

SAS[®] Model Manager 3.1

User's Guide



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® Model Manager 3.1: User's Guide*. Cary, NC: SAS Institute Inc.

SAS® Model Manager 3.1: User's Guide

Copyright © 2011, SAS Institute Inc., Cary, NC, USA

For a hardcopy 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.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

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 53.127–19 Commercial Computer Software-Restricted Rights (June 1987).

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

1st electronic book, July 2011

2nd electronic book, April 2012

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 3.1</i>	<i>ix</i>
<i>Accessibility Features of SAS Model Manager</i>	<i>xiii</i>
<i>Recommended Reading</i>	<i>xix</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
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	29
Creating Project Input and Output Tables	32
Creating Scoring Task Input and Output Tables	34
Creating a Test Table	35
Creating a Performance Table	35
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	46
Prerequisites for Creating Projects	47
About Defining Project Input and Output Variables	48
Create a Project	48
Modify Project Definition	51
Lock or Unlock Project Metadata	53
Setting the Project Champion Model Status	53
Project Properties	54
Chapter 6 • Working with Versions	59
Overview of Versions	59

Creating Life Cycle Templates	62
Create a Version	76
Version Properties	77
Working with Life Cycles	79
Chapter 7 • Using Workflow Console	85
Overview of Workflow Console	86
User Interface Layout	86
Customizing Category Views	88
Setting Preferences	94
Working with Objects	96
Viewing Workflow Activities	99
Working with Workflow Activities	100
Editing Activity Properties	100
Working with Comments	101
Managing the Workflow Process	105
Viewing Workflow Process Definitions	106
Creating a New Workflow Instance	107
Viewing Workflow Instances	109
Editing a Workflow Instance	110
Editing Instance Properties	112
Working with Workflow Participants	112
Terminating a Workflow Instance	116
 PART 3 Importing, Scoring, and Validating Models	117
Chapter 8 • Importing Models	119
Overview of Importing Models	119
Import Models from the SAS Metadata Repository	120
Import Package Files from SAS Enterprise Miner	122
Import SAS Code Models and R Models Using Local Files	123
Import PMML Models	135
Import Partial Models	136
Set Model Properties	137
Map Model Variables to Project Variables	138
User-Defined Model Templates	139
Specific Properties for a Model	145
Chapter 9 • Scoring Models	149
Overview of Scoring Tasks	149
Scoring Task Tabbed Views	151
Create Scoring Output Tables	153
Create a Scoring Task	155
Modify a Scoring Task	156
Map Scoring Task Output Variables	157
Execute a Scoring Task	158
Graph Scoring Task Results	159
Generated Scoring Task Content Files	163
Scoring Task Properties	163
Result Set Properties	164
Chapter 10 • Validating Models Using Comparison Reports	167
Overview of Model Comparison Reports	167
Model Profile Reports	169

Creating Delta Reports	171
Creating Dynamic Lift Reports	173
View Reports	176
Chapter 11 • Validating Models Using User Reports	177
Overview of User Reports	177
Ad Hoc Reports	179
User-Defined Reports	182
 PART 4 Deploying and Delivering Champion Models	191
Chapter 12 • Deploying Models	193
Overview of Deploying Models	193
Champion Models	194
Freezing Models	196
The Default Version for a Project	198
Chapter 13 • Delivering Models	203
Overview of Model Delivery	203
Publishing Models	204
Exporting Models	208
Publish Scoring Functions	210
Chapter 14 • Replacing a Champion Model	223
Overview of Replacing a Champion Model	223
Select a New Default Version	224
Retire a Project	224
 PART 5 Performance Monitoring and Retraining Models	225
Chapter 15 • What Are Performance Monitoring Reports?	227
Overview of Performance Monitoring Reports	227
Types of Performance Monitoring Reports	228
Performance Index Warnings and Alerts	234
The Process of Monitoring Champion Models	236
Formatting Performance Monitoring Reports	237
View Reports	240
Chapter 16 • Create Reports by Defining a Performance Task	241
Overview of Creating Reports Using a Performance Task	241
Prerequisites for Running the Define Performance Task Wizard	243
Run the Define Performance Task Wizard	245
Chapter 17 • Create Reports Using Batch Programs	251
Overview of SAS Programs to Monitor Model Performance	252
Prerequisites for Running Batch Performance Reports	253
Report Output in Test and Production Modes	256
Define the Report Specifications	257
Extracting the Champion Model from a Channel	268
SAS Code to Run Performance Reports	271

Chapter 18 • Using Dashboard Reports	279
Overview of Project Dashboard Reports	279
Define Dashboard Report Indicators	279
Generate Dashboard Reports	283
View Dashboard Reports	285
Chapter 19 • Retraining Models	289
Overview of Retraining Models	289
Prerequisites for Retraining a Model	290
Define a Model Retrain Task	290
Execute a Model Retrain Task	296
Viewing Retrained Models and Model Comparison Reports	297
 PART 6 Appendixes	 301
Appendix 1 • Query Utility	303
Overview of the Query Utility	303
Search for Models	304
Search by Using a UUID	306
Search Life Cycles for Tasks Assigned to Users	307
Appendix 2 • SAS Model Manager Access Macros	311
Overview of Access Macros	311
Using the SAS Model Manager Access Macros	312
Dictionary of Access Macros	317
Appendix 3 • SAS Model Manager Macro Variables	351
Appendix 4 • Properties	357
General Properties	357
System Properties	358
Specific Properties for a Project	359
User-Defined Properties	362
Specific Properties for a Version	363
Specific Properties for Milestones and Tasks	365
Specific Properties for a Model	366
Scoring Task Properties	368
Result Set Properties	369
Appendix 5 • SAS Model Manager R Model Support	371
Overview of Using R Models with SAS Model Manager	371
Preparing R Model Files to Use with SAS/IML	372
Appendix 6 • Report and Performance Monitoring Examples	379
Dashboard Report Examples	379
Model Retrain Comparison Report Example	387
Monitoring Performance of a Model without Score Code	397
 Glossary	 399
Index	407

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

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 3.1

Overview

SAS Model Manager 3.1 has the following new features and enhancements:

- ability to retrain models
- ability to customize dashboard reports
- enhanced support for modifying project definitions
- ability to import R models
- ability to modify and upload templates and SAS files
- ability to manage the progress of a project or version using SAS Workflow
- enhanced support for PMML models

The first maintenance release of SAS Model Manager 3.1 adds the ability to publish scoring functions to a Greenplum database.

Ability to Retrain Models

SAS Model Manager now supports the retraining of models. Multiple attributes can be specified when you define a model retrain task. You can also register the models and create model comparison reports when executing a model retain task for a project. For more information, see [“Overview of Retraining Models” on page 289](#).

Ability to Customize Dashboard Reports

The SAS Model Manager Dashboard reports are produced from the same data sets that are used to create the performance monitoring reports. The data sets are created by running performance tasks. For each project a user can define dashboard report indicators that are used to create the dashboard reports. Users can now specify additional report options and exclude project types when generating the dashboard reports. The dashboard reports are not displayed through the SAS Model Manager user interface, but instead are viewed in an HTML browser. For more information, see [“Overview of Project Dashboard Reports” on page 279](#).

Enhanced Support for Modifying Project Definitions

SAS Model Manager administrators can now modify the project definition that contains project properties, project input variables, and project output variables, after a project has been created. This change enables users to modify the project properties, select new project input and output variables, or modify existing project input and output variables. A SAS Model Administrator can also lock the project metadata so that it cannot be modified. For more information, see [“Lock or Unlock Project Metadata” on page 53](#) and [“Modify Project Definition” on page 51](#).

Ability to Import R Models

SAS Model Manager now supports importing R models. R is a freely available language and environment for statistical computing and graphics. Using the open architecture of SAS Model Manager, you can register and import R models. For more information, see [“Import SAS Code Models and R Models Using Local Files” on page 123](#).

Ability to Manage Templates and SAS Code Files

Using the SAS Model Manager Template Editor, you can create, edit, or delete life cycle templates, model templates, and user report templates or user report SAS code files. You can then upload new or modified templates, or SAS code files to the SAS Content Server. For more information, see [“Creating Life Cycle Templates” on page 62](#), [“User-Defined Model Templates” on page 139](#), and [“User-Defined Reports” on page 182](#).

Ability to Manage the Progress of a Project or Version Using SAS Workflow

The Workflow Console for SAS Model Manager can be used to track the progress of a modeling project or version. Process definitions are created using SAS Workflow Studio and are activated with the SAS Workflow Engine. A SAS Model Manager administrator can then use the Workflow Console to create instances of the process definitions to be used with SAS Model Manager. For more information, see [“Overview of Workflow Console” on page 86](#).

Enhanced Support for PMML Models

SAS Model Manager now supports setting a PMML model as the champion model, publishing a PMML model to a SAS channel, and exporting a PMML model to the SAS Metadata Repository.

Ability to Publish Scoring Functions to a Greenplum Database

In the first maintenance release of SAS Model Manager 3.1, you can publish scoring functions to a Greenplum database. SAS Model Manager uses the scoring publishing macros that are included in the SAS/ACCESS Interface to Greenplum to publish the scoring information for a model to the database. For more information, see [“Publish Scoring Functions” on page 210](#).

Accessibility Features of SAS Model Manager

Overview

SAS Model Manager 3.1 has been tested with assistive technology tools. It includes accessibility and compatibility features that improve the usability of the product for users with disabilities, with exceptions noted below. These features are related to accessibility standards for electronic information technology that were adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973 (2008 draft proposal initiative update). Applications are also tested against Web Content Accessibility Guidelines (WCAG) 2.0, part of the Web Accessibility Initiative (WAI) of the Worldwide Web Consortium (W3C). For detailed information about the accessibility of this product, send e-mail to accessibility@sas.com or call SAS Technical Support.

User Interface Layout

SAS Model Manager Client

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 based on your perspective and the component that is selected. Inactive options are dimmed.

For more information about the layout and features of the application window, see [“Layout of the SAS Model Manager Window” on page 9](#).

SAS Model Manager Workflow Console

The SAS Model Manager Workflow Console provides a framework for working with SAS workflow. The application window contains three main sections:

- The left side of the window contains a collapsible navigation pane. This pane can contain one or more views (depending on your permissions and roles) that you select using buttons at the bottom of the pane. The views contain lists of objects. You can open these objects (one at a time) into tabs in the work area next to the navigation pane.

- The work area in the middle of the window contains tabs that display open objects, workflow process definitions, instances of process definitions and activities.
- An optional docking pane is available on the right for docking certain pop-up windows such as the object previewer, the permissions inspector, and the list of how-to topics. This pane is not displayed until a window is docked.

To customize the application window and its features, select **File** ⇒ **Preferences**. For more information about the layout and features of the application window, see [Chapter 7, “Using Workflow Console,”](#) on page 85.

Themes

An application’s theme is the collection of colors, graphics, fonts, and effects that appear in the application. The following themes are provided with Workflow Console: SAS Corporate, SAS Light, and SAS Dark. To change the theme for the application, select **File** ⇒ **Preferences** and go to the **Global Preferences** page.

Keyboard Shortcuts

SAS Model Manager Client

The following table contains standard keyboard shortcuts for the application. Keyboard shortcuts are also documented in tooltips and menu labels.

Task	Keyboard Shortcut
Display a Help pop-up window	F1
Display the contents of a menu	ALT + first letter of the menu name. For example, press ALT+F to access the File menu.
Cancel an action in a pop-up window or wizard	ALT+C
Acknowledge a message or accept an action in a pop-up window.	ALT+O
View the next page of a wizard	ALT+N
Go back to the previous page of a wizard	ALT+B
Finish entering information in a wizard.	ALT+F

SAS Model Manager Workflow Console

The following table contains standard keyboard shortcuts for the Workflow Console application. Keyboard shortcuts are also documented in tooltips and menu labels.

Task	Keyboard Shortcut
Display a Help pop-up window	Ctrl+?
	<i>Note:</i> This shortcut does not work on some keyboards (for example, the Italian keyboard).
Increase font size	Ctrl++
Decrease font size	Ctrl+-

Exceptions to Accessibility Standards

SAS Model Manager Client

All known exceptions to accessibility standards are documented in the following table.

Accessibility Issue	Workaround
You cannot use the keyboard to open context menus. SHIFT + F10 does not open the context menus when focus is on the relevant object in the Project Tree.	<p>Use MouseKeys to open the context menus. MouseKeys lets you control the mouse pointer by using the numeric keypad on your keyboard. In Windows XP, press the Left ALT, Left SHIFT, and NUM LOCK keys, to turn on MouseKeys. You can also use the Control Panel. Select Accessibility Options ⇒ Mouse tab and then select Use MouseKeys.</p> <p>For Windows 7, from the Control Panel select Ease of Access Center ⇒ Make the mouse easier to use, and then select Turn on Mouse Keys.</p>
You cannot use the keyboard to expand or collapse the objects in the Project Tree.	
You cannot use the keyboard shortcut ALT + F10 to open drop-down menus.	

Accessibility Issue	Workaround
The JAWS screen reading software utility cannot read most of the text in the SAS Model Manager main window and pop-up windows. For example, not all of the text in the Project Tree, icon toolbar, Properties pane, Resources pane, Annotations pane, and New Report Wizard is readable by the screen reader.	
You can use the keyboard shortcut CTRL + Tab to move focus to another component in the main window. However, the toolbar icons and left navigation icons are not active, even though they might look active.	
SAS Model Manager does not properly inherit the Windows high contrast and large text settings.	
The ENTER key does not activate a default button, such as OK , Next , or Finish in a window after the required fields have been completed.	
When JAWS is in use, some parts of SAS Model Manager have performance issues. For example, when you are navigating the Project Tree, JAWS pauses before reading the contents of the component that is currently in focus. If you move quickly through the list before JAWS reads each entry, JAWS reads only some but not all of the selections before reading the current selection. The performance issues seem to worsen with the number of levels that are expanded in the Project Tree, and there is a delay when expanding and collapsing objects when navigating.	

SAS Model Manager Workflow Console

All known exceptions to accessibility standards for Workflow Console are documented in the following table.

Accessibility Issue	Workaround
When you are in a table cell, if you press Home, End, Page Up, or Page Down, the selected cell will change to be one in the first column of the currently displayed columns for the table.	Use the arrow keys to navigate through the cells of the table.
You cannot use the keyboard to navigate to the column headings in a table. JAWS also cannot read the column headings.	

Accessibility Issue	Workaround
<p>You cannot use the keyboard to access the column-sorting feature that is available from the first cell of each column in a table.</p>	
<p>When a table is in a secondary window, you cannot use the Enter or Esc keys to exit Edit mode. Pressing Enter performs the default action on the window (for example, it might click the OK button). Pressing Esc will close the window.</p>	
<p>If you are tabbing through a table that contains both editable and non-editable cells, when you get to the first non-editable cell, focus moves to outside of the table instead of to the next editable cell. Also, when you are in an editable cell and you press Enter the focus does not move to the next editable cell.</p>	<p>Use arrow keys to navigate to a cell. If the cell is editable, then press Enter to enter Edit mode. Edit the cell. Press Esc to exit Edit mode. Then, use arrow keys to navigate to the next cell.</p> <p><i>Note:</i> You cannot use this workaround if the table is in a secondary window.</p>
<p>You cannot use the keyboard to close the overflow menu of a toolbar after you tab into the menu. (The overflow menu is displayed for a toolbar when the width of the toolbar is reduced to a point where all of the tools in the toolbar cannot be displayed in a single row.)</p>	
<p>JAWS refers to table controls as list boxes.</p>	<p>When JAWS reports that a control is a list box, keep in mind that it might actually be a table.</p>
<p>The keyboard shortcuts that are used to interact with editable tables can conflict with keyboard shortcuts for JAWS' forms mode.</p>	<p>As a best practice, disable the JAWS virtual PC cursor when you work with tables. Tab to the table and press Insert+Z to disable the JAWS virtual PC cursor. When you finish interacting with the table, press Insert+Z to re-enable the JAWS virtual PC cursor.</p>
<p>Some controls in the application, such as images, icons, and buttons, are unlabeled or cannot be read by JAWS.</p>	
<p>When a modal secondary window is open in the application, JAWS can still access the controls (which should be blocked) in the background of the application.</p>	
<p>When JAWS reads the information in a table, it reads an entire row at a time instead of cell by cell.</p>	
<p>The Windows high-contrast and large-text settings are not properly inherited.</p>	

If you have questions or concerns about the accessibility of SAS products, send e-mail to accessibility@sas.com.

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Model Manager: Administrator's Guide*
- *Getting Started with SAS Enterprise Miner*
- *SAS Data Set Options: Reference*
- *SAS Formats and Informats: Reference*
- *SAS Functions and CALL Routines: Reference*
- *SAS In-Database Products: User's Guide*
- *SAS Macro Language: Reference*
- *SAS Statements: Reference*
- *SAS System Options: Reference*

For a complete list of SAS publications, go to support.sas.com/bookstore. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-3228
Fax: 1-919-677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/bookstore

xx *Recommended Reading*

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, publish and monitor champion models in a production environment, and retrain models. 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 a specific database 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 by using either 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.
- You can retrain models to respond to data or market changes.

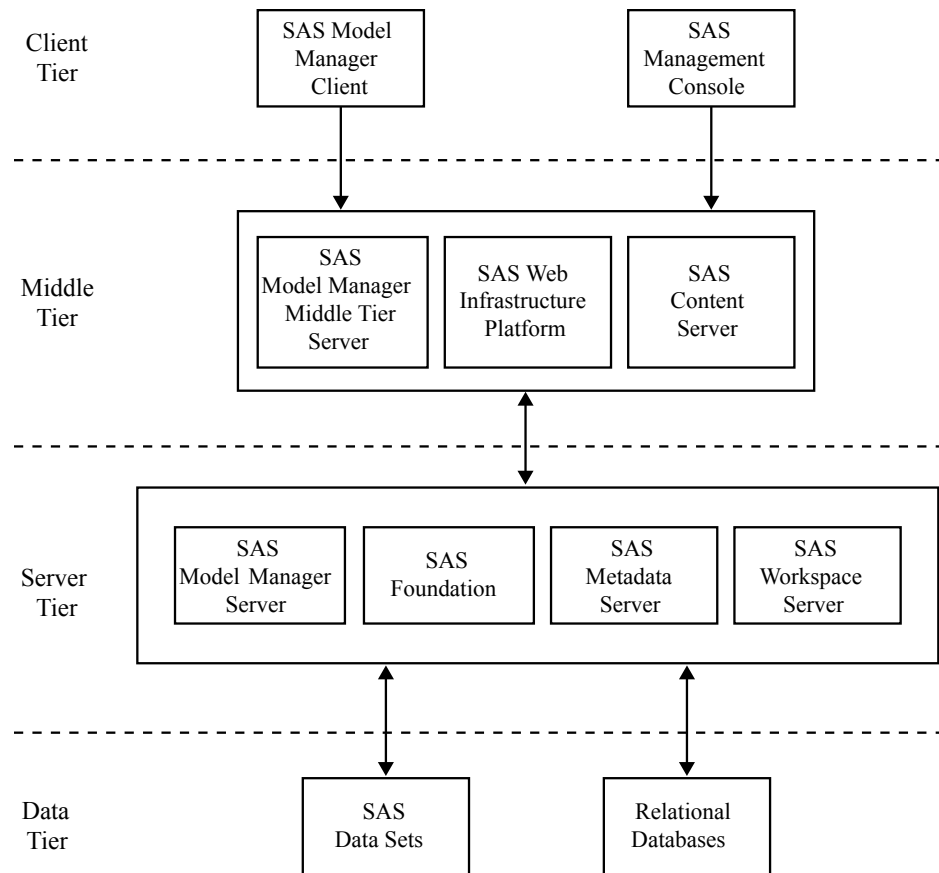
Any user who is registered in SAS Management Console can be assigned to a SAS Model Manager group, and can then work in SAS Model Manager. SAS Model Manager has three groups.

- Users in the Model Manager Administrator Users 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 Users group can perform some of the tasks that the Model Manager Administrator Users group can perform as well as all tasks that users in the Model Manager User group can perform.
- Users in the Model Manager Users 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. You can use 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. 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 uses relational 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 Middle Tier Server

The SAS Model Manager Middle Tier Server is a collection of services and applications that orchestrates the communication and movement of data between all servers and components in the SAS Model Manager operational environment.

SAS Web Infrastructure Platform

The SAS Web Infrastructure Platform (or WIP) is a collection of middle tier services and applications that provides basic integration services. It is delivered as part of the Integration Technologies package. As such, all Business Intelligence applications, Data Integration applications, and SAS Solutions have access to the Web Infrastructure Platform as part of their standard product bundling.

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 Model Manager Server

The SAS Model Manager Server is a collection of macros that run under the SAS Workspace Server or a SAS session in the SAS Model Manager operational environment.

SAS Foundation

SAS Foundation includes Base SAS and a superset of SAS software that 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 stores and retrieves 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.

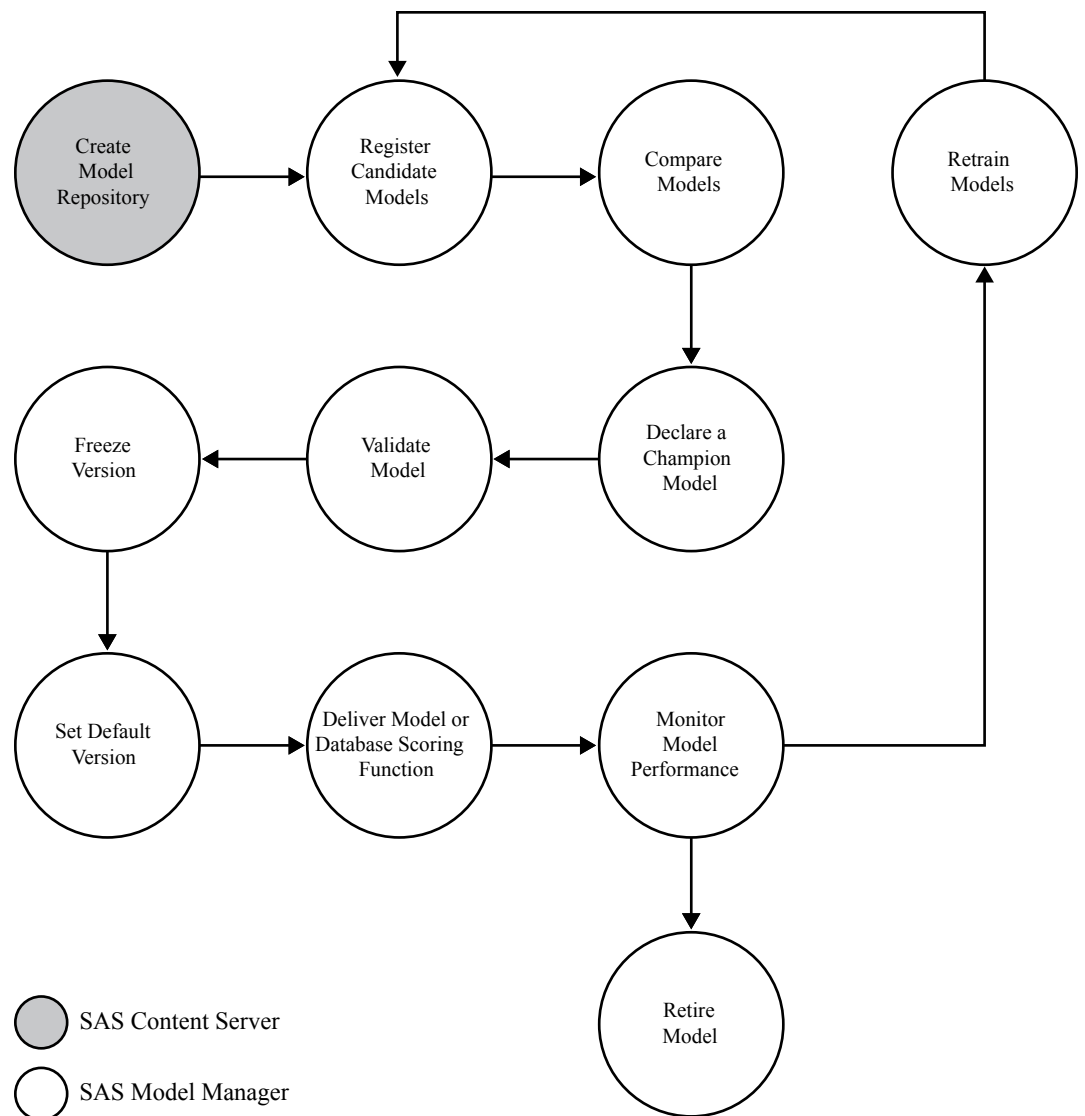
See Also

[“Publish Scoring Functions” on page 210](#)

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



Here is a 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 a model to a SAS publish channel or publish a champion model for a project as a scoring function to a database.
- **Monitor Model Performance:** provide comparative model performance benchmarking.
- **Retrain Models:** select models to retrain in response to data or market changes.
- **Retire Model:** retire a model from production.

Chapter 2

Introduction to SAS Model Manager

Layout of the SAS Model Manager Window	9
Overview of the SAS Model Manager Window	9
SAS Model Manager Perspectives	10
SAS Model Manager Toolbar and Menus	14
SAS Model Manager User Groups, Roles, and Tasks	18
SAS Model Manager Groups	18
SAS Model Manager Roles	19
Setting Up SAS Model Manager	20
Setting Up Projects and Versions	20
Importing and Assessing Models	21
Deploying and Delivering Models	22
Monitor Champion Model Performance and Retrain Models	22
General Tasks	23

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 based on 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 14](#).

The perspective button bar enables you to select a perspective to display. 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 Cycles 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 major 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 Cycles 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 14](#).

The SAS Model Manager menus enable you to perform general tasks such as renaming an object or accessing Help. The menus 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 Projects perspective  button to display the Projects perspective.

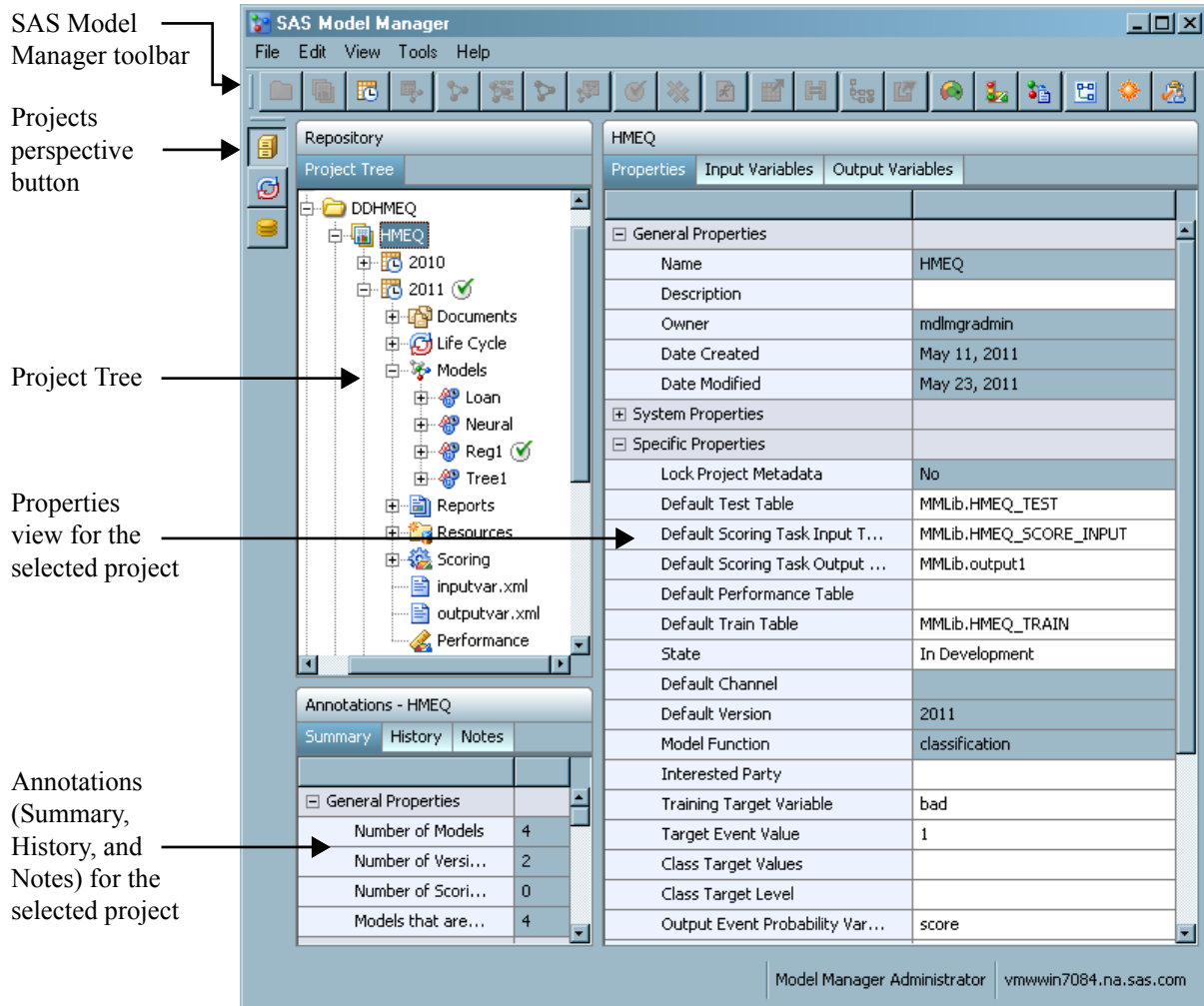
Most of your work in SAS Model Manager is performed in the Projects perspective, where you manage model projects and their components. The Projects perspective contains three major views: the Repository view, the Properties view, and the Annotations view.

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 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 in 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 **2011**. The **2011** version folder contains a **Models** folder that contains three models.

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, when 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 a component 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 229](#). The **History** tab displays a time-stamped log that documents the following transactions by user ID for the selected repository component:

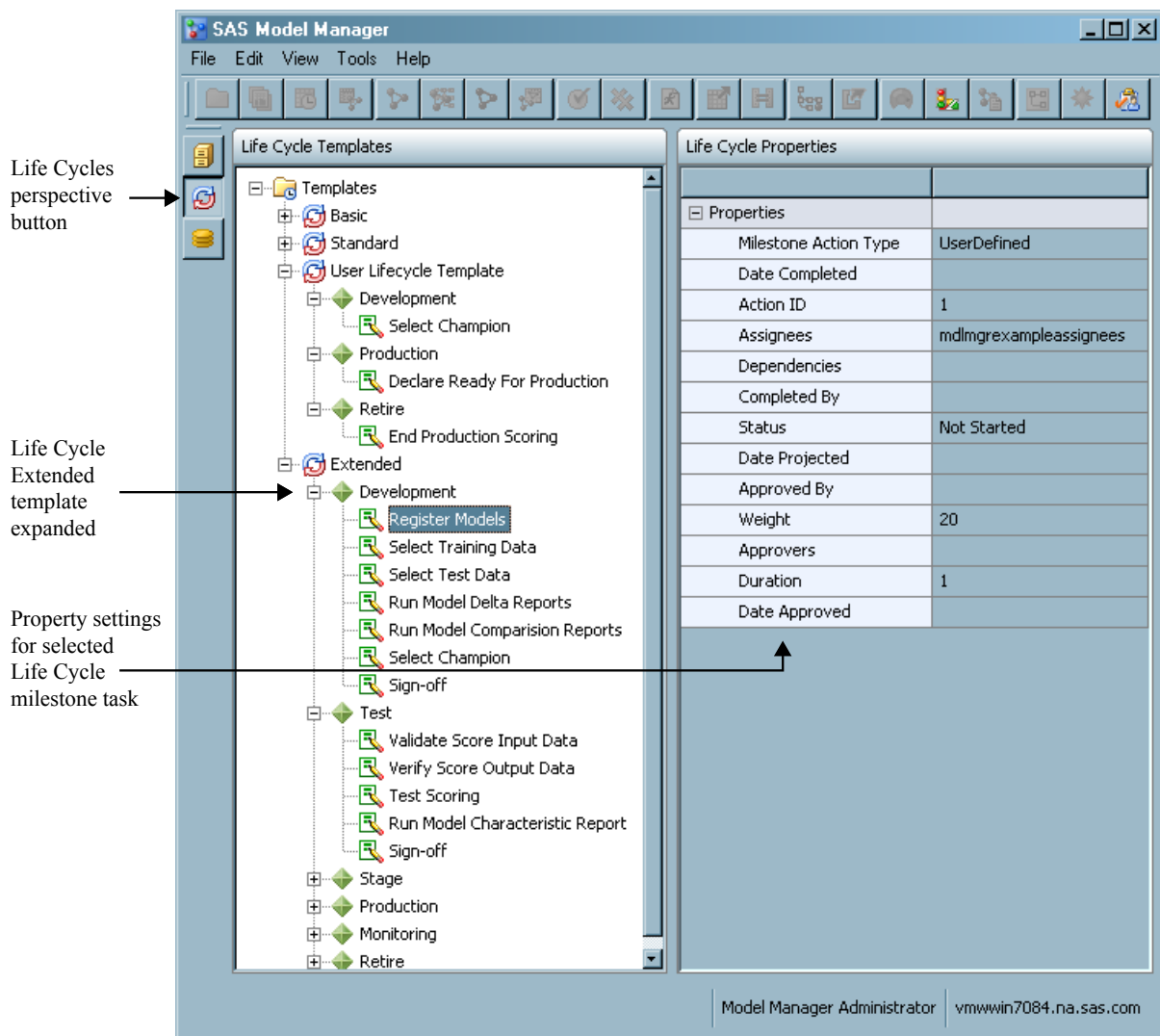
- Create
- Modify
- Import
- Publish
- Export
- Delete

The **Notes** tab enables you to record information about the selected component that can be useful for later reference. For more information about SAS Model Manager projects, see [Chapter 5, “Working with Projects,”](#) on page 43.

Life Cycles Perspective

Click the Life Cycle perspective  button to view the model life cycle templates. Each life cycle template contains milestones that correspond to key events in the life span of a modeling project 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 Cycles perspective displays three example life cycle templates: Basic, Standard, and Extended.

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

Figure 2.2 The Life Cycles Perspective

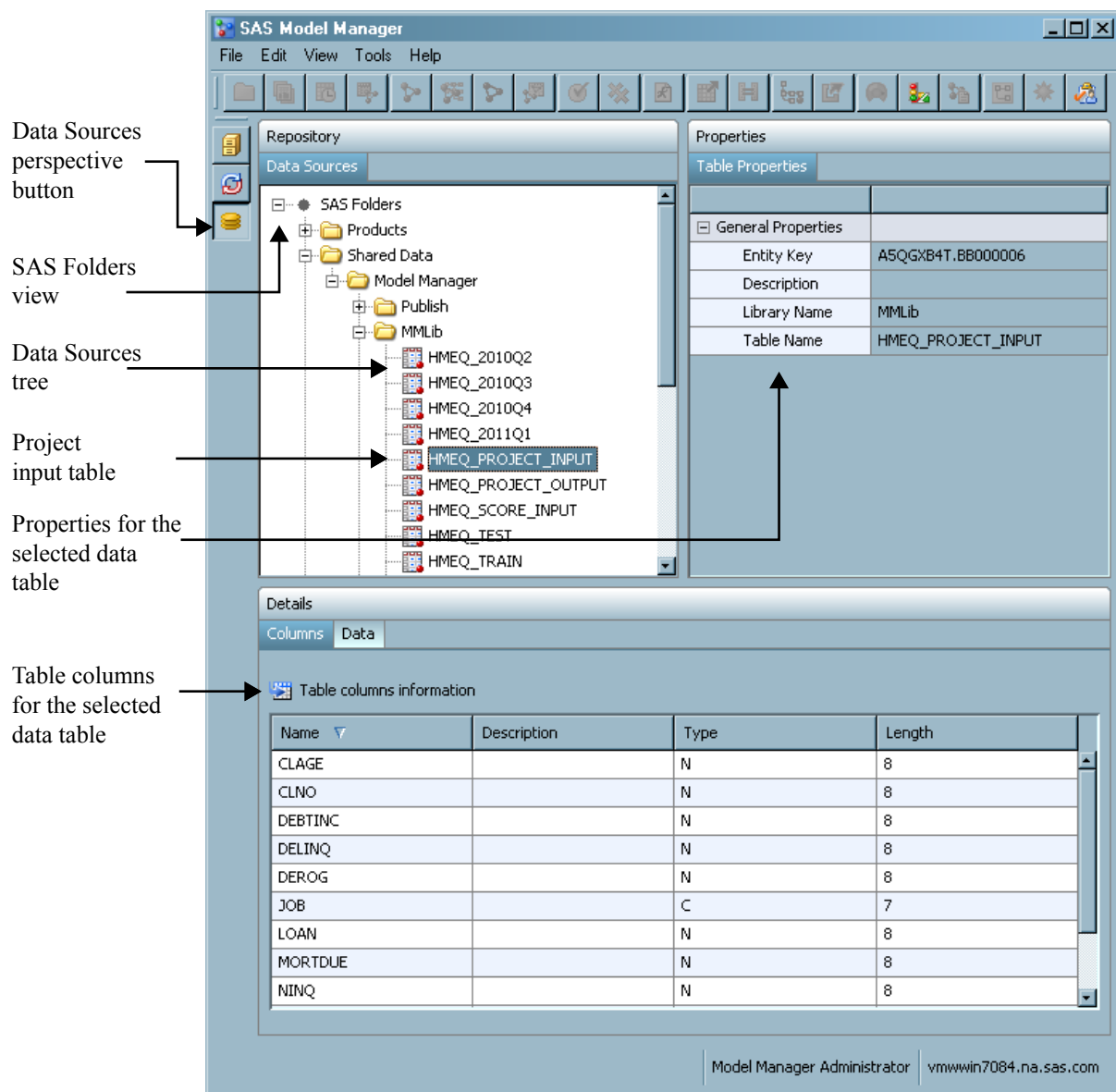
For more information about life cycles, see [“Working with Life Cycles”](#) on page 79.

Data Sources Perspective



Click the Data Sources perspective button to view data sources to SAS Model Manager. SAS Model Manager data sources are populated from libraries that are defined through SAS Management Console.

In the following examples, the Repository view displays the SAS Folders. This component lists the folders that are available in the SAS Metadata Repository. The folders contain 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

For more information about data sources, see [Chapter 3, “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 a pop-up menu. The list of active tasks varies based on your perspective and 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 for that usage context 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 modeling project folder under an organizational folder. Project folders contain one or more version folders. For more information, see [“Create a Project” on page 48](#).
- 3 **New Version** creates a version folder. A version folder contains the models and their related files. Related files are typically associated with a chronological period, such as a fiscal year or a quarter. You can create version folders only under a project folder. For more information, see [“Create a Version” on page 76](#).
- 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. For more information, see [“Create a Scoring Task” on page 155](#).
- 5 **SAS 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 the SAS Metadata Repository” on page 120](#).
- 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 122](#).
- 7 **Local Files** imports SAS code models that were not developed in SAS Enterprise Miner (such as PROC LOGISTIC models) into SAS Model Manager. For more information, see [“Import SAS Code Models and R Models Using Local Files” on page 123](#).
- 8 **PMML Model File** imports a PMML model. For more information, see [“Import PMML Models” on page 135](#).
- 9 **Set Champion Model or Default Version** selects the champion model or the default version. When you select a model in the Project Tree, click the **Set Champion Model or Default Version** button to promote the model to champion status in the version folder. When you select a version folder in the Project Tree, click the **Set Champion Model or Default Version** button to promote the version folder to the default version within the project for model scoring and life cycle management. For more information, see [Chapter 12, “Deploying Models,” on page 193](#).
- 10 **Clear Champion Model or Default Version** deselects the champion model, the default version, or both. When you select a default version in the Project Tree, click the **Clear Champion Model or Default Version** button to clear that version's status as default. When you select a champion model in the Project Tree, click the **Clear Champion Model or Default Version** button to clear that model's status as the champion model. For information, see [Chapter 12, “Deploying Models,” on page 193](#).
- 11 **Execute** performs the scoring task that is selected 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 158](#).

- 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 153](#).
- 13 **Quick Mapping Check** enables you to compare the input data source variable names that were submitted to a scoring task with the required input variable names in the model. If the variables in the scoring input table are an incomplete subset of the model's required input variables, then the scoring results might be statistically invalid. The variable data type is not validated. For more information, see [“Overview of Scoring Tasks” on page 149](#).
- 14 **Advanced View** displays a SAS Enterprise Miner Package Viewer window that displays the contents of the SAS Enterprise Miner SPK file if it was created with the model that is selected in the Project Tree.

When you register a model in the SAS Metadata Repository for SAS Enterprise Miner, a SAS Enterprise Miner package file is registered if a Web folder to store the file was defined using SAS Management Console. For more information, see model deployment in the Help for SAS Enterprise Miner 7.2.
- 15 **Export Model into SAS Metadata Repository** exports the model that is selected in the Project Tree to the SAS Metadata Repository. For more information, see [“Exporting Models” on page 208](#).
- 16 **Define Dashboard Report Indicators** defines the conditions for the performance monitoring data that you want to see in a dashboard reports. For more information, see [“Define Dashboard Report Indicators” on page 279](#).
- 17 **Generate Dashboard Reports** displays a window that you can use to select style and report options. Dashboard reports are then created for projects that have performance monitoring data and have dashboard report indicators that have been defined. For more information, see [“Generate Dashboard Reports” on page 283](#).
- 18 **Define Model Retrain Task** starts a wizard that retrains one or more models. For more information, see [Chapter 19, “Retraining Models,” on page 289](#).
- 19 **View Workflow** enables the user to view the workflow instances that are associated with the selected project or version. For more information, see [“Viewing Workflow Instances” on page 109](#).
- 20 **New Workflow Instance** creates a new workflow instance of a process definition and associates it with the selected project or version. For more information, see [“Creating a New Workflow Instance” on page 107](#).
- 21 **My Workflow Inbox** opens the SAS Model Manager Workflow Console to view the workflow activities that have been assigned to the user as a potential owner, actual owner, or business administrator. For more information, see [“Viewing Workflow Activities” on page 99](#).

SAS Model Manager Menus

The SAS Model Manager main menu varies in content based on whether you select the Projects perspective, the Life Cycles perspective, or the Data Sources perspective. If a menu item is not applicable in the selected perspective, then it is dimmed and is not available. Here is a list of all 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 48](#).
- **New Version** creates a new version folder within a project folder. For more information, see [“Create a Version” on page 76](#).
- **New Scoring Task** creates a new scoring task within a **Scoring** folder. For more information, see [“Create a Scoring Task” on page 155](#).
- **Import From** specifies the method that you want to use to import models into a version's **Models** folder. For more information, see [Chapter 8, “Importing Models,” on page 119](#).
 - **SAS Metadata Repository** displays a SAS Folders 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 the SAS Metadata Repository” on page 120](#).
 - **SAS Enterprise Miner Package File** displays a local file browser window that you can use to import a model from a SAS Enterprise Miner package (SPK) file. For more information, see [“Import Package Files from SAS Enterprise Miner” on page 122](#).
 - **Local Files** displays a local file browser window that you can use to import SAS code models that were not developed in SAS Enterprise Miner (such as PROC LOGISTIC models) into SAS Model Manager. For more information, see [“Import SAS Code Models and R Models Using Local Files” on page 123](#).
 - **PMML Model File** displays a local file browser window that you can use to import PMML models. For more information, see [“Import PMML Models” on page 135](#).
- **Exit** ends the SAS Model Manager session, and then closes the window.

Edit

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

View

- **Projects** displays the Projects perspective. For more information, see [“Projects Perspective” on page 10](#).
- **Life Cycles** displays the Life Cycles perspective. For more information, see [“Life Cycles Perspective” on page 12](#).
- **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 14](#).

Tools

- **Manage Templates** displays the SAS Model Manager Template Editor, which enables you to create, edit, and upload to the SAS Content Server life cycle templates and model templates. For more information, see [“Creating Life Cycle Templates” on page 62](#) and [“User-Defined Model Templates” on page 139](#).
- **Generate Dashboard Report** displays a window that you can use to select the style and report options. Dashboard reports are then created for projects that have

performance monitoring data and have dashboard report indicators that have been defined. For more information, see [“Generate Dashboard Reports” on page 283](#).

- **Manage Workflow** opens the SAS Model Manager Workflow Console, which can be used to manage instances of workflow process definitions and workflow activities. For more information, see [“Managing the Workflow Process” on page 105](#).
- **My Workflow Inbox** opens the SAS Model Manager Workflow Console to view the workflow activities that have been assigned to the user as a potential owner, actual owner, or business administrator. For more information, see [“Viewing Workflow Activities” on page 99](#).

Help

- **Help** displays the table of contents of the SAS Model Manager Help.
- **About SAS Model Manager** 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 Users and Model Manager Users groups are more restrictive. See the tables in the subsequent sections for a list of SAS Model Manager tasks and the groups whose users can perform the task.

A SAS Model Manager administrator can create custom groups for your organization as well as 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 correspond 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
Comment Administrator	A user who can manage comments in the SAS Model Manager Workflow Console. This role is assigned to the group Model Manager Administrators.	:CAdmin
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

Role	Description	Abbreviation
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>
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	Chapter 3, “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>
Configure the SAS Content Server for SAS Model Manager	MM Admin	See <i>SAS Intelligence Platform: Web Application Administration Guide</i>
Create workflow process definitions	MM Admin	See <i>SAS Model Manager: Administrator's 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
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 48
Create versions	MM Adv User, MM Admin	“Create a Version” on page 76
Delete a node in the Project Tree	MM Adv User, MM Admin	“Deleting an Object in the Project Tree” on page 42
Create and upload life cycle templates	MM User, MM Adv User, MM Admin	“Creating Life Cycle Templates” on page 62
Create a workflow instance	MM Admin	“Creating a New Workflow Instance” (p. 107)
View workflow instances associated with a project or version	MM Admin	“Viewing Workflow Instances” (p. 109)

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 139
Import models	MM Adv User, MM Admin	Chapter 8, “Importing Models,” on page 119
Configure model properties	MM Adv User, MM Admin	“Set Model Properties” on page 137
Map model variables to project variables	MM Adv User, MM Admin	“Map Model Variables to Project Variables” on page 138
Run model comparison reports	MM Adv User, MM Admin	Chapter 10, “Validating Models Using Comparison Reports,” on page 167
Create user reports	MM Adv User, MM Admin	Chapter 11, “Validating Models Using User Reports,” on page 177

Task	Group	Topic
Create scoring task output tables	MM Adv User, MM Admin	“Create Scoring Output Tables” on page 153
Create and run scoring tasks	MM Adv User, MM Admin	Chapter 9, “Scoring Models,” on page 149

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 194
Validate the champion model by running a scoring task using test data and reviewing the scoring task output	MM Adv User, MM Admin	Chapter 10, “Validating Models Using Comparison Reports,” on page 167 Chapter 11, “Validating Models Using User Reports,” on page 177
Set the default version	MM Adv User, MM Admin	“The Default Version for a Project” on page 198
Freeze or unfreeze versions	MM Admin	“Freezing Models” on page 196
Publish a project, version, or model	MM Adv User, MM Admin	“Publishing Models” on page 204
Extract a model	any user who has the appropriate access rights to the SAS Metadata Repository	“Extract a Published Model” on page 207
Export a model	MM Adv User, MM Admin	“Exporting Models” on page 208
Publish a scoring function	MM Adv User, MM Admin	“Publish Scoring Functions” on page 210

Monitor Champion Model Performance and Retrain Models

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

Task	Group	Topic
Set project properties	MM Adv User, MM Admin	“Project Properties” on page 54
Run the Define Performance Task wizard	MM Adv User, MM Admin	Chapter 16, “Create Reports by Defining a Performance Task,” on page 241
Execute the performance monitoring SAS programs from the PerformanceMonitor node.	MM Adv User, MM Admin	“Run the Define Performance Task Wizard” on page 245
Run performance monitoring batch jobs	in Test mode: MM User, MM Adv User, MM Admin in Production mode: MM Adv, MM Admin	Chapter 17, “Create Reports Using Batch Programs,” on page 251
View monitoring reports and charts	MM User, MM Adv User, MM Admin	“View Reports” on page 240
Define dashboard report indicators	MM Adv User, MM Admin	“Define Dashboard Report Indicators” (p. 279)
Generate dashboard reports	MM Adv User, MM Admin	“Generate Dashboard Reports” (p. 283)
View dashboard reports	MM User, MM Adv User, MM Admin	“View Dashboard Reports” (p. 285)
Define a model retrain task	MM Adv User, MM Admin	“Define a Model Retrain Task” (p. 290)
Execute a model retrain task	MM Adv User, MM Admin	“Execute a Model Retrain Task” (p. 296)
View retrained models and the associated model comparison reports	MM User, MM Adv User, MM Admin	“Viewing Retrained Models and Model Comparison Reports” (p. 297)

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 81
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 81
Use the Query utility	MM User, MM Adv User, MM Admin	Appendix 1, “Query Utility,” on page 303
Set the status of a project champion model	MM Adv User, MM Admin	“Setting the Project Champion Model Status” on page 53
Replacing a champion model	MM Adv User, MM Admin	“Overview of Replacing a Champion Model” on page 223
View workflow process activities	<p>MM User, MM Adv User, MM Admin</p> <p>A user must be the actual owner of an activity or assigned the workflow participant role of potential owner or business administrator to view activities in their workflow inbox.</p>	“Viewing Workflow Activities” (p. 99)
Work with workflow process activities	<p>MM User, MM Adv User, MM Admin</p> <p>A user who is a workflow participant can claim, release, and complete activities.</p>	“Working with Workflow Activities” (p. 100)

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	59
<i>Chapter 7</i>	
Using Workflow Console	85


Chapter 3

Working with Data Sources

Overview of Data Sources	27
Project Tables	29
Project Input Tables	29
Project Output Tables	30
Scoring Task Input Tables	30
Scoring Task Output Tables	30
Train Tables	31
Test Tables	31
Performance Tables	32
Creating Project Input and Output Tables	32
Create a Project Input Table	32
Create a Project Output Table	33
Creating Scoring Task Input and Output Tables	34
About Scoring Task Input and Output Tables	34
Create a Scoring Task Input Table	34
Create a Scoring Task Output Table	35
Creating a Test Table	35
Creating a Performance Table	35
About Performance Tables	35
Naming a Performance Table for Use with the Define	
Performance Task Wizard	36
Create a Performance Table	37

Overview of Data Sources

Data sources are 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.

You can use prototype tables to import the input and output variables that SAS Model Manager uses to define projects. SAS Model Manager does not use any data in a prototype table except for 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

If you use prototype tables to define project input and output variables, the tables must be registered as data sources in SAS Management Console before you create a project.

Note: An alternative to using prototype tables to define the project input and output variables is to copy the variables from the champion model when you declare a default version or to modify the project definition. For more information, see [“The Default Version for a Project” on page 198](#) and [“Modify Project Definition” on page 51](#).

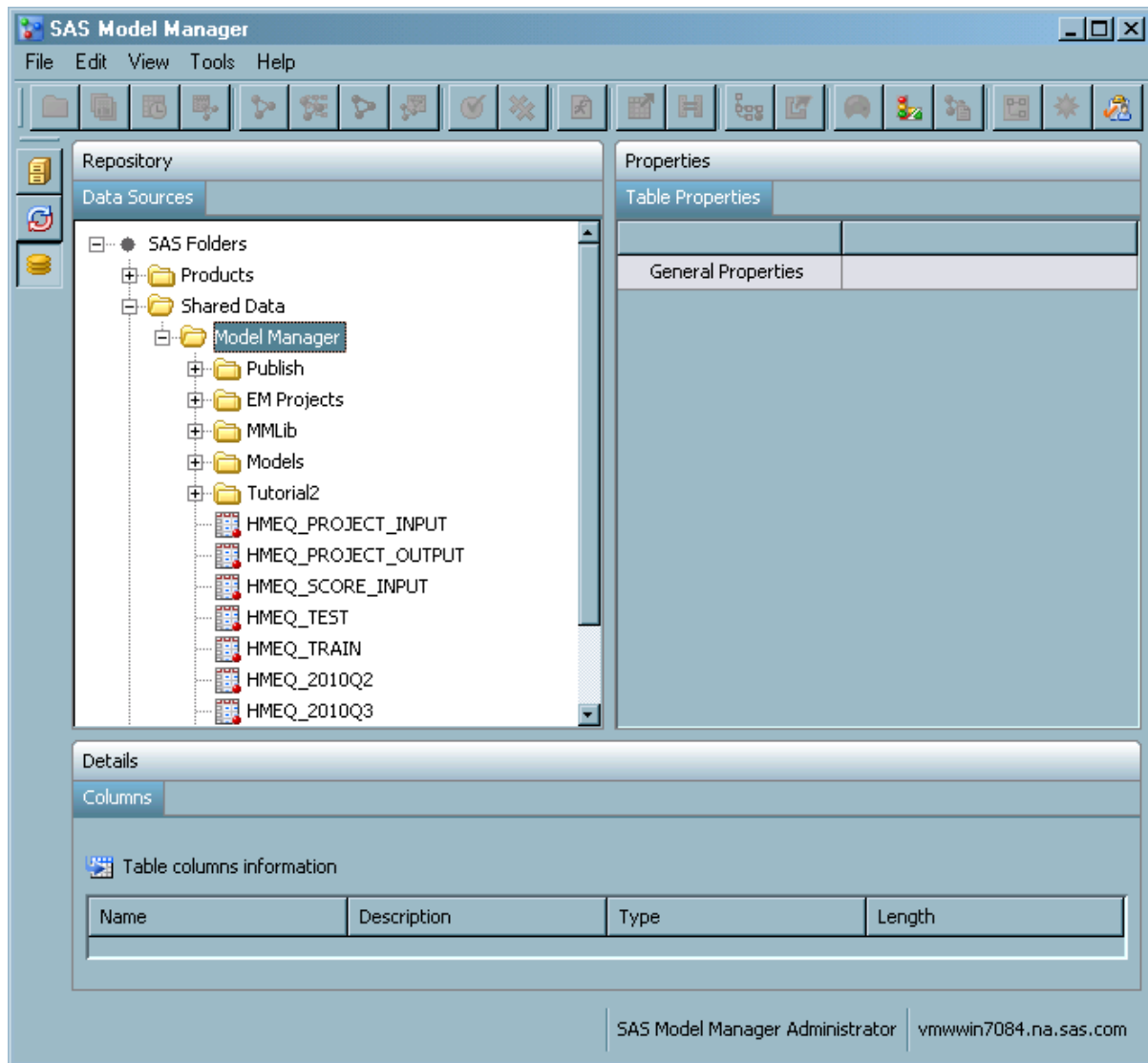
Use the following data tables to train or validate models, test models, and create monitoring reports:

- scoring task input tables
- scoring task output tables
- train tables
- test tables
- performance tables

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

Tables can be registered only by a user who has Write access to the SAS Metadata Repository. After you create a table, it must be registered in the SAS Metadata Repository using SAS Management Console.

After tables are registered in the SAS Metadata Repository, you can view them through the SAS Model Manager Data Sources perspective.



For information about registering data sources, see the *SAS Model Manager: Administrator's Guide*.

Project Tables

Project Input Tables

A project input table is an optional SAS data set that contains the champion model input variables and their attributes. It is a prototype table that can be 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 all champion model input variables, the variables in the project input table are a superset of all 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.

If you use a prototype table to define the project input variables, create the table and register the table using SAS Management Console before you create the project. In SAS Model Manager, you can view project input tables in the Data Sources perspective.

Note: An alternative to using prototype tables to define the project input and output variables is to copy the variables from the champion model when you declare a default version or to modify the project definition. For more information, see [“The Default Version for a Project” on page 198](#) and [“Modify Project Definition” on page 51](#).

See Also

- [“About Defining Project Input and Output Variables” on page 48](#)
- [“Creating Project Input and Output Tables” on page 32](#)

Project Output Tables

A project output table is an optional SAS data set or database table that defines project output variables and variable attributes such as data type and length. It is a prototype table that contains a subset of the output variables that any model in the project might create.

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

If you use a prototype table to define the project output variables, create the table and register the table using SAS Management Console before you create the project. In SAS Model Manager, you can view project output tables in the Data Sources perspective.

Note: An alternative to using prototype tables to define the project input and output variables is to copy the variables from the champion model when you declare a default version or to modify the project definition. For more information, see [“The Default Version for a Project” on page 198](#) and [“Modify Project Definition” on page 51](#).

See Also

- [“About Defining Project Input and Output Variables” on page 48](#)
- [“Creating Project Input and Output Tables” on page 32](#)

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 must create a scoring task input table and register it in the SAS Metadata Repository using SAS Management Console. In SAS Model Manager, you can view scoring task input tables in the Data Sources perspective.

See Also

[“Creating Scoring Task Input and Output Tables” on page 34](#)

Scoring Task Output Tables

A scoring task output table is used by a scoring task to define the variables for the scoring results table.

Depending on the mode in which a scoring task is run, the scoring task output table can be a prototype table or a physical data 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 scoring task runs, it creates a scoring results table. In test mode, the scoring results table is stored in the SAS Model Manager model repository. You can view the scoring results table under the scoring task node in the **Score** folder. The scoring task output table in the SAS Metadata Repository is not updated in test mode. In production mode, the contents of the scoring task output table in the SAS Metadata Repository are replaced by the contents of the 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 and accessible from the Data Sources perspective. To add the scoring task output table to SAS Model Manager, perform one of the following actions:

- Add the table manually by creating the table, and then registering it in the SAS Metadata Repository using SAS Management Console.
- Use the SAS Model Manager Create Output Table window. When you use the Create Output Table window, SAS Model Manager creates the table, and then registers it in the SAS Metadata Repository.

In SAS Model Manager, you can view scoring task output tables in the Data Sources perspective.

See Also

[“Creating Scoring Task Input and Output Tables” on page 34](#)

Train Tables

A train table is used to build predictive models. Whether your predictive models are created using SAS Enterprise Miner or you created SAS code models, you used a train table to build your predictive model. SAS Model Manager uses this same train table. The train table must be registered in the SAS Metadata Repository.

