



THE  
POWER  
TO KNOW.

# **SAS<sup>®</sup> Model Manager 12.3**

## **User's Guide**

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *SAS® Model Manager 12.3: User's Guide*. Cary, NC: SAS Institute Inc.

**SAS® Model Manager 12.3: User's Guide**

Copyright © 2013, SAS Institute Inc., Cary, NC, USA

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

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

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

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 52.227-19, Commercial Computer Software-Restricted Rights (June 1987).

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

November 2014

SAS provides a complete selection of books and electronic products to help customers use SAS® software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit [support.sas.com/bookstore](http://support.sas.com/bookstore) 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>ix</i>
<i>What's New in SAS Model Manager 12.3</i> . . . . .	<i>xi</i>
<i>Accessibility Features of SAS Model Manager</i> . . . . .	<i>xv</i>
<i>Recommended Reading</i> . . . . .	<i>xxvii</i>

## PART 1 What Is SAS Model Manager? 1

<b>Chapter 1 • Overview of SAS Model Manager</b> . . . . .	<b>3</b>
Managing Models Using SAS Model Manager . . . . .	3
The SAS Model Manager Operational Environment . . . . .	5
Model Management Process . . . . .	6
<b>Chapter 2 • Introduction to SAS Model Manager</b> . . . . .	<b>9</b>
Layout of the SAS Model Manager Window . . . . .	9
SAS Model Manager User Groups, Roles, and Tasks . . . . .	20

## PART 2 Working with Projects and Versions 29

<b>Chapter 3 • Working with Data Sources</b> . . . . .	<b>31</b>
Overview of Data Sources . . . . .	31
Project Tables . . . . .	33
Creating Project Input and Output Tables . . . . .	37
Creating Scoring Task Input and Output Tables . . . . .	39
Creating a Test Table . . . . .	40
Creating a Performance Table . . . . .	40
Using Tables from a Local or Network Drive . . . . .	42
<b>Chapter 4 • Organizing the Project Tree</b> . . . . .	<b>47</b>
Overview of the Project Tree . . . . .	47
Create an Organizational Folder . . . . .	48
Associate Documents with a Folder . . . . .	49
Deleting an Object in the Project Tree . . . . .	52
Archive and Restore Organizational Folders . . . . .	52
<b>Chapter 5 • Working with Projects</b> . . . . .	<b>55</b>
Overview of Projects . . . . .	55
Planning a Project . . . . .	58
Prerequisites for Creating Projects . . . . .	60
About Defining Project Input and Output Variables . . . . .	61
Create a Project . . . . .	61
Modify Project Definition . . . . .	64
Lock or Unlock Project Metadata . . . . .	66
Setting the Project Champion Model Status . . . . .	66
Project Properties . . . . .	67

<b>Chapter 6 • Working with Versions</b> .....	<b>71</b>
Overview of Versions .....	71
Creating Life Cycle Templates .....	75
Create a Version .....	89
Version Properties .....	90
Working with Life Cycles .....	92
<b>Chapter 7 • Working with Project Control Groups</b> .....	<b>99</b>
Overview of Project Control Groups .....	99
Planning a Project Control Group .....	100
Prerequisites for Creating Project Control Groups .....	101
Creating a Project Control Table .....	103
Create a Project Control Group .....	103
Create Projects from a Control Table .....	104
Add a New Version .....	113
Add an Input Variable .....	114
Publish Project Champion Models from a Project Control Group .....	115
Monitor Performance of Project Champion Models .....	117
<b>PART 3 Importing, Scoring, and Validating Models</b>	<b>123</b>
<b>Chapter 8 • Importing Models</b> .....	<b>125</b>
Overview of Importing Models .....	125
Import Models from the SAS Metadata Repository .....	127
Import SAS Model Package Files .....	128
Import SAS Code Models and R Models Using Local Files .....	130
Import PMML Models .....	143
Import Partial Models .....	144
Set Model Properties .....	145
Map Model Variables to Project Variables .....	146
User-Defined Model Templates .....	148
Specific Properties for a Model .....	154
<b>Chapter 9 • Scoring Models</b> .....	<b>157</b>
Overview of Scoring Tasks .....	157
Scoring Task Tabbed Views .....	159
Create Scoring Output Tables .....	162
Create a Scoring Task .....	164
Modify a Scoring Task .....	167
Map Scoring Task Output Variables .....	167
Execute a Scoring Task .....	168
Schedule Scoring Tasks .....	170
Graph Scoring Task Results .....	172
Generated Scoring Task Content Files .....	175
Scoring Task Properties .....	176
Result Set Properties .....	177
<b>Chapter 10 • Validating Models Using Reports</b> .....	<b>179</b>
Overview of Model Comparison, Validation, and Summary Reports .....	180
Model Profile Reports .....	183
Delta Reports .....	184
Dynamic Lift Reports .....	186
Interval Target Variable Report .....	189
Basel II Reports .....	191

Training Summary Data Set Reports . . . . .	196
View Reports . . . . .	198
<b>Chapter 11 • Validating Models Using User Reports . . . . .</b>	<b>199</b>
Overview of User Reports . . . . .	199
Ad Hoc Reports . . . . .	201
User-Defined Reports . . . . .	204
<b>PART 4 Deploying and Publishing Models 213</b>	
<b>Chapter 12 • Deploying Models . . . . .</b>	<b>215</b>
Overview of Deploying Models . . . . .	215
Champion Models . . . . .	216
Challenger Models . . . . .	219
Freezing Models . . . . .	220
<b>Chapter 13 • Publishing Models . . . . .</b>	<b>223</b>
Overview of Publishing Models . . . . .	223
Publishing Models to a SAS Channel . . . . .	224
Publish Models to the SAS Metadata Repository . . . . .	228
Publishing Models to a Database . . . . .	231
Remove Models from a Database . . . . .	244
View Publish History . . . . .	246
<b>Chapter 14 • Replacing a Champion Model . . . . .</b>	<b>247</b>
Overview of Replacing a Champion Model . . . . .	247
Retire a Project . . . . .	248
<b>PART 5 Performance Monitoring and Retraining Models</b>	
<b>249</b>	
<b>Chapter 15 • What is Performance Monitoring? . . . . .</b>	<b>251</b>
Overview of Performance Monitoring . . . . .	251
Types of Performance Monitoring . . . . .	252
Performance Index Warnings and Alerts . . . . .	259
The Process of Monitoring Champion Models . . . . .	260
<b>Chapter 16 • Create Reports by Defining a Performance Task . . . . .</b>	<b>263</b>
Overview of Creating Reports Using a Performance Task . . . . .	263
Prerequisites for Running the Define Performance Task Wizard . . . . .	266
Run the Define Performance Task Wizard . . . . .	268
Schedule Performance Monitoring Tasks . . . . .	273
View Performance Monitoring Job History . . . . .	275
Delete Performance Summary Data Sets . . . . .	276
<b>Chapter 17 • Create Reports Using Batch Programs . . . . .</b>	<b>277</b>
Overview of SAS Programs to Monitor Model Performance . . . . .	278
Prerequisites for Running Batch Performance Reports . . . . .	279
Report Output in Test and Production Modes . . . . .	282
Define the Report Specifications . . . . .	283
Extracting the Champion Model from a Channel . . . . .	294

SAS Code to Run Performance Reports .....	297
<b>Chapter 18 • Formatting Performance Reports .....</b>	<b>305</b>
Format a Monitoring Report .....	305
Format a Champion and Challenger Performance Report .....	308
Performance Report Output Files .....	310
View Reports .....	311
<b>Chapter 19 • Using Dashboard Reports .....</b>	<b>313</b>
Overview of Project Dashboard Reports .....	313
Create a Dashboard Report Definition .....	314
Generate Dashboard Reports .....	319
View Dashboard Reports .....	320
Edit a Dashboard Report Definition .....	324
Manage All Project Dashboard Definitions .....	325
Delete a Project Dashboard Report Definition .....	325
<b>Chapter 20 • Retraining Models .....</b>	<b>327</b>
Overview of Retraining Models .....	327
Prerequisites for Retraining a Model .....	328
Define a Model Retrain Task .....	328
Execute a Model Retrain Task .....	333
Viewing Retrained Models and Model Comparison Reports .....	334
<b>PART 6 Combining Multiple Reports 337</b>	
<b>Chapter 21 • Aggregated Reports .....</b>	<b>339</b>
About Aggregated Reports .....	339
Create an Aggregated Report .....	340
View an Aggregated Report .....	341
Edit an Aggregated Report Definition .....	341
Delete an Aggregated Report .....	342
<b>PART 7 SAS Model Manager Workflow Console 343</b>	
<b>Chapter 22 • Using Workflow Console .....</b>	<b>345</b>
Overview of Workflow Console .....	346
User Interface Layout .....	346
Customizing Category Views .....	349
Setting Preferences .....	355
Working with Objects .....	357
Viewing Workflow Activities .....	359
Working with Workflow Activities .....	360
Editing Activity Properties .....	361
Working with Comments .....	362
Viewing Workflow Milestones .....	365
<b>Chapter 23 • Managing Workflows .....</b>	<b>367</b>
Overview of Managing Workflows .....	367
Viewing Workflow Definitions .....	368
Creating a New Workflow .....	369

Viewing Workflows . . . . .	370
Editing a Workflow . . . . .	372
Editing Workflow Properties . . . . .	374
Working with Workflow Participants . . . . .	374
Terminating a Workflow . . . . .	378
<b>Chapter 24 • SAS Workflow Model Management Components . . . . .</b>	<b>379</b>
Overview of SAS Workflow Model Management Components . . . . .	379
Importing Models . . . . .	380
Viewing Models . . . . .	381
Setting Champion and Challenger Models . . . . .	383
Publishing Models . . . . .	384
Add, View, or Delete Attachments . . . . .	386
Creating and Viewing Reports . . . . .	386
Viewing Performance Results . . . . .	388
Viewing All Model Management Components . . . . .	389
<b>PART 8 Appendixes 391</b>	
<b>Appendix 1 • Query Utility . . . . .</b>	<b>393</b>
Overview of the Query Utility . . . . .	393
Search for Models . . . . .	394
Search By Using a UUID . . . . .	396
Search Life Cycles for Tasks Assigned to Users . . . . .	398
<b>Appendix 2 • SAS Model Manager Access Macros . . . . .</b>	<b>401</b>
Overview of Access Macros . . . . .	401
Using the SAS Model Manager Access Macros . . . . .	402
Dictionary . . . . .	407
<b>Appendix 3 • SAS Model Manager Macro Variables . . . . .</b>	<b>441</b>
<b>Appendix 4 • Macros for Registering Models to the SAS Metadata Repository . . . . .</b>	<b>449</b>
Using Macros to Register Models Not Created by SAS Enterprise Miner . . . . .	449
Dictionary . . . . .	453
<b>Appendix 5 • Macros for Adding Folders, Projects, Versions, and Setting Properties . . . . .</b>	<b>459</b>
Adding Folders, Projects, Versions, and Properties Using Macros . . . . .	459
Dictionary . . . . .	464
Example: Add a Folder, Project, and Version, Set Properties . . . . .	470
<b>Appendix 6 • Macros for Generating Score Code . . . . .</b>	<b>473</b>
Generating Score Code for COUNTREG Procedure Models . . . . .	473
Generating Score Code for PROC SEVERITY Models . . . . .	474
Dictionary . . . . .	474
<b>Appendix 7 • Properties . . . . .</b>	<b>503</b>
General Properties . . . . .	503
System Properties . . . . .	504
Specific Properties for a Project . . . . .	505
User-Defined Properties . . . . .	508
Specific Properties for a Version . . . . .	511
Specific Properties for Milestones and Tasks . . . . .	512
Specific Properties for a Model . . . . .	514

Scoring Task Properties .....	516
Result Set Properties .....	517
Schedule Properties .....	518
<b>Appendix 8 • SAS Model Manager R Model Support .....</b>	<b>519</b>
Overview of Using R Models with SAS Model Manager .....	519
Preparing R Model Files to Use with SAS/IML .....	520
<b>Appendix 9 • Statistical Measures Used in Basel II Reports .....</b>	<b>527</b>
<b>Appendix 10 • Report and Performance Monitoring Examples .....</b>	<b>535</b>
Dashboard Report Examples .....	535
Model Retrain Comparison Report Example .....	544
Monitoring Performance of a Model without Score Code .....	551
<b>Glossary .....</b>	<b>553</b>
<b>Index .....</b>	<b>561</b>



# 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 20](#).

---

## 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 12.3

---

## Overview

SAS Model Manager 12.3 has the following new features and enhancements:

- ability to create and manage multiple projects in a control group
- enhanced performance monitoring and reporting
- support for SAS Enterprise Miner Random Forest and SAS/ETS models
- ability to manage models published to a database
- support for multiple SAS application servers
- ability to add folders, projects, versions, and to set properties by using macros
- ability to create and view reports within a workflow activity
- ability to view the process flow diagram for a workflow

---

## Create and Manage Multiple Projects in a Control Group

SAS Model Manager enables you to create multiple projects in a control group. Additional versions can then be created for all projects within the control group. Champion models for all projects within the control group can be monitored for performance, and published to the SAS Metadata Repository. For more information, see [“Overview of Project Control Groups” on page 99](#).

---

## Enhanced Performance Monitoring and Reporting

The following enhancements have been made to performance monitoring and reporting:

- ability to schedule a performance task for a certain date and time on available servers, and to specify where to save the performance job
- support for multiple data sources and collection dates when defining a performance task at the project level

- ability to delete performance summary data sets

For more information, see [“Schedule Performance Monitoring Tasks”](#) on page 273, [“Run the Define Performance Task Wizard”](#) on page 268, and [“Delete Performance Summary Data Sets”](#) on page 276.

---

## Support for SAS Enterprise Miner Random Forest and SAS/ETS Models

SAS Model Manager supports importing SAS Enterprise Miner Random Forest (HPFOREST), as well as SAS/ETS COUNTREG and SEVERITY models that are contained in a SAS package (SPK) file. Macros have been added to generate the score code for SAS/ETS COUNTREG and SEVERITY models. The models can then be registered in the SAS Metadata Repository. For more information, see [Appendix 6, “Macros for Generating Score Code,”](#) on page 473 and [Chapter 8, “Importing Models,”](#) on page 125.

---

## Manage Models Published to a Database

You can view the publish history for champion and challenger models, as well as remove the published models from a database. For more information, see [“View Publish History”](#) on page 246 and [“Remove Models from a Database”](#) on page 244.

---

## Support for Multiple SAS Application Servers

When scoring a model, retraining a model, or monitoring performance of champion and challenger models, you can select the SAS Application Server. For more information, see [“Create a Scoring Task”](#) on page 164, [“Define a Model Retrain Task”](#) on page 328, and [“Run the Define Performance Task Wizard”](#) on page 268.

---

## Add Folders, Projects, Versions, and Set Properties By Using Macros

SAS Model Manager provides macros that you can use to programmatically add folders, projects, and versions to the Project Tree, and to set project and version properties. For more information, see [Appendix 5, “Macros for Adding Folders, Projects, Versions, and Setting Properties,”](#) on page 459.

---

## **Create and View Reports within a Workflow Activity**

When the Create and View Reports model management component is associated with a workflow activity, you can create model comparison reports, performance monitoring reports, and a workflow milestones report using the SAS Model Manager Workflow Console. For more information, see [Chapter 22, “Using Workflow Console,”](#) on page 345.

---

## **View the Process Flow Diagram for a Workflow**

You define a process flow diagram when you create a workflow definition using SAS Workflow Studio. The process flow diagram is displayed when you select a workflow in the SAS Model Manager Workflow Console. For more information, see [Chapter 22, “Using Workflow Console,”](#) on page 345.



# Accessibility Features of SAS Model Manager

---

## Overview

SAS Model Manager 12.3 has been tested with assistive technology tools. It includes accessibility and compatibility features that improve the usability of the product for users with disabilities. (Some accessibility issues remain and are 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](mailto:accessibility@sas.com) or call SAS Technical Support.

---

## Documentation Format

Please contact [accessibility@sas.com](mailto:accessibility@sas.com) if you need this document in an alternative digital format.

---

## Landmarks

Landmarks are references to the primary areas of an application's user interface. They provide a quick and easy way for keyboard users to navigate to these areas of the application.

To access the list of landmarks that are available for a specific context, press Ctrl+F6 to open the Landmarks window. Use the arrow keys to select a landmark, and then press Enter to navigate to that area of the application.

---

## 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 category view button bar, and category 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 category view and the component that is selected. Inactive options are dimmed. The category views provide information and metadata about projects, life cycles, and data sources.

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 top of the window contains the application name and an application bar that includes a menu bar and a **Log Off** button.
- The left side of the window contains a collapsible navigation pane. This pane can contain one or more views (depending on your assigned capabilities and roles) that you select using buttons at the bottom of the pane. The views contain either trees or lists of objects. You can open these objects (one at a time) into tabs in the work area next to the navigation pane.
- The center of the window (the work area) contains tabs: personal tabs, open objects, system reports, and administrative tools. The work area can be split (either vertically or horizontally) into two groups of tabs.
- The right side of the window can contain a pane for docking the Object Preview window, the Permissions Inspector window, and the How To window. This pane is not displayed until a window is docked.
- The bottom of the window contains a status bar that displays information about your connection to the metadata server.

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 22, “Using Workflow Console,” on page 345](#).

---

## **Themes**

An application’s theme is the collection of colors, graphics, and fonts that appear in the application. The following themes are provided with this application: SAS Corporate, SAS Blue Steel, 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 the keyboard shortcuts for the application. In the user interface, the shortcuts are displayed within parentheses in tooltips and menu labels.

*Note:* When you use a keyboard shortcut to activate a button, move the focus to the field or section that the button is associated with before you use the keyboard shortcut. For example, if a table has an associated  button, you must first move the focus to the table before you press Ctrl+?.

#### Keyboard Shortcuts

Task	Keyboard Shortcut
Open a Help pop-up window from the  button.	Ctrl+? <i>Note:</i> This shortcut does not work on some keyboards (for example, the Italian keyboard).
Zoom in.	Ctrl+plus sign
Zoom out.	Ctrl+minus sign

Task	Keyboard Shortcut
Reset the zoom state.	Ctrl+0
<p>Maximize view (collapses the category pane and the tile pane, and hides the status bar and the application bar, which includes the menu bar and the workspace bar).</p> <p>or</p> <p>Exit maximized view (expands the category pane and the tile pane, and shows the status bar and the application bar).</p>	Ctrl+Alt+Shift+M
Open a pop-up menu.	<p>Shift+F9 (if a menu is available in that context)</p> <p><i>Note:</i> If you use Shift+F9 to display the pop-up menu, then it is always displayed in the top left corner of the user interface control that you are using.</p>
Open the Landmarks window.	Ctrl+F6
<p>Temporarily invert or revert application colors (for the current session only).</p> <p><i>Note:</i> You can set the <b>Invert application colors</b> preference in the Preferences window if you want the color change to persist across sessions.</p>	Ctrl+~
Rename the selected tab.	<p>Make sure that the focus is on the tab. Press F2, and specify the new name. To commit your changes, press Enter. To cancel your changes, press Esc.</p>
Close the selected tab.	<p>Make sure that the focus is on the tab, and then press Delete.</p> <p><i>Note:</i> Some tabs cannot be closed.</p>
Switch in and out of Edit mode for a table cell.	<p>To enter Edit mode, select a cell, and press F2.</p> <p>To exit Edit mode, press Esc.</p>
Navigate between table headings and table content.	<p>For a two-dimensional table, make sure that the focus is on the table and that you are not in Edit mode. Press Ctrl+F8 to switch the focus between column headings and table cells. Use the arrow keys to navigate from heading to heading.</p> <p>For a multidimensional table, make sure that the focus is on a table cell and that you are not in Edit mode. Press Ctrl+F8 to switch the focus between column headings, row headings, and table cells. Use the arrow keys to navigate from heading to heading.</p>

Task	Keyboard Shortcut
Navigate the content rows of a table.	<p>When table cells are in Edit mode:</p> <ul style="list-style-type: none"> <li>• Press Tab and Shift+Tab to move from cell to cell horizontally across columns.</li> <li>• Press Enter and Shift+Enter to move from cell to cell vertically across rows.</li> </ul> <p>When table cells are not in Edit mode, use the arrow keys to move from cell to cell.</p>
Sort columns in a table.	<p>To sort a single column, navigate to its column heading (press Ctrl+F8). Press the spacebar to sort the column.</p> <p>To sort additional columns, navigate to the column heading of each additional column that you want to sort. Press Ctrl+spacebar.</p>
Change the width of the current column.	<p>Navigate to the column heading (press Ctrl+F8). Then press Ctrl+left arrow or Ctrl+right arrow to change the width of the column.</p>
Move the current column.	<p>Navigate to the column heading (press Ctrl+F8). Then press Shift+left arrow to move one column to the left, and press Shift+right arrow to move one column to the right.</p>
Automatically re-size the current column to fit its contents.	<p>Navigate to the column heading (press Ctrl+F8). Then press Enter.</p>

---

## Exceptions to Accessibility Standards

### **SAS Model Manager Client**

Exceptions to accessibility standards are documented in the following table.

Accessibility Issue	Workaround
<p>You cannot use the keyboard to open context menus in the object details section. SHIFT + F10 does not open the context menus when focus is on the relevant field in the details section, such a property field.</p>	<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 <b>Accessibility Options</b> ⇒ <b>Mouse</b> tab and then select <b>Use MouseKeys</b>.</p> <p>For Windows 7, from the Control Panel select <b>Ease of Access Center</b> ⇒ <b>Make the mouse easier to use</b>, and then select <b>Turn on Mouse Keys</b>.</p>
<p>You cannot use the keyboard to expand or collapse the objects in the Project Tree.</p>	<p>No workaround is available.</p>
<p>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.</p>	<p>No workaround is available.</p>
<p>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.</p>	<p>No workaround is available.</p>
<p>SAS Model Manager does not properly inherit the Windows high contrast and large text settings.</p>	<p>No workaround is available.</p>
<p>The ENTER key does not activate a default button, such as <b>OK</b>, <b>Next</b>, or <b>Finish</b> in a window after the required fields have been completed.</p>	<p>No workaround is available.</p>
<p>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.</p>	<p>No workaround is available.</p>

Accessibility Issue	Workaround
When JAWS is in use, the Library drop-down does not respond correctly to keyboard input. The Alt+Down Arrow key combination places focus on the Input Variable table when using the Create Output Table feature.	A user can use the arrow keys to change the selection, and then navigate away and back immediately to enable JAWS to read the current selection.

## SAS Model Manager Workflow Console

Exceptions to accessibility standards are documented in the following table.

*Note:* The JAWS issues occur when JAWS is used with Internet Explorer. Other browsers were not tested with JAWS, unless noted.

### *Exceptions to Accessibility Standards*

Accessibility Issue	Workaround
Sometimes, you cannot use the keyboard to sequentially navigate through the interface and move the focus in a meaningful order.	No workaround is available.
The SAS High Contrast theme has a few unresolved focus and contrast issues.	For contrast issues, select a different theme, and then press Ctrl+~ to invert the colors.
The SAS Light theme and SAS Dark theme might not provide sufficient color contrast for some users.	Use the SAS Corporate theme or the SAS High Contrast theme.
JAWS cannot read some of the controls in the application, such as images, icons, and buttons.	No workaround is available.
JAWS cannot read the tooltips of items in trees, lists, and menus.	No workaround is available.
JAWS refers to table controls as list boxes.	When JAWS reports that a control is a list box, keep in mind that it might actually be a table.
JAWS can sometimes read controls that have been disabled.	No workaround is available.
Sometimes, JAWS does not correctly work with the controls in the Preferences window.	When you are in Virtual PC cursor mode in JAWS, traverse the entire window to familiarize yourself with its contents before you change any of the settings. You might need to switch between Forms mode and Virtual PC cursor mode to access all of the controls.

Accessibility Issue	Workaround
JAWS does not correctly read the states in a tri-state check box tree if JAWS is not in Forms mode.	Disable the JAWS Virtual PC cursor when you work with the check box tree. Tab to the tree, and press Insert+Z to disable the Virtual PC cursor. When you finish interacting with the tree, press Insert+Z to re-enable the Virtual PC cursor.
The keyboard shortcuts that are used to interact with editable tables can conflict with keyboard shortcuts for the Forms mode in JAWS.	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 Virtual PC cursor. When you finish interacting with the table, press Insert+Z to re-enable the Virtual PC cursor.
JAWS cannot read two-column property tables.	No workaround is available.
<p>JAWS does not correctly read the information in a table:</p> <ul style="list-style-type: none"> <li>• JAWS cannot read the column headings of a table.</li> <li>• When table cells are not editable and the focus is on the body of the table, JAWS reads an entire row at a time instead of cell by cell.</li> <li>• When table cells are editable and the focus is on the body of the table, JAWS reads only the first row of the table. If you use the arrow keys to select a cell or row, then JAWS does not read anything. If you press Enter to edit a cell, then JAWS reads the row that contains the edited cell.</li> </ul>	No workaround is available.
When a table cell is selected and you press Home, End, Page Up, or Page Down, the focus moves to the first displayed column, regardless of which column you were in.	Use the arrow keys to navigate through the cells of the table.
You cannot use the keyboard to scroll to the left and the right in some tables.	No workaround is available.
You cannot use the keyboard to activate the links within how-to topics and Help pop-up windows.	Use the <b>Help</b> menu to access the linked documents.

Accessibility Issue	Workaround
You cannot use Shift+F10 to open a pop-up menu.	<p>Use Shift+F9 to open pop-up menus that are created for the SAS application. The generic menu that is provided by the Flash player cannot be opened by Shift+F9.</p> <p><i>Note:</i> If you press Shift+F10 in Internet Explorer and no context menu is available, the browser moves the focus to the <b>File</b> menu for the browser tab. To return focus to the application area of the browser window, press Esc.</p>
You cannot use the keyboard to access the close (x) button that is in the top right corner of a tab.	Make sure that the focus is on the tab, and then press Delete to close the tab.
You cannot use the keyboard to access the close (x) button that is in the top right corner of a tile in the tile pane.	Make sure that the focus is on the tile, and then press Delete to close the tile. (The object that is displayed in the tile is not deleted.)
Visual focus for the menu bar is indicated with an outline around the entire menu bar instead of around individual menus.	To select individual menus, use the left or right arrow key.
Sometimes, you cannot use the Tab key to move the focus to the application area of a web browser (that is, the part of the browser window that is controlled by the Flash player).	<p>The following workaround is applicable to Internet Explorer only.</p> <p>Press Ctrl+<i>number</i>, where <i>number</i> is the ordinal position of the application's tab in the set of tabs that are open in your browser window. Then press Tab to move the focus to the application area.</p>
You cannot use the Tab key to move the focus outside of a code or expression editor. Pressing Tab within the editor only inserts tabs.	<p>For Internet Explorer, press Shift+F10, and then press Esc to move the focus outside of the editor.</p> <p>For Firefox, press Alt+Tab to switch to another application. When you switch back, the focus will be outside of the editor.</p>
You cannot use Ctrl+Alt+Shift+M to minimize or maximize the view if the focus is on the workspace bar.	No workaround is available.
If you tab to an item that is partially or entirely off-screen, the item is not automatically scrolled back into view.	Sometimes, you can use the arrow keys or the Tab key to scroll the item back into view.
When you use the Ctrl+plus sign keyboard shortcut to zoom in, some portions of the interface can become hidden from view.	Use the keyboard to access the hidden parts of the interface.

Accessibility Issue	Workaround
The Ctrl+plus sign and Ctrl+minus sign keyboard shortcuts for zooming in and out do not work on some menus unless the menus are first opened.	Open the menu before you use the keyboard shortcut.
The Ctrl+plus sign and Ctrl+minus sign keyboard shortcuts for zooming in and out do not work on all elements in the application window (for example, tooltips and button labels).	No workaround is available.
If you maximize a tile in the <b>Home</b> workspace and then use the Tab key to navigate, the focus appears to be lost after you tab away from the <b>Log Off</b> button.	After you tab away from the <b>Log Off</b> button, press the Tab key 5 more times to return the focus to the maximized tile.
You cannot use the keyboard to navigate in the <b>Layout</b> section because it is a Read-Only interface that is used for the visual verification of the elements that have been created.	Use the test button that is in the <b>Layout</b> section to preview your elements in a secondary window. The items that are displayed in the secondary window are identical to the items that are displayed in the <b>Layout</b> section, but unlike the items in the Read-Only <b>Layout</b> section, you can interact with the items in the secondary window.  <i>Note:</i> After the application opens the secondary window, press Tab to move the focus to the window.
JAWS cannot read the labels for the <b>Red</b> , <b>Green</b> , and <b>Blue</b> fields in the Custom Colors window.	No workaround is available.
You cannot use the keyboard to access the color blocks in the <b>Recently used</b> section of the color selection control.	No workaround is available.
JAWS does not explain how to open a drop-down menu or drop-down list.	Press Ctrl+down arrow to open the control.
When JAWS reads the control names in a breadcrumb, it does not distinguish between the breadcrumb buttons that contain drop-down menus and those that do not.	Check for a drop-down menu by pressing Ctrl+down arrow on a breadcrumb button. A drop-down menu will open if one exists for that button.
When you use the down arrow to scroll through the items in a "combo box," any item that opens a secondary window will do so when you scroll down to it. This will prevent you from navigating to items that are farther down in the drop-down list.	Press Ctrl+down arrow to scroll through the items in the drop-down list, and then press Enter or Tab to make a selection.



Accessibility Issue	Workaround
When you add a date value to the predefined list for a date element, you cannot use the keyboard to access the date-selection button in the table cells in the Customize Data window for the predefined list.	Enter the date value in the field that is next to the date-selection button.
JAWS cannot read the contents of a tree table (that is, a table that contains a tree) unless the table is in Edit mode.	Make sure that the focus is in the tree table, and press F2 to enter Edit mode.
JAWS cannot read the <name-of-UI-control>.	No workaround is available.
JAWS cannot read the content selection tree.	No workaround is available.
Sometimes, after you close a tab to hide it from view, you can still use the keyboard to access the contents of the tab.	No workaround is available.
After you edit or delete a comment, the focus does not return to the comment.	Use the Tab key to return the focus to the comment.
If the list of additional search options contains a secondary level of options, you cannot use the keyboard to select the check boxes that are associated with that secondary level of options.	No workaround is available.



# 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 books, go to [support.sas.com/bookstore](http://support.sas.com/bookstore). If you have questions about which titles you need, please contact a SAS Book Sales Representative:

SAS Books  
SAS Campus Drive  
Cary, NC 27513-2414  
Phone: 1-800-727-3228  
Fax: 1-919-677-8166  
E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web address: [support.sas.com/bookstore](http://support.sas.com/bookstore)



## **Part 1**

---

# What Is SAS Model Manager?

<i>Chapter 1</i>	
<b>Overview of SAS Model Manager</b> .....	<b>3</b>
<i>Chapter 2</i>	
<b>Introduction to SAS Model Manager</b> .....	<b>9</b>



## Chapter 1

# Overview of SAS Model Manager

---

<b>Managing Models Using SAS Model Manager</b> .....	<b>3</b>
<b>The SAS Model Manager Operational Environment</b> .....	<b>5</b>
<b>Model Management Process</b> .....	<b>6</b>

---

## Managing 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 can track the progress of your project's version either by creating processes and activities using the SAS Model Manager Workflow Console or by creating milestones and tasks using a life cycle. You create custom processes and activities or milestones and tasks to meet your business requirements and to match your business processes.
- Data tables that are registered in SAS Management Console or in an accessible SAS library can be used in SAS Model Manager.
- The models that you import into SAS Model Manager can be SAS Enterprise Miner models, SAS/STAT linear models, SAS/ETS COUNTEG and SEVERITY models, models that you develop using SAS code, PMML models, or R models. 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 schedule and 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. The aggregated reporting facility enables you to combine multiple reports into a single report.  
SAS Model Manager can also create Basel II model validation reports.
- 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 flag challenger models and publish them to a production environment.
- You can publish models to the SAS Metadata Repository, or you can publish the champion model and challenger models to a database for scoring using the SAS Scoring Accelerator.
- You can monitor the performance of a champion model in a production environment by scheduling a performance monitoring task to run on a specific day and time, by executing the performance monitoring task using the SAS Model Manager window, or by using SAS Model Manager macros in a batch environment. The performance of challenger models can be monitored in a production environment using the SAS Model Manager window. After the data is collected, you can create a report that compares the performance of the champion and challenger models.
- SAS Model Manager provides macro programs for you to run model registration to SAS Model Manager and scoring in a batch environment. Another macro registers SAS/STAT item store models and High-Performance Analytic models that were not created in SAS Enterprise Miner to the SAS Metadata Repository. If you create models using the COUNTREG or SEVERITY procedures, SAS Model Manager provides macros for you to generate score code for the models.
- You can retrain models to respond to data or market changes.
- SAS Model Manager provides dashboard reports for you to monitor the state of projects using performance monitoring reports, and enables you to view the reports in a web browser.
- If your environment supports multiple SAS Application Servers, you can select the application server to execute scoring tasks, performance monitoring tasks, and model retrain tasks.
- Using a query utility, you can look for models by name or identifier, or you can look for tasks.

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

- Users in the Model Manager Administrator 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

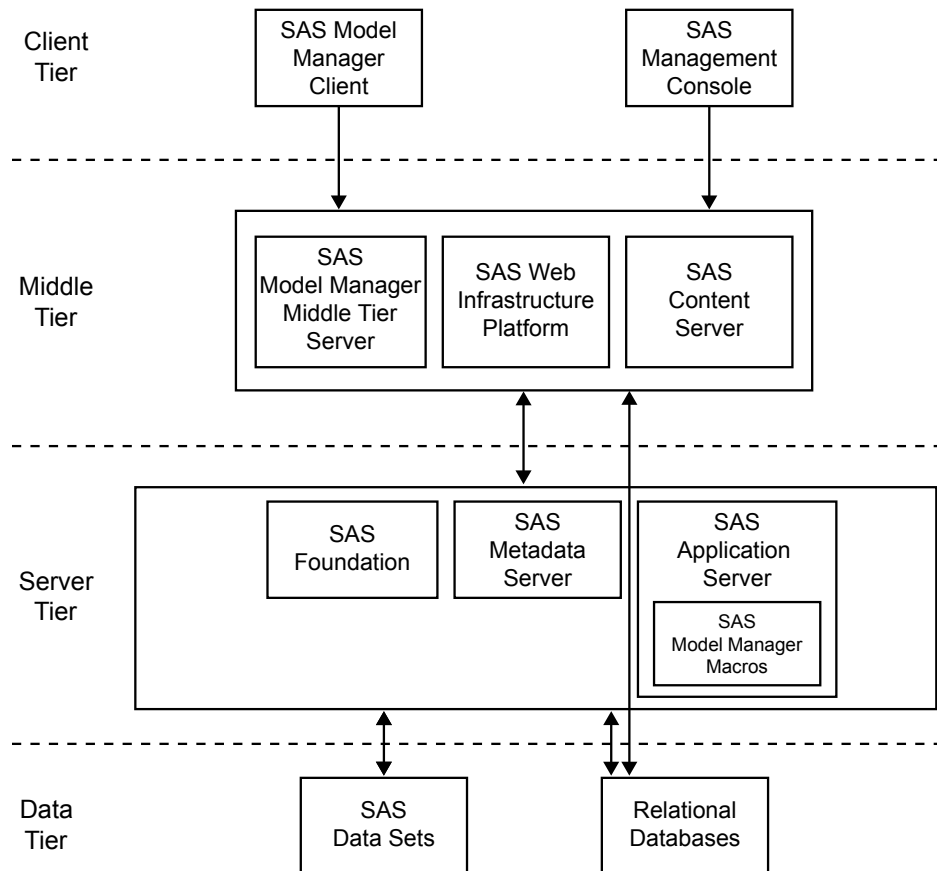


to SAS Model Manager. Data tables are used for scoring, testing, and performance monitoring. Performance data can be created from your operational data.

You can also create multiple projects in a control group. Additional versions can then be created for all projects within the control group. Champion models for all projects within the control group can be monitored for performance, and published to the SAS Metadata Repository. For more information, see [“Overview of Project Control Groups” on page 99](#).

## 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 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 Application Server

There are multiple servers that are used for different processes on the SAS Application Server. SAS Model Manager uses the JES Java Batch Server to schedule jobs for both performance tasks and scoring tasks. The execution of all other SAS Model Manager tasks such as SAS score code, scoring tasks, performance tasks, and mode retraining tasks is performed on the SAS Workspace Server.

#### SAS Model Manager Macros

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

#### 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

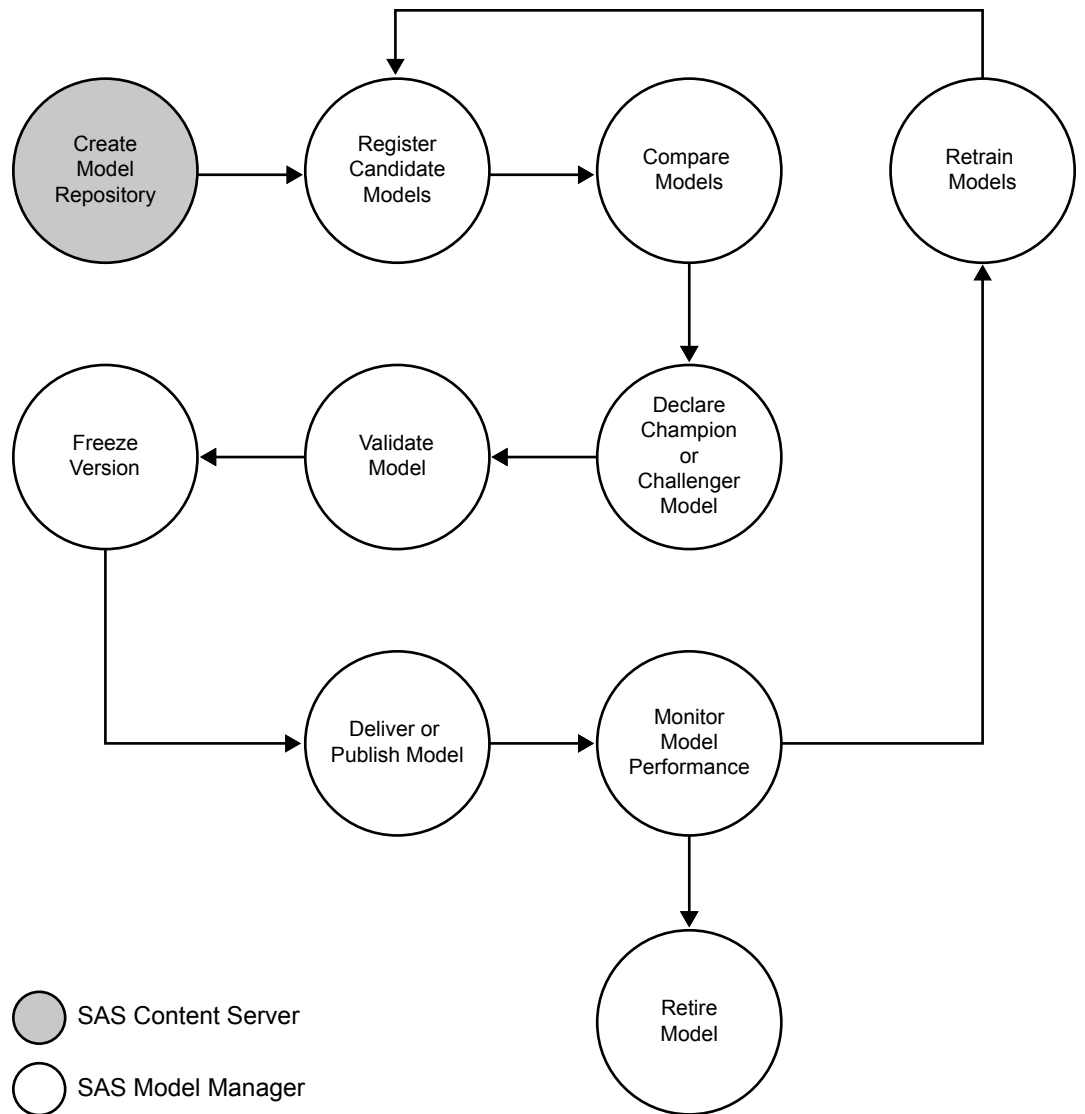
[“Publishing Models to a Database” on page 231](#)

---

## 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 the models by using test data sources.
- **Declare Champion or Challenger Model:** declare the model as champion or challenger to use for testing and production phases of the life cycle or workflow.
- **Validate Model:** perform scoring tests and create validation reports for the champion model and challenger models by using test data sources.
- **Freeze Version:** lock a version when the champion model in a version folder is approved for production.

- **Deliver or Publish Model:** publish a champion or challenger models to a SAS publish channel, to a database, or to the SAS Metadata Repository.
- **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.

Here is an example of the model management process for comparing a challenger model to the champion model to determine the best champion model:

1. Register candidate models in the version that is under development.
2. Create a Dynamic Lift report and compare the model to the champion model. Flag the model as a challenger based on the results of the Dynamic Lift report.
3. Perform scoring tasks with the champion and challenger models in real time or in batch. This step can be performed outside SAS Model Manager.
4. Publish the challenger model to a database or to the SAS Metadata Repository.
5. Prepare performance data sources, which include both the actual outcome variable and predicted variable.
6. Create and execute the performance tasks for the champion and challenger models to create reports to compare and validate the champion model and challenger models. One of the reports that are available for this comparison is the Champion and Challenger Performance report.
7. Set the challenger model as the project champion if the challenger is good enough to be promoted. Go to step 3, or consider building another model as a challenger with existing or a new input training data source.
8. Publish the new project champion model with or without a new challenger model.

## Chapter 2

# Introduction to SAS Model Manager

---

<b>Layout of the SAS Model Manager Window</b> . . . . .	<b>9</b>
Overview of the SAS Model Manager Window . . . . .	9
SAS Model Manager Category Views . . . . .	10
SAS Model Manager Toolbar and Menus . . . . .	15
<b>SAS Model Manager User Groups, Roles, and Tasks</b> . . . . .	<b>20</b>
SAS Model Manager Groups . . . . .	20
SAS Model Manager Roles . . . . .	20
Setting Up SAS Model Manager . . . . .	21
Setting Up Projects and Versions . . . . .	22
Importing and Assessing Models . . . . .	23
Deploying and Delivering Models . . . . .	24
Monitor Champion Model Performance and Retrain Models . . . . .	25
General Tasks . . . . .	26

---

## 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 category view button bar, and category views that enable you to work with modeling projects and to view life cycle templates and project tables. 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 category view and the component that is selected. Inactive options are dimmed. For more information about the SAS Model Manager toolbar and menus, see [“SAS Model Manager Toolbar and Menus” on page 15](#).

The category view button bar enables you to select a category view to display. Each category view 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 Category Views




The SAS Model Manager interface is divided into three category views.

- [Projects category view](#)
- [Life Cycles category view](#)
- [Data Sources category view](#)

Each category view is a work area for an aspect of model management. The category views enable you to access specific information and functionality to manage your projects and models. For example, the Projects category view has three major sections: the Repository section, the Details section, and the Annotations section.

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

**Table 2.1** The Category View Buttons

	Projects category view button
	Life Cycles category view button
	Data Sources category view 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 champion model and challenger models. For more information about the SAS Model Manager toolbar, see [“SAS Model Manager Toolbar” on page 15](#).

The SAS Model Manager menus enable you to perform general tasks such as renaming an object or accessing Help. The menus 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 18](#).

## SAS Model Manager Category Views

### Projects Category View

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

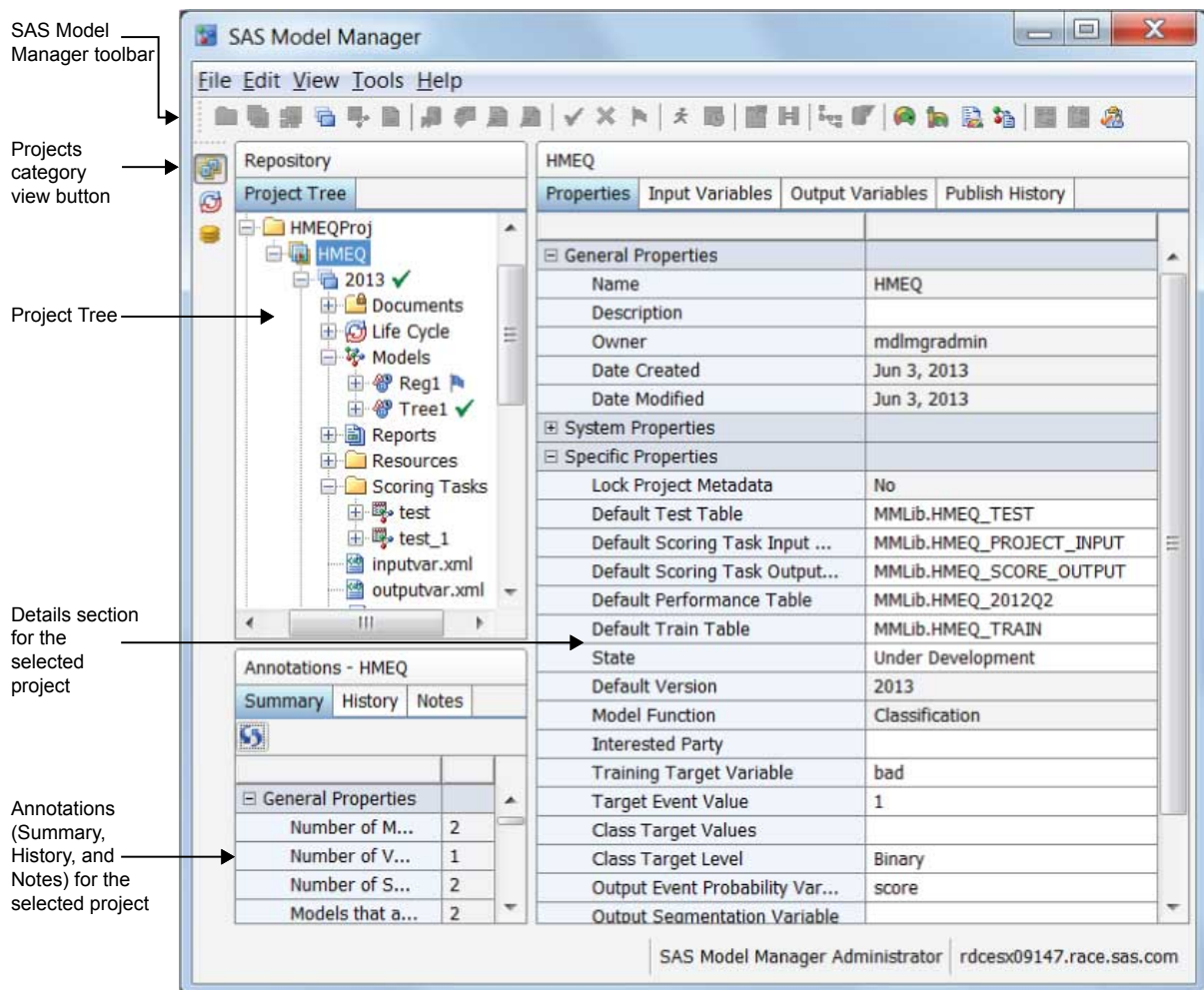
Most of your work in SAS Model Manager is performed in the Projects category view, where you manage model projects and their components. The Projects category view contains three major sections: the Repository section, a Details section, and the Annotations section.

The Repository section 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

single projects or project control groups. Inside a project or project control group, 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 Details section and in the Annotations section. In the following example, the Project Tree displays an open organizational folder that is named **HMEQProj**. The **HMEQProj** folder contains a project folder that is named **HMEQ**. The **HMEQ** Project folder contains two version folders, **2012** and **2013**. The **2013** version folder contains a **Models** folder that contains three models.

**Figure 2.1** The Projects Category View



The Details section 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 Details section displays model-level metadata. When you select a version folder in the Project Tree, the Details section 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 Results](#)” on page 253. 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
- 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 55.

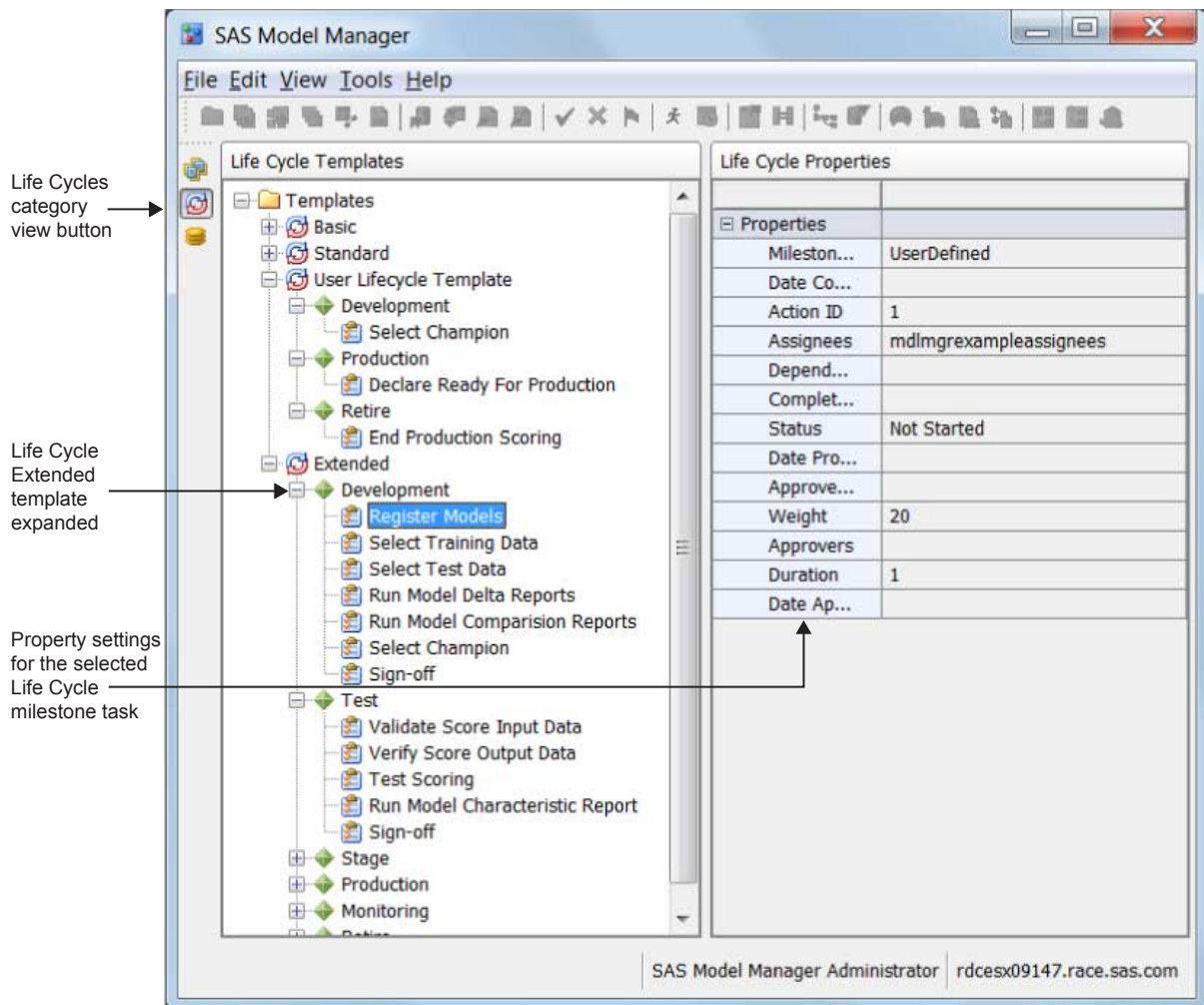
### ***Life Cycles Category View***

Click the Life Cycles category view  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 category view 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 Category View



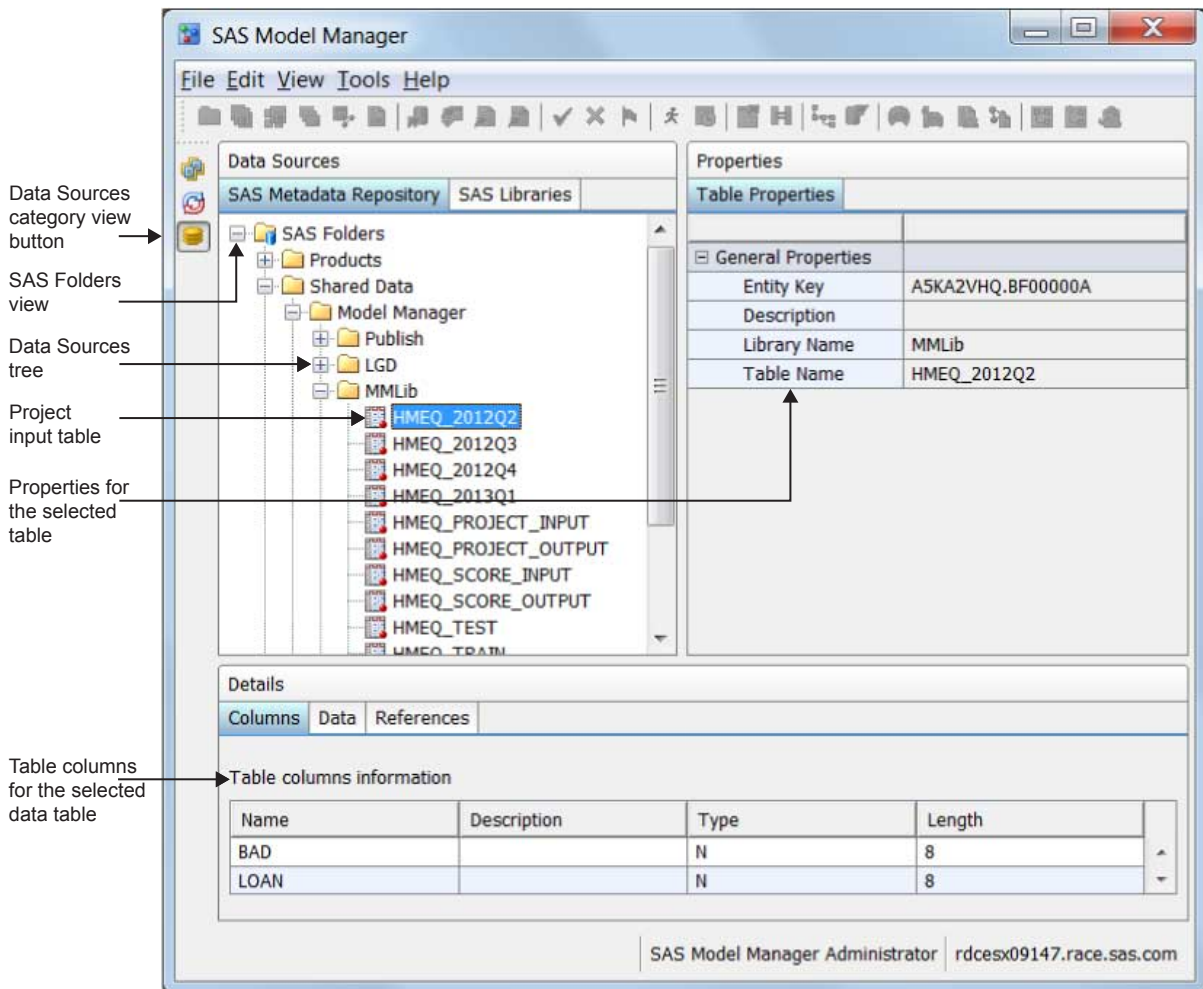
For more information about life cycles, see “Working with Life Cycles” on page 92.

### Data Sources Category View

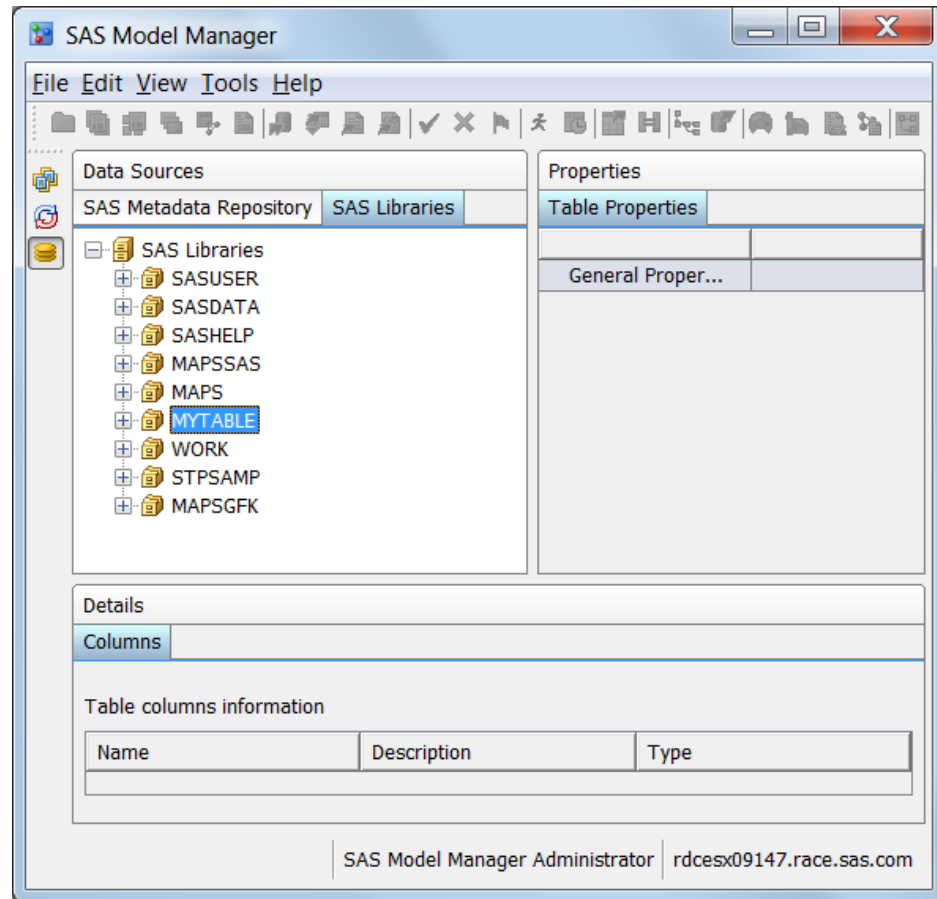
Click the Data Sources category view  button to view data sources to SAS Model Manager. SAS Model Manager data sources are populated from libraries that are defined through SAS Management Console.

In the following examples, the **SAS Metadata Repository** tab of the **Data Sources** section displays the SAS Folders. This component lists the folders that are available in the SAS Metadata Repository. The folders contain data tables that are available to SAS Model Manager projects. The **Properties** section displays a list of the metadata for the selected data table. The Details section displays information about the contents of the selected prototype or table and what objects in the Project Tree are associated with the prototype or table.

Figure 2.3 The Data Sources Category View



If you have data source tables on a local SAS Workspace Server or network drive, you can define a libref to point to the tables. In the Data Sources category view, you can view the tables using the **SAS Libraries** tab. The **SAS Libraries** tab shows the **MYTABLES** libref that was created by using the Edit Start-up Code window:



For more information about data sources, see [Chapter 3, “Working with Data Sources,”](#) on page 31.

## 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 category view 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. To enable the New Folder button and menu, select an organizational folder. For more information, see [“Create an Organizational Folder”](#) on page 48.

- 2 **New Project** creates a mining project folder under an organizational folder. Project folders normally contain one or more version folders. To enable the New Project button and menu, select an organizational folder. For more information, see [“Create a Project” on page 61](#).
- 3 **New Project Control Group** creates a project control group under an organizational folder. The actions that are taken on a project control group apply to all of the projects that are defined in the control group. For more information, see [“Create a Project Control Group”](#).
- 4 **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. To enable the New Version button and menu, select a project folder. For more information, see [“Create a Version” on page 89](#).
- 5 **New Scoring Task** creates a scoring task in the **Scoring Task** folder that you have selected in the Project Tree. To enable the **New Scoring Task** button and menu, select the **Scoring Tasks** folder. For more information, see [“Create a Scoring Task” on page 164](#).
- 6 **New Report** creates model comparison, validation, and summary reports. To enable the New Report button and menu, select the **Reports** folder. For more information, see [“Overview of Model Comparison, Validation, and Summary Reports” on page 180](#).
- 7 **Import model from the SAS Metadata Repository** imports a model from the SAS Metadata Repository into the **Models** folder that is selected in the Project Tree. To enable the import models button and menu, select the **Models** folder. For more information, see [“Import Models from the SAS Metadata Repository” on page 127](#).
- 8 **Import model from a SAS Package File** imports a SAS Enterprise Miner package file (SPK) or a SAS/STAT package file from the user's client machine to the **Models** folder that is selected in the Project Tree. To enable the import model buttons and menu, select the **Models** folder. For more information, see [“Import SAS Model Package Files” on page 128](#).
- 9 **Import model from Local Files** imports SAS code models that were not developed in SAS Enterprise Miner (such as PROC LOGISTIC models) into SAS Model Manager. To enable the import models buttons and menu, select the **Models** folder. For more information, see [“Import SAS Code Models and R Models Using Local Files” on page 130](#).
- 10 **Import model from a PMML Model File** imports a PMML model. To enable the import models buttons and menu, select the **Models** folder. For more information, see [“Import PMML Models” on page 143](#).
- 11 **Set as Champion** sets a model as the champion model. Setting the champion model sets the model's version as the default version. To enable the Set as Champion button and menu, select a model. For more information, see [Chapter 12, “Deploying Models,” on page 215](#).
- 12 **Clear Champion Model** or **Clear Challenger Model** deselects the champion model or the challenger model. To enable this button, select the champion or challenger model. For more information, see [Chapter 12, “Deploying Models,” on page 215](#).
- 13 **Flag as Challenger** identifies a model as a challenger model. To enable the button and menu, select a model. For more information, see [“Challenger Models” on page 219](#).
- 14 **Execute** performs the scoring, performance, or model retrain task that is selected in the Project Tree. To enable the button and menu, select a scoring task, the

**Performance Monitor** node, or the **Model Retrain** node. For more information, see [“Execute a Scoring Task” on page 168](#), [“Run the Define Performance Task Wizard” on page 268](#), or [Chapter 20, “Retraining Models,” on page 327](#).

- 15 **New Schedule** schedules the execution of a scoring or performance monitoring task by specifying the date and time intervals. To enable the button and menu, select a scoring task or the **Performance Monitor** node. For more information, see [“Schedule Scoring Tasks” on page 170](#) and [“Schedule Performance Monitoring Tasks” on page 273](#).
- 16 **Create Output Table** creates a new output table structure for one or more SAS Model Manager scoring tasks. To enable the button and menu, select a model. For more information, see [“Create Scoring Output Tables” on page 162](#).
- 17 **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 score results might be statistically invalid. The variable data type is not validated. To enable the button and menu, select a scoring task. For more information, see [“Overview of Scoring Tasks” on page 157](#).
- 18 **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 12.1.
- 19 **Publish Model** publishes the model that is selected in the Project Tree to the SAS Metadata Repository. For more information, see [“Publish Models to the SAS Metadata Repository” on page 228](#).
- 20 **Dashboard Report Definition** defines the indicators for the performance monitoring data that you want to see in a dashboard reports. To enable the button and menu, select a project. For more information, see [“Create a Dashboard Report Definition” on page 314](#).
- 21 **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 319](#).
- 22 **Define Performance Task** starts a wizard that creates a performance task. A performance task creates performance monitoring reports. To enable the button and menu, select a project. For more information, see [Chapter 16, “Create Reports by Defining a Performance Task,” on page 263](#).
- 23 **Define Model Retrain Task** starts a wizard that retrains one or more models. To enable the button and menu, select a project. For more information, see [Chapter 20, “Retraining Models,” on page 327](#).
- 24 **View Workflow** enables the user to view the workflows that are associated with the selected version. To enable the button and menu, select a version. For more information, see [“Viewing Workflows” on page 370](#).
- 25 **New Workflow** creates a new workflow from a process definition and associates it with the selected project or version. To enable the button and menu, select a version. For more information, see [“Creating a New Workflow” on page 369](#).

- 26 **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 359](#).

### **SAS Model Manager Menus**

The SAS Model Manager main menu varies in content based on whether you select the Projects category view, the Life Cycles category view, or the Data Sources category view. If a menu item is not applicable in the selected category view, 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 48](#).
- **New Project** creates a new modeling project folder. For more information, see [“Create a Project” on page 61](#).
- **New Project Control Group** creates a new project control group folder. For more information, see [“Create a Project Control Group” on page 103](#).
- **New Document Folder** creates a **Documents** folder. For more information, see [“Create a Documents Folder” on page 50](#).
- **New Version** creates a new version folder within a project folder. For more information, see [“Create a Version” on page 89](#).
- **New Scoring Task** creates a new scoring task within a **Scoring Tasks** folder. For more information, see [“Create a Scoring Task” on page 164](#).
- **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 125](#).
  - **SAS Metadata Repository** displays the **SAS Metadata Repository** window that you can use to select a mining results file and then import it into a **Models** folder. For more information, see [“Import Models from the SAS Metadata Repository” on page 127](#).
  - **SAS Model Package File** displays a local file browser window that you can use to import a model from a SAS Enterprise Miner or SAS/STAT package (SPK) file. For more information, see [“Import SAS Model Package Files” on page 128](#).
  - **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 130](#).
  - **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 143](#).
- **Exit** ends the SAS Model Manager session, and then closes the window.

#### **Edit**

- **Copy** creates a copy of the selected object.
- **Paste** places the 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 folder or file.

## View

- **Projects** displays the Projects category view. For more information, see [“Projects Category View” on page 10](#).
- **Life Cycles** displays the Life Cycles category view. For more information, see [“Life Cycles Category View” on page 12](#).
- **Data Sources** displays the Data Sources category view. For more information, see [“Data Sources Category View” on page 13](#).
- **Toolbar** toggles the SAS Model Manager toolbar on and off. For more information, see [“SAS Model Manager Toolbar” on page 15](#).

## Tools

- **Manage Templates** displays the SAS Model Manager Template Editor, which enables you to create, edit, update, or delete life cycle templates, model templates, report templates, and SAS files. For more information, see [“Creating Life Cycle Templates” on page 75](#) and [“User-Defined Model Templates” on page 148](#).
- **Edit Start-up Code** displays the Edit Start-up Code window. Use this window to create a libref for libraries that are not defined in the SAS Metadata Repository or in SAS code. For more information, see [“Using Tables from a Local or Network Drive” on page 42](#).
- **Remove Models from Database** removes published models from DB2, Greenplum, Netezza, Oracle, or Teradata databases. For more information, see [“Remove Models from a Database” on page 244](#).
- **Manage Project Dashboard Indicators** displays dashboard indicators for projects that are used to generate dashboard reports. Use this window to view or delete dashboard indicators from projects. For more information, see [“Manage All Project Dashboard Definitions” on page 325](#).
- **Generate Dashboard Reports** 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 319](#).
- **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 [“Overview of Managing Workflows” on page 367](#).
- **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 359](#).

## 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 that are supplied by SAS should not be modified.

### SAS Model Manager Roles

The SAS Model Manager roles enable specific users or groups to be assigned in order 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 to 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
Comments 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
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	<a href="#">Chapter 3, "Working with Data Sources," on page 31</a>
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	<a href="#">"Create an Organizational Folder" on page 48</a>
Create project control groups	MM Adv User, MM Admin	<a href="#">"Create a Project Control Group" (p. 103)</a>
Create projects from a control table	MM Adv User, MM Admin	<a href="#">"Create Projects from a Control Table" (p. 104)</a>
Create projects	MM Adv User, MM Admin	<a href="#">"Create a Project" on page 61</a>

Task	Group	Topic
Create versions	MM Adv User, MM Admin	“Create a Version” on page 89
Delete a node in the Project Tree	MM Adv User, MM Admin	“Deleting an Object in the Project Tree” on page 52
Archive and restore folders	MM Admin	“Archive and Restore Organizational Folders” on page 52
Create and upload life cycle templates	MM User, MM Adv User, MM Admin	“Creating Life Cycle Templates” on page 75
Create a workflow	MM Admin	“Creating a New Workflow” (p. 369)
Assign participants to a workflow	MM Admin	“Working with Workflow Participants” (p. 374)
View workflows that are associated with a version	MM Admin	“Viewing Workflows” (p. 370)

### **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 148
Import models	MM Adv User, MM Admin	Chapter 8, “Importing Models,” on page 125
Configure model properties	MM Adv User, MM Admin	“Set Model Properties” on page 145
Map model variables to project variables	MM Adv User, MM Admin	“Map Model Variables to Project Variables” on page 146
Run model comparison and model validation reports	MM Adv User, MM Admin	Chapter 10, “Validating Models Using Reports,” on page 179
Create user reports	MM Adv User, MM Admin	Chapter 11, “Validating Models Using User Reports,” on page 199

Task	Group	Topic
Create aggregated reports	MM Adv User, MM Admin	Chapter 21, “Aggregated Reports,” on page 339
Create scoring task output tables	MM Adv User, MM Admin	“Create Scoring Output Tables” on page 162
Create and run scoring tasks	MM Adv User, MM Admin	Chapter 9, “Scoring Models,” on page 157
Schedule a scoring task to execute	MM Adv User, MM Admin	“Schedule Scoring Tasks” on page 170

### 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
Set a champion model	MM Adv User, MM Admin	“Champion Models” on page 216
Flag a challenger model	MM Adv User, MM Admin	“Challenger Models” on page 219
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 Reports,” on page 179 Chapter 11, “Validating Models Using User Reports,” on page 199
Freeze or unfreeze versions	MM Admin	“Freezing Models” on page 220
Publish a project, version, or model to a SAS channel	MM Adv User, MM Admin	“Publishing Models to a SAS Channel” on page 224
Extract a model	any user who has the appropriate access rights to the SAS Metadata Repository	“Extract a Published Model” on page 228
Publish a model to the SAS Metadata Repository	MM Adv User, MM Admin	“Publish Models to the SAS Metadata Repository” on page 228
Publish a scoring function or model scoring files to a database	MM Adv User, MM Admin	“Publishing Models to a Database” on page 231

## 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 67
Monitor performance of project champion models that are within a project control group	MM Adv User, MM Admin	“Monitor Performance of Project Champion Models” (p. 117)
Run the Define Performance Task wizard	MM Adv User, MM Admin	Chapter 16, “Create Reports by Defining a Performance Task,” on page 263
Schedule a performance monitoring task to execute	MM Adv User, MM Admin	“Schedule Performance Monitoring Tasks” on page 273
Execute the performance monitoring SAS programs from the <b>Performance Monitor</b> node.	MM Adv User, MM Admin	“Run the Define Performance Task Wizard” on page 268
Run performance monitoring batch jobs	in <b>Test</b> mode: MM User, MM Adv User, MM Admin in <b>Production</b> mode: MM Adv, MM Admin	Chapter 17, “Create Reports Using Batch Programs,” on page 277
View monitoring reports and charts	MM User, MM Adv User, MM Admin	“View Reports” on page 311
Delete performance monitoring files from the <b>Resources</b> folder	MM Adv User, MM Admin	“Delete Performance Summary Data Sets” on page 276
Define dashboard report indicators	MM Adv User, MM Admin	“Create a Dashboard Report Definition” (p. 314)
Generate dashboard reports	MM Adv User, MM Admin	“Generate Dashboard Reports” (p. 319)
View dashboard reports	MM User, MM Adv User, MM Admin	“View Dashboard Reports” (p. 320)
Define a model retrain task	MM Adv User, MM Admin	“Define a Model Retrain Task” (p. 328)

Task	Group	Topic
Execute a model retrain task	MM Adv User, MM Admin	“Execute a Model Retrain Task” (p. 333)
View retrained models and the associated model comparison reports	MM User, MM Adv User, MM Admin	“Viewing Retrained Models and Model Comparison Reports” (p. 334)

## General Tasks

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

Task	Group or Role	Topic
Manage <b>Documents</b> folders and subfolders	MM User, MM Adv User, MM Admin	“Associate Documents with a Folder” on page 49
Update life cycle status	For a version life cycle, any SAS Model Manager user or group that is assigned to the role :LC Assignee.  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.	“Update Milestone Status” on page 95
Approve a life cycle task	For a version life cycle, any SAS Model Manager user or group that is assigned to the role :LC Approver.  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.	“Update Milestone Status” on page 95
Use the Query utility	MM User, MM Adv User, MM Admin	Appendix 1, “Query Utility,” on page 393
Set the status of a project champion model	MM Adv User, MM Admin	“Setting the Project Champion Model Status” on page 66
Replacing a champion model	MM Adv User, MM Admin	“Overview of Replacing a Champion Model” on page 247

Task	Group or Role	Topic
View workflow activities	MM User, MM Adv User, MM Admin  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.	<a href="#">“Viewing Workflow Activities” (p. 359)</a>
Work with workflow activities	MM User, MM Adv User, MM Admin  A user who is a workflow participant can claim, release, and complete activities.	<a href="#">“Working with Workflow Activities” (p. 360)</a>





## Part 2

---

# Working with Projects and Versions

<i>Chapter 3</i>	
<b>Working with Data Sources</b> .....	31
<i>Chapter 4</i>	
<b>Organizing the Project Tree</b> .....	47
<i>Chapter 5</i>	
<b>Working with Projects</b> .....	55
<i>Chapter 6</i>	
<b>Working with Versions</b> .....	71
<i>Chapter 7</i>	
<b>Working with Project Control Groups</b> .....	99



## Chapter 3

# Working with Data Sources

---

<b>Overview of Data Sources</b> . . . . .	<b>31</b>
<b>Project Tables</b> . . . . .	<b>33</b>
Project Control Tables . . . . .	33
Project Input Tables . . . . .	34
Project Output Tables . . . . .	34
Scoring Task Input Tables . . . . .	35
Scoring Task Output Tables . . . . .	35
Train Tables . . . . .	36
Test Tables . . . . .	36
Performance Tables . . . . .	36
<b>Creating Project Input and Output Tables</b> . . . . .	<b>37</b>
Create a Project Input Table . . . . .	37
Create a Project Output Table . . . . .	38
<b>Creating Scoring Task Input and Output Tables</b> . . . . .	<b>39</b>
About Scoring Task Input and Output Tables . . . . .	39
Create a Scoring Task Input Table . . . . .	39
Create a Scoring Task Output Table . . . . .	40
<b>Creating a Test Table</b> . . . . .	<b>40</b>
<b>Creating a Performance Table</b> . . . . .	<b>40</b>
About Performance Tables . . . . .	40
Naming a Performance Table for Use with the Define Performance Task Wizard . . . . .	41
Create a Performance Table . . . . .	42
<b>Using Tables from a Local or Network Drive</b> . . . . .	<b>42</b>
About Using Tables from a Local or Network Drive . . . . .	42
Create a Libref for a Local or Network Drive . . . . .	43
Modify the Path for a Libref . . . . .	45
Delete a Libref . . . . .	45

---

## Overview of Data Sources

Data sources are prototype tables and data tables that reside in the SAS Metadata Repository or in a SAS library on a local SAS Workspace Server or network drive, and are used by SAS Model Manager. You can view all registered data sources in SAS

Model Manager by clicking the Data Sources category view  button.

Project control tables are used to create the structure of the projects within a project control group. The tables must be registered as data sources in the SAS Management Console or a libref must be created to access tables on a local or network drive, before you create a project control table.

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 or a libref must be created to access tables on a local or network drive, 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 or challenger model, or to modify the project definition. For more information, see [“Modify Project Definition” on page 64](#).

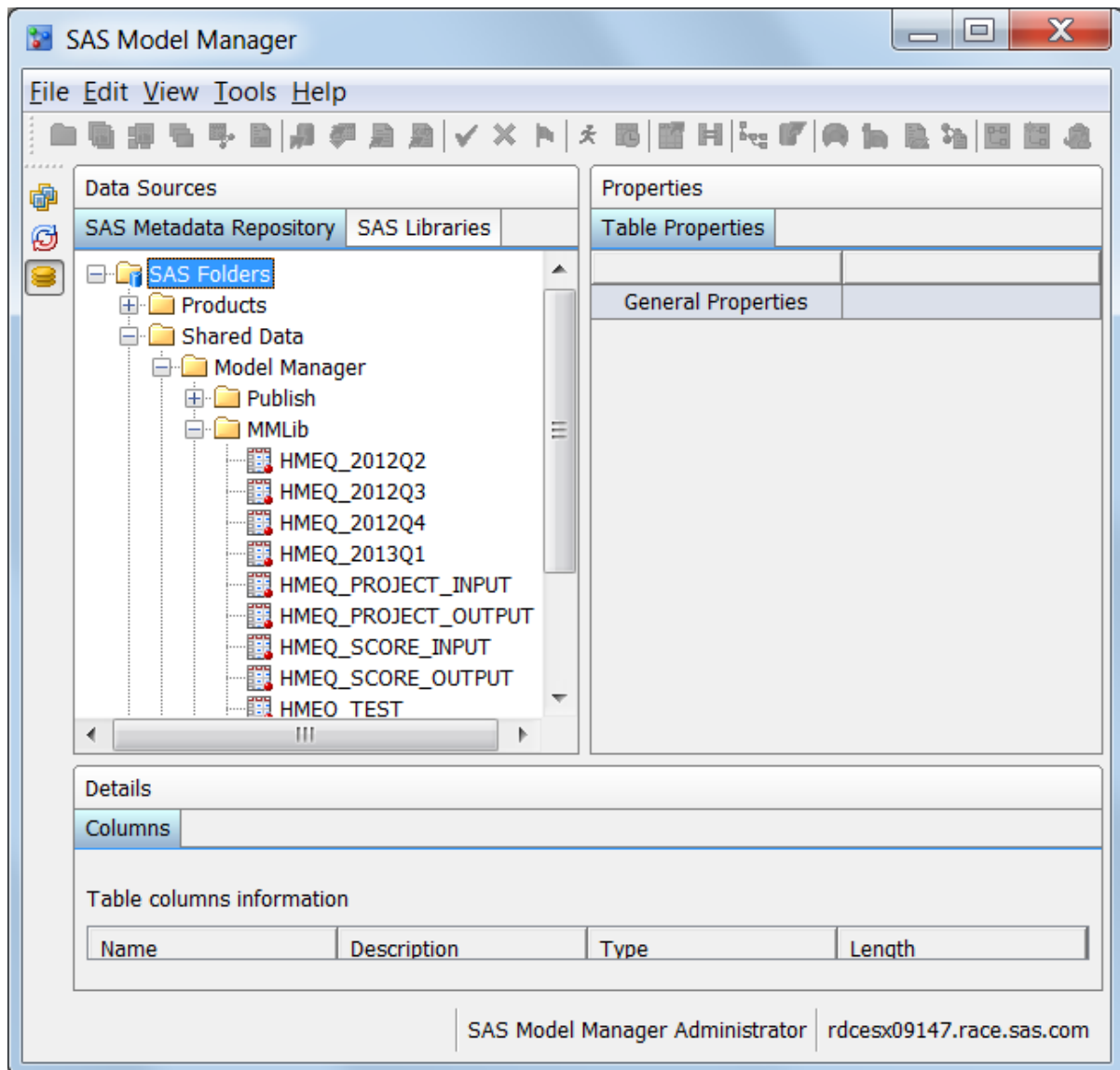
Use the following data tables to train or validate models, test models, and create monitoring reports:

- 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 or accessible with a libref 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 or you must define a libref using the **Edit Start-up Code** window to be able to access the table using SAS Model Manager.

After tables are accessible, you can view them through the SAS Model Manager Data Sources category view.



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

### See Also

“Using Tables from a Local or Network Drive” on page 42

## Project Tables

### Project Control Tables

A project control table is a data set that contains the projects, models, and segments that are used to create the structure of the projects within a project control group in the Project Tree. The project control table must at least contain a project variable with the name of project\_name. If you want to monitor the performance of the champion models

within a control group, then the project control table must also contain a segment ID variable. The segment ID variable must also be in the performance tables that are used to monitor performance. If you want to include the models for each project when creating a project control group, then the control table must also contain the model variable.

### **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 super set 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, either create the table and register the table using SAS Management Console or create a libref to access the table on a local SAS Workspace Server or network drive.

The project input variables must be available to SAS Model Manager either by specifying a project input table or by defining individual variables before you set a champion model. You can view input variables for a project in the **Input Variables** tab of the project's Details view or in the Data Sources category view.

*Note:* An alternative to using prototype tables to define the project input and output variables is to copy the variables from the champion or challenger model, or to modify the project definition. For more information, see [“Modify Project Definition” on page 64](#).

#### **See Also**

- [“About Defining Project Input and Output Variables” on page 61](#)
- [“Creating Project Input and Output Tables” on page 37](#)

### **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, either create the table and register the table using SAS Management Console or create a libref to access the table on a local or network drive.

SMM 12.3 Update where the user can view the output table and that the output variables must be available before you set a champion mode.

The project output variables must be available to SAS Model Manager either by specifying a project output table or by defining individual variables before you set a champion model. You can view output variables for a project in the **Output Variables** tab of the project's Details view or in the Data Sources category view.

*Note:* An alternative to using prototype tables to define the project input and output variables is to copy the variables from the champion or challenger model, or to modify the project definition. For more information, see [“Modify Project Definition” on page 64](#).

### **See Also**

- [“About Defining Project Input and Output Variables” on page 61](#)
- [“Creating Project Input and Output Tables” on page 37](#)

## **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 or create a libref to access the table on a local or network drive. In SAS Model Manager, you can view scoring task input tables in the Data Sources category view.

### **See Also**

[“Creating Scoring Task Input and Output Tables” on page 39](#)

## **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 or on a local or network drive. You can view the scoring results table under the scoring task node in the **Scoring Task** folder. The scoring task output table in the SAS Metadata Repository or on a local or network drive is not updated in test mode. In production mode, the contents of the scoring task output table in the SAS Metadata Repository or the local or network drive are replaced by the contents of the scoring results table. The scoring results table is not stored in the SAS Model Manager model repository or on a local or network drive.

Before you can create a scoring task, the scoring task output table must be added and accessible from the Data Sources category view. 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. Then, register the table in the SAS Metadata Repository using SAS Management Console or create a libref for the library to where the table resides.
- Use the SAS Model Manager Create Output Table window. When you use the Create Output Table window, SAS Model Manager creates the table in the library that is specified in the **Library** box. The table is registered in the SAS Metadata Repository and is available in the Data Sources category view.

You can view scoring task output tables in the Data Sources category view.

**See Also**

“Creating Scoring Task Input and Output Tables” on page 39

**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 or a libref must exist for a local or network drive.

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 publish a scoring function or model scoring files, and if the **Validate scoring results** box is selected when you publish scoring functions or model scoring files to a database.

*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 the Dynamic Lift report and the Interval Target Variable report 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.

To view test tables in SAS Model Manager, the tables must be registered in the SAS Metadata Repository or a libref must be defined for a local or network drive. In SAS Model Manager, you can view test tables in the Data Sources category view.

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 40

**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 2013. You might want to take a new sampling of the operational data in June 2013, September 2013, and



January 2014. These new tables are performance tables in the context of SAS Model Manager.

To view a performance table in SAS Model Manager, you must register the tables in the SAS Metadata Repository using SAS Management Console or you must define a libref for a local or network drive. You can view performance tables in the Data Sources category view. After a performance table is registered or a libref is defined, 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 40](#)

### See Also

- [“Using Tables from a Local or Network Drive” on page 42](#)
- “Deleting a Data Table” in Chapter 6 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 64](#).

### See Also

- [“About Defining Project Input and Output Variables” on page 61](#)
- *SAS 9.4 Statements: Reference*
- *SAS 9.4 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 output table, you can use the Modify Project Definition window to modify the project variables. For more information, see [“Modify Project Definition” on page 64](#).

### See Also

- [“About Defining Project Input and Output Variables” on page 61](#)

---

## 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 contains the data that is produced when you execute a scoring task. You can provide a scoring task output table or you can create a scoring output table definition in SAS Model Manager. When a scoring task is executed, SAS Model Manager uses the scoring output table definition to create the scoring output table. The name of the scoring output table definition is used as the name of the scoring output table.

You can create a scoring output table definition by selecting the Create Output Table function directly from the model. In the Create Output Table function, you select variables from a scoring task input table as well as variables from the model's output. The variables in the **Input Variables** table are variables from the scoring task input table if one is specified for the **Default Scoring Task Input Table** property for a project, version, or model property. Otherwise, the **Input Variables** table is empty. The **Output Variables** that appear in the window are model output variables. You use the variables from both tables to create the scoring output table.

SAS Model Manager saves the table definition as metadata in the SAS Metadata Repository. The location of the metadata is defined by the SAS library that you specify when you create the output table definition. After SAS Model Manager creates the table definition, the table can be selected as the output table for subsequent scoring tasks.

A SAS Model Manager scoring task can run in test mode, which is the default mode, or it can run in production mode. When the task runs, it populates a scoring task output table. In test mode, the scoring task output table is stored in the SAS Model Manager model repository. You view the table under the scoring task node in the version **Scoring Tasks** folder. In production mode, if the scoring output table is a table that you provided, that table is updated. If you created a scoring task output definition, the scoring output table is located in the designated SAS library that you specified when you created the table definition in the Create Output Table window. The production scoring task output table is not stored in the SAS Model Manager repository.

### *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. If the library that you select in the Create Output Table window is a folder in the SAS Metadata Repository, SAS Model Manager registers the table in the repository. You can view the table in the Data Sources category view of SAS Model Manager. For information, see [“Create Scoring Output Tables” on page 162](#).

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

[“Dynamic Lift Reports” on page 186](#)

---

## 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 or challenger models

- all output variables that are used by the champion model or challenger models
- 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, or you specify a performance table on a local or network drive. When you create a performance table, you can collect and name the performance table using a method that is most suitable for 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 or a libref must be defined for the local or network drive where the data set resides. 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. If the data set resides in the SAS Metadata Repository, 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 or where the data set is saved on the local or network drive.

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.

