 3. If the scoring function name is validated successfully, the SAS Model Manager middle-tier server creates a MiningResult metadata object based on the champion model and exports that MiningResult metadata object to the folder that was specified in the Publish Teradata Scoring Function window.
 4. The SAS Model Manager middle-tier server then makes the user-defined formats accessible to the SAS Workspace Server. The format catalog is stored in the corresponding **Resources** folder.
 5. The MM_PUBLISH2TERADATA macro is called, which performs the following tasks:
 - calls the #MM_TRANSFORM4TD macro that creates a metadata XML file. This XML file is used by the %INDTD_PUBLISH_MODEL macro.
 - calls the %INDTD_PUBLISH_MODEL macro, which transforms user-defined formats to C code and generates C code from the DATA step score code. The C code is used to create the Teradata scoring function.
 - validates scoring results by performing the following tasks:
 - creates a benchmark scoring result with the SAS Workspace Server using DATA step score code.
 - copies a scoring input data set to create an equivalent Teradata table.
- Note:* The default train table that is specified in the properties of the version that contains the champion model is used as the scoring input data set during validation.
- scores the model with the new scoring function using the new scoring table.
 - compares scoring results.
 6. The middle-tier server parses the SAS Workspace Server logs to extract the return code.
 7. The middle-tier server updates the scoring function metadata tables (for example, table project_metadata). For more information see, [“Scoring Function Metadata Tables” on page 175](#).
 8. The middle-tier server then creates a History entry in the SAS Model Manager project history.
- Note:* A history entry is always created whether the publishing job is completed successfully or not.
9. The middle-tier server updates the project user-defined properties with the Function Name that was entered in the Publish Teradata Scoring Function window.
- Note:* The user-defined part of the function name is used as the default Teradata scoring function name in future scoring function publishing from the same SAS Model Manager project.
10. A message indicates that the scoring function has been successfully created and that the scoring results have been successfully validated.

Note: If the publishing job fails, an error message appears. Users can view the workspace logs that are accessible from the message box. If the scoring function is

not published successfully, then the previous scoring function for the same project is used in subsequent scoring, based on the SAS Model Manager Java Scoring API.

Prerequisites for Publishing a Teradata Scoring Function

The following prerequisites must be completed before users can publish a Teradata scoring function:

- The user must have the proper authorization to publish approved models from SAS Model Manager 2.2 to Teradata 12 for SAS In-Database scoring.
- The default scoring version for the project and the champion model for the default version must be set.
- A predictive (classification or prediction) model must have been selected for production scoring deployment via SAS Model Manager 2.2.
- A SAS metadata folder must have been created for models that are to be used for Teradata SAS In-Database scoring. The folder name is created in SAS Management Console.

Note: the SAS metadata folder is created via SAS Management Console.

- A project user-defined property **DbmsTable** must have been defined for the default version of the SAS Model Manager project from which to publish the scoring function.
- The Teradata JDBC driver must be accessible from the SAS Model Manager middle-tier server.
- A Teradata EDW must have been configured to install Teradata scoring functions.
- (Optional) The user might have contacted the scoring application development team about what scoring function name should be used for the SAS Model Manager project that contains the approved model.

Publish Teradata Scoring Function Field Descriptions

Here is a list of the field names and descriptions for the Publish Teradata Scoring Function window.

Publish Teradata Scoring Function

Publish Teradata Scoring Function for HomeEquity

SAS Metadata Location: Browse...

Function Name: Y090122***_

Teradata Server:

Database:

User ID:

Password:

Options

☒ Validate scoring results ☐ Keep scoring function if validation fails

☒ Protected mode Sample Size:

☐ Display detailed log messages

OK Cancel

SAS Metadata Location

specifies a folder location in the SAS Metadata Repository.

Function Name

specifies the name of the Teradata Scoring Function, which includes a prefix and a user-defined value. The prefix has a length of 11 characters and in the format of **Yyyymmddnnn_**.

- **Y** is a literal character and is fixed for all prefixes.
- **yy** is the two digit year.
- **mm** is the month that ranges from 01 to 12.
- **dd** is the day that ranges from 01 to 31.
- **nnn** is a counter that increments by one each time that a scoring function is completed successfully. The value can range from 001 to 999.
- **_** is the underscore that ends the prefix.

The **yyymmdd** value in the prefix is the GMT timestamp that identifies the date when you selected the **Publish Teradata Scoring Function** menu option. The **nnn** is a counter that increments by one each time the scoring function is successfully completed. An example of a function name is **Y081107001_user_defined_value**. The following are the naming convention requirements:

- The user-defined value is case-insensitive. The maximum length is 19 alphanumeric characters, and no spaces are allowed. An underscore is the only special character that can be included in the function name.

UNIX Specifics

The user-defined portion of the function name in an AIX environment has a maximum length of 16 alphanumeric characters.

- The user-defined value portion of the scoring function name is used by default for subsequent use of the Publish Teradata Scoring function from the same project. Only the user-defined value of the scoring function name can be modified.

Teradata Server

specifies the name of the server where the Teradata Enterprise Data Warehouse (EDW) resides.

Database

specifies the name of the database.

User ID

specifies the user identification that is required to access the database.

Password

specifies the password that is associated with the **User ID** property and that is required to access the database.

Options**Validate scoring results**

specifies to validate the scoring results when publishing the Teradata scoring function. This option creates a benchmark scoring result on the SAS Workspace Server using the DATA step score code. The scoring input data set is used to create an equivalent Teradata table. Scoring is performed using the new scoring function and Teradata table. The scoring results are then compared.

Note: The default training table is used as the scoring input data set during validation.

Keep scoring function if validation fails

specifies to save the scoring function if the validation of the scoring results fails. This option is enabled by default. You might find that keeping the scoring function is useful for debugging if the scoring function validation fails.

Protected mode

specifies the mode of operation to use for the Publish Teradata Scoring function. There are two modes of operation, protected and unprotected. You specify the mode by selecting or deselecting the **Protected mode** option. The default mode of operation is protected. Protected mode means that the macro code is isolated in a separate process from the Teradata database, and an error does not cause the database processing to fail. It is recommended that you run the Publish Teradata Scoring function in protected mode during acceptance tests. When the model is ready for production, you can run the Publish Teradata Scoring function in unprotected mode. Note that you might see a significant performance advantage when you run it in unprotected mode.

Display detailed log messages

provides detailed information, which includes warning and error messages that occur during the publish process of a Teradata scoring function.

Sample Size

specifies the size of the sample to use for the scoring function validation. The default value is 100. The maximum number of digits that are allowed is 8.

How to Publish a Teradata Scoring Function

To publish a Teradata scoring function, follow these steps:

1. Verify that you have set the default scoring version for the project and have set the champion model for the default version. For more information, see [“Set a Default Version” on page 154](#) or [“Set a Champion Model” on page 151](#).
2. Select the project name and enter a value for the **DbmsTable** user-defined property. For more information, see [“User-Defined Properties” on page 287](#).

User-Defined Properties	
KeyType	
TableKey	
PreCode	
ScoringInputTable	
DbmsTable	

3. Right-click the project's name in the Project Tree and select **Publish Teradata Scoring Function**. The Publish Teradata Scoring Function window opens.

Publish Teradata Scoring Function

Publish Teradata Scoring Function for HomeEquity

SAS Metadata Location:

Function Name:

Teradata Server:

Database:

User ID:

Password:

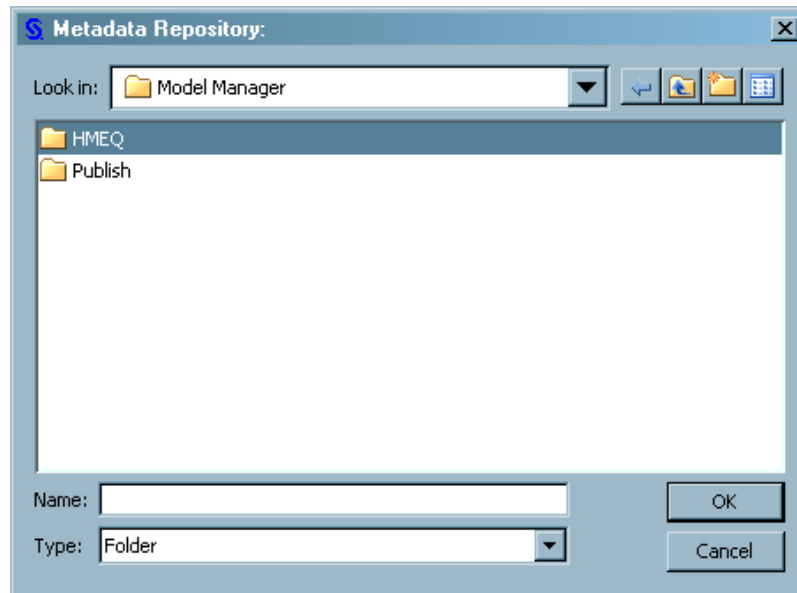
Options

☒ Validate scoring results ☐ Keep scoring function if validation fails

☒ Protected mode Sample Size:

☐ Display detailed log messages

4. To select a **SAS Metadata Location** in which to publish the model, click **Browse**, select a folder name, and then click **OK**.



5. Enter a name for the scoring function. The function name includes a prefix and a user-defined value. Only the user-defined value can be modified. Use the following naming conventions:

- The user-defined value must be unique across all projects.
- The maximum length for the user-defined value is 19, and no spaces are allowed.
- The only special character that can be included in the function name is an underscore.

Note: The user-defined value portion of the scoring function name is used by default for subsequent use of the scoring function from the same project. For more information, see [“User-Defined Properties” on page 287](#).

6. Enter a text value for the following fields:
 - **Teradata Server**
 - **Database**
 - **User ID**
 - **Password**
7. In the **Options** section, select one of the following check boxes for the desired validation options.
 - **Validate scoring results**
 - **Keep scoring function if validation fails**
 - **Protected mode**
 - **Display detailed log messages**

Note: The **Validate scoring results** and **Protected mode** options are selected by default.

8. Enter a numeric value for **Sample Size**.

SAS Publish Teradata Scoring Function

Publish Teradata Scoring Function for HomeEquity

SAS Metadata Location:

Function Name:

Teradata Server:

Database:

User ID:

Password:

Options

☒ Validate scoring results ☐ Keep scoring function if validation fails

☒ Protected mode Sample Size:

☐ Display detailed log messages

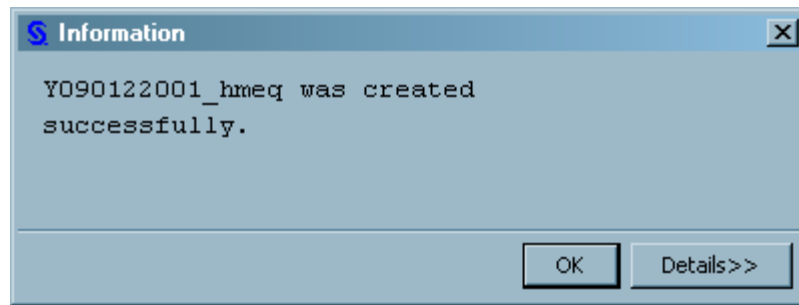
9. Click **OK**. A message is displayed that contains the scoring function name and the project champion model that will be published to the Teradata EDW.

Note: The user-defined value of the **Function Name** is validated against the target Teradata database. If the user-defined value is not unique, an error message is displayed.

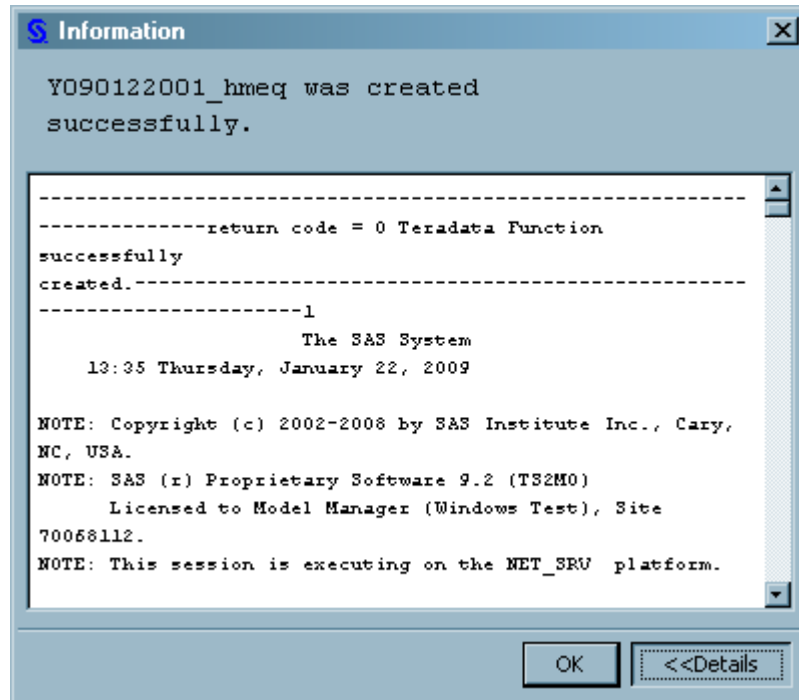
SAS Information

Complete Function Name is:
Y090122001_hmeq

10. Click **OK**. A message indicating that the scoring function has been created successfully or unsuccessfully is displayed.



11. Click **Details** to display information about the publish scoring function actions or error messages.



12. Click **OK** to complete the publishing process.

Log Messages

The log table is named UPDATE_LOG. All modules that are in the process of obtaining candidate projects to update the timestamp in the controller table are recorded in the log table. The time the process starts, who initiates the process, and the project being published are also recorded. Error messages can also be found in this table, which provides an audit trail of all relevant actions that take place within the publishing process. The UPDATE_LOG is a valuable source of information for auditing and debugging. A user can view the log file in the Details window when publishing a Teradata scoring function or by using a database viewer (for example, Teradata SQL Assistant 12). The log messages have the following formats that are held by the following columns of the table:

Action Type

specifies the type of action such as start update service, get project candidates, and load metadata tables.

Message Type

specifies the type of message, such as **info** for information message, **warning** for a warning message, or **error** for an error message.

Message

provides a full description of the event or action that took place

Timestamp

specifies the date and time when the action took place.

The UPDATE_LOG codes are as follows:

```
// Message Types
INFO = "10";
WARNING = "20";
ERROR = "30";

// Action Types

START_UPDATE_SERVICE = "100";
GET_PROJECT_CANDIDATES = "110";
GET_PROJECT_CANDIDATE_COUNT = "120";
CREATE_UDF = "200";
LOADING_META_TABLES = "300";
ROLLBACK = "400";
EXIT_ON_ERROR = "410";
EXIT_ON_NO_UPDATES = "480";
EXIT_ON_SUCCESS = "490";
RETRY_MM_CONNECT = "500";
RETRY_LOAD_META_TABLES = "510";
CONNECT_MODEL_MANAGER = "600";
DROPPING_UDF = "700";
```

Scoring Function Metadata Tables

The following tables are created in the Teradata database when publishing a scoring function:

project_metadata

provides information about the scoring project

model_metadata

provides information about the champion model, such as the function name and signature that are stored within this table.

project_model_info

maps a project to the champion model, as well as provides information about the current active scoring model.

update_log

provides information about the update service process such as when it was started, errors and maintenance messages about what has occurred during the publishing process. It also serves as the main audit trail.

rollback_error_log

records fatal errors that occur in the publishing process.

Chapter 13

Replacing a Champion Model

Overview of Replacing a Champion Model	177
Select a New Default Version	178
Retire a Project	178

Overview of Replacing a Champion Model

After reviewing production monitoring reports, you might find that it is necessary to retire a champion model and replace it with a champion model that performs better. Or, you might find that the project is no longer viable and you need to retire the project.

You can retire a champion model and replace it with a new champion model that resides in any version folder for the project.



Use the following table to determine the tasks to perform when you retire a champion model:

State of the Champion Model	Tasks to Perform
A new champion model is available in the same version folder	<p>Set the new model as the champion model.</p> <p>Update the model life cycle milestones status as appropriate. Approval of some tasks were based on the previous champion model.</p> <p><i>Note:</i> If the characteristic or stability analysis shows significant changes, set a new champion model in a different version folder. The bin definition that is used in characteristic and stability analysis is based on the data that was developed for the first champion model selected in the version and does not change after it is created. An out-of-date bin definition might cause incorrect characteristic or stability analysis.</p>
A new champion model is available in a different version folder.	<p>Set the model as the champion model.</p> <p>Set the version as the default version.</p>

State of the Champion Model	Tasks to Perform
A new champion model is not available when you want to retire a model but not retire the project.	<p>Set the project State property to be inactive.</p> <p>After a new model is ready, do the following:</p> <ul style="list-style-type: none"> • Set the model as the champion model. • Update the appropriate life cycle milestone tasks. If the new champion model is in the same version folder as the previous champion model, check that milestone tasks are based on the new champion model and not the previous champion model. • Set the project State property to be active.
No other champion models are planned for the project and the project is to be retired.	<p>Set the project State property to be retired.</p> <p>(Optional) Set the default version life cycle retire status as completed.</p>

Select a New Default Version

To retire a champion model at the version level, you either select a new default version. To select a new default version, follow these steps:

1. Ensure that a champion model is set in the version that you want to be the default version. The icon that indicates a champion model is a checkmark in a circle .
2. Right-click the new default version and select **Set Default Version**. The default champion icon  is indicated next to the version name.

Retire a Project

To retire a project, follow these steps:

1. Select the project in the Project Tree.
2. Click the **Status** property and select **Retired**.

Note: The **Status** property is also set to **Retired** when you sign off on the milestone action **RetireChampion** in the life cycle for the version.

Part 5

Performance Monitoring

Chapter 14

What Are Performance Monitoring Reports? 181

Chapter 15

Create Reports by Defining a Performance Task 197

Chapter 16

Create Reports Using Batch Programs 205

Chapter 14

What Are Performance Monitoring Reports?

Overview of Performance Monitoring Reports	181
Types of Performance Monitoring Reports	182
Overview of the Types of Performance Monitoring Reports	182
Summary Report	183
Data Composition Reports	183
Model Monitoring Reports	185
Performance Index Warnings and Alerts	188
Performance Data Sets Versus Performance Data Sources	190
The Process of Monitoring Champion Models	190
Formatting Performance Monitoring Reports	192
About Monitoring Reports	192
Create Monitoring Reports	193
Monitoring Report Output files	194
View Reports	195

Overview of Performance Monitoring Reports

To ensure that a champion model in a production environment is performing efficiently, you can collect performance data that has been created by the model at intervals that are determined by your organization. A performance data set is used to assess model prediction accuracy. It includes all of the required input variables as well as one or more actual target variables. For example, you might want to create performance data sets monthly or quarterly and then use SAS Model Manager to create performance monitoring reports for each time interval. After you create the reports, you can view the report charts in SAS Model Manager that give a graphical representation of the model's performance.

SAS Model Manager provides the following types of performance monitoring reports:

- Summaries of the types of information in project folders such as the number of models, model age distribution, input variables, and target variables.
- Reports that detect and quantify shifts in the distribution of variable values over time that occur in input data and scored output data.
- Performance monitoring reports that evaluate the predicted and actual target values for a champion model at multiple points in time.

You can create the performance monitoring reports, except for summaries, using either of the following methods:

- In the SAS Model Manager window, use the Define Performance Task wizard to generate the SAS code that creates the reports and then execute the generated code.
- Write your own SAS program using the report creation macros that are provided with SAS Model Manager and submit your program as a batch job. You can run your SAS program in any SAS session as long as the SAS session can access the SAS Content Server.

After you create the reports, you view the report charts in the SAS Model Manager window by selecting the **Performance** node in the default version. The report charts are interactive charts in which you modify charts to help you assess the champion model performance. For example, you can select different variables for the x-axis and y-axis, filter observations, and change chart types.

Types of Performance Monitoring Reports

Overview of the Types of Performance Monitoring Reports

After a champion model is in production, you can monitor the performance of the model by analyzing the SAS Model Manager performance reports. You can create the reports interactively using the Define Performance Task wizard and the **PerformanceMonitor** node from the project folder in the Project Tree or you can submit batch programs within SAS.

You can create the following types of performance reports:

Summary Report

The **Summary** report uses the information within organizational folders, project folders, and version folders to summarize the number of champion models, the number of models not in production, the model age, the number of reports, the input variables, and the target variables. The summary information enables you to compare the contents of organizational folders, projects, and versions. You view the Summary report from the Annotations view in the Project perspective.

Data Composition Reports

The two data composition reports, the **Characteristic Report** and the **Stability Report**, detect and quantify shifts in the distribution of variable values that occur in input data and scored output data over a period of time. The Characteristic report detects shifts in the distribution of input variables over time. The Stability report measures shifts in the scored output data that a model produces. By analyzing these shifts, you can gain insights on scoring input and output variables.

Model Monitoring Reports

The model monitoring reports are a collection of performance assessment reports that evaluate the predicted and actual target values. The model monitoring reports create a Lift Chart, a Gini-ROC Chart (Receiver Operating Characteristic), a Gini - Trend Chart, and KS charts.

When you create Data Composition reports and Model Monitoring reports, you can set performance index warnings and alerts. When certain thresholds are met, SAS Model Manager can send a warning and alert notification by e-mail to e-mail addresses that you configure either in the Define Performance Task wizard or in a SAS program.

You view the Data Composition reports and the Model Assessment reports from the version **Performance** node.

To explore the degradation of a model's performance over time using these charts, right-click in the chart and select **Data Options**. From the Data Options window, you can modify various values to further explore the degradation of a model. You can set different data filters and select different variables to replot a chart.

Summary Report

The Summary report summarizes the contents of different organizational folders, projects, and versions.

The contents of the Summary report is dynamic and is updated according to the folder that you select in the Project Tree. The scope of the information reported is defined by the collection of folders and objects that exist beneath the folder that is selected.

To view the Summary report, click the **Summary** tab that is in the Annotations view of the Projects perspective.

Use the following sections to evaluate and compare the contents of the different folders in the Project Tree:

General Properties

Use the **General Properties** section to browse the number of champion models, the number of versions, the number of scoring tasks, and the number of candidate models.

Production Models Aging Report

Use the **Production Models Aging Report** to view the number and aging distribution of champion models. The binned chronology report lists the number of champion models by deployment age, using six intervals to classify the deployment ages. The first four intervals combine to create a span of 365 days. The fifth interval adds another 365 days. The sixth interval reports the number of models that have been in production for two years or more.

Summary of Reports

Use the **Summary of Reports** section to browse the number of reports that have been generated in the **Reports** folder for the selected folder.

Model Target Variable Report

Use the **Model Target Variable Report** to browse the frequency that target variables are used in the models that exist for the selected folder. Each unique model target variable is reported, listing the number of models that use that variable as a target variable.

Model Input Variable Report

Use the **Model Input Variable Report** to browse the frequencies that input variables are used in the models for an organizational folder, a project, or a version. Each unique model input variable is reported, listing the number of models that use that variable as an input variable.

Data Composition Reports

Overview of Characteristic and Stability Reports

Together, the Characteristic and Stability reports detect and quantify shifts that can occur in the distribution of model training data, scoring input data, and model score output data.

The Characteristic report detects shifts in the distributions of input variables that are submitted for scoring over time. The Stability report measures shifts in the scored output data that a model produces. If a Characteristic report identifies a distribution shift in the input data, then the corresponding Stability report can help to assess the model's sensitivity to the distribution shift in the input data, in terms of the predictive performance of the scoring input variables.

While the Characteristic report indicates changes to the scope and composition of the submitted data sets over time, the Stability report evaluates the impact of the data variation on the model's predictive output during the same interval.

The Characteristic report does not require scoring. The Stability report requires output data from scoring to generate the deviation statistics of the output variable.

Note: For each time period that you execute the performance task, SAS Model Manager creates a new point on the Characteristic and Stability charts. Line segments between points in time do not appear on the charts until after the third iteration of executing the performance reports.

Characteristic Report

The Characteristic report detects and quantifies the shifts in the distribution of variable values in the input data over time. Input data variable distribution shifts can point to significant changes in customer behavior due to new technology, competition, marketing promotions, new laws, or other influences.

To find shifts, the Characteristic report compares the distributions of the variables in these two data sets:

- the training data set that was used to develop the model
- a current data set

If a large enough shifts occur in the distribution of variable values over time, the original model might not be the best predictive or classification tool to use with the current data.

The characteristic report uses a deviation index to quantify the shifts in a variable's values distribution that can occur between the training data set and the current data set. The deviation index is computed for each predictor variable in the data set, using this equation:

$$\text{Deviation_Index} = \Sigma (\%Actual - \%Expected) * \ln (\%Actual / \%Expected)$$

Numeric predictor variable values are placed into bins for frequency analysis. Outlier values are removed to facilitate better placement of values and to avoid scenarios that can aggregate most observations into a single bin.

If the training data set and the current data set have identical distributions for a variable, the variable's deviation index is equal to 0. A variable with a deviation index value that is $P1 > 2$ is classified as having a mild deviation. The Characteristic report uses the performance measure P1 to count the number of variables that receive a deviation index value that is greater than 0.1.

A variable that has a deviation index value that is $P1 > 5$ or $P25 > 0$ is classified as having a significant deviation. A performance measure P25 is used to count the number of variables that have significant deviations, or the number of input variables that receive a deviation index score value that is greater than or equal to 0.25.

Stability Report

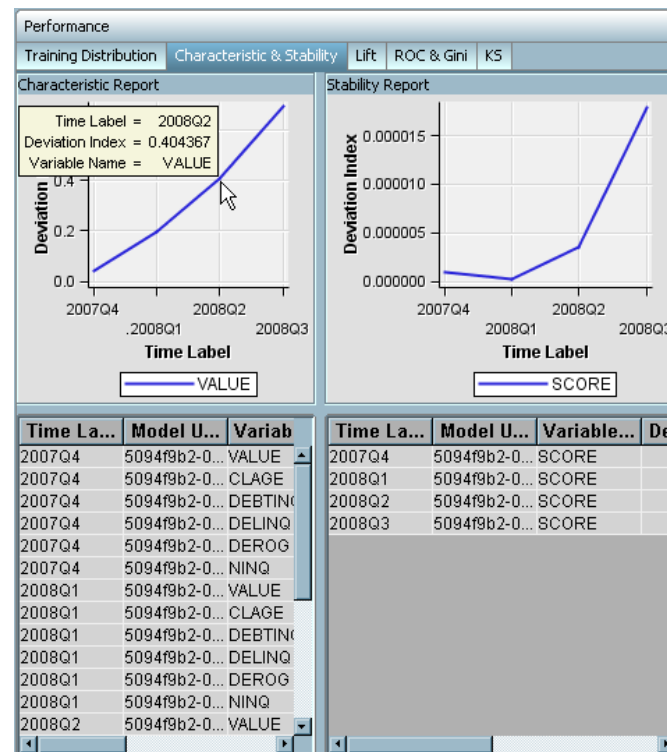
The Stability report evaluates changes in the distribution of scored output variable values as models score data over time. It uses the same deviation index function that is used by

the Characteristic report, except that the Stability report detects and quantifies shifts in the distribution of output variable values in the data that is produced by the models.

If an output variable from the training data set and the output variable from the current data set have identical distributions, then that output variable's deviation index is equal to zero. An output variable with a deviation index value that is greater than 0.10 and less than 0.25 is classified as having a mild deviation. A variable that has a deviation index value that is greater than 0.03 is classified as having a significant deviation. Too much deviation in predictive variable output can indicate that model tuning, retraining, or replacement might be necessary.

Example Characteristic and Stability Reports

The following report is an example of Characteristic and Stability reports. By placing the cursor over a point in the chart, you can view the data for that point.