You specify a train table as a version-level property. When you define the train table at the version level, the table can be used to build all predictive models that are defined in the version's **Models** folder.

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

Note: A train table cannot contain an input variable name that starts with an underscore.

For information about registering train tables using SAS Management Console, see the *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 they 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 to be viewed in SAS Model Manager. In SAS Model Manager, you can view test tables in the Data Sources perspective.

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 test tables using SAS Management Console, see the *SAS Model Manager: Administrator's Guide*.

See Also

[“Creating a Test Table” on page 35](#)

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 the predicted target variable values. A performance table is a sampling of operational data that is taken at a single point in time. Each time you run a performance task, you use a new performance table to take a new sampling of the operational data. For example, a champion model is deployed to a production environment for the first time in March 2010. You might want to take a new sampling of the operational data in June 2010, September 2010, and January 2011. These new tables are performance tables in the context of SAS Model Manager.

Performance tables must be registered in the SAS Metadata Repository using SAS Management Console to be viewed in SAS Model Manager. You can view performance tables in the Data Sources perspective. After a performance table is registered, 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 SAS Model Manager report macros outside of SAS Model Manager to monitor a champion model's performance, the macros cannot access the performance tables in SAS Model Manager to create model performance monitoring reports.

See Also

[“Creating a Performance Table” on page 35](#)

See Also

[“Deleting a Data Table” in Chapter 5 of *SAS Model Manager: Administrator's Guide*](#)

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.

Here is one method that you can use to create the project input table from the train table. 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
           ninq 8 job $7 clno 8 derog 8
           clag 8 loan 8;
run;
```

If you find that you need to modify the project input variables after you have created a project input table, you can use the Modify Project Definition window to modify the project variables. For more information, see [“Modify Project Definition” on page 51](#).

See Also

- [“About Defining Project Input and Output Variables” on page 48](#)
- *SAS 9.3 Statements: Reference*
- *SAS 9.3 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 use 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 output table using the LENGTH statement to specify the output variable and its type and variable length:

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

If you find that you need to modify the project output variables after you have created a project input table, you can use the Modify Project Definition window to modify the project variables. For more information, see [“Modify Project Definition” on page 51](#).

See Also

[“About Defining Project Input and Output Variables” on page 48](#)

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 you can view the table in the Data Sources perspective of SAS Model Manager. 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 in the SAS Metadata Repository using SAS Management Console.

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 then you can view the table in the Data Sources perspective of SAS Model Manager. For information, see [“Create Scoring Output Tables” on page 153](#).

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 173](#)

Creating a Performance Table

About Performance Tables

Here are the requirements for a performance table:

- the input variables that you want reported in a Characteristic report
- if you have score code:
 - all input variables that are used by the champion model
 - all output variables that are used by the champion model

- if you have no score code:
 - the actual value of the dependent variable and the predicted score variable
 - all output variables that you want reported in a Stability Report

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. When you create a performance table, you can collect and name the performance table using a method that best suits your business process. Here are two methods of collecting performance data:

- Method 1: You periodically take a snapshot of an operational data set to create a performance data set. Each time you take a snapshot, you give the performance data set a new name. Each performance data set must be registered in SAS Management Console. For each time interval, you name a new performance data source when you run the Define Performance Task wizard.
- Method 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 each performance data set every time that you take a snapshot. You register the performance data set with SAS Management Console 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.

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.

For more information, see [“Determine How to Use the Performance Data Sets”](#) on page 242.

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;
```


Chapter 4

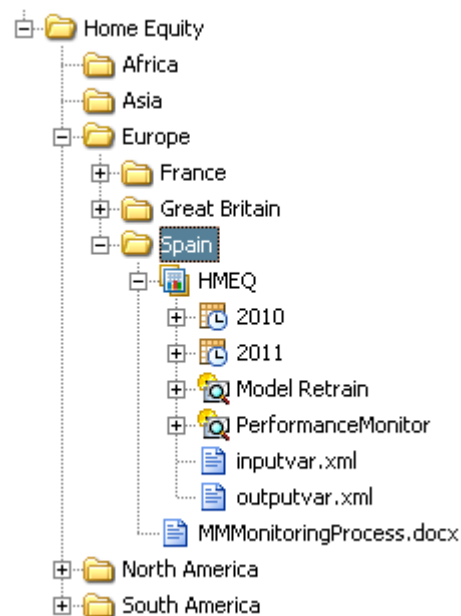
Organizing the Project Tree

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, **2010** and **2011**.

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 48
- “Associate Documents with a Folder” on page 41
- “Create a Version” on page 76

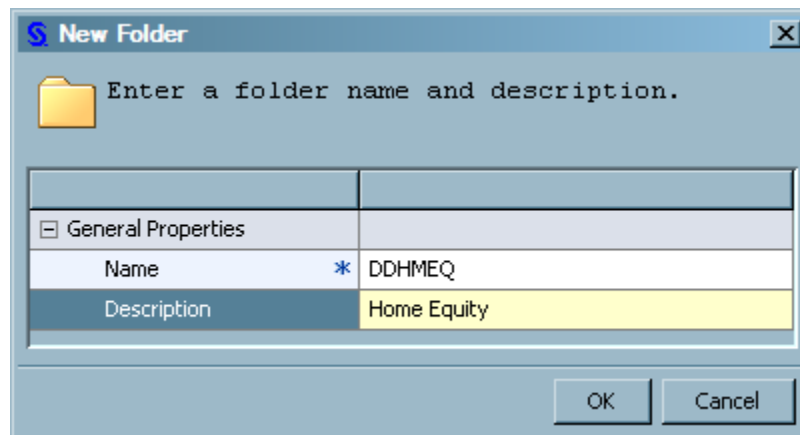
Here are some other common tasks that you can perform on an organizational folder, a project folder, or a version folder:

- Publish models. For more information, see “Publishing Models” on page 204.
- Search for models or tasks that are assigned to SAS Model Manager users. For more information, see Appendix 1, “Query Utility,” on page 303.

Create an Organizational Folder

To create a folder, follow these steps:

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



2. In the **Name** field, type a folder name. The characters @ \ / * % # & \$ () ! ? < > ^ + ~ ` = { } [] | ; : ' " cannot be used in the 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 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 File**.



The Attach File window appears.

2. In the Attach File window, select the document that you want to attach to the folder.
3. The file format is preselected for you when you select a file. To change the file format, select **Binary or XML** or **Text**.
4. Click **OK**. SAS Model Manager attaches the document to the folder.

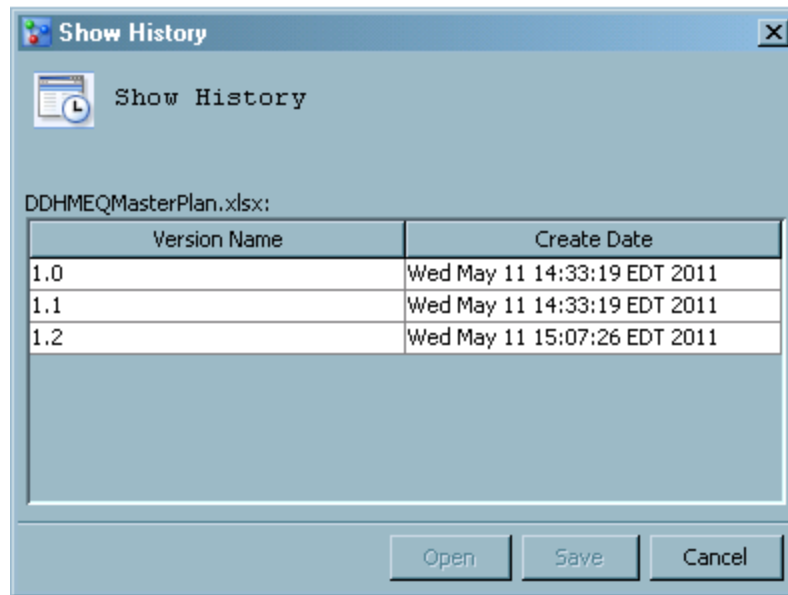
Here is an example of a document that is attached to a folder:



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 appears. 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**.

Note: When a version is frozen, **Delete** is disabled for the frozen version as well as all models in the **Models** folder.

2. Click **Yes** 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	46
Prerequisites for Creating Projects	47
About Defining Project Input and Output Variables	48
Create a Project	48
Modify Project Definition	51
Lock or Unlock Project Metadata	53
Setting the Project Champion Model Status	53
About the Champion Model Status	53
Setting the Champion Model Status	53
Project Properties	54
About Project Properties	54
Project-Specific Properties	54

Overview of Projects

What Is a SAS Model Manager Project?

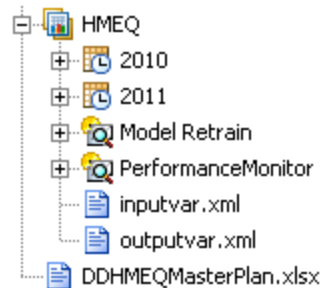
A SAS Model Manager project consists of the models, reports, documents, scoring tasks, and other resources that you use to determine a champion model. For example, a banking project might include models, data, and reports that are used to determine the champion model for a home equity scoring application. 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 [Chapter 6](#), “Working with Versions,” on page 59.

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, a **Model Retrain** folder, a **PerformanceMonitor** folder, the project input and output XML files, 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. They contain life cycle information, candidate model files, model comparison reports, resource files, scoring tasks, and model performance reports. This Project Tree has two versions, **2010** and **2011**.

Model Retrain

contains reports for models that have been retrained.

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 variables that are defined when you create a project, modify a project, or declare a default version.

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 variables that are defined when you create a project, modify a project, or declare a default version.

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 **DDHMEQMasterPlan.xlsx**.

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 76](#).

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 205](#).

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 208](#).

Publish Scoring Function

transforms DATA step score code of a predictive model to user-defined function (UDF) executable files and writes the UDF files to the database. For more information, see [“How to Publish a Scoring Function” on page 217](#).

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 241](#).

Define Dashboard Report Indicators

starts a wizard to create performance indicators that are used to execute the dashboard reports. For more information, see [“Define Dashboard Report Indicators” on page 279](#).

Define Model Retrain Task

starts a wizard that retrains one or more models. For more information, see [Chapter 19, “Retraining Models,” on page 289](#).

Lock Project Metadata

enables a SAS Model Manager administrator to lock the project metadata so that the project definition cannot be modified while it is locked. For more information, see [“Lock or Unlock Project Metadata” on page 53](#).

Unlock Project Metadata

enables a SAS Model Manager administrator to unlock the project metadata so that the project definition can be modified. For more information, see [“Lock or Unlock Project Metadata” on page 53](#).

Modify Project Definition

enables a user to modify the project properties and the project input and output table variables for the selected project. For more information, see [“Modify Project Definition” on page 51](#).

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 303](#).

View Workflow

enables the user to view the workflow instance that is associated with the selected project. For more information, see [“Viewing Workflow Instances” on page 109](#).

New Workflow Instance

creates a new workflow instance of a process definition and associates it with the selected project. For more information, see [“Creating a New Workflow Instance” on page 107](#).

Planning a Project

Before you begin a project, you must plan your project resources. Here is a list of questions to consider and conditions to meet 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 [Chapter 4, “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 or a model created by third-party software such as R, you must ensure that you have imported all of the model component files. For more information, see [“Import SAS Code Models and R Models Using Local Files” on page 123](#).
- How do you want to define your project input and output variables? When you create a project, you can import the variables using input and output prototype tables, you can copy the variables from an existing champion model, or you can define individual variables. If you use prototype tables to define the project input and output variables, the tables must be registered in the SAS Metadata Repository using SAS Management Console before you create the project. For more information, see [“About Defining Project Input and Output Variables” on page 48](#).
- What method do you want to use to track the progress of a project or version? The Workflow Console enables you to track the progress of activities from the project level or a version level. A SAS Model Manager administrator can create and associate a workflow instance with a project or version. You can also use the life cycle feature to track the life cycle of a model at the version level.
 - If you decide to use the workflow process to track the progress of activities for a project or version, you do not need to use the life cycle feature to monitor the progress of milestones and tasks. For more information, see [“Overview of Workflow Console” on page 86](#).
 - If you decide to use the life cycle feature to monitor the progress of your project or version, you must plan your milestones and the tasks for each milestone before you can create a version for a project. 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 62](#).
- 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 [Chapter 10, “Validating Models Using Comparison Reports,” on page 167](#) and [Chapter 11, “Validating Models Using User Reports,” on page 177](#).
- When you export a project champion model to the SAS Metadata Repository, you must specify a folder to where you export the project champion model. 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 208](#).
- 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 35](#).
- 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 234](#).

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 that you know the model function type before you create a project.

SAS Model Manager has several model function types:

- Analytical
- Prediction
- Classification
- Segmentation
- Any

To determine the model function type for your project, compare your model to the descriptions in the table [Types of Model Functions on page 57](#).

If you use prototype tables to define the project input and output variables, the project input and output tables must be created and registered in the SAS Metadata Repository using SAS Management Console before you can create a project. You then can view the data tables from the Data Sources perspective in SAS Model Manager. 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 32](#).
- The *SAS Model Manager: Administrator's Guide* has instructions on registering project input and output tables in SAS Management Console.

About Defining Project Input and Output Variables

Project input and output variables are the input and output variables that are used by the champion model. SAS Model Manager requires that the project input and output variables be defined before a champion model can be exported or published to a production environment. You can define the project input and output variables when you create a project, during the champion model selection process, or when you declare a default version.

You define the project input and output variables using one of the following methods:

- Create input and output prototype tables and import the variables using these tables. You import the variables using the **New Project** wizard or the Modify Project Definition window.
- Copy the input and output variables from an existing champion model. You copy the champion model variables using the **New Project** wizard or the Modify Project Definition window.
- Use the Modify Project Definition window to add , modify, or delete individual input or output variables.
- If you declare a default version and the project variables have not been defined, SAS Model Manager uses the champion model variables to define the input and output variables. If the project variables have been defined, you are prompted to map model output variables to project output variables.

SAS Model Manager saves the input variables' definitions in the file `inputvar.xml` and the output variables definitions in the file `outputvar.xml`. The `inputvar.xml` file and the `outputvar.xml` file are saved in the project folder. When you declare a default version, the `inputvar.xml` file and the `outputvar.xml` file are copied to the version folder.

See Also

- [“Project Tables” on page 29](#)
- [“Creating Project Input and Output Tables” on page 32](#)
- [“Create a Project” on page 48](#)
- [“Modify Project Definition” on page 51](#)
- [“Set a Default Version” on page 198](#)

Create a Project

To create a project, follow these steps:

Note: SAS Model Manager does not support tables, models, or variables that contain special characters and that were created when the system options `VALIDVARNAME=ANY` and `VALIDMEMNAME=EXTEND` were set.

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

Property	Value
General Properties	
Name *	HMEQ
Description	
Project Properties	
Model Function *	Classification

Buttons: Back, Next, Finish, Cancel, Help

2. Enter a name and a description for the project that you are creating. The characters @ \ / * % # & \$ () ! ? < > ^ + ~ ` = { } [] | ; : ' " cannot be used in the name. The **Name** field is required.
3. Click the **Model Function** list box and select the function type for the model. Click **Next**.
4. Specify the project input and output variables for the project.
 - Click **Import Variables** to import input variables or output variables from a data set that is located in the SAS Metadata Repository.
 - Click **Copy Variables** to copy variables from another project.
 - Click **Add** to manually enter a new variable.

New Project

Set Project Variables

Specify the project input and output variables for this project.

Step 2 of 2

Project Input Variables

Name	Type	Measur...	Length	Descrip...
YOJ	N		8	
MORTDUE	N		8	

Import Variables Copy Variables Add Edit Delete

Project Output Variables

Name	Type	Measur...	Length	Descrip...
score	N		8	

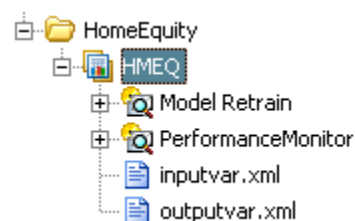
Import Variables Copy Variables Add Edit Delete

Back Next Finish Cancel Help

Note: You can also edit or delete existing input variables and output variables.

5. Click **Finish**. 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 **Model Retrain** node and a **PerformanceMonitor** node. It uses the project input and output variable tables to create the two XML files.

- You use the **Model Retrain** node to retrain a champion model after it has been in production for a while.
- 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.

See Also

[“About Defining Project Input and Output Variables” on page 48](#)

Modify Project Definition

This feature is used to modify the project properties and the project input and output variables for a project when the existing table structure is changed using SAS Management Console. Alternatively, you can use this feature to assign different data variables. For example, suppose you selected the wrong variables when you created a project. You do not realize the mistake until you try to set a champion model.

Note: Only SAS Model Manager administrators can use this feature.

The project input and output variables cannot be modified in the following cases:

- The project contains a frozen default version.
- The project metadata is locked.

CAUTION:

Modifying the project definition could invalidate the project champion model in the default version. If you modify the project input or output variables, inconsistencies might occur in the champion model input and output variables. The default version for the project is also cleared.

To modify a project input table or project output table property, follow these steps:

1. Right-click the project folder in the Project Tree and select **Modify Project Definition** from the pop-up menu. The Modify Project Definition window appears.

Modify the project properties, or project input and output variables.

Project Properties

Property	Value
General Properties	
Name *	HMEQ
Description	

Project Input Variables

Name	Type	Measure...	Length	Description
YOJ	N		8	
MORTDUE	N		8	

Import Variables Copy Variables Add Edit Delete

Project Output Variables

Name	Type	Measur...	Length	Description
score	N		8	

Import Variables Copy Variables Add Edit Delete

OK Cancel

2. To modify a project property, click the property field in the **Value** column and type a new value.
3. Modify the project input and output variables for the project.
 - Click **Import Variables**, to import input variables or output variables from a data set that is located in the SAS Metadata Repository.
 - Click **Copy Variables** to copy variables from another project.
 - Click **Add** to manually enter a new variable.
 - Select a variable and click **Edit** to modify the variable information.
 - Select a variable and click **Delete** to remove the variable from the project. A confirmation message is displayed. Click **Yes** to delete the variable.
4. Click **OK**.

See Also

- [“Specific Properties for a Project” on page 359](#)

- [“Project Tables” on page 29](#)
- [“Create a Project” on page 48](#)

Lock or Unlock Project Metadata

- A SAS Model Manager administrator can lock or unlock the metadata for a project. If the metadata is locked for a project, you cannot modify the project definition.
- To lock the metadata for a project, right-click the project folder and select **Lock Project Metadata**.
 - To unlock the metadata for a project, right-click the project folder and select **Unlock Project Metadata**.

See Also

[“Modify Project Definition” on page 51](#)

Setting the Project Champion Model Status

About the Champion Model Status

You can set the project **State** property to indicates 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 can 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

Property Name	Description
Lock Project Metadata	Specifies that the project metadata is locked and the project definition cannot be modified.
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.

Property Name	Description
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 scoring functions when a user publishes the associated project champion model to a database.</p> <p>The value of the Default Train Table property is used to validate scoring functions only if a default train table is not specified for a version or for the model.</p>
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	<p>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.</p>
Default Version	<p>Specifies the version that contains the champion model in a production environment.</p>

Property Name	Description
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 57 .
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, ordinal, 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.
Class Target Level	Specifies the class target level of binary, nominal, ordinal, or interval.
Output Event Probability Variable	The output event probability variable name, when the Model Function property is set to Classification .
Output Prediction Variable	The output prediction variable name, when the Model Function property is set to Prediction .
Output Classification Variable	The output classification 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
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	59
What Is a SAS Model Manager Version?	59
How a Version Folder Is Organized	60
Version Folder Tasks	61
Creating Life Cycle Templates	62
Overview of Creating Life Cycle Templates	62
The SAS Model Manager Template Editor Window	63
Life Cycle Template Participants	65
The Browse Templates Window	67
Create a Life Cycle Template from a Sample Template	68
Create a New Life Cycle Template	69
Modify a Life Cycle Template	71
Delete a Life Cycle Template	72
Life Cycle Template Properties	72
Create a Version	76
Version Properties	77
About Version Properties	77
Version-Specific Properties	77
Working with Life Cycles	79
Overview of a Life Cycle	79
Life Cycle Tasks	80
Life Cycle Properties	82

Overview of Versions

What Is a SAS Model Manager Version?

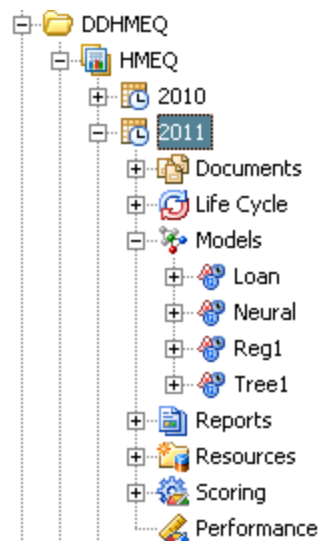
After a project is created, you create a version folder to import your models, score your models, run reports, and monitor the life cycle of these models. A version is often the time-phased container for your SAS Model Manager projects. The time interval for a project cycle is specified when you create 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 2011. 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 roadmap, 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.

inputvar.xml

contains a copy of the project input variables. The project inputvar.xml file is copied to the version inputvar.xml file when a model is set as a champion model. If the project input variables are modified after a champion model is selected, SAS Model Manager knows the required input variables for the champion model.

outputvar.xml

contains a copy of the project output variables. The project outputvar.xml file is copied to the version outputvar.xml file when a model is set as a champion model. If the project output variables are modified after a champion model is selected, SAS Model Manager knows the required output variables for the champion model.

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 198](#).

Clear Default Version

deselects the version as the default version. For more information, see [“Clear a Default Version” on page 200](#).

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 196](#).

Unfreeze Version

enables modifications of version model properties and files. For more information, see [“Unfreeze a Version” on page 197](#).

Publish

publishes models to defined channels and notifies subscribers of the publication channel when the models are delivered. For more information, see, [“Publishing Models” on page 204](#).

Query

searches for model and tasks that are assigned to SAS Model Manager users. For more information, see [Appendix 1, “Query Utility,” on page 303](#).

View Workflow

enables the user to view the workflow instance that is associated with the selected version. For more information, see [“Viewing Workflow Instances” on page 109](#).

New Workflow Instance

creates a new workflow instance of a process definition and associates it with the selected version. For more information, see [“Creating a New Workflow Instance” on page 107](#).

See Also

- [“Overview of Importing Models” on page 119](#)
- [“Exporting Models” on page 208](#)

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 Cycles perspective. Templates that appear in the Life Cycle perspective are the life cycle templates that have been uploaded to the SAS Content Server.

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. When you create a life cycle template using the **SAS Model Manager Template Editor**, you can browse existing templates and select one to modify using a new name or you can create a new life cycle template.

SAS supplies four life cycle templates that you can use to create a template: Basic, Standard, Extended, and User Lifecycle templates. The Basic, Standard, and Extended templates are reserved templates and are provided only as examples. They are not intended for use by any organization. Reserved templates cannot be modified, but they can be saved using another name to create a new template. Templates that are not

reserved can be uploaded to the SAS Content Server. The Browse Templates window indicates which templates are reserved.

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 a template, the template is saved to a local or network location as an XML file using the required XML element structure. You save templates to create a backup of a template. A template can be used in SAS Model Manager only by uploading the template to the SAS Content Server. The SAS Model Manager Template Editor provides an **Upload File** menu selection.

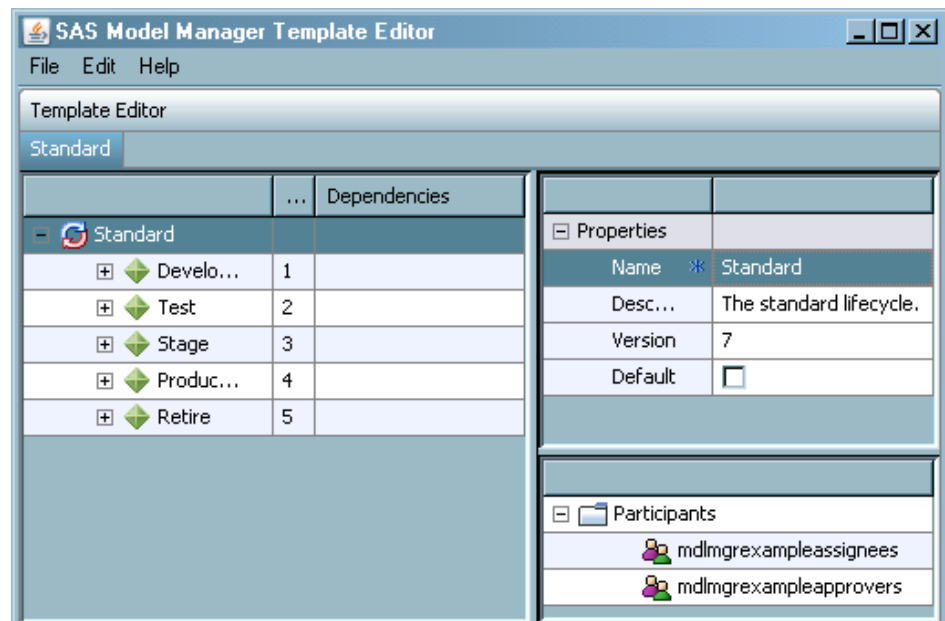
If you create a life cycle template using an XML file, you can copy any life cycle template from the user-templates directory and modify the file with any text editor. When you modify an XML template file, you specify the milestone and task properties as XML element and element attributes. SAS Model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

The SAS Model Manager Template Editor Window

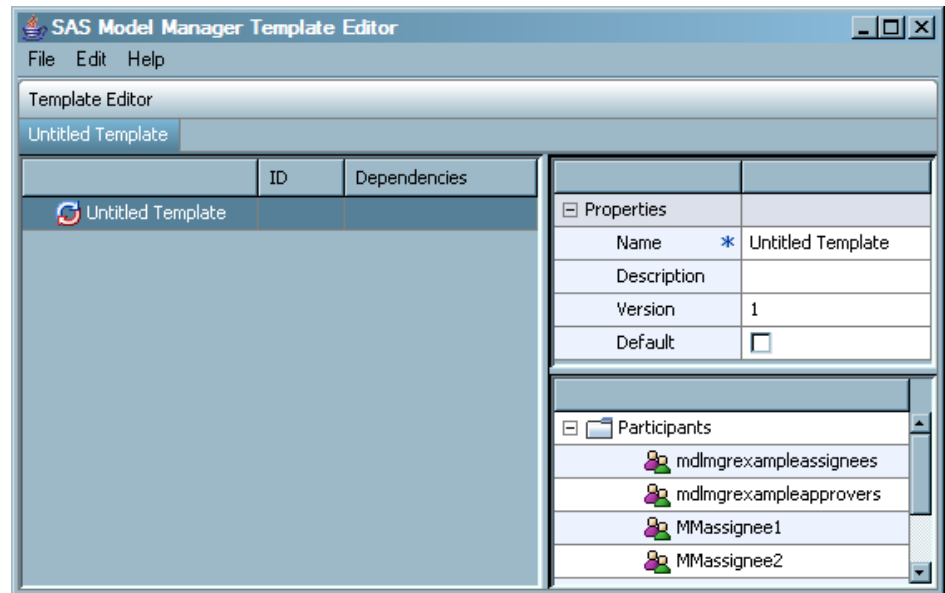
You use the SAS Model Manager Template Editor window to create or modify a life cycle or model template.

To open a life cycle template in the **SAS Model Manager Template Editor** window, follow these steps:

1. From the SAS Model Manager window, select **Tools** ⇒ **Manage Templates**.
2. From the **File** menu, open a life cycle template:
 - To open an existing template on the SAS Content Server, select **Browse** ⇒ **Browse Templates**. The Browse Templates window appears. Select a template and click **Open**.



- To open a new life cycle template, select **New Life Cycle Template**.



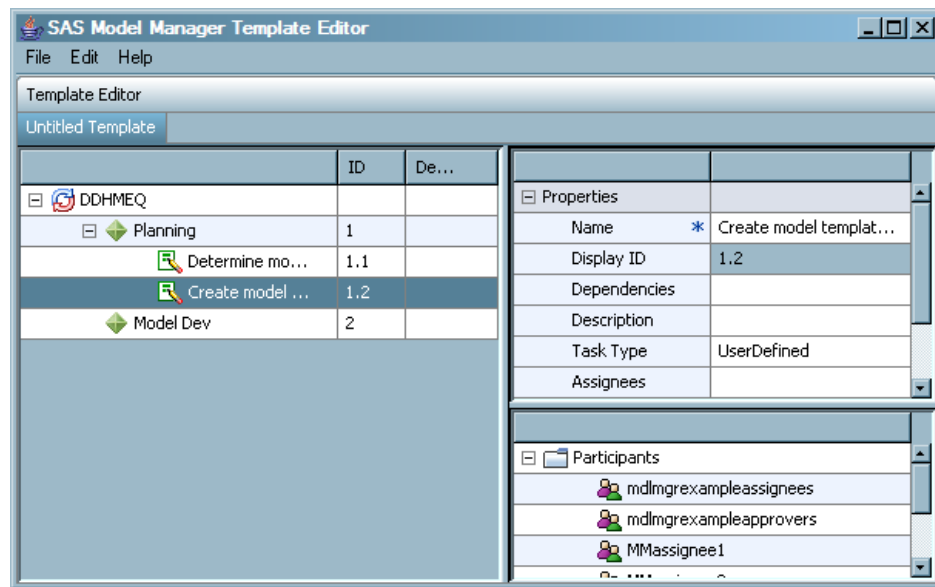
- To open a life cycle template that is stored on a local or network location, select **Open**.

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 72](#).

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 or approved.

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 or approved:

- **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.

Note: In order to change life cycle properties in SAS Model Manager, a user or a group must be assigned to the respective life cycle roles and the role of either Model Manager: Administration Usage or Model Manager: Advanced Usage.

- **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.

In the **Properties** pane, 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 as a value in a version's **Life Cycle** node task properties **To Be Completed By** and **To Be Approved By**.

- If any user or group that you select in the Select Participants window is assigned the role **Model Manager: Life Cycle Assignee Usage** in the SAS Management Console **User Manager**, 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** in SAS Management Console, 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** in SAS Management Console, 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.

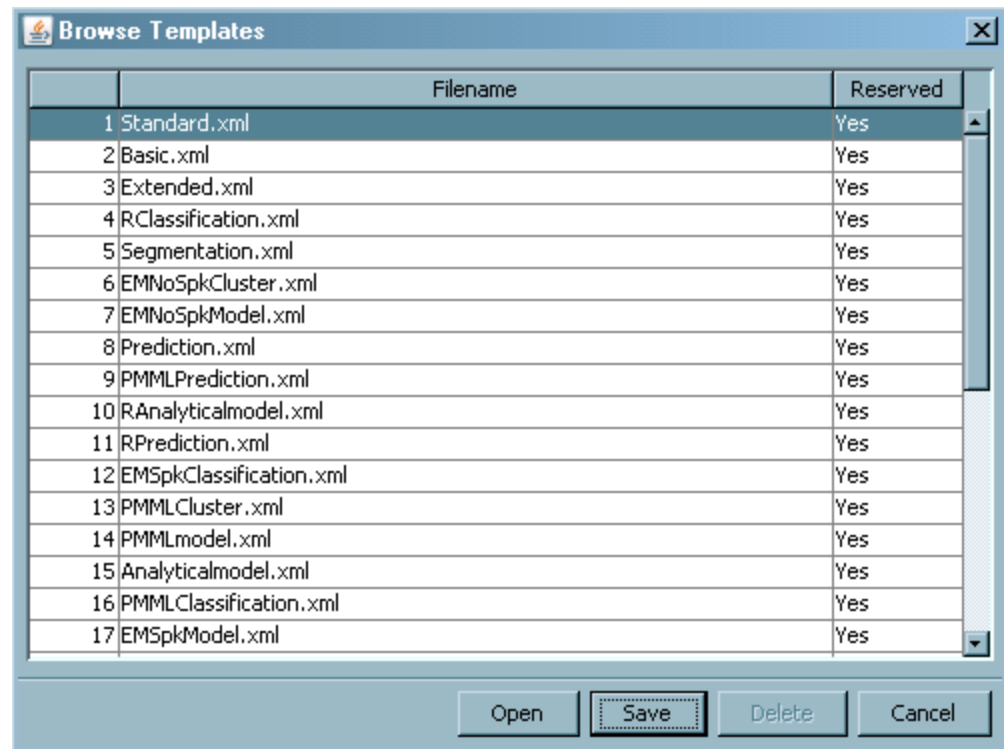
When you assign a group to be an **Assignee** or an **Approver**, all users and groups in that group have the authority to change the task status. Therefore, ensure that the users and groups that are defined in the group are those users and groups that you intend to be an **Assignee** or **Approver**.

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.

The Browse Templates Window

Using the Browse Templates window, you can access life cycle templates that are used by SAS Model Manager and are stored on the SAS Content Server.

To open the Browse Templates window, from the SAS Model Manager Templates Editor, select **File** ⇒ **Browse** ⇒ **Browse Templates**.



The Browse Templates window lists life cycle and model templates that are stored on the SAS Content Server. The first three templates, Standard.xml, Basic.xml, and Extended.xml are life cycle templates that are supplied by SAS. A **Yes** value in the **Reserved** column indicates that the template cannot be modified. These templates are supplied by SAS. A **No** value indicates that the template can be modified.

You can perform the following tasks in the Browse Templates window:

- To open a template in the Template Editor, select a template and click **Open**.
- To save a template to a local or network location, select the template and click **Save**.
- To delete a template, select the template and click **Delete**.

Create a Life Cycle Template from a Sample Template


SAS Model Manager supplies sample life cycle templates (Basic, Standard, Extended, and UserLifecycleTemplate) that you can use to create a life cycle template. To view the sample templates, select **File** ⇒ **Browse** ⇒ **Browse Templates**. The templates that have **Yes** in the **Reserved** column of the Browse Templates window cannot be modified. Select a template and click **Open**. You can also view sample templates in the Life Cycles Perspective of the SAS Model Manager window.

Note: The UserLifecycleTemplate.xml template that is supplied by SAS is not a reserved template and can be modified. When you create a life cycle using this template, rename the template before you modify it. Only a user or group with the role of Model Manager: Administration Usage can upload a template.

To create a life cycle template from a sample template, follow these steps:

1. In the Template Editor window, select **File** ⇒ **Browse** ⇒ **Browse Templates**.
2. Select a template and click **Open**. The template name appears under **Template Editor**.

3. Rename the template filename and **Name** property:
 - a. Select **File** ⇒ **Save As**.
 - b. Select a directory and a filename for the template. The filename must have an extension of .xml (for example, myLifeCycle.xml). The new template filename appears under **Template Editor**.
 - c. Modify the template **Name** property value.

Note: You cannot upload a template if the value of the **Name** property is the same for a template that has been uploaded to the SAS Content Server.
4. Modify the template:
 - a. To add a milestone, right-click the template name and select **New Milestone**. In the New Task window, complete the **Name**, **Description**, and **Type** fields. The **Name** and **Type** fields are required.
 - b. To modify milestone properties, click the property and modify the properties in the **Properties** pane. Properties with an * are required. For a description of the properties, see [“Milestone Properties” on page 73](#).
 - c. To add a task, right-click a milestone and select **New Task**. In the New Task window, complete the **Name**, **Description**, and **Type** fields. The **Name** and **Type** fields are required.
 - d. For each task, complete the task properties. For tasks that have multiple values, such as Dependencies, Assignees, and Approvers, click the value field for a list of values or click  to assign a value. For a description of the properties, see [“Task Properties” on page 74](#).
 - e. 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 no tasks are dependent on later tasks in the tree structure.

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.
 - f. To delete a milestone or a task, right-click the milestone name or task name and select **Delete**.
5. When you have completed your changes, upload the template by selecting **File** ⇒ **Upload File**.

Note: Only a user or group with the role of Model Manager: Administration Usage can upload a template.
6. To save the template, select **File** ⇒ **Save**. Select a directory and a filename for the template. Saving the template creates a backup copy of the 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 Cycles perspective. Only a user or group with the role of Model Manager: Administration Usage can upload a template.

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 window, select the folder to save the template to. In the **File name** field, type the life cycle template name with an extension of .XML and click **Save**.
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 window, 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 72.
6. Create milestones for the life cycle. For each new milestone, right-click the template name and select **New Milestone**. The New Milestone window appears.

New Milestone	
General Properties	
Name *	Testing
Description	
Type *	Develop
	Develop Test Staging Production Retire User-defined

Complete these fields:

- a. Enter a name and an optional 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 73. 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 appears.

New Task	
[-] General Properties	
Name	Determine reports needed
Description	
Type	User-defined
<div>OK Cancel</div>	

Complete these fields:

- a. Enter a name and an optional 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 74](#).
8. For each task, complete the task properties. For more information, see [“Task Properties” on page 74](#).
 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. When you have completed your changes, upload the template by selecting **File** ⇒ **Upload File**.
Note: Only a user or group with the role of Model Manager: Administration Usage can upload a template.
 12. To save the template, select **File** ⇒ **Save**. Select a directory and a filename for the template. Saving the template creates a backup copy of the template.



Modify a Life Cycle Template

To modify a life cycle template, follow these steps:

1. From the SAS Model Manager window, select **Tools** ⇒ **Manage Templates**.
2. Select one of the following templates to open:
 - a. To open a template on the SAS Content Server, select **File** ⇒ **Browse** ⇒ **Browse Templates**. Select a template and click **Open**.

- b. To open a backup copy of a template, select **File** ⇒ **Open**. Select the local or network location, select the file, and click **OK**.
3. To modify life cycle properties, select the property and make changes to the property value. For more information, see [“Template Properties” on page 72](#).
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 73](#).

5. Create or modify a task:
 - To create a new task, right-click a milestone and select **New Task**. Complete the task properties. Click  to make changes to the property value.
 - To modify a task property, select the property and make changes to the property value. Click  to make changes to the property value.

For more information, see [“Task Properties” on page 74](#).

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. When you have completed your changes, upload the template by selecting **File** ⇒ **Upload File**.

Note: Only a user or group with the role of Model Manager: Administration Usage can upload a template.

Each time you upload a template to the SAS Content Server, SAS Model Manager increments the template’s **Version** property value by 1. If you create a backup copy of the template after you upload the template to the SAS Content Server, increment the **Version** property value by 1 and then save the template. This action ensures that your backup copy is the same version as the version on the SAS Content Server.

8. To save the template, select **File** ⇒ **Save**. Select a directory and a filename for the template. Saving the template creates a backup copy of the template.

Delete a Life Cycle Template

To delete a life cycle template, follow these steps:

1. Open the Browse Templates window. From the SAS Model Manager Template Editor, select **File** ⇒ **Browse** ⇒ **Browse Templates**.
2. Select the template. Templates with a **Yes** in the **Reserved** column cannot be deleted.
3. Click **Delete**. Click **Yes** to confirm the deletion.

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 you upload a version of a template to the SAS Content Server, the version number is incremented 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. Select the check box to set the template to be the default template.</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.

Property Name	Description
Milestone Phase	<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>User-defined 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.
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 the display ID for a task that must be completed before this task can be completed.
Description	Displays user-defined information about the task.

Property Name	Description
Task Type	<p>Specifies a type for the task. Here is a list of valid task types:</p> <p>User-defined 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>Sign-off specifies that all of the milestone tasks are complete and have been approved.</p> <p>Declare Production specifies that the champion model is ready to be exported to the production environment.</p> <p>Set Champion specifies that the task is to determine a champion model. Before this task can be completed a champion model must be set for the version that contains the champion model.</p> <p>Retire Champion 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 the user who 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. • If any users or groups are assigned the role Model Manager: Life Cycle Assignee Usage, then only those users and groups can modify the task status. • If the role of Model Manager: Life Cycle Assignee Usage is not assigned to any user or group, then any SAS Model Manager user can modify the task status.
Approvers	<p>Specifies a user or a group from the Participants list. The specified user or any member of the specified group is the user who is responsible for approving the task and changing the approval status. 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. • If any users or groups are assigned the role Model Manager: Life Cycle Approval Usage, then only those users and groups can modify the task approval status. • If the role of Model Manager: Life Cycle Approval Usage is not assigned to any user or group, then any SAS Model Manager user can modify the task approval status.

Property Name	Description
Weight	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. <i>Note:</i> User-defined weights are not explicitly adjusted. Weights remain as they were entered and are not adjusted.
Duration	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.

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 appears.

2. Specify a name and an optional description for the new version. Use a name that is unique among versions in this project. The characters @ \ / * % # & \$ () ! ? < > ^ + ~ ` = { } [] | ; : ' " cannot be used in the name.
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 **Date Created** 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 59](#)
- [“Creating Life Cycle Templates” on page 62](#)

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.

Property Name	Description
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 scoring functions when a user publishes the associated project champion model to a database. • checks the Validate scoring results box in the Publish Scoring Function window. <p>The value of the Default Train Table property is used to validate 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.
Date Frozen	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 .
Date Retired	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

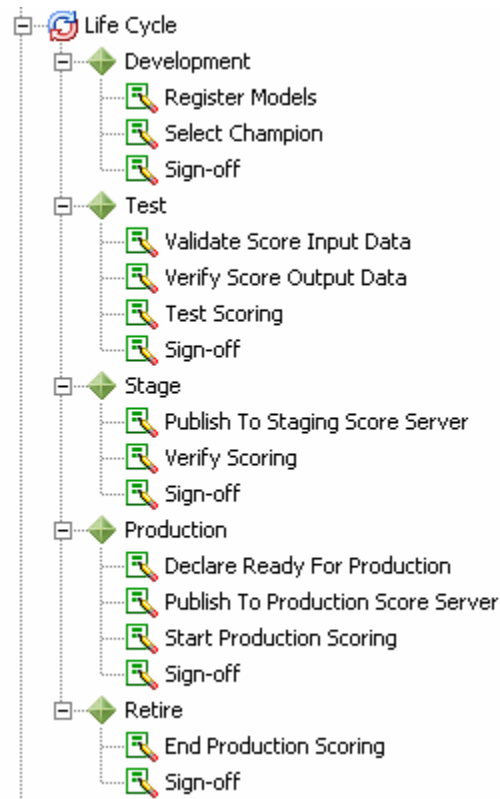
What Is a Life Cycle?

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 who 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 Cycles perspective. For more information, see [“View Life Cycle Templates” on page 80](#).
- Update the status for a milestone. For more information, see [“Update Milestone Status” on page 81](#).
- Search for a task. For more information, see [“Search Life Cycles for Tasks Assigned to Users” on page 307](#).

For information about users and groups who can update the **Status** property of a task, see [“Participant Roles” on page 65](#).

View Life Cycle Templates

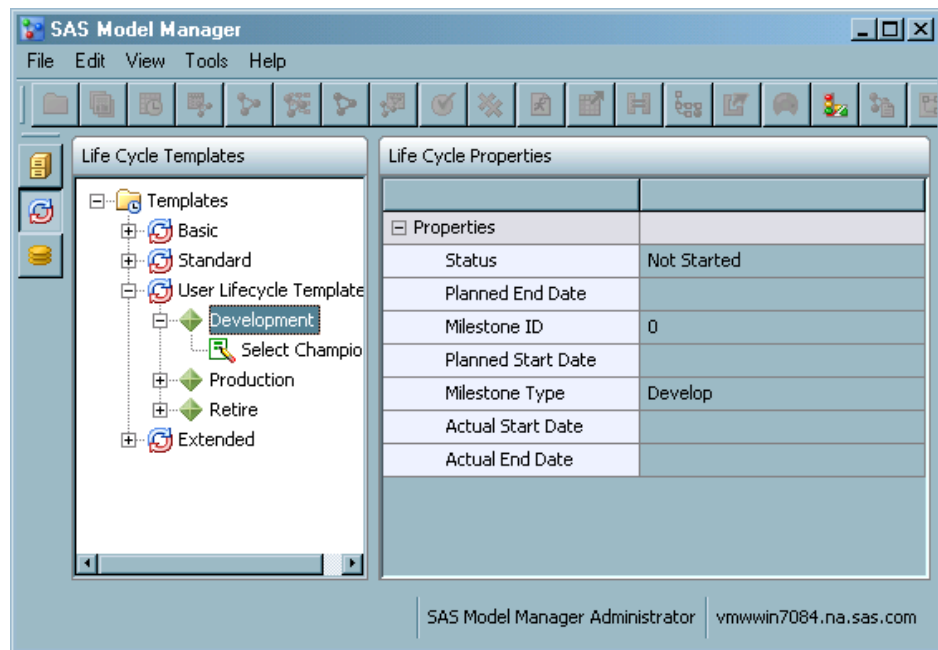
To view the life cycle templates that are available in SAS Model Manager, follow these steps:

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 Manager:
 - **Basic**
 - **Standard**

- **Extended**
- **User Lifecycle Template**

These life cycle templates are example templates that you can use as a model to create life cycle templates that meet your organization's 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 **Milestone ID** property. At the task level, sequential ordering is determined by the **Action ID** property.



4. Expand the milestone phase to view the milestone tasks. Examine the task requirements in the **Life Cycle Properties** pane.
 - **Action ID** 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 65](#).

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 **Date Modified** 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** contains 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.

Property Name	Description
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 .
Date Completed	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.
Date Approved	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.

Chapter 7

Using Workflow Console

Overview of Workflow Console	86
User Interface Layout	86
About the User Interface Layout	86
Navigation Pane	88
Customizing Category Views	88
Overview of Customizing Category Views	88
Manage Columns in a List	89
Sort Lists by Column Content	90
Filtering List Content	91
Setting Preferences	94
About Setting Preferences	94
Global Preferences	94
General Preferences	95
Alert Notifications	95
Working with Objects	96
About the Object Bar	96
Open Objects	97
Open Multiple Objects	97
Rearrange Open Objects	98
Save Layouts	98
Close Objects	99
Viewing Workflow Activities	99
Working with Workflow Activities	100
Editing Activity Properties	100
Working with Comments	101
About Comments	101
Adding Comments	102
Reply to a Comment	103
Attach a File	103
Sort and Filter Items in a Comment Topic Thread	104
Search for Comments and Replies	104
Managing the Workflow Process	105
Viewing Workflow Process Definitions	106
Creating a New Workflow Instance	107
Viewing Workflow Instances	109

Editing a Workflow Instance	110
Editing Instance Properties	112
Working with Workflow Participants	112
Assigning Participants to Activities	112
Removing Participants from Activities	114
Releasing an Activity	115
Terminating a Workflow Instance	116

Overview of Workflow Console

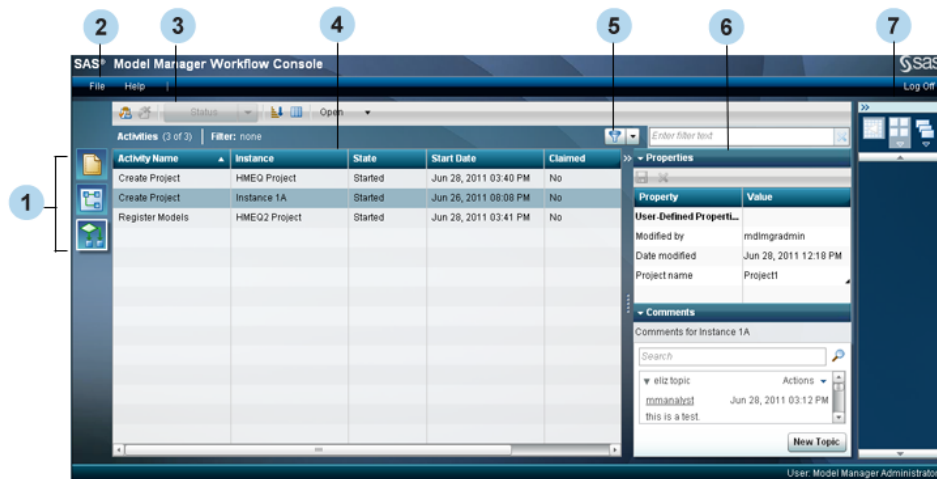
The SAS Model Manager Workflow Console is an interface to SAS Workflow that you can use to track the progress of a modeling project or version. A SAS Model Manager administrator or a SAS administrator uses SAS Workflow Studio to define process definitions and to make them available to SAS Model Manager for use. Process definitions contain the set of activities, participants, policies, statuses, and data objects that comprise a business task. After the process definitions are made available, the SAS Model Manager administrator uses Workflow Console to create workflow instances to be used with SAS Model Manager. A *workflow instance* is a working version of a workflow process definition. Each workflow instance consists of activities. Activities can contain user-defined properties and comments so that you can share information with other users, or make notes. The status that you select when completing an activity determines the next activity in the workflow process.

From the SAS Model Manager client application, you can view workflow instances, create a new workflow instance for a project or version, and view your workflow inbox to work with activities, depending on the user permissions. The option that is selected and the user permissions determine the category view and content that are displayed when Workflow Console is launched. SAS Model Manager administrators can view the Process Definitions, Instances, and Activities category views of Workflow Console. SAS Model Manager advanced users can view only the Activities category view. For more information about user permissions, see *SAS Model Manager: Administrator's Guide*.

User Interface Layout

About the User Interface Layout







Workflow Console is a workspace that includes a navigation pane with category buttons, a menu bar, a toolbar, a list area, a filter bar, property panes, and an object bar.



- 1 **Navigation pane:** Click a category button to switch to a different category. Your user privileges determine which categories and which category-specific features are available to you. Each category view displays a list of items and has a toolbar for working with the view and the displayed items. At the top of each view, you can filter the list by specifying the type of items that you want to display or group by. You can also search for specific types of items.
- 2 **Menu bar:** A menu bar is available at the top, left-hand side of the window. The actions and features that are available from the menu bar are available regardless of which part of Workflow Console you are working with. If an action or feature is available only for a particular part of Workflow Console, then that action or feature is instead available via menus and toolbars that are located with that part of Workflow Console.
- 3 **Toolbar:** Use the toolbar to create, delete, and copy category-related items; sort items; manage columns; open and display selected objects; and select different types of viewing formats. The toolbar options change depending on the category view.
- 4 **List area:** Use lists to select an item with which to work. The list area displays items that relate to the current category. Depending on user permissions, the list area displays the process definitions, instances, or activities with which to work. Select an item in the list to work with it. To customize the display, you can sort the items by column content, and add, remove, and reposition columns as desired.
- 5 **Filter bar:** Use the filter bar to specify the types of items that you want to display in the list area.
- 6 **Property panes:** View information or modify details about selected items. The panes that are displayed change depending on the current category. For example, the Comments pane is available only in the Activities and Instances category views.
- 7 **Object bar:** Use the object bar to switch between category and details views, display thumbnail images of open objects, and manage the layout within a workspace. For more information, see [“Working with Objects” on page 96](#).




Although window controls change depending on the workspace category, some controls are common to all categories. [Table 7.1](#) describes the common window controls.

Table 7.1 Window Controls

Control	Name	Description
	Collapse	Hides the object with which it is associated.
	Expand	Shows the object with which it is associated.
	Resize Vertical	Vertically resizes a pane when you drag the pointer () up and down.
	Resize Horizontal	Horizontally resizes a pane when you drag the pointer () to the left and right.

Navigation Pane

The following table shows the category views that are available in the navigation pane of Workflow Console.

Category View Button	Description
	The Process Definitions category view contains a list of the process definitions that are available to SAS Model Manager. To view the process definition details, double-click a process definition in the list.
	The Instances category view contains a list of the workflow instances. To view detailed information for an instance, double-click an instance in the list. You can view the properties and participants that are associated with an activity by selecting an activity.
	The Activities category view contains the activities that you can participate in and that have a state of Started . To view all of the properties for an activity, double-click an activity in the list.


Customizing Category Views

Overview of Customizing Category Views

You can customize the category views in the following ways:

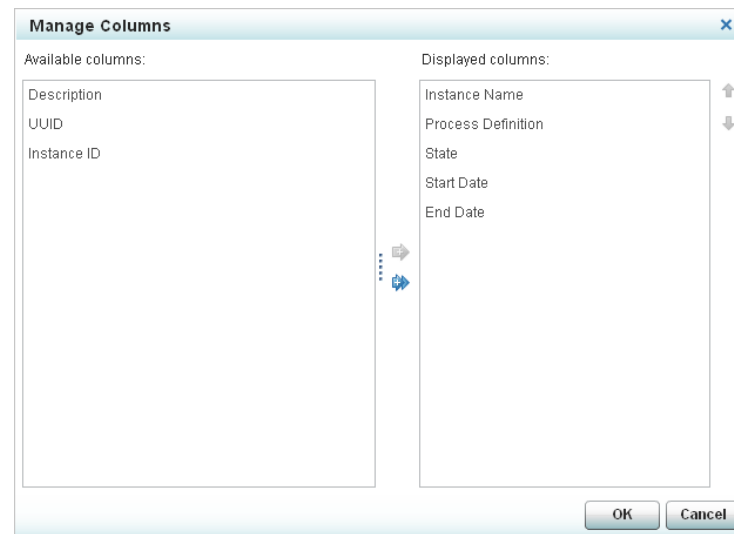
- manage which columns are displayed, and determine the order of the columns in a list
- set the sort order and direction of the attributes
- create, save, and manage filters

Manage Columns in a List

To add, remove, or reposition columns in a list, click the **Manage Columns** button  on the toolbar.

The **Manage Columns** button displays the Manage Columns window, which reflects the current settings of the list. Here is an example of the Manage Columns window.

Figure 7.1 Manage Columns Window









To add, remove, and reposition columns, use the control buttons that are described below.

Note: Changes that you make affect only the category views on your computer.

TIP You can select a set of noncontiguous columns in the list and then add, remove, or reposition the selected set as a single entity.

Table 7.2 Manage Columns Window Buttons

Button	Description
	Adds one or more selected columns to the list.
	Adds all available columns to the list.
	Removes one or more selected columns from the list.
	Removes all columns from the list.
	Moves one or more selected columns to the left in the list.
	Moves one or more selected columns to the right in the list.

Note: The workspace filter is applied to all columns in the view, even to columns that you remove from the **Displayed columns** list as long as they are still in the **Available columns** list. For example, if you remove the **Name** column from a list, the filter still searches for name values and returns results based on the contents of that column. For more information about using filters, see “[Filtering List Content](#)”.

Sort Lists by Column Content


Sort Single Columns in a List

When you display a list of items in a category view, click a column label to sort rows in ascending order, based on the values in the selected column. Clicking the column heading toggles between ascending and descending order.

To indicate that a column is sorted, a sort icon appears next to the column label and shows whether the sort is in ascending (▲) or descending (▼) order. The sort icon does not appear next to the label until that column is sorted.

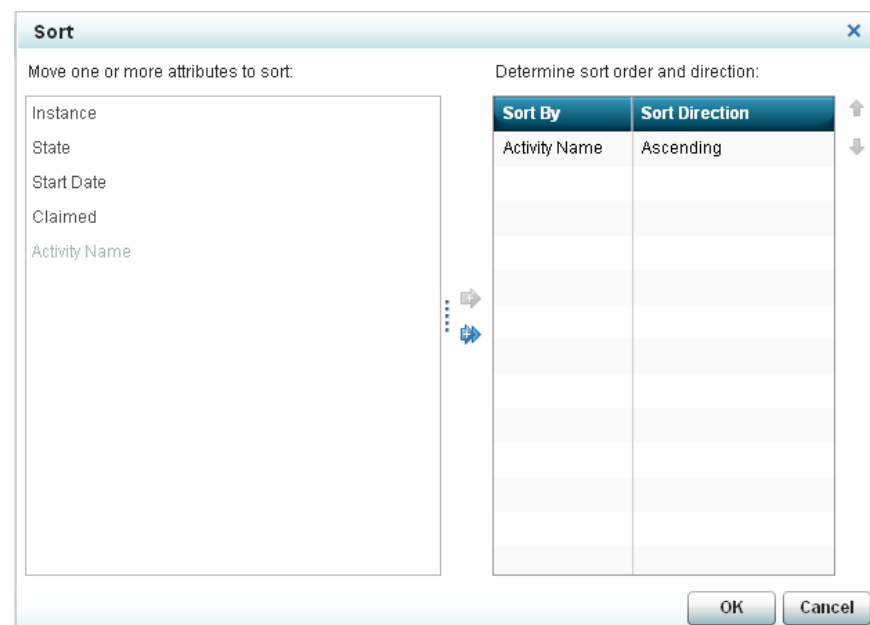
Sort Multiple Columns in a List

You can also sort a list by more than one column and, if desired, in ascending order by some columns and in descending order by other columns.

To sort a list by more than one column, click the **Sort** button  on the toolbar.







The **Sort** button displays the Sort window, which reflects the current sort order of the list. Here is an example of the Sort window.

Figure 7.2 Sort Window



To add, remove, or reposition columns in the sort order, use the control buttons that are described below.

Table 7.3 Sort Window Control Buttons

Control	Description
	Adds selected columns to the sort order.
	Adds all columns to the sort order.
	Removes selected columns from the sort order.
	Removes all columns from the sort order.
	Moves one or more selected columns higher in the sort order. <i>Note:</i> The first column that is listed in the sort order becomes the primary sort column.
	Moves one or more selected columns lower in the sort order.

TIP You can select a set of noncontiguous columns in the list and then add or remove the set as a single entity.

To specify the direction in which to sort a particular column, click in the **Sort Direction** cell, and then select **Ascending** or **Descending**.

Filtering List Content

About Filtering List Content

You can filter the list in a category view to view only particular process definitions, instances, or activities. For example, suppose you want to see the entry for a project and you use the UUID to identify the project.

To limit the number of items that appear in a list, you can use the filter bar to perform the following tasks:

- Create and save filter criteria for reuse
- Modify a saved filter
- Apply a saved filter
- Manage filters

Here are the different parts of the filter bar.

Figure 7.3 Filter Bar


- 1 **Filter menu:** Use the filter menu to save and manage filters. The menu expands to display the first ten saved filters, based on the order that is specified in the Manage Filters window.
- 2 **Search criteria:** Use the **Enter filter text** box to specify search criteria.