### See Also

- [“Determine How to Use the Performance Data Sets” on page 264](#)
- [“Using Tables from a Local or Network Drive” on page 42](#)

## 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 past 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;
```

---

## Using Tables from a Local or Network Drive

### About Using Tables from a Local or Network Drive

You can use tables from the local SAS Workspace Server or network drive to complete these SAS Model Manager tasks:

- Create a project
- Create projects from a control table
- Modify a project definition
- Create a scoring task
- Create a model retrain task
- Create reports
- Create a performance monitoring task

To use tables from the local or network drive, you must submit a LIBNAME statement to define a libref for the drive before you execute the SAS Model Manager task. You submit LIBNAME statements using the Edit Start-up Code window.



Here is the syntax:

**LIBNAME** *libref*<*engine*> '*SAS-library*' <*options*>

*libref*

specifies the one- to eight-character name of the SAS library. The first character must be a letter or an underscore (\_), and all other characters can be either letters or numbers. The *libref* cannot contain spaces.

*engine*

is a SAS engine name. This argument is optional. The default engine is the BASE engine.

*SAS-library*

is the physical pathname that is recognized by your operating environment. Enclose *SAS-library* in single quotation marks.

*options*

are LIBNAME statement options or engine-host options. For more information about options, see the LIBNAME statement in *SAS Statements: Reference* and the documentation for your operating environment.

Here is an example LIBNAME statement:

```
libname SalesLib 's:\sales\2013\october';
```

The BASE engine is used for tables in this library because the engine name was omitted from this statement. For more information about the LIBNAME statement, see *SAS Statements: Reference*.

SAS Model Manager provides the Edit Start-up Code window where you can submit the LIBNAME statement. Before you submit the LIBNAME statement, make sure that your SAS Model Manager user ID has permission to access the directory that you use in *SAS-library*.

### Create a Libref for a Local or Network Drive

To submit the LIBNAME statement, follow these steps:

1. Ensure that the path to the library that you want to create exists and that your SAS Model Manager user ID has access to the library.
2. Select **Tools** ⇒ **Edit Start-up Code**. The Edit Start-up Code window appears.

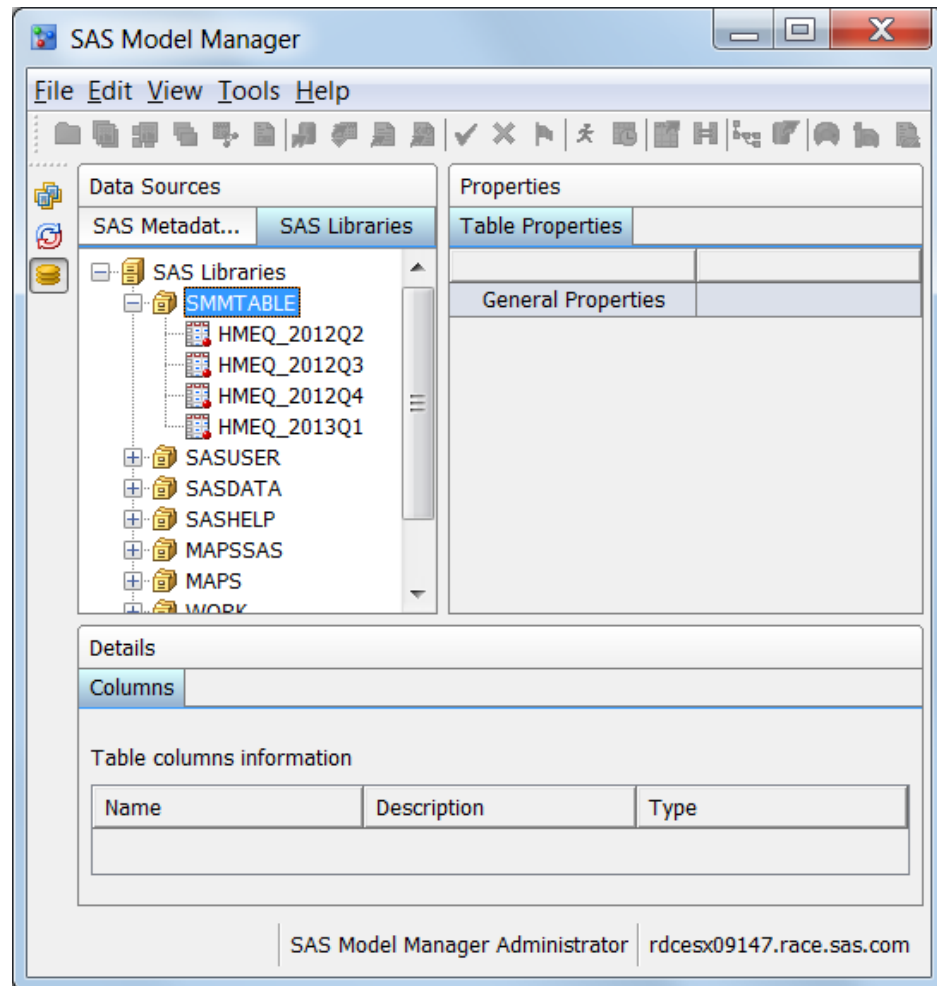


3. Enter the LIBNAME statement.
4. Click **Run Now**.  
A message indicates whether the libref was created. Click the **Log** tab to see the SAS log.
5. Click **OK**. The LIBNAME statement is saved in the Edit Start-up Code window.  
*Note:* If you save the code without running it by clicking **OK**, the code automatically runs the next time the middle-tier server starts.

If multiple LIBNAME statements are submitted for the same libref, the last LIBNAME statement defines the libref.

The librefs that you create can be viewed in the Data Sources category view. Select the **SAS Libraries** tab to view the list:





### **Modify the Path for a Libref**

You can modify the path that is used for a libref by resubmitting the LIBNAME statement using the same libref with a different path.

To modify a libref, delete the libref and resubmit the LIBNAME statement.

### **Delete a Libref**

You delete a libref using the Edit Start-up Code window.

1. Select **Tools** ⇨ **Edit Start-up Code**
2. Type `libname libref clear.`
3. Click **Run Now**.



## Chapter 4

# Organizing the Project Tree

---

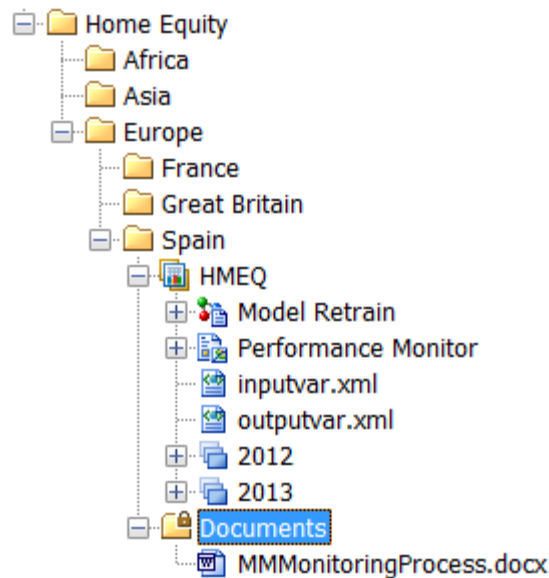
<b>Overview of the Project Tree</b> . . . . .	<b>47</b>
<b>Create an Organizational Folder</b> . . . . .	<b>48</b>
<b>Associate Documents with a Folder</b> . . . . .	<b>49</b>
About Associating Documents . . . . .	49
Create a Documents Folder . . . . .	50
Create a User-Defined Documents Subfolder . . . . .	50
Attach a Document to a Folder or a Subfolder . . . . .	51
Show Document Versions and Open or Save Documents . . . . .	51
<b>Deleting an Object in the Project Tree</b> . . . . .	<b>52</b>
<b>Archive and Restore Organizational Folders</b> . . . . .	<b>52</b>
About Archiving and Restoring Organizational Folders . . . . .	52
Archive an Organizational Folder . . . . .	52
Restore an Organizational Folder . . . . .	53

---

## 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 for your modeling projects. Create as many organizational folder levels as you need. Then, you can create projects, create project control groups, 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**. The **HMEQ** project has two versions, **2012** and **2013**, the **Model Retrain** folder, the **Performance Monitor** folder, XML files that contain the project input and output variables, and the **Documents** folder. The **Documents** folder contains the attached document **MMMMonitoringProcess.docx**.

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 48
- “Create a Project Control Group” on page 103
- “Create a Project” on page 61
- “Associate Documents with a Folder” on page 49
- “Create a Version” on page 89
- “Create an Aggregated Report” on page 340

*Note:* SAS Model Manager provides macros that you can use to programmatically add folders, projects, and versions. For more information, see [Appendix 5, “Macros for Adding Folders, Projects, Versions, and Setting Properties,”](#) on page 459.

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 to a SAS Channel](#)” on page 224.
- Search for models or tasks that are assigned to SAS Model Manager users. For more information, see [Appendix 1, “Query Utility,”](#) on page 393.

---

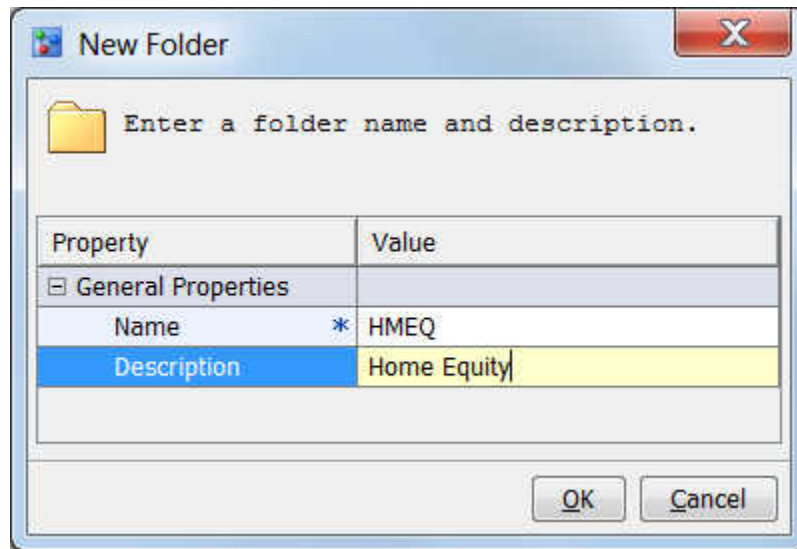
## Create an Organizational Folder

To create a folder:

1. Do one of the following:

- To create a folder under the **MMRoot** node, right-click **MMRoot** and select **New Folder**.
- To create a subfolder, right-click a folder and select **New** ⇒ **Folder**.

The New Folder window appears.



2. In the **Name** field, enter a folder name. The name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ). This is a required field.
3. (Optional) In the **Description** field, enter a folder description.
4. Click **OK**. SAS Model Manager adds the folder to the Project Tree.

### See Also

- [Appendix 5, “Macros for Adding Folders, Projects, Versions, and Setting Properties,” on page 459](#)

---

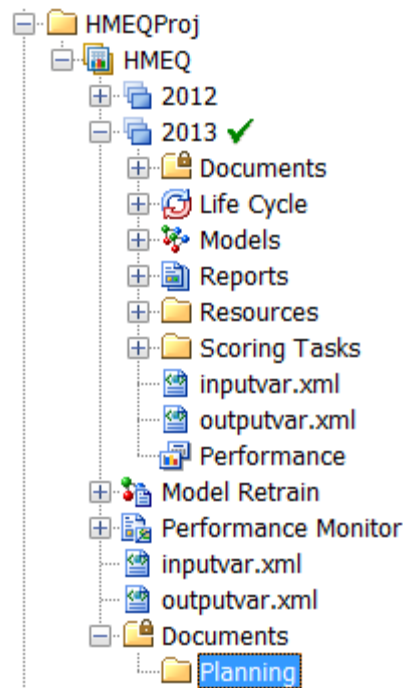
## Associate Documents with a Folder

### 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 a **Documents** folder or the **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.

SAS Model Manager creates a **Documents** folder when you create a version. You can create a **Documents** folder for an organization folder or for a project folder. You cannot add a **Documents** folder to a project control group folder. Documents can be organized by adding user-defined document subfolders to a **Documents** folder. The **Documents** folder cannot be deleted. Subfolders can be copied to other **Documents** folders or deleted.

Here is the user-defined subfolder **Planning** that is located in the **Documents** folder:



The lock on the **Documents** folder icon indicates that the folder cannot be deleted.

Documents that were attached to folders in previous releases of SAS Model Manager and migrated to the current version of SAS Model Manager are copied to the folders where they originally were attached.

You can edit only those file types that can be viewed in the SAS Model Manager text editor. Some of the file types are .txt, .sas, .log and .html. 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 on which the document was attached to the folder. From the Show History window, you can open the document or save a version of a document.

### Create a Documents Folder

To create a **Documents** folder for an organization folder or a project folder, right-click the folder and select **New** ⇒ **Documents Folder**.

### Create a User-Defined Documents Subfolder

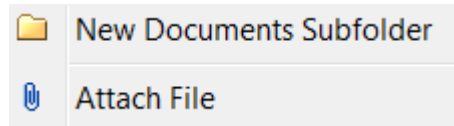
To create a user-defined **Documents** subfolder:

1. Right-click the **Documents** folder and select **New Documents Subfolder**.
2. In the New Documents Subfolder window **Name** box, enter a folder name. The name cannot be Documents.
3. (Optional) In the **Description** box, enter a folder description.
4. Click **OK**.

### Attach a Document to a Folder or a Subfolder

To attach a document to a **Documents** folder or to a **Resources** folder, follow these steps:

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



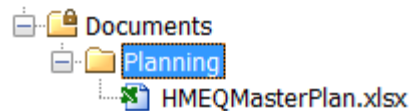
The Attach File window appears.

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**.

*Note:* If the selected file format is Text, the file encoding field is enabled. You can select the system recognized encoding from the drop-down list.

4. Click **OK**. SAS Model Manager attaches the document to the folder.

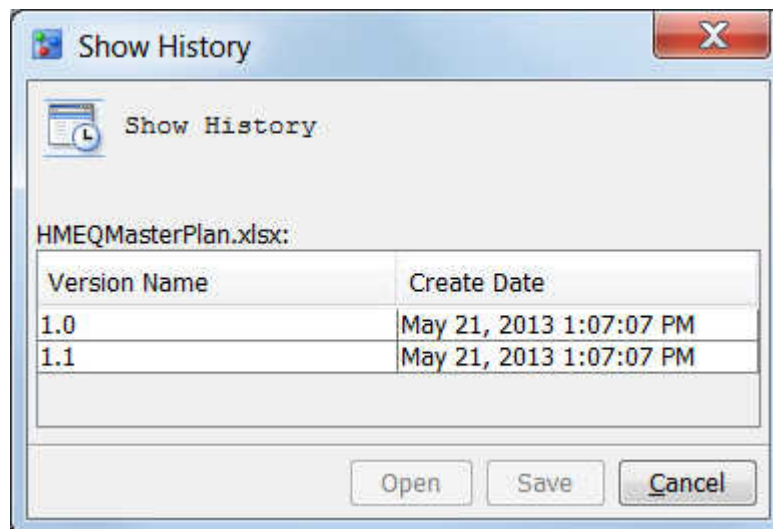
Here is an example of a document that is attached to a subfolder:



### Show Document Versions and Open or Save Documents

To show all of a document's versions, open 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 open 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:

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.

---

## Archive and Restore Organizational Folders

### *About Archiving and Restoring Organizational Folders*

Using the archive and restore facility, a SAS Model Manager administrator can back up the **MMRoot** folder or an organizational folder in one repository and restore it to another repository. The folder is archived as a compressed ZIP file.

When you restore an organizational folder, you must first create an organizational folder to restore it to. A best practice is to give the restored organizational folder the same name as the archived ZIP file. The contents of the archived organizational folder are restored to the new organizational folder.

Organizational folders cannot be restored in these situations:

- The organizational folder to be restored is the **MMRoot** folder but the archive folder is a folder other than **MMRoot**.
- The name of the organizational folder to be restored is the same as the project name in the archived folder.
- The same archived ZIP file has already been restored in an organizational folder on the same WebDAV server.

### *Archive an Organizational Folder*

To archive an organizational folder:



1. If the organizational folder name is the same name as a project in the folder, rename either the folder or the project. Organizational folders that have the same name as a project name cannot be restored.
2. Right-click the organizational folder and select **Archive**.
3. In the **Save** window, select the folder where the ZIP file is to be saved.
4. In the **File name** field, name the ZIP file. A best practice is to use the default name, which is the name of the organizational folder.
5. Click **Save**.

### **Restore an Organizational Folder**

To restore an archived **MMRoot** folder:

1. Right-click the **MMRoot** folder and select **Restore**.
2. In the Open window, navigate to the archived **MMRoot** folder and select the ZIP file. The **File name** field contains the name of the archived ZIP file.
3. Click **Open**.
4. A message indicates that the restore was successful. Click **Close**.

To restore an organizational folder:

1. Right-click the **MMRoot** folder and select **New Folder**. Enter a name for the organizational folder and click **OK**.
2. Right-click the organizational folder and select **Restore**.
3. In the Open window, navigate to the archived organizational folder and select the ZIP file. The **File name** field contains the name of the archived ZIP file.
4. Click **Open**.
5. A message indicates that the restore was successful. Click **Close**.



## Chapter 5

# Working with Projects

---

<b>Overview of Projects</b> . . . . .	<b>55</b>
What Is a SAS Model Manager Project? . . . . .	55
How a Project Folder Is Organized . . . . .	56
Project Folder Tasks . . . . .	57
Project Metadata . . . . .	58
<b>Planning a Project</b> . . . . .	<b>58</b>
<b>Prerequisites for Creating Projects</b> . . . . .	<b>60</b>
<b>About Defining Project Input and Output Variables</b> . . . . .	<b>61</b>
<b>Create a Project</b> . . . . .	<b>61</b>
<b>Modify Project Definition</b> . . . . .	<b>64</b>
<b>Lock or Unlock Project Metadata</b> . . . . .	<b>66</b>
<b>Setting the Project Champion Model Status</b> . . . . .	<b>66</b>
About the Champion Model Status . . . . .	66
Setting the Champion Model Status . . . . .	66
<b>Project Properties</b> . . . . .	<b>67</b>
About Project Properties . . . . .	67
Project-Specific Properties . . . . .	67

---

## 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.

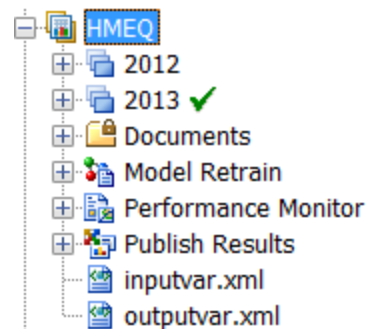
Project work is organized in one or more time-based intervals that are called versions. Each version contains the documents, models, reports, resources, scoring tasks, and performance information for a time interval. Each version can have its own life cycle definition for tracking the progress of a project. For more information, see [Chapter 6](#), “Working with Versions,” on page 71.

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

Multiple projects can be created within a project control group. Some project tasks cannot be performed for the projects that are within a project control group. For more information, see “[Overview of Project Control Groups](#)” on page 99.

### How a Project Folder Is Organized

The SAS Model Manager project is a folder that contains versions, an optional **Documents** folder, a **Model Retrain** folder, a **Performance Monitor** folder, a **Publish Results** folder, and the project input and output XML files. 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, **2012** and **2013**.

#### Model Retrain

contains reports for models that have been retrained.

#### Performance Monitor

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

#### Publish Results

contains the results after publishing models to a database.

#### 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 champion or challenger model.

#### 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 champion or challenger model.

#### Documents

contain documents that are associated with the project. This folder is optional.

*Note:* Documents that were associated with a project in previous releases of SAS Model Manager and migrated to this release remain in the Project Tree as they originally were attached and not in the **Documents** folder.

## 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 89](#).

### **New Documents Folder**

creates a new **Documents** folder to contain auxiliary documents, such as project notes and schedules. For more information, see [“Associate Documents with a Folder” on page 49](#).

### **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 66](#).

### **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 64](#).

### **Publish Models to a Database**

transforms DATA step score code of a model to a scoring function or model scoring files, and writes them to the database. For more information, see [“Publishing Models to a Database” on page 231](#).

### **Publish Models to a SAS Channel**

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 225](#).

### **Publish Models to the SAS Metadata Repository**

publishes the champion model to the SAS Metadata Repository in order for the champion model to be run in a test or production scoring environment. For more information, see [“Publish Models to the SAS Metadata Repository” on page 228](#).

### **Define Performance Task**

starts a wizard that generates the SAS code to monitor the performance of a champion model or challenger model. For more information, see [“Overview of Creating Reports Using a Performance Task” on page 263](#).

### **Dashboard Report Definition**

starts a wizard to create performance indicators that are used to execute the dashboard reports. For more information, see [“Create a Dashboard Report Definition” on page 314](#).

### **Define Model Retrain Task**

starts a wizard that retrains one or more models. For more information, see [Chapter 20, “Retraining Models,” on page 327](#).

### **Define Aggregated Report**

specifies reports from the **Reports** folder that are used to create a single aggregated report. For more information, see [Chapter 21, “Aggregated Reports,” on page 339](#).

**Query**

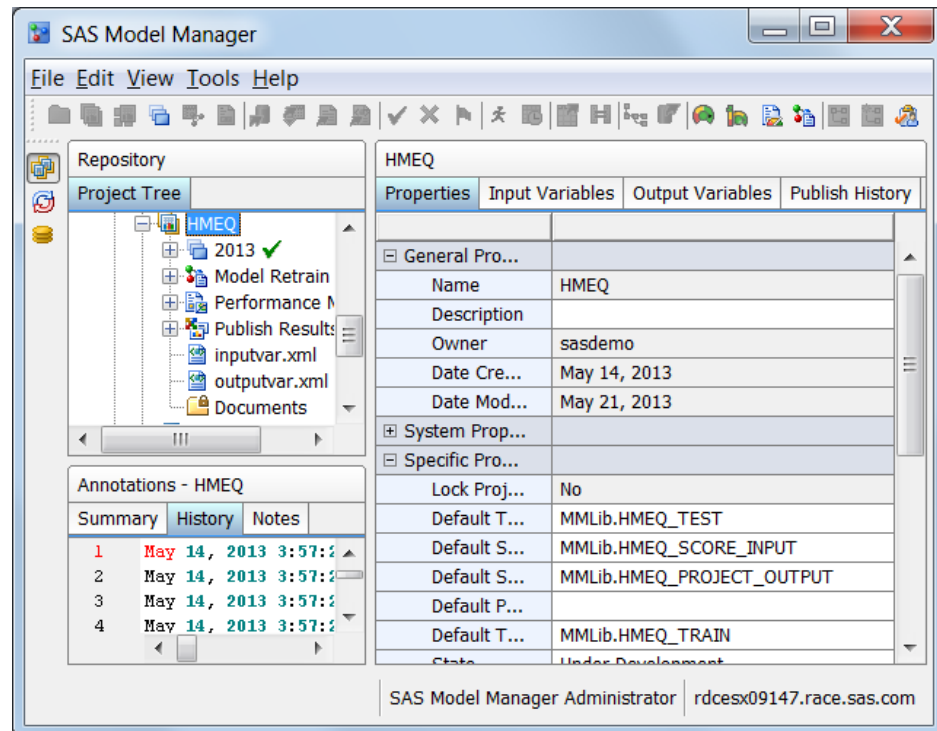
searches for models and tasks that are assigned to SAS Model Manager users. For more information, see “[Overview of the Query Utility](#)” on page 393.

**Project Metadata**

When you click on a project folder, the detail view on the right has tabs that categorize the project metadata:

- The **Properties** tab shows the general metadata for the project. For more information, see “[Project Properties](#)” on page 67.
- The **Input Variables** tab shows the project input variables and their attributes.
- The **Output Variables** tab shows the output variables and their attributes.
- The **Publish History** tab shows the models that have been published to a database, a SAS channel or to the SAS Metadata Repository. The metadata includes the published name, the destination, the method of publishing, the server name, the user ID that published the model, the date on which the model was published, the model function, whether the model is a champion model or a challenger model, the project name, the version name, the model name, and the model level.

Here are the project metadata tabs:

**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:

*Note:* For more information about creating multiple projects in a control group, see [“Overview of Project Control Groups” on page 99](#).

- 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 20](#) 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 47](#).
- What models do you want to use in the project? If the models were created using SAS Enterprise Miner, SAS/STAT, or the SAS/ETS procedures COUNTREG and SEVERITY, all model components are available to SAS Model Manager when you import the model. If your model is a SAS code model that is not contained in a miningresult.spk file or a model that was 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 130](#) and [“Import PMML Models” on page 143](#).
- 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, copy the variables from an existing champion model, or 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 or you must create a libref for files that are stored on a local or network drive before you create the project. For more information, see [“About Defining Project Input and Output Variables” on page 61](#).
- What method do you want to use to track the progress of a version? The Workflow Console enables you to track the progress of activities from the version level. A SAS Model Manager administrator can create a workflow and associate it with a 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 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 346](#).
  - If you decide to use the life cycle feature to monitor the progress of your 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 75](#).
- You might have project documents that you would like to access from SAS Model Manager. SAS Model Manager enables you to attach documents to a **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 49](#).
- SAS Model Manager provides several reports that you can use to help you 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 Reports,”](#) on page 179 and [Chapter 11, “Validating Models Using User Reports,”](#) on page 199.

- When you publish a project champion model to the SAS Metadata Repository, you must specify a folder to which you can publish 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 [“Publish Models to the SAS Metadata Repository”](#) on page 228.
- 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 40.
- 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 259.

---

## 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 70.

If you use prototype tables to define the project input and output variables, you must do one of the following two things before you can create a project: either create the project input and output tables and register them in the SAS Metadata Repository using SAS Management Console or create a libref for files on a local SAS Workspace Server or network drive. You then can view the data tables from the Data Sources category view 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 37.
- The *SAS Model Manager: Administrator's Guide* has instructions on registering project input and output tables in SAS Management Console.



For information about prerequisites for project control groups, see [“Prerequisites for Creating Project Control Groups” on page 101](#).

---

## 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 and challenger models. SAS Model Manager requires that the project input and output variables be defined before a champion model can be published to a production environment. You can define the project input and output variables when you create a project or during the champion model selection process.

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 champion model and the project variables have not been defined, SAS Model Manager prompts you to add model input variables to the project and 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 set a champion model, the `inputvar.xml` file and the `outputvar.xml` file are copied to the version folder.

### See Also

- [“Project Tables” on page 33](#)
- [“Creating Project Input and Output Tables” on page 37](#)
- [“Create a Project” on page 61](#)
- [“Modify Project Definition” on page 64](#)

---

## Create a Project

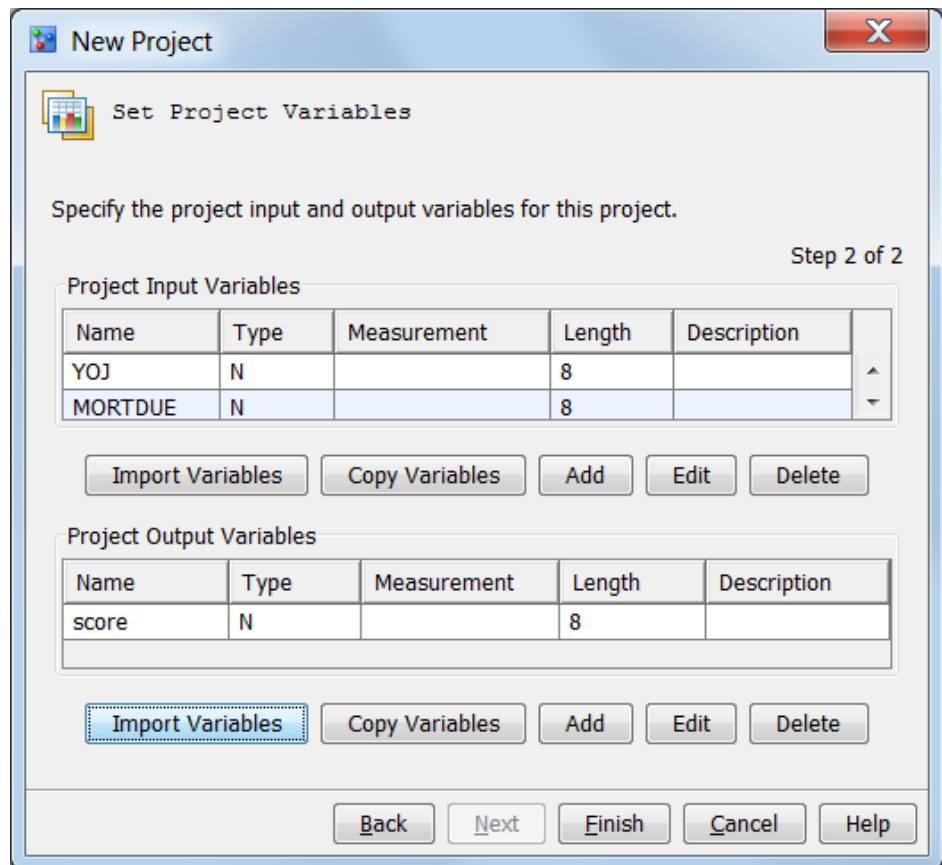
*Note:* SAS Model Manager does not support tables or models whose data sets contain special characters and that were created when the system option `VALIDMEMNAME=EXTEND` was set. SAS Model Manager can use tables whose variables contain special characters only when the SAS Model Manager administrator has enabled the use of the `VALIDVARNAME=` system option in SAS Management Console. For more information, see the *SAS Model Manager Administrator's Guide*.

To create a project:

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

Property	Value
[-] General Properties	
Name *	
Description	
[-] Project Properties	
Model Function *	Classification

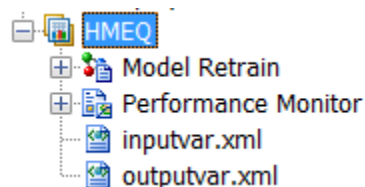
2. Enter a name and a description for the project that you are creating. The name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ). The **Name** field is required.
3. Click the **Model Function** box and select the function type for the model. Click **Next**.
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.



*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 **Performance Monitor** 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 **Performance Monitor** 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. The input and output XML files 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 61](#)

- “Create a Project Control Group” on page 103

---

## 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 and advanced users can use this feature.

The project definition cannot be modified if it has been locked. Only SAS Model Manager administrators can lock or unlock a project definition. The project definition for projects that are within a project control group cannot be modified.

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.

*Note:* SAS Model Manager does not support tables or models whose data sets contain special characters and that were created when the system option VALIDMEMNAME=EXTEND was set. SAS Model Manager can use tables whose variables contain special characters only when the SAS Model Manager administrator has enabled the use of the VALIDVARNAME= system option in SAS Management Console. For more information, see the *SAS Model Manager Administrator's Guide*.

To modify a project input table or project output table property:

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	Measu...	Length	Descri...
YOJ	N		8	
MORTDUE	N		8	

Import Variables Copy Variables Add Edit Delete

**Project Output Variables**

Name	Type	Measu...	Length	Descri...
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 enter 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 or from a SAS library.
  - 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.
4. Click **OK**.

### See Also

- “Specific Properties for a Project” on page 505
- “Project Tables” on page 33
- “Create a Project” on page 61

- [“Create a Project Control Group” on page 103](#)

---

## 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.

*Note:* The project definition for projects that are within a project control group cannot be modified.

- To lock the metadata for a project, right-click the project folder and select **Lock Project Metadata**. Click **Close**. A check mark appears to the left of the task in the pop-up menu.
- To unlock the metadata for a project, right-click the project folder and select **Lock Project Metadata**. Click **Close**. The check mark is removed from the task in the pop-up menu.

### See Also

[“Modify Project Definition” on page 64](#)

---

## Setting the Project Champion Model Status

### About the Champion Model Status

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

#### **Under 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:

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

State	Under Development
Default Version	Under Development
Model Function	Active
Interested Party	Inactive
Training Target Varia...	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 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.

### Project-Specific Properties

Property Name	Description
<b>Lock Project Metadata</b>	Specifies that the project metadata is locked and the project definition cannot be modified.
<b>Default Test Table</b>	Specifies a default SAS data set that can be used to create model assessment reports such as dynamic lift charts.
<b>Default Scoring Task Input Table</b>	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 <b>Default Scoring Task Input Table</b> property, the value is used as the default input table in the New Scoring Task window.

Property Name	Description
<b>Default Scoring Task Output Table</b>	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 <b>Default Scoring Task Output Table</b> property, the value is used as the default output table in the New Scoring Task window.
<b>Default Performance Table</b>	<p>Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager project.</p> <p>The value of the <b>Default Performance Table</b> property is used as the default value for the <b>Data Source</b> column in the Define Performance Task wizard if a default performance table is not specified for a version or for the model.</p>
<b>Default Train Table</b>	<p>The train table is optional and is used only as information. However, when a value is specified for a model's <b>Default Train Table</b> property, SAS Model Manager does the following:</p> <ul style="list-style-type: none"> <li>• uses default train table to validate scoring functions or scoring model files when a user publishes the associated project champion model or challenger models to a database.</li> <li>• checks the <b>Validate scoring results</b> box in the Publish Scoring Function window.</li> </ul> <p>The value of the <b>Default Train Table</b> property is used to validate scoring functions or scoring model files only if a default train table is not specified for a version or for the model.</p>
<b>State</b>	<p>Specifies the current state of the project:</p> <p><b>Under Development</b> specifies the time period from the project start to the time where the champion model is in a production environment.</p> <p><b>Active</b> specifies the time period where the champion model is in a production environment.</p> <p><b>Inactive</b> specifies the time period when a project is temporarily suspended from the production environment.</p> <p><b>Retired</b> specifies that the champion model for this project is no longer in production.</p>
<b>Default Version</b>	Specifies the version that contains the champion model in a production environment.



Property Name	Description
<b>Model Function</b>	Specifies the type of output that your predictive model project generates. The <b>Model Function</b> 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 <b>Model Function</b> 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 <a href="#">Types of Model Functions on page 70</a> .
<b>Interested Party</b>	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.
<b>Training Target Variable</b>	Specifies the name of the target variable that was used to train the model.
<b>Target Event Value</b>	The target variable value that defines the desired target variable event.
<b>Class Target Values</b>	For class, nominal, ordinal, or interval targets, the set of possible outcome classes, separated by commas. For example, binary class target values might be <b>1, 0</b> or <b>Yes, No</b> . Nominal class target values might be <b>Low, Medium, High</b> . These values are for information only.
<b>Class Target Level</b>	Specifies the class target level of binary, nominal, ordinal, or interval.
<b>Output Event Probability Variable</b>	The output event probability variable name, when the <b>Model Function</b> property is set to <b>Classification</b> .
<b>Output Prediction Variable</b>	The output prediction variable name, when the <b>Model Function</b> property is set to <b>Prediction</b> .
<b>Output Classification Variable</b>	The output classification variable name, when the <b>Model Function</b> property is set to <b>Classification</b> .
<b>Output Segmentation Variable</b>	The output segmentation variable name, when the <b>Model Function</b> property is set to <b>Segmentation</b> .

**Table 5.1** Types of Model Functions

Model Function	Description	Example
<b>Analytical</b>	Function for any model that is not Prediction, Classification, or Segmentation.	None
<b>Prediction</b>	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.
<b>Classification</b>	Function for models that have target variables that contain binary, categorical, or ordinal values.	<b>DEFAULT_RISK</b> = {Low, Med, High}
<b>Segmentation</b>	Function for segmentation or clustering models.	Clustering models
<b>Any</b>	Specify <b>Any</b> when you import a SAS code model and you want a choice of the model template to use in the <b>Local Files</b> window. When you specify <b>Any</b> , SAS Model Manager lists the available model templates in the <b>Choose a model template</b> list in the Local Files window.	None

## Chapter 6

# Working with Versions

---

<b>Overview of Versions</b> . . . . .	<b>71</b>
What Is a SAS Model Manager Version? . . . . .	71
How a Version Folder Is Organized . . . . .	72
Version Folder Tasks . . . . .	73
Version Metadata . . . . .	74
<b>Creating Life Cycle Templates</b> . . . . .	<b>75</b>
Overview of Creating Life Cycle Templates . . . . .	75
The SAS Model Manager Template Editor Window . . . . .	76
Life Cycle Template Participants . . . . .	78
The Browse Templates Window . . . . .	80
Create a Life Cycle Template from a Sample Template . . . . .	81
Create a New Life Cycle Template . . . . .	82
Modify a Life Cycle Template . . . . .	84
Delete a Life Cycle Template . . . . .	85
Life Cycle Template Properties . . . . .	86
<b>Create a Version</b> . . . . .	<b>89</b>
<b>Version Properties</b> . . . . .	<b>90</b>
About Version Properties . . . . .	90
Version-Specific Properties . . . . .	90
<b>Working with Life Cycles</b> . . . . .	<b>92</b>
Overview of a Life Cycle . . . . .	92
Life Cycle Tasks . . . . .	93
Life Cycle Properties . . . . .	96

---

## Overview of Versions

### *What Is a SAS Model Manager Version?*

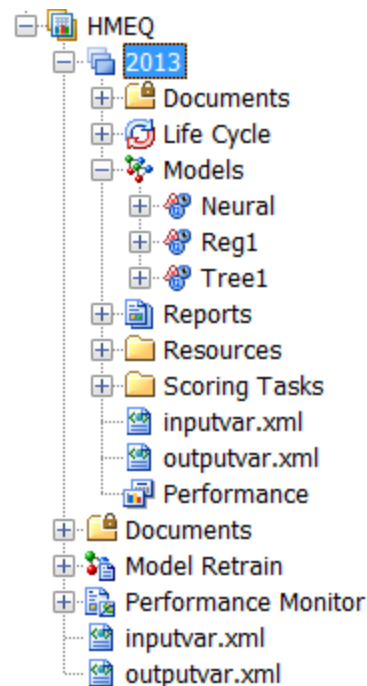
After a project is created, you create a version folder to import your models, score your models, run reports, and monitor the life cycle of these models. A version is often the time-phased container for your SAS Model Manager projects. The time interval for a project cycle is specified when you create 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 2013. 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, a champion model after it is selected, and challenger models. 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 Tasks**

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:

**New Workflow**

creates a new workflow from a process definition and associates it with the selected version. For more information, see [“Creating a New Workflow” on page 369](#).

**Freeze Version**

disables or enables modifications of some version models properties and files. When a version is frozen, a check mark appears next to the task. 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 220](#).

**Publish Models to a SAS Channel**

publishes models to defined channels and notifies subscribers of the publication channel when the models are delivered. For more information, see, [“Publishing Models to a SAS Channel” on page 224](#).

**Define Aggregate Report**

combines multiple reports from the **Reports** folder to create a single report. For more information, see [Chapter 21, “Aggregated Reports,” on page 339](#).

**Query**

searches for model and tasks that are assigned to SAS Model Manager users. For more information, see [Appendix 1, “Query Utility,” on page 393](#).

**View Workflow**

enables the user to view the workflow instance that is associated with the selected version. For more information, see [“Viewing Workflows” on page 370](#).

**Generate Training Summary Data Set**

generates a data set that is used to create the Train Table Summary report. For more information, see [“Training Summary Data Set Reports” on page 196](#).

**See Also**

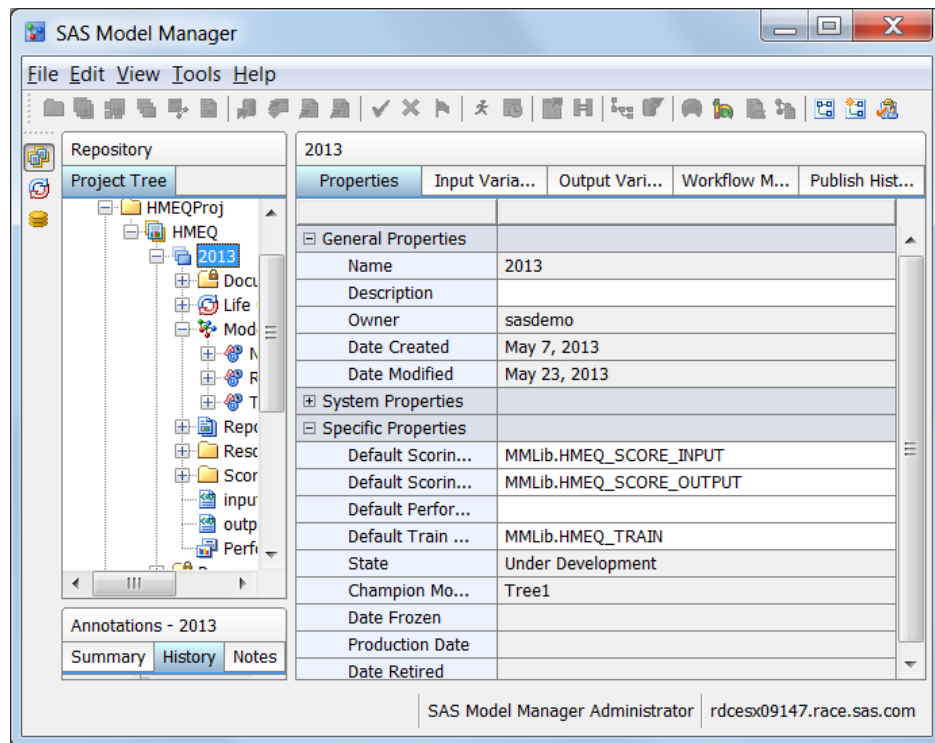
- [“Overview of Importing Models” on page 125](#)
- [“Publish Models to the SAS Metadata Repository” on page 228](#)

**Version Metadata**

When you click on a version folder, the detail view on the right has tabs that categorize the version metadata:

- The **Properties** tab shows the general metadata for the version. For more information, see [“Version Properties” on page 90](#).
- The **Input Variables** tab shows the input variables and their attributes for the champion model. Metadata appears on this tab only after the champion model has been set.
- The **Output Variables** tab shows the output variables and their attributes for the champion model. Metadata appears on this tab only after the champion model has been set.
- The **Workflow Milestones** shows the activities from the SAS Model Manager Workflow Console that are associated with milestones. The metadata includes the milestone or task, the status, the date started, the date completed, and the user ID that modified the milestone or task status.
- The **Publish History** tab shows the models that have been published to a database, a SAS channel or to the SAS Metadata Repository. The metadata includes the published name, the destination, the method of publishing, the server name, the user ID that published the model, the date on which the model was published, the model function, whether the model is a champion model or a challenger model, the project name, the version name, the model name, and the model level.

Here are the version metadata tabs:



## 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 category view. Templates that appear in the Life Cycles category view 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 by using a new name. Alternatively, 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 by 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.

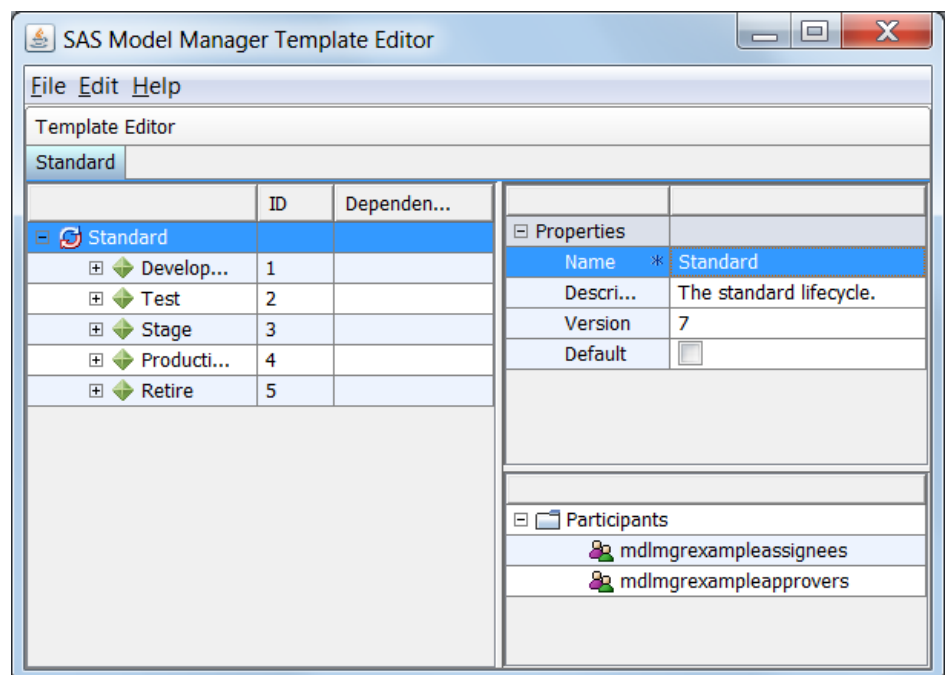
If you create a life cycle template using an XML file, you can copy any life cycle template from the user-templates directory and modify the file with any text editor. When you modify an XML template file, you specify the milestone and task properties as XML elements and element attributes. SAS Model 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.

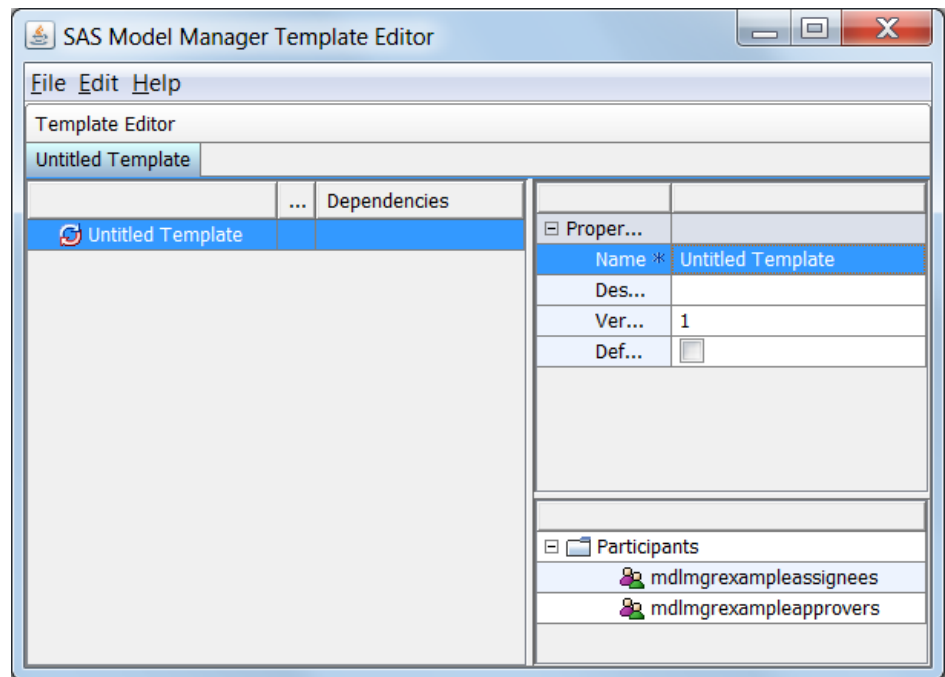
To open a life cycle template in the **SAS Model Manager Template Editor** window:

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**.





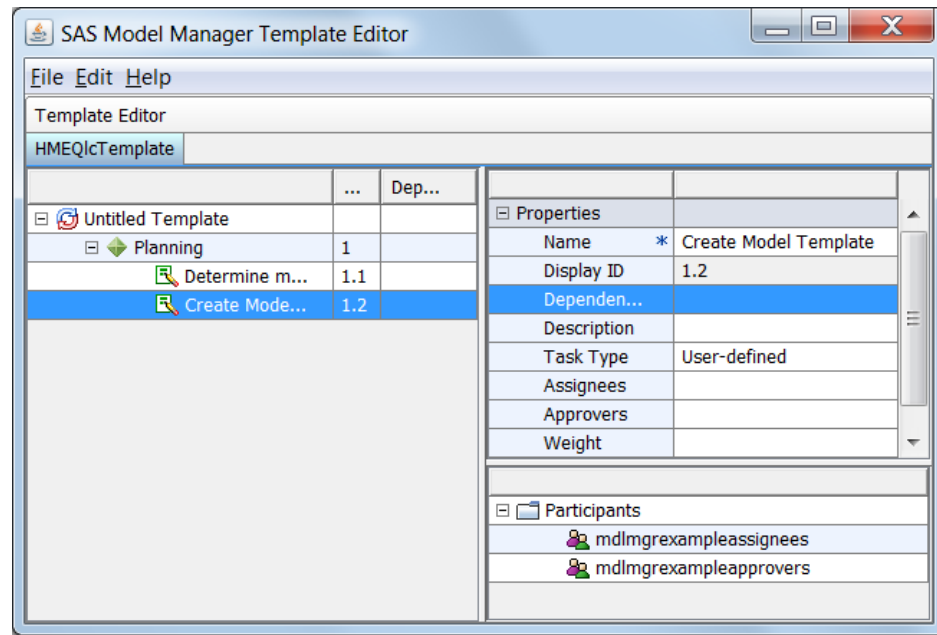
- To open a life cycle template that is stored on a local or network location, select **Open**.

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 on 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 86.

The lower right pane displays the life cycle participants. Participants 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 **Participants** list. Only users and groups that are assigned this role for a life cycle, and are in the **Participants** 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** box 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** box 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 a life cycle template in 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 task **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** box. 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 boxes **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 boxes. These boxes 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 the boxes.

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** box. If you specify a group, any member of the group can update the task **Status** box. The user ID of the group member who changed the status appears in the **Completed By** or **Approved By** boxes.

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** box without having to create a new version and a life cycle template that includes that individual user.

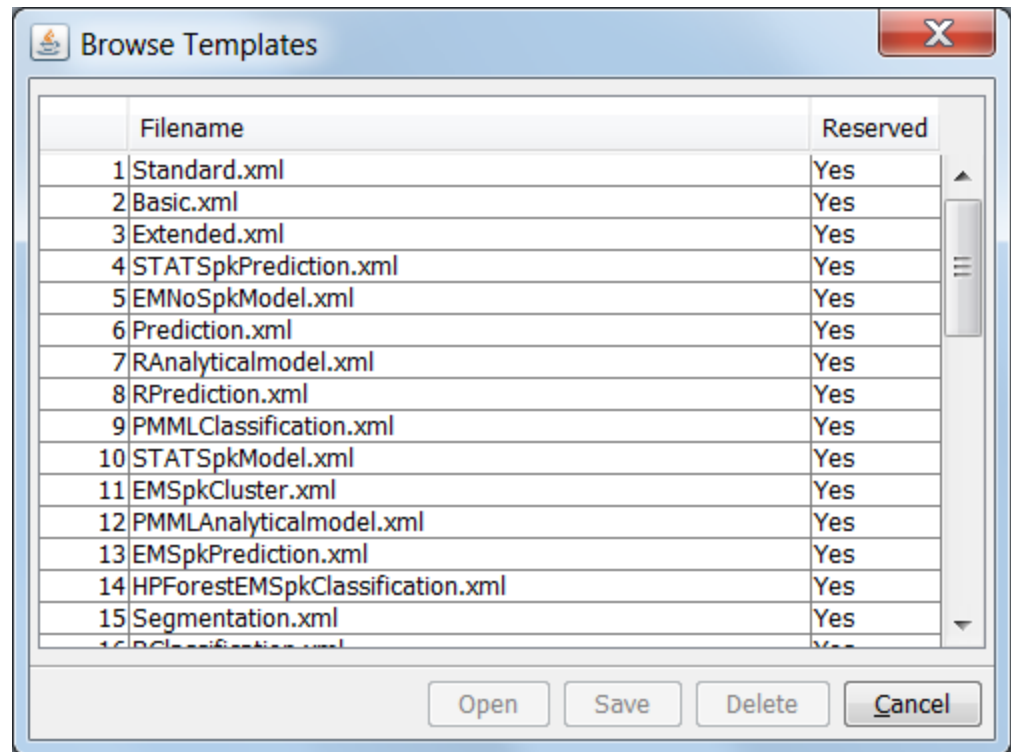
When you assign a group to be an **Assignee** or an **Approver**, all users and groups in that group have the authority to change the task status. Therefore, ensure that the users and groups that are defined in the group are those users and groups that you intend to be an **Assignee** or **Approver**.

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. 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 category view 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:

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** boxes. The **Name** and **Type** boxes are required.
  - b. To modify milestone properties, click the property and modify the properties in the **Properties** pane. Properties with an asterisk (\*) are required. For a description of the properties, see “[Milestone Properties](#)” on page 86.
  - c. To add a task, right-click a milestone and select **New Task**. In the New Task window, complete the **Name**, **Description**, and **Type** boxes. The **Name** and **Type** boxes 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 box for a list of values or click  to assign a value. For a description of the properties, see “[Task Properties](#)” on page 87.
  - 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:

*Note:* You can view sample life cycle templates in the Life Cycles category view. Only a user or group with the role of Model Manager: Administration Usage can upload a template.

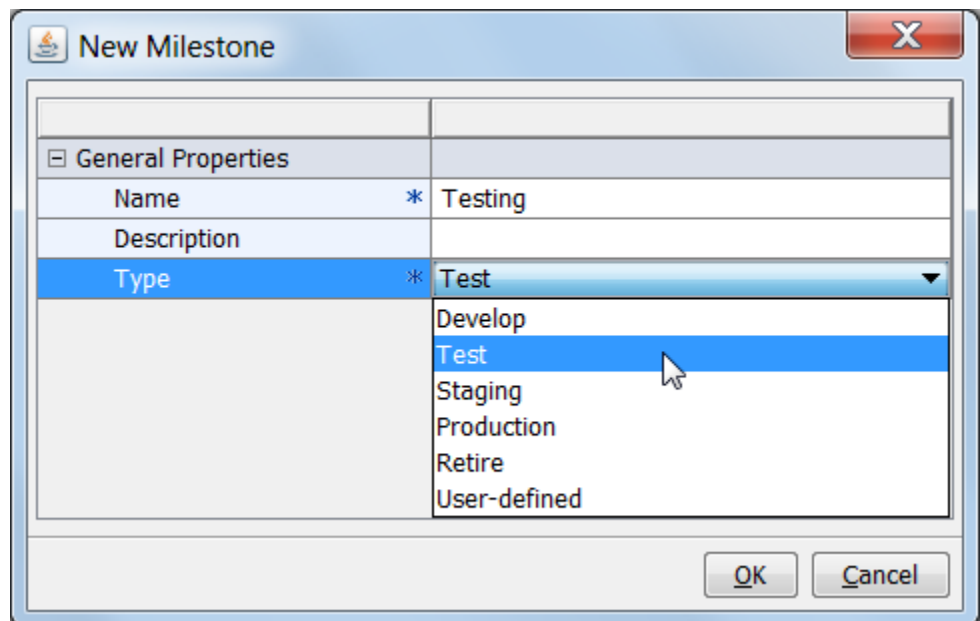
1. From the SAS Model Manager Template Editor window, select **File** ⇒ **New Life Cycle Template**.

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. Enter a name in the **Name** box and select **File** ⇒ **Save As**. In the Save window, select the folder to save the template to. In the **File name** box, enter 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 remove these example groups, the **To Be Completed By** and the **To Be Approved By** version life cycle task properties displays 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 86.
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.



Complete these boxes:

- a. Enter a name and an optional description for the new milestone. The **Name** box is required.

- b. Click the **Type** box, select a milestone type, and then click **OK**. For more information, see “[Milestone Properties](#)” on page 86. 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							
<div style="border: 1px solid gray; padding: 5px;"> <div style="display: flex; justify-content: space-between; align-items: center;"> <span>☐ General Properties</span> </div> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr style="background-color: #e6f2ff;"> <td style="padding: 5px;">Name *</td> <td style="padding: 5px;">Determine reports needed</td> </tr> <tr> <td style="padding: 5px;">Description</td> <td style="padding: 5px;"></td> </tr> <tr style="background-color: #e6f2ff;"> <td style="padding: 5px;">Type *</td> <td style="padding: 5px;">User-defined</td> </tr> </table> </div>		Name *	Determine reports needed	Description		Type *	User-defined
Name *	Determine reports needed						
Description							
Type *	User-defined						
<div style="display: flex; justify-content: flex-end; gap: 20px;"> <span>OK</span> <span>Cancel</span> </div>							

Complete these boxes:

- a. Enter a name and an optional description for the task. The **Name** box is required.
  - b. Click the **Type** box and select a task type. Click **OK**. For more information, see “[Task Properties](#)” on page 87.
8. For each task, complete the task properties. For more information, see “[Task Properties](#)” on page 87.
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:



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 86](#).
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 86](#).

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 87](#).

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:

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
<b>Name</b>	Identifies the name of the life cycle template. This property is required.
<b>Description</b>	Specifies user-defined information about the life cycle template.
<b>Version</b>	<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>
<b>Default</b>	<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
<b>Name</b>	Identifies the name of the milestone. This property is required.
<b>Display ID</b>	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.
<b>Description</b>	Specifies user-defined information about the milestone.

Property Name	Description
<b>Milestone Phase</b>	<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><b>Develop</b> 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><b>Test</b> 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><b>Staging</b> 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><b>Production</b> 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><b>Retire</b> specifies that the milestone has retirement tasks such as removing a model from a production environment.</p> <p><b>User-defined</b> 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
<b>Name</b>	Identifies the name of the task. This property is required.
<b>Display ID</b>	Displays a system-supplied task identifier in the form <b>milestone#. task#</b> (for example <b>1.1</b> ) 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.
<b>Dependencies</b>	Identifies the display ID for a task that must be completed before this task can be completed.
<b>Description</b>	Displays user-defined information about the task.

Property Name	Description
<b>Task Type</b>	<p>Specifies a type for the task. Here is a list of valid task types:</p> <p><b>User-defined</b> 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><b>Sign-off</b> specifies that all of the milestone tasks are complete and have been approved.</p> <p><b>Declare Production</b> specifies that the champion model is ready to be exported to the production environment.</p> <p><b>Set Champion</b> 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><b>Retire Champion</b> specifies that the champion model is retired.</p>
<b>Assignees</b>	<p>Specifies a user or group name from the <b>Participants</b> 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 <b>Status</b> box to <b>Not Started</b>, <b>Started</b>, or <b>Completed</b>.</p> <p><b>Assignees</b> can be unassigned. If this box is unassigned, the following rules apply:</p> <ul style="list-style-type: none"> <li>• Updates to the task status are not required.</li> <li>• If any users or groups are assigned the role <b>Model Manager: Life Cycle Assignee Usage</b>, then only those users and groups can modify the task status.</li> <li>• If the role of <b>Model Manager: Life Cycle Assignee Usage</b> is not assigned to any user or group, then any SAS Model Manager user can modify the task status.</li> </ul>
<b>Approvers</b>	<p>Specifies a user or a group from the <b>Participants</b> 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 <b>Status</b> box to <b>Approved</b>.</p> <p><b>Approvers</b> can be unassigned. If this box is unassigned, the following rules apply:</p> <ul style="list-style-type: none"> <li>• The task approval status is not required to be updated.</li> <li>• If any users or groups are assigned the role <b>Model Manager: Life Cycle Approval Usage</b>, then only those users and groups can modify the task approval status.</li> <li>• If the role of <b>Model Manager: Life Cycle Approval Usage</b> is not assigned to any user or group, then any SAS Model Manager user can modify the task approval status.</li> </ul>

Property Name	Description
<b>Weight</b>	<p>Specifies a percentage either as an integer or a fractional number that indicates the relative work effort that is required by the task to complete the milestone. SAS Model Manager uses weight values to calculate the percentage that is complete for a milestone. The weight appears as a property for a version's <b>Life Cycle</b> folder. If you use the <b>Weight</b> property, the weight values for all tasks in a milestone should add up to 100. When weights for a milestone do not add up to 100, SAS Model Manager performs a weight proportion adjustment so that the sum of those weights within a milestone adds up to 100.</p> <p><i>Note:</i> User-defined weights are not explicitly adjusted. Weights remain as they were entered and are not adjusted.</p>
<b>Duration</b>	<p>Specifies a number that indicates the amount of time that is allocated to complete the task. The default duration unit is the number of days.</p>

## Create a Version

To create a new version:

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

Property	Value
<input type="checkbox"/> General Properties	
Name *	2013
Description	
<input type="checkbox"/> Version Properties	
Life Cycle Template	Extended

2. Specify a name and an optional description for the new version. Use a name that is unique among versions in this project. The name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ).
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 category view to view the contents of a template. If you plan to use SAS Model Manager Workflow Console to track activities (milestones and tasks), see [Chapter 22, “Using Workflow Console,” on page 345](#).

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 71](#)
- [“Creating Life Cycle Templates” on page 75](#)

---

## 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
<b>Default Scoring Task Input Table</b>	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 <b>Default Scoring Task Input Table</b> 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.
<b>Default Scoring Task Output Table</b>	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 <b>Default Scoring Task Output Table</b> 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.
<b>Default Performance Table</b>	<p>Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager version.</p> <p>The value of the <b>Default Performance Table</b> property is used as the default value for the <b>Performance data source</b> field in the Define Performance Task wizard if a default performance table is not specified for the model.</p>
<b>Default Train Table</b>	<p>The train table is optional and is used for information as well as the Training Data Set Summary report. When a value is specified for a model's <b>Default Train Table</b> property, SAS Model Manager does the following:</p> <ul style="list-style-type: none"> <li>• uses default train table to validate scoring functions or scoring model files when a user publishes the associated project champion model or challenger models to a database.</li> <li>• checks the <b>Validate scoring results</b> box in the Publish Scoring Function window.</li> </ul> <p>The value of the <b>Default Train Table</b> property is used to validate scoring functions or scoring model files only if a default train table is not specified for the model.</p>
<b>State</b>	Specifies the current status of the version.

Property Name	Description
<b>Champion Model Name</b>	<p>If a champion model has been set for a project, specifies one of the following:</p> <ul style="list-style-type: none"> <li>the name of the present champion model for the project</li> <li>the name of the model that was last set as the champion model for the project, when the champion model has been retired or cleared</li> </ul>
<b>Date Frozen</b>	Specifies the date on which the version was frozen.
<b>Production Date</b>	Specifies the date on which the status of the <b>Production</b> milestone task in the version's life cycle was changed from <b>Started</b> to <b>Complete</b> .
<b>Date Retired</b>	Specifies the date on which the status of the <b>Retire</b> milestone task in the version's life cycle was changed from <b>Started</b> to <b>Complete</b> .

---

## Working with Life Cycles

### Overview of a Life Cycle

#### **What Is a Life Cycle?**

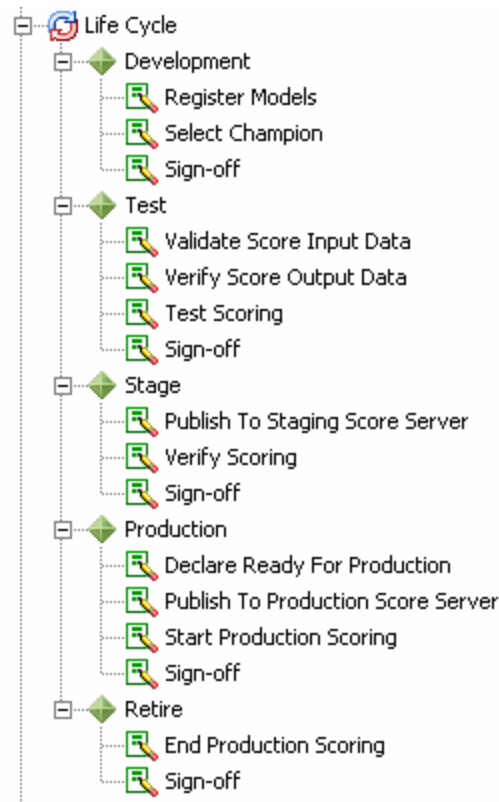
A SAS Model Manager life cycle defines the milestones and tasks that your organization uses to monitor the progress of a modeling project. The life cycle template controls the milestone tasks and the sequence of activities that are required to implement and deploy scoring models. SAS Model Manager provides 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 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 category view. For more information, see [“View Life Cycle Templates” on page 94](#).
- Update the status for a milestone. For more information, see [“Update Milestone Status” on page 95](#).
- Search for a task. For more information, see [“Search Life Cycles for Tasks Assigned to Users” on page 398](#).

For information about users and groups who can update the **Status** property of a task, see [“Participant Roles” on page 78](#).

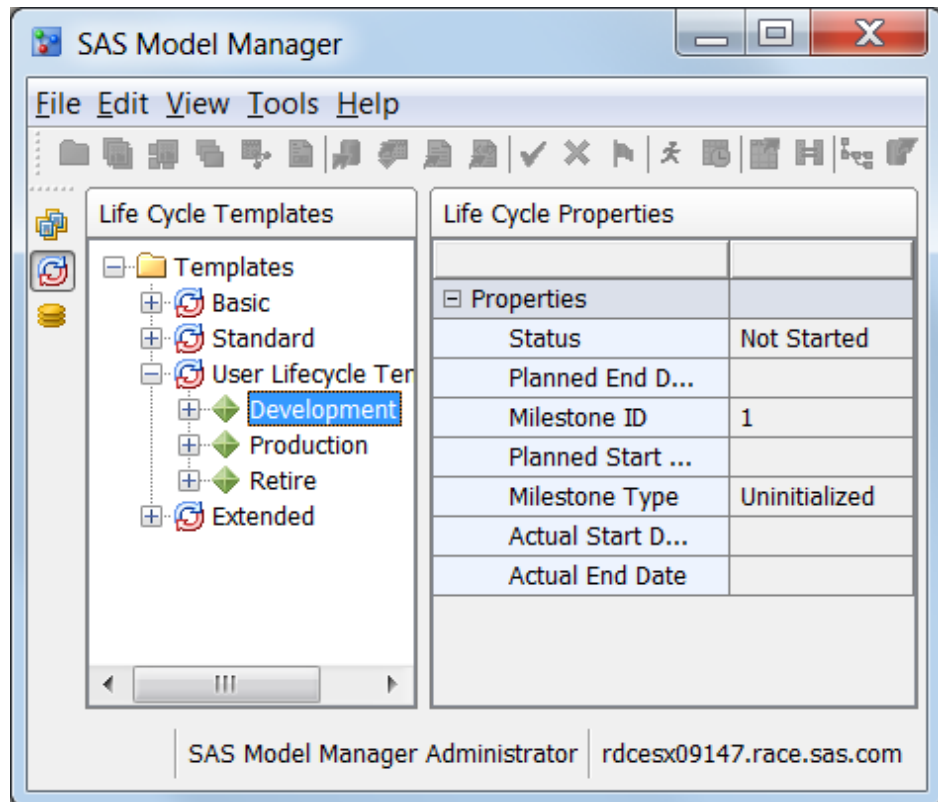
### **View Life Cycle Templates**

To view the life cycle templates that are available in SAS Model Manager:

1. Click the Life Cycles category view button  in the SAS Model Manager window.
2. Expand the **Templates** folder to display 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**
  - **User Lifecycle Template**
  - **Extended**

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 78](#).

### **Update Milestone Status**

To modify the status for a life cycle milestone:

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** contains 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 boxes that are white. When you click in a box, 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
<b>Actual Start Date</b>	Specifies the actual date that the first task for the milestone is started. This property is Read-only.
<b>Actual End Date</b>	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 to <b>Completed</b> .
<b>Planned Start Date</b>	Specifies the expected date to start the first task for milestone.

Property Name	Description
<b>Planned End Date</b>	Specifies the expected date to complete all tasks for the milestone.

Here is a list of task properties:

Property Name	Description
<b>Status</b>	Specifies the status of task. Possible values are <b>Not Started</b> , <b>Started</b> , <b>Completed</b> , or <b>Approved</b> .
<b>Date Completed</b>	Specifies the date on which 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 <b>Completed</b> .
<b>Completed By</b>	Specifies the name of the user who completed the task. This property is Read-only.
<b>Date Approved</b>	Specifies the date on which completion of the task is approved. This property is Read-only.
<b>Approved By</b>	Specifies the name of the user who approved completion of the task. This property is Read-only.
<b>Planned Completion Date</b>	Specifies the expected date to complete the task.
<b>To Be Completed By</b>	Specifies the user who is responsible for completing the task.
<b>To Be Approved By</b>	Specifies the user who can approve that the task is completed.



## Chapter 7

# Working with Project Control Groups

---

<b>Overview of Project Control Groups</b> .....	<b>99</b>
<b>Planning a Project Control Group</b> .....	<b>100</b>
<b>Prerequisites for Creating Project Control Groups</b> .....	<b>101</b>
<b>Creating a Project Control Table</b> .....	<b>103</b>
<b>Create a Project Control Group</b> .....	<b>103</b>
<b>Create Projects from a Control Table</b> .....	<b>104</b>
<b>Add a New Version</b> .....	<b>113</b>
<b>Add an Input Variable</b> .....	<b>114</b>
<b>Publish Project Champion Models from a Project Control Group</b> .....	<b>115</b>
<b>Monitor Performance of Project Champion Models</b> .....	<b>117</b>

---

## Overview of Project Control Groups

SAS Model Manager enables you to create a project control group in the model repository. From a project control group level, you can create multiple projects from a control table, and then add new versions or new input variables to all projects within the project control group. After you set the champion model for each project, you can monitor the performance of the champion models for all projects, and publish the champion models to the SAS Metadata Repository.

Here are the tasks that can be performed for a project control group:

- [“Create a Project Control Group” on page 103](#)
- [“Create Projects from a Control Table” on page 104](#)
- [“Add a New Version” on page 113](#)
- [“Add an Input Variable” on page 114](#)
- [“Publish Project Champion Models from a Project Control Group” on page 115](#)
- [“Monitor Performance of Project Champion Models” on page 117](#)

---

## Planning a Project Control Group

Before you begin a project control group, you must plan your project control group resources. Here are questions to consider and conditions to meet for a modeling projects within a project control group:

- After you know which users are assigned to the projects within a project control group, 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 20](#) and the *SAS Model Manager: Administrator's Guide*.
- How do you want to structure the projects within the project control group in the Project Tree? A project control group 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 47](#).
- What models do you want to use in each project of the control group? If the models were created using SAS Enterprise Miner, SAS/STAT, or the SAS/ETS procedures COUNTREG and SEVERITY, all model components are available to SAS Model Manager when you import the model. Only models that are contained in an SPK file can be imported using the **Create Projects from a Control Table** feature. At least one SPK file must be prepared for each project and the SPK files should be placed in the same location. If your model is a SAS code model or a PMML model that is not contained in an SPK file, you must import it separately into the desired project within the project control group, and 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 130](#) and [“Import PMML Models” on page 143](#).
- What model function do you want to use in each project of the control group?  
SAS Model Manager has several model function types:
  - Analytical
  - Prediction
  - Classification
  - Segmentation
  - Any

After the model function is specified for the project control group, the **Model Function** property for a project cannot be changed. Ensure that the types of models that you are going to use in each project of the project control group fit within the selected model function type. For more information, see [Table 5.1 on page 70](#).

- How do you want to define your project input and output variables? When you create a project control group, you can import the variables using input and output prototype tables, copy the variables from an existing champion model, or define individual variables. The project variables are set for each project within the project control group. 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, or you must create a libref for files that are stored on a local or network drive before you create the project control group. For more information, see [“About Defining Project Input and Output Variables” on page 61](#).

- What method do you want to use to track the progress of a version? The Workflow Console enables you to track the progress of activities from the version level for each individual project within a control group. A SAS Model Manager administrator can create a workflow and associate it with a version. You can also use the life cycle feature to track the life cycle of models at the version level.
  - If you decide to use the workflow process to track the progress of activities for a version, you do not need to use the life cycle feature to monitor the progress of milestones and tasks. However, you must select a life cycle when creating a project control group, but you do not need to use the life cycle feature. You can associate a workflow at the version level for each project with the control group. For more information, see [“Overview of Workflow Console” on page 346](#).
  - If you decide to use the life cycle feature to monitor the progress of your version, you must plan your milestones and the tasks for each milestone before you can create a version for each individual project within a project control group. When you have that information, you then create a life cycle template. The life cycle template enables you to assign users at the version level to complete projects and to monitor the progress of your project. For more information, see [“Creating Life Cycle Templates” on page 75](#).
- When you publish project champion models from a project control group to the SAS Metadata Repository, you must specify a folder to which you can publish the project champion models. You might need to create a folder in the SAS Metadata Repository, if one does not already exist. For more information, see [“Publish Models to the SAS Metadata Repository” on page 228](#).
- After your project champion models are in a production environment, you can monitor the performance of the project champion models within a project control group in SAS Model Manager using your organization's operational data. If you use SAS Model Manager to monitor performance of projects within a project control group, you must first prepare performance tables using the operational data and then register the tables in the SAS Metadata Repository using SAS Management console or create a libref for files that are stored on a local or network drive. For more information, see [“Creating a Performance Table” on page 40](#).
- 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 259](#).

---

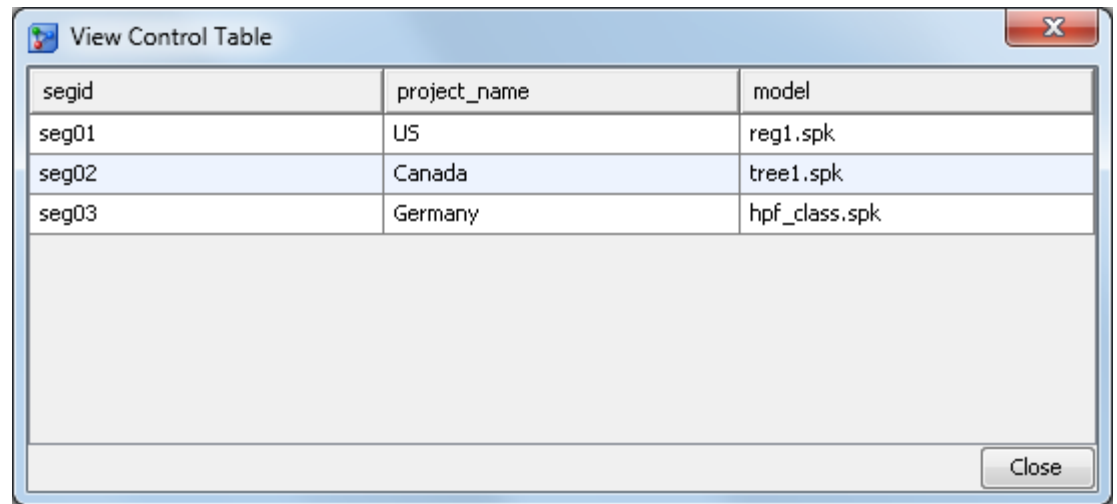
## Prerequisites for Creating Project Control Groups

After you have planned the projects and models that you want to have in your project control group, you must create a project control table that contains the segment identifiers, projects, and models. The project control table can then be used by the Create Projects from a Control Table feature to create a hierarchy of your project control group.

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

The project control table must contain the project names (project\_name variable) to create the projects within the control group. At least one segment identifier variable (for example, segid) is required, and that segment identifier variable must also be in the performance data set. When you want to monitor the performance of project champion models, you must also associate the model name (model variable) with each project (project\_name) and segment identifier (segid, or another name for the segments) in the table.

This is an example of a control table that can be viewed by using the Create Projects from a Control Table feature:



segid	project_name	model
seg01	US	reg1.spk
seg02	Canada	tree1.spk
seg03	Germany	hpf_class.spk

You must know the model function type before you create a project control group.

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 [Table 5.1 on page 70](#).

If you use prototype tables to define the project input and output variables, you must do one of the following two things before you can create a project control group. Create the project input and output tables and register them in the SAS Metadata Repository using SAS Management Console. Create a libref for files on a local SAS Workspace Server or network drive. You then can view the data tables from the Data Sources category view in SAS Model Manager. See the following documents for details:

- For instructions about creating project input and output tables, see [“Creating Project Input and Output Tables” on page 37](#).
- *SAS Model Manager: Administrator's Guide* has instructions on registering project input and output tables in SAS Management Console.

---

## Creating a Project Control Table

After you have planned the projects and models that you want to have in your project control group, you must create a project control table that contains the segment identifiers, projects, and models. The project control table is then used by the Create Projects from a Control Table feature to create the hierarchy of your project control group. The variable names that are required in the project control table are at least one segment identifier (for example, `segid`), `project_name`, and `model`. All variables other than `project_name` and `model` are treated as segment identifier variables. The segment identifier variables do not have a required naming convention.

Here is an example of the code to create a project control table.

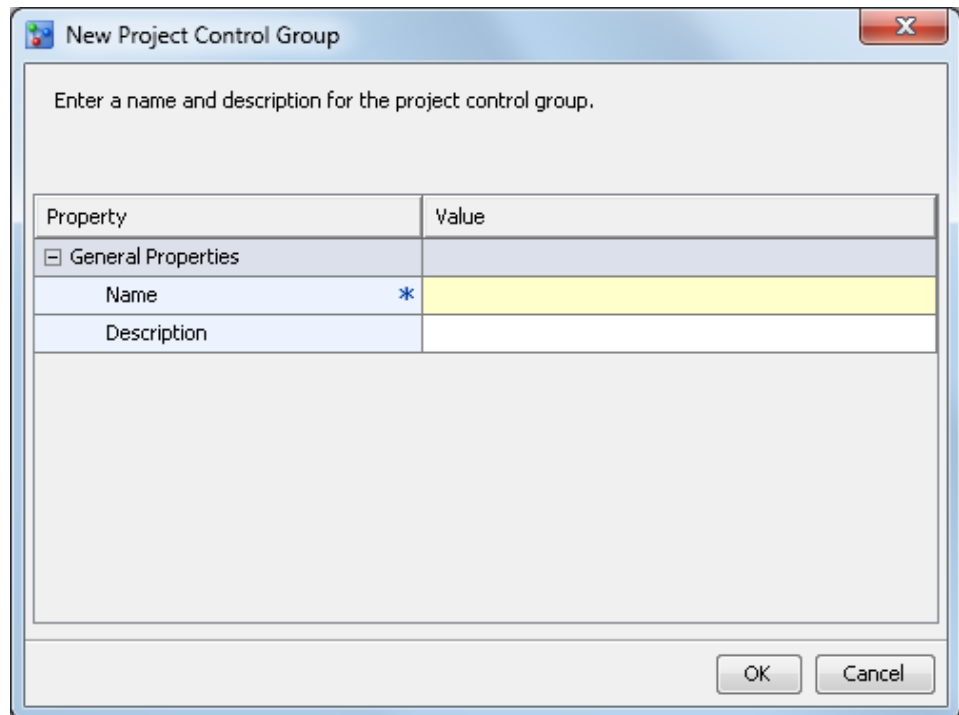
```
data control_Table;
  length segid project_name model $20;
  infile datalines dsd dlm=' ' missover;
  input segid project_name model;
  datalines;
  seg01,US,reg1.spk
  seg02,Canada,tree1.spk
  seg03,Germany,hpf_class.spk
  ;
run;
```

---

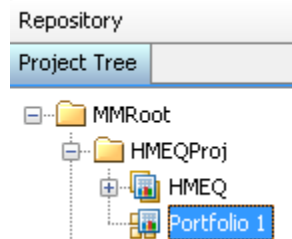
## Create a Project Control Group

To create a project control group:

1. Right-click an organizational folder and select **New** ⇒ **Project Control Group**. The New Project Control Group window appears.



2. Enter a name for the project control group.
3. (Optional) Enter a description for the project control group.
4. Click **OK**. The project control group is created in the Project Tree.

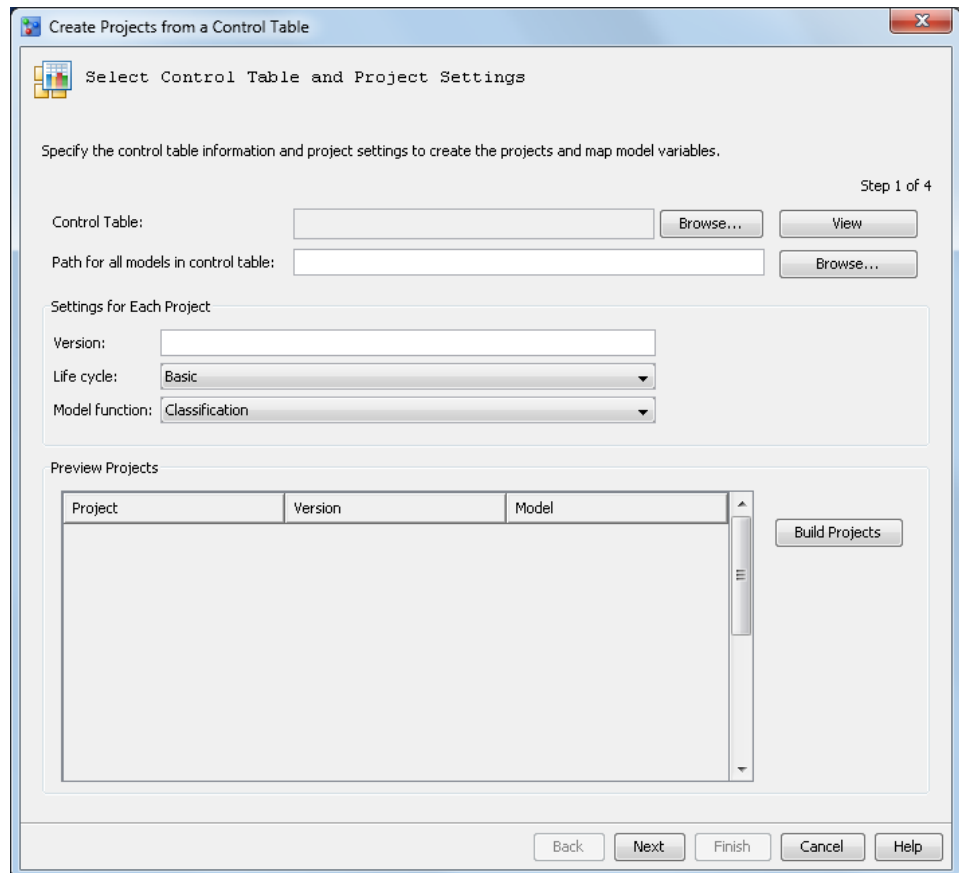



---

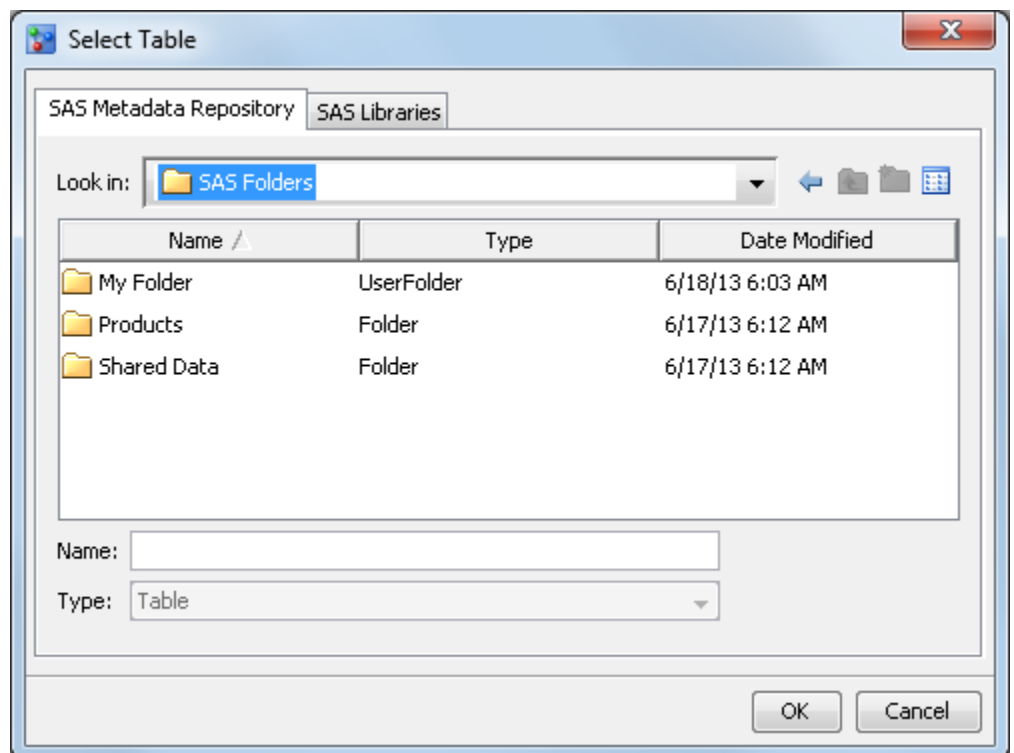
## Create Projects from a Control Table

To create projects from a control table:

1. Verify that the project control table contains the required variables. For more information, see [“Prerequisites for Creating Project Control Groups” on page 101](#).
2. Right-click a project control group in the Project Tree and select **Create Projects from a Control Table**. The Create Projects from a Control Table wizard appears.



3. Click **Browse** to select the control table from the SAS Metadata Repository or from local SAS libraries.

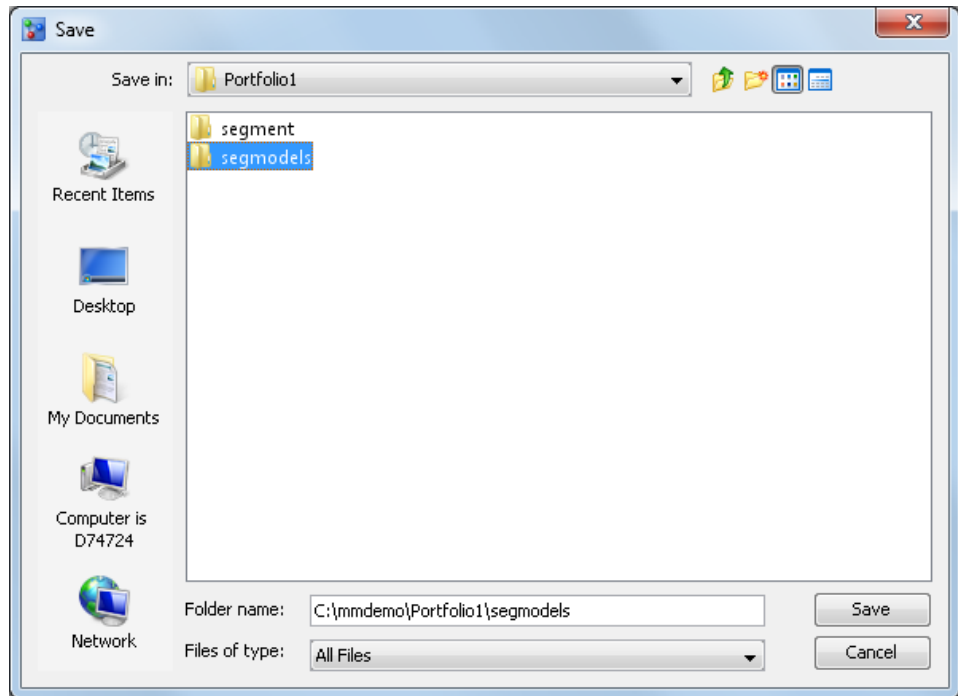


Click **View** to view the columns and values in the control table.

segid	project_name	model
seg01	US	reg1.spk
seg02	Canada	tree1.spk
seg03	Germany	hpf_class.spk

For information about the required columns in a control table, see “Creating a Project Control Table” on page 103.

4. Click **Browse** to select the location of the model SPK files that are in the control table.



Click **Save**.

5. Specify the version, life cycle, and model function settings that are to be applied to all projects in the control table when they are added to the project control group.
6. Click **Build Projects** to add the projects, versions, and models to the **Preview Projects** list.

Step 1 of 4

Control Table: Portfolio1.CONTROL\_TABLE Browse... View

Path for all models in control table: C:\mmdemo\Portfolio1\segmodels Browse...

Settings for Each Project

Version: 2013

Life cycle: Basic

Model function: Classification

Preview Projects

Project	Version	Model
US	2013	reg1
Canada	2013	tree1
Germany	2013	hpf_class

Build Projects

Back Next Finish Cancel Help

7. Click **Next**. The **Set Project Variables** page appears.

Set Project Variables for "Portfolio1.CONTROL\_TABLE"

Specify the input and output variables to apply to all of the projects in the control group.

Step 2 of 4

Project Input Variables

Name	Type	Measurement	Length	Description

Import Variables Copy Variables Add Edit Delete

Project Output Variables

Name	Type	Measurement	Length	Description

Import Variables Copy Variables Add Edit Delete

Back Next Finish Cancel Help

8. Specify the input and output variables to apply to all of the projects in the control group using one of these methods:
  - Click **Import Variables** to import input variables or output variables. Select the **SAS Metadata Repository** tab or the **SAS Libraries** tab, select the table, and click **OK**.
  - Click **Copy Variables** to copy variables from another project.
  - Click **Add** to manually enter a new variable.



Create Projects from a Control Table

Set Project Variables for "Portfolio1.CONTROL\_TABLE"

Specify the input and output variables to apply to all of the projects in the control group. Step 2 of 4

Project Input Variables

Name	Type	Measurement	Length	Description
LOAN	N		8	
CLAGE	N		8	
DELINQ	N		8	
DEBTINC	N		8	
JOB	C		7	
NINQ	N		8	

Import Variables Copy Variables Add Edit Delete

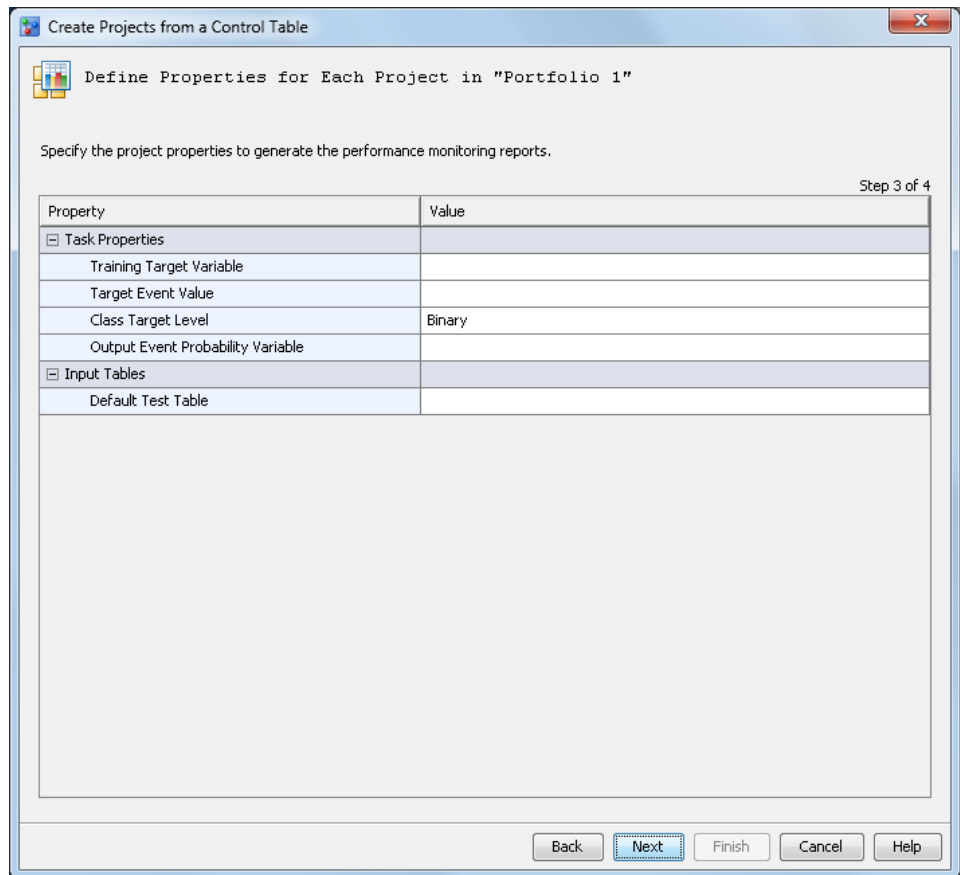
Project Output Variables

Name	Type	Measurement	Length	Description
score	N		8	

Import Variables Copy Variables Add Edit Delete

Back Next Finish Cancel Help

- Click **Next**. The **Define Properties for Each Project** page appears.



- Specify the task properties and input tables to apply to all projects within the control group.

Define Properties for Each Project in "Portfolio 1"

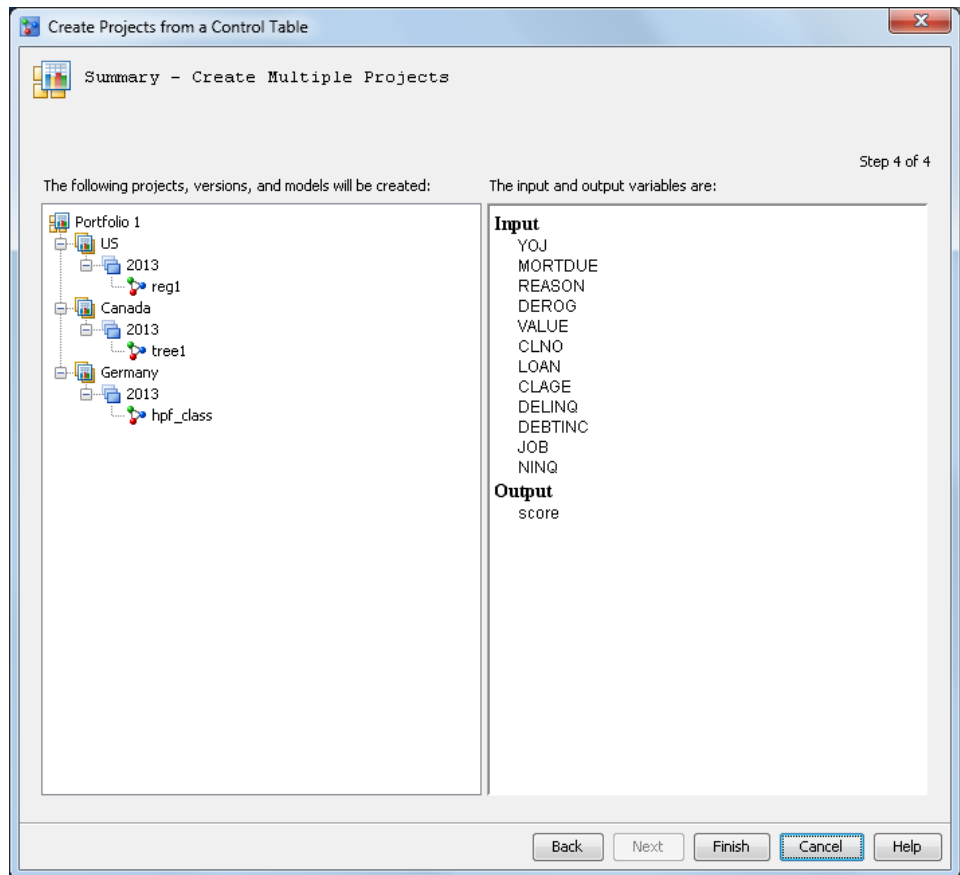
Specify the project properties to generate the performance monitoring reports.

Step 3 of 4

Property	Value
<input type="checkbox"/> Task Properties	
Training Target Variable	bad
Target Event Value	1
Class Target Level	Binary
Output Event Probability Variable	score
<input type="checkbox"/> Input Tables	
Default Test Table	MMLib.HMEQ_TEST

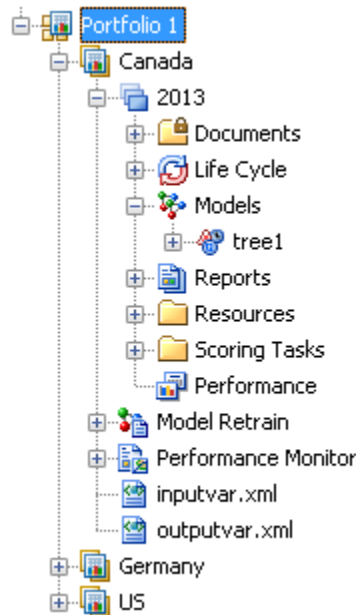
Back Next Finish Cancel Help

11. Click **Next** to view the summary of information that has been specified.



12. Click **Finish**. The projects are created with the version, life cycle, and model function that was specified. The models for each project in the control table are added to the **Models** folder within each version.

*Note:* The champion model for each project within the project control group must be set manually. For more information, see “[Set a Champion Model](#)” on page 217.



## Add a New Version

To add a new version to all projects within a project control group:

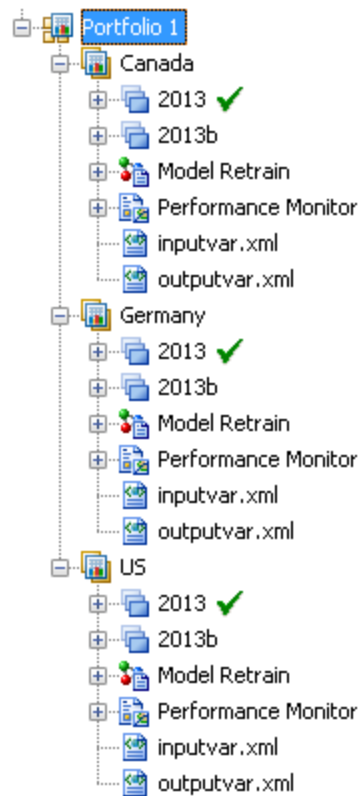
1. Right-click a project control group in the Project Tree and select **Add a New Version**. The Add a New Version window appears.

Property	Value
[-] General Properties	
Name *	
Description	
[-] Version Properties	
Life Cycle Template	Basic

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

*Note:* The life cycle template is not associated with a workflow. If you are using the Workflow Console to track the progress of workflow activities for the version in each individual project within the project control group, you can choose any of the life cycle templates.

- SAS Model Manager administrators and advanced users can use the SAS Model Manager Template Editor to customize life cycle templates. Use the Life Cycles category view to view the contents of a template.
  - SAS Model Manager administrators can use the Workflow Console to create a workflow and track the progress of activities for an individual version. A workflow cannot be created at the project control group level or the project level.
4. Review the selections and click **OK**.
  5. Click **Close** in the success message. The new version is added to each project within the control group.



6. Examine the properties of the version folder of each project within the project control group. 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.

For more information, see “[Overview of Versions](#)” on page 71.

---

## Add an Input Variable

To add an input variable to each project within a control table:

1. Right-click a project control group and select **Add a New Input Variable**. The Add a New Input Variable window appears.

Specify an input variable to add to each project in the control group.		
General Properties		
Name	*	
Description		
Type	*	N
Measurement		
Length	*	

2. Specify a name, type, and length for the input variable.  
*Note:* Only the values of character and number are valid for the variable type.
3. (Optional) Specify a description and measurement for the input variable.
4. Click **OK**.

---

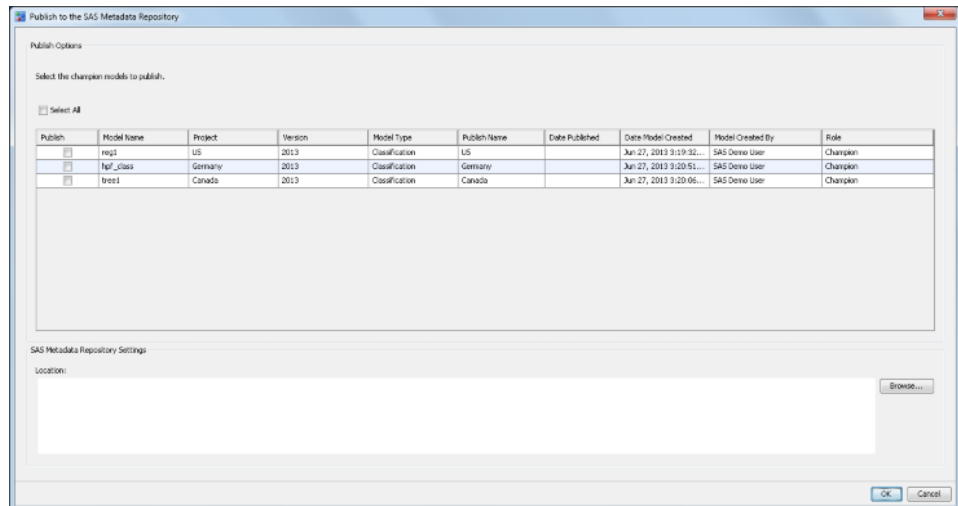
## Publish Project Champion Models from a Project Control Group

To publish the champion models for projects within a control group, you must have already set the models that you want to publish as project champion models. When a champion model is selected, the version that contains the model is automatically set as the default version for the project. SAS Model Manager examines the projects and always publishes the champion models. When the champion model for a project changes and you publish the model again to the same location, the scoring application automatically uses the latest score code.

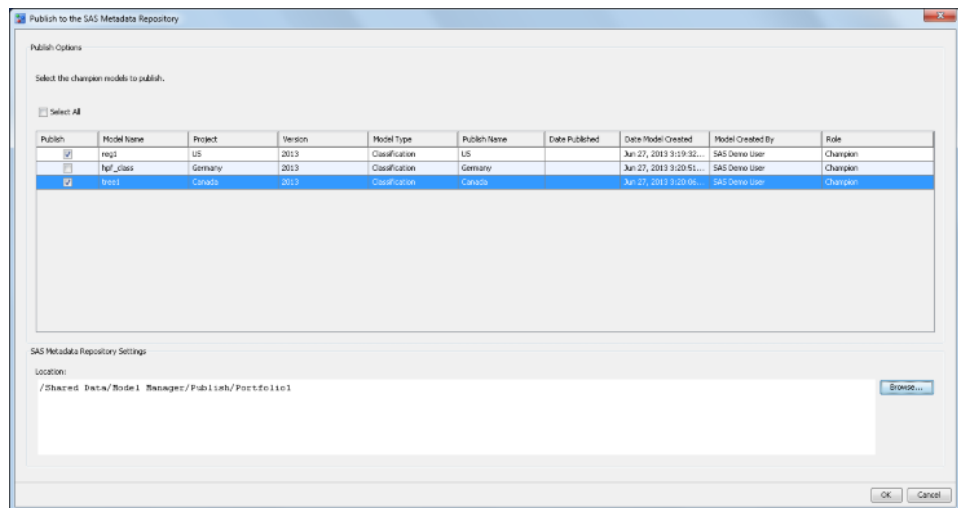
*Note:* SAS Model Manager cannot publish R models.

To publish champion models for projects in a control group:

1. Verify that a champion model has been assigned to all of the projects within a project control group that you want to publish. Select a project folder to examine its properties. The **Default Version** property contains the name of the default version. For more information, see [“Champion Models” on page 216](#).
2. Right-click the project control group and select **Publish**. The Publish to the SAS Metadata Repository window appears.

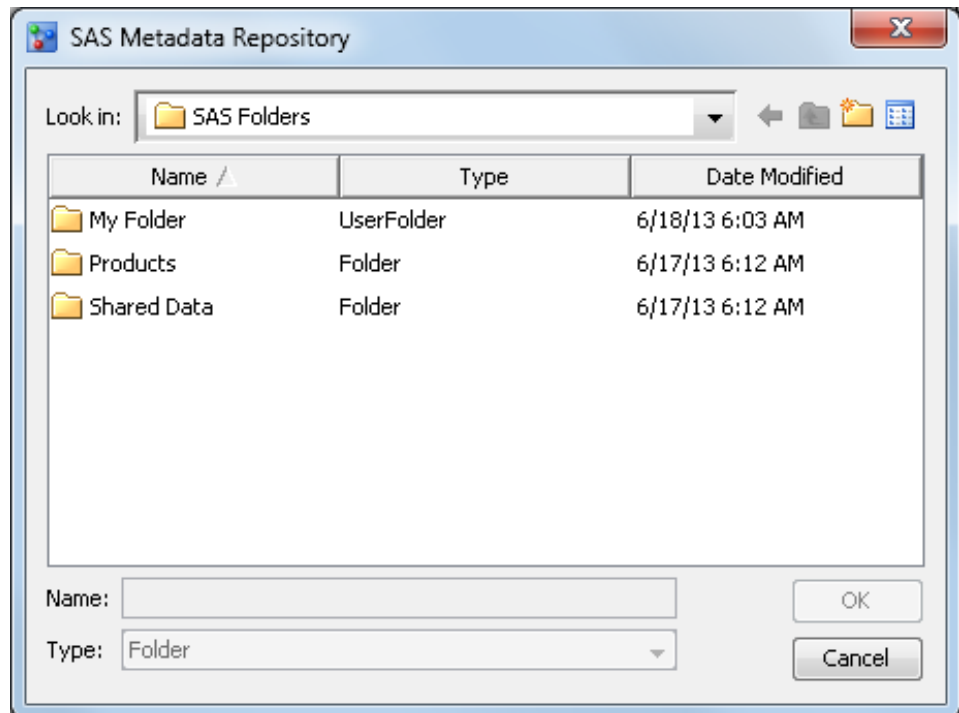


3. Select one or more champion models to publish.



4. Click **Browse** to select the location of the folder from the SAS Metadata Repository that you want to publish the selected champion models to.





*Note:* The selected champion models can be published only to a folder and you must have Write permission to the folder on the SAS Metadata Repository.

5. Click **OK**.

*Note:* If a MiningResults object is in the SAS Metadata Repository that has the same name or model UUID, then you are asked whether to overwrite the metadata for this stored object. Do not overwrite an existing MiningResults object unless you are certain that the model is from the same project in SAS Model Manager.

6. Click **Close** for the success message.

### See Also

- [“Verify the Model Publish” on page 230](#)
- [“Publish Models to the SAS Metadata Repository” on page 228](#)

---

## Monitor Performance of Project Champion Models

To create performance monitoring reports for all projects within a project control group, you run the Monitor Performance of Champion Models wizard to generate and execute the code. Execution of the generated code creates the SAS data sets that are used to display the performance monitoring reports from the version **Performance** node.

To monitor the performance of the champion models for all projects:

1. Right-click the project control group node in the project tree, and select **Monitor Performance**. The Monitor Performance of Champion Models wizard appears.

*Note:* The **Monitor Performance** menu item is available only to SAS Model Manager administrators and advanced users.

Monitor Performance of Champion Models

Performance Data Source Specifications for "Portfolio 1"

Step 1 of 5

Performance Data Options

Performance data source:  Browse...

Run model score code

Collection date:

Report label:

Champion Models to Monitor

Model	Version	Project	Date Published
reg1	2013	US	Jun 27, 2013
tree1	2013	Canada	Jun 27, 2013
hpf_class	2013	Germany	

Back Next Finish Cancel Help

2. Specify the performance data options:

- Click **Browse** to select the performance data source.

*Note:* The performance data source must contain the same segment identifier variables as the control table.

- To run the scoring task code in the performance monitor job, select the **Run model score code** check box. If the check box is not selected, all of the output variables for stability analysis must be in the performance data source.
- Click to select a collection date to associate with the performance data. The date can be any date in the time period when the performance data was collected.
- Enter a report label to associate with the performance data. The report label represents the time point of the performance data source. Because the report label appears in the performance charts, use a label that has not been used for another time period, is short, and is understandable (for example, 2013Q1 or 2013).

Click **Next**. The Input and Output Variables page appears.

Monitor Performance of Champion Models

Input and Output Variables for "Portfolio 1"

Select the input and output variables to use for analysis of the champion models in this project control group. Step 2 of 5

Output Variables for Stability Analysis

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

Select All Clear All

Input Variables for Characteristic Analysis

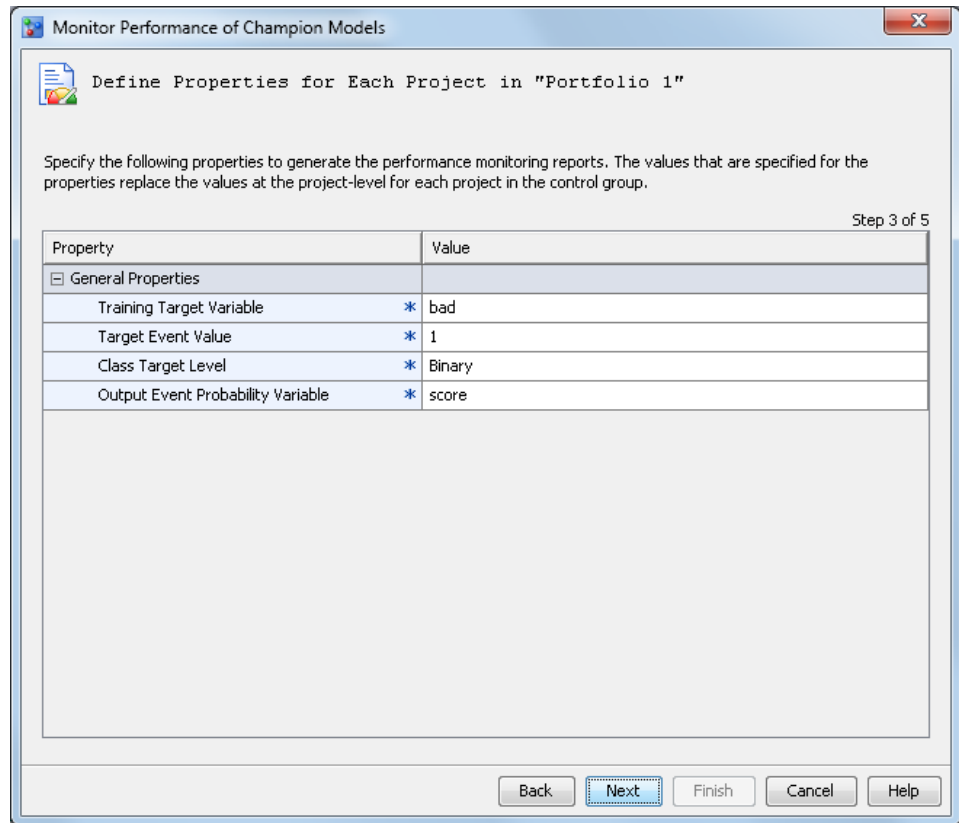
Select	Keep Variables	Description
<input type="checkbox"/>	YOJ	
<input type="checkbox"/>	MORTDUE	
<input type="checkbox"/>	REASON	
<input type="checkbox"/>	DEROG	
<input type="checkbox"/>	VALUE	
<input type="checkbox"/>	CLNO	
<input type="checkbox"/>	LOAN	

Select All Clear All

Back Next Finish Cancel Help

- In the **Output Variables for Stability Analysis** table, select one or more output variables. To select all output variables, click **Select All**.
- In the **Input Variables for Characteristic Analysis** table, select one or more input variables. To select all input variables, click **Select All**.

Click **Next**. The Define Properties for Each Project page appears.



5. Specify the properties that are required to generate the performance monitoring reports.

*Note:* The values that are specified for the properties replace the values at the project level for each project in the control group.

Click **Next**. The Warnings, Alerts, and E-mail Notifications page appears.

Monitor Performance of Champion Models

Warnings, Alerts, and E-mail Notifications for "Portfolio 1"

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

Step 4 of 5

Project-level Settings

Override the values of the project-level settings

Property	Value
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 condition	* (lift5Decay>0.15 and lift10Decay>0.12) or giniDecay>0.1 or ksD...
Model assessment warning condition	* lift5Decay>0.05

Specify when to receive e-mail notifications

E-mail Notifications

E-mail Address	Send Alert Warning	Send Job Status

Add Delete

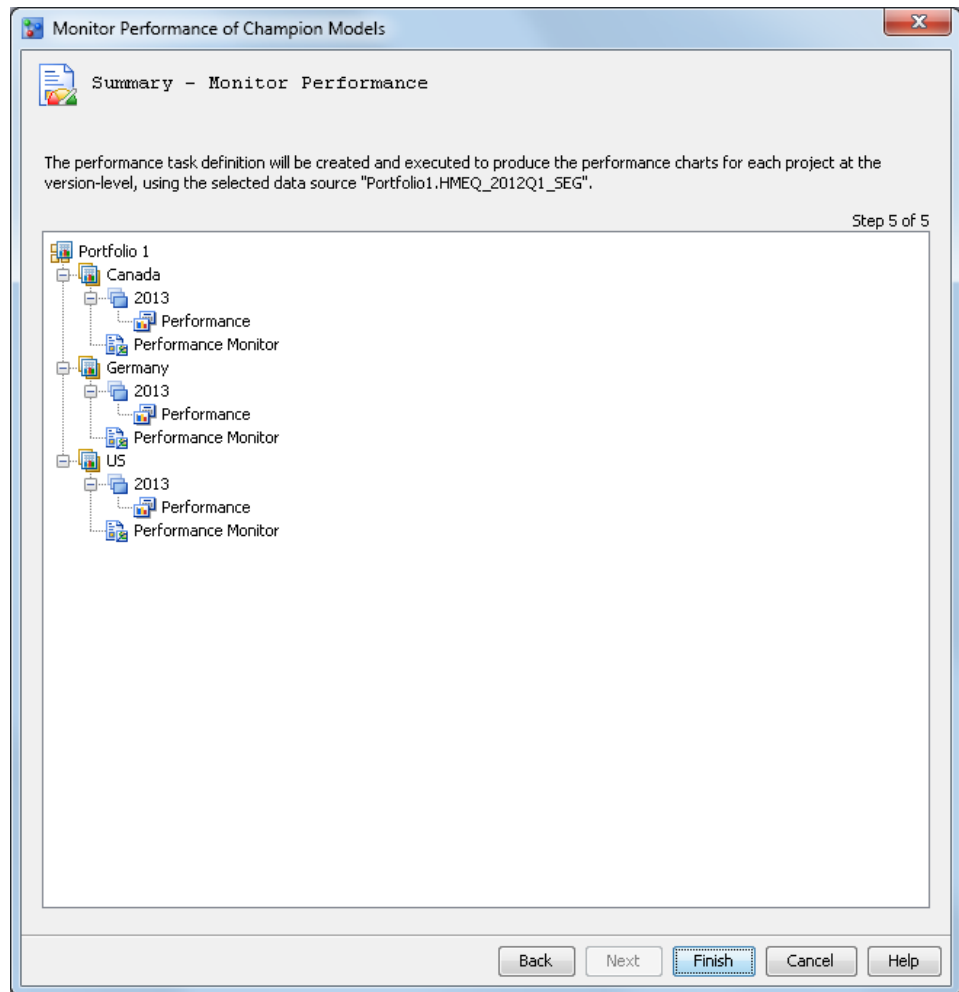
Back Next Finish Cancel Help

6. Verify or set the condition values for Characteristic Analysis, and specify when and who should receive e-mail notifications.

(Optional) To send the scoring results by e-mail, click **Add** in the **E-mail Notifications** table. The Add Contact window appears.

- a. Enter an e-mail address.
- b. Select either **Yes** or **No** if you want an alert warning 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 with the job status to be sent by e-mail every time the report runs.

Click **Next**. The summary of the monitor performance task that has been created is displayed.



7. Click **Finish**. The **running background process** status bar appears. The generated SAS code executes automatically. When it is finished, a status message for the monitoring results appears. You can view the generated code in the project **Performance Monitor** folder for each project within the control group.

To execute the generated code again, right-click the **Performance Monitor** node in the project control group folder, and select **Execute**. The performance task is executed as a background process. SAS Model Manager saves the data sets that create the monitoring reports in the version **Resources** folder of each project.

*Note:* The **Execute** pop-up menu item is available only to SAS Model Manager administrators and advanced users.

8. (Optional) Repeat steps 1–7 to monitor performance of the champion models for multiple performance data sources.
9. To view Model Monitoring Performance reports, select the **Performance** node at the default version level for each project within the control group. The details section displays a tab for each report. Select a tab to see a report.

## See Also

[“Prerequisites for Running the Define Performance Task Wizard” on page 266](#)

## Part 3

---

# Importing, Scoring, and Validating Models

<i>Chapter 8</i>	
<b>Importing Models</b> .....	125
<i>Chapter 9</i>	
<b>Scoring Models</b> .....	157
<i>Chapter 10</i>	
<b>Validating Models Using Reports</b> .....	179
<i>Chapter 11</i>	
<b>Validating Models Using User Reports</b> .....	199





## Chapter 8

# Importing Models

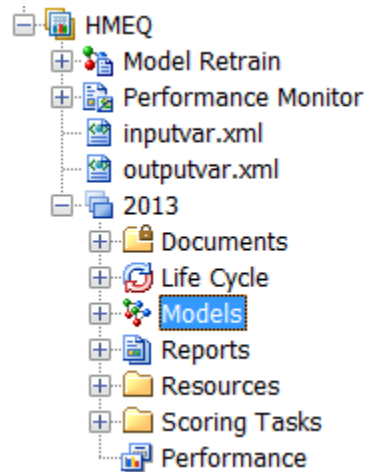
---

<b>Overview of Importing Models</b> .....	<b>125</b>
<b>Import Models from the SAS Metadata Repository</b> .....	<b>127</b>
<b>Import SAS Model Package Files</b> .....	<b>128</b>
What Is a SAS Model Package File? .....	128
Create SAS Package Files in SAS Enterprise Miner .....	128
About Creating SAS Package Files Using the %AA_Model_Register Macro ...	129
Import Package Files .....	129
<b>Import SAS Code Models and R Models Using Local Files</b> .....	<b>130</b>
Overview of Importing SAS Code Models .....	130
Model Templates .....	131
Model Template Component Files .....	133
Viewing Model Template Files .....	140
Importing a SAS Code Model .....	140
Importing an R Model .....	142
<b>Import PMML Models</b> .....	<b>143</b>
<b>Import Partial Models</b> .....	<b>144</b>
<b>Set Model Properties</b> .....	<b>145</b>
<b>Map Model Variables to Project Variables</b> .....	<b>146</b>
<b>User-Defined Model Templates</b> .....	<b>148</b>
Creating a New Model Template .....	148
Model Template Properties .....	152
<b>Specific Properties for a Model</b> .....	<b>154</b>

---

## Overview of Importing Models

After you create a project and version, the next step is to import models into SAS Model Manager. The **Models** folder is the container for all of the models under a 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 127
- “Import SAS Model Package Files” on page 128
- “Import SAS Code Models and R Models Using Local Files” on page 130
- “Import Partial Models” on page 144
- “Import PMML Models” on page 143

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 401 and [Appendix 4, “Macros for Registering Models to the SAS Metadata Repository ,”](#) on page 449.

*Note:*

- Scorecard models can be imported using the SAS Code Models local files method and the SAS Model Package File import method.
- HPFOREST procedure models can be imported using the SAS Metadata Repository import and the SAS Model Package File import. You cannot import PROC HPFOREST models using local files.
- High-Performance analytics models that are not created with SAS Enterprise Miner can be registered to the SAS Metadata Repository using the %AA\_Model\_Repository. These models can then be imported to SAS Model Manager by importing the models from the SAS Metadata Repository and from a SAS model package file..
- Before you import COUNTREG procedure and SEVERITY procedure models, create the model score code using the %MM\_Countreg\_Create\_Scorecode macro and the %MM\_Severity\_Create\_Scorecode macro. After the score code is generated, you can use the %MM\_Model\_Register macro or the local files method to import these models. For more information, see [Appendix 6, “Macros for Generating Score Code,”](#) on page 473.
- SAS Model Manager cannot publish models to a database whose **Score Code Type** model property is set to **SAS Program** or **PMML**.
- When you import models using the local file method, the table names that you specify as model components must start with a letter or underscore, can contain a period, and cannot be more than 32 characters long. Spaces or special characters (for example, ~!@#%&\*()+={}|\\:;’<>?/”) are not valid in a table name. For

more information about the types of model component tables, see “[Model Template Component Files](#)” on page 133.

- Model component table variable names must start with a letter or underscore, and cannot contain special characters (for example, `~!@#\$\$%^&\*()-+=[\{}|;:'"/<?>` or spaces. Variables with special characters can be used in SAS Model Manager only when the SAS Model Manager administrator has set the **Valid Variable Name** option to **Yes** in the SAS Management Console. For more information, see the *SAS Model Manager Administrator's Guide*.
- Unexpected results might occur if you import a model that was previously exported using SAS Model Manager. A best practice is to import models that were not previously exported by SAS Model Manager.

## Import Models from the SAS Metadata Repository

If your SAS Enterprise Miner 5.1 (or later) model files or your models that are created by the %AA\_Model\_Register macro are registered in your SAS Metadata Repository, you can import them into SAS Model Manager from the repository.

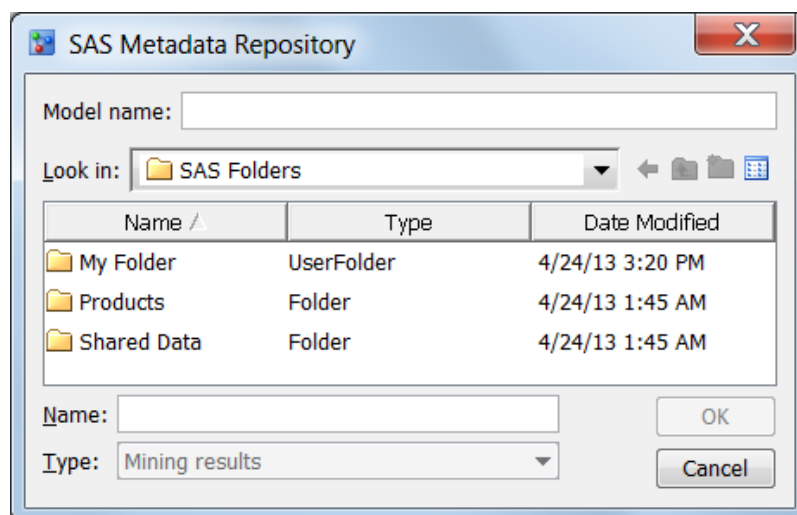
*Note:* Before you import a model into your project's version, verify that the model type matches the **Model Function** property setting on the project **Properties** tab or set the **Model Function** property to **Any**. For more information about model functions, see “[Specific Properties for a Project](#)” on page 505.

To import a 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 model.
4. In the **Model name** box, 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 box, 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 then view your newly imported model's data structures using the **Model Input**, **Model Output**, and **SAS Code** tabs on the right.

### See Also

- “Import SAS Model Package Files” on page 128
- “Import Partial Models” on page 144
- “Set Model Properties” on page 145
- “Map Model Variables to Project Variables” on page 146
- Appendix 4, “Macros for Registering Models to the SAS Metadata Repository ,” on page 449

---

## Import SAS Model Package Files

### *What Is a SAS Model Package File?*

A SAS model package (SPK) file is a SAS Enterprise Miner SPK file or an SPK file that was created by using the %AA\_Model\_Register macro. SPK files contain complete model information. They enable a user to import into SAS Model Manager a complete model that is not registered in a SAS Metadata Repository.

### *Create SAS Package Files in SAS Enterprise Miner*

To create an SPK file for an existing SAS Enterprise Miner model:

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 node 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.ta**, and save the file to your workstation folder.

## About Creating SAS Package Files Using the %AA\_Model\_Register Macro

These models can be created by SAS procedures and are supported by SAS Model Manager:

- SAS/STAT item store models
- High-performance models
- SAS/ETS COUNTREG procedure models
- SAS/ETS SEVERITY procedure models

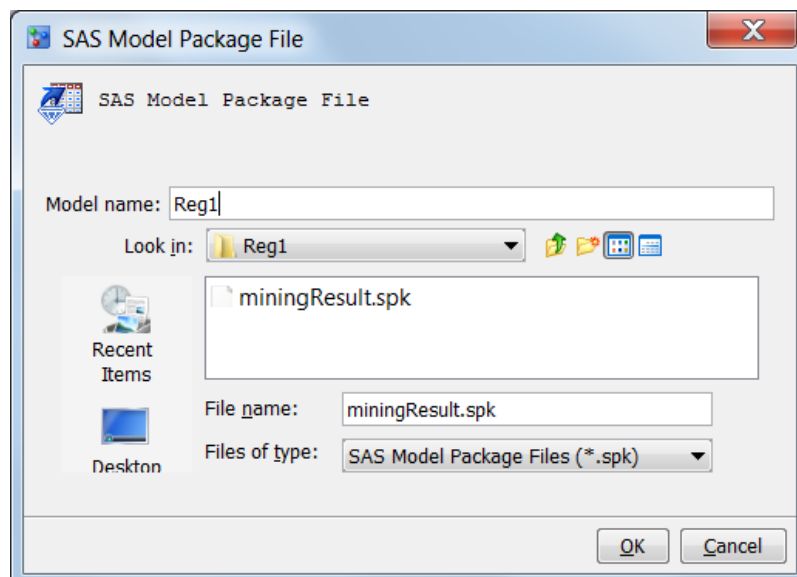
You can use the %AA\_Model\_Register macro to create an SPK file to contain these models. For more information, see [“Create a SAS Package File Using a SAS/STAT Item Store” on page 451](#).

## 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 505](#).

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 Model Package File**.



3. Navigate to the location of the SAS model package (SPK) file and select the file.
4. To change the name of the model, enter a text value in the **Model name** box. The value of the **Model name** box 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 144](#)
- [“Set Model Properties” on page 145](#)
- [“Map Model Variables to Project Variables” on page 146](#)

---

## Import SAS Code Models and R Models Using Local Files

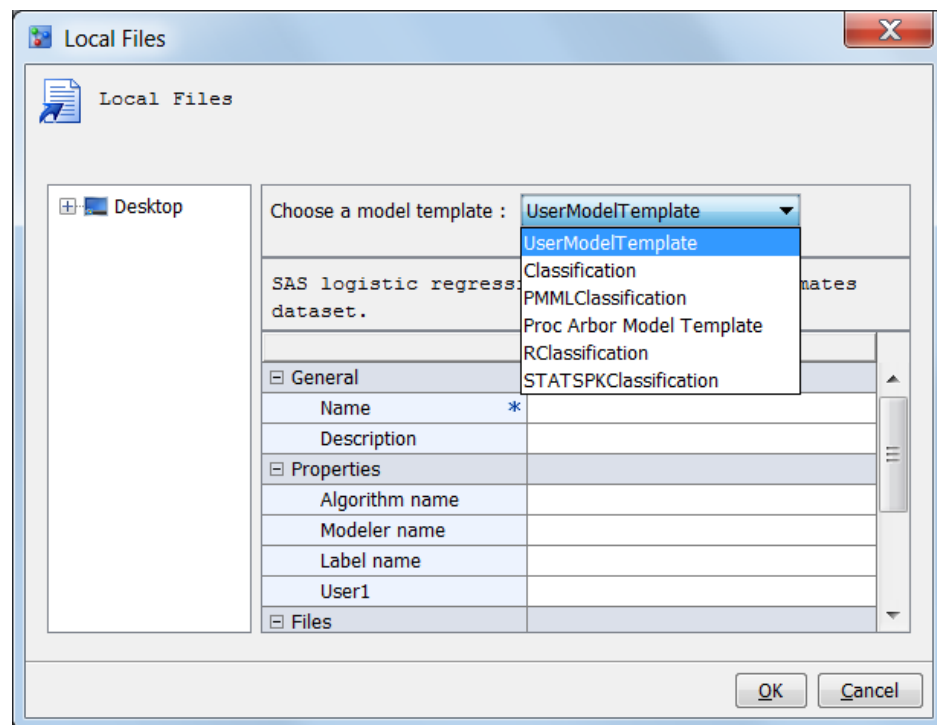
### Overview of Importing SAS Code Models

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 to import R models. An example of a model might be a SAS LOGISTIC procedure model, a SEVERITY model, or an R logistic model.

*Note:* HPFOREST models cannot be imported using local files.

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 133](#). 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 144](#).

After you have your model component files, you use the Local Files window to import the component files. To open the Local Files window, right-click the **Models** folder and select **Import from** ⇒ **Local 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 using either SAS data sets or XML files. 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, 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:

modelinput.sas7bdat

myModelInput.sas7bdat

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

For information about component files for the different model types, see “[Model Template Component Files](#)” on page 133.

### SAS Model Manager Model Templates

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

Model Type	Description
Analytical	The <b>Analytical</b> 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 <b>Classification</b> 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 <b>Prediction</b> model template is used for predictive models. Predictive models declare in advance the outcome of an interval target. A model that assigns a numeric credit score to an applicant is an example of a prediction model.
Segmentation	The <b>Segmentation</b> 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 <b>Any</b> 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 <b>Any</b> , SAS Model Manager lists the available model templates in the <b>Choose a model template</b> 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 144.



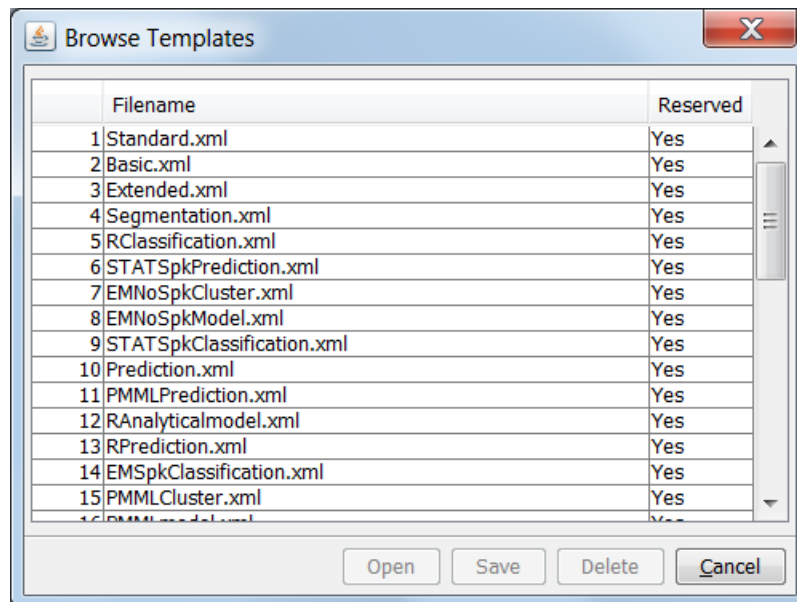
## User Model Templates

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 SAS 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 148](#).

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 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 on page 135	—	✓	—	—
EMPublishScore.sas on page 135	—	✓	—	—
Scorecard_GainsTable.csv on page 135	—	✓	—	—
score.sas on page 135	✓	✓	✓	✓
modelinput.sas7bdat on page 135	✓	✓	✓	✓
modeloutput.sas7bdat on page 136	✓	✓	✓	✓
target.sas7bdat on page 136	—	✓	✓	—
inputvar.xml on page 136	✓	✓	✓	✓
outputvar.xml on page 137	✓	✓	✓	✓
targetvar.xml on page 138	—	✓	✓	—
smmpostcode.sas on page 138	✓	✓	✓	✓
trainingvariables.csv on page 138	—	✓	—	—
training.sas on page 139	✓	✓	✓	✓
training.log on page 139	✓	✓	✓	✓
training.lst on page 139	✓	✓	✓	✓
outest.sas7bdat on page 139	✓	✓	✓	—
outmodel.sas7bdat on page 139	✓	✓	✓	—
output.spk on page 139	✓	✓	✓	✓
miningResult.spk on page 139	✓	✓	✓	—
layout.xml on page 139	—	✓	✓	—

Filename	Analytical	Classification	Prediction	Segmentation
<a href="#">format.sas7bcat on page 139</a>	✓	✓	✓	✓
<a href="#">dataprep.sas on page 139</a>	✓	✓	✓	✓
<a href="#">batch.sas on page 139</a>	✓	✓	✓	✓
<a href="#">pmml.xml on page 139</a>	✓	✓	✓	—
<a href="#">training.r on page 139</a>	✓	✓	✓	—
<a href="#">outmodel.rda on page 140</a>	✓	✓	✓	—
<a href="#">score.r on page 140</a>	✓	✓	✓	—
<a href="#">fitstats.xml on page 140</a>	—	✓	✓	—
<a href="#">HPDMForest_VARIMPO RT.csv on page 140</a>	—	✓	✓	—
<a href="#">HPDMForest_ITERATION.csv on page 140</a>	—	✓	✓	—
<a href="#">outmdlfile.bin on page 140</a>	—	✓	✓	—

#### **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.sas7bd` 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.sas7bd` 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.sas7bd` file.

*Note:* If the same variables appear in your `modelinput.sas7bd` file and your `modeloutput.sas7bd` file, when you import the model, SAS Model Manager removes the duplicate variables in the `outputvar.xml` file.

### **modeloutput.sas7bd**

The value of `modeloutput.sas7bd` 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.sas7bd` 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.sas7bd` is ignored.

You can create a `modeloutput.sas7bd` file by running the `score.sas` file against the `modelinput.sas7bd` file.

### **target.sas7bd**

The value of `target.sas7bd` 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.sas7bd` 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.sas7bd` is ignored.

You can create a `target.sas7bd` 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.sas7bd`, 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.

### **smmpostcode.sas**

SAS Model Manager creates this file to document the mapping that the user specified between the model variables and the project variables.

### **trainingvariables.csv**

This optional file contains a list of the training variables.

**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.

**miningresult.spk**

This is a SAS package file that stores detailed information about SAS Enterprise Miner nodes in the flow from which the model is created and the detailed information for SAS/STAT item store models.

**layout.xml**

This optional file contains information about the SAS Enterprise Miner diagram topology.

**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.

**batch.sas**

This file is created by SAS Enterprise Miner and is used for model retraining by SAS Model Manager.

**pmml.xml**

This file contains score code in PMML format.

**training.r**

This is an optional R script file that is used to retrain 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.

**fitstats.xml**

This file is created by SAS Enterprise Miner and contains the basic Fit Statistics for the model.

**HPDMForest\_VARIMPORT.csv**

This CSV file contains the variable importance data for a PROC HPFOREST model.

**HPDMForest\_ITERATION.csv**

This CSV file contains statistics across each iteration of a PROC HPFOREST model.

**OUTMDLFILE.bin**

This is a binary file that contains the PROC HPFOREST model information to be used for scoring.

For information about preparing R model component files, see [Appendix 8, “SAS Model Manager R Model Support,”](#) on page 519.

### 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.

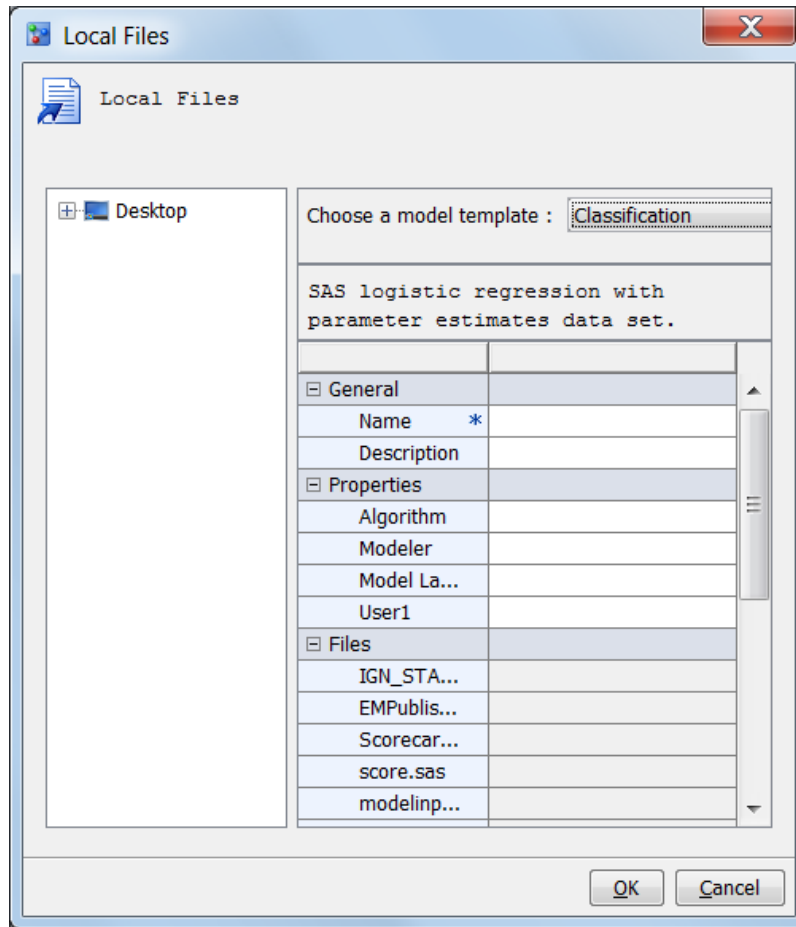
Reserved templates cannot be modified.

### Importing a SAS Code Model

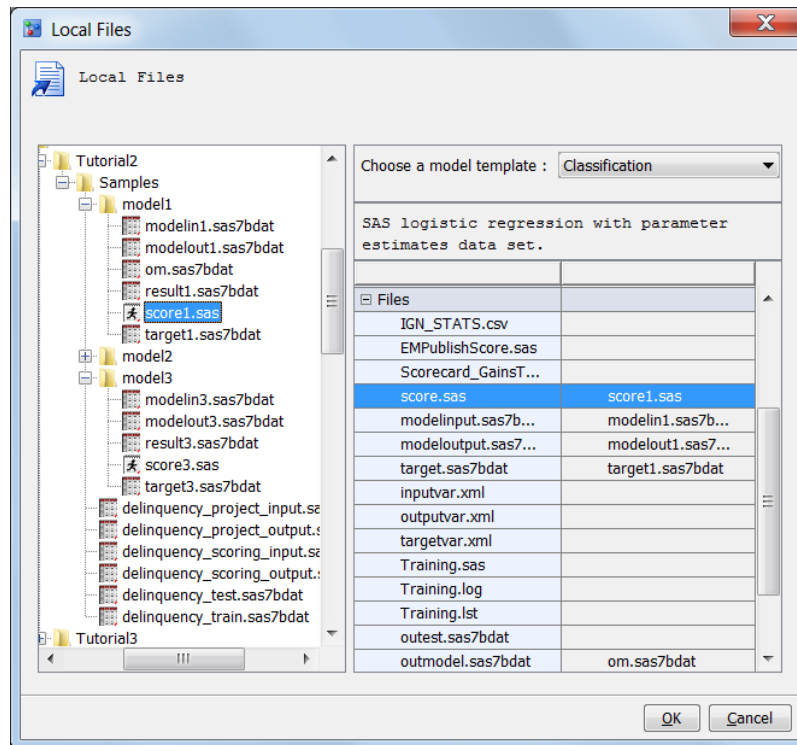
To import a SAS code model:

1. Copy your SAS code model and all of the associated metadata files to a location on your local workstation, or map the server that contains these model files to 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 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 131](#).
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.



*Note:* The filenames do not have to match, provided that the file contents meets the file property requirements. For more information, see the following topics:

- “Model Template Component Files” on page 133
- “Model Template Properties” on page 152

8. 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.

*Note:* If the same variables appear in your modelinput.sas7bdat file and your modeloutput.sas7bdat file, SAS Model Manager removes the duplicate variables in the outputvar.xml file when you import the model.

## Importing an R Model

For information about preparing R model files, see [Appendix 8, “SAS Model Manager R Model Support,”](#) on page 519.

To import an R model into SAS Model Manager:

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** ⇒ **Local Files**.
3. In the Local Files window, select the R model template in the **Choose a model template** box.
4. Enter **R model - type** in the **Name** field (for example, R Logistic).
5. Expand **Desktop** and navigate to the location of the R model files. Complete these required model properties:
  - a. Drag the model input table to the **modelinput.sas7bdat** property.

- b. Drag the model output table to the **modeloutput.sas7bdat** property.
  - c. Drag the target table to the **target.sas7bdat** property.
  - d. Drag the *model.rda* file to the **outmodel.rda** property.
  - 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

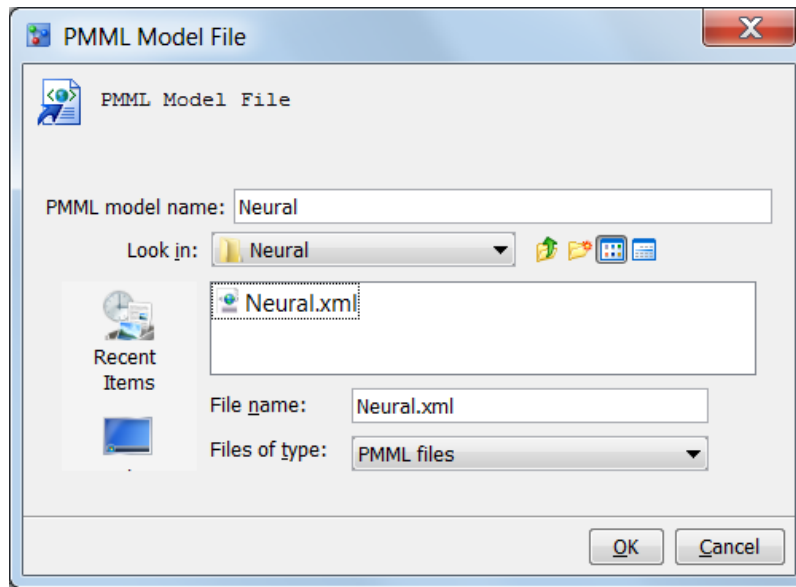
PMML (Predictive Modeling Markup Language) is an XML-based standard for representing data mining results. PMML is designed to enable the sharing and deployment of data mining results between vendor applications and across data management systems. The SAS Model Manager Import PMML Models feature enables users to import PMML models that are produced by using other applications. PMML 4.0 (or later) is supported by SAS Model Manager. Models that are created using PMML 4.0 support DATA step score code. For more information about PMML support in SAS Enterprise Miner, see the *SAS Enterprise Miner Help*.

*Note:* Before you import a PMML model, 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 505](#).

*Note:* SAS Model Manager does not support the importing of a PMML file that contains multiple models.

To import a PMML model 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** ⇒ **PMML Model File**. The PMML Model File window appears.



3. Navigate to the location of the PMML file and select the file.
4. Enter a text value in the **PMML model name** field. The name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ).
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.

SAS Model Manager generates the score code for the model that is based on the PMML file. The score.sas file is included with the model under the **Models** node.

*Note:* After you import a PMML 4.0 model, you can score the model using a scoring task and you can create a model comparison report for the model. You can score tree, regression, and neural PMML models. You cannot define a performance task if the PMML model is set as the champion model.

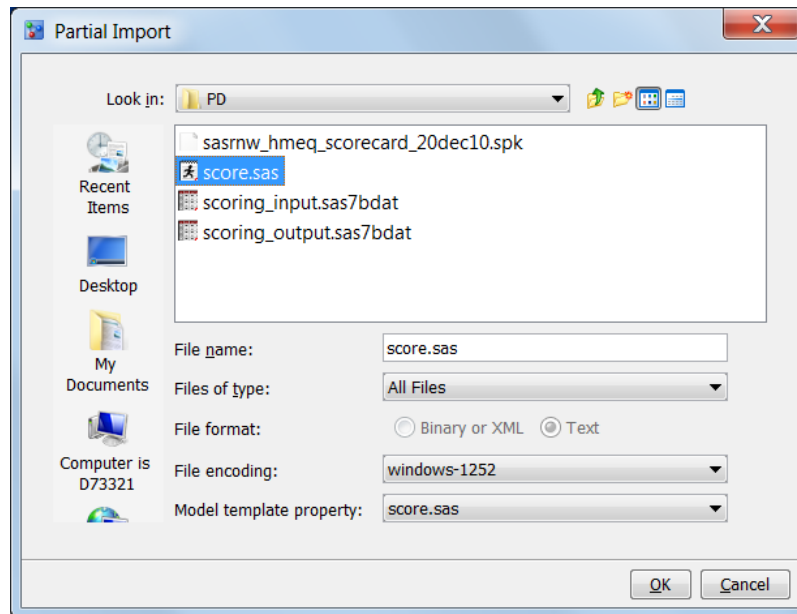
---

## Import Partial Models

Suppose 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 Import window appears.



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 an existing model component file that has 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 130
- [“Set Model Properties”](#) on page 145
- [“Map Model Variables to Project Variables”](#) on page 146

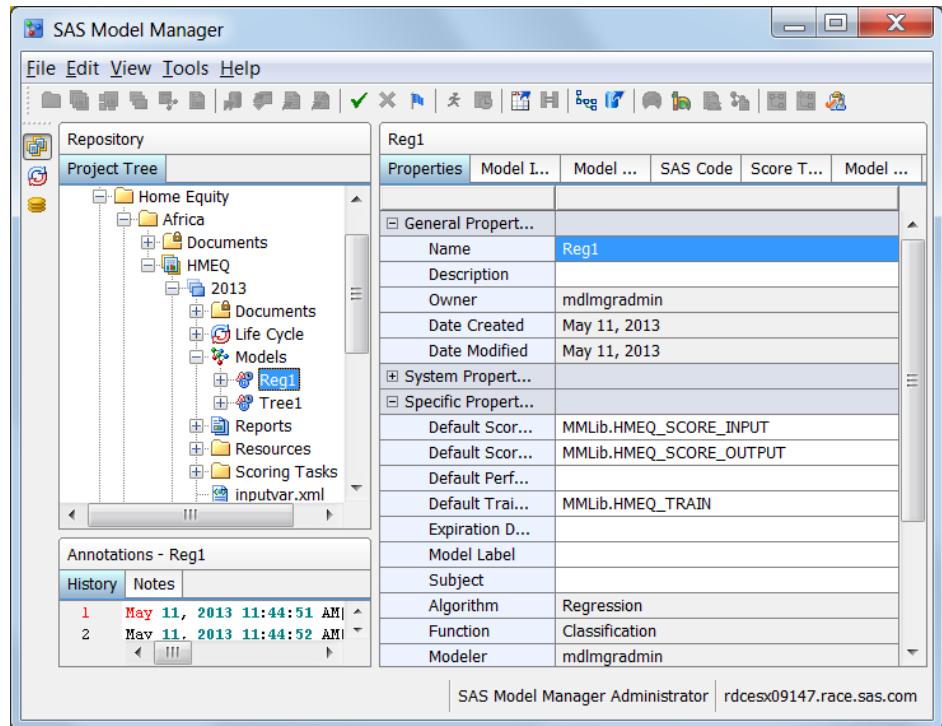
---

## 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.



- In the General Properties section, enter a model description for your model, if you did not do so when the model was imported. The only property that you can edit in the General Properties section is the **Description**. For more information, see “General Properties” on page 503.
- 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 504.
- 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 154.

### See Also

- “Import Partial Models” on page 144
- “Map Model Variables to Project Variables” on page 146

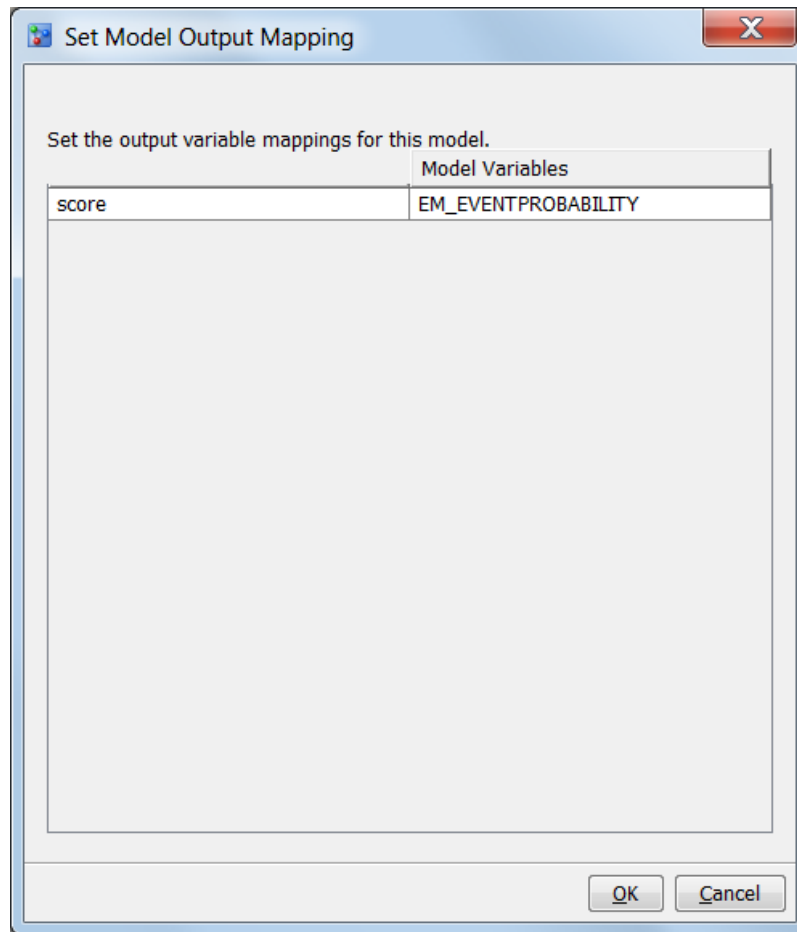
## Map Model Variables to Project Variables

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 33.

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

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

2. 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**.

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

### See Also

- [“Set Model Properties” on page 145](#)
- [“Overview of Importing Models” on page 125](#)

## 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 133. For a list of properties, see “[Model Template Properties](#)” on page 152.

*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.

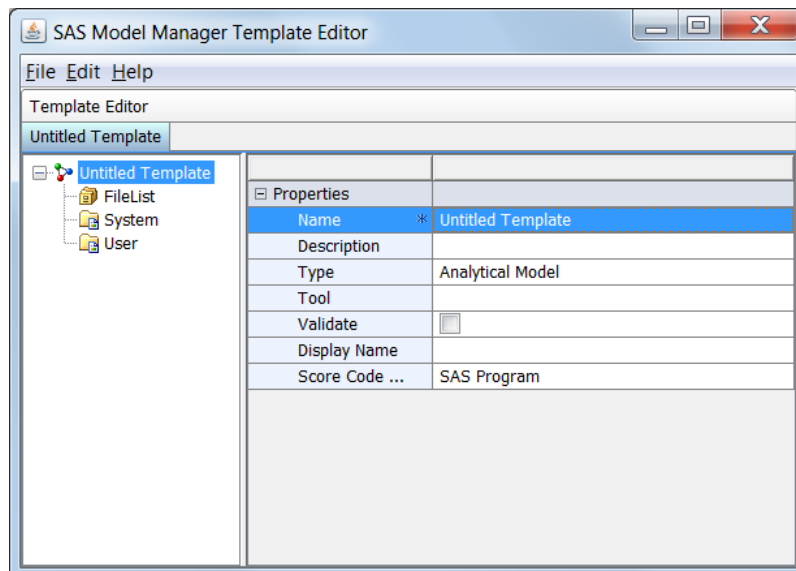
To open the Template Editor, select **Tools** ⇒ **Manage Templates**.

#### 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 150.

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** box is required. Replace **Untitled Template** with the template name. For more information, see [“Model Template Properties”](#) on page 152.
3. Create a list of component files for the model. For each new filename, right-click the **File List** folder and select **New File Item**. The New File Item window appears.

New File Item	
[-] General Properties	
Name *	
Description	
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

4. Enter a **Name** and **Description** for the file, and then click **OK**. For more information, see [“Model Template Component Files”](#) on page 133.
5. After all required files have been created, select each filename and assign values for the properties on the right. For more information, see [“File List Properties”](#) on page 153.
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.

New Property	
[-] General Properties	
Name *	
Description	
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

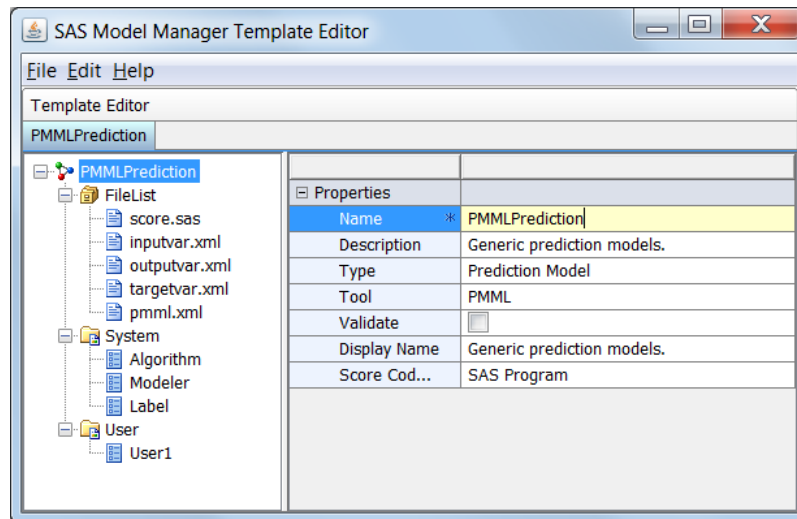
- b. Enter a **Name** and **Description** for the new property, and then click **OK**.
  - c. After all required properties have been created, select each property name and enter the property field values on the right. For more information, see [“System and User Properties”](#) on page 153.
7. To save 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

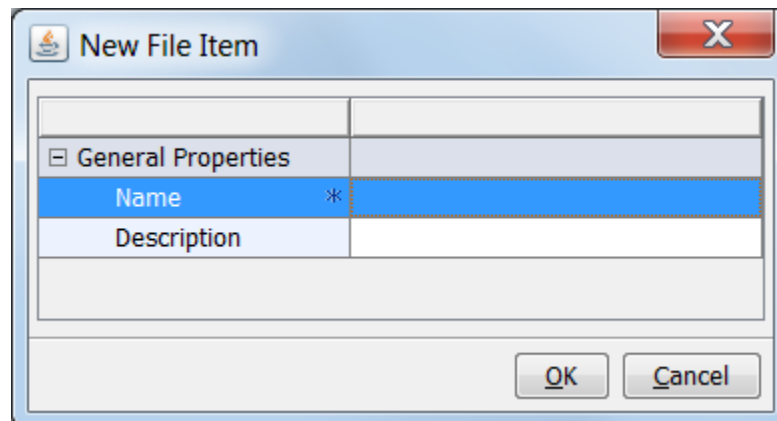
Only user templates can be modified with the use of the SAS Model Manager Template Editor. Reserved templates can be used as a model to create a customized model template. When you open a reserved template, modify the template, and upload the template to the SAS Content Server, SAS Model Manager creates 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:

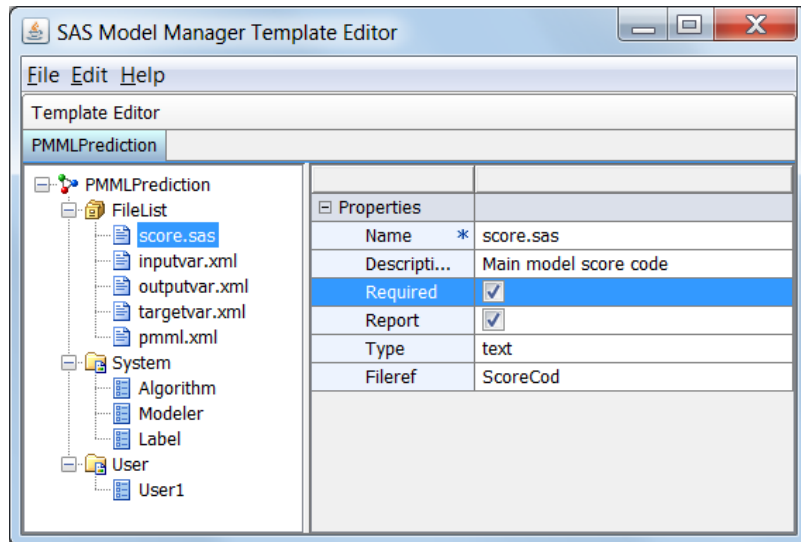
1. From the SAS Model Manager Template Editor window, select **File** ⇒ **Browse** ⇒ **Browse Templates**.
2. Select a model template, and then click **Open**. The template properties appear on the right.



3. To modify model template properties, select the property and make changes to the property value. For more information, see [“Template Properties”](#) on page 152.
4. Create or modify file properties:
  - To create a new file property, right-click the **File List** folder and select **New File Item**. Complete the properties, and then click **OK**.



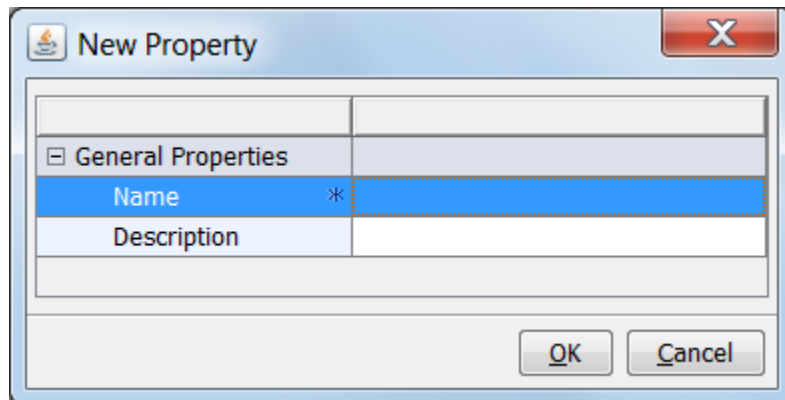
- To modify file properties, select the filename and make changes to the property values.



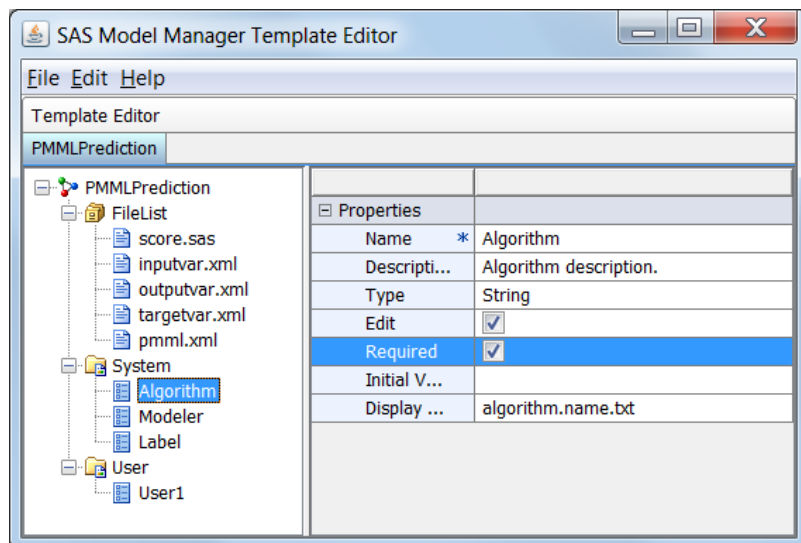
For more information, see “File List Properties” on page 153.

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 153](#).

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

## Model Template Properties

### Template Properties

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

Property Name	Description
<b>Name</b>	Identifies the name of the template. This property is required. The characters @ \ / * % # & \$ ( ) ! ? < > ^ + ~ ` = { } [ ]   ; : ‘ ” cannot be used in the name.
<b>Description</b>	Specifies user-defined information about the template.
<b>Type</b>	<p>Specifies the type of the model. SAS Model Manager supports the following model types:</p> <p><b>Analytical Model</b> specifies the type of model that is associated with the Analytical model function.</p> <p><b>Classification Model</b> specifies the type of model that is associated with the Classification model function.</p> <p><b>Prediction Model</b> specifies the type of model that is associated with the Prediction model function.</p> <p><b>Clustering Model</b> specifies the type of model that is associated with the Segmentation model function.</p> <p>For more information about the model function types, see <a href="#">“SAS Model Manager Model Templates” on page 132</a>.</p>
<b>Tool</b>	Specifies a text value that describes which tool is used to produce this type of model.
<b>Validate</b>	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.

Property Name	Description
<b>Display Name</b>	Specifies a text value that is displayed as the name of the model template.
<b>Score Code Type</b>	Specifies whether the imported model score code runs by using a <b>DATA Step</b> fragment, <b>SAS Program</b> code, or <b>PMML</b> .

### **File List Properties**

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

Property Name	Definition
<b>Name</b>	Identifies the name of the file. This property is required.
<b>Description</b>	Specifies user-defined information about the file.
<b>Required</b>	When it is selected, indicates that the file is a required component file of the model that must be imported before using the model.
<b>Report</b>	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.
<b>Type</b>	Specifies a file whose type is text or binary.
<b>Fileref</b>	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
<b>Name</b>	Identifies the name of the property. This is a required field.
<b>Description</b>	Specifies user-defined information about the property.

Property Name	Description
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
<b>Default Scoring Task Input Table</b>	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 <b>Default Scoring Task Input Table</b> property inherits the property value from the associated version or project, if one is specified.
<b>Default Scoring Task Output Table</b>	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 <b>Default Scoring Task Output Table</b> property inherits the property value from the associated version or project, if one is specified.
<b>Default Performance Table</b>	Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager project.  A model's <b>Default Performance Table</b> 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.

Property Name	Description
<b>Default Train Table</b>	<p>The train table is optional and is used only as information. However, when a value is specified for a model's <b>Default Train Table</b> property, SAS Model Manager does the following:</p> <ul style="list-style-type: none"> <li>• uses default train table to validate scoring functions or scoring model files when a user publishes the associated project champion model or challenger models to a database.</li> <li>• checks the <b>Validate scoring results</b> box in the Publish Scoring Function window.</li> </ul>
<b>Expiration Date</b>	<p>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.</p>
<b>Model Label</b>	<p>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 <b>Model Label</b> property, SAS Model Manager uses the text string that is specified for the <b>Model Name</b> property. The <b>Model Label</b> property can be useful if the Model Name property that is specified is too long for use in plots. This property is optional.</p>
<b>Subject</b>	<p>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.</p>
<b>Algorithm</b>	<p>Specifies the computational algorithm that is used for the selected model. This property cannot be modified.</p>
<b>Function</b>	<p>Specifies the SAS Model Manager function class that was chosen when the SAS Model Manager associated project was created. The <b>Function</b> property specifies the type of output that models in the predictive model project generate. For more information, see <a href="#">“Overview of Importing Models” on page 125.</a></p>
<b>Modeler</b>	<p>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.</p>
<b>Tool</b>	<p>Specifies whether the imported model came from SAS Enterprise Miner or from other modeling tools.</p>

Property Name	Description
<b>Tool Version</b>	Specifies the version number of the tool that is specified in the <b>Tool</b> property.
<b>Score Code Type</b>	<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 <b>DATA step</b>, <b>SAS Program</b>, and <b>PMML</b>.</p> <p><i>Note:</i> If the model is created using PMML 4.0, the <b>Score Code Type</b> is <b>DATA step</b> and not <b>PMML</b>.</p> <p><i>Note:</i> SAS Model Manager cannot publish models to a database whose <b>Score Code Type</b> model property is set to <b>SAS Program</b> and <b>PMML</b>.</p>
<b>Template</b>	Specifies the SAS Model Manager model template that was used to import the model and to create pointers to its component files and metadata.
<b>Copied From</b>	Specifies where the original model is if this model is copied from another model in the SAS Model Manager repository.
<b>Target Variable</b>	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 is <b>GENDER</b> .
<b>Target Event Value</b>	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 <b>Target Variable</b> property. For example, if a model predicts when GENDER=M, then the target event value is <b>M</b> .



## Chapter 9

# Scoring Models

---

<b>Overview of Scoring Tasks</b> .....	<b>157</b>
<b>Scoring Task Tabbed Views</b> .....	<b>159</b>
<b>Create Scoring Output Tables</b> .....	<b>162</b>
What Is a Scoring Output Table? .....	162
How to Create a Scoring Output Table Definition .....	162
<b>Create a Scoring Task</b> .....	<b>164</b>
<b>Modify a Scoring Task</b> .....	<b>167</b>
<b>Map Scoring Task Output Variables</b> .....	<b>167</b>
<b>Execute a Scoring Task</b> .....	<b>168</b>
<b>Schedule Scoring Tasks</b> .....	<b>170</b>
About Scoring Task Schedules .....	170
Schedule a Scoring Task .....	171
Delete a Scoring Task Schedule .....	172
<b>Graph Scoring Task Results</b> .....	<b>172</b>
<b>Generated Scoring Task Content Files</b> .....	<b>175</b>
<b>Scoring Task Properties</b> .....	<b>176</b>
<b>Result Set Properties</b> .....	<b>177</b>

---

## 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 &

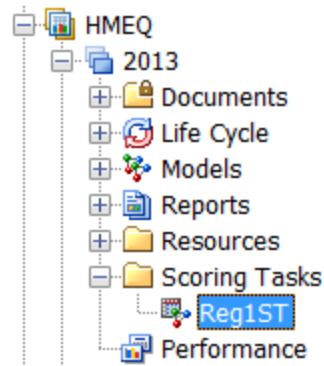
Execute, Read, and Write security permissions. For more information, see [“Scoring Task Output Tables” on page 35](#).

**CAUTION:**


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

*Note:* In order to run scoring tasks in a high-performance environment, the scoring output table must be a SAS table and not a database table.

You create a new scoring task in the **Scoring** folder of your version. Here is an example of a **Scoring** folder under the version **2013**.



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 162](#) and [“Project Tables” on page 33](#).
- 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 164](#) and [“Scoring Task Tabbed Views” on page 159](#).
- 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 167](#).
- 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 168](#).
- To run a scoring task at a scheduled time, you can open the New Schedule window to specify the date, time and frequency that you want the scoring task to run. For more information, see [“Schedule Scoring Tasks” on page 170](#).
- 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 172](#).

**See Also**

- [“Create Scoring Output Tables” on page 162](#)

- [“Create a Scoring Task” on page 164](#)
- [“Modify a Scoring Task” on page 167](#)
- [“Map Scoring Task Output Variables” on page 167](#)
- [“Execute a Scoring Task” on page 168](#)
- [“Graph Scoring Task Results” on page 172](#)

---

## 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 through graphing the scoring results. The following is a list of these views:

Tabbed View	Description
<b>Properties</b>	This view contains five groupings of properties, some of which have fields that can be modified. Click <b>Save</b> , when you update any of the properties in this view. Here is a list of the groupings: <ul style="list-style-type: none"> <li>• <a href="#">“General Properties” on page 503</a></li> <li>• <a href="#">“System Properties” on page 504</a></li> <li>• <a href="#">“Scoring Task Properties” on page 176</a></li> <li>• <a href="#">“Result Set Properties” on page 177</a></li> <li>• <a href="#">“Schedule Properties” on page 518</a></li> </ul>
<b>Model Input Variables</b>	This view shows the model input variables.
<b>Input Table</b>	This view shows the variables and attributes of the scoring task input table. For each input variable the model lists on the <b>Model Input Variables</b> tab, there must be a matching variable in the input table. The input table can contain additional variables.
<b>Model Output Variables</b>	This view shows the model output variables.

Tabbed View	Description
<p><b>Output Table</b></p>	<p>This view shows the variables and attributes of the scoring task output table. The value of the <b>Map to Model Variable Name</b> 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 the model output variable name is the same as the output variable name. If these are not correct or are incomplete, correct them and then click <b>Save</b> 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>
<p><b>Pre-code</b></p>	<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>
<p><b>Post-code</b></p>	<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>
<p><b>SAS Code</b></p>	<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>Note that this view shows you the same code as the model's <b>SAS Code</b> view. To see the code that was actually executed for a scoring task, expand the scoring task's folder and select the <b>taskCode.sas</b> 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																
<b>Results</b>	<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"> <li>• <b>Result Set</b> is the view of the result data set.</li> <li>• <b>Log</b> is the view of the log from the last execution of this scoring task. This is the default view.</li> <li>• <b>Output</b> is the view of the listing file from the last execution of this scoring task.</li> </ul> <p>Both the Log and the Output views are visible 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 <b>Scoring Task Type</b> property. If the value of this property is <b>Test</b>, then the results are read from the .sas7bdat file that is specified in the <b>Output Table</b> property. This file is in the SAS Content server and can be found in the scoring task folder. If the value of this property is <b>Production</b>, then the results are read from the location of the data source that is known to the SAS Metadata Repository. For more information, see <a href="#">“Create Scoring Output Tables” on page 162</a>.</p>																
<b>Graph</b>	<p>This view displays graphs of the scoring task output that you create by clicking the <b>Graph Wizard</b> button. For more information, see <a href="#">“Graph Scoring Task Results” on page 172</a>.</p>																
<b>Job History</b>	<p>This view displays the following metadata about the execution of a scoring task job:</p> <table border="0"> <tr> <td data-bbox="876 1245 975 1272">Job Name</td> <td data-bbox="1043 1245 1289 1299">specifies the name of the scoring task job name.</td> </tr> <tr> <td data-bbox="876 1318 975 1346">Job Status</td> <td data-bbox="1043 1318 1347 1398">specifies whether the job is queued to start, running, or has been completed.</td> </tr> <tr> <td data-bbox="876 1417 975 1472">Execution Status</td> <td data-bbox="1043 1417 1331 1533">specifies whether the job was completed successfully, was completed with warnings, or was completed with errors.</td> </tr> <tr> <td data-bbox="876 1551 999 1579">Date Started</td> <td data-bbox="1043 1551 1347 1606">specifies the date on which the job started.</td> </tr> <tr> <td data-bbox="876 1625 986 1675">Date Completed</td> <td data-bbox="1043 1625 1347 1675">specifies the date on which the job was completed.</td> </tr> <tr> <td data-bbox="876 1694 916 1722">Log</td> <td data-bbox="1043 1694 1331 1749">specifies the revision number for the taskCode.log file.</td> </tr> <tr> <td data-bbox="876 1768 948 1795">Output</td> <td data-bbox="1043 1768 1331 1822">specifies the revision number for the taskcode.lst file.</td> </tr> <tr> <td data-bbox="876 1841 979 1869">SAS Code</td> <td data-bbox="1043 1841 1331 1896">specifies the revision number for the taskCode.sas file.</td> </tr> </table>	Job Name	specifies the name of the scoring task job name.	Job Status	specifies whether the job is queued to start, running, or has been completed.	Execution Status	specifies whether the job was completed successfully, was completed with warnings, or was completed with errors.	Date Started	specifies the date on which the job started.	Date Completed	specifies the date on which the job was completed.	Log	specifies the revision number for the taskCode.log file.	Output	specifies the revision number for the taskcode.lst file.	SAS Code	specifies the revision number for the taskCode.sas file.
Job Name	specifies the name of the scoring task job name.																
Job Status	specifies whether the job is queued to start, running, or has been completed.																
Execution Status	specifies whether the job was completed successfully, was completed with warnings, or was completed with errors.																
Date Started	specifies the date on which the job started.																
Date Completed	specifies the date on which the job was completed.																
Log	specifies the revision number for the taskCode.log file.																
Output	specifies the revision number for the taskcode.lst file.																
SAS Code	specifies the revision number for the taskCode.sas file.																

**See Also**

[“Modify a Scoring Task” on page 167](#)

## Create Scoring Output Tables

### **What Is a Scoring Output Table?**

A scoring output table is a SAS data set that contains the data from executing a scoring task. The scoring output table cannot be a database table. You can provide a scoring output table or you can create a scoring output table definition using SAS Model Manager. When you create a scoring task, you specify either the scoring output table that you provide or the scoring task output definition as the scoring task output table.

A SAS data set that you provide as a scoring output table must be registered to the SAS Metadata Repository or a libref must exist in the Data Sources category view for the library where the data set resides.

You create a scoring output table definition by using the Create Output Table function directly from the model. In the Create Output Table function, you select variables from a scoring task input table as well as variables from the model’s output. The variables in the **Input Variables** table are variables from the scoring task input table if one is specified for the **Default Scoring Task Input Table** property for a project, version, or model property. Otherwise, the **Input Variables** table is empty. The **Output Variables** that appear in the window are model output variables. You use the variables from both tables to create the scoring output table.

SAS Model Manager saves the table definition as metadata in the SAS Metadata Repository. The location of the metadata is defined by the SAS library that you specify when you create the output table definition. After SAS Model Manager creates the table definition, the table can be selected as the output table for subsequent scoring tasks.

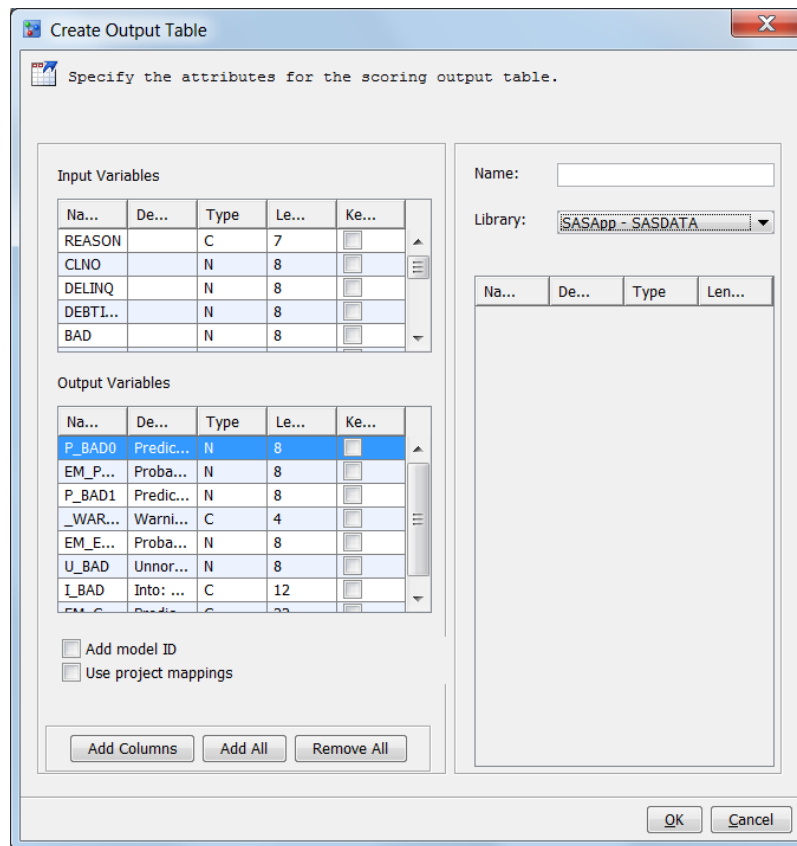
You can view a scoring output table definition on the **SAS Folders** tab of the Data Sources category view. If you do not see the table, right-click the library name and select **Refresh**.

When you execute a scoring task with a type of **Test**, the output table is located under the scoring task node in the Project Tree. If the scoring task type is **Production**, the output table is stored at the designated SAS library location that you specified in the **Create Output Table** window.

### **How to Create a Scoring Output Table Definition**

To create a scoring output table definition, 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 definition that is unique to the SAS library. The names in the **Library** selection list are the SAS libraries that are defined in the SAS Metadata Repository under the **Data Library Manager** folder in SAS Management Console. The name can contain letters and the underscore ( \_ ). Spaces and special characters are not allowed.
  4. Select a SAS library name from the **Library** list.
5. Select the check boxes in the **Keep** column for the input and output variables that you want to include in the output table definition.

*Note:* If you want to use all of the variables in the output table definition, you can click **Add All** instead of selecting individual variables. If you use **Add All** and you want to use the project output variable names in the scoring table instead of the model variable names, click **Use Project Mappings** before you click **Add All**.

6. Select the **Add model ID** check box to add the **ModelID** variable to the output table definition. The model UUID appears in all rows of the output table.
7. Select the **Use Project Mappings** check box to use the project's output variable names in the output table definition for model variables that are mapped to project variables.

*Note:* If you want to use all of the variables in the output table definition and you want to use the project output variable names in the scoring table instead of the model variable names, click **Use Project Mappings** before you click **Add All**.

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 Variables** table and the **Output Variables** table.
  - Click **Add All** to add all the columns in the **Input Variables** table and all the variables in the **Output Variables** table.

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

9. Click **OK**. The output table definition is created and you receive a confirmation message. Click **Close** to close the confirmation message.

*Note:* You can view the output table definition in the **SAS Folders** tab or the **SAS Libraries** tab of the Data Sources category view. You might need to right-click the library name and select **Refresh** to see the table definition.

### See Also

- [“Overview of Scoring Tasks” on page 157](#)
- [“Project Tables” on page 33](#)

---

## Create a Scoring Task

To create a new scoring task, follow these steps:

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



2. Enter a **Name** for the scoring task. The name can contain only letters and the underscore character ( \_ ). As an option, enter a **Description**.
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**.

By default, a test scoring task scores 1000 records. The scoring output table is saved to the SAS Content Server. A production scoring task generates either a SAS data set or a database table and uses all the records in the scoring input table. The scoring output table is saved to the library that is specified in score.sas.

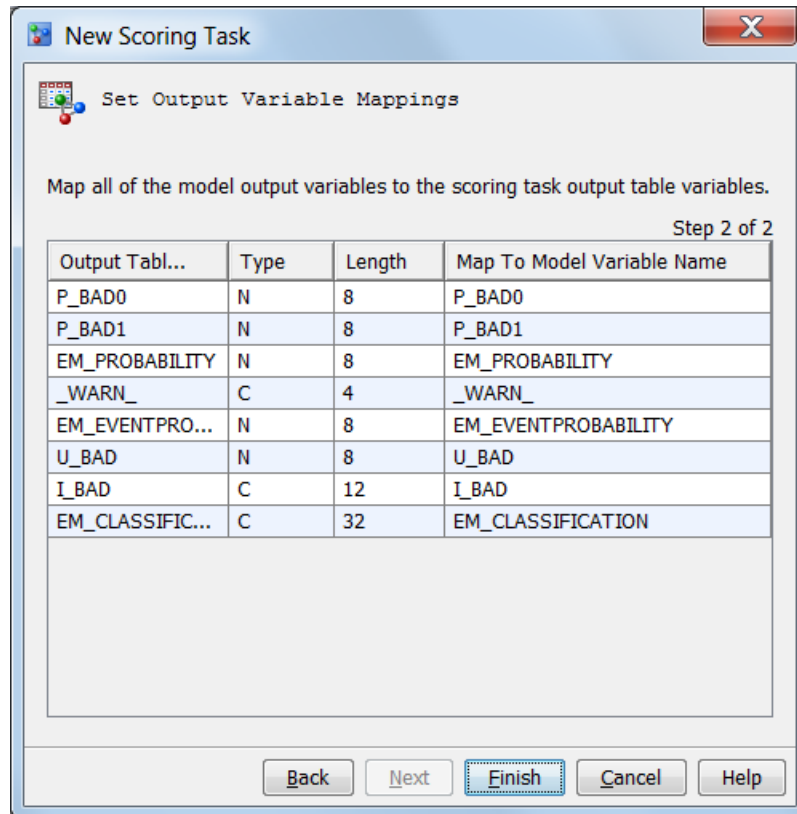
*Note:* A best practice is to select **Test** before beginning all scoring tasks. Later, when you are satisfied with the results of running the scoring task and you are ready to put the task into production, you can change the type to **Production**. When you run in production mode in the Windows and UNIX environments, the scoring task output table definition in the SAS Metadata Server must have Modify, Read and Execute, Read, and Write security permissions. The SAS Model Manager user who is executing the scoring task must have Write permission to the physical folder on the SAS Application 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 **Browse** and select an input table.

*Note:* The scoring input table must have at least one column.

6. Click **Browse** and select an output table.

7. (Optional) Select a SAS Application Server from the **Default server** list.
8. Click **Next**.
9. 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.



10. Click **Finish**. The scoring task is created under the **Scoring Tasks** node.

*Note:* Four of the **Scoring Task** properties cannot be modified after the scoring task has been created. To change the following properties, you must create a new scoring task.

- Name
- Model
- Input Table
- Output Table

### See Also

- “Modify a Scoring Task” on page 167
- “Execute a Scoring Task” on page 168
- “Schedule Scoring Tasks” on page 170
- “Generated Scoring Task Content Files” on page 175
- “Scoring Task Properties” on page 176

## Modify a Scoring Task

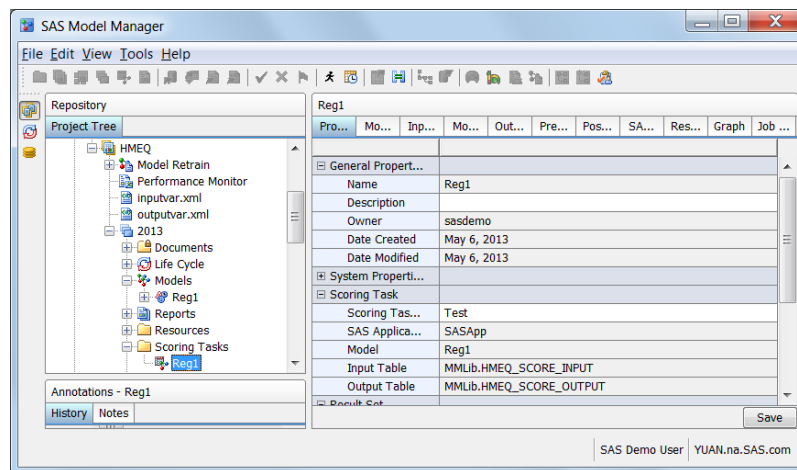
The following are the only four tabbed views that can be modified:

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

For more information, see [“Scoring Task Tabbed Views”](#) on page 159.

To modify a scoring task, follow these steps:

1. Expand the **Scoring Tasks** folder and select your scoring task.
2. Click the tab information that you want to modify. The default is the **Properties** tab.



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

### See Also

- [“Create a Scoring Task”](#) on page 164
- [“Execute a Scoring Task”](#) on page 168
- [“Schedule Scoring Tasks”](#) on page 170
- [“Generated Scoring Task Content Files”](#) on page 175
- [“Scoring Task Tabbed Views”](#) on page 159

## Map Scoring Task Output Variables

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

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

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

Reg1ST										
Propert...	Model I...	Input T...	Model ...	Output ...	Pre-code	Post-co...	SAS C...	Results	Graph	Job His...
Output Table Variable Na...	Type	Length	Map To Model Variable Name ↵							
REASON	C	7	REASON							
P_BAD1	N	8	REASON							
EM_PROBABILITY	N	8	CLNO							
DELINQ	N	8	DELINQ							
CLNO	N	8	DEBTINC							
score	N	8	BAD							
BAD	N	8	JOB							
			YOJ							
			MORTDUE							

4. Click **Save**.


### See Also

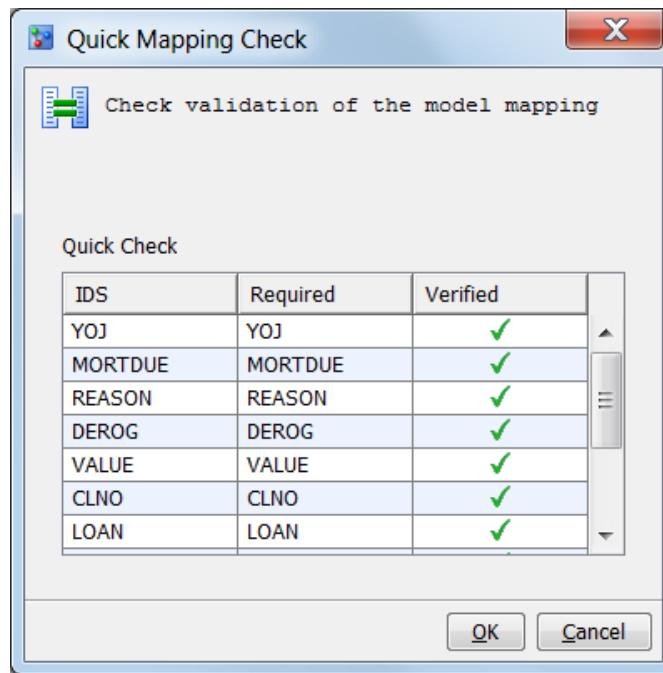
- “Create Scoring Output Tables” on page 162
- “Create a Scoring Task” on page 164
- “Modify a Scoring Task” on page 167
- “Execute a Scoring Task” on page 168

---

## 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 167.
2. Select the scoring task name, and click  in the toolbar to validate the input variables.



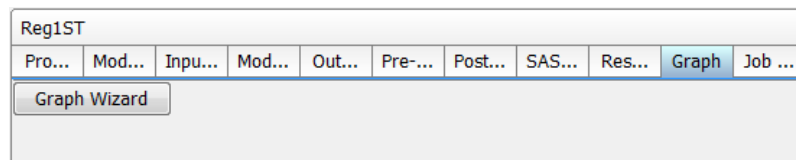
The table in the Quick Mapping Check window compares the column names from the Scoring Input Table (that is, the input data source) against the model's input variable names.

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 might be unreliable.

3. Expand the **Scoring Tasks** folder. Right-click the scoring task 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.

Reg1ST										
Prope...	Model...	Input ...	Model...	Outpu...	Pre-c...	Post-c...	SAS ...	Results	Graph	Job Hi...
	BAD	REASON	DELINQ	CLNO	P_BAD1		EM_PROBABILITY		score	
1	1.0	HomeImp	0.0	9.0	0.0891795482		0.9108204518		1.0	
2	1.0	HomeImp	2.0	14.0	0.0891795482		0.9108204518		1.0	
3	1.0	HomeImp	0.0	10.0	0.0891795482		0.9108204518		1.0	
4	1.0		.	.	0.0891795482		0.9108204518		1.0	
5	0.0	HomeImp	0.0	14.0	0.0891795482		0.9108204518		0.0	
6	1.0	HomeImp	0.0	8.0	0.115876633914781	0.884123366085219		1.0		
7	1.0	HomeImp	2.0	17.0	0.0891795482		0.9108204518		1.0	
8	1.0	HomeImp	0.0	8.0	0.10785931509276908	0.8921406849072309		1.0		
9	1.0	HomeImp	2.0	12.0	0.0891795482		0.9108204518		1.0	
10	1.0	HomeImp	0.0	13.0	0.0891795482		0.9108204518		1.0	
11	1.0		.	.	0.0891795482		0.9108204518		1.0	
12	1.0	HomeImp	1.0	9.0	0.0891795482		0.9108204518		1.0	
13	1.0	HomeImp	0.0	25.0	0.0891795482		0.9108204518		1.0	
14	0.0		0.0	24.0	0.0891795482		0.9108204518		0.0	
15	1.0	HomeImp	1.0	16.0	0.0891795482		0.9108204518		1.0	
16	1.0	HomeImp	1.0	8.0	0.0891795482		0.9108204518		1.0	
17	1.0	HomeImp	0.0	22.0	0.0891795482		0.9108204518		1.0	

- Click the **Graph** tab and then **Graph Wizard** to graph the results. For more information, see “[Graph Scoring Task Results](#)” on page 172.



*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 175.

## See Also

- “[Create a Scoring Task](#)” on page 164
- “[Schedule Scoring Tasks](#)” on page 170
- “[Modify a Scoring Task](#)” on page 167
- “[Generated Scoring Task Content Files](#)” on page 175
- “[Graph Scoring Task Results](#)” on page 172

---

## Schedule Scoring Tasks

### About Scoring Task Schedules

Instead of executing a scoring task from the SAS Model Manager Project Tree, you can schedule a scoring task to run on a particular date and time. You can also schedule how often you want the scoring task to run. Advanced settings enable you to set the scheduling server, the batch server to run the scoring task, and the location of the scoring results.


Before you can schedule a scoring task, the password for your user ID must be made available to the SAS Metadata Repository. Passwords can be added using SAS

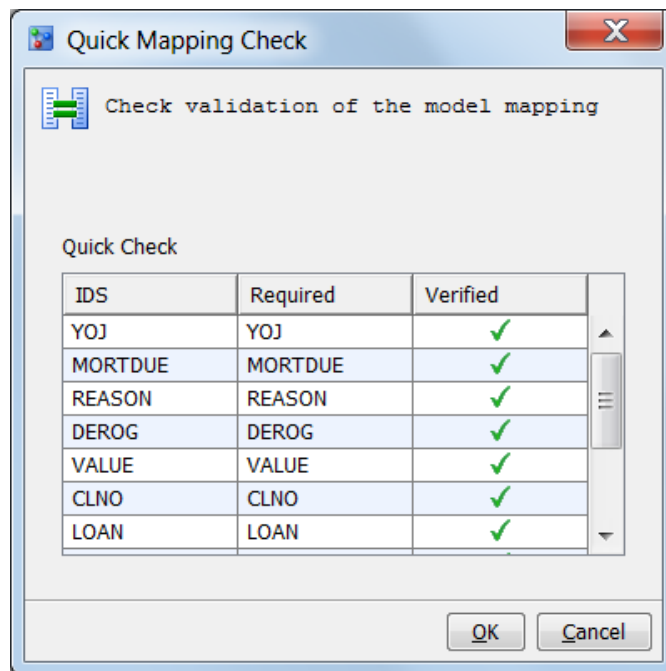
Management Console. To add or update your password, contact your SAS Model Manager Administrator.

Scoring task schedules cannot be edited. If you need to modify a scoring schedule, delete the schedule and create a new schedule.


## Schedule a Scoring Task

To schedule a scoring task:

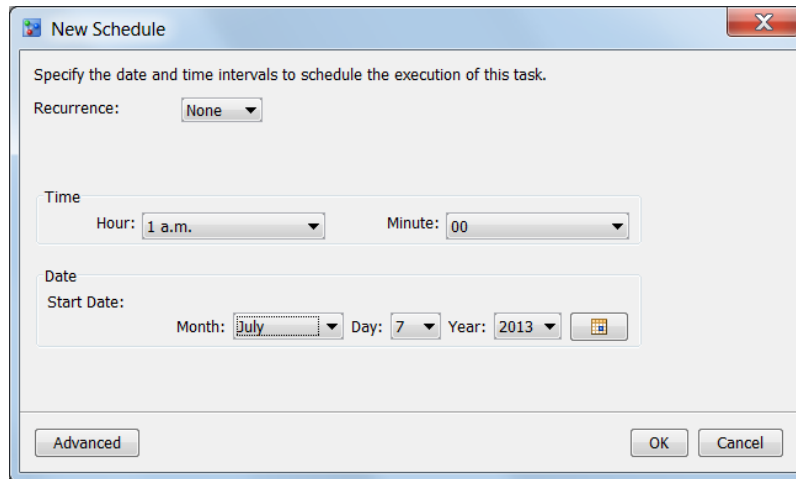
1. Verify that you have mapped the model output variables to the scoring task output variables. For more information, see “[Map Scoring Task Output Variables](#)” on page 167.
2. Select the scoring task name, and click  in the toolbar to validate the input variables.




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 might be unreliable.

3. Right-click the scoring task and select **New Schedule**. The New Schedule window appears.



4. To set how often to run the scoring task, click the **Recurrence** list box and select a time interval.
5. To set the time to run the job, select an hour from the **Hour** list box and select a minute from the **Minute** list box.
6. To set the start date, click the calendar  and select a start date. Instead of using the calendar, you can select a month from the **Month** list box, select a day from the **Day** list box, and select a year from the **Year** list box.
7. (Optional) Click **Advanced**. Select the server that schedules the job from the **Scheduling server** list box. Select the batch server that runs the job from the **Batch server** list box. Select a location for the scoring task output from the **Location** list box. Click **OK**.
8. Click **OK**. A dialog box message confirms that the schedule was created. Click **Close**.

*Note:* Scoring task schedules cannot be edited. To change the schedule, delete the schedule and create a new schedule.

### Delete a Scoring Task Schedule

To delete a scoring task schedule:

1. Right-click the scoring task and select **Delete Schedule**.
2. A message box confirms that the schedule was deleted. Click **Close**.

*Note:* Scoring task schedules cannot be edited. To change the schedule, delete the schedule and create a new schedule.

---

## Graph Scoring Task Results

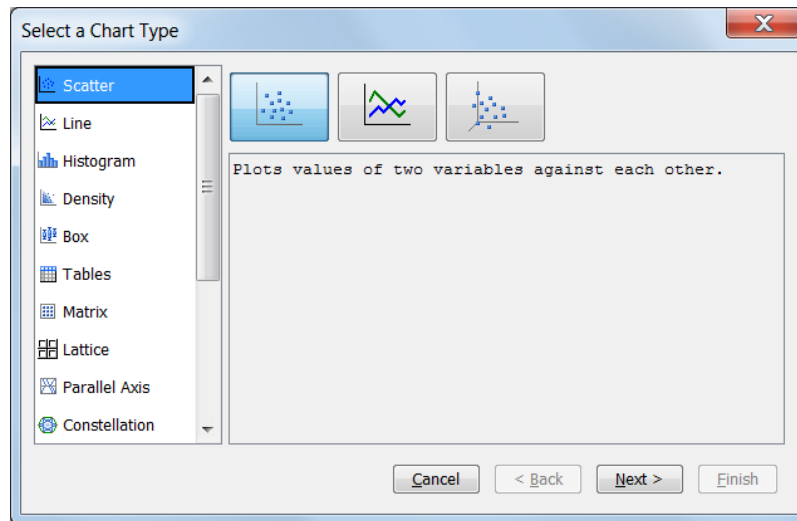
After the scoring task successfully executes, you can use the graphics wizard to plot your customized charts.

To graph scoring task results, follow these steps:

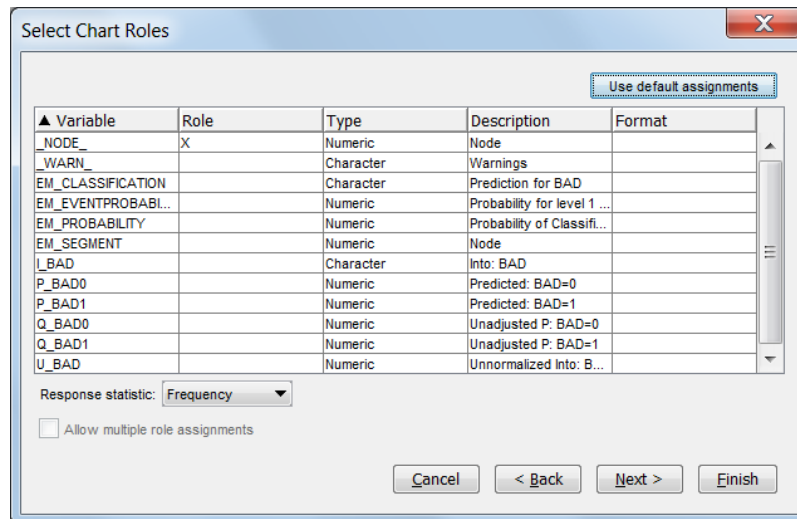
1. Select the scoring task in the **Scoring Tasks** node.



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



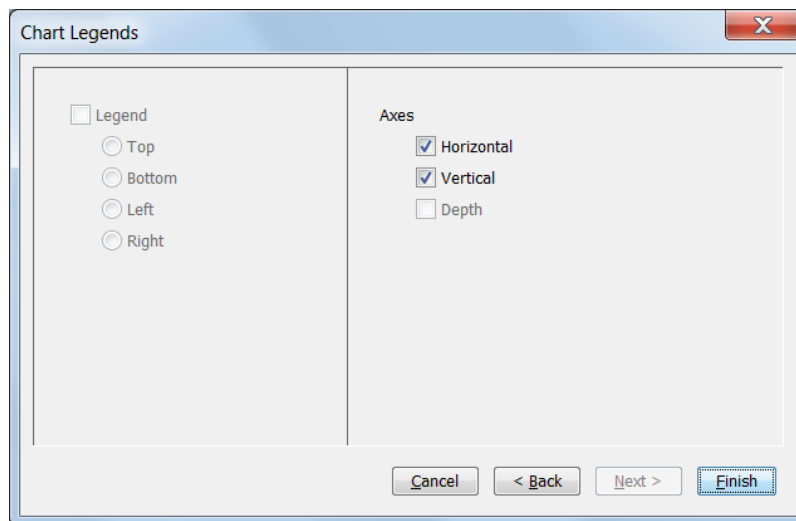
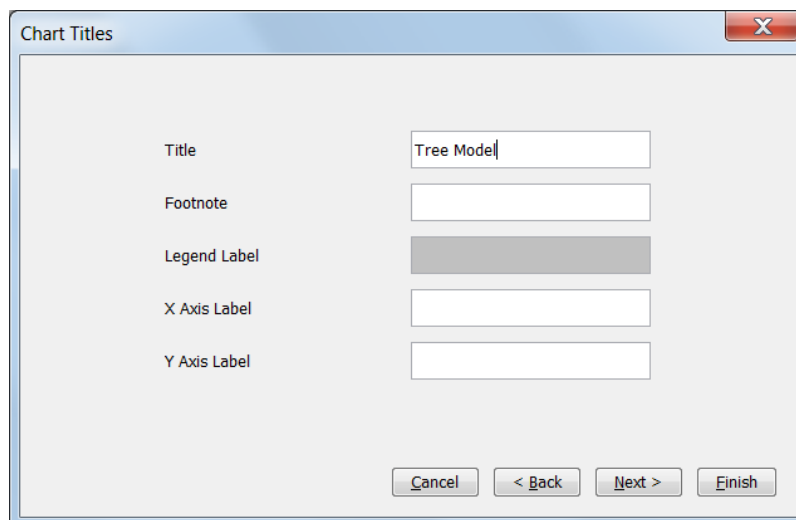
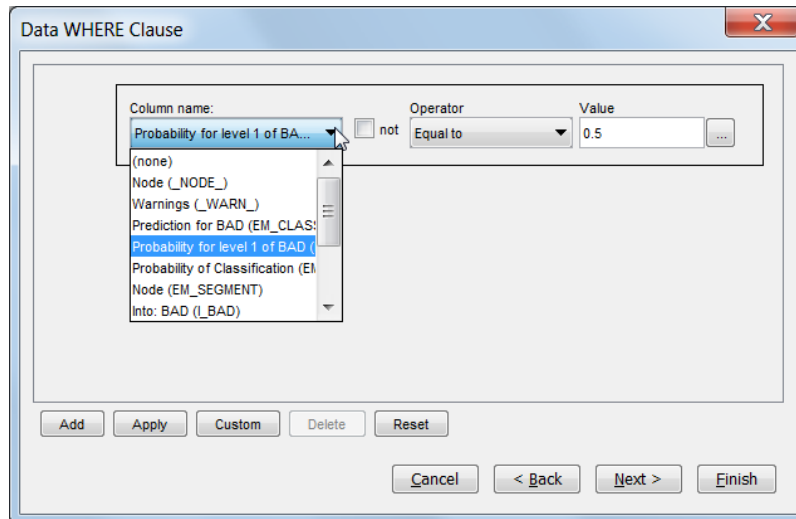
3. Select a chart type from the list box on the left, and then select the chart display button on the right. A description of how the results are graphed for a chart appears.
4. Click **Next**. The Select Chart Roles window appears.
5. For each variable, select the row and then select a value from the **Role** list.



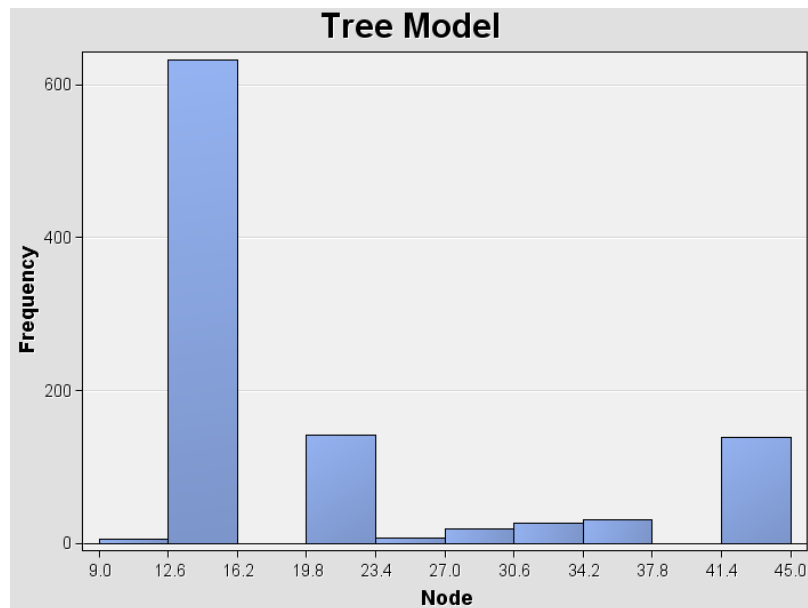
*Note:* You can also click **Use default assignments** to assign variables to the required roles.

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

Here are examples of the options that you can define:



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

**See Also**

[“Execute a Scoring Task” on page 168](#)

---

## 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
<i>&lt;Output Table Name&gt;</i> .sas7bdat	This file is created whenever the scoring task is executed and the scoring task property <b>Scoring Task Type</b> is set to <b>Test</b> . The contents are not the most recent scoring output results if the type of the scoring task was changed from <b>Test</b> to <b>Production</b> , and the scoring task is executed.
taskCode.log	This file is the SAS log for the scoring task code that is executed on the SAS Application 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.
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 SAS Application Server for execution.

Filename	Description
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 <a href="#">“Scoring Task Tabbed Views” on page 159</a> .
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 <a href="#">“Scoring Task Tabbed Views” on page 159</a> .

### See Also

- [“Execute a Scoring Task” on page 168](#)
- [“Scoring Task Tabbed Views” on page 159](#)

---

## 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
<b>Scoring Task Type</b>	Specifies a value of <b>Test</b> or <b>Production</b> for the type of scoring task.
<b>SAS Application Server</b>	Specifies the name of the SAS Application Server to which SAS Model Manager is connected. This value is taken from the SAS Metadata Repository.
<b>Model</b>	Specifies the name of the model whose score code is to be executed on the SAS Application Server. This value is set when the scoring task is created and cannot be modified.
<b>Input Table</b>	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
<b>Output Table</b>	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 <b>Test</b> , this property identifies the name of the output file ( <i>output_filename.sas7bdat</i> ) that is created by the SAS Application 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 <b>Production</b> , then this setting identifies the output table where the results of the scoring are written.

### See Also

- “Create a Scoring Task” on page 164
- “Modify a Scoring Task” on page 167
- “Execute a Scoring Task” on page 168
- “Schedule Scoring Tasks” on page 170

---

## Result Set Properties

The following property provides information that is specific to the scoring task.

Property Name	Description
<b>Number of Observations</b>	<p>When <b>Scoring Task Type</b> is set to <b>Test</b>, 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 <b>Scoring Task Type</b> is set to <b>Production</b>, this property specifies how many observations are to be read from the scoring task input table and displayed when you select <b>Result Set</b> from the <b>Results</b> tab. The default value is 0, indicating that there is no limit. This value cannot be changed in the SAS Model Manager client. The SAS Model Manager administrator can modify the value by using SAS Management Console. For more information, see <i>SAS Model Manager: Administrator's Guide</i>.</p>

---

**See Also**

- [“Create a Scoring Task” on page 164](#)
- [“Modify a Scoring Task” on page 167](#)

## Chapter 10

## Validating Models Using Reports

---

<b>Overview of Model Comparison, Validation, and Summary Reports</b> . . . . .	<b>180</b>
What Are Model Comparison, Validation, and Summary Reports? . . . . .	180
The Model Comparison, Validation, and Summary Report Input Files . . . . .	181
The Model Comparison, Validation, and Summary Report Output Files . . . . .	182
<b>Model Profile Reports</b> . . . . .	<b>183</b>
About Model Profile Reports . . . . .	183
Create a Model Profile Report . . . . .	183
<b>Delta Reports</b> . . . . .	<b>184</b>
About Delta Reports . . . . .	184
Create a Delta Report . . . . .	185
<b>Dynamic Lift Reports</b> . . . . .	<b>186</b>
About Dynamic Lift Reports . . . . .	186
Verify Project and Model Property Settings . . . . .	187
Create a Dynamic Lift Report . . . . .	188
<b>Interval Target Variable Report</b> . . . . .	<b>189</b>
About Interval Target Variable Reports . . . . .	189
Verify Project and Model Properties . . . . .	189
Create an Interval Target Variable Report . . . . .	189
<b>Basel II Reports</b> . . . . .	<b>191</b>
About Basel II Reports . . . . .	191
The Loss Given Default and Probability of Default Model Validation Report Properties . . . . .	191
Prerequisites for Loss Given Default Reports . . . . .	192
Create a Loss Given Default Report . . . . .	193
Prerequisites for Probability of Default Model Validation Reports . . . . .	194
Create a Probability of Default Model Validation Report . . . . .	195
<b>Training Summary Data Set Reports</b> . . . . .	<b>196</b>
About Training Summary Data Set Reports . . . . .	196
Generate the Training Summary Data Sets . . . . .	196
Create a Training Summary Data Set Report . . . . .	197
<b>View Reports</b> . . . . .	<b>198</b>

---

## Overview of Model Comparison, Validation, and Summary Reports

### ***What Are Model Comparison, Validation, and Summary Reports?***

The SAS Model Manager model comparison, validation, and summary 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 resilience of candidate models. The model validation reports use statistical measures to validate the stability, performance, and calibration of Basel II risk models and parameters. The training summary data set report creates frequency and distribution charts that summarize the train table variables.

The reports present information about a number of attributes that can affect 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 that is 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.

#### **Interval Target Variable Report**

The Interval Target Variable report creates two plots for you to view the actual versus predicted values for a model and the actual versus residual values for a model. Interval Target Variable report can be created only for prediction models.

These are the Basel II model validation reports:

#### **Loss Given Default Report**

The Loss Given Default (LGD) report calculates the amount that might be lost in an investment and calculates the economic or regulatory capital for Basel II compliance.

#### **Probability of Default Model Validation Report**

The Probability of Default (PD) Validation report estimates the probability of defaulting on a debt that is owed. Probability of default is used to calculate economic or regulatory capital for Basel II compliance.

The model validation reports use statistical measures that report on these model validation measures:

- The model stability measures track the change in distribution for the modeling data and scoring data.
- The model performance measures check the model's ability to distinguish between accounts that have not defaulted and accounts that have defaulted, as well as report



on the relationship between actual default probability and predicted default probability.

- The model calibration measures check the accuracy of the selected models for the LGD and the PD reports by comparing the correct quantification of the risk components with the available standards.

This is the train table data set summary report:

### Training Summary Data Set Report

The Training Summary Data Set report creates frequency and distribution charts for a training data set.

You create the model comparison, validation, and summary reports using the New Report window that you start from a version's **Reports** node:

**Report Options**

Type: Delta Report

Format: PDF

Style: SAS default

**Select Models**

Select	ID	Name	Version	Type	Champi...
<input checked="" type="checkbox"/>	MMRoot/H...	Reg1	2013	Classification	NO
<input checked="" type="checkbox"/>	MMRoot/H...	Tree1	2013	Classification	NO

**Report Properties**

Property	Value
General Properties	
Name	* Delta_D2013-05-11T12.08
Description	The Delta Report

OK Cancel

### The Model Comparison, Validation, and Summary Report Input Files

SAS Model Manager uses a test table as the input table for the Dynamic Lift report and the Interval Target Variable report.

Before you can create a Dynamic Lift report or the Interval Target Variable report, make sure that a test table has been added to the SAS Metadata Repository using SAS Management Console or that a libref has been defined in the **Edit Start-up Code** window for the SAS library where the test table resides. The test table can be viewed in the Data Sources category view. Then, specify the test table in the project property **Default Test Table**.

You specify the input table for Basel II validation reports in the **New Report** window. The input file for the validation reports can contain only input variables or it can contain input and output variables. If the input table contains input and output variables, the report generation does not need to run a scoring task to obtain the output variables.

To create a train table summary report, use a train table to create training summary data sets. The training summary data sets are used to create the train table summary report. Either the train table must be added to the SAS Metadata Repository or a libref must be defined in the Edit Start-up Code window for the train table library. The train table must then be specified in the project or version property for **Default Train Table**. You create the training summary data sets by using the Generate Training Summary Data Set feature of SAS Model Manager from the version node in the Project Tree.

### See Also

- “Specific Properties for a Project” on page 505
- “Creating a Test Table” on page 40

## The Model Comparison, Validation, and Summary Report Output Files

The New Reports window stores the model comparison, validation, and summary report output files in a report node under the **Reports** node. The name of the report node is the value of the **Name** box that you specified in the New Report window **Report Properties** table.

Each time you create a report, these files are generated:

- the report in either HTML, PDF, RTF, or EXCEL format

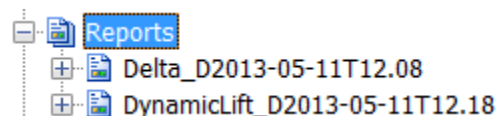
*Note:* The Loss Given Default and Probability of Default model validation reports can be created only in PDF and RTF formats.

- taskCode.log
- taskCode.sas

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 can view the report from the **Reports** node:



*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** node.

---

## 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 that is associated with the variable
Format	the SAS format that is 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

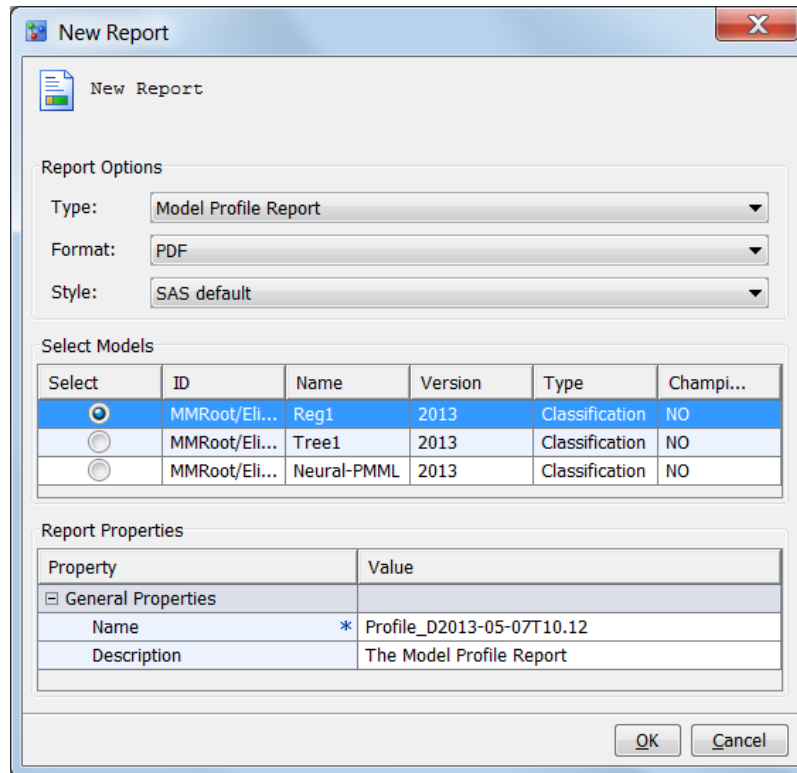
These are the tasks that you perform for Model Profile reports:

- [“Create a Model Profile Report” on page 183](#)
- [“View Reports” on page 198](#)

### Create a Model Profile Report

To create a Model Profile report:

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



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 **Style** list box, select the style for the output. The default is **SAS default**. Other options are **Seaside**, **Meadow**, and **Harvest**.
6. In the **Select** column of the **Select Models** table, select the models that you want to include in the report. This report requires only one model.
7. 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 name can contain letters, spaces, the underscore ( `_` ), the hyphen ( `-` ), and the period ( `.` ).
8. Click **OK**. A message confirms that the report was created successfully.

### See Also

[“View Reports” on page 198](#)

---

## 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.

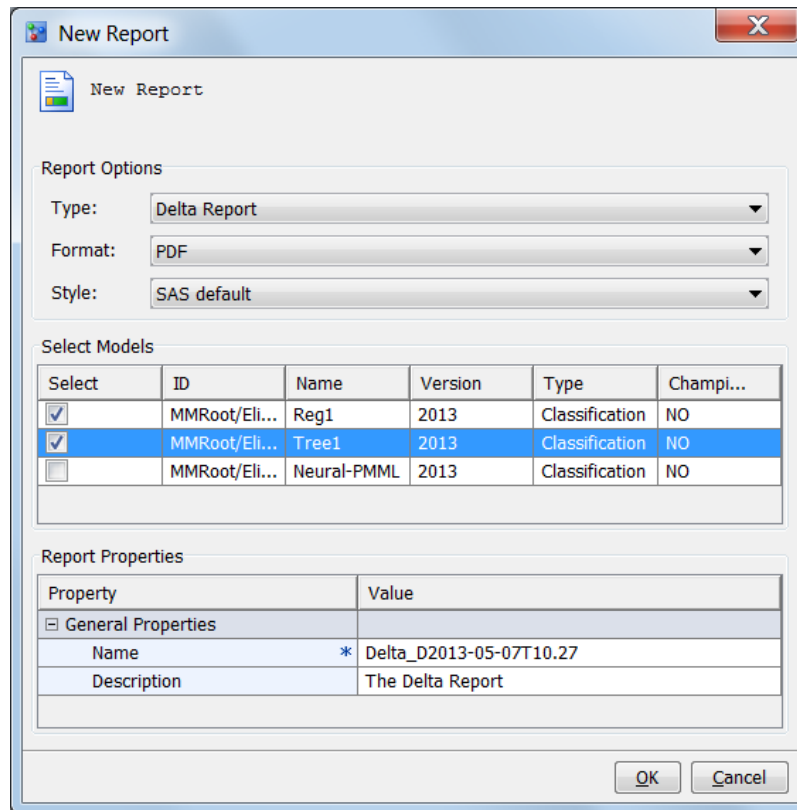
Here is a description of each of the columns in the output of a Delta report:

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.

### Create a Delta Report

To create a Delta report:

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



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 **Style** list box, select the style for the output. The default is **SAS default**. Other options are **Seaside**, **Meadow**, and **Harvest**.
6. 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.
7. 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 name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ).
8. Click **OK**. A message confirms that the report was created successfully.

### See Also

[“View Reports” on page 198](#)

---

## 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. Before you can create a Dynamic Lift report, certain project and model property settings must be set.

For PMML 4.0 and later models, the **Valid variable name** option in SAS Management Console must be set to **Yes** by a SAS Model Manager administrator. For more information, see the *SAS Model Manager 12.3 Administrator's Guide*.

## 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. The model must have the same training target variable as the project.

#### 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's 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 **Edit**. Click the **Model Variables** property for each project variable and select a model variable.

#### 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 <b>RESPONSE</b> .

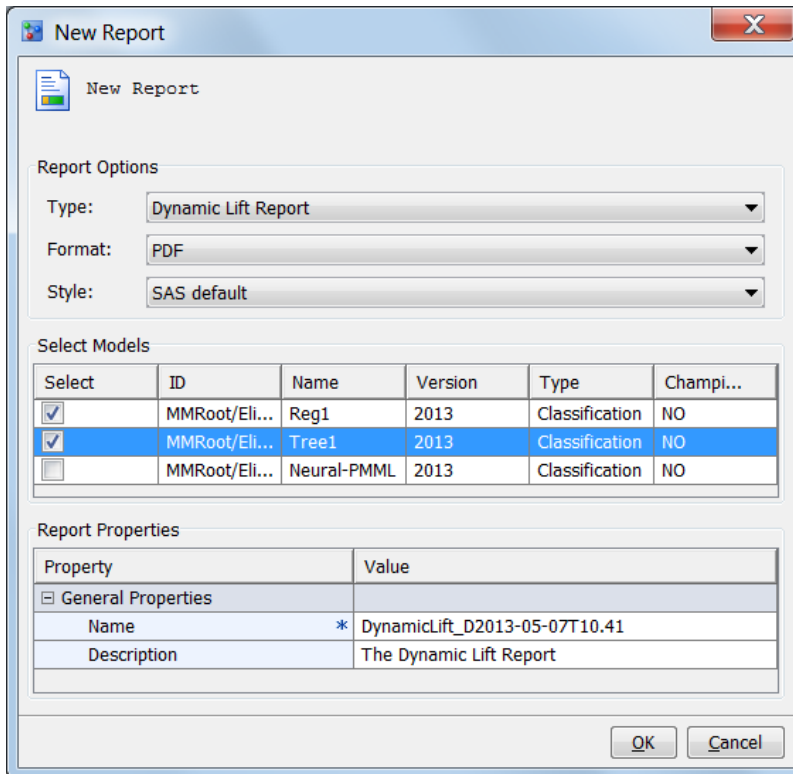
Property	Description
<b>Score Code Type</b>	Specifies whether the score code runs using a DATA step fragment or SAS code that is not a DATA step fragment.

*Note:* Dynamic Lift reports are not applicable to models whose **Score Code Type** property has a value of PMML. For PMML 4.0 and later, a Dynamic Lift report can be created for a PMML model whose **Score Code Type** is DATA step.

## Create a Dynamic Lift Report

After ensuring that the appropriate project and model properties have been set, create the report.

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



**Report Options**

Type: **Dynamic Lift Report**

Format: **PDF**

Style: **SAS default**

**Select Models**

Select	ID	Name	Version	Type	Champi...
<input checked="" type="checkbox"/>	MMRoot/Eli...	Reg1	2013	Classification	NO
<input checked="" type="checkbox"/>	MMRoot/Eli...	Tree1	2013	Classification	NO
<input type="checkbox"/>	MMRoot/Eli...	Neural-PMML	2013	Classification	NO

**Report Properties**

Property	Value
General Properties	
Name	* DynamicLift_D2013-05-07T10.41
Description	The Dynamic Lift Report

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 **Style** list box, select the style for the output. The default is **SAS default**. Other options are **Seaside**, **Meadow**, and **Harvest**.
6. In the **Select** column of the **Select Models** table, select the boxes for the models that you want to include in the report.



7. 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 name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ).
8. Click **OK**. A message confirms that the report was created successfully.
9. If errors occurred, ensure that the prerequisite properties have been set correctly or correct the reported model configuration error.

**See Also**

“View Reports” on page 198

---

## Interval Target Variable Report

### *About Interval Target Variable Reports*

The Interval Target Variable report creates two plots for you to view the actual versus predicted values for a model and the actual versus residual values for a model. Interval Target Variable report can be created only for prediction models. Before you can create an Interval Target Variable report, certain project and model property settings must be set.

### *Verify Project and Model Properties*

Before you can run an Interval Target Variable report, you must set the following project properties:

**Default Test Table**

Specifies a test table that is listed in the **SAS Metadata Repository** tab or the **SAS Libraries** tab in the Data Sources category view. 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. The model must have the same training target variable as the project.

**Output Prediction Variable**


Specifies the name of the output prediction variable.

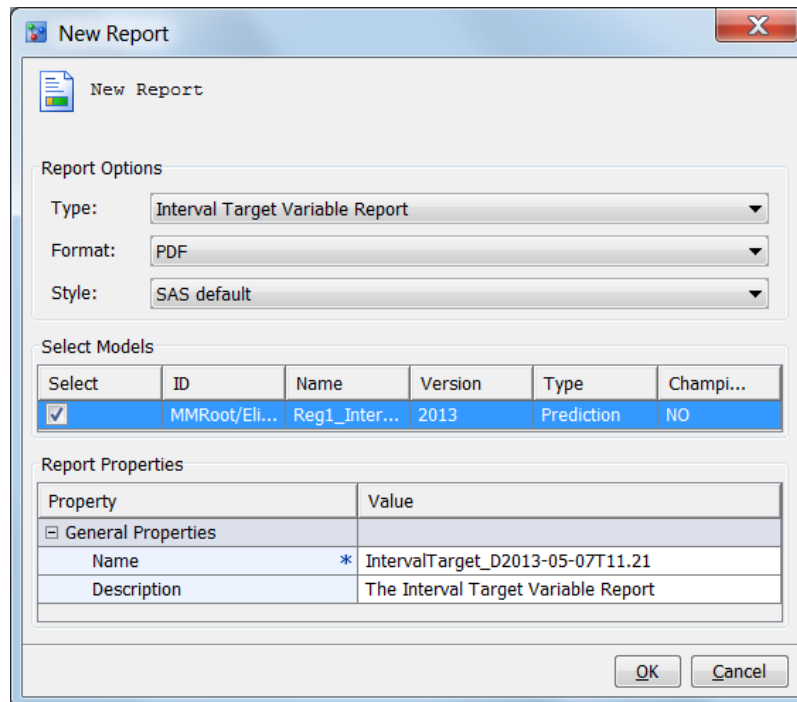
To verify the model mapping, click on the model name and verify that the model variables are mapped to the project variables. If the variable names are the same, you do not need to map the variables. If they are not mapped, for each project variable, click **Edit**, click the project variable’s **Model Variable** field, and select a variable name.

### *Create an Interval Target Variable Report*

After ensuring that the appropriate project properties have been set and the model mapping is set, create the report.

1. (Optional) Change the sample seed in SAS Management Console. The report is created based on the sample data of the default test table. By default, the sample seed is 12345.

2. Expand the version folder .
3. Right-click the **Reports** node and select **Reports** ⇒ **New Report**. The New Report window appears.



4. Select **Interval Target Variable Report** from the **Type** list box.
5. In the **Format** 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 **Style** box, select a report style. The default is **SAS default**. Other options are **Seaside**, **Meadow**, and **Harvest**.
7. In the **Select** column of the **Select Models** table, select the box for the models that you want to include in the report.
8. 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 `IntervalTarget_DdateTtime`. The name can contain letters, spaces, the underscore ( `_` ), the hyphen ( `-` ), and the period ( `.` ). Click **OK**. A dialog box message confirms that the report was created successfully.
9. If errors occurred, ensure that the prerequisite properties have been set correctly or correct the reported model configuration error.

### See Also

“View Reports” on page 198

---

## Basel II Reports

### About Basel II Reports

Basel II reports in SAS Model Manager provide several statistical measures and tests to validate the stability, performance, and calibration using Loss Given Default (LGD) and Probability of Default (PD) models.

#### Model stability measures

The model stability measures track the change in distribution of the modeling data and the scoring data.

#### Model performance measures

The model performance measures report this information:

- The model's ability to discriminate accounts that have defaulted with those that have not defaulted. The score difference between the accounts that default and those that do not helps determine the cut-off score, which is used to predict whether a credit exposure is a default.
- The relationship between the actual default probability and the predicted probability. This information is used to understand a model's performance over a period of time.

#### Model calibration measures

The model calibration measures check the accuracy of the LGD and PD models by comparing the correct quantification of the risk components with the available standards.

For a description of the statistical measures, see [Appendix 9, "Statistical Measures Used in Basel II Reports,"](#) on page 527.

### The Loss Given Default and Probability of Default Model Validation Report Properties

In order to create the Basel II reports, SAS Model Manager must know the input and output variables for the model. To run the reports, the New Report window requires the name of an input table. The input table can contain only input variables, or it can contain input and output variables. If the input table contain only input variables only, a scoring task must be run to obtain the output variable. If the input table contains the input and output variables, no scoring is necessary. You specify whether a scoring task must be run by setting the **Run Scoring Task** property in the New Report window. If the input table contains the input and output variables, the value of the **Run Scoring Task** can be **No**. If the input table contains only input variables, the **Run Scoring Task** property must be set to **Yes**.

The New Report window report properties requires the name of the variables from the input and output tables in order to map these variables to variables that are used by SAS Model Manager uses to create the reports.

The LGD report properties map these variables:

Time Period Variable	specifies the variable that is used to indicate a time period. The first time period begins with 1 and typically increments by 1. The default is <b>period</b> .
----------------------	--

Time Label Variable	(optional) specifies a label for the time period. If this variable exists in the input table, the report output contains a table that maps time periods to time labels.
Actual Variable	specifies the actual LGD variable. The default is <b>lgd</b> .
Predicted Variable	specifies the output prediction variable that is used only if scoring for the report is not performed by SAS Model Manager. If the report scoring is done by SAS Model Manager, this variable should be excluded by the input data set. The default is <b>p_lgd</b> .
Pool Variable	specifies the variable that names pool IDs. The default is <b>pool_id</b> .

The PD report properties map these variables:

Time Period Variable	specifies the variable that is used to indicate a time period. The first time period begins with 1 and typically increments by 1. The default is <b>period</b> .
Time Label Variable	(optional) specifies a label for the time period. If this variable exists in the input table, the report output contains a table that maps time periods to time labels.
Scorecard Bin Variable	specifies the scoring output variable that names the scorecard bins. The input table must include this variable if scoring for the PD report is performed outside of SAS Model Manager. If scoring is done by SAS Model Manager, do not include this variable in the input data set. The default is <b>scorecard_bin</b> .
Scorecard Points Variable	specifies the scoring output variable that names the scorecard points. The input table must include this variable if scoring for the PD report is performed outside of SAS Model Manager. If scoring is done by SAS Model Manager, do not include this variable in the input data set. The default is <b>scorecard_points</b> .
Cut-off Value	specifies the variable that is used to derive whether a credit exposure is a default. The cut-off value is also used to compute accuracy, sensitivity, specificity, precision, and error rate measures. You can use the score difference between accounts that default on loans and those that do not default on loans to determine the cut-off value. The default is <b>100</b> .

### **Prerequisites for Loss Given Default Reports**

Before you run an LGD report, select the project name and verify that the following project properties are set:

#### **Training Target Variable**

Specifies the name of the target variable that was used to train the model. The model must have the same training target variable as the project.

#### **Model Function**

Specifies the type of model function. For an LGD report, the model function must be Prediction.

#### **Class Target Level**


Specifies an **Interval** class target level.

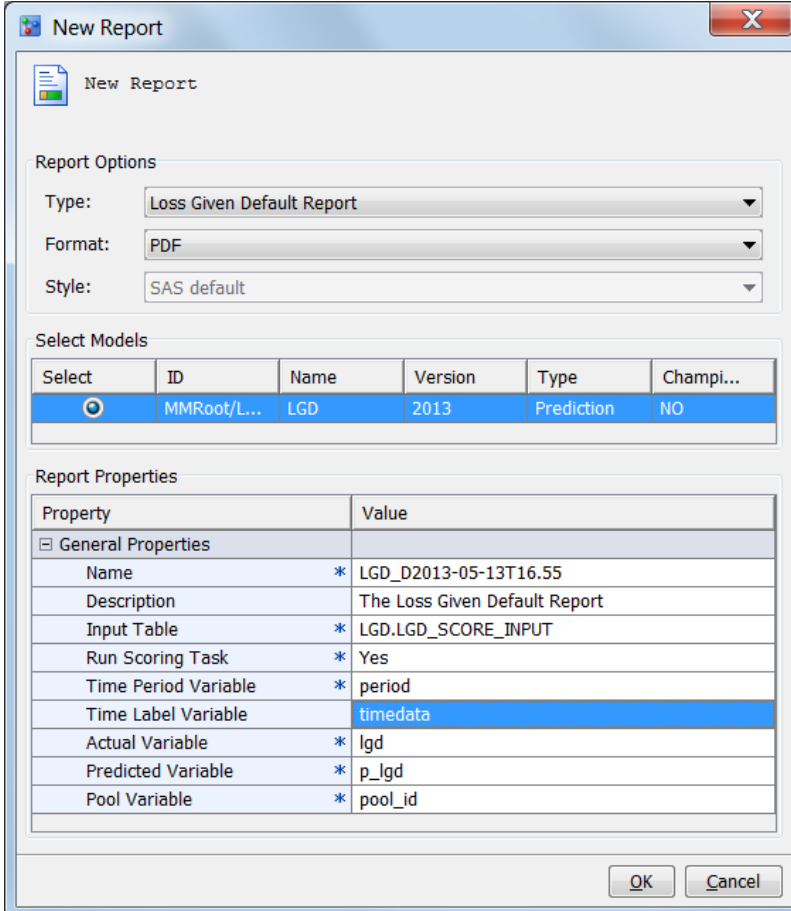
## Output Prediction Variable

Specifies the name of the output prediction variable.

### Create a Loss Given Default Report

To create an LGD report:

1. Expand the version folder .
2. Right-click the **Reports** node and select **Reports** ⇒ **New Report**. The New Report window appears.
3. Select **Loss Given Default Report** from the **Type** box.



**Report Options**

Type: Loss Given Default Report

Format: PDF

Style: SAS default

**Select Models**

Select	ID	Name	Version	Type	Champi...
<input checked="" type="radio"/>	MMRoot/L...	LGD	2013	Prediction	NO

**Report Properties**

Property	Value
General Properties	
Name	* LGD_D2013-05-13T16.55
Description	The Loss Given Default Report
Input Table	* LGD.LGD_SCORE_INPUT
Run Scoring Task	* Yes
Time Period Variable	* period
Time Label Variable	* <b>timedata</b>
Actual Variable	* lgd
Predicted Variable	* p_lgd
Pool Variable	* pool_id

OK Cancel

4. In the **Format** box, select the type of output that you want to create. The default is **PDF**. The other option is **RTF**.
5. In the **Select Models** box, select the model that you want to report on. This report requires one model.
6. Complete the **Report Properties**. The properties with an asterisk are required:
  - Enter a report name if you do not want to use the default value for the **Name** property. The name can contain letters, the underscore ( \_ ), hyphen ( - ), and the period ( . ).
  - (Optional) Enter a report description.

- For the **Input Table** property, click the **Browse** button and select a table from the **SAS Metadata Repository** tab or from the **SAS Libraries** tab that is used for scoring during the creation of the LGD report. The table can contain only input variables or it can contain input and output variables.
- If the input table that is specified in the **Input Table** property contains only input variables, set **Run Scoring Task** to **Yes**. If the input table contains input and output variables, set **Run Scoring Task** to **No**.
- The **Time Period Variable** specifies the variable from the input table whose value is a number that represents the development period. This value is numeric. The default is **period**.
- (Optional) In the **Time Label Variable** field, enter the variable from the input table that is used for time period labels. When you specify the time label variable, the report appendix shows the mapping of the time period to the time label.
- **Actual Variable** is the actual LGD variable. The default is **lgd**.
- **Predicted Variable** is the project scoring output variable. If the scoring for the LGD report is performed outside SAS Model Manager, the input data set must include this variable. If the scoring for the LGD report is done by SAS Model Manager, the input data set should not include this variable. The default is **p\_lgd**.
- **Pool Variable** is the variable from the input table that is used to identify a two-character pool identifier. The default is **pool\_id**.

*Note:* The variable names that you specify can be user-defined variables. A variable mapping feature maps the user-defined variables to required variables.

7. Click **OK**. A dialog box message confirms that the report was created successfully.

### **See Also**

[“View Reports” on page 198](#)

## **Prerequisites for Probability of Default Model Validation Reports**

Before you can create a PD report, verify that the following project settings are specified and that the output variables have been mapped:

1. Select the project name and ensure that these properties are set:

#### **Training Target Variable**

Specifies the name of the target variable that was used to train the model. The model must have the same training target variable as the project.

#### **Class Target Level**

Specifies a **Binary** class target level.


#### **Output Event Probability Variable**

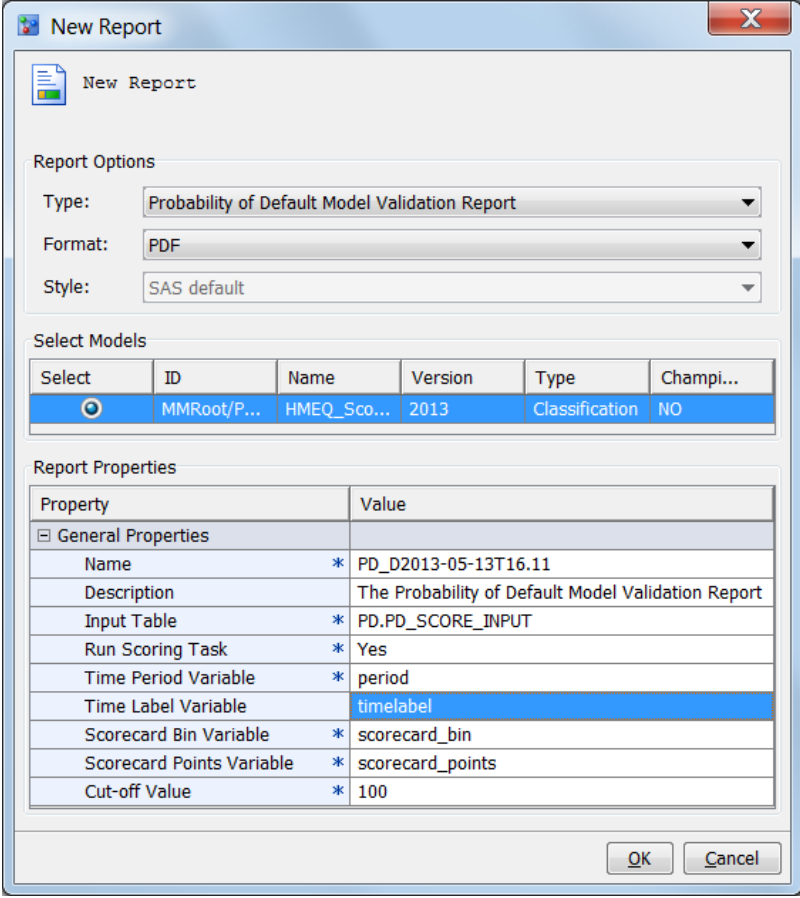
Specifies the name of the output event probability variable.

2. Expand the **Models** folder and select the PD model.
3. In the model Details view, select the **Model Mapping** tab and verify that the project variables are mapped to model variables. To map output variables, click **Edit**, select the cell in the **Model Variables** for the unmapped project variable, and select an output variable. Click **OK**.

## Create a Probability of Default Model Validation Report

To create a PD report:

1. Expand the version folder .
2. Right-click the **Reports** node and select **Reports** ⇒ **New Report**. The New Report window appears.
3. Select **Probability of Default Model Validation Report** from the **Type** box.



**Report Options**

Type: Probability of Default Model Validation Report

Format: PDF

Style: SAS default

**Select Models**

Select	ID	Name	Version	Type	Champi...
<input checked="" type="radio"/>	MMRoot/P...	HMEQ_Sco...	2013	Classification	NO

**Report Properties**

Property	Value
General Properties	
Name	* PD_D2013-05-13T16.11
Description	The Probability of Default Model Validation Report
Input Table	* PD.PD_SCORE_INPUT
Run Scoring Task	* Yes
Time Period Variable	* period
Time Label Variable	timelabel
Scorecard Bin Variable	* scorecard_bin
Scorecard Points Variable	* scorecard_points
Cut-off Value	* 100

OK Cancel

4. In the **Format** box, select the type of output that you want to create. The default is **PDF**. The other option is **RTF**.
5. In the **Select Models** box, select the box for the model that you want to report on.
6. Complete the **Report Properties**:
  - Enter a report name if you do not want to use the default value for the **Name** property. The name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ).
  - (Optional) Enter a report description.
  - For the **Input Table** property, click the **Browse** button and select a table from the **SAS Metadata Repository** tab or from the **SAS Libraries** tab. The table can contain only input variables or it can contain both input and output variables.

*Note:* When a scoring input table for a PD report contains data and one or more time periods do not contain default or non-default loan information, these

time periods are not used to calculate the PD measurements. Time periods that are not used to calculate the PD measurements are represented in a chart with dashed lines.

- If the input table that contains only input variables, set **Run Scoring Task** to **Yes**. If the input table contains input and output variables, set **Run Scoring Task** to **No**.
- The **Time Period Variable** specifies the variable from the input table whose value is a number that represents the development period. This value is numeric. The time period for PD reports begin with 1. The default is **period**.
- (Optional) In the **Time Label Variable** field, enter the variable from the input table that is used for time period labels. When you specify the time label variable, the report appendix shows the mapping of the time period to the time label.
- **Scorecard Bin Variable** is the variable from the input table that contains the scorecard bins. If the scoring job for the PD report is run outside SAS Model Manager, the scorecard bin variable must be a variable in the input table. If scoring is done within SAS Model Manager, do not include the variable in the input table. The default is **scorecard\_bin**.
- **Scorecard Points Variable** is the variable that contains the scorecard points. If the scoring job for the PD report is run outside SAS Model Manager, the scorecard points variable must be a variable in the input table. If scoring is done within SAS Model Manager, do not include the variable in the input table. The default is **scorecard\_points**.
- **Cut-off Value** is the maximum value that can be used to derive the predicted event and to further compute accuracy, sensitivity, specificity, precision, and error rate. The default is **100**.

*Note:* The variable names that you specify can be user-defined variables. A variable mapping feature maps the user-defined variables to required variables.

7. Click **OK**. A dialog box message confirms that the report was created successfully.

### **See Also**

[“View Reports” on page 198](#)

---

## **Training Summary Data Set Reports**


### **About Training Summary Data Set Reports**

A Training Summary Data Set report creates frequency and distribution charts that summarize the train table variables. Using the default train table, SAS Model Manager generates data sets in the **Resources** folder that contain numeric and character variable summaries, and variable distributions. These data sets are used to create the summary report. Before you can create the report, you must generate the training summary data sets.

### **Generate the Training Summary Data Sets**


To generate the training summary data sets:

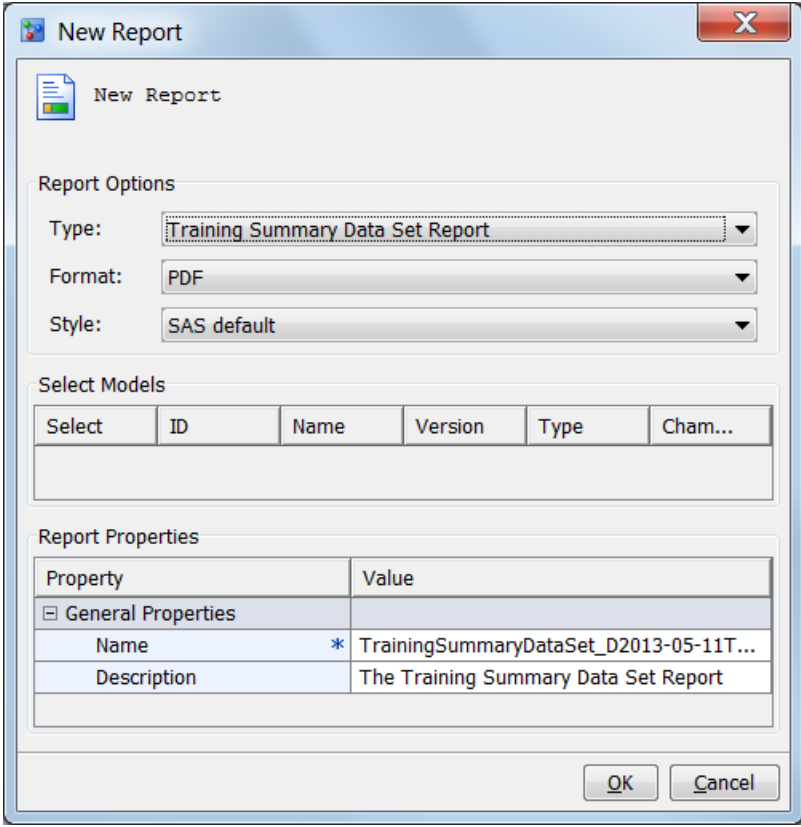


1. Click on the version folder  and verify that the **Default Train Table** property contains the train table for the report.
2. Right-click the version and select **Generate Training Summary Data Set**. The Generate Training Summary Data Set window appears.
3. Select the variables for the report by selecting the box that is next to each variable in the **Select** column. To select all variables, click **Select All**.
4. Click **OK**. SAS Model Manager creates data sets in the **Resources** folder.

### Create a Training Summary Data Set Report

To generate a train table summary report for a version:

1. Expand the version folder .
2. Right-click the **Reports** folder and select **New Report**. The **New Report** window appears.



**Report Options**

Type: Training Summary Data Set Report

Format: PDF

Style: SAS default

**Select Models**

Select	ID	Name	Version	Type	Cham...

**Report Properties**

Property	Value
[-] General Properties	
Name	* TrainingSummaryDataSet_D2013-05-11T...
Description	The Training Summary Data Set Report

OK Cancel

3. Select **Training Summary Data Set 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 **Style** list box, select a style for the output. The default is **SAS default**. Other options are **Seaside**, **Meadow**, and **Harvest**.
6. In the **Report Properties** box, complete **Name** and **Description** properties if you do not want to use the default values. The default value for the **Name** property uses the

form `TrainingSummaryDataSet_DateTime`. The name can contain letters, spaces, the underscore ( `_` ), the hyphen ( `-` ), and the period ( `.` ). Click **OK**.

7. A dialog box message confirms that the report was created successfully.

## See Also

[“View Reports” on page 198](#)

---

## View Reports

To view a report:

1. Expand the version folder and the **Reports** folder.



2. Right-click the report name and select **View Report**. If authentication is required, enter your user ID and password. The report appears.

## Chapter 11

# Validating Models Using User Reports

---

<b>Overview of User Reports</b> . . . . .	<b>199</b>
Ad Hoc Reports and User-Defined Reports . . . . .	199
Comparison of Ad Hoc and User-Defined Reports . . . . .	200
Output Created by User Reports . . . . .	200
<b>Ad Hoc Reports</b> . . . . .	<b>201</b>
Overview of Ad Hoc Reports . . . . .	201
Create an Ad Hoc Report . . . . .	202
Example Ad Hoc Report . . . . .	202
<b>User-Defined Reports</b> . . . . .	<b>204</b>
Overview of User-Defined Reports . . . . .	204
Create a User-Defined Report . . . . .	204
Defining SAS Model Manager Macro Variables for a User-Defined Report . . . . .	205
Upload SAS Programs to the SAS Content Server . . . . .	205
The Report Template . . . . .	205
Edit a SAS Program on the SAS Content Server . . . . .	208
Delete a SAS Program from the SAS Content Server . . . . .	208
Run a User-Defined Report . . . . .	208
View a User-Defined Report . . . . .	209
Example User-Defined Report . . . . .	209

---

## 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 customize reports to meet your business requirements. The ad hoc report enables you to develop, test, and run your report 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 that is available to SAS Model Manager and is run using the New Report window.

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 <b>Reports</b> 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 create a new ad hoc report, SAS Model Manager creates these files:

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

Each time you run create a new user-defined report, SAS Model Manager 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 an output file of the same name already exists.

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 of Ad Hoc Reports

To create an ad hoc report, you must first write a SAS report program. When the report code is ready, you copy your code to the **SAS Editor** tab in the Create Ad Hoc Report window. 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, or to specify a style other than the default style for your report, you modify your report with code that is provided by SAS and that enables you specify the report output format and style. 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 submit the code in the Create Ad Hoc Report window 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. After the code runs successfully, you can create the ad hoc report.

To create an ad hoc report:

1. Expand the version folder .
2. Right-click the **Reports** node and select **Create Ad Hoc Report**. The Create Ad Hoc Report window appears.
3. In the **Select Models** table, select any number of models.
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, and the report style:

```
Filename mmreport catalog "sashelp.modelmgr.reportexportmacros.source";
%include mmreport;
%MM_ExportReportsBegin(reportFormat=report-format, reportStyle=report-style,
  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-style* with one of the following values: SAS default, Seaside, Meadow, or Harvest.

Replace *report-name* with the name of your ad hoc report. The name can contain letters, the underscore ( \_ ), hyphen ( - ), and the period ( . ).

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. Enter 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 was 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:

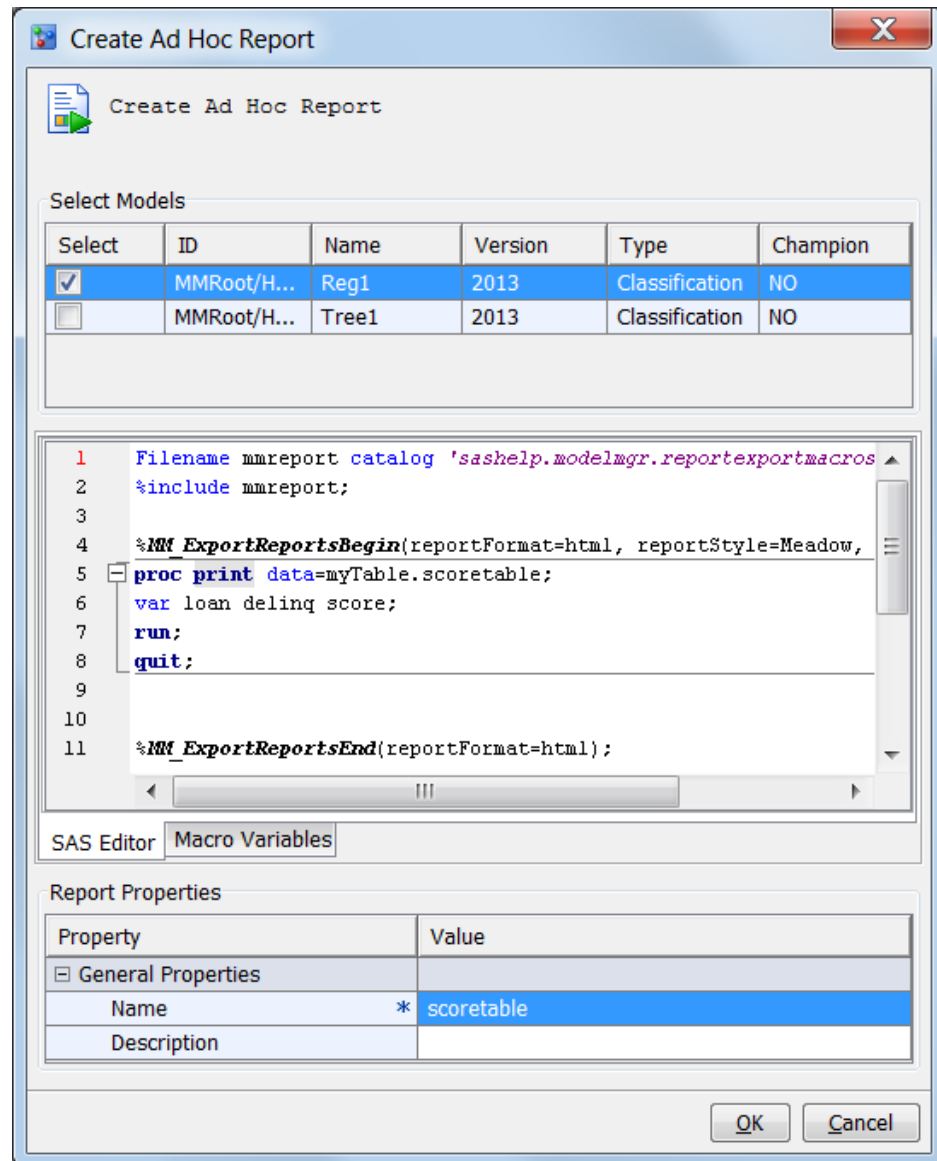
```
Filename mmreport catalog "sashelp.modelmgr.reportexportmacros.source";
%include mmreport;

%MM_ExportReportsBegin(reportFormat=html, reportStyle=Meadow, fileName=PerfDS);
```

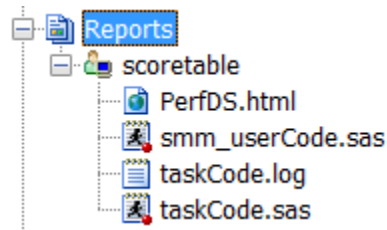
```
proc print data=myTable.scoretable;
var loan delinq score;
run;
quit;
```

```
%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 the output:

Obs	LOAN	DELINQ	score
1	1100.00	0	0.08918
2	162.06	2	0.08918
3	1292.02	0	0.08918
4	783.13	.	0.08918
5	1700.00	0	0.08918

---

## 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 **Type** list box of the New Reports window.

The New Report window includes controls to specify the type of output that the report creates, such as HTML or PDF, and a style for the report. You can modify your report to include the SAS code so that the New Report window offers the report output controls for your report.

### Create a User-Defined Report

To create a user-defined report:

1. Write and test your SAS program that creates a report.
2. 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 and the **Style** list box in the New Report window. The **Select Formats** list box enables you to select a report output format of HTML, PDF, RTF, or Excel. The **Style** list box enables you to select a report output style for your report.



Replace *report-name* with the name of your user-defined report. The name can contain letters, the underscore ( \_ ), hyphen ( - ), and the period ( . ). 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;
```

3. In the report XML file, add this SAS program name to the FILENAME= argument of the <Code> element (for example, <Code filename="myUserReport.sas"/>). See [“The Report Template” on page 205](#).

For an example of a report, see [“Example User-Defined Report” on page 209](#).

### 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 FILENAME= argument of the <PreCode> element (for example, <PreCode filename="myMacroDefs.sas"/>). See [“The Report Template” on page 205](#).

For an example of a SAS Model Manager macro variable program, see [“Example User-Defined Report” on page 209](#).

For a list of SAS Model Manager macro variables, see [Appendix 3, “SAS Model Manager Macro Variables,” on page 441](#).

### 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**. Enter 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 window.

description="model-description"

specifies a description of the report that displays at the bottom of the New Reports window 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**


Deleting a User Report SAS Content Server is a two-step process. You must delete the SAS program and the report template.

To delete a user report, follow these steps:

1. Select **File** ⇒ **Browse** ⇒ **Browse SAS Templates**.
2. In the Browse Templates window, select the template. You can delete only templates that have a value of **No** in the **Reserved** column.
3. Click **Delete**.
4. In the SAS Model Manager Template Editor window, select **File** ⇒ **Browse** ⇒ **Browse SAS Files**.
5. 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**. The New Report window appears.
3. From the **Type** list box, select a user-defined report.
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 Models** table, select the appropriate model or models for the report.
6. In the **Style** list box, select a report output style. The default is **SAS default**. Other options are **Seaside**, **Meadow**, and **Harvest**.
7. 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*.

8. 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 NOdate;
%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:

```

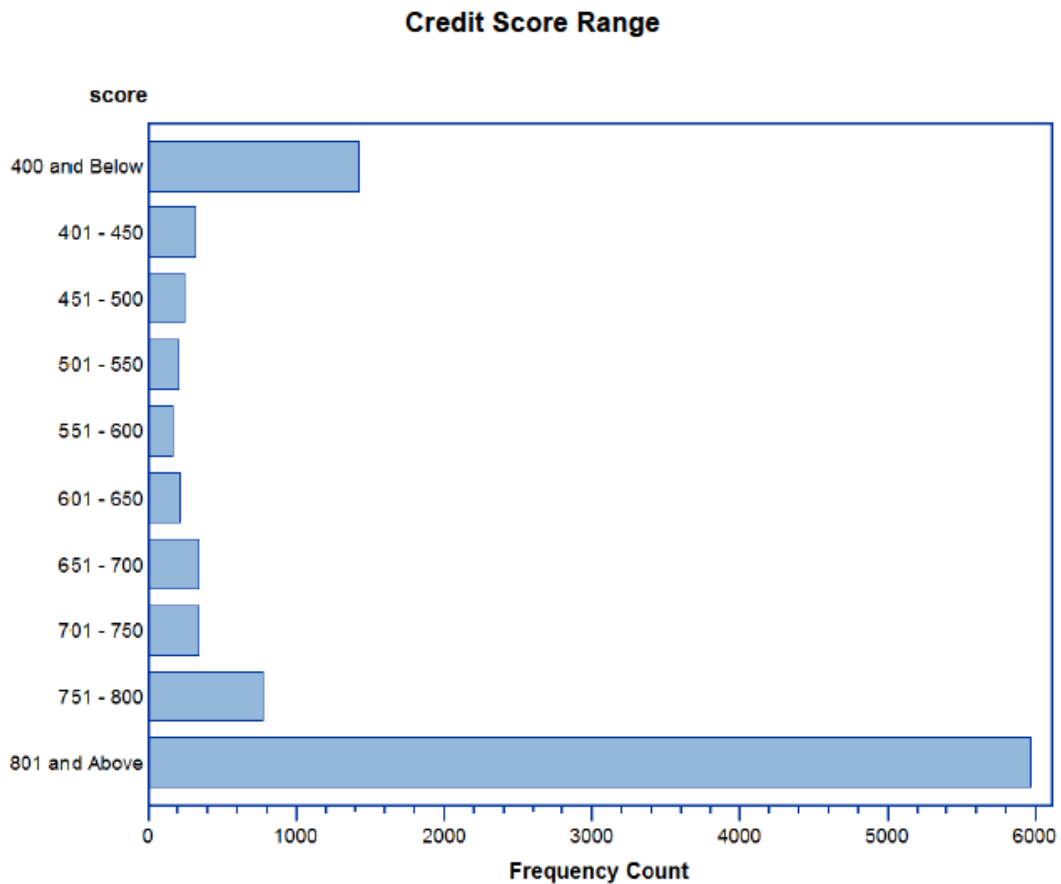
<?xml version="1.0" encoding="UTF-8" ?>
<ReportTemplate
  name="Score Range Report"
  type="UserDefinedReport"
  displayName="Score Range Report"
  description="Score Range Report"

```

```
>  
<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:







## Part 4

---

# Deploying and Publishing Models

<i>Chapter 12</i>	
<b>Deploying Models</b> .....	215
<i>Chapter 13</i>	
<b>Publishing Models</b> .....	223
<i>Chapter 14</i>	
<b>Replacing a Champion Model</b> .....	247



## Chapter 12

# Deploying Models

---

<b>Overview of Deploying Models</b> .....	<b>215</b>
<b>Champion Models</b> .....	<b>216</b>
About Champion Models .....	216
Requirements for a Champion Model .....	217
Set a Champion Model .....	217
Clear a Champion Model .....	218
<b>Challenger Models</b> .....	<b>219</b>
About Challenger Models .....	219
Flag a Challenger Model .....	219
Clear a Challenger Model .....	220
<b>Freezing Models</b> .....	<b>220</b>
About Freezing Models .....	220
Freeze a Version .....	221
Unfreeze a Version .....	221

---

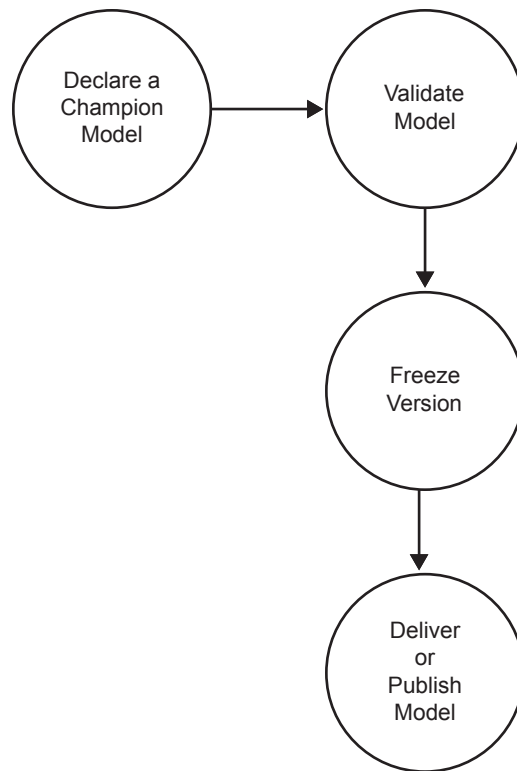
## Overview of Deploying Models

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. You can also flag challenger models for the champion model.
2. Test and validate the model before you declare the model ready for production.
3. Freeze the model version to prevent changes to the model.
4. Update your life cycle milestones.
5. Publish the champion model and challenger models (optional) so that you can deploy them to a production environment.

The following figure illustrates activities that might occur before a model is deployed.


**Figure 12.1** Deploying Analytical Models

*Note:* Prior to SAS Model Manager 12.1, all versions of a project could contain a champion model, and one of the tasks in deploying models was to set the version that contained the champion model as the default version for a project. Beginning in Model Manager 12.1, when you set a champion model, SAS Model Manager sets the version that contains the champion model as the default version. A project can have only one champion model. Therefore, the task of setting a default version has been eliminated.

---

## Champion Models

### About Champion Models

The champion model is the best predictive model that is chosen from a pool of candidate models. Before you identify the champion model, you can evaluate the structure, performance, and resilience of candidate models. You select the champion model from the models in a version. When a champion model is ready for production scoring, you set the model as the champion model. A check mark  appears next to the model and version that are associated with the champion model. The version that contains the champion model is considered the default version for the project. There can be only one champion model for a project.

You can publish the champion model to a database, the SAS Metadata Repository, and a SAS channel.

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

- [“Set a Champion Model” on page 217](#)
- [“Clear a Champion Model” on page 218](#)
- [Publish Models on page 224](#)

### **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, you cannot set the model as a champion model.

*Note:* An authorized user can reset the expiration date to a later date so that it is possible to set the champion model. To reset the expiration date, click the **Properties** tab for the 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. For example, you might use 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 flag and publish 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 because of the regulatory requirements.


#### **See Also**

[“Flag a Challenger Model” on page 219](#)

### **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 as Champion** from the pop-up menu. A dialog box appears. Click **Yes** to confirm.
3. If there are model input variables that are not defined as project input variables, you are prompted to add the input variables. Click **Yes** to confirm. The model input variables are copied to the project input variables. If project output variables are not defined, then the Select Project Output Variables window appears for you to select the output variables. After you select the output variables, click **OK**.

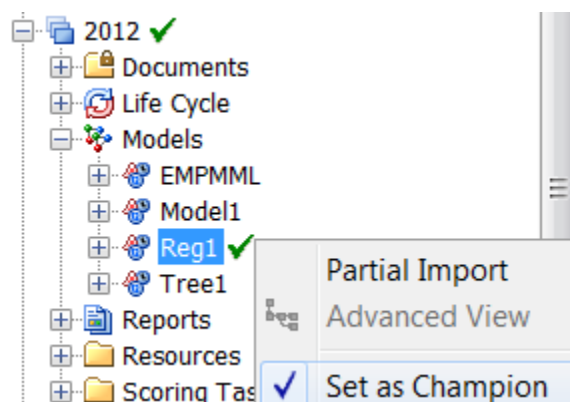
4. If the project output variables have not been mapped to the model output variables, the Set Model Output Mapping window appears. For each project variable, click the **Model Variables** field and select the model output variable. Click **OK**.
5. If you are replacing the champion model, follow these steps:
  - a. If there are no challenger models flagged, select **Yes** to replace the champion model. If a challenger model is flagged, go to the next step.
  - b. To keep the current challenger models, select **Keep models flagged as challengers**. If you do not select **Keep models flagged as challengers**, challenger models are cleared.
  - c. Click **Yes** to confirm replacing the current champion model.
  - d. If the project output variable has not been mapped to the model output variable, the Set Model Output Mapping window appears. For each project variable, click the **Model Variables** field and select the model output variable. Click **OK**.
6. Verify that the  icon appears beside the champion model. and the version
7. Select the version folder to examine its properties. The value for **Date Modified** is today's date. The value for the **Champion Model Name** is the name of the champion model.
 

*Note:* SAS Model Manager automatically annotates the **History** tab. To document the reasons for your selection of the champion model, use the version **Notes** tab.
8. 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 **Set as Champion** from the pop-up menu. A message box appears.



3. Click **Yes** to confirm. The **Set as Champion** check mark is cleared. SAS Model Manager also clears all challenger models and clears the version check mark.


*Note:* If the version is frozen, you cannot clear the champion model unless a SAS Model Manager administrator unfreezes the version first.

4. Select the version folder to examine its properties. Verify that the value for **Date Modified** property is today's date. The model that you just cleared as the champion model remains as the value for the **Champion Model Name** property until another champion model is set.
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**.

---

## Challenger Models

### About Challenger Models

You use challenger models to test the strength of champion models. The champion model for a project can have one or more challenger models. A model can be flagged as a challenger model only after a champion model for the project has been selected. A challenger model can be flagged in any version of a project. When you flag a model as a challenger model, the  icon appears next to the model.

Before you flag a model as the challenger model, verify that the model is active. If the model expiration date has passed, then you cannot set the model as a challenger model.

*Note:* An authorized user can reset the expiration date to a later date so that it is possible to set the challenger model.

To compare a challenger model to a champion model, you can create and run performance monitoring tasks for the champion model and any challenger models. Then, using the performance data, you can create a Champion and Challenger Performance report. You can also compare challenger models to the champion model using other reports such as the Delta report and Dynamic Lift report that are available through the New Report wizard. For more information, see [“Format a Champion and Challenger Performance Report” on page 308](#).


*Note:* The batch programs for performance monitoring do not support creating challenger model performance reports.


Challenger models can be published to a database, the SAS Metadata Repository, or to a SAS channel that has the champion model. They can also be published by themselves. If testing determines that the challenger model is the better model, you can replace the champion model by setting the challenger model as the champion model.

Here are the tasks that are associated with challenger models:



- [“Flag a Challenger Model” on page 219](#)
- [“Clear a Challenger Model” on page 220](#)
- [Publish Models on page 224](#)

### Flag a Challenger Model


1. Expand the **Models** folder under the version folder 
2. Right-click the model that you want to use as the challenger model and select **Flag as Challenger** from the pop-up menu. Click **Yes** to confirm.

3. If there are model input variables that are not defined as project input variables, you are prompted to add the input variables. Click **Yes** to confirm. The model input variables are copied to the project input variables.
4. If the project output variable has not been mapped to the model output variable, the Set Model Output Mapping window appears. For each project variable, click the **Model Variables** field and select the model output variable. Click **OK**.
5. Verify that the  appears beside the challenger model.

### Clear a Challenger Model

1. Expand the **Models** folder under the version folder .
2. Right-click the challenger model and select **Flag as Challenger**.
3. Click **Yes** to confirm. The **Flag as Challenger** check mark is cleared and the challenger flag icon  to the right of the model is cleared.

*Note:* If the version is frozen, then you cannot clear the challenger model unless a SAS Model Manager administrator unfreezes the version first.

Challenger models  can also be cleared when the champion model is cleared or replaced.

---

## 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, rename, 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 Tasks** 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 221
- “Unfreeze a Version” on page 221


If you attempt to delete a project that contains a frozen version, SAS Model Manager displays a message indicating that you cannot delete a project that contains frozen versions. You must have a SAS Model Manager administrator unfreeze the versions before the project can be deleted.

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

## Unfreeze a Version


If changes to a model are required after the version is frozen, 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 **Freeze Version** from the pop-up menu.



The check mark is cleared.

3. Verify that the  icon next to version folder does not contain a lock.
4. Select the version folder to examine its properties. The value for **Date Frozen** is cleared.

*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.

## Chapter 13

# Publishing Models

---

<b>Overview of Publishing Models</b> .....	<b>223</b>
<b>Publishing Models to a SAS Channel</b> .....	<b>224</b>
About Publishing Models to a SAS Channel .....	224
Publish a Model to a Channel .....	225
Extract a Published Model .....	228
<b>Publish Models to the SAS Metadata Repository</b> .....	<b>228</b>
About Publishing Models to the SAS Metadata Repository .....	228
Publish a Model .....	228
Publish a Project Champion Model .....	229
Verify the Model Publish .....	230
<b>Publishing Models to a Database</b> .....	<b>231</b>
About Publishing Models to a Database .....	231
Process Flow .....	233
Prerequisites for Publishing to a Database .....	234
Make User-Defined Formats Available When Publishing Models to a Database ..	235
Publish Models to a Database Field Descriptions .....	235
How to Publish Models to a Database .....	239
Log Messages .....	244
Scoring Function Metadata Tables .....	244
<b>Remove Models from a Database</b> .....	<b>244</b>
<b>View Publish History</b> .....	<b>246</b>

---

## Overview of Publishing Models

SAS Model Manager provides a comprehensive publishing environment for model delivery that supports sharing life cycle, performance, and scoring data. SAS Model Manager publishes models to different channels, and to the SAS Metadata Repository. SAS Model Manager can also publish classification, prediction, and segmentation (cluster) models with the score code type of DATA step 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.

The publish history of models can be viewed from the **MMRoot** folder, and the project or version level. You can also remove models that have been published to a database.

---

## Publishing Models to a SAS Channel

### *About Publishing Models to a SAS Channel*

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 **Models** 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:* Before you can deploy a model to a publications channel, a SAS administrator must configure the publication channel in SAS Management Console 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, an HTTP server, or a path in WebDAV.

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 of inputvar.xml
sas002.ref	URL of outputvar.xml
sas003.ref	URL 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 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 225
- “Extract a Published Model” on page 228

## Publish a Model to a Channel

To publish a model to a channel:

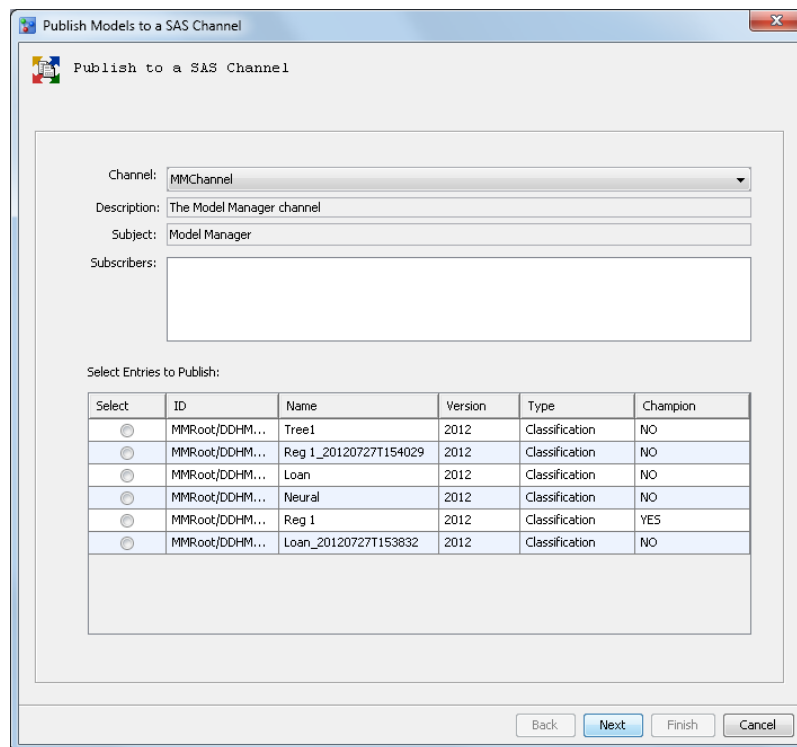
1. Right-click the organizational, project, version, or **Models** folder that contains the model that you want to publish, and select one of the following menu options to display the Publish to a SAS Channel window:

- Select **Publish Models to a SAS Channel** for an organizational or version folder.

*Note:* If you select an organization folder, all of models for each project within a control group are also included in the list of models.

- Select **Publish Models ⇨ to a SAS Channel** for a project folder.
- Select **Publish to a SAS Channel** for the **Models** folder.

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



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 selected folder. To view the entire filepath for the location of the model, expand the **ID** column heading.
4. Click **Next**.

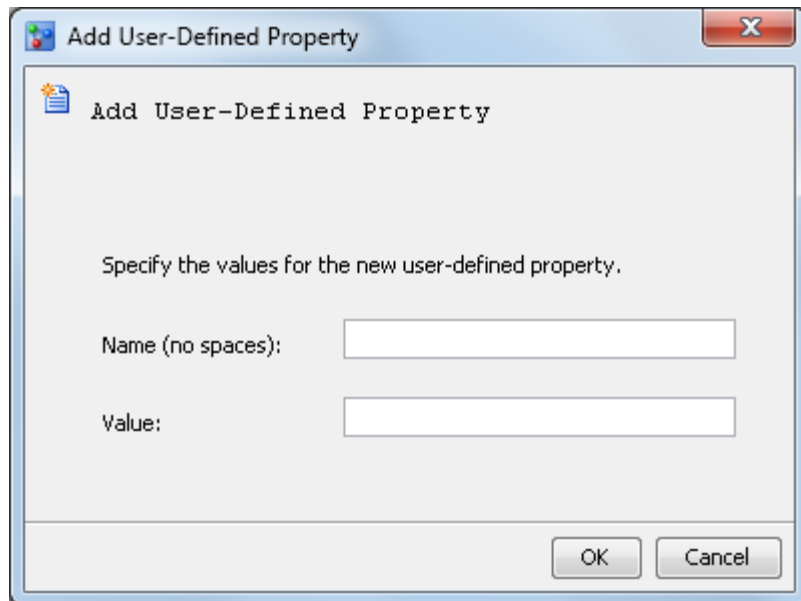
The screenshot shows a dialog box titled "Publish Models to a SAS Channel". It contains the following elements:

- Message Subject:** A text input field.
- Notes:** A larger text area for additional information.
- Add User-Defined Property:** A table with two columns: "Name" and "Value".
- Buttons:** "Back", "Next", "Finish", and "Cancel" at the bottom right.

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**. The Add User-Define Property window appears.



- b. Specify the property name and its value. For example, specify **CreditScoreModelCode** for the property name and **V3test** for its value. The property name can contain letters, an underscore ( \_ ), and a hyphen ( - ). However, a hyphen ( - ) cannot be the first character.
- 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 appears 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. (Optional) Update the specific properties for the appropriate milestone task in the **Life Cycle** node.
11. (Optional) Add a user-defined property to identify the publication channel:
  - a. Select the model. Right-click in the Properties view and select **Add User-Defined Property**.
  - b. In the **Name** field, enter a property name that depicts the channel name that was used to publish the model, such as **PublishedToChannel**. The property name can contain letters, an underscore ( \_ ), and a hyphen ( - ). However, a hyphen ( - ) cannot be the first character.
  - c. In the **Value** field, enter the channel that was used to publish the model. Click **OK**.

If you extract a model using the `%MM_GetModels()` macro to create performance reports, you must know the name of 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 294](#).

---

## **Publish Models to the SAS Metadata Repository**

### **About Publishing Models to the SAS Metadata Repository**

SAS Model Manager publishes a model by creating a MiningResults object in the SAS Metadata Repository. You can use the model information in the MiningResults object to set up a scoring environment. A scoring application can 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. Therefore, when you publish a project champion model, challenger model, or other models (with proper configuration), the scoring application always uses the most current champion model. Only the project champion model can be published from the project level and from the project control group level.

*Note:* SAS Model Manager cannot publish R models.

SAS Model Manager uses the SAS Folders view to publish 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 publish models:

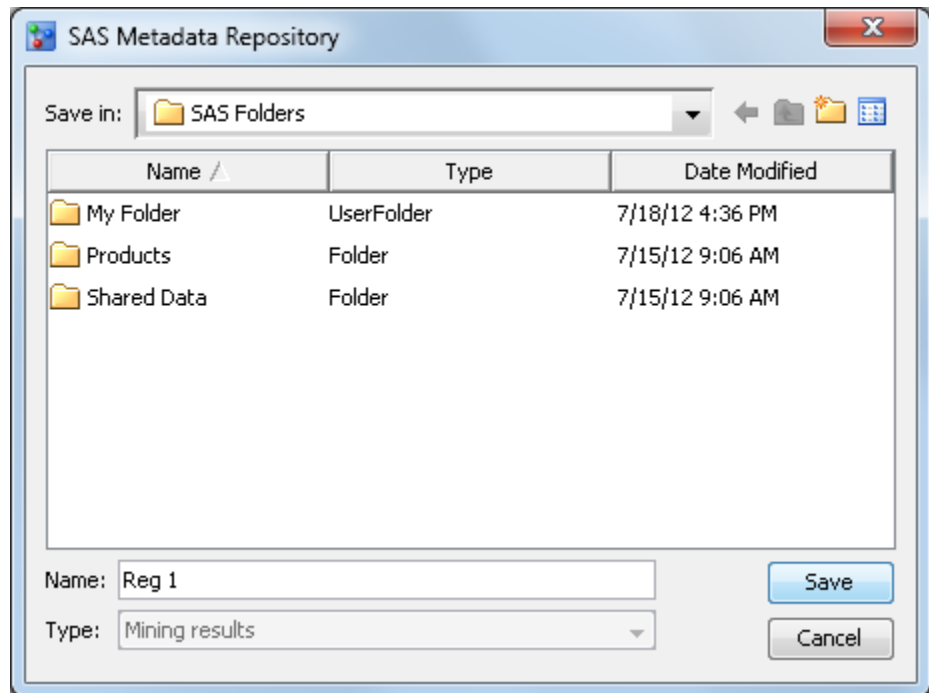
- [“Publish a Model” on page 228](#)
- [“Publish a Project Champion Model” on page 229](#)
- [“Verify the Model Publish” on page 230](#)

### **Publish a Model**

To publish a model:



1. Right-click the model that you want to publish and select **Publish Model**. The SAS Metadata Repository window appears. SAS Model Manager displays metadata folders.



2. Select the folder to store the model. You must have Write permission to this folder.  
*Note:* If you created your own subfolders, then select a folder to store the metadata for the published model.
3. Enter the name for the model and click **Save**. If a MiningResults 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. In addition, if a MiningResults object is in the repository that has the same model UUID, you are asked whether you want to continue with publishing the model.

*Note:* Do not overwrite an existing MiningResults 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 publish the model again to ensure that your score application uses the current scoring code. If you change the status of model that has been published, then it is recommended that you publish the model again. For example, flag or clear a model as a challenger, or set or clear a model as champion.

### **Publish a Project Champion Model**

To publish the champion model for a project, you must have already set a model as the champion. When a champion model is selected, the version that contains the model is automatically set as the default version for the project. SAS Model Manager examines the project and always publishes the project champion model. When the champion model for a project changes and you publish the model again to the same location, the scoring application automatically uses the latest score code.

*Note:* SAS Model Manager cannot publish R models.

To publish the champion model for a project:

1. Verify that the project that you want to publish has a champion model assigned. Select the project folder to examine its properties. The **Default Version** property contains the name of the version that contains the champion model.
2. Right-click the project name and select **Publish Models** ⇒ **to the SAS Metadata Repository**. If the project is not locked, you are prompted to confirm publishing the champion model. The SAS Metadata Repository window appears. SAS Model Manager displays the metadata folders.
3. Select a folder that you want to publish the model to.

*Note:* A champion model can be published only to a folder and you must have permissions to access the folder on the SAS Metadata Repository. If a folder is selected when the SAS Metadata Repository window appears, navigate to a list that contains your folder and select the folder.

4. Click **OK**. If a MiningResults object is in the SAS Metadata Repository that has the same name or model UUID, then you are asked whether to overwrite the metadata for this stored object.

*Note:* Do not overwrite an existing MiningResults object unless you are certain that the model is from the same project in SAS Model Manager.

### Verify the Model Publish

To verify that SAS Model Manager successfully created the MiningResults object in the SAS Metadata Repository for the published model, use SAS Management Console. To view the contents of the published model, you can use SAS Data Integration Studio. You can also use SAS Management Console to export the MiningResult objects to a SAS package.

*Note:* You can also view the publish history of models in the MMRoot, project, or version object details view. For more information, see [“View Publish History” on page 246](#).

To view a MiningResult object in the SAS Metadata Repository:

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 published the model. When you select the folder, the right pane lists the MiningResult objects for the published models.
3. Right-click the MiningResults object that has the name of published 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 MiningResults object contains the UUID of the published project or model.

*Note:* You can use the UUID to conduct filtered searches and query the published models. For more information, see [Query Folders, Projects, and Versions on page 393](#).

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

---

## Publishing Models to a Database

### *About Publishing Models to a Database*

SAS Model Manager enables you to publish the project champion model and challenger models that are associated with the **DATA Step** score code type to a configured database. SAS Model Manager uses the SAS Scoring Accelerator and SAS/ACCESS interface to the database to publish models to the database. The Scoring Accelerator takes the models from SAS Model Manager and translates them into scoring files or functions that can be deployed inside the database. After the scoring functions are published using the SAS/ACCESS interface to the database, the functions extend the database's SQL language and can be used in SQL statements such as other database functions. After the scoring files are published, they are used by the SAS Embedded Process to run the scoring model.

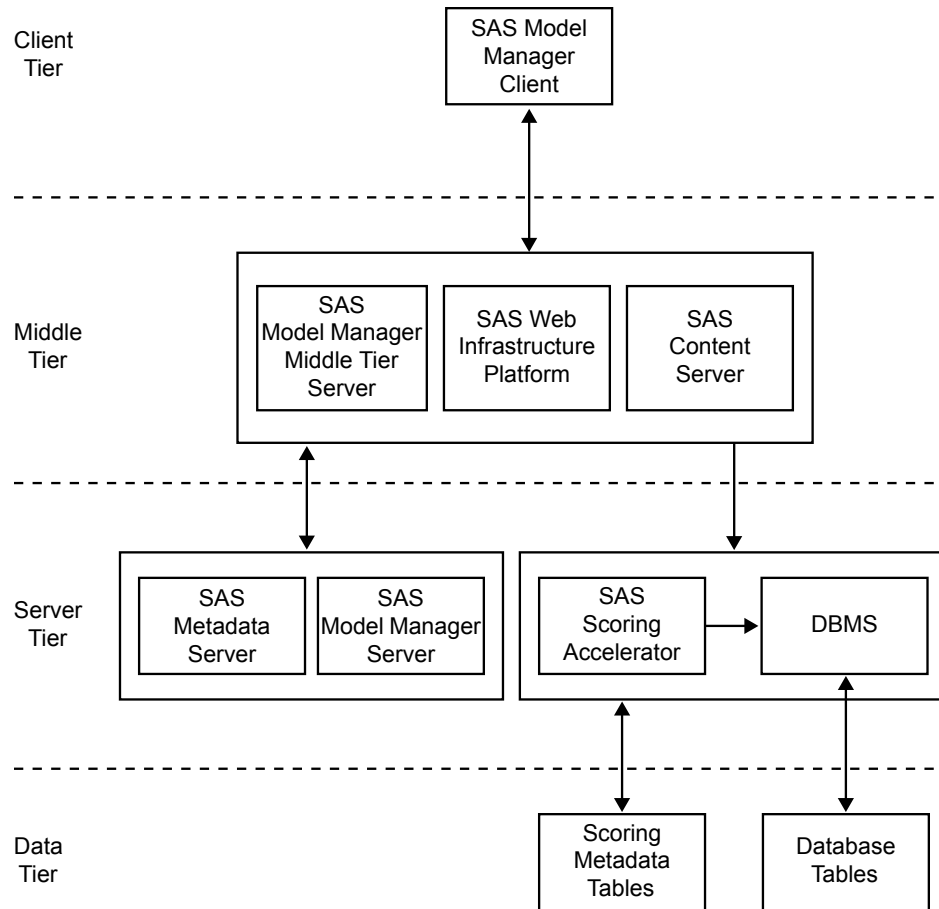
If the scoring function publish method is chosen, the scoring metadata tables in the database are populated with information about the project and pointers to the scoring function. This feature enables users to review descriptions and definitions of the published model. The audit logs track the history of the model's usage and any changes that are made to the scoring project.

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

*Note:* For more information about the prerequisites before publishing models to a database, see [“Prerequisites for Publishing to a Database”](#) on page 234.

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 12.3 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 project 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 or model files that can be deployed inside a database. The scoring functions or model files are based on the project's champion model score code or challenger 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 are populated in the database when you publish a scoring function in SAS Model Manager.

#### Database Tables

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

## Process Flow

This is an example of the process flow to publish a scoring model to a database. For more information, see [“How to Publish Models to a Database” on page 239](#).

1. From SAS Model Manager, you select the **Publish Models** ⇒ **to a Database** for the project that contains the champion model or challenger model that you want to publish to a specific database. For more information, see [“How to Publish Models to a Database” on page 239](#).
2. After you select the publish method and complete all the required information to publish the model to a database, SAS Model Manager establishes a connection to the database using the credentials that were entered. The publish name is validated against the target database. If the publish name is not unique, an error message is displayed.
3. 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.
4. 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 or model files, and publishes the scoring functions or model files 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 published model is used as the scoring input data set during validation.
  - scores the model with the new scoring function or model files using the new scoring table.
  - compares scoring results.
5. The middle-tier server parses the SAS Workspace Server logs to extract the return code.
  6. The middle-tier server updates the scoring metadata tables (for example, table `project_metadata`). For more information see, “[Scoring Function Metadata Tables](#)” on page 244.  
*Note:* This step is performed only for the scoring function publish method and the metadata usage option is enabled in SAS Management Console.
  7. The middle-tier server then creates a history entry in the SAS Model Manager project history.
  8. The middle-tier server updates the project user-defined properties with the publish name that was entered in the Publish Models to a Database window.  
*Note:* For more information about the user-defined properties that are created when publishing, see “[SAS User-Defined Properties](#)” on page 510.
  9. A message indicates that the scoring function or model files 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 a folder that is created for the publish model in the **Publish Results** folder in the Project Tree.

### Prerequisites for Publishing to a Database

The following prerequisites must be completed before users can publish a model scoring function using the scoring function publish method, or publish a model’s scoring files using the SAS Embedded Process publish method:

- The user must have the proper authorization to publish approved models from SAS Model Manager to the database for SAS In-Database scoring.
- The champion model for the project 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 can publish only the models that are associated with the **DATA step** score code type to a database. Models with a score code type of **SAS Program** or **PMML** cannot be published to a database.

- A database must have been configured to install scoring functions or model scoring files.
- If the model contains user-defined formats, a file that contains the user-defined formats must be attached to the **Resources** folder.
- The following prerequisites are only for the scoring function publish method.

- (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 when using the scoring function publish method.
- The scoring function metadata tables are required in the target database if the **Metadata usage** option is enabled in SAS Management Console.

### **Make User-Defined Formats Available When Publishing Models to a Database**

In order to publish models with user-defined formats to a database using the Publish Models to a Database feature, you must make the user-defined formats available to SAS Model Manager.

To make the user-defined formats available for publishing:

1. Translate the user-defined formats SAS data set (formats.sas7bcat) that was created with the model into a formats.cport file.

Here is an example:

```
filename tranfile "C:\formats.cport";
libname source "C:\myformats";

proc cport library=source file=tranfile memtype=catalog;
run;
quit;
```

2. Attach the formats.cport file to the **Resources** folder within the version that contains the project champion model or challenger models.
3. Send a request to the SAS administrator and ask them to either put the user-defined formats SAS data set (formats.sas7bcat) in the \\SASConfigDirectory\Lev1\SASApp\SASEnvironment\SASFormats directory or add the LIBNAME definition for the formats library to the \\SASConfigDirectory\Lev1\SASApp\appserver\_autoexec\_usermods file.

Here is an example of a LIBNAME definition:

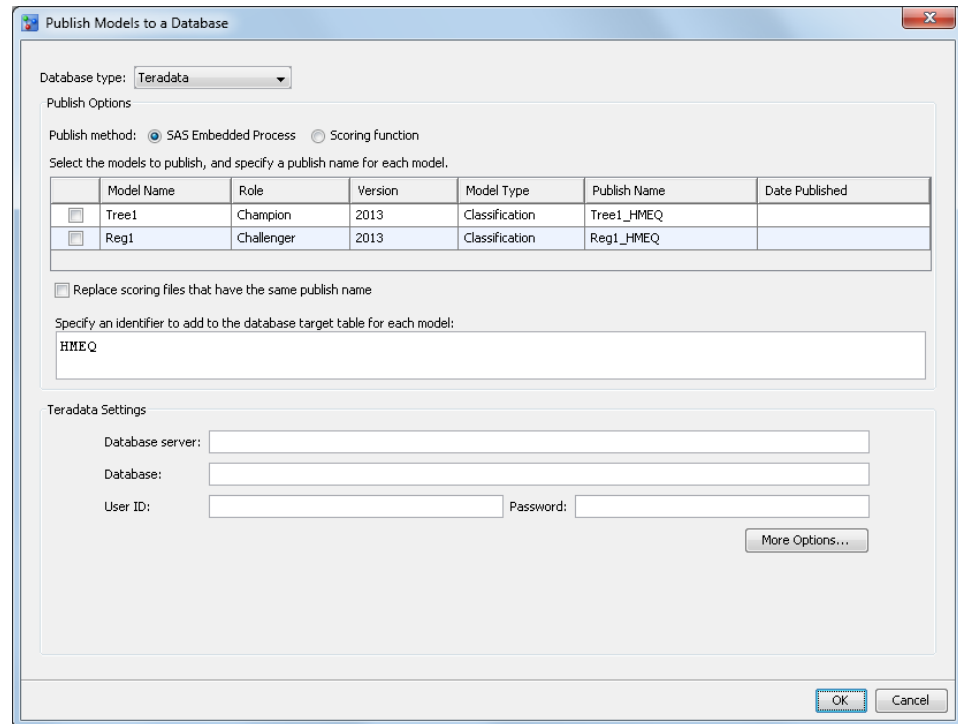
```
libname mylib "C:\myformats";
options fmtsearch = (mylib.formats);
```

### **See Also**

*SAS In-Database Products: User's Guide*

### **Publish Models to a Database Field Descriptions**

Here is a list of the field names and descriptions for the Publish Models to a Database window.



### Database type

specifies the type of database to which the scoring function or model scoring files are published.

### Publish method

specifies the method to use when publishing the scoring function or model files to the database.

### Publish name

specifies the name to use when publishing a scoring function or model files to the database. The publish name is a user-defined value that can be modified. The SAS Embedded Process publish method uses the **publish name** as the model name to publish the model files to the database. The scoring function publish method has a system-generated **prefix** and the **publish name** that makes up the scoring function name. These are used to publish the model scoring function.

The prefix portion of the scoring function name 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 on which you selected the **Publish Models** ⇒ **to a Database** menu option. An example of a function name is **Y081107001\_user\_defined\_value**. Here are the naming convention requirements:



Here are the naming convention requirements for the publish name:

- The user-defined value is case insensitive. The maximum length of alphanumeric characters is determined by the database type and publish method that is selected. No spaces are allowed. An underscore is the only special character that can be included in the publish name.
- The recommended maximum lengths of the publish name for the scoring function publish method are the following:
  - 19 alphanumeric characters for Teradata
  - 32 alphanumeric characters for Netezza, Greenplum, and DB2

#### *UNIX Specifics*

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

- The recommended maximum length of the publish name for the SAS Embedded Process publish method is 32 alphanumeric characters for all database types. The database types that are currently supported by SAS Model Manager are Teradata, Oracle, Greenplum, and DB2.

The value of the publish name is validated against the target database, when the option **Replace scoring files that have the same publish name** is not selected for the SAS Embedded Process publish method. If the publish name is not unique, an error message is displayed.

#### **Replace scoring files that have the same publish name**

specifies to replace the model scoring files that have the same publish name when you are using the SAS Embedded Process publish method. The value of the publish name is validated against the target database when this option is not selected. If the publish name is not unique, an error message is displayed.

#### **Specify an identifier to add to the database table for each model**

specifies the value of the identifier that is added to each model in the database so that the Database administrator or other users can query the database. The default value is the project name. This option is available only for the SAS Embedded Process publish method.

#### **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, Oracle, and DB2)**

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

**Options****Validate scoring results**

specifies to validate the scoring results when publishing a model scoring function or model scoring files. 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 or model scoring files 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**(scoring function) or **Keep scoring files if validation fails** (SAS Embedded Process)

specifies to save the scoring function or model scoring files if the validation of the scoring results fails. Saving the scoring function or model scoring files is useful for debugging if validation fails.

**Use model input**

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

Here are the limitations for the number of model input variables when publishing a champion model or challenger model to a database:

Database Type	SAS Embedded Process	Scoring Function
Teradata	If you use Teradata version 13.1 or 14.0, the maximum is 1024. If you use the SAS Embedded Process and Teradata version 14.10, the maximum is 2048.	128
Netezza	1600	64

Database Type	SAS Embedded Process	Scoring Function
Oracle	1000	Not applicable
Greenplum	1660	100
DB2	The maximum depends on the page size of the database table space. For a 4K page size database, the limit is 500. If you have it configured for any of the larger page sizes (8K, 16K, 32K), then the limit is 1012.	90

#### **Protected mode** (Teradata only)

specifies the mode of operation to use when publishing a model using the scoring function publish method. 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 and Netezza only)

specifies the mode of operation to use when publishing a model using the scoring function publish method. 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 or scoring model files.

#### **Sample size**

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

## **How to Publish Models to a Database**

To publish a model to a database:

1. Verify that you have set the champion model and challenger models that you want to publish. For more information, see [“Set a Champion Model” on page 217](#).

- (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 can 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 or model files, 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 508.

<input type="checkbox"/> User-Defined Properties	
BusinessContext	
<b>DbmsTable</b>	

- Right-click the project's name in the Project Tree and select **Publish Models** ⇒ to a **Database** from the pop-up menu. The Publish Models to a Database window appears.

Database type: Teradata

Publish Options

Publish method:  SAS Embedded Process  Scoring function

Select the models to publish, and specify a publish name for each model.

	Model Name	Role	Version	Model Type	Publish Name	Date Published
<input type="checkbox"/>	Tree1	Champion	2013	Classification	Tree1_HMEQ	
<input type="checkbox"/>	Reg1	Challenger	2013	Classification	Reg1_HMEQ	

Replace scoring files that have the same publish name

Specify an identifier to add to the database target table for each model:  
HMEQ

Teradata Settings

Database server:

Database:

User ID:  Password:

More Options...

OK Cancel

- Select a database type and publish method. The type of database and the publish method that you choose determine which database settings and options are required.

#### *Operating Environment Information*

The SAS Embedded Process can be used with the SAS Scoring Accelerator for Netezza to run scoring models with the release of SAS 9.4. The SAS Administrator can enable Netezza support for SAS Model Manager so that the Netezza database type appears when using the SAS Embedded Process publish method. For more information, see the *SAS Model Manager: Administrator's Guide*.

- Select the check box next to the models that you want to publish.

6. Enter a publish name for each model that you selected to publish. The scoring function publish method has a system-generated **prefix** and the **publish name**. These are used to publish the model scoring function. The SAS Embedded Process publish method uses only the **publish name** to publish the model files to the database. The publish name is a user-defined value that can be modified.

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 and publish method that is selected. No spaces are allowed. An underscore is the only special character that can be included in the publish name.
- The recommended maximum lengths of the publish names for the scoring function publish method are the following:
  - 19 alphanumeric characters for Teradata
  - 32 alphanumeric characters for Netezza, Greenplum, and DB2

#### *UNIX Specifics*

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

- The maximum length of the publish name for the SAS Embedded Process publish method is 128 alphanumeric characters for all databases. However, the recommended maximum length of the publish name is 32 alphanumeric characters for all database types. The database types that are currently supported by SAS Model Manager are Teradata, Oracle, Netezza, Greenplum, and DB2.

*Note:* The publish name for each model is reserved by default for subsequent use of the publishing models for a project.

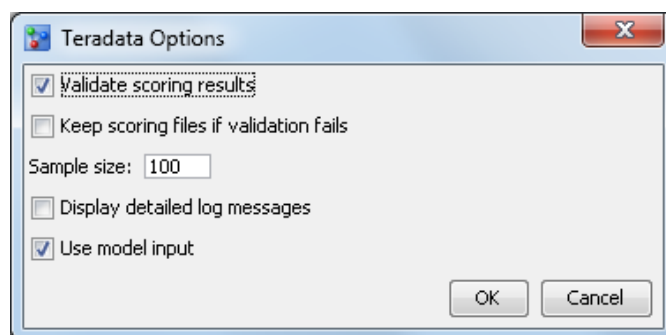
7. (Optional) Select the **replace scoring files that have the same publish name** check box. This option is available only for the SAS Embedded Process publish method.
8. Specify an identifier to add to the database target table for each model. The default value is the project name. This option is available only for the SAS Embedded Process publish method.
9. Enter a value for the database settings that appear for the selected database type.

Here are the available database settings according to the publish method and database type:

Database Settings	SAS Embedded Process	Scoring Function
<b>Database server</b>	<ul style="list-style-type: none"> <li>• Teradata</li> <li>• Oracle</li> <li>• Netezza</li> <li>• Greenplum</li> <li>• DB2</li> </ul>	<ul style="list-style-type: none"> <li>• Teradata</li> <li>• Netezza</li> <li>• Greenplum</li> <li>• DB2</li> </ul>
<b>Database</b>	<ul style="list-style-type: none"> <li>• Teradata</li> <li>• Oracle</li> <li>• Netezza</li> <li>• Greenplum</li> <li>• DB2</li> </ul>	<ul style="list-style-type: none"> <li>• Teradata</li> <li>• Netezza</li> <li>• Greenplum</li> <li>• DB2</li> </ul>

Database Settings	SAS Embedded Process	Scoring Function
<b>User ID</b>	<ul style="list-style-type: none"> <li>• Teradata</li> <li>• Oracle</li> <li>• Netezza</li> <li>• Greenplum</li> <li>• DB2</li> </ul>	<ul style="list-style-type: none"> <li>• Teradata</li> <li>• Netezza</li> <li>• Greenplum</li> <li>• DB2</li> </ul>
<b>Password</b>	<ul style="list-style-type: none"> <li>• Teradata</li> <li>• Oracle</li> <li>• Netezza</li> <li>• Greenplum</li> <li>• DB2</li> </ul>	<ul style="list-style-type: none"> <li>• Teradata</li> <li>• Netezza</li> <li>• Greenplum</li> <li>• DB2</li> </ul>
<b>Server user ID</b>	Not applicable	DB2
<b>Compile database</b>	Not applicable	Netezza
<b>Jazlib database</b>	Not applicable	Netezza
<b>Schema</b>	<ul style="list-style-type: none"> <li>• Oracle</li> <li>• Greenplum</li> <li>• DB2</li> </ul>	<ul style="list-style-type: none"> <li>• Greenplum</li> <li>• DB2</li> </ul>
<b>Initial wait time</b> (in seconds)	Not applicable	DB2
<b>FTP time out</b> (in seconds)	Not applicable	DB2

10. Click **More Options**. The Database Options window appears.



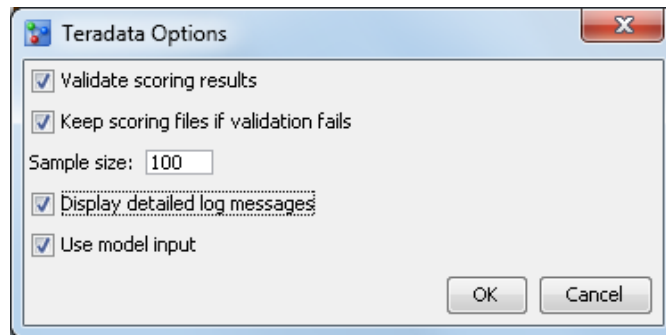
Select the check box for the desired validation options that appear for the selected database type:

- **Validate scoring results**
- **Keep scoring files if validation fails** (SAS Embedded Process) or **Keep scoring function if validation fails** (scoring function)
- **Display detailed log messages**

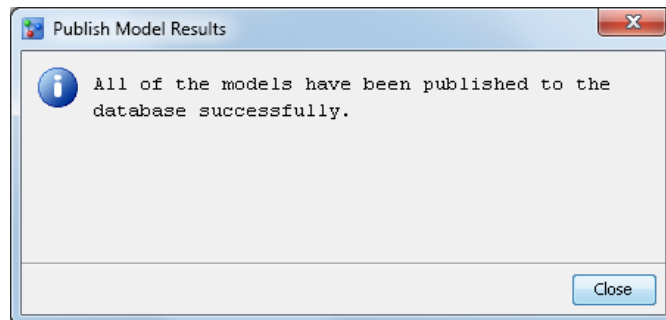
- **Use model input**
- **Protected mode** (Teradata scoring function option) or **Fenced mode** (DB2 and Netezza scoring function option)

*Note:* By default, the **Validate scoring results** and **Use model input** options are selected for both publish methods. The **Protected mode** or the **Fenced mode** options are selected by default for the scoring function publish method.

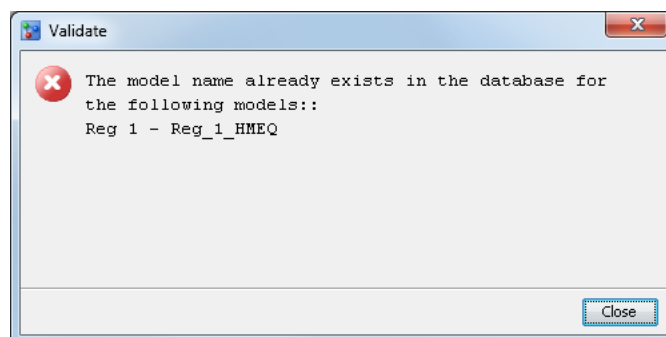
11. Enter a numeric value for **Sample Size**. The default value for sample size is 100 if the value is null or zero. The maximum number of digits that are allowed is 8.



12. Click **OK**. A message is displayed to indicate whether the models were published to the database successfully or not.



*Note:* The value of the publish name is validated against the target database, when the option **Replace scoring files that have the same publish name** is not selected for the SAS Embedded Process publish method. If the publish name is not unique, an error message is displayed.



13. Click **Close** to complete the publishing process.

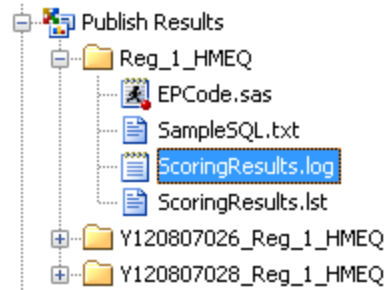
### See Also

- [“Remove Models from a Database” on page 244](#)

- [“View Publish History” on page 246](#)

## Log Messages

A user can view the log file after publishing a scoring model to a database. The **Publish Results** folder in the Project Tree contains a folder for each model that was published. The publish name is used to create a folder for each model that is published. That folder contains the ScoringResults.log file. The time at which the process started, details about who initiated the process, and the time at which 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

If the metadata tables are created and configured for use in SAS Management Console, the following tables are populated 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.

---

## Remove Models from a Database

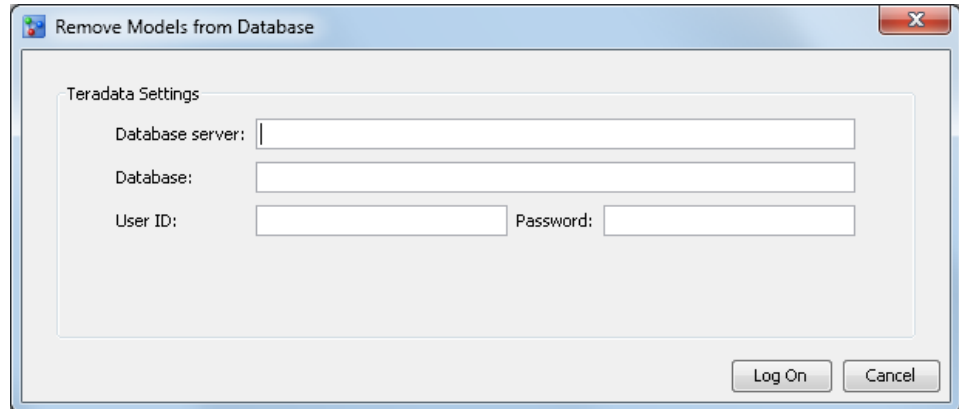
The SAS Embedded Process publish method enables you to replace the model scoring files, but the scoring function publish method publishes the model as a separate entry in the database each time. The Remove Models from a Database feature enables you to remove previously published models, so that you can clean up the test or production



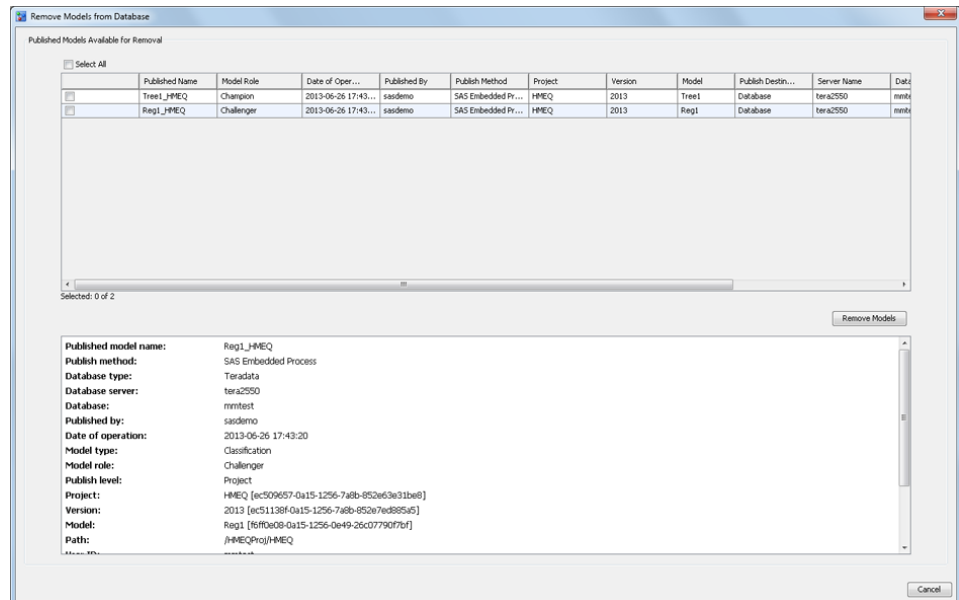
database. After you have published models to a database, if you modify the previously published models or change the champion model or challenger models, you can remove them to clean up the database for future publishing of models.

To remove models from a database:

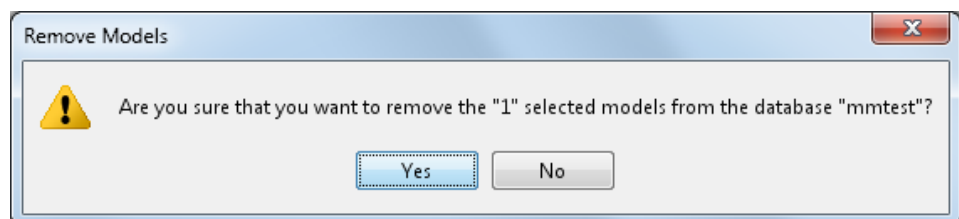
1. Select a project and then select **Tools** ⇒ **Remove Models from Database** ⇒ **<database type>** from the menu toolbar. The Remove Models from Database window appears.



2. Enter the database settings and click **Log On**. The **Published Models Available for Removal** list is displayed.



3. Select from the list the models that you want to remove from the database. When you select a model from the list, the publish details are displayed in the box at the bottom of the window.
4. Click **Remove Models**. A warning message appears.



5. Click **Yes** to remove the models from the database.
6. Click **Close** in the success message that is displayed.
7. Click **Cancel** to exit the tool.

## View Publish History

To view the publish history of models, select the **MMRoot** folder, a project or a version from the Project Tree, and then select the **Publish History** tab in the details section. All models that have been published to a SAS Channel, to the SAS Metadata Repository, and to a database are displayed. Select an item from the list to view the full publish details.

HMEQ					
Properties	Input Variables	Output Variables	Publish History		
Published Name	Publish Destination	Publish Method	Operation Status	Server Name	Database Name
HMEQ	SAS Metadata Repository		Published	RDCESX09147.ra...	
Tree1_HMEQ	Database	SAS Embedded Process	Modified	tera2550	mmtest
Reg1_HMEQ	Database	SAS Embedded Process	Modified	tera2550	mmtest
Tree1	SAS Channel		Published	RDCESX09147.ra...	
Reg1_HMEQ	Database	SAS Embedded Process	Removed	tera2550	mmtest

<b>Published model name:</b>	Tree1_HMEQ
<b>Publish method:</b>	SAS Embedded Process
<b>Operation status:</b>	Modified
<b>Database type:</b>	Teradata
<b>Database server:</b>	tera2550
<b>Database:</b>	mmtest
<b>Published by:</b>	sasdemo
<b>Date of operation:</b>	2013-06-26 17:43:00
<b>Model type:</b>	Classification
<b>Model role:</b>	Champion
<b>Publish level:</b>	Project
<b>Project:</b>	HMEQ [ec509657-0a15-1256-7a8b-852e63e31be8]
<b>Version:</b>	2013 [ec51138f-0a15-1256-7a8b-852e7ed885a5]
<b>Model:</b>	Tree1 [ec51cbe6-0a15-1256-7a8b-852e32d3a8de]
<b>Path:</b>	/HMEQProj/HMEQ
<b>User ID:</b>	mmtest
<b>Validate scoring results:</b>	Y
<b>Keep scoring files if validation fails:</b>	Y
<b>Sample size:</b>	100
<b>Display detailed log messages:</b>	Y
<b>Use model input:</b>	Y

## Chapter 14

# Replacing a Champion Model

---

Overview of Replacing a Champion Model . . . . .	247
Retire a Project . . . . .	248

---

## 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 new 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.	Set the model as the champion model.

---

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.	Set the project <b>State</b> property to be inactive. After a new model is ready, do the following: <ul style="list-style-type: none"> <li>• Set the model as the champion model.</li> <li>• 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.</li> <li>• Set the project <b>State</b> property to be active.</li> </ul>
No other champion models are planned for the project and the project is to be retired.	Set the project <b>State</b> property to be retired. (Optional) Set the default version life cycle retire status as completed.

---

## Retire a Project

To retire a project:

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>	
<b>What is Performance Monitoring?</b> .....	251
<i>Chapter 16</i>	
<b>Create Reports by Defining a Performance Task</b> .....	263
<i>Chapter 17</i>	
<b>Create Reports Using Batch Programs</b> .....	277
<i>Chapter 18</i>	
<b>Formatting Performance Reports</b> .....	305
<i>Chapter 19</i>	
<b>Using Dashboard Reports</b> .....	313
<i>Chapter 20</i>	
<b>Retraining Models</b> .....	327



## Chapter 15

# What is Performance Monitoring?

---

<b>Overview of Performance Monitoring</b> .....	<b>251</b>
<b>Types of Performance Monitoring</b> .....	<b>252</b>
Overview of the Types of Performance Monitoring .....	252
Summary Results .....	253
Data Composition Reports .....	254
Model Monitoring Reports .....	256
<b>Performance Index Warnings and Alerts</b> .....	<b>259</b>
<b>The Process of Monitoring Champion Models</b> .....	<b>260</b>

---

## Overview of Performance Monitoring

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 tasks for each time interval. After you create and execute the performance monitoring tasks, you can view the performance data through report charts in SAS Model Manager that give a graphical representation of the model's performance. SAS Model Manager enables you to create performance monitoring reports in PDF, HTML, RTF, and Excel output formats from the **Reports** node.

*Note:* Performance monitoring is designed to work only with a project that is associated with a classification model function and has a binary target, or a prediction model function and has an interval target. Only models that are associated with the classification and prediction model types and are set as champion and challenger models can be monitored for performance.

SAS Model Manager provides the following types of output for performance monitoring:

- 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 output, 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 performance output 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 and execute the performance task, you view the report charts in the SAS Model Manager window by selecting the **Performance** node in the default champion model's 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.

If you have flagged a challenger model to compare with the champion model, you can use the performance data that you collected for the champion model to create reports for the challenger model. After all of the performance monitoring tasks have been run, you can use the New Report window to create a Champion and Challenger Performance report that compares the champion model to the challenger model.

## Types of Performance Monitoring

### Overview of the Types of Performance Monitoring

After a champion model is in production, you can monitor the performance of the model by analyzing the SAS Model Manager performance results. You can create the performance output interactively using the Define Performance Task wizard and the **Performance Monitor** 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 output:

#### Summary Results

The **Summary** results 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 results from the Annotations section in the Project category view.

#### 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 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
- Gini - ROC\*
- Gini - Trend
- KS
- MSE (Mean Squared Error) for prediction models

\* 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 Monitoring 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 Results

The Summary results summarizes the contents of different organizational folders, projects, and versions.

The contents of the Summary results 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 results, click the **Summary** tab that is in the Annotations section of the Projects category view. Then click .

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 models, the number of versions, the number of scoring tasks, and the number of candidate models.

Candidate models are models that are not currently in production.

### 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.

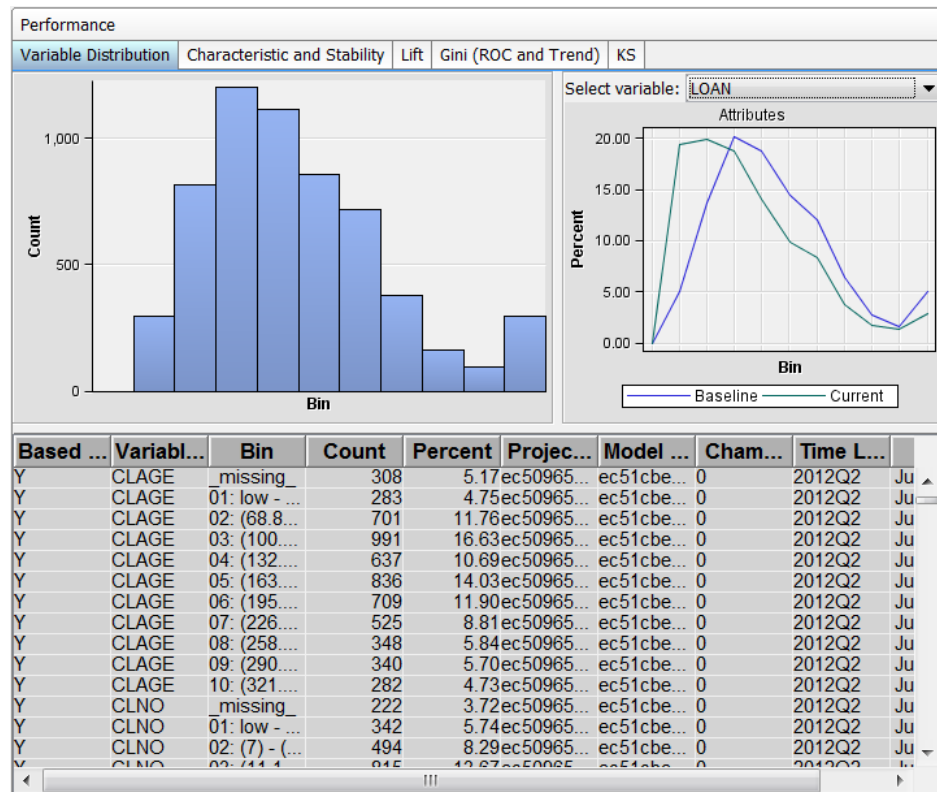
## Data Composition Reports

### Variable Distribution Report

When you select the **Performance** node for a version in the Project Tree, the **Variable Distribution** report appears with two charts. The variable distribution chart on the left is a histogram. This histogram is a graphical representation of the distribution of a selected variable from the training data set. The Y-axis is the count of observations in a bin.

The variable distribution chart on the right is a line chart. The chart is a graphical representation of two distributions of a selected variable from the training data set and the current data set, respectively. The Y-axis is the percentage of observations in a bin that is proportional to the total count.

To change the variable that appears in the two charts, click the **Select variable** box and select a variable.



### Overview of Characteristic and Stability Reports

Together, the Characteristic and Stability reports detect and quantify shifts that can occur in the distribution of model performance data, scoring input data, and the scored output data that a model produces.

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 assess the model's sensitivity to the distribution shift in the input data, in terms of the predictive performance of the scoring input variables.

The Characteristic report indicates changes to the scope and composition of the submitted data sets over time, and 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:

$$\text{Deviation\_Index} = \Sigma (\%Actual - \%Expected) * 1n (\%Actual / \%Expected)$$

Numeric predictor variable values are placed into bins for frequency analysis. Outlier values are removed to facilitate better placement of values and to avoid scenarios that can aggregate most observations into a single bin.

If the training data set and the current data set have identical distributions for a variable, the variable's deviation index is equal to 0. A variable with a deviation index value that is  $P1 > 2$  is classified as having a mild deviation. The Characteristic report uses the performance measure P1 to count the number of variables that receive a deviation index value that is greater than 0.1.

A variable that has a deviation index value that is  $P1 > 5$  or  $P25 > 0$  is classified as having a significant deviation. A performance measure P25 is used to count the number of variables that have significant deviations, or the number of input variables that receive a deviation index score value that is greater than or equal to 0.25.

### **Stability Report**

The Stability report evaluates changes in the distribution of scored output variable values as models score data over time. It uses the same deviation index function that is used by the Characteristic report, except that the Stability report detects and quantifies shifts in the distribution of output variable values in the data that is produced by the models.

If an output variable from the training data set and the output variable from the current data set have identical distributions, then that output variable's deviation index is equal to zero. An output variable with a deviation index value that is greater than 0.10 and less than 0.25 is classified as having a mild deviation. A variable that has a deviation index value that is greater than 0.30 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.



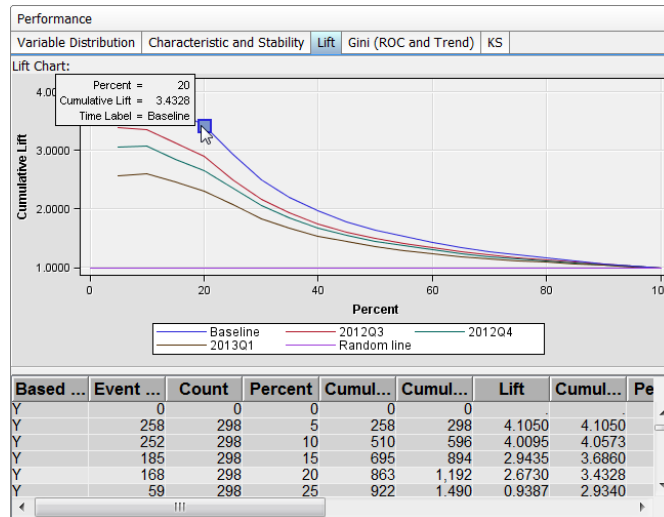
## Model Monitoring Reports

### Lift Report

The 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.



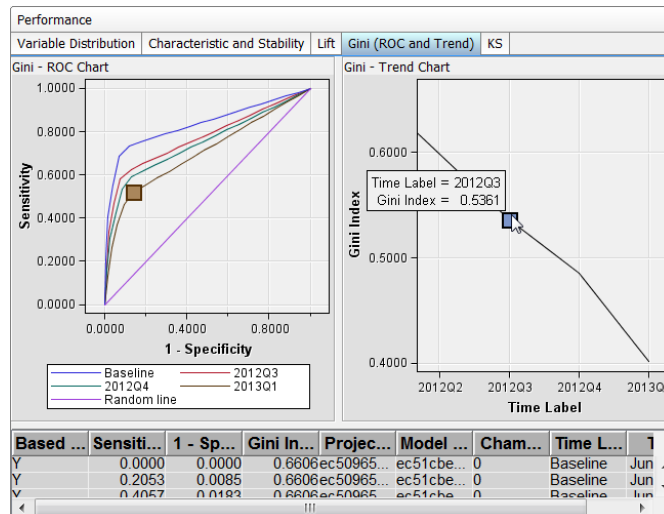
### Gini (ROC and Trend) Report

The **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

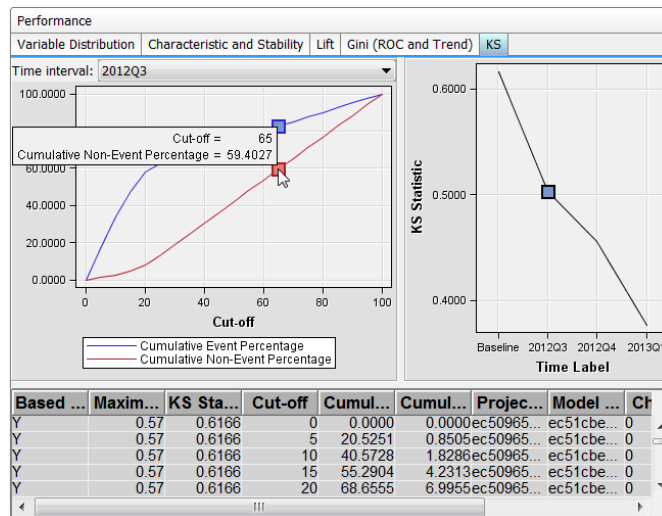
summary data set that plots the KS statistic and the KS probability cutoff values over time.

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.

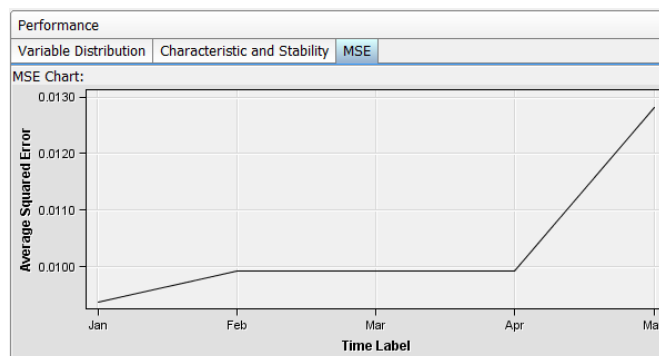
The following report is an example of a KS report. By placing the cursor over a point in the chart, you can view the data for that point.



### Mean Squared Error Report

The Mean Squared Error (MSE) report checks the accuracy of a prediction model with an interval target by comparing the estimation derived from the test data and the actual outcomes that are associated with the test data for different time periods.

The following report is an example of an MSE report.



---

## Performance Index Warnings and Alerts

The production model performance reports use performance measurement thresholds to benchmark and gauge the performance of a predictive model. When one of the performance measurements exceeds one or more specified indexes or thresholds, warning and alert events occur. When warning or alert events occur, warning and alert notifications are automatically sent by e-mail to recipients whose e-mail address is configured either in the Define Performance Task wizard or in the batch program that runs the reports.

Use the following assignment statements to set warning and alert conditions:

```
alertCondition='alert-condition';
warningCondition='warning-condition';
```

The condition must be enclosed in quotation marks if you use SAS code to create the report. An error occurs if you enclose the condition in quotation marks in the Define Performance Task wizard.

The following indexes and thresholds can be configured in either the Define Performance Task wizard or in a batch program that creates the report specifications:

### Characteristic report

You can configure the thresholds for the performance indexes P1 and P25. The P1 and P25 indexes represent the count of input variables with deviation index scores exceeding 0.1 and 0.25, respectively. 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.

For the prediction model MSE report, you can configure the mseDecay statistic threshold values. The mseDecay statistic is the performance decay of the MSE 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 indexes 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 warnings 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” on page 252

---

## 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. Both methods can process one or more performance data sources. When you create a



performance task, you can specify one or more data sources to process. When you use a batch program, you use a separate DATA step to process each data source.

If you run batch programs, you can find example programs in the `sashelp.modelmgr.source` catalog. These reports' filenames are **reportexample<sub>x</sub>**, 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 That 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 one or more SAS data sets that are a snapshot of model output. They can be registered in SAS Management Console or they can be accessed by using a libref that has been defined by using the Edit Start-up Code window.	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 sets do not need to be registered with SAS Management Console.
Ensure access to the champion or challenger 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.
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"> <li>• <code>mm_jobs.project</code></li> <li>• <code>mm_jobs.emailaddr</code></li> <li>• <code>mm_jobs.reportdef</code></li> <li>• <code>mm_jobs.jobtime</code></li> </ul>
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 <code>%MM_RunReports()</code> macro.

Task	Reports Created by Using the Define Performance Task Wizard	Reports Created Using SAS Programs That Run in Batch
Run the reports	Execute the code from the <b>Performance Monitor</b> node that was generated by the Define Performance Task wizard or schedule the performance task from the <b>Performance Monitor</b> pop-up menu. The data sets that underlie the monitoring reports are stored in the <b>Resources</b> 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 <b>Resources</b> 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 <b>Performance</b> folder for the champion model's version to view the reports.	Select the <b>Performance</b> folder for the champion model's version to view the reports.

## Chapter 16

# Create Reports by Defining a Performance Task

---

<b>Overview of Creating Reports Using a Performance Task</b> . . . . .	<b>263</b>
Creating Reports Using a Performance Task . . . . .	263
Determine How to Use the Performance Data Sets . . . . .	264
<b>Prerequisites for Running the Define Performance Task Wizard</b> . . . . .	<b>266</b>
Overview of Prerequisites . . . . .	266
Ensure That the Champion Model Is Set or That the Challenger Model Is Flagged . . . . .	266
Ensure That the Champion Model Function and Class Target Level Are Valid . .	266
Ensure That the Performance Data Source Is Available . . . . .	267
Ensure That Project and Model Properties Are Set . . . . .	267
Map Model and Project Output Variables . . . . .	268
<b>Run the Define Performance Task Wizard</b> . . . . .	<b>268</b>
<b>Schedule Performance Monitoring Tasks</b> . . . . .	<b>273</b>
Overview of Scheduling Performance Monitoring Tasks . . . . .	273
Create the Schedule . . . . .	274
Delete a Performance Task Schedule . . . . .	274
Schedule Properties for the Performance Monitor . . . . .	275
<b>View Performance Monitoring Job History</b> . . . . .	<b>275</b>
<b>Delete Performance Summary Data Sets</b> . . . . .	<b>276</b>

---

## 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 either the project champion model or a challenger model that is flagged in any version 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 performance data results. Then, you execute the generated code. You can execute the code immediately, or you can schedule a date and time at which the task is to run. Information about performance tasks is recorded and can be viewed in the **Performance Monitor Job History** tab. SAS Model Manager stores the output data in the **Resources** folder of the default version. To view the performance data results, you select the **Performance** node in the champion model's version.

To create performance reports in SAS Model Manager, follow this process:

- Ensure that one or more performance data sources are registered using SAS Management Console or that a libref has been defined for the location where the performance data sets are stored.
- 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 or schedule when the generated code is to be executed.
- To view the reports, select the **Performance** node in the champion model's version.

### **Determine How to Use the Performance Data Sets**

Before you run the Define Performance Task wizard, the performance data sets must be registered in the SAS Metadata Repository by using SAS Management Console or a libref must be defined in SAS Model Manager for the library that contains the performance data sets. For each SAS Model Manager project, you can set up your environment to use the performance data source that is most appropriate for 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 or stored in a library that has a libref that has been defined in SAS Model Manager. You can create and execute a performance monitoring task each time you take a snapshot, or you can create a performance monitoring task to execute multiple performance data sets in the same task.
- **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 or in the SAS library that you defined for performance data sets.

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 **Data Source** column of the **Performance Data Options** table in the wizard is completed by SAS Model Manager. You modify only the **Collection Date** and **Report Label** columns in the table.

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	<p>Each month, take a snapshot of the operational data and create a performance data set with a different name:</p> <ul style="list-style-type: none"> <li>• Jul13</li> <li>• Aug13</li> <li>• Sep13</li> <li>• Oct13</li> <li>• Nov13</li> <li>Dec13</li> </ul>	<p>Every month, take a snapshot of the operational data and name the performance data set using the same name:</p> <p>2013perf</p>
If you are registering the performance data sets in the SAS Metadata Repository, register the performance data sets using the 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.
If the performance data set is accessed by using a libref, store the data set in the SAS library.	Save the performance data set in the SAS library that is defined by a libref in SAS Model Manager.	Save the performance data set in the SAS library that is defined by a libref in SAS Model Manager.
Modifications to make in the Define Performance Task wizard	In Step 3, select one or more performance data sources. For each data source, select a data collection date and enter a date label.	<p>In Step 3, select a data collection date and enter a date label.</p> <p>The <b>Performance data source</b> 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.</p>
Create the reports	<p>Run the Define Performance Task wizard and execute the reports from the <b>Performance Monitor</b> project node or schedule when the task is to execute.</p> <p>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.</p>	<p>Monthly, run the Define Performance Task wizard and execute the reports from the <b>Performance Monitor</b> project node or schedule when the task is to execute.</p> <p>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.</p>

---

## 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 is set or the challenger model is flagged.
- Ensure that the champion or challenger model is within a project that is associated with a classification model function and has a binary target, or a prediction model function and has an interval target.
- Ensure that the champion or challenger 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 551](#).
- Ensure that the performance data sets for the time period that you want to monitor are registered in SAS Management Console or that a libref has been defined for the SAS library where the performance data sets are saved.
- 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 Is Set or That the Challenger Model Is Flagged

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 champion model is not set, right-click the champion model name and select **Set as Champion**. The  icon appears next to the champion model name and the version for the champion model.

You can determine the challenger model by looking for the  icon next to the challenger model name.

If the challenger model is not set, right-click the challenger model name and select **Flag as Challenger**. The  icon appears next to the challenger model.

### Ensure That the Champion Model Function and Class Target Level Are Valid

Performance monitoring is valid only for a project that is associated with a classification model function and has a binary target, or for a prediction model function that has an

interval target. You should define only performance tasks for classification and prediction models. The champion model must have a function type of classification and must contain a binary target, or a function type of prediction and must contain an interval target.

From the Projects category view, select the champion model name and verify that the **Function** property in the specific properties section is set to **Classification** or **Prediction**. 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** for a classification model or to **INTERVAL** for a prediction model.

### **Ensure That the Performance Data Source Is Available**

The Define Performance Task wizard requires that the performance data either be registered in the SAS Metadata Repository using SAS Management Console, or that a libref is defined from the Edit Start-up Code window to access the performance data from a SAS library.

If your performance table is not available for selection, do one of the following actions:

- Contact your administrator to add the table to the Data Library Manager using SAS Management Console. For more information, see the *SAS Model Manager Administrator's Guide*.
- Define a libref to access the performance data in a SAS library. For more information about defining a libref for the performance data, see [“Using Tables from a Local or Network Drive” on page 42](#).

#### **See Also**

[“Project Tables” on page 33](#)

### **Ensure That Project and Model Properties Are Set**

Several properties must be defined in order to generate the model performance reports. Verify that the appropriate project and model properties are set. Here is a list of properties.

#### Classification Project Properties

- Training Target Variable
- Target Event Value
- Class Target Level
- Output Event Probability Variable

#### Prediction Project Properties

- Training Target Variable
- Class Target Level
- Output Prediction Variable

#### Model Properties

- Score Code Type

### 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 **Edit**.
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 or create a schedule to execute the generated code on a specific day and time. Execution of the generated code creates the SAS data sets that are used to display reports: either the monitoring reports from the version **Performance** node, or the Monitoring report or Champion and Challenger Performance report that you create from the New Report wizard.


To create the reports:

1. Right-click the project name and select **Define Performance Task**. The **Define Performance Task** wizard appears.

*Note:* The **Define Performance Task** pop-up menu item is available to only SAS Model Manager administrators and advanced users.



**Define Performance Task** [X]

 **Input and Output Variables**

Select one or more input and output variables for analysis. Step 1 of 4

**Output Variables for Stability Analysis**

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

[ Select All ] [ Clear All ]

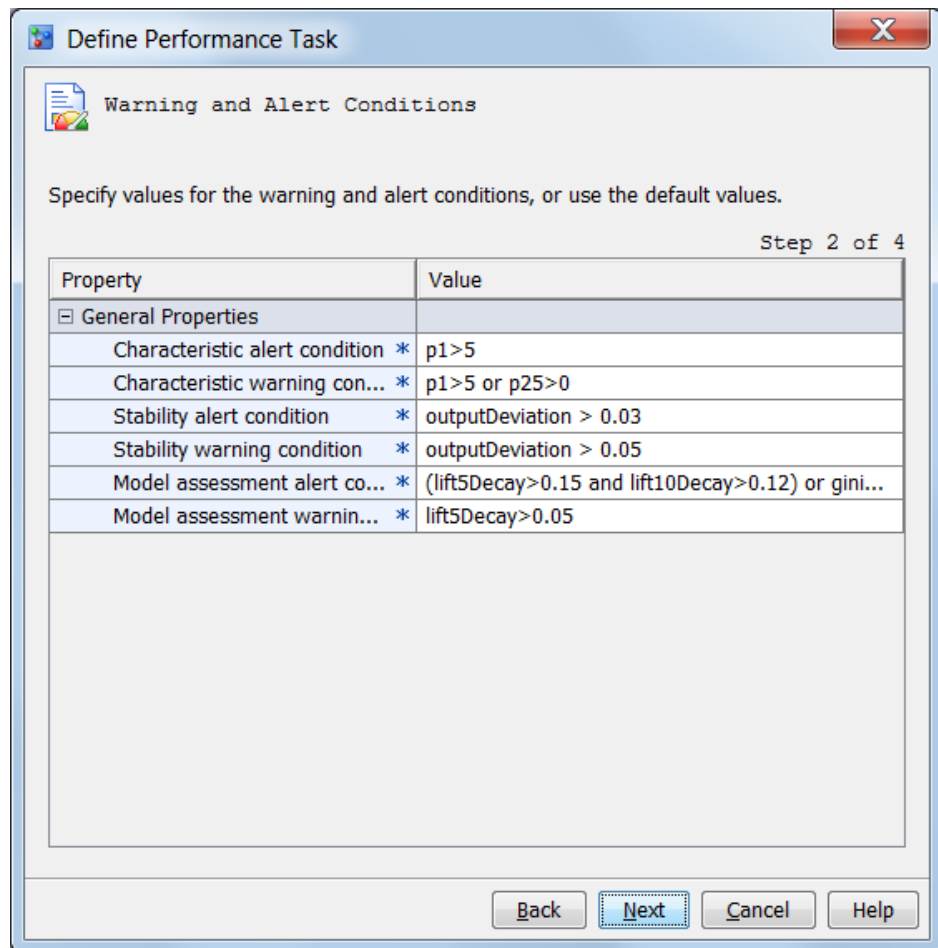
**Input Variables for Characteristic Analysis**

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

[ Select All ] [ Clear All ]

[ Back ] [ **Next** ] [ Cancel ] [ Help ]

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**.



5. Choose the data processing methods.
  - To run a standard environment, select **Standard configuration**.
  - To run the performance monitoring task in a High-Performance Analytic environment, select **High-performance configuration**.
  - To run the scoring task code in the performance monitor job, select **Run model score code**. If **High-performance configuration** is selected, the **Run model score code** check box is not available. If you do not select **Run model score code**, make sure that all output variables for stability analysis exist in the performance data source.

*Note:* The scoring task is not run when **High-performance configuration** is selected. To use the high-performance configuration, the High-Performance Analytics server product must be licensed.

**Define Performance Task**

**Data and Model Specifications**

Select the data processing method, specify the performance data options and select a model.

Step 3 of 4

**Data Processing Method**

Standard configuration     High-performance configuration

Run model score code

**Models**

Select	Model Na...	Role	Version	Model Type
<input type="radio"/>	Tree1	Champion	2013	Classification
<input checked="" type="radio"/>	Reg1	Challenger	2013	Classification


**Performance Data Options**


Baseline collection date: Jun 29, 2012

Data Source	Collection Date	Report Label
MMLib.HMEQ_2012Q2	Jun 29, 2012	2012Q2
MMLib.HMEQ_2012Q3	Sep 28, 2012	2012Q3
MMLib.HMEQ_2012Q4	Dec 31, 2012	2012Q4
MMLib.HMEQ_2013Q1	Mar 29, 2013	2013Q1

**SAS Application Server**

Default server: SASApp


- Select a model from the **Models** table. If a challenger model has been flagged, the challenger model is listed in the **Models** table.
- Click , click the empty cell in the **Data Source** column, and click **Browse**. Select a performance data table and click **OK**.
 

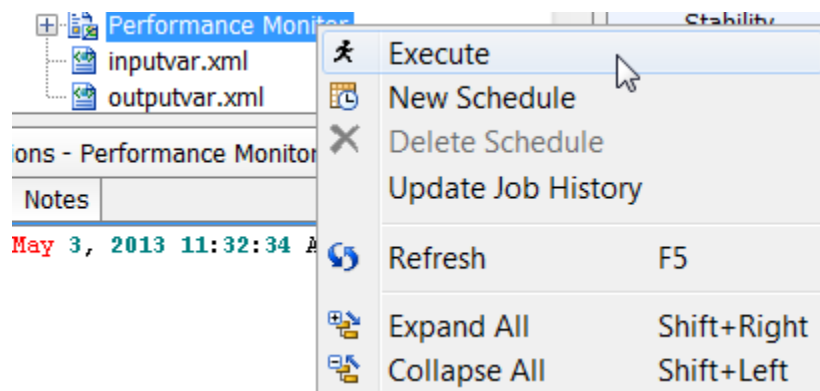
*Note:* When you add multiple tables, the baseline performance table is the table with the earliest collection date.
- Click the empty cell in the **Collection Date** column and click . Select a date for the collection date. The date can be any date in the time period when the performance data was collected.
 

*Note:* The date that you select is used by SAS Model Manager for sequencing data and does not appear in any charts. If the performance data is for the first quarter of 2013, the date could be any date between January 1 and March 31, 2013.
- To add a label for the date, enter the label in the **Report Label** column. The report label represents the time point of the performance data source. Because the report label appears in the performance charts, use a label that has not been used in another report, is short, and is understandable (for example, 2013Q1 or 2013).

*Note:* Duplicate report labels result in previous performance results being overwritten.

When the performance monitoring report is for a challenger model and when the data will be used in a Champion and Challenger Performance report, some requirements apply. Namely, the value of the **Report Label** field must be the same report label that was used for the same time period when the performance monitoring report was run for the champion model. For example, if the report label for the champion model's data from the first quarter of 2013 is 2013Q1, the date label for the challenger model's data from the first quarter of 2013 must be 2013Q1.

10. (Optional) Repeat Steps 7–9 to add multiple performance data tables to the performance monitoring task.
11. (Optional) To delete a table from the performance monitoring task, select the table and click .
12. (Optional) Click **Validate** to verify that the selected input variables and target variables are included in the performance data source.
13. Click the **Default server** list and select a SAS Application Server where the performance task is to execute.
14. Click **Next**.
15. (Optional) To send the scoring results by e-mail, click the **Add** button in the **E-mail Notifications** table. The Add Contact window appears.
  - a. Enter an e-mail address.
  - b. Select either **Yes** or **No** if you want an alert warning 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 with the job status to be sent by e-mail every time the report runs.
16. Click **Finish**. The **Working** status box appears while the code is generated.
17. To execute the generated code at a particular time, see “[Schedule Performance Monitoring Tasks](#)” on page 273. To execute the generated code from the Project Tree immediately, right-click the **Performance Monitor** folder and select **Execute**. The performance task is executed as a background process. SAS Model Manager saves the data sets that create the monitoring reports in the **Resources** folder of the default version.



*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 **Performance Monitor** node.

18. To view the reports, click the **Performance** node in the champion model's version. On the right, click the tab for the report that you want to view.

The screenshot shows the SAS Model Manager interface. The Project Tree on the left shows the 'Performance' node selected. The main area displays two reports: 'Characteristic Report' and 'Stability Report'. The 'Characteristic Report' shows a line graph of Deviation Index vs Time Label (2012Q3, 2012Q4, 2013Q1) with four lines. Below it is a table with columns 'Based ...', 'Variabl...', and 'Deviati...'. The 'Stability Report' shows a line graph of Deviation Index vs Time Label (2012Q3, 2012Q4, 2013Q1) with one line labeled 'score'. Below it is a table with columns 'Based ...', 'Variabl...', 'Deviati...', and 'Pr'.

Based ...	Variabl...	Deviati...
Y	CLAGE	0.0001
Y	CLNO	0.0001
Y	DEBTINC	0.0001
Y	DELINQ	0.0001
Y	DEROG	0.0001
Y	JOB	0.0001
Y	LOAN	0.0001
Y	MORTDUE	0.0001
Y	NINQ	0.0001
Y	REASON	0.0001

Based ...	Variabl...	Deviati...	Pr
Y	score	0.0000	ec5
Y	score	0.0009	ec5
Y	score	0.0031	ec5
Y	score	0.0086	ec5

## Schedule Performance Monitoring Tasks

### Overview of Scheduling Performance Monitoring Tasks

After you create a performance monitoring task, you can create a schedule to execute the task to run on a specific day and at a specific time. You can schedule the task to run hourly, daily, weekly, monthly, or yearly.

Before you can schedule a performance task, the password for your user ID must be made available to the SAS Metadata Repository. Passwords can be added using SAS Management Console. To add or update your password, contact your SAS Model Manager Administrator.


You cannot edit performance monitoring schedules. To modify a schedule, delete the schedule and create a new schedule.

After performance monitoring jobs execute, you can view the job history using the **Job History** tab of the **Performance Monitor** node.

## Create the Schedule

To schedule a performance monitoring task:

1. Right-click the **Performance Monitor** node and select **New Schedule**. The New Schedule window appears.

2. To set how often to run the performance monitoring task, click the **Recurrence** list box and select a time interval and the interval options in the **Interval** box.
3. To set the time to run the job, select an hour from the **Hour** list box and select a minute from the **Minute** list box.
4. To set the start date, click the calendar  and select a start date. Instead of using the calendar, you can select a month from the **Month** list box, select a day from the **Day** list box, and select a year from the **Year** list box.
5. (Optional) Click **Advanced**. Select the server that schedules the job from the **Scheduling server** list box. Select the batch server that runs the job from the **Batch server** list box. Select a location for the performance monitoring job from the **Location** list box. Click **OK**.
6. Click **OK**. A message box confirms that the schedule was created. Click **Close**.

*Note:* Performance monitoring task schedules cannot be edited. To change the schedule, delete the schedule and create a new schedule.

## Delete a Performance Task Schedule

To delete the schedule for a performance monitoring task:

1. Expand the project folder, right-click **Performance Monitor**, and select **Delete Schedule**.
2. A message box confirms that the schedule was deleted. Click **Close**.

### **Schedule Properties for the Performance Monitor**

Here is a list of the **Schedule** properties for the **Performance Monitor**:

<b>Property Name</b>	<b>Description</b>
<b>Job Name</b>	Specifies the name of the performance monitoring task. This name cannot be changed.
<b>Location</b>	Specifies the location of the performance monitoring task job definition in the SAS Metadata Repository.
<b>Scheduling Server</b>	Specifies the name of the server that schedules the job for the performance monitoring task.
<b>Batch Server</b>	Specifies the name of the server that executes the job for the performance monitoring task.
<b>Recurrence</b>	Specifies how often the scheduled job for the performance monitoring task is to be executed.
<b>SAS Application Server</b>	Specifies the name of the SAS Application Server where the performance monitoring task is to be executed.

---

## **View Performance Monitoring Job History**

Use the **Performance Monitor Job History** tab to verify whether a performance monitoring task was run. The performance monitoring job appears on the **Job History** tab only after the job has begun.

To view the job history of a performance monitoring task:

1. Expand the project folder and select **Performance Monitor**.
2. In the details section, click the **Job History** tab. A table appears that lists the performance monitoring jobs that have been executed.

Here is a description of the columns in the job history table:

#### **Job Name**

is the name of the performance monitoring task.

#### **Job Status**

specifies whether the job status is **Running** or **Completed**.

**Execution Status**

shows a green indicator for a successful job execution. A yellow indicator shows that the performance monitoring task ran with warnings. A red indicator shows that the performance monitoring task ran with errors.

**Date Started**

is the date and time that the performance monitoring task started.

**Date Completed**

is the date and time that the performance monitoring task ended.

**Log**

is the revision number for the SAS log.

**Output**

is the revision number for the job output.

**SAS Code**

is the revision number for the performance monitoring task program.

---

## Delete Performance Summary Data Sets

After a performance monitoring task has run, the summary data sets reside in the **Resources** folder.

To delete the performance summary data sets:

1. Expand the version folder.
2. Right-click the **Resources** folder and select **Delete Performance Data Sets**.
3. Click **Yes** to confirm the deletion, and click **Close** after the files have been deleted.



## Chapter 17

# Create Reports Using Batch Programs

<b>Overview of SAS Programs to Monitor Model Performance</b> . . . . .	<b>278</b>
<b>Prerequisites for Running Batch Performance Reports</b> . . . . .	<b>279</b>
Overview of Prerequisites for Running Batch Performance Reports . . . . .	279
Publish the Champion Model from the Project Folder . . . . .	279
Create a Folder Structure . . . . .	279
Obtain Performance Data . . . . .	281
Determine the Publish Channel . . . . .	281
Copy Example Batch Programs . . . . .	281
Determine SAS Model Manager User ID and Password . . . . .	282
<b>Report Output in Test and Production Modes</b> . . . . .	<b>282</b>
Report Output in Test Mode . . . . .	282
Report Output in Production Mode . . . . .	283
<b>Define the Report Specifications</b> . . . . .	<b>283</b>
Overview of Code to Define Report Specifications . . . . .	283
Required Libref . . . . .	283
Project Specifications . . . . .	284
E-mail Recipient Specifications . . . . .	286
Report Specifications . . . . .	287
Job Scheduling Specifications . . . . .	290
Example Code to Create the Report Specifications . . . . .	291
<b>Extracting the Champion Model from a Channel</b> . . . . .	<b>294</b>
Using the %MM_GetModels() Macro . . . . .	294
Accessing SAS Model Manager Report Macros . . . . .	295
%MM_GetModels() Macro Syntax . . . . .	295
Example Program to Extract a Model from a Channel . . . . .	295
The current.sas7bdat Data Set . . . . .	296
<b>SAS Code to Run Performance Reports</b> . . . . .	<b>297</b>
Overview of the SAS Code to Run the Performance Reports . . . . .	297
Accessing SAS Model Manager Report Macros . . . . .	298
Required Librefs . . . . .	298
Macro Variables to Define Report Local Folders and Data Sets . . . . .	298
Macro Variables That Are Used by the %MM_RunReports() Macro . . . . .	299
The DATA Step to Access the Performance Data Set . . . . .	300
The %MM_RunReports() Macro . . . . .	300
Example Code to Run the Reports . . . . .	302

---

## 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.

*Note:* SAS programs for performance monitoring reports can be run only for champion models. Performance monitoring reports for challenger models can be run only by creating a performance task using the SAS Model Manager client.

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` catalog. The location of this catalog for Windows is `\sasinstalldir\SASFoundation\9.4\mmcommon\sashelp`. The default value for `sasinstalldir` in Windows is `C:\Program Files\SASHome`. The location of this file for UNIX is `/sasinstalldir/SASFoundation/9.4/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 282](#)

**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 40](#)

**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 the **History** tab. Look for a publish model entry. In this example, the channel name is **MMChannel**: May 29, 2013 4:55:11 PM[mdlmgradmin] "Tree1" was published to "MMChannel(sas-oma://RDCEX09147.race.sas.com:8561/reposid=A5DB5KPY/ITChannel;id=A5DB5KPY.BC000001)" successfully.
- Select the **MMRoot**, project, or version folder, and then select the **Publish History** tab. Look for the publish model entry and then select the entry to view the publish details. In this example, the channel name **MMChannel** that can be found from the value of **SAS destination location**. Here is an example: **/Channels/Model Manager Channels/MMChannel**.
- 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 the channel and select **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 20](#)

## **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 279](#)

- “Delete Performance Summary Data Sets” on page 276

### **Report Output in Production Mode**

When you run the %MM\_RunReports() macro in Production mode, ensure that you complete the following code changes:

- In the DATA step mm\_jobs.project, remove the assignment of the variable testDestination=reportOutputPath.
- In the %MM\_RunReports() macro, set the macro variable \_MM\_ReportMode=Production.

Production report output is written to the **Resources** folder in the default version of the project. To view the report output, you select the **Performance** folder in the default version.

## **Define the Report Specifications**

### **Overview of Code to Define Report Specifications**

Before you can create a monitoring report for a project, you must create several data sets that define the report specifications:

mm_jobs.project	defines the project information, such as the project UUID, project variables, and the model repository URL for the project. It is recommended that you create only one observation in this data set.
mm_jobs.emailaddr	defines the e-mail addresses for the recipients of job status and the notification flags for alert and warning notifications.
mm_jobs.reportdef	defines the types of reports to create and the warning and alert conditions that are associated with those reports.
mm_jobs.jobtime	defines the date and time to run the reports and a label that describes the time performance data set period.

The code that you write to create the report specifications might need to be run only after it is created and only whenever it is modified. These data sets might not need to be created every time 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_ReportDatasc=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 282](#)

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 expand **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 conditions for each type of report:

Report Type	Example Alert Condition
Characteristic	<code>alertCondition='p1&gt;5 or p25&gt;0';</code>
Stability	<code>alertCondition='outputDeviation &gt; 0.03';</code>
Model Assessment	<code>alertCondition='lift5Decay&gt;0.15 and lift10Decay&gt;0.12) or giniDecay&gt;0.1 or ksDecay&gt;0.1';</code> <code>alertCondition='msedecay &gt; 20';</code>

See also: see [“Performance Index Warnings and Alerts” on page 259](#).

`warningConditions='warningConditions';`

specifies a warning condition for the type of report. Enclose *warningConditions* in quotation marks.

Report Type	Example Warning Condition
Characteristic	<code>warningCondition='p1&gt;2';</code>
Stability	<code>warningCondition='outputDeviation &gt; 0.01';</code>
Model Assessment	<code>warningCondition='lift5Decay&gt;0.05';</code> <code>warningCondition='msedecay &gt; 10';</code>

See also: see [“Performance Index Warnings and Alerts” on page 259](#).

`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 `ddmmyyyy:hh:mm:ss` where `dd` is a two-digit year, `mmm` is the first three letters of the month, `yyyy` is a four-digit year, `hh` is a two-digit hour, `mm` is a two-digit minute, and `ss` is a two-digit second. Enclose `dateTime` in quotation marks.

The values of `scheduledTime` are used by the `%MM_RunReports()` macro, rather than by your job scheduler. Each time that the `%MM_RunReports()` macro runs, it checks the values of the `scheduleTime` variable. If the scheduled time has passed, the report runs. If it has not passed, the performance data sets are not created.

Example: `scheduledTime='03Jun2012:23:59:00'`;

`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='2012Q4'`;

### 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='01OCT2012:23:59:59';time='2012Q3';output;
scheduledTime='01JAN2013:23:59:59';time='2012Q4';output;
scheduledTime='01APR2013:23:59:59';time='2013Q1';output;
scheduledTime='01JUL2013:23:59:59';time='2013Q2';output;
scheduledTime='01OCT2013:23:59:59';time='2013Q3';output;

run;

```

**See Also**

- “Extracting the Champion Model from a Channel” on page 294
- “SAS Code to Run Performance Reports” on page 297

---

## 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 296.

## 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 281. 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.

```
%let _MM_Service_Registry_URL=
  %nrstr(http://myServer:80/SASWIPClientAccess/remote/Ser
```

```

viceRegistry);

/* Source file name: sashelp.modelmgr.reportExample1.source */

FILENAME mmmac
  catalog 'sashelp.modelmgr.reportmacros.source';
%inc mmmac;

%MM_GetModels(
  channel=\\network1\MMChampion\channel1,
  localPath=c:\mm.test\model.extraction);

```

### The current.sas7bdat Data Set

When models are extracted from a publishing channel, the current.sas7bdat data set contains the following information for each model:

Variable Name for the Project or Model Information	Description
algorithm	The algorithm that was used to create the model
fileName	Not used
isChampionModel	True or False to indicate whether the model is the champion model
keyWords	Keywords
miningFunction	The type of mining function, such as classification, prediction, segmentation
model	Not used
modeler	The name of the person who created the model
modelName	The name of the model
modelProductionTimestamp	The time at which 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

Variable Name for the Project or Model Information	Description
scoreCodeType	DATA Step or SAS Program
subject	The subject name
targetName	The Training Target Variable name
userAttr	User-defined attributes, such as "MODELER='sasguest' MODELPROJECTVARMAP='predictedProbability eq P_BAD1; predictedClass eq I_BAD;'"
versionName	The name of the version that contains the model
whenPublished	The date and time at which 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 283](#)
- [“SAS Code to Run Performance Reports” on page 297](#)

---

## 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 the 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_ReportDatsrc`, in the precode variable of the `mm_jobs.project` data set. Here is an example: `%let _MM_ReportDatsrc=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 [Appendix 3, “SAS Model Manager Macro Variables,”](#) on page 441.

## 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_ServiceRegistry_URL`

specifies the service registry to set the environment.

Example: `%let _MM_Service_Registry_URL=%nrstr(http://myServer:80/SASWIPClientAccess/remote/ServiceRegistry);`

`_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 300

For a description of the macro variables, see [Appendix 3, “SAS Model Manager Macro Variables,”](#) on page 441.

### 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 [Appendix 3, “SAS Model Manager Macro Variables,”](#) on page 441.

### **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_2013q1;
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 283](#).

### 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 score code 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 300](#)

**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=03Jul2013:12:15:30**

### 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 283 and “Extracting the Champion Model from a Channel” on page 294.

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_Service_Registry_URL=
  %nrstr(http://myServer:80/SASWIBClientAccess/remote/ServiceRegistry);
%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 2012Q4 report.It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 01Jan2013 in order to trigger the report */
/* execution scheduled for the 2012Q4 interval. */
/*****/

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2012Q4;
run;

%let currentTime=01Jan2013:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,
    user=&_MM_User,
    password=&_MM_Password);

/*****/
/* Now, run the 2012Q1 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Apr2012 in order to trigger the report */
/* execution scheduled for the 2012Q1 interval. */
/*****/

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2012q1;
run;

%let currentTime=03Apr2012:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,
    user=&_MM_User,
    password=&_MM_Password);

/*****/
/* Now, run the 2012Q2 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Jul2012 in order to trigger the report */
/* execution scheduled for the 2012Q2 interval. */
/*****/

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2012q2;
run;

%let currentTime=03Jul2012:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,

```

```

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

/*****
/* Now, run the 2012Q3 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Oct2012 in order to trigger the report */
/* execution scheduled for the 2012Q3 interval. */
*****/

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2012q3;
run;

%let currentTime=03Oct2012:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,
    user=&_MM_User,
    password=&_MM_Password);

/*****
/* Now, run the 2012Q4 report. It is necessary to */
/* artificially declare a "currentTime" argument */
/* of 03Jan2013 in order to trigger the report */
/* execution scheduled for the 2012Q4 interval. */
*****/

DATA scoreIn.hmeq0;
    set scoreIn.hmeq_2012q4;
run;