Model Monitoring Reports

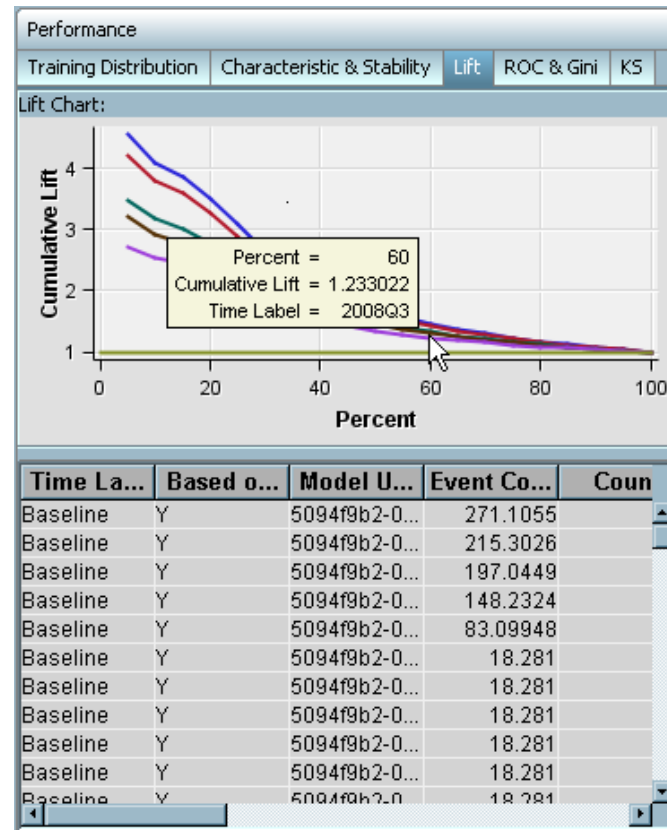
Monitoring Lift Report

The monitoring Lift report provides a visual summary of the usefulness of the information provided by a model for predicting a binary outcome variable. Specifically, the chart summarizes the utility that one can expect by using the champion model as compared to using baseline information only. Baseline information is the prediction accuracy performance of the initial performance monitoring task or batch program using operational data.

A monitoring Lift chart can view a model's cumulative lift at a given point in time or the sequential lift performance of a model's lift over time. To detect model performance degradation, you can set the Lift report performance indexes Lift5Decay, Lift10Decay,

Lift15Decay, and Lift20Decay. The data that underlies the Lift chart is contained in the report file mm_lift.sas7bdat in the **Resources** folder.

The following chart is an example of a monitoring Lift report. By placing the cursor over a point in the chart, you can view the data for that point.



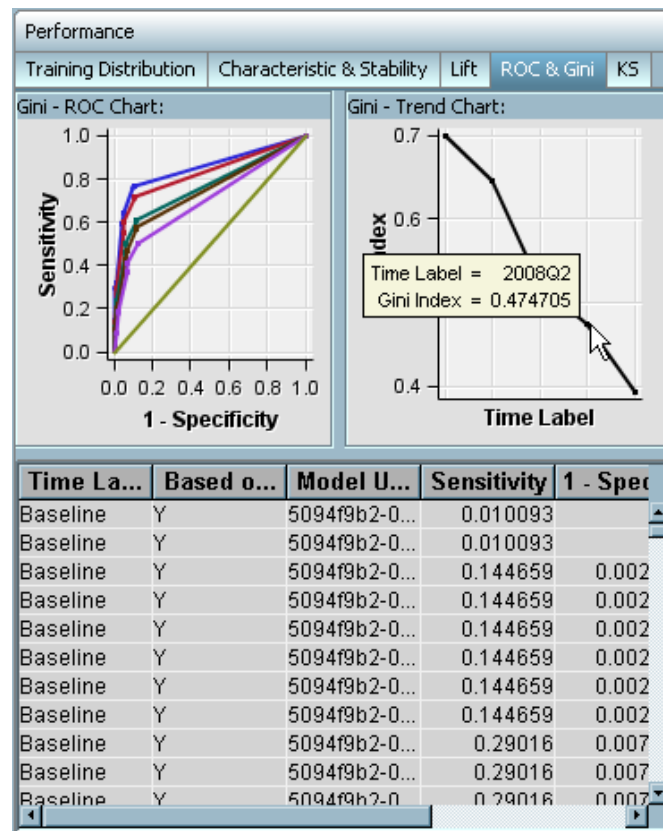
Monitoring ROC & Gini Report

The monitoring ROC & Gini plot reports the predictive accuracy of a model that has a binary target. The plot displays Sensitivity information about the y axis and 1–Specificity information about the x axis. Sensitivity is the proportion of true positive events. Specificity is the proportion of true negative events. The Gini index is calculated for each ROC curve. The Gini coefficient, which represents the area under the ROC curve, is a benchmark statistic that can be used to summarize the predictive accuracy of a model.

Use the monitoring ROC & Gini report to detect degradations in the predictive power of a model.

The data that underlies the monitoring ROC & Gini report is contained in the report component file mm_roc.sas7bdat.

The following chart is an example of a monitoring ROC & Gini report. By placing the cursor over a point in the chart, you can view the data for that point.



KS Report

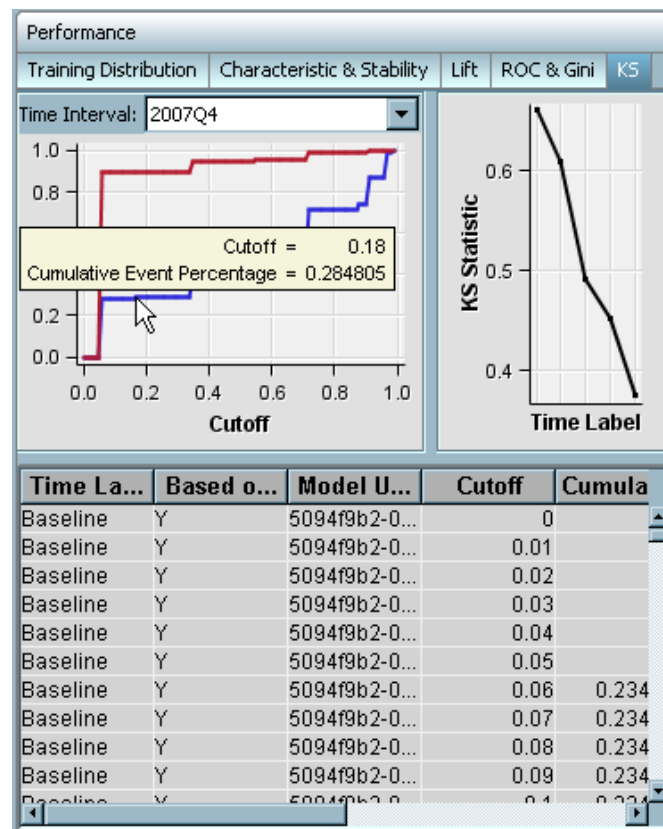
The KS report contains the Kolmogorov-Smirnov (KS) test plots for models with a binary target. The KS statistic measures the maximum vertical separation, or deviation between the cumulative distributions of events and non-events. This trend report uses a summary data set that plots the KS statistic and the KS probability cutoff values over time.

Use the KS report to detect degradations in the predictive power of a model. To scroll through a successive series of KS performance depictions, select a time interval from the **Time Interval** list box. If model performance is declining, it is indicated by the decreasing distances between the KS plot lines.

To detect model performance degradation, you can set the ksDecay performance index in the KS report.

The data that underlies the KS chart is contained in the report component file mm_ks.sas7bdat.

The following report is an example of a KS report. By placing the cursor over a point in the chart, you can view the data for that point.



Performance Index Warnings and Alerts

The production model performance reports use performance measurement thresholds to benchmark and gauge the performance of a predictive model. When one of the performance measurements exceeds one or more specified indexes or thresholds, warning and alert events occur. When warning or alert events occur, warning and alert notifications are automatically sent by e-mail to recipients whose e-mail address is configured either in the Define Performance Task wizard or in the batch program that runs the reports.

Use the following assignment statements to set warning and alert conditions:

```
alertCondition='alert-condition';
warningCondition='warning-condition';
```

The condition must be enclosed in quotation marks if you use SAS code to create the report. An error occurs if you enclose the condition in quotation marks in the Define Performance Task wizard.

The following indexes and thresholds can be configured in either the Define Performance Task wizard or in a batch program that creates the report specifications:

Characteristic report

You can configure the thresholds for the performance indexes P1 and P25. The P1 and P25 indexes represent the count of input variables with deviation index scores exceeding 0.1 and 0.25 respectively. The following is an example of alert and warning thresholds:

```
alertCondition='p1>5 or p25>0';
warningCondition='p1>2';
```

Stability report

You can configure output deviation index scores for a model's output variable. The output deviation index scores represent the deviation levels in the distribution of the model's scored output variables. The following is an example of alert and warning thresholds:

```
alertCondition='outputDeviation>0.03';
warningCondition='outputDeviation>0.01';
```

Model Assessment reports

For the Lift, ROC&Gini, and KS reports, you can configure threshold values for the following decay statistics.

lift5Decay

is the lift performance decay based on the top 5% of the target population of interest from time A to time B.

lift10Decay

is the lift performance decay based on the top 10% of the target population of interest from time A to time B.

lift15Decay

is the lift performance decay based on the top 15% of the target population of interest from time A to time B.

lift20Decay

is the lift performance decay based on the top 20% of the target population of interest from time A to time B.

giniDecay

is the performance decay of the Gini index from time A to time B.

ksDecay

is the performance decay of the KS statistic from time A to time B.

The following is an example of alert and warning thresholds:

```
alertCondition='(lift5Decay>0.15 and lift10Decay>0.12)
               or giniDecay>0.1 or ksDecay>0.1';
warningCondition='lift5Decay>0.05';
```

The following table is an example of a warnings and alerts notification table:

perfIndex	perfDecay	alertCondition	alertEval	warningCondition	warningEval
p1=0; p25=0;		p1>5 or p25>0	False	p1>2	False
lift5=4.055; lift10=4.037; lift15=3.690; lift20=3.416; giniIndex=0.689; ksStatistic=0.628;	lift5Decay=0.077; lift10Decay=0.071; lift15Decay=0.079; lift20Decay=0.070; giniDecay=0.088; ksDecay=0.077;	(lift5Decay>0.15 and lift10Decay>0.12) or giniDecay>0.1 or ksDecay>0.1	False	lift5Decay>0.05	True
outputDeviation=0.000;		outputDeviation > 0.03	False	outputDeviation > 0.01	False

The warnings and alerts notification table displays the computed performance index and performance decay statistics that were calculated for the model, as well as summaries of

the alert and warning threshold settings that were specified for the model. The calculated statistics are compared with the alert and warning threshold settings.

When an alertEvaluation or warningEvaluation column displays a "True" value, the warning and alerts table is e-mailed to the configured recipients. When the value is "False", no e-mail notification is sent.

See Also

[“Types of Performance Monitoring Reports” on page 182](#)

Performance Data Sets Versus Performance Data Sources

To monitor the performance of the champion model, you take a snapshot of the performance data. The snapshot is used to assess model prediction accuracy. It includes all of the required scoring input variables as well as one or more actual target variables.

When you add a table to SAS Model Manager to be used as a data source, the table must first be registered in SAS Management Console. SAS Model Manager data sources are pointers to tables that are registered in SAS Management Console. When you add performance data as a SAS Model Manager performance data source, SAS Model Manager uses the location of the performance data set that is registered in SAS Management Console.

When you create performance monitoring reports using the Define Performance Task wizard, you are required to specify a SAS Model Manager data source. When you create performance monitoring reports by submitting batch programs, the performance data must be in a performance data set that can be accessed by the SAS session that runs the batch program.

See Also

[“Determine How to Use the Performance Data Sets” on page 198](#)

The Process of Monitoring Champion Models

Your project plan might include a schedule to monitor the champion model performance, or your plan might require that you monitor the performance at any time. For each time period that you monitor the champion model, you take a snapshot of the data for that time period and use that data as the performance data source for creating the monitoring reports.

You can create the monitoring reports from the Project Tree in the SAS Model Manager window, or you can submit batch programs to create the reports. Both methods require the same information. If you run batch programs to create the reports, you can find example programs in the sashelp.modelmgr.source catalog. These reports filenames are **reportexample_x**, where *x* is a number from 1 to 4.

The following table lists the tasks that are required to create performance reports:

Task	Reports Created by Using the Define Performance Task Wizard	Reports Created Using SAS Programs Run in Batch
Create a folder structure for report files	The folder structure is inherent in the Project Tree. No action is necessary.	Create a folder structure on a local computer.
Obtain performance data	The performance data is a SAS data set that is snapshot of model output. It must be registered in SAS Management Console and added as a performance data source in SAS Model Manager.	The performance data is used to assess model prediction accuracy. It includes all of the required scoring input variables as well as one or more actual target variables. You can store performance data sets anywhere as long as they can be accessed by the SAS session that runs the batch program. The data set does not need to be registered with SAS Management Console or added as a SAS Model Manager data source.
Ensure access to the champion model	This process is performed by the Define Performance Task wizard. No action is necessary.	Run the %MM_GetModels() macro to extract the champion model in a channel to the local computer.
Map model and project output variables.	Map the model and project output variables using the Project Tree.	Map the model and project output variables using the Project Tree.
Define report specifications	The report specification are derived from project data and input that you specify in the Define Performance Task wizard. The wizard generates the SAS code to create the performance reports.	Write the following DATA steps: <ul style="list-style-type: none"> • mm_jobs.project • mm_jobs.emailaddr • mm_jobs.reportdef • mm_jobs.jobtime
Specify the report execution operational environment	The operational environment is known to SAS Model Manager. No action is necessary.	Define the required macro variables that are used by the %MM_RunReports() macro.

Task	Reports Created by Using the Define Performance Task Wizard	Reports Created Using SAS Programs Run in Batch
Run the reports	Execute the code that was generated by the Define Performance Task wizard. The data sets that underlie the monitoring reports are stored in the Resources folder.	Create a DATA step that points to the performance data sets and execute the %MM_RunReports() macro. The data sets that underlie the monitoring reports are stored in the Resources folder when the reports are created in production mode. In Test mode, the monitoring reports data sets reside in the location specified in the mm_jobs.project data set.
View the reports	Select the version Performance folder to view the reports.	Select the version Performance folder to view the reports.

Formatting Performance Monitoring Reports

About Monitoring Reports

After you execute a performance task from the SAS Model Manager window or run the %MM_RunReports() macro in production mode, as a batch job, SAS Model Manager stores the output data sets in the default version **Resources** folder. You can use the New Reports wizard to format the performance monitoring results in PDF, HTML, RTF, or Excel output formats, or you can view the performance monitoring results by selecting the default version **Performance** node. When you create the Monitoring reports using the New Reports wizard, the report creates the following charts:

Assessment Charts

The Assessment Charts summarize the utility that one can expect by using the respective models, as compared to using only baseline information. The Assessment Charts can view a model's lift at a given point in time or the sequential lift performance of a model's lift over time. The Monitoring report creates the following Assessment Charts:

- Lift
- Cumulative Lift
- Percent Response
- Cumulative Percent Response
- Captured Response
- Cumulative Captured Response

Lift Trend Chart

A Lift Trend chart displays the cumulative lift of the champion model, over time.

ROC Chart

Sensitivity is the proportion of true positive events and specificity is the proportion of true negative events. The ROC Chart plots Sensitivity on the Y axis and 1 - Specificity on the X axis.

Gini Trend

When the ROC Chart is created, the Gini index for each ROC curve is also created. The Gini coefficient represents the area under the ROC curve and is a benchmark statistic that can be used to summarize the predictive accuracy of a model. The Gini Trend Chart plots a model's Gini index scores over time, which is used to monitor model degradation over time.

KS Chart

The KS Chart uses the Kolmogorov-Smirnov statistic to measure the maximum vertical separation, or deviation between the cumulative distributions of events and non-events.

KS Trend Chart

When you create a Kolmogorov-Smirnov report, the KS statistic and the corresponding probability cutoff are computed for each Kolmogorov-Smirnov table. The KS Trend Chart uses a summary data set that plots the KS Statistic and the probability cutoff values over time. The KS Trend Chart is used to monitor model degradation over time.

Before you create the Monitoring Report, you must ensure that certain project and model properties are set. For more information, see [“Verify Project and Model Property Settings” on page 131](#).

These are the tasks that you perform for Monitoring reports:


- [“View Reports” on page 134](#)

See Also

- [“Create Reports by Defining a Performance Task” on page 197](#)
- [“Create Reports Using Batch Programs ” on page 205](#)

Create Monitoring Reports

To create a Monitoring report, follow these steps:

1. Expand the version folder .
2. Right-click the **Reports** node and select **Reports** ⇒ **New Report Wizard**. The New Report Wizard opens.
3. Select **Monitoring Report** from the **Reports** list box.

New Report Wizard

Reports: Monitoring Report

Select Format: PDF

Select Model(s):

S...	ID	T...	...
------	----	-----	-----	------	-----

Report Properties

General Properties	
Name *	Monitoring_D2009-02-16T17.21
Description	Monitoring

OK Cancel

4. In the **Select Format** list box, select the type of output that you want to create. The default is **PDF**. Other options are **HTML**, **RTF**, and **Excel**.
5. In the **Report Properties** table, complete the **Name** and **Description** properties if you do not want to use the default values. The default value for the **Name** property uses the form **Monitoring_DdateTtime**. The following characters cannot be used in the name: **@, \, /, *, %, #, &, \$, (,), !, ?, <, >, ^, +, **.
6. Click **OK**. A message box confirms that the report was created successfully.

See Also

[“View Reports” on page 134](#)

Monitoring Report Output files

The New Reports Wizard stores the Monitoring Report output files in a report node under the **Reports** folder. The name of the report node is the value of the **Name** field that you specified in the New Report Wizard **Report Properties**.

Each time you run the New Report Wizard, the wizard creates four files:

- the report in either HTML, PDF, RTF, or Excel format
- taskCode.log
- taskCode.lst
- taskCode.sas

The wizard overwrites the output files if output file of the same name already exist.

Here is a description of the model comparison output files:

Report File	Description
<i>report-name.html</i>	This file is the report output in HTML format.
<i>report-name.pdf</i>	This file is the report output in PDF format.
<i>report-name.rtf</i>	This file is the report output in RTF format.
<i>report-name.xls</i>	This file is the report output in Excel format.
taskCode.log	This file is the log file that contains messages from running the SAS code to create the report.
taskCode.lst	This file is the report output in the SAS listing output if the ODS Listing destination was set; otherwise, an empty file.
taskCode.sas	This file is the SAS code that is used to create the report.

After you create a report, you view the report from the **Reports** folder.

View Reports

To view the performance monitoring reports in the SAS Model Manager window, select the **Performance** node. The right pane displays a view where each tab is one of the reports. Select a tab to see a report.

You can format the performance monitoring reports in PDF, HTML, RTF, or Excel as well by creating Monitoring reports using the New Reports wizard. For more information, see [“Formatting Performance Monitoring Reports” on page 192](#).

Chapter 15

Create Reports by Defining a Performance Task

Overview of Creating Reports Using a Performance Task	197
Creating Reports Using a Performance Task	197
Determine How to Use the Performance Data Sets	198
Prerequisites for Running the Define Performance Task Wizard	199
Overview of Prerequisites	199
Ensure That the Champion Model and Default Version Are Set	200
Add the Performance Data Source	200
Ensure That Project Model Properties Are Set	200
Map Model and Project Output Variables	201
Run the Define Performance Task Wizard	201

Overview of Creating Reports Using a Performance Task

Creating Reports Using a Performance Task

You define and execute a performance task for a SAS Model Manager project. The model that you monitor is always the champion model that is set in the default version folder for the project. The process of creating performance reports is a two-step process. First, you run the Define Performance Task wizard to generate the code that creates the reports. Then, you execute the generated code. SAS Model Manager stores the output data in the **Resources** folder of the default version. To view the resulting data, you select the **Performance** node in the default version.

To create performance reports in SAS Model Manager, follow this process:

- Ensure that a performance data source is added in the **Data Sources** perspective **Performance Tables** folder.
- Ensure that all prerequisites have been completed.
- Run the Define Performance Task wizard to generate the SAS code that creates the performance reports.
- Execute the generated code.
- To view the reports, select the **Performance** node in the default version.

Determine How to Use the Performance Data Sets

Before you run the Define Performance Task wizard, the performance data set must be available as a SAS Model Manager performance table. For each SAS Model Manager project, you can set up your environment to use the performance data source that best suits your business processes:

- **Scenario 1:** You periodically take a snapshot of an operational data set to create a performance data set. Each time that you take a snapshot, you give the performance data set a new name. Each performance data set must be registered in SAS Management Console and added as a SAS Model Manager performance table. For each time interval, you name a new performance data source when you run the Define Performance Task wizard.
- **Scenario 2:** You take a snapshot of the operational data set to create a performance data set over time, and you reuse the same name for the performance data set. You register the performance data set with SAS Management Console only once and add the data set as a SAS Model Manager performance table only once. Each time you take a snapshot, you replace the performance data set at the location where the performance data set is registered in SAS Management Console. SAS Model Manager uses the registered location in SAS Management Console to determine the location of a data source.

When you run the Define Performance Task wizard, the name of the performance data source does not change. Because you used the performance data source static name as the **Default Performance Table** in the project properties, the **Performance data source** box in the wizard is completed by SAS Model Manager.

The following table summarizes the tasks that are performed if performance reports are run after six months or for reports that are run every month. Use this task and example table to help you determine how you want to name your performance data sets and your SAS Model Manager performance data sources.

Task	Scenario 1: The Performance Data Set Name Changes	Scenario 2: The Performance Data Set Name Remains Static
Create a performance data set from model output data	Each month, take a snapshot of the operational data and create a performance data set with a different name: <ul style="list-style-type: none"> • Jul09 • Aug09 • Sep09 • Oct09 • Nov09 Dec09 	Every month, take a snapshot of the operational data and name the performance data set using the same name: 2009perf
Register the performance data set in SAS Management Console	Register the data sets monthly or register them all at once before you run the reports.	Register the data sets the first month only.

Task	Scenario 1: The Performance Data Set Name Changes	Scenario 2: The Performance Data Set Name Remains Static
Add the performance data set as a SAS Model Manager performance data source	Add the data sets monthly or add them all at once before you run the reports.	Add the data source the first month only.
Modifications to make in the Define Performance Task wizard	In Step 3, select a performance data source, select a date, and a date label.	In Step 3, select a date and a date label. The Performance data source field contains the static name of the performance data source name because it was specified for the previous execution of the task for this project.
Create the reports	For each performance data source, run the Define Performance Task wizard and execute the reports from the PerformanceMonitor project node. Because each performance data source has a different name, you can run the performance task as desired; the task does not need to be run monthly.	Monthly, run the Define Performance Task wizard and execute the reports from the PerformanceMonitor project node. To ensure that you do not write over important performance data, run the performance task before a new snapshot of the operational data is taken.

Prerequisites for Running the Define Performance Task Wizard

Overview of Prerequisites


Before you run the Define Performance Task wizard, the environment must be set appropriately as follows:

- Ensure that the champion model and default version are set.
- Ensure that the performance data sets for the time period that you want to monitor are available as SAS Model Manager performance data sources.
- Ensure that the appropriate project and model properties are set.
- Ensure that the project output variable is mapped to the model output variable.



After the environment is set, you can run the Define Performance Task wizard.

Ensure That the Champion Model and Default Version Are Set

The Define Performance Task wizard generates report code for the champion model in the default version.

You can determine the default version and the champion model by looking for the  icon next to the default version name and the champion model name.

If the default version or the champion model are not set, follow these steps:

1. Right-click the champion model name and select **Set Champion Model**. The  icon appears next to the champion model name.
2. Right-click the version name where the champion model resides and select **Set Default Version**. The  icon appears next to the version name.

Add the Performance Data Source

The Define Performance Task wizard requires that the performance data be a SAS Model Manager performance data source. Before you can add a SAS Model Manager performance data source, the performance data set must be registered in SAS Management Console.

To add a performance data source, follow these steps:

1. Select the Data Sources perspective button.
2. Right-click the **Performance Tables** folder and select **Add Data Source**. The Add Data Source window opens.
3. Click the **Library** list box and select a library.
4. Select a table and click **OK**.

If your performance table is not listed in the Performance Tables folder, your administrator must add the table to the SAS Metadata Repository Data Library Manager using SAS Management Console. For more information, see the *SAS Model Manager 2.2 Administrator's Guide*.

See Also

[“Project Tables” on page 28](#)

Ensure That Project Model Properties Are Set

Several properties must be defined in order to generate the model performance reports. From the Project perspective, verify that the following properties are specified:

- the project **Training Target Variable** property
- the project **Target Event Value** property
- the project **Output Event Probability Variable**
- the champion model **Score Code Type** property

Map Model and Project Output Variables

In order to create the model performance reports, the model output variable must be mapped to the project output variable if the corresponding project variable and the model variable have different names. To map these output variables, follow these steps:

1. Select the model from the **Models** node.
2. In the right pane, click the **Model Mapping** tab.
3. For each project output variable, select a variable from the **Model Variables** list box.

Run the Define Performance Task Wizard

To create the monitoring reports you run the Define Performance Task wizard to generate SAS code. You then execute the generated code. Execution of the generated code creates the SAS data sets that are used to display the monitoring reports from the version **Performance** node.

To create the reports, follow these steps:

1. Right-click the project name and select **Define Performance Task**. The **Define Performance Task** wizard appears, and creates a **PerformanceMonitor** node if one does not exist.

Define Performance Task (Step 1 of 3)

Input and Output Variables

Select Output Variable(s):

Select	Keep Variables	Description
<input checked="" type="checkbox"/>	score	

Select All Deselect All

Select Input Variable(s):

Select	Keep Variables	Description
<input checked="" type="checkbox"/>	MORTDUE	
<input checked="" type="checkbox"/>	REASON	
<input checked="" type="checkbox"/>	DELINQ	
<input checked="" type="checkbox"/>	DEBTINC	

Select All Deselect All

Back Next Cancel Help

2. In the **Select Output Variable(s)** table, select the output variable or variables. To select all output variables, click **Select All**.
3. In the **Select Input Variable(s)** table, select the input variables. To select all input variables, click **Select All**. Click **Next**.
4. In the **General Properties** table, verify the values for the warning and alert conditions. Modify the values as necessary. Make sure that the values are not enclosed in quotation marks. Click **Next**.

Define Performance Task [X]

Warning and Alert Conditions

Step 2 of 3

General Properties	
Characteristic alert condition *	p1>5 or p25>0
Characteristic warning condition *	p1>2
Stability alert condition *	outputDeviation > 0.03
Stability warning condition *	outputDeviation > 0.01
Model assessment alert condi... *	(lift5Decay>0.15 and lift10Decay>0.12) or giniDeca...
Model assessment warning c... *	lift5Decay>0.05

Back Next Cancel Help

- Click the **Performance data source** box and select a performance data source.

Define Performance Task [X]

Other Reporting Specifications

Step 3 of 3

Performance data source: NewPerfData.HMEQ_2008Q1

Choose a date: Mar 31, 2008 ... Date label: 2008Q1

E-mail to:

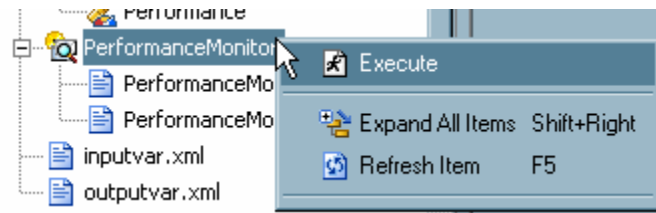
E-mail:	Send alert warning:	Send job status:
sasadm@mycompany.com	Yes	Yes

Add Delete

Back Next Finish Cancel Help

- Click the ellipsis button (...) for the **Choose a date** box and select a date. The date can be any date in the time period when the performance data was collected. SAS Model Manager uses the date value to sequence data. Therefore, you can select any date in the time period when the performance data source was collected.
- To add a label for the date, type the label in the **Date Label** box. The date label represents the timeframe of the performance data source. Date labels appear on the report charts. In order for the chart to be readable, select short, clear date labels.
- (Optional) To send the scoring results by e-mail, click the **Add** button in the **E-mail to** table. The Add Contact window opens.
 - Type an e-mail address.
 - Select either **Yes** or **No** if you want an alert and warnings to be sent by e-mail when alert or warning thresholds have been exceeded.

- c. Select either **Yes** or **No** if you want a completion notice to be sent by e-mail every time the performance task runs.
9. Click **Finish**. The **Working** status box appears while the code is generated.
10. When the code generation is complete, the Information window opens with a success message. Click **OK**. You can view the generated code in the project **PerformanceMonitor** folder.
11. To execute the generated code, right-click the **PerformanceMonitor** folder and select **Execute**. SAS Model Manager saves the data sets that create the monitoring reports in the default version **Resources** folder.