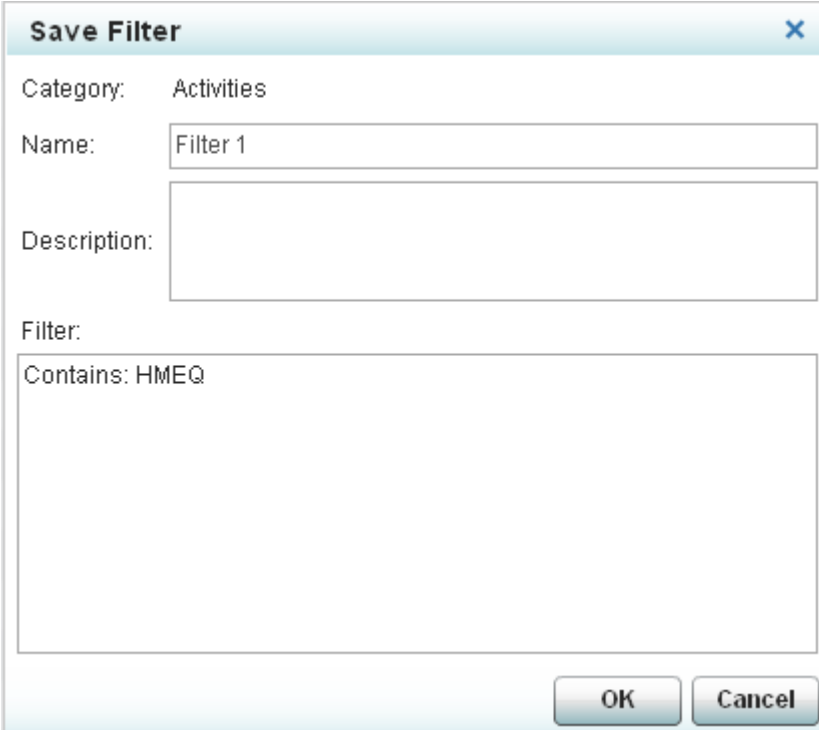
Create and Save Filter Criteria for Reuse

You can save filter criteria for reuse. Saved filters are available only for the category view in which the filter was saved.

To save filter criteria, follow these steps:

1. On the filter bar, specify filter criteria in the **Enter filter text** area. You can specify the criteria in any of the following formats:
 - alphanumeric combinations
 - alphabetic characters only
 - numbers only
 - special characters
 - compound words
 - mixed case

Note: Filter criteria are not case-sensitive.
2. Click . The Save Filter window appears.



The image shows a 'Save Filter' dialog box with a title bar containing a close button (X). The dialog has four main sections: 'Category:' with the value 'Activities'; 'Name:' with a text box containing 'Filter 1'; 'Description:' with an empty text box; and 'Filter:' with a text box containing 'Contains: HMEQ'. At the bottom right, there are two buttons: 'OK' and 'Cancel'.

3. In the Save Filter window, specify a name for the filter and an optional description. Click **OK**.

Note: The filter name can be modified in the Manage Filters window. The rule and description cannot be modified for an existing filter. If you want to change the rule or description, you must create a filter that has the same name to replace the existing filter.

4. In the confirmation window, click **OK** to replace the existing filter.

Filter Information Area

When you filter items in a list, the filter information appears in the area beneath the toolbar. The information shows the filter criteria that are in effect and provides a tally of how many items are filtered from the total number of items.

Here is a sample of the filter information area. In this sample, two activities out of three contain the instance name HMEQ.


Figure 7.4 Filter Information Area



Modify a Saved Filter

Although you cannot modify an existing filter, you can create a new filter and replace the existing filter with the new one.

To create a new filter and replace the existing one, follow these steps:

1. In the **Enter filter text** area on the filter bar, enter the filter text that you want to include in the new filter.
2. Click the down arrow in the filter menu  and select **Save Filter**.
3. In the Save Filter window, enter the same name as the filter that you want to replace.
4. (Optional) Enter a description.
5. Click **OK**.
6. In the confirmation window, click **OK** to replace the existing filter.

Apply a Saved Filter

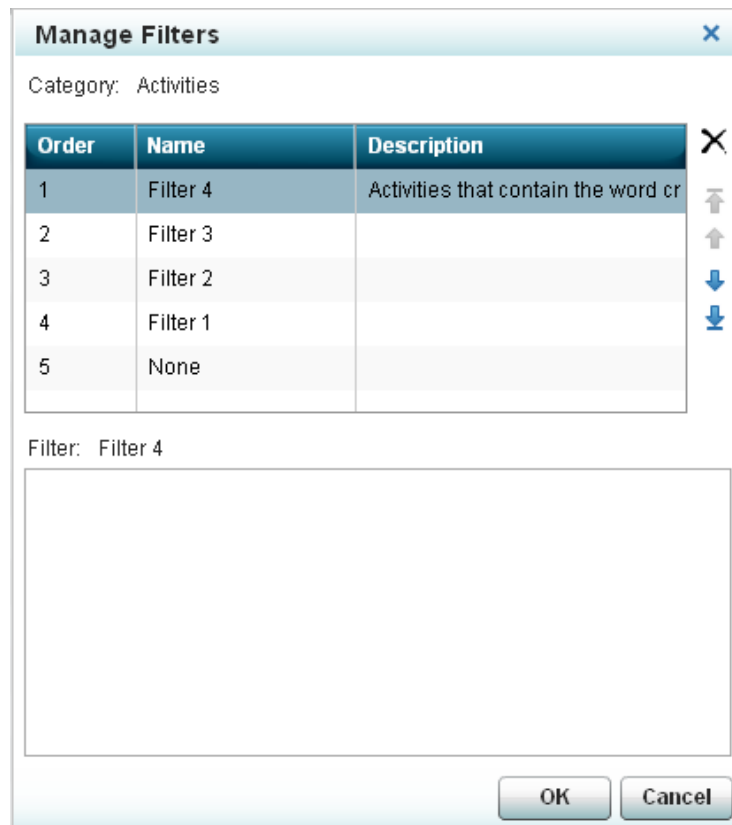
To apply a saved filter, click the filter menu down arrow, and select the filter from the list. If there are more than ten saved filters, follow these steps:

1. Click **More** on the filter menu to display the Select Filter window.
2. Select the filter that you want to use, and click **OK** to apply the filter criteria to the current list.

Manage Filters

You can rename, reorder, and remove saved filters as necessary by using the Manage Filters window.

1. Click the down arrow in the filter menu  and select **Manage Filters**. The Manage Filters window appears.



2. To rename a saved filter, click in the **Name** column and enter a new name. The change is saved when you navigate away from the column or press ENTER.
3. The **Order** column indicates the order in which saved filters appear in the filter menu. To reorder the list, use the move down (↓ and ⇓) and move up (↑ and ⇑) buttons.
4. To remove a filter, select the filter and click ✕.
5. Click **OK**.

Setting Preferences

About Setting Preferences

Preferences provide a way for you to customize the user interface of Workflow Console. To set preferences, select **File** ⇒ **Preferences** from the menu bar. The following topics describe the available preferences and explain how to use the Preferences window. Additional information is also provided for how to use the SAS Preferences Manager to specify individual preferences to override the default notification type and user locale.

Global Preferences

Global preferences are applied to all SAS Web applications that are displayed with the Adobe Flash player. When you set a global preference, it applies to the user who you are logged on as and to no other users.

Note: If the user locale that you specify in the preferences is different from the user locale for the SAS Workspace Server, you might receive an error when you try to log on to the SAS Model Manager client application. You might also receive encoding errors when executing tasks in SAS Model Manager.

To set global preferences, select the **Global Preferences** page. These global preferences are available:

- **User locale** specifies the geographic region whose language and conventions are used in the applications. This setting might also apply to some SAS Web applications that are not displayed with the Adobe Flash player.
- **Theme** specifies the collection of colors, graphics, fonts, and effects that appear in the applications. Theme changes take effect after you log off and log back on.

Note: You can also set the **User locale** setting by using the SAS Preferences Manager. Enter in your browser window the URL **http://host-name:port/SASPreferences** to launch the SAS Preferences Manager, and select the **Regional** menu option. For more information, see “SAS Preferences Manager” in Chapter 5 of *SAS Intelligence Platform: Middle-Tier Administration Guide*.

General Preferences

General preferences are settings that are applied across Workflow Console only. To set general preferences, select the **General** page. These general settings are available:


- **Open application using this workspace** specifies the workspace that is displayed when logging on to this application.
- **Show this number of recent items** specifies how many objects to display in the **Recent Objects** tile.

Alert Notifications


Use SAS Preferences Manager to override the default notification delivery type for an individual user. The SAS Preferences Manager is a Web application that provides a central facility for users to manage their preferences and settings. The SAS default delivery type for notifications is specified in the properties for the SAS Application Infrastructure using the Configuration Manager plug-in to SAS Management Console. The default type after the deployment of SAS 9.3 is **My alerts portlet**. However, a SAS administrator can modify the default notification type. For more information about modifying the SAS default notification type for all users, see “Configuring Alert Notifications for SAS Workflow” in Chapter 3 of *SAS Model Manager: Administrator's Guide*.

To specify the notification delivery preference for an individual user, follow these steps:

1. Enter the URL **http://host-name:port/SASPreferences** in your browser window to launch the SAS Preferences Manager. Replace the values for host-name and port based on the location of the configured SAS Web Infrastructure Platform.
2. Enter the user ID and password for the user account that you use to access SAS Web applications.
3. Select **General** ⇒ **Notifications**.
4. Select a format type for the E-mail notifications. The options are **HTML-formatted e-mail** and **Plain-text e-mail**.

5. Select the notification types from the **Available** list and click  to add the selected notification types. The available options are the following:

- Via e-mail
- My alerts portlet
- Via SMS text message
- Via digested e-mail

TIP To remove a notification type, select the type from the list and click  to remove the selected item.

6. Click **Apply** to update the notification settings and click **OK** to save the changes.

For more information, see “SAS Preferences Manager” in Chapter 5 of *SAS Intelligence Platform: Middle-Tier Administration Guide*.

Working with Objects

When you open objects, you can add them to the object bar as minimized items or open them and work with them in the layout. A layout is simply a collection of one or more open objects in a specific order or arrangement in the object view.

About the Object Bar



The object bar provides quick access to open objects and to tools that control how objects are displayed. By using the object bar, you can perform the following activities:


- switch between the category view and one or more open objects
- view thumbnail images of open objects
- tile open object views
- open, minimize, and close objects and object groups

Note: The object bar is workspace-specific. When you switch to a different workspace, the object bar changes to reflect the open objects in that workspace.

Here is a description of the buttons on the object bar.

Table 7.4 Object Bar Buttons

Button	Name	Description
	Switch to Category View	Displays the category view.
	View Objects	<p>Tiles the display of open objects.</p> <p>From the drop-down list, select the objects to tile and select Show Details to display object details in the thumbnail images.</p>

Button	Name	Description
	Manage Objects	<p>Displays the Manage Objects menu that contains the following options:</p> <ul style="list-style-type: none"> • Show All Objects • Add Minimized Objects • Add Group to Layout • Minimize All • Minimize Group • Close All • Close Displayed • Close Minimized • Close Groups • Save Layout • Open Layout


Open Objects

To open an object in a category view:

1. Double-click the object, or select **Open** on the toolbar and then select one of the following options:
 - **Open:** Opens the object in the object view and adds the object's thumbnail image to the object bar. This option enables you to work with the object immediately.
 - **Open Minimized:** Opens the object's thumbnail image on the object bar. This enables you to continue selecting other objects in the object view.

TIP You can drag an object from a list to the object bar.

- **Add:** Adds a selected object to the object bar (when one or more objects are open).
2. (Optional) To switch to the object view, click the **View Objects** button on the object bar.

TIP To obtain a list of open objects, click  beneath the **View Objects** button. In this list, a check mark indicates that the object is open. You can select an object to open that object in the object view, or clear the check mark to return the object to a minimized (thumbnail) state on the object bar.

Open Multiple Objects

To open multiple objects, use either of the following methods:

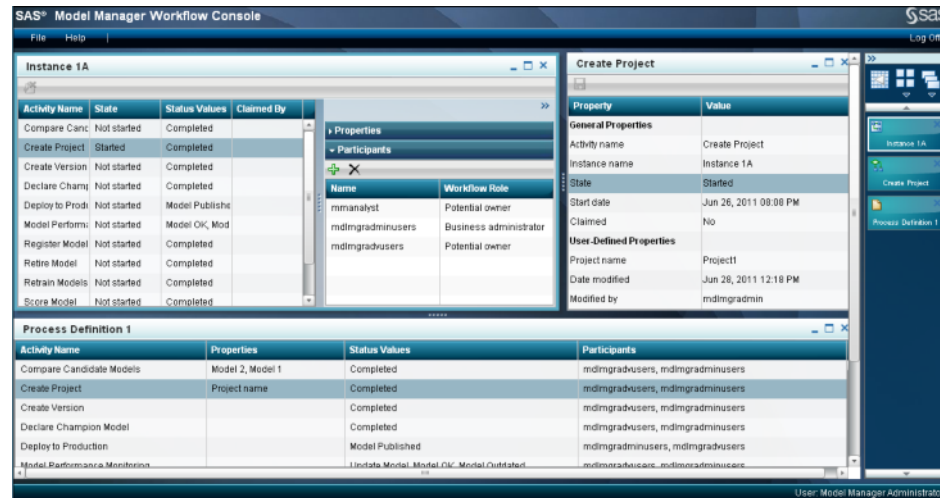
- Use the SHIFT key to select multiple adjacent objects or use the CTRL key to select noncontiguous objects. The objects are tiled within the view and object thumbnail images appear on the object bar.
- Select an object and click **Open**, click the **Switch to Category View** button, select another object, and then click **Open**. The second object opens in the workspace,

replacing the first item, and thumbnail images for both objects appear on the object bar. Repeat this process to continue opening objects.

TIP To display all open objects, rather than the most recently opened object, click the **Manage Objects** button on the object bar, and select **Show All Objects**.

Here is an example of how three open objects appear when they are tiled within Workflow Console and how the relevant thumbnail images appear on the object bar.

Figure 7.5 Tiled Open Objects



Rearrange Open Objects

To change the way open objects are displayed, you can do any of the following:

- Minimize, maximize, or close each object by using the window controls.
- Minimize objects by using **View Objects** on the object bar.

To minimize an object, select the object in the list. This action clears the check mark and removes the object from the object view. The thumbnail image is retained on the object bar for quick access when necessary.

- Open, minimize, or close objects by using **Manage Objects** on the object bar.
- Add objects to the object view by dragging their icons from the object bar to the object view.
- Resize windows by using the dividers between the item windows.
- Rearrange objects by clicking their title bars and dragging them to a new position in the object view.


Save Layouts

To save the current object view layout for reuse, click the **Manage Objects** button on the object bar, select **Save Layout**, and provide a name for the saved layout.


To open a saved layout, click the **Manage Objects** button on the object bar, select **Open Layout**, and select the layout that you want to apply to the object view.

Close Objects

You can close objects in the following ways:


- If the object is open in the object view, click  in the upper right corner of the thumbnail image. This closes the object in both the object view and on the object bar.

Note: To minimize an open object, click the minimize button in the upper right corner of the thumbnail image. This removes the object from the object view. The object thumbnail is visible on the object bar.

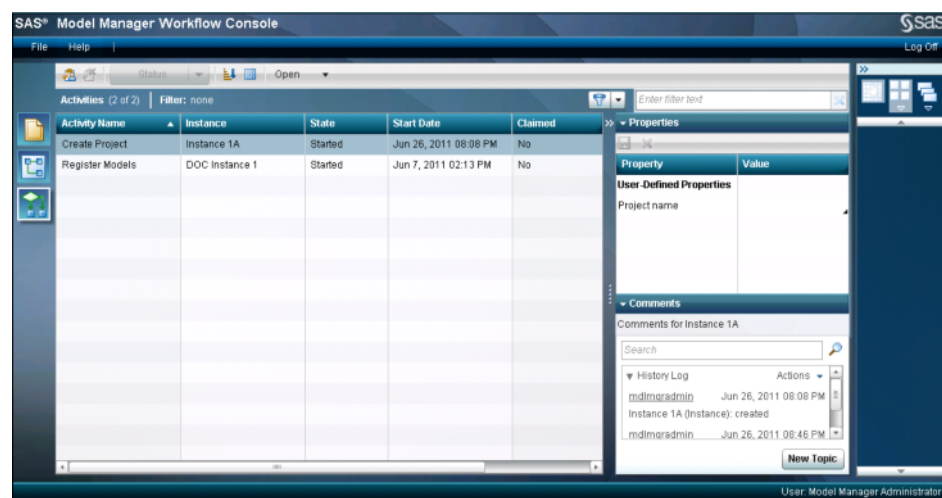
- If the object thumbnail image is visible on the object bar, click  in the upper right corner of the thumbnail image. This action removes the object thumbnail from the object bar.

After you close the object, you can open it again from the category view or open a previously saved layout that contains the object.

Viewing Workflow Activities

To view the activities in your workflow inbox from the SAS Model Manager window, select **Tools** ⇒ **My Workflow Inbox** or click .

Workflow Console is launched in a Web browser, and displays the Activities category view. The list displays only the activities for which you are the actual owner or are assigned as a potential owner, and that have the state of **Started**. If you are assigned as a participant with the workflow role of business administrator, you can see all of the activities that contain that workflow role assignment and that have the state of **Started**. After you claim an activity, you become the actual owner of the activity.





Working with Workflow Activities

The Activities category view of Workflow Console displays the activities that you have been assigned as potential owner or business administrator, and that have a state of **Started**. From the Activities category view, you can perform the following tasks:

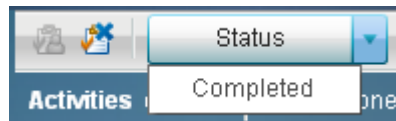
- claim an activity
- add comments to an activity
- change values of user-defined properties
- release an activity
- mark an activity completed
- view the history log for an activity

To complete an activity, follow these steps:

1. From the Activities category view, select an activity name, and click  to claim an activity.

Note: You can select an activity name and click  to release an activity that you had previously claimed. Only a SAS administrator or SAS Model Manager administrator can release an activity that has been claimed by another participant. For more information, see [“Releasing an Activity” on page 115](#).

2. (Optional) Enter a property value or change an existing property value in the Properties pane. For more information, see [“Editing Activity Properties” on page 100](#).
3. (Optional) Add a comment to the activity using the Comments pane. For more information, see [“Working with Comments” on page 101](#).
4. Select a status value to complete the activity. The workflow process continues to the next activity.



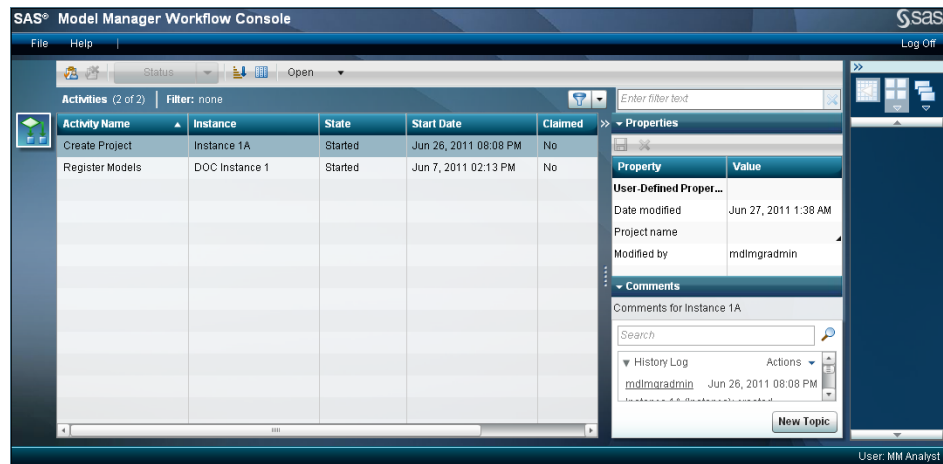
5. Repeat steps 1 through 4 until the workflow process has been completed.


Editing Activity Properties


An activity can contain user-defined properties. User-defined properties that are editable display a triangular icon in the bottom right corner of the property value in the data grid.

To edit the properties for an activity, follow these steps:

1. From the Activities category view, select an activity. The user-defined properties that are associated with the activity are displayed to the right in the Properties pane.



2. Click on the property value, and then enter a value or change the existing value.
3. To save the changes to the properties, click .

If you do not want to save the changes to the properties, click .

The activity properties can also be modified from the activity details view. Double-click an activity name to display the activity details view. Here is an example of a details view.



Working with Comments

About Comments

Use the Comments pane in the Instances and Activities category views to share information about an instance or activity with other users, or to make notes.

The Comments pane enables you to add new topics or reply to an existing topic that is associated with an instance or an activity. You can also search or filter the comments.

You can add multiple comments to an activity or instance. Each comment becomes its own topic, to which other users can reply. The original comment and its replies are a *topic thread*. Each entry in the topic thread includes the user ID of the person who created it, and the date and time when the comment or reply was last modified. Topic threads are separated by lines in the Comments pane.

The following table describes the items in the Comments pane. Before you add a comment, only the **New Topic** button is available.

Table 7.5 Plan Comments Pane Items

Item	Description
Search box	Enables you to display all topics that contain a specific text string.
Topic thread	Displays a comment and its replies, if any. Topic threads are separated by lines in the Comments pane.
Actions menu	Provides options for sorting and filtering the entries in a topic thread.
Reply button	Displays a topic window in which you can enter a response to a comment or add a comment to the History Log topic.
New Topic button	Displays the New Topic window in which you can create a new comment.

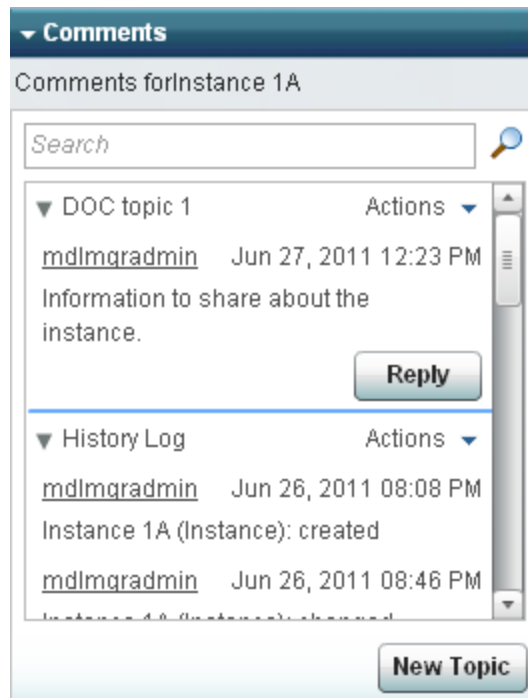
Adding Comments

To add comments to an instance or an activity, follow these steps:

1. Select the instance or activity.
2. In the Comments pane, click **New Topic**. The New Topic window appears.



3. Enter a topic name and a comment.
4. (Optional) Click **Attachment** to add a file with the new topic. For more information, see [“Attach a File” on page 103](#).
5. Click **Save**. The new comment now appears in the Comments pane.



Reply to a Comment

To reply to a comment, follow these steps:

1. In the Comments pane, expand the topic thread to which you want to add a reply.
2. Click **Reply**. A topic window appears. The title bar displays the name of the topic to which you are replying.
3. Enter your comment in the box.
4. (Optional) Click **Attachment** to add a file to the new topic. For more information, see [“Attach a File” on page 103](#).
5. Click **Save**. The reply is added below the original comment.


Attach a File

When you add a comment or reply, you can include an attachment. There are no restrictions on the types of files that you can attach, or on their size. However, including large attachments in a comment or reply can affect Workflow Console performance.

To add an attachment, follow these steps:

1. Click the **Attachment** button in the Comment or Reply window.
The **Select file(s) to upload** window appears.
2. Using the **Look in** field, navigate to the location of the file.
3. In the list, click the file that you want to attach. Its name appears in the **File Name** box.
4. Click **Open**.

The topic window appears. The filename and size appear below the comment box.

Note: Before you save a comment or reply, you can use the remove attachment button  to the right of the attachment name to remove the attachment. After you save a comment or reply, you cannot remove its attachments.

Sort and Filter Items in a Comment Topic Thread


Use the **Actions** menu to sort the comments and replies.

Table 7.6 Comments Pane Sort Options

Option	Description
Sort by Date Ascending	Displays the comment and replies in ascending order by their date and time.
Sort by Date Descending	Displays the comment and replies in descending order by their date and time.
Sort by User Ascending	Displays the comment and replies in ascending order by user ID.
Sort by User Descending	Displays the comment and replies in descending order by user ID.
Show Only My Replies	Displays only the replies that have your user ID.

Search for Comments and Replies

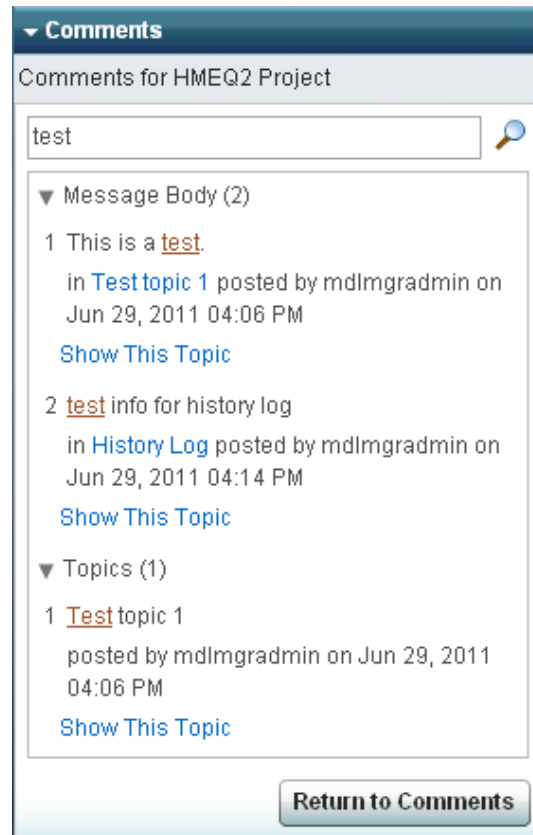
For a selected instance or activity, use the search box at the top of the Comments pane to find all the comments and replies that contain a specific text string.

To initiate a search, enter the text that you want to find in the search box and click .

The application applies the search to all the topic threads, and divides the results into two categories:

- **Message Body:** displays the topics that contain the search string in their comment text.
- **Topics:** displays the topics that include the search string in their titles.

As shown in the following example, the search returns all of the topics that contain the string “test” and underlines the relevant area of text in the display:



To clear the search and display all topics, click **Return to Comments**.

To display the full content of a topic, click **Show This Topic**. The application displays the original comment and all its replies.

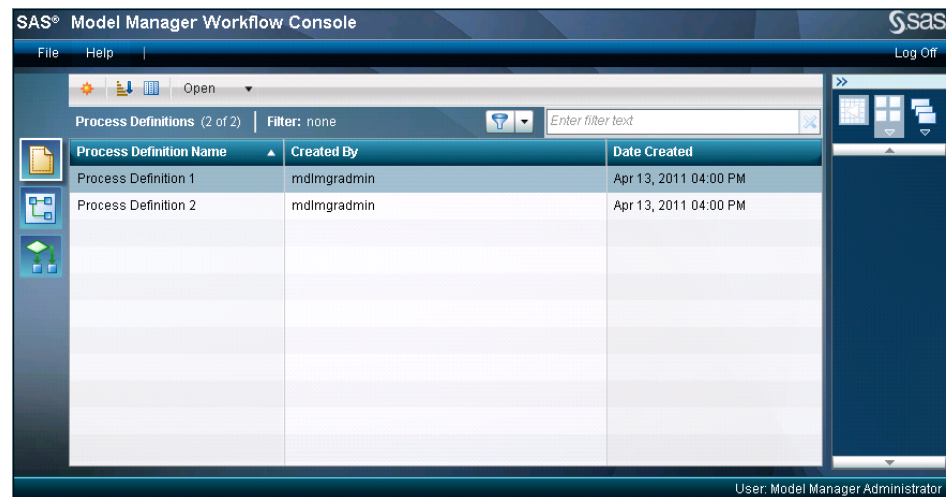
After you review a selected topic, do one of the following:

- To return to the list of topics that match the search criteria, click **Back to Search Results**.
- To clear the search criteria, click **Show All Topics**.

Managing the Workflow Process

SAS Model Manager Workflow Console can be used to manage instances of workflow process definitions. A SAS Model Manager administrator can create new workflow instances, view workflow process definitions, and interact with activities that are associated with a workflow instance. If the SAS Model Manager administrator is assigned to the workflow role of business administrator, the administrator can influence the progress of an activity by actions such as assigning an activity, or releasing the activity that is claimed by another user.

To manage the workflow process, from the SAS Model Manager window select **Tools** ⇒ **Manage Workflow**. Workflow Console is launched in a Web browser and displays the Process Definitions category view, as shown.



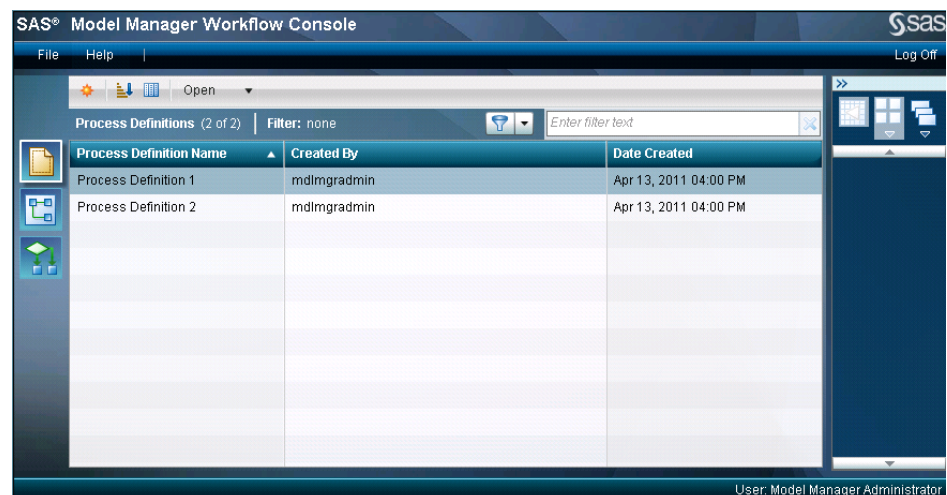
See Also

- “Creating a New Workflow Instance” on page 107
- “Viewing Workflow Instances” on page 109
- *SAS Model Manager: Administrator's Guide*

Viewing Workflow Process Definitions

Only a SAS Model Manager administrator can view process definitions. Process definitions are created and made available to SAS Model Manager using SAS Workflow Studio. A SAS Model Manager administrator can use Workflow Console to create instances of process definitions. You use instances of process definitions to manage the process of model development and implementation in SAS Model Manager.

To view the available process definitions, from the SAS Model Manager window select **Tools** ⇒ **Manage Workflow**. Workflow Console is launched in a Web browser and displays the Process Definitions category view.



To view detailed information for a process definition, double-click the process definition. A process definition contains the activities that define a workflow process. The activities contain properties, status values, and participants that are associated with the activity.



The screenshot shows the SAS Model Manager Workflow Console interface. It features a table titled 'Process Definition 1' with four columns: Activity Name, Properties, Status Values, and Participants. The table lists various activities such as 'Compare Candidate Models', 'Create Project', 'Create Version', 'Declare Champion Model', 'Deploy to Production', 'Model Performance Monitoring', 'Register Models', 'Retire Model', 'Retrain Models', 'Score Model', and 'Validate Score Code'. Each activity has associated properties, status values, and participants. The interface also includes a menu bar with 'File' and 'Help', a 'Log Off' button, and a sidebar with a 'Process Definition 1' icon. The user is identified as 'Model Manager Administrator'.

Activity Name	Properties	Status Values	Participants
Compare Candidate Models	Model 2, Model 1	Completed	mdlmggradusers, mdlmggradminusers
Create Project	Project name	Completed	mdlmggradminusers, mdlmggradvusers
Create Version		Completed	mdlmggradminusers, mdlmggradvusers
Declare Champion Model		Completed	mdlmggradusers, mdlmggradminusers
Deploy to Production		Model Published	mdlmggradusers, mdlmggradminusers
Model Performance Monitoring		Model OK, Model Outdated, Update Model	mdlmggradusers, mdlmggradminusers
Register Models		Completed	mdlmggradminusers, mdlmggradvusers
Retire Model		Completed	mdlmggradusers, mdlmggradminusers
Retrain Models		Completed	mdlmggradusers, mdlmggradminusers
Score Model		Completed	mdlmggradusers, mdlmggradminusers
Validate Score Code		Completed	mdlmggradminusers, mdlmggradvusers


For more information about process definitions, see [“Managing the Workflow Process” on page 105](#) and *SAS Model Manager: Administrator's Guide*.

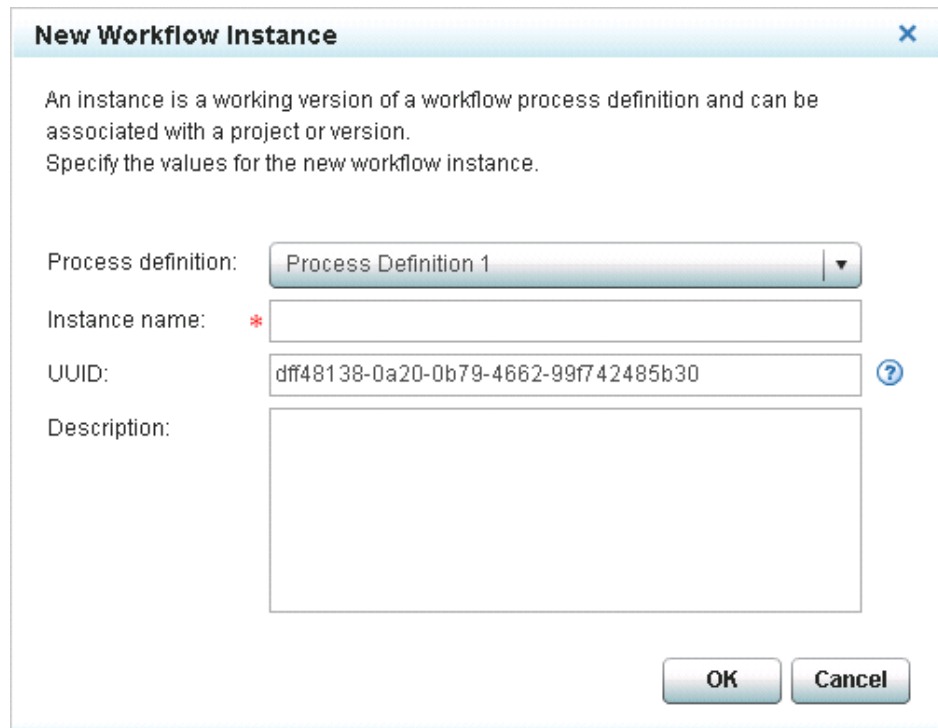
Creating a New Workflow Instance

An instance is a working version of a workflow process definition. Only a SAS Model Manager administrator can create a new workflow instance.

To create a new workflow instance, follow these steps:

1. From the SAS Model Manager window, right-click a project or version and select **New Workflow Instance**. Workflow Console is launched in a Web browser and displays the New Workflow Instance window.

Note: If you are already logged on to Workflow Console, from the Process Definitions category view, select a process definition and click .



New Workflow Instance [X]

An instance is a working version of a workflow process definition and can be associated with a project or version.
Specify the values for the new workflow instance.

Process definition:

Instance name: *

UUID: [?]

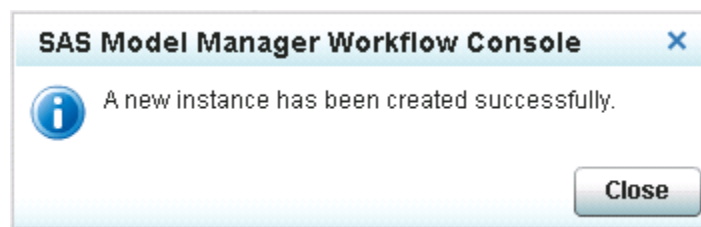
Description:


[OK] [Cancel]

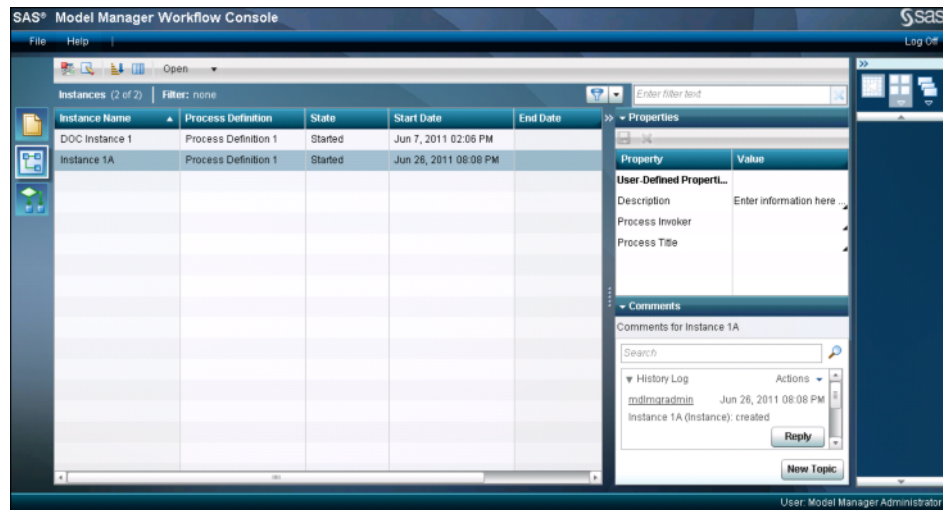
2. Enter a name for the instance.
3. The UUID of the selected project or version is already populated.

Note: If the UUID is not already populated, you can copy the UUID property value for a project or version from the Properties view in the SAS Model Manager window. The field label and other characters that precede the UUID value must be removed.

4. (Optional) Enter a description for the instance.
5. Click **OK**. A message appears, indicating that the instance has been successfully created.




6. Click **Close**. The new workflow instance is now available in the Instances category view.
7. To view the new instance, click . Select the instance to view information that is associated with the new instance.

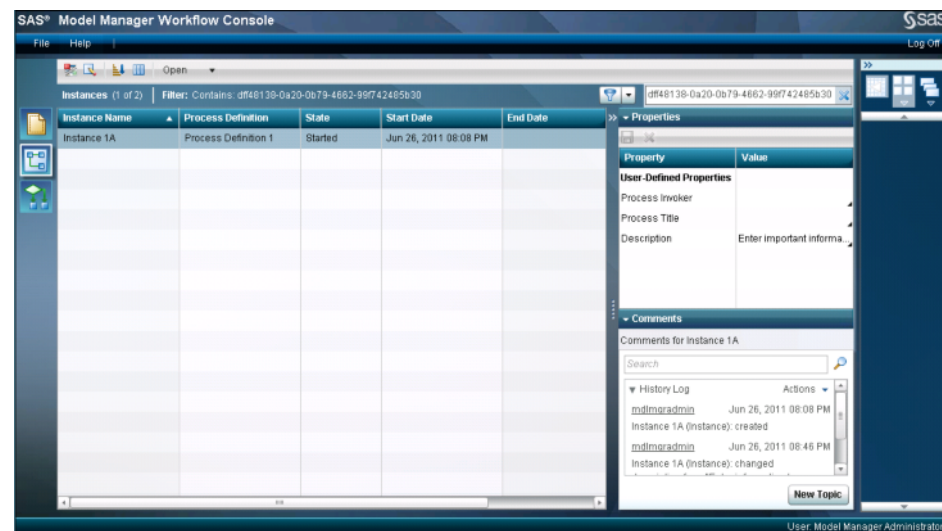


For more information about managing process definitions, see the *SAS Model Manager: Administrator's Guide*.

Viewing Workflow Instances

Only a SAS Model Manager administrator can view instances that are associated with a project or version. To view the workflow instances that are associated with a project or version, from the SAS Model Manager window right-click a project or version and select **View Workflow** or click .

Workflow Console is launched in a Web browser, and displays the Instances category view with the value for the UUID as the applied filter. Only the workflow instances that are associated with the selected project or version are displayed in the list. You can select an instance from the list to view the properties and comments that are associated with the instance.



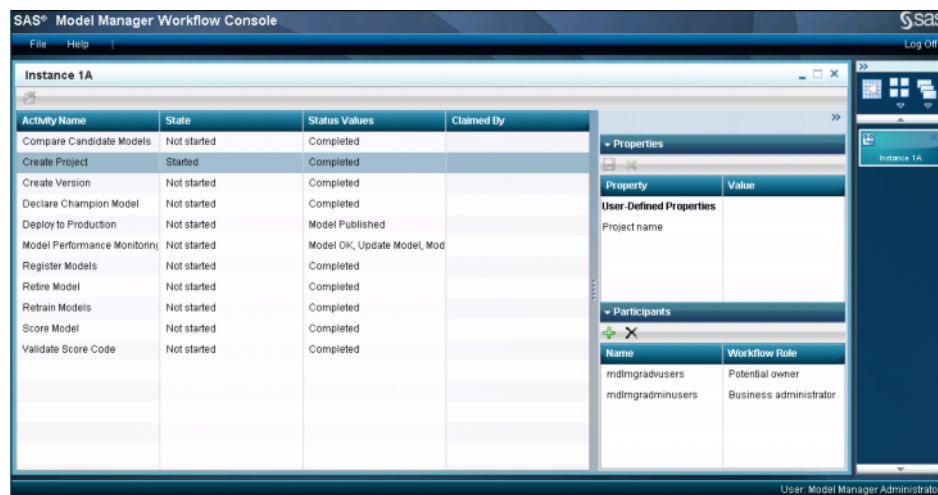
Note: If a workflow instance is not associated with the selected project or version, the following message appears after Workflow Console is launched.



From the instances category view, you can perform the following tasks:

- [edit a workflow instance](#)
- [edit instance properties](#)
- [add comments](#)
- [terminate a workflow instance](#)

To view detailed information for an instance, double-click an instance name. You can then view the properties and participants that are associated with an activity by selecting an activity name.

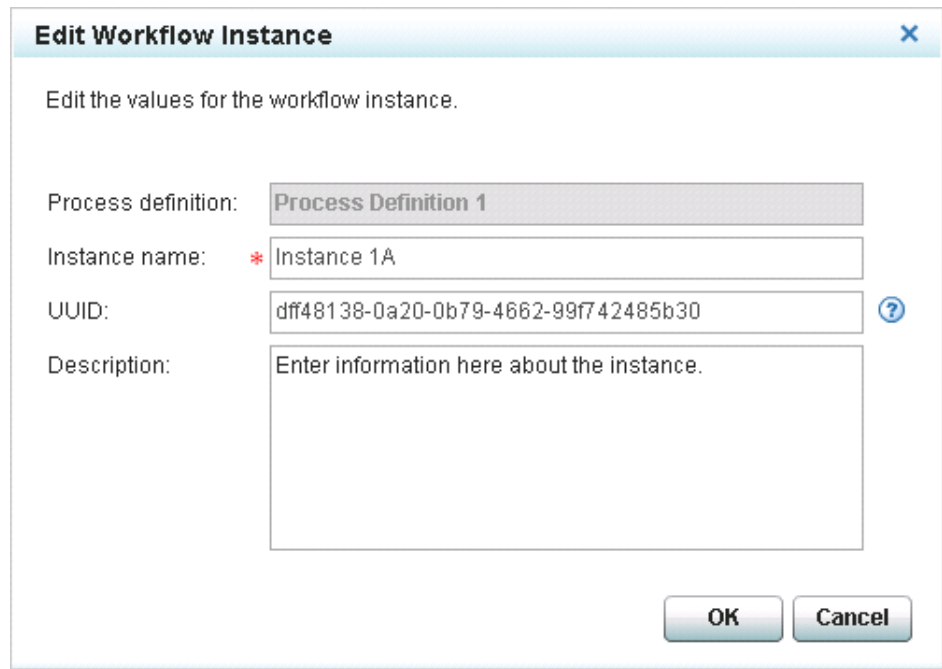


For more information, see [“Working with Workflow Participants”](#) on page 112.

Editing a Workflow Instance

To edit a workflow instance, follow these steps:

1. From the SAS Model Manager window, right-click on a project or version and select **View Workflow**. Workflow Console appears.
2. From the Instances category view, select an instance.
3. Click . The Edit Workflow Instance window appears.



Edit Workflow Instance [X]

Edit the values for the workflow instance.

Process definition: **Process Definition 1**

Instance name: *

UUID: [?]

Description:

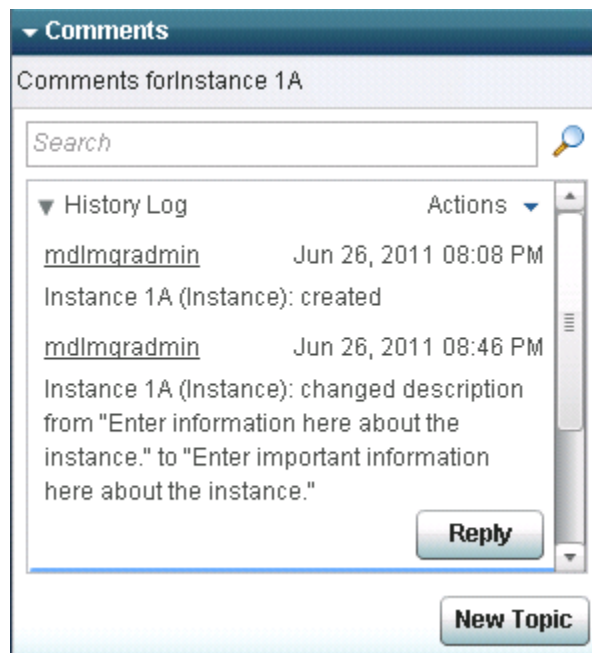
[OK] [Cancel]

4. Make changes to the instance name, UUID, or description.

Note: To associate the instance with a project or version, enter the UUID for a specific project or version.

5. Click **OK**.

A record of the changes is added to the **History Log** in the Comments pane.



Comments

Comments for Instance 1A

[Search Icon]

▼ History Log Actions ▼

mdlmgadmin	Jun 26, 2011 08:08 PM
Instance 1A (Instance): created	
mdlmgadmin	Jun 26, 2011 08:46 PM
Instance 1A (Instance): changed description from "Enter information here about the instance." to "Enter important information here about the instance."	

[Reply]

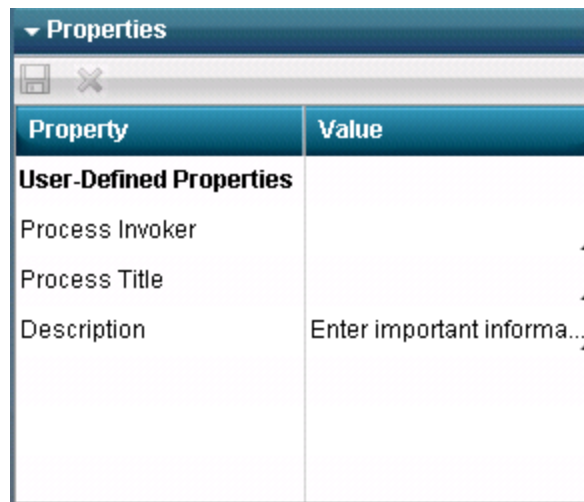
[New Topic]


Editing Instance Properties


A process definition can contain user-defined properties for instances and for activities. User-defined properties that are editable display a triangular icon in the bottom right corner of the property value field in the data grid.

To edit the properties for an instance, follow these steps:

1. From the Instances category view, select an instance. The user-defined properties that are associated with the instance are displayed in the Properties pane.



2. Click on the property value, and enter a value or change the existing value.
3. To save the changes to the properties, click .

If you do not want to save the changes to the properties, click .

For more information about editing activity properties, see [“Editing Activity Properties” on page 100](#).

Working with Workflow Participants

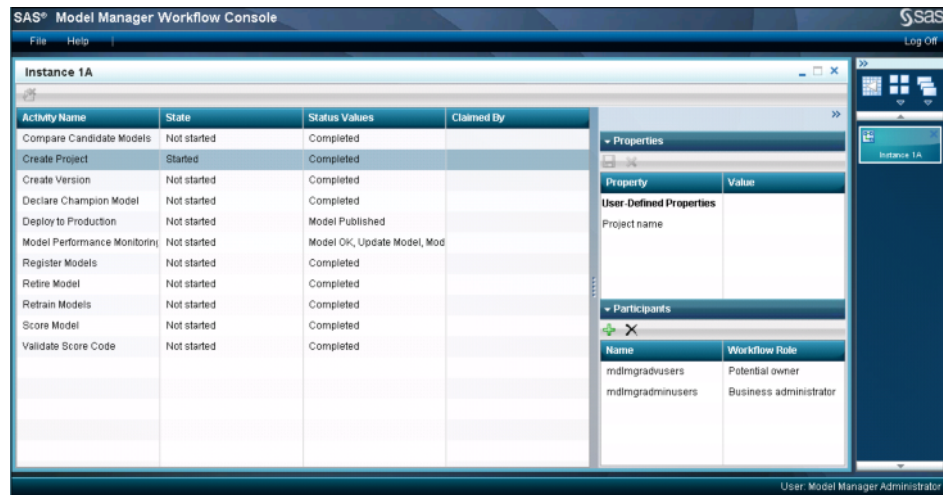
From the Instance details view you can view the properties and participants that are associated with an activity by selecting an activity. If you are a SAS Model Manager administrator and you are associated with the workflow role of business administrator, you can assign or remove participants, and release activities that have been claimed by another user.


Assigning Participants to Activities

Default participants might have been assigned already to activities when a process definition was created.

To assign an additional participant to an activity, follow these steps:

1. From the Instances category view, double-click an instance. The Instance details view appears.




2. Select an activity, and then click  in the Participants pane. The Assign a Participant window appears.


Assign a Participant

A participant can be an individual user, user group, or user role that is assigned to a workflow role.
Specify the values for the participant that you want to assign to this activity.

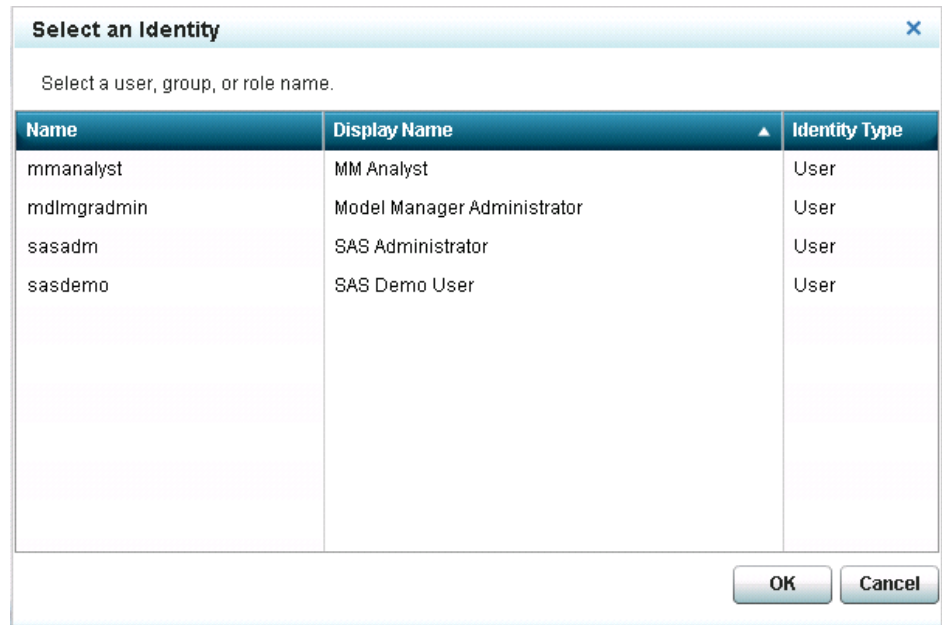
Identity type:

Name: * 

Workflow role: 

3. Select one of the identity types: user, group, or role.
4. Enter part of the user, group, or role name, and click 

Note: If you do not enter part of the name, all of the names for the selected identity type are displayed. In addition, if you manually enter a name value and do not click the search button, the name is not verified against the SAS identity participant list.



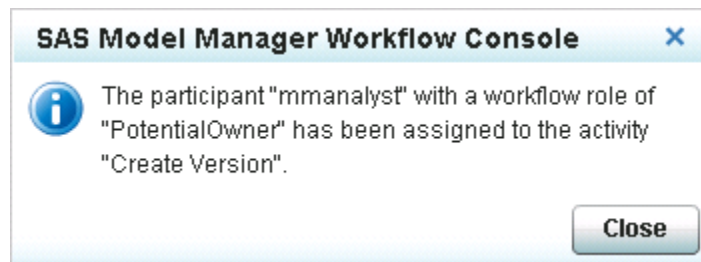
Select a name and click **OK**.

5. Select a workflow role for the participant.

Here are the workflow roles that you can assign to participants for a workflow activity:

- **Business administrator:** a participant who can influence the progress of an activity by actions such as adding comments, assigning an activity, or releasing the activity claimed by another user.
- **Potential owner:** a participant who can claim an activity in a workflow process and who becomes the actual owner of an activity

Click **OK**. A message appears, indicating whether the participant was successfully assigned to the activity.



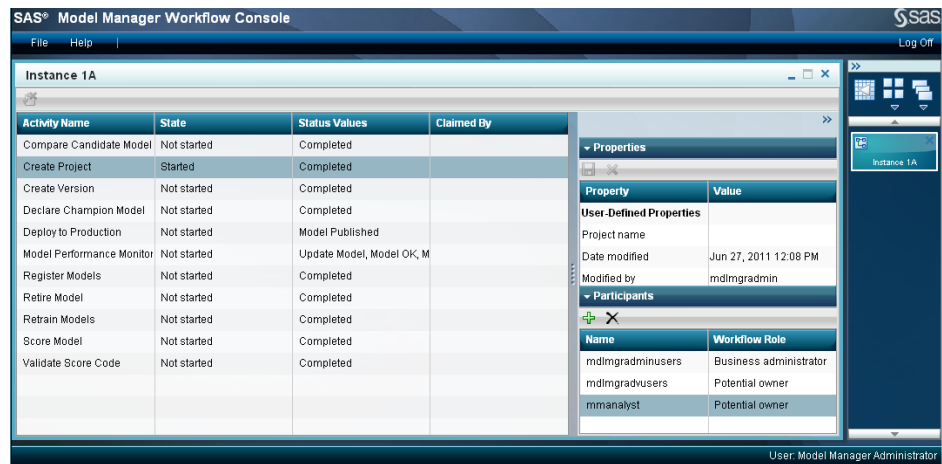
6. Click **Close**.

Removing Participants from Activities

To remove a participant from an activity, follow these steps:

1. From the Instances category view, double-click an instance name.
2. Select an activity, and then select a participant from the Participants pane.

Note: You cannot remove a participant who is associated with the workflow roles of business administrator or actual owner.



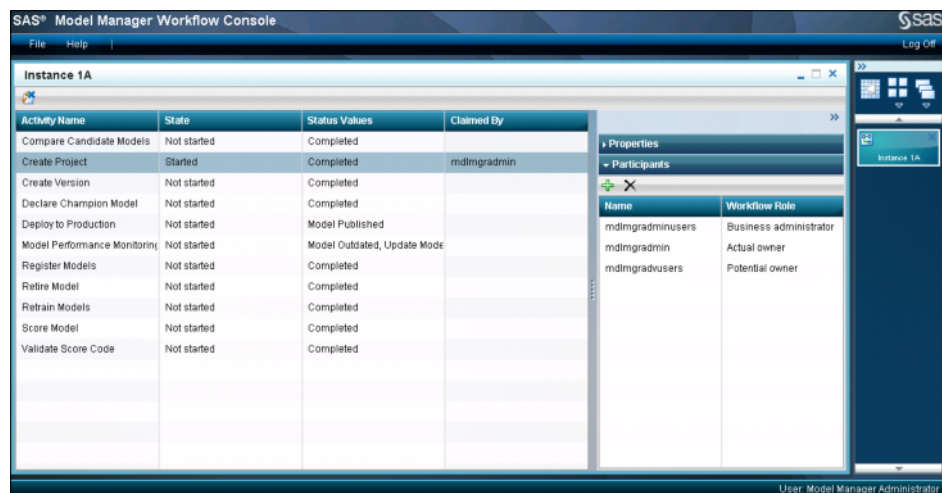
3. Click **X**. A message is displayed asking if you are sure that you want to remove the participant from the activity.
4. Click **Yes**. A message is displayed indicating that the participant was successfully removed from the activity.
5. Click **Close**.

Releasing an Activity

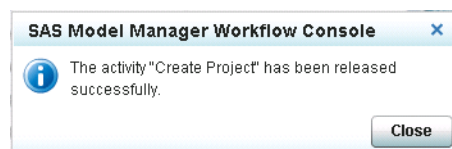
A SAS Model Manager administrator can release an activity that has been claimed by a workflow participant. The name of the actual owner is displayed in the Participants pane.

To release an activity, follow these steps:

1. From the Instances category view, double-click an instance name. The Instance details view is displayed.



2. Select an activity name, and click . A message is displayed indicating that the activity was successfully released.




3. Click **Close**.

Terminating a Workflow Instance

When you terminate a workflow instance, all activities that have not yet been completed in the workflow process are changed to a state of **Terminated**. After you terminate a workflow process for an instance, the process cannot be restarted.

To terminate a workflow process, follow these steps:

1. From the Instances category view, select an instance name and click .
2. Click **Yes** to terminate the workflow process for the selected instance.
3. Click **Close** to return to the Instances category view.