%let currentTime=03Jan2013:12:30:15;
%MM_RunReports(
    localpath=&_MM_JobLocalPath,
    currentTime=&currentTime,
    mapTable=&mapTable,
    user=&_MM_User,
    password=&_MM_Password);

```

**See Also**

- [“Define the Report Specifications” on page 283](#)
- [“Extracting the Champion Model from a Channel” on page 294](#)

## Chapter 18

# Formatting Performance Reports

---

<b>Format a Monitoring Report</b> .....	<b>305</b>
About Monitoring Reports .....	305
Create a Monitoring Report .....	307
<b>Format a Champion and Challenger Performance Report</b> .....	<b>308</b>
About the Champion and Challenger Performance Report .....	308
Verify Performance Data and Model Status .....	309
Create a Champion and Challenger Performance Report .....	309
<b>Performance Report Output Files</b> .....	<b>310</b>
<b>View Reports</b> .....	<b>311</b>
View Reports in SAS Model Manager .....	311
View Formatted Monitoring Reports .....	311

---

## Format a Monitoring Report

### *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 window 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 window, 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
- Actual vs. Predicted for prediction models
- Actual vs. Residual for prediction models
- Population Stability Trend for prediction models

Assessment charts are created for the Monitoring Report.

#### 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 Chart

When the Gini - ROC chart is created, the Gini index for each ROC curve is also created. The Gini index 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 underlying KS statistic and the corresponding probability cutoff are read from a summary data set in the Resources folder. The KS Trend chart uses a summary data set that plots the KS Statistic over time. The KS Trend chart is used to monitor model degradation over time.

#### Actual vs. Predicted

You use the Actual vs. Predicted plot to see how predicted values match actual values.

#### Actual vs. Residual

You use the Actual vs. Residual plot to determine how good the model is at predicting values by examining errors and error trending, and comparing them to the actual values.

#### Population Stability Trend

The Population Stability Trend chart measures the shift of the scoring output variable distribution over time. Scoring output that is based on a development sample is used as the baseline distribution. The deviation index is used to indicate the shift for a given point in time.

Before you create a Monitoring Report or a Champion and Challenger Performance Report, you must ensure that certain project and model properties are set. For more information, see [“Verify Project and Model Property Settings” on page 187](#).

These are the tasks that you perform for monitoring reports:


- [“Create a Monitoring Report” on page 307](#)
- [“View Reports” on page 198](#)

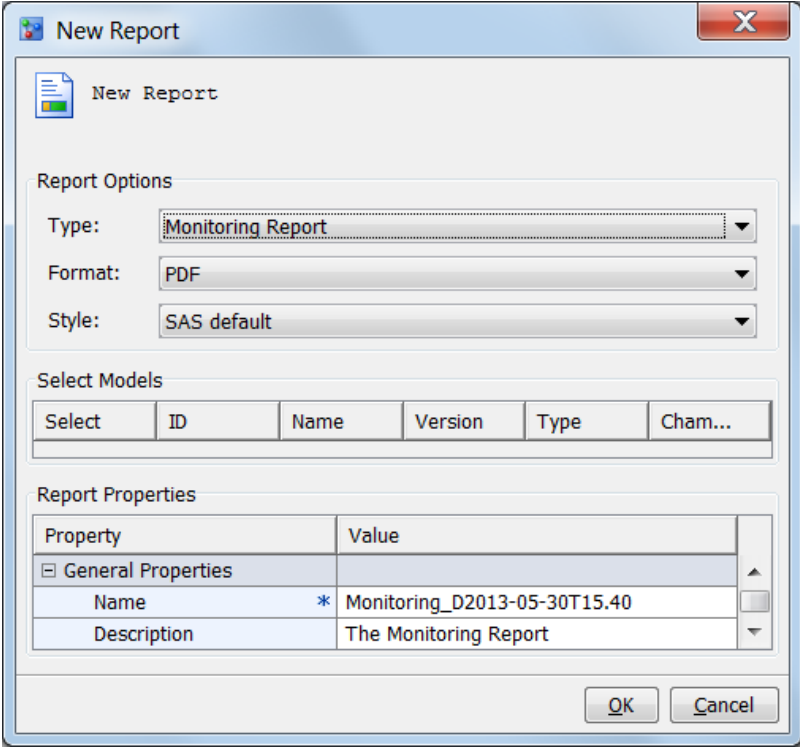
**See Also**

- Chapter 16, “Create Reports by Defining a Performance Task,” on page 263
- Chapter 17, “Create Reports Using Batch Programs,” on page 277

**Create a Monitoring Report**

To create a monitoring report:

1. Expand the version folder .
2. Right-click the **Reports** node and select **Reports** ⇒ **New Report**. The New Report window appears.
3. Select **Monitoring Report** from the **Type** list box.



**Report Options**

Type:

Format:

Style:

**Select Models**

Select	ID	Name	Version	Type	Cham...

**Report Properties**

Property	Value
[-] General Properties	
Name	* Monitoring_D2013-05-30T15.40
Description	The Monitoring Report

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**, **EXCEL**, and **RTF**.
5. In the **Style** list box, select a style for the output. The default is SAS default. Other options are **Seaside**, **Meadow**, and **Harvest**.
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 `Monitoring_DateTime`. The name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ).
7. Click **OK**. A message box confirms that the report was created successfully.

**See Also**

- “View Reports” on page 198
- “Performance Report Output Files” on page 310

---

## Format a Champion and Challenger Performance Report

### *About the Champion and Challenger Performance Report*

After you execute a performance task for the champion model, you can execute a performance task for the challenger model using the same performance data sets. SAS Model Manager updates the output data sets in the **Resources** folder with the performance data for the challenger model. You can create a Champion and Challenger Performance report that compares the performance of the two models.

The Champion and Challenger Performance report contains these charts:

#### Number of Predictors Exceeding Deviation Threshold

This characteristic report creates a chart for each index exceeding a deviation threshold (either 0.1 or 0.25) as indicated in the define performance task. The characteristic report detects shifts in the distribution of input variables over time.

#### Lift Trend Chart

A Lift Trend chart displays the cumulative lift of the champion model over time.

#### 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.

#### 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.

#### 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.

#### 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.

#### Score Histogram

The Score Histogram compares the scoring result distribution at different time periods using a histogram.

#### Score Distribution Line Plot




The Score Distribution Line Plot compares the scoring result distribution at different time periods using a line plot

Before you create a Champion and Challenger Performance report, verify the performance data and model status.




## Verify Performance Data and Model Status

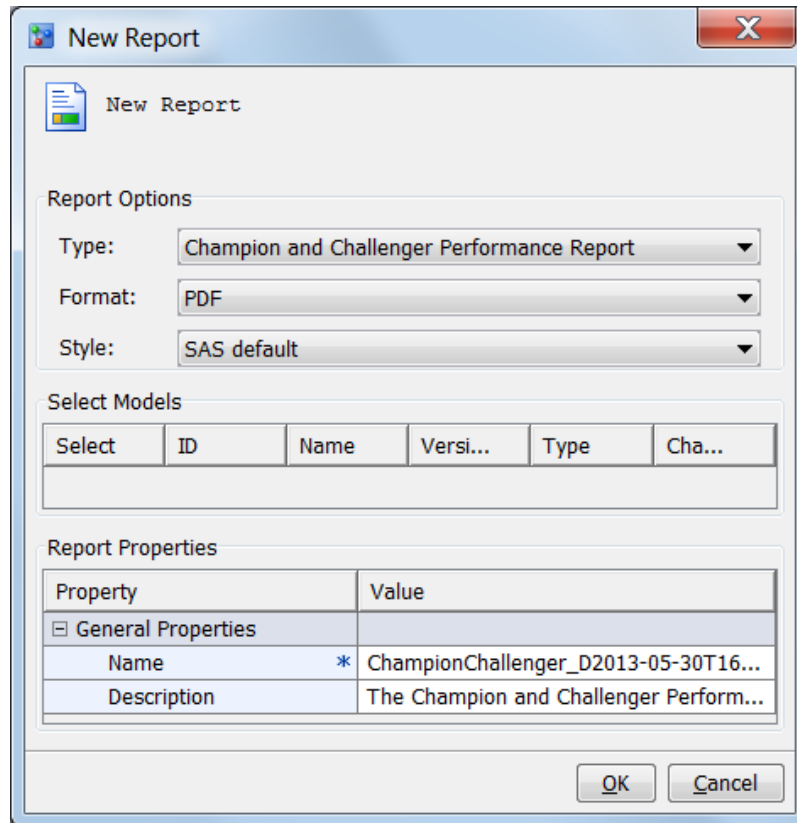
Before you can create a Champion and Challenger Performance report:

1. Click the version folder  and verify that the champion model has been set. The champion model has the check mark . If it does not, right-click the model and select **Set as Champion**.
2. Ensure that the challenger model is flagged . If it is not, right-click the model and select **Flag as Challenger**.
3. Verify that performance monitoring data is available for the champion model and the challenger model. Performance monitoring results must exist for the same performance data using the same time periods and data labels. Click the **Resources** node and select the file **jobstatus.sas7bdat**. The **Content** tab displays performance monitoring status data.
  - a. Verify that the UUIDs for the champion and challenger models are in the **Model UUID** column.
  - b. Using the **name** column and the **time** column, verify that matching date labels exist for the champion and challenger models for each type of report. If there are multiple date labels for a model for any given report, SAS Model Manager uses the most recent job.

## Create a Champion and Challenger Performance Report

To create a monitoring report:

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



3. In the **Format** list box, select the type of output that you want to create. The default is **PDF**. Other options are **HTML**, **EXCEL**, and **RTF**.
4. In the **Style** list box, select a style for the output. The default is SAS default. Other options are **Seaside**, **Meadow**, and **Harvest**.
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 `ChampionChallenger_DdateTtime`. The name can contain letters, spaces, the underscore ( `_` ), the hyphen ( `-` ), and the period ( `.` ).
6. Click **OK**. A message box confirms that the report was created successfully.

### See Also

- [“View Reports” on page 198](#)
- [“Performance Report Output Files” on page 310](#)

---

## Performance Report Output Files

The Monitoring report and Champion and Challenger report output files are stored 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 window **Report Properties** table.

Here are the files that are created each time you create a report:

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

- taskCode.sas

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

### *View Reports in SAS Model Manager*

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.

### *View Formatted Monitoring Reports*

To view the report:

1. Expand the **Reports** folder.
2. Right-click the report name and select **Open**.
3. If you are prompted to do so, enter a user ID and a password. Click **OK**.



## Chapter 19

# Using Dashboard Reports

---

<b>Overview of Project Dashboard Reports</b> .....	<b>313</b>
<b>Create a Dashboard Report Definition</b> .....	<b>314</b>
<b>Generate Dashboard Reports</b> .....	<b>319</b>
<b>View Dashboard Reports</b> .....	<b>320</b>
<b>Edit a Dashboard Report Definition</b> .....	<b>324</b>
<b>Manage All Project Dashboard Definitions</b> .....	<b>325</b>
<b>Delete a Project Dashboard Report Definition</b> .....	<b>325</b>

---

## Overview of Project Dashboard Reports

The SAS Model Manager Dashboard can provide reports that show the overall state of 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 by creating a dashboard report definition. The dashboard report definition is used to create the dashboard reports. You view the dashboard reports through the SAS Model Manager **Tools** menu. 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 268
- “Create a Dashboard Report Definition” on page 314
- “Generate Dashboard Reports” on page 319
- “View Dashboard Reports” on page 320

To manage all dashboard reports or to delete a project’s dashboard report, complete these tasks:

- Edit a Dashboard Report Definition on page 324
- Manage All Dashboard Report Definitions on page 325
- Delete a Project Dashboard Report Definition on page 325

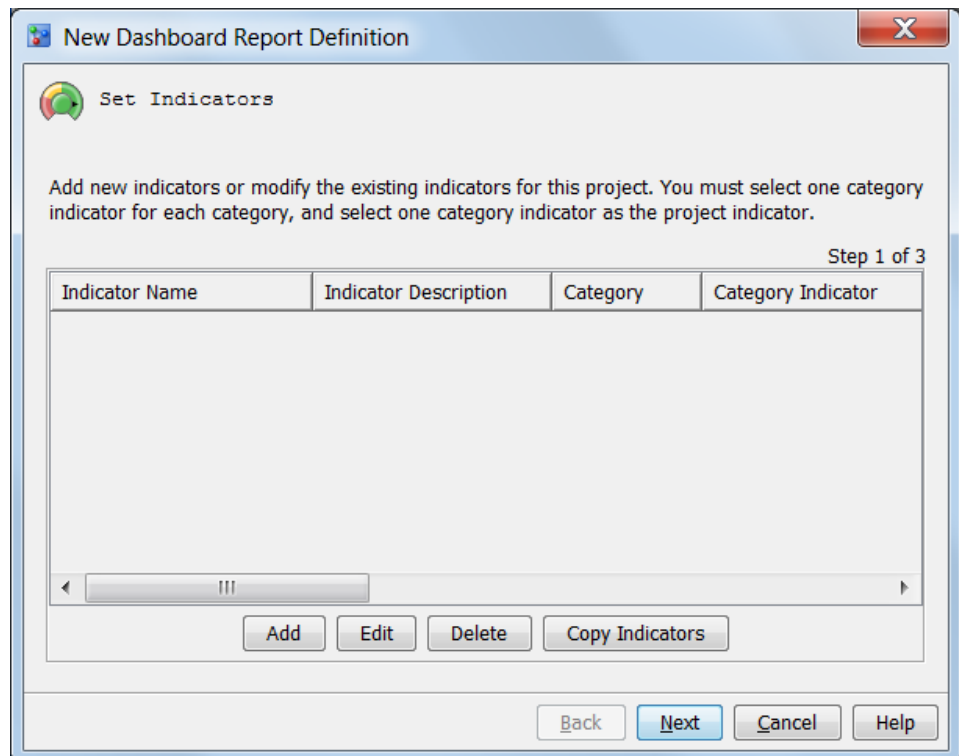
For more information, see the *SAS Model Manager: Administrator's Guide*.

---

## Create a Dashboard Report Definition

To define dashboard report indicators:

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 268.
2. Right-click the project folder in the Project Tree, and select **Dashboard Report Definition** ⇒ **New** from the pop-up menu. The New Dashboard Report Definition 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.

**Add Indicator**

Template: CHAR\_P1

Name: CHAR\_P1

Description: Number of predictors with deviation index exceeding 0.1

Condition:

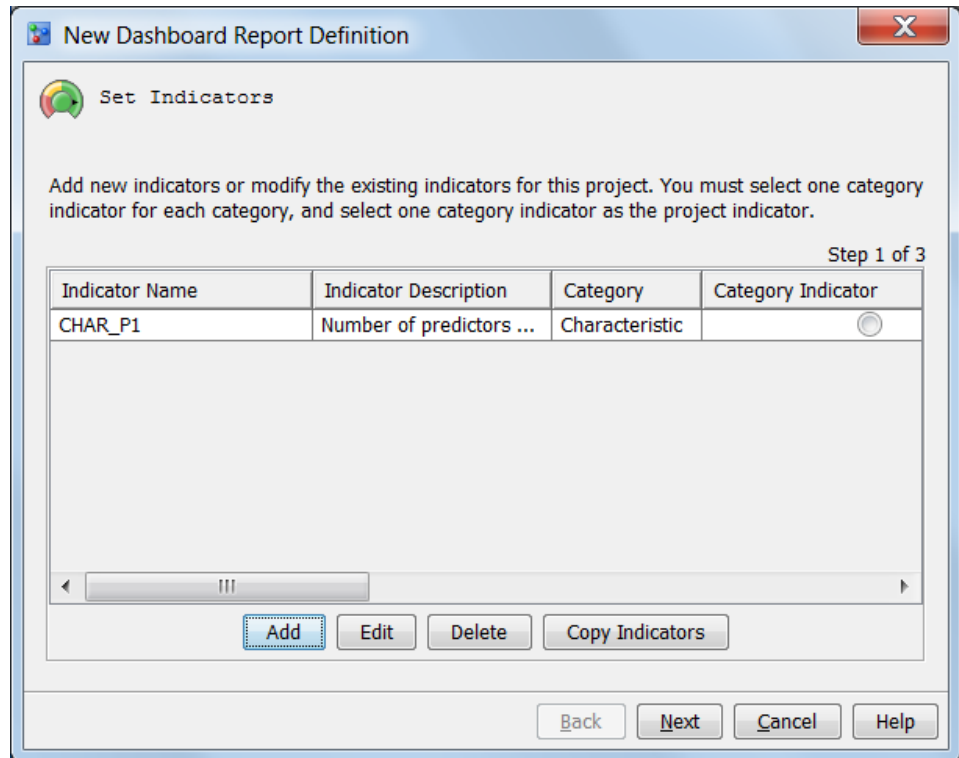
Range Definition

Normal:   ●

Warning:   ▲

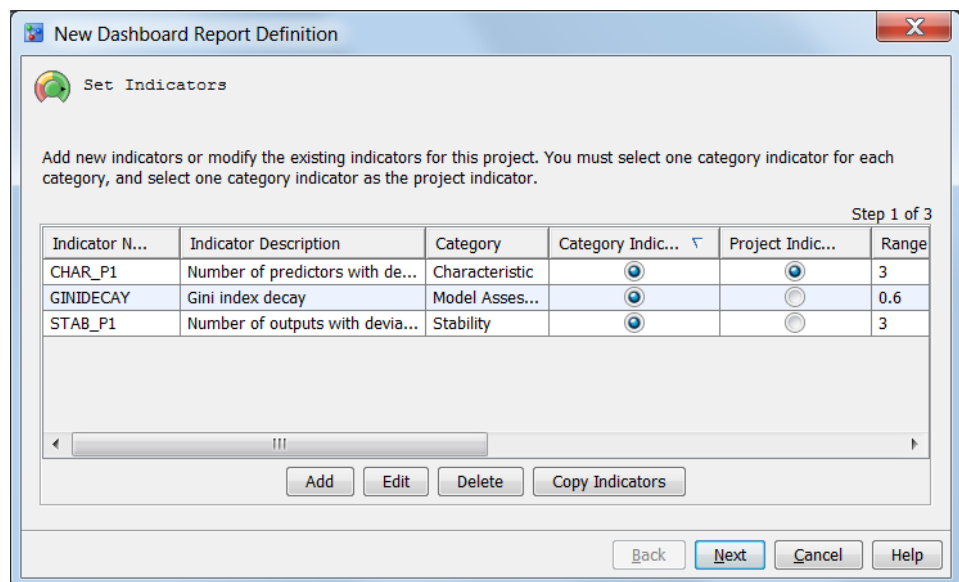
Alert:   ■

- 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 New Dashboard Report Definition window appears with information about the new indicator.



- Repeat step 4 for each indicator that you want to add. To edit an existing indicator, select the indicator, and click **Edit**. To delete an existing indicator, select the indicator and click **Delete**.
- 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.



- Click **Next**. The New Dashboard Report Definition window appears with information about setting up E-mail notifications.



New Dashboard Report Definition

Set E-mail Notifications

Specify an e-mail address for each recipient who you want to send an e-mail notification about the project status.

Step 2 of 3

Status	E-mail Address
--------	----------------

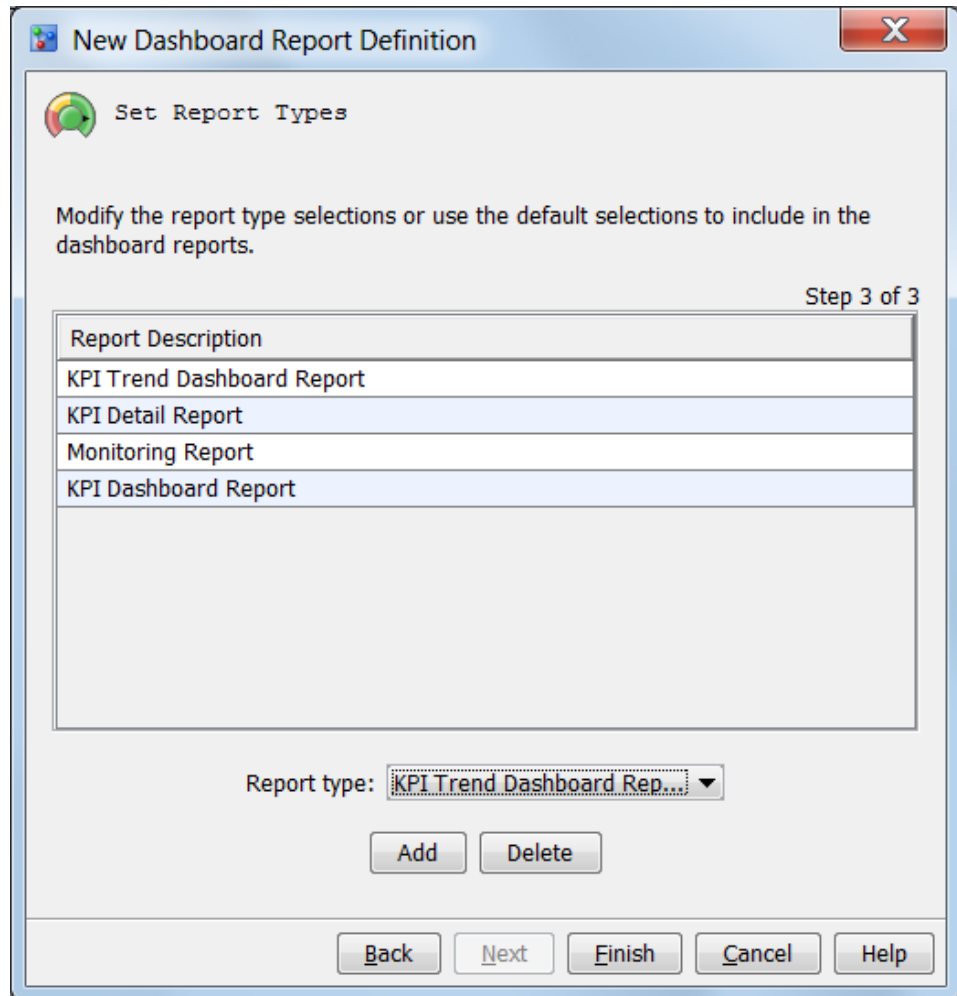
Project status: Normal

E-mail address:

Add Delete

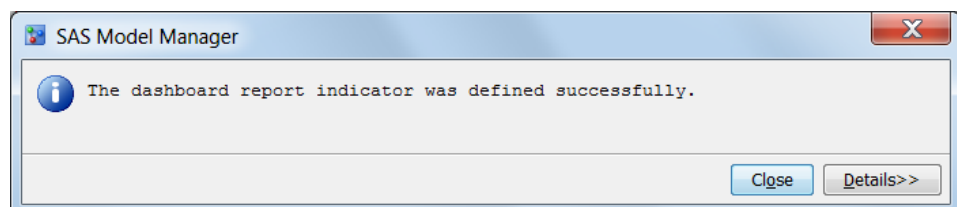
Back Next Cancel Help

8. Select a value from the **Project status** drop-down list, enter a value for **E-mail address**, and click **Add**. Repeat this step as needed. Each recipient will receive an e-mail notification about the status. To delete an e-mail notification, select an item from the list, and click **Delete**.
9. Click **Next**. The New Dashboard Report Definition window appears with information about setting report types.



10. By default, all of the report types are selected. To change report types:
  - a. To add a report type, select a value from the **Report type** drop-down list, and click **Add**.
  - 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 report definition and indicators were created 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 313
- “Generate Dashboard Reports” on page 319
- “View Dashboard Reports” on page 320

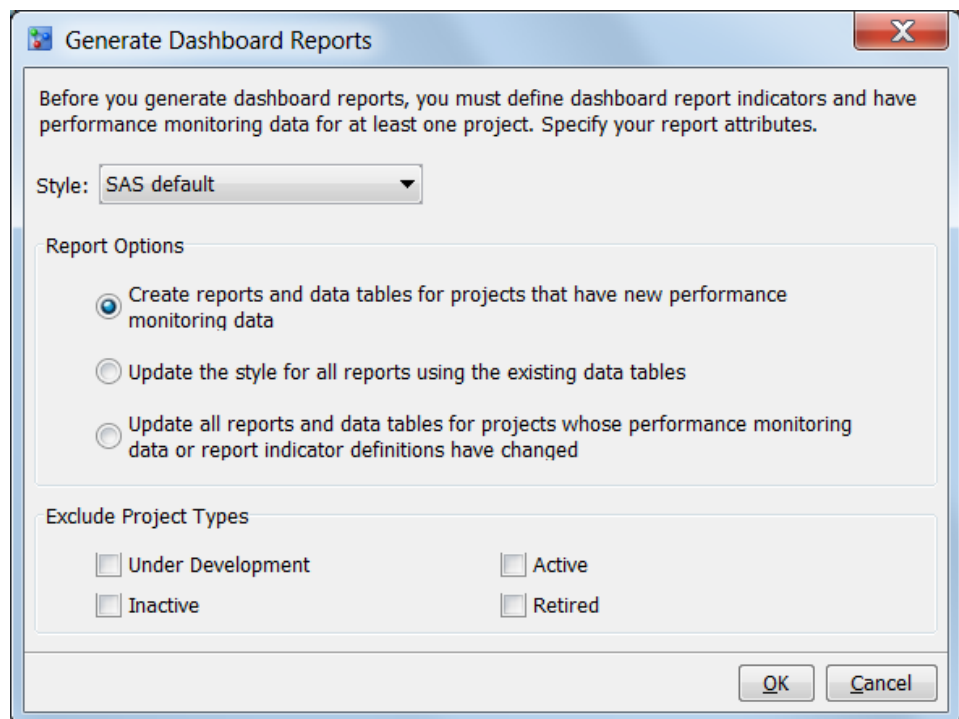
---

## Generate Dashboard Reports

To generate the dashboard reports:

*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.



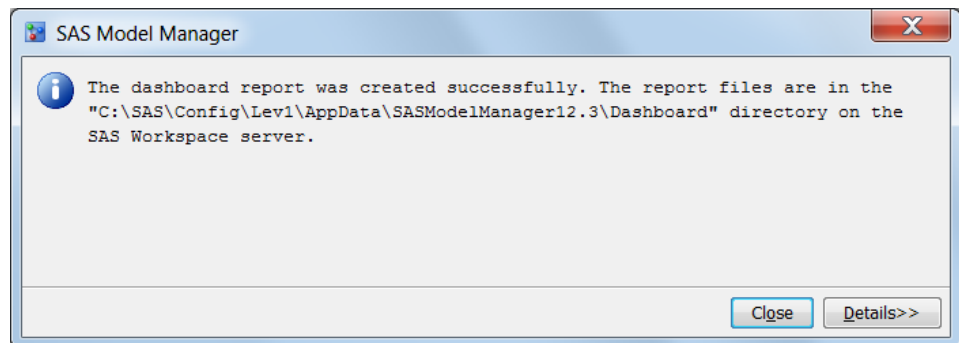
2. Select a style for the report from the drop-down list.

*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\SASModelManager12.3\Dashboard`.

*Note:* The location of the dashboard report was configured in the SAS Metadata Repository during the installation and configuration process. You must have access to the location in order to create the report.



6. To view the dashboard reports, select **Tools** ⇒ **View Dashboard Reports**. For more information, see “[View Dashboard Reports](#)” on page 320.

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

### See Also

- “[Overview of Project Dashboard Reports](#)” on page 313
- “[Create a Dashboard Report Definition](#)” on page 314

---

## View Dashboard Reports

To view the dashboard reports:

1. Select **Tools** ⇒ **View Dashboard Reports**

A web page displays a table with dashboard reports for each project that has a dashboard definition.

The screenshot displays a web browser window with the following content:

**All projects**

Project Name	Current Status	Owner	Model Age (days)
<a href="#">/MMRoot/HMEQProj/HMEQ</a>	<span style="color: red;">■</span> <a href="#">2013Q1</a>	mdlmgadmin	3

**History Status**

Project Name	Current	Current - 1	Current - 2	Current - 3
<a href="#">/MMRoot/HMEQProj/HMEQ</a>	<span style="color: red;">■</span> <a href="#">2013Q1</a>	<span style="color: red;">■</span> <a href="#">2012Q4</a>	<span style="color: green;">●</span> <a href="#">2012Q3</a>	<span style="color: green;">●</span> <a href="#">2012Q2</a>

2. Select a **Project Name** or status link to view the associated dashboard reports.

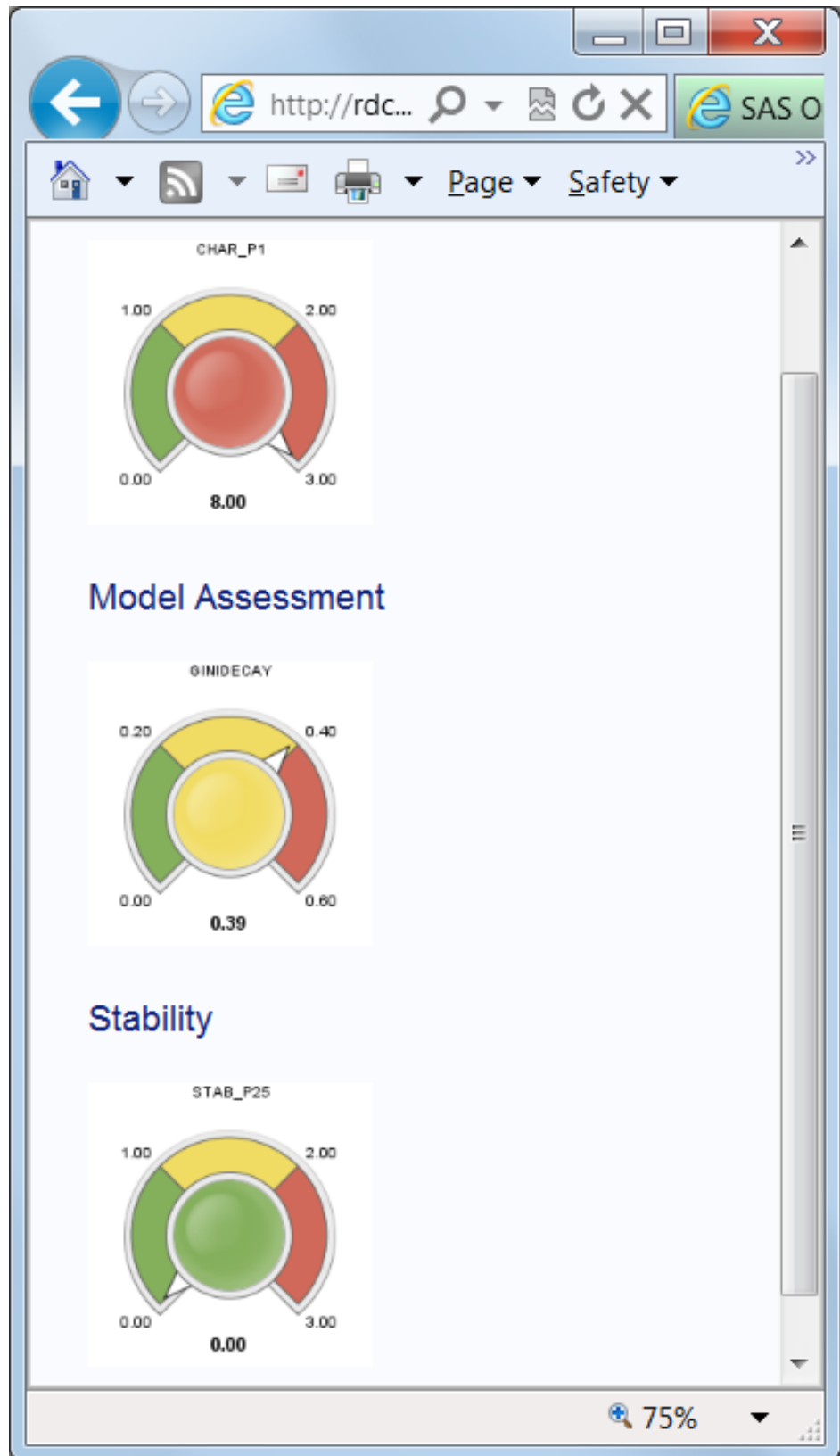
*Note:* You can also view the report by opening the index.html file directly on the SAS Workspace Server dashboard reports location (for example, `c:\SAS\Config\Lev1\AppData\SASModelManager12.3\Dashboard\report`).

The screenshot shows a web browser window with the URL <http://rdc...> and the SAS logo. The page title is "Project Reports Index". Below the title is a table with the following data:

Time	Status	Project Indicator	Report
2013Q1		Number of predictors with deviation index exceeding 0.1	<a href="#">KPI Dashboard Report</a>
			<a href="#">KPI Detail Report</a>
			<a href="#">KPI Trend Dashboard Report</a>
			<a href="#">Monitoring Report</a>
2012Q4		Number of predictors with deviation index exceeding 0.1	<a href="#">KPI Dashboard Report</a>
			<a href="#">KPI Detail Report</a>
			<a href="#">KPI Trend Dashboard Report</a>
			<a href="#">Monitoring Report</a>
2012Q3		Number of predictors with deviation index exceeding 0.1	<a href="#">KPI Dashboard Report</a>
			<a href="#">KPI Detail Report</a>

The browser's status bar at the bottom shows a magnifying glass icon and "100%".

3. Select a link from the **Report** column to view the report details.



For more report examples, see “Dashboard Report Examples” on page 535.

**See Also**

- “Overview of Project Dashboard Reports” on page 313
- “Create a Dashboard Report Definition” on page 314
- “Generate Dashboard Reports” on page 319

---

## Edit a Dashboard Report Definition

To edit a dashboard report definition:

1. Right-click the project folder in the Project Tree and select **Dashboard Report Definition** ⇒ **Edit**. The Edit Dashboard Report Definition window appears.
2. (Optional) To copy indicators from another project, click **Copy Indicators**. The Copy Indicators from Another Project window appears.
  - a. Select the project name to copy the indicators from another project.
  - b. Click **OK**.
3. Click **Add**. The Add Indicator window appears.
  - a. From the **Template** list box, select a template. For information about the selected template, click **Details**.
  - 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 field is available.
  - d. Enter values for the **Normal**, **Warning**, and **Alert** range definitions.
  - e. Click **OK**.
4. Repeat Step 3 for each indicator that you want to add. To edit an existing indicator, select the indicator, and then click **Edit**. To delete an existing indicator, select the indicator and click **Delete**.
 

*Note:* At least one indicator is required to continue.
5. Select one **Category Indicator** for each category, select one indicator as the **Project Indicator**, and then click **Next**.
 

*Note:* The indicator that you select as a project indicator must also be a category indicator.
6. Select a value from the **Project status** list box, enter a value for **E-mail address**, and then click **Add**. Repeat this step as needed. Each recipient will receive an e-mail notification about the status. To delete an e-mail notification, select an item from the list, and click **Delete**. Click **Next**.
7. By default, all of the report types are selected. To change the types of reports, follow these steps:
  - a. To add a report type, select a value from the **Report type** list box and click **Add**.
  - b. To delete a report type, select a value from the **Report description** list and click **Delete**.
8. Click **Finish**.

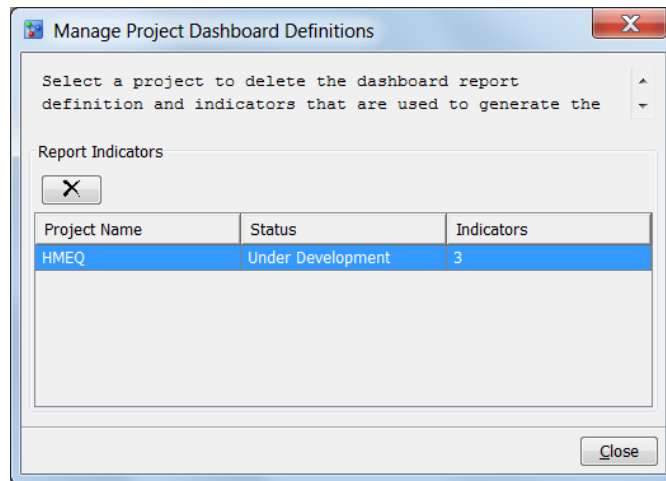


---

## Manage All Project Dashboard Definitions

The Manage Project Dashboard Definition window lists all dashboard report definitions in SAS Model Manager. To delete one or more dashboard report definitions:

1. Select **Tools** ⇒ **Manage Project Dashboard Definitions**. The Manage Project Dashboard Definitions window appears.



2. For each dashboard report definition that you want to delete, select the project and click **X**.
3. Click **Close**.

### See Also

[“Edit a Dashboard Report Definition” on page 324](#)

---

## Delete a Project Dashboard Report Definition

To delete a dashboard report definition for a project:

1. Right-click the project folder and select **Dashboard Report Definition** ⇒ **Delete**.
2. In the **Delete Dashboard Report Definition** window, select **Yes**.
3. Click **Close**.

### See Also

[“Edit a Dashboard Report Definition” on page 324](#)



## Chapter 20

# Retraining Models

---

<b>Overview of Retraining Models</b> . . . . .	<b>327</b>
<b>Prerequisites for Retraining a Model</b> . . . . .	<b>328</b>
<b>Define a Model Retrain Task</b> . . . . .	<b>328</b>
<b>Execute a Model Retrain Task</b> . . . . .	<b>333</b>
<b>Viewing Retrained Models and Model Comparison Reports</b> . . . . .	<b>334</b>
View Retrained Models . . . . .	334
View Model Comparison Reports for Retrained Models . . . . .	335

---

## 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 for the selected project is selected for retrain. 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. The task is executed on the SAS Application Server that is specified. The report folder is stored on the SAS Content Server.

*Note:* Only models that are created by using SAS Enterprise Miner, SAS/STAT, SAS/ETS, and R models can be retrained.

To retrain models in SAS Model Manager:

- 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 define and execute a model retrain task, complete the following prerequisites:

- If you want to retrain the project champion model, ensure that the champion model is set. For more information, see [“Champion Models” on page 216](#).
- Verify that the training data set that you want to use as the training data source has been registered in the SAS Metadata Repository, either by using SAS Management Console or by defining a libref using the Edit Start-up Code window. For more information, see [“Train Tables” on page 36](#).
- Verify that the appropriate project and model properties are set. Here is a list of properties.

### Classification Model Project Properties

- Training Target Variable
- Target Event Value
- Class Target Level
- Output Event Probability Variable

### Prediction Model Project Properties

- Training Target Variable
- Class Target Level
- Output Prediction Variable

### Model Properties

- Score Code Type

For more information, see [“Specific Properties for a Project” on page 505](#) and [“Specific Properties for a Model” on page 514](#).

- 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 146](#).
- Verify that the retrain file that is specified in the model template exists in the model folder that you want to retrain and that the content is correct.

---

## Define a Model Retrain Task

To define a model retrain task:

1. Right-click the project name and select **Define Model Retrain Task**. The Define Model Retrain Task wizard appears.

Define Model Retrain Task

Select Models for Retrain

Step 1 of 3

Select Models

Select All

Select	ID	Name	Version	Type	Champion
<input checked="" type="checkbox"/>	MMRoot/Eliz/...	Tree1	2013	Classification	YES
<input type="checkbox"/>	MMRoot/Eliz/...	Reg1	2013	Classification	NO
<input type="checkbox"/>	MMRoot/Eliz/...	HMEQ_STAT...	2013	Classification	NO

Model Retrain Settings

Destination version for new models: 2013

Training data source: MMLibrary.HMEQ\_TRAIN

SAS application server: SASApp

Report folder:

Retrain result folder:

Register new trained model  Trace on

Data Processing Method

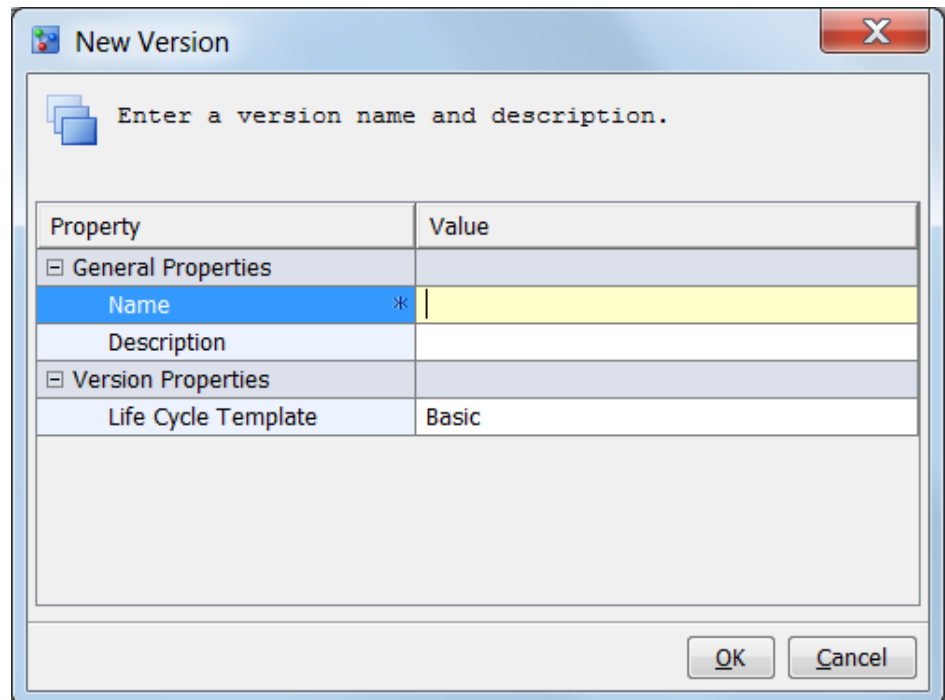
Standard configuration  High performance configuration

2. (Optional) Select one or more models to be retrained. To select all models, select the **Select All** check box.
3. Select a destination version for new models.

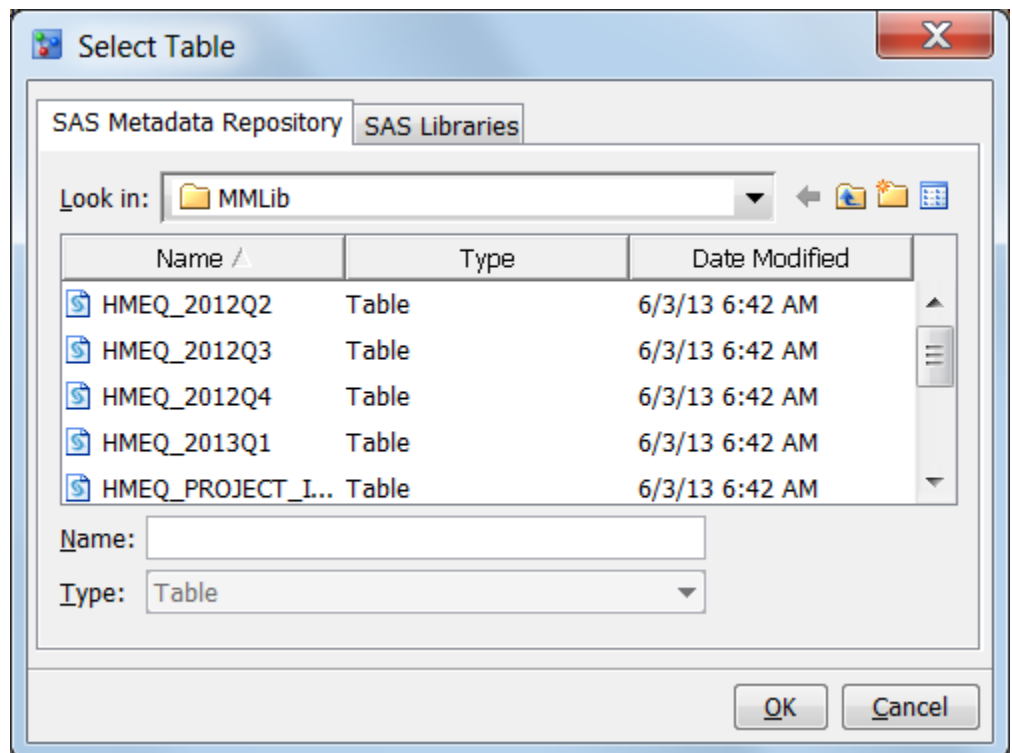
*Note:* If you do not select a destination version, the default location is used for the destination of the new retrained models.

(Optional) To create a new version to store the new retrained models:

- a. From the Select Models for Retrain page of the wizard, click **New**. The New Version window appears.



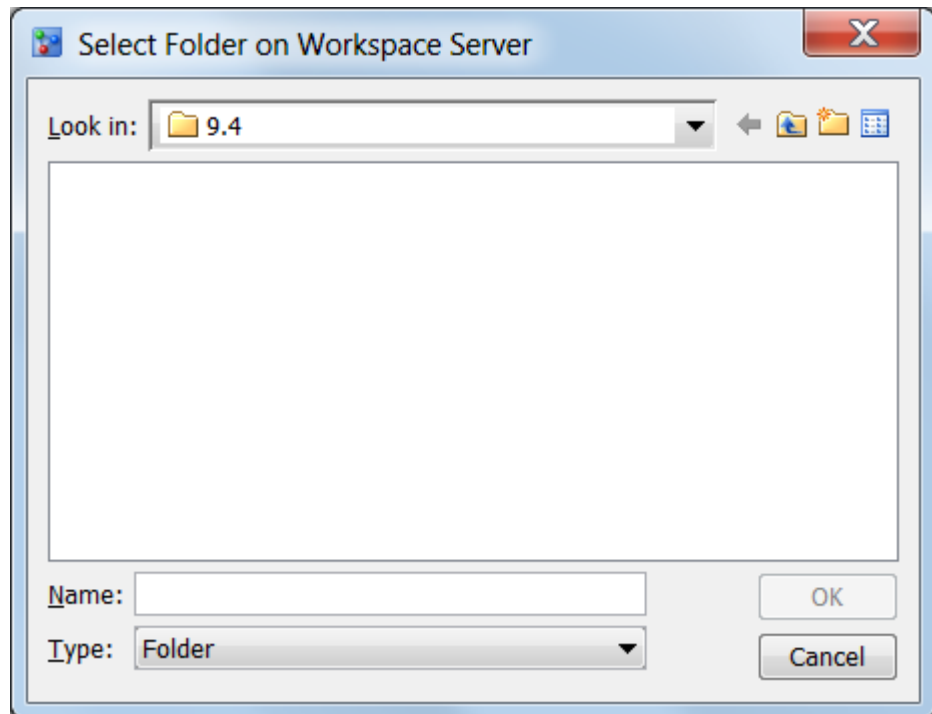
- b. Enter a name of the new version and select a life cycle template. Entering a description of the new version is optional.
  - c. Click **OK**. You are then returned to the Define Model Retrain Task wizard.
4. Click **Browse** to select a training data source from the **SAS Metadata Repository** tab or the **SAS Libraries** tab.



Click **OK**. You are then returned to the Define Model Retrain Task wizard.

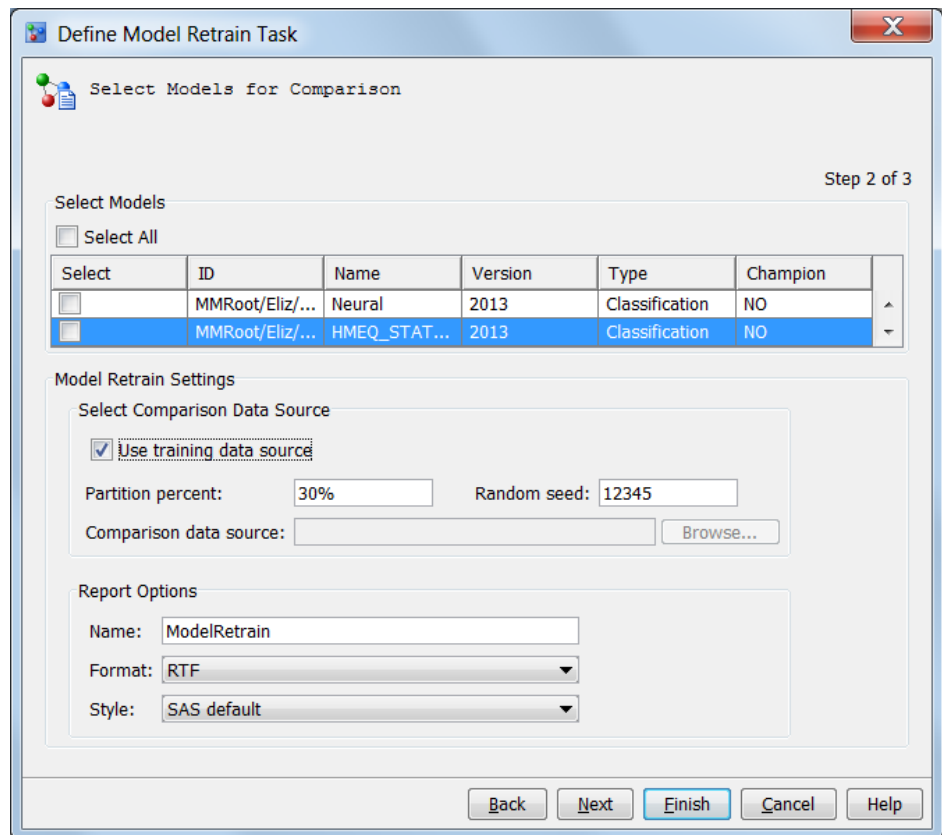
5. Select a SAS Application Server.

6. (Optional) Specify a location to store the report. By default, the report is stored in the SAS session's working folder on the SAS Workspace Server. To change the location, click **Browse**. The Select Folder on Workspace Server window appears.



Select a folder and click **OK**.

7. (Optional) Specify where to store the retraining results. Click **Browse**. The Select Folder on Workspace Server appears. Select a folder and click **OK**
- 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.
9. (Optional) Select **Trace On** to print trace information to the SAS log file.
10. Select a data processing method of either **Standard configuration** or **High-performance configuration**.
- Note:* To use the high-performance configuration, you must license the SAS High-Performance Analytics server.
11. Click **Next**. The **Select Models for Comparison** page appears.



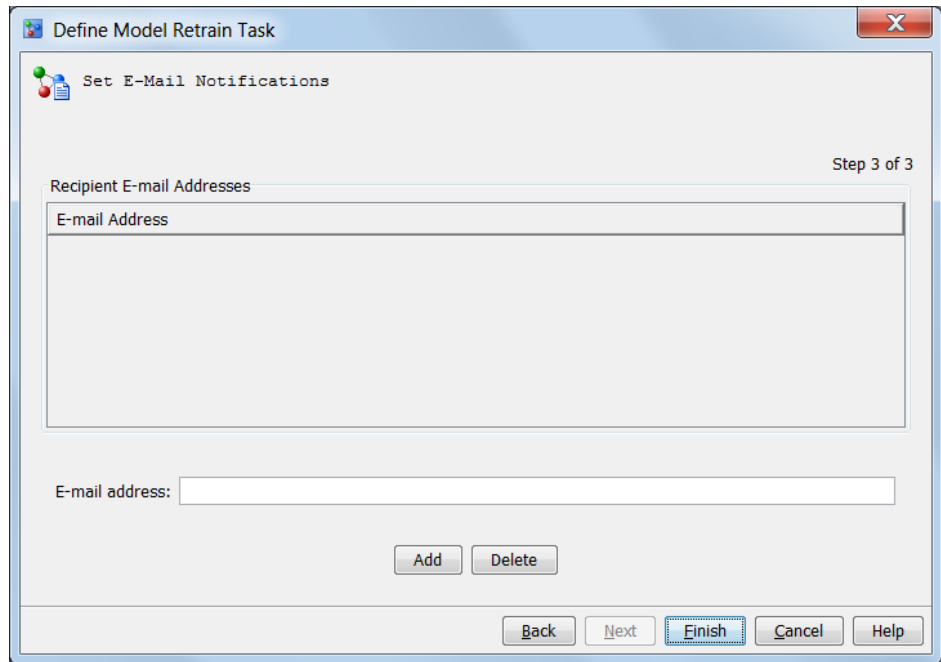
12. Select the models to be compared to the retrained model. To select all models, click **Select All**.
 

*Note:* If you do not select a model, the champion model in the default version for the project is used to perform the comparison.
13. Select a comparison data source. Take one of the following steps:
  - Select **Use training data source** if you want to use the training data 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 **Browse** to select a performance data set as the comparison data source.
14. Specify the report options.
  - a. Enter a report name.
  - b. Select a format for the report output. The standard formats that are available are **RTF**, **PDF**, **HTML**, and **EXCEL**. The default is **RTF**.
 

*Note:* SAS Model Manager administrators can configure the report formats that are available using SAS Management Console.
  - c. Select a style for the report. The available styles are **SAS default**, **Seaside**, **Meadow**, and **Harvest**. The default is **SAS default**.
 

*Note:* SAS Model Manager administrators can configure the report styles that are available using SAS Management Console.
15. Click **Next**. In the **Set E-Mail Notifications** page, you can specify e-mail recipients to receive notification the retrained model.





(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 333](#)

---

## Execute a Model Retrain Task

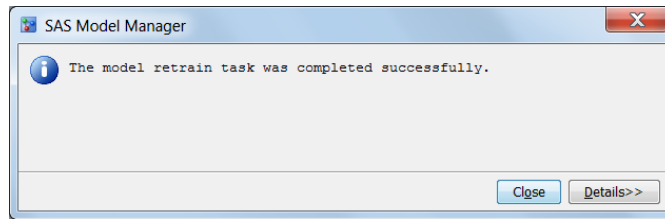
The prerequisites for retraining a model must be completed and a model retrain task must be defined before you can execute a model retrain task.

To execute a model retrain task:

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 334](#)

---

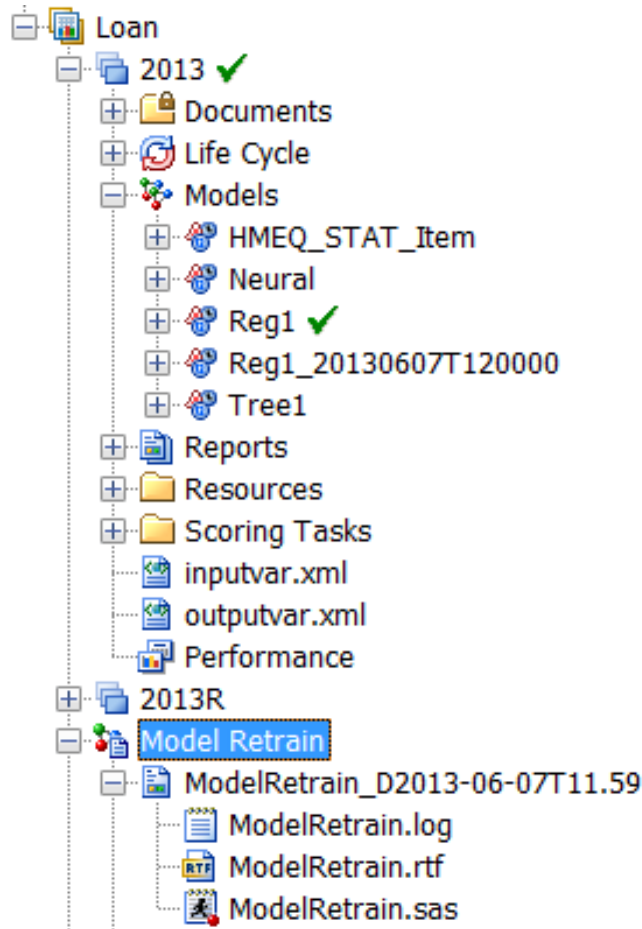
## Viewing Retrained Models and Model Comparison Reports

After a model retrain task is executed and 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:

1. Expand the destination version node to see the new models in the **Models** folder.

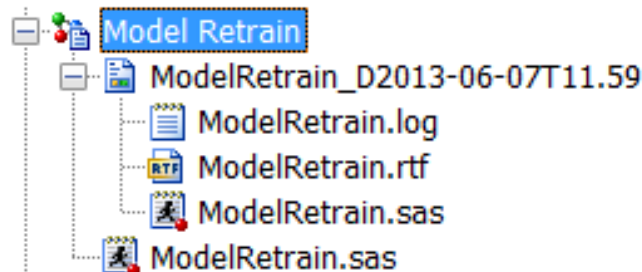


2. Expand the new retrained model folder to view its contents.

### View Model Comparison Reports for Retrained Models

To view a model comparison report:

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 RTF file that was sent as an attachment 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 544.

## Part 6

---

# Combining Multiple Reports

*Chapter 21*

**Aggregated Reports** ..... 339



## Chapter 21

# Aggregated Reports

---

<b>About Aggregated Reports</b> .....	<b>339</b>
<b>Create an Aggregated Report</b> .....	<b>340</b>
<b>View an Aggregated Report</b> .....	<b>341</b>
<b>Edit an Aggregated Report Definition</b> .....	<b>341</b>
<b>Delete an Aggregated Report</b> .....	<b>342</b>

---

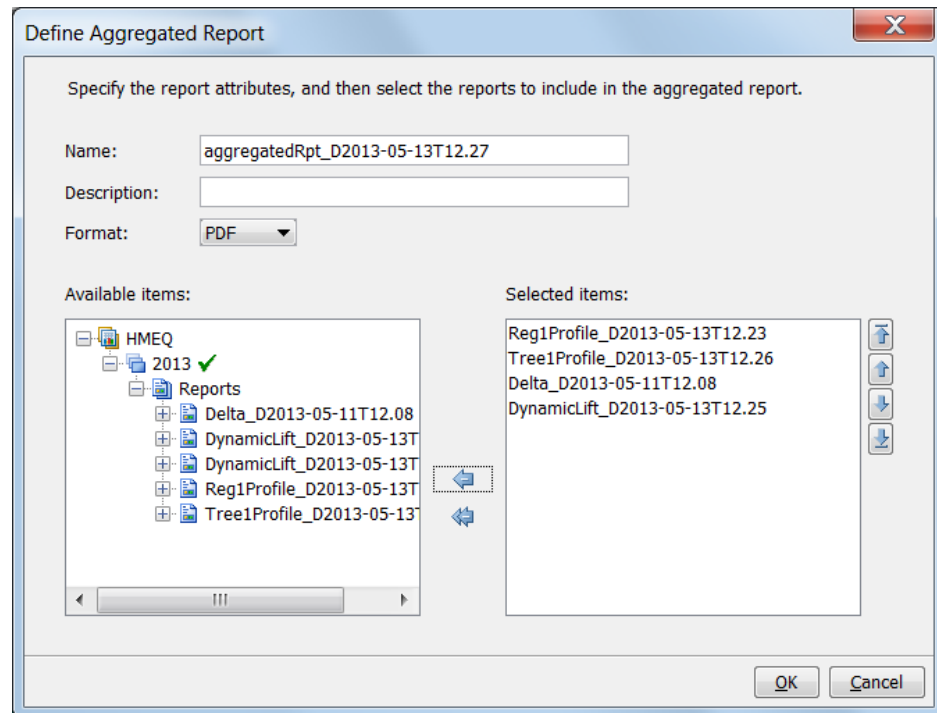
## About Aggregated Reports.

You can combine multiple reports from the **Reports** node to create a single, aggregated report. SAS Model Manager administrators and advanced users can create an aggregated report in two steps. First, you open the Define Aggregated Reports window from an organizational folder, a project, or a version. Using reports that reside in the **Reports** folder, you select the reports that you want in your aggregated report to create an aggregated report definition. Next, you generate the aggregated report using the aggregated report definition. The format of the report can be PDF, HTML, or RTF. Aggregated reports are stored in the **Documents** folder.

Ad hoc reports, LGD reports, and PD reports cannot be added to an aggregated report.

If a report contains an error, the report is not listed in the **Reports** folder.

Here is the **Define Aggregated Report** window.



These are the tasks that you perform for aggregated reports:

- “Create an Aggregated Report” on page 340
- “View an Aggregated Report” on page 341
- “Edit an Aggregated Report Definition” on page 341
- “Delete an Aggregated Report” on page 342

---

## Create an Aggregated Report

To create an aggregated report, you must have existing reports in the **Reports** node. You first create an aggregated report definition, and then generate the report.

To create an aggregated report:

1. Right-click an organizational folder, a project, or a version, and select **Define Aggregated Report**.
 

*Note:* The **Define Aggregated Report** pop-up menu item is available to only SAS Model Manager administrators and advanced users.
2. In the **Name** field, enter a name for the report.
3. (Optional) In the **Description** field, enter a description for the report.
4. Click the **Format** list box and select an output format. The default is PDF.
5. In the **Available items** box, expand the organizational folder, the project, or version to show the **Reports** node.
6. To select all reports in the **Report** node, click the double right arrow. If you do not want all reports, skip this step.



7. For each individual report that you want to include in the aggregated report, expand **Reports** and select the report in the **Available items** box. Click the single right arrow. The report appears in the **Selected items** box.
8. To remove reports from the **Selected items** box, click the single left arrow to remove one report or click the double right arrow to remove all reports.
9. To order the reports, select a report and use the up and down arrows.
10. When all of the reports are in the **Selected items** box and in the correct order, click **OK**. An object for the aggregated report definition and an **Output** node appear in the **Documents** folder.
11. Expand the **Documents** folder.
12. Right-click the aggregated report definition name and select **Generate Aggregated Report**. A message appears to indicate whether the report was generated. If it was successful, click **Close**. If it was not successful, click **Details** to view the SAS log.
13. To view the report, expand the **Output** folder for the aggregated report, right-click the report, and select **Open**.

---

## View an Aggregated Report

To view an aggregated report, follow these steps:

1. Expand the **Documents** folder.
2. For the report that you want to view, expand the report definition folder and the **Output** folder.
3. Right-click the report and select **Open**.

---

## Edit an Aggregated Report Definition

To edit an aggregated report, follow these steps:

1. Expand the **Documents** folder. Right-click the aggregated report and select **Edit Aggregated Report**. The Edit Aggregated Report window appears.
2. Modify the report definition:
  - a. In the **Name** field, modify the name.
  - b. (Optional) In the **Description** field, enter a description for the report.
  - c. Click the **Format** list box and select an output format. The default is PDF.
  - d. To add reports to the report definition, select the report in the **Available items** box and click the single right arrow.
  - e. To delete reports from the report definition, select the report from the **Selected items** box and click the single left arrow. To remove all reports, click the double left arrow.
  - f. To order the reports, select a report and use the up and down arrows.
  - g. Click **OK**.

- h. Right-click the aggregated report name and select **Generate Aggregated Report**. A message appears to indicate whether the report was generated. If it was successful, click **Close**. If it was not successful, click **Details** to view the SAS log.

---

## Delete an Aggregated Report

To delete an aggregated report, follow these steps:

1. Right-click the report name in the **Documents** folder and select **Delete**.
2. Click **Yes** for the confirmation message. The aggregated report definition and **Output** folder are deleted.

## Part 7

---

# SAS Model Manager Workflow Console

<i>Chapter 22</i>	
<b>Using Workflow Console</b> .....	345
<i>Chapter 23</i>	
<b>Managing Workflows</b> .....	367
<i>Chapter 24</i>	
<b>SAS Workflow Model Management Components</b> .....	379



## Chapter 22

# Using Workflow Console

---

<b>Overview of Workflow Console</b> .....	<b>346</b>
<b>User Interface Layout</b> .....	<b>346</b>
About the User Interface Layout .....	346
Navigation Pane .....	348
<b>Customizing Category Views</b> .....	<b>349</b>
Overview of Customizing Category Views .....	349
Manage Columns in a List .....	349
Sort Lists by Column Content .....	350
Searching List Content .....	352
<b>Setting Preferences</b> .....	<b>355</b>
About Setting Preferences .....	355
Global Preferences .....	355
General Preferences .....	356
Alert Notifications .....	356
<b>Working with Objects</b> .....	<b>357</b>
About the Tile Pane .....	357
Open Objects .....	357
Open Multiple Objects .....	358
Rearrange Open Objects .....	359
Save Layouts .....	359
Close Objects .....	359
<b>Viewing Workflow Activities</b> .....	<b>359</b>
<b>Working with Workflow Activities</b> .....	<b>360</b>
<b>Editing Activity Properties</b> .....	<b>361</b>
<b>Working with Comments</b> .....	<b>362</b>
About Comments .....	362
Adding Comments .....	363
Respond to a Topic .....	364
Attach a File .....	364
Search for Comments and Replies .....	365
<b>Viewing Workflow Milestones</b> .....	<b>365</b>

---

## 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 models at the version level of a project. A SAS Model Manager administrator or a SAS administrator uses SAS Workflow Studio to define workflow definitions and to make them available to SAS Model Manager for use. Workflow definitions contain the set of activities (or tasks), participants, policies, statuses, and data objects that comprise a business task. Activities can be associated with a milestone and model management components. After the workflow definitions are made available, the SAS Model Manager administrator uses the Workflow Console to create a workflow to be used with SAS Model Manager. A *workflow* is a copy of a workflow definition. Each workflow consists of activities. Activities can contain 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.

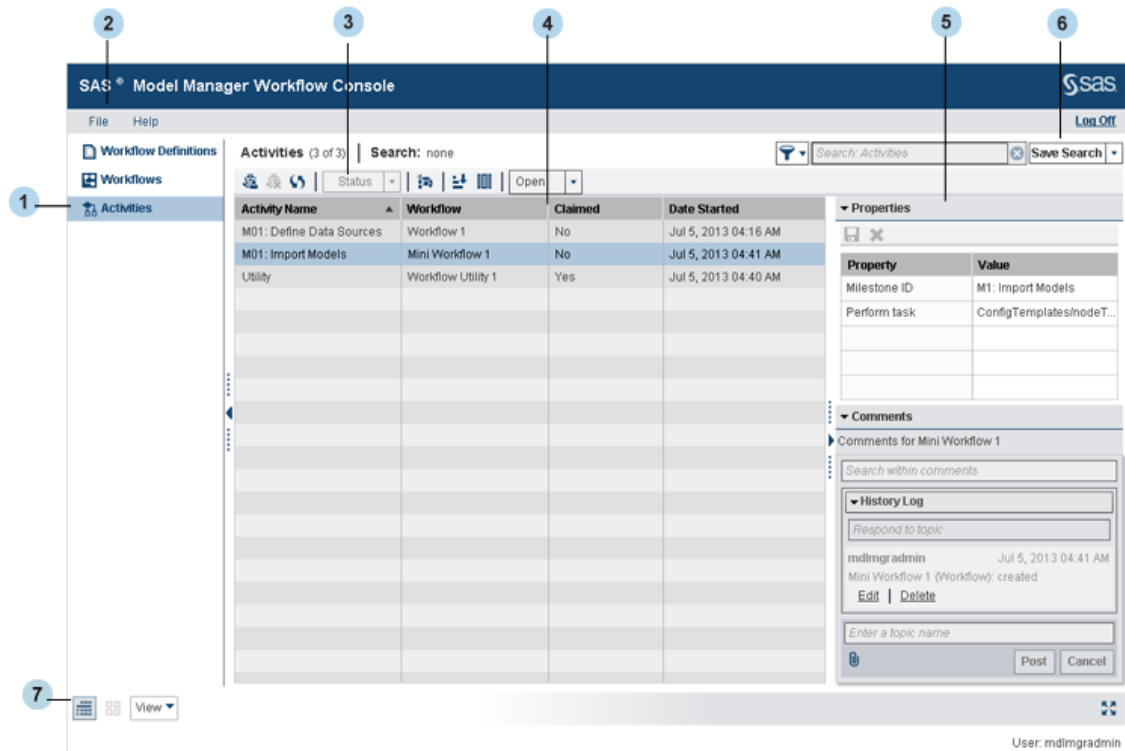
From the SAS Model Manager client application, you can launch the Workflow Console to create a new workflow or view the workflow for a version, manage all workflows, and view your workflow inbox to work with activities, depending on the user permissions. The option that is selected and the user permissions determine the category view and content that are displayed when the Workflow Console is launched. SAS Model Manager administrators can view the Workflow Definitions, Workflows, and Activities category views of the Workflow Console. Other SAS Model Manager 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 search bar, details panes, and a tile pane.









- 1 **Category 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 workflow definitions, workflows 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 **Details 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 Workflows category views.
- 6 **Search bar:** Use the search bar to specify the types of items that you want to display in the list area.

- 7 **Tile pane:** Use the tile pane 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 357.




Although window controls change depending on the workspace category, some controls are common to all categories. Table 22.1 describes the common window controls.

**Table 22.1** Window Controls

Control	Name	Description
	Collapse	Hides the object with which it is associated.
	Expand	Shows the object with which it is associated.
	Resize Horizontal	Horizontally resizes a pane when you drag the pointer (  ) to the left and right.
	Resize Vertical	Vertically 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 Workflow 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 Workflows category view contains a list of the workflows. To view detailed information for a workflow, double-click a workflow 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 <b>Started</b> . To view all of the properties for an activity or start model management tasks, 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:

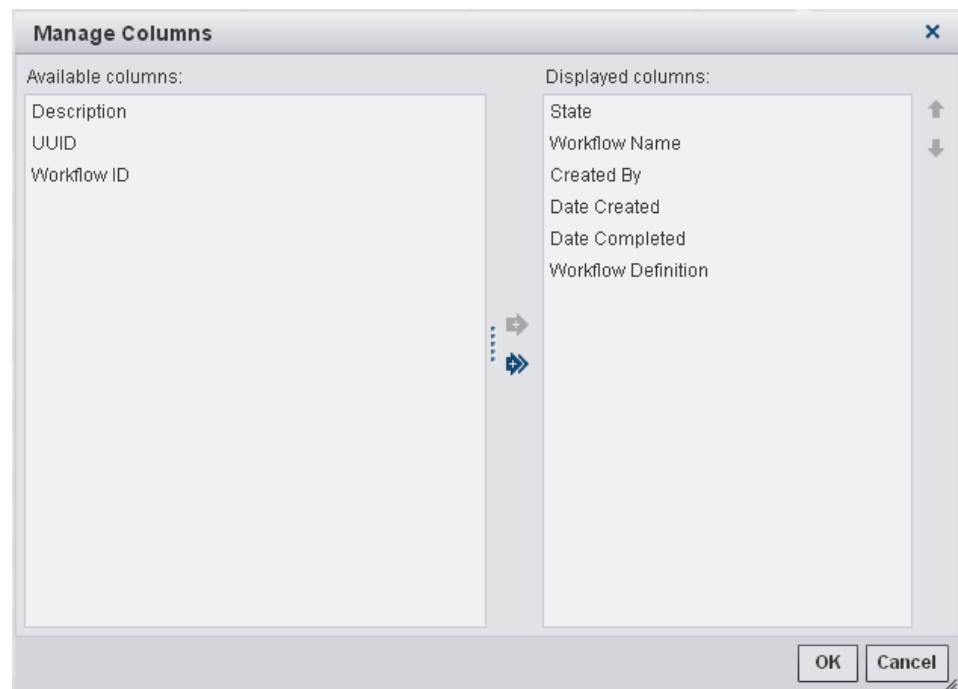
- manage which columns are displayed, and determine the order of the columns in a list
- set the sort order and direction of the attributes
- create, save, and manage searches

### 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 22.1** 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 22.2** 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 search 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 search still searches for name values and returns results based on the contents of that column. For more information about using searches, see “[Searching List Content](#)” on page 352.

## 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.

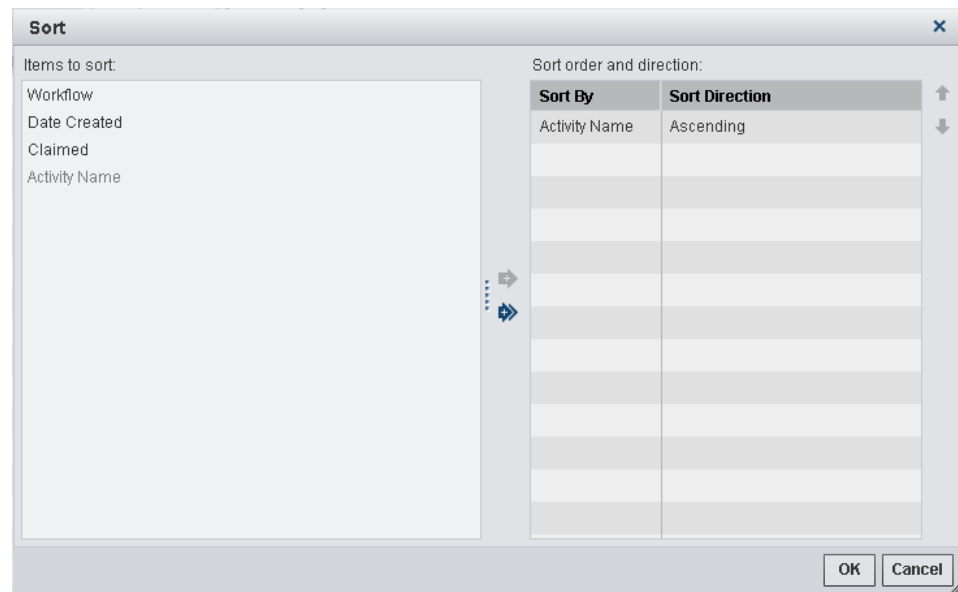
To indicate that a column is sorted, a sort icon appears next to the column label and shows whether the sort is in ascending () or descending () order. The sort icon does not appear next to the label until that column is sorted.

### 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 22.2** Sort Window

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

**Table 22.3** 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**.

## Searching List Content

### About Searching List Content

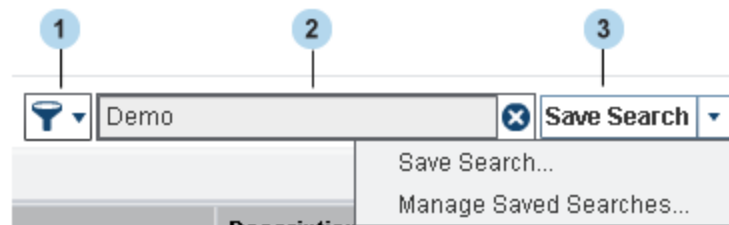
You can search the list in a category view to view only particular workflow definitions, workflows, or activities. For example, suppose you want to see the entry for a version and you use the UUID to identify the version.

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

- Create and save search criteria for reuse
- Modify a saved search
- Apply a saved search
- Manage saved searches

Here are the different parts of the search bar.

**Figure 22.3** Search Bar



- 1 **Additional search options:** Use the additional search options menu to filter the results by dates that are available in the list.
- 2 **Search criteria:** Use the search text box to specify search criteria.
- 3 **Search menu:** Use the search menu to save and manage searches. The menu expands to display the last ten saved searches by default. The order can be different based on the order that is specified in the Manage Saved Searches window.

### Create and Save Search Criteria for Reuse

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

To save search criteria:

1. On the search bar, specify search criteria in the search 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:* Search criteria are not case-sensitive.

2. Click **Save Search**. The Save Search window appears.

3. In the Save Search window, specify a name for the search and an optional description. Click **Save**.

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

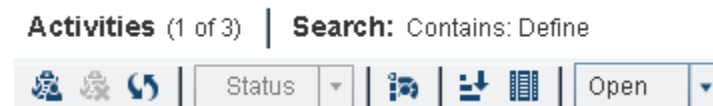
4. In the confirmation window, click **OK** to replace the existing search.

### **Search Information Area**

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

Here is a sample of the search information area. In this sample, six activities out of ten contain the text “test”.

**Figure 22.4** Search Information Area



### **Modify a Saved Search**

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

To create a new search and replace the existing one:

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

### **Apply a Saved Search**

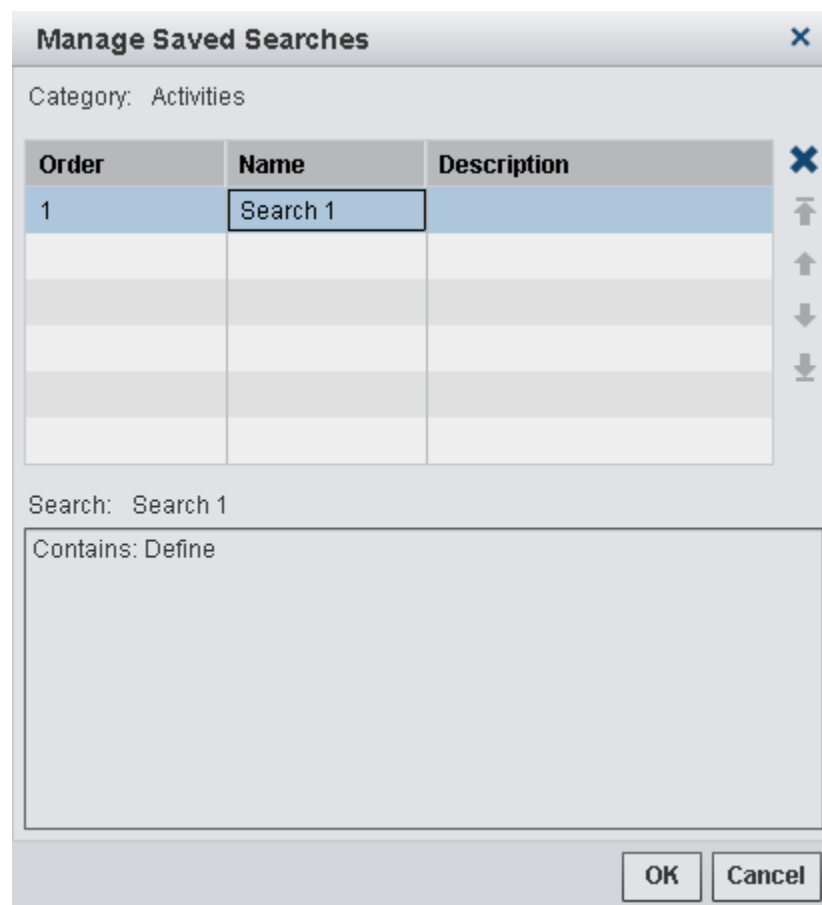
To apply a saved search, click the search menu down arrow, and select the search from the list. If there are more than ten saved searches:






1. Click **More** on the search menu to display the list of all saved searches.
2. Select the search that you want to use, and click **OK** to apply the search criteria to the current list.

### **Manage Saved Searches**

You can rename, reorder, and remove saved searches as necessary by using the Manage Saved Searches window.

1. Click the down arrow in the save search menu and select **Manage Saved Searches**. The Manage Saved Searches window appears.



2. To rename a saved search, 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 searches appear in the search menu. To reorder the list, use the move down ( and ) and move up ( and ) buttons.
4. To remove a search, select the search 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. If you receive an error change the updated locale back to the original locale for the user.

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. The default is the browser locale.
- **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 7 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:

- **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 4 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 7 of *SAS Intelligence Platform: Middle-Tier Administration Guide*.



## Working with Objects

When you open objects, you can add them to the tile pane 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 Tile Pane



The tile pane provides quick access to open objects and to tools that control how objects are displayed. By using the tile pane, 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 tile pane is workspace-specific. When you switch to a different workspace, the tile pane changes to reflect the open objects in that workspace.

Here is a description of the buttons in the tile pane.

**Table 22.4** Tile Pane Buttons

Button	Name	Description
	View object list	Returns to the category view and displays the list of objects.
	View layout	Tiles the display of open objects. From the drop-down list, select the objects to tile and select <b>Show Details</b> to display object details in the thumbnail images.
View actions button	View actions	Displays the View actions menu that contains the following options: <ul style="list-style-type: none"> <li>• <b>Save Layout</b></li> <li>• <b>Open Layout</b></li> <li>• <b>Show All Items</b></li> <li>• <b>Close All</b></li> </ul>

### 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 tile pane. This option enables you to work with the object immediately.
- **Add:** Adds a selected object to the tile pane (when one or more objects are open).
- **Sent to Tile Pane:** Opens the object's thumbnail image in the tile pane. This enables you to continue selecting other objects in the object view.

**TIP** You can drag an object from a list to the tile pane.

- (Optional) To switch to the object view, click the **View object list** button in the tile pane.

**TIP** To obtain a list of open objects, click **View**. 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 tile pane.

## Open Multiple Objects

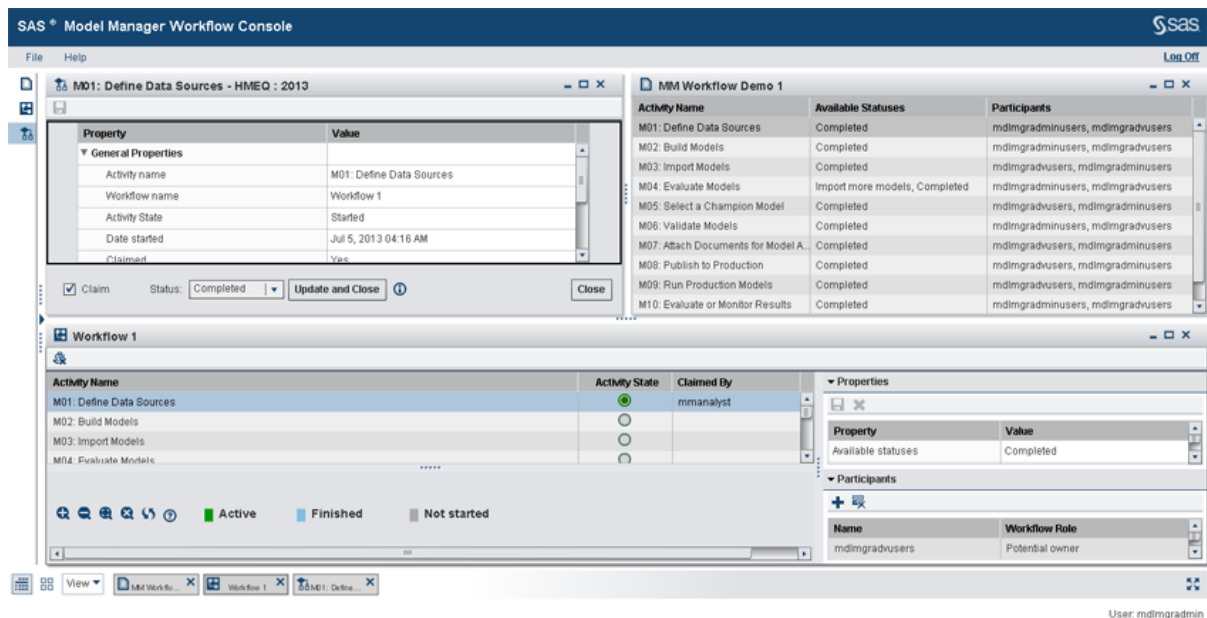
To open multiple objects, use either of the following methods:

- 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 tile pane.
- Select an object and click **Open**, click the **View object list** 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 tile pane. Repeat this process to continue opening objects.

**TIP** To display all open objects, rather than the most recently opened object, click the **View actions** button on the tile pane, and select **Show All Items**.

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 tile pane.

Figure 22.5 Tiled Open Items



## 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 actions** on the tile pane.

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 in the tile pane for quick access when necessary.

- Add objects to the object view by dragging their icons from the tile pane 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 **View actions** button in the tile pane, select **Save Layout**, and provide a name for the saved layout.

To open a saved layout, click the **View actions** button on the tile pane, 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 **X** in the upper right corner of the thumbnail image. This closes the object in both the object view and in the tile pane.

*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 in the tile pane.

- If the object thumbnail image is visible in the tile, click **X** in the upper right corner of the thumbnail image. This action removes the object thumbnail from the tile pane.

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 .

SAS Model Manager 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.

*Note:* If another users claims an activity, it no longer appears in your activities list. A SAS Model Manager administrator that has the workflow role of business administrator can view the activity and modify the activity properties from the Workflow details view.

The screenshot displays the SAS Model Manager Workflow Console. The main area shows a table of activities:

Activity Name	Workflow	Claimed	Date Started
M01: Define Data Sources	Workflow 1	Yes	Jul 5, 2013 04:16 AM
M01: Import Models	Mini Workflow 1	No	Jul 5, 2013 04:41 AM
Utility	Workflow Utility 1	Yes	Jul 5, 2013 04:40 AM

The right-hand pane shows details for the selected activity, including a Properties section with a table:

Property	Value
Date modified	Jul 9, 2013 04:08 AM

Below the properties is a History Log section with the following entries:


- mdlmgradmin Jul 9, 2013 03:41 AM: M01: Define Data Sources (Activity): removed participant "mmanalyst" as potential owner. [Edit](#) | [Delete](#)
- mdlmgradmin Jul 9, 2013 03:42 AM: M01: Define Data Sources (Activity): assigned participant "mmanalyst" as potential owner. [Edit](#) | [Delete](#)
- mmanalyst Jul 9, 2013 04:08 AM: M01: Define Data Sources (Activity): claimed. [Edit](#) | [Delete](#)


## 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 properties
- release an activity
- mark an activity completed
- view the history log for an activity
- view dashboard reports

To complete an activity:


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 377](#).

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 361](#).
3. (Optional) Double-click an activity to view the activity details. From the activity details view you can modify activity properties or perform the model management tasks that are associated with the activity.

*Note:* If you did not claim the activity from the Activities category view, you can select the **Claim** check box in the activity status bar.

The following actions can be performed from the activity status bar.


- Select the information button  to view the description of the activity. The description window can contain a brief description of the activity or special instructions. The information button is displayed only if a description was provided for the activity when the workflow definition was created.
- If you have no other actions to perform for the activity, you can select a status to complete the activity, and click **Update and Close**. The workflow continues to the next activity.
- Click **Close** if you want to return to the Activities category view without updating the status of the activity.

For more information about the model management tasks that might be available, see [Chapter 24, “SAS Workflow Model Management Components,” on page 379](#).

4. (Optional) Add a comment to the activity using the Comments pane. For more information, see [“Working with Comments” on page 362](#).
5. Select a status value to complete the activity. The workflow continues to the next activity.



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

From the Activities category view, you can also view the dashboard reports for all projects. Click  to view the dashboard reports in a browser window.

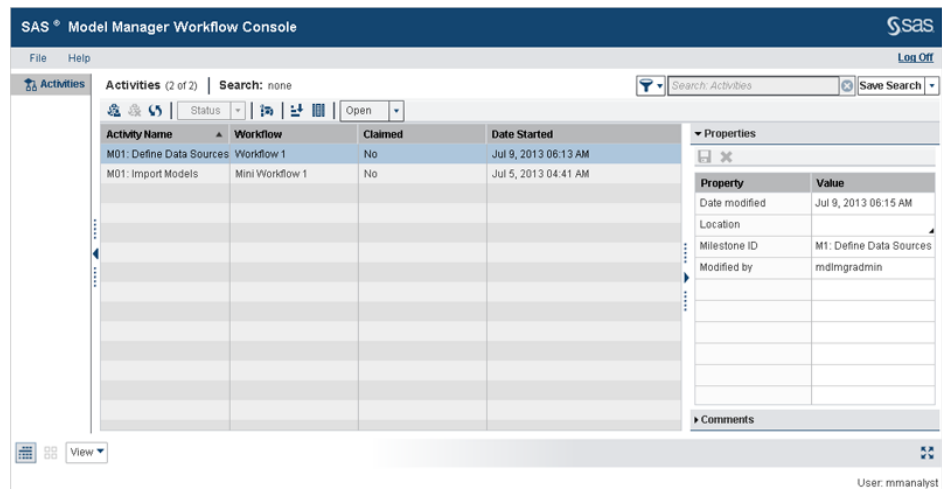
---


## Editing Activity Properties


An activity can contain properties. 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:

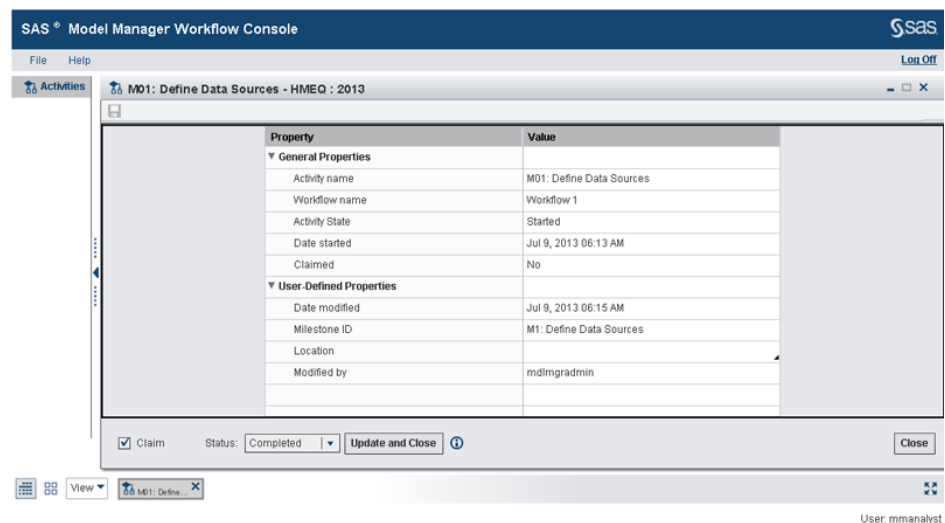
1. From the Activities category view, select an activity. The properties that are associated with the activity are displayed to the right in the Properties pane.



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 Workflows and Activities category views to share information about a workflow 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 a workflow or an activity. You can also search, sort, or filter the comments.

You can add multiple comments to an activity or workflow. 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 22.5** 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 <b>History Log</b> topic.
New Topic button	Displays the New Topic window in which you can create a new comment.

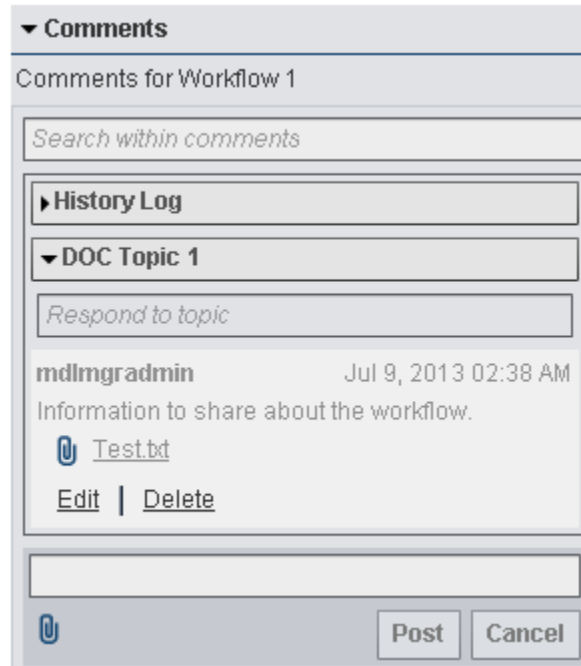
## Adding Comments

To add comments to a workflow or an activity:

1. Select the workflow or activity.
2. In the Comments pane, enter a topic name and a comment.


The screenshot shows a web interface for adding comments. At the top is a 'Comments' header with a dropdown arrow. Below it is a section titled 'Comments for Workflow 1'. This section contains a search box with the placeholder text 'Search within comments'. Below the search box is a 'History Log' section with a dropdown arrow. Under the History Log is a 'Respond to topic' button. Below that is a comment entry by user 'mdlmgadmin' dated 'Jul 5, 2013 04:16 AM'. The comment text is 'Workflow 1 (Workflow): created'. Below the comment are 'Edit' and 'Delete' links. Below the comment entry is a 'DOC Topic 1' section with a text input field containing the placeholder text 'Information to share about the workflow.'. At the bottom of the form are a paperclip icon for attachments, and 'Post' and 'Cancel' buttons.

- (Optional) Click  to add a file with the new topic. For more information, see “Attach a File” on page 364.
- Click **Post**. The new comment now appears in the Comments pane.



### Respond to a Topic


To respond to a topic:

- In the Comments pane, expand the topic thread to which you want to respond.
- Enter a comment in the box.
- (Optional) Click  to add a file to the new topic. For more information, see “Attach a File” on page 364.
- Click **Post**. The response is added below the original comment.

### Attach a File

When you add a comment or reply, you can include attachments. 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:

- Click the  button in the Comments pane.  
The **Select file(s) to upload** window appears.
- Using the **Look in** field, navigate to the location of the file.
- In the list, click the file that you want to attach. Its name appears in the **File name** box.
- Click **Open**.



The filename appears below the comment box.

*Note:* Before you post a comment or response, you can click **Remove** to the right of the attachment name to remove the attachment.

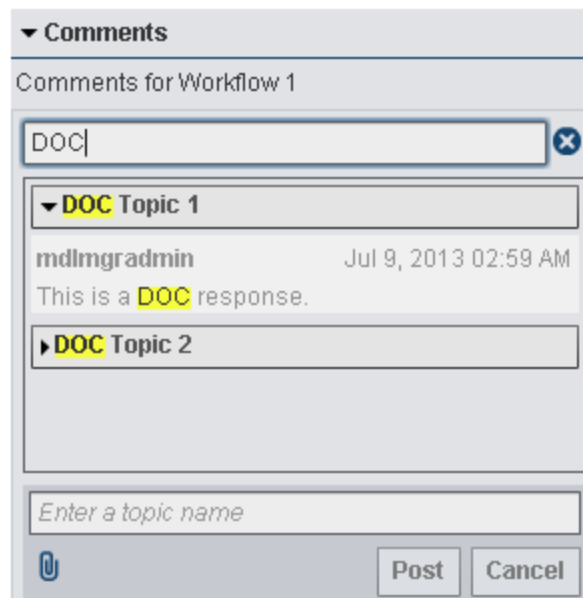
- Repeat steps 1 through 4 to attach more files.


### Search for Comments and Replies

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

To initiate a search, enter the text that you want to find in the search box and press **Enter**. The application applies the search to all the topic threads and highlights the search text in yellow.

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



To clear the search and display all topics, click .

---

## Viewing Workflow Milestones

SAS Model Manager enables you to associate milestones with workflow activities as part of the workflow process definition. Activities that are associated with a milestone appear on the Workflow Milestones tab in the version details section of the SAS Model Manager client. You can also create a Workflow Milestones report using the Workflow Console, if an activity is associated with the Create and View Reports component, or by using the New Report wizard that is available within a version from the **Reports** context menu in the SAS Model Manager client application. For more information, see “How to Associate a Milestone with a Workflow Activity” in Chapter 3 of *SAS Model Manager: Administrator's Guide*.

Here is an example of the **Workflow Milestones** tab in the version details view of the SAS Model Manager client

2013				
Properties	Input Variables	Output Variables	Workflow Milestones	Publish History
Workflow: Workflow 1				
Milestone or Task	Status	Date Started	Date Completed	Modified By
M01: Define Data Sources	✔	Jul 9, 2013	Jul 9, 2013	mmanalyst
M02: Build Models	✔	Jul 9, 2013	Jul 9, 2013	mdlmgadmin
M03: Import Models	✔	Jul 9, 2013	Jul 9, 2013	mdlmgadmin
M04: Evaluate Models	✔	Jul 9, 2013	Jul 9, 2013	mdlmgadmin
M05: Select a Champion Model	✔	Jul 9, 2013	Jul 9, 2013	mdlmgadmin
M06: Validate Models	✔	Jul 9, 2013	Jul 9, 2013	mdlmgadmin
M07: Attach Documents for Model Approval Process (MAP)	✔	Jul 9, 2013	Jul 9, 2013	mdlmgadmin
M08: Publish to Production	🟡	Jul 9, 2013		
M09: Run Production Models	⬜			
M10: Evaluate or Monitor Results	⬜			

Here is an example of the Workflow Milestones report:

Workflow Milestones Report				
Project: HMEQ Version: 2013				
Milestone or Task	Status	Date Started	Date Completed	Modified By
M01: Define Data Sources	✔	July 9, 2013 10:13:20 AM	July 9, 2013 10:33:20 AM	mmanalyst
M02: Build Models	✔	July 9, 2013 10:33:20 AM	July 9, 2013 03:18:20 PM	mdlmgadmin
M03: Import Models	✔	July 9, 2013 03:18:20 PM	July 9, 2013 03:25:00 PM	mdlmgadmin
M04: Evaluate Models	✔	July 9, 2013 03:25:00 PM	July 9, 2013 03:28:20 PM	mdlmgadmin
M05: Select a Champion Model	✔	July 9, 2013 03:28:20 PM	July 9, 2013 04:55:00 PM	mdlmgadmin
M06: Validate Models	✔	July 9, 2013 04:55:00 PM	July 9, 2013 05:20:00 PM	mdlmgadmin
M07: Attach Documents for Model Approval Process (MAP)	✔	July 9, 2013 05:20:00 PM	July 9, 2013 05:25:00 PM	mdlmgadmin
M08: Publish to Production	🟡	July 9, 2013 05:25:00 PM		
M09: Run Production Models	⬜			
M10: Evaluate or Monitor Results	⬜			

✔ Completed
🟡 In progress
⬜ Not started

## Chapter 23

# Managing Workflows

---

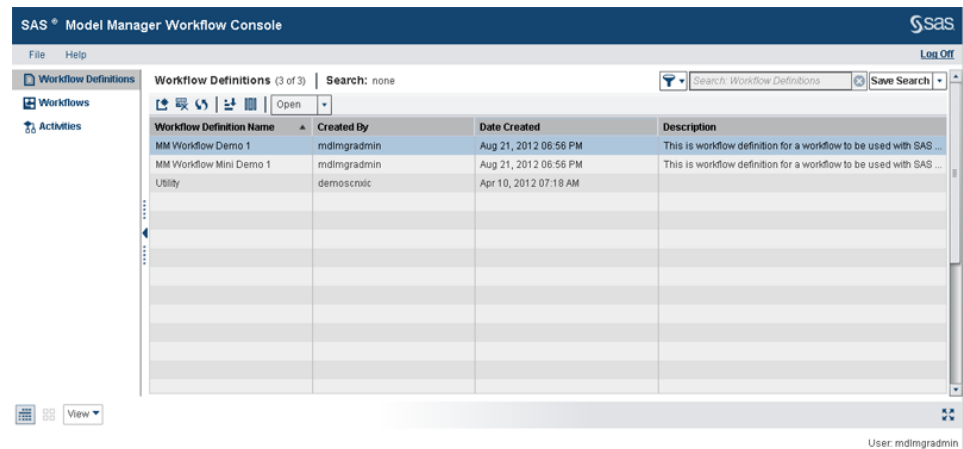
<b>Overview of Managing Workflows</b> .....	<b>367</b>
<b>Viewing Workflow Definitions</b> .....	<b>368</b>
<b>Creating a New Workflow</b> .....	<b>369</b>
<b>Viewing Workflows</b> .....	<b>370</b>
<b>Editing a Workflow</b> .....	<b>372</b>
<b>Editing Workflow Properties</b> .....	<b>374</b>
<b>Working with Workflow Participants</b> .....	<b>374</b>
Assigning Participants to Activities .....	375
Removing Participants from Activities .....	376
Releasing an Activity .....	377
<b>Terminating a Workflow</b> .....	<b>378</b>

---

## Overview of Managing Workflows

SAS Model Manager Workflow Console can be used to manage workflows. A SAS Model Manager administrator can create new workflows, view workflow definitions, and interact with activities that are associated with a workflow. 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 Workflow Definitions category view, as shown.



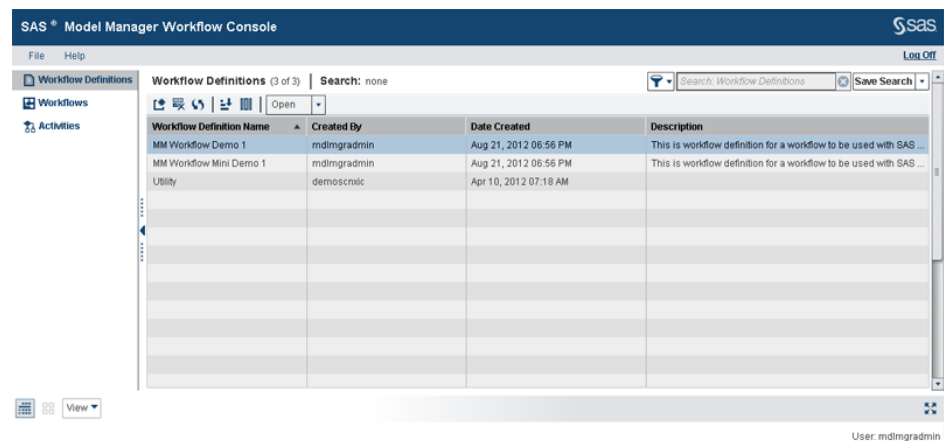
## See Also

- “Creating a New Workflow” on page 369
- “Viewing Workflows” on page 370
- *SAS Model Manager: Administrator's Guide*

## Viewing Workflow Definitions

Only a SAS Model Manager administrator can view workflow definitions. Workflow 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 a workflow from a workflow definition. You use workflows to manage the process of model development and implementation in SAS Model Manager.

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



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

Activity Name	Available Statuses	Participants
M01: Define Data Sources	Completed	mdlmggradusers, mdlmggradminusers
M02: Build Models	Completed	mdlmggradminusers, mdlmggradvusers
M03: Import Models	Completed	mdlmggradvusers, mdlmggradminusers
M04: Evaluate Models	Import more models, Completed	mdlmggradminusers, mdlmggradminusers
M05: Select a Champion Model	Completed	mdlmggradvusers, mdlmggradminusers
M06: Validate Models	Completed	mdlmggradminusers, mdlmggradvusers
M07: Attach Documents for Model Approval Process (MAP)	Completed	mdlmggradminusers, mdlmggradvusers
M08: Publish to Production	Completed	mdlmggradvusers, mdlmggradminusers
M09: Run Production Models	Completed	mdlmggradminusers, mdlmggradvusers
M10: Evaluate or Monitor Results	Completed	mdlmggradminusers, mdlmggradvusers

User: mdlmggradmin


For more information about workflow definitions, see [“Overview of Managing Workflows”](#) on page 367 and *SAS Model Manager: Administrator's Guide*.

## Creating a New Workflow

A workflow is a copy of a workflow definition. Only a SAS Model Manager administrator can create a new workflow.

To create a new workflow:

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

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

**New Workflow**

Specify the values for the new workflow.

Workflow definition: MM Workflow Demo 1

Workflow name: \*


UUID: ec51138f-0a15-1256-7a8b-852e7ed885a5

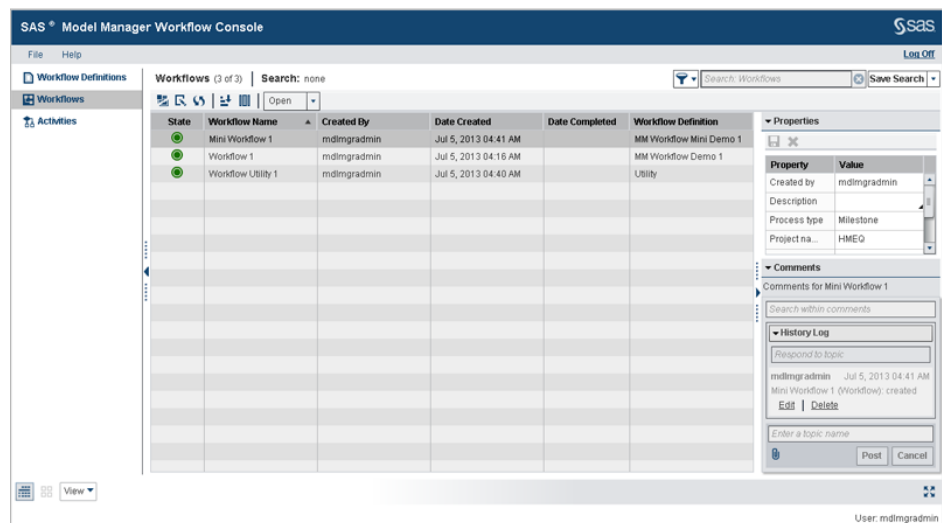
Description:

OK Cancel

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


*Note:* If the UUID is not already populated, you can copy the UUID property value for a version from the Properties view in the SAS Model Manager window. The UUID must be unique, if you enter a value that is already associated with another workflow, a confirmation window appears for you to terminate the existing workflow and replace it with the new workflow. For more information about the UUID property, see “System Properties” on page 504.

4. (Optional) Enter a description for the workflow.
5. Click **OK**. A message appears, indicating that the workflow has been successfully created.
6. Click **Close**. The new workflow is now available in the Workflows category view.
7. To view the new workflow, click **Workflows** . Select the workflow to view information that is associated with the new workflow.

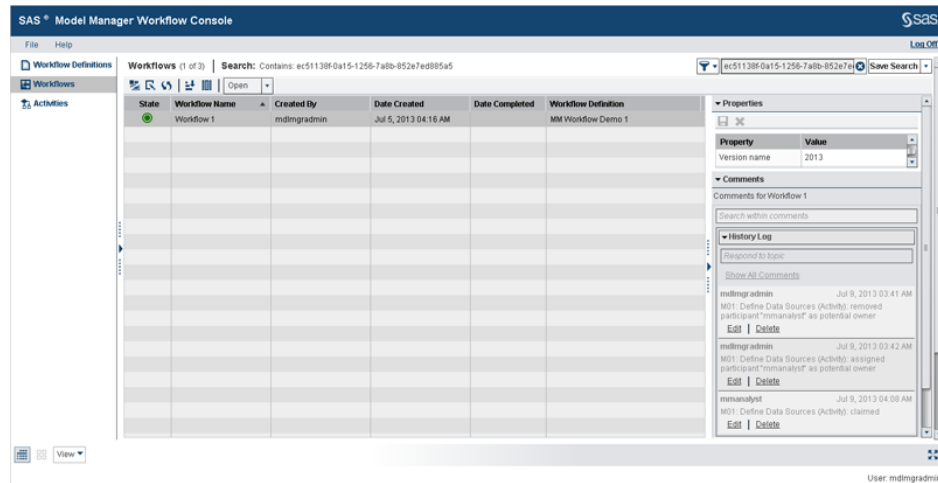


For more information about managing workflow definitions, see the *SAS Model Manager: Administrator's Guide*.

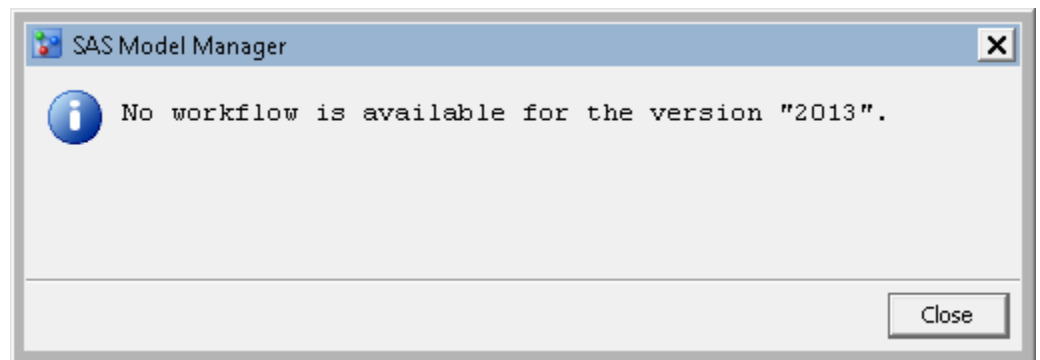
## Viewing Workflows

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

SAS Model Manager Workflow Console is launched in a web browser, and displays the Workflows category view with the value for the UUID as the applied filter. Only the workflows that are associated with the selected version are displayed in the list. You can select a workflow from the list to view the properties and comments that are associated with the workflow.



*Note:* If a workflow is not associated with the selected version, the following message appears and the Workflow Console is not launched.



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

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

To view detailed information for a workflow, double-click a workflow name. The list of activities, the activity state, and who the activity is claimed by are displayed. You can then view the properties and participants that are associated with an activity by selecting an activity name. The workflow diagram is also displayed with the current status of the workflow and its activities..

The screenshot shows the SAS Model Manager Workflow Console interface. The main window displays a workflow named "Workflow 1" with a list of activities and their states. The activities are:

Activity Name	Activity State	Claimed By
M01: Define Data Sources	Completed	
M02: Build Models	Not started	
M03: Import Models	Not started	
M04: Evaluate Models	Not started	
M05: Select a Champion Model	Not started	
M06: Validate Models	Not started	
M07: Attach Documents for Model Approval Process (MAP)	Not started	
M08: Publish to Production	Not started	
M09: Run Production Models	Not started	
M10: Evaluate or Monitor Results	Not started	

The flow diagram below the table shows the sequence of activities. M01 is completed and leads to M02. M02 leads to M03. M03 leads to M04. M04 leads to M05. M05 leads to M06. M06 leads to M07. M07 leads to M08. M08 leads to M09. M09 leads to M10. The flow diagram also shows a feedback loop from M10 back to M03 labeled "Import more models".

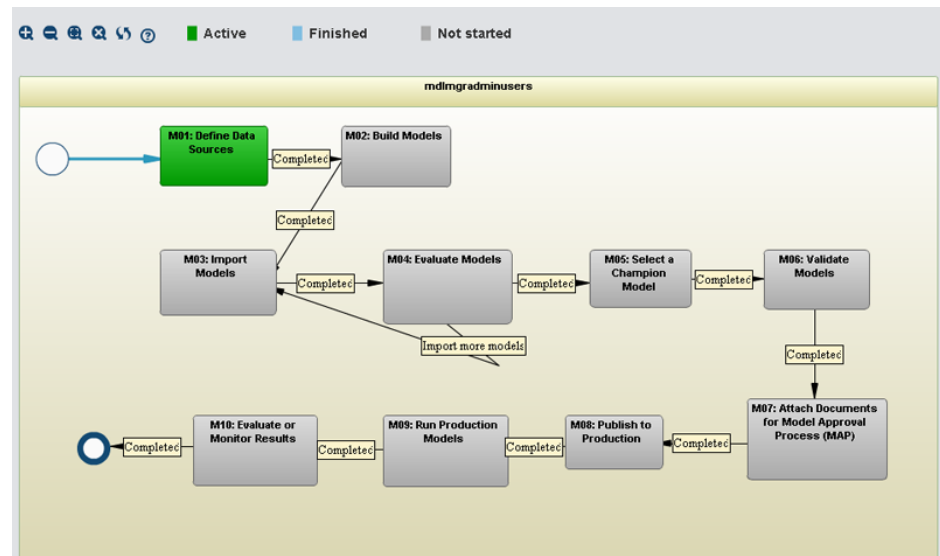
The Properties panel on the right shows the following information:

Property	Value
Available statuses	Completed
Milestone ID	M1: Define Data Sources

The Participants panel on the right shows the following information:

Name	Workflow Role
mdimgadmins	Potential owner
mdimgadministrators	Business administrator


The user is identified as "User: mdimgadmin".



For more information, see [“Working with Workflow Participants”](#) on page 374.

## Editing a Workflow

To edit a workflow:

1. From the SAS Model Manager window, right-click on a version and select **View Workflow**. Workflow Console appears.
2. From the Workflows category view, select a workflow and click . The Edit Workflow window appears.



3. Make changes to the workflow name or description.

*Note:* The UUID only can be modified if the workflow is not associated with an existing version. If the UUID field is empty or is not a valid version UUID, to associate the workflow with a version, enter the UUID for a specific version. The UUID cannot be modified after the first workflow activity has been claimed. The UUID must be unique, if you enter a value that is already associated with another workflow, a confirmation window appears for you to terminate the existing workflow and replace it with the new workflow. For more information about the UUID property, see “[System Properties](#)” on page 504.

4. Click **OK**.


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


## Editing Workflow Properties


A workflow process definition can contain properties that a user can modify for workflows and for activities. The 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 a workflow:

1. From the Workflows category view, select a workflow. The properties that are associated with the workflow are displayed in the Properties pane.

▼ Properties	
Property	Value
Created by	mdlmgadmin
Description	Test edit workflow. 
Process type	Milestone
Project name	HMEQ
Version name	2013
Version UUID	ec51138f-0a15-1256-7a8b-85...
Workflow name	Workflow 1

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 361.

## Working with Workflow Participants

From the Workflow 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 workflow definition was created.

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

1. From the Workflows category view, double-click a workflow. The Workflow details view appears.

The screenshot displays the SAS Model Manager Workflow Console interface. The main window shows a workflow definition with a list of activities and their states. The activities are:

Activity Name	Activity State	Claimed By
M01: Define Data Sources	Completed	
M02: Build Models	Not started	
M03: Import Models	Not started	
M04: Evaluate Models	Not started	
M05: Select a Champion Model	Not started	
M06: Validate Models	Not started	
M07: Attach Documents for Model Approval Process (MAP)	Not started	
M08: Publish to Production	Not started	
M09: Run Production Models	Not started	
M10: Evaluate or Monitor Results	Not started	

The workflow diagram below the table shows the sequence of activities: M01 (Define Data Sources) is completed and leads to M02 (Build Models), which leads to M03 (Import Models), M04 (Evaluate Models), and M05 (Select a Champion Model). M03, M04, and M05 are also marked as completed. The Participants pane on the right shows the following table:


Name	Workflow Role
mdimgadmins	Potential owner
mdimgadmins	Business administrator

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

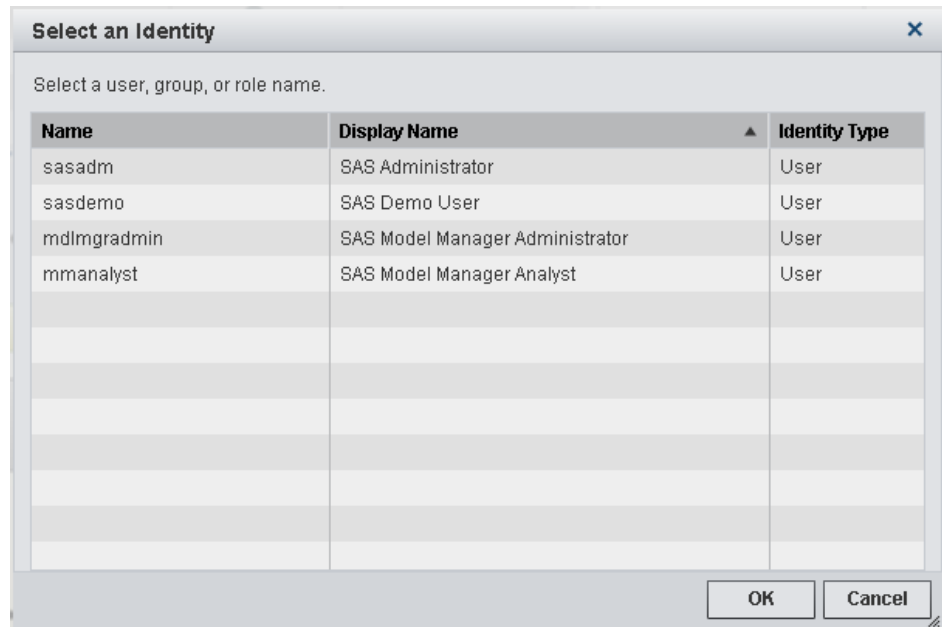
The 'Assign a Participant' dialog box is shown with the following fields and options:

- Identity type: User
- Name: \* Enter a name
- Workflow role: Potential owner

Buttons: OK, Cancel

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.



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**. The new participant is added to the **Participants** list.

6. Click **Close**.

### **Removing Participants from Activities**

To remove a participant from an activity:

1. From the Workflows category view, double-click a workflow 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.

The screenshot shows the SAS Model Manager Workflow Console interface. The main window displays a workflow named 'Workflow 1' with a list of activities (M01 to M10) and their states. Activity M01, 'Define Data Sources', is marked as 'Completed' and 'Claimed By' mdingradusers. The 'Participants' list on the right shows mdingradusers as the 'Actual owner' of the activity. The workflow diagram below the list shows a sequence of activities: M01 (Define Data Sources) -> M02 (Build Models) -> M03 (Import Models) -> M04 (Evaluate Models) -> M05 (Select a Champion Model) -> M06 (Validate Models). All activities in the diagram are marked as 'Completed'.

3. Click . A message is displayed asking if you are sure that you want to remove the participant from the activity.
4. Click **Yes**. The participant is no longer displayed in the **Participants** list.

## 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 Workflows category view, double-click a workflow name. The Workflow details view is displayed.

This screenshot shows the same SAS Model Manager Workflow Console interface, but the 'Claimed By' field for activity M01 is now 'mmanalyst'. The 'Participants' list on the right has updated to show mmanalyst as the 'Actual owner', while mdingradusers remains as a 'Potential owner'. The workflow diagram and activity list remain the same as in the previous screenshot.


2. Select an activity name, and click . The **Claimed By** value for the selected activity is cleared. If you click on the Activities category view, you should now see the released activity in your activities list.

---

## Terminating a Workflow

When you terminate a workflow, all activities that have not yet been completed in the workflow are changed to a state of **Terminated** and are no longer displayed in the Workflows category view list. After you terminate a workflow, it cannot be restarted.

To terminate a workflow:

1. From the Workflows category view, select a workflow name and click .
2. Click **Yes** to terminate the selected workflow.
3. Click **Close** to return to the Workflows category view.

## Chapter 24

# SAS Workflow Model Management Components

---

<b>Overview of SAS Workflow Model Management Components</b> . . . . .	<b>379</b>
<b>Importing Models</b> . . . . .	<b>380</b>
<b>Viewing Models</b> . . . . .	<b>381</b>
<b>Setting Champion and Challenger Models</b> . . . . .	<b>383</b>
<b>Publishing Models</b> . . . . .	<b>384</b>
<b>Add, View, or Delete Attachments</b> . . . . .	<b>386</b>
<b>Creating and Viewing Reports</b> . . . . .	<b>386</b>
<b>Viewing Performance Results</b> . . . . .	<b>388</b>
<b>Viewing All Model Management Components</b> . . . . .	<b>389</b>

---

## Overview of SAS Workflow Model Management Components

SAS Model Manager enables you to integrate SAS Workflow with some of the model management tasks that are normally performed in the SAS Model Manager client. Workflow definitions can be configured to use model management components with the workflow activities. When the workflow definition is activated for use, the model management components are available through the object view for the associated activity in the SAS Model Manager Workflow Console.

If a model management component is associated with an activity, you can perform one or more of the following tasks:

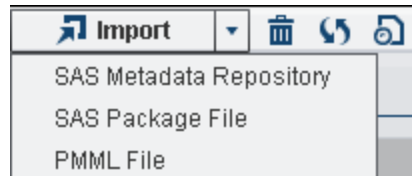
- import models
- view models
- set champion and challenger models
- publish models
- add, view, or delete attachments
- create and view reports
- view performance results

## Importing Models

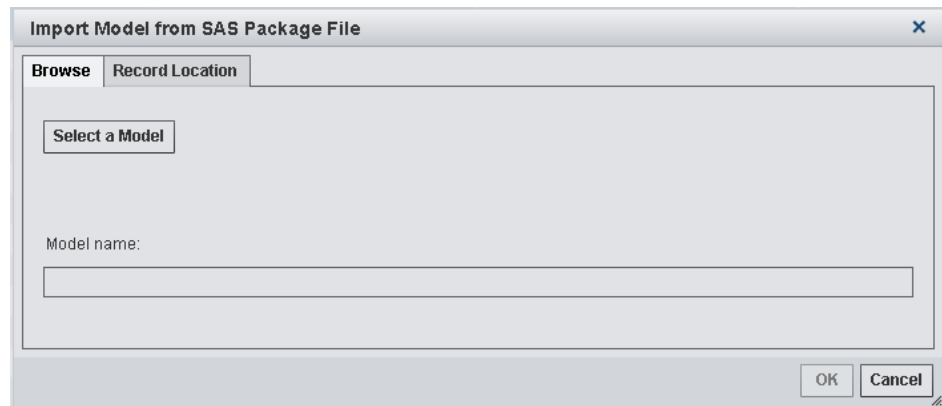
If the Import Models component is associated with an activity, you can import models. You can import models from the SAS Metadata Repository, a SAS package file (.spk), or a PMML file (.xml) into the version that is associated with the workflow.

To import a model:

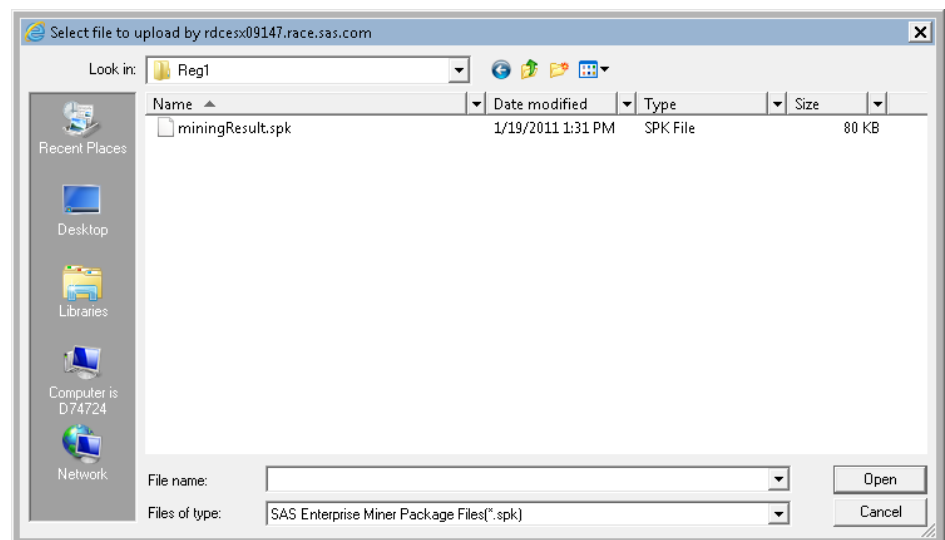
1. Select an import method from the **Import** drop-down list.



**Display 24.1** Example of Importing a Model from a SAS Package File

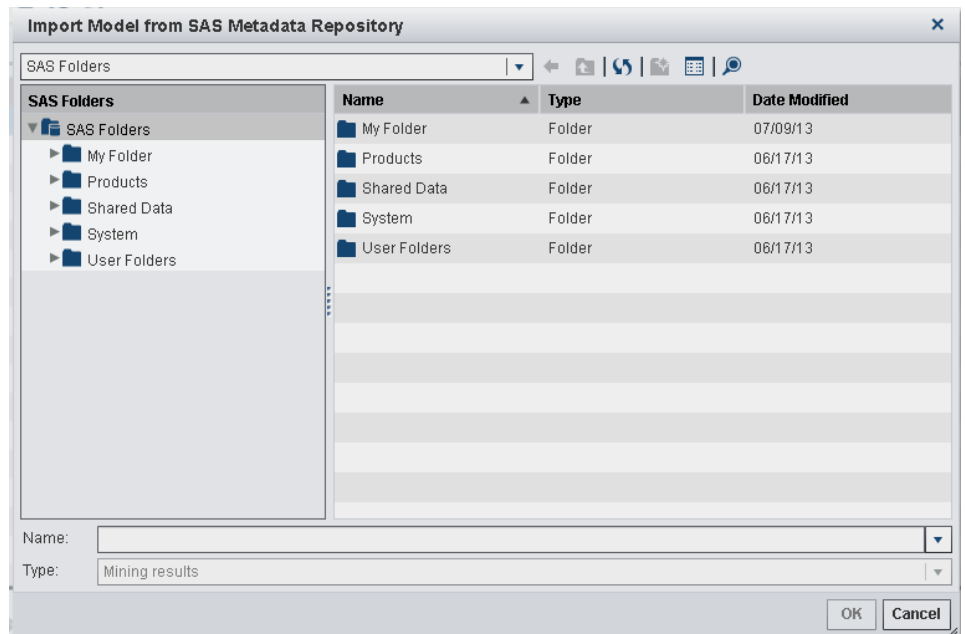


2. Navigate to the location of the file and select the model file to import.
  - When importing a model from a SAS package file or from a PMML File, select **Open**.



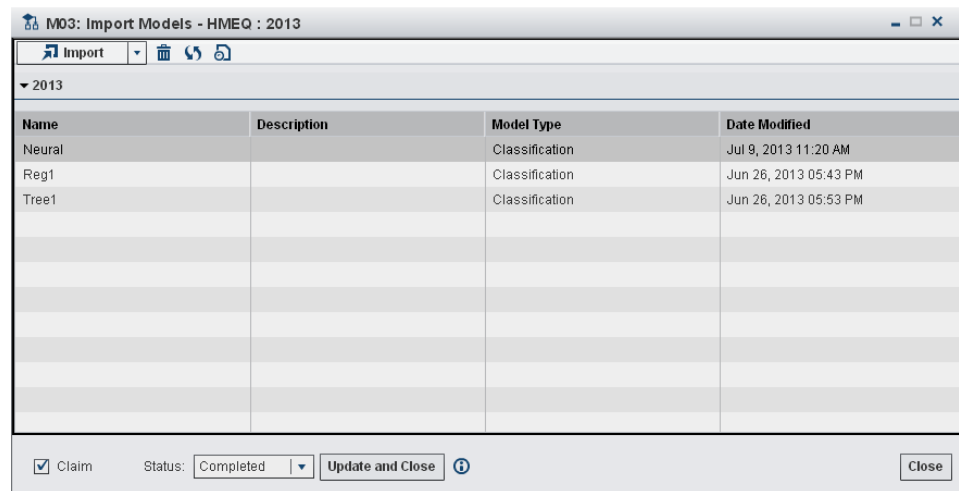


- When importing a model from SAS Metadata Repository, you select the file and specify a name in the same window.



3. Enter a name for the model and click **OK**.
4. Click **Close** for the success message.
5. Repeat steps 1 through 4 for each model.

**Display 24.2** Import Models Object Window After Models Are Imported



For more information about importing models using SAS Model Manager, see [Chapter 8, "Importing Models,"](#) on page 125.

## Viewing Models

If the View Models component is associated with an activity, you can view a list of the models. You can also view model information such as properties, model variables, score



## Setting Champion and Challenger Models

If the Set Champion and Challenger component is associated with an activity, you can set a project champion model and challenger models.



Perform one or more of the following actions:

- Select a model from the list, and click to set the model as the project champion model.

*Note:* The **Set Model Output Mapping** window appears if you have not mapped the model output variables to the project output variables. If there are model input variables that are not defined as project input variables, you are prompted to add the input variables. Click **Yes** to confirm. The model input variables are copied to the project input variables. If project output variables are not defined, the **Select Project Output Variables** window appears for you to select the output variables. After you select the output variables, click **OK**.

- Select a model from the list, and click to flag a model as a challenger to the project champion model.

*Note:* The **Set Model Output Mapping** window appears if you have not mapped the model output variables to the project output variables. If there are model input variables that are not defined as project input variables, you are prompted to add the input variables. Click **Yes** to confirm. The model input variables are copied to the project input variables.

- Select a model from the list, and click to clear a flagged challenger or champion model.
- Select a model from the list and click to view the model information.

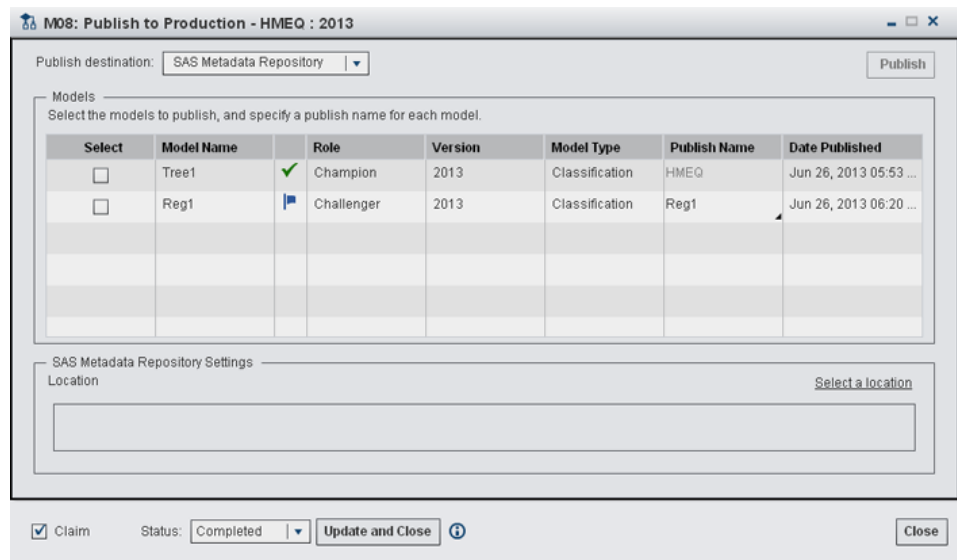
For more information about champion and challenger models, see [Chapter 12, “Deploying Models,”](#) on page 215.

## Publishing Models

You can publish champion and challenger models from the model repository to the SAS Metadata Repository or to a database.

To publish models to the SAS Metadata Repository:

1. Select **SAS Metadata Repository** for the publish destination.



2. Select the models that you want to publish from the **Models** list.
3. Specify a publish name for each model.
 

*Note:* The default format of the publish name is configured by the SAS administrator. You cannot modify the publish name for a champion model.
4. Select the location in the SAS Metadata Repository to publish the models.
5. Click **Publish**.

To publish models to a database:

1. Select the type of database for the publish destination.

Publish destination: Teradata Publish

Publish Options

Publish method:  SAS Embedded Process  Scoring function

Select the models to publish, and specify a publish name for each model.

Select	Model Name	Role	Version	Model Type	Publish Name	Date Published
<input type="checkbox"/>	Tree1	✓ Champion	2013	Classification	Tree1	Jun 26, 2013 05:53 PM
<input type="checkbox"/>	Reg1	■ Challenger	2013	Classification	Reg1_HMEQ	Jun 26, 2013 06:20 PM

Replace scoring files that have the same publish name

Specify an identifier to add to the database target table for each model:

HMEQ

Database Settings

Database server:

Database:

User ID:  Password:  [More options...](#)

Claim Status: Completed Update and Close ? Close

2. Select a publish method.
3. Select the models that you want to publish from the **Models** list.
4. Specify a publish name for each model.
 

*Note:* The default format of the publish name is configured by the SAS administrator.
5. (Optional) Select the **replace scoring files that have the same publish name** check box. This option is available only for the SAS Embedded Process publish method.
6. Specify an identifier to add to the database target table for each model. The default value is the project name. This option is available only for the SAS Embedded Process publish method.
7. Specify the settings to connect to the database, and click **More Options** to specify the processing options that should be used when publishing the models.

**Teradata Options** ✕

Validate scoring results

Keep scoring files if validation fails

Sample size:

Display detailed log messages

Use model input

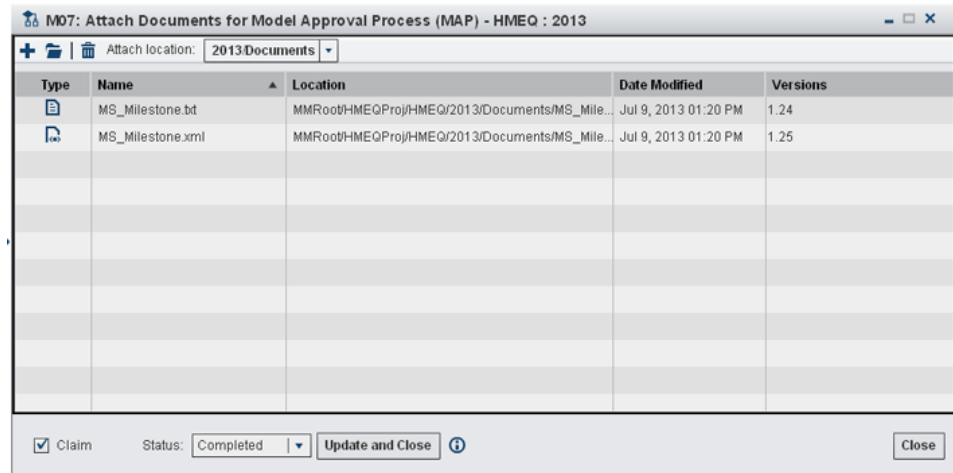
8. Click **Publish**.

For more information about publishing models using SAS Model Manager, see [Chapter 13, “Publishing Models,”](#) on page 223.

---

## Add, View, or Delete Attachments

If the Add Documents component is associated with an activity, you can add, view, or delete attachments. The attachments are stored in the **Documents** folder that is located within a project or version in the model repository.



Perform one of the following actions:

- Click to attach a file to the specified **attach location**.
- Click or double click the selected file to view the contents of the file.
- Click to delete an attachment.

---

## Creating and Viewing Reports

If the Create and View Reports component is associated with an activity, you can create reports and also view reports. The reports can be created by using the Workflow Console as well as by using the New Report wizard in the SAS Model Manager client application.

Here is an example of the **Validate Models** activity that uses this component.

Type	Name	Date Created	Location	Code	Log	Owner
ChampionChallenger	ChampionChallenger_D2013-05-30T16.06	Jun 17, 2013 09:44 AM	MMRootHMEQProjHMEQ/20...			mdimgadmin
DynamicLift	DynamicLift_D2013-06-25T16.48	Jun 25, 2013 04:48 PM	MMRootHMEQProjHMEQ/20...			sasdemo
Monitoring	Monitoring_D2013-05-30T15.38	Jun 17, 2013 09:44 AM	MMRootHMEQProjHMEQ/20...			mdimgadmin
Profile	Profile_D2013-06-20T22.33	Jun 20, 2013 10:33 PM	MMRootHMEQProjHMEQ/20...			sasdemo
Profile	Profile_D2013-06-25T16.41	Jun 25, 2013 04:41 PM	MMRootHMEQProjHMEQ/20...			sasdemo

To create a report:

1. Click and select a type of report. The New Report window appears.

Report: Delta Report

Output:  PDF  RTF  HTML  EXCEL

Style: SAS default

Select models from HMEQ:

All	Name	Version	Role	Model Type	Date Published
<input type="checkbox"/>	Reg1	2013	Challenger	Classification	Jun 26, 2013 05:43 PM
<input type="checkbox"/>	Neural	2013		Classification	
<input type="checkbox"/>	Reg1	2013b		Classification	Jun 26, 2013 05:43 PM
<input type="checkbox"/>	Tree1	2013a		Classification	Jun 26, 2013 05:53 PM
<input type="checkbox"/>	Tree1	2013	Champion	Classification	Jun 26, 2013 05:53 PM
<input type="checkbox"/>	Reg1	2013a		Classification	Jun 26, 2013 05:43 PM
<input type="checkbox"/>	Tree1	2013b		Classification	Jun 26, 2013 05:53 PM
<input type="checkbox"/>	Neural	2013a		Classification	

Name: \* Delta\_D2013-07-09T13.10

Description: The Delta Report

Run Report Cancel

2. Select an output type. The default is PDF.
3. Select a style for the report. When the SAS default option is selected, the default style and themes are used when generating the report. For example, the SAS default style for the HTML output type is HTMLBLUE.
4. From the list, select the models that you want to include in the report.

5. Specify a name and description if you do not want to use the default values.
6. Specify the report options for the selected report type. Some reports have additional options that can be set. Click **More options** to view the additional report options.

Here is an example of the **More Options** window for the Probability of Default Model Validation Report

The screenshot shows a dialog box titled "More Options" with a light gray background. It contains three rows of input fields, each with a red asterisk icon to its left. The first row is "Scorecard bin variable:" with the text "scorecard\_bin" in the input box. The second row is "Scorecard points variable:" with the text "scorecard\_points" in the input box. The third row is "Cut-off value:" with the text "100" in the input box. At the bottom right of the dialog are two buttons: "OK" and "Cancel".

7. Click **Run Report**. The new report is generated and appears in the default viewer for the selected output type.

To view a report:

1. Select a type of report from the left navigation menu.
2. Right-click a report from the list and select **Open** to view the report.

*Note:* You can also view the SAS code and SAS log if the report is not displayed.

To view the SAS code or SAS log for a report, select a report from the list and click on the icon in the **Code** or **Log** column.

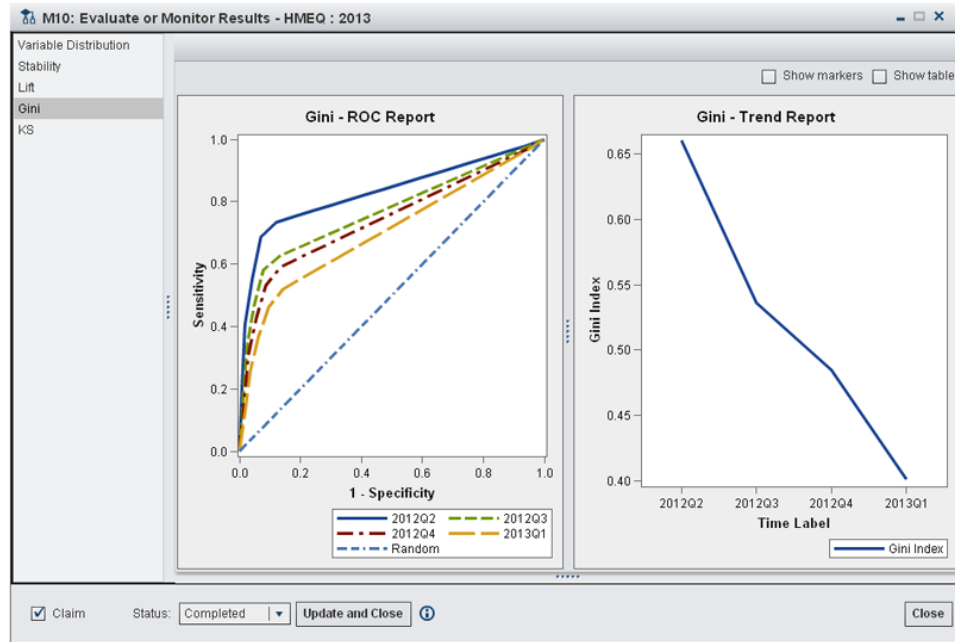
For more information, see [Chapter 10, “Validating Models Using Reports,”](#) on page 179.

---

## Viewing Performance Results

If the Model Performance Viewer component is associated with an activity, you can view the performance of the project champion model through a series of charts. The performance charts are generated using performance tasks in the SAS Model Manager client application. Select a type of report from the left navigation menu to view the performance charts.





For more information, see [Chapter 15, “What is Performance Monitoring?”](#) on page 251.

## Viewing All Model Management Components

The Utility component consists of all the model management components. You can associate the Utility component with an activity so that you can perform administrative tasks or review all of the content that is available for a workflow without having to step through multiple activities. To switch between the model management tasks select a link from the object window navigation bar.

The screenshot shows the 'Utility - HMEQ : 2013b' window. At the top, there is a navigation bar with the following links: [Import Models](#), [Models](#), [Set Champion and Challenger](#), [Publish Models](#), [Attachments](#), [View and Create Reports](#), and [Performance](#). Below the navigation bar, there is a table with the following data:

Name	Description	Model Type	Date Modified
Neural		Classification	Jul 9, 2013 02:33 PM
Reg1		Classification	Jul 5, 2013 04:46 AM
Tree1		Classification	Jul 5, 2013 04:47 AM

The status bar at the bottom shows 'Claim' checked, 'Status: Done', and 'Update and Close' buttons.



## Part 8

---

# Appendixes

<i>Appendix 1</i>	
<b>Query Utility</b> .....	393
<i>Appendix 2</i>	
<b>SAS Model Manager Access Macros</b> .....	401
<i>Appendix 3</i>	
<b>SAS Model Manager Macro Variables</b> .....	441
<i>Appendix 4</i>	
<b>Macros for Registering Models to the SAS Metadata Repository</b> .	449
<i>Appendix 5</i>	
<b>Macros for Adding Folders, Projects, Versions,     and Setting Properties</b> .....	459
<i>Appendix 6</i>	
<b>Macros for Generating Score Code</b> .....	473
<i>Appendix 7</i>	
<b>Properties</b> .....	503
<i>Appendix 8</i>	
<b>SAS Model Manager R Model Support</b> .....	519
<i>Appendix 9</i>	
<b>Statistical Measures Used in Basel II Reports</b> .....	527
<i>Appendix 10</i>	
<b>Report and Performance Monitoring Examples</b> .....	535



## Appendix 1

# Query Utility

---

Overview of the Query Utility .....	393
Search for Models .....	394
Search By Using a UUID .....	396
Search Life Cycles for Tasks Assigned to Users .....	398

---

## 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 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, or a version. 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 that is shown below the search criteria. The **Path** column in the search Component and Life Cycle 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/2013Q1/Models/Model%201
```