Note: If the report creation fails, you can view the SAS log to look for error messages by selecting the PerformanceMonitor.log file in the **PerformanceMonitor** node.

Chapter 16

Create Reports Using Batch Programs

Overview of SAS Programs to Monitor Model Performance	206
Prerequisites for Running Batch Performance Reports	207
Overview of Prerequisites for Running Batch Performance Reports	207
Publish the Champion Model from the Project Folder	207
Create a Folder Structure	207
Obtain Performance Data	209
Determine the Export Channel	209
Copy Example Batch Programs	209
Determine SAS Model Manager User ID and Password	210
Report Output in Test and Production Modes	210
Report Output in Test Mode	210
Report Output in Production Mode	210
Define the Report Specifications	211
Overview of Code to Define Report Specifications	211
Required libref	211
Project Specifications	211
E-mail Recipient Specifications	214
Report Specifications	215
Job Scheduling Specifications	218
Example Code to Create the Report Specifications	219
Extracting the Champion Model from a Channel	222
Using the %MM_GetModels() Macro	222
Accessing SAS Model Manager Report Macros	223
%MM_GetModels() Macro Syntax	223
Example Program to Extract a Model from a Channel	223
The current.sas7bdat Data Set	223
SAS Code to Run Performance Reports	225
Overview of the SAS Code to Run the Performance Reports	225
Accessing SAS Model Manager Report Macros	225
Required Librefs	225
Macro Variables to Define Report Local Folders and Data Sets	226
Macro Variables That Are Used by the %MM_RunReports() Macro	226
The DATA Step to Access the Performance Data Set	228
The %MM_RunReports() Macro	228
Example Code to Run the Reports	229

Overview of SAS Programs to Monitor Model Performance

A SAS program that creates performance monitoring reports consists of three conceptual sections:

- The first section defines the report specifications that identify the project, the types of reports that you want to create, alert and warning conditions, and the date and time to run the batch jobs.
- The second section extracts the champion model from a channel. Any batch program that creates performance monitoring reports must extract models from a publishing channel. The champion model must have been published to the channel from the project folder.
- The third section defines the operational environment and the performance data set, and then calls a SAS macro that creates the reports.

You define the report specifications by writing four DATA steps:

- `mm_jobs.project` defines the project specifications.
- `mm_jobs.emailaddr` defines the e-mail address where you send job, alert, and warning notifications.
- `mm_jobs.reportdef` defines which type of reports you want to create and the alert and warning conditions for those reports.
- `mm_jobs.jobtime` defines the date and time to run the batch jobs.

After the report specification data sets have been created, you extract the model from the publishing channel to the local computer using the `%MM_GetModels()` macro. Finally, you set macro variables to define the operating environment, specify the performance data set, and run the `%MM_RunReports()` macro to create the reports.

You view the reports by selecting the version **Performance** node in the SAS Model Manager window.

SAS Model Manager provides these performance monitoring macros:

- `%MM_GetModels()` to extract models from a publishing channel
- `%MM_UpdateCharacteristicTable` to create a Characteristic report
- `%MM_UpdateStabilityTable` to create a Stability report
- `%MM_UpdateAssessmentTable` to create model monitoring reports
- `%MM_RunReports()` which sets the operating environments and launches the macros that create the reports.

SAS supplies example SAS programs in the `sashelp.modelmgr` catalog that you can modify for your environment.

Prerequisites for Running Batch Performance Reports

Overview of Prerequisites for Running Batch Performance Reports

Batch performance reporting requires you to complete several tasks before you can modify the example programs. After the following tasks have been completed, you are ready to modify the example programs:

- Ensure that the champion model has been published from the project folder.
- Create a folder structure on the local computer.

Note: The local computer and the folder that are used in the process of creating batch performance reports must be accessible to the batch performance program.

- Store performance data sets on the local computer.
- If you are using SAS example programs, copy the example programs to the local computer.
- Determine the channel that is used to publish the project or model.
- Determine a SAS Model Manager user ID and password to authorize the batch processing.

Publish the Champion Model from the Project Folder

In order to run performance reports in batch, you must publish the champion model from the project folder. The SAS Model Manager performance macros use project metadata when running performance reports.

Whenever you have a new champion model, you must publish the new champion model again, from the project folder.

Create a Folder Structure

Create a folder structure on your local computer to contain the report monitoring files. First, create a root folder to contain performance reporting files for one or more SAS Model Manager projects. You might further organize your file structure by project. The examples in the following table use a classification of HMEQ for the files that are used to create home equity performance monitoring reports. Create folders to contain the following types of files:

Folder Contents	Description	Example
job local path	Specifies the folder that contains the reporting specification data sets that are used by the %MM_RunReports() macro.	c:\mmReports\HMEQ\reportJobs

Folder Contents	Description	Example
report output	Specifies the folder that contains data sets and auxiliary files that are created during the creation of the performance reports when the %MM_RunReports() macro is run in test mode.	c:\mmReports\HMEQ\testReportOutput\
performance data	Specifies the folder that contains the performance data sets for each time period. Performance data sets can be stored in a DBMS as well. If your performance data set is in a DBMS, then this folder is not necessary.	c:\mmReports\HMEQ\scoreIn
channel	Specifies the folder on the local computer to save the SPK file that is created during the processing of the %MM_GetModels() macro. The SPK file contains the model. When you publish a model to a channel, the published package is placed in this folder. A channel can be shared by multiple SAS Model Manager projects. You can define the channel to any location as long as it can be accessed by the %MM_GetModels() macro.	c:\mmReports\HMEQ\channel2
model	Specifies the folder to where the SPK model is extracted to by the %MM_GetModels() macro. The macro creates a \scorecode folder that contains the model score code and saves the data set current.sas7bdat, logs.sas7bdat, and processingspk.sas7bdat in the model folder. The current.sas7bdat data set contains project and model information that is used to create the performance monitoring reports.	c:\mmReports\HMEQ\model c:\mmReports\HMEQ\model\scorecode

To ensure that your report data is not lost, regularly back up these report folders.

See Also

[“Report Output in Test and Production Modes” on page 210](#)

Obtain Performance Data

The performance data set is a snapshot of a data set that includes scoring input variables and one or more target variables. After the snapshot is available, store the data set in a performance data folder on the local computer.

See Also

[“Creating a Performance Table” on page 34](#)

Determine the Export Channel

You can determine the channel that was used to publish the model from the project folder to the SAS Metadata repository from SAS Management Console. In SAS Management Console, select the **Plug-ins** tab and expand the following nodes under **SAS Management Console** to find the publishing channels that are used by SAS Model Manager:

Environment Management ⇒ **Publishing Framework** ⇒ **Foundation** ⇒ **Channels** ⇒ **Model Manager Channels**. You can attempt to extract the model using each of the SAS Model Manager publishing channels. Right-click **Properties**. The channel path is located on the **Persistent Store** tab.

Note: A publish channel can be shared by multiple SAS Model Manager projects.

Copy Example Batch Programs

SAS provides several example programs that you can use to create a batch program that monitors the performance of the champion model. You can find the example programs in the sashelp.modelmgr catalog. The catalog includes these example programs:

reportExample1

contains example SAS code to extract a project or model from the channel using the %MM_GetModels() macro.

reportExample2

contains DATA steps to create performance data that can be used to test the batch programs that create performance monitoring reports.

reportExample3

contains example DATA steps to create the SAS data sets that contain report specifications, such as the project UUID and path, various input variables, the location of the performance data source, alert and warning conditions, and e-mail addresses for report notifications.

reportExample4

contains an example program that are used to define the operating environment using macro variables. This program also contains the DATA steps that are used to create the reports.

You can copy these example programs to the job local path folder and you can modify them for your operating environment.

Determine SAS Model Manager User ID and Password

The performance monitoring reports must specify a valid SAS Model Manager user ID and password. The user ID can have any of the following roles:

- Model Manager User
- Model Manager Advanced User
- Model Manager Administrator

See Also

[“SAS Model Manager User Groups, Roles, and Tasks” on page 18](#)

Report Output in Test and Production Modes

Report Output in Test Mode

When you run the %MM_RunReports() macro, you can either run the report in Test mode or Production mode, by using the _MM_ReportMode macro variable.

To run in Test mode, ensure that you make the following assignments:

- In the DATA step mm_jobs.project, set the variable `testDestination=reportOutputPath`, where `reportOutputPath` is the report output folder on the local computer or network. This is the location that you defined when you completed the prerequisites for running batch performance jobs.
- In the %MM_RunReports() macro, set the macro variable `_MM_ReportMode=TEST`.

Test report output is then written to the local computer or network location. You can test your %MM_RunReports() macro any number of times without corrupting the integrity of your model repository. You can delete the contents of the report output folder and resubmit your macro as necessary.

To view the report output, you can copy the files from the report output folder to any version folder whose **Resources** folder is empty. A best practice would be to create a test version and copy the files to the test version **Resources** folder. After the files are in the **Resources** folder, you can select the **Performance** folder in the version to view the test output. If you do not create a test version, ensure that you delete the files from the **Resources** folder when you no longer need these files.

See Also

[“Prerequisites for Running Batch Performance Reports” on page 207](#)

Report Output in Production Mode

When you run the %MM_RunReports() macro in PRODUCTION mode, ensure that you complete the following code changes:

- In the DATA step mm_jobs.project, remove the assignment of the variable `testDestination=reportOutputPath`.
- In the %MM_RunReports() macro, set the macro variable `_MM_ReportMode=PRODUCTION`.

Production report output is written to the **Resources** folder in the default version of the project. To view the report output, you select the **Performance** folder in the default version.

Define the Report Specifications

Overview of Code to Define Report Specifications

Before you can create a monitoring report for a project, you must create several data sets that define the report specifications:

`mm_jobs.project`

defines the project information, such as the project UUID, project variables, and the model repository URL for the project. It is recommended that you create only one observation in this data set.

`mm_jobs.emailaddr`

defines the e-mail addresses for the recipients of job status and the notification flags for alert and warning notifications.

`mm_jobs.reportdef`

defines the types of reports to create and the warning and alert conditions that are associated with those reports.

`mm_jobs.jobtime`

defines the date and time to run the reports and a label that describes the time performance data set period.

The code that you write to create the report specifications might need to be run only after it is created and only whenever it is modified. These data sets might not need to be created every time that you want to create reports.

Required libref

To create the report specifications, you need to define the following libref:

`mm_jobs`

defines the local path to the folder that contains the report job files.

Example: `libname mm_jobs "c:\mmReports\project1";`

Project Specifications

DATA Step `mm_jobs.project`

This DATA step defines the project specifications.

```

/*****
/* DATA step mm_jobs.project          */
/*                                     */
/* Create a data set to initialize the  */
/* performance monitoring batch program */
/* project specifications              */
*****/

```

```

DATA mm_jobs.project;
  length testDestination %150
         projectuuid $36
         projectpath $2000
         projectAlias $50
         precode $32000
         isActive $1
         notes $500;

  isActive='Y';

  /*****
  /* Specify the destination for the report      */
  /* output when the report is run in TEST mode */
  *****/

  testDestination='reportOutputPath';

  /*****
  /* Specify the project UUID                    */
  *****/

  projectuuid='projectuuid';

  /*****
  /* Map project specification variables to      */
  /* macro variables                            */
  *****/

  precode='
    %let _MM_EventProbVar=eventProbabilityVariable;
    %let _MM_TargetVar=targetVariable;
    %let _MM_TargetEvent=targeEventValue;
    %let _MM_ReportDataSrc=scoreIn.dataSetName;
    %let _MM_KeepVars=variablesToKeep;
    %let _MM_DropVars=variableToDrop;';

  /*****
  /* Specify the URL to the project in the model */
  /* repository and a description of the project */
  *****/

  projectPath='projectURL';
  projectAlias='alternateProjectName';

run;

```

Variable Descriptions for mm_jobs.project

The following variables are used in the mm_jobs.project DATA step:

isActive='Y | N'

specifies whether to enable the project definitions. Valid values are Y (yes) and N (no). Specifying N means that project files do not need to be removed from the local computer to deactivate a project entry. Enclose the value of isActive in quotation marks.

Interaction: Always set isActive='Y' when the data set mm_jobs.project has only one observation.

testDestination=*reportOutputPath*;

specifies the local path that contains the output files that are created when the %MM_RunReports() macro report mode macro variable _MM_ReportMode is set to TEST. Enclose the value of testDestination in quotation marks.

Example: testDestination='c:\mmReports\project1\testOutput';

See: [“Report Output in Test and Production Modes” on page 210](#)

projectuuid

specifies the universally unique identifier for a SAS Model Manager project. To obtain the project UUID, in the SAS Model Manager window, select the project. Expand System Properties. You can copy the UUID from the UUID property. projectuuid is used to redirect reporting job output data sets to the appropriate project folders in the model repository when the %MM_RunReports() macro is run in Production mode.

Note: If you copy the UUID from the SAS Model Manager window, you might need to remove leading text and spaces that are copied with the UUID.

precode=*macroVariableDefinitions*

specifies the macro variables that are used by the %MM_RunReports() macro. Enclose the value of the precode variable in quotation marks.

%let _MM_EventProbVar=*outputEventProbabilityVariable*;

specifies the output event probability variable name. To obtain the name, select the project in the Project Tree and expand **Specific Properties**. Use one of the values for the **Output Event Probability Variable** property list box.

%let _MM_TargetVar=*targetVariable*;

specifies the target variable name. To obtain the name, select the project in the Project Tree and expand **Specific Properties**. The target event variable is found in the property **Training Target Variable**. If a target variable is not specified, see your performance data set or the model for the name of the target variable.

%let _MM_TargetEvent=*targetEventValue*;

specifies the target event value. To obtain the name, select the project in the Project Tree and expand **Specific Properties**. The value is found in the property **Target Event Value**. If a target event value is not specified, see your performance data set or the model to determine the value.

Requirement: The value of _MM_TargetEvent must be an unformatted, raw value even if the original target variable has a SAS format applied to it.

%let _MM_ReportDataSrc=*scoreIn.dataSetName*;

specifies the libref and the data set name for the performance data set that is being analyzed.

If you process multiple data sets at one time, you can specify a generic data set name in this macro definition. The generic data set name is used to process all performance data sets. Before you run the %MM_RunReports() macro, you should create a DATA step with the name scoreIn.*genericDataSetName*, where the only statement in the DATA step is the SET statement that specifies the performance data set to process.

%let _MM_KeepVars=*variablesToKeep*;

specifies one or more output variables, separated by a space, that are kept in the performance data source to create the Stability report data set.

`%let _MM_DropVars=variablesToDrop;`
 specifies one or more input variables, separated by a space, that are dropped from the performance data source to create the Characteristic report data set.

`projectPath='projectURL'`
 specifies the project URL. To obtain the project URL, select the project in the Project Tree and select **System Properties**. You can copy the URL from the **URL** property. The project URL is used for information purposes only; it is not used to access project resources. *projectURL* is dynamically retrieved when the %MM_RunReports() macro runs. Enclose the value of projectPath in quotation marks.

Note: If you copy the URL from the SAS Model Manager window, you might need to remove leading text and spaces that are copied with the URL.

`projectAlias='alternateProjectName'`
 specifies an alternate project name. The alternate project name can be used to help identify the project when the projectPath is long. If you do not have an alternate project name, you can leave this variable blank. Enclose the value of projectAlias in quotation marks.

`notes='userNotes'`
 specifies any notes that the user want to add to the project specifications. Enclose the value of notes in quotation marks.

E-mail Recipient Specifications

DATA Step *mm_jobs.emailaddr*

This DATA step defines the e-mail recipient specifications:

```

/*****/
/* DATA mm_jobs.emailaddr                               */
/*                                                         */
/* Create a data set that specifies the e-mail             */
/* addresses of the users who will receive job             */
/* status notification as well as warnings and            */
/* alerts.                                                 */
/*****/

DATA mm_jobs.emailaddr;
  length address $50 sendAlertWarning sendJobStatus $1;
  address='e-mailAddress';
  sendAlertWarning='Y';
  sendJobStatus='N';
  output;
  address='e-mailAddress';
  sendAlertWarning='Y';
  sendJobStatus='Y';
  output;
run;

```

Variable Descriptions for *mm_jobs.emailaddr*

The following variables are used in the mm_jobs.emailaddr DATA step:

`address='e-mailAddress'`

specifies the e-mail address of the user to receive job, alert, and warning notices.
Enclose the value of address in quotation marks.

`sendAlertWarning='Y | N'`

specifies whether alert warning notifications are sent to the e-mail address specified in address. Valid values are Y (yes) and N (no). Enclose the value of sendAlertWarning in quotation marks.

`sendJobStatus='Y | N'`

specifies whether the job status report is sent to the e-mail address specified in address. Valid values are Y (yes) and N (no). Enclose Y or N in quotation marks.

Report Specifications

DATA Step `mm_jobs.reportdef`

This DATA step defines the type of reports to create, provides the macro syntax for the report type, and defines alert and warning specifications. You can specify one, two, or three report types in the DATA step. The %MM_RunReports() macro runs the reports that are defined in the mm_jobs.reportdef data set. For each type of report, assign the reportName, the macro, and alert and warning conditions.

```

/*****
/* DATA set mm_jobs.reportdef
/*
/* Create a data set that defines the report
/* metadata and alarm thresholds for the
/* Characteristic, Stability, and Model Assessment
/* reporting jobs.
*****/

```

```

DATA mm_jobs.reportdef;
    length reportName $20
    macro $1000
    alertCondition $200
    warningCondition $200
    isActive $1
    notes $500;

    isActive='Y';

/*****
/* Characteristic Report
*****/

reportName='Characteristic';
macro='
    %MM_UpdateCharacteristicTable(
        datasrc=&_MM_ReportDataSrc,
        dropVars=&_MM_DropVars;';

alertCondition='alertConditions';
warningCondition='warningConditions';
output;

```

```

/*****
/* Stability Report */
*****/

reportName='Stability';
macro='
%MM_UpdateStabilityTable(
    datasrc=&_MM_ReportDatasrc,
    keepVars=&_MM_KeepVars;';

alertCondition='alertConditions';
warningCondition='warningConditions';
output;

/*****
/* Model Assessment Report */
*****/

reportName='Model Assessment';
macro='
%MM_UpdateAssessmentTable(
    datasrc=&_MM_ReportDatasrc);';

alertCondition='alertConditions';
warningCondition='warningConditions';
output;
run;

```

Variable Descriptions for *mm_job.reportdef*

The following variable definitions are used in the *mm_jobs.reportdef* DATA step:

isActive

specifies whether to enable the report definitions. Valid values are Y (yes) and N (no). Specifying N means that a report definition file does not need to be removed from the local computer to deactivate a report definition entry.

Interaction: Always set *isActive*='Y' when the data set *mm_jobs.project* has only one observation.

reportName='reportName'

specifies the name of the report. The following are valid report types:

- Characteristic
- Stability
- Model Assessment

Enclose *reportName* in quotation marks. This argument is required.

macro='macroDefinition';

specifies the report macro that is executed when the %MM_RunReports() macro is executed. This argument is required.

alertConditions='alertConditions';

specifies an alert condition for the type of report. Enclose *alertConditions* in quotation marks. Here are example alert and warning conditions for each type of report:

Report Type	Example Alert Condition
Characteristic	alertCondition='p1>5 or p25>0';
Stability	alertCondition='outputDeviation > 0.03';
Model Assessment	alertCondition='lift5Decay>0.15 and lift10Decay>0.12) or giniDecay>0.1 or ksDecay>0.1';

See also: see [“Performance Index Warnings and Alerts”](#) on page 188.

warningConditions='warningConditions';
specifies a warning condition for the type of report. Enclose *warningConditions* in quotation marks.

Report Type	Example Warning Condition
Characteristic	warningCondition='p1>p2';
Stability	alertCondition='outputDeviation > 0.01';
Model Assessment	warningCondition='lift5Decay>0.05';

See also: see [“Performance Index Warnings and Alerts”](#) on page 188.

notes='userNotes';
specifies a note to add to the report definition data set. Enclose *userNotes* in quotation marks.

%MM_UpdateCharacteristicTable() Macro

Here is the syntax for the %MM_UpdateCharacteristicTable() macro:

```
%MM_UpdateCharacteristicTable(datasrc=&_MM_ReportDatasrc,
<dropvars=&_MM_DropVars>);
```

datasrc=&_MM_ReportDatasrc
specifies the macro variable that defines the performance data set which is used to create the Characteristic report.

dropvars=&_MM_DropVars
specifies the macro variable that defines the input variables to drop from the performance data set. Consider dropping variables from the performance data set whose values do not need to be monitored.

%MM_UpdateStabilityTable() Macro

Here is the syntax for the %MM_UpdateStabilityTable() macro:

```
%MM_UpdateStabilityTable(datasrc=&_MM_ReportDatasrc,
<keepvars=&_MM_KeepVars>);
```

datasrc=&_MM_ReportDatasrc
specifies the macro variable that defines the performance data set which is used to create the Stability report.

keepvars=&_MM_KeepVars

specifies the macro variable that defines the output variables to keep in the performance data set. Consider keeping only the variables in the performance data set whose values are to be monitored.

%MM_UpdateAssessmentTable() Macro

Here is the syntax for the %MM_UpdateAssessmentTable() macro:

%MM_UpdateAssessmentTable(datasrc=&_MM_ReportDatasrc);

datasrc=&_MM_ReportDatasrc

specifies the macro variable that defines the performance data set which is used to create the Model Assessment reports.

Job Scheduling Specifications

DATA Step mm_jobs.jobtime

This DATA step defines the dates and times that the data sets that underlie the performance monitoring reports are to be created or updated.

```

/*****
/* DATA step mm_jobs.jobtime                               */
/*                                                         */
/* Define the report schedule by specifying the             */
/* dates and times for each incremental reporting           */
/* interval. You can schedule as many jobs as you          */
/* would like. The following jobs are scheduled to         */
/* run one second before midnight on the dates             */
/* listed below.                                           */
*****/

DATA mm_jobs.jobtime;
    length scheduledTime $18 time $10;
    scheduledTime='dateTime';time='timePeriodLabel';output;
run;

```

Variable Descriptions for mm_jobs.jobtime

Here are the variables that are used in the DATA step mm_jobs.jobtime:

scheduledTime='dateTime'

specifies the date and time to run the report. The value of scheduledTime must be in the form *ddmmmyyyy:hh:mm:ss* where *dd* is a two-digit year, *mmm* is the first three letters of the month, *yyyy* is a four-digit year, *hh* is a two digit hour, *mm* is a two-digit minute, and *ss* is a two-digit second. Enclose *dateTime* in quotation marks.

The values of scheduledTime are used by the %MM_RunReports() macro, rather than by your job scheduler. Each time that the %MM_RunReports() macro runs, it checks the values of the scheduleTime variable. If the scheduled time has passed, the report runs. If it has not passed, the performance data sets are not created.

Example: **scheduledTime='03Jun2009:23:59:00';**

time='timePeriodLabel'

specifies a label that represents the time period for which the performance data was collected. Enclose *timePeriodLabel* in quotation marks. Use short and clear labels to create charts that can be easily read.

Example: `time='2008Q4';`

Example Code to Create the Report Specifications

This example creates a single SAS program to create the report specification data sets. After you copy the example code from the `sashelp.modelmgr` library, you providing values for the required variables and macros. The variable and macro names are highlighted in the example code to identify the values that you would modify to create the report specifications.

```
/* Source file name: sashelp.modelmgr.reportExample3.source */
```

```
LIBNAME mm_jobs 'c:\mm.test\report.auto';
```

```

/*****
/* DATA step mm_jobs.project          */
/*                                     */
/* Create a data set to initialize the  */
/* performance monitoring report batch  */
/* job project specification metadata and */
/* report precode metadata.            */
*****/

```

```

DATA mm_jobs.project;
  length testDestination $50
         projectuuid $36
         projectpath $200
         projectAlias $50
         precode $32000
         isActive $1
         notes $500;

```

```
  isActive='Y';
```

```

/*****
/* Specify the destination path for the report */
/* and the universal unique ID for the project */
*****/

```

```

testDestination=
  'c:\mm.test\report.test.output\project_123';
projectuuid=
  '8817ea06-0a28-0c10-0034-68f4ba396538';

```

```

/*****
/* The precode section uses macro variables to */
/* map individual model metadata components     */
/* to their respective variables, target event  */
/* values, and data used to create the report.  */
*****/

```

```

precode='
  %let _MM_EventProbVar=p_bad1;
  %let _MM_TargetVar=bad;
  %let _MM_TargetEvent=1;

```

```

%let _MM_ReportDataSrc=scoreIn.hmeq0;
%let _MM_KeepVars=p_bad1;
%let _MM_DropVars=bad job;
';

/*****
/* Specify the path to the project and provide */
/* an Alias name for the project reports.      */
*****/

projectPath=
'http://myserver:8300/ModelManager/MMRoot/demo/Creditcardpromotion';
projectAlias=
'credit risk for younger customers';
run;

/*****
/* DATA set mm_jobs.emailaddr                */
/*                                             */
/* Create a data set that specifies the e-mail */
/* recipient notification list, and whether to */
/* send the alert, warning, and job status     */
/* notifications.                             */
*****/

DATA mm_jobs.emailaddr;
  length address $50 sendAlertWarning sendJobStatus $1;
  address='recipient1@mail.com';
  sendAlertWarning='Y';
  sendJobStatus='N';
  output;
  address='recipient2@mail.com';
  sendAlertWarning='Y';
  sendJobStatus='Y';
  output;
run;

/*****
/* DATA set mm_jobs.reportdef                */
/*                                             */
/* Create a data set that defines the report   */
/* metadata and alarm thresholds for the      */
/* Characteristic, Stability, and Model Assessment */
/* reporting jobs.                            */
*****/

DATA mm_jobs.reportdef;
  length reportName $20
    macro $1000
    alertCondition $200
    warningCondition $200
    isActive $1
    notes $500;

  isActive='Y';

```

```

/*****
/* Characteristic Report */
*****/

reportName='Characteristic';
macro='
    %MM_UpdateCharacteristicTable(
        datasrc=&_MM_ReportDatasrc,
        dropVars=&_MM_DropVars;';

alertCondition='p1>5 or p25>0';
warningCondition='p1>2';
output;

/*****
/* Stability Report */
*****/

reportName='Stability';
macro='
    %MM_UpdateStabilityTable(
        datasrc=&_MM_ReportDatasrc,
        keepVars=&_MM_KeepVars;';

alertCondition='outputDeviation > 0.03';
warningCondition='outputDeviation > 0.01';
output;

/*****
/* Model Assessment Report */
*****/

reportName='Model Assessment';
macro='
    %MM_UpdateAssessmentTable(
        datasrc=&_MM_ReportDatasrc);';

alertCondition='
    (lift5Decay>0.15 and lift10Decay>0.12)
    or giniDecay>0.1
    or ksDecay>0.1';
warningCondition='lift5Decay>0.05';
output;
run;

/*****
/* DATA step mm_jobs.jobtime */
/* */
/* Define the report schedule by specifying the */
/* dates and times for each incremental reporting */
/* interval. The jobs below are scheduled to run */
/* one second before midnight on the dates listed */
/* below. */
/* */
/* For each scheduledTime variable you need a */
/* separate DATA step to execute whose SET */

```

```

/* statement names the appropriate performance */
/* data source. */
/*****

DATA mm_jobs.jobtime;
    length scheduledTime $18 Time $10;
    scheduledTime='01OCT2007:23:59:59';time='2007Q3';output;
    scheduledTime='01JAN2008:23:59:59';time='2007Q4';output;
    scheduledTime='01APR2008:23:59:59';time='2008Q1';output;
    scheduledTime='01JUL2008:23:59:59';time='2008Q2';output;
    scheduledTime='01OCT2008:23:59:59';time='2008Q3';output;

run;

```

See Also

- [“Extracting the Champion Model from a Channel” on page 222](#)
- [“SAS Code to Run Performance Reports” on page 225](#)

Extracting the Champion Model from a Channel

Using the %MM_GetModels() Macro

Before you run the %MM_RunReports() macro, you must extract the model from the publishing channel to a local computer. The model must have been published to the channel from the project folder. The %MM_GetModels() macro extracts models and auxiliary files from a SAS Publishing Framework SPK file to the local computer. All models that were published to the specified channel are included in the SPK file for a given SAS Model Manager project. If a model has been published multiple times over the channel, the latest model is used in the extraction. The macro then extracts the files from the SPK file to their respective folders on the local computer. The auxiliary files are extracted to the model folder and the model score code is extracted to a folder named \scorecode, which the macro creates as a subfolder of the model folder.