Part 3

Importing, Scoring, and Validating Models

<i>Chapter 8</i>	
Importing Models	119
<i>Chapter 9</i>	
Scoring Models	149
<i>Chapter 10</i>	
Validating Models Using Comparison Reports	167
<i>Chapter 11</i>	
Validating Models Using User Reports	177

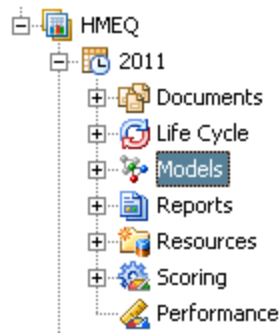
Chapter 8

Importing Models

Overview of Importing Models	119
Import Models from the SAS Metadata Repository	120
Import Package Files from SAS Enterprise Miner	122
What Is a SAS Enterprise Miner Package File?	122
Create Package Files	122
Import Package Files	122
Import SAS Code Models and R Models Using Local Files	123
Overview of Importing SAS Code Models	123
Model Templates	125
Model Template Component Files	127
Viewing Model Template Files	133
Importing a SAS Code Model	133
Importing an R Model	134
Import PMML Models	135
Import Partial Models	136
Set Model Properties	137
Map Model Variables to Project Variables	138
User-Defined Model Templates	139
Creating a New Model Template	139
Model Template Properties	143
Specific Properties for a Model	145

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 Documents, Life Cycle, 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 version's **Models** folder.

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

- “Import Models from the SAS Metadata Repository” on page 120
- “Import Package Files from SAS Enterprise Miner” on page 122
- “Import SAS Code Models and R Models Using Local Files” on page 123
- “Import Partial Models” on page 136
- “Import PMML Models” on page 135

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 [Appendix 2, “SAS Model Manager Access Macros,”](#) on page 311.

Note:

- Scorecard models can be imported using the SAS Code Models and SAS Enterprise Miner Package File import methods.
- SAS Model Manager cannot export or publish models whose **Score Code Type** model property is set to **SAS Program**.
- SAS Model Manager does not support tables, variables, or models that have special characters or double-byte character sets, and that were created when the system options `VALIDVARNAME=ANY` or `VALIDMEMNAME=EXTEND` were set.

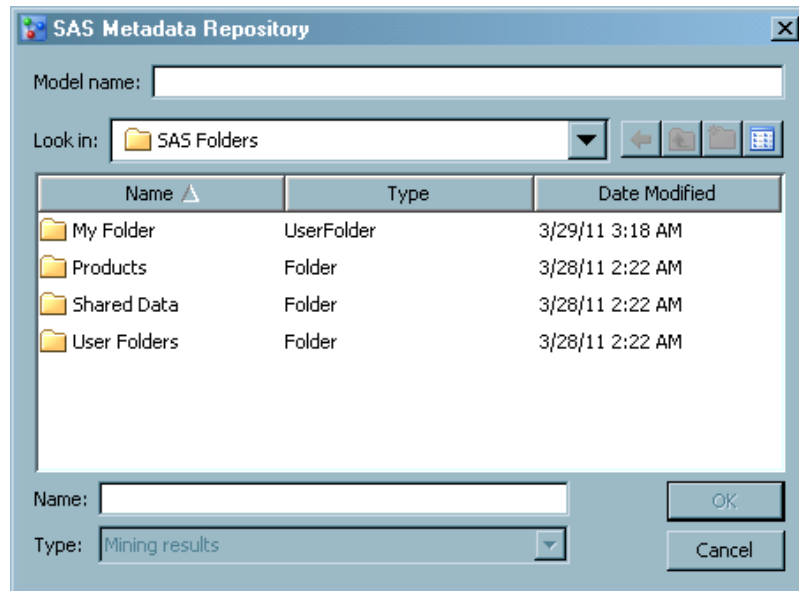
Import Models from the SAS Metadata Repository

If your SAS Enterprise Miner 5.1 (or later) 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 or set the **Model Function** property to **Any**. For more information about model functions, see “Specific Properties for a Project” on page 359.

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** ⇒ **SAS Metadata Repository**. The SAS Metadata Repository window appears.



3. Navigate to the location of the folder that contains the SAS Enterprise Miner model.
4. In the **Model name** field, enter a model name. The name that you enter is used as the name of the model in the **Models** folder. If you do not complete this field, SAS Model Manager imports the model using the model name as it is registered in the SAS Metadata Repository.
5. Select a model from the folder.

Note: You can import only one model at a time in the SAS Metadata Repository window.

6. Click **OK**. After 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 and view the tabs in the properties panel on the right. You can view your newly imported model's data structures on the **Model Input**, **Model Output**, and **SAS Code** tabs.

See Also

- [“Import Package Files from SAS Enterprise Miner” on page 122](#)
- [“Import Partial Models” on page 136](#)
- [“Set Model Properties” on page 137](#)
- [“Map Model Variables to Project Variables” on page 138](#)

Import Package Files from SAS Enterprise Miner

What Is a SAS Enterprise Miner Package File?

SAS Enterprise Miner package files are SPK files that 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:

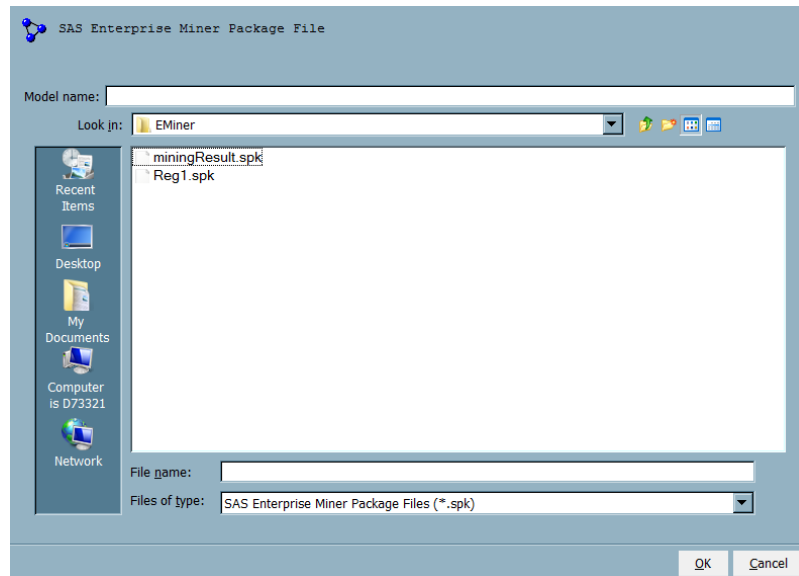
1. Open the SAS Enterprise Miner diagram that contains the model, and then run the model.
2. After the model run is complete, you can right-click the model in the SAS Enterprise Miner Diagram Workspace, and select **Create Model Package**. The new SPK filename appears under the Model Packages folder 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 359](#).

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. Navigate to the location of the SAS Enterprise Miner package (SPK) file and select the file.
4. To change the name of the model, 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.
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 136](#)
- [“Set Model Properties” on page 137](#)
- [“Map Model Variables to Project Variables” on page 138](#)

Import SAS Code Models and R Models Using Local Files

Overview of Importing SAS Code Models

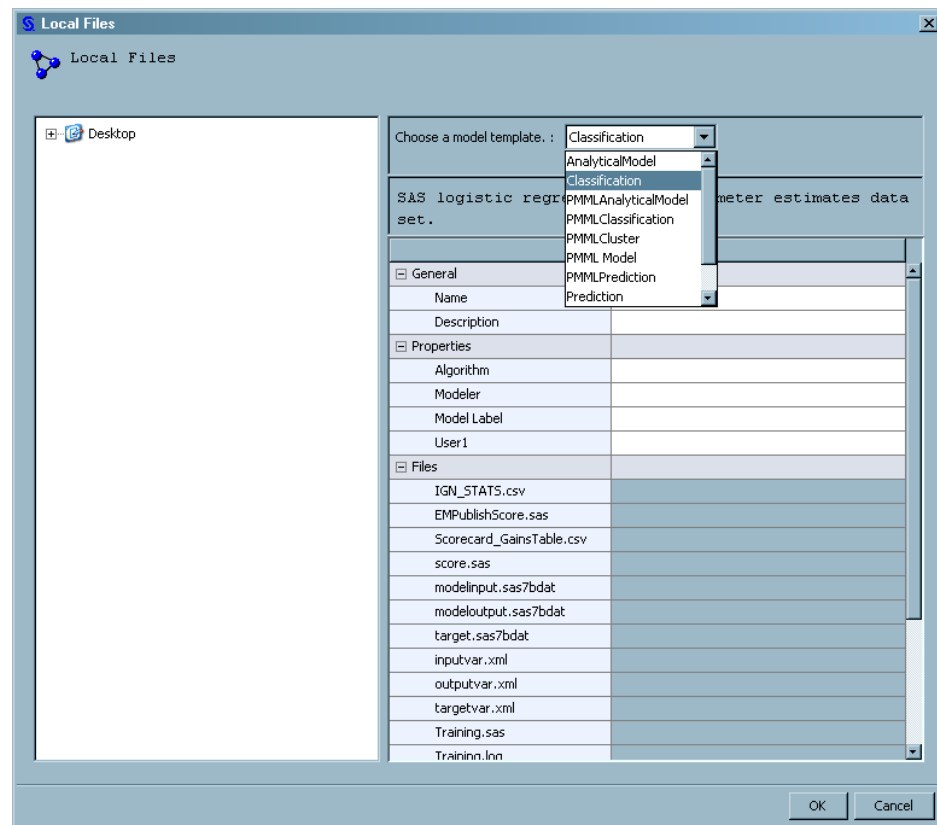
You use the Local Files method to import models that you create using SAS code, but that were not created in or exported from SAS Enterprise Miner, and R models. An example of such a model might be a SAS LOGISTIC procedure model or an R logistic model.

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 for SAS code models are analytical, classification, prediction, segmentation or any. The model functions for R

models are analytical, classification, or prediction. For a list of component files by model function, see “[Model Template Component Files](#)” on page 127. 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 136.

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 8.1 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.

R models, at a minimum, require SAS and R score code component files, a file for the output parameter estimate, and the 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 has properties and 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 dragging a component filename from a tree view to the corresponding SAS Model Manager filename. For example, most templates have a `modelinput.sas7bdat` component file. In the Local Files window, navigate to the location of model component files and drag your `myModelInput.sas7bdat` file to the **Files** section and drop it as the value for `modelinput.sas7bdat`. Here is the local file model template component `myModelInput.sas7bdat` as the value for the `modelinput.sas7bdat` component file:

<code>modelinput.sas7bdat</code>	<code>myModelInput.sas7bdat</code>
----------------------------------	------------------------------------

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 127](#).

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.

Model Type	Description
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. A segmentation or cluster model is designed 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. R models cannot have segmentation model function.
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.

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 136.

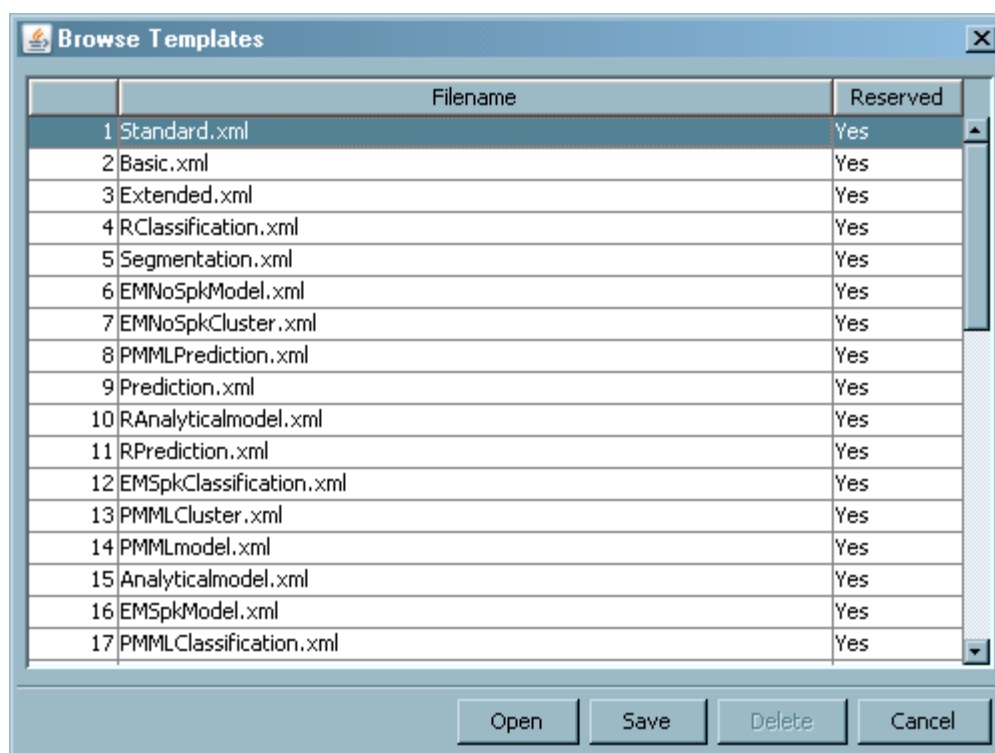
User Model Templates

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 139.

You can view the user model template files (as well as all of the other SAS Model Manager template files) in the Browse window of the SAS Model Manager Template Editor:

1. Select **Tools** ⇒ **Manage Templates**. The SAS Model Manager Template Editor appears.
2. Select **File** ⇒ **Browse** ⇒ **Browse Templates**. The Browse Templates window appears.



Note: The **Reserved** column indicates that whether the template can be modified.

Yes indicates that the template cannot be modified. **No** indicates that the template can be modified. You use reserved templates as a model to create a customized template. When a reserved template is uploaded to the SAS Content Server, SAS Model Manager creates a new file with the same name and changes the new file's **Reserved** value to **No**.

3. Select a template and click **Open**. The template opens in the **Template Editor**.

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	✓	✓	✓	✓

Filename	Analytical	Classification	Prediction	Segmentation
modeloutput.sas7bdat	✓	✓	✓	✓
target.sas7bdat	—	✓	✓	—
inputvar.xml	✓	✓	✓	✓
outputvar.xml	✓	✓	✓	✓
targetvar.xml	—	✓	✓	—
training.sas	✓	✓	✓	✓
trainin.log	✓	✓	✓	✓
training.lst	✓	✓	✓	✓
outest.sas7bdat	✓	✓	✓	—
outmodel.sas7bdat	✓	✓	✓	—
output.spk	✓	✓	✓	✓
format.sas7bcat	✓	✓	✓	✓
dataprep.sas	✓	✓	✓	✓
training.r	✓	✓	✓	—
outmodel.rda	✓	✓	✓	—
score.r	✓	✓	✓	—

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.

For R models, this file transforms a scoring data set to an R data frame.

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 that 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.

training.r

This is an optional R script file that is used to retain R models in SAS Model manager.

outmodel.rda

SAS Model Manager requires this file to save the output parameter estimate for R models.

score.r

This file is an R script that is used to predict new data.

For information about preparing R model component files, see [Appendix 5, “SAS Model Manager R Model Support,”](#) on page 371.

Viewing Model Template Files

You can view model template files using the Browse Templates window. You open the Browse Templates window from the SAS Model Manager Template Editor

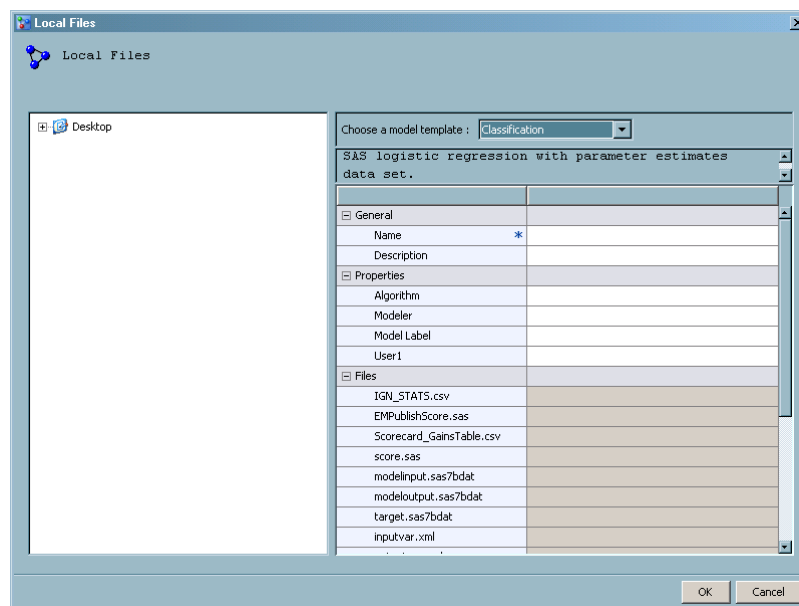
1. From the SAS Model Manager window, select **Tools** ⇒ **Manage Templates**.
2. From the SAS Model Manager Template Editor, select **File** ⇒ **Browse** ⇒ **Browse Templates**.
3. In the Browse Templates window, select the template.
4. Click **Open**. The template opens in the SAS Model Manager Template Editor.

Reserved templates cannot be modified.

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 125](#).
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.

- e. Drag the R script file for scoring to the **score.r** property.
- f. Drag the SAS scoring program to the **score.sas** property.
6. If you have other component files, drag them to their respective properties.
7. Click **OK**. After SAS Model Manager processes the model import request, the new model appears in the **Models** folder of your project's version.
8. Repeat steps 2 through 7 to import additional R files from your client computer folder.

Import PMML Models

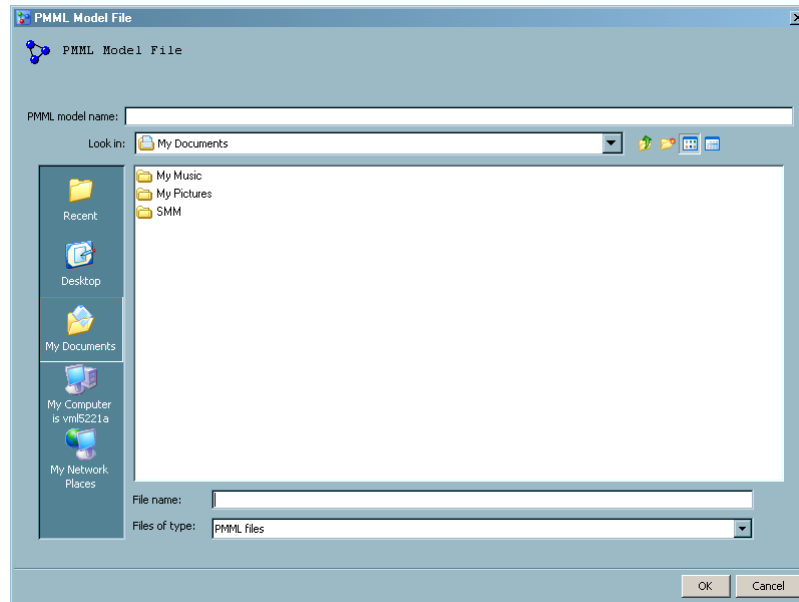
PMML (Predictive Modeling Markup Language) is an XML-based standard for representing data mining results. PMML is designed to enable the sharing and deployment of data mining results between vendor applications and across data management systems. The SAS Model Manager Import PMML Models feature enables users to import PMML models that are produced by using another application, such as SAS Enterprise Miner. SAS Model Manager does not support the importing of a PMML file that contains multiple models. PMML 3.1 (or later) is supported by SAS Model Manager. For more information about PMML support in SAS Enterprise Miner, see the *SAS Enterprise Miner Reference Help*.

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, see [“Specific Properties for a Project”](#) on page 359.

To import a PMML model into SAS Model Manager, follow these steps:

Note: After you import a PMML model, you cannot score the model, publish a scoring function to a database or define a performance task if the PMML model is set as the champion model.

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** ⇒ **PMML Model File**. The PMML Model File window is displayed.



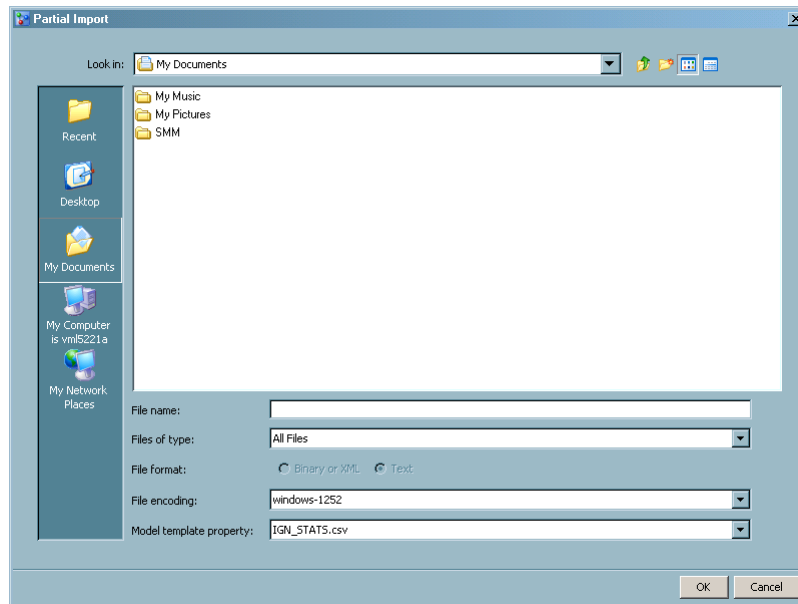
3. Navigate to the location of the PMML file and select the file.
4. Enter a text value in the **PMML model name** field.
5. Click **OK**. After the SAS Model Manager processes the model import request, the new model appears in the **Models** folder of your project's version.
6. Repeat steps 2 through 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 *model name* and select **Partial Import**. The Partial File Import window is displayed.



3. Select a model template component file from the **Model template property** list.
4. Select a file encoding in the **File encoding** list.
5. Use the navigation icons to locate the file to be imported and select the file to complete the **File name** field.
6. 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

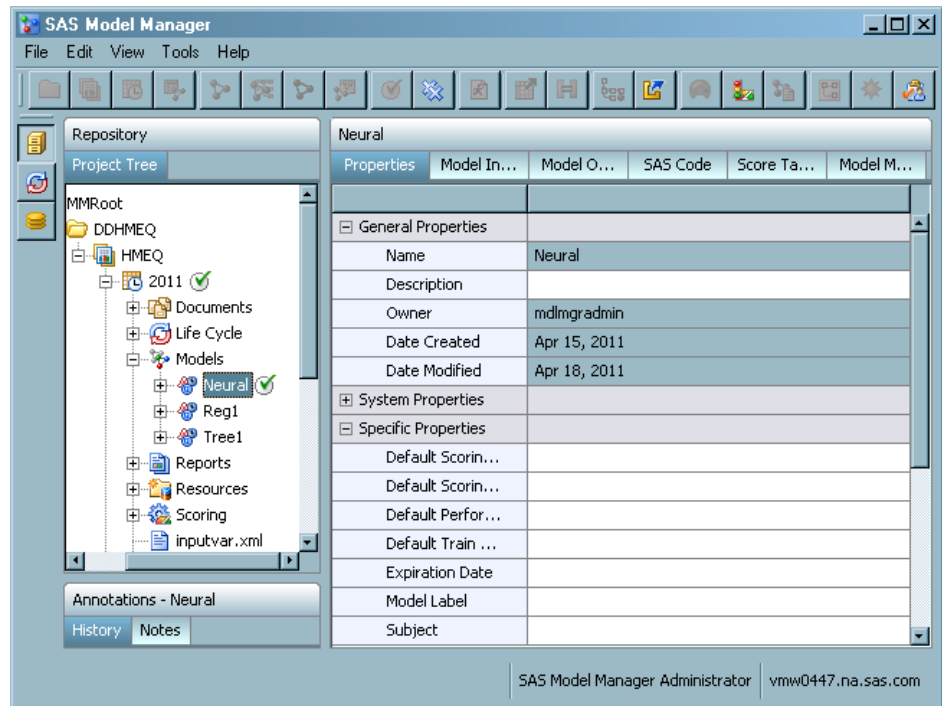
- [“Import SAS Code Models and R Models Using Local Files” on page 123](#)
- [“Set Model Properties” on page 137](#)
- [“Map Model Variables to Project Variables” on page 138](#)

Set Model Properties

After you import a model into SAS Model Manager, you 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. In the General Properties section, enter a model description 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 357](#).
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 358](#).
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 145](#).

See Also

- [“Import Partial Models” on page 136](#)
- [“Map Model Variables to Project Variables” on page 138](#)

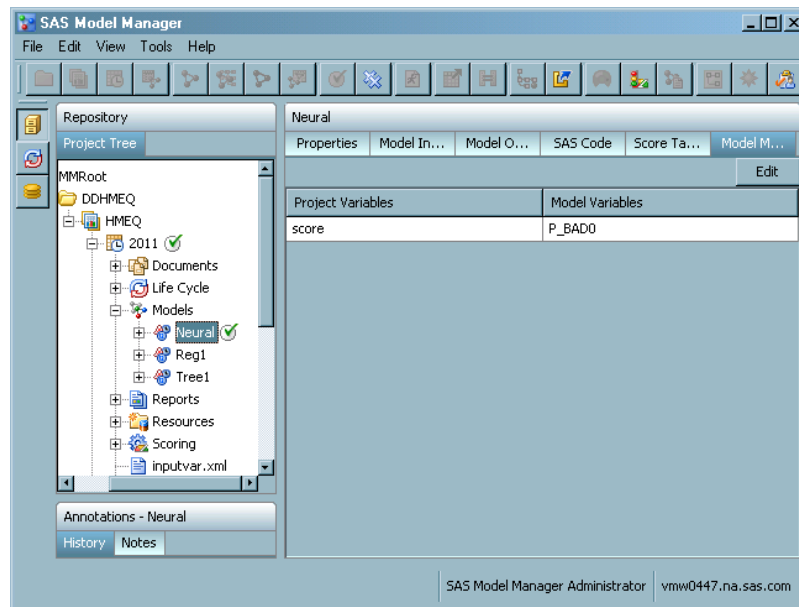
Map Model Variables to Project Variables

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. For more information about project input and output tables, see [“Project Tables” on page 29](#).

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

1. Select the model in the SAS Model Manager Project Tree.

- Click the **Model Mapping** tab on the right and click **Edit**. The **Set Model Output Mapping** window appears.



Note: Alternatively, you can open the **Set Model Output Mapping** window by right-clicking the model name and selecting **Set Model Output Mapping**.

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

See Also

- “Set Model Properties” on page 137
- “Overview of Importing Models” on page 119

User-Defined Model Templates

Creating a New Model Template

Overview of Model Templates

When you import a SAS code model or R 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 127. For a list of properties, see “[Model Template Properties](#)” on page 143.

Note: Only a user or group with the role of Model Manager: Administration Usage can upload a template after it has been created by a user. Several sample user template XML files are included with the SAS Model Manager installation package and are available to be used as a starting point for creating your own model template.

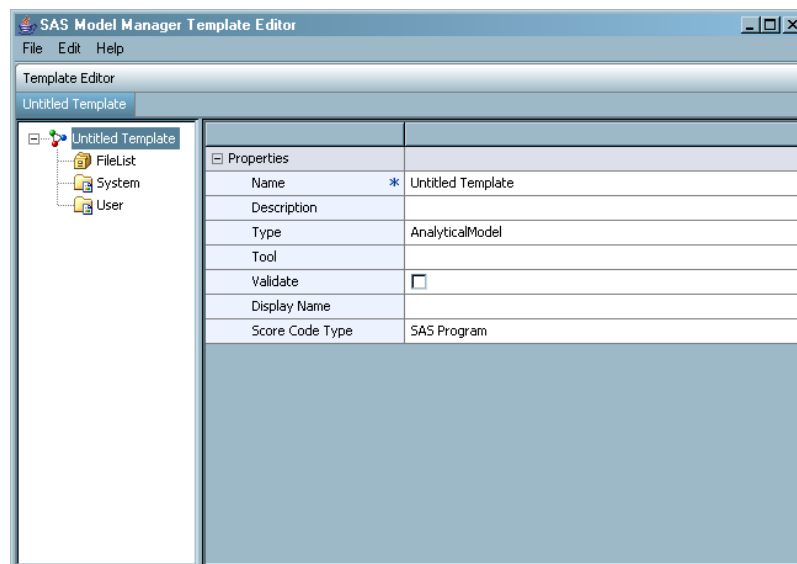
To open the Template Editor from the SAS Model Manager menu, select **Tools** ⇒ **Manage Templates**.

Create a New Model Template

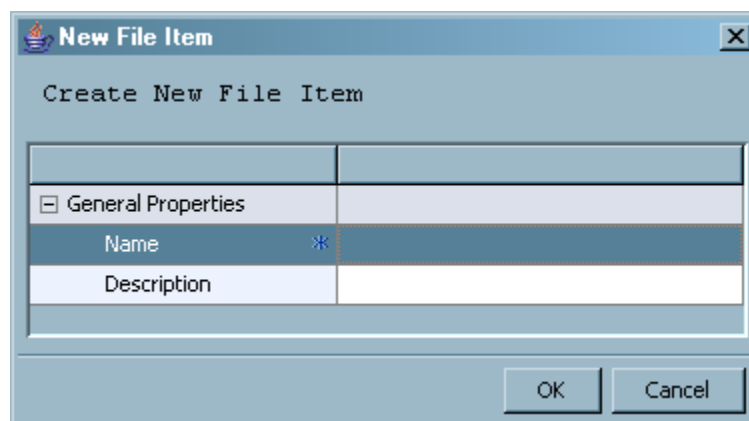
You can create a new model template either by modifying an existing model template or by starting from an empty model template. To create a model template by modifying an existing template, see “[Modify a Model Template](#)” on page 141.

To create a new model template from an empty 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 143.
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 appears.



4. Enter a **Name** and **Description** for the file, and then click **OK**. For more information, see “[Model Template Component Files](#)” on page 127.
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 144.
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 appears.

Create New Property	
<input checked="" type="checkbox"/> General Properties	
Name *	
Description	

OK Cancel

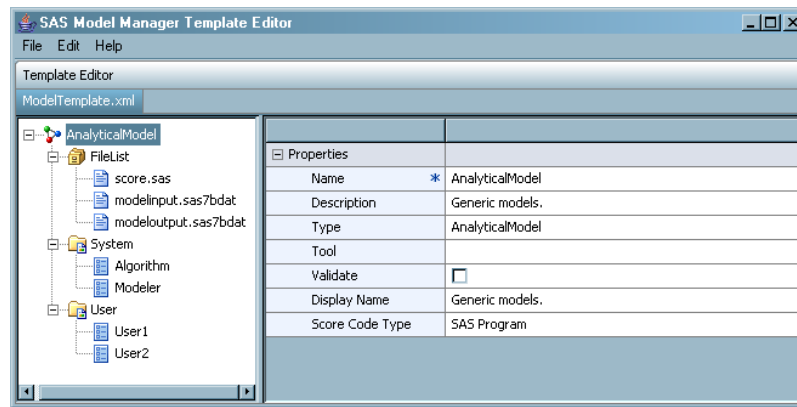
- b. Enter a **Name** and **Description** for the new property, and then click **OK**.
 - c. After all required properties have been created, select each property name and enter the property field values on the right. For more information, see “[System and User Properties](#)” on page 145.
7. To save the template, select **File** ⇒ **Save As**. Then select a directory and filename for the template. This creates a backup of the template at a local or network location.
8. Upload the template to the SAS Content Server. From the Browse Templates window, select **Upload File**.
9. In the Upload File window, ensure that the name is correct and click **OK**.

Modify a Model Template

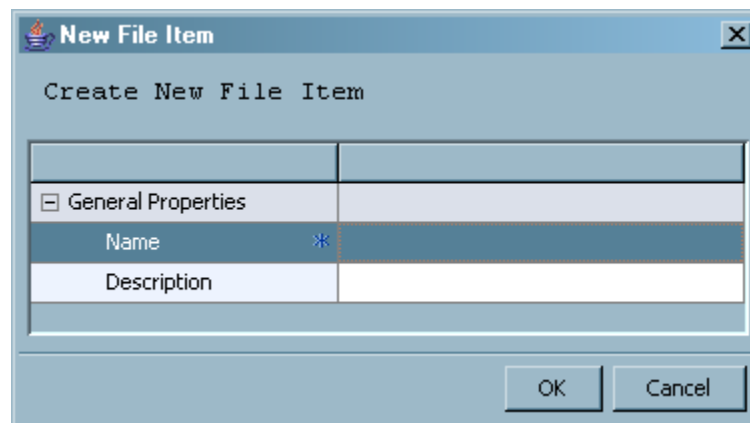
Only user templates can be modified with the use of the SAS Model Manager Template Editor. Reserved templates can be used as a model to create a customized model template. When you open a reserved template, modify the template, and upload the template to the SAS Content Server, SAS Model Manager creates a new user template using the same name as the reserved template, the template is not reserved.

To modify a model template, follow these steps:

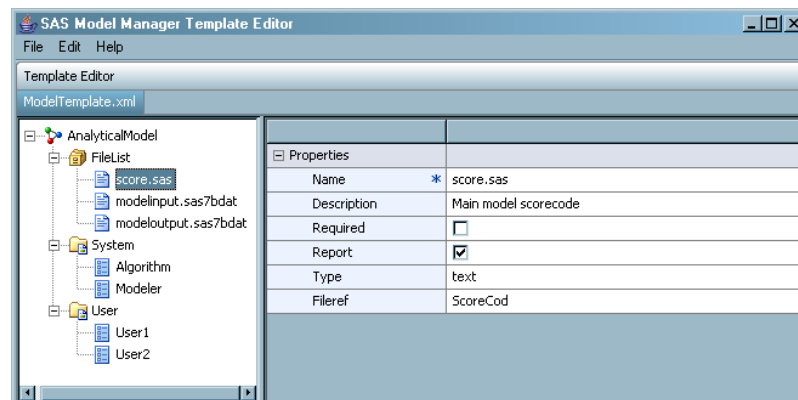
1. From the SAS Model Manager Template Editor window, select **File** ⇒ **Browse** ⇒ **Browse Templates**.
2. Select a model template, and then click **Open**. The template properties appear on the right.



3. To modify model template properties, select the property and make changes to the property value. For more information, see [“Template Properties” on page 143](#).
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**.

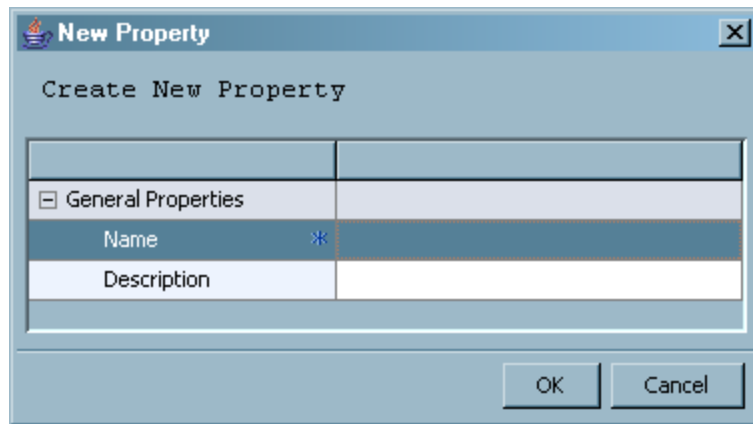


- To modify file properties, select the filename and make changes to the property values.

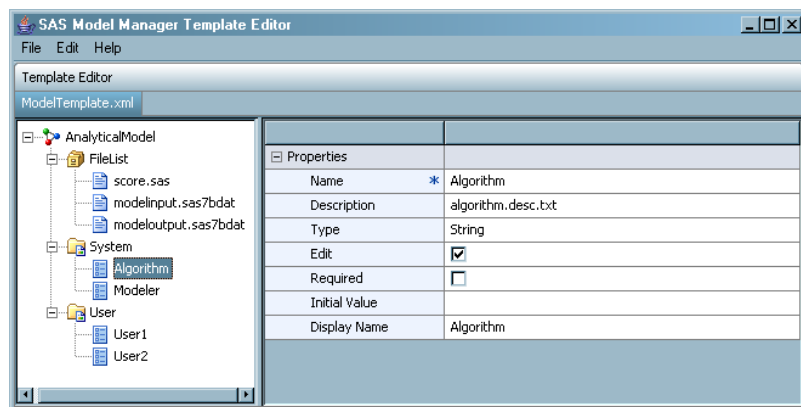


For more information, see [“FileList Properties” on page 144](#).

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 145.

- 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.

Saving the template creates a backup file of the template.

- Upload the template to the SAS Content Server. From the Browse Templates window, select **Upload File**.
- In the Upload File window, ensure that the name is correct and click **OK**.

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 characters @ \ / * % # & \$ () ! ? < > ^ + ~ ` = { } [] ; : ' " cannot be used in the name.

Property Name	Description
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 125.</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 before using the model.
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.

Property Name	Definition
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 must 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	<p>Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager project.</p> <p>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.</p>
Default Train Table	<p>The train table is optional and is used only as information.</p> <p>However, when a value is specified for a model's Default Train Table property, it is used to validate scoring functions when a user publishes the associated project champion model to a database.</p>
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.

Property Name	Description
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 119 .
Modeler	Specifies the Modeler ID or, when Modeler ID is missing, specifies the user ID of the individual who created the model that 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	<p>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.</p> <p><i>Note:</i> SAS Model Manager cannot export or publish models whose Score Code Type model property is set to SAS Program.</p>
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 .

Chapter 9

Scoring Models

Overview of Scoring Tasks	149
Scoring Task Tabbed Views	151
Create Scoring Output Tables	153
What Is a Scoring Output Table?	153
How to Create a Scoring Output Table	153
Create a Scoring Task	155
Modify a Scoring Task	156
Map Scoring Task Output Variables	157
Execute a Scoring Task	158
Graph Scoring Task Results	159
Generated Scoring Task Content Files	163
Scoring Task Properties	163
Result Set Properties	164

Overview of Scoring Tasks

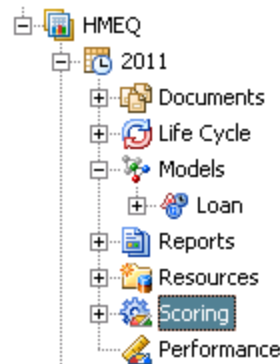
The purpose of a scoring task within SAS Model Manager is to run the score code of a model and produce scoring results that you can use for scoring accuracy and performance analysis. The scoring task uses data from a scoring task input table to generate the scoring task output table. The types of score 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 is mostly limited to testing the score code. Otherwise, you can use the SAS Model Manager scoring tasks both to 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 location that the output table metadata specifies. 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 30](#).


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 "2011".



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 153](#) and [“Project Tables” on page 29](#).
- 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 155](#) and [“Scoring Task Tabbed Views” on page 151](#).
- 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 157](#).
- 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 158](#).
- 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 159](#).

See Also

- [“Create Scoring Output Tables” on page 153](#)
- [“Create a Scoring Task” on page 155](#)
- [“Modify a Scoring Task” on page 156](#)
- [“Map Scoring Task Output Variables” on page 157](#)
- [“Execute a Scoring Task” on page 158](#)
- [“Graph Scoring Task Results” on page 159](#)

Scoring Task Tabbed Views

Associated with each scoring task are tabbed views that 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	<p>This view contains four groupings of properties, some of which have fields that can be modified. Click Save, when 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 357 • “System Properties” on page 358 • “Scoring Task Properties” on page 163 • “Result Set Properties” on page 164
Model Input Variables	This view shows the model input variables.
Input Table	<p>This view shows the variables and attributes of the scoring task input table. For each input variable the model lists in the Model Input Variables, there must be a matching input variable in the input table. The input table can contain additional variables.</p>
Model Output Variables	This view shows the model output variables.
Output Table	<p>This view shows the variables and attributes of 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, correct them and then click Save at the bottom right of the view.</p> <p>A Permission Denied message is displayed if a user accesses this view before the creation of the scoring task. Because 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	<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. 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 that are specified in the generated code section.</p>
Post-code	<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. 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 that are specified in the generated code section.</p>
SAS Code	<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 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	<p>This view shows three separate views depending on which button you click. Here are descriptions of each view:</p> <ul style="list-style-type: none"> • Result Set is the view of the result data set. • Log is the view of the log from the last execution of this scoring task. This is the default view. • Output is the view of the listing file from the last execution of this scoring task. <p>Both the Log and the Output views 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 that is known to the SAS Metadata Repository. For more information, see “Create Scoring Output Tables” on page 153.</p>
Graph	<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 159.</p>

See Also

[“Modify a Scoring Task” on page 156](#)

Create Scoring Output Tables

What Is a Scoring Output Table?

A scoring output table contains the definition of one or more output variables. After a scoring task is run, the scoring table contains 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 definition of that 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**
2. Right-click *model name* and select **Create Output Table** from the pop-up menu.

Note: If no value for the model property **Default Scoring Task Input Table** is specified on the **Properties** tab, the **Input Variables** section is empty.

3. 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.
4. Select a SAS library name from the **Library** list.

Note: You must have Write access to the SAS library.

5. 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.
6. Select the **Add model ID** check box to add the **ModelID** variable to the output table.
7. 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.

8. 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**.

9. 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 149
- “Project Tables” on page 29

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, and then select **New** ⇒ **New Scoring Task** from the pop-up menu.

New Scoring Task

Set Scoring Task Properties

Specify values for the scoring task properties.

Step 1 of 2

General Properties

Name:

Description:

Model:

Scoring task type: ☒ Test ☐ Production

Select Scoring Task Tables

Input table: ...

Output table: ...



Back Next Finish Cancel Help

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

Note: When a model is selected, the values in the **Input Table** field and **Output Table** fields are cleared.

4. 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 to put the task into production, you can change the type to **Production**. When you run in production mode in the Windows and UNIX environments, the scoring task output table definition in the SAS Metadata Server must have Modify, Read & Execute, Read, and Write security permissions. The SAS Model Manager user that is executing the scoring task must have Write permission to the physical folder on the SAS Workspace server where the scoring task output table is written. If the user does not have Write permission, the scoring task fails, and an error message appears.

5. Click  and select an input table.
6. Click  and select an output table. Click **Next**.
7. Map the scoring output table variables to the model output variables. For each variable, click in the **Map to Model Variable Name** field and select a model output variable.
8. Click **Finish**. The scoring task is created under the **Scoring** node.

Note: Four of the **Scoring Task** properties cannot be modified after 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 156](#)
- [“Execute a Scoring Task” on page 158](#)
- [“Generated Scoring Task Content Files” on page 163](#)
- [“Scoring Task Properties” on page 163](#)

Modify a Scoring Task

The following are the only four tabbed views that can be modified:

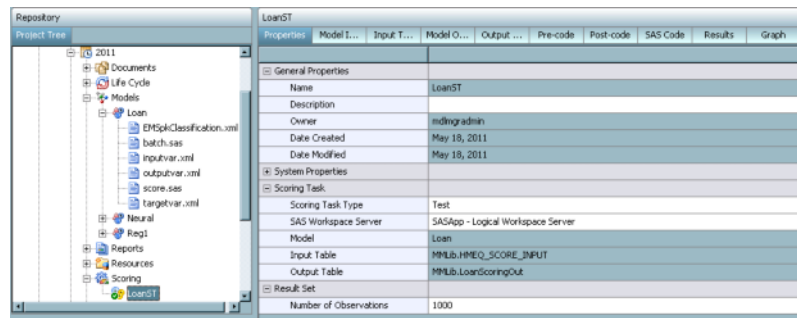
- **Properties**
- **Output Table**
- **Pre-code**
- **Post-code**

For more information, see [“Scoring Task Tabbed Views” on page 151](#).

To modify a scoring task, follow these steps:

1. Expand the **Scoring** folder and select your scoring task.

- Click the tab for which you want to modify the information. The default is the **Properties** tab.



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

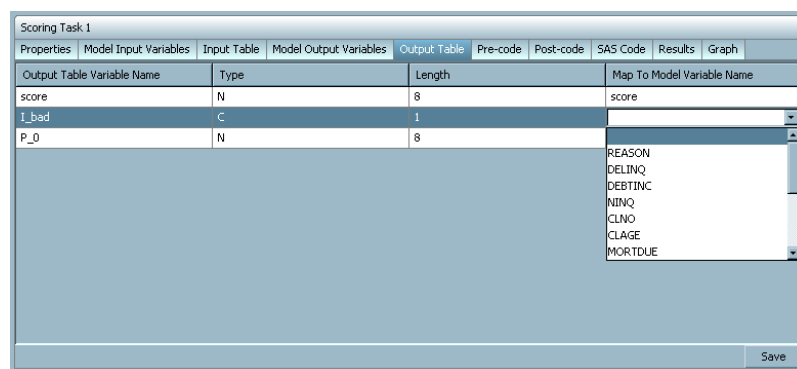
See Also

- “Create a Scoring Task” on page 155
- “Execute a Scoring Task” on page 158
- “Generated Scoring Task Content Files” on page 163
- “Scoring Task Tabbed Views” on page 151

Map Scoring Task Output Variables

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

- Expand the **Scoring** folder and select your scoring task.
- Click the **Output Table** tab.



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


See Also

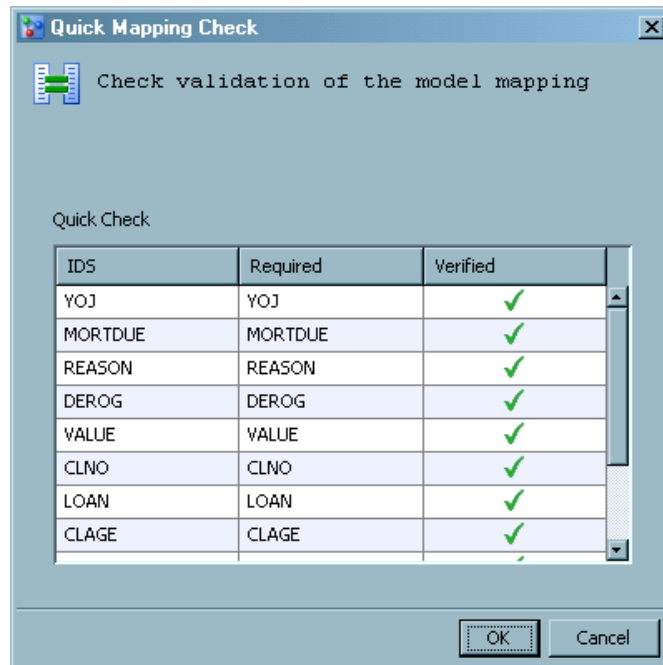
- “Create Scoring Output Tables” on page 153

- “Create a Scoring Task” on page 155
- “Modify a Scoring Task” on page 156
- “Execute a Scoring Task” on page 158


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 157.
2. Select the scoring task name, and click  in the toolbar to validate the input variables.



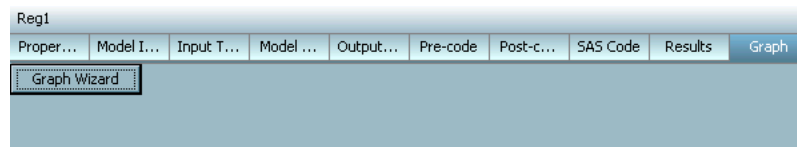
The table in the Quick Mapping Check window compares the column names from the Scoring Input Table (that is, the input data source) against the model's input variable names.

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. A progress bar appears at the bottom of the **SAS Model Manager** window.
4. After the task has been completed, a success or failure message is displayed. Click **Close**, and then review the log for error messages.
5. Click the **Result** tab and then **Result Set** to view the scoring task results.

Reg1			
Proper...	Model I...	Input T...	Model ...
Output...	Pre-code	Post-c...	SAS Code
Results	Graph		
customer_id	EM_PROBABILITY		
1	118-296-340	0.9108204518	
2	126-291-396	0.9108204518	
3	154-253-305	0.9108204518	
4	107-281-352	0.9108204518	
5	184-207-395	0.9108204518	
6	129-227-368	0.884123366085219	
7	197-222-368	0.9108204518	
8	141-255-328	0.8921406849072309	
9	147-284-363	0.9108204518	
10	158-258-337	0.9108204518	
11	172-250-392	0.9108204518	
12	192-258-329	0.9108204518	
13	139-247-367	0.9108204518	
14	117-216-386	0.9108204518	
15	130-293-389	0.9108204518	
16	156-205-313	0.9108204518	
17	151-243-317	0.9108204518	
18	166-240-312	0.9108204518	
19	145-220-357	0.9108204518	
20	173-244-305	0.9599852657134139	
21	152-234-302	0.9108204518	
22	171-293-344	0.9108204518	
23	194-271-310	0.9108204518	
24	117-227-361	0.9108204518	
25	142-207-335	0.9108204518	

6. Click the **Graph** tab and then **Graph Wizard** to graph the results. For more information, see [“Graph Scoring Task Results” on page 159](#).



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 163](#).

See Also

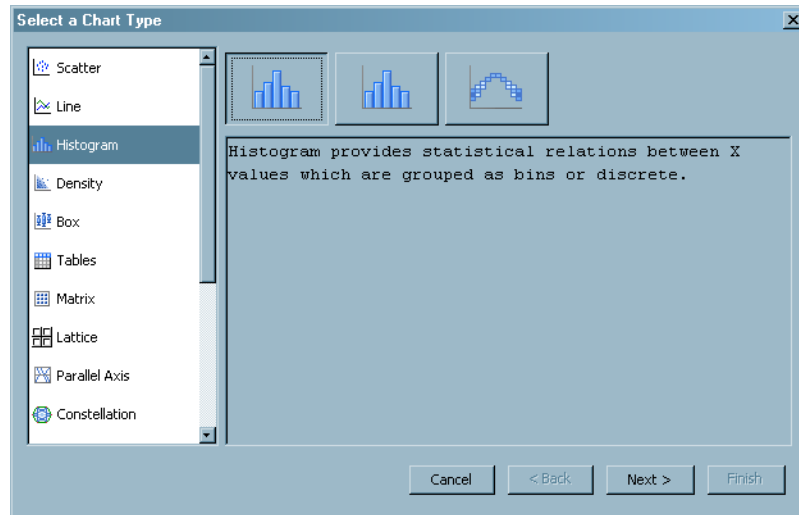
- [“Create a Scoring Task” on page 155](#)
- [“Modify a Scoring Task” on page 156](#)
- [“Generated Scoring Task Content Files” on page 163](#)
- [“Graph Scoring Task Results” on page 159](#)

Graph Scoring Task Results

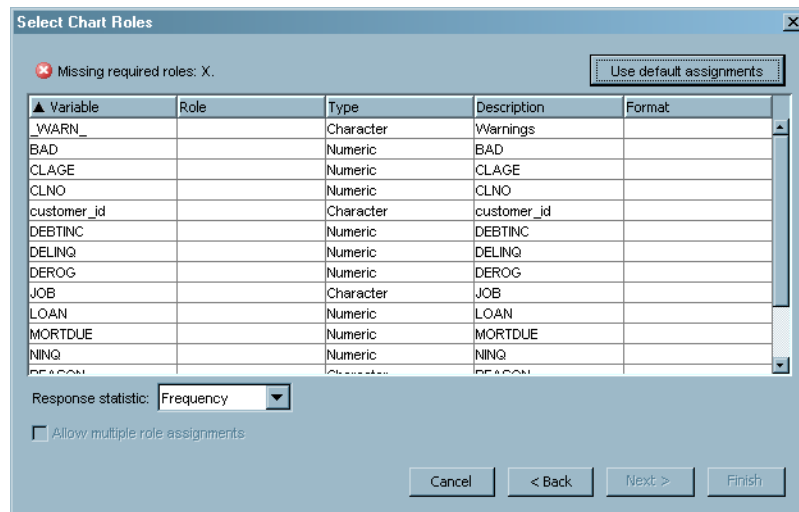
After the scoring task successfully executes, you can use the graphics wizard to plot your tailored charts.

To graph scoring task results, follow these steps:

1. Select the scoring task in the **Scoring** node.
2. Select the **Graph** tabbed view of the scoring task and then click **Graph Wizard**. The Select a Chart Type window appears.



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 appears.
4. Click **Next**. The Select Chart Roles window appears.



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

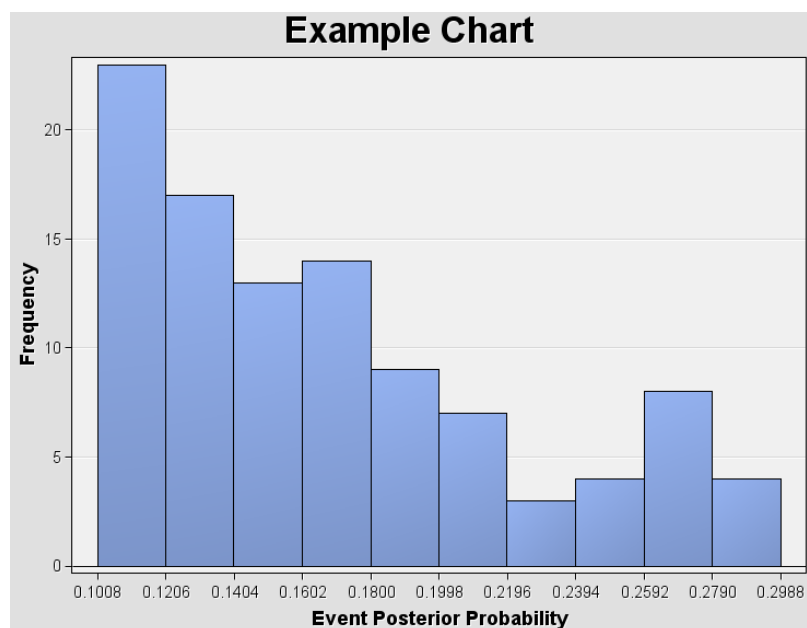
☒ Horizontal

☒ Vertical

☐ Depth

Cancel < Back Next > Finish

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



See Also

[“Execute a Scoring Task” on page 158](#)

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 that are 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 that 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 151 .
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 151 .
<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 158](#)
- [“Scoring Task Tabbed Views” on page 151](#)

Scoring Task Properties

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

Property Name	Description
Scoring Task Type	Specifies a value of Test or Production for the type of scoring task.
SAS Workspace Server	Specifies the name of the Workspace Server to which SAS Model Manager is connected. This value is taken from the SAS Metadata Repository.
Model	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 , 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 setting identifies the output table where the results of the scoring are written.

See Also

- [“Create a Scoring Task” on page 155](#)
- [“Modify a Scoring Task” on page 156](#)
- [“Execute a Scoring Task” on page 158](#)

Result Set Properties

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

Property Name	Description
Number of Observations	<p>When Scoring Task Type is set to Test, this property specifies how many observations are to be read from the scoring task input table. This setting enables you to limit the number of records that are 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.</p> <p>When Scoring Task Type is set to Production, this property specifies how many observations show in a result set in order to limit the number of observations to display. This value can be set only by a SAS Model Manager administrator using SAS Management Console. If a value is not specified, the default value is 0. A 0 indicates no limit on the number of observations to display. For information about setting Number of Observations when Scoring Task Type is Production, see <i>SAS Model Manager: Administrator's Guide</i>.</p>

See Also

- [“Create a Scoring Task” on page 155](#)
- [“Modify a Scoring Task” on page 156](#)

Chapter 10

Validating Models Using Comparison Reports

Overview of Model Comparison Reports	167
What Is a Model Comparison Report ?	167
Model Comparison Input File	168
Model Comparison Output Files	168
Model Profile Reports	169
About Model Profile Reports	169
Create a Model Profile Report	169
Creating Delta Reports	171
About Delta Reports	171
Create a Delta Report	171
Creating Dynamic Lift Reports	173
About Dynamic Lift Reports	173
Verify Project and Model Property Settings	173
Create a Dynamic Lift Report	174
View Reports	176

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 SAS Metadata Repository using SAS Management Console and that the table can be viewed in the Data Sources perspective. Then, specify the test table in the project property **Default Test Table**.

See Also

- [“Specific Properties for a Project” on page 359](#)
- [“Creating a Test 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** table.

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

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

Note: If you save a report to a local drive, images in the reports, such as graphs, do not appear. The report images are separate files and are stored in the SAS Content Server. Always view reports 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 169](#)
- [“View Reports” on page 176](#)

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.

New Report Wizard

New Report Wizard

Report Options

Type: Model Profile Report

Format: PDF

Select Models

Select	ID	Name	Version	Type	Cham...
<input type="checkbox"/>	MMRoot/D...	Neural	2011	Classificati...	NO
<input checked="" type="checkbox"/>	MMRoot/D...	Tree1	2011	Classificati...	NO
<input type="checkbox"/>	MMRoot/D...	Loan	2011	Classificati...	NO
<input type="checkbox"/>	MMRoot/D...	Reg1	2011	Classificati...	YES

Report Properties

Property	Value
[-] General Properties	
Name *	profile_D2011-05-18T15.25
Description	profile

OK Cancel

3. Select **Model Profile Report** from the **Type** list box.
4. In the **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 Models** table, check the box for one or more models 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 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 176](#)

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 from 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 from 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 171](#)
- [“View Reports” on page 176](#)

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

New Report Wizard

Report Options

Type: Delta Report

Format: PDF

Select Models

Select	ID	Name	Version	Type	Cham...
<input type="checkbox"/>	MMRoot/D...	Neural	2011	Classificati...	NO
<input type="checkbox"/>	MMRoot/D...	Tree1	2011	Classificati...	NO
<input checked="" type="checkbox"/>	MMRoot/D...	Loan	2011	Classificati...	NO
<input checked="" type="checkbox"/>	MMRoot/D...	Reg1	2011	Classificati...	YES

Report Properties

Property	Value
[-] General Properties	
Name	* delta_D2011-05-18T16.01
Description	delta

OK Cancel

3. Select **Delta Report** from the **Type** list box.
4. In the **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 Models** 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 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 176](#)

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 173](#)
- [“Create a Dynamic Lift Report” on page 174](#)
- [“View Reports” on page 176](#)

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, click the **Model Variables** property for each project name, 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

New Report Wizard

Report Options

Type:

Format:

Select Models

Select	ID	Name	Version	Type	Champion
<input type="checkbox"/>	MMRoot/DD...	Neural	2011	Classificatio...	NO
<input type="checkbox"/>	MMRoot/DD...	Tree1	2011	Classificatio...	NO
<input type="checkbox"/>	MMRoot/DD...	Loan	2011	Classificatio...	NO
<input checked="" type="checkbox"/>	MMRoot/DD...	Reg1	2011	Classificatio...	YES

Report Properties

Property	Value
[-] General Properties	
Name *	dynamicLift_D2011-05-18T16.14
Description	dynamicLift

OK Cancel

3. Select **Dynamic Lift Report** from the **Type** list box.
4. In the **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 Models** table, check the boxes for the models that you want to include in the report.
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 `dynamicLift_DdateTtime`. The 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 173
- “View Reports” on page 176

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 11

Validating Models Using User Reports

Overview of User Reports	177
Ad Hoc Reports and User-Defined Reports	177
Comparison of Ad Hoc and User-Defined Reports	178
Output Created by User Reports	178
Ad Hoc Reports	179
Overview	179
Create an Ad Hoc Report	179
Example Ad Hoc Report	180
User-Defined Reports	182
Overview of User-Defined Reports	182
Create a User-Defined Report	182
Defining SAS Model Manager Macro Variables for a User-Defined Report	182
Upload SAS Programs to the SAS Content Server	183
The Report Template	183
Edit a SAS Program on the SAS Content Server	185
Delete a SAS Program from the SAS Content Server	185
Run a User-Defined Report	186
View a User-Defined Report	186
Example User-Defined Report	186

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

After you have these three files, you use the SAS Model Manager Template Editor to upload the files to the SAS Content Server.