**MMRoot** is the top node in the Project Tree. From the Project Tree, expand the project folder **HomeEquity**, expand the version folder **2013Q1**, 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.

The Path column for the Model results contains the model repository path (the SMM Path system property) that you can use to determine the path to the model. Here is an example model repository path:

```
//ModelManagerDefaultRepo/MMRoot/HomeEquity/2013Q1/Model/Model%201
```

You can find the model that is called Model 1 in the project **HomeEquity**, the version **2013Q1**, and the **Models** folder.

To search the Project Tree, use the following instructions:

- “Search for Models” on page 394
- “Search By Using a UUID” on page 396
- “Search Life Cycles for Tasks Assigned to Users” on page 398

---

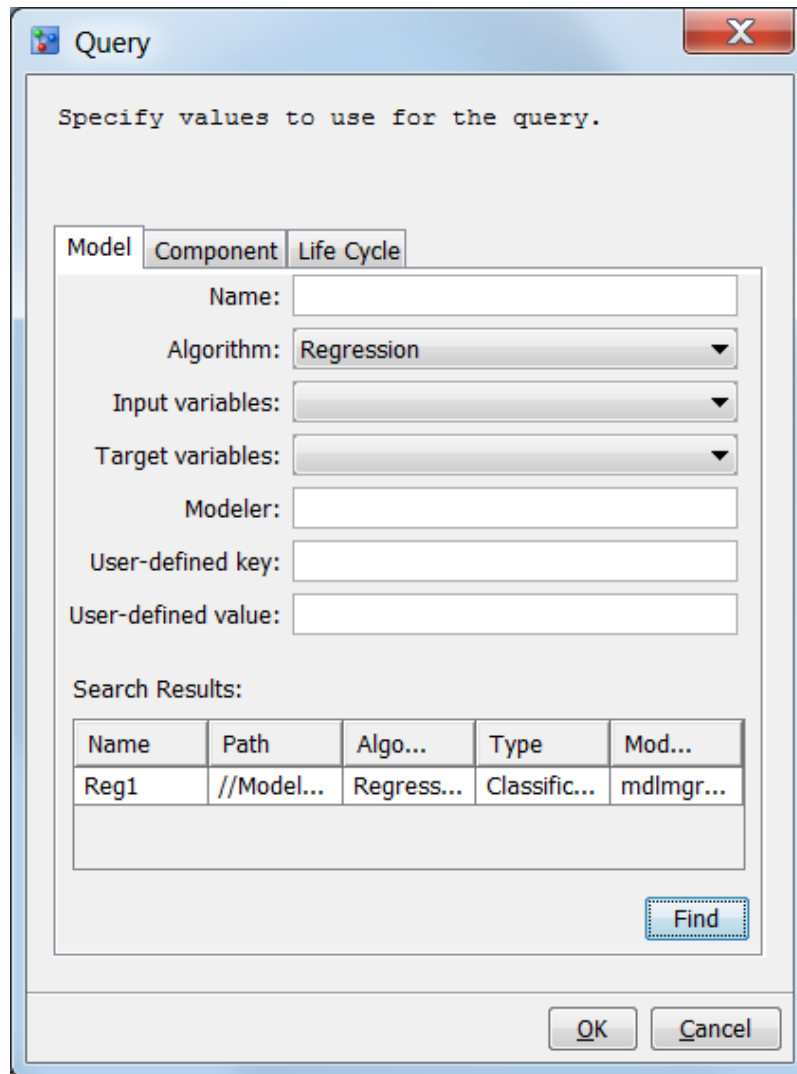
## Search for Models

To search for models:

1. From the Project Tree, right-click **MMRoot**, an organizational folder, a project, or a version, 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.
Modeler	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 <b>User-defined key</b> 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 <b>User-defined value</b> field.

**TIP** To deselect a value from a list box, select the blank line at the bottom of the list.



3. Click **Find**.

The search results display the following information:

Column	Description
Name	Specifies the name of the model.
Path	Specifies the <b>SMM Path</b> system property of the model in the Project Tree. Use the path to locate the model in the Project Tree, following the path from <b>MMRoot</b> . For example, using the path //ModelManagerDefaultRepo/MMRoot/HomeEquity/2013Q1/Model/Model%201, you can find the model Model 1 in the project <b>HomeEquity</b> , the version <b>2013Q1</b> , and the <b>Models</b> folder.
Algorithm	Specifies the name of the algorithm, such as regression or logistic, that is used by the model.

Column	Description
Type	Specifies one of the model function types: <ul style="list-style-type: none"><li>• Analytical</li><li>• Classification</li><li>• Prediction</li><li>• Cluster</li></ul>
Modeler	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 398](#)
- [“Search By Using a UUID” on page 396](#)

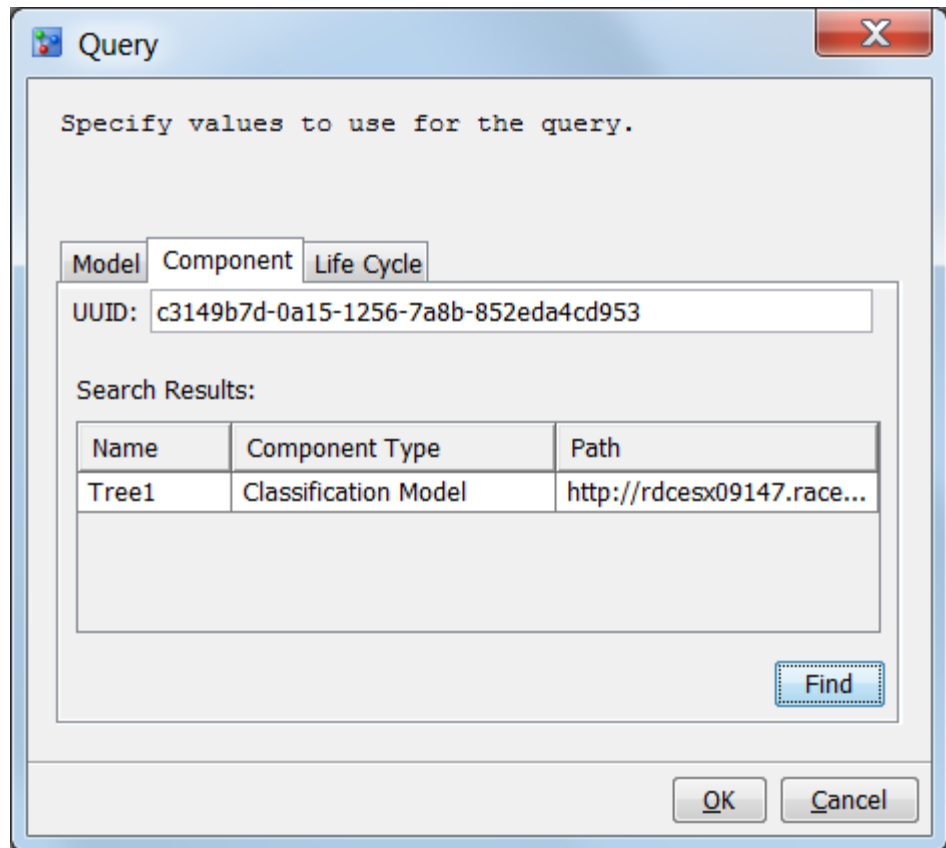
---

## 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.





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.
Component Type	Specifies one of following component types: Analytical Model the component is an analytical model Classification Model the component is a classification model Cluster Model the component is a clustering or segmentation model Model Group the component is an organizational folder Prediction Model the component is a prediction model Project the component is a project Version the component is a version

Column	Description
Path	Specifies the URL of the component in the Project Tree. Use the URL to locate the task in the Project Tree, following the path from <b>MMRoot</b> . For example, using the URL <code>http://SMMserver:8080/SASContentServer/repository/default/ModelManager/MMRoot/HomeEquity/2013Q1</code> you can find the version in the project <b>HomeEquity</b> .

### See Also

- “Search for Models” on page 394
- “Search Life Cycles for Tasks Assigned to Users” on page 398

## Search Life Cycles for Tasks Assigned to Users

To search for tasks that are assigned to a user:

1. From the Project Tree, right-click **MMRoot**, an organizational folder, a project, or a version, and select **Query**. The Query utility opens.
2. Click the **Life Cycle** tab.
3. Click the **User** field and select a user from the list box.

Specify values to use for the query.

Model Component **Life Cycle**

User: mdlmgrexampleapprovers

Assignee:

Name	Project	Version	Miles...	Status	Path

Approver:

Name	Proj...	Ver...	Mile...	Status	Path
End Prod...	HMEQ	2013	Retire	Not Sta...	http://r...
Select Ch...	HMEQ	2013	Develop...	Not Sta...	http://r...
Declare R...	HMEQ	2013	Producti...	Not Sta...	http://r...

Find

OK Cancel

#### 4. Click **Find**.

The search results display tasks in the **Assignee** results that are assigned to the user and tasks in the **Approver** results that the user is assigned to approve. The **Assignee** query results return only the tasks that have a status of **Started** or **Not Started**. Results that have a status of **Complete** or **Approved** are not returned. The **Approver** query results return tasks that have a status of **Started**, **Not Started**, and **Completed**.

The search results display the following information for tasks where the user is designated as an **Assignee** and an **Approver**:

Column	Description
Name	Specifies the name of the task.
Project	Specifies the project name for which the task must be completed.
Version	Specifies the version name for which the task must be completed.
Milestone	Specifies the milestone for which the task must be completed.
Status	Specifies the state of the task at the time of the query. Values for Status can be <b>Not Started</b> or <b>Started</b> .
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 <b>MMRoot</b> . For example, using the URL <code>http://SMMserver:8080/SASContentServer/repository/default/ModelManager/MMRoot/HomeEQ/2013Q1/Mortgages/Testing/Signoff</code> , you can find the user in the project <b>HomeEQ</b> , the version <b>2013Q1</b> , and the life cycle <b>Mortgages</b> .

#### See Also

- [“Search for Models” on page 394](#)
- [“Search By Using a UUID” on page 396](#)



## Appendix 2

# SAS Model Manager Access Macros

---

<b>Overview of Access Macros</b> . . . . .	<b>401</b>
<b>Using the SAS Model Manager Access Macros</b> . . . . .	<b>402</b>
Global Macro Variables . . . . .	402
Accessing the Macros . . . . .	404
Identifying SAS Model Manager Model Repository Objects . . . . .	404
Identifying Files Used by Access Macros . . . . .	405
Required Tables . . . . .	405
<b>Dictionary</b> . . . . .	<b>407</b>
%MM_AddModelFile Macro . . . . .	407
%MM_GetModelFile Macro . . . . .	410
%MM_GetURL Macro . . . . .	414
%MM_Register Macro . . . . .	415
%MM_RegisterByFolder Macro . . . . .	432
%MM_CreateModelDataset Macro . . . . .	437

---

## 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\9.4\mmcommon\sashelp`. The default value for `sasinstalldir` in Windows is `C:\Program Files\SAS`. The location of this file for UNIX is `/sasinstalldir/SASFoundation/9.4/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:

- Use the SAS Model Manager global macro variables to define the SAS Model Manager Service Registry URL and to define a valid SAS Model Manager user and password.
- Create a fileref to the SAS Model Manager access macro catalog and include that fileref, using the `%INCLUDE` statement.
- 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

### Global Macro Variables

Your SAS program and SAS Model Manager use global macro variables to pass information about the SAS environment 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 300

**\_MM\_RC**

contains one of the following return codes after processing a SAS Model Manager macro:

<b>_MM_RC</b> Return Value	<b>Access Macro</b> Status
0	All OK
1	Macro parameter error
2	Macro parameter processing error
3	Repository login failed
4	Repository operation failed
5	Generic critical Java error
6	Generic DATA step error

**\_MM\_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\_Service\_Registry\_URL**

contains the URL for a SAS environment file that defines the SAS environment.

**\_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**.

When you use the access macros, the macros need to know the following information:

- how to access the SAS environment XML file and environment name
- 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\_Service\_Registry\_URL**
- **\_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 300](#).

For a description of these macro variables as well as their default values, see [“Global Macro Variables” on page 402](#).

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_Service_Registry_URL=
    %STR(http://abcdef.sas.com:7980/SASWIPClientAccess/remote/ServiceRegistry);
%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

[Appendix 3, “SAS Model Manager Macro Variables,” on page 441](#)

## 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/2013*.

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 402.

## 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 set 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;

```

---

## Dictionary

---

### %MM\_AddModelFile Macro

Add model component files to an existing SAS Model Manager model.

---

#### 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.

**Examples** ModelId=8904daa1-0a29-0c76-011a-f7bb587be79f

---

```
ModelId=//ModelManagerDefaultRepo/MMRoot/DDHMEQ/
HomeEquity/2013/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** SASDataFile=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** SASCatalog=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_Service_Registry_URL=
    %STR(http://abcdef.sas.com:7980/SASWIPClientAccess/remote/ServiceRegistry);
%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/2013/hmeqDecTree1,
    Name=modelinput.sas7bdat,
    SASDataFile=mtbls.myInputVariables,
    Trace=Off
);

/*****/
/* A FILENAME for a text file.                  */
/*****/

```

```

FILENAME tcode 'c:\myModel\inputvar.xml';

/*****
/* Set to detect failure in case macro load fails */
/* and add the xml file for the input data source */
*****/

%let _MM_RC= -1;

%MM_AddModelFile (
  ModelId=
    //ModelManagerRepo/MMRoot/HomeEquity/HMEQ/2013/hmeqDecTree1,
  TextFile=tcode,
  Trace=on);

```

---

## %MM\_GetModelFile Macro

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

```

%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.

**Examples** ModelId=b2341a42-0a29-0c76-011a-f7bb7bc4f1e9

---

```

ModelId=//ModelManagerDefaultRepo/MMRoot/DDHMEQ/
HomeEquity/2013/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.

**Examples** VersionId=b23327cb-0a29-0c76-011a-f7bb3d790340

---

```
VersionId=//ModelManagerDefaultRepo/MMRoot/DDHMEQ/
HomeEquity/2013
```

---

**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.

**Examples** VersionId=b232d766-0a29-0c76-011a-f7bb50921b42

---

```
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** SASDataFile=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** SASCatalog=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, you can write SAS code that includes the score code in 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_Service_Registry_URL=
    %STR(http://abcdef.sas.com:7980/SASWIPClientAccess/remote/ServiceRegistry);
%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\2013;
    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/2013/
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\2013\DTree';
DATA score;
    set input.dTreeInp;
    %include dest;
run;
```

---

## %MM\_GetURL Macro

Translates a specified SAS Model Manager UUID to a URL-style path address and sets the URL as the value of the `_MM_URL` and `_MM_ResourcesURL` macro variables.

---

### 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_Service_Registry_URL=
    %STR(http://abcdef.sas.com:7980/SASWIPClientAccess/remote/ServiceRegistry);
%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

Registers a model to an existing version in the SAS Model Manager model hierarchy.

---

## Syntax

```
%MM_Register(
  VersionId=destination-version-UUID,
  ModelTemplate=model-template-name,
  EMModelPackage=SAS-fileref-for-EM-package-file,
  ScoreDataStepCode=fileref-to-data-step-fragment-score-code,
  ScoreProgram=fileref-to-SAS-program-score-code,
  InDataSamp=SAS-data-set-reference-to-input-data-sample-table,
  InDataInfo=SAS-data-set-reference-to-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.

**Default** the value of the \_MM\_CId macro variable

**Note** This argument is required.

#### **ModelTemplate=model-template-name**

specifies the SAS Model Manager model template that was used to register and validate this model.

**Defaults** 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.

---

**EMMModelPackage=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.

**Note** 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.

**Note** 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.

**Note** 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.

**Note** 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** [KeepInVars argument on page 419](#)

---

**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.

**Note** 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** The CONTENTS Procedure in the *Base SAS 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.

**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.

**Note** The OutDataSamp argument must be specified unless you use the OutDataInfo argument.

**See** [KeepOutVars argument on page 419](#)

**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.

**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.

**Note** The OutDataInfo argument must be specified unless you use the OutDataSamp 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.

**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** [KeepTargetVars argument on page 418](#)

**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.

**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.

**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.

**Note** This argument is 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.

**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.

**See** [InDataSamp argument on page 417](#)

**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.

**See** [OutDataSamp argument on page 418](#)

**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.

See [TargetDataSamp argument on page 418](#)

**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.

**Note** This argument is required.

**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.

**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.

**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.

**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.

**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. Valid values are 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.

**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.

**ToolVersion=*model-tool-version***

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.

**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.
<p>model score code</p> <p>Specify one of the following arguments:</p> <ul style="list-style-type: none"> <li>• ScoreDataStepCode</li> <li>• ScoreProgram</li> <li>• LogisticOutModelTable</li> </ul>	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"> <li>• InDataSamp and OutDataSamp</li> <li>• InDataInfo and OutDataInfo</li> <li>• InDataSamp and OutDataInfo</li> </ul>
	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"> <li>• InDataSamp and OutDataSamp</li> <li>• InDataInfo and OutDataInfo</li> </ul>

Required Information	Argument	Usage
	LogisticOutModelTable	<p data-bbox="1082 243 1374 499">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 data-bbox="1082 520 1374 806">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 data-bbox="1082 827 1374 1167">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"> <li>• You can use the KeepInVars argument to specify the variables in the InDataSamp data set that are used to create the inputvar.xml file.</li> <li>• 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.</li> </ul> <p>For more information, see <a href="#">KeepInVars argument on page 419</a>.</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 data-bbox="1082 243 1382 470">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 data-bbox="1082 491 1382 718">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 data-bbox="1082 739 1382 905">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 data-bbox="1082 926 1382 1255" style="list-style-type: none"> <li data-bbox="1082 926 1382 1092">• You can use the KeepOutVars argument to specify the variables in the OutDataSamp data set that are used to create the outputvar.xml file.</li> <li data-bbox="1082 1113 1382 1255">• If you do not specify the KeepOutVars argument, input variables and target variables are filtered from the output table.</li> </ul> <p data-bbox="1082 1276 1382 1354">For more information, see <a href="#">KeepOutVars argument on page 419</a>.</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 131 and “[%MM\\_AddModelFile Macro](#)” on page 407.

## Examples

### Example 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_Service_Registry_URL=
    %STR(http://abcdef.sas.com:7980/SASWIPClientAccess/remote/ServiceRegistry);
%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/2013,
  EMModelPackage=EMPak,
  ModelName=HMEQ,
  Description=Home Equity Score Code,
  Modeler=Titus Groan,
  Function=Reg,
  Tool=SAS Enterprise Miner,
  ToolVersion=v12.1,
  Subject= Loan,
  Trace=ON);

/*****
/* Display MM_Register defined variables.        */
*****/

Options nosource;
%PUT _MM_RC = &_MM_RC;
%PUT _MM_CId = &_MM_CId;
Options source;

```

### **Example 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_Service_Registry_URL=
    %STR(http://abcdef.sas.com:7980/SASWIPClientAccess/remote/ServiceRegistry);
%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/2013,
    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 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_Service_Registry_URL=
    %STR(http://abcdef.sas.com:7980/SASWIPClientAccess/remote/ServiceRegistry);
%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/2013,
    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

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.

**Default** the value of the `_MM_CId` macro variable

---

**Note** This argument is required.

---

**Examples** VersionId=b23327cb-0a29-0c76-011a-f7bb3d790340

---

VersionId=//ModelManagerDefaultRepo/MMRoot/DDHMEQ/  
HomeEquity/2013

---

#### ReportDir=*path-to-folder*

specifies the directory that contains the models to be registered.

**Note** This argument is required.

---

#### Trace=ON | OFF

specifies whether to supply verbose trace messages to the SAS log.

**Default** OFF

---

**Example** Trace=on

---

### Details

You can register SAS Enterprise Miner models and SAS code models using the %MM\_RegisterByFolder macro. The directory that you specify in the ReportDir argument is the parent folder. Each model has its own subfolder under the parent folder.

Each type of model has requirements for the subfolder name and the contents of the subfolder:

**Table A2.1** Requirements for Registering Models in a Directory

Requirement Type	Enterprise Miner Models	SAS Code Models
Value of ReportDir	a valid directory name	a valid directory name
Model subdirectory name	the subdirectory name must be the name of the model	the subdirectory name must be the name of the model
Contents of the subdirectory	one file named miningResult.spk	Required files: <ul style="list-style-type: none"> <li>• Modelmeta.xml</li> <li>• ModelInput.sas7bdat</li> <li>• Score.sas</li> </ul> Optional files: <ul style="list-style-type: none"> <li>• ModelOutput.sas7bdat</li> <li>• ModelTarget.sas7bdat</li> </ul>

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

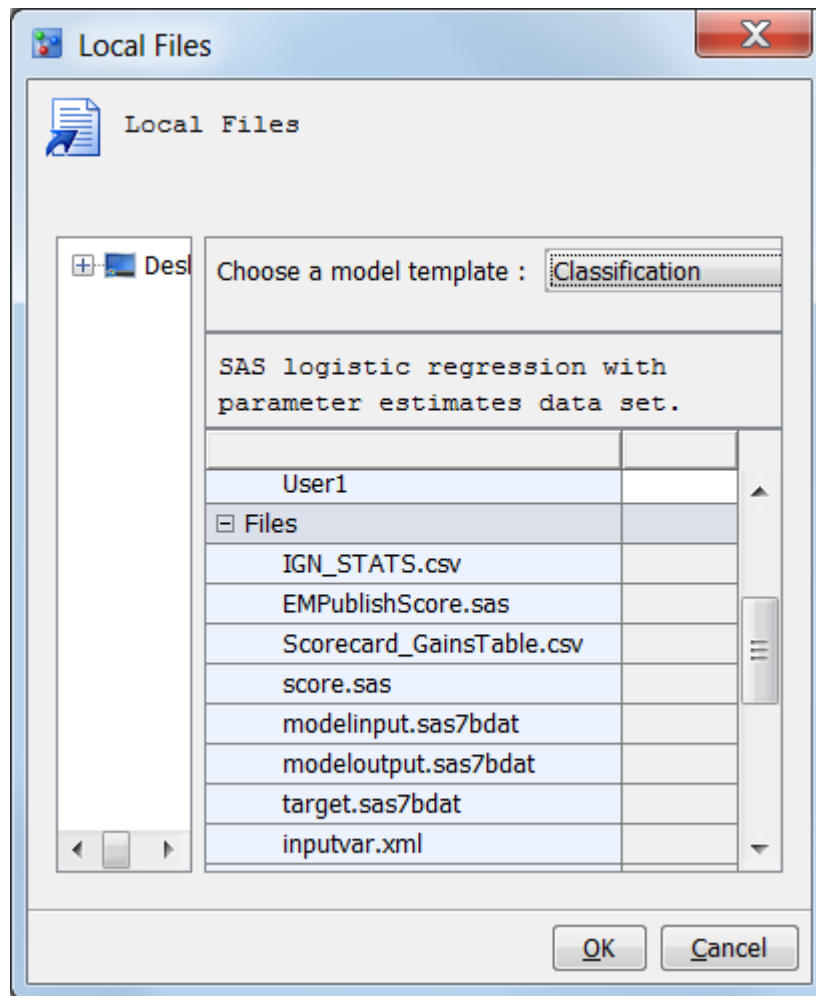
ModelOutput.sas7bdat is a SAS data set that contains one or more model output variables.

#### 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 514](#)
- <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 133.

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.sas7bcats</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 A2.1 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_Service_Registry_URL=
    %STR(http://abcdef.sas.com:7980/SASWIPClientAccess/remote/ServiceRegistry);
%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 hmeq2013 = //ModelManagerModelRepos/MMRoot/HomeEquity/HMEQ/2013;

```



```

/*****
/* Set to detect failure in case macro load fails      */
/* and register the models in SAS Model Manager.      */
/*****

%let _MM_RC= -1;

%MM_RegisterByFolder(VersionId=&hmeq2013, ReportDir=&modelFolder, Trace=ON);

/*****
/* Display the SAS Model Manager-defined variables.  */
/*****

Options nosource;
%PUT _MM_RC = &_MM_RC;
Options source;

```

---

## %MM\_CreateModelDataset Macro

Creates a data set that contains information about models. SAS Model Manager can provide information for the champion mode or for all models that are in the specified model repository path. The repository path that you specify can be MMRoot, an organizational folder, a project, a version, or a model. The data set contains the information for models that exist under the specified path.

---

### Syntax

```
%MM_CreateModelDataset (mDatasetName = name-of-data-set,
  smmPath=folder-project-verion-or-model-path <isChampion=Y | N><, 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** mDatasetName=work.models

---

#### **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 unless isChampion=Y. If isChampion=Y, the information that is returned is for only the champion model. If the path is a project, the data set contains model information for models under that project. If the path is a version, the data set contains model information for models under that version. If the path is a model, the data set contains model information for only that model.

**Default** MMRoot

---

#### **isChampion=Y | N**

specifies whether the information that is returned contains information for only the champion model or for all models.

**Y** specifies that the information that is returned is for only the champion model.

N specifies that the information that is returned is for all models.

**Default** Y

**Trace=ON | OFF**

specifies whether to supply verbose trace messages to the SAS log.

**Default** OFF

**Example** Trace=on

## Details

By default, the %MM\_CreateModelDataset returns data only about the champion model. If you want information about models other than the champion model, specify isChampion=N. The data set that is created contains these variables:

Algorithm	Name	ScoreCodeType
CreationDate	Owner	Template
Description	ProductionDate	TemplateFileName
ExpirationDate	ProjectName	Tool
FolderName	ProjectPath	UserProperties
Function	ProjectState	VersionName
ModelLabel	ProjectURL	VersionState
ModelUUID	ProjectUUID	isChampion
Modeler	PublishedDate	isDefaultVersion
ModificationDate	RetiredDate	isPublished

## Example

### Example Code A2.2 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_Service_Registry_URL=

```

```

%STR(http://abcdef.sas.com:7980/SASWIPClientAccess/remote/ServiceRegistry);
%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 hmeq2013 = //ModelManagerModelRepos/MMRoot/HomeEquity/HMEQ/2013;

/*****
/* 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/2013/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

### SAS Environment Macro Variables

The following table lists the macro variables that are used to set the SAS environment:

Macro Variable Name	Description	Example Value
<code>_MM_Service_Registry_URL</code>	the URL for a SAS environment that is defined in a SAS environment file.	<code>%let _MM_Service_Registry_URL= %STR(http://abcdef.sas.com: 7980/SASWIPClientAccess/ remote/ServiceRegistry);</code>
<code>_MM_Password</code>	the password of the user ID that is running the macro	<code>mdlmgrpw2</code>
<code>_MM_User</code>	the user ID of the user that is running the macro	<code>mdlmgradmin</code>

### Scoring Task Macro Variables

The following table lists the macro variables that are used to run a scoring task:

Macro Variable Name	Description	Example Value
<code>_MM_InputDS</code>	the location of the input data source file	<code>http://abc123.sas.com:8080/ SASContentServer/repository/ default/sasfolders/Shared Data/ Model Manager/MMLib/ HMEQ_SCORE_INPUT.sas7bdat</code>
<code>_MM_InputLib</code>	the libref that is associated with the location of the input data source file	<code>inlib</code>

Macro Variable Name	Description	Example Value
<code>_MM_ModelID</code>	the UUID of the model	4622bdda-ac1b-12d5-0196-021edec54347
<code>_MM_OutputDS</code>	the location of the output data source file	http://abc123.sas.com:8080/SASContentServer/repository/default/sasfolders/Shared Data/Model Manager/MMLib/HMEQ_SCORE_OUTPUT.sas7bdat
<code>_MM_OutputLib</code>	the libref that is associated with the location of the output data source file	outdslib
<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://abc123.sas.com:8080/SASContentServer/repository/default/sasfolders/Shared Data/Model Manager/MMLib/HMEQ_perf2013Q2.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/2013/Scoring
<code>_MM_TestDS</code>	the location of the test data source file	http://abc123.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://abc123.sas.com:8080/SASContentServer/repository/default/sasfolders/Shared Data/Model Manager/MMLib/HMEQ_train.sas7bdat

Macro Variable Name	Description	Example Value
_MM_TrainLib	the libref that is associated with the location of the train source file	trainline
_MM_User	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 Value
_MM_LocationInfo	the location information for a model	/MMRoot/Mortgages/HMEQ/2013
_MM_ModelFlag	the value of the champion model flag 0 - champion model 1 - challenger model	0
_MM_ModelLabel	a label for a model	reg
_MM_ModelName	the name of the model	Tree
_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_ProjectName	the name of the project	HMEQ
_MM_ReportFormat	the output format of the generated report	html
_MM_ReportLib	the libref for the <b>Report</b> node	report
_MM_ResourcesLib	the libref for the <b>Resources</b> node	resources
_MM_SampleSize	the size of a sample	1000
_MM_SampleSeed	the sample seed	12345
_MM_SourceCodeType	the type of score code	SAS Program

Macro Variable Name	Description	Example Value
<code>_MM_TargetEvent</code>	the target event value	1
<code>_MM_TargetVar</code>	the target variable name	bad
<code>_MM_TaskDir</code>	the URL of the stored report	http://myserver.mycompany:8080/SASContentServer/repository/default/ModelManager/MMRoot/DDHMEQ/HMEQ/2013/Reports
<code>_MM_User</code>	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 Value
<code>_MM_Agg_Mail</code>	specifies whether to send aggregated mail for performance monitoring with multiple data sources	Y or N
<code>_MM_DateTime</code>	the time that the performance task is to run	1Sep2013:05:00:00
<code>_MM_Hpds2_Flg</code>	enables high-performance monitoring if set it to 1, is used with the <code>_MM_Hpdm_Performance</code> macro variable	1
<code>_MM_Hpdm_Performance</code>	the configuration settings for high-performance monitoring	%nrstr(performance commit=10000 cpucount=ACTUAL dataserver='tera2650' timeout=120 host='tms2650' install='/opt/v940/ laxno/TKGrid';)
<code>_MM_ModelName</code>	the name of the champion model	reg1
<code>_MM_ModelID</code>	the UUID of the champion model	7514d6e- ac1b-12d5-01e4-878abeb04505



Macro Variable Name	Description	Example Value
_MM_ModelLocalPath	the location of the SAS Work library in the SAS Model Manager server	C: \DOCUME~1\ADMINI~1\LOCAL S~1\Temp\1\SAS Temporary Files \_TD2032_BRDVM0199_
_MM_Password	the password of the user ID that is running the report	mdlmgrpw2
_MM_ProjectPath	the network path to the SAS Model Manager project.	//ModelManagerDefaultRepo/ MMRoot/DDHMEQ/HMEQ
_MM_ProjectURLPath	the URL to the SAS Model Manager project	http://myserver.mycompany.com: 8080/SASContentServer/ repository/default/ModelManager/ MMRoot/HMEQ
_MM_ProjectUUID	the project UUID	27514d6e- ac1b-12d5-01e4-878abeb04505
_MM_Seg_Filter	filters the performance data for each sub-project from the top level performance datasource by using this macro variable.	%nrstr(Location='USA')
_MM_ScoreCodeType	the type of score code	SAS Program
_MM_VersionName	the name of the default version	2013
_MM_ReportDatascrc	the project's performance data set	jun13perf.sas7bdat
_MM_PreCode	one or more macro variables that set values to performance variables	%let _MM_EventProbVar=score; %let _MM_TargetVar=bad;
_MM_ResultURLPath	the URL to the version's <b>Resources</b> node	http://myserver.mycompany.com: 8080/SASContentServer/ repository/default/ModelManager/ MMRoot/HMEQ/2013/Resources
_MM_TimeLabel	the label that is used in reports to represent the time period of the data in the performance data set	2013Q2
_MM_Trace	indicates whether to write a trace log	ON or OFF

Macro Variable Name	Description	Example Value
_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 Value
_MM_Dashboard_Dir	the path to the directory where the dashboard report is stored	C:\SAS\Config\Lev1\AppData\SASModelManager12.3\Dashboard
_MM_Force_Run_Dash_Reports	whether to force running the report and updating all tables	Y or N
_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 Value
_MM_Hpds2_Flg	enables high-performance monitoring if set it to 1, is used with the _MM_Hpdm_Performance macro variable	1

Macro Variable	Description	Example Value
<code>_MM_Hpdm_Performance</code>	the configuration settings for high-performance monitoring	<code>%nrstr(performance commit=10000 cpucount=ACTUAL dataserver='tera2650' timeout=120 host='tms2650' install='/opt/v940/laxno/ TKGrid');</code>
<code>_MM_Password</code>	the password of the user ID that is running the report	<code>mdlmgrpw2</code>
<code>_MM_Service_Registry_URL</code>	the URL for a SAS environment that is defined in a SAS environment file.	<code>%let _MM_Service_Registry_URL= %STR(http://abcdef.sas.com: 7980/SASWIPClientAccess/ remote/ServiceRegistry);</code>
<code>_MM_User</code>	the user ID of the user who is running the report	<code>mdlmgradmin</code>



## Appendix 4

# Macros for Registering Models to the SAS Metadata Repository

---

<b>Using Macros to Register Models Not Created by SAS Enterprise Miner . . . . .</b>	<b>449</b>
About the %AA_Model_Register Macro . . . . .	449
Register a Model in the SAS Metadata Repository Using a SAS/STAT Item Store . . . . .	451
Create a SAS Package File Using a SAS/STAT Item Store . . . . .	451
Register a Model in the SAS Metadata Repository Using Model Component Files . . . . .	452
<b>Dictionary . . . . .</b>	<b>453</b>
%AAModel Autocall Macro . . . . .	453
%AA_Model_Register Autocall Macro . . . . .	454

---

## Using Macros to Register Models Not Created by SAS Enterprise Miner

### *About the %AA\_Model\_Register Macro*

You can use the %AAModel macro and the %AA\_Model\_Register macro to register the SAS Metadata Repository models that are not created by SAS Enterprise Miner. These models are created by SAS procedures and are supported by SAS Model Manager:

- SAS/STAT item store models
- High-performance models
- PROC COUNTREG models
- PROC SEVERITY models

If you do not want to register the model, you can create SAS package files (SPK) without registering the model. After the model is registered to the SAS Metadata Repository, you can import the model to SAS Model Manager using the import from SAS Metadata Repository method. If you create an SPK file, you would import the model using the import from SAS Model Package File method.

The %AAModel macro is an autocall macro that loads the %AA\_Model\_Register macro. This macro must be submitted before you submit the %AA\_Model\_Register macro.

You specify these types of arguments in the %AA\_Model\_Register macro:

- The model identification argument's name. You must also describe a model and identify a SAS/STAT item store.
- Action arguments specify whether to create an SPK file and whether to register the model in the SAS Metadata Repository.
- You specify model component arguments when a SAS/STAT procedure does not create an item store, if a model is created using high performance analytic procedures, or if you are registering PROC COUNTREG or PROC SEVERITY models. The model component arguments identify the train data set, the model level, and the score code file. The arguments also identify whether the score code is only DATA step code or a SAS program that includes DATA step code, macros, procedures.
- The Lookup=Select option if a SAS/STAT model's input variable includes non-latin1 characters. This option ensures the generation of correct score code.
- Other options are available to add information to the model or to specify whether to keep or delete the data sets that the macro produces.

For more information, see [“%AA\\_Model\\_Register Autocall Macro” on page 454](#).

When you are registering the model to the SAS Metadata Repository, you can specify the metadata server connection system options before you run the %AAModel and %AAModel\_Register macros. If these options are not specified, dialog boxes appear to prompt you for the information. Here is a sample OPTIONS statement that specifies these options:

```
options metaPort=8561
        metaServer=server-address
        metaRepository=Foundation
        metaUser=user-ID
        metaPass=password;
```

These SAS/STAT procedures can create an item store using the STORE statement:

Procedure	Item Store Restrictions
GENMOD	Training code is not included
GLIMMIX	Training code is not included
GLM	Training code is not included
GLMSELECT	Fit statistics are not included
LOGISTIC	None
MIXED	Training code is not included
REG	Training code or fit statistics are not included

If you want to retrain models using SAS Model Manager and if the procedure item store does not include training code, you must create the SAS training code before you run the %AA\_Model\_Register macro.

*Note:* Item store restrictions have not been evaluated for other SAS/STAT procedures that have a STORE statement. Using the %AA\_Model\_Register macro might cause undesirable results.

### Register a Model in the SAS Metadata Repository Using a SAS/STAT Item Store

After you run a SAS/STAT procedure using the STORE statement, you use the %AA\_Model\_Register macro to register the model to the SAS Metadata Repository.

In the following example program, the PROC LOGISTICS STORE statement creates an item store in work.logisticStore. The %AA\_Model\_Register macro uses the item store in work.logisticStore to create the register file.

```

/* PROC LOGISTIC specifies the STORE statement to create an item store. */
/
proc logistic data=sampsio.hmeq;
  class job;
  model bad = loan value job;
  store work.logisticStore;
run;

/* Set up the meta data connection system options. */

options metaPort=8561
        metaServer=server-address
        metaRepository=Foundation
        metaUser=user-ID
        metaPass=password;

/* Load the macros. */

%aamodel;

/* Register the model in the SAS Metadata Repository. */

%aa_model_register(modelname=LogisticTest,
                   modeldesc=%nrquote(Logistic Test),
                   itemstore=work.logisticstore,
                   register=Y,
                   mrPath=%NRBQUOTE(/User Folders/user-ID/My Folder/),
                   spk=N,
                   spkfolder=c:\temp\,
                   data=sampsio.hmeq)
;

```

The model can now be imported to SAS Model Manager using the import from SAS Metadata Repository method.

### Create a SAS Package File Using a SAS/STAT Item Store

To create a SAS package (SPK) file without registering it to the SAS Metadata Repository, you specify the Register=Y, SPK=Y, and the SPKFolder= arguments. This example shows these modifications using the previous example:

```

/* PROC LOGISTIC specifies the STORE statement to create an item store. */

```

```

proc logistic data=sampsio.hmeq;
  class job;
  model bad = loan value job;
  store work.logisticStore;
run;

/* Set up the meta data connection system options. */

options metaPort=8561
  metaServer=server-address
  metaRepository=Foundation
  metaUser=user-ID
  metaPass=password;

/* Load the macros. */

%aamodel;

/* Create an SPK file; do not register the model in the SAS Metadata Repository. */

%aa_model_register(modelname=LogisticTest,
  modeldesc=%nrquote(Logistic Test),
  itemstore=work.logisticstore,
  register=N,
  spk=Y,
  spkfolder=c:\temp\,
  data=sampsio.hmeq)
;

```

The macro creates a folder for the model in the `c:\temp` folder. The folder name is the UUID of the model. The name of the SPK file is `miningResults.spk`. The SPK file can be imported to SAS Model Manager using the import from SAS Model Package File method.

### Register a Model in the SAS Metadata Repository Using Model Component Files

If you do not have an item store, or if you have the information and files that you need for a model, you can use the `%AA_Model_Register` macro to register the model in the SAS Metadata Repository. In addition to the macro's model identification arguments and the action arguments, you can use these arguments to register the model:

- `Data=training-data-set-name`
- `Level=Binary | Ordinal | Nominal | Interval`
- `ScoreCodeFile=filename`
- `ScoreCodeFormat=Datastep | Program`
- `Target=target-variable`

The following SAS program uses model component arguments to register the model to the SAS Metadata Repository. Other arguments identify the mining function and mining algorithm.

```

/* Train high performance model */

```



```

proc hplogistic data=gplib.hmeqid; class job reason;
  id value;
  class bad ;
  model bad = clage clno debtinc delinq derog mortdue job reason;
  output out=gplib.hpregid_score pred;
  code file='c:\temp\score.sas' ;
run;

/* Set up metadata connections */

options metaPort=8561
        metaServer=server-address
        metaRepository=Foundation
        metaUser=user-ID
        metaPass=password;

/* Load the macros. */

%aamodel;

/* Register the model in the SAS Metadata Repository */

%aa_model_register
(modelname=Modell,
 modeldesc=%nrquote(First Model for registration),
 register=Y,
 mrPath=%NRQUOTE(/User Folders/user-ID/My Folder/),
 spk=N,
 spkfolder=c:\temp\,
 data=sampsio.hmeq,
 target=bad,
 level=BINARY,
 miningfunction=Classification,
 miningalgorithm=Regression,
 scorecodefile=c:\temp\score.sas)
;

```

The model can now be imported to SAS Model Manager using the import from SAS Metadata Repository method.

---

## Dictionary

---

### %AAModel Autocall Macro

Loads the %AA\_Model\_Register macro.

---

#### Syntax

**%AAModel**

## Details

The %AAModel macro loads the %AA\_Model\_Register macro. You must specify %aamodel; before you use the %AA\_Model\_Register macro. The %AAModel macro produces these messages in the SAS log:

```
NOTE: Loading the aa_model_eval macro
NOTE: Loading the aa_model_register macro
```

*Note:* The %AA\_Model\_Eval macro is used internally by SAS Model Manager.

---

## %AA\_Model\_Register Autocall Macro

Creates an SPK package file and registers models to the SAS Metadata Repository.

---

### Syntax

```
%AA_Model_Register(
  ModelName model-name,
  ModelDesc=description,
  Register=Y | N,
  MRPath=SAS-Metadata-Repository-folder,
  SPK=Y | N,
  SPKFolder=SPK-folder-path,
  ItemStore=item-store-name,
  Data=training-data-set-name,
  Target=target-variable,
  Level=Binary | Ordinal | Nominal | Interval,
  ScoreCodeFile=filename,
  ScoreCodeFormat=Datastep | Program,
  <Score=scored-data-set-name>,
  <PMMLFile=filename>,
  <TrainFile=train-program-filename>,
  <MiningAlgorithm=algorithm>,
  <MiningFunction=mining-function>,
  <Segment=segment-variable-name>,
  <Lookup=lookup-method>,
  <Debug=Y | N>)
```

### Model Identification Arguments

**ModelName=***model-name*

specifies the name of the model.

**Default** aa\_model\_&sysuserid, where &sysuserid contains the user ID or login of the current SAS process.

---

**ModelDesc=***description*

is a description of the model.

**ItemStore=***item-store-name*

specifies the name of the item store that is created by some SAS/STAT procedures. The item store is used to retrieve input and target variable metadata, data set names, score code, training code, the mining algorithm, and the mining function.

**Note** Item store data is not available from these SAS/STAT procedures: REG, GLM, GENMOD, GLIMMIX, PHREG, and SURVEYPHREG.

**Tip** If you do not specify the ITEMSTORE= option, you must specify these options: DATA=, TARGET=, SCORECODEFILE=, SCORECODEFORMAT=. If you specify the ITEMSTORE= option, you do not need to specify these options.

**Action Arguments****Register=Y | N**

specifies whether to register the model in the SAS Metadata Repository.

Y indicates to register the model in the SAS Metadata Repository.

N indicates not to register the model in the SAS Metadata Repository.

**Default** Y

**MRPath=***SAS-Metadata-Repository-Folder*

specifies a folder, using **SAS Folders** as the root node in the SAS Metadata Repository, where the model is registered.

**Default** /Shared Data/

**Note** The forward slash (/) after the last folder in the path is not required.

**Example** /Shared Data/Model Manager/Models/

**SPK=Y | N**

specifies whether to create a SAS package file:

Y indicates to create a SAS package file.

N indicates not to create a SAS package file.

**Requirement** If SPK=Y, you must use the SPKFOLDER= option to specify a location to store the SPK file.

**SPKFolder=***SPK-folder-path*

specifies the location to store the SPK file.

**Requirement** The option is required when you specify SPK=Y.

**Model Component Arguments**

These arguments must be specified if you do not specify the ITEMSTORE= option:

**Data=***training-data-set-name*

specifies the name of the training data set for the model.

**Level=**Binary | Ordinal | Nominal | Interval

specifies the class target level of the model.

Binary	the variable can contain two discrete values (for example, Yes and No).
Ordinal	the variable can contain discrete values that have a logical order (for example, 1, 2, 3, 4).
Nominal	the variable contains discrete values that do not have a logical order (for example, car, truck, bus, and train).
Interval	the variable contains values across a range. For example, temperature ranges could be between 0–100.

**ScoreCodeFile=filename**

specifies the name of the file that contains the score code.

**Tip** If you specify the ITEMSTORE= option, you do not need to specify this option.

**ScoreCodeFormat=Datastep | Program**

specifies the format of the score code.

DATASTEP	the score code contains only DATA step statements
PROGRAM	the score code contains DATA step statements, procedures, or macros.

**Target=target-variable**

specifies the name of the target variable for model.

**Optional Arguments****Debug=Y | N**

specifies whether to prevent deletion of the generated data sets:

Y indicates to keep the generated data sets.

N indicates not to keep the generated data sets.

**Lookup=lookup-method**

specifies the algorithm for looking up CLASS levels in SAS/STAT models. Here are the valid lookup methods:

**Auto**

selects the LINEAR algorithm if a CLASS variable has fewer than five categories. Otherwise, the Binary algorithm is used. This is the default.

**Binary**

specifies to use a binary search. This method is fast, but it might produce incorrect results. The normalized category values might contain characters that collate in different orders in ASCII and EBCDIC, if you generate the code on an ASCII machine and execute the code on an EBCDIC machine, or vice versa.

**Linear**

uses a linear search with IF statements that have categories in the order of the class levels. This method is slow if there are many categories.

**Select**

uses a SELECT statement.

**Requirement** Use Lookup=Select when a SAS/STAT model contains non-latin1 characters to ensure the generation of the correct score code. If a model with non-latin1 characters is published to a

database and Lookup=Select is not specified, the scoring results might be incorrect.

---

**MiningAlgorithm=*algorithm***

specifies the type of algorithm that is used to create the mode (for example, DecisionTree or logistic).

**MiningFunction=*mining-function***

specifies one of the following mining functions:

- classification
- prediction
- segmentation

**PMMLFile=*filename***

specifies the name of the file that contains the PMML score code. This option is optional.

**Score=*scored-data-set-name***

specifies the name of the scored training data set. This data set is used when there is no score code available to determine the output variables.

**Segment=*variable***

specifies the name of the segment variable.

**TrainFile=*train-program-filename***

specifies the name of the training program file.



## Appendix 5

# Macros for Adding Folders, Projects, Versions, and Setting Properties

---

<b>Adding Folders, Projects, Versions, and Properties Using Macros . . . . .</b>	<b>459</b>
Overview of Using a SAS Program to Add Folders, Projects, Versions, and Properties . . . . .	459
Writing Your SAS Program . . . . .	460
Creating the Properties Table . . . . .	461
<b>Dictionary . . . . .</b>	<b>464</b>
%mdlmgr_AddFolder Macro . . . . .	464
%mdlmgr_AddProject Macro . . . . .	465
%mdlmgr_AddVersion Macro . . . . .	467
%mdlmgr_SetProperty Macro Function . . . . .	468
<b>Example: Add a Folder, Project, and Version, Set Properties . . . . .</b>	<b>470</b>

---

## Adding Folders, Projects, Versions, and Properties Using Macros

### *Overview of Using a SAS Program to Add Folders, Projects, Versions, and Properties*

SAS Model Manager provides four macros that you can use in a SAS program to add folders, project, and versions, and to set properties:

`%mdlmgr_AddFolder()`

Adds a folder under **MMRoot** or adds a subfolder.

`%mdlmgr_AddProject()`

Adds a project under a folder or a subfolder.

`%mdlmgr_AddVersion()`

Adds a version to a project.

`%mdlmgr_SetProperty()`

Sets project and version properties that appear in the **Specific Properties** section of the project or version **Properties** tab in the SAS Model Manager client.

After you have added the Project Tree nodes or set properties, you refresh the **MMRoot** node to see the new nodes and property settings in the SAS Model Manager client. You can then use these nodes in the client to further define your projects and versions.

To delete a folder, project, or version, you use the SAS Model Manager client.

## Writing Your SAS Program

Include these language elements in your SAS program:

Global macro variable to set the environment

```
%let _MM_Service_Registry_URL=
    %str(http://your-server.com:7980/SASWIPClientAccess/remote/ServiceRegistry);
```

Global macro variable to define the user and a DATA step to provide the password

```
%let _MM_User = user_ID;
data _null_;
    infile pwfile obs=1 length=1;
    input @;
    input @1 line $varying1024. 1;
    call symput('_MM_Password', substr(line,1,1));
run;
```

If you are setting properties, use a DATA step to create a table that contains property and value pairs.

One of the %mdlmgr\_SetProperty( ) arguments is the name of a table that contains property-value pairs. “[Creating the Properties Table](#)” on page 461 lists the properties that you can include in the table. When you create the table, the first column must be Name and the second column must be Value. Both columns must be character. See “[Example: Creating a Properties Table](#)” on page 464.

Access the macros by using the FILENAME and %INCLUDE statements.