Note: You can run the %MM_GetModels() macro when no new model has been published to the channel for a SAS Model Manager project.

The auxiliary files include three SAS data sets:

- current.sas7bdat contains project and model metadata
- logs.sas7bdat contains the SAS logs that were created during the model extraction process
- processingspk.sas7bdat contains information that is necessary to process the SPK file.

The models in the \scorecode folder are named using the project UUID as the model folder name. The %MM_RunReports() macro uses the mm_jobs.project data set to determine the project UUID. The project UUID is then used as the name of the model on the local computer for scoring when the performance monitoring reports are created.

The current data set contains project and model information and is used by the %MM_RunReports() macro. To ensure that the %MM_RunReports() macro is using the most current project and model metadata, always run the %MM_GetModels() macro before

you run the %MM_RunReports() macro. For a list of the information that is contained in the current data set, see “The current.sas7bdat Data Set” on page 223.

Accessing SAS Model Manager Report Macros

The %MM_RunReports() macro, the %MM_GetModel() macro, and all other SAS Model Manager macros are available in the catalog sashelp.modelmgr.reportmacros.source. Use the following FILENAME statement to make these macros available to your program:

```
filename repmacro catalog 'sashelp.modelmgr.reportmacros.source';
%inc repmacro;
```

%MM_GetModels() Macro Syntax

Here is the syntax for the %MM_GetMacros() macro:

```
%MM_GetModels(channel=channelPath
localPath=localModelPath);
```

channel=channelPath

specifies the path of the channel to extract the models from. To obtain the channel path, open SAS Management Console. Click the **Plug-ins** tab, and expand the following nodes under **SAS Management Console** to find the publishing channels that are used by SAS Model Manager: **Environment Management** ⇒ **Publishing Framework** ⇒ **Foundation** ⇒ **Channels** ⇒ **Model Manager Channels**. Select a channel. Right-click **Properties**. The channel path is located in the **Persistent Store** tab. Do not enclose the value of channel in quotation marks.

Note: The %MM_GetModels() macro supports only publishing channels that have a persistent store type of **Archive**.

localPath=localModelPath

specifies a folder on the local computer to where the model and auxiliary files are extracted from the SPK file. Do not enclose *localModelPath* in quotation marks.

Example Program to Extract a Model from a Channel

The following SAS code uses the %MM_GetModel macro to extract a champion model from a channel.

```
/* Source file name: sashelp.modelmgr.reportExample1.source */

FILENAME mmmac
    catalog 'sashelp.modelmgr.reportmacros.source';
%inc mmmac;

%MM_GetModels(
    channel=\\network1\MMChampion\channel1,
    localPath=c:\mm.test\model.extraction);
```

The current.sas7bdat Data Set

When models are extracted from a publishing channel, the current.sas7bdat data set contains the following information for each model:

Variable Name for the Project or Model Information	Description
algorithm	The algorithm that was used to create the model.
fileName	Not used
isChampionModel	True or False to indicate whether the model is the champion model.
keyWords	Keywords
miningFunction	The type of mining function, such as classification, prediction, segmentation
model	Not used
modeler	The name of the person who created the model.
modelName	The name of the model.
modelProductionTimestamp	The time that the model was declared as a production model.
modelTool	The name of the tool that was used to train the model.
modelUUID	The UUID for the model.
nodeDescription	Not used
projectPath	The project URL.
project UUID	The UUID for the project.
repository	The repository URL.
scoreCodeType	DATA Step or SAS Program
subject	The subject name.
targetName	The Training Target Variable name
userAttr	User-defined attributes, such as "MODELER='sasquest' MODELPROJECTVARMAP='predictedProbability eq P_BAD1; predictedClass eq I_BAD;' "
versionName	The name of the version that contains the model.
whenPublished	The date and time when the project or model was published to the channel.
whoPublished	The SAS Model Manager user who published the model.

See Also

- [“Define the Report Specifications” on page 211](#)
- [“SAS Code to Run Performance Reports” on page 225](#)

SAS Code to Run Performance Reports

Overview of the SAS Code to Run the Performance Reports

After you have created the data sets that define the report specifications and have extracted the model from the publishing channel, you then run the %MM_RunReports() macro to create the reports for one or more time periods. Using the data sets that were created to define the report specifications, the %MM_RunReports() macro uses the report specifications to create the reports. The report specifications include the type of report to create, such as characteristic, stability, or model assessment. Other report specifications include the target variable, the libref and data set name that is used as the performance data source, variables to keep and drop from reports, e-mail addresses to send report notifications, and performance index warnings and alerts.

To run the %MM_RunReports() macro, your code must accomplish the following tasks:

- access the reporting macros
- define the librefs and the macro variables that are required by the %MM_RunReports() macro
- specify the performance data set to process. To do this, execute a DATA step before each %MM_RunReports() macro.

To ensure that you have the latest model, extract the model from the channel each time that you create the performance reports. For this reason you could combine into one SAS program the extraction process and the code to run the reports.

If you run a set of batch jobs every night, you could include this batch job with that set of batch jobs. The reports would be created only after the scheduled date and time that is specified in the mm_jobs.jobtime data set.

The following sections describe each of these components of your SAS program. The last section is an example of a program that is used to test the %MM_RunReports() macro.

Accessing SAS Model Manager Report Macros

The %MM_RunReports() macro, the %MM_GetModel() macro, and all other SAS Model Manager macros are available in the catalog sashelp.modelmgr.reportmacros.source. Use the following FILENAME statement to make these macros available to your program:

```
filename repmacro catalog 'sashelp.modelmgr.reportmacros.source';
%inc repmacro;
```

Required Librefs

The following librefs are required in your report monitoring program:

mm_jobs

defines the local path to the folder that contains the report job files.

Example: `libname mm_jobs "c:\mmReports\project1";`

`mm_meta`

defines the local path to the folder that stores the data sets that are created from running the %MM_GetModels() macro. The value of this libref must have the same value as the localPath argument for the %MM_GetModels() macro.

Example: `libname mm_meta "c:\mmReports\project1\model";`

`scoreIn`

specifies a user-defined libref that points to the local path that contains the performance data sources.

Interaction: You can use this libref when you set the value of SAS Model Manager macro variables, such as `_MM_ReportDataSrc`, in the precode variable of the `mm_jobs.project` data set. For example, `%let _MM_ReportDataSrc=scoreIn.foo`.

Example: `libname scoreIn "c:\mmReports\project1\perfdatasets";`

Macro Variables to Define Report Local Folders and Data Sets

Define the following macro variables in your report monitoring program. Then define the location of the job and model on the local computer:

`_MM_JobLocalPath`

specifies the path on the local computer that contains the root folder for the reporting files of a given SAS Model Manager project.

Example: `%let _MM_JobLocalPath=c:\mmReports\project1;`

`_MM_ModelLocalPath`

specifies the path on the local computer that contains the model after it has been extracted from the SAS Metadata Repository.

Example: `%let _MM_ModelLocalPath=c:\mmReports\project1\model;`

`mapTable`

specifies a libref and data set in the form *libref.dataSet* that contains the mapping of the project output variables to the model output variables. When the model is extracted from the channel, the data set `current.sas7bdat` is extracted to the folder that contains the model. Use this data set as the value of `mapTable`.

Example: `mapTable=mm_meta.current`. The data set name `current` is arbitrary. It is recommended that you use the name `current`.

Macro Variables That Are Used by the %MM_RunReports() Macro

Required Macro Variables

The following macro variables are required to run the %MM_RunReports() macro:

`_MM_ServerName`

specifies the server domain name for the server that hosts the model repository.

Example: `%let _MM_ServerName=myserver.com;`

`_MM_PortNumber`

specifies the port number for the server that hosts the model repository.

Example: `%let _MM_PortNumber=6411;`

_MM_RepositoryId

specifies the UUID for the SAS Model Manager repository. To obtain the UUID, select MMRoot from the Project Tree. Expand the System Properties. The first property is the repository UUID.

Example: `%let _MM_RepositoryId=ModelManagerDefaultRepoUUID;`

_MM_User

specifies a valid SAS Model Manager user.

_MM_Password

specifies the password for the SAS Model Manager user who is identified in the _MM_User macro variable.

See: [“Encoding SAS Model Manager User Passwords” on page 227](#)

Optional Macro Variable

The example programs use the following global macro variable, which you might find useful in your report monitoring program:

_MM_ReportMode

specifies the mode to run the %MM_RunReports() macro. Valid values are TEST and PRODUCTION. The default value is PRODUCTION. You might want to use a value of TEST while you are testing your program. When the value is TEST, the report output files are written to the local computer. When the value is PRODUCTION, the report output files are written to the appropriate project folders in the SAS Model Manager model repository.

Interaction: If _MM_ReportMode is set to TEST, you must supply a value for the testDestination variable in the mm_jobs.project data set.

Example: `%let _MM_ReportMode=TEST;`

Encoding SAS Model Manager User Passwords

Each time that you run a SAS program to be processed by SAS Model Manager, you specify a SAS Model Manager user ID and assign the user's password to the global macro variable _MM_Password. In order to not store passwords in clear text, you can use the PWENCODE procedure to encode a password and store it in a file, in a network-accessible directory. Then, in your SAS program, you create a fileref to the network file that contains the encoded password and you use a DATA step to assign the encoded password to the _MM_Password global macro variable.

In a separate SAS program, encode your password:

```
filename pwfile "my-network-path\pwfile";

proc pwencode in="12345" out=pwfile;
run;
```

In your SAS Model Manager program, use a DATA step to access the encoded password file:

```
filename pwfile "my-network-path\pwfile";
%let _MM_User=mmuser1;
data _null_;
  infile pwfile obs=1 length=1;
  input @;
  input @1 line $varying1024. 1;
  call symput('_MM_Password', substr(line,1,1));
run;
```

The DATA Step to Access the Performance Data Set

You use a DATA step to access the performance data set before you run the %MM_RunReports() macro:

```
DATA libref.dataStepName;
    set libref.performanceDataSetName;
run;
```

Here is an example of a DATA step to access the performance data set:

```
DATA scoreIn.hmeq;
    set scoreIn.hmeq_2008q4;
run;
```

The %MM_RunReports() Macro

Description of the %MM_RunReports() Macro

You use the %MM_RunReports() macro to create or update the data sets that underlie the performance monitoring reports. Before each %MM_RunReports() macro that you specify in your program, you might want to update the performance data set by including a DATA step that accesses the performance data set input file.

The %MM_RunReports() macro uses the data sets that are stored in the library that is specified by the mm_jobs libref. These data sets define the report specifications and are the data sets that are created in the report specification program. For more information about the report specification program, see [“Define the Report Specifications” on page 211](#).

Syntax

Use the following syntax for the %MM_RunReports() macro:

```
%MM_RunReportss(localPath=&_MM_JobLocalPath, mapTable=&mapTable,
user=&_MM_User, password=&_MM_Password, <currentTime=&currentTime>);
```

Syntax Description

localPath=&_MM_ModelLocalPath

specifies the path on the local computer to the location where the %MM_GetModels() macro stores the files extracted from the channel. The %MM_RunReports() macro retrieves the score code from the scorecode folder, which is a subfolder of &_MM_ModelLocalPath.

Example: **localPath=&_MM_ModelLocalPath**

mapTable=&mapTable

specifies the name of the data set that contains metadata about the extracted model. mapTable is the data set named current.sas7bdat that is created when the model is extracted using the %MM_GetModels() macro. No modification of this argument is necessary.

Example: **mapTable=&mapTable**

user=&_MM_User

specifies a valid SAS Model Manager user. Use the macro variable that defines the valid SAS Model Manager user.

Example: `user=&_MM_User`

`password=&_MM_Password`

specifies the password for `_MM_User`. Use the `_MM_Password` global macro variable that defines the password for the SAS Model Manager user. The value of `_MM_Password` is a text string.

Example: `password=&_MM_Password`

See: “[Encoding SAS Model Manager User Passwords](#)” on page 227

`currentTime=currentTime`

specifies a time to use for the current time. Use this argument for testing the `%MM_RunReports()` macro. You do not need to specify an argument for `currentTime` when you run the macro in a production environment, where the system timestamp is used as a value for `currentTime`.

The value of `currentTime` must be in the form `ddmmmyyyy:hh:mm:ss` where `dd` is a two-digit year, `mmm` is the first three letters of the month, `yyyy` is a four-digit year, `hh` is a two digit hour, `mm` is a two-digit minute, and `ss` is a two-digit second.

Example: `currentTime=03Oct2009:12:15:30`

Example %MM_RunReports() Macro

The following code is an example of using the `%MM_RunReports()` macro:

```
%MM_RunReports (
  localPath=&_MM_ModelLocalPath,
  mapTable=&mapTable,
  user=&_MM_User,
  password=&_MM_Password);
```

Example Code to Run the Reports

The following example program defines the librefs and macro variables to test the `%MM_RunReports()` macro's ability to assess home equity performance data for multiple time periods. Before this section of code can be run, the report specifications must be defined in SAS data sets and the model must be extracted from the publishing channel. For more information, see “[Define the Report Specifications](#)” on page 211 and “[Extracting the Champion Model from a Channel](#)” on page 222.

The example program sets the current time to a time that would trigger the creation of data sets or the updating of data sets that underlie the model monitoring reports. When you run your batch program in a production environment, you do not need a variable to set the current time. When no value is set for the current time, the `%MM_RunReports()` macro uses the system timestamp as the value of the current time variable.

The highlighted values are user-supplied values.

```
/* Source file name: sashelp.modelmgr.reportExample4.source */

FILENAME repmacro catalog 'sashelp.modelmgr.reportmacros.source';
%inc repmacro;

/* Fileref to the encoded password */

FILENAME pwfile "my-network-path\pwfile";

/*****/
```

```

/* Specify the report execution metadata and      */
/* configure the _MM_ macro variables to run the */
/* report job in TEST mode.                      */
/*****/

%let _MM_ReportMode=TEST;
%let _MM_User=mmuser1;
data _null_;
    infile pwfile obs=1 length=1;
    input @;
    input @1 line $varying1024. 1;
    call symput('_MM_Password',substr(line,1,1));
run;
;

%let _MM_ServerName=myserver.com;
%let _MM_PortNumber=6411;
%let _MM_RepositoryId=ModelManagerDefaultRepo;
%let _MM_PathMayChange=Y;

%let _MM_JobLocalPath=c:\mm.test\report.auto;
%let _MM_ModelLocalPath=c:\mm.test\model.extraction;

LIBNAME mm_jobs "&_MM_JobLocalPath";
LIBNAME mm_meta "&_MM_ModelLocalPath";
LIBNAME scoreIn 'c:\mm.test\score.in';

%let mapTable=mm_meta.current;

/*****/
/* DATA step scoreIn.hmeq0                      */
/*                                              */
/* First, run the 2007Q3 report. It is necessary to */
/* artificially declare a "currentTime" argument  */
/* of 03Oct2007 in order to trigger the report     */
/* execution scheduled for the 2007Q3 interval.    */
/*****/

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2007q3;
run;

%let currentTime=03Oct2007:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,
    user=&_MM_User,
    password=&_MM_Password);

/*****/
/* Now, run the 2007Q4 report. It is necessary to */
/* artificially declare a "currentTime" argument  */
/* of 03Jan2008 in order to trigger the report     */
/* execution scheduled for the 2007Q4 interval.    */
/*****/

```

```

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2007q4;
run;

%let currentTime=03Jan2008:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,
    user=&_MM_User,
    password=&_MM_Password);

/*****
/* Now, run the 2008Q1 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Apr2008 in order to trigger the report */
/* execution scheduled for the 2008Q1 interval. */
*****/

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2008q1;
run;

%let currentTime=03Apr2008:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,
    user=&_MM_User,
    password=&_MM_Password);

/*****
/* Now, run the 2008Q2 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Jul2008 in order to trigger the report */
/* execution scheduled for the 2008Q2 interval. */
*****/

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2008q2;
run;

%let currentTime=03Jul2008:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,
    user=&_MM_User,
    password=&_MM_Password);

/*****
/* Now, run the 2008Q3 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Oct2008 in order to trigger the report */
/* execution scheduled for the 2008Q3 interval. */
*****/

```

```

/*****

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2008q3;
run;

%let currentTime=03Oct2008:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,
    user=&_MM_User,
    password=&_MM_Password);

```

See Also

- [“Define the Report Specifications” on page 211](#)
- [“Extracting the Champion Model from a Channel” on page 222](#)

Part 6

Appendixes

<i>Appendix 1</i>	
Query Utility	235
<i>Appendix 2</i>	
SAS Model Manager Access Macros	243
<i>Appendix 3</i>	
Properties	283
<i>Appendix 4</i>	
SAS Model Manager In-Database Scoring for Teradata	297

Appendix 1

Query Utility

Overview of the Query Utility	235
Search for Models	236
Search by Using a UUID	238
Search Life Cycles for Tasks Assigned to Users	239

Overview of the Query Utility

Using the Query utility you can search for models based on certain criteria, SAS Model Manager project, version or model components, and tasks that are assigned to users. You can perform a query from an organizational folder, a project folder, or a version folder.

The Query utility has three tabs that you can use to enter search criteria, depending on the type of search that you want to perform:

- Use the **Model** tab to search for models based on criteria such as name, model algorithm, or variable name.
- Use the **Component** tab to search for project folders, version folders, or models when you know the UUID of the component. This is helpful when a model is published to a channel outside of Model Manager. You can then use the UUID to search for the model in SAS Model Manager.
- Use the **Life Cycle** tab to search for tasks that are assigned to users or tasks that users are assigned to approve.

You begin a query in the Project Tree by selecting the **MMRoot** folder, an organizational folder, a project folder, or a version folder. The Query utility searches the selected folder and all sub-folders and subcomponents. If you are searching for a model using the **Model** tab, the values in the search criteria list boxes depend on what folder you select to begin your search. A list box contains values for models in the selected folder and all sub-folders and subcomponents.

SAS Model Manager returns the search results in a table shown below the search criteria. The **Path** column in the search results table contains a URL that you can use to determine the path to the component in the Project Tree. Here is an example URL:

```
http://SMMserver:8808/SASContentServer/repository/default/ModelManager
/MMRoot/HomeEquity/2009Q3/Models/Model%1
```

MMRoot is the top node in the Project Tree. From the Project Tree, expand the project folder **HomeEquity**, expand the version folder **2009Q3**, and expand the **Models** folder. Model 1 is in the **Models** folder.

If no results are found, SAS Model Manager issues a message that informs you that there are no entries in the Project Tree for the search criteria.

To search the Project Tree, use the following instructions:

- “Search for Models” on page 236
- “Search by Using a UUID” on page 238
- “Search Life Cycles for Tasks Assigned to Users” on page 239

Search for Models

To search for models, follow these steps:

1. From the Project Tree, right-click **MMRoot**, an organizational folder, a project folder, or a version folder, and select **Query**. The Query utility opens.
2. Enter the search criteria that are listed here along with their explanations.

Name

Enter the name of a model.

Algorithm

Select an algorithm from the list box. The list box lists all algorithms for models that have been imported to SAS Model Manager.

Input Variables

Select an input variable from the list box.

Target Variables

Select a target variable from the list box.

Model Creator

Enter the name of the person who imported the model.

User Defined Key

Enter a user-defined property name. If you specify a value in this field, you must specify a value in the User Defined Value field.

User Defined Value

Enter a user-defined property value. If you specify a value in this field, you must specify a value in the User Defined Key field.

TIP To deselect a value from a list box, select the blank line at the bottom of the list.

Query

Enter query values.

Model

Component

Life Cycle

Name:

Algorithm:

DecisionTree

Input Variables:

Target Variables :

Model Creator:

User Defined Key:

User Defined Value:

Search Results:

Name	Path	Algor...	Type	Model...
HMEQ Lo...	//ModelM...	DecisionT...	Predictio...	

Find

OK

Cancel

3. Click **Find**.

The search results display the following information:

Column	Description
Name	Specifies the name of the model.
Path	Specifies the URL of the model in the Project Tree. Use the URL to locate the model in the Project Tree, following the path from MMRoot . For example, using the URL <code>http://SMMserver:8080/SASContentServer/repository/default/ModelManager/MMRoot/HomeEquity/2007Q3/Model/Model%201</code> , you can find the model Model 1 in the project HomeEquity , the version 2007Q3 , and the Models folder.
Algorithm	Specifies the name of the algorithm, such as regression or logistic, that is used by the model.

Column	Description
Type	Specifies one of the model function types: <ul style="list-style-type: none"> AnalyticalModel ClassificationModel PredictionModel ClusteringModel
Model Creator	Specifies the user that created or imported the model, depending on the model import method.

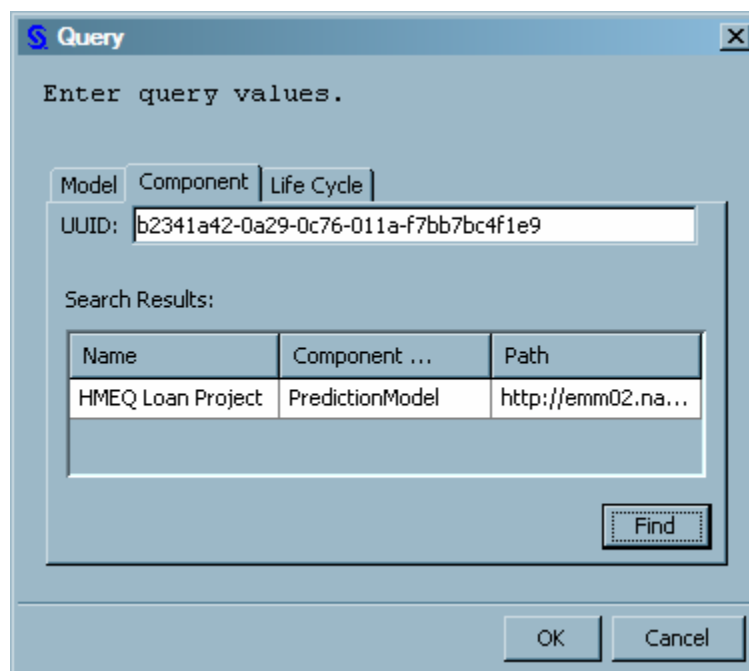
See Also

- “Search Life Cycles for Tasks Assigned to Users” on page 239
- “Search by Using a UUID” on page 238

Search by Using a UUID

To search for organizational folders, projects, versions, or models by using a UUID, follow these steps:

1. From the Project Tree, right-click **MMRoot**, an organizational folder, a project folder, or a version folder, and select Query. The Query utility opens.
2. Select the **Component** tab.
3. Enter the UUID. You can copy and paste the UUID into the **UUID** field. Be sure you do not include leading blank spaces or undesired text that is not part of the UUID.



4. Click Find. The **Search Results** display below the **UUID** field.

The search results displays the following information :

Column	Description
Name	Specifies the name of the project, version or model.
Component Type	Specifies one of following component types: AnalyticalModel the component is an analytical model ClassificationModel the component is a classification model ClusteringModel the component is a clustering or segmentation model ModelGroup the component is an organizational folder PredictionModel the component is a prediction model Project the component is a project Version the component is a version
Path	Specifies the URL of the component in the Project Tree. Use the URL to locate the task in the Project Tree, following the path from MMRoot . For example, using the URL <code>http://SMMserver:8080/SASContentServer/repository/default/ModelManager/MMRoot/HomeEquity/2009Q3</code> you can find the version in the project HomeEquity .

See Also

- [“Search for Models” on page 236](#)
- [“Search Life Cycles for Tasks Assigned to Users” on page 239](#)

Search Life Cycles for Tasks Assigned to Users

To search for tasks that are assigned to a user, follow these steps:

1. From the Project Tree, right-click **MMRoot**, an organizational folder, project folder, or version folder, and select **Query**. The Query utility opens.
2. Click the **Life Cycle** tab.
3. Click in the **User** field and select a user from the list box.

Enter query values.

Model Component **Life Cycle**

User: mdlmgrexampleapprovers

Assignee:

N...	P...	V...	M...	S...	P...
Sign-off	Test	2008	Devel...	Not S...	http:/...
Sign-off	Home...	2007	Produ...	Not S...	http:/...

Approver:

N...	P...	V...	M...	S...	P...
End P...	HMEQ...	2008	Retire	Not S...	http:/...
Decla...	HMEQ...	2008	Produ...	Not S...	http:/...

Find

OK Cancel

4. Click **Find**.

The search results display tasks in the **Assignee** results that are assigned to the user and tasks in the **Approver** results that the user is assigned to approve. The **Assignee** query results return only the tasks that have a status of **Started** or **Not Started**. Results that have a status of **Complete** or **Approved** are not returned. The **Approver** query results return tasks that have a status of **Started**, **Not Started**, and **Completed**.

The search results display the following information for tasks where the user is designated as an **Assignee** and an **Approver**:

Column	Description
Name	Specifies the name of the task.
Project	Specifies the project name for which the task must be completed.
Version	Specifies the version name for which the task must be completed.
Milestone	Specifies the milestone for which the task must be completed.
Status	Specifies the state of the task at the time of the query. Values for Status can be Not Started or Started .

Column	Description
Path	Specifies the URL of the task in the Project Tree. Use the URL to locate the task in the Project Tree, following the path from MMRoot . For example, using the URL <code>http://SMMserver:8080/SASContentServer/repository/default/ModelManager/MMRoot/HomeEQ/2008Q2/Mortgages/Testing/Signoff</code> , you can find the user in the project HomeEQ , the version 2008Q2 , and the life cycle Mortgages .

See Also

- [“Search for Models” on page 236](#)
- [“Search by Using a UUID” on page 238](#)

Appendix 2

SAS Model Manager Access Macros

Overview of SAS Model Manager Access Macros	243
Using the SAS Model Manager Access Macros	244
Accessing the SAS Model Manager Macros	244
Identifying SAS Model Manager Model Repository Objects	244
Identifying Files Used by Access Macros	245
Required Tables	245
SAS Model Manager Global Macro Variables	247
Required Global Macro Variables	248
Dictionary of SAS Model Manager Access Macros	249
%MM_AddModelFile Macro	249
%MM_GetModelFile Macro	252
%MM_GetURL Macro	256
%MM_Register Macro	257
%MM_RegisterByFolder Macro	274
%MM_CreateModelDataset Macro	279

Overview of SAS Model Manager Access Macros

The SAS Model Manager access macros provide a way to use SAS code to perform basic operations on a SAS Model Manager repository. The SAS Model Manager access macros are a combination of SAS macros and Java libraries. The SAS Model Manager access macros and Java libraries are delivered with the SAS Model Manager software.

Here is a list of the SAS Model Manager access macros:

- %MM_AddModelFile adds a model component file to a model that is already registered with SAS Model Manager.
- %MM_GetModelFile retrieves a model from the model repository and saves it to a specified destination.
- %MM_GetURL retrieves the SAS Model Manager path to an object in the model repository and saves it in the global macro variable _MM_URL.
- %MM_Register registers a model in the SAS Model Manager model repository. You can use the %MM_Register macro in the same SAS program that you create models using SAS Enterprise Miner to register the model for use with SAS Model Manager in the same program.

- %MM_RegisterByFolder registers multiple models simultaneously to the SAS Model Manager model repository. Model files for a single model are contained in a subdirectory, and all subdirectories have the same parent directory.
- %MM_CreateModelDataset creates a data set that contains information for all models in a specified folder. Model information can be retrieved in a data set for all models in MMRoot, an organizational folder, a project, a version, and a single model.

To use the SAS Model Manager access macros, you can structure your SAS program as follows:

- Create a fileref to the SAS Model Manager access macro catalog and include that fileref, using the %INCLUDE statement.
- Use the SAS Model Manager global macro variables to define the server, the server port, a valid SAS Model Manager user and password.
- Define the SAS Model Manager _MM_RepositoryId global macro variable.
- Set up librefs to access the necessary directories and filerefs to access the necessary files.
- Set up macro variables as necessary.
- Execute the macro.
- Check for successful completion.