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 in the User-defined report file structure.

Comparison of Ad Hoc and User-Defined Reports

Report Difference	Ad Hoc Report	User-Defined Report
Version	An ad hoc report is defined and can be run only under the version where it was created.	A user-defined report can be run under any project version.
Report template	An ad hoc report does not require a template.	A user-defined report requires 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 uploaded to the SAS Content Server.

Output Created by User Reports

The first time 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.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.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 that was 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 uploaded to the SAS Content Server.


To create your report output in either HTML, PDF, RTF, or Excel, you modify your report with code that is 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 appears.

3. In the **Select Models** 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 characters @ \ / * % # & \$ () ! ? < > ^ + ~ ` = { } [] | ; : ' " cannot be used in the 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 click the **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 that 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

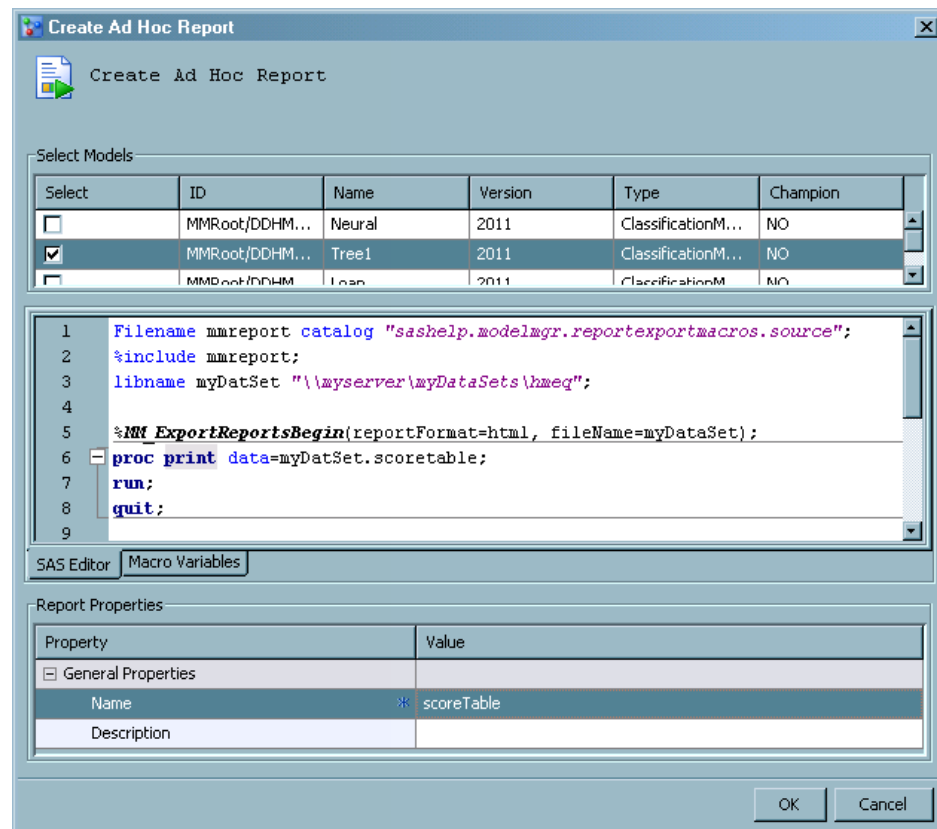
The following example code lists the score results in an HTML output format:

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

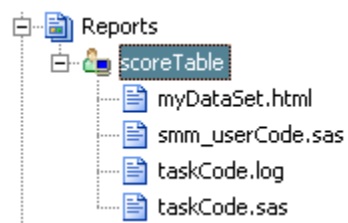
%MM_ExportReportsBegin(reportFormat=html, fileName=myDataSet);
proc print data=myDataSet.scoretable;
run;
quit;

%MM_ExportReportsEnd(reportFormat=html);
```

When you include this program in the Create Ad Hoc Report 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 selected rows of 5000 possible 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 the following files to be uploaded to the SAS Content Server:

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

After these three files have been uploaded to the SAS Content Server, the user-defined report type is included as a report in the New Reports Wizard **Reports** list.

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

Create a User-Defined Report

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, add the SAS code below to 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. End your user-defined report with the %MM_ExportReportsEnd macro.

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

In the report XML file, add this SAS program name to the `<Code filename=" ">` argument. For example, `<Code filename="myUserReport.sas"/>`. See [“The Report Template” on page 183](#).

For an example of a report, see [“Example User-Defined Report” on page 186](#).

Defining SAS Model Manager Macro Variables for a User-Defined Report

Executing a user-defined report requires a SAS program that lists the report code’s SAS Model Manager macro variables. If you do not have SAS Model Manager macro variables in your report, create a SAS program file with a comment in it. This file is required.

Here is an example program to define macro variables:

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

In the report XML file, add this SAS program name to the `<PreCode filename=" ">` element. For example, `<PreCode filename="myMacroDefs.sas"/>`. See [“The Report Template” on page 183](#).

For an example of a SAS Model Manager macro variable programs, see [“Example User-Defined Report” on page 186](#).

For a list of SAS Model Manager macro variables, see [“SAS Model Manager Macro Variables” on page 351](#).

Upload SAS Programs to the SAS Content Server

After you have the two SAS programs for your user report, follow these steps to upload them to the SAS Content Server:

1. Select **Tools** ⇒ **Manage Templates** to open the SAS Model Manager Template Editor.
2. Select **File** ⇒ **Open**, select the program in the Open window, and click **OK**.
3. Select **File** ⇒ **Upload File** to upload the program to the SAS Content Server.
4. Repeat steps 2 and 3 to upload the second file.

The Report Template

You create a report template XML definition file to describe your user-defined report. After you create the report template, upload the template to the SAS Content Server.

SAS Model Manager provides a sample report template that you can use as a model for your XML template. You can use any template as a model or you can create an XML file with the required XML elements. A best practice is to open the model XML template and save the template using another name. To open a sample report template, follow these steps:

1. Select **Tools** ⇒ **Manage Templates** to open the SAS Model Manager Template Editor.
2. Select **File** ⇒ **Browse** ⇒ **Browse Templates**. Select **UserReportTemplate.xml** and click **OK**.
3. Select **File** ⇒ **Save As**. Type a name in the **File name** field and click **OK**.
4. The UserReportTemplate.xml file has arguments in quotation marks that you modify for your report. Replace the text in quotation marks with values that are appropriate for your report. See the argument descriptions below.
5. When the report template is complete, select **File** ⇒ **Upload File** to upload the report template to the SAS Content Server.

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 characters @ \ / * % # & \$ () ! ? < > ^ + ~ ` = { } [] | ; : ' " cannot be used in the 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.

Valid values: ANALYTICAL, CLASSIFICATION, PREDICTION, SEGMENTATION, ANY

`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.

`<Parameters>` Element Argument

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

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

Edit a SAS Program on the SAS Content Server

To edit the program after the file has been uploaded to the SAS Content Server, follow these steps:

1. Select **File** ⇒ **Browse** ⇒ **Browse SAS Files**.
2. Select the program and click **Open**.
3. Modify the program. When you have finished making changes, upload the file to the SAS Content Server by selecting **File** ⇒ **Upload File**.
4. (Optional) To save a backup of the template, from the Browse Templates window, select **File** ⇒ **Save As**. Select a file location, enter a filename, and click **Save**.

Delete a SAS Program from the SAS Content Server


To delete a SAS program from the SAS Content Server, follow these steps:

1. Select **File** ⇒ **Browse** ⇒ **Browse SAS Files**.

2. Select the program and click **Delete**.

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 appears.
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.

View a User-Defined Report

To view a user-defined report, right-click the report name under the Reports node and select **Reports** ⇒ **View Report**.

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 SAS programs and report template file 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_ExportReportsEnd;
```

The SAS Program File for Macro Variables

The file ScoreRangeMacro.sas contains only a comment in it because macro variables are not used in the report code:

```
/* ScoreRangeMacro.sas empty file */
```

The Report Template XML File

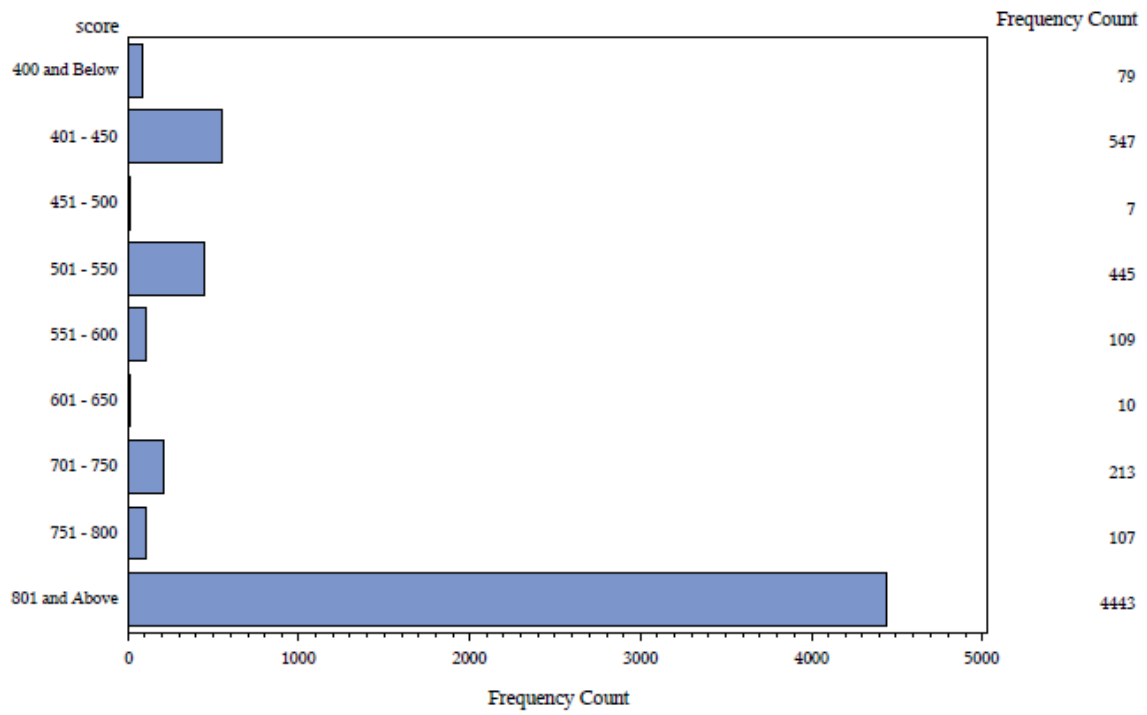
Here is the report template XML file for the user-defined Score Range report:

```
<?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 12</i>	
Deploying Models	193
<i>Chapter 13</i>	
Delivering Models	203
<i>Chapter 14</i>	
Replacing a Champion Model	223

Chapter 12

Deploying Models

Overview of Deploying Models	193
Champion Models	194
About Champion Models	194
Requirements for a Champion Model	194
Set a Champion Model	195
Clear a Champion Model	195
Freezing Models	196
About Freezing Models	196
Freeze a Version	197
Unfreeze a Version	197
The Default Version for a Project	198
About the Default Version	198
Set a Default Version	198
Clear a Default Version	200

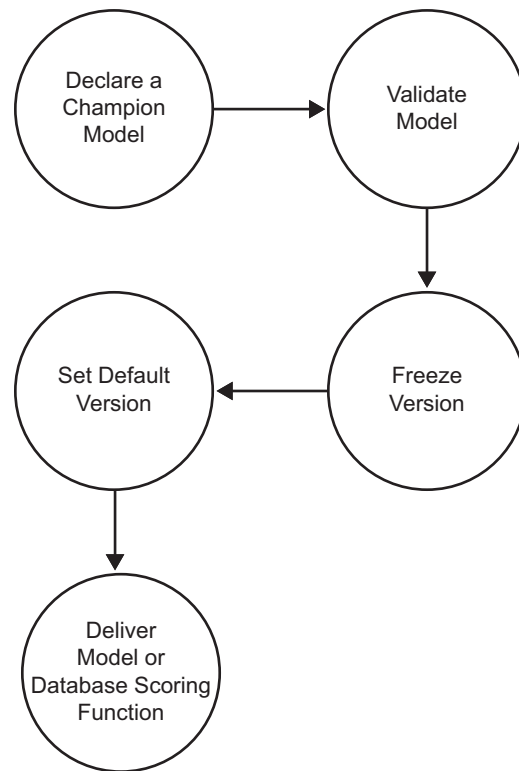
Overview of Deploying Models

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 12.1 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.

These are the tasks that you perform to use champion models:

- [“Set a Champion Model” on page 195](#)
- [“Clear a Champion Model” on page 195](#)

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 appears.
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 **Date Modified** is today's date. The value for the **Champion Model ID** is the champion model's UUID.

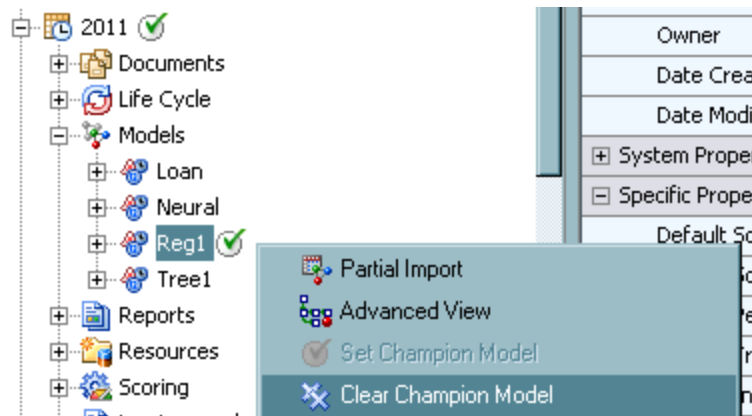
Note: To document the reasons or assumptions for your selection of the champion model, use the version **Notes** tab. SAS Model Manager automatically annotates the History tab.

6. Update the specific properties for the appropriate milestone task in the **Life Cycle** node. Specify that **Status** for selecting a champion model is set to **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 appears.



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 **Date Modified** property is today's date. The value for the **Champion Model ID** property has been cleared.
5. Update the specific properties for the appropriate milestone task in the **Life Cycle** node. Change **Status** for selecting a 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 are 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 197
- “Unfreeze a Version” on page 197


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.

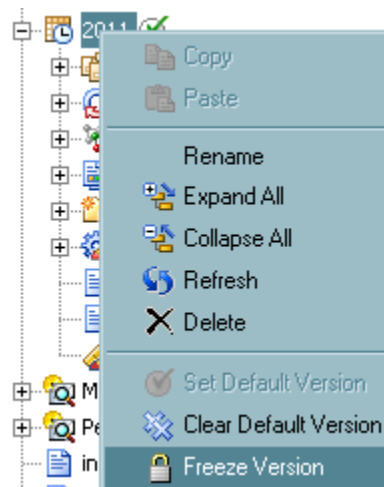
Freeze a Version


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 **Date Frozen** is today's date.

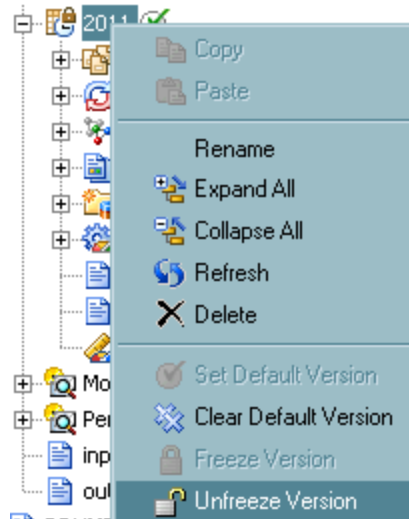
Note: To document the reasons or assumptions for freezing the version, use the version **Notes** tab. SAS Model Manager automatically annotates the **History** tab.


Unfreeze a Version

If changes to a model are required after the version is frozen, then a SAS Model Manager administrator can 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 **Date Frozen** 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 action 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 198
- “Clear a Default Version” on page 200

See Also

“Exporting Models” on page 208

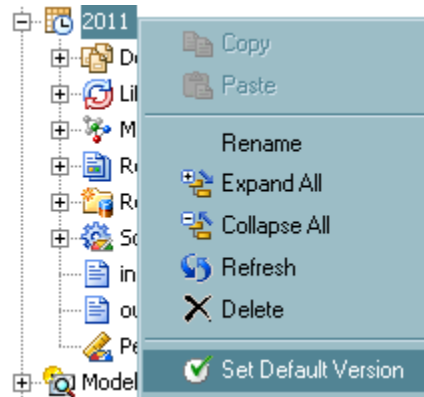
Set a Default Version

To set the default version for a project, follow these steps:

1. Right-click the version folder .

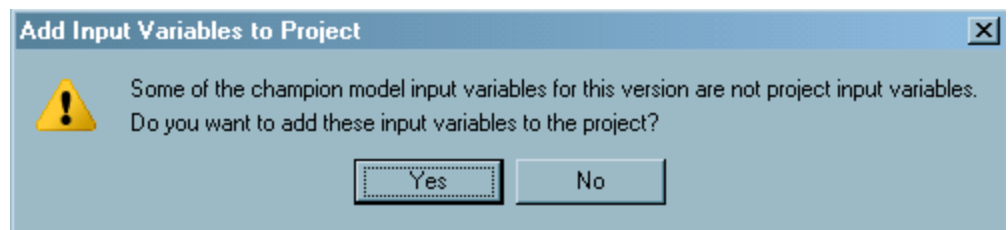
Note: The version must have a champion model assigned.

2. Select **Set Default Version** from the pop-up menu.



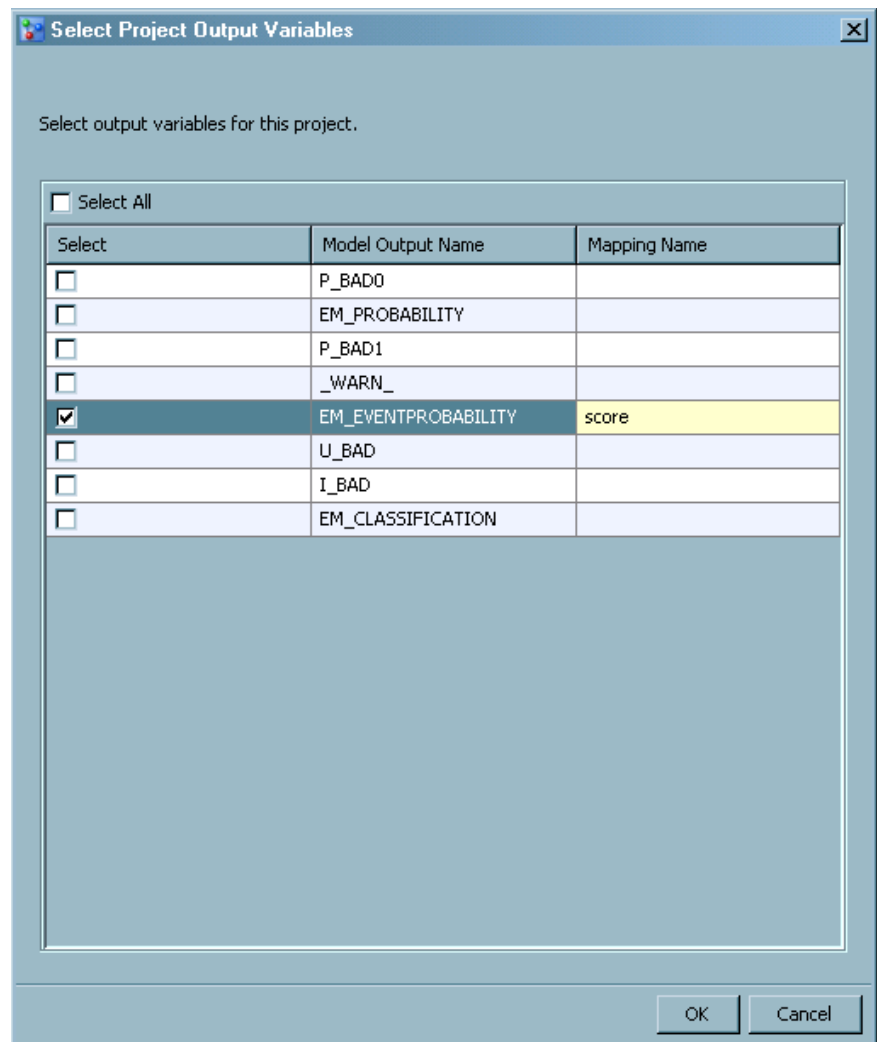
3. Click **Yes** to confirm.


If the project inputvar.xml and outputvar.xml files have not been defined, the Add Variables to Project message window appears.



Follow these steps to define the inputvar.xml and outputvar.xml files.


- a. Click **Yes** in the Add Variables to Project message window.
- b. SAS Model Manager uses the variables in the champion model as the prototype for the inputvar.xml file. In the Select Project Output Variables window, check the box for each model variable that is to be mapped to a project output variable.




- c. For each checked model output variable, enter a project output variable name in the **Mapping Name** column. Click **OK**. SAS Model Manager creates **inputvar.xml** and **outputvar.xml** in the version folder.
4. Verify that the  icon appears beside the version folder.
5. Select the project folder to examine its properties. The value for **Date Modified** 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 **Date Modified** 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 13

Delivering Models

Overview of Model Delivery	203
Publishing Models	204
About Publishing Models	204
Publish a Model to a Channel	205
Extract a Published Model	207
Exporting Models	208
About Exporting Models	208
Export a Model	208
Export a Project Champion Model	209
Verify the Model Export	210
Publish Scoring Functions	210
What Is a Scoring Function?	210
Scoring Function Process Flow	212
Prerequisites for Publishing a Scoring Function	214
Publish Scoring Function Field Descriptions	214
How to Publish a Scoring Function	217
Log Messages	221
Scoring Function Metadata Tables	222

Overview of Model Delivery

SAS Model Manager provides a comprehensive publishing environment for model delivery that supports sharing life cycle and performance data. SAS Model Manager publishes models to different channels, and exports champion models to the SAS Metadata Repository. SAS Model Manager can also publish scoring functions for predictive (classification or prediction) models and segmentation models to a database. 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 who subscribe to a 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. From a publication channel, 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 the SAS Publishing Framework to publish models to defined channels. The SAS Publishing Framework notifies subscribers of the publication channel when the models are delivered. You can publish models from the organizational, project, version, or model folder in the Project Tree.

SAS Model Manager creates a SAS package (SPK) file for the model in a publication channel. A user who subscribes to the publication channel can choose to receive e-mail that includes the SAS package as an attachment.

Note: To deploy a model to a publications channel, in SAS Management Console the publication channel must be configured by a SAS Model Manager administrator to publish models as archive (binary .SPK) files to a persistent store location. The archive persistent store location 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	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 inputvar.xml
sas002.ref	URL address of outputvar.xml
sas003.ref	URL address of score.sas

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 folder/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 205
- “Extract a Published Model” on page 207

Publish a Model to a Channel

To publish a model to a channel, follow these steps.

1. Right-click the project, version, or models folder that contains the model that you want to publish and select **Publish** from the pop-up menu. The Publish to a SAS Channel window appears.

Note: To publish a model, you must specify that the project contains a version folder with at least one model.

Channel: MMChannel

Description: The Model Manager channel

Subject: Model Manager

Subscribers:

Select Entries to Publish:

Select	ID	Name	Version	Type	Cham...
<input checked="" type="radio"/>	MMRoot/H...	Neural	2011	Classificati...	NO
<input type="radio"/>	MMRoot/H...	Loan	2011	Classificati...	NO
<input type="radio"/>	MMRoot/H...	Reg1	2011	Classificati...	NO

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 appears.

Publish to a SAS Channel

Message Subject:

Notes:

Add User-Defined Property:

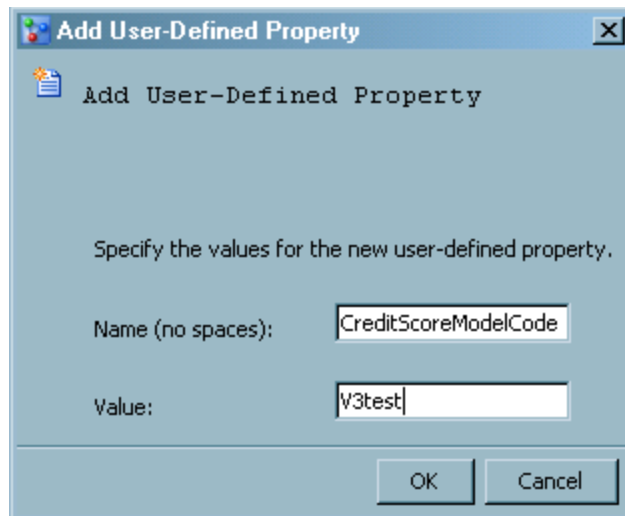
Name	Value
<input type="text"/>	<input type="text"/>

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, 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 that are 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 **Close**.
10. Update the specific properties for the appropriate milestone task in the **Life Cycle** node.
11. Add a user-defined property to identify the publication channel:
 - a. Select the project. Right-click in the Properties view and select **Add User-Defined Property**.
 - b. In the **Name** field, type in a property name that depicts the channel name that was used to export the project, such as `PublishedToChannel`.
 - c. In the **Value** field, enter the channel that was used to export the project. Click **OK**.

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 more information, see [“Extracting the Champion Model from a Channel” on page 268](#).

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, the scoring application always uses the most current champion model.

Note: SAS Model Manager cannot export models whose **Score Code Type** model property is set to **SAS Program**.

SAS Model Manager uses the SAS Folders 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.

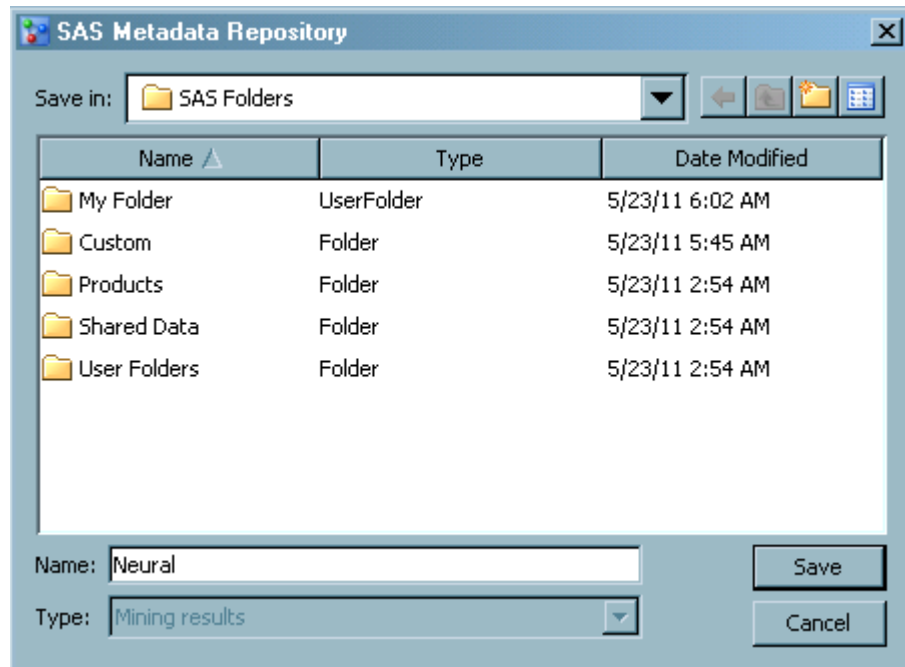
These are the tasks that you perform to export models:

- [“Export a Model” on page 208](#)
- [“Export a Project Champion Model” on page 209](#)
- [“Verify the Model Export” on page 210](#)

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 SAS Metadata Repository window appears. 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 as the object, 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 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 the project that 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. If the project is not locked, you are prompted to confirm exporting the champion model. The SAS Metadata Repository window appears. SAS Model Manager displays the metadata folders.
3. Select a folder to which you want to export the model.

4. Click **OK**. If a MiningResult object is in the repository that has the same name as the object, 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 folder to which 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 appears.
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 303](#).

5. Examine the metadata on the **Advanced** tab to determine when the MiningResult object was created or most recently updated.
6. Click **OK**.

Publish Scoring Functions

What Is a Scoring Function?

The **Publish Scoring Function** of SAS Model Manager enables you to publish models that are associated with the **Data Step** score code type to a configured database. When you publish a scoring function for a project, SAS Model Manager exports the project's champion model to the SAS Metadata Repository. The SAS Scoring Accelerator then creates scoring functions in the default version that can be deployed inside the database based on the project's champion model score code. The scoring function is validated automatically against a default train table to ensure that the scoring results are correct. A scoring application or SQL code can then execute the scoring functions in the database. The scoring functions extend the database's SQL language and can be used in SQL statements like other database functions.

The Scoring Function metadata tables are populated with information about the project and pointers to the scoring function. This feature 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.

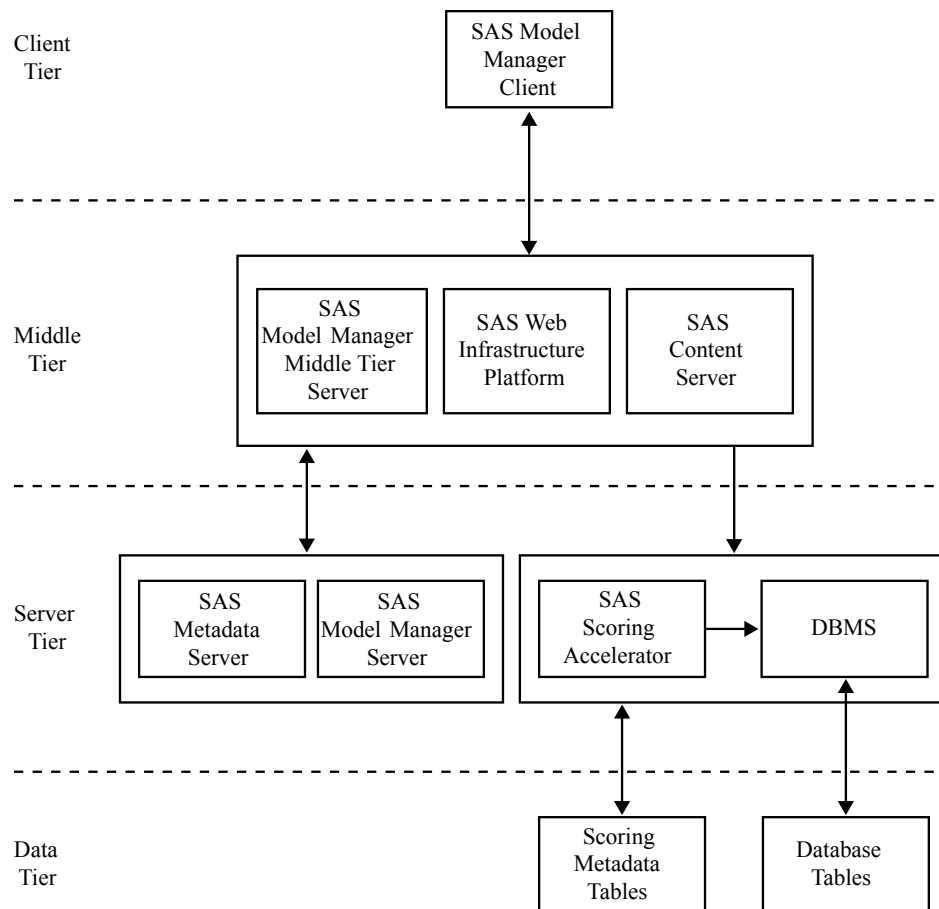
The Publish Scoring Function also creates a MiningResult metadata object that is stored in the SAS Metadata Repository. A typical use of a MiningResult object is to serve input and output metadata queries made by scoring applications at the design time of application development.

For more information about the SAS Scoring Accelerator, see the [SAS In-Database Technology](http://support.sas.com) page on <http://support.sas.com>.

Note: For more information about the prerequisites before publishing a scoring function, see “[Prerequisites for Publishing a Scoring Function](#)” on page 214.

Here is a diagram that represents the relationship between SAS Model Manager and SAS Model Manager In-Database Support.

Figure 13.1 The Relationship between SAS Model Manager 3.1 and SAS Model Manager In-Database Support



Here are descriptions of the diagram'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.

SAS Model Manager Middle Tier Server

The SAS Model Manager Middle Tier Server is a collection of services that are hosted by an application server that orchestrates the communication and movement of data between all servers and components in the SAS Model Manager operational environment.

SAS Web Infrastructure Platform

The SAS Web Infrastructure Platform (or WIP) is a collection of middle tier services and applications that provides basic integration services. It is delivered as part of the Integration Technologies package. As such, all Business Intelligence applications, Data Integration applications, and SAS Solutions have access to the Web Infrastructure Platform as part of their standard product bundling.

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 Model Manager Server

The SAS Model Manager Server is a collection of macros on the SAS Workspace Server that generate SAS code to perform SAS Model Manager tasks.

SAS Scoring Accelerator

The SAS Scoring Accelerator creates scoring functions that can be deployed inside a database. Scoring functions are based on the project's champion model score code.

DBMS

The relational databases in the database management system (DBMS) serve as output data sources for SAS Model Manager.

Scoring Metadata Tables

These tables contain metadata, and the tables serve as data sources in SAS Model Manager.

Database Tables

These database tables in relational databases serve as data sources for a scoring application.

Scoring Function Process Flow

This is an example of the process flow to publish a scoring function. For more information, see [“How to Publish a Scoring Function” on page 217](#).

1. From SAS Model Manager, you select the Publish Scoring Function for the project that contains the champion model that they want to publish to a specific database. For more information, see [“How to Publish a Scoring Function” on page 217](#).
2. After you complete all the required information about the Publish Scoring Function, SAS Model Manager establishes a JDBC connection to the database using the credentials that were entered. The user-defined part of the scoring function name is validated against the target 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 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 SAS Model Manager publishing macro is called, which performs the following tasks:
 - calls the SAS Model Manager transform macro that creates a metadata XML file. This XML file is used by the model publishing macro.
 - calls the SAS model publishing macro, which creates the files that are needed to build the scoring functions and publishes the scoring functions with those files to the specified database.
 - 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 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 222.
 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 Scoring Function window.
- Note:* The user-defined part of the function name is used as the default 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 Scoring Function

The following prerequisites must be completed before users can publish a scoring function:

- The user must have the proper authorization to publish approved models from SAS Model Manager to the database 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) or segmentation model must have been selected for production scoring deployment via SAS Model Manager.

Note: SAS Model Manager cannot publish scoring functions for PMML models. A model is supported when the score code is a SAS DATA step fragment and has been created by SAS Enterprise Miner.

- A SAS metadata folder must have been created for models that are to be used for SAS In-Database scoring. The folder name is created in SAS Management Console.

Note: the SAS metadata folder is created via SAS Management Console.

- (Optional) A project user-defined property **DbmsTable** is defined for the default version of the SAS Model Manager project from which to publish the scoring function.

Note: The **DbmsTable** property must be defined if you plan to use a scoring application or SQL code to score your model.

- The JDBC driver must be accessible from the SAS Model Manager middle-tier server.
- A database must have been configured to install 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 Scoring Function Field Descriptions

Here is a list of the field names and descriptions for the Publish Scoring Function window.

SAS metadata location

specifies a folder location in the SAS Metadata Repository.

Function name

specifies the name of the scoring function, which includes a prefix and a user-defined value. The prefix is 11 characters long and is 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 and ranges from 01 to 12.
- **dd** is the day and ranges from 01 to 31.
- **nnn** is a counter that increments by 1 each time that a scoring function completes 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 Scoring Function** menu option. An example of a function name is **Y081107001_user_defined_value**. Here are the naming convention requirements:

- The user-defined value is case insensitive. The maximum length of alphanumeric characters is determined by the database type that is selected. No spaces are allowed. An underscore is the only special character that can be included in the function name.
 - 19 alphanumeric characters for Teradata
 - 117 alphanumeric characters for Netezza and DB2
 - 52 alphanumeric characters for Greenplum

UNIX Specifics

The user-defined portion of the function name in an AIX environment has a maximum length of 16 alphanumeric characters for Teradata.

- The user-defined value part of the scoring function name is used by default for subsequent use of the Publish Scoring Function from the same project. Only the user-defined value in the scoring function name can be modified.

Database type

specifies the type of database that the scoring function will be published to.

Database server

specifies the name of the server where the database 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**.

Server user ID (DB2 only)

specifies the user ID for SAS SFTP. This value enables you to access the machine on which you have installed the DB2 database. If you do not specify a value for **Server user ID**, the value of **User ID** is used as the user ID for SAS SFTP.

Schema (Greenplum only)

specifies the schema name for the database. The schema name is owned by the user that is specified in the **User ID** field. The schema must be created by your database administrator.

Initial wait time (DB2 only)

specifies the initial wait time in seconds for SAS SFTP to parse the responses and complete the SFTP –batch file process.

Default: 15 seconds

FTP time out (DB2 only)

specifies the time-out value in seconds if SAS SFTP fails to transfer the files.

Default: 120 seconds

Compile database (Netezza only)

specifies the name of the database where the SAS_COMPILEUDF function is published.

Default: SASLIB

See Also: For more information about publishing the SAS_COMPILEUDF function, see the *SAS In-Database Products: Administrator's Guide*.

Jazlib database (Netezza only)

specifies the name of the database where the SAS 9.3 Formats Library for Netezza is published.

Default: SASLIB

Restriction: This argument is supported only for TwinFin systems.

Options

Validate scoring results

specifies to validate the scoring results when publishing the 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 database table. Scoring is performed using the new scoring function and database 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. Saving the scoring function is useful for debugging if the scoring function validation fails.

Use champion model input

specifies to use the champion model input when publishing the scoring function instead of using the project input, which is the default. This is useful when the project input variables exceed the limitations for a database.

Note: Here are the limitations for the number of project input variables when publishing a scoring function for a champion model to a database:

- 128 input variables for Teradata
- 64 input variables for Netezza
- 100 input variables for Greenplum
- 90 input variables for DB2

Protected mode (Teradata only)

specifies the mode of operation to use for the Publish 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 database processing to fail. You should run the Publish Scoring Function in protected mode during validation. When the model is ready for production, you can run the Publish Scoring Function in unprotected mode. You might see a significant performance advantage when you run the Publish Scoring Function in unprotected mode.

Fenced mode (DB2 only)

specifies the mode of operation to use for the Publish Scoring Function. There are two modes of operation, fenced and unfenced. You specify the mode by selecting or deselecting the **Fenced mode** option. The default mode of operation is fenced. Fenced mode means that the macro code is isolated in a separate process from the DB2 database, and an error does not cause database processing to fail. You should run the Publish Scoring Function in fenced mode during validation. When the model is ready for production, you can run the Publish Scoring Function in unfenced mode. You might see a significant performance advantage when you run the Publish Scoring Function in unfenced mode.

Display detailed log messages

provides detailed information, which includes warnings and error messages that occur when you publish a scoring function.

Sample size

specifies the size of the sample to use for validating the scoring function. The default value is 100. The maximum number of digits that are allowed is 8.

How to Publish a Scoring Function

To publish a 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 198](#) or [“Set a Champion Model” on page 195](#).
2. (Optional) Select the project name and enter a value for the **DbmsTable** user-defined property.

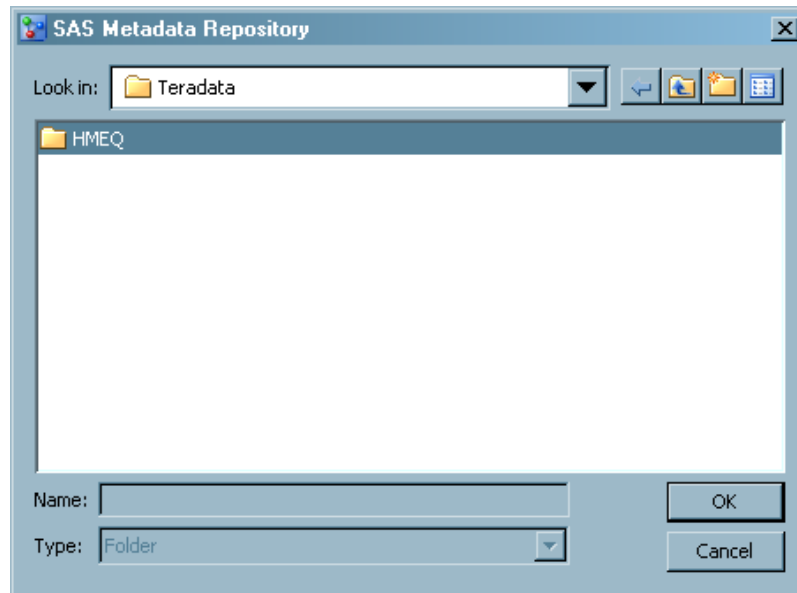
Note: If you plan to use scoring application or SQL Code to score this project, you must first set the **DbmsTable** property to the name of input table in your database that you want to use for scoring the champion model. When you publish a scoring function the information that is associated with the input table in the database is updated to contain the value of the **DbmsTable** property. The scoring application or SQL code can then query the database for the input table name to use as the scoring input table.

For more information, see [“User-Defined Properties” on page 362](#).

User-Defined Properties	
BusinessContext	
KeyType	
ScoringInputTable	
DbmsTable	
PreCode	
TableKey	

3. Right-click the project's name in the Project Tree and select **Publish Scoring Function**. The Publish Scoring Function window appears.

4. To select a SAS metadata Location in which to publish the model, click **Browse**, select a folder name, and then click **OK**.



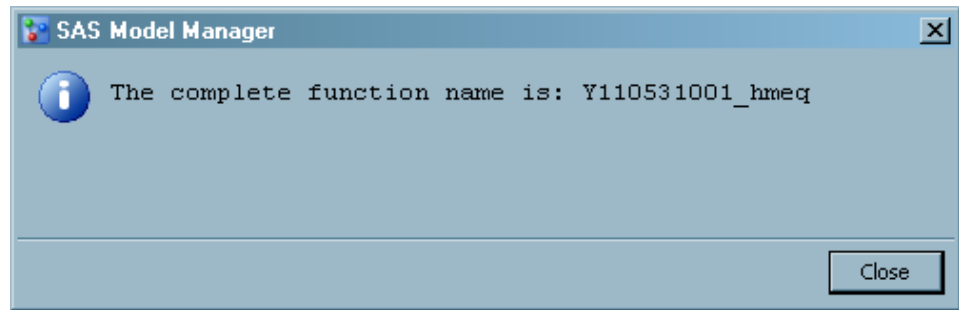
5. Enter a name for the scoring function. The function name includes a prefix and a user-defined value. Only the user-defined value can be modified. The naming conventions are the following:
 - The user-defined value must be unique across all projects.
 - The maximum length for the user-defined value is determined by which database type is selected, and no spaces are allowed. It is recommended that you limit the user-defined value to 20 characters or fewer. Here are the limitations for the number of characters for each database:
 - 19 alphanumeric characters for Teradata
 - 117 alphanumeric characters for Netezza and DB2
 - 52 alphanumeric characters for Greenplum
 - 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 362](#).
6. Select a database type from the drop-down list. The type of database that you choose determines which fields are required.
7. Enter a text value for the following fields:
 - **Database server**
 - **Database**
 - **User ID**
 - **Password**
 - **Server user ID** (DB2 only)
 - **Schema** (Greenplum only)
 - **Compile database** (Netezza only)
 - **Jazlib database** (Netezza only)
8. (DB2 only) Enter a numeric value for the fields that appear for the database type:

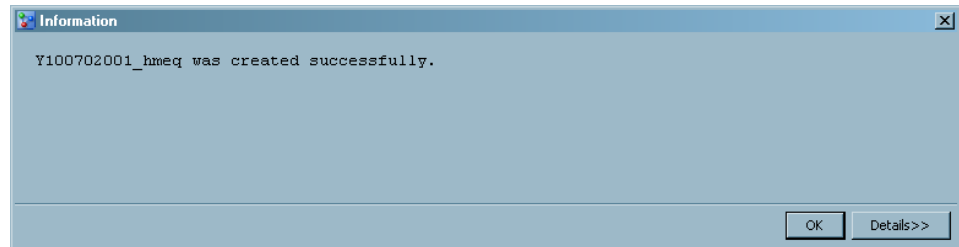
- **Initial wait time** (in seconds)
 - **FTP time out** (in seconds)
9. In the **Options** section, select the check box for the desired validation options that appear for the selected database type:
- **Validate scoring results**
 - **Keep scoring function if validation fails**
 - **Use champion model input**
 - **Protected mode** (Teradata only) or **Fenced mode** (DB2 only)
 - **Display detailed log messages**
- Note:* By default, the **Validate scoring results**, and **Protected mode** or **Fenced mode** options are selected. The **Keep scoring function if validation fails** option is available for selection only when **Validate scoring results** is selected.
10. Enter a numeric value for **Sample Size**. The default value for sample size is 100 if the value is null or zero. The maximum number of digits that are allowed is 8.

11. Click **OK**. A message is displayed that contains the scoring function name that will be published to the database.

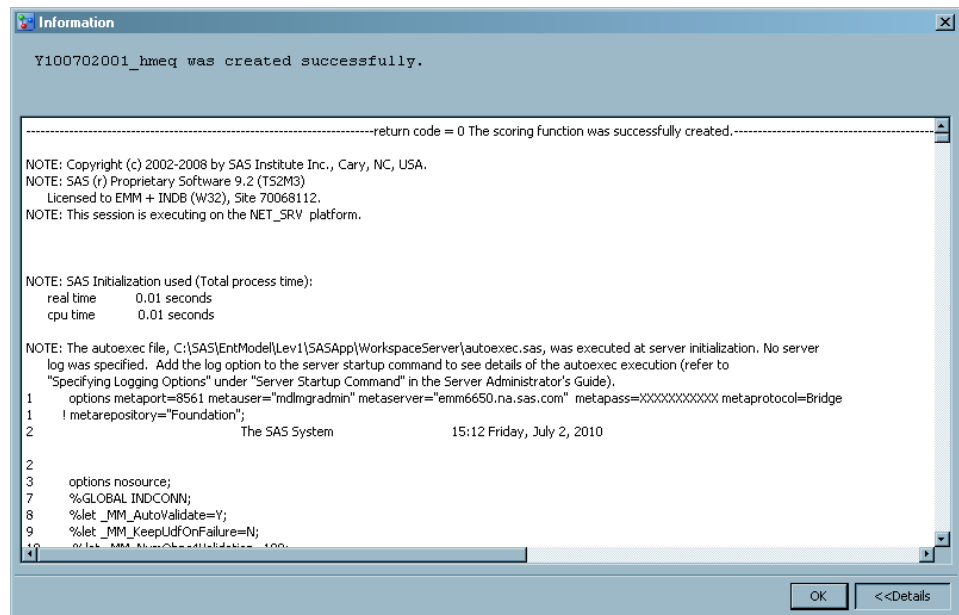
Note: The user-defined value of the **Function Name** is validated against the target database. If the user-defined value has already being used by another project to publish a model to the target database, an error message is displayed.



12. Click **OK**. A message indicating that the scoring function has been created successfully or unsuccessfully is displayed.



13. Click **Details** to display information about the publish scoring function actions or error messages.



14. Click **OK** to complete the publishing process.

Log Messages

A user can view the log file in the Details window when publishing a scoring function. As an alternative, a user can view the log file in the ScoringFunction.log file in the Project Tree. The log file is named ScoringFunction.log. The time that the process started, who initiated the process, and when the project was published are recorded. Error messages are also recorded in the log file. The log file provides an audit trail of all relevant actions in the publishing process.

Scoring Function Metadata Tables

The following tables are created in the database when you are 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 that occurred, 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 14

Replacing a Champion Model

Overview of Replacing a Champion Model	223
Select a New Default Version	224
Retire a Project	224

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. You might also 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 was 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 that was 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, ensure that milestone tasks are based on the new champion model and not on 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 or retire the project. 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 **State** property, and select **Retired**.

Note: The **State** property is not set to **Retired** when you sign off on the milestone action **RetireChampion** in the life cycle for the version. You must set the state for the project manually.

Part 5

Performance Monitoring and Retraining Models

<i>Chapter 15</i>	
What Are Performance Monitoring Reports?	227
<i>Chapter 16</i>	
Create Reports by Defining a Performance Task	241
<i>Chapter 17</i>	
Create Reports Using Batch Programs	251
<i>Chapter 18</i>	
Using Dashboard Reports	279
<i>Chapter 19</i>	
Retraining Models	289

Chapter 15

What Are Performance Monitoring Reports?

Overview of Performance Monitoring Reports	227
Types of Performance Monitoring Reports	228
Overview of the Types of Performance Monitoring Reports	228
Summary Report	229
Data Composition Reports	230
Model Monitoring Reports	231
Performance Index Warnings and Alerts	234
The Process of Monitoring Champion Models	236
Formatting Performance Monitoring Reports	237
About Monitoring Reports	237
Create Monitoring Reports	238
Monitoring Report Output Files	239
View Reports	240

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.

Note: The performance monitoring reports are designed to work only with classification models that contain a binary target.

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 several charts:

- Lift Chart
- Gini - ROC* Chart
- Gini - Trend Chart
- KS Chart

* receiver operating characteristic

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 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 see the frequency with which 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 see the frequency with which 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, 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 that are 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 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:

$$\text{Deviation_Index} = \sum (\% \text{Actual} - \% \text{Expected}) * \ln (\% \text{Actual} / \% \text{Expected})$$

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.

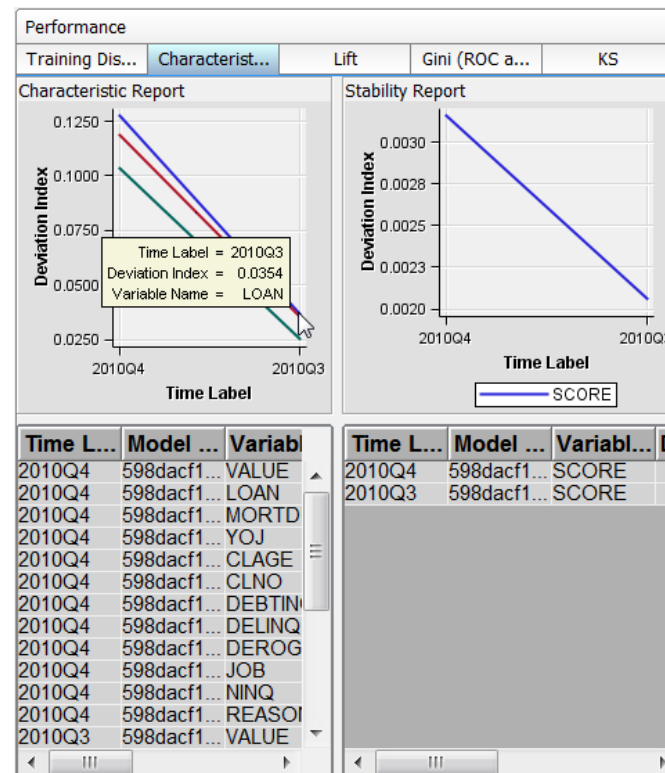
Stability Report

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 0.30 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

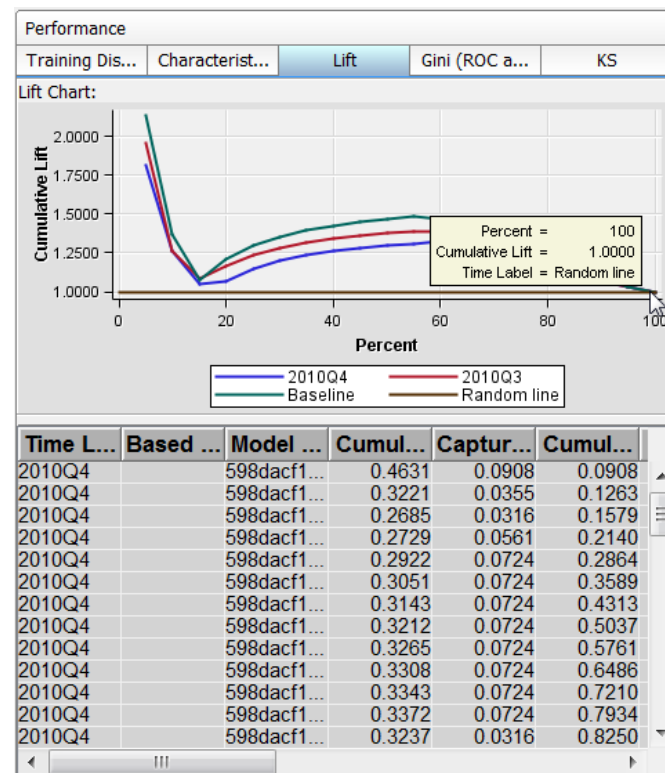
Monitoring Lift Report

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 report 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 report can show 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 report is contained in the report file mm_lift.sas7bdat in the **Resources** folder.

Here is an example of a monitoring Lift report. By placing the cursor over a point in the report, you can view the data for that point.



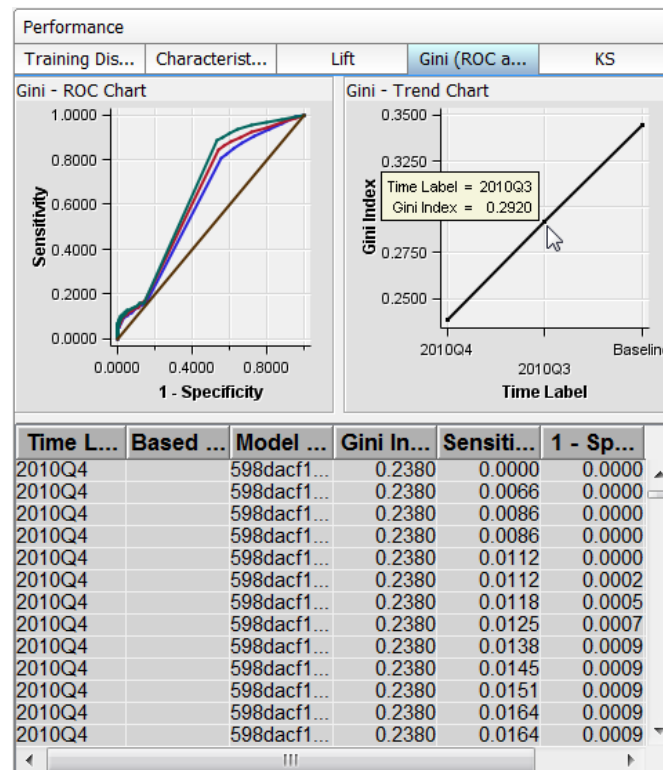
Monitoring Gini (ROC and Trend) Report

The monitoring **Gini (ROC and Trend)** reports show you 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 **Gini (ROC and Trend)** report to detect degradations in the predictive power of a model.

The data that underlies the monitoring **Gini (ROC and Trend)** report is contained in the report component file mm_roc.sas7bdat.

The following chart is an example of a monitoring **Gini (ROC and Trend)** report. By placing the cursor over a point in the chart, you can view the data for that point.



KS Report

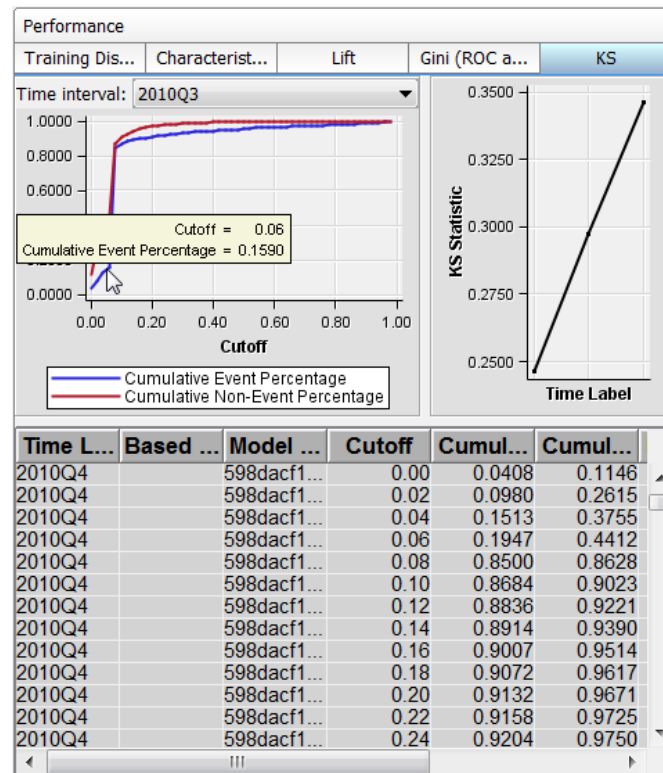
The KS report contains the Kolmogorov-Smirnov (KS) test plots for models with a binary target. The KS statistic measures the maximum vertical separation, or deviation between the cumulative distributions of events and non-events. This trend report uses a summary data set that plots the KS statistic and 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:

```
alertConditon='alert-condition';
warningCondition='warning-conditon';
```

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. Here 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. Here is an example of alert and warning thresholds:

```
alertCondition='outputDeviation>0.03';
warningCondition='outputDeviation>0.01';
```

Model Assessment reports

For the Lift, Gini (ROC and Trend), 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.

Here 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 228](#)

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 by creating and executing a performance task, 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.	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.
Ensure access to the champion model	This process is performed by the Define Performance Task wizard. No action is necessary.	Run the <code>%MM_GetModels()</code> 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.

Task	Reports Created by Using the Define Performance Task Wizard	Reports Created Using SAS Programs Run in Batch
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.
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 monitoring reports using the New Reports Wizard, the report creates the following charts:

Assessment Charts

Assessment charts summarize the utility that one can expect by using the respective models, as compared to using only baseline information. Assessment charts can present a model's lift at a given point in time or the sequential lift performance of a model's lift over time. A 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.

Gini - ROC Chart

Sensitivity is the proportion of true positive events and specificity is the proportion of true negative events. The Gini - ROC Chart plots Sensitivity on the Y axis and 1 - Specificity on the X axis.

Gini - Trend

When the Gini - 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, and these are 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 a 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 173](#).

These are the tasks that you perform for monitoring reports:


- [“View Reports” on page 176](#)

See Also

- [Chapter 16, “Create Reports by Defining a Performance Task,” on page 241](#)
- [Chapter 17, “Create Reports Using Batch Programs,” on page 251](#)

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 **Type** list box.

New Report Wizard

New Report Wizard

Report Options

Type: Monitoring Report

Format: PDF

Select Models

Sel...	ID	Name	Ve...	Type	C...

Report Properties

Property	Value
[-] General Properties	
Name *	Monitoring_D2011-05-06T14.44
Description	Monitoring

OK Cancel

4. In the **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 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 176](#)

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** table.

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

- the report in either HTML, PDF, RTF, or Excel format
- taskCode.log
- 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.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 performance monitoring reports in the SAS Model Manager window, select the **Performance** node. The right pane displays a view in which each tab is one of the reports. Click 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 237](#).

Chapter 16

Create Reports by Defining a Performance Task

Overview of Creating Reports Using a Performance Task	241
Creating Reports Using a Performance Task	241
Determine How to Use the Performance Data Sets	242
Prerequisites for Running the Define Performance Task Wizard	243
Overview of Prerequisites	243
Ensure That the Champion Model and Default Version Are Set	244
Ensure That the Champion Model Is a Classification Model That Contains a Binary Target	244
Ensure That the Performance Data Source Is Available	244
Ensure That Project Model Properties Are Set	244
Map Model and Project Output Variables	245
Run the Define Performance Task Wizard	245

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 registered using SAS Management Console.
- 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 registered in the SAS Metadata Repository by using SAS Management Console. For each SAS Model Manager project, you can set up your environment to use the performance data source that best suits your business process. Here are two methods of collecting performance data:

- **Method 1:** You periodically take a snapshot of an operational data set to create a performance data set. Each time you take a snapshot, you give the performance data set a new name. Each performance data set must be registered in SAS Management Console. For each time interval, you name a new performance data source when you run the Define Performance Task wizard.
- **Method 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 each performance data set every time you take a snapshot. You register the performance data set with SAS Management Console 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.

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	Method 1: The Performance Data Set Name Changes	Method 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"> • Jul11 • Aug11 • Sep11 • Oct11 • Nov11 Dec11 	Every month, take a snapshot of the operational data and name the performance data set using the same name: 2011perf
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	Method 1: The Performance Data Set Name Changes	Method 2: The Performance Data Set Name Remains Static
Modifications to make in the Define Performance Task wizard	In Step 3, select a performance data source, select a data collection date, and a date label.	In Step 3, select a data collection 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 champion model is a classification model that contains a binary target.
- Ensure that the champion model contains a score.sas file. If the performance data set contains the predicted values, the score.sas file can be empty. For more information, see [“Monitoring Performance of a Model without Score Code” on page 397](#).
- Ensure that the performance data sets for the time period that you want to monitor are registered in SAS Management Console.
- Ensure that the appropriate project and model properties are set.
- Ensure that the model output variables are mapped to the project output variables.



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.

Ensure That the Champion Model Is a Classification Model That Contains a Binary Target

Performance monitoring reports are valid only for classification models that contain a binary target. You should define only performance tasks for classification models. The champion model must have a function type of classification and must contain a binary target. From the Projects perspective, select the champion model name and verify that the **Function** property in the specific properties section is set to **Classification**. For models that are created using SAS Enterprise Miner, select the **targetvar.xml** file in the model folder and verify that the LEVEL attribute is set to **BINARY**.

Ensure That the Performance Data Source Is Available

The Define Performance Task wizard requires that the performance data be registered in the SAS Metadata Repository, using SAS Management Console. After the performance data set is registered, update the **Default Performance Table** model property. From the Projects perspective, select the champion model name. Click the value field of the **Default Performance Table** property, click the ellipsis button, and select a performance data set from the Select Table window.

If your performance table is not available for selection, 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 3.1 Administrator's Guide*.

See Also

[“Project Tables” on page 29](#)

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 **Class Target Level** 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 and click the **Edit** button.
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

Input and Output Variables

Select one or more input and output variables for analysis.

Step 1 of 3

Output Variables for Stability Analysis


Select	Keep Variables	Description
<input checked="" type="checkbox"/>	score	

Input Variables for Characteristic Analysis

Select	Keep Variables	Description
<input checked="" type="checkbox"/>	YOJ	
<input checked="" type="checkbox"/>	MORTDUE	
<input checked="" type="checkbox"/>	REASON	
<input checked="" type="checkbox"/>	DEROG	

2. In the **Output Variables for Stability Analysis** table, select the output variable or variables. To select all output variables, click **Select All**.
3. In the **Input Variables for Characteristic Analysis** table, select the input variables. To select all input variables, click **Select All**. Click **Next**.
4. For each general property in the 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

 **Warning and Alert Conditions**

Specify values for the warning and alert conditions, or use the default values.

Step 2 of 3

Property	Value
<input type="checkbox"/> 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 condit... *	(lift5Decay>0.15 and lift10Decay>0.12) or giniDecay...
Model assessment warning co... *	lift5Decay>0.05

- Click the ellipsis button  for the **Performance data source** box and select a performance data source.

Define Performance Task

Other Reporting Specifications

Specify the report data specifications and e-mail notifications.

Step 3 of 3

Performance data source: HMEQ.HMEQ_2010Q2

Data collection date: May 10, 2011


Date label:

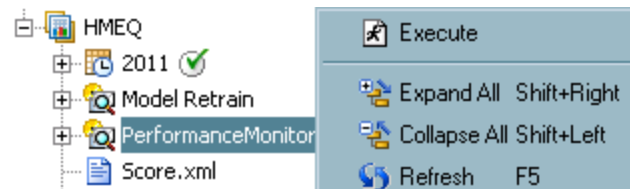
E-mail Notifications

E-mail Address	Send Alert Warning	Send Job Status
mdlmgadm@mycompany.com	Yes	Yes

Add Delete

Back Next Finish Cancel Help

6. Click the ellipsis button  for the **Data collection 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.
7. 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.
8. (Optional) To send the scoring results by e-mail, click the **Add** button in the **E-mail to** table. The Add Contact window appears.
 - a. Type an e-mail address.
 - b. 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 displays a confirmation 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.

12. To view the reports, click on the **Performance** node. On the right, click the tab for the report that you want to view.

SAS Model Manager

File Edit View Tools Help

Repository

Project Tree

- HMEQ
 - 2011 ✓
 - Documents
 - Life Cycle
 - Models
 - Loan
 - Neural
 - Reg1 ✓
 - Tree1
 - Reports
 - Resources
 - Scoring
 - inputvar.xml
 - outputvar.xml
 - Performance
 - Model Retrain
 - PerformanceMonitor
 - outputvar.xml

Annotations - Performance

Performance

Training Distribution Characteristic and Stability Lift Gini (ROC and Trend) KS

Characteristic Report

Deviation Index

Time Label

2010Q3 2010Q4 2011Q1

— VALUE — LOAN — MORTDUE — YOJ

Stability Report

Deviation Index

Time Label

2010Q3 2010Q4 2011Q1

— SCORE

Time La...	Model U...	Variable...
2010Q3	dff8d220-0...	VALUE
2010Q3	dff8d220-0...	LOAN
2010Q3	dff8d220-0...	MORTDUE
2010Q3	dff8d220-0...	YOJ
2010Q3	dff8d220-0...	CLAGE
2010Q3	dff8d220-0...	CLNO
2010Q3	dff8d220-0...	DEBTINC
2010Q3	dff8d220-0...	DELINQ
2010Q3	dff8d220-0...	DEROG
2010Q3	dff8d220-0...	JOB
2010Q3	dff8d220-0...	NINQ
2010Q3	dff8d220-0...	REASON

Time La...	Model U...	Variable.
2010Q3	dff8d220-0...	SCORE
2010Q4	dff8d220-0...	SCORE
2011Q1	dff8d220-0...	SCORE

SAS Model Manager Administrator vmwwin7084.na.sas.com

Chapter 17

Create Reports Using Batch Programs

Overview of SAS Programs to Monitor Model Performance	252
Prerequisites for Running Batch Performance Reports	253
Overview of Prerequisites for Running Batch Performance Reports	253
Publish the Champion Model from the Project Folder	253
Create a Folder Structure	253
Obtain Performance Data	255
Determine the Publish Channel	255
Copy Example Batch Programs	255
Determine SAS Model Manager User ID and Password	256
Report Output in Test and Production Modes	256
Report Output in Test Mode	256
Report Output in Production Mode	256
Define the Report Specifications	257
Overview of Code to Define Report Specifications	257
Required Libref	257
Project Specifications	257
E-mail Recipient Specifications	260
Report Specifications	261
Job Scheduling Specifications	264
Example Code to Create the Report Specifications	265
Extracting the Champion Model from a Channel	268
Using the %MM_GetModels() Macro	268
Accessing SAS Model Manager Report Macros	269
%MM_GetModels() Macro Syntax	269
Example Program to Extract a Model from a Channel	269
The current.sas7bdat Data Set	269
SAS Code to Run Performance Reports	271
Overview of the SAS Code to Run the Performance Reports	271
Accessing SAS Model Manager Report Macros	271
Required Librefs	271
Macro Variables to Define Report Local Folders and Data Sets	272
Macro Variables That Are Used by the %MM_RunReports() Macro	272
The DATA Step to Access the Performance Data Set	274
The %MM_RunReports() Macro	274
Example Code to Run the Reports	275

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 publishing channel. Any batch job 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 operating environment and the performance data set. This section 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 champion model from the publishing channel to the local computer using the `%MM_GetModels()` macro. 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 the following performance monitoring macros:

- `%MM_GetModels()` extracts models from a publishing channel.
- `%MM_UpdateCharacteristicTable` creates a Characteristic report.
- `%MM_UpdateStabilityTable` creates a Stability report.
- `%MM_UpdateAssessmentTable` creates model monitoring reports.
- `%MM_RunReports()` sets the operating environment and runs the macros to create the reports.

Note: The macros are in the `modelmgr.sas7bcat` file. The location of this file for Windows is `\sasinstalldir\SASFoundation\9.3\mmcommon\sashelp`. The default value for `sasinstalldir` in Windows is `C:\Program Files\SASHome`. The location of this file for UNIX is `/sasinstalldir/SASFoundation/9.3/sashelp`. The default value for `sasinstalldir` in UNIX is `/usr/local/SASHome`.

SAS provides 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.

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 256](#)

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 35](#)

Determine the Publish Channel

You can determine the channel that was used to publish the model by using one of these methods:

- Select the model and click on the **History** tab. Look for a publish model entry. In this example, the channel name is **MMChannel**: May 10, 2011 9:49:09 AM[mdlmgradmin] "Reg1" was published to "MMChannel(sas-oma://vmwwin7084.na.sas.com:8561/reposid=A5OW51K8/ITChannel;id=A5OW51K8.B8000001)" successfully.
- In 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**. 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: If the **Plug-ins** tab does not appear in your view of SAS Management Console, contact your SAS administrator.

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 253](#)

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:

<code>mm_jobs.project</code>	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.
<code>mm_jobs.emailaddr</code>	defines the e-mail addresses for the recipients of job status and the notification flags for alert and warning notifications.
<code>mm_jobs.reportdef</code>	defines the types of reports to create and the warning and alert conditions that are associated with those reports.
<code>mm_jobs.jobtime</code>	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 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\HMEQ";`

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\HMEQ\testOutput';`

See: [“Report Output in Test and Production Modes” on page 256](#)

`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 might 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 234.

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 234.

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 that 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 that 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 that 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 scheduledTime variable. If the scheduled time has passed, the report runs. If it has not passed, the performance data sets are not created.

Example: **scheduledTime='03Jun2011:23:59:00';**

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='2011Q4';`

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:8080/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='01OCT2010:23:59:59';time='2010Q3';output;
  scheduledTime='01JAN2011:23:59:59';time='2010Q4';output;
  scheduledTime='01APR2011:23:59:59';time='2011Q1';output;
  scheduledTime='01JUL2011:23:59:59';time='2011Q2';output;
  scheduledTime='01OCT2011:23:59:59';time='2011Q3';output;

run;

```

See Also

- [“Extracting the Champion Model from a Channel” on page 268](#)
- [“SAS Code to Run Performance Reports” on page 271](#)

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 269.

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=channelPathlocalPath=localModelPath);

channel=channelPath

specifies the path of the channel to extract the models from. To obtain the channel path, see “[Determine the Publish Channel](#)” on page 255. 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

Variable Name for the Project or Model Information	Description
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 257](#)
- [“SAS Code to Run Performance Reports” on page 271](#)

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 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\HMEQ";`

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\HMEQ\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. Here is an 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\HMEQ1;`

_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\HMEQ\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**.

For a description of the macro variables, see “SAS Model Manager Macro Variables” on page 351.

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_MulticastAddress

specifies the multicast address of the network server where the Web Infrastructure Platform is installed.

Example: `%let _MM_MulticastAddress=239.27.18.213;`

_MM_MulticastPort

specifies the multicast port number for the server that the Web Infrastructure Platform server listens for SAS Model Manager requests.

Example: `%let _MM_MulticastPort=8561;`

_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 273](#)

For a description of the macro variables, see [“SAS Model Manager Macro Variables” on page 351](#).

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;`

For a description of the macro variables, see [“SAS Model Manager Macro Variables” on page 351](#).

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_2010q4;
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 257](#).

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 273](#)

`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=03Oct2010:12:15:30`

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 257](#) and [“Extracting the Champion Model from a Channel” on page 268](#).

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_MulticastAddress=239.27.18.213;
%let _MM_MulticastPort=8561;
%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 2010Q4 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 01Jan2011 in order to trigger the report */
/* execution scheduled for the 2010Q4 interval. */
*****/

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2010Q4;
run;

%let currentTime=01Jan2011:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,

```



```

user=&_MM_User,
password=&_MM_Password);

/*****
/* Now, run the 2011Q1 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Apr2011 in order to trigger the report */
/* execution scheduled for the 2010Q1 interval. */
*****/

DATA scoreIn.hmeq0;
  set scoreIn.hmeq_2011q1;
run;

%let currentTime=03Apr2011:12:30:15;
%MM_RunReports(
  localpath=&_MM_JobLocalPath,
  currentTime=&currentTime,
  mapTable=&mapTable,
  user=&_MM_User,
  password=&_MM_Password);

/*****
/* Now, run the 2011Q2 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Jul2011 in order to trigger the report */
/* execution scheduled for the 2011Q2 interval. */
*****/

DATA scoreIn.hmeq0;
  set scoreIn.hmeq_2011q2;
run;

%let currentTime=03Jul2011:12:30:15;
%MM_RunReports(
  localpath=&_MM_JobLocalPath,
  currentTime=&currentTime,
  mapTable=&mapTable,
  user=&_MM_User,
  password=&_MM_Password);

/*****
/* Now, run the 2011Q3 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Oct2011 in order to trigger the report */
/* execution scheduled for the 2011Q3 interval. */
*****/

DATA scoreIn.hmeq0;
  set scoreIn.hmeq_2011q3;
run;

%let currentTime=03Oct2011:12:30:15;
%MM_RunReports(
  localpath=&_MM_JobLocalPath,
  currentTime=&currentTime,

```

```

mapTable=&mapTable,
user=&_MM_User,
password=&_MM_Password);

/*****
/* Now, run the 2011Q4 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Jan2012 in order to trigger the report */
/* execution scheduled for the 2011Q4 interval. */
*****/

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2011q4;
run;

%let currentTime=03Jan2012:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,
    user=&_MM_User,
    password=&_MM_Password);

```

See Also

- [“Define the Report Specifications” on page 257](#)
- [“Extracting the Champion Model from a Channel” on page 268](#)

Chapter 18

Using Dashboard Reports

Overview of Project Dashboard Reports	279
Define Dashboard Report Indicators	279
Generate Dashboard Reports	283
View Dashboard Reports	285

Overview of Project Dashboard Reports

The SAS Model Manager Dashboard can provide reports that show the overall state of all projects that are being monitored. The dashboard reports are produced from existing performance monitoring reports. For each project, a user can define dashboard report indicators that are then used to create the dashboard reports. The dashboard reports are not displayed through the SAS Model Manager user interface. Instead, use a browser window to view the dashboard reports that are located on the SAS Workspace Server. These reports are generated in HTML by SAS Model Manager.

Note: The dashboard reports can be defined and generated only by SAS Model Manager administrators and advanced users.

You must complete the following tasks:

- “Run the Define Performance Task Wizard” on page 245
- “Define Dashboard Report Indicators” on page 279
- “Generate Dashboard Reports” on page 283
- “View Dashboard Reports” on page 285

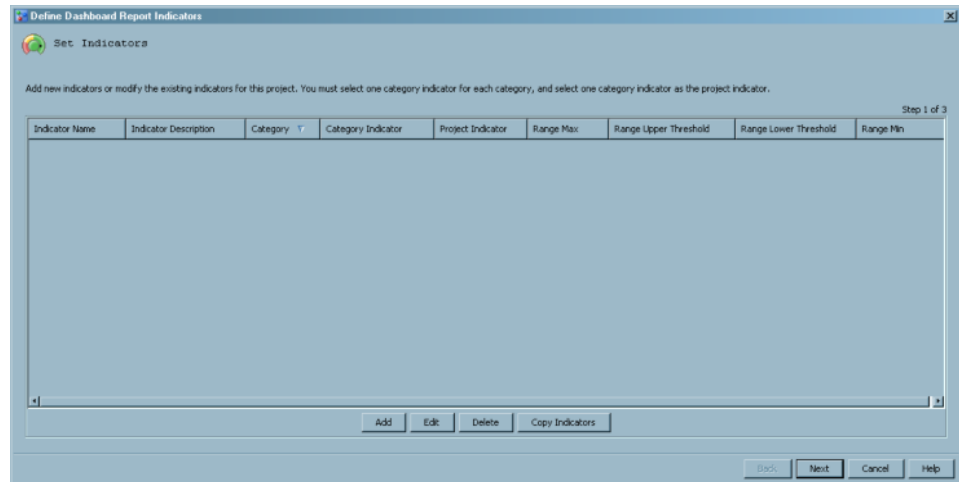
For more information, see the *SAS Model Manager: Administrator's Guide*.

Define Dashboard Report Indicators

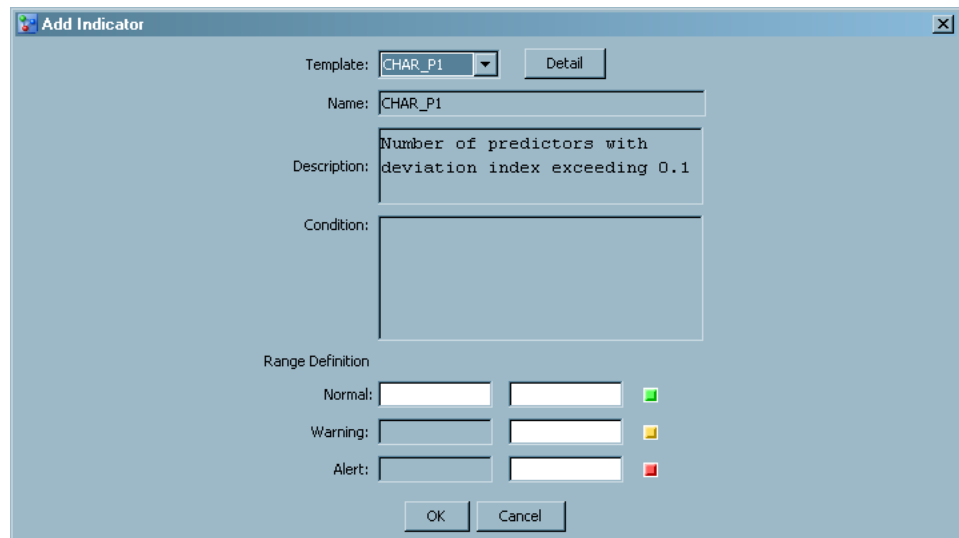
To define dashboard report indicators, follow these steps:

1. You must have at least one project that contains performance data before you continue to the next step. For more information, see “Run the Define Performance Task Wizard” on page 245.

2. Right-click the project folder in the Project Tree, and select **Define Dashboard Report Indicators** from the pop-up menu. The Define Dashboard Report Indicators window appears.



3. To copy dashboard report indicators from another project, click **Copy Indicators**. Otherwise, continue to step 4 to add indicators.
 - a. Select the project name from which to copy the indicators.
 - b. Click **OK**.
4. Click **Add**. The Add Indicator window appears.



- a. Select a template from the **Template** drop-down list.

Note: Click **Detail** to view information about the selected indicator template.
- b. The **Name** and **Description** values are populated from the selected template. If the selected indicator template requires a condition, the name and description can be modified.
- c. Enter a condition for the indicator if the **Condition** field has been configured for use.
- d. Enter values for the **Normal**, **Warning**, and **Alert** range definitions.
- e. Click **OK**. The Define Dashboard Report Indicators window appears with information about the new indicator.

Define Dashboard Report Indicators

Set Indicators

Add new indicators or modify the existing indicators for this project. You must select one category indicator for each category, and select one category indicator as the project indicator.

Step 1 of 3

Indicator Name	Indicator Description	Category	Category Indicator	Project Indicator	Range Max	Range Upper Threshold	Range Lower Threshold	Range Min
CHAR_F1	Number of predictors with de...	Characteristic	<input type="radio"/>	<input type="radio"/>	3	2	1	0

Buttons: Add, Edit, Delete, Copy Indicators

Navigation: Back, Next, Cancel, Help

5. Repeat step 4 for each indicator that you want to add. To edit an existing indicator, select the indicator, and click **Edit**.
6. Select one **Category Indicator** for each category, and select one indicator as the **Project Indicator**.

Note: The indicator that you select as a project indicator must also be a category indicator.

Define Dashboard Report Indicators

Set Indicators

Add new indicators or modify the existing indicators for this project. You must select one category indicator for each category, and select one category indicator as the project indicator.

Step 1 of 3

Indicator Name	Indicator Description	Category	Category Indicator	Project Indicator	Range Max	Range Upper Threshold	Range Lower Threshold	Range Min
CHAR_F1	Number of predictors with de...	Characteristic	<input checked="" type="radio"/>	<input checked="" type="radio"/>	3	2	1	0
GINDECAY	Gini index decay	Model Assessm...	<input checked="" type="radio"/>	<input type="radio"/>	0.6	0.4	0.2	0
STAB_F1	Number of outputs with devi...	Stability	<input checked="" type="radio"/>	<input type="radio"/>	3	2	1	0

Buttons: Add, Edit, Delete, Copy Indicators

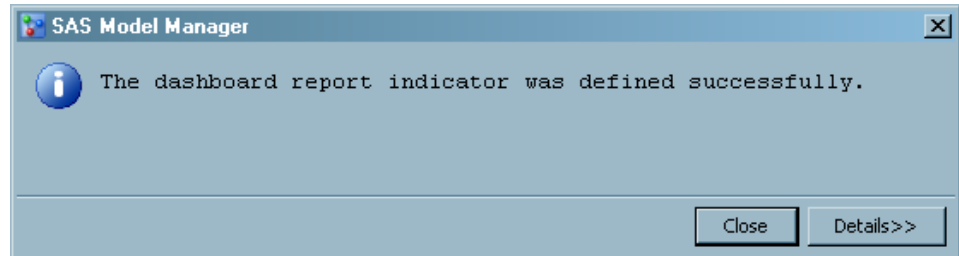
Navigation: Back, Next, Cancel, Help

7. Click **Next**. The Define Dashboard Report Indicators window appears with information about setting up notifications.

- b. To delete a report type, select a value from the **Report Description** list, and click **Delete**.

Note: If all report types are deleted, this project is not included in the generated dashboard reports. SAS Model Manager displays a confirmation message.

11. Click **Finish**. An informational message is displayed indicating that the dashboard indicator was defined successfully.



Note: You must define dashboard report indicators for all projects that you want to be included in your dashboard reports.

See Also

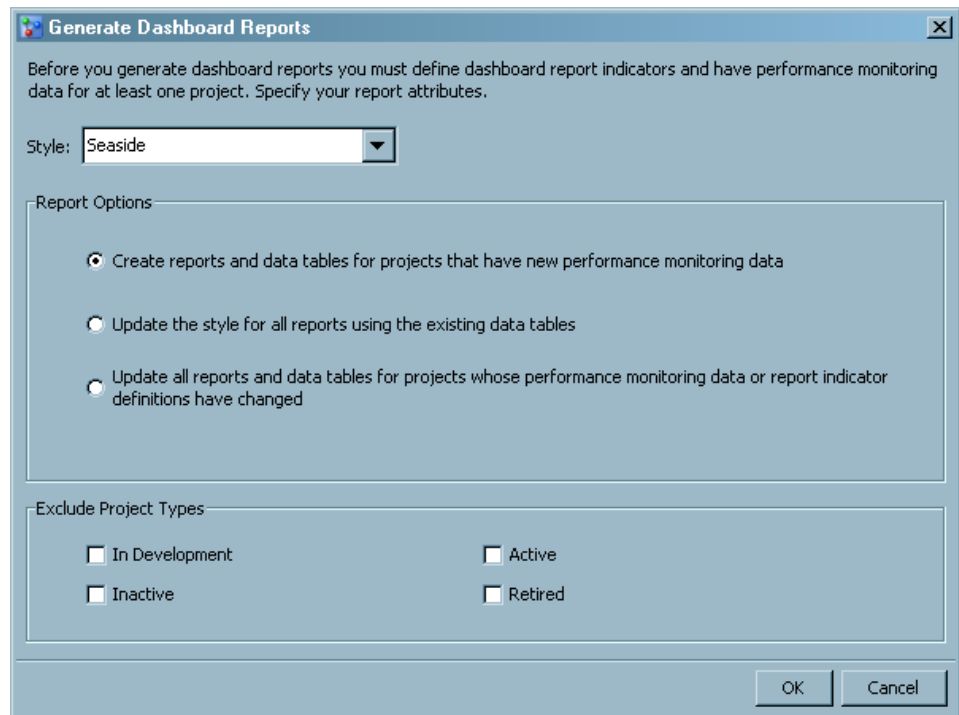
- [“Overview of Project Dashboard Reports” on page 279](#)
- [“Generate Dashboard Reports” on page 283](#)
- [“View Dashboard Reports” on page 285](#)

Generate Dashboard Reports

To generate the dashboard reports, follow these steps:

Note: Before you execute the dashboard report, you must have at least one project that contains performance data. That project must also have at least one dashboard report indicator defined.

1. Select **Tools** ⇒ **Generate Dashboard Reports** from the menu. The Generate Dashboard Reports window appears.



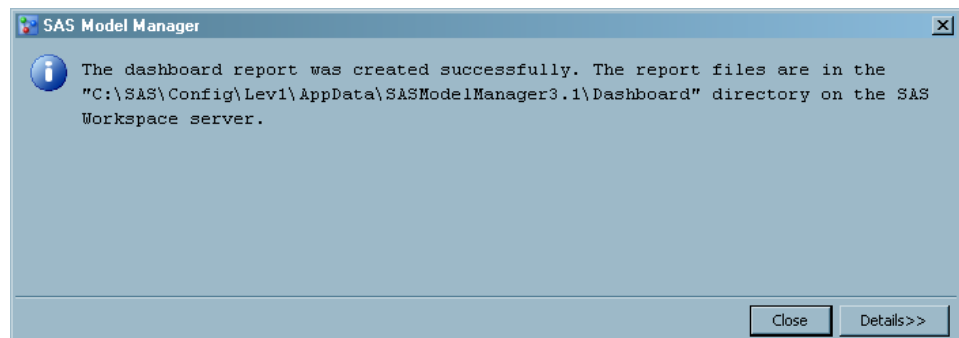
2. Select a style for the report from the drop-down list.

Note: SAS Model Manager administrators can configure the report styles that are available using SAS Management Console.

3. Select one of the following report options:
 - Create reports and data tables for projects that have new performance monitoring data.
 - Update the style for all reports, using the existing data tables.
 - Update all reports and data tables for projects whose performance monitoring data or report indicator definitions have changed.
4. (Optional) Select one or more project types that you want to exclude from the dashboard reports.

5. Click **OK**. A message appears that indicates whether the report was created successfully. The message also displays the location of the dashboard reports on the SAS Workspace Server. Here is an example: **C:\SAS\Config\Lev1\AppData\SASModelManager3.1\Dashboard**.

Note: SAS Model Manager administrators can configure the location of the dashboard report directory.



6. To view the dashboard report, navigate to the directory location specified in the message. For more information, see [“View Dashboard Reports” on page 285](#).

For more information, see the *SAS Model Manager: Administrator's Guide*.

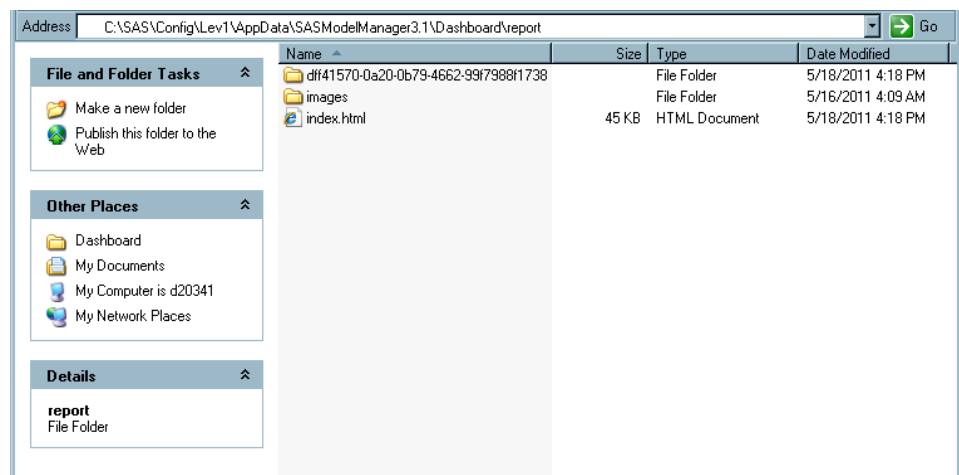
See Also

- [“Overview of Project Dashboard Reports” on page 279](#)
- [“Define Dashboard Report Indicators” on page 279](#)

View Dashboard Reports


To view the dashboard reports, follow these steps:

1. Navigate to the **report** folder in the dashboard report directory on the SAS Workspace Server. For example, **C:\SAS\Config\Lev1\AppData\SASModelManager3.1\Dashboard**.







2. Open the **index.html** file in your browser window. The SAS Model Manager Dashboard appears.

All projects





Project Name	Current Status	Owner	Model Age (days)
/MMRoot/DDHMEQ/HMEQ	 2011Q1	mdlmgadmin	7

History Status

Project Name	Current	Current - 1	Current - 2	Current - 3
/MMRoot/DDHMEQ/HMEQ	 2011Q1	 2010Q4	 2010Q3	 2010Q2










3. Select a project name or a status to view the associated dashboard reports. The Project Reports Index appears in a new window. If you select a status, only the dashboard reports for that time frame are displayed.

Project Reports Index

Time	Status	Project Indicator	Report
2011Q1		Number of predictors with deviation index exceeding 0.1	KPI Dashboard Report
			KPI Detail Report
			KPI Trend Dashboard Report
			Monitoring Report
2010Q4		Number of predictors with deviation index exceeding 0.1	KPI Dashboard Report
			KPI Detail Report
			KPI Trend Dashboard Report
			Monitoring Report
2010Q3		Number of predictors with deviation index exceeding 0.1	KPI Dashboard Report
			KPI Detail Report
			KPI Trend Dashboard Report
			Monitoring Report
2010Q2		Number of predictors with deviation index exceeding 0.1	KPI Dashboard Report
			KPI Detail Report
			KPI Trend Dashboard Report
			Monitoring Report

4. Select a report to view report details. The report details appear in the same window.

**KPI Detail Report
2011Q1**

Category	Category Status	Category Indicator	Indicator	Indicator Status	Value
Characteristic			Number of predictors with deviation index exceeding 0.1		3.0000
Model Assessment			Gini index decay		0.4564
Stability			Number of outputs with deviation index exceeding 0.1		0.0000

Note: To return to the Project Reports Index, select the browser's back button.

For more report examples, see [“Dashboard Report Examples”](#) on page 379.

See Also

- [“Overview of Project Dashboard Reports”](#) on page 279
- [“Define Dashboard Report Indicators”](#) on page 279
- [“Generate Dashboard Reports”](#) on page 283

Chapter 19

Retraining Models

Overview of Retraining Models	289
Prerequisites for Retraining a Model	290
Define a Model Retrain Task	290
Execute a Model Retrain Task	296
Viewing Retrained Models and Model Comparison Reports	297
View Retrained Models	297
View Model Comparison Reports for Retrained Models	298

Overview of Retraining Models

Using SAS Model Manager, you can retrain models to respond to data and market changes. Retraining models enables you to update out-of-date models and improve model performance. When you define a model retrain task, you can select multiple models to be retrained at the same time. The definition of the model retrain task includes the destination version and training data source. The destination version is an existing version or new version that is associated with the selected project and stores the retrained model information.

The training data source contains new data for retraining the selected models. You can also specify a location to store the comparison reports and retrain results. When you select the models to include in the comparison report, you can use the training data source or select a different data source to compare the performance of the new models. You can also specify the report options, including the name, format and style of the comparison report. E-mail notifications can also be defined for a model retrain task and are sent after you execute a model retrain task.

By default, the champion model in the default version for the selected project is retrained if a model is not selected. After you execute a model retrain task, if the **Register new trained model** option was selected, SAS Model Manager registers the new models to the destination version. The comparison report is stored in the **Model Retrain** folder, as well as in the report folder on the SAS Workspace Server that was specified when the model retrain task was defined.

Note: Only models that are created by using SAS Enterprise Miner or R models can be retrained.

To retrain models in SAS Model Manager, take the following steps:

- Ensure that all [prerequisites](#) have been completed.

- [Define a model retrain task](#) to generate the SAS code that retrains models.
- [Execute](#) the generated SAS code.
- [View](#) the new models and comparison report.

Prerequisites for Retraining a Model

Before you execute the Define a Model Retrain Task, complete the following prerequisites:

- If you want to retrain the project champion model, ensure that the champion model in the default version is set. For more information, see [“Champion Models” on page 194](#) and [“Set a Default Version” on page 198](#).
- Verify that the training data set that you want to use as the training data source has been registered in the SAS Metadata Repository using SAS Management Console. For more information, see [“Train Tables” on page 31](#).

- Verify that the appropriate project and model properties are set. Here is a list of properties.

Project Properties

- Training Target Variable
- Target Event Value
- Class Target Level
- Output Event Probability Variable

Model Properties

- Score Code Type

For more information, see [“Specific Properties for a Project” on page 359](#) and [“Specific Properties for a Model” on page 145](#).

- Verify that all of the project output variables are mapped to the corresponding model output variables. For more information, see [“Map Model Variables to Project Variables” on page 138](#).

Define a Model Retrain Task

To define a model retrain task, follow these steps:

1. Right-click the project name and select **Define Model Retrain Task**. The Define Model Retrain Task wizard appears.

Define Model Retrain Task Step 1 of 3

Select Models

☐ Select All

Sel...	ID	Name	Version	Type	Champion
<input type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Loan/type	Loan	2011	ClassificationModel	NO
<input type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Reg1/type	Reg1	2011	ClassificationModel	YES
<input type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Tree1/type	Tree1	2011	ClassificationModel	NO

Model Retrain Settings

Destination version for new models: 2011

Training data source:

Report folder:

Retrain result folder:

☒ Register new trained model ☐ Trace on

2. (Optional) Select one or more models to be retrained. To select all models, select the **Select All** check box.

Note: If you do not select a model, the champion model in the default version for the selected project is retrained.

3. Select a destination version for new models.


Note: If you do not select a destination version, the default location is used for the destination of the new retrained models.

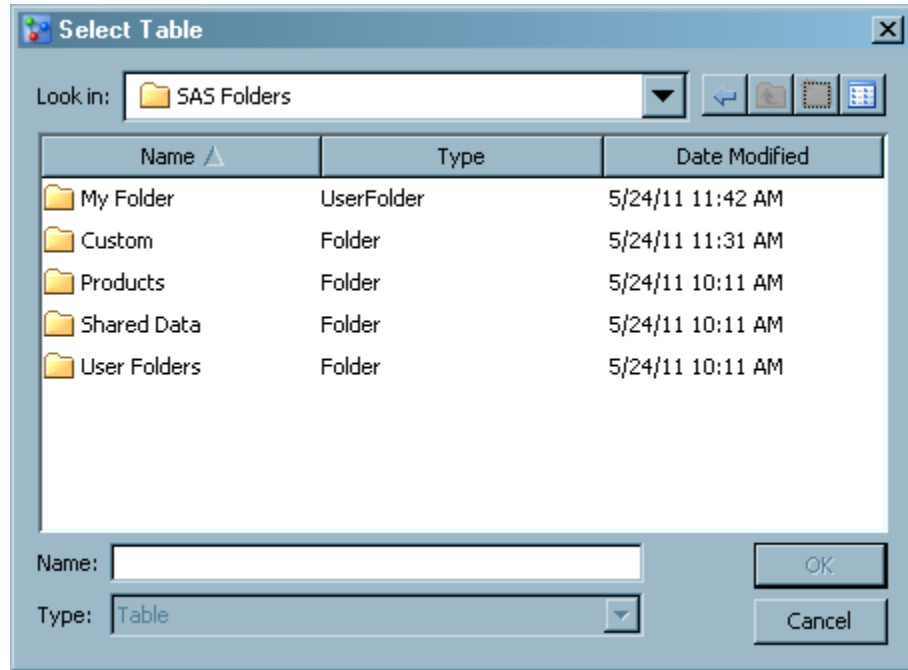
(Optional) To create a new version to store the new retrained models, follow these steps:

- a. From the Select Models for Retrain page of the wizard, click **New**. The New Version window appears.


New Version Enter a version name and description.

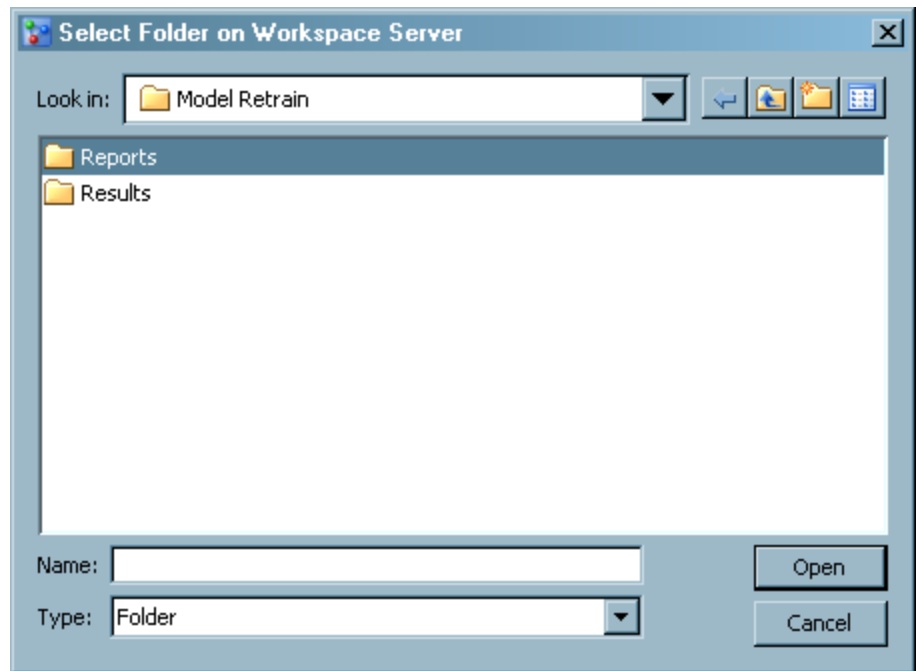
Property	Value
<input checked="" type="checkbox"/> General Properties	
Name *	
Description	
<input checked="" type="checkbox"/> Version Properties	
Life Cycle Template	Basic


- b. Enter a name of the new version and select a life cycle template. The description of the new version is optional.
 - c. Click **OK**. You are then returned to the Define Model Retrain Task wizard.
4. From the Select Models for Retrain page of the wizard, click  to select a data set from the SAS Metadata Repository as the training data source.



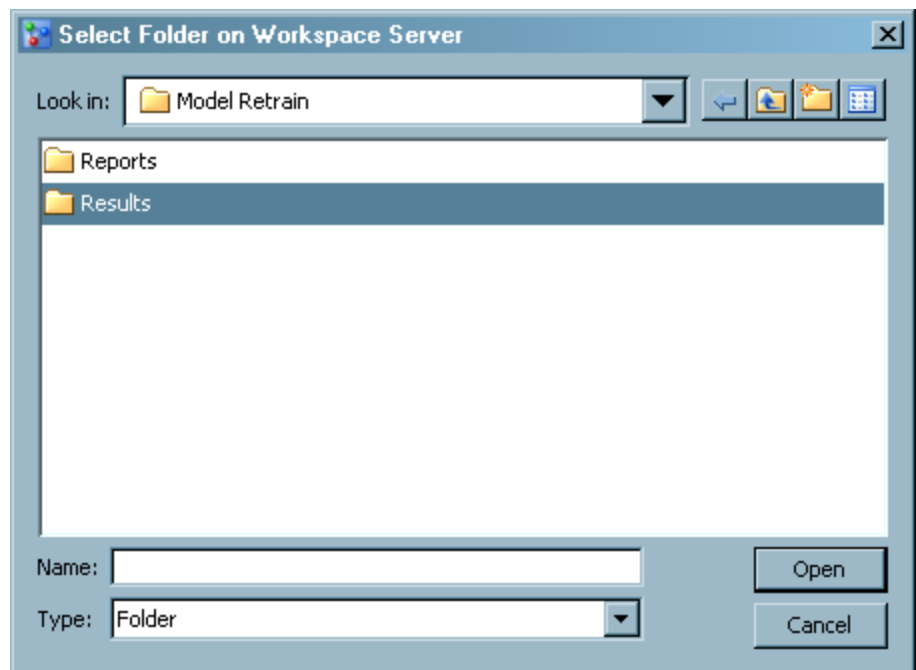
Click **OK**. You are then returned to the Define Model Retrain Task wizard.

5. (Optional) Click **Finish** to generate the model retrain task code if you do not want to specify the additional parameters.
 6. From the Select Models for Retrain page of the wizard, click  to select a report folder to store the comparison report. By default, the report is stored in the SAS session's working folder on the SAS Workspace Server. You can also create a subfolder in which to store the report.

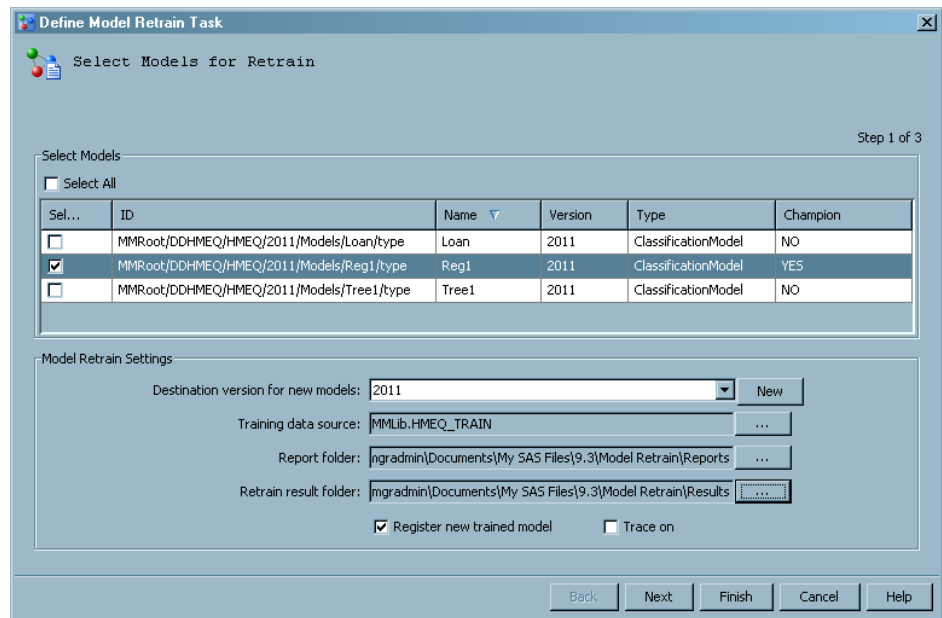


7. From the Select Models for Retrain page of the wizard, click  to select a retrain result folder to store the model training results.

Note: This setting is for informational purposes only. The data sets and files that are created during model retraining are stored in this location. By default, the training results are stored in the SAS session's working folder on the SAS Workspace Server.



8. (Optional) Select **Register new trained model** to specify whether to register the new models. If this option is not selected, the new models are not registered in the destination version in the Project Tree, and they are not saved to the SAS Content Server.



Define Model Retrain Task (Step 1 of 3)

Select Models for Retrain

☐ Select All

Select	ID	Name	Version	Type	Champion
<input type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Loan/type	Loan	2011	ClassificationModel	NO
<input checked="" type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Reg1/type	Reg1	2011	ClassificationModel	YES
<input type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Tree1/type	Tree1	2011	ClassificationModel	NO

Model Retrain Settings

Destination version for new models: 2011

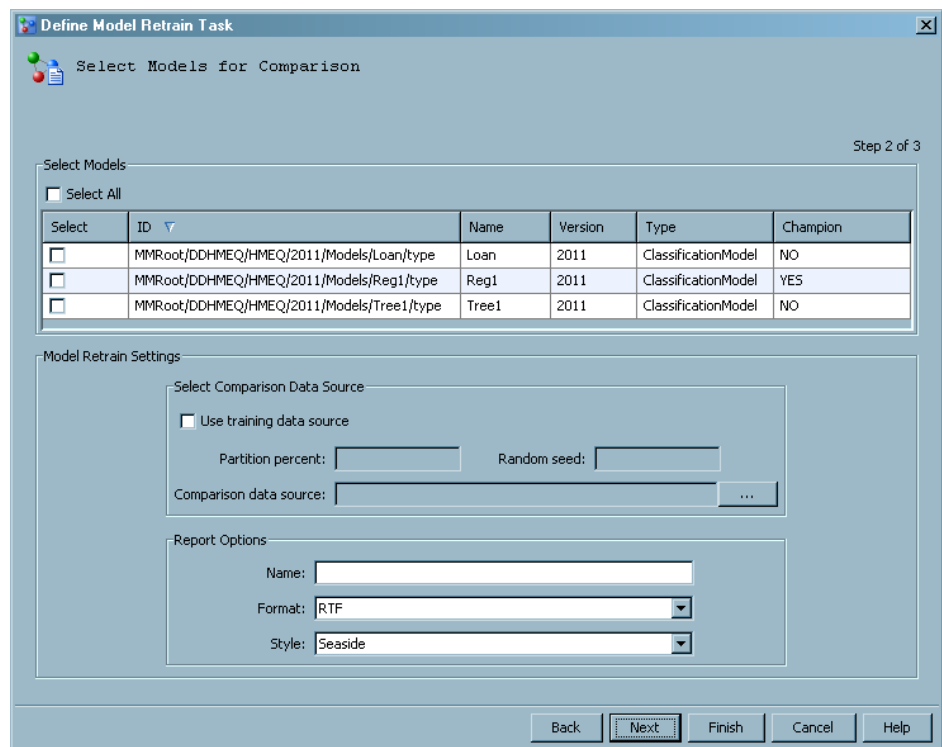
Training data source: MMLib.HMEQ_TRAIN

Report folder: mgradmin\Documents\My SAS Files\9.3\Model Retrain\Reports

Retrain result folder: mgradmin\Documents\My SAS Files\9.3\Model Retrain\Results

☒ Register new trained model ☐ Trace on

9. (Optional) Select **Trace On** to print trace information to the SAS log file.
10. Click **Next**. The **Select Models for Comparison** page is displayed in the Define Model Retrain Task wizard.



Define Model Retrain Task (Step 2 of 3)

Select Models for Comparison

☐ Select All

Select	ID	Name	Version	Type	Champion
<input type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Loan/type	Loan	2011	ClassificationModel	NO
<input type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Reg1/type	Reg1	2011	ClassificationModel	YES
<input type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Tree1/type	Tree1	2011	ClassificationModel	NO

Model Retrain Settings

Select Comparison Data Source

☐ Use training data source

Partition percent: Random seed:

Comparison data source:

Report Options

Name:


Format: RTF

Style: Seaside

11. Select the models to be compared to the retrained model. To select all models, click **Select All**.

Note: If you do not select a model, the champion model in the default version for the project is used to perform the comparison.

12. Select a comparison data source. Take one of the following steps:

- Select **Use training data source** if you want to use it as the comparison data source. You can either use the whole training data source to compare or partition it into two parts, based on partition percent and random seed. The percent that is specified is the percentage of data that is used for model comparison; the other part of the data is used for training. The random seed value is used to generate the training data based on the random sampling method.
- Click the  to select a performance data set as the comparison data source.

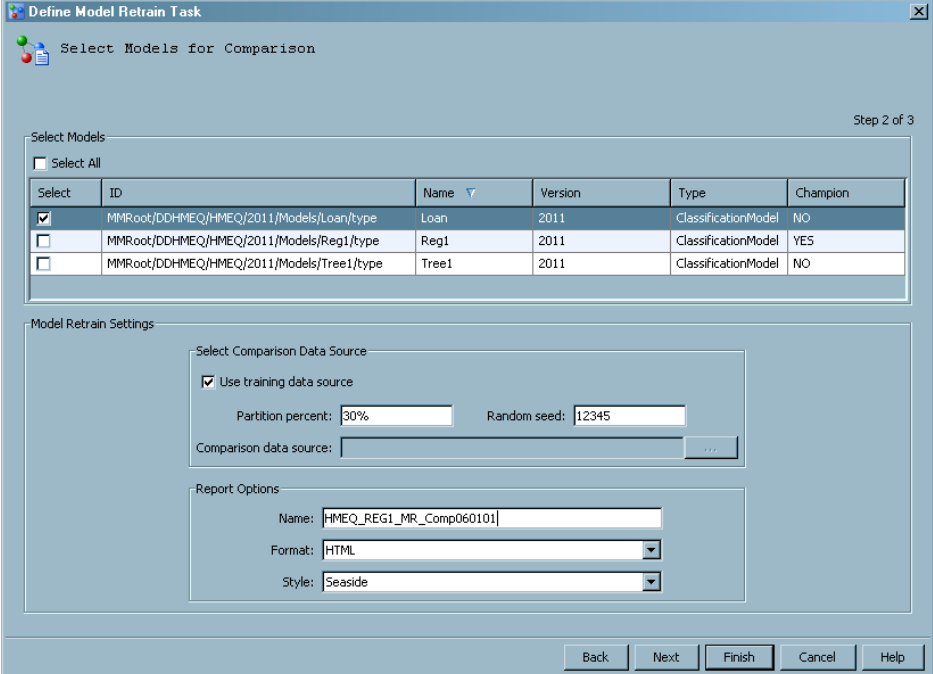
13. Specify the report options by taking the following steps.

- Enter a report name.
- Select a format for the report output. The standard formats that are available are **RTF**, **PDF**, **HTML**, and **Excel**.

Note: SAS Model Manager administrators can configure the report formats that are available using SAS Management Console.

- Select a style for the report.

Note: SAS Model Manager administrators can configure the report styles that are available using SAS Management Console.



Define Model Retrain Task (Step 2 of 3)

Select Models for Comparison

Select Models

☐ Select All


Select	ID	Name	Version	Type	Champion
<input checked="" type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Loan/type	Loan	2011	ClassificationModel	NO
<input type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Reg1/type	Reg1	2011	ClassificationModel	YES
<input type="checkbox"/>	MMRoot/DDHMEQ/HMEQ/2011/Models/Tree1/type	Tree1	2011	ClassificationModel	NO

Model Retrain Settings

Select Comparison Data Source

☒ Use training data source

Partition percent: Random seed:

Comparison data source: 

Report Options

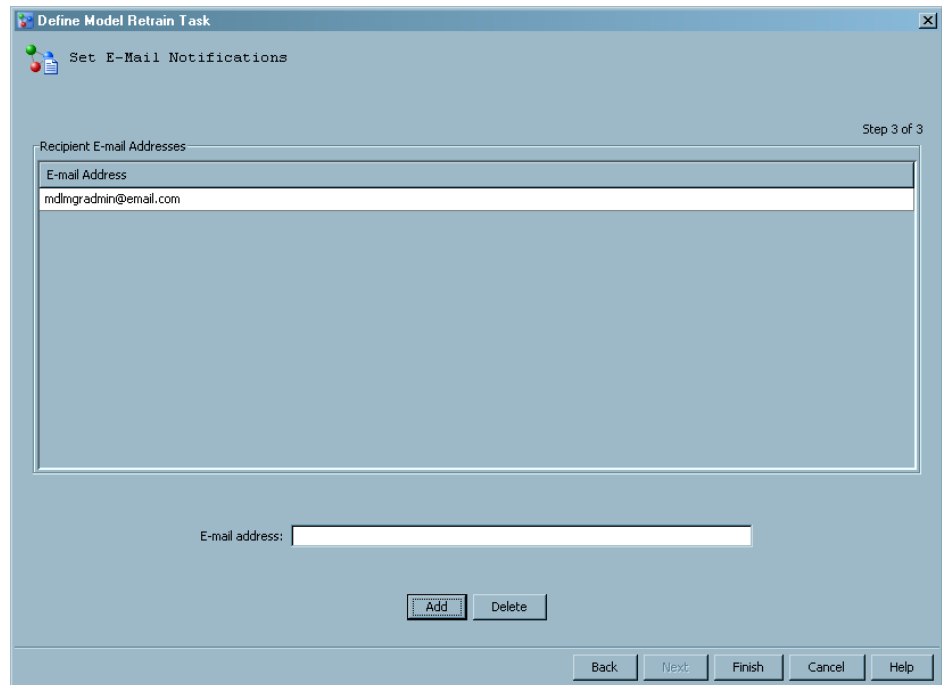
Name:

Format:

Style:

Back Next Finish Cancel Help

14. Click **Next**. The **Set E-Mail Notifications** dialog box is displayed in the Define Model Retrain Task window.



15. (Optional) To send the training results by e-mail, enter an e-mail address or multiple e-mail addresses that are separated by a comma or blank, and then click **Add**. To delete a recipient, select the recipient's e-mail address and click **Delete**.
16. Click **Finish**. The SAS code is generated and placed in the **Model Retrain** folder of the associated project.

See Also

[“Execute a Model Retrain Task” on page 296](#)

Execute a Model Retrain Task

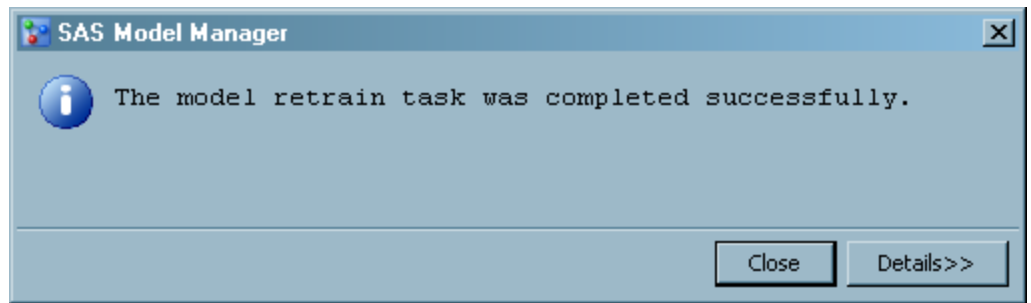
The prerequisites for retraining a model must be completed and a model retrain task must be defined before you can execute a model retrain task.

To execute a model retrain task, follow these steps:

1. Expand the project folder.
2. Right-click the **Model Retrain** folder, and then select **Execute** from the pop-up menu.

Note: The model retrain task is executed as a background process. You can view the progress of the model retrain task in a status bar at the bottom of the SAS Model Manager application window.

3. When the model retrain has finished executing a success message is displayed. Click **Close**.



Note: If you chose to register the retrained model, it is now available in the **Models** folder of the selected destination version. If this option was not selected, the new models are not registered in the destination version in the Project Tree and they are not saved to the SAS Content Server. If the model retrain task does not execute successfully, click **Details**, or look for error messages in the SAS log (**ModelRetrain.log**) that is located in the report folder. The report folder for the retrained model comparison report has been created in the **Model Retrain** folder.

See Also

[“Viewing Retrained Models and Model Comparison Reports” on page 297](#)

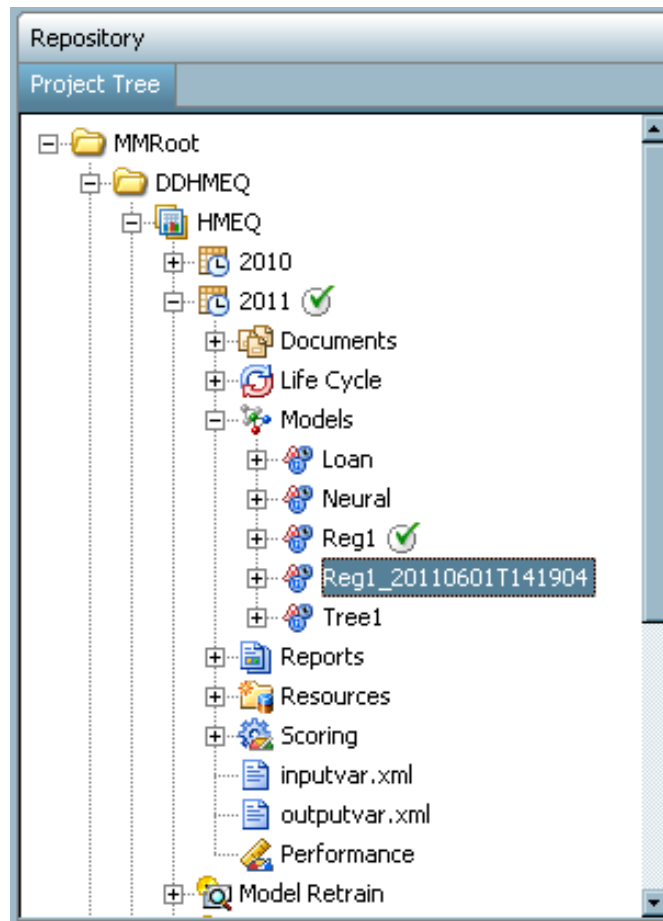
Viewing Retrained Models and Model Comparison Reports

After a model retrain task is executed, if you chose to register the retrained models when defining the model retrain task, the new retrained models are available in the **Models** folder. In addition, if you specified that the model retrain task should create a model comparison report, the report is available in the **Model Retrain** folder for the associated project.

View Retrained Models

To view retrained models, follow these steps:

1. Expand the destination version node to see the new models in the **Models** folder.

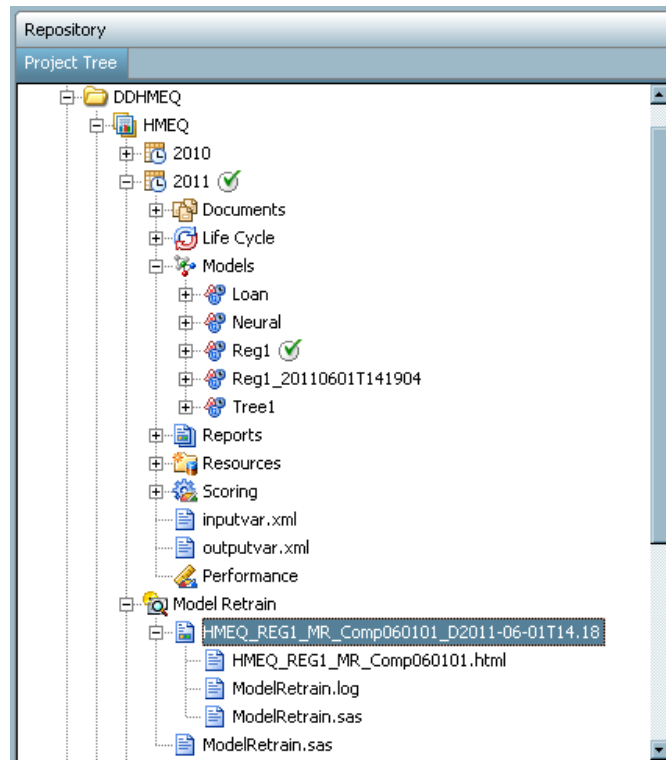


2. Expand the new retrained model folder to view its contents.

View Model Comparison Reports for Retrained Models

To view a model comparison report, follow these steps:

1. Expand the report folder that is located in the **Model Retrain** folder for the associated project.



2. Right-click the report output file, and select **Open** from the pop-up menu. Specify user credentials if you are prompted to.

Note:

You can also view the model retrain report in the following ways:

- Navigate to the report folder location that you specified when defining the model retrain task.
- Open the SPK file that was sent in the e-mail notification. This action is available only if you set a notification when defining the model retrain task.

For report examples, see [“Model Retrain Comparison Report Example”](#) on page 387.

Part 6

Appendixes

<i>Appendix 1</i>	
Query Utility	303
<i>Appendix 2</i>	
SAS Model Manager Access Macros	311
<i>Appendix 3</i>	
SAS Model Manager Macro Variables	351
<i>Appendix 4</i>	
Properties	357
<i>Appendix 5</i>	
SAS Model Manager R Model Support	371
<i>Appendix 6</i>	
Report and Performance Monitoring Examples	379

Appendix 1

Query Utility

Overview of the Query Utility	303
Search for Models	304
Search by Using a UUID	306
Search Life Cycles for Tasks Assigned to Users	307

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 subfolders 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 subfolders 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/2010Q3/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 **2010Q3**, 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 304](#)
- [“Search by Using a UUID” on page 306](#)
- [“Search Life Cycles for Tasks Assigned to Users” on page 307](#)

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

Specify values to use for the query.

Model

Component

Life Cycle

Name:

Algorithm:

DecisionTree

Input variables:

Target variables:

Model creator:

User-defined key:

User-defined value:

Search Results:

Name	Path	Algorithm	Type	Model Cr...
Tree1	//ModelMana...	DecisionTree	Classification...	mdlmgadmin

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/2010Q3/Model/Model%201</code> , you can find the model Model 1 in the project HomeEquity , the version 2010Q3 , and the Models folder.
Algorithm	Specifies the name of the algorithm, such as regression or logistic, that is used by the model.
Type	Specifies one of the model function types: <ul style="list-style-type: none">AnalyticalModelClassificationModelPredictionModelClusteringModel
Model Creator	Specifies the user who created or imported the model, depending on the model import method.

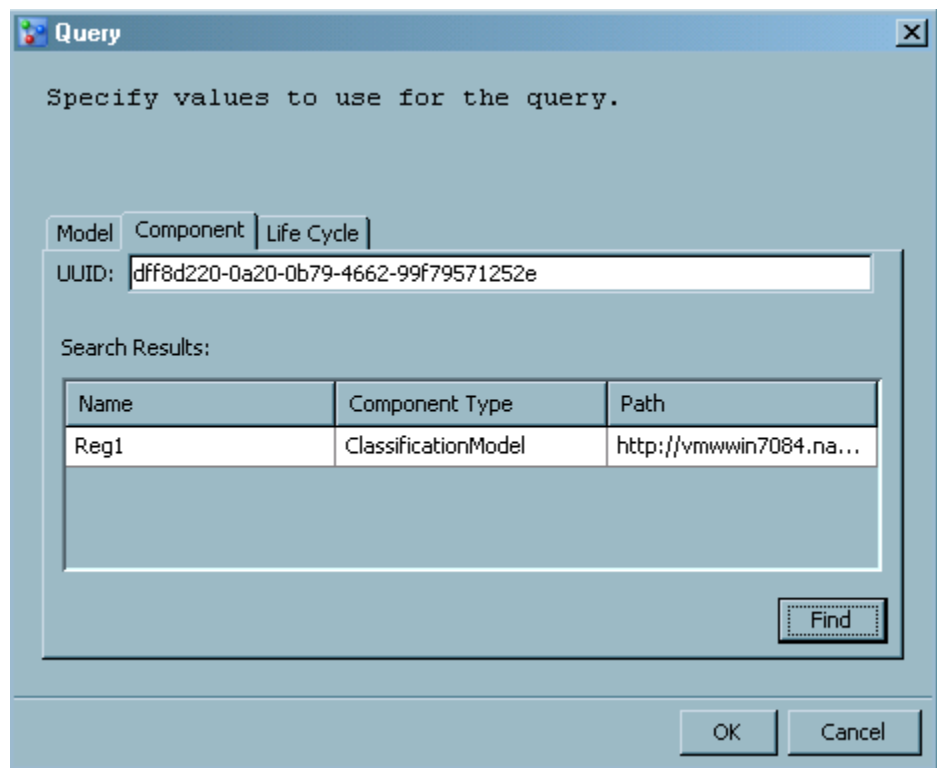
See Also

- “Search Life Cycles for Tasks Assigned to Users” on page 307
- “Search by Using a UUID” on page 306

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. Click 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** appear below the **UUID** field.

The search results displays the following information :

Column	Description
Name	Specifies the name of the project, version or model.

Column	Description
Component Type	<p>Specifies one of following component types:</p> <p>AnalyticalModel the component is an analytical model</p> <p>ClassificationModel the component is a classification model</p> <p>ClusteringModel the component is a clustering or segmentation model</p> <p>ModelGroup the component is an organizational folder</p> <p>PredictionModel the component is a prediction model</p> <p>Project the component is a project</p> <p>Version the component is a version</p>
Path	<p>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/2010Q3</code> you can find the version in the project HomeEquity.</p>

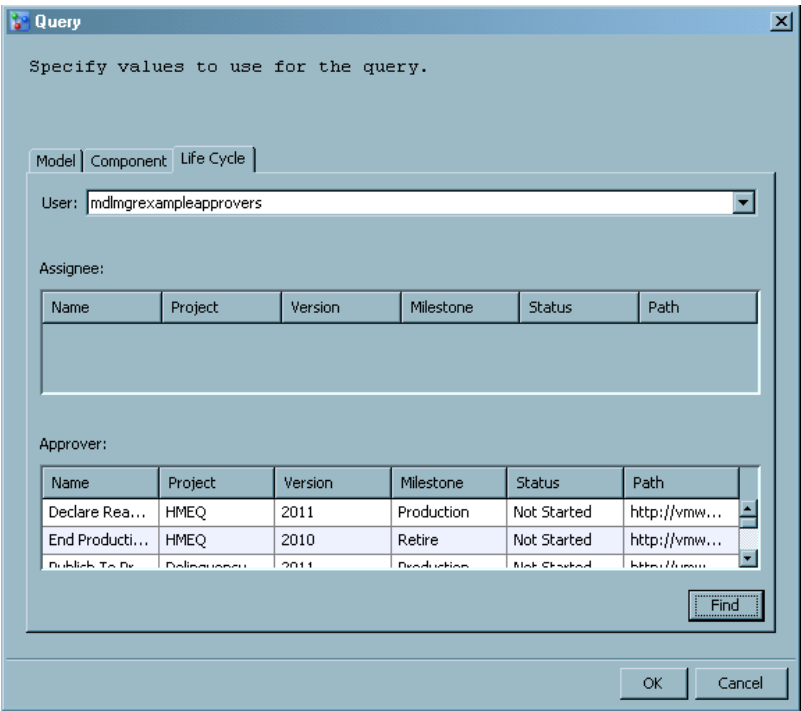
See Also

- [“Search for Models” on page 304](#)
- [“Search Life Cycles for Tasks Assigned to Users” on page 307](#)

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.



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 .
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/2011Q2/Mortgages/Testing/Signoff</code> , you can find the user in the project HomeEQ , the version 2011Q2 , and the life cycle Mortgages .

See Also

- [“Search for Models” on page 304](#)
- [“Search by Using a UUID” on page 306](#)

Appendix 2

SAS Model Manager Access Macros

Overview of Access Macros	311
Using the SAS Model Manager Access Macros	312
Accessing the Macros	312
Identifying SAS Model Manager Model Repository Objects	312
Identifying Files Used by Access Macros	313
Required Tables	313
Global Macro Variables	315
Required Global Macro Variables	316
Dictionary of Access Macros	317
%MM_AddModelFile Macro	317
%MM_GetModelFile Macro	320
%MM_GetURL Macro	324
%MM_Register Macro	325
%MM_RegisterByFolder Macro	342
%MM_CreateModelDataset Macro	347

Overview of 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.
- %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.

Note: The macros are in the modelmgr.sas7bcat file. The location of this file for Windows is \sasinstalldir\SASFoundation\9.3\mmcommon\sashelp. The default value for sasinstalldir in Windows is C:\Program Files\SAS. The location of this file for UNIX is /sasinstalldir/SASFoundation/9.3/sashelp. The default value for sasinstalldir in UNIX is /usr/local/SASHome.

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.
- 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 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 **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/2011*.

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 315.

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 `SMMMModel` to access model component files, as in this example:

```
libname smmmodel "c:\myModel\HMEQ\scorecode";
```

You can define the libref `SMMMModel` 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 ninq 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;

```

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 273](#)

`_MM_MulticastPort`

contains the multicast port number that the Web Infrastructure Platform server listens for SAS Model Manager requests.

Default: 8080

`_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

_MM_RC Return Value	Access Macro Status
6	Generic DATA step error

_MM_ResourceURL

contains the URL of the **Resources** folder. the **_MM_ResourceURL** is set by the **%MM_GetURL** macro when the macro returns a version URL in the **_MM_URL** global macro variable.

_MM_MulticastAddress

contains the multicast address of the network server where the Web Infrastructure Platform is installed.

Example: %let _MM_MulticastAddress=239.27.18.213

_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_MulticastAddress** global macro variable.

Default: the value of SAS automatic macro variable &SYSUSERID.

See Also

[“SAS Model Manager Macro Variables” on page 351](#)

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 Web Infrastructure Platform is installed
- a user and password for processing requests to SAS Model Manager
- the URL or path to the SAS Model Manager repository

Make sure that your SAS program defines values for these macro variables when you use the access macros:

- **_MM_MulticastAddress**
- **_MM_MulticastPort**
- **_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 273](#).

For a description of these macro variables as well as their default values, see [“Global Macro Variables” on page 315](#).

Here is a code example that uses the four macro variables to describe how to the access to the server for the Web Infrastructure Platform.


```

Filename pwfile "my-network-drive\pwfile";

%let _MM_MulticastAddress = 239.27.18.213;
%let _MM_MulticastPort = 8561;
%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;

```

See Also

[“SAS Model Manager Macro Variables” on page 351](#)

Dictionary of 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 317](#)
- [“Arguments” on page 317](#)
- [“Details” on page 318](#)
- [“Example” on page 319](#)

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/2011/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 WIP Server variables.            */
*****/

%let _MM_MulticastAddress=239.27.18.213;
%let _MM_MulticastPort=8561;
%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;

/*****
/* 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/2011/hmeqDecTree1,

```

```

Name=modelinput.sas7bdat,
SASDataFile=mtb1s.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/2011/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 320
- “Arguments” on page 320
- “Details” on page 322
- “Example” on page 322

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/2011/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/2011

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 the SAS WIP Server variables.          */
*****/

%let _MM_MulticastAddress=239.27.18.213;
%let _MM_MulticastPort=8561;
%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;

/*****
/* Specify the model component file name and  */
/* destination.                               */
*****/

%let WorkPath = c:\myProject\2011;
  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/2011/
DecisionTree, TextFile=dest);

/*****
/* Display SAS Model Manager set macro variables.  */
*****/

Options nosource;
%PUT _MM_RC = &_MM_RC;
%PUT _MM_CId = &_MM_CId;
Options source;

/*****
/* Run score code. Sepcify the LIBNAME input path.  */
*****/

LIBNAME input 'c:\mysascode\2011\DTree';
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 324](#)
- [“Arguments” on page 324](#)
- [“Details” on page 324](#)
- [“Example” on page 324](#)

Syntax

%MM_GetURL(UUID=UUID, <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

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 the SAS WIP Server variables. */
*****/

%let _MM_MulticastAddress=239.27.18.213;
%let _MM_MulticastPort=8561;
%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,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 326
- “Arguments” on page 326
- “Details” on page 331
- “Examples” on page 338

Syntax

```
%MM_Register(
  VersionId=destination-version-UUID,
  ModelTemplate=model-template-name,
  EMMModelPackage=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-for-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, you could specify %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 SAS Enterprise Miner model package file.
- Models that were registered using the LogisticOutModelTable parameter are registered with the Classification template.
- All other registrations default to the AnalyticalModel template.

EMModelPackage=*SAS-filereference-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 329](#)

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 329](#)

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 329](#)

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.

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 that is specified by the InDataSamp argument can be specified in this list.

Required: No

See also: [InDataSamp argument on page 327](#)

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 that is specified by the OutDataSamp argument can be specified in this list.

Required: No

See also: [KeepOutVars argument on page 328](#)

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 328](#)

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 appear 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 argument on page 329.</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 argument on page 329.</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 125](#) and [“Syntax” on page 317](#).

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 WIP Server variables. *****/
*****/

%let _MM_MulticastAddress=239.27.18.213;
%let _MM_MulticastPort=8561;
%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;

/*****
/* 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/2011,
  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 WIP Server variables. */
*****/

%let _MM_MulticastAddress=239.27.18.213;
%let _MM_MulticastPort=8561;
%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;

/*****
/* 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/2011,
    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 WIP Server variables. */
*****/

%let _MM_MulticastAddress=239.27.18.213;
%let _MM_MulticastPort=8561;
%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;

/*****
/* 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/2011,
    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_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/2011

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

Requirement Type	Enterprise Miner Models	SAS Code Models
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

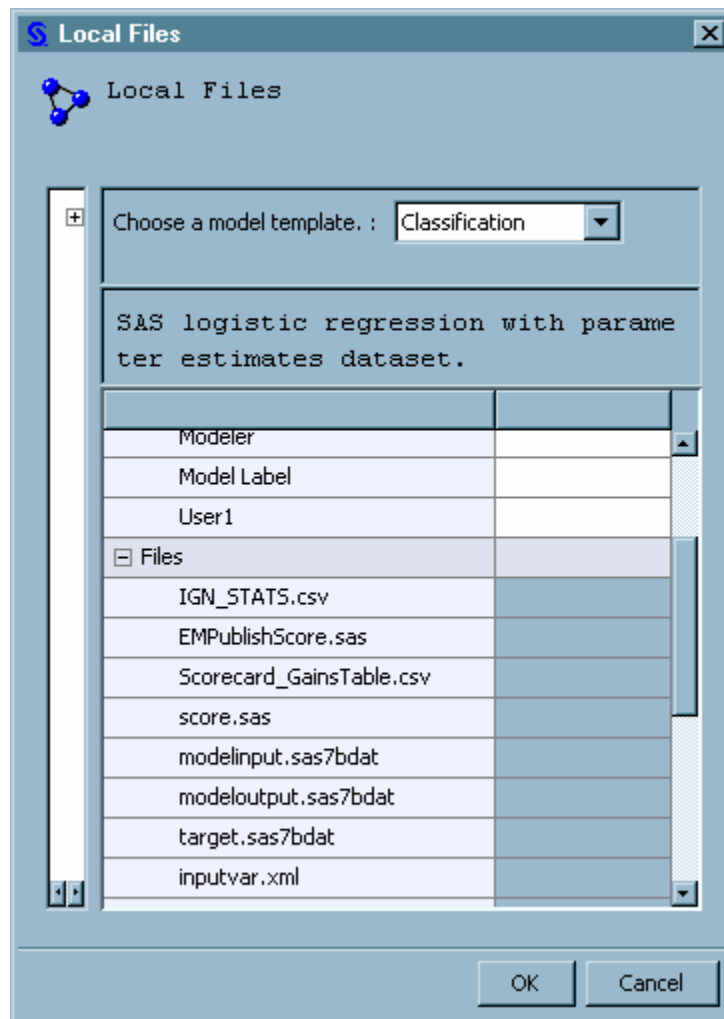
ModelOutput.sas7bdat is a SAS data set that contains one or more model output variables.

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 145](#)
- <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 127.

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 WIP Server variables. */
*****/

%let _MM_MulticastAddress=239.27.18.213;
%let _MM_MulticastPort=8561;
%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;

/*****
/* Specify the location of the folder. */
*****/

%let modelFolder = c:\myModel;
%let hmeq2011 = //ModelManagerModelRepos/MMRoot/HomeEquity/HMEQ/2011;

```

```

/*****
/* Set to detect failure in case macro load fails      */
/* and register the models in SAS Model Manager.      */
*****/

%let _MM_RC= -1;

%MM_RegisterByFolder(VersionId=&hmq2011, 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 WIP Server variables. */
*****/

%let _MM_MulticastAddress=239.27.18.213;
%let _MM_MulticastPort=8561;
%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;

/*****
/* Specify the location of the data set and model */
/* path. */
*****/

libname modelDS 'c:\myModel\ModelInfo';
%let hmeq2011 = //ModelManagerModelRepos/MMRoot/HomeEquity/HMEQ/2011;

/*****
/* 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/2011/Models/
    Regression,
    Trace=ON);

/*****
/* Display the SAS Model Manager-defined variables. */
*****/

```



```
Options nosource;  
%PUT _MM_RC = &_MM_RC;  
Options source;
```


Appendix 3

SAS Model Manager Macro Variables

Scoring Task Macro Variables

The following table lists the macro variables that are used to run a scoring task:

Macro Variable Name	Description	Example
_MM_InputDS	the location of the input data source file	http:// vmwwin7084.na.sas.com: 8080/SASContentServer/ repository/default/sasfolders/ Shared Data/Model Manager/ MMLib/ HMEQ_SCORE_INPUT.sas7 bdat
_MM_InputLib	the libref that is associated with the location of the input data source file	inlib
_MM_ModelID	the UUID of the model	4622bdda- ac1b-12d5-0196-021edec543 47
_MM_MulticastAddress	the SAS Model Manager server multicast address	239.27.18.213
_MM_MulticastPort	the SAS Model Manager server multicast port	8561
_MM_OutputDS	the location of the output data source file	http:// vmwwin7084.na.sas.com: 8080/SASContentServer/ repository/default/sasfolders/ Shared Data/Model Manager/ MMLib/ HMEQ_SCORE_OUTPUT.s as7bdat
_MM_OutputLib	the libref that is associated with the location of the output data source file	outdslib

Macro Variable Name	Description	Example
<code>_MM_Password</code>	the password of the user ID that is running the report	mdlmgrpw2
<code>_MM_PerformanceDS</code>	the location of the performance data source file	http:// vmwwin7084.na.sas.com: 8080/SASContentServer/ repository/default/sasfolders/ Shared Data/Model Manager/ MMLib/ HMEQ_perf2011Q2.sas7bdat
<code>_MM_PerformanceLib</code>	the libref that is associated with the location of the performance data source file	perflib
<code>_MM_TaskDir</code>	the URL of the stored scoring task	http://myserver.mycompany: 8080/SASContentServer/ repository/default/ ModelManager/MMRoot/ DDHMEQ/HMEQ/2011/ Scoring
<code>_MM_TestDS</code>	the location of the test data source file	http:// vmwwin7084.na.sas.com: 8080/SASContentServer/ repository/default/sasfolders/ Shared Data/Model Manager/ MMLib/ HMEQ_TEST.sas7bdat
<code>_MM_TestLib</code>	the libref that is associated with the location of the test source file	testlib
<code>_MM_TrainDS</code>	the location of the train data source file	http:// vmwwin7084.na.sas.com: 8080/SASContentServer/ repository/default/sasfolders/ Shared Data/Model Manager/ MMLib/ HMEQ_train.sas7bdat
<code>_MM_TrainLib</code>	the libref that is associated with the location of the train source file	trainlib
<code>_MM_User</code>	the user ID of the user that is running the report	mdlmgradmin

Validating Model Report Macro Variables

The following tables lists the macro variables that are used to create model comparison reports, model profile reports, delta reports, dynamic lift reports, and user reports:

Macro Variable Name	Description	Example
_MM_ModelLabel	a label for a model	reg
_MM_Password	the password of the user ID that is running the report	mdlmgrpw2
_MM_PosteriorVa	the model's posterior variable name	EM_EVENTPROBABILITY
_MM_ReportFormat	the output format of the generated report	html
_MM_ReportLib	the libref for the Report node	report
_MM_ResourcesLib	the libref for the Resources node	resources
_MM_SourceCodeType	the type of score code	SAS Program
_MM_TargetEvent	the target event value	1
_MM_TargetVar	the target variable name	bad
_MM_TaskDir	the URL of the stored report	http://myserver.mycompany:8080/SASContentServer/repository/default/ModelManager/MMRoot/DDHMEQ/HMEQ/2011/Reports
_MM_User	the user ID of the user that is running the report	mdlmgradmin

Performance Monitoring Report Macro Variables

The following table lists the macro variables that are used to create performance monitoring reports:

Macro Variable Name	Description	Example
_MM_DateTime	the time that the performance task is to run	1Sep2011:05:00:00
_MM_ModelName	the name of the champion model	reg1
_MM_ModelID	the UUID of the champion model	7514d6e-ac1b-12d5-01e4-878abeb04505

Macro Variable Name	Description	Example
<code>_MM_ModelLocalPath</code>	the location of the SAS Work library in the SAS Model Manager server	C:\ \\DOCUME~1\ADMINI~1\LOCALS~1\Temp\1\SAS Temporary Files \\TD2032_BRDVM0199_
<code>_MM_MulticastAddress</code>	the SAS Model Manager server multicast address	2.27.18.213
<code>_MM_MulticastPort</code>	the SAS Model Manager server multicast port	8561
<code>_MM_Password</code>	the password of the user ID that is running the report	mdlmgrpw2
<code>_MM_ProjectAlias</code>	the alias of the project	
<code>_MM_ProjectPath</code>	the network path to the SAS Model Manager project.	//ModelManagerDefaultRepo/ MMRoot/DDHMEQ/HMEQ
<code>_MM_ProjectURLPath</code>	the URL to the SAS Model Manager project	http:// myserver.mycompany.com: 8080/SASContentServer/ repository/default/ ModelManager/MMRoot/ HMEQ
<code>_MM_ProjectUUID</code>	the project UUID	27514d6e- ac1b-12d5-01e4-878abeb045 05
<code>_MM_ScoreCodeType</code>	the type of score code	SAS Program
<code>_MM_VersionName</code>	the name of the default version	2011
<code>_MM_ReportDataSrc</code>	the project's performance data set	jun11perf.sas7bdat
<code>_MM_PreCode</code>	one or more macro variables that set values to performance variables	%let _MM_EventProbVar=score; %let _MM_TargetVar=bad;
<code>_MM_ResultURLPath</code>	the URL to the version's Resources node	http:// myserver.mycompany.com: 8080/SASContentServer/ repository/default/ ModelManager/MMRoot/ HMEQ/2011/Resources
<code>_MM_TimeLabel</code>	the label that is used in reports to represent the time period of the data in the performance data set	2011Q2

Macro Variable Name	Description	Example
_MM_Trace	indicates whether to write a trace log	ON or OFF
_MM_User	the user ID of the user that is running the report	mdlmgradmin

Dashboard Report Macro Variables

The following table lists the macro variables that are used to create dashboard reports:

Macro Variable Name	Description	Example
_MM_Dashboard_Dir	the path to the directory where the dashboard report is stored	C:\SAS\Config\Lev1\AppData\SASModelManager\Dashboard
_MM_Force_Run_Dash_Reports	whether to force running the report and updating all tables	Y or N
_MM_MulticastAddress	the SAS Model Manager server multicast address	2.27.18.213
_MM_MulticastPort	the SAS Model Manager server multicast port	8561
_MM_Password	the password of the user whose user ID is running the report	mdlmgrpw2
_MM_ReportFormat	the output format of the generated report	html
_MM_Report_Style	the style used in the generated report	Seaside
_MM_SAS_Locale	the SAS session locale	en_US
_MM_User	the user ID of the user who is running the report	mdlmgradmin

Model Retrain Report Macro Variables

The following table lists the macro variables that are used to retrain models:

Macro Variable	Description	Example
_MM_MulticastAddress	the SAS Model Manager server multicast address	2.27.18.213
_MM_MulticastPort	the SAS Model Manager server multicast port	8561
_MM_Password	the password of the user ID that is running the report	mdlmgrpw2
_MM_User	the user ID of the user who is running the report	mdlmgradmin

Appendix 4

Properties

General Properties	357
System Properties	358
Specific Properties for a Project	359
User-Defined Properties	362
Organization-Specific User-Defined Properties	362
SAS User-Defined Properties	362
Specific Properties for a Version	363
Specific Properties for Milestones and Tasks	365
Specific Properties for a Model	366
Scoring Task Properties	368
Result Set Properties	369

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 who created the component. This property is Read-only. SAS Model Manager assigns the value when the component is created.
Date Created	Specifies the date that the component was created. This property is Read-only.

Property Name	Description
Date Modified	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

Property Name	Description
Lock Project Metadata	Specifies that the project metadata is locked and the project definition cannot be modified.
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 scoring functions when a user publishes the associated project champion model to a database.</p> <p>The value of the Default Train Table property is used to validate 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 57 .
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, ordinal, 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
Class Target Level	Specifies the class target level of binary, nominal, ordinal, or interval.
Output Event Probability Variable	The output event probability variable name, when the Model Function property is set to Classification .
Output Prediction Variable	The output prediction variable name, when the Model Function property is set to Prediction .
Output Classification Variable	The output classification 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 A4.1 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 that is 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 and select **Add User-Defined Property**. The Add User-Defined Property window appears.
2. In the **Name** field, add a name beginning with a letter. The characters @ \ / * % # & \$ () ! ? < > ^ + ~ ` = { } [] | ; : ' " cannot be used in the name. Spaces cannot be used 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.

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.
BusinessContext	Specifies the business context for which the project is used.
ScoringInputTable	Specifies the name of the input table that the model uses for scoring.

When you publish a 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 Scoring Function window.

Here are the scoring function user-defined properties.

Property Name	Description
DbmsTable	Specifies the name of the input table that is used in scoring functions. This field should be specified for the project properties before you publish a scoring function.
ScoringFunctionName	Specifies the user-defined portion of the scoring function name. You can modify this property value only by changing the name in the Publish Scoring Function window. The name must start with a letter or underscore. The name cannot start with a special character or number such as 0123456789@/*%#&()?<^_.
ScoringFunctionPrefix	<p>Specifies a prefix for the 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 Greenwich Mean time (GMT) timestamp of when you select the Publish 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	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.

Property Name	Description
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 scoring functions when a user publishes the associated project champion model to a database. • checks the Validate scoring results box in the Publish Scoring Function window. <p>The value of the Default Train Table property is used to validate 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.
Date Frozen	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 .
Date Retired	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 .
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 .
Date Completed	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.
Date Approved	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.

Property Name	Description
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 scoring functions when a user publishes the associated project champion model to a 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 119 .
Modeler	Specifies the Modeler ID or, when Modeler ID is missing, specifies the user ID of the individual who created the model that 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	<p>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.</p> <p><i>Note:</i> SAS Model Manager cannot export or publish models whose Score Code Type model property is set to SAS Program.</p>
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.

Property Name	Description
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 that is specific to the scoring task.

Property Name	Description
Scoring Task Type	Specifies a value of Test or Production for the type of scoring task.
SAS Workspace Server	Specifies the name of the Workspace Server to which SAS Model Manager is connected. This value is taken from the SAS Metadata Repository.
Model	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.

Property Name	Description
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 , 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 setting identifies the output table where the results of the scoring are written.

See Also

- [“Create a Scoring Task” on page 155](#)
- [“Modify a Scoring Task” on page 156](#)
- [“Execute a Scoring Task” on page 158](#)

Result Set Properties

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

Property Name	Description
Number of Observations	<p>When Scoring Task Type is set to Test, this property specifies how many observations are to be read from the scoring task input table. This setting enables you to limit the number of records that are 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.</p> <p>When Scoring Task Type is set to Production, this property specifies how many observations show in a result set in order to limit the number of observations to display. This value can be set only by a SAS Model Manager administrator using SAS Management Console. If a value is not specified, the default value is 0. A 0 indicates no limit on the number of observations to display. For information about setting Number of Observations when Scoring Task Type is Production, see <i>SAS Model Manager: Administrator's Guide</i>.</p>

See Also

- [“Create a Scoring Task” on page 155](#)
- [“Modify a Scoring Task” on page 156](#)

Appendix 5

SAS Model Manager R Model Support

Overview of Using R Models with SAS Model Manager	371
Preparing R Model Files to Use with SAS/IML	372
Build an R Model	372
Prepare an R Model Template File	373
Prepare R Model Component Files	374

Overview of Using R Models with SAS Model Manager

R is a freely available language and environment for statistical computing and graphics. Using the open architecture of SAS Model Manager, you can register and import R models. SAS Model Manager requires a model template file and model component files that are created specifically for R models.

The following SAS components are required to use R models in SAS Model Manager:

- SAS/IML. You must license SAS/IML because the IML procedure is required to export SAS data sets to R and to submit R code.
- the RLANG system option. You must set this system option.

SAS Model Manager supplies three R model templates that you can use, or you can create your own using the SAS Model Manager Template Editor. The R model templates that are provided by SAS Model Manager support the analytic, classification, and prediction model functions. The segmentation model function is not supported for R models.

After the model component files are registered, you can perform all SAS Model Manager functions except for exporting an R model to the SAS Metadata Repository.

To use R models in SAS Model Manager, do the following tasks:

1. Ensure that the RLANG system option is set. To have the RLANG system option set when SAS starts, have your site administrator add the RLANG system option to the SAS configuration file.
2. Build an R model. For more information, see [“Build an R Model” on page 372](#).
SAS/IML must be installed before you build an R model.
3. Ensure that you have a model template file. For more information, see [“Prepare an R Model Template File” on page 373](#).

4. Ensure that you have the required model component files. For more information, see [“Prepare R Model Component Files” on page 374](#).
5. Import the R model. For more information, see [“Importing an R Model” on page 134](#).

Preparing R Model Files to Use with SAS/IML

Build an R Model

Use the following SAS code to create an R model and save it in the outmodel.rda model component file:

```
/* Define the libref to the SAS input data set.    */

libname libref "path-to-input-data-set";

/* Use PROC IML to export the SAS input data set to the R input data set. */

proc iml;
    run ExportDatasetToR("input-data-set" , "R-matrix-input");

/* Submit the model-fitting R code.    */

submit /R;
    attach(R-matrix-input)

    # -----
    # FIT THE MODEL
    # -----
    model-name<- model-fitting-function

    # -----
    # SAVE THE PARAMETER ESTIMATE TO LOCAL FILE OUTMODEL.RDA
    # -----
    save(model-name, file="path/outmodel.rda")
endsubmit;

run;
quit;
```

Supply the following values:

path-to-input-data-set

is the path to the library where the input data set is stored.

input-data-set

is the name of the input data set.

R-matrix-input

is the R input data.

model-name

is the name of the model.

model-fitting-function

is the R formula that is used to fit the model.

path

is the path to where outmodel.rda is to be stored.

Here is an example of creating an R model using the HMEQ train data set as the SAS input data set:

```
libname mmsamp "!sasroot\mmcommon\sample";
proc iml;
  run ExportDatasetToR("mmsamp.hmeq_train" , "mm_inds");
  submit /R;
    attach(mm_inds)

    # -----
    # FIT THE LOGISTIC MODEL
    # -----
    logiten<- glm(BAD ~ VALUE + factor(REASON) + factor(JOB) + DEROG +
                  CLAGE + NINQ + CLNO , family=binomial)

    # -----
    # SAVE THE PARAMETER ESTIMATE TO LOCAL FILE OUTMODEL.RDA
    # -----
    save(logiten, file="c:/RtoMMfiles/outmodel.rda")
  endsubmit;
run;quit;
```

Prepare an R Model Template File

SAS Model Manager provides three R model templates that you can use as a model template for your R model:

- RClassification
- RPrediction
- RAnalyticalmodel

To view these model templates, use the SAS Model Manager Template Editor:

1. Select **Tools** ⇒ **Manage Templates**. The SAS Model Manager Template Editor appears.
2. Select **File** ⇒ **Browse** ⇒ **Browse Templates**. The Browse Templates window appears.
3. Select an R model template and click **Open**.
4. Review the model template to make sure that it contains all of the model component files and properties for your model. If it does, you can use this template to import your R model. To customize the model template, you can save one of the supplied template files using a different name and make modifications. You then upload the model template to the SAS Content Server.

To create a custom R model template, see [“Model Template Component Files” on page 127](#) and [“User-Defined Model Templates” on page 139](#).

Prepare R Model Component Files

R Model Component Files for Executing R Models Using SAS/IML

To submit R models from SAS Model Manager using SAS/IML, you need several model component files:

- modelinput.sas7bdat
- modeloutput.sas7bdat
- target.sas7bdat
- inputvar.xml
- outputvar.xml
- targetvar.xml
- outmodel.rda
- score.r
- score.sas
- training.r (not required if you do not retrain your R model)
- training.sas (not required if you do not retrain your R model)

You create the modelinput.sas7bdat, modeloutput.sas7bdat, target.sas7bdat, inputvar.xml, outputvar.xml, and targetvar.xml files as you would for importing a SAS code file. For more information, see [“Model Template Component Files” on page 127](#).

The remaining files, outmodel.rda, score.r, score.sas training.r, and training.sas require additional file preparation.

Create outmodel.rda

The outmodel.rda file contains the output parameter estimate. This file is used by SAS Model Manager to register and score the model. You create outmodel.rda when you build an R model. See [“Build an R Model” on page 372](#). The outmodel.rda file uses the R function save() to save the scoring results.

Here is the syntax of an outmodel.rda file:

```
save(model-name, file="path/outmodel.rda")
```

Supply the following values:

model-name

is the name of the R model.

path

is the system path to the location where outmodel.rda is stored.

Here is an example outmodel.rda file:

```
save(logiten, file="c:/temp/outmodel.rda")
```

Create score.r

The score.r script is an R script that is used to score data. You can use the following R script to create score.r:

```
attach(R-matrix-input)
```

```

# -----
# LOAD THE OUTPUT PARAMETER ESTIMATE FROM FILE OUTMODEL.RDA
# -----
load('&_mm_scorefilesfolder/outmodel.rda')

# -----
# SCORE THE MODEL
# -----
score<- predict(model-name, type="response", newdata=R-matrix-input)

# -----
# MERGING PREDICTED VALUE WITH MODEL INPUT VARIABLES
# -----
mm_outds <- cbind(R-matrix-input, score)

```

Supply the following values:

R-matrix-input

is the name of the input R matrix file that you specified in the ExportDatasetToR function in the IML procedure. See [“Build an R Model” on page 372](#).

score

is the output variable. The value for *score* must match the output variable that is defined in modeloutput.sas7bdat and outputvar.xml.

model-name

is the name of the R model. The value of *model-name* must match the R save function *model-name* argument that is specified in the outmodel.rda file.

Here is an example score.r file:

```

attach(mm_inds)

# -----
# LOAD THE OUTPUT PARAMETER ESTIMATE FROM FILE OUTMODEL.RDA
# -----
load('&_mm_scorefilesfolder/outmodel.rda')

# -----
# PREDICT
# -----
score<- predict(logiten, type="response", newdata=mm_inds)

# -----
# MERGE THE PREDICTED VALUE WITH MODEL INPUT VARIABLES
# -----
mm_outds <- cbind(mm_inds, score)

```

Create score.sas

The score.sas program defines the score task information in a data set and calls the %mmbatch macro. When you submit the %mmbatch macro, the task mm_r_model_train_main completes the following tasks:

- transforms a scoring data set to an R data frame
- generates and submits R code for scoring
- transforms the scored output to a SAS data set for reporting in SAS Model Manager

Here is the score.sas program:

```

filename tmp catalog "sashelp.modelmgr.mm_include.source";
%include tmp;
filename tmp;

data work.mm_score_task_information;
  length role $ 8;
  length name $ 80;
  length value $ 200;

  role = "input";
  name = "importedData";
  value = "&_mm_inputds";
  output;

  role = "input";
  name = "modelID";
  value = "&_mm_modelID";
  output;

  role = "output";
  name = "exportedData";
  value = "&_mm_outputds";
  output;

  role = "input";
  name = "dataRole";
  value = "output-variable-name";
  output;

  role = "input";
  name = "p_Target";
  value = "output-variable-name";
  output;
run;

/* mm_r_model_score_main is a SAS Model Manager process flow that is used to run */
/* R model scripts using PROC IML.                                           */

%mmbatch(task=mm_r_model_score_main, taskprops= mm_score_task_information);

```

Supply the following value:

output-variable-name

is the output variable that is defined in modeloutput.sas7bdat or modeloutput.xml.

To print verbose SAS logs, add the following lines before the RUN statement in the previous DATA step:

```

role = "input";
name = "_mm_trace";
value = "ON";
output;

```

Create training.r

The training.r script is an R script that is used to build a train model. Use the following script for the training.r file. In the R save function, the path in the file= argument must be &_MM_TrainResultFolder.

You can use the following script to create training.r:

```
attach(R-matrix-input)

# -----
# FIT THE LOGISTIC MODEL
# -----
model-name<- model-fitting-function

# -----
# SAVE THE OUTPUT PARAMETER ESTIMATE TO LOCAL FILE OUTMODEL.RDA
# -----
save(model-name, file="&_MM_TrainResultFolder/outmodel.rda")
```

Supply the following values:

R-matrix-input

is the name of the R matrix that is specified in the ExportMatrixToR function that is used to build a model using the IML procedure.

model-name

is the name of the R model.

model-fitting-function

is an R model fitting function, such as lm() or glm().

Here is an example training.r R script to build the HMEQ R train model:

```
attach(mm_inds)

# -----
# FIT THE LOGISTIC MODEL
# -----
logiten<- glm(BAD ~ VALUE + factor(REASON) + factor(JOB) + DEROG + CLAGE +
              NINQ + CLNO , family=binomial)

# -----
# SAVE THE OUTPUT PARAMETER ESTIMATE TO LOCAL FILE OUTMODEL.RDA
# -----
save(logiten, file="&_MM_TrainResultFolder/outmodel.rda")
```

Create training.sas

If you do not need to retrain your R model in SAS Model Manager, you do not need this file.

The training.sas program defines the train task information in a data set and calls the %mmbatch macro. When you submit the %mmbatch macro, the task mm_r_model_train_main completes the following tasks:

- transforms a training data set to an R data frame
- generates and submits R code for training
- registers the training output parameter estimate file in SAS Model Manager

Here is the training.sas file:

```
filename tmp catalog "sashelp.modelmgr.mm_include.source";
%include tmp;
filename tmp;

data work.mm_train_task_information;
```

```

length role $ 8;
length name $ 80;
length value $ 200;

role = "input";
name = "trainData";
value = "&_mm_inputds";
output;

role = "input";
name = "modelID";
value = "&_mm_modelID";
output;
run;

/* mm_r_model_train_main is a SAS Model Manager process flow that is used to run */
/* R model scripts using PROC IML. */

%mmbatch(task=mm_r_model_train_main, taskprops= mm_train_task_information);

```

To print verbose SAS logs, add the following lines before the RUN statement in the previous DATA step:

```

role = "input";
name = "_mm_trace";
value = "ON";
output;

```

Appendix 6

Report and Performance Monitoring Examples

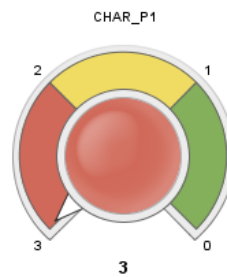
Dashboard Report Examples	379
KPI Dashboard Report	379
KPI Detail Report	380
KPI Trend Report	381
Monitoring Report	382
Model Retrain Comparison Report Example	387
Lift Charts	388
ROC Charts	392
KS Charts	394
Monitoring Performance of a Model without Score Code	397

Dashboard Report Examples

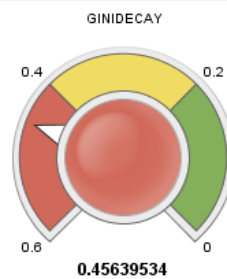
KPI Dashboard Report

Here is an example of the KPI Dashboard Report for the HMEQ project:

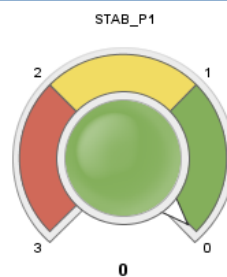
Characteristic



Model Assessment



Stability



KPI Detail Report

Here is an example of the KPI Detail Report for the HMEQ project:

KPI Detail Report 2011Q1

Category	Category Status	Category Indicator	Indicator	Indicator Status	Value
Characteristic			Number of predictors with deviation index exceeding 0.1		3.0000
Model Assessment			Gini index decay		0.4564
Stability			Number of outputs with deviation index exceeding 0.1		0.0000

KPI Trend Report

Here is an example of the KPI Trend Report for the HMEQ project.

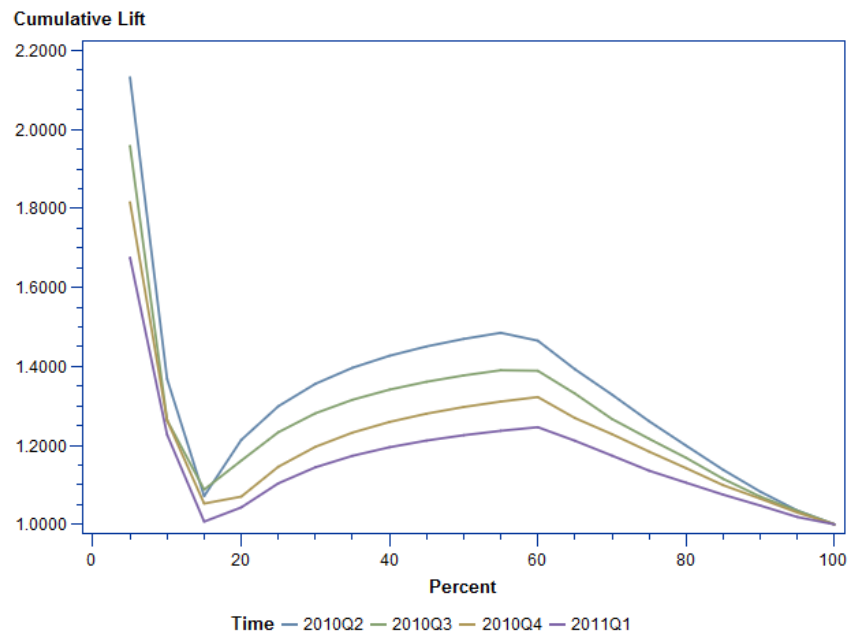


Monitoring Report

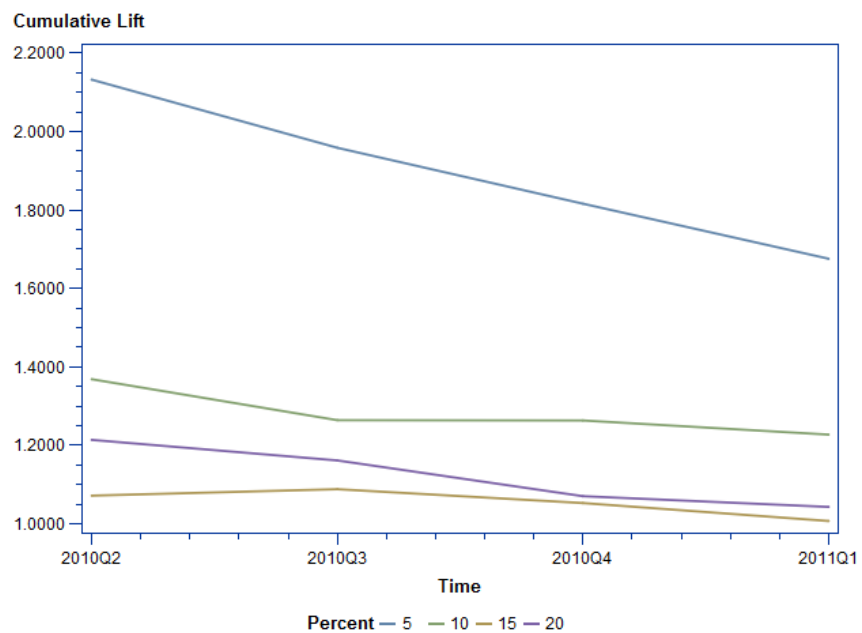
Here is an example of the graphs that are displayed in the Monitoring Report for the HMEQ project.

Multiple assessment charts are displayed in the Monitoring Report. This is a cumulative lift example.

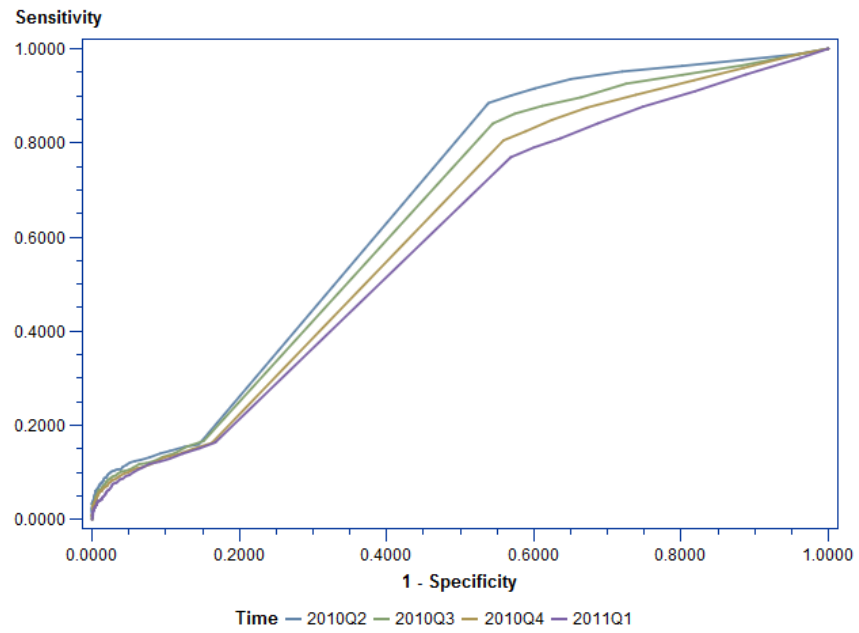
Assessment Chart



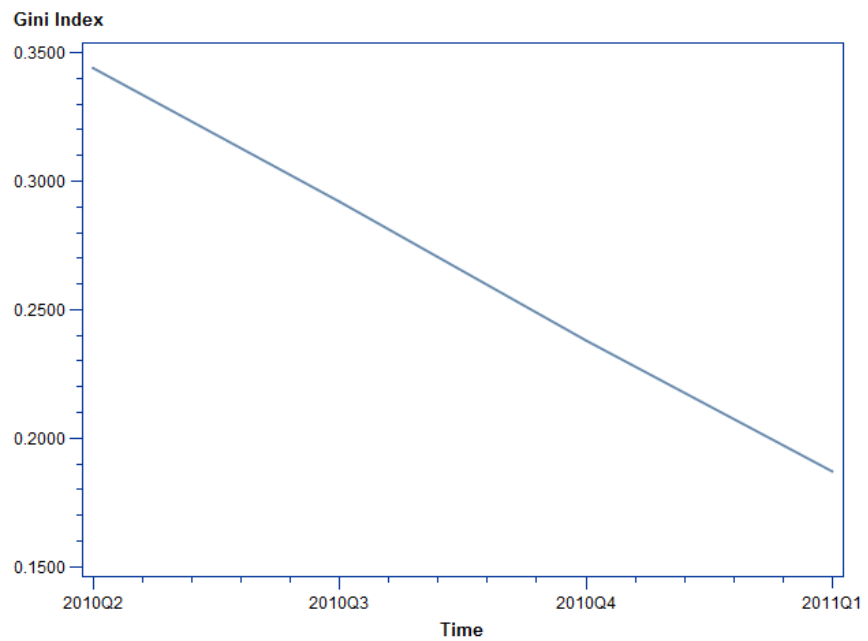
Lift Trend Chart



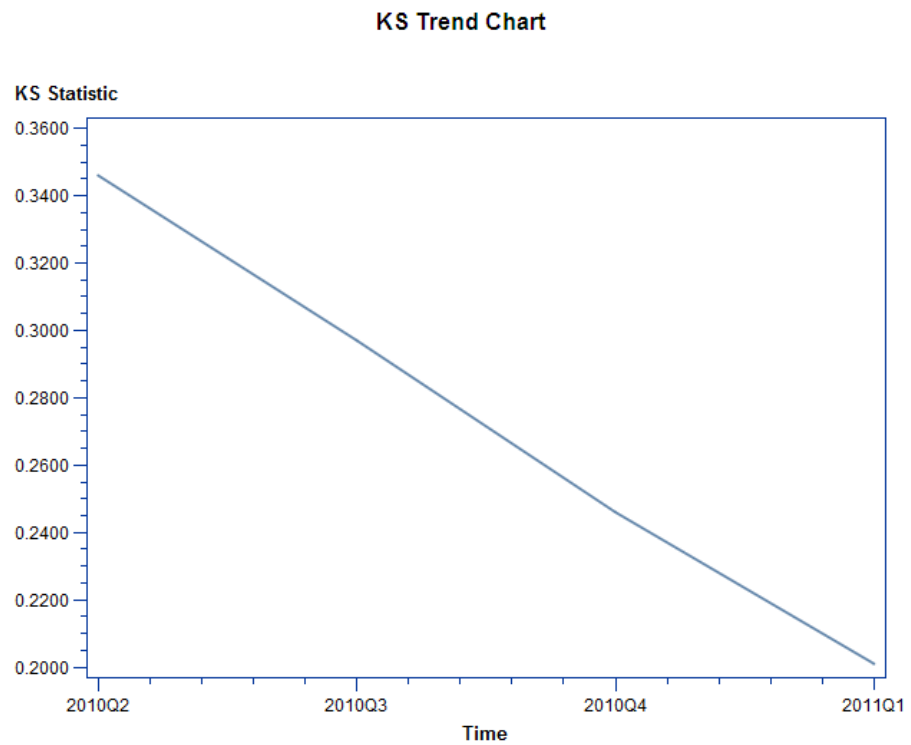
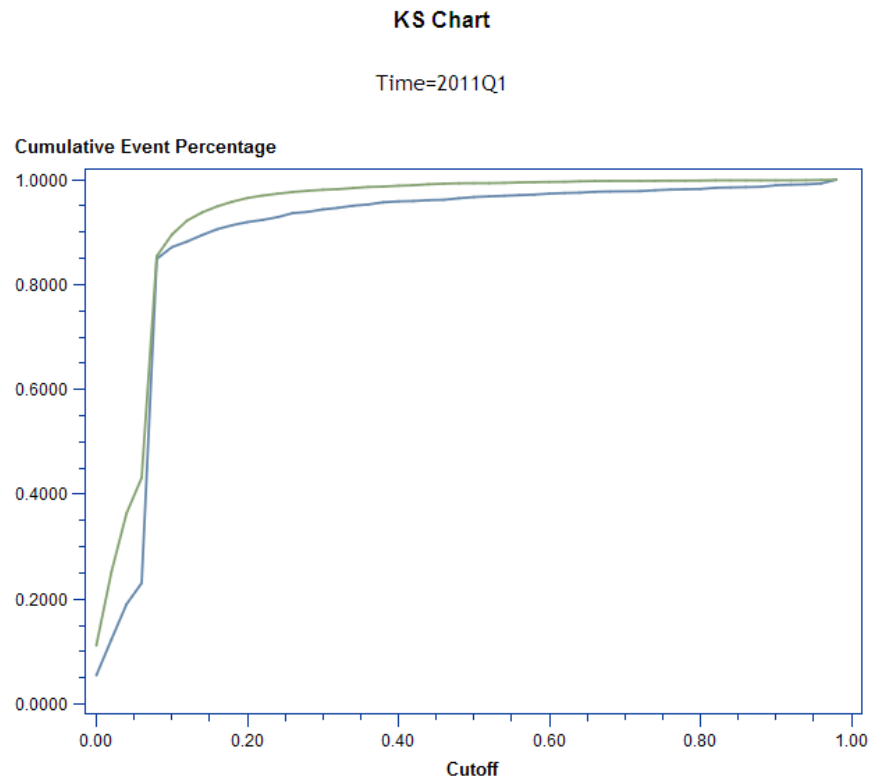
ROC Chart



Gini Trend Chart



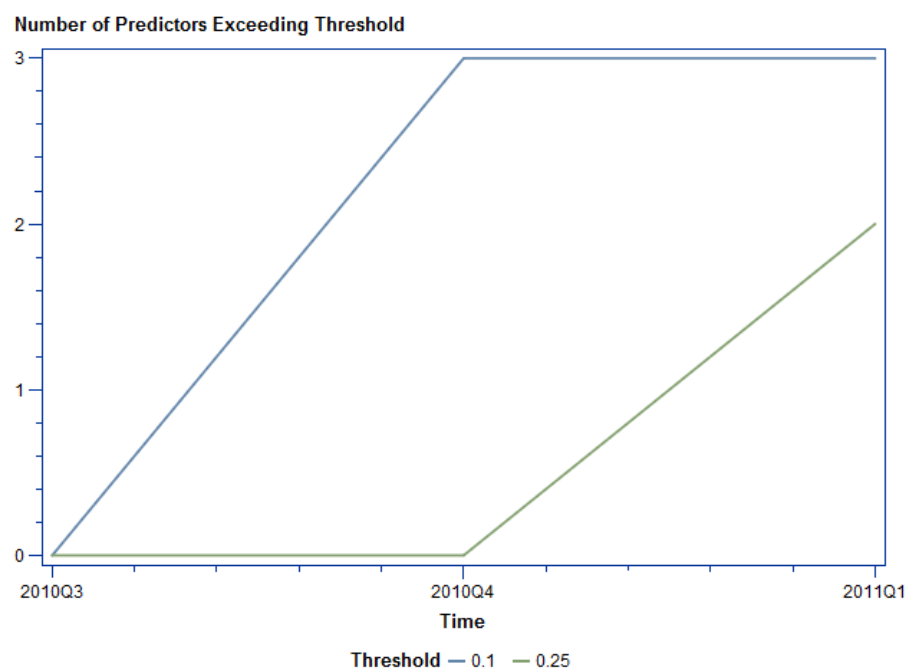
A KS Chart for each time frame is displayed on the Monitoring Report. Here is an example for the first quarter of 2011.