Using the SAS Model Manager Access Macros

Accessing the SAS Model Manager Macros

Before you can use the access macros, your SAS program must access the catalog where the macros are located and load the macros into memory. Here is example code to do this:

```

/*****
/* Specify the macro code location          */
/*****

Filename MMAccess catalog "sashelp.modelmgr.accessmacros.source";

/*****
/* Load the Access macros                  */
/*****

%include MMAccess;

```

Identifying SAS Model Manager Model Repository Objects

The access macros use a SAS Model Manager identifier to specify a unique object such as the version or a model, in the SAS Model Manager model repository. The identifier can be in the form of a Universal Unique Identifier (UUID) or a SAS Model Manager path.

- A UUID is a case sensitive, 36-character string that uniquely identifies the repository object. An example UUID is cca1ab08-0a28-0e97-0051-0e3991080867.

If you need to find the UUID or the exact SAS Model Manager path for an object, you can look it up in the SAS Model Manager Project Tree. Select the object and then expand the **System Properties** in the Properties view. The UUID and path values are listed there.

- The format for a SAS Model Manager path is *//repositoryID/MMRoot/folder/project/version/Models/model*

The name of *repositoryID* is defined during installation. The names of the folder, project, version, and model that follow in the path are user-defined. SAS Model Manager path specifications always use the forward slash character (/) as a separator.

For example, a version path might look like *//MMModelRepository/MMRoot/HomeEquity/HMEQ/2009*.

You use the `_MM_CId` global macro variable to pass a model repository identifier to an access macro. For more information, see “[_MM_CId](#)” on page 247.

Identifying Files Used by Access Macros

All SAS Model Manager access macros that accept SAS file references require the file references to point to a single physical file. File references in the form *libref.filename* must resolve to a single physical file. Specific logical library references in the form *libref* must resolve to a directory or a folder.

Concatenated library references cannot be used.

Here is a list of libraries to which you must assign a libref in your SAS programs:

- the directory that contains your model files
- the directory that contains the training data
- the directory that contains your input, output, and target data sets

SAS Model Manager macros use the libref `SMMMODEL` to access model component files, as in this example:

```
libname smmmodel "c:\myModel\HMEQ\scorecode";
```

You can define the libref `SMMMODEL` at the beginning of your SAS program and use it to access model component files in any of the SAS Model Manager access macros that your program executes.

Here is a list of files that you can identify with a fileref in your SAS programs:

- a catalog fileref to the SAS Model Manager access macro code
- the source path and filename for a single file to be registered by the `%MM_AddModelFile` macro
- the source path and filename for a SAS Enterprise Miner package file to be registered by the `%MM_Register` macro
- the destination path and filename for the `%MM_GetModelFile` macro

Required Tables

Whether you use the SAS Model Manager window or the access macros, SAS Model Manager must know the model input variables, the output variables, and the target variables to register a model. SAS Model Manager uses an XML file to describe each of these types of files. Before you can register a SAS code model, you must create a SAS data sets that represents the input, output, and target variables:

- The model input table contains the variables that are used as input by the model. During model registration, SAS Model Manager uses this table to create the inputvar.xml file.
- The model output table is a table whose variables contain the model output values. During model registration, SAS Model Manager uses this table to create the outputvar.xml file.
- The model target variable table is a table whose one variable is the target variable that is used in the training data. During model registration, SAS Model Manager uses this file to create the targetvar.xml file.

Each of these tables can be a one-row table. The tables' purpose is to define and describe the variables that are used by the model.

You can create each of these tables using the training data that you used to train your model. The following example SAS program uses the training data to create all three tables:

```

/*****
/* Set the location for the model tables */
*****/

libname hmeqtabl "c:\myModel\hmeq\tables";

/*****
/* DATA step to create the target variable table. */
/* Because there is only one target variable, keep only */
/* that variable. */
*****/

data hmeqtabl.target;
    set hmeqtabl.training(obs=1);
    keep bad;
run;

/*****
/* DATA step to create the input variable table. */
/* Keep only the variables used for input by the model. */
*****/

data hmeqtabl.invars;
    set hmeqtabl.training (obs=1);
    keep debtinc delinq derog job loan mortdue ning reason value yoj;
run;

/*****
/* DATA step to create the output variable table. */
/* Keep only the variables used for output by the model.*/
/* Include the score code to get the output variables. */
*****/

data hmeqtabl.outvars;
    set hmeqtabl.training;
    %include "c:\myModel\hmeq\score.sas"
    keep f_bad i_bad p_0 p_1;
run;

```

SAS Model Manager Global Macro Variables

Your SAS program and SAS Model Manager use global macro variables to pass information about the server and the SAS Model Manager model repository to the access macros. Some macros set these global macro variables. You can set any of these global macro variables in your SAS program. At the end of each macro execution, the global macro variable `_MM_RC` is set to a number that indicates either that the macro executed successfully or that there was an error.

Here is a description of the SAS Model Manager global macro variables:

`_MM_CId`

contains the name of the current SAS Model Manager object identifier. `_MM_CId` is either the URL or the SAS Model Manager path to the object in the model repository. You can use the `%MM_GetURL` to obtain a URL for any object in the SAS Model Manager repository.

The `%MM_Register` macro sets `_MM_CId` to contain the SAS Model Manager identifier for the registered model. The `%MM_AddModelFile` macros sets `_MM_CId` to the SAS Model Manager identifier for the model to which the file was added.

`_MM_Password`

contains a password for the SAS Model Manager user. If you do not encode the password using the `PWENCODE` procedure, the password is printed in the SAS log.

See: [“Encoding SAS Model Manager User Passwords” on page 227](#)

`_MM_PortNumber`

contains the port number that the SAS Analytics Platform server listens for SAS Model Manager requests.

Default: 6411

`_MM_RC`

contains one of the following return codes after processing a SAS Model Manager macro:

<code>_MM_RC</code> Return Value	Access Macro Status
0	All OK
1	Macro parameter error
2	Macro parameter processing error
3	Repository login failed
4	Repository operation failed
5	Generic critical Java error
6	Generic DATA step error

`_MM_RepositoryId`

contains the name of the SAS Metadata Repository.

Example: %let _MM_RepositoryId=ModelManagerDefaultRepo

_MM_ResourceURL

contains the URL of the **Resources** folder. the _MM_Resource URL is set by the %MM_GetURL macro when the macro returns a version URL in the _MM_URL global macro variable.

_MM_ServerName

contains the name of the network server where the SAS Analytics Platform is installed.

Example: %let _MM_ServerName=myserver.com

_MM_URL

contains a URL for a SAS Model Manager object. The %MM_GetURL macro returns a URL in the _MM_URL global macro variable.

_MM_User

contains the name of a SAS Model Manager user on the server that is specified by the _MM_ServerName global macro variable.

Default: the value of SAS automatic macro variable &SYSUSERID.

Required Global Macro Variables

When you use the access macros, the macros need to know the following information:

- how to access the server where the SAS Analytics Platform is installed
- a user and password for processing requests to SAS Model Manager
- the URL or path to the SAS Model Manager repository

Be sure that your SAS program defines values for these macro variables when you use the access macros:

- _MM_ServerName
- _MM_PortNumber
- _MM_User
- _MM_Password

To secure the Model Manager user password, encode the password using the PWENCODE procedure and save it in a file on the network. You can then use a fileref to access the password file and a DATA step to assign the password to the _MM_Password global macro variable. For more information, see [“Encoding SAS Model Manager User Passwords” on page 227](#).

For a description of these macro variables as well as their default values, see [“SAS Model Manager Global Macro Variables” on page 247](#).

Here is a code example that uses the four macro variables to describe how to the access to the server for the SAS Analytics Platform.

```
Filename pwfile "my-network-drive\pwfile";
```

```
%let _MM_ServerName = DMS9.loc.co.com;
%let _MM_PortNumber = 6411;
%let _MM_User = miller;
data _null_;
  infile pwfile obs=1 length=1;
  input @;
  input @1 line $varying1024. 1;
```

```
call symput('_MM_Password', substr(line,1,1));
run;
```

When you use a UUID as an identifier, you must also name the SAS Model Manager model repository. Here is a code example that sets the `_MM_RepositoryId` global macro variable.:

```
%let _MM_RepositoryId = MMModelRepository;
```

Dictionary of SAS Model Manager Access Macros

%MM_AddModelFile Macro

Overview of the %MM_AddModelFile Macro

You use the %MM_AddModelFile macro to add model component files to an existing SAS Model Manager model.

- “Syntax” on page 249
- “Arguments” on page 249
- “Details” on page 250
- “Example” on page 251

Syntax

```
%MM_AddModelFile (ModelId=path-to-model,
  SASDataFile=path-to-SAS-file | SASCatalog=path-to-SAS-catalog | TextFile=path-to-text-file | BinaryFile=path-to-binary-file
  <, Name=alternateFileName><, Trace=OFF | ON>)
```

Arguments

ModelId=*path-to-model*

specifies a SAS Model Manager identifier of the model in the SAS Model Manager repository. The identifier specifies the location in the SAS Model Manager repository where the file is to be added. *path-to-model* can be either a SAS Model Manager UUID or a SAS Model Manager path. ModelId is a required argument. The default value is the value of the `_MM_CId` macro variable.

Example: ModelId=8904daa1-0a29-0c76-011a-f7bb587be79f

Example: ModelId=//ModelManagerDefaultRepo/MMRoot/DDHMEQ/HomeEquity/2009/Models/HMEQ%20Loan%20Project

SASDataFile=*path-to-SAS-file*

specifies the path to a SAS data set to add to a model in the SAS Model Manager repository. *path-to-SAS-file* must be a two-level path in the form *libref.filename*.

Example: mysascode.hmeqloan

SASCatalog=*path-to-SAS-catalog*

specifies the path to one or more SAS code model component files to add to a model in the SAS Model Manager repository. *path-to-SAS-catalog* must be a two-level path in the form *libref.catalog*. Use the SASCatalog argument to add the catalog to a model.

Example: mylib.modelinput

TextFile=*path-to-text-file*

specifies the path to a SAS code model component file that is an ASCII text file. *path-to-text-file* is a one-level SAS name to a model component file.

Example: TextFile=inputxml

BinaryFile=*path-to-binary-file*

specifies the path to a SAS code model component file that is a binary file. *path-to-binary-file* is a one-level SAS name to a model component file that is not a text file.

Example: BinaryFile=gainscsv

Name=*alternateFileName*

specifies a name for the file that you are adding. Use the Name argument when your model component filename does not follow the SAS Model Manager model component file naming convention that is specified in the model's template file or your model requires a file to have a particular filename. If Name is not specified, the filename that is registered is the name of the file.

Example: Name=score.sas

Trace=ON | OFF

specifies whether to supply verbose trace messages to the SAS log.

Default: OFF.

Example: Trace=on

Details

For models that require model component files other than the score code, you can use the %MM_AddModelFile macro to add model component files to a registered model, one file at a time. All files that are added using the %MM_AddModelFile macro are placed in the SAS Model Manager repository. After files have been added, you can view the files in the model folder in the Project Tree.

The %MM_AddModelFile macro supports two types of files, text and binary. Text files are ASCII files that contain character data. Binary files are files created by an application in a format specific to that application. If you are adding a text file, you must use the TextFile argument to specify the file. To avoid any unintentional character translations, all non-text files should be added using the BinaryFile argument.

SAS data sets and SAS catalogs are both binary files. Instead of using the BinaryFile argument to add SAS files, you can use the SASDataFile and SASCatalog arguments respectively to add files using the SAS two-level references *libref.filename* or *libref.catalog*. The TextFile and BinaryFile arguments require a single SAS filename that can be a fileref.

The ModelId argument defaults to the value of the global variable _MM_CId. For example, after a call to the %MM_Register macro, the _MM_CId variable is set to the identifier for the registered model. In this case, you can use the %MM_AddModelFile macro to add additional component files to your model without having to explicitly specify the ModelId argument.

When you use the %MM_AddModelFile macro to add a component file to your SAS Model Manager model, the name of the added component file remains unchanged by default. If you need to change the name of the component file when you save it to a SAS Model Manager model, you can use the Name argument to specify the new component filename. Whenever possible, you should try to follow the component file naming conventions that are specified in the model's template file. When you use the model template file naming conventions, you are less likely to be confused about filenames.

Example

```

/*****
/* Adding a file to a registered model.          */
*****/

Options NOMlogic NOMprint NOSpool;

/*****
/* Get the SAS Model Manager macro code.          */
*****/

Filename MMAccess catalog 'SASHELP.modelmgr.AccessMacros.source';
%include MMAccess;

/* Fileref to the encoded password          */

FILENAME pwfile 'my-network-path\pwfile';
/*****
/* Set the SAS AP Server variables.          */
*****/

%let _MM_ServerName=myserver.com;
%let _MM_PortNumber=6411;
%let _MM_User=sasdemo;
data _null_;
  infile pwfile obs=1 length=1;
  input @;
  input @1 line $varying1024. 1;
  call symput('_MM_Password',substr(line,1,1));
run;

/*****
/* If you use a UUID to specify the target, then  */
/* the value for the _MM_RepositoryId must be set. */
*****/

%let _MM_RepositoryId=ModelManagerDefaultRepo;

/*****
/* A LIBNAME for a table.          */
*****/

LIBNAME mtbls 'c:\mysascode';

/*****
/* Set to detect failure in case macro load fails */
/* and add the input data source.          */
*****/

%let _MM_RC= -1;

%MM_AddModelFile(
  ModelId=
    //ModelManagerRepo/MMRoot/HomeEquity/HMEQ/2009/hmeqDecTree1,
  Name=modelinput.sas7bdat,
  SASDataFile=mtbls.myInputVariables,

```

```

Trace=Off
);

/*****
/* A FILENAME for a text file.
*****/

FILENAME tcode 'c:\myModel\inputvar.xml';

/*****
/* Set to detect failure in case macro load fails */
/* and add the xml file for the input data source */
*****/

%let _MM_RC= -1;

%MM_AddModelFile(
  ModelId=
    //ModelManagerRepo/MMRoot/HomeEquity/HMEQ/2009/hmeqDecTree1,
  TextFile=tcode,
  Trace=on);

```

%MM_GetModelFile Macro

Overview of the %MM_GetModelFile Macro

Use the %MM_GetModelFile macro in a SAS programs to access files in the SAS Model Manager repository. This macro copies the specified model file to the specified location on a local or network computer.

- [“Syntax” on page 252](#)
- [“Arguments” on page 252](#)
- [“Details” on page 254](#)
- [“Example” on page 254](#)

Syntax

```

%MM_GetModelFile (
  ModelId=path-to-model | VersionId=path-to-version | ProjectId=path-to-project,
  SASDataFile=path-to-SAS-data-file | SASCatalog=path-to-SAS-catalog |
  TextFile=path-to-text-file | BinaryFile=path-to-binary-file
  <, Name=alternateFileName>
  <, Trace=ON | OFF>

```

Arguments

ModelId=*path-to-model*

specifies a SAS Model Manager identifier to the model in the SAS Model Manager repository. *path-to-model* can be either a SAS Model Manager UUID or a SAS Model Manager path that describes the location of the specific model. ModelId is a required argument. The default value is the value of the _MM_CId macro variable.

Example: ModelId=ModelId=b2341a42-0a29-0c76-011a-f7bb7bc4f1e9

Example: ModelId=//ModelManagerDefaultRepo/MMRoot/DDHMEQ/HomeEquity/
2009/Models/HMEQ%20Loan%20Project

VersionId

specifies a SAS Model Manager identifier of the version folder to where a champion model resides in the SAS Model Manager repository. *path-to-version* can be either a SAS Model Manager UUID or a SAS Model Manager path that describes the location of the version.

Example: VersionId=b23327cb-0a29-0c76-011a-f7bb3d790340

Example: VersionId=//ModelManagerDefaultRepo/MMRoot/DDHMEQ/
HomeEquity/2009

ProjectId

specifies a SAS Model Manager identifier of the project folder. The identifier specifies the location where the champion model under the default version resides in the SAS Model Manager repository. *path-to-project* can be either a SAS Model Manager UUID or a SAS Model Manager path that describes the location of the project.

Example: VersionId=b232d766-0a29-0c76-011a-f7bb50921b42

Example: VersionId=//ModelManagerDefaultRepo/MMRoot/DDHMEQ/
HomeEquity

SASDataFile=*path-to-SAS-file*

specifies the destination path for a SAS data set. *path-to-SAS-file* must be a two-level path in the form *libref.filename*.

Example: mylib.modelinput

SASCatalog=*path-to-SAS-catalog*

specifies the SAS catalog to store a SAS catalog file. *path-to-SAS-catalog* must be a two-level path in the form *libref.catalog*.

Example: mylib.format

TextFile=*path-to-text-file*

specifies the destination path for a component file that is an ASCII text file. *path-to-text-file* is a one-level path to a model component file. The path can be a fileref.

Example: TextFile=myfileref

BinaryFile=*path-to-binary-file*

specifies the destination path for a model component file that is a binary file. *path-to-binary-file* is a one-level pathname to a model component file that is not a text file. The pathname can be a fileref.

Example: BinaryFile=myfileref

Name=*alternateFileName*

specifies a name for the model component file that you are retrieving. Use the Name argument when the name of the destination file does not match the name of the file in the SAS Model Manager repository. The Name argument is the filename within the SAS Model Manager repository. If Name is not specified, the filename that is registered in the SAS Model Manager repository is the name of the file.

Example: Name=score.sas

Trace=ON | OFF

specifies whether to supply verbose trace messages to the SAS log.

Default: OFF.

Example: Trace=on

Details

Use the %MM_GetModelFile macro to retrieve a component file for a model that has been registered in the SAS Model Manager repository. You can retrieve a component file for any model by specifying the repository location of the model, or you can retrieve a component file for a champion model by specifying the version or project location in the SAS Model Manager repository.

The %MM_GetModelFile macro supports two types of files, text and binary files. Text files are ASCII files that contain character data. Binary files are files that are created by an application in a format that is specific to that application. If you are retrieving a text file, you must use the TextFile argument to specify the file. To avoid any unintentional character translations, all non-text files should be retrieved by using the BinaryFile argument.

SAS data files and SAS catalogs are binary files. Instead of using the BinaryFile argument to retrieve model component files to store as a SAS file or in a SAS catalog, you can use the SASDataFile and SASCatalog arguments respectively to specify the SAS location to store the file. The TextFile and BinaryFile arguments require a single SAS filename.

You can use the optional Name argument if you want to save the model component file with a different name from the name within the SAS Model Manager repository.

After you use the %MM_GetModelFile macro to copy a model component file to its new location, you can use the model component file for any purpose. For example, a simple application might use the %MM_GetModelFile macro to copy a registered model's score code file to the SAS WORK library. After the score code is copied to WORK, SAS code can be written that includes the score code into a SAS DATA step and is executed for experimental purposes.

If the destination file argument or the two-level SAS library reference name that is invoked in the macro uses the original filename, you do not need to specify the Name argument. In other words, the macro can use the SAS logical names to determine the name of the file in the model hierarchy. If the name of the destination file needs to be different from the name of the original file that was copied, use the Name argument to specify the new name for the model component file.

Example

```

/*****/
/* Get the score code from a registered model and run */
/* it.                                                    */
/*****/

Options NOMlogic NOMprint NOspool;

/*****/
/* Get the SAS Model Manager macro code.                */
/*****/

FILENAME MMAccess catalog 'sashelp.modelmgr.accessmacros.source';
%include MMAccess;

/* Fileref to the encoded password                        */

FILENAME pwfile 'my-network-path\pwfile';

/*****/
/* Set SAS AP Server variables.                          */
/*****/

```

```

%let _MM_ServerName = myserver.com;
%let _MM_PortNumber = 6411;
%let _MM_User = miller;
data _null_;
  infile pwfile obs=1 length=1;
  input @;
  input @1 line $varying1024. 1;
  call symput('_MM_Password',substr(line,1,1));
run;

/*****
/* If you use a UUID to specify the target, then      */
/* the value for the _MM_RepositoryId must be set.    */
*****/

%let _MM_RepositoryId=ModelManagerDefaultRepo;

/*****
/* Specify the model component file name and          */
/* destination.                                       */
*****/

%let WorkPath = c:\myProject\2009;
FILENAME dest '&WorkPath.\score.sas';

/*****
/* Set to detect failure in case macro load fails.   */
*****/

%let _MM_RC = -1;

/*****
/* Get score code.                                   */
*****/

%MM_GetModelFile(ModelId=//ModelManagerRepo/MMRoot/HomeEquity/HMEQ/2009/
DecisionTree,
  TextFile=dest);

/*****
/* Display SAS Model Manager set macro variables.    */
*****/

Options nosource;
%PUT _MM_RC = &_MM_RC;
%PUT _MM_RepositoryId = &_MM_RepositoryId;
%PUT _MM_CId = &_MM_CId;
Options source;

/*****
/* Run score code. Sepcify the LIBNAME input path.   */
*****/

LIBNAME input 'c:\mysascode\2009\DTTree';
DATA score;

```

```

set input.dTreeInp;
%include dest;
run;

```

%MM_GetURL Macro

Overview of the %MM_GetURL Macro

The MM_GetURL macro translates a specified SAS Model Manager UUID to a URL-style path address and sets the URL address as the value of the _MM_URL and _MM_ResourcesURL macro variables.

- [“Syntax” on page 256](#)
- [“Arguments” on page 256](#)
- [“Details” on page 256](#)
- [“Example” on page 257](#)

Syntax

```
%MM_GetURL(UUID=UUID, RepositoryName=path-to-repository, <Trace=ON | OFF> );
```

Arguments

UUID=UUID

specifies the UUID of the object for which an URL is desired. A SAS Model Manager UUID is a 36-character string that identifies a single object in the SAS Model Manager model repository. The UUID argument is required.

Example: UUID=cca1ab08-0a28-0e97-0051-0e3991080867

RepositoryName=SAS-Model-Manager-repository-name

specifies the name of the SAS Model Manager model repository. This argument is required if the global macro variable _MM_RepositoryId is not set.

Default: the value of the _MM_RepositoryId

Example: RepositoryName=ModelManagerDefaultRepo

Trace=ON | OFF

specifies whether to supply verbose trace messages to the SAS log.

Default: OFF

Example: Trace=on

Details

The %MM_GetURL macro sets the value of the global macro variable _MM_URL to the URL of the specified SAS Model UUID.

If the UUID argument specifies a SAS Model Manager version or model, then the macro sets the global macro variable _MM_ResourcesURL to the URL of that object's associated Resources folder.

The %MM_GetURL macro does not set a value for the global macro variable, _MM_CID.

Example

```

/*****
/* Get the URL for the location of a model.          */
*****/

Options nomlogic nomprint nospool;

/*****
/* Get the SAS Model Manager macro code.          */
*****/
/
FILENAME MMAccess catalog 'sashelp.modelmgr.accessmacros.source';
%include MMAccess;

/* Fileref to the encoded password                  */

FILENAME pwfile 'my-network-path\pwfile';

/*****
/* Set SAS AP Server variables.                    */
*****/

%let _MM_ServerName=myserver.com;
%let _MM_PortNumber=6411;
%let _MM_User=miller;
data _null_;
    infile pwfile obs=1 length=1;
    input @;
    input @1 line $varying1024. 1;
    call symput('_MM_Password',substr(line,1,1));
run;

/*****
/* Set to detect failure in case macro load fails */
/* and get the URL.                               */
*****/

%let _MM_RC= -1;

%let target=aef7a78e-0a28-0e97-01c0-b8a0e5ba15c7;
%MM_GetURL(UUID=&target,RepositoryName=ModelManagerDefaultRepo, Trace=on);
%put _MM_URL=&_MM_URL;
%put _MM_ResourcesURL=&_MM_ResourcesURL;

```

%MM_Register Macro**Overview of the %MM_Register Macro**

The %MM_Register macro registers a model to an existing version in the SAS Model Manager model hierarchy.

- [“Syntax” on page 258](#)
- [“Arguments” on page 258](#)
- [“Details ” on page 263](#)

- [“Examples ” on page 270](#)

Syntax

```
%MM_Register(
  VersionId=destination-version-UUID,
  ModelTemplate=model-template-name,
  EMModelPackage=SAS-fileref-for-EM-package-file,
  ScoreDataStepCode=fileref-to-data-step-fragment-score-code,
  ScoreProgram=fileref-to-SAS-program-score-code,
  InDataSamp=SAS-data-set-reference-to-input-data-sample-table,
  InDataInfo=SAS-data-set-reference-tor-input-variable-metadata-table,
  OutDataSamp=SAS-data-set-reference-for-output-data-sample-table,
  OutDataInfo=SAS-data-set-reference-for-output-variable-metadata-table,
  TargetDataSamp=SAS-data-set-reference-for-target-data-sample-table,
  TargetDataInfo=SAS-data-set-reference-for-target-variable-metadata-table,
  TrainingDataSamp=SAS-data-set-reference-for-training-data-sample-table,
  LogisticOutModelTable=SAS-data-set-reference-for-PROC-LOGISTIC-outmodel-
    table,
  ReportDir=path-to-EMREPORT-directory,
  KeepInVars=keep-variable-list-for-InDataSamp,
  KeepOutVars=keep-variable-list-for-OutDataSamp,
  KeepTargetVars=keep-variable-list-for-TargetDataSamp,
  ModelName=model-name,
  Description=model-description,
  Label=model-label,
  Subject=model-subject,
  Algorithm=model-algorithm,
  Function=model-function,
  Modeler=modeler-property,
  Tool=model-tool-property,
  ToolVersion=model-tool-version,
  Trace=ON | OFF
);
```

Arguments

Note: If a %MM_Register macro parameter contains a semicolon, comma, apostrophe, or quotation mark (; , ' ") character, you must add %bquote to the macro parameter. For example, %MM_Register(..., Description=%bquote(My Division's Model), ...);

VersionId=*destination-version-UUID*

specifies the SAS Model Manager UUID for an existing version in the SAS Model Manager model repository.

Required: Yes

Default: the value of the _MM_CId macro variable

ModelTemplate=*model-template-name*

specifies the SAS Model Manager model template that was used to register and validate this model.

Required: No

Default:

- For models that were registered using the EMMModelPackage parameter, the template is set according to the information that is contained within the named EM model package.
- Models that were registered using the LogisticOutModelTable parameter are registered with the Classification template.
- All other registrations default to the AnalyticalModel template.

EMModelPackage=*SAS-fileref-for-EM-package-file*

specifies a SAS file reference that points to the Enterprise Miner model package file (SPK) that contains the model to be registered.

Required: The EMMModelPackage argument is required unless you use the ReportDir argument, the ScoreDataStepCode argument, or the ScoreProgram argument to specify the model code filename.

ScoreDataStepCode=*fileref-to-data-step-fragment-score-code*

specifies a SAS file reference for the model score code that is a fragment of SAS code that can be included in a DATA step. A DATA step fragment contains no DATA, PROC, or RUN statements.

Required: The ScoreDataStepCode argument is required unless you use the EMMModelPackage argument, the ReportDir argument, or the ScoreProgram argument to specify the model code filename.

ScoreProgram=*fileref-to-SAS-program-score-code*

specifies a SAS file reference for a text file containing the SAS program, including all step code that is required for successful execution of the model score code.

Required: The ScoreProgram argument is required unless you use the EMMModelPackage argument, the ReportDir argument, or the ScoreDataStepCode argument to specify the model code filename.

InDataSamp=*SAS-data-set-reference-to-input-data-sample-table*

specifies a two-level SAS data set reference in the form *libref.filename* that points to a model input data sample table. The input data sample table is a table that contains all model input variables and is used to create the inputvar.xml file that is required for model registration. The input data sample table is not required for models that were imported as SAS Enterprise Miner package files.

Required: The InDataSamp argument is required unless you use the InDataInfo argument.

Tip: When you use the %MM_Register macro to register a model, the inputvar.xml file should contain only input variables for the model that you are registering. If the input data sample table includes variables that are not used by the model, use the KeepInVars argument to remove these variables. If no variables are specified by the KeepInVars argument, SAS filters the target variables from the table specified by the InDataSamp argument.

See also: [KeepInVars argument on page 261](#)

InDataInfo=*SAS-data-set-reference-for-input-variable-metadata-table*

specifies a two-level SAS data set reference in the form *libref.filename* that points to a model input variable metadata table. The input variable metadata table should be in the form of a CONTENTS procedure output file, which has the columns NAME, TYPE, LENGTH, LABEL, FORMAT, LEVEL, and ROLE. Each row of the table is a variable.

The model input variable metadata table is used to create the inputvar.xml file that is required for model registration.

Required: The InDataInfo argument must be specified unless you use the InDataSamp argument.

Tip: When you use the %MM_Register macro to register a model, the inputvar.xml file should contain only variables for the model that you are registering. If no variables are specified in the KeepInVars argument, SAS filters the target variables from the table specified by the InDataInfo argument.

See also: The CONTENTS Procedure in the *Base SAS 9.2 Procedures Guide*

OutDataSamp=SAS-data-set-reference-for-output-data-sample-table

specifies a two-level SAS data set reference in the form *libref.filename* that points to a model output data sample table. The output data sample table should contain all variables that are created or modified by the model and is used to create the outputvar.xml file that is required for model registration. The output data sample table is not required for models that were imported as SAS Enterprise Miner package files.

Required: The OutDataSamp argument must be specified unless you use the OutDataInfo argument.

Interaction: If the output data sample table includes variables that are created or modified by the model, use the KeepOutVars argument to remove these variables. If no variables are specified in the KeepOutVars argument, SAS filters the input variables and the target variables from the table that is specified by the OutDataSamp argument.

See also: [KeepOutVars argument on page 261](#)

OutDataInfo=SAS-data-set-reference-for-output-variable-metadata-table

specifies a two-level SAS data set reference in the form *libref.filename* that points to a model output variable metadata table. The output variable metadata table should contain all of the variables that are created or modified by the model. The SAS file should be in the form of the CONTENTS procedure output file, which has the columns NAME, TYPE, LENGTH, LABEL, FORMAT, LEVEL, and ROLE. Each row of the table contains a variable. The output variable metadata table is used to create the outputvar.xml file that is required for model registration.

Required: The OutDataInfo argument must be specified unless you use the OutDataSamp argument.

Interaction: If no variables are specified by the KeepOutVars argument, SAS filters the input variables and target variables from the table that is specified by the OutDataInfo argument.

TargetDataSamp=SAS-data-set-reference-for-target-data-sample-table

specifies a two-level SAS data set reference in the form *libref.filename*. The data set reference points to a SAS table that contains the model target variable. The SAS file should contain the variable that was used as the model target during training. The SAS file is used to create the target variable information in the targetvar.xml file that is used for SAS Model Manager model registration.

Required: No

Tip: If the target data sample table includes other variables that are not model target variables, use the KeepTargetVars argument to remove these variables.

See also: [KeepTargetVars argument on page 261](#)

TargetDataInfo=SAS-data-set-reference-for-target-variable-metadata-table

specifies a two-level SAS data set reference in the form *libref.filename*. The data set reference points to a SAS table that contains the model's target variable and its metadata.

The SAS file should be in the form of the CONTENTS procedure output file, which has the columns NAME, TYPE, LENGTH, LABEL, FORMAT, LEVEL, and ROLE. Each row of the table contains a variable. The metadata in the SAS file is used to create the target variable information in the target.xml file that is used for SAS Model Manager model registration. F

Required: No

TrainingDataSamp=SAS-data-set-reference-for-training-data-sample-table specifies a two-level SAS data set reference in the form *libref.filename*. The data set reference points to a SAS file that contains the training data that is used for a model created by the LOGISTIC procedure. The training data sample must be an exact sample of the training data that is submitted to the LOGISTIC procedure. When the TrainingDataSamp argument and the LogisticOutModelTable argument are specified, the %MM_Register macro can derive the input, output, and target variables to create the inputvar.xml file, the outputvar.xml file, and the targetvar.xml file.

Required: No

LogisticOutModelTable==SAS-data-set-reference-for-PROC-LOGISTIC-outmodel-table specifies a two-level SAS data set reference in the form *libref.filename* that points to a LOGISTIC procedure fit table that was created by using the PROC LOGISTIC OUTMODEL= statement, and is suitable for use with the PROC LOGISTIC INMODEL statement. If the TrainingDataSamp argument is specified, then SAS generates the input, output, and target variable metadata from this table. In this case, the InDataSamp and the OutDataSamp arguments do not need to be specified.

Required: Only if the model is created by the LOGISTIC procedure using the OUTMODEL statement.

ReportDir=path-to-EMREPORT-directory specifies an absolute file path to the EMREPORT directory that was created by the SAS Enterprise Miner batch code. All SAS Enterprise Miner model packages that are named miningResult.spk and that reside in a subdirectory of the EMREPORT directory are registered to the target version. The ReportDir argument is valid only for use with SAS Enterprise Miner model package files.

Required: No

KeepInVars=keep-variable-list-for-InDataSamp specifies a list of input variables or columns that are retained in the model's inputvar.xml file. Only variables from the table specified by the InDataSamp argument can be specified in this list.

Required: No

See also: [InDataSamp argument on page 259](#)

KeepOutVars==keep-variable-list-for-OutDataSamp specifies a list of variables or columns that are retained in the model's outputvar.xml file. Only variables from the table specified by the OutDataSamp argument can be specified in this list.

Required: No

See also: [KeepOutVars argument on page 260](#)

KeepTargetVars=keep-variable-list-for-TargetDataSamp specifies a list of variables or columns that are retained in the model's targetvar.xml file. Only variables from the tables that are specified by the TargetDataSamp argument can be specified in this list.

Required: No

See also: [TargetDataSamp argument on page 260](#)

ModelName=*model-name*

specifies the name of the model, which will be used as the value of the model **Model Name** property in the Project Tree.

Required: Yes

Description=*model-description*

specifies a description of the model, which will be used as the value of the model **Description** property in the Project Tree.

Required: No

Label=*model-label*

specifies a model's label, which will be used as the value for the model **Model Label** property in the Project Tree. *model-label* is a text string that is used as the label for the selected model in the model assessment charts that SAS Model Manager creates. If *model-label* is not specified, SAS Model Manager uses the text string that is specified for the **ModelName** argument.

Required: No

Subject=*model-subject*

specifies the model's subject, which will be used as the value for the model **Subject** property in the Project Tree. *model-subject* provide an additional description for a model, such as a promotional or campaign code. This property is not tied to any computational action by SAS Model Manager.

Required: No

Algorithm=*model-algorithm*

specifies the model's computation algorithm, which will be used as the value of the model **Algorithm** property in the Project Tree.

Required: No

Example: **Algorithm**=Decision Tree

Function=*model-function*

specifies the model's function class, which will be used as the value for the model **Function** in the Project Tree.

Required: No

Valid values: Classification, Prediction, Association, Clustering, Sequence, Forecasting, TextMining, Transformation, and EMCreditScoring

Modeler=*model-creator*

specifies the SAS Model Manager user ID for the person who created the model, which will be used as the value of the model **Modeler** property in the Project Tree.

Required: No

Tool=*model-tool*

specifies the modeling tool that was used to create the model, and that will be used as the value of the model **Tool** property in the Project Tree.

Required: No

ToolVersion

specifies the version of the tool that was used to create the model, and that will be used as the value of the model **Tool Version** property in the Project Tree.

Required: No

Trace=ON | OFF

specifies whether to supply verbose trace messages to the SAS log.

Default: OFF.

Example: Trace=on

Details

Overview of Using the %MM_Register Macro

The %MM_Register macro registers the following types of models to an existing version in the SAS Model Managers repository:

- a model as a SAS Enterprise Miner package
- a SAS DATA step fragment
- a SAS program

In order to register a model using the %MM_Register macro, the macro must know the model name, the version in which the model is registered, the model source code, the model template, and the model input and output variables. If you register a SAS Enterprise Miner model, this information is included in a SAS Enterprise Miner package file (SPK file). When you register SAS code models, you must specify the model name, version, and model score code, as well as the model input and output variables in the respective macro arguments. Several %MM_Register macro arguments enable you to provide values for model property values that display in the Project Tree.

Registering SAS Enterprise Miner Models

Models that were created in SAS Enterprise Miner and saved as a SAS Enterprise Miner SPK file contain all of the information that is needed to register a model in SAS Model Manager. Registering SAS Enterprise Miner SPK files requires you to specify the following arguments:

- ModelName
- VersionId
- EMMModelPackage or ReportDir arguments

To register one SAS Enterprise Miner model, you can specify the EMMModelPackage argument. To register multiple SAS Enterprise Miner models, you use the ReportDir argument to name a directory whose subdirectories each contain a miningResult.spk file. You can register multiple models simultaneously in SAS Model Manager.

SAS Enterprise Miner generates a program, EMBatch, to create multiple models in a batch program. You can modify the EMBatch program to include the %MM_Register macro, using the macro variable &EMREPORT as the value of the ReportDir argument. By making this change to the EMBatch program, you can create and register SAS Enterprise Miner models in a batch program for use in SAS Model Manager.

Registering SAS Code Models

When you register SAS code models, the information that is required is not contained in an SPK file and you must specify the required information using the %MM_Register arguments. Each model that you register must specify the model name, the model version, the model template, the model code, and the SAS data sets that describe the input, output, and target variables.

Use the following table for usage information about using the %MM_Register arguments:

Required Information	Argument	Usage
model name	ModelName	Specify the name of the model, which is used to identify the model in the SAS Model Manager model repository.
version	VersionId	Specify the name of the version in which the model is registered.
model score code Specify one of the following arguments: <ul style="list-style-type: none"> ScoreDataStepCode ScoreProgram LogisticOutModelTable 	ScoreDataStepCode	<p>Specify a fileref that points to a file that contains score code that is a DATA step fragment. A DATA step fragment contains no DATA, PROC or RUN statements.</p> <p>When you specify the ScoreDataStepCode argument, your model input and output variables can be defined using one of the following pairs of arguments:</p> <ul style="list-style-type: none"> InDataSamp and OutDataSamp InDataInfo and OutDataInfo InDataSamp and OutDataInfo
	ScoreProgram	<p>Specify a LOGISTIC procedure FIT table in the form <i>libref.filename</i> that was created by the PROC LOGISTIC OUTMODEL= statement. The FIT table can be used as the value in a PROC LOGISTIC INMODEL= statement.</p> <p>When you specify the ScoreProgram argument, your model input and output variables can be defined using one of the following pairs of arguments:</p> <ul style="list-style-type: none"> InDataSamp and OutDataSamp InDataInfo and OutDataInfo

Required Information	Argument	Usage
	LogisticOutModelTable	<p>Specify a <i>libref.filename</i> that points to a LOGISTIC procedure FIT table that was created by the PROC LOGISTIC OUTMODEL= statement, which can be used as the value to a PROC LOGISTIC INMODEL= statement.</p> <p>If the model does not contain data transmission and you specify a value for the TrainingDataSamp argument, SAS Model Manager uses the training sample data set and the FIT table to create the model inputvar.xml file, the outputvar.xml file, and the targetvar.xml file.</p> <p>If you do not specify a value for the TrainingDataSamp argument or if your program transforms the model input before running the LOGISTICS procedure, you must provide the model input and output variables using the InDataSamp or InDataInfo argument, and the OutDataSamp or OutDataInfo argument.</p>

Required Information	Argument	Usage
input variables	InDataSamp	<p>Specify a fileref to a SAS data set whose variables contain the input variables that are used by the SAS code model. An example would be a data set that was used for training the model.</p> <p>SAS Model Manager reads one observation in the data set that is specified by the InDataSamp argument to create the inputvar.xml file for the model. The inputvar.xml file defines the model input variables and their metadata.</p> <p>Based on the arguments that were specified, the %MM_Register macro uses arguments to filter variables from the data set to create the inputvar.xml file.</p> <ul style="list-style-type: none"> You can use the KeepInVars argument to specify the variables in the InDataSamp data set that are used to create the inputvar.xml file. If you do not specify the KeepInVars argument, you can specify a value for the TargetDataSamp argument or the TargetDataInfo argument to filter variables based on this target data sample data set. <p>For more information, see “KeepInVars=keep-variable-list-for-InDataSamp” on page 261.</p>

Required Information	Argument	Usage
	InDataInfo	<p>Specify a fileref that points to a SAS data set whose variables are NAME, TYPE, LENGTH, LABEL, FORMAT, LEVEL, and ROLE. These variables define metadata for the model input variables. Each row in this data set contains the metadata for model input variables. Such a table can be created by the CONTENTS procedure.</p> <p>SAS Model Manager reads the data set that is specified by the InDataInfo argument to create the inputvar.xml file for the model. The inputvar.xml file defines the model input variables and their metadata.</p> <p>The variables in the data set that are specified by the TargetDataSamp argument or the TargetDataInfo argument are used as a filter to create the inputvar.xml file.</p>

Required Information	Argument	Usage
output variables	OutDataSamp	<p>Specify a fileref that points to a SAS data set whose variables contain the output variables that are created or modified by the SAS code model. An example is a data set that was the scored output of the model.</p> <p>SAS Model Manager reads the data set that is specified by the OutDataSamp argument to create the outputvar.xml file for the model. The outputvar.xml file defines the model output variables and their metadata.</p> <p>Based on the arguments that were specified, the %MM_Register macro uses arguments to filter variables from the data set to create the outputvar.xml file.</p> <ul style="list-style-type: none"> You can use the KeepOutVars argument to specify the variables in the OutDataSamp data set that are used to create the outputvar.xml file. If you do not specify the KeepOutVars argument, input variables and target variables are filtered from the output table. <p>For more information, see “KeepOutVars==keep-variable-list-for-OutDataSamp” on page 261.</p>

Required Information	Argument	Usage
	OutDataInfo	<p>Specify a fileref that points to a SAS data set whose variables are NAME, TYPE, LENGTH, LABEL, FORMAT, LEVEL, and ROLE. These variables define metadata for the model output variables. Each row in this data set contains the metadata for model output variables. Such a table can be created by the CONTENTS procedure.</p> <p>SAS Model Manager reads the data set that is specified by the OutDataInfo argument to create the outputvar.xml file for the model. The outputvar.xml file defines the model output variables and their metadata. If you do not specify the KeepOutVars argument, input variables and target variables are filtered from the output table.</p>
target variable	TargetDataSamp	<p>Specify a fileref that points to a SAS data set whose variables contain the target variable that is created or modified by the SAS code model. An example is a data set that was the scored output of the model.</p> <p>SAS Model Manager reads the data set that is specified by the TargetDataSamp argument to create the targetvar.xml file for the model. The targetvar.xml file defines the target output variable and its metadata.</p> <p>You can use the KeepTargetVars argument to specify the variable in the TargetDataSamp data set that is used to create the targetvar.xml file.</p>

Required Information	Argument	Usage
	TargetDataInfo	<p>Specify a fileref that points to a SAS data set whose variables are NAME, TYPE, LENGTH, LABEL, FORMAT, LEVEL, and ROLE. These variables define metadata for the model target variable. A row in this data set contains the metadata for the model target variable. Such a table can be created by the CONTENTS procedure.</p> <p>SAS Model Manager reads the data set that is specified by the TargetDataInfo argument to create the targetvar.xml file for the model. The targetvar.xml file defines the model target variable and its metadata.</p>

Use the %MM_AddModelMfile macro to register other model component files that are not registered by the %MM_Register macro. For more information, see [“Model Templates” on page 84](#) and [“Syntax” on page 249](#).

Examples

Example Code A2.1 Registering a SAS Enterprise Miner Model Package

```

/*****
/* Registering a SAS Enterprise Miner Model Package. */
*****/

Options NOmlogic NOmprint NOSpool;

/*****
/* Access and load the SAS Model Manager macro code.*/
*****/

Filename MMAccess catalog 'SASHELP.modelmgr.AccessMacros.source';
%include MMAccess;

/* Fileref to the encoded password */

FILENAME pwfile 'my-network-path\pwfile';

/*****
/* Set SAS AP Server variables. *****/
*****/

%let _MM_ServerName = myserver.com;
%let _MM_PortNumber = 6411;
%let _MM_User = miller;
data _null_;
    infile pwfile obs=1 length=1;
    input @;

```

```

input @1 line $varying1024. 1;
call symput('_MM_Password',substr(line,1,1));
run;

/*****
/* If you use a UUID to specify the target, then      */
/* the value for the _MM_RepositoryId must be set.    */
*****/

%let _MM_RepositoryId = ModelManagerDefaultRepo;

/*****
/* Specify the path for a SAS Enterprise              */
/* Miner Model Package file miningResult.spk.        */
*****/

FILENAME EMPak 'c:\myscorecode\EM\miningResult.spk';

/*****
/* Set to detect failure in case macro load fails    */
/* and register the Enterprise Miner model.          */
*****/

%let _MM_RC= -1;

%MM_Register(
  VersionId=
    //ModelManagerModelRepos/MMRoot/HomeEquity/HMEQ/2009,
  EMModelPackage=EMPak,
  ModelName=HMEQ,
  Description=Home Equity Score Code,
  Modeler=Titus Groan,
  Function=Reg,
  Tool=SAS/Enterprise Miner,
  ToolVersion=v902,
  Subject= Loan,
  Trace=ON);

/*****
/* Display MM_Register defined variables.             */
*****/

Options nosource;
%PUT _MM_RC = &_MM_RC;
%PUT _MM_CId = &_MM_CId;
Options source;

```

Example Code A2.2 Registering a Generic Model

```

/*****
/* Registering a generic model.                      */
*****/

Options nomlogic nomprint nospool;

/*****

```

```

/* Load and access the SAS Model Manager macro code. */
/*****

Filename MMAccess catalog 'SASHELP.modelmgr.AccessMacros.source';
%include MMAccess;

/* Fileref to the encoded password */

FILENAME pwfile 'my-network-path\pwfile';

/*****
/* Set the SAS AP Server variables. */
*****/

%let _MM_ServerName = myserver.com;
%let _MM_PortNumber = 6411;
%let _MM_User = miller;
data _null_;
    infile pwfile obs=1 length=1;
    input @;
    input @1 line $varying1024. 1;
    call symput('_MM_Password',substr(line,1,1));
run;

/*****
/* If you use a UUID to specify the target, then */
/* the value for the _MM_RepositoryId must be set. */
*****/

%let _MM_RepositoryId = ModelManagerDefaultRepo;

/*****
/* Specify the location of the files. */
*****/

LIBNAME modelTbl 'c:\myModel\tables';
FILENAME Code 'c:\myModel\scoreCode';

/*****
/* Set to detect failure in case macro load fails */
/* and register the model in SAS Model Manager */
*****/

%let _MM_RC= -1;

%MM_Register(
    VersionId=
        //ModelManagerModelRepos/MMRoot/HomeEquity/HMEQ/2009,
    ScoreDataStepCode=CODE,
    InDataSamp=modelTbl.HMEQInput,
    OutDataSamp=modelTbl.HMEQOutput,
    TargetDataSamp=modelTbl.HMEQTarget,
    ModelName=HMEQDTree,
    Description= Home Equity model Added with a SMM Macro,
    Trace=ON);

```

```

/*****/
/* Display the SAS Model Manager defined variables. */
/*****/

```

```

Options nosource;
%PUT _MM_RC = &_MM_RC;
%PUT _MM_CId = &_MM_CId;
Options source;

```

Example Code A2.3 *Registering a PROC LOGISTIC OUTMODEL-Style Model*

```

/*****/
/* Registering a PROC LOGISTIC OUTMODEL-style model. */
/*****/

```

```
Options nomlogic nomprint nospool;
```

```

/*****/
/* Load and access the SAS Model Manager macro code. */
/*****/

```

```

Filename MMAccess catalog 'SASHELP.modelmgr.AccessMacros.source';
%include MMAccess;

```

```
/* Fileref to the encoded password */
```

```
FILENAME pwfile 'my-network-path\pwfile';
```

```

/*****/
/* Set the SAS AP Server variables. */
/*****/

```

```

%let _MM_ServerName = myserver.com;
%let _MM_PortNumber = 6411;
%let _MM_User = miller;
data _null_;
    infile pwfile obs=1 length=1;
    input @;
    input @1 line $varying1024. 1;
    call symput('_MM_Password',substr(line,1,1));
run;

```

```

/*****/
/* If you use a UUID to specify the target, then */
/* the value for the _MM_RepositoryId must be set. */
/*****/

```

```
%let _MM_RepositoryId = ModelManagerDefaultRepo;
```

```

/*****/
/* Specify the location of the files. */
/*****/

```

```

LIBNAME modelTbl 'c:\myModel\Tables';
LIBNAME trainTbl 'c:\HomeEquity\Tables';
FILENAME ProgCode 'c:\myModel\scoreCode';

```

```

/*****
/* Set to detect failure in case macro load fails      */
/* and register the model                             */
*****/

%let _MM_RC= -1;

%MM_Register(
  VersionId=
    //ModelManagerModelRepos/MMRoot/HomeEquity/HMEQ/2009,
  ScoreProgram=ProgCODE,
  LogisticOutModelTable=modelTbl.HMEQProcLogisticOutput,
  TrainingDataSamp=trainTbl.HMEQTraining,
  ModelName=HMEQLogisticOutmodel,
  Description=HMEQ Logistic OUTMODEL model added by macro,
  Trace=off);

/*****
/* Display the SAS Model Manager-defined variables.    */
*****/

Options nosource;
%PUT _MM_RC = &_MM_RC;
%PUT _MM_RepositoryId = &_MM_RepositoryId;
%PUT _MM_CId = &_MM_CId;
Options source;

```

%MM_RegisterByFolder Macro

Overview of the %MM_RegisterByFolder Macro

You use the %MM_RegisterByFolder macro to register one model or multiple models simultaneously to the SAS Model Manager model repository from a single directory. Each model is located in a subdirectory under the specified directory.

Syntax

%MM_RegisterByFolder (VersionId=*path-to-version*, ReportDir=*path-to-folder*,
<Trace=On | OFF>);

Arguments

VersionId=*path-to-version*

specifies the SAS Model Manager UUID for an existing version in the SAS Model Manager model repository where the models are registered. *path-to-version* can be either a SAS Model Manager UUID or a SAS Model Manager version path.

Required: Yes

Default: the value of the _MM_CId macro variable

Example: VersionId=b23327cb-0a29-0c76-011a-f7bb3d790340

Example: VersionId=//ModelManagerDefaultRepo/MMRoot/DDHMEQ/
HomeEquity/2009

ReportDir=*path-to-folder*

specifies the directory that contains the models to be registered.

Required: Yes

Trace=ON | OFF

specifies whether to supply verbose trace messages to the SAS log.

Default: OFF.

Example: Trace=on

Details

You can register SAS Enterprise Miner models and SAS code models using the %MM_RegisterByFolder macro. The directory that you specify in the ReportDir argument is the parent folder. Each model has its own subfolder under the parent folder. Each type of model has requirements for the subfolder name and the contents of the subfolder:

Table A2.1 Requirements for Registering Models in a Directory

Requirement Type	Enterprise Miner Models	SAS Code Models
Value of ReportDir	a valid directory name	a valid directory name
Model subdirectory name	the subdirectory name must be the name of the model	the subdirectory name must be the name of the model
Contents of the subdirectory	one file named miningResult.spk	Required files: <ul style="list-style-type: none"> Modelmeta.xml ModelInput.sas7bdat Score.sas Optional files: <ul style="list-style-type: none"> ModelOutput.sas7bdat ModelTarget.sas7bdat

Here is a description of the files that reside in the model subfolders:

miningResult.spk

The miningResult.spk file contains the model component files for a model that was created in SAS Enterprise Miner.

Modelmeta.xml

The Modelmeta.xml file uses XML to define the model component files and values for model properties.

ModelInput.sas7bdat

ModelInput.sas7bdat is a table that contains the model input variables. This file is used to create the model inputvar.xml file.

Score.sas

Score.sas contains the SAS score code, which can be a DATA step fragment or a SAS program.

ModelOutput.sas7bdat

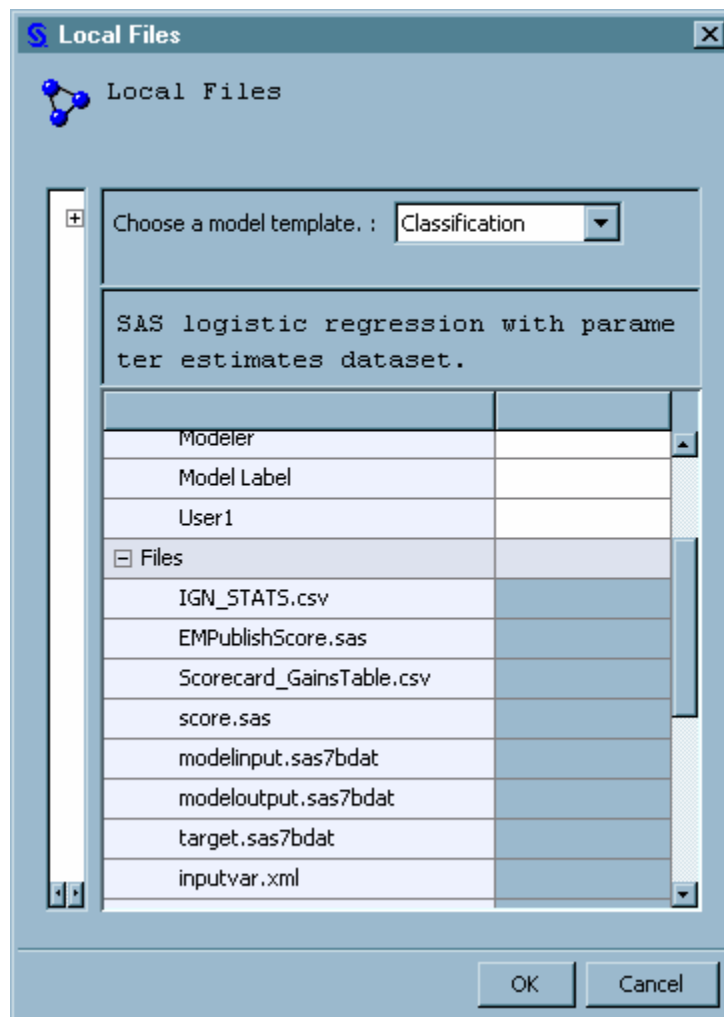
ModelOutput.sas7bdat is a SAS data set that contains one or more model output variables.

ModelTarget.sas7bdat

ModelTarget.sas7bdat is a SAS data set that contains only the target variable.

The Modelmeta.xml file is an XML file that is a mapping of SAS Model Manager component filenames to user-defined component filenames. The <Model> element has two main sections:

- <ModelMetadata> to define model properties
See: “[Specific Properties for a Model](#)” on page 103
- <FileList> to list the model component files. This list is comparable to the **Files** section of the Local Files window, which you use to import SAS code models in the SAS Model Manager window:



For a list of files for each model type, see: “[Model Template Component Files](#)” on page 86

Within the <File> element, put the name of the file that is defined in the model template, in the <name> element. The contents of the <value> element is the filename under the model directory.