Characteristic Deviation Index

Obs	Time	Variable Name	Deviation Index
1	2010Q3	VALUE	0.0370
2	2010Q3	LOAN	0.0354
3	2010Q3	MORTDUE	0.0256
4	2010Q3	YOJ	0.0095
5	2010Q3	CLAGE	0.0000
6	2010Q3	CLNO	0.0000
7	2010Q3	DEBTINC	0.0000
8	2010Q3	DELINQ	0.0000
9	2010Q3	DEROG	0.0000
10	2010Q3	JOB	0.0000
11	2010Q3	NINQ	0.0000
12	2010Q3	REASON	0.0000
13	2010Q4	VALUE	0.1277
14	2010Q4	LOAN	0.1182
15	2010Q4	MORTDUE	0.1035
16	2010Q4	YOJ	0.0389
17	2010Q4	CLAGE	0.0000
18	2010Q4	CLNO	0.0000
19	2010Q4	DEBTINC	0.0000
20	2010Q4	DELINQ	0.0000
21	2010Q4	DEROG	0.0000
22	2010Q4	JOB	0.0000
23	2010Q4	NINQ	0.0000
24	2010Q4	REASON	0.0000
25	2011Q1	VALUE	0.2902
26	2011Q1	LOAN	0.2529
27	2011Q1	MORTDUE	0.2045
28	2011Q1	YOJ	0.0871
29	2011Q1	CLAGE	0.0000
30	2011Q1	CLNO	0.0000
31	2011Q1	DEBTINC	0.0000
32	2011Q1	DELINQ	0.0000
33	2011Q1	DEROG	0.0000
34	2011Q1	JOB	0.0000
35	2011Q1	NINQ	0.0000
36	2011Q1	REASON	0.0000

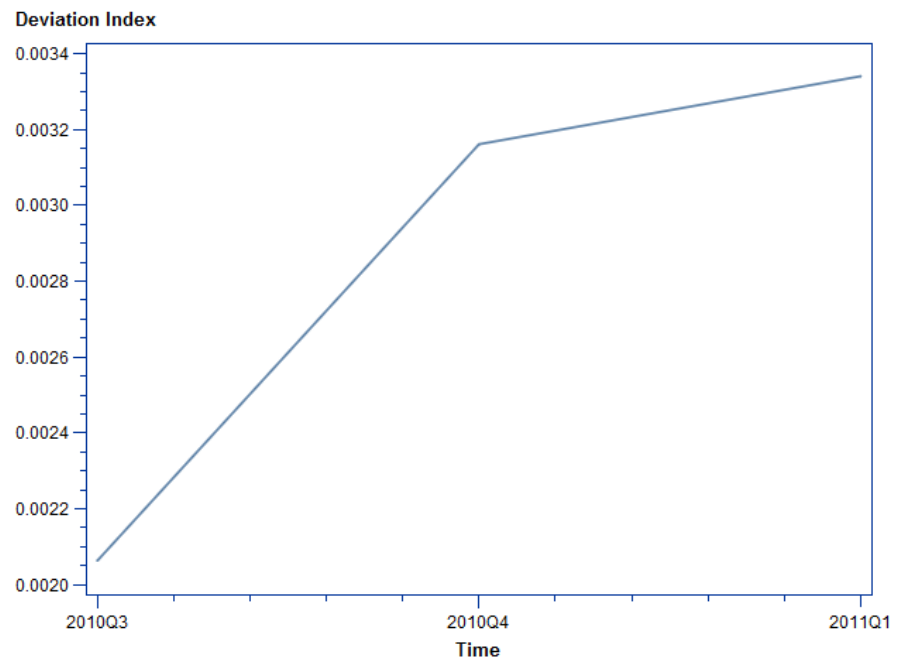
Number of Predictors with Index > 0.1 or 0.25



Stability Deviation Index

Obs	Time	Variable Name	Deviation Index
1	2010Q3	SCORE	0.0021
2	2010Q4	SCORE	0.0032
3	2011Q1	SCORE	0.0033

Population Stability Trend Chart

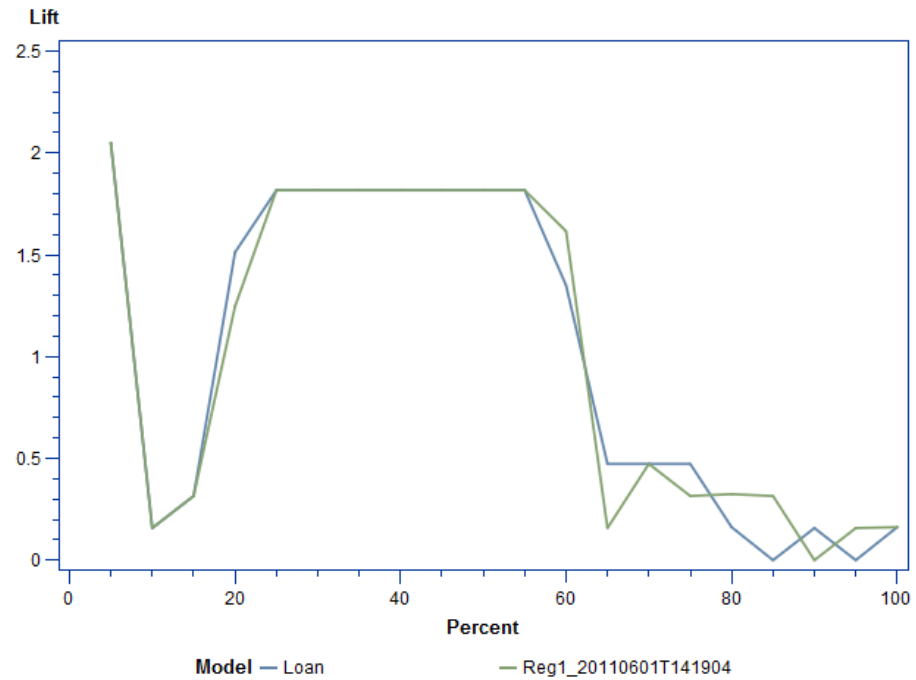


Model Retrain Comparison Report Example

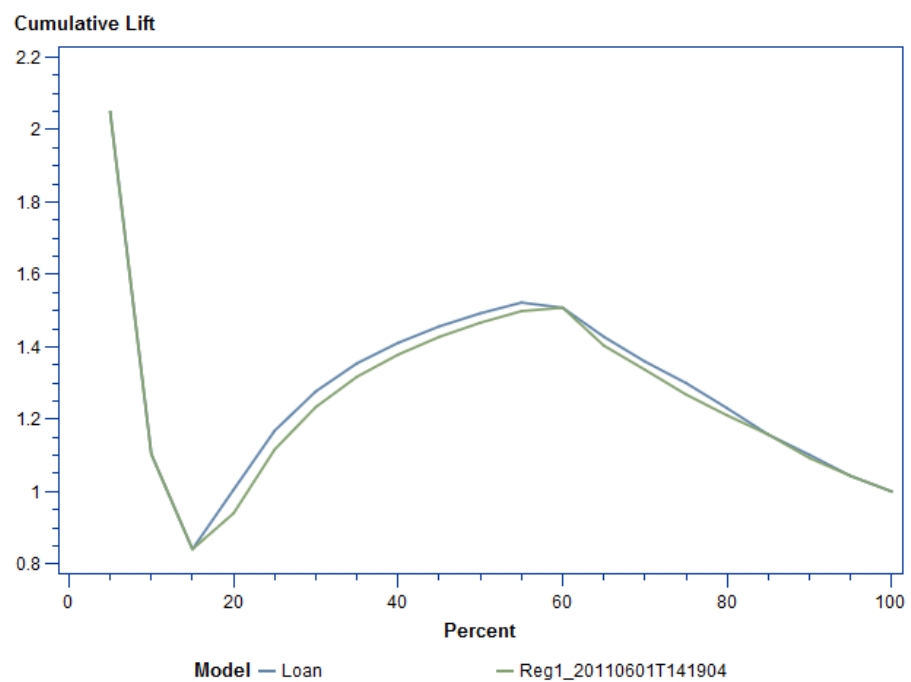
Here is an example of the model retrain comparison report for the **Loan** model, and **Reg1** retrained model.

Lift Charts

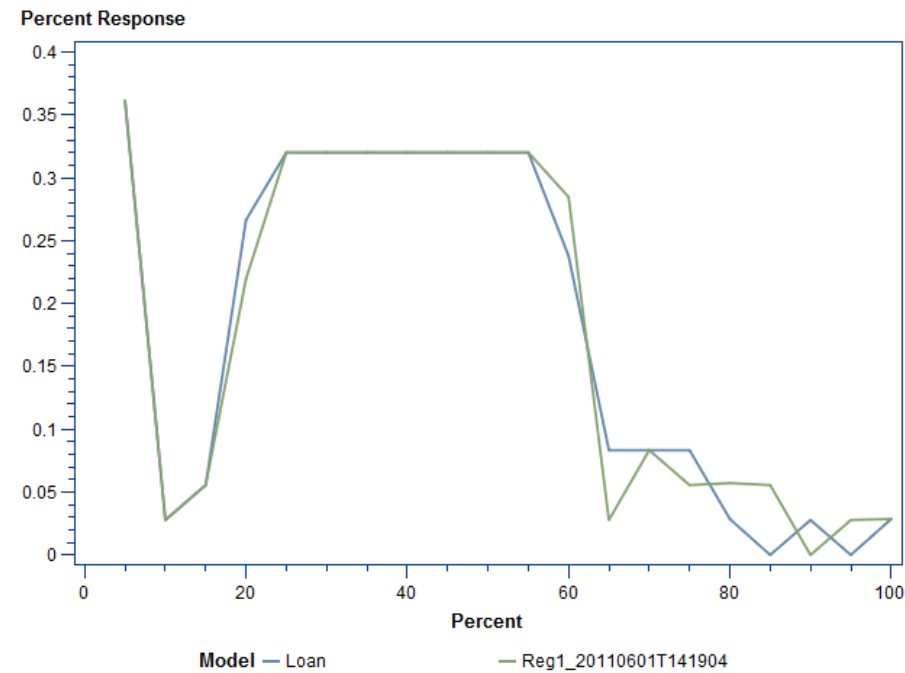
Lift Chart



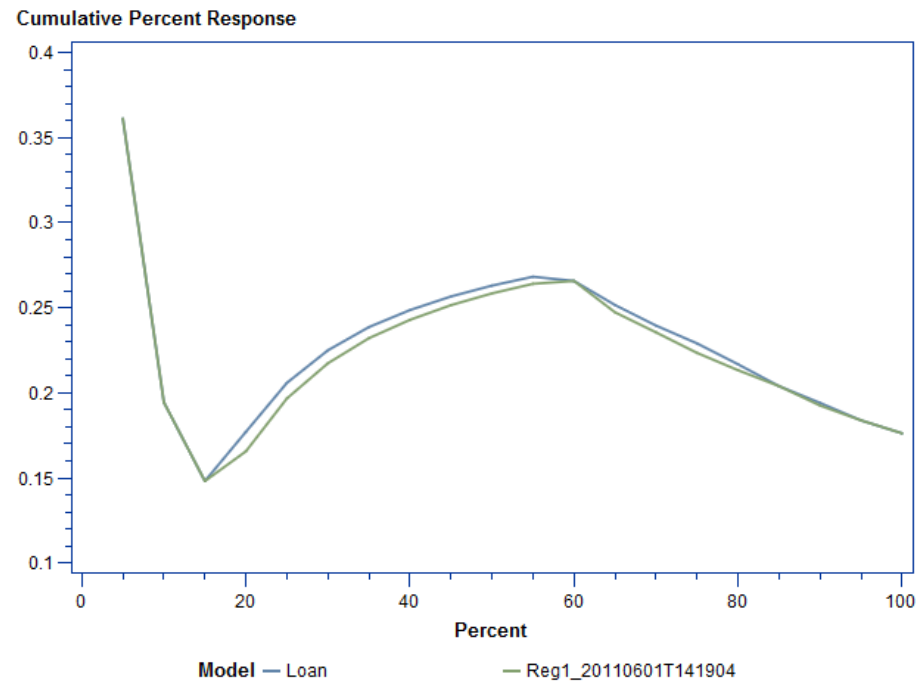
Lift Chart



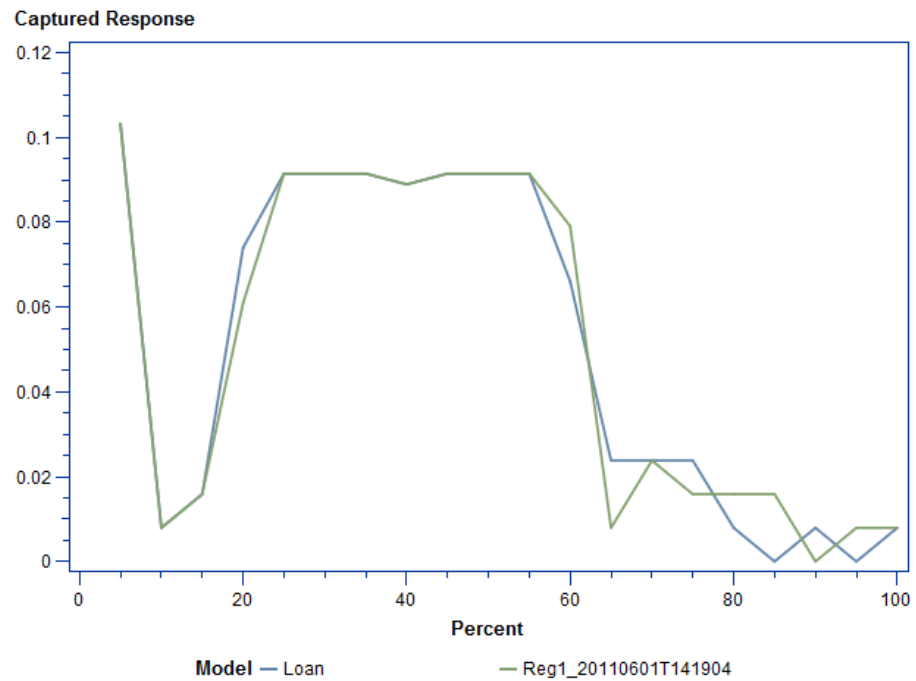
Lift Chart



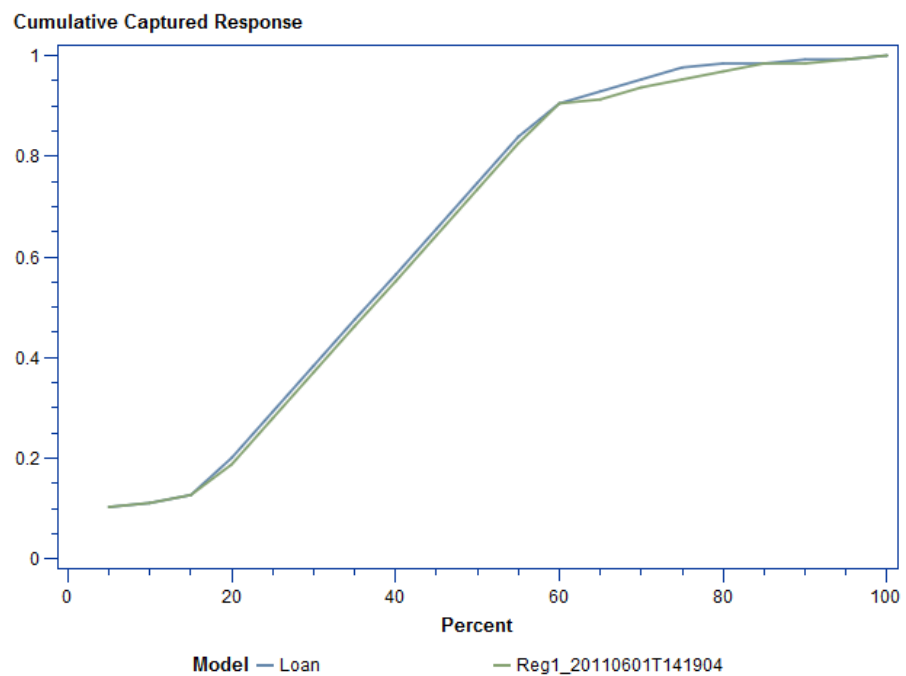
Lift Chart



Lift Chart



Lift Chart

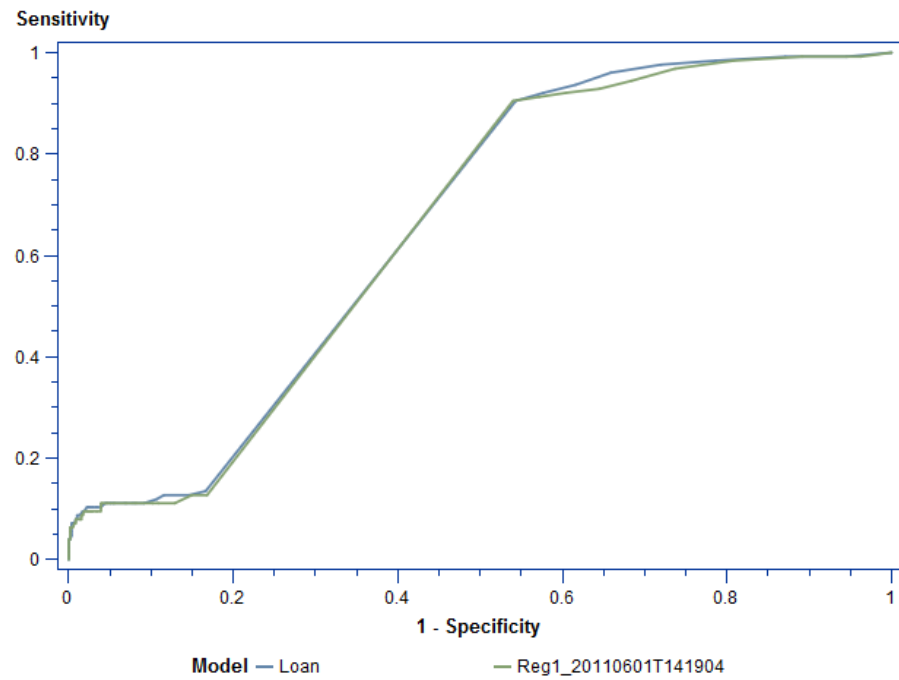


Lift Chart

Obs	Model	Cumulative Percent Response	Captured Response	Cumulative Captured Response	Percent	Count	Event Count	Percent Response	Lift	Cumulative Lift	Cumulative Event Count	Cumulative Count
1	Loan	0.3611	0.1032	0.1032	5	36	13.0000	0.3611	2.0492	2.0492	13.0000	36
2	Loan	0.1944	0.0079	0.1111	10	36	1.0000	0.0278	0.1576	1.1034	14.0000	72
3	Loan	0.1481	0.0159	0.1270	15	36	2.0000	0.0556	0.3153	0.8407	16.0000	108
4	Loan	0.1771	0.0740	0.2010	20	35	9.3234	0.2664	1.5116	1.0049	25.3234	143
5	Loan	0.2059	0.0915	0.2924	25	36	11.5248	0.3201	1.8166	1.1681	36.8482	179
6	Loan	0.2250	0.0915	0.3839	30	36	11.5248	0.3201	1.8166	1.2767	48.3729	215
7	Loan	0.2386	0.0915	0.4754	35	36	11.5248	0.3201	1.8166	1.3542	59.8977	251
8	Loan	0.2486	0.0889	0.5643	40	35	11.2046	0.3201	1.8166	1.4108	71.1023	286
9	Loan	0.2566	0.0915	0.6558	45	36	11.5248	0.3201	1.8166	1.4561	82.6271	322
10	Loan	0.2630	0.0915	0.7472	50	36	11.5248	0.3201	1.8166	1.4924	94.1518	358
11	Loan	0.2682	0.0915	0.8387	55	36	11.5248	0.3201	1.8166	1.5220	105.6766	394
12	Loan	0.2657	0.0661	0.9048	60	35	8.3234	0.2378	1.3495	1.5079	114.0000	429
13	Loan	0.2516	0.0238	0.9286	65	36	3.0000	0.0833	0.4729	1.4278	117.0000	465
14	Loan	0.2395	0.0238	0.9524	70	36	3.0000	0.0833	0.4729	1.3592	120.0000	501
15	Loan	0.2291	0.0238	0.9762	75	36	3.0000	0.0833	0.4729	1.2998	123.0000	537
16	Loan	0.2168	0.0079	0.9841	80	35	1.0000	0.0286	0.1621	1.2302	124.0000	572
17	Loan	0.2039	0.0000	0.9841	85	36	0.0000	0.0000	0.0000	1.1573	124.0000	608
18	Loan	0.1941	0.0079	0.9921	90	36	1.0000	0.0278	0.1576	1.1014	125.0000	644
19	Loan	0.1838	0.0000	0.9921	95	36	0.0000	0.0000	0.0000	1.0431	125.0000	680
20	Loan	0.1762	0.0079	1.0000	100	35	1.0000	0.0286	0.1621	1.0000	126.0000	715
21	Reg1_20110601T141904	0.3611	0.1032	0.1032	5	36	13.0000	0.3611	2.0492	2.0492	13.0000	36
22	Reg1_20110601T141904	0.1944	0.0079	0.1111	10	36	1.0000	0.0278	0.1576	1.1034	14.0000	72
23	Reg1_20110601T141904	0.1481	0.0159	0.1270	15	36	2.0000	0.0556	0.3153	0.8407	16.0000	108
24	Reg1_20110601T141904	0.1656	0.0610	0.1880	20	35	7.6832	0.2195	1.2457	0.9398	23.6832	143
25	Reg1_20110601T141904	0.1967	0.0915	0.2794	25	36	11.5248	0.3201	1.8166	1.1162	35.2079	179
26	Reg1_20110601T141904	0.2174	0.0915	0.3709	30	36	11.5248	0.3201	1.8166	1.2334	46.7327	215
27	Reg1_20110601T141904	0.2321	0.0915	0.4624	35	36	11.5248	0.3201	1.8166	1.3171	58.2574	251
28	Reg1_20110601T141904	0.2429	0.0889	0.5513	40	35	11.2046	0.3201	1.8166	1.3782	69.4620	286
29	Reg1_20110601T141904	0.2515	0.0915	0.6428	45	36	11.5248	0.3201	1.8166	1.4272	80.9868	322
30	Reg1_20110601T141904	0.2584	0.0915	0.7342	50	36	11.5248	0.3201	1.8166	1.4664	92.5116	358
31	Reg1_20110601T141904	0.2641	0.0915	0.8257	55	36	11.5248	0.3201	1.8166	1.4984	104.0363	394
32	Reg1_20110601T141904	0.2657	0.0791	0.9048	60	35	9.9637	0.2847	1.6154	1.5079	114.0000	429
33	Reg1_20110601T141904	0.2473	0.0079	0.9127	65	36	1.0000	0.0278	0.1576	1.4034	115.0000	465
34	Reg1_20110601T141904	0.2355	0.0238	0.9365	70	36	3.0000	0.0833	0.4729	1.3365	118.0000	501
35	Reg1_20110601T141904	0.2235	0.0159	0.9524	75	36	2.0000	0.0556	0.3153	1.2681	120.0000	537
36	Reg1_20110601T141904	0.2133	0.0159	0.9683	80	35	2.0000	0.0571	0.3243	1.2103	122.0000	572
37	Reg1_20110601T141904	0.2039	0.0159	0.9841	85	36	2.0000	0.0556	0.3153	1.1573	124.0000	608
38	Reg1_20110601T141904	0.1925	0.0000	0.9841	90	36	0.0000	0.0000	0.0000	1.0926	124.0000	644
39	Reg1_20110601T141904	0.1838	0.0079	0.9921	95	36	1.0000	0.0278	0.1576	1.0431	125.0000	680
40	Reg1_20110601T141904	0.1762	0.0079	1.0000	100	35	1.0000	0.0286	0.1621	1.0000	126.0000	715

ROC Charts

ROC Chart



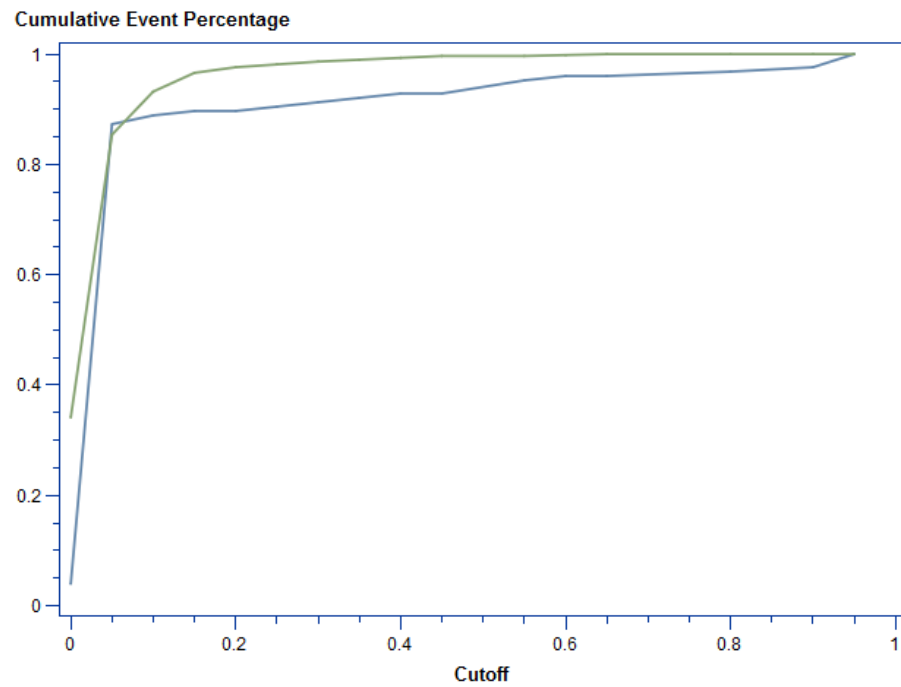
ROC Chart

Obs	Model	Gini Index	Sensitivity	1 - Specificity
1	Loan	0.3320	0.0000	0.0000
2	Loan	0.3320	0.0079	0.0000
3	Loan	0.3320	0.0159	0.0000
4	Loan	0.3320	0.0238	0.0000
5	Loan	0.3320	0.0317	0.0000
6	Loan	0.3320	0.0397	0.0000
7	Loan	0.3320	0.0397	0.0017
8	Loan	0.3320	0.0476	0.0034
9	Loan	0.3320	0.0556	0.0034
10	Loan	0.3320	0.0635	0.0034
11	Loan	0.3320	0.0714	0.0034
12	Loan	0.3320	0.0714	0.0051
13	Loan	0.3320	0.0714	0.0068
14	Loan	0.3320	0.0794	0.0085
15	Loan	0.3320	0.0794	0.0102
16	Loan	0.3320	0.0873	0.0102
17	Loan	0.3320	0.0873	0.0136
18	Loan	0.3320	0.0952	0.0170
19	Loan	0.3320	0.0952	0.0187
20	Loan	0.3320	0.1032	0.0221
21	Loan	0.3320	0.1032	0.0238
22	Loan	0.3320	0.1032	0.0255
23	Loan	0.3320	0.1032	0.0289
24	Loan	0.3320	0.1032	0.0340
25	Loan	0.3320	0.1032	0.0390

KS Charts

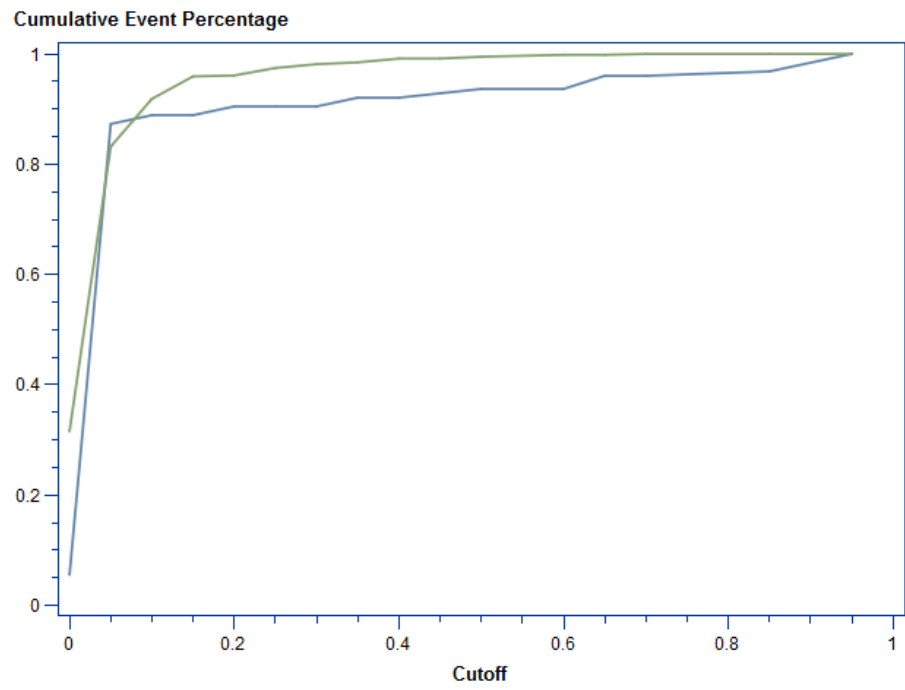
KS Chart

Model=Loan



KS Chart

Model=Reg1_20110601T141904



KS Chart

Obs	Model	Cutoff	Cumulative Event Percentage	Cumulative Non-Event Percentage	Maximum Cutoff	KS Statistic
1	Loan	0.00	0.0397	0.3413	0.08	0.3610
2	Loan	0.05	0.8730	0.8540	0.08	0.3610
3	Loan	0.10	0.8889	0.9321	0.08	0.3610
4	Loan	0.15	0.8968	0.9660	0.08	0.3610
5	Loan	0.20	0.8968	0.9762	0.08	0.3610
6	Loan	0.25	0.9048	0.9813	0.08	0.3610
7	Loan	0.30	0.9127	0.9864	0.08	0.3610
8	Loan	0.35	0.9206	0.9898	0.08	0.3610
9	Loan	0.40	0.9286	0.9932	0.08	0.3610
10	Loan	0.45	0.9286	0.9966	0.08	0.3610
11	Loan	0.55	0.9524	0.9966	0.08	0.3610
12	Loan	0.60	0.9603	0.9983	0.08	0.3610
13	Loan	0.65	0.9603	1.0000	0.08	0.3610
14	Loan	0.80	0.9683	1.0000	0.08	0.3610
15	Loan	0.90	0.9762	1.0000	0.08	0.3610
16	Loan	0.95	1.0000	1.0000	0.08	0.3610
17	Reg1_20110601T141904	0.00	0.0556	0.3158	0.09	0.3650
18	Reg1_20110601T141904	0.05	0.8730	0.8319	0.09	0.3650
19	Reg1_20110601T141904	0.10	0.8889	0.9185	0.09	0.3650
20	Reg1_20110601T141904	0.15	0.8889	0.9593	0.09	0.3650
21	Reg1_20110601T141904	0.20	0.9048	0.9610	0.09	0.3650
22	Reg1_20110601T141904	0.25	0.9048	0.9745	0.09	0.3650
23	Reg1_20110601T141904	0.30	0.9048	0.9813	0.09	0.3650
24	Reg1_20110601T141904	0.35	0.9206	0.9847	0.09	0.3650
25	Reg1_20110601T141904	0.40	0.9206	0.9915	0.09	0.3650
26	Reg1_20110601T141904	0.45	0.9286	0.9915	0.09	0.3650
27	Reg1_20110601T141904	0.50	0.9365	0.9949	0.09	0.3650
28	Reg1_20110601T141904	0.60	0.9365	0.9983	0.09	0.3650
29	Reg1_20110601T141904	0.65	0.9603	0.9983	0.09	0.3650
30	Reg1_20110601T141904	0.70	0.9603	1.0000	0.09	0.3650
31	Reg1_20110601T141904	0.85	0.9683	1.0000	0.09	0.3650
32	Reg1_20110601T141904	0.95	1.0000	1.0000	0.09	0.3650

Monitoring Performance of a Model without Score Code

If you want to monitor the performance of a model for which you no longer have the score code, you can import a model without SAS score code in the score.sas file. If the performance data set contains the predicted values, the score.sas file can be empty.

To monitor the performance of a model without score code, follow these steps:

1. Prepare the following model files:
 - XML file that defines the model input variables (inputvar.xml)
 - XML file that defines the model output variables (outputvar.xml)
 - XML file that defines the model target variables (targetvar.xml)
 - empty SAS score code file (score.sas)
2. Create a project that has a model function type of **Classification**, and create a version. You can skip this step if you have already created a project and version.
3. Verify that the following project properties are set:
 - Training Target Variable (for example, *bad*)
 - Target Event Value (for example, *1*)
 - Class Target Level as **Binary** (SAS Model Manager 3.1 or later)
 - Output Event Probability Variable (for example, *score*)
4. In the Project tree, navigate to the project's version.
MMRoot ⇒ *organizational folder* ⇒ *project folder* ⇒ *version folder*
5. Right-click the **Models** folder, and select **Import From** ⇒ **Local Files**.
Note: If the model already exists you can right-click the model name and select **Partial Import** to import an empty score.sas file, and then skip to step 11. For more information, see [“Import Partial Models” on page 136](#).
6. Navigate to the folder on your computer that contains the component files for your model.
7. Select a classification template from the **Choose a model template** list.
8. Enter a text value in the model **Name** field.
9. Complete the template fields. Drag the files from the left of the window to the corresponding file property on the right. The following files are required:
 - inputvar.xml
 - outputvar.xml
 - targetvar.xml
 - score.sas

Note: The filenames that you created for the model do not have to match the template filenames. However, the file contents must meet the file property requirements. For more information, see [“Model Template Component Files” on page 127](#) or [“Model Template Properties” on page 143](#).

10. Click **OK**. After SAS Model Manager processes the model import request, the new model appears in the **Models** folder of your project's version.
11. Select the model in the project tree, and set the model-specific properties. The value for the **Score Code Type** property must be set to **Data Step**.
12. Right-click the model, and select **Set Model Output Mapping** in order to set the output variable mappings for the model.
13. Set the model as the champion model, and set the version that contains the model as the default version. For more information, see [“Ensure That the Champion Model and Default Version Are Set” on page 244](#).
14. Before defining a performance task, verify that the performance data set is registered in SAS Management Console and contains the following variables:
 - model input variables
Note: You must have the variable columns in the table, but the values can be missing.
 - target variable
 - prediction variables
 - variables for characteristic analysis
15. Define a performance task using the performance data set that contains the predicted values. For more information, see [“Run the Define Performance Task Wizard” on page 245](#).

Glossary

activity

See workflow process activity

analytical model

a statistical model that is designed to perform a specific task or to predict the probability of a specific event.

attribute

See variable attribute

baseline

the initial performance prediction against which the output data from later tasks are compared.

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 model training

the process of building a predictive model from data.

data set

See SAS data set

data source

a table, view, or file from which you will extract information. Sources can be in any format that SAS can access, on any supported hardware platform. The metadata for a source is typically an input to a job.

DATA step

in a SAS program, a group of statements that begins with a DATA statement and that ends with either a RUN statement, another DATA statement, a PROC statement, or the end of the job. The DATA step enables you to read raw data or other SAS data sets and to create SAS data sets.

DATA step fragment

a block of SAS code that does not begin with a DATA statement. In SAS Model Manager, all SAS Enterprise Miner models use DATA step fragments in their score code.

diagram

See process flow diagram

file reference

See fileref

fileref

a name that is temporarily assigned to an external file or to an aggregate storage location such as a directory or a folder. The fileref identifies the file or the storage location to SAS.

format

See SAS format

Gini coefficient

a benchmark statistic that is a measure of the inequality of distribution, and that can be used to summarize the predictive accuracy of a model.

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.

informat

See SAS informat

input variable

a variable that is used in a data mining process to predict the value of one or more target variables.

instance

See workflow process instance

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.

library reference

See libref

libref

a SAS name that is associated with the location of a SAS library. For example, in the name MYLIB.MYFILE, MYLIB is the libref, and MYFILE is a file in the SAS library.

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

descriptive data about data that is stored and managed in a database, in order to facilitate access to captured and archived data for further use.

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.

model scoring

the process of applying a model to new data in order to compute outputs.

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.

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

See SAS package file

package file

See SAS package file

participant

See workflow participant

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.

PFD

See process flow diagram

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

See workflow process

process definition

See workflow process definition

process flow diagram

a graphical sequence of interconnected symbols that represent an ordered set of steps or tasks that, when combined, form a process designed to yield an analytical result.

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 of the characteristics of a component that collectively determine the component's appearance and behavior. Examples of types of properties are attributes and methods.

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.

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 chart

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.

SAS format

a type of SAS language element that applies a pattern to or executes instructions for a data value to be displayed or written as output. Types of formats correspond to the data's type: numeric, character, date, time, or timestamp. The ability to create user-defined formats is also supported. Examples of SAS formats are BINARY and DATE. Short form: format.

SAS informat

a type of SAS language element that applies a pattern to or executes instructions for a data value to be read as input. Types of informats correspond to the data's type: numeric, character, date, time, or timestamp. The ability to create user-defined informats is also supported. Examples of SAS informats are BINARY and DATE. Short form: informat.

SAS Metadata Repository

a container for metadata that is managed by the SAS Metadata Server.

SAS package file

a container for data that has been generated or collected for delivery to consumers by the SAS Publishing Framework. Packages can contain SAS files, binary files, HTML files, URLs, text files, viewer files, and metadata.

SAS publication channel

See publication channel

SAS variable

a column in a SAS data set or in a SAS data view. The data values for each variable describe a single characteristic for all observations (rows).

scoring

See model scoring

scoring function

a user-defined function that is created by the SAS Scoring Accelerator from a scoring model and that is deployed inside the database.

scoring task

a 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 are associated with an attribute of interest.

source

See data source

SPK

See SAS package file

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 data

data that contains input values and target values that are used to train and build predictive models.

universal unique identifier

a number that is used to uniquely identify information in distributed systems without significant central coordination. There are 32 hexadecimal digits in a UUID, and these are divided into five groups with hyphens between them as follows: 8-4-4-4-12. Altogether the 16-byte (128 bit) canonical UUID has 32 digits and 4 hyphens, or 36 characters.

UUID

See universal unique identifier

variable

See SAS variable

variable attribute

any of the following characteristics that are associated with a particular variable: name, label, format, informat, data type, and length.

version folder

a folder in the Project Tree that typically represents a time phase and that contains models, scoring tasks, life cycle data, reports, documents, resources, and model performance output.

view

a particular representation of a model's data.

workflow

a model for a sequence of activities, declared as work of a person, a group, an organization, or one or more mechanisms. Workflows are generally designed to enable a work process that can be documented and learned.

workflow participant

a user, group, or role that is assigned to an activity of a workflow instance.

workflow process

a set of one or more linked activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships.

workflow process activity

a task or step in a workflow process.

workflow process definition

an activated process template that is available in the SAS Workflow Engine for use. Process definitions contain the set of activities, participants, policies, statuses, and operands that comprise a business task.

workflow process instance

a running process in the SAS Workflow Engine or a working version of a process definition.

Index

Special Characters

%MM_AddModelFile macro [317](#)
 %MM_CreateModelDataset macro [347](#)
 %MM_GetModel macro [269](#)
 %MM_GetModelFile macro [320](#)
 %MM_GetModels macro [268](#), [269](#)
 %MM_GetURL macro [324](#)
 %MM_Register macro [325](#)
 %MM_RegisterByFolder macro [342](#)
 %MM_RunReports macro [274](#)
 macro variables used by [272](#)

A

access macros [311](#)
 %MM_AddModelFile [317](#)
 %MM_CreateModelDataset [347](#)
 %MM_GetModelFile [320](#)
 %MM_GetURL [324](#)
 %MM_Register [325](#)
 %MM_RegisterByFolder [342](#)
 accessing [312](#)
 dictionary of [317](#)
 global macro variables and [315](#)
 identifying files used by [313](#)
 identifying model repository objects
 [312](#)
 required global macro variables [316](#)
 required tables [313](#)
 ad hoc reports [177](#), [179](#)
 compared with user-defined reports [178](#)
 creating [179](#)
 example [180](#)
 alert notifications [234](#)
 analytical model
 query [307](#)
 specifying for model import [144](#)
 template [125](#)
 Approvers
 groups as, for life cycle templates [66](#)
 life cycle template participants [66](#)

Model Manager Example Life Cycle

 Approvers [18](#)

assessing models

 tasks by user groups [21](#)

Assessment Charts [237](#)

Assignees

 groups as, for life cycle templates [66](#)

 life cycle template participants [66](#)

 Model Manager Example Life Cycle

 Assignees [18](#)

associating documents with folders [41](#)

attaching documents [41](#)

B

batch performance reports [253](#)
 accessing performance data set [274](#)
 copying example batch programs [255](#)
 creating folder structure for [253](#)
 defining report folders and data sets
 [272](#)
 defining specifications [257](#)
 e-mail recipient specifications [260](#)
 encoding passwords [273](#)
 example code [275](#)
 export channel for [255](#)
 extracting champion model from a
 channel [268](#)
 job scheduling specifications [264](#)
 librefs for running [271](#)
 performance data for [255](#)
 prerequisites for running [253](#)
 project specifications [257](#)
 publishing champion model from
 project folder [253](#)
 report output in production mode [256](#)
 report output in test mode [256](#)
 report specifications [261](#), [265](#)
 SAS code for running [271](#)
 user ID and password for [256](#)
 browse templates [133](#)

Browse Templates window 67, 126

C

champion models 194
 clearing 195
 default version 198
 deploying 194
 exporting 209
 extracting from a channel 268
 monitoring performance tasks by user groups 22
 monitoring process 236
 performance monitoring reports 227
 publishing for batch performance reports 253
 replacing/retiring 223
 requirements for 194
 selecting new default version 224
 setting 195
 setting status 53
 channels
 export channel for batch performance reports 255
 extracting champion model from 268
 publishing models to 205
 Characteristic reports 228, 230
 example 231
 overview 230
 performance index warnings and alerts 234
 classification model
 query 307
 specifying for model import 144
 template 125
 clustering model
 query 307
 specifying for model import 144
 columns
 managing 89
 comparison reports
 See [model comparison reports](#)
 component files
 model templates 127
 content files, scoring task 163
 current.sas7bdat data set 269

D

dashboard reports 279
 defining indicators 279
 generate 283
 viewing 285
 Data Composition reports 228, 230
 Characteristic report 228, 230
 Stability report 228, 230, 231

data sets 6

 containing model information 347
 current.sas7bdat 269
 performance data sets 242, 274
 performance tables 32
 project input tables 29
 project output tables 30
 scoring task input tables 30

data sources 27

 performance table 35
 project input table 32
 project output table 32
 registering 28
 scoring task input tables 34
 scoring task output tables 34
 tables 4, 27
 test table 35

Data Sources perspective 9, 13

DATA step

 accessing performance data set 274

default version

 clearing 200
 for projects 198
 selecting a new default version 224
 setting 198

Define Performance Task wizard

 naming performance tables for use with 36
 prerequisites for running 243
 running 245

degradation of models 229

delivering models

See also [model delivery](#)
 tasks by user groups 22

Delta reports 171

deploying models 193

 champion models 194
 default version for projects 198
 freezing models 196
 tasks by user groups 22

documents

 associating with folders 41
 attaching to folders 41
 saving 41
 showing versions 41
 viewing 41

Dynamic Lift reports 173

 creating 174
 creating test tables 35
 verifying model properties 173
 verifying project properties 173

E

Edit menu 17

encoding passwords 273

- environment, operational 4
- export channel
 - for batch performance reports 255
- exporting
 - models 208
 - project champion models 209
 - verifying model exports 210
- extracting
 - champion model from a channel 268
 - published models 207

F

- File menu 16
- filtering
 - managing filters 93
 - retrieving and applying 93
 - save filter criteria 92
- folders
 - associating documents with 41
 - attaching documents to 41
 - creating organizational folders 40
 - project folder organization 44
 - project folder tasks 44
 - structure for batch performance reports 253
 - version folder organization 60
 - version folder tasks 61
- freezing
 - models 196
 - versions 197

G

- general properties 357
- general tasks 23
- Gini plots 232
- Gini Trend Chart 238
- global macro variables 315
- graphing scoring task results 159
- groups 4, 18

H

- Help menu 18

I

- importing models 119
 - from metadata repository 120
 - mapping model variables to project variables 138
 - model templates 125
 - package files from SAS Enterprise Miner 122
 - partial models 136

- PMML models 135
- R model 134
- SAS code models 123, 133
- setting model properties 137
- tasks by user groups 21
- user-defined model templates 139

- In-Database Scoring 211

- indexes

- warnings and alerts 234

- input data variable distribution shifts 230

- input variables 48, 51

- input variables 199

J

- job scheduling specifications
 - batch performance reports 264

K

- Kolmogorov-Smirnov (KS) plots 233
- KS Chart 238
- KS reports 233
- KS Trend Chart 238

L

- librefs

- %MM_CreateModelDataset macro 347

- %MM_GetModelFile macro 317

- %MM_Register macro 325

- %MM_RunReports macro 274

- access batch performance data sets 274

- batch performance reports 257

- R model 372

- running batch performance reports 271

- SAS Model Manager access macros 311, 313

- user-defined report 184, 362

- Life Cycle perspective 9

- life cycle templates 62

- accessing 67

- adding milestones 70

- creating 62, 69

- creating from a sample 68

- deleting 67

- groups as Assignees and Approvers 66

- milestone properties 73

- modifying 71

- participants 65

- properties 72

- SAS Model Manager Template Editor window 63

- saving 67

- selecting participants 66

- task properties 74

- template properties 72
- viewing 80
- life cycles 79
 - definition 79
 - milestone organization 79
 - properties 82
 - searching for tasks assigned to users 307
 - tasks 80
 - updating milestone status 81
- Life Cycles perspective 12
- life style templates
 - adding tasks 71
- Lift Trend chart 238
- Local Files method
 - importing R models 134
 - importing SAS code models 123
- logs
 - publishing scoring functions 221

M

- macro variables
 - %MM_RunReports macro 272
 - ad hoc report 180
 - defining for user-defined reports 182
 - description of 351
 - global 184, 315
 - optional for report monitoring 273
 - required for access macros 316
 - to define report folders and data sets 272
- macros
 - See also* access macros
 - accessing report macros 269, 271
- manage
 - workflow 105
- managing
 - columns 89
- mapping variables
 - model variables to project variables 138
 - scoring task output variables 157
- menus 10, 16
- metadata
 - project metadata 54
- metadata repository
 - importing models from 120
 - viewing MiningResult objects in 210
- metadata tables
 - publishing scoring functions 222
- milestones
 - adding to life cycle template 69
 - organization of life cycle milestones 79
 - specific properties for 365
 - updating status 81
- MiningResult objects 208
 - viewing in metadata repository 210
- Model Assessment reports
 - performance index warnings and alerts 235
- model comparison reports 167
 - Delta 171
 - Dynamic Lift 173
 - input files 168
 - Model Profile 169
 - output files 168
 - viewing 176
- model component files 123, 317
 - import partial models 136
 - importing R model 134
 - model templates 125
 - naming for model template 140
 - project planning 46
 - R model 374
 - SAS package file 132
 - specifying for model import 133
 - used by access macros 313
- model delivery 203
 - exporting models 208
 - publishing models 204
 - publishing scoring functions 210
- model function types 57, 361
- Model Input Variable Report 229
- model management process 6
- Model Manager Example Life Cycle
 - Approvers 18
- Model Manager Example Life Cycle
 - Assignees 18
- Model Monitoring reports 228, 231
 - KS reports 233
 - monitoring Gini (ROC and Trend) reports 232
 - monitoring Lift reports 231
- Model Profile reports 169
 - creating 169
- model repository objects 312
- Model Target Variable Report 229
- model templates 125, 139
 - component files 127
 - creating 139
 - FileList properties 144
 - modifying 141
 - properties 143
 - system and user properties 145
 - template properties 143
 - user model templates 126
 - user-defined 139
- model variables
 - mapping to project variables 138
- models
 - See also* champion models
 - assessing tasks by user group 21

- data sets containing model information 347
- degradation of 229
- deploying 193
- deploying and delivering tasks by user groups 22
- exporting 208
- extracting published models 207
- freezing 196
- function types 47
- importing 119
- importing tasks by user groups 21
- managing 3
- mapping model variables to project variables 138
- process of managing 6
- publishing 204, 205
- registering 325, 342
- searching for 304
- specific properties for 145, 366
- unfreezing 196
- validating with model comparison reports 167
- validating with user reports 177
- verifying exports 210
- monitoring champion model performance 227
 - tasks by user groups 22
- monitoring Lift reports 231
- Monitoring reports 237
 - creating 238
 - output files 239
- monitoring ROC & Gini reports 232

N

- naming performance tables 36

O

- objects
 - deleting in Project Tree 42
- opening
 - objects 97
- operational environment 4
- organization-specific user-defined properties 362
- organizational folders 40
- output data variable distribution shifts 231
- output variables 48, 51, 199

P

- package files 122
 - creating 122

- importing from SAS Enterprise Miner 122
 - publishing models 204
- Partial Model Import utility 136
- partial models
 - importing 136
- participants
 - selecting for life cycle templates 66
- passwords
 - encoding 273
 - for batch performance reports 256
- performance
 - data for batch performance reports 255
 - monitoring champion model
 - performance by user groups 22
- performance data sets
 - creating performance reports 242
 - DATA step for accessing 274
- performance indexes
 - warnings and alerts 234
- performance indicator
 - See [dashboard reports](#)
- performance monitoring reports 227
 - See also [batch performance reports](#)
 - Data Composition reports 228, 230
 - formatting 237
 - Model Monitoring reports 228, 231
 - Monitoring reports 237
 - performance index warnings and alerts 234
 - SAS programs for creating 252
 - Summary reports 228, 229
 - types of 228
 - viewing 240
- performance reports 241
 - See also [batch performance reports](#)
 - creating with performance tasks 241
 - performance data sets and 242
 - prerequisites for running Define
 - Performance Task wizard 243
 - running Define Performance Task wizard 245
- performance tables 28, 32, 35
 - creating 35
 - naming for use with Define
 - Performance Task wizard 36
- performance tasks
 - creating reports with 241
 - prerequisites for running Define
 - Performance Task wizard 243
 - running Define Performance Task wizard 245
- perspectives 9
 - Data Sources 13
 - Life Cycle 12
 - Projects 10

- planning projects 46
 - PMML models
 - importing 135
 - prediction model
 - query 307
 - specifying for model import 144
 - template 125
 - preferences
 - Workflow Console 94
 - production mode
 - batch performance reports 256
 - Production Models Aging Report 229
 - profile data 169
 - project folders
 - organization of 44
 - publishing champion model from 253
 - tasks 44
 - project input tables 27, 29
 - creating 32
 - project metadata 54
 - project output tables 27, 30
 - creating 33
 - project properties 54
 - project tables 29
 - performance tables 32
 - project input tables 29
 - project output tables 30
 - scoring task input tables 30
 - scoring task output tables 30
 - test tables 31
 - train tables 31
 - Project Tree 10
 - associating documents with folders 41
 - creating organizational folders 40
 - deleting objects in 42
 - organizing 39
 - project variables 199
 - mapping to model variables 138
 - planning 46
 - projects 43
 - controlling access to versions 196
 - creating 48
 - default version for 198
 - exporting champion models 209
 - input variables 48, 51
 - lock metadata 53
 - model function types 47
 - modify 51
 - output variables 48, 51
 - planning 46
 - prerequisites for creating 47
 - properties 47
 - publishing scoring functions 210
 - retiring 224
 - setting champion model status 53
 - setup tasks by user groups 20
 - specific properties for 359
 - unlock metadata 53
 - Projects perspective 9, 10
 - properties
 - Dynamic Lift reports 173
 - general 357
 - life cycle properties 82
 - life cycle templates 72
 - model function types 57, 361
 - model template properties 143
 - organization-specific user-defined 362
 - project properties 54
 - result set properties 164, 369
 - scoring task properties 163, 368
 - setting for imported models 137
 - specific for a model 145, 366
 - specific for a project 359
 - specific for a version 363
 - specific for milestones and tasks 365
 - system properties 358
 - user-defined 362
 - version properties 77
 - prototype tables 27
 - project input tables 29
 - project output tables 30
 - scoring task output tables 30, 34
 - Publish Scoring Function 210
 - Publish Scoring Function window 214
 - publishing
 - champion model, for batch performance reports 253
 - extracting published models 207
 - models 204
 - models to a channel 205
 - scoring functions 210
- ## Q
- Query utility 303
 - searching for models 304
 - searching life cycles for tasks assigned to users 307
 - searching with UUID 306
- ## R
- R model 134
 - R models
 - building 372
 - model component files 374
 - model template file 373
 - using in SAS Model Manager 371
 - registering
 - data sources 28
 - models 325, 342
 - relational databases 6

- report macros
 - accessing 269, 271
 - report templates
 - for user-defined reports 183
 - reports
 - See also* performance reports
 - ad hoc reports 177, 179
 - batch performance reports 253
 - Characteristic reports 228, 230
 - creating with performance tasks 241
 - dashboard 279
 - Data Composition reports 228, 230
 - Delta reports 171
 - Dynamic Lift reports 173
 - KS reports 233
 - Model Assessment reports 235
 - model comparison reports 167
 - Model Input Variable Report 229
 - Model Monitoring reports 228, 231
 - Model Profile reports 169
 - Model Target Variable Report 229
 - monitoring Lift reports 231
 - Monitoring reports 237
 - monitoring ROC & Gini reports 232
 - performance monitoring reports 227
 - Production Models Aging Report 229
 - Stability reports 228, 230, 231
 - Summary of Reports 229
 - Summary reports 228, 229
 - user reports 177
 - user-defined reports 177, 182
 - repository
 - accessing files in 320
 - model repository objects 312
 - result set properties 164, 369
 - retiring
 - champion models 223
 - projects 224
 - retrain model 289
 - define task 290
 - execute 296
 - prerequisites 290
 - view comparison report 297
 - view models 297
 - ROC Chart 238
 - ROC plots 232
 - roles 19
 - life cycle template participant roles 65
- S**
- SAS code models
 - importing 123, 133
 - SAS Content Server 5
 - SAS Enterprise Miner
 - importing package files from 122
 - SAS Foundation 6
 - SAS Management Console 5
 - SAS Metadata Server 6
 - SAS Model Manager
 - interface 9
 - managing models 3
 - model management process 6
 - operational environment 4
 - services provided by 3
 - setup tasks by user groups 20
 - user groups 4
 - SAS Model Manager Client 5
 - SAS Model Manager Middle Tier Server 5
 - SAS Model Manager Server 6
 - SAS Model Manager Template Editor
 - window 63
 - SAS Model Manager window 9
 - layout 9
 - perspectives 9, 10
 - toolbar and menus 10
 - SAS programs
 - creating performance monitoring reports 252
 - delete from SAS Content Server 185
 - edit on SAS Content Server 185
 - upload to SAS Content Server 183
 - user-defined report 182
 - SAS user-defined properties 362
 - SAS Web Infrastructure Platform 5
 - SAS Workspace Server 6
 - saving documents 41
 - scoring functions 210
 - log messages for publishing 221
 - metadata tables 222
 - prerequisites for publishing 214
 - process flow 212
 - publishing 210
 - steps for publishing 217
 - scoring output tables 153
 - creating 153
 - scoring task content files 163
 - scoring task input tables 28, 30, 34
 - creating 34
 - scoring task output tables 27, 30, 34
 - adding to SAS Model Manager 31
 - creating 34
 - scoring tasks 149
 - creating 155
 - creating scoring output tables 153
 - executing 158
 - generated content files 163
 - graphing results 159
 - mapping output variables 157
 - modifying 156
 - properties 163, 368

- result set properties 164, 369
- tabbed views 151
- searches
 - for life cycle tasks assigned to users 307
 - for models 304
 - with UUID 306
- segmentation model
 - template 125
- setup
 - of projects and versions by user groups 20
- setup tasks by user groups
 - SAS Model Manager 20
- sorting
 - multiple columns 90
 - single columns 90
- SPK files
 - See [package files](#)
- Stability reports 228, 230, 231
 - example 231
 - overview 230
 - performance index warnings and alerts 235
- Summary of Reports 229
- Summary reports 228, 229
- system properties 358

T

- tabbed views
 - scoring tasks 151
- tasks
 - adding to life cycle template 69
 - creating reports using performance tasks 241
 - general, by user groups 23
 - life cycle tasks 80
 - project folder tasks 44
 - scoring task properties 163, 368
 - scoring tasks 149
 - searching for life cycle tasks assigned to users 307
 - specific properties for 365
 - version folder tasks 61
- templates
 - See also [model templates](#)
 - analytical model 125
 - classification model 125
 - creating from a sample 68
 - life cycle 62
 - prediction model 125
 - report templates 183
 - segmentation model 125
 - user model templates 126
 - user-defined model templates 139

- viewing 133
- test mode
 - for batch performance reports 256
- test tables 28, 31
 - creating 35
- thresholds
 - warnings and alerts 234
- toolbar 10, 14
- train tables 28, 31

U

- unfreezing
 - models 196
 - versions 197
- URL addresses 324
- user groups 4, 18
- user ID
 - for batch performance reports 256
- user model templates 126
- user reports 177
 - ad hoc 177, 179
 - output created by 178
 - user-defined 177, 182
 - validating models with 177
- user-defined model templates 139
- user-defined properties 362
 - organization-specific 362
 - SAS 362
- user-defined reports 177, 182
 - compared with ad hoc reports 178
 - creating 182
 - example 186
 - macro variables 182
 - report template for 183
 - running 186
- users
 - searching for life cycle tasks assigned to 307
- UUIDs
 - searching with 306
 - translating to URL address 324

V

- validating models
 - with model comparison reports 167
 - with user reports 177
- verifying model exports 210
- version folders
 - organization of 60
 - tasks 61
- versions 59
 - clearing default version 200
 - controlling access to project versions 196

- creating 76
 - creating life cycle templates 62
 - default version for projects 198
 - freezing 197
 - properties 77
 - SAS Model Manager functionality for 60
 - selecting new default version 224
 - setting default version 198
 - setup tasks by user groups 20
 - showing document versions 41
 - specific properties for 363
 - unfreezing 197
 - View menu 17
 - viewing
 - documents 41
 - life cycle templates 80
 - model comparison reports 176
 - performance monitoring reports 240
- W**
- warning notifications 234
 - workflow activity
 - adding comments 102
 - edit properties 100
 - release 115
 - replying to comments 103
 - searching comments 104
 - sorting and filtering comments 104
 - workflow console
 - customizing category views 88
 - filtering 91
 - Workflow Console 86
 - activities 100
 - alert notifications 95
 - comments 101
 - create instance 107
 - edit instance 110
 - manage 105
 - navigation pane 88
 - object bar 96
 - opening objects 97
 - participants 112
 - rearranging objects 98
 - setting preferences 94
 - user interface layout 86
 - view activities 99
 - view instances 109
 - view process definitions 106
 - window controls 87
 - workflow instance
 - adding comments 102
 - assign participant 112
 - create 107
 - edit 110
 - edit properties 112
 - release activity 115
 - remove participant 114
 - replying to comments 103
 - searching comments 104
 - sorting and filtering comments 104
 - terminate 116
 - workflow participant
 - assign 112
 - remove 114
 - workspaces
 - saving 98