Here is an example Modelmeta.xml file for a classification model named HMEQ:

```
<?xml version="1.0" encoding="utf-8" ?>
<Model>
  <ModelMetadata>
    <name>hmeq</name>
    <description>Home Equity Model</description>
    <label>HMEQ</label>
```

```

<algorithm></algorithm>
<function>classification</function>
<modeler></modeler>
<tool>SASProc</tool>
<toolversion></toolversion>
<subject></subject>
<modelTemplate>Classification</ModelTemplate>
<scoreCodeType>SAS Program</scoreCodeType>
</ModelMetadata>
<FileList>
  <File>
    <name>score.sas</name>
    <value>myScoreFile.sas</value>
  </File>
  <File>
    <name>modelinput.sas7bdat</name>
    <value>hmeqIn</value>
  </File>
  <File>
    <name>modeloutput.sas7bdat</name>
    <value>hmeqOut</value>
  </File>
  <File>
    <name>target.sas7bdat</name>
    <value>hmeqTar</value>
  </File>
  <File>
    <name>inputvar.xml</name>
    <value></value>
  </File>
  <File>
    <name>outputvar.xml</name>
    <value></value>
  </File>
  <File>
    <name>targetvar.xml</name>
    <value></value>
  </File>
  <File>
    <name>train.sas7bdat</name>
    <value></value>
  </File>
  <File>
    <name>Training.sas</name>
    <value></value>
  </File>
  <File>
    <name>Training.log</name>
    <value></value>
  </File>
  <File>
    <name>Training.lst</name>
    <value></value>
  </File>
  <File>
    <name>outest.sas7bdat</name>

```

```

        <value></value>
    </File>
    <File>
        <name>outmodel.sas7bdat</name>
        <value>om</value>
    </File>
    <File>
        <name>Output.spk</name>
        <value></value>
    </File>
    <File>
        <name>Format.sas7bcat</name>
        <value></value>
    </File>
    <File>
        <name>Dataprep.sas</name>
        <value></value>
    </File>
    <File>
        <name>Notes.txt</name>
        <value></value>
    </File>
</FileList>
</Model>

```

Example

Example Code A2.4 Registering a Generic Model

```

/*****
/* Register a SAS Code Model By Folder */
*****/

Options nomlogic nomprint nospool;

/*****
/* Load and access the SAS Model Manager macro code. */
*****/

Filename MMAccess catalog 'SASHELP.modelmgr.AccessMacros.source';
%include MMAccess;

/* Fileref to the encoded password */

FILENAME pwfile 'my-network-path\pwfile';

/*****
/* Set the SAS AP Server variables. */
*****/

%let _MM_ServerName = myserver.com;
%let _MM_PortNumber = 6411;
%let _MM_User = miller;
data _null_;
    infile pwfile obs=1 length=1;
    input @;
    input @1 line $varying1024. 1;

```

```

        call symput('_MM_Password', substr(line,1,1));
run;

/*****
/* If you use a UUID to specify the target, then      */
/* the value for the _MM_RepositoryId must be set.    */
*****/

%let _MM_RepositoryId = ModelManagerDefaultRepo;

/*****
/* Specify the location of the folder.                */
*****/

%let modelFolder = c:\myModel;
%let hmeq2009 = //ModelManagerModelRepos/MMRoot/HomeEquity/HMEQ/2009;

/*****
/* Set to detect failure in case macro load fails    */
/* and register the models in SAS Model Manager.     */
*****/

%let _MM_RC= -1;

%MM_RegisterByFolder(VersionId=&hmeq2009, ReportDir=&modelFolder, Trace=ON);

/*****
/* Display the SAS Model Manager-defined variables.  */
*****/

Options nosource;
%PUT _MM_RC = &_MM_RC;
Options source;

```

%MM_CreateModelDataset Macro

Overview of the %MM_CreateModelDataset Macro

The %MM_CreateModelDataset macro creates a data set that contains information about models. SAS Model Manager provides information for all models in the specified model repository path. You can specify MMRoot, an organizational folder, a project, a version, or a model. The data set contains the information for all models that exist under the specified path.

Syntax

```

%MM_CreateModelDataset (
    mDatasetName = name-of-data-set,
    smmPath=folder-project-verion-or-model-path <, Trace=ON | OFF>)
;

```

Arguments

mDatasetName = *name-of-data-set*

specifies the name of the data set that the macro creates. The macro can be created in a data set that you specify by using a two-level name in the form *libref.filename*.

Default: work.models

Required: No

`smmPath=folder-project-version-or-model-path`

specifies the path from which to obtain the model data. If the path is a folder, the data set contains model information for all models under that folder. If the path is a project, the data set contains model information for all models under that project. If the path is a version, the data set contains model information for all models under that version. If the path is a model, the data set contains model information for only that model.

Default: MMRoot

Required: No

`Trace=ON | OFF`

specifies whether to supply verbose trace messages to the SAS log.

Default: OFF.

Example: `Trace=on`

Example

Example Code A2.5 Extracting Model Information

```

/*****
/* Create a data set to contain model information */
*****/

Options nomlogic nomprint nospool;

/*****
/* Load and access the SAS Model Manager macro code. */
*****/

Filename MMAccess catalog 'SASHELP.modelmgr.AccessMacros.source';
%include MMAccess;

/* Fileref to the encoded password */

FILENAME pwfile 'my-network-path\pwfile';

/*****
/* Set the SAS AP Server variables. */
*****/

%let _MM_ServerName = myserver.com;
%let _MM_PortNumber = 6411;
%let _MM_User = miller;
data _null_;
    infile pwfile obs=1 length=1;
    input @;
    input @1 line $varying1024. 1;
    call symput('_MM_Password',substr(line,1,1));
run;

/*****
/* If you use a UUID to specify the model, then */
/* the value for the _MM_RepositoryId must be set. */
*****/

```

```

/*****/

%let _MM_RepositoryId = ModelManagerDefaultRepo;

/*****/
/* Specify the location of the data set and model */
/* path. */
/*****/

libname modelDS = 'c:\myModel\ModelInfo';
%let hmeq2009 = //ModelManagerModelRepos/MMRoot/HomeEquity/HMEQ/2009;

/*****/
/* Set to detect failure in case macro load fails */
/* and create the model data set. */
/*****/

%let _MM_RC= -1;

%MM_CreateModelDataset(mDatasetName=modelDS.models,
    smmpath=//ModelManagerDefaultRepo/MMRoot/DDHMEQ/HMEQ/2009/Models/
    Regression,
    Trace=ON);

/*****/
/* Display the SAS Model Manager-defined variables. */
/*****/

Options nosource;
%PUT _MM_RC = &_MM_RC;
Options source;

```


Appendix 3

Properties

General Properties	283
System Properties	284
Specific Properties for a Project	285
User-Defined Properties	287
Organization-Specific User-Defined Properties	287
SAS User-Defined Properties	288
Specific Properties for a Version	289
Specific Properties for Milestones and Tasks	290
Specific Properties for a Model	291
Scoring Task Properties	293
Result Set Properties	294

General Properties

Here is a list of general properties that describe how the repository component is named, created, and last updated.

Property Name	Description
Name	Identifies the name of the component. This property is read-only. You can assign the name when the component is created, a model is imported, or a report is generated.
Description	Specifies user-defined information about the component. You can modify this property.
Owner	Specifies the name of the user that created the component. This property is read-only. SAS Model Manager assigns the value when the component is created.
Creation Date	Specifies the date that the component was created. This property is read-only.

Property Name	Description
Modification Date	Specifies the date that the component was last modified. This property is read-only. SAS Model Manager assigns the current date when a change occurs.

System Properties

Here is a list of the system properties that describe the physical storage attributes of a repository component. To view system properties, click the + icon beside the System Properties heading to expand the section.

Property Name	Description
UUID	Specifies the universal unique identifier (UUID), which is a case sensitive, 36-character string that uniquely identifies the repository component. If the component is renamed, an application that uses the UUID always maintains the relationship between components. You can also use the UUID to query the repository. An example UUID is cca1ab08-0a28-0e97-0051-0e3991080867 .
SMMPath	Specifies the SAS Model Manager path (SMMPath) that is a network path to the directory that contains the model folders and files. The format for an SMM version path is //<RepositoryID>/MMRoot/<Folder>/<Project>/<Version>/<ComponentType> . A SAS Model Manager DATA Step client uses this information. For example, the DSC uses the SMMPath to identify the location to import a model.
URL	Specifies the Uniform Resource Locator (URL) that is the external Web address for the SAS Content Server location that stores model folders and files. You can paste the address in a Web browser to view the storage location.
Entity Key	Specifies a key that provides access data and metadata in the SAS Model Manager repository. The key consists of a component type, repository name, and a path to the component.
Repository Name	Specifies the name of the repository that contains the component.

Specific Properties for a Project

Here is a list of the specific properties for a project.

Property Name	Description
Project Input Table	Specifies the input variables that can be used by all models within a SAS Model Manager project.
Project Output Table	Specifies the list of output variables that must be created by all models within a SAS Model Manager project. If a model output variable is not the same as the project output variable, you can select the model in the Project Tree and map the model output variable to the project output variable.
Default Test Table	Specifies a default SAS data set that can be used to create model assessment reports such as dynamic lift charts.
Default Scoring Task Input Table	Specifies a default SAS data set that is used as the input data table for all scoring tasks within the SAS Model Manager project. If you specify a value for the Default Scoring Task Input Table property, the value is used as the default input table in the New Scoring Task window.
Default Scoring Task Output Table	Specifies a default SAS data set that defines the variables to keep in the scoring results table and the scoring task output table. If you specify a value of the Default Scoring Task Output Table property, the value is used as the default output table in the New Scoring Task window.
Default Performance Table	<p>Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager project.</p> <p>The value of the Default Performance Table property is used as the default value for the Performance data source field in the Define Performance Task wizard if a default performance table is not specified for a version or for the model.</p>
Default Train Table	<p>The train table is optional and is used only as information. However, when a value is specified for a model's Default Train Table property, it is used to validate Teradata scoring functions when a user publishes the associated project champion model to a Teradata database.</p> <p>The value of the Default Train Table property is used to validate Teradata scoring functions only if a default train table is not specified for a version or for the model.</p>

Property Name	Description
State	<p>Specifies the current state of the project:</p> <p>In Development specifies the time period from the project start to the time where the champion model is in a production environment.</p> <p>Active specifies the time period where the champion model is in a production environment.</p> <p>Inactive specifies the time period when a project is temporarily suspended from the production environment.</p> <p>Retired specifies that the champion model for this project is no longer in production.</p>
Default Channel	Specifies the channel that is used, by default, to publish a project. The default channel appears in the Channel box of the Channel Usage window.
Default Version	Specifies the version that contains the champion model in a production environment.
Model Function	Specifies the type of output that your predictive model project generates. The Model Function property that you specify affects the model templates that SAS Model Manager provides when you are ready to import models into one of your project's version folders. Once declared, the Model Function property for a project cannot be changed. Ensure that the types of models that you are going to use in the project fit within the selected model function type. For more information about the types of model functions, see Types of Model Functions on page 51 .
Interested Party	Specifies any person or group that has an interest in the project. For example, an interested party would be the business department or the business analyst whose request led to the creation of a SAS Model Manager project.
Training Target Variable	Specifies the name of the target variable that was used to train the model.
Target Event Value	The target variable value that defines the desired target variable event.
Class Target Values	For class, nominal, or interval targets, the set of possible outcome classes, separated by commas. For example, binary class target values might be 1, 0 or Yes, No . Nominal class target values might be Low, Medium, High . These values are for information only.

Property Name	Description
Output Event Probability Variable	The output event probability variable name, when the Model Function property is set to Classification .
Output Segmentation Variable	The output segmentation variable name, when the Model Function property is set to Segmentation .

Table A3.1 Types of Model Functions

Model Function	Description	Example
Analytical	Function for any model that is not Prediction, Classification, or Segmentation.	None
Prediction	Function for models that have interval targets with continuous values.	The score output of a prediction model could estimate the weight of a person. The output of a model would be P_Weight.
Classification	Function for models that have target variables that contain binary, categorical or ordinal values.	DEFAULT_RISK = {Low, Med, High}
Segmentation	Function for segmentation or clustering models.	Clustering models
Any	Specify Any when you import a SAS code model and you want a choice of the model template to use in the Local Files window. When you specify Any , SAS Model Manager lists the available model templates in the Choose a model template list in the Local Files window.	None

User-Defined Properties

Organization-Specific User-Defined Properties

You can create user-defined properties to keep information specific to your organization or business in the appropriate folders in the Project Tree. User-defined property names must begin with a letter. To create a user-defined property, follow these steps:

1. In the Properties view, right-click the mouse and select **Add User-Defined Property**. The Add User-Defined Property window opens.
2. In the **Name** field, add a name beginning with a letter. Spaces are not allowed in the name.
3. In the **Value** field, type a value for the user-defined property. Click **OK**. The user-defined property is added to the Properties view under **User-Defined Properties**.

SAS User-Defined Properties

SAS creates some user-defined properties that are used by the SAS Real-Time Decision Manager and for scoring that is performed using SAS In-Database processing, such as scoring within Teradata.

Here are the user-defined properties that are used by the SAS Real-Time Decision Manager. These fields must be completed by the user:

Property Name	Description
KeyType	Specifies a value of C if TableKey has a type of character specifies a value of N if TableKey has a type of numeric
TableKey	Specifies the primary key in the table that the model uses for scoring
PreCode	Specifies the libref definition that is used to access the table that the model uses for scoring
ScoringInputTable	Specifies the name of the input table that the model uses for scoring.

When you publish a Teradata scoring function for the first time, SAS creates some project user-defined properties. Some of these property values are assigned by SAS after you complete the **Function Name** field in the Publish Teradata Scoring Function window.

Here are the Teradata scoring function user-defined properties.

Property Name	Description
DbmsTable	Specifies the name of the input table that is used in Teradata scoring functions. This field should be specified for the project properties before you publish a Teradata scoring function.
ScoringFunctionName	Specifies the user-defined portion of the Teradata scoring function name. You can modify this property value only by changing the name in the Publish Teradata Scoring Function window.

Property Name	Description
ScoringFunctionPrefix	<p>Specifies a prefix for the Teradata scoring function name. The prefix has a length of 10 characters and in the format of "Yyyymmddnnn". You cannot modify this value. The naming convention for the prefix is the following:</p> <ul style="list-style-type: none"> • Y is a literal character and is fixed for all prefixes. • yyymmdd is the GMT timestamp of when you select the Publish Teradata Scoring Function menu option. • nnn is a counter that increments by one each time the scoring function is successfully completed.

Specific Properties for a Version

Here is a list of the specific properties for a version.

Property Name	Description
Default Scoring Task Input Table	<p>Specifies a default SAS data set that is used as the input data table for all scoring tasks within the SAS Model Manager version. If you specify a value for the Default Scoring Task Input Table property, the value is used as the default input table in the New Scoring Task window if a default scoring task input table is not specified for the model.</p>
Default Scoring Task Output Table	<p>Specifies a default SAS data set that defines the variables to keep in the scoring results table of the scoring task. If you specify a value for the Default Scoring Task Output Table property, the value is used as the default output table in the New Scoring Task window if a default scoring task output table is not specified for the model.</p>
Default Performance Table	<p>Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager version.</p> <p>The value of the Default Performance Table property is used as the default value for the Performance data source field in the Define Performance Task wizard if a default performance table is not specified for the model.</p>

Property Name	Description
Default Train Table	<p>The train table is optional and is used only as information. However, when a value is specified for a model's Default Train Table property, SAS Model Manager does the following:</p> <ul style="list-style-type: none"> • uses default train table to validate Teradata scoring functions when a user publishes the associated project champion model to a Teradata database • checks the Validate scoring results box in the Publish Teradata Scoring Function window. <p>The value of the Default Train Table property is used to validate Teradata scoring functions only if a default train table is not specified for the model.</p>
State	Specifies the current status of the version.
Champion Model ID	Specifies the UUID for the champion model in the version, if one exists.
Frozen Date	Specifies the date that the version was frozen.
Production Date	Specifies the date that the status of the Production milestone task in the version's life cycle was changed from Started to Complete .
Retired Date	Specifies the date that the status of the Retire milestone task in the version's life cycle was changed from Started to Complete .

Specific Properties for Milestones and Tasks

Here is a list of the milestone properties.

Property Name	Description
Actual Start Date	Specifies the actual date that the first task for the milestone is started. This property is Read-only.
Actual End Date	Specifies the actual date when all tasks for the milestone are finished. This property is read-only. SAS Model Manager assigns the value when the status of every milestone task is set Completed .

Property Name	Description
Planned Start Date	Specifies the expected date to start the first task for milestone.
Planned End Date	Specifies the expected date to complete all tasks for the milestone.

Here is a list of task properties:

Property Name	Description
Status	Specifies the status of task. Possible values are Not Started , Started , Completed , or Approved .
Completed Date	Specifies the date when the task is finished. This property is read-only. SAS Model Manager assigns the value when the status of the milestone task was changed to Completed .
Completed By	Specifies the name of the user who completed the task. This property is Read-only.
Approved Date	Specifies the date when completion of the task is approved. This property is Read-only.
Approved By	Specifies the name of the user who approved completion of the task. This property is Read-only.
Planned Completion Date	Specifies the expected date to complete the task.
To Be Completed By	Specifies the user who is responsible for completing the task.
To Be Approved By	Specifies the user who can approve that the task is completed.

Specific Properties for a Model

Here is a list of specific properties for a model that identify the fundamental model data structures and some of the critical model life cycle dates. Where applicable, project-based or version-based data structures automatically populate properties for model-based data structures.

Property Name	Description
Default Scoring Task Input Table	Specifies a default SAS data set that is used as the input data table for all of scoring tasks within the SAS Model Manager project. The model's Default Scoring Task Input Table property inherits the property value from the associated version or project, if one is specified.
Default Scoring Task Output Table	Specifies a default SAS data set that defines the variables to keep in the scoring results table and the scoring task output table. The model's Default Scoring Task Output Table property inherits the property value from the associated version or project, if one is specified.
Default Performance Table	Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager project. A model's Default Performance Table property inherits the property value from the associated version or project, if one is specified. If you do not specify a performance table, some of the SAS Model Manager Model Monitoring reports might not be enabled.
Default Train Table	The train table is optional and is used only as information. However, when a value is specified for a model's Default Train Table property, it is used to validate Teradata scoring functions when a user publishes the associated project champion model to a Teradata database.
Expiration Date	Specifies a date property by which the selected model is obsolete or needs to be updated or replaced. This property is for informational purposes and is not associated with any computational action by SAS Model Manager. This property is optional.
Model Label	Specifies a text string that is used as a label for the selected model in the model assessment charts that SAS Model Manager creates. If no value is provided for the Model Label property, SAS Model Manager uses the text string that is specified for the Model Name property. The Model Label property can be useful if the Model Name property that is specified is too long for use in plots. This property is optional.
Subject	Specifies a text string that is used to provide an additional description for a model, such as a promotional or campaign code. This property is for informational purposes and is not associated with any computational action by SAS Model Manager. This property is optional.
Algorithm	Specifies the computational algorithm that is used for the selected model. This property cannot be modified.

Property Name	Description
Function	Specifies the SAS Model Manager function class that was chosen when the SAS Model Manager associated project was created. The Function property specifies the type of output that models in the predictive model project generate. For more information, see “Overview of Importing Models” on page 79 .
Modeler	Specifies the Modeler ID or, when Modeler ID is missing, specifies the user ID of the individual that created the model which is stored in the SPK file for SAS Enterprise Miner models. Otherwise, the modeler can be specified during model import for local files into SAS Model Manager.
Tool	Specifies whether the imported model came from SAS Enterprise Miner or from other modeling tools.
Tool Version	Specifies the version number of the tool that is specified in the Tool property.
Score Code Type	Specifies whether the imported model score code is a DATA step fragment, ready-to-run SAS code, or a PMML file. Valid values are Data Step, SAS Program, and PMML.
Template	Specifies the SAS Model Manager model template that was used to import the model and to create pointers to its component files and metadata.
Copied From	Specifies where the original model is if this model is copied from another model in the SAS Model Manager repository.
Target Variable	Specifies the name of the target variable for a classification or prediction model. This property can be ignored for segmentation, cluster, and other models that do not use target variables. For example, if a model predicts when GENDER=M, then the target variable value is GENDER .
Target Event Value	Specifies a value for the target event that the model attempts to predict. This property is used only when a value is specified for the Target Variable property. For example, if a model predicts when GENDER=M, then the target event value is M .

Scoring Task Properties

Here is a list of the scoring task properties that provide information specific to the scoring task.

Property Name	Description
Scoring Task Type	Specifies a value of Test or Production for the type of scoring task.
Workspace Server	Specifies the name of the Workspace Server in which SAS Model Manager is connected. This value is taken from the SAS Metadata Repository.
Model Name	Specifies the name of the model whose score code is to be executed on the Workspace Server. This value is set when the scoring task is created and cannot be modified.
Input Table	Specifies the name of the input table (data source) to be used in scoring. This value is set when the scoring task is created and cannot be modified.
Output Table	Specifies the name of the output table to be used in scoring. This value is set when the scoring task is created. If the Scoring Task Type is Test then this property identifies the name of the output file (<i>output_filename.sas7bdat</i>) that is created by the SAS Workspace Server when the score code is executed. Upon creation the output file is placed in the scoring task's folder. If the Scoring Task Type is Production , then this identifies the output table where the results of the scoring are written.

See Also

- [“Create a Scoring Task” on page 113](#)
- [“Modify a Scoring Task” on page 114](#)
- [“Execute a Scoring Task” on page 115](#)

Result Set Properties

Here is a list of result set properties that provide information specific to the scoring task.

Property Name	Description
Number of Observations	This is an editable field whose value is used only when the Scoring Task Type is set to Test . It specifies how many observations are to be read from the scoring task input table. This enables you to limit the amount of records written to the scoring task output table on the SAS Content Server in order to reduce operation costs. If a value is not specified, the default value of 1000 rows is used for the number of observations.

See Also

- [“Create a Scoring Task” on page 113](#)
- [“Modify a Scoring Task” on page 114](#)

Appendix 4

SAS Model Manager In-Database Scoring for Teradata

Overview of SAS Model Manager In-Database Scoring for Teradata	297
About	297
Getting Started	298
Using the Java Scoring API	298
Overview of Using the Java Scoring API	298
Establishing a JDBC Connection	299
Logging a Scoring Request	299
Submitting a Scoring Request	299
Using Dynamic Predictors	300
Parsing the Scoring Result	300
Examples	301

Overview of SAS Model Manager In-Database Scoring for Teradata

About

SAS Model Manager In-Database Scoring for Teradata (Java Scoring API) works with SAS Model Manager. When you publish a Teradata scoring function for a project, SAS Model Manager exports the project's champion model to the SAS Metadata Repository and the SAS Scoring Accelerator creates scoring functions under the default version. The scoring functions can be deployed inside Teradata based on the project's champion model score code. The Java Scoring API provides an interface that enables you, as an application developer, to access a scoring model that is stored as a function in the Teradata EDW. The application developer is responsible for creating or maintaining the Java Scoring API that submits a scoring request and parses the scoring result that is returned.

Here is an example from a call center: a customer service representative responds to a customer request for an increase in credit limit. While on the telephone, the customer service representative gathers and verifies new information about salary, length of time on the job, and so on. The representative enters this new information into the scoring application and the developer submits a scoring query with the new information to the previously registered Teradata scoring function. The scoring result is returned to the application, and the representative then uses it to determine whether to approve or deny the request, along with the new limit, if the request is approved.

For more information, see [“Using the Java Scoring API” on page 298](#) or [“Publish Teradata Scoring Functions” on page 164](#).

Getting Started

In order for the Java Scoring API to access a scoring function that was published to Teradata by the SAS Model Manager Client, the application needs to have two sets of jars made available to it. There are a number of ways to accomplish this, one of which is adding them to the CLASSPATH. Choose the one that is appropriate for your environment. The following jars need to be included:

- **`sas.modelmanager.td.jar`**
- Teradata JDBC jars (**`terajdbc4.jar`** and **`tdgssconfig.jar`**)

The System Administrator who installed SAS 9.2 can provide you with the **`sas.modelmanager.td.jar`**. For more information, see *SAS Model Manager: Administrator's Guide*.

The Teradata JDBC jars can be found on the Teradata Web site (<http://www.teradata.com>). Select **Support & Downloads** ⇒ **Downloads** ⇒ **Teradata JDBC Driver**. Select the version of the JDBC driver that supports Teradata version 12. SAS has validated the minimum version requirements of 12.0.0.106 for use with SAS Model Manager 2.2. It is possible to use updated versions of the third-party software. For more information, see the SAS Third Party Software Requirements - Baseline and Higher Support at the URL http://support.sas.com/resources/thirdpartysupport/baseline_plus.html#.

In addition to the jar files, you also need the following information from the Teradata DBA to use the Java Scoring API:

- Server host name of where the Teradata EDW resides
- Database name
- User ID
- Password

Using the Java Scoring API

Overview of Using the Java Scoring API

Access to the Java Scoring API to submit scoring requests and process the returned results is an added benefit of using SAS Model Manager solution. The Java Scoring API consists of the following two classes:

`com.sas.modelmanager.td.Scoring`

is used to establish the connection and submits the scoring requests.

`com.sas.modelmanager.td.Results`

is the class that helps you parse the returned results.

To produce a scoring result using the Java Scoring API, follow these steps:

1. Establish a JDBC Connection.
2. (optional) Enable logging of scoring actions.
3. Set the **`projectId`**, parameters and predictors.
4. Parse the scoring result.

5. Compile and run your Java Scoring application (for example ScoreAPI.java) to score your model. The scoring results are displayed.

Establishing a JDBC Connection

There are two constructors for the **Scoring** class. Which one you use depends on how your application accesses Teradata. For example, if your application makes multiple Teradata requests and therefore has set up a JDBC connection pool, then it is best to use the constructor for the **Scoring** class that accepts an existing JDBC connection. By contrast, if your Teradata requests are infrequent and you do not keep a JDBC connection open, then you should use the scoring constructor that accepts host name, user ID, password, and database name.

When you use an existing connection, you make the following call in your code:

```
Scoring scoringTask = new Scoring(existingJDBCConnection, userId);
```

existingJDBCConnection

This argument is the connection that your application has already established and therefore the one you want the scoring function to use.

userId

This argument should be set to null in this version of the interface.

Note: When an existing connection is used, the Java Scoring API does not release the connection after the scoring task is complete.

If you want the Java Scoring API to establish the connection (and release it when the scoring task is complete) you make the following call in your code:

```
Scoring scoringTask = new Scoring(fullyQualifiedHostName, userId, password,
databaseName);
```

Logging a Scoring Request

For many different reasons including legal ones, you might need to access the scoring log. The scoring log shows what scoring request was made, what parameters were used and what results were returned. So by default the Java Scoring API writes log messages to the SAS Model Manager Teradata table called **scoring_log**. To turn off logging, you can use the following call:

```
scoringTask.setWriteScoringLog(false);
```

Submitting a Scoring Request

There are two ways to submit a scoring request. Both use the same method call. In one case you can override some of the values in the customer's record in the table (that is, you want to pass in dynamic predictors). In the other case you want to use only the information that is available in the customer record. Either way, you use the same scoring function:

```
scoringTask.score(projectId, whereClause, dynamicPredictors);
```

projectId

The **projectId** is a Java string. The value is a unique identifier that is set when a project is created in SAS Model Manager. To obtain this value, open the SAS Model Manager Client and navigate to the desired project. On the **Properties** tab, click the

plus sign to expand the **System Properties**. You can then see a property value pair where the property name is **UUID**. This is the value that you use as the **projectId**.

For an example, see [“SAS Model Manager Project System Properties” on page 301](#).

whereClause

The **whereClause** is a Java string. It is the clause that you would use to uniquely select the customer record if you were submitting an SQL select statement. The clause can be as simple or as complex an SQL WHERE clause as you need.

dynamicPredictor

The **dynamicPredictor** is a Java string. This argument is used to specify values that are different from the values stored in the table from which the customer record is being retrieved. If you are using the existing values in the table, you can set the value of this argument to null, or to an empty string.

Example Code A4.1 Scoring Request

```
scoringTask.score("ea04019b-0a29-0910-012e-7c6a5a8f6e2c", "custId=44", "");
```

For an example of the scoring results for a scoring request, see [“Java Scoring API Scoring Results” on page 302](#).

Using Dynamic Predictors

Specifying dynamic predictors as the third argument to the **score()** method enables you to override some or all of the column values for the customer record you are retrieving. The syntax of this string is a comma separated list of name and value pairs. When writing this string, use the SET syntax of the SQL UPDATE statement as an example.

Note: The scoring request does not update your data tables. The existing column values are used for any column values that you do not override.

Example Code A4.2 Dynamic Predictors Scoring Request

```
scoringTask.score("ea04019b-0a29-0910-012e-7c6a5a8f6e2c",  
"custId=44", "yoj=17.0,debtinc=45.0,job='OFFICE'");
```

Note: The string values in the dynamic predictors are enclosed in single quotation marks.

Parsing the Scoring Result

After the successful execution of the **score()** method, the value that is returned is a Java string. The structure of that string depends on the output variables that were specified for the model. However, using the **Results** object to parse the returned value provides a simple mechanism for accessing the returned values. The type of each value can be either a string or a double array.

In order to help you parse the returned scoring result you can pass the returned string as the single argument to the **Results** class. For example:

```
String score = scoringTask.score("ea04019b-0a29-0910-012e-7c6a5a8f6e2c",  
"custId=44", "");  
Results results = new Results(score);
```

The constructor for the **Results** class parses the passed-in string. Using this constructor enables you to do several things:

- determine the size of the resulting array (using **size()**)
- determine the names of the keys, if any (using **keys()**)

- use the `get` method to find a specific value (using `get()`)

After you have created the **Results** object, use the following methods to access the contents of the **Results** object:

```
int size = results.size();
```

This method returns a one-based integer that equals the number of output variables that were returned after the execution of the scoring task. A value of zero can be returned. If the value is not zero, use this value in a loop with the `results.get(ndx)` method call.

```
String[] names = results.keys();
```

This method returns a string array of the names of each of the output variables that were returned. These names are the keys that you would use in the `results.get(key)` method call. The array can have a length of zero.

```
Object value = results.get(key);
```

This method takes a string as the key, looks up the key, and returns the value as an object. If the key does not exist, then NULL is returned. The type of the returned object can be either a string or a double array. After each call to the `get(key)` method you can use the `isDouble()` method of the **Results** object to determine the object type of the returned value.

```
Object value = results.get(ndx);
```

This method takes an integer as the argument, and returns the value at the index position as an object. If the index is out of range, then NULL is returned. The type of the returned object can be either a string or a double array. After each call to the `get(ndx)` method you can use the `isDouble()` method of the **Results** object to determine the object type.

Examples

SAS Model Manager Project System Properties

Home Equity	
Properties	Input Variables
Output Variables	
[-] General Properties	
Name	Home Equity
Description	Home Equity Example
Owner	mdlmgadmin
Creation Date	Feb 1, 2009
Modification Date	Feb 9, 2009
[-] System Properties	
UUID	34b6ccd1-0a29-0c76-0082-d60311dbde66
SMM Path	///ModelManagerDefaultRepo/MMRoot/Kristen/Home%20Equity
URL	http://emm02.na.sas.com:8080/SASContentServer/repository/default/ModelMa...
Entity Key	Project+dav:///ModelManagerDefaultRepo/MMRoot/Kristen/Home%20Equity/Pro...
Repository Name	ModelManagerDefaultRepo
[-] Specific Properties	
[-] User-Defined Properties	

Java Scoring API Scoring Results

```
Creating new TD JDBC connection 'jdbc:teradata://sl9-1200/DATABASE=pubs, pubs,
pubs1' took 7422 milliseconds.
Result=score=0.15704010353232
ScoreAPI Time: 1781
```

Glossary

analytical model

in SAS Model Manager, a model that is neither a classification model, nor a prediction model, nor a segmentation model.

baseline

using operational data, the prediction accuracy of the initial performance monitoring task or batch program.

bin

a grouping of predictor variable values that is used for frequency analysis.

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.

classification model

a predictive model that has a categorical, ordinal, or binary target.

clustering model

a model in which data sets are divided into mutually exclusive groups in such a way that the observations for each group are as close as possible to one another, and different groups are as far as possible from one another.

component files

the files that define a predictive model. Component files can be SAS programs or data sets, XML files, log files, SPK files, or CSV files.

data source

a data object that represents a SAS data set or a DBMS table. SAS Model Manager has seven types of data sources: project input and output tables, test tables, scoring task input and output tables, performance tables, and training tables.

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, the end of the job, or the semicolon that immediately follows lines of data. The DATA step enables you to read raw data or other SAS data sets and to use programming logic to create a SAS data set, to write a report, or to write to an external file.

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.

fileref

a short name (or alias) for the full physical name of an external file. A SAS FILENAME statement maps the fileref to the full physical name.

folder

an object that contains other container objects and files.

format

a pattern or set of instructions that SAS uses to determine how the values of a variable (or column) should be written or displayed. SAS provides a set of standard formats and also enables you to define your own formats.

Gini coefficient

a benchmark statistic that is used to summarize the predictive accuracy of a model that has a binary target. The Gini coefficient represents the area under the ROC curve.

Gini index

See Gini coefficient

hold-out data

a portion of the historical data that is set aside during model development. Hold-out data can be used as test data to benchmark the fit and accuracy of the emerging predictive model. See also model.

informat

a pattern or set of instructions that SAS uses to determine how data values in an input file should be interpreted. SAS provides a set of standard informats and also enables you to define your own informats.

input variable

a variable that is used in a data mining process to predict the value of one or more target variables.

Kolmogorov-Smirnov chart

a chart that shows the measurement of the maximum vertical separation, or deviation between the cumulative distributions of events and non-events.

libref

a short name (or alias) for the full physical name of a SAS library. A SAS LIBNAME statement maps the libref to the full physical name. A libref is the first part of a multi-level SAS filename and indicates the SAS library in which a SAS file is stored. For example, in the name SASUSER.ACCTS, SASUSER is the libref, and ACCTS is a file in the library that the SASUSER libref refers to. See also SAS library.

life cycle phase

a collection of milestones that complete a major step in the process of selecting and monitoring a champion model. Typical life cycle phases include development, test, production, and retire.

logistic regression

a form of regression analysis in which the target variable (response variable) represents a binary-level, categorical, or ordinal-level response.

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

a description or definition of data or information.

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 assessment

the process of determining how well a model predicts an outcome.

model function

the type of statistical model, such as classification, prediction, or segmentation.

neural networks

a class of flexible nonlinear regression models, discriminant models, data reduction models, and nonlinear dynamic systems that often consist of a large number of neurons. These neurons are usually interconnected in complex ways and are often organized into layers. See also neuron.

observation

a row in a SAS data set. All of the data values in an observation are associated with a single entity such as a customer or a state. Each observation contains either one data value or a missing-value indicator for each variable.

organizational folder

a folder in the SAS Model Manager Project Tree that is used to organize project and document resources. An organizational folder can contain zero or more organizational folders in addition to other objects.

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 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 (SAS catalogs; SAS data sets; various types of SAS databases, including cubes; and SAS SQL views), binary files (such as Excel, GIF, JPG, PDF, PowerPoint and Word files), HTML files (including ODS output), reference strings (such as URLs), text files (such as SAS programs), and viewer files (HTML templates that format SAS file items for viewing).

performance table

a table that contains response data that is collected over a period of time. Performance tables are used to monitor the performance of a champion model that is in production.

PMML

See Predictive Modeling Markup Language.

prediction model

a model that predicts the outcome of an interval target.

Predictive Modeling Markup Language

an XML-based standard for representing data mining results for scoring purposes. PMML enables the sharing and deployment of data mining results between applications and across data management systems. Short form: PMML.

process flow diagram

a graphical representation of the various data mining tasks that are performed by individual Enterprise Miner nodes during a data mining analysis. A process flow diagram consists of two or more individual nodes that are connected in the order in which the data miner wants the corresponding statistical operations to be performed. Short form: PFD.

profile data

information that consists of the model name, type, length, label, format, level, and role.

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.

property

any characteristics of an object that collectively determine the object's appearance and behavior. UUIDs, table names, and input and output variable names are types of properties.

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. See also publish.

Receiver Operating Characteristic chart

a chart that plots the specificity of binary data values against 1-specificity of binary data values. A ROC chart is used to assess a model's predictive performance. Short form: ROC

ROC

See Receiver Operating Characteristic

SAS code model

a SAS program or a DATA step fragment that computes output values from input values. An example of a SAS code model is the LOGISTIC procedure.

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 data files contain data values in addition to descriptor information that is associated with the data. SAS data views contain only the descriptor information plus other information that is required for retrieving data values from other SAS data sets or from files whose contents are in other software vendors' file formats. See also descriptor information.

SAS Metadata Repository

a repository that is used by the SAS Metadata Server to store and retrieve metadata. See also SAS Metadata Server.

SAS package file

See package file

scoring

the process of applying a model to new data in order to compute outputs.

scoring function

a user-defined function that is created by the SAS Scoring Accelerator for Teradata from a scoring model and that is deployed inside the Teradata Enterprise Data Warehouse (EDW).

scoring task

a process that executes a model's score code.

scoring task input table

a table that contains the variables and data that are used as input in a SAS Model Manager scoring task.

scoring task output table

a table that contains the output variables and data that result from performing a SAS Model Manager scoring task. Before executing a scoring task, the scoring task output table defines the variables to keep as the scoring results.

segmentation model

a model that identifies and forms segments, or clusters, of individual observations that is associated with an attribute of interest.

target event value

for binary models, the value of a target variable that a model attempts to predict. In SAS Model Manager, the target event value is a property of a model.

target variable

a variable whose values are known in one or more data sets that are available (in training data, for example) but whose values are unknown in one or more future data sets (in a score data set, for example). Data mining models use data from known variables to predict the values of target variables.

test table

a SAS data set that is used as input to a model that tests the accuracy of a model's output.

training

the process of building a predictive model from data.

Universal Unique Identifier

a 36-character identifier that is used to uniquely identify an object. UUIDs are also called Global Unique Identifiers (GUIDs). Short form: UUID.

UUID

See Universal Unique Identifier.

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. Each SAS variable can have the following attributes: name, data type (character or numeric), length, format, informat, and label.

variable attribute

any of the following characteristics that are associated with a particular variable: name, label, format, informat, data type, and length.

version

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.

view

a standardized way of looking at project and model metadata. A view is also a way of looking at the resulting output that is based on model calculations.

Index

Special Characters

%MM_AddModelFile macro [249](#)
 %MM_CreateModelDataset macro [279](#)
 %MM_GetModel macro [223](#)
 %MM_GetModelFile macro [252](#)
 %MM_GetModels macro [222](#), [223](#)
 %MM_GetURL macro [256](#)
 %MM_Register macro [257](#)
 %MM_RegisterByFolder macro [274](#)
 %MM_RunReports macro [228](#)
 macro variables used by [226](#)

A

access macros [243](#)
 %MM_AddModelFile [249](#)
 %MM_CreateModelDataset [279](#)
 %MM_GetModelFile [252](#)
 %MM_GetURL [256](#)
 %MM_Register [257](#)
 %MM_RegisterByFolder [274](#)
 accessing [244](#)
 dictionary of [249](#)
 global macro variables and [247](#)
 identifying files used by [245](#)
 identifying model repository objects [244](#)
 required global macro variables [248](#)
 required tables [245](#)
 accessibility features [9](#)
 ad hoc reports [135](#), [137](#)
 compared with user-defined reports [136](#)
 creating [137](#)
 example [138](#)
 alert notifications [188](#)
 Analytical model template [85](#)
 analytical models [3](#)

Approvers

groups as, for life cycle templates [60](#)
 life cycle template participants [59](#)
 Model Manager Example Life Cycle
 Approvers [18](#)

assessing models [21](#)

Assessment Charts [192](#)

Assignees

groups as, for life cycle templates [60](#)
 life cycle template participants [59](#)
 Model Manager Example Life Cycle
 Assignees [18](#)

associating documents with folders [41](#)

attaching documents [41](#)

B

batch performance reports [207](#)
 accessing performance data set [228](#)
 copying example batch programs [209](#)
 creating folder structure for [207](#)
 defining report local folders and data sets [226](#)
 defining specifications [211](#)
 e-mail recipient specifications [214](#)
 encoding passwords [227](#)
 example code [229](#)
 export channel for [209](#)
 extracting champion model from a
 channel [222](#)
 job scheduling specifications [218](#)
 librefs for running [225](#)
 performance data for [209](#)
 prerequisites for running [207](#)
 project specifications [211](#)
 publishing champion model from project
 folder [207](#)

- report output in production mode 210
- report output in test mode 210
- report specifications 215, 219
- SAS code for running 225
- user ID and password for 210

C

- champion models 150
 - clearing 151
 - deploying 150
 - exporting 163
 - extracting from a channel 222
 - monitoring performance 22
 - monitoring process 190
 - performance monitoring reports 181
 - publishing for batch performance reports 207
 - replacing/retiring 177
 - requirements for 150
 - selecting new default version 178
 - setting 151
 - setting status 48
- channels
 - export channel for batch performance reports 209
 - extracting champion model from 222
 - publishing models to 159
- Characteristic reports 182, 183, 184
 - example 185
 - overview 183
 - performance index warnings and alerts 188
- Classification model template 85
- comparison reports
 - See* [model comparison reports](#)
- component files
 - model templates 86
- content files
 - scoring task content files 121
- current.sas7bdat data set 223

D

- Data Composition reports 182, 183
 - Characteristic report 182, 183, 184
 - Stability report 182, 183, 184
- data sets 6
 - containing model information 279
 - current.sas7bdat 223
 - performance data sets 190, 198, 228
 - performance tables 30
 - project input tables 28
 - project output tables 28
 - scoring task input tables 29
- data source tables 4

- adding 35
- deleting 36
- data sources 27
 - performance data sources 190
 - registering 28
- Data Sources perspective 10, 13
- DATA step
 - accessing performance data set 228
- data tables 27
 - scoring task input tables 32
- default version
 - clearing 155
 - for projects 154
 - selecting a new default version 178
 - setting 154
- Define Performance Task wizard
 - naming performance tables for use with 34
 - prerequisites for running 199
 - running 201
- degradation of models 183
- delivering models 21
 - See also* [model delivery](#)
- Delta reports 129
- deploying models 21, 149
 - champion models 150
 - default version for projects 154
 - freezing models 152
- documents
 - associating with folders 41
 - attaching to folders 41
 - saving 41
 - showing versions 41
 - viewing 41
- Dynamic Lift reports 131
 - creating 132
 - creating test tables 33
 - verifying model properties 131
 - verifying project properties 131
- dynamic predictors 300

E

- Edit menu 17
- encoding passwords 227
- environment, operational 4
- export channel
 - for batch performance reports 209
- exporting
 - models 162
 - project champion models 163
 - verifying model exports 164
- extracting
 - champion model from a channel 222
 - published models 161

F

File menu 16
 folder structure
 creating for batch performance reports 207
 folders
 associating documents with 41
 attaching documents to 41
 creating organizational folders 40
 project folder organization 44
 project folder tasks 44
 version folder organization 54
 version folder tasks 55
 freezing
 models 152
 versions 153

G

general properties 283
 general tasks 22
 Gini plots 186
 Gini Trend Chart 193
 global macro variables 247
 graphing scoring task results 118
 groups 4, 18

H

Help menu 17

I

importing models 21, 79
 from metadata repository 80
 importing package files from SAS
 Enterprise Miner 81
 mapping model variables to project
 variables 95
 model templates 84
 overview 79
 partial models 93
 PMML models 92
 SAS code models 83, 91
 setting model properties 94
 user-defined model templates 96
 In-Database Scoring for Teradata 165,
 297
 getting started 298
 Java Scoring API 298
 indexes
 warnings and alerts 188
 input data variable distribution shifts 184

J

Java Scoring API 297, 298
 establishing JDBC connection 299
 examples 301
 logging a scoring request 299
 parsing the scoring result 300
 specifying dynamic predictors 300
 submitting a scoring request 299
 JDBC connections 299
 job scheduling specifications
 batch performance reports 218

K

Kolmogorov-Smirnov (KS) plots 187
 KS Chart 193
 KS reports 187
 KS Trend Chart 193

L

librefs
 running batch performance reports 225
 Life Cycle perspective 10, 13
 life cycle templates 56
 creating 56, 60
 creating new versions 63
 groups as Assignees and Approvers 60
 middle-tier server user-templates
 directory 56
 milestone properties 64
 modifying 62
 participant roles 58
 participants 58
 properties 64
 SAS Model Manager Template Editor
 window 57
 selecting participants 59
 task properties 65
 template properties 64
 viewing 72
 life cycles 70
 definition 70
 milestone organization 70
 properties 73
 searching for tasks assigned to users
 239
 tasks 71
 updating milestone status 73
 Lift Trend chart 192
 Local Files method
 importing SAS code models 83
 logs
 logging a scoring request 299
 publishing Teradata scoring functions
 174

M

- macro variables
 - defining for user-defined reports 143
 - defining report local folders and data sets 226
 - global 247
 - used by %MM_RunReports macro 226
- macros
 - See also* [access macros](#)
 - accessing report macros 223, 225
- mapping variables
 - model variables to project variables 95
 - scoring task output variables 115
- menus 10, 16
- metadata
 - project metadata 49
- metadata repository
 - importing models from 80
 - viewing MiningResult objects in 164
- metadata tables
 - publishing Teradata scoring functions 175
- middle-tier server
 - user-templates directory 56
- milestones
 - organization of life cycle milestones 70
 - specific properties for 290
 - updating status 73
- MiningResult objects 162
 - viewing in metadata repository 164
- Model Assessment reports
 - performance index warnings and alerts 189
- model comparison reports 125
 - Delta 129
 - Dynamic Lift 131
 - input files 126
 - Model Profile 127
 - output files 126
 - viewing 134
- model component files 249
- model delivery 157
 - exporting models 162
 - publishing models 158
 - publishing Teradata scoring functions 164
- model function types 51, 287
- Model Input Variable Report 183
- model management process 6
- Model Manager Administrators 18
- Model Manager Advanced Users 18
- Model Manager Example Life Cycle Approvers 18
- Model Manager Example Life Cycle Assignees 18
- Model Manager Users 18
- Model Monitoring reports 182, 185
 - KS reports 187
 - monitoring Lift reports 185
 - monitoring ROC & Gini reports 186
- Model Profile reports 127
 - creating 127
- model repository objects 244
- Model Target Variable Report 183
- model templates 84, 96
 - component files 86
 - creating 96
 - FileList properties 101
 - modifying 98
 - properties 101
 - system and user properties 102
 - template properties 101
 - user model templates 85
 - user-defined 96
- model variables
 - mapping to project variables 95
- models
 - See also* [champion models](#)
 - assessing 21
 - data sets containing model information 279
 - degradation of 183
 - deploying 149
 - deploying and delivering 21
 - exporting 162
 - extracting published models 161
 - freezing 152
 - function types 46
 - importing 21, 79
 - management process 6
 - mapping model variables to project variables 95
 - publishing 158
 - publishing to a channel 159
 - registering 257, 274
 - searching for 236
 - specific properties for 103, 291
 - unfreezing 152
 - validating with model comparison reports 125
 - validating with user reports 135
 - verifying exports 164
- monitoring champion model performance 22
- monitoring Lift reports 185
- Monitoring reports 192
 - creating 193
 - output files 194
- monitoring ROC & Gini reports 186

N

naming performance tables 34

O

objects

deleting in Project Tree 42

operational environment 4

organization-specific user-defined
properties 287

organizational folders 40

output data variable distribution shifts
184

P

package files 81

creating 81

importing from SAS Enterprise Miner
81

publishing models 158

parsing scoring results 300

Partial Model Import utility 93

partial models

importing 93

participants

selecting for life cycle templates 59

passwords

encoding 227

for batch performance reports 210

performance

data for batch performance reports 209

monitoring champion model

performance 22

performance data sets

creating performance reports 198

DATA step for accessing 228

versus performance data sources 190

performance data sources

versus performance data sets 190

performance indexes

warnings and alerts 188

performance monitoring reports 181

See also [batch performance reports](#)

Data Composition reports 182, 183

formatting 192

Model Monitoring reports 182, 185

Monitoring reports 192

performance data sets versus

performance data sources 190

performance index warnings and alerts
188

SAS programs for creating 206

Summary reports 182, 183

types of 182

viewing 195

performance reports 197

See also [batch performance reports](#)

creating with performance tasks 197

performance data sets and 198

prerequisites for running Define

Performance Task wizard 199

running Define Performance Task wizard
201

performance tables 28, 30, 34

creating 34

naming for use with Define Performance
Task wizard 34

performance tasks

creating reports with 197

prerequisites for running Define

Performance Task wizard 199

running Define Performance Task wizard
201

perspectives 10

Data Sources 13

Life Cycle 13

Projects 11

planning projects 45

PMML models

importing 92

Prediction model template 85

predictors, dynamic 300

production mode

batch performance reports 210

Production Models Aging Report 183

profile data 127

project folders

organization of 44

publishing champion model from 207

tasks 44

project input tables 27, 28

creating 31

project metadata 49

project output tables 27, 28

creating 32

project properties 49

project tables 28

performance tables 30

project input tables 28

project output tables 28

scoring task input tables 29

scoring task output tables 29

test tables 30

train tables 30

Project Tree 11

associating documents with folders 41

creating organizational folders 40

deleting objects in 42

organizing 39

project variables

mapping to model variables 95

- projects 43
 - controlling access to versions 152
 - creating 47
 - default version for 154
 - exporting champion models 163
 - model function types 46
 - planning 45
 - prerequisites for creating 46
 - properties 46
 - publishing Teradata scoring functions 164
 - retiring 178
 - setting champion model status 48
 - setting up 20
 - specific properties for 285
- Projects perspective 10, 11
- properties
 - Dynamic Lift reports 131
 - general 283
 - life cycle properties 73
 - life cycle templates 64
 - model function types 51, 287
 - model template properties 101
 - organization-specific user-defined 287
 - project properties 49
 - result set properties 122, 294
 - scoring task properties 121, 293
 - setting for imported models 94
 - specific for a model 103, 291
 - specific for a project 285
 - specific for a version 289
 - specific for milestones and tasks 290
 - system properties 284
 - user-defined 287
 - version properties 68
- prototype tables 27
 - project input tables 28
 - project output tables 28
 - scoring task output tables 29, 32
- Publish Teradata Scoring Function 164
- Publish Teradata Scoring Function window 168
- publishing
 - champion model, for batch performance reports 207
 - extracting published models 161
 - models 158
 - models to a channel 159
 - Teradata scoring functions 164

Q

- Query utility 235
 - searching for models 236
 - searching life cycles for tasks assigned to users 239

- searching with UUID 238

R

- registering
 - data sources 28
 - models 257, 274
- relational databases 6
- report macros
 - accessing 223, 225
- report templates
 - for user-defined reports 141
- reports
 - See also performance reports*
 - ad hoc reports 135, 137
 - batch performance reports 207
 - Characteristic reports 182, 183, 184
 - creating with performance tasks 197
 - Data Composition reports 182, 183
 - Delta reports 129
 - Dynamic Lift reports 131
 - KS reports 187
 - Model Assessment reports 189
 - model comparison reports 125
 - Model Input Variable Report 183
 - Model Monitoring reports 182, 185
 - Model Profile reports 127
 - Model Target Variable Report 183
 - monitoring Lift reports 185
 - Monitoring reports 192
 - monitoring ROC & Gini reports 186
 - performance monitoring reports 181
 - Production Models Aging Report 183
 - Stability reports 182, 183, 184
 - Summary of Reports 183
 - Summary reports 182, 183
 - user reports 135
 - user-defined reports 135, 140
- repository
 - accessing files in 252
 - model repository objects 244
- result set properties 122, 294
- retiring
 - champion models 177
 - projects 178
- ROC Chart 193
- ROC plots 186
- roles 18
 - life cycle template participant roles 58

S

- SAS Analytics Platform 5
- SAS code models
 - importing 83, 91
- SAS Content Server 5

- SAS Enterprise Miner
 - importing package files from 81
 - SAS Foundation 5
 - SAS Management Console 5
 - SAS Metadata Server 6
 - SAS Model Manager
 - interface 9
 - managing analytical models 3
 - model management process 6
 - operational environment 4
 - services provided by 3
 - setting up 19
 - user groups 4
 - SAS Model Manager Client 5
 - SAS Model Manager Server 5
 - SAS Model Manager Template Editor
 - window 57
 - SAS Model Manager window 9
 - layout 9
 - perspectives 10, 11
 - toolbar and menus 10
 - SAS programs
 - creating performance monitoring reports 206
 - SAS user-defined properties 288
 - SAS Workspace Server 6
 - saving documents 41
 - scoring functions, Teradata 164
 - publishing 164
 - scoring output tables 111
 - creating 111
 - scoring requests
 - logging 299
 - submitting 299
 - scoring results
 - parsing 300
 - scoring task content files 121
 - scoring task input tables 28, 29, 32
 - creating 32
 - scoring task output tables 27, 29, 32
 - adding to SAS Model Manager 29
 - creating 32
 - scoring tasks 107
 - creating 113
 - creating scoring output tables 111
 - executing 115
 - generated content files 121
 - graphing results 118
 - mapping output variables 115
 - modifying 114
 - properties 121, 293
 - result set properties 122, 294
 - tabbed views 109
 - searches
 - for life cycle tasks assigned to users 239
 - for models 236
 - with UUID 238
 - Section 508 standards 9
 - Segmentation model template 85
 - setup
 - projects and versions 20
 - SAS Model Manager 19
 - SPK files
 - See [package files](#)
 - Stability reports 182, 183, 184
 - example 185
 - overview 183
 - performance index warnings and alerts 189
 - Summary of Reports 183
 - Summary reports 182, 183
 - system properties 284
- ## T
- tabbed views
 - scoring tasks 109
 - tasks
 - creating reports using performance tasks 197
 - general tasks 22
 - life cycle tasks 71
 - project folder tasks 44
 - scoring task properties 121, 293
 - scoring tasks 107
 - searching for life cycle tasks assigned to users 239
 - specific properties for 290
 - version folder tasks 55
 - templates
 - See also [life cycle templates](#)
 - See also [model templates](#)
 - Analytical model template 85
 - Classification model template 85
 - Prediction model template 85
 - report templates 141
 - Segmentation model template 85
 - user model templates 85
 - user-defined model templates 96
 - user-templates directory 56
 - Teradata In-Database Scoring 165, 297
 - Teradata scoring functions 164
 - log messages for publishing 174
 - metadata tables 175
 - prerequisites for publishing 168
 - process flow 166
 - publishing 164
 - steps for publishing 171
 - test mode
 - for batch performance reports 210
 - test tables 28, 30

- creating 33
- thresholds
 - warnings and alerts 188
- toolbar 10, 15
- train tables 28, 30

U

- unfreezing
 - models 152
 - versions 153
- URL addresses 256
- user groups 4, 18
- user ID
 - for batch performance reports 210
- user model templates 85
- user reports 135
 - ad hoc 135, 137
 - output created by 136
 - user-defined 135, 140
 - validating models with 135
- user-defined model templates 96
- user-defined properties 287
 - organization-specific 287
 - SAS 288
- user-defined reports 135, 140
 - compared with ad hoc reports 136
 - creating 140
 - defining macro variables for 143
 - example 143
 - report template for 141
 - running 143
- user-templates directory 56
- users
 - searching for life cycle tasks assigned to 239
- UUIDs
 - searching with 238
 - translating to URL address 256

V

- validating models
 - with model comparison reports 125
 - with user reports 135
- verifying model exports 164
- version folders
 - organization of 54
 - tasks 55
- versions 53
 - clearing default version 155
 - controlling access to project versions 152
 - creating 67
 - creating life cycle templates 56
 - creating new version of a life cycle template 63
 - default version for projects 154
 - freezing 153
 - properties 68
 - SAS Model Manager functionality for 54
 - selecting new default version 178
 - setting default version 154
 - setting up 20
 - showing document versions 41
 - specific properties for 289
 - unfreezing 153
- View menu 17
- viewing
 - documents 41
 - life cycle templates 72
 - model comparison reports 134
 - performance monitoring reports 195

W

- warning notifications 188

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **`yourturn@sas.com`**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **`suggest@sas.com`**.

SAS® Publishing Delivers!

Whether you are new to the work force or an experienced professional, you need to distinguish yourself in this rapidly changing and competitive job market. SAS® Publishing provides you with a wide range of resources to help you set yourself apart. Visit us online at support.sas.com/bookstore.

SAS® Press

Need to learn the basics? Struggling with a programming problem? You'll find the expert answers that you need in example-rich books from SAS Press. Written by experienced SAS professionals from around the world, SAS Press books deliver real-world insights on a broad range of topics for all skill levels.

support.sas.com/saspress

SAS® Documentation

To successfully implement applications using SAS software, companies in every industry and on every continent all turn to the one source for accurate, timely, and reliable information: SAS documentation. We currently produce the following types of reference documentation to improve your work experience:

- Online help that is built into the software.
- Tutorials that are integrated into the product.
- Reference documentation delivered in HTML and PDF – **free** on the Web.
- Hard-copy books.

support.sas.com/publishing

SAS® Publishing News

Subscribe to SAS Publishing News to receive up-to-date information about all new SAS titles, author podcasts, and new Web site features via e-mail. Complete instructions on how to subscribe, as well as access to past issues, are available at our Web site.

support.sas.com/spn



**THE
POWER
TO KNOW®**