```
filename file1 catalog 'sashelp.mdlmgr.accessmacros.source';
%include file1;
filename file1;

filename file2 catalog 'sashelp.mdlmgr.mdlmgr_addfolder.source';
%include file2;
filename file2;

filename file3 catalog 'sashelp.mdlmgr.mdlmgr_addproject.source';
%include file3;
filename file3;

filename file4 catalog 'sashelp.mdlmgr.logtrace.source';
%include file4;
filename file4;

filename file5 catalog 'sashelp.mdlmgr.mdlmgr_addversion.source';
%include file5;
filename file5;

filename file6 catalog 'sashelp.mdlmgr.mdlmgr_setproperty.source';
%include file6;
filename file6;
```

You can change the fileref name.

Call the macros:

```
%mdlmgr_AddFolder(ParentId=, Name=, Desc=, NewFolderId=, Trace=);

%mdlmgr_AddProject(ParentId=, Name=, Desc=, ModelFunction=,
    InputVarTable=, OutputVarTable=, NewProjectId=, Trace=);
```



```
%mdlmgr_AddVersion(ParentId=, Name=, Desc=, NewVersionId=, Trace=);
```

```
%mdlmgr_SetProperty(FolderId=, Table=, PropertyType=, FolderType=, Trace=);
```

There is no requirement to call all of the macros in the same SAS program.

When SAS returns from a macro call that adds a node, the value of `NewFolderId=`, `NewProjectId=`, and `NewVersionId=` is used to create a global macro variable that can be referenced by other macros in the same SAS session. The value of the macro variable is the UUID or the SAS Model Manager repository path for the node that is added. You can then use that macro reference as a value for the `ParentId=` argument of another macro or for the `%mdlmgr_SetProperty()` macro `FolderId=` argument. For example, in the `%mdlmgr_AddProject()` macro, if you set

```
NewProject=projectId, the variable name projectId is used to create the global macro variable %projectId. The &projectId macro reference can now be used as the value of the ParentId= argument in the %mdlmgr_AddVersion() macro, ParentId=&projectId. The same macro reference can be used as a value for the FolderId= argument in the %mdlmgr_SetProperty() macro, FolderId=&projectId.
```

## Creating the Properties Table

### Property Table Requirements

To set project properties, you use a DATA step to create a data set that contains property-value pairs. The data set variables must be `Name` and `Value`, and they must be character variables.

In the data set, property names can be mixed case. The required appended text, `:sas-libraries`, must be lower case. For more information, see [“Specifying Data Sets” on page 461](#).

### Specifying Data Sets

Some property values specify the name of a default table, such as the default train table or the default performance table. You specify tables using the form `SMRLibrary.table` for libraries in the SAS Metadata Repository and `libref.table` for SAS libraries. See the Data Sources category view for valid library and table names. In the **SAS Metadata Repository** tab, `SMRLibrary` is the folder-name where the data set is stored. In the **SAS Libraries** tab, `libref` can be one of the librefs under the **SAS Libraries** node.

When your DATA step specifies a library in the **SAS Libraries** tab, the text `:sas-library` must be appended to `libref.table` in lower case (for example, **MySASLib.Property:sas-library** and **Work.ProjProp:sas-library**). Libraries that are defined in the SAS Metadata Repository do not require the appended text.

### Properties That You Can Set

Use a property in the following Property Name column as a value for the `Name` variable in the property table.

**Table A5.1** Project and Version Properties That Can Be Set by %mdlmgr\_SetProperty() Macro

Property Name	Property Name As It Appears in the SAS Model Manager Client	Valid Values
ClassificationRole	Output Event Probability Variable	A text string that specifies the output event probability variable.  Set for a project with a model function of classification.
ClassTargetEvent	Class Event Value	A number that represents the target event value.  Set for a project.
ClassTargetEventValues	Class Target Values	A text string that represents the class target values.  Set for a project.
ClassTargetLevel	Class Target Level	One of the following text strings: "BINARY", "NOMINAL", "ORDINAL", or "INTERVAL"  Set for a project.
ClassTargetVar	Training Target Variable	A text string that indicates the training target variable  Set for a project.
EventProbabilityRole	Output Event Probability Variable	A text string that specifies the output event probability variable.  Specify this property only if you specify the outputVarTable= argument in the %mdlmgr_AddProject() macro. The value of EventProbabilityRole must be a variable in the project output table.  Set for a project.
Function	Model Function	A text string that specifies the model function. Valid values are "CLASSIFICATION", "PREDICTION", "SEGMENTATION", and "ANALYTICAL".  Set for project.
InterestedParty	Interested Party	A text string that specifies a person or group that has an interest in the project.  Set for a project.

Property Name	Property Name As It Appears in the SAS Model Manager Client	Valid Values
MetadataLock	Lock Project Metadata	Specify "YES" or "NO" to indicate whether the project metadata is locked. Set for a project.
PredictionRole	Output Prediction Variable	A text string that specifies the output prediction variable. Set for a project with a model function of prediction.
ProjectInputDS	None, it is used to create inputvar.xml.	The project input table in the form <i>libref.table</i> . Set for a project.
ProjectOutputDS	None, it is used to create outputvar.xml.	The project output table in the form <i>libref.table</i> . Set for a project.
ResponseDS	Default Performance Table	The default performance table in the form <i>libref.table</i> . Set for projects and versions.
ScoreInputDS	Default Scoring Task Input Table	The default scoring task input table in the form <i>libref.table</i> . Set for projects and versions.
ScoreOutputDS	Default Scoring Task Output Table	The default scoring task output table in the form <i>libref.table</i> . Set for projects and versions.
SegmentRole	Output Segmentation Variable	A text string that specifies the output segmentation variable. Set for a project with a model function type of segmentation.
State	State	Select one: 0 Under Development 1 Active 2 Inactive 3 Retired Set for a project.
TestDS	Default Test Table	The default test table in the form <i>libref.table</i> . Set for projects and versions.

Property Name	Property Name As It Appears in the SAS Model Manager Client	Valid Values
TrainDS	Default Train Table	The default train table in the form <i>libref.table</i> . Set for projects and versions.

### Example: Creating a Properties Table

Here is a sample DATA step to create a properties table:

```
data HMEQProp;
  length name $20.;
  length value $40.;
  input name $ value$;
  datalines;
TestDS MMLIB.HMEQ_TEST
ScoreInputDS MMLIB.HMEQ_SCORE_INPUT
ScoreOutputDS MMLIB.OUTPUT
TrainDS MMLIB.HMEQ_TRAIN
ResponseDS PERFDS.2013Q1:sas-library
ClassTargetEvent 1
ClassTargetLevel BINARY
EventProbabilityRole SCORE
ClassTargetVar BAD
;
run;
```

Note the difference in values for the ResponseDS property and the other table properties. In the Data Sources category view, the library MMLIB is defined in the **SAS Metadata Repository** tab and the library PERFDS is defined in the **SAS Libraries** tab. Because PERFDS is defined in the **SAS Libraries** tab, the value requires **:sas-library** to be appended to the *libref.table* value. Libraries that are defined in the SAS Metadata Repository do not require the appended text.

---

## Dictionary

---

### %mdlmgr\_AddFolder Macro

Adds a folder to the Project Tree.

---

## Syntax

```
%mdlmgr_AddFolder(
  ParentId=parent-UUID-or-path
  Name=folder-name
  <Desc=description>
  NewFolderId=folder-Id-variable
  <Trace=On | Off>
);
```

### Required Arguments

#### **ParentId=parent-UUID-or-path**

specifies the UUID or the SAS Model Manager repository path of the parent folder.

If the folder that you are creating is a subfolder, you can use the value of **NewFolderId=** that was specified during the macro call of parent folder as the value for *parent-UUID-or-path*. For example, if a parent folder exists and **NewFolderId=&folderId** was set in the macro call for the parent folder, then you can specify **ParentId=&folderId** in the subfolder macro call.

If you specify the repository path, use one of these forms:

```
//ModelManagerDefaultRepo/MMRoot/
//ModelManagerDefaultRepo/MMRoot/folder-name/
```

**Restriction** A folder can be added only to the **MMRoot** node or a folder in the Project Tree.

#### **Name=folder-name**

specified the name of the folder. The name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ).

#### **NewFolderId=folder-Id-variable**

specifies a variable that is used to identify the new folder.

SAS Model Manager creates a global macro variable, *%folder-Id-variable* whose value is the folder UUID or the path in the SAS Metadata Repository. You can use *&folder-Id-variable* as the value of a **ParentId=** argument in the **%mdlmgr\_AddFolder()** or **%mdlmgr\_AddProject()** macros. For example, if **NewFolderId=folderId**, then you can use **ParentId=&folderId** in the **%mdlmgr\_AddProject()** macro.

### Optional Arguments

#### **Desc=description**

specifies a description of the folder.

#### **Trace=On | Off**

specifies whether to supply verbose trace messages to the SAS log.

**Default** Off

---

## %mdlmgr\_AddProject Macro

Adds a project to a folder.

## Syntax

```
%mdlmgr_AddProject(
  ParentId=parent-UUID
  Name=folder-name
  <Desc=description>
  ModelFunction=model-function
  <InputVarTable=project-input-variable-table>
  <OutputVarTable=project-output-variable-table>
  NewProjectId=project-Id-variable
  <Trace=On | Off>
);
```

### Required Arguments

#### **ParentId=***parent-UUID-or-path*

specifies the UUID of the parent folder or the SAS Model Manager path for the parent folder.

You can use *&folder-Id-variable* that is set for the NewFolderId= argument in the %mdlmgr\_AddFolder( ) macro as the value of *parent-UUID-or-path*.

The SAS Model Manager path is in this form:

```
//ModelManagerDefaultRepo/MMRoot/folder-name/
```

#### **Name=***project-name*

specified the name of the project. The name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ).

#### **ModelFunction=***model-function*

specifies the project model function type. These are the valid values:

- classification
- prediction
- segmentation
- analytical

**Default** classification

#### **NewProjectId=***project-Id-variable*

specifies a variable or a macro variable that is used to identify the new project.

SAS Model Manager creates a global macro variable, *%project-Id-variable* whose value is the project UUID or the path in the SAS Metadata Repository. You can use *&project-Id-variable* as the value of a ParentID= argument in the %mdlmgr\_AddVersion( ) macro or the FolderId= argument in the %mdlmgr\_SetProperty( ) macro. For example, if you set **NewProjectId=projectId**, you can use **ParentId=&projectId** in the %mdlmgr\_AddVersion( ) macro.

The SAS Model Manager path is in this form:

```
//ModelManagerDefaultRepo/MMRoot/folder-name/project-name
```

## Optional Arguments

### **Desc=description**

specifies a description of the project.

### **InputVarTable=project-input-variable-table**

specifies a data set that must include the input variables that are used by the champion model. If you have several candidate models for your project, make sure that all candidate model input variables are included in the project input table. The data set does not need to contain data. If you use the train table as a project input table, be sure to exclude the target variable.

The input variable table is used to create the inputvar.xml file, which describes all of the model input variables.

**Requirement** The data set must be a local or network file. This macro does not support project input tables in the SAS Metadata Repository.

**Tip** The project input table can be defined after the project is created. It must be defined before the project champion model is set.

**See** [“Create a Project Input Table” on page 37](#)

### **OutputVarTable=project-output-variable-table**

specifies a data set that includes only output variables that are created or modified by the champion model. 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. If you use the train table as the project output 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 output variable table is used to create the outputvar.xml file, which describes all of the model output variables.

**Requirement** The data set must be a local or network file. This macro does not support project output tables in the SAS Metadata Repository.

**Tip** The project output table can be defined after the project is created. It must be defined before the project champion model is set.

**See** [“Create a Project Output Table” on page 38](#)

### **Trace=On | Off**

specifies whether to supply verbose trace messages to the SAS log.

**Default** Off

---

## %mdlmgr\_AddVersion Macro

Adds a version to a project.

---

## Syntax

```
%mdlmgr_AddVersion(
  ParentId=parent-UUID-or-path
  Name=version-name
  <Desc=description>
  NewVersionId=version-Id-variable
  <Trace=On | Off>
);
```

### Required Arguments

#### ParentId=parent-UUID-or-path

specifies the UUID of the project for which the version is to be created.

You can use *&project-Id-variable* that is set for the NewProjectId= argument in the %mdlmgr\_AddProject() macro as the value of *parent-UUID-or-path*. For example, if **NewProjectId=projectId**, you can specify **ParentId=&projectId**.

The SAS Model Manager path is in the form

```
//ModelManagerDefaultRepo/MMRoot/folder-name/project-name
```

#### Name=version-name

specifies the name of the version. The name can contain letters, spaces, the underscore ( \_ ), the hyphen ( - ), and the period ( . ).

#### NewVersionId=version-Id-variable

specifies a variable name that is used to identify the new version.

SAS Model Manager creates a global macro variable, *%version-Id-variable* whose value is the version UUID or the path in the SAS Metadata Repository. You can use *&version-Id-variable* as the value of the FolderId= argument in the %mdlmgr\_SetProperty() macro. For example, if you set **NewVersionId=versionId**, then you can specify **FolderId=&versionId** in the %mdlmgr\_SetProperty() macro.

The version path is in this form:

```
//ModelManagerDefaultRepo/MMRoot/folder-name/project-name/version-name
```

### Optional Arguments

#### Desc=description

specifies a description of the version.

#### Trace=Of | Off

specifies whether to supply verbose trace messages to the SAS log.

Default Off

---

## %mdlmgr\_SetProperty Macro Function

Sets project properties in the Project Tree.

---



## Syntax

```
%mdlmgr_SetProperty(
  FolderId=folder-UUID-or-path
  Table=property-value-table-name
  PropertyType=System | User
  FolderType=UUID-or-folder-type
  <Trace=On | Off>
```

### Required Arguments

#### **FolderId=***folder-UUID-or-path*

specifies the project folder UUID or path.

To add a project property, you can use *&project-Id-variable* that is set for the `NewProjectId=` argument in the `%mdlmgr_AddProject()` macro as the value of *project-folder-UUID-or-path*. For example, if `NewProjectId=projectId`, then you can specify `FolderId=&projectId`.

To add a version property, you can use *&version-Id-variable* that is set for the `NewVersionId=` argument in the `%mdlmgr_AddVersion()` macro as the value of *project-folder-UUID-or-path*. For example, if `NewVersionId=versionId`, then you can specify `FolderId=&versionId`.

#### **Table=***property-value-data-set*

specifies the data set that contain the properties to set. *property-value-table-name* must be in the form *libref.data-set*.

See [“Creating the Properties Table” on page 461](#)

#### **PropertyType=**System | User

specifies whether the property is a SAS Model Manager property or if the property is user-defined. Specify **system** for all SAS Model Manager properties.

Default System

#### **FolderType=***folder-type*

specifies the folder type for the properties that are being set. If FolderId is a UUID, this argument is optional. Here are the valid values for Folder type:

- Project
- Version

### Optional Argument

#### **Trace=**On | Off

specifies whether to supply verbose trace messages to the SAS log.

Default Off

---

## Example: Add a Folder, Project, and Version, Set Properties

```
%let _MM_User=your-userID;
%let _MM_Password=your-password;
%let _MM_Service_Registry_URL=%STR(http://your-web-service.com:7980/
SASWIPClientAccess/remote/ServiceRegistry);
libname temp 'your-path';
data temp.property;
    length name $ 30 value $ 40;
    input name $ value $;
    infile datalines;
datalines;
ProjectInputDS MMLIB.HMEQ_PROJECT_INPUT
ProjectOutputDS MMLIB.HMEQ_PROJECT_OUTPUT
ScoreInputDS MMLIB.HMEQ_SCORE_INPUT
ScoreOutputDS MMLIB.HMEQ_SCORE_OUTPUT
TrainDS MMLIB.HMEQ_TRAIN
TestDS MMLIB.HMEQ_TEST
ClassTargetEvent 1
ClassTargetLevel BINARY
ClassTargetVar BAD
EventProbabilityRole SCORE
;
run;

/* Access the macros */

filename file1 catalog 'sashelp.modelmgr.accessmacros.source';
%include file1;
filename file1;

filename file2 catalog 'sashelp.modelmgr.mdlmgr_addfolder.source';
%include file2;
filename file2;

filename file3 catalog 'sashelp.modelmgr.mdlmgr_addproject.source';
%include file3;
filename file3;

filename file4 catalog 'sashelp.modelmgr.logtrace.source';
%include file4;
filename file4;

filename file5 catalog 'sashelp.modelmgr.mdlmgr_addversion.source';
%include file5;
filename file5;

filename file6 catalog 'sashelp.modelmgr.mdlmgr_setproperty.source';
%include file6;
filename file6
```

```
/*add folder*/
%mdlmgr_AddFolder( parentId=//ModelManagerDefaultRepo/MMRoot,
                  name=Bank3,
                  desc=,
                  newFolderId=newFolderIdVar,
                  Trace=on);

/*add project*/
%mdlmgr_AddProject( parentId=&newFolderIdVar,
                   name=HMEQ,
                   desc=Home Equity,
                   modelFunction=classification,
                   inputVarTable=,
                   outputVarTable=,
                   newProjectId=newProjectIdVar1,
                   Trace=on);

/*set properties*/
%mdlmgr_SetProperty( folderId=&newProjectIdVar1,
                    table=temp.property,
                    propertyType=system,
                    folderType=project,
                    Trace=on);

/*add version*/
%mdlmgr_AddVersion( parentId=&newProjectIdVar1,
                   name=2013,
                   desc=,
                   newVersionId=newVersionIdVar1,
                   Trace=off);
```



## Appendix 6

# Macros for Generating Score Code

---

<b>Generating Score Code for COUNTREG Procedure Models</b> .....	<b>473</b>
<b>Generating Score Code for PROC SEVERITY Models</b> .....	<b>474</b>
<b>Dictionary</b> .....	<b>474</b>
%MM_Countreg_Create_Scorecode Autocall Macro .....	474
%MM_Severity_Create_Scorecode Autocall Macro .....	488

---

## Generating Score Code for COUNTREG Procedure Models

The %MM\_Countreg\_Create\_Scorecode macro creates DATA step statements to compute the predicted values of a model that you create using the COUNTREG procedure. Input to the macro is the ODS output data set ParameterEstimates that is created by the COUNTREG procedure. You can also specify the location to save the score code and other macro output files, and prefix values for the dependent variable and the variable for the probability of having a zero-generating process.

*Note:* SAS Model Manager does not support PROC COUNTREG models when VALIDVARNAME="ANY".

The score code generation supports the following COUNTREG procedure features:

PROC COUNTREG Feature	Supported Functionality
Categorical predictor	Character and numeric class variables
Continuous predictor	Variable values are used as is.
MODEL specification	Effect specifications that are allowed by the MODEL statement, including main effects, interactions, and powers of continuous predictors. Only one MODEL statement can be specified.
ZEROMODEL specification	Effect specifications that are allowed in the MODEL statement, including the intercept, main effects, interactions, and powers of continuous predictors.

PROC COUNTREG Feature	Supported Functionality
OFFSET variables	The offset variables in the MODEL and ZEROMODEL statements are retrieved from the FitSummary table.
ZEROMODEL statement LINK function	The LOGISTIC and the NORMAL link distribution functions that are allowed in the ZEROMODEL statement.

BY-group processing is not supported.

After you have created the score code, you can register the score code and other COUNTREG procedure model component files by using the \$AA\_Model\_Register macro or you can import the model using the local files method. For more information, see [“Using Macros to Register Models Not Created by SAS Enterprise Miner”](#) on page 449 and [“Import SAS Code Models and R Models Using Local Files”](#) on page 130.

---

## Generating Score Code for PROC SEVERITY Models

The %MM\_Severity\_Create\_Scorecode macro generates score code for PROC SEVERITY models. Inputs to the macro are the ODS output data sets ParameterEstimates and ModelInformation that are created by the SEVERITY procedure. You can also specify the location to save the score code and other macro output files, and the prefix value for the dependent variable.

Custom distributions and BY-group processing are not supported by the macro.

After you have created the score code, you can register the score code and other SEVERITY procedure model component files by using the \$AA\_Model\_Register macro or you can import the model using the local files method. For more information, see [“Using Macros to Register Models Not Created by SAS Enterprise Miner”](#) on page 449 and [“Import SAS Code Models and R Models Using Local Files”](#) on page 130.

---

## Dictionary

---

### %MM\_Countreg\_Create\_Scorecode Autocall Macro

Generates score code for a model that is created by the COUNTREG procedure.

#### Syntax

```
%MM_Countreg_Create_Scorecode (
    ParmEst=countreg-parameter-estimate-data-set
    <FileRef=output-fileref>
    <PredPrefix=dependent-variable-prefix>
    <PZPrefix=probability-zero-variable-prefix>
);
```

## Arguments

**ParmEst=***countreg-parameter-estimate-dataset*

specifies the name of the parameter estimations ODS output data. This ParameterEstimates data set is created when PROC COUNTREG executes. To capture this data set, use the ODS OUTPUT statement before PROC COUNTREG executes.

**Tip** In the PROC COUNTREG code, include the PREDICTION= and the PREOBZERO= options in the OUTPUT statement.

**FileRef=***output-fileref*

specifies the fileref that defines the location of the macro output files.

**Default** The SAS log

**PredPrefix=***dependent-variable-prefix*

specifies a prefix for the predicted dependent variable. The variable is named in the PRED= option of the PROC COUNTREG OUTPUT= statement. When is prefix is applied to the dependent variable, this new name becomes the prediction variable.

**Default** P\_

**PZPrefix=***probability-zero-variable-prefix*

specifies a prefix for the variable that indicates the probability that the response variable will take on the value of zero as a result of the zero-generating process. The variable is named in the PROBZERO= option of the PROC COUNTREG OUTPUT= statement. When the prefix is applied to the probability zero variable, this new name becomes the probability zero variable.

**Default** PHI\_

## Details

To create score code for a model that you create with PROC COUNTREG, include the following SAS code:

1. Use a LIBNAME statement to identify the location of the output that you create using PROC COUNTREG.
2. Before PROC COUNTREG, use the ODS OUTPUT statement to capture the ParameterEstimates output data set. Here is an example:

```
ods output ParameterEstimates=CntReg.ParameterEstimates;
```

3. Build your model using PROC COUNTREG and close the ODS OUTPUT destination.
4. Use the FILENAME statement to define a fileref for the macro output location.
5. Invoke the %mm\_countreg\_create\_scorecode macro.
6. Execute the score code within a DATA step.

---

## Example: Generate the PROC COUNTREG Score Code for Insurance Risk

### Create the Sample Insurance Data

The following SAS program creates sample data that resembles an automobile policy history file for a property and casualty insurance program:

```

%let MyProj = C:\Users\myID;
%let MyProj = C:\Users\minlam\Documents\Projects;
libname CntReg "&MyProj.\CountReg\Test";
options fmtsearch = (CntReg.formats);
proc format library = CntReg cntlout = phf_fmt;
  value $ Gender_fmt
    'Male' = 'Man'
    'Female' = 'Woman';
  value HO_fmt
    0 = 'No'
    1 = 'Yes';
run;

data CntReg.phf;
  length CarType $ 5;
  label CarType = 'Type of Car';
  length Gender $ 6;
  format Gender $ Gender_fmt.;
  label Gender = 'Gender Identification';

  /* This variable name will test how the macro will resolve name conflicts */
  length Estimate $ 6;
  label Estimate = 'Gender Identification (Copy)';
  label AgeDriver = 'Driver Age';
  format HomeOwner HO_fmt.;
  call streaminit(27513);

  do PolicyId = 00001 to 99999;
    StartYr = 2000 +
      rand('table', 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1);
    do ExpYr = StartYr to 2011;
      EExp = rand('uniform');
      MyOffset = 0;
      select (rand('table', 0.499, 0.299, 0.199, 0.003));
        when (1)
          do;
            CarType = 'SEDAN';
            fCarType = 0;
          end;
        when (2)
          do;
            CarType = 'TRUCK';
            fCarType = 0.5;
          end;
      end;
    end;
  end;

```



```

        when (3)
        do;
            CarType = 'SPORT';
            fCarType = 1.0;
        end;
        otherwise CarType = ' ';
    end;

AgeDriver = 18 + rand('binomial',0.375, 72);
fAgeDriver = 0.0123 * (AgeDriver - 17);

HomeOwner = rand('bernoulli', 0.25);
if (HomeOwner eq 0) then fHomeOwner = 0.7;
    else if (HomeOwner eq 1) then fHomeOwner = 0;
if (HomeOwner eq 1) then
    do;
        IS = round(rand('uniform') * 5) - 2.5;
        fIS = -0.0456 * IS * IS;
    end;
if (EExp lt 0.5) then
    do;
        Gender = 'Male';
        fGender = 0;
    end;
    else if (EExp lt 0.9) then
        do;
            Gender = 'Female';
            fGender = -1.5;
        end;
        else Gender = ' ';
Estimate = Gender;
if (missing(HomeOwner) eq 0 and missing(IS) eq 0)
    then mu_zero = 0.987 + fHomeOwner + fIS;
    else mu_zero = 0.987;
phi = cdf('normal', mu_zero, 0, 1);
if (rand('bernoulli', phi) eq 0) then
    do;
        if (missing(CarType) eq 0 and missing(AgeDriver) eq 0 and
            missing(Gender) eq 0)
            then mu = 2 + fCarType + fAgeDriver + fGender;
            else mu = 2;
        nClaim = rand('poisson', exp(mu));
    end;
    else nClaim = 0;
output;
end;
end;
drop fCarType fAgeDriver fHomeOwner fGender;
drop mu_zero mu;
run;

```

### Run the Sample Program

Here is the sample program:

```
%let MyProj = C:\Users\emdev;
```

```

libname CntReg "&MyProj.\CountReg\Test";
options fmtsearch = (CntReg.formats);

/* Original Model */
%let model = 1;

/* Build the model and deliver the required ODS datasets */
ods output ParameterEstimates = CntReg.ParameterEstimates_&model.;

proc countreg data = CntReg.phf;
  class CarType Gender HomeOwner;
  model nClaim = CarType AgeDriver Gender / dist = poisson;
  zeromodel nClaim ~ HomeOwner IS * IS / link = normal;
  output out = CntReg.phf_pred_&model.
  predicted = Pred_nClaim probzero = Phi_nClaim;
run;

ods output close;

/* Define the fileref for the output syntax */
filename ThisFile "&MyProj.\CountReg\Test\ScoreCode_&Model..sas";

/* Invoke the macro */
%mm_countreg_create_scorecode(
  ParamEst = CntReg.ParameterEstimates_&Model.,
  FileRef = ThisFile,
  PredPrefix = MyPred_,
  PZPrefix = MyPhi_,
);

/* Execute the score codes within a DATA STEP */
data CntReg.phf_pred_compare;
  set CntReg.phf_pred_&Model.;
  %include ThisFile;
  IsMiss_Pred_nClaim = missing(Pred_nClaim);
  IsMiss_Phi_nClaim = missing(Phi_nClaim);
  IsMiss_MyPred_nClaim = missing(MyPred_nClaim);
  IsMiss_MyPhi_nClaim = missing(MyPhi_nClaim);
  if (IsMiss_Pred_nClaim eq 0 and IsMiss_MyPred_nClaim eq 0)
    then MyDiffPred = MyPred_nClaim - Pred_nClaim;
  if (IsMiss_Phi_nClaim eq 0 and IsMiss_MyPhi_nClaim eq 0)
    then MyDiffPhi = MyPhi_nClaim - Phi_nClaim;
run;

proc contents data = CntReg.phf_pred_compare;
run;

/* If the score codes work correctly, then the MyDifference variable should be
   a constant variable of all zero values */
proc freq data = CntReg.phf_pred_compare;
  tables _WARN_;
run;

proc tabulate data = CntReg.phf_pred_compare;
  class IsMiss_Pred_nClaim IsMiss_MyPred_nClaim

```

```
IsMiss_Phi_nClaim IsMiss_MyPhi_nClaim;
var Pred_nClaim MyPred_nClaim MyDiffPred Phi_nClaim
    MyPhi_nClaim MyDiffPhi;
table IsMiss_Pred_nClaim * IsMiss_MyPred_nClaim *
    (n nmiss mean*f=e22. stddev*f=e22. min*f=e22. max*f=e22.),
    (Pred_nClaim MyPred_nClaim MyDiffPred);
table IsMiss_Phi_nClaim * IsMiss_MyPhi_nClaim *
    (n nmiss mean*f=e22. stddev*f=e22. min*f=e22. max*f=e22.),
    (Phi_nClaim MyPhi_nClaim MyDiffPhi);
run;
quit;
```

## The Generated Score Code and Output Tables

### Output A6.1 The Score Code That Is Generated by SAS Model Manager

```

/*****
/* Begin scoring code for COUNTREG
/* Model: ZIP
/* Created By: emdev
/* Date: April 26, 2013
/* Time: 09:27:39
*****/

LENGTH _WARN_ $ 4;
_WARN_ = ' ';
LABEL _WARN_ = &quot;Warnings&quot; ;

_nInputMiss = 0;

/*****
/* Check the continuous predictors
*****/

IF ( MISSING( AgeDriver ) EQ 1 ) THEN _nInputMiss = _nInputMiss + 1;

IF ( MISSING( IS ) EQ 1 ) THEN _nInputMiss = _nInputMiss + 1;

_nInputOutOfRange = 0;

/*****
/* Check the CLASS predictors
*****/

LENGTH _UFormat_1 $ 5 ;
LABEL _UFormat_1 = &quot;Formatted Value of CarType&quot; ;
IF ( MISSING( CarType ) EQ 0 ) THEN DO;
  _UFormat_1 = STRIP( PUT( CarType , $5. ) );
  IF ( _UFormat_1
      NOTIN ( &quot;SEDAN&quot;
            , &quot;SPORT&quot;
            , &quot;TRUCK&quot;
            )
      ) THEN _nInputOutOfRange = _nInputOutOfRange + 1;
END;
ELSE _nInputMiss = _nInputMiss + 1;

LENGTH _UFormat_2 $ 5 ;
LABEL _UFormat_2 = &quot;Formatted Value of Gender&quot; ;
IF ( MISSING( Gender ) EQ 0 ) THEN DO;
  _UFormat_2 = STRIP( PUT( Gender , $GENDER_FMT5. ) );
  IF ( _UFormat_2
      NOTIN ( &quot;Man&quot;
            , &quot;Woman&quot;
            )
      ) THEN _nInputOutOfRange = _nInputOutOfRange + 1;
END;
ELSE _nInputMiss = _nInputMiss + 1;

```

```

LENGTH _UFormat_3 $ 3 ;
LABEL _UFormat_3 = &quot;Formatted Value of HomeOwner&quot; ;
IF ( MISSING( HomeOwner ) EQ 0 ) THEN DO;
  _UFormat_3 = STRIP( PUT( HomeOwner , HO_FMT3. ) );
  IF ( _UFormat_3
      NOTIN ( &quot;No&quot;
            , &quot;Yes&quot;
            )
      ) THEN _nInputOutOfRange = _nInputOutOfRange + 1;
END;
ELSE _nInputMiss = _nInputMiss + 1;

/*****
/* Set _WARN_ value */
*****/

_VALID2SCORE = 1;
LABEL _VALID2SCORE = &quot;Is this record valid to be scored? 1=Yes, 0=No&quot; ;

IF ( _nInputMiss GT 0 ) THEN DO;
  SUBSTR(_WARN_,1,1) = 'M';
  _VALID2SCORE = 0;
END;
IF ( _nInputOutOfRange GT 0 ) THEN DO;
  SUBSTR(_WARN_,2,1) = 'U';
  _VALID2SCORE = 0;
END;

/*****
/* Calculate scores only if current record contains valid values */
*****/

IF ( _VALID2SCORE EQ 1 ) THEN DO;

  _NU_MODEL = 0 ;
  _NU_ZEROMODEL = 0 ;

  _NU_MODEL = _NU_MODEL + 7.889048183464800E-01
  ;

  IF ( _UFormat_1 EQ &quot;SEDAN&quot;
      ) THEN DO;
    _NU_MODEL = _NU_MODEL - 4.983426513164500E-01
    ;
  END;

  IF ( _UFormat_1 EQ &quot;SPORT&quot;
      ) THEN DO;
    _NU_MODEL = _NU_MODEL + 4.985885591940500E-01
    ;
  END;

  _NU_MODEL = _NU_MODEL + 1.227923016048900E-02
  * AgeDriver
  ;

  IF ( _UFormat_2 EQ &quot;Man&quot;
      ) THEN DO;
    _NU_MODEL = _NU_MODEL + 1.503894036936300E+00
    ;
  END;

  _NU_ZEROMODEL = _NU_ZEROMODEL + 9.925866013120000E-01
  ;

```

```

IF ( _UFormat_3 EQ &quot;No&quot;
  ) THEN DO;
  _NU_ZEROMODEL = _NU_ZEROMODEL + 6.905739218180000E-01
  ;
END;

_NU_ZEROMODEL = _NU_ZEROMODEL - 4.346588113784800E-02
  * IS
  * IS
  ;

_LOG_TAIL_P_ = LOGSDF( 'NORMAL' , _NU_ZEROMODEL );

IF ( ( _NU_MODEL + _LOG_TAIL_P_ ) LE 709.780 )
THEN MyPred_nClaim = EXP( _NU_MODEL + _LOG_TAIL_P_ );
ELSE MyPred_nClaim = .;

MyPhi_nClaim = 1 - EXP( _LOG_TAIL_P_ );

END; /* END ( _VALID2SCORE EQ 1) IF BLOCK */

LABEL MyPred_nClaim = &quot;Predicted value of nClaim&quot; ;
LABEL MyPhi_nClaim = &quot;Probability of nClaim being zero as a result
of the zero-generating process&quot; ;

DROP _nInputMiss _VALID2SCORE _NU_MODEL;
DROP _NU_ZEROMODEL _LOG_TAIL_P_;
DROP _nInputOutOfRange
  _UFormat_1
  _UFormat_2
  _UFormat_3
  ;

/*****
/* End scoring code for COUNTREG */
*****/

```

Output A6.2 The Tables Created by the Sample Program

## The SAS System

### The COUNTREG Procedure

Class Level Information		
Class	Levels	Values
CarType	3	SEDAN SPORT TRUCK
Gender	2	Man Woman
HomeOwner	2	No Yes

Model Fit Summary	
Dependent Variable	nClaim
Number of Observations	582162
Missing Values	67258
Data Set	CNTREG.PHF
Model	ZIP
ZI Link Function	Normal
Log Likelihood	-283522
Maximum Absolute Gradient	0.00229
Number of Iterations	9
Optimization Method	Newton-Raphson
AIC	567066
SBC	567190

### The SAS System

#### The COUNTREG Procedure

Class Level Information		
Class	Levels	Values
CarType	3	SEDAN SPORT TRUCK
Gender	2	Man Woman
HomeOwner	2	No Yes

Model Fit Summary	
Dependent Variable	nClaim
Number of Observations	582162
Missing Values	67258
Data Set	CNTREG.PHF
Model	ZIP
ZI Link Function	Normal
Log Likelihood	-283522
Maximum Absolute Gradient	0.00229
Number of Iterations	9
Optimization Method	Newton-Raphson
AIC	567066
SBC	567190

Algorithm converged.

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	t Value	Approx Pr >  t
Intercept	1	0.788905	0.015473	50.99	<.0001
CarType SEDAN	1	-0.498343	0.003396	-146.76	<.0001
CarType SPORT	1	0.498589	0.003280	151.99	<.0001
CarType TRUCK	0	0	.	.	.
AgeDriver	1	0.012279	0.000329	37.27	<.0001
Gender Man	1	1.503894	0.003976	378.23	<.0001
Gender Woman	0	0	.	.	.
Inf_Intercept	1	0.992587	0.004642	213.84	<.0001
Inf_HomeOwner No	1	0.690574	0.004973	138.87	<.0001
Inf_HomeOwner Yes	0	0	.	.	.
Inf_IS*IS	1	-0.043466	0.001068	-40.70	<.0001



The SAS System

Obs	NAME	_LABEL_	_VALUE_1	_VALUE_2	_VALUE_3	MacVar
1	CARTYPE	Type of Car	SEDAN	SPORT	TRUCK	CarType

The SAS System

Obs	NAME	_LABEL_	_VALUE_1	_VALUE_2	MacVar
1	GENDER	Gender Identification	Man	Woman	Gender

The SAS System

Obs	NAME	_VALUE_1	_VALUE_2	MacVar
1	HOMEOWNER	No	Yes	HomeOwner

The SAS System

The CONTENTS Procedure

<b>Data Set Name</b>	CNTREG.PHF_PRED_COMPARE	<b>Observations</b>	649420
<b>Member Type</b>	DATA	<b>Variables</b>	25
<b>Engine</b>	V9	<b>Indexes</b>	0
<b>Created</b>	04/25/2013 16:49:55	<b>Observation Length</b>	192
<b>Last Modified</b>	04/25/2013 16:49:55	<b>Deleted Observations</b>	0
<b>Protection</b>		<b>Compressed</b>	NO
<b>Data Set Type</b>		<b>Sorted</b>	NO
<b>Label</b>			
<b>Data Representation</b>	WINDOWS_64		
<b>Encoding</b>	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1911
First Data Page	1
Max Obs per Page	340
Obs in First Data Page	321
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Users\lemdev\CountReg\Test\phf_pred_compare.sas7bdat
Release Created	9.0401B0
Host Created	X64_S08R2

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
6	AgeDriver	Num	8		Driver Age
3	CarType	Char	5		Type of Car
11	EExp	Num	8		
5	Estimate	Char	6		Gender Identification (Copy)
10	ExpYr	Num	8		
4	Gender	Char	6	\$GENDER_FMT.	Gender Identification
7	HomeOwner	Num	8	HO_FMT.	
13	IS	Num	8		
23	IsMiss_MyPhi_nClaim	Num	8		
22	IsMiss_MyPred_nClaim	Num	8		
21	IsMiss_Phi_nClaim	Num	8		
20	IsMiss_Pred_nClaim	Num	8		
25	MyDiffPhi	Num	8		
24	MyDiffPred	Num	8		
12	MyOffset	Num	8		
19	MyPhi_nClaim	Num	8		Probability of nClaim being zero as a result of the zero-generating process
18	MyPred_nClaim	Num	8		Predicted value of nClaim
2	Phi_nClaim	Num	8		Probability of nClaim being zero
8	PolicyId	Num	8		
1	Pred_nClaim	Num	8		Predicted value of nClaim
9	StartYr	Num	8		
17	_WARN_	Char	4		Warnings
14	fiS	Num	8		
16	nClaim	Num	8		
15	phi	Num	8		

The SAS System

The FREQ Procedure

Warnings				
_WARN_	Frequency	Percent	Cumulative Frequency	Cumulative Percent
M	67258	100.00	67258	100.00
Frequency Missing = 582162				

The SAS System

			Predicted value of nClaim	Predicted value of nClaim	MyDiffPred
IsMiss_Pred_nClaim	IsMiss_MyPred_nClaim				
0	0	N	582162	582162	582162
		NMiss	0	0	0
		Mean	9.429365968097200E-01	9.429365968097100E-01	-2.495081706328000E-14
		StdDev	1.064955276559700E+00	1.064955276559700E+00	4.181980199999600E-14
		Min	9.128948368471800E-02	9.128948368471800E-02	-3.437250484239400E-13
		Max	7.825944704397700E+00	7.825944704397300E+00	3.053113317719100E-16
	1	N	67258	0	0
		NMiss	0	67258	67258
		Mean	3.362464926679000E-01	.	.
		StdDev	3.193524985874900E-01	.	.
		Min	9.128948368471800E-02	.	.
		Max	4.525549373539800E+00	.	.

The SAS System

			Probability of nClaim being zero	Probability of nClaim being zero as a result of the zero-generating process	MyDiffPhi
IsMiss_Phi_nClaim	IsMiss_MyPhi_nClaim				
0	0	N	582162	582162	582162
		NMiss	0	0	0
		Mean	9.104499873516800E-01	9.104499873516800E-01	-8.436792562164200E-16
		StdDev	5.836674095312900E-02	5.836674095312900E-02	3.895804002996700E-16
		Min	7.645221218601200E-01	7.645221218601200E-01	-1.443289932012700E-15
		Max	9.527668240990100E-01	9.527668240990000E-01	-2.220446049250300E-16
	1	N	67258	0	0
		NMiss	0	67258	67258
		Mean	9.105934041870400E-01	.	.
		StdDev	5.811672446138200E-02	.	.
		Min	7.645221218601200E-01	.	.
		Max	9.527668240990100E-01	.	.

---

## %MM\_Severity\_Create\_Scorecode Autocall Macro

Creates DATA step statements to compute the predicted values of a model that you create using the SEVERITY procedure.

---

### Syntax

```
%MM_Severity_Create_Scorecode (
    ParmEst=severity-parameter-estimate-data-set
    ModelInfo=model-info-data-set<FileRef=output-fileref>
    <PredPrefix=dependent-variable-prefix>
) / store secure;
```

### Arguments

**ParmEst=*severity-parameter-estimate-dataset***

specifies the name of the parameter estimations output data. This data set is created when you specify the OUTEST= option in the PROC SEVERITY statement.

**ModelInfo=*model-info-data-set***

specifies the name of the model information output data set. This data set is created when you specify the OUTMODELINGINFO= option in the PROC SEVERITY statement.

**FileRef=*output-fileref***

specifies the fileref that defines the location of the macro output files.

**Default** The SAS log

**PredPrefix=*dependent-variable-prefix***

specifies a prefix for the predicted dependent variable. The variable is named in the PROC SEVERITY LOSS= statement. When is prefix is applied to the dependent variable, this new name becomes the prediction variable.

**Default** P\_

### Details

To create score code for a model that you create with PROC SEVERITY, include the following SAS code:

1. Use a LIBNAME statement to identify the location of the output that you create using PROC SEVERITY.
2. Build your model using PROC SEVERITY. Specify the OUTEST= option to create the ParameterEstimates data. Specify OUTMODELINGINFO= option to create the ModelInformation data set. Close the ODS OUTPUT destination.
3. Use the FILENAME statement to define a fileref for the macro output location.
4. Invoke the %MM\_Severity\_Create\_Scorecode Macro.

---

## Example: Generate the PROC SEVERITY Score Code for Insurance Risk

### Create the Sample Insurance Data

```
%let MyProj = C:\Users\myID;
%let MyProj = C:\MyJob\Projects;
libname Severity &quot;&amp;MyProj.\Severity\Test&quot;;

data Severity.phf;

    /* Regression Coefficient for the Intercept Term */
    retain fIntercept 6.8024;

    /* Regression Coefficient for continuous AgeDriver */
    retain fAgeDriver 0.01234;

    /* Regression Coefficient for the three dummy indicators for nominal CarType */
    retain fCarType_SEDAN 0;
    retain fCarType_SPORT 1.0;
    retain fCarType_TRUCK 0.5;

    /* Regression Coefficient for the two dummy indicators for nominal Gender */
    retain fGender_Female -1.5;
    retain fGender_Male 0;

    /* Regression Coefficient for the two dummy indicators for nominal HomeOwner */
    retain fHomeOwner_NO 0;
    retain fHomeOwner_YES 0.7;

    /* Regression Coefficient for continuous IS */
    retain fIS -0.00456;

    /* Regression Coefficient for continuous MileageDriven */
    retain fMileageDriven 0.013579;

    /* Variable Labels */
    label AgeDriver = 'Age of Driver';

    label AmountLoss = 'Amount of Loss in Dollars';
    format AmountLoss dollar.;

    label CarType_SEDAN = 'Indicator of Car Type is Sedan';
    label CarType_SPORT = 'Indicator of Car Type is Sport';
    label CarType_TRUCK = 'Indicator of Car Type is Truck';

    label EExp = 'Earned Exposure in Units of One Year';

    label ExpYr = 'Exposure Year';

    label Gender_Female = 'Indicator of Gender is Female';
    label Gender_Male = 'Indicator of Gender is Male';
```

```

label HomeOwner_NO = 'Indicator of Home Ownership is No';
label HomeOwner_YES = 'Indicator of Home Ownership is Yes';

label IS = 'Insurance Score of Driver';

label MileageDriven = 'Mileage Driven in Units of 1,000 Miles';

label PolicyId = 'Policy Identifier';

call streaminit(27513);
do PolicyId = 00001 to 99999;
  StartYr = 2000 +
    rand('table', 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1);
  do ExpYr = StartYr to 2011;
    EExp = rand('uniform');

    AgeDriver = 18 + rand('binomial',0.375, 72);

    CarType_SEDAN = 0;
    CarType_SPORT = 0;
    CarType_TRUCK = 0;
    select (rand('table', 0.4999, 0.2999, 0.1999, 0.0003));
      when (1) CarType_SEDAN = 1;
      when (2) CarType_SPORT = 1;
      when (3) CarType_TRUCK = 1;
      otherwise
      do;
        CarType_SEDAN = .;
        CarType_SPORT = .;
        CarType_TRUCK = .;
      end;
    end;
  end;

  Gender_Female = 0;
  Gender_Male = 0;
  if (EExp lt 0.4999) then Gender_Female = 1;
  else if (EExp lt 0.9999) then Gender_Male = 1;
  else
  do;
    Gender_Female = .;
    Gender_Male = .;
  end;

  HomeOwner_NO = 0;
  HomeOwner_YES = 0;
  if (rand('bernoulli', 0.25) eq 1) then HomeOwner_YES = 1;
  else HomeOwner_NO = 1;

  IS = round(rand('gamma', 600));
  if (IS gt 800) then IS = 800;
  else if (IS lt 1) then IS = 1;

  MileageDriven = rand('gamma', 12);
  /* Annual Mileage Driven in unit of 1000 miles */

```

```

if (nmiss(MileageDriven, AgeDriver, CarType_SEDAN, CarType_TRUCK,
         CarType_SPORT,
         Gender_Male, Gender_Female,
         HomeOwner_YES, HomeOwner_NO, IS) eq 0)
  then
do;
  mu = fIntercept
      + fAgeDriver * (28 - AgeDriver)
      + fCarType_SEDAN * CarType_SEDAN + fCarType_SPORT
      * CarType_SPORT
      + fCarType_TRUCK * CarType_TRUCK
      + fGender_Female * Gender_Female + fGender_Male
      * Gender_Male
      + fHomeOwner_NO * HomeOwner_NO + fHomeOwner_YES
      * HomeOwner_YES
      + fIS * IS
      + fMileageDriven * (MileageDriven - 12);
  AmountLoss = exp(mu) * rand('gamma', 25);
end;
else AmountLoss = .;
output;

end;
end;
drop fAgeDriver fCarType_SEDAN fCarType_TRUCK fCarType_SPORT fGender_Male
     fGender_Female fHomeOwner_YES fHomeOwner_NO fIntercept fIS fMileageDriven;
drop mu StartYr;
run;

```

### Run the Sample Program

```

%let MyProj = C:\Users\emdev;
%let MyProj = C:\Users\minlam\Documents\Projects;
libname Severity "&MyProj.\Severity\Test";

title "SCALEMODEL and all applicable distributions";

%let model = 1;
%let predlist = AgeDriver CarType_SEDAN CarType_TRUCK CarType_SPORT
               Gender_Male Gender_Female HomeOwner_YES HomeOwner_NO IS MileageDriven;

/* Build the model and obtain the required datasets */
proc severity data = Severity.phf
              outest = Severity.ParamEst_&Model.
              outmodelinfo = Severity.ModelInfo_&model.;
  loss AmountLoss;
  dist _predefined_ stweedie;
  scalemodel &predlist.;
  nloptions maxiter = 1000;
run;

/* Define the fileref for the output syntax */
filename ThisFile "&MyProj.\Severity\Test\ScoreCode_&Model..sas";

/* Invoke the macro */
%mm_severity_create_scorecode
(

```

```

        ParamEst = Severity.ParamEst_&Model.,
        ModelInfo = Severity.ModelInfo_&model.,
        FileRef = ThisFile,
        PredPrefix = MyPred_,
    );

    /* Execute the score codes within a DATA STEP */
    data Severity.phf_wPrediction;
        set Severity.phf;
        %include ThisFile;
    run;

    proc contents data = Severity.phf_wPrediction;
    run;

```

## The Generated Score Code and Output Tables

### Output A6.3 The Score Code That Is Generated by SAS Model Manager

```

/*****
/* Begin scoring code for SEVERITY                                     */
/* Created By: emdev                                                 */
/* Date: May 15, 2013                                               */
/* Time: 12:06:28                                                  */
*****/

LENGTH _WARN_ $ 4;
_WARN_ = ' ';
LABEL _WARN_ = "Warnings" ;

_nInputMiss = 0;

/*****
/* Check the SCALEMODEL regression variables                         */
*****/

IF ( MISSING( MileageDriven ) EQ 1 ) THEN _nInputMiss = _nInputMiss + 1;

IF ( MISSING( IS ) EQ 1 ) THEN _nInputMiss = _nInputMiss + 1;

/* NOTE: HomeOwner_NO is not checked for missing values because it is a redundant
regressor. */

IF ( MISSING( HomeOwner_YES ) EQ 1 ) THEN _nInputMiss = _nInputMiss + 1;

IF ( MISSING( Gender_Female ) EQ 1 ) THEN _nInputMiss = _nInputMiss + 1;

/* NOTE: Gender_Male is not checked for missing values because it is a redundant
regressor. */

IF ( MISSING( CarType_SPORT ) EQ 1 ) THEN _nInputMiss = _nInputMiss + 1;

IF ( MISSING( CarType_TRUCK ) EQ 1 ) THEN _nInputMiss = _nInputMiss + 1;

/* NOTE: CarType_SEDAN is not checked for missing values because it is a
redundant regressor. */

```



```

IF ( MISSING( AgeDriver ) EQ 1 ) THEN _nInputMiss = _nInputMiss + 1;

/*****
/* Calculate scores only if current record contains valid values */
*****/

IF ( _nInputMiss EQ 0 ) THEN DO;

/*****
/* Distribution: BURR */
*****/

_XBETA_ = 0;
_XBETA_ = _XBETA_ + 1.344413338897300E-02 * MileageDriven ;
_XBETA_ = _XBETA_ - 4.570022585401000E-03 * IS ;
/* NOTE: HomeOwner_NO is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 7.004995716974000E-01 * HomeOwner_YES ;
_XBETA_ = _XBETA_ - 1.499608973328200E+00 * Gender_Female ;
/* NOTE: Gender_Male is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 9.997068084571000E-01 * CarType_SPORT ;
_XBETA_ = _XBETA_ + 4.992376225576000E-01 * CarType_TRUCK ;
/* NOTE: CarType_SEDAN is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ - 1.240862927421100E-02 * AgeDriver ;

/* NOTE: MyPred_AmountLoss_BURR = GAMMA(1 + 1/Gamma) * GAMMA(Alpha - 1/
Gamma) / GAMMA(Alpha) * Theta * EXP(_XBETA_) */
MyPred_AmountLoss_BURR = 2.729080537097400E+04 * EXP(_XBETA_);

/*****
/* Distribution: EXP */
*****/

_XBETA_ = 0;
_XBETA_ = _XBETA_ + 1.344997925413300E-02 * MileageDriven ;
_XBETA_ = _XBETA_ - 4.570912461736200E-03 * IS ;
/* NOTE: HomeOwner_NO is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 7.006247728147500E-01 * HomeOwner_YES ;
_XBETA_ = _XBETA_ - 1.499520049919700E+00 * Gender_Female ;
/* NOTE: Gender_Male is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 9.998154659889200E-01 * CarType_SPORT ;
_XBETA_ = _XBETA_ + 4.991760040523900E-01 * CarType_TRUCK ;
/* NOTE: CarType_SEDAN is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ - 1.240552808055000E-02 * AgeDriver ;

/* NOTE: MyPred_AmountLoss_EXP = 1 * Theta * EXP(_XBETA_) */
MyPred_AmountLoss_EXP = 2.730403090782800E+04 * EXP(_XBETA_);

/*****
/* Distribution: GAMMA */
*****/

_XBETA_ = 0;
_XBETA_ = _XBETA_ + 1.344997933121800E-02 * MileageDriven ;
_XBETA_ = _XBETA_ - 4.570912459089100E-03 * IS ;
/* NOTE: HomeOwner_NO is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 7.006247729158700E-01 * HomeOwner_YES ;
_XBETA_ = _XBETA_ - 1.499520049929900E+00 * Gender_Female ;
/* NOTE: Gender_Male is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 9.998154662540800E-01 * CarType_SPORT ;
_XBETA_ = _XBETA_ + 4.991760039384900E-01 * CarType_TRUCK ;
/* NOTE: CarType_SEDAN is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ - 1.240552804902500E-02 * AgeDriver ;

/* NOTE: MyPred_AmountLoss_GAMMA = Alpha * Theta * EXP(_XBETA_) */
MyPred_AmountLoss_GAMMA = 2.730405549354900E+04 * EXP(_XBETA_);

```

```

/*****
/* Distribution: GPD
*****/

_XBETA_ = 0;
_XBETA_ = _XBETA_ + 1.345090198095600E-02 * MileageDriven ;
_XBETA_ = _XBETA_ - 4.569987445112400E-03 * IS ;
/* NOTE: HomeOwner_NO is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 7.006260117108500E-01 * HomeOwner_YES ;
_XBETA_ = _XBETA_ - 1.499518185889600E+00 * Gender_Female ;
/* NOTE: Gender_Male is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 9.998173115643000E-01 * CarType_SPORT ;
_XBETA_ = _XBETA_ + 4.991779592407300E-01 * CarType_TRUCK ;
/* NOTE: CarType_SEDAN is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ - 1.240306956097400E-02 * AgeDriver ;

/* NOTE: MyPred_AmountLoss_GPD = 1 / (1 - Xi) * Theta * EXP(_XBETA_) */
MyPred_AmountLoss_GPD = 2.728530960810300E+04 * EXP(_XBETA_);

/*****
/* Distribution: IGAUSS
*****/

_XBETA_ = 0;
_XBETA_ = _XBETA_ + 1.344333259472700E-02 * MileageDriven ;
_XBETA_ = _XBETA_ - 4.572143035437300E-03 * IS ;
/* NOTE: HomeOwner_NO is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 7.006130278360900E-01 * HomeOwner_YES ;
_XBETA_ = _XBETA_ - 1.499413162869200E+00 * Gender_Female ;
/* NOTE: Gender_Male is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 9.997768449055400E-01 * CarType_SPORT ;
_XBETA_ = _XBETA_ + 4.991467916958200E-01 * CarType_TRUCK ;
/* NOTE: CarType_SEDAN is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ - 1.241894446056000E-02 * AgeDriver ;

/* NOTE: MyPred_AmountLoss_IGAUSS = 1 * Theta * EXP(_XBETA_) */
MyPred_AmountLoss_IGAUSS = 2.734200065401500E+04 * EXP(_XBETA_);

/*****
/* Distribution: LOGN
*****/

_XBETA_ = 0;
_XBETA_ = _XBETA_ + 1.344165732039300E-02 * MileageDriven ;
_XBETA_ = _XBETA_ - 4.571943993968500E-03 * IS ;
/* NOTE: HomeOwner_NO is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 7.006057373962000E-01 * HomeOwner_YES ;
_XBETA_ = _XBETA_ - 1.499427525786600E+00 * Gender_Female ;
/* NOTE: Gender_Male is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 9.997737501240600E-01 * CarType_SPORT ;
_XBETA_ = _XBETA_ + 4.991635085860000E-01 * CarType_TRUCK ;
/* NOTE: CarType_SEDAN is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ - 1.241995587270100E-02 * AgeDriver ;

/* NOTE: MyPred_AmountLoss_LOGN = EXP(Sigma*Sigma/2) * EXP(Mu) * EXP(_XBETA_)
*/
MyPred_AmountLoss_LOGN = 2.734808258530300E+04 * EXP(_XBETA_);

```

```

/*****
/* Distribution: PARETO
/*****

_XBETA_ = 0;
_XBETA_ = _XBETA_ + 1.344702841631400E-02 * MileageDriven ;
_XBETA_ = _XBETA_ - 4.573870525107900E-03 * IS ;
/* NOTE: HomeOwner_NO is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 7.006208104018000E-01 * HomeOwner_YES ;
_XBETA_ = _XBETA_ - 1.499526007562500E+00 * Gender_Female ;
/* NOTE: Gender_Male is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 9.998095632861300E-01 * CarType_SPORT ;
_XBETA_ = _XBETA_ + 4.991697518801100E-01 * CarType_TRUCK ;
/* NOTE: CarType_SEDAN is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ - 1.241339040886400E-02 * AgeDriver ;

/* NOTE: MyPred_AmountLoss_PARETO = 1 / (Alpha - 1) * Theta * EXP(_XBETA_) */
MyPred_AmountLoss_PARETO = 2.736400276713000E+04 * EXP(_XBETA_);

/*****
/* Distribution: STWEEDIE
/*****

_XBETA_ = 0;
_XBETA_ = _XBETA_ + 1.345898239828500E-02 * MileageDriven ;
_XBETA_ = _XBETA_ - 4.570041179676800E-03 * IS ;
/* NOTE: HomeOwner_NO is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 7.006207334939600E-01 * HomeOwner_YES ;
_XBETA_ = _XBETA_ - 1.499496885374900E+00 * Gender_Female ;
/* NOTE: Gender_Male is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 9.998043617045600E-01 * CarType_SPORT ;
_XBETA_ = _XBETA_ + 4.991607418998200E-01 * CarType_TRUCK ;
/* NOTE: CarType_SEDAN is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ - 1.239129336545400E-02 * AgeDriver ;

/* NOTE: MyPred_AmountLoss_STWEEDIE = Lambda * (2 - P) / (P - 1) * Theta *
EXP(_XBETA_) */
MyPred_AmountLoss_STWEEDIE = 2.726265339822600E+04 * EXP(_XBETA_);

/*****
/* Distribution: WEIBULL
/*****

_XBETA_ = 0;
_XBETA_ = _XBETA_ + 1.350103959448200E-02 * MileageDriven ;
_XBETA_ = _XBETA_ - 4.569462924201300E-03 * IS ;
/* NOTE: HomeOwner_NO is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 7.007182928623600E-01 * HomeOwner_YES ;
_XBETA_ = _XBETA_ - 1.499697786981000E+00 * Gender_Female ;
/* NOTE: Gender_Male is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ + 1.000109511872200E+00 * CarType_SPORT ;
_XBETA_ = _XBETA_ + 4.989436512356800E-01 * CarType_TRUCK ;
/* NOTE: CarType_SEDAN is skipped because it is a redundant regressor. */
_XBETA_ = _XBETA_ - 1.233857684563600E-02 * AgeDriver ;

/* NOTE: MyPred_AmountLoss_WEIBULL = GAMMA(1 + 1/Tau) * Theta * EXP(_XBETA_) */
MyPred_AmountLoss_WEIBULL = 2.707248194039600E+04 * EXP(_XBETA_);

END;

```

```

ELSE DO;

  /*****
  /* Set _WARN_ value
  *****/

  SUBSTR(_WARN_,1,1) = 'M';
END;

LABEL MyPred_AmountLoss_BURR = "Predicted Mean for the Burr Distribution" ;

LABEL MyPred_AmountLoss_EXP = "Predicted Mean for the Exponential Distribution" ;

LABEL MyPred_AmountLoss_GAMMA = "Predicted Mean for the Gamma Distribution" ;

LABEL MyPred_AmountLoss_GPD = "Predicted Mean for the Generalized Pareto
Distribution" ;

LABEL MyPred_AmountLoss_IGAUSS = "Predicted Mean for the Inverse Gaussian
Distribution" ;

LABEL MyPred_AmountLoss_LOGN = "Predicted Mean for the Lognormal Distribution" ;

LABEL MyPred_AmountLoss_PARETO = "Predicted Mean for the Pareto Distribution" ;

LABEL MyPred_AmountLoss_STWEEDIE = "Predicted Mean for the Tweedie Distribution
with Scale Parameter" ;

LABEL MyPred_AmountLoss_WEIBULL = "Predicted Mean for the Weibull Distribution" ;

DROP _nInputMiss _XBETA_;

/*****
/* End scoring code for SEVERITY
*****/

```

The following tables are a sampling of the output tables that are created by the example. For each distribution type, PROC SEVERITY creates these tables: Distribution

Information, Convergence Status, Optimization Summary, Fit Statistics, and Parameterization Estimation. The output displays the tables for the stweedie distribution.

### SCALEMODEL and all applicable distributions

#### The SEVERITY Procedure

##### Input Data Set

<b>Name</b>	SEVERITY.PHF
-------------	--------------

##### Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
stweedie	Yes	8686129	No
Burr	Yes	8690692	No
Exp	Yes	10242371	No
Gamma	Yes	8681780	Yes
Igauss	Yes	8686498	No
Logn	Yes	8686067	No
Pareto	Yes	10242384	No
Gpd	Yes	10242371	No
Weibull	Yes	8749362	No

### The SEVERITY Procedure stweedie Distribution

Distribution Information	
Name	stweedie
Description	Tweedie Distribution with Scale Parameter
Distribution Parameters	3
Regression Parameters	7

Convergence Status
Convergence criterion (GCONV=1E-8) satisfied.

Optimization Summary	
Optimization Technique	Trust Region
Iterations	4
Function Calls	22
Log Likelihood	-4343064.7

Fit Statistics	
-2 Log Likelihood	8686129
AIC	8686149
AICC	8686149
BIC	8686263
Kolmogorov-Smirnov	273.53175
Anderson-Darling	9096662
Cramer-von Mises	30389

Parameter Estimates				
Parameter	Estimate	Standard Error	t Value	Approx Pr >  t
Theta	40.79128	10.38232	3.93	<.0001
Lambda	25.99009	0.25302	102.72	<.0001
P	1.03743	0.00951	109.07	<.0001
AgeDriver	-0.01239	0.0000599	-206.80	<.0001
CarType_TRUCK	0.49916	0.0006506	767.27	<.0001
CarType_SPORT	0.99980	0.0005676	1761.48	<.0001
Gender_Female	-1.49950	0.0004918	-3049.2	<.0001
HomeOwner_YES	0.70062	0.0005680	1233.52	<.0001
IS	-0.00457	.	.	.
MileageDriven	0.01346	0.0000709	189.82	<.0001

### SCALEMODEL and all applicable distributions

#### The CONTENTS Procedure

Data Set Name	SEVERITY.PHF_WPREDICTION	Observations	648370
Member Type	DATA	Variables	24
Engine	V9	Indexes	0
Created	05/15/2013 12:06:28	Observation Length	192
Last Modified	05/15/2013 12:06:28	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1908
First Data Page	1
Max Obs per Page	340
Obs in First Data Page	318
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Users\lemdev\Severity\phf_wprediction.sas7bdat
Release Created	9.0401B0
Host Created	X64_S08R2



Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
1	AgeDriver	Num	8		Age of Driver
2	AmountLoss	Num	8	DOLLAR.	Amount of Loss in Dollars
3	CarType_SEDAN	Num	8		Indicator of Car Type is Sedan
4	CarType_SPORT	Num	8		Indicator of Car Type is Sport
5	CarType_TRUCK	Num	8		Indicator of Car Type is Truck
6	EExp	Num	8		Earned Exposure in Units of One Year
7	ExpYr	Num	8		Exposure Year
8	Gender_Female	Num	8		Indicator of Gender is Female
9	Gender_Male	Num	8		Indicator of Gender is Male
10	HomeOwner_NO	Num	8		Indicator of Home Ownership is No
11	HomeOwner_YES	Num	8		Indicator of Home Ownership is Yes
12	IS	Num	8		Insurance Score of Driver
13	MileageDriven	Num	8		Mileage Driven in Units of 1,000 Miles
16	MyPred_AmountLoss_BURR	Num	8		Predicted Mean for the Burr Distribution
17	MyPred_AmountLoss_EXP	Num	8		Predicted Mean for the Exponential Distribution
18	MyPred_AmountLoss_GAMMA	Num	8		Predicted Mean for the Gamma Distribution
19	MyPred_AmountLoss_GPD	Num	8		Predicted Mean for the Generalized Pareto Distribution
20	MyPred_AmountLoss_IGAUSS	Num	8		Predicted Mean for the Inverse Gaussian Distribution
21	MyPred_AmountLoss_LOGN	Num	8		Predicted Mean for the Lognormal Distribution
22	MyPred_AmountLoss_PARETO	Num	8		Predicted Mean for the Pareto Distribution
23	MyPred_AmountLoss_STWEEDIE	Num	8		Predicted Mean for the Tweedie Distribution with Scale Parameter
24	MyPred_AmountLoss_WEIBULL	Num	8		Predicted Mean for the Weibull Distribution
14	PolicyId	Num	8		Policy Identifier
15	_WARN_	Char	4		Warnings



## Appendix 7

# Properties

---

<b>General Properties</b> . . . . .	<b>503</b>
<b>System Properties</b> . . . . .	<b>504</b>
<b>Specific Properties for a Project</b> . . . . .	<b>505</b>
<b>User-Defined Properties</b> . . . . .	<b>508</b>
Organization-Specific User-Defined Properties . . . . .	508
SAS User-Defined Properties . . . . .	510
<b>Specific Properties for a Version</b> . . . . .	<b>511</b>
<b>Specific Properties for Milestones and Tasks</b> . . . . .	<b>512</b>
<b>Specific Properties for a Model</b> . . . . .	<b>514</b>
<b>Scoring Task Properties</b> . . . . .	<b>516</b>
<b>Result Set Properties</b> . . . . .	<b>517</b>
<b>Schedule Properties</b> . . . . .	<b>518</b>

---

## General Properties

Here is a list of general properties that describe how the repository component is named, created, and updated.

Property Name	Description
<b>Name</b>	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.
<b>Description</b>	Specifies user-defined information about the component. You can modify this property.
<b>Owner</b>	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.
<b>Date Created</b>	Specifies the date on which the component was created. This property is Read-only.

Property Name	Description
<b>Date Modified</b>	Specifies the date on which 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
<b>UUID</b>	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 <b>cca1ab08-0a28-0e97-0051-0e3991080867</b> .
<b>SMM Path</b>	Specifies the SAS Model Manager path (SMM Path) that is a network path to the directory that contains the model folders and files. The format for an SMM version path is <b>//&lt;RepositoryID&gt;/MMRoot/&lt;Folder&gt;/&lt;Project&gt;/&lt;Version&gt;/&lt;ComponentType&gt;</b> . 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.
<b>URL</b>	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.
<b>Entity Key</b>	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.
<b>Repository Name</b>	Specifies the name of the repository that contains the component.

---

## Specific Properties for a Project

Property Name	Description
<b>Lock Project Metadata</b>	Specifies that the project metadata is locked and the project definition cannot be modified.
<b>Default Test Table</b>	Specifies a default SAS data set that can be used to create model assessment reports such as dynamic lift charts.
<b>Default Scoring Task Input Table</b>	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 <b>Default Scoring Task Input Table</b> property, the value is used as the default input table in the New Scoring Task window.
<b>Default Scoring Task Output Table</b>	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 <b>Default Scoring Task Output Table</b> property, the value is used as the default output table in the New Scoring Task window.
<b>Default Performance Table</b>	<p>Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager project.</p> <p>The value of the <b>Default Performance Table</b> property is used as the default value for the <b>Data Source</b> column in the Define Performance Task wizard if a default performance table is not specified for a version or for the model.</p>
<b>Default Train Table</b>	<p>The train table is optional and is used only as information. However, when a value is specified for a model's <b>Default Train Table</b> property, SAS Model Manager does the following:</p> <ul style="list-style-type: none"> <li>• uses default train table to validate scoring functions or scoring model files when a user publishes the associated project champion model or challenger models to a database.</li> <li>• checks the <b>Validate scoring results</b> box in the Publish Scoring Function window.</li> </ul> <p>The value of the <b>Default Train Table</b> property is used to validate scoring functions or scoring model files only if a default train table is not specified for a version or for the model.</p>

---

Property Name	Description
<b>State</b>	<p>Specifies the current state of the project:</p> <p>Under 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>
<b>Default Version</b>	<p>Specifies the version that contains the champion model in a production environment.</p>
<b>Model Function</b>	<p>Specifies the type of output that your predictive model project generates. The <b>Model Function</b> 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 <b>Model Function</b> 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 <a href="#">Types of Model Functions on page 507</a>.</p>
<b>Interested Party</b>	<p>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.</p>
<b>Training Target Variable</b>	<p>Specifies the name of the target variable that was used to train the model.</p>
<b>Target Event Value</b>	<p>The target variable value that defines the desired target variable event.</p>
<b>Class Target Values</b>	<p>For class, nominal, ordinal, or interval targets, the set of possible outcome classes, separated by commas. For example, binary class target values might be <b>1, 0</b> or <b>Yes, No</b>. Nominal class target values might be <b>Low, Medium, High</b>. These values are for information only.</p>
<b>Class Target Level</b>	<p>Specifies the class target level of binary, nominal, ordinal, or interval.</p>

Property Name	Description
<b>Output Event Probability Variable</b>	The output event probability variable name, when the <b>Model Function</b> property is set to <b>Classification</b> .
<b>Output Prediction Variable</b>	The output prediction variable name, when the <b>Model Function</b> property is set to <b>Prediction</b> .
<b>Output Classification Variable</b>	The output classification variable name, when the <b>Model Function</b> property is set to <b>Classification</b> .
<b>Output Segmentation Variable</b>	The output segmentation variable name, when the <b>Model Function</b> property is set to <b>Segmentation</b> .

Table A7.1 Types of Model Functions

Model Function	Description	Example
<b>Analytical</b>	Function for any model that is not Prediction, Classification, or Segmentation.	None
<b>Prediction</b>	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.
<b>Classification</b>	Function for models that have target variables that contain binary, categorical, or ordinal values.	<b>DEFAULT_RISK</b> = {Low, Med, High}
<b>Segmentation</b>	Function for segmentation or clustering models.	Clustering models
<b>Any</b>	Specify <b>Any</b> when you import a SAS code model and you want a choice of the model template to use in the <b>Local Files</b> window. When you specify <b>Any</b> , SAS Model Manager lists the available model templates in the <b>Choose a model template</b> list in the Local Files window.	None

---

## User-Defined Properties

### *Organization-Specific User-Defined Properties*

#### **Overview of 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 properties can be added by using the Details section of the Project category view or by using the SAS Model Manager Template Editor. When you use the Template Editor, you can add a user-defined property to a user-defined model template or you can upload a UserDefinedProperties.xml file. For information about adding user-defined properties to a user-defined model template, see [“User-Defined Model Templates” on page 148](#).

User-defined properties that are added using the Details section of the Project category view are added as a property only for the selected node. You name the property and can assign it a value.

When you add a user-defined property using the UserDefinedProperties.xml file, you specify the name of the property, the initial value of the property, and the type of node in the project tree for which it applies. The user-supplied property is created for the specified node type when the node is added to the project tree. For example, if the XML file specifies a user-defined property Version Date for a node type of Document, each time a Documents folder is added to the project tree, the folder has a property of Version Date. User-supplied properties are not added to existing nodes in the project tree.

You can specify these node types in the UserDefinedPropertyies.xml file:

- AdHocReport
- AggregatedReport
- AnalyticalModel
- ClassificationModel
- ClusteringModel
- Document
- Milestone
- MilestoneAction
- ModelGroup
- ModelRetrain
- ModelRetrainReport
- PerformanceMonitor
- PredictionModel
- Project
- ReportingTask
- ScoringTask
- Version



### Create User-Defined Properties Using the Properties Tab

To create a user-defined property:

1. In the **Properties** tab, right-click and select **Add User-Defined Property**. The Add User-Defined Property window appears.
2. In the **Name** field, add a name. The name can contain letters, the underscore character ( \_ ), and hyphens ( - ). The name cannot begin with a hyphen. Spaces cannot be used in the name.
3. In the **Value** field, enter a value for the user-defined property. Click **OK**. The user-defined property is added to the **Properties** section under **User-Defined Properties**.

### Create User-Defined Properties Using the UserDefinedProperties.xml File

#### CAUTION:

**Deleting the BusinessContext and DbmsTable properties might result in unexpected results.** SAS Model Manager provides user-supplied properties for use with SAS Real-Time Decision Manager and SAS In-Database scoring. Unexpected results might occur in these environments if you delete the BusinessContext property or the DbmsTable property. For more information, see [“SAS User-Defined Properties” on page 510](#).

To create a user-defined property:

1. Open the Template Editor by selecting **Tools** ⇒ **Manage Templates**
2. Select **Browse** ⇒ **Browse Templates** ⇒ **UserDefinedProperties.xml** ⇒ **Open**.
3. Add properties using a <Property/> element for each property. These arguments are required:

`name="property-name"`

specifies the name of the property.

`initial="initial-value "`

specifies a value for the property when it is added as a property for the specified project tree node. If you do not want to specify an initial value, use two double quotation marks, `initial=""`

`target="node-type"`

specifies the project tree node for which the user-supplied value applies. For a list of node types, see [“Overview of User-Defined Properties” on page 508](#).

Example: `<Property name="Version Date" initial=" " target="Documents"/>`

4. To upload the file to the SAS Content Server, select **File** ⇒ **Upload File** ⇒ **OK**.

Here is an example of a UserDefinedProperties.xml file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<FolderTemplate
  name="User_Defined_Properties">
<!--
  Target defines to apply the property to which folder type . Basically,
  it is referred as the argument type in MMRepositoryImpl.createNamedObject()
-->
<Properties>
  <Property name="BusinessContext" initial="" target="Project"/>
  <Property name="DbmsTable" initial="" target="Project"/>
```

```

<Property name="Division" initial="" target="Version"/>
<Property name="Consultant" initial="" target="ClassificationModel"/>
<Property name="Tester" initial="" target="ClassificationModel"/>
<Property name="Version Date" initial="" target="Documents"/>

</Properties>
</FolderTemplate>

```

## SAS User-Defined Properties

SAS creates some user-defined properties that are used by the SAS Real-Time Decision Manager and for scoring that is performed using SAS In-Database processing, such as scoring within a database.

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
<b>BusinessContext</b>	Specifies the business context for which the project is used.
<b>DbmsTable</b>	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 if you plan to reference it with a scoring application or in SAS code.

When you publish a model to a database for the first time, SAS creates some project user-defined properties. Some of these property values are assigned by SAS after you complete the **Publish Name** field in the Publish Models to a Database window.

Here are the publish models to a database user-defined properties.

Property Name	Description
<b>ModelNameForEP</b>	Specifies the publish name of the model that has been published to the database using the SAS Embedded Process.
<b>ScoringFunctionName</b>	For a model that has been published to the database using the scoring function, specifies the user-defined portion of the scoring function name. The name can contain letters, the underscore character ( _ ), and hyphens ( - ). The name cannot begin with a hyphen. Spaces cannot be used in the name.

Property Name	Description
<b>ScoringFunctionPrefix</b>	<p>Specifies a prefix for the scoring function name. The prefix has a length of 10 characters and in the format of <b>Yyyymmddnnn</b>. You cannot modify this value. The naming convention for the prefix is the following:</p> <ul style="list-style-type: none"> <li>• <b>Y</b> is a literal character and is fixed for all prefixes.</li> <li>• <b>yyymmdd</b> is the Greenwich Mean time (GMT) timestamp of when you select the <b>Publish Models</b> ⇒ <b>to a database</b> menu option.</li> <li>• <b>nnn</b> is a counter that increments by one each time the scoring function is successfully completed.</li> </ul>

---

## Specific Properties for a Version

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

Property Name	Description
<b>Default Scoring Task Input Table</b>	<p>Specifies a default SAS data set that is used as the input data table for all scoring tasks within the SAS Model Manager version. If you specify a value for the <b>Default Scoring Task Input Table</b> property, the value is used as the default input table in the New Scoring Task window if a default scoring task input table is not specified for the model.</p>
<b>Default Scoring Task Output Table</b>	<p>Specifies a default SAS data set that defines the variables to keep in the scoring results table of the scoring task. If you specify a value for the <b>Default Scoring Task Output Table</b> property, the value is used as the default output table in the New Scoring Task window if a default scoring task output table is not specified for the model.</p>
<b>Default Performance Table</b>	<p>Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager version.</p> <p>The value of the <b>Default Performance Table</b> property is used as the default value for the <b>Performance data source</b> field in the Define Performance Task wizard if a default performance table is not specified for the model.</p>

Property Name	Description
<b>Default Train Table</b>	<p>The train table is optional and is used for information as well as the Training Data Set Summary report. When a value is specified for a model's <b>Default Train Table</b> property, SAS Model Manager does the following:</p> <ul style="list-style-type: none"> <li>• uses default train table to validate scoring functions or scoring model files when a user publishes the associated project champion model or challenger models to a database.</li> <li>• checks the <b>Validate scoring results</b> box in the Publish Scoring Function window.</li> </ul> <p>The value of the <b>Default Train Table</b> property is used to validate scoring functions or scoring model files only if a default train table is not specified for the model.</p>
<b>State</b>	Specifies the current status of the version.
<b>Champion Model Name</b>	<p>If a champion model has been set for a project, specifies one of the following:</p> <ul style="list-style-type: none"> <li>• the name of the present champion model for the project</li> <li>• the name of the model that was last set as the champion model for the project, when the champion model has been retired or cleared</li> </ul>
<b>Date Frozen</b>	Specifies the date on which the version was frozen.
<b>Production Date</b>	Specifies the date on which the status of the <b>Production</b> milestone task in the version's life cycle was changed from <b>Started</b> to <b>Complete</b> .
<b>Date Retired</b>	Specifies the date on which the status of the <b>Retire</b> milestone task in the version's life cycle was changed from <b>Started</b> to <b>Complete</b> .

---

## Specific Properties for Milestones and Tasks

Here is a list of the milestone properties.

Property Name	Description
<b>Actual Start Date</b>	Specifies the actual date that the first task for the milestone is started. This property is Read-only.
<b>Actual End Date</b>	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 to <b>Completed</b> .
<b>Planned Start Date</b>	Specifies the expected date to start the first task for milestone.
<b>Planned End Date</b>	Specifies the expected date to complete all tasks for the milestone.

Here is a list of task properties:

Property Name	Description
<b>Status</b>	Specifies the status of task. Possible values are <b>Not Started</b> , <b>Started</b> , <b>Completed</b> , or <b>Approved</b> .
<b>Date Completed</b>	Specifies the date on which 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 <b>Completed</b> .
<b>Completed By</b>	Specifies the name of the user who completed the task. This property is Read-only.
<b>Date Approved</b>	Specifies the date on which completion of the task is approved. This property is Read-only.
<b>Approved By</b>	Specifies the name of the user who approved completion of the task. This property is Read-only.
<b>Planned Completion Date</b>	Specifies the expected date to complete the task.
<b>To Be Completed By</b>	Specifies the user who is responsible for completing the task.
<b>To Be Approved By</b>	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
<b>Default Scoring Task Input Table</b>	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 <b>Default Scoring Task Input Table</b> property inherits the property value from the associated version or project, if one is specified.
<b>Default Scoring Task Output Table</b>	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 <b>Default Scoring Task Output Table</b> property inherits the property value from the associated version or project, if one is specified.
<b>Default Performance Table</b>	<p>Specifies the default performance table for all model performance monitoring tasks within a SAS Model Manager project.</p> <p>A model's <b>Default Performance Table</b> 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>
<b>Default Train Table</b>	<p>The train table is optional and is used only as information. However, when a value is specified for a model's <b>Default Train Table</b> property, SAS Model Manager does the following:</p> <ul style="list-style-type: none"> <li>• uses default train table to validate scoring functions or scoring model files when a user publishes the associated project champion model or challenger models to a database.</li> <li>• checks the <b>Validate scoring results</b> box in the Publish Scoring Function window.</li> </ul>
<b>Expiration Date</b>	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
<b>Model Label</b>	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 <b>Model Label</b> property, SAS Model Manager uses the text string that is specified for the <b>Model Name</b> property. The <b>Model Label</b> property can be useful if the Model Name property that is specified is too long for use in plots. This property is optional.
<b>Subject</b>	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.
<b>Algorithm</b>	Specifies the computational algorithm that is used for the selected model. This property cannot be modified.
<b>Function</b>	Specifies the SAS Model Manager function class that was chosen when the SAS Model Manager associated project was created. The <b>Function</b> property specifies the type of output that models in the predictive model project generate. For more information, see <a href="#">“Overview of Importing Models” on page 125.</a>
<b>Modeler</b>	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.
<b>Tool</b>	Specifies whether the imported model came from SAS Enterprise Miner or from other modeling tools.
<b>Tool Version</b>	Specifies the version number of the tool that is specified in the <b>Tool</b> property.
<b>Score Code Type</b>	<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 <b>DATA step</b>, <b>SAS Program</b>, and <b>PMML</b>.</p> <p><i>Note:</i> If the model is created using PMML 4.0, the <b>Score Code Type</b> is <b>DATA step</b> and not <b>PMML</b>.</p> <p><i>Note:</i> SAS Model Manager cannot publish models to a database whose <b>Score Code Type</b> model property is set to <b>SAS Program</b> and <b>PMML</b>.</p>
<b>Template</b>	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
<b>Copied From</b>	Specifies where the original model is if this model is copied from another model in the SAS Model Manager repository.
<b>Target Variable</b>	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 is <b>GENDER</b> .
<b>Target Event Value</b>	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 <b>Target Variable</b> property. For example, if a model predicts when GENDER=M, then the target event value is <b>M</b> .

---

## 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
<b>Scoring Task Type</b>	Specifies a value of <b>Test</b> or <b>Production</b> for the type of scoring task.
<b>SAS Application Server</b>	Specifies the name of the SAS Application Server to which SAS Model Manager is connected. This value is taken from the SAS Metadata Repository.
<b>Model</b>	Specifies the name of the model whose score code is to be executed on the SAS Application Server. This value is set when the scoring task is created and cannot be modified.
<b>Input Table</b>	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
<b>Output Table</b>	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 <b>Test</b> , this property identifies the name of the output file ( <i>output_filename.sas7bdat</i> ) that is created by the SAS Application 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 <b>Production</b> , then this setting identifies the output table where the results of the scoring are written.

### See Also

- “Create a Scoring Task” on page 164
- “Modify a Scoring Task” on page 167
- “Execute a Scoring Task” on page 168
- “Schedule Scoring Tasks” on page 170

---

## Result Set Properties

The following property provides information that is specific to the scoring task.

Property Name	Description
<b>Number of Observations</b>	<p>When <b>Scoring Task Type</b> is set to <b>Test</b>, 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 <b>Scoring Task Type</b> is set to <b>Production</b>, this property specifies how many observations are to be read from the scoring task input table and displayed when you select <b>Result Set</b> from the <b>Results</b> tab. The default value is 0, indicating that there is no limit. This value cannot be changed in the SAS Model Manager client. The SAS Model Manager administrator can modify the value by using SAS Management Console. For more information, see <i>SAS Model Manager: Administrator's Guide</i>.</p>

---

**See Also**

- [“Create a Scoring Task” on page 164](#)
- [“Modify a Scoring Task” on page 167](#)

---

## Schedule Properties

Here are **Schedule** scoring task properties:

Property Name	Description
<b>Job Name</b>	Specifies the name of the scheduled job for a scoring task. The name of the scoring task is used by default as the job name.
<b>Location</b>	Specifies the location of the scoring job definition in the SAS Metadata Repository. You set this property using the <b>Advanced Settings</b> window.
<b>Scheduling Server</b>	Specifies the name of the server that schedules the job for the scoring task. You set this property using the <b>Advanced Settings</b> window.
<b>Batch Server</b>	Specifies the name of the server that executes the job for the scoring task. You set this property using the <b>Advanced Settings</b> window.
<b>Recurrence</b>	Specifies how often the scheduled job for the scoring task is to be executed: <b>None</b> , <b>Hourly</b> , <b>Daily</b> , <b>Weekly</b> , <b>Monthly</b> , and <b>Yearly</b> .
<b>SAS Application Server</b>	Specifies the name of the SAS Application Server where the scoring task is to be executed.

**See Also**

- [“Execute a Scoring Task” on page 168](#)
- [“Schedule Scoring Tasks” on page 170](#)

## Appendix 8

# SAS Model Manager R Model Support

---

<b>Overview of Using R Models with SAS Model Manager</b> . . . . .	<b>519</b>
<b>Preparing R Model Files to Use with SAS/IML</b> . . . . .	<b>520</b>
Build an R Model . . . . .	520
Prepare an R Model Template File . . . . .	521
Prepare R Model Component Files . . . . .	522

---

## 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:

- Ensure that the installed R language version is 2.13.0 or later.
- 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 520](#). 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 521](#).
4. Ensure that you have the required model component files. For more information, see [“Prepare R Model Component Files” on page 522](#).
5. Import the R model. For more information, see [“Importing an R Model” on page 142](#).

---

## 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_inde");
  submit /R;
    attach(mm_inde)

    # -----
    # 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 133 and [“User-Defined Model Templates”](#) on page 148.

## 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 133](#).

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 520](#). 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 520](#).

***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 Manage

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 9

# Statistical Measures Used in Basel II Reports

### Overview of Statistical Measures Used for Basel II Reports

SAS Model Manager Basel II reports use several statistical measures to validate the stability, performance, and calibration for the two key types of Basel II risk models: the Probability of Default (PD) model and the Loss Given Default (LGD) model.

The statistical measures for model validation are grouped into three categories:

Category	Description
Model Stability	Tracks the change in distribution of the modeling data and scoring data.
Model Performance	<ul style="list-style-type: none"> <li>Measures the ability of a model to discriminate between customers with accounts that have defaulted, and customers with accounts that have not defaulted. The score difference between non-default and default accounts helps determine the required cutoff score. The cutoff score helps predict whether a credit exposure is a default account.</li> <li>Measures the relationship between the actual default probability and the predicted default probability. This helps you understand the performance of a model over a time period.</li> </ul>
Model Calibration	Checks the accuracy of the PD and LGD models by comparing the correct quantification of the risk components with the available standards.

The sections that follow describe the measures, statistics, and tests that are used to create the PD and LGD reports. For more information about these measures, statistics, and tests, see the SAS Model Manager product documentation page on [support.sas.com](http://support.sas.com).

### Model Stability Measure

The following table describes the model stability measure that is used to create the PD report and the LGD reports.

Measure	Description	PD Report	LGD Report
System Stability Index (SSI)	SSI monitors the score distribution over a time period.	Yes	Yes

### Model Performance Measures and Statistics

The following table describes the model performance measures that are used to create the PD and LGD reports.

Measure	Description	PD Report	LGD Report
Accuracy	Accuracy is the proportion of the total number of predictions that were correct.	Yes	No
Accuracy Ratio (AR)	AR is the summary index of Cumulative Accuracy Profile (CAP) and is also known as Gini coefficient. It shows the performance of the model that is being evaluated by depicting the percentage of defaulted accounts that are captured by the model across different scores.	Yes	Yes
Area Under Curve (AUC)	AUC can be interpreted as the average ability of the rating model to accurately classify non-default accounts and default accounts. It represents the discrimination between the two populations. A higher area denotes higher discrimination. When AUC is 0.5, it means that non-default accounts and default accounts are randomly classified, and when AUC is 1, it means that the scoring model accurately classifies non-default accounts and default accounts. Thus, the AUC ranges between 0.5 and 1.	Yes	No
Bayesian Error Rate (BER)	BER is the proportion of the whole sample that is misclassified when the rating system is in optimal use. For a perfect rating model, the BER has a value of zero. A model's BER depends on the probability of default. The lower the BER, and the lower the classification error, the better the model.	Yes	No
D Statistic	The D Statistic is the mean difference of scores between default accounts and non-default accounts, weighted by the relative distribution of those scores.	Yes	No
Error Rate	The Error Rate is the proportion of the total number of incorrect predictions.	Yes	No

Measure	Description	PD Report	LGD Report
Information Statistic (I)	The Information Statistic value is a weighted sum of the difference between conditional default and conditional non-default rates. The higher the value, the more likely a model can predict a default account.	Yes	No
Kendall's Tau-b	Kendall's tau-b is a nonparametric measure of association based on the number of concordances and discordances in paired observations. Kendall's tau values range between -1 and +1, with a positive correlation indicating that the ranks of both variables increase together. A negative association indicates that as the rank of one variable increases, the rank of the other variable decreases.	Yes	No
Kullback-Leibler Statistic (KL)	KL is a non-symmetric measure of the difference between the distributions of default accounts and non-default accounts. This score has similar properties to the information value.	Yes	No
Kolmogorov-Smirnov Statistic (KS)	KS is the maximum distance between two population distributions. This statistic helps discriminate default accounts from non-default accounts. It is also used to determine the best cutoff in application scoring. The best cutoff maximizes KS, which becomes the best differentiator between the two populations. The KS value can range between 0 and 1, where 1 implies that the model is perfectly accurate in predicting default accounts or separating the two populations. A higher KS denotes a better model.	Yes	No
1-PH Statistic (1-PH)	1-PH is the percentage of cumulative non-default accounts for the cumulative 50% of the default accounts.	Yes	No
Mean Square Error (MSE), Mean Absolute Deviation (MAD), and Mean Absolute Percent Error (MAPE)	MSE, MAD, and MAPE are generated for LGD reports. These statistics measure the differences between the actual LGD and predicted LGD.	No	Yes

Measure	Description	PD Report	LGD Report
Pietra Index	<p>The Pietra Index is a summary index of Receiver Operating Characteristic (ROC) statistics because the Pietra Index is defined as the maximum area of a triangle that can be inscribed between the ROC curve and the diagonal of the unit square.</p> <p>The Pietra Index can take values between 0 and 0.353. As a rating model's performance improves, the value is closer to 0.353. This expression is interpreted as the maximum difference between the cumulative frequency distributions of default accounts and non-default accounts.</p>	Yes	No
Precision	Precision is the proportion of the actual default accounts among the predicted default accounts.	Yes	No
Sensitivity	Sensitivity is the ability to correctly classify default accounts that have actually defaulted.	Yes	No
Somers' D (p-value)	Somers' D is a nonparametric measure of association that is based on the number of concordances and discordances in paired observations. It is an asymmetric modification of Kendall's tau. Somers' D differs from Kendall's tau in that it uses a correction only for pairs that are tied on the independent variable. Values range between -1 and +1. A positive association indicates that the ranks for both variables increase together. A negative association indicates that as the rank of one variable increases, the rank of the other variable decreases.	Yes	No
Specificity	Specificity is the ability to correctly classify non-default accounts that have not defaulted.	Yes	No
Validation Score	The Validation Score is the average scaled value of seven distance measures, anchored to a scale of 1 to 13, lowest to highest. The seven measures are the mean difference (D), the percentage of cumulative non-default accounts for the cumulative 50% of the default accounts (1-PH), the maximum deviation (KS), the Gini coefficient (G), the Information Statistic (I), the Area Under the Curve (AUC), or Receiver Operating Characteristic (ROC) statistic, and the Kullback-Leibler statistic (KL).	Yes	No

## Model Calibration Measures and Tests

The following table describes the model calibration measures and tests that are used to create the PD and LGD reports:

Measure	Description	PD Report	LGD Report
Binomial Test	<p>The Binomial Test evaluates whether the PD of a pool is correctly estimated. It does not take into account correlated defaults, and it generally yields an overestimate of the significance of deviations in the realized default rate from the forecast rate. The Modified Binomial Test now addresses the overestimate. This test takes into account the correlated defaults<sup>1</sup>. The default correlation coefficient in SAS Model Manager is 0.04. By using past banking evaluations, you can use these rho values<sup>2</sup>:</p> <p>rho=0.04    Qualifying revolving retail</p> <p>rho=0.15    Residential mortgage</p> <p>rho=0.16    Other retail</p> <p>rho=0.24    Corporations, sovereign, and banks</p> <p>If the number of default accounts per pool exceeds either the low limit (binomial test at 0.95 confidence) or high limit (binomial test at 0.99 confidence), the test suggests that the model is poorly calibrated.</p> <p>To change the default rho value, contact your SAS Model Manager administrator. The value is a SAS Model Manager report option in SAS Management Console.</p>	Yes	No
Brier Skill Score (BSS)	BSS measures the accuracy of probability assessments at the account level. It measures the average squared deviation between predicted probabilities for a set of events and their outcomes. Therefore, a lower score represents a higher accuracy.	Yes	No

<sup>1</sup> Rauhmeier, Robert, and Englemann, Bernd. "PD Validation - Experience from Banking Practice." Available at <http://d.yimg.com/kq/groups/12093474/1121755262/name/The+Basel+II+Risk+Parameters.pdf>

<sup>2</sup> Rauhmeier, Robert, and Englemann, Bernd. "PD Validation - Experience from Banking Practice." Available at <http://d.yimg.com/kq/groups/12093474/1121755262/name/The+Basel+II+Risk+Parameters.pdf>

Measure	Description	PD Report	LGD Report
Confidence Interval	<p>The Confidence Interval indicates the confidence interval band of the PD or LGD for a pool. The Probability of Default report compares the actual and estimated PD rates with the CI limit of the estimate. If the estimated PD lies in the CI limits of the actual PD model, the PD performs better in estimating actual outcomes.</p> <p>For the Loss Given Default (LGD) report, confidence intervals are based on the pool-level average of the estimated LGD, plus or minus the pool-level standard deviation, and multiplied by the 1-(alpha/2) quantile of the standard normal distribution.</p>	Yes	Yes
Correlation Analysis	The model validation report for LGD provides a correlation analysis of the estimated LGD with the actual LGD. This correlation analysis is an important measure for a model's usefulness. The Pearson correlation coefficients are provided at the pool and overall levels for each time period are examined.	No	Yes
Hosmer-Lemeshow Test (p-value)	The Hosmer-Lemeshow test is a statistical test for goodness-of-fit for classification models. The test assesses whether the observed event rates match the expected event rates in pools. Models for which expected and observed event rates in pools are similar are well calibrated. The p-value of this test is a measure of the accuracy of the estimated default probabilities. The closer the p-value is to zero, the poorer the calibration of the model.	Yes	No
Mean Absolute Deviation (MAD)	MAD is the distance between the account level estimated and the actual loss LGD, averaged at the pool level.	No	Yes
Mean Absolute Percent Error (MAPE)	MAPE is the absolute value of the account-level difference between the estimated and the actual LGD, divided by the estimated LGD, and averaged at the pool level.	No	Yes
Mean Squared Error (MSE)	MSE is the squared distance between the account level estimated and actual LGD, averaged at the pool level.	No	Yes



Measure	Description	PD Report	LGD Report
Normal Test	The Normal Test compares the normalized difference of predicted and actual default rates per pool with two limits estimated over multiple observation periods. This test measures the pool stability over time. If a majority of the pools lie in the rejection region, to the right of the limits, then the pooling strategy should be revisited.	Yes	No
Observed versus Estimated Index	The observed versus estimated index is a measure of closeness of the observed and estimated default rates. It measures the model's ability to predict default rates. The closer the index is to zero, the better the model performs in predicting default rates.	Yes	No
Traffic Lights Test	The Traffic Lights Test evaluates whether the PD of a pool is underestimated, but unlike the binomial test, it does not assume that cross-pool performance is statistically independent. If the number of default accounts per pool exceeds either the low limit (Traffic Lights Test at 0.95 confidence) or high limit (Traffic Lights Test at 0.99 confidence), the test suggests the model is poorly calibrated.	Yes	No



*Appendix 10*

# Report and Performance Monitoring Examples

---

<b>Dashboard Report Examples</b> .....	<b>535</b>
KPI Dashboard Report .....	535
KPI Detail Report .....	537
KPI Trend Report .....	538
Monitoring Report .....	540
<b>Model Retrain Comparison Report Example</b> .....	<b>544</b>
Lift Charts .....	544
ROC Charts .....	548
KS Charts .....	549
<b>Monitoring Performance of a Model without Score Code</b> .....	<b>551</b>

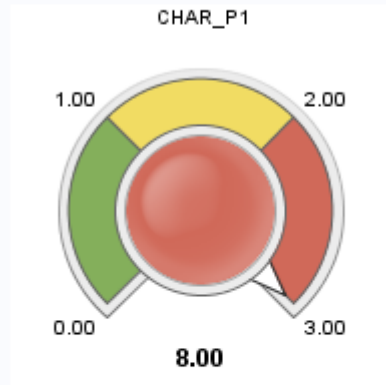
---

## Dashboard Report Examples

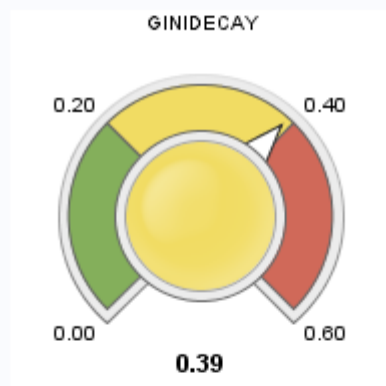
### *KPI Dashboard Report*

Here is an example of the KPI Dashboard Report for the HMEQ project:

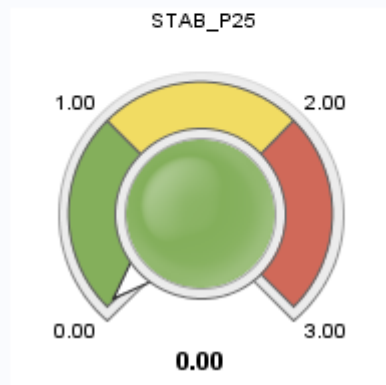
## Characteristic



## Model Assessment












## Stability



**KPI Detail Report**

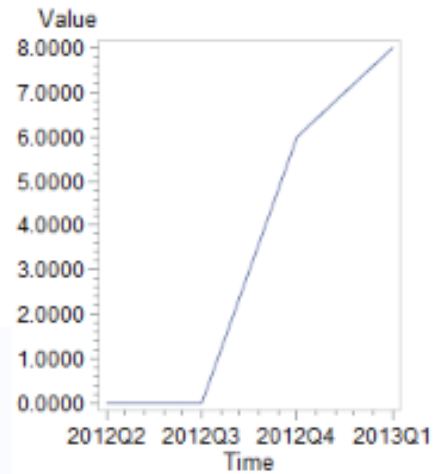
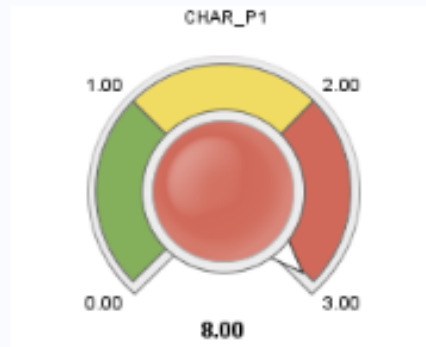
Here is an example of the KPI Detail Report for the HMEQ project:

KPI Detail Report 2013Q1					
Category	Category Status	Category Indicator	Indicator	Indicator Status	Value
Characteristic			Number of predictors with deviation index exceeding 0.1		8.0000
Model Assessment			Gini index decay		0.3929
Stability			Number of outputs with deviation index exceeding 0.25		0.0000

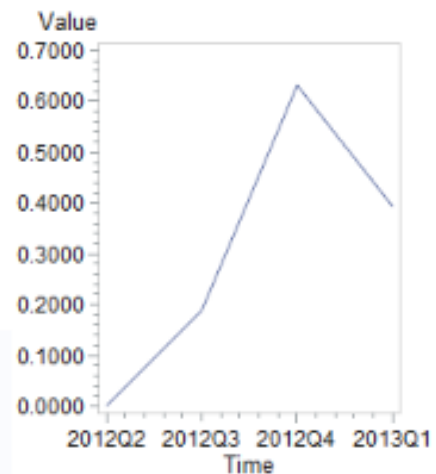
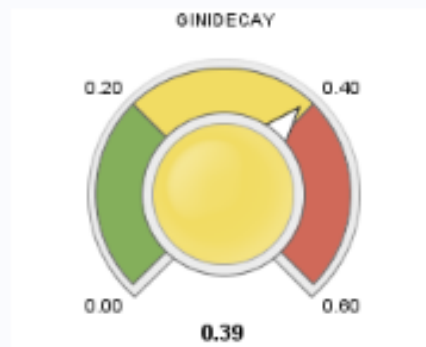
### ***KPI Trend Report***

Here is an example of the KPI Trend Report for the HMEQ project.

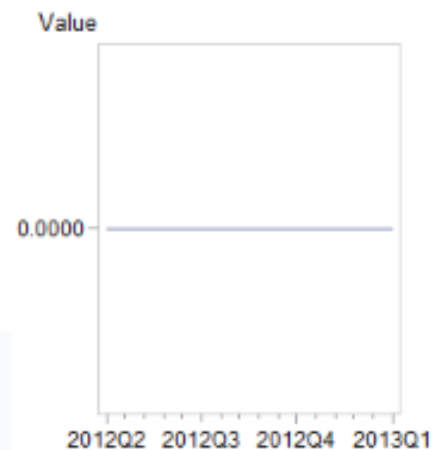
## Characteristic



## Model Assessment

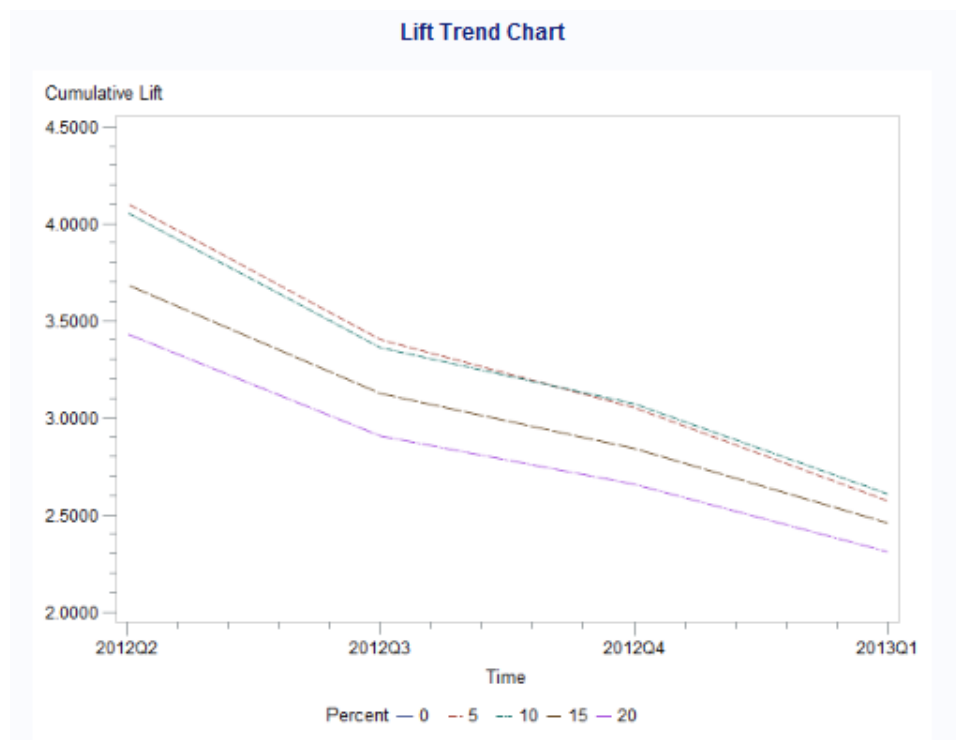
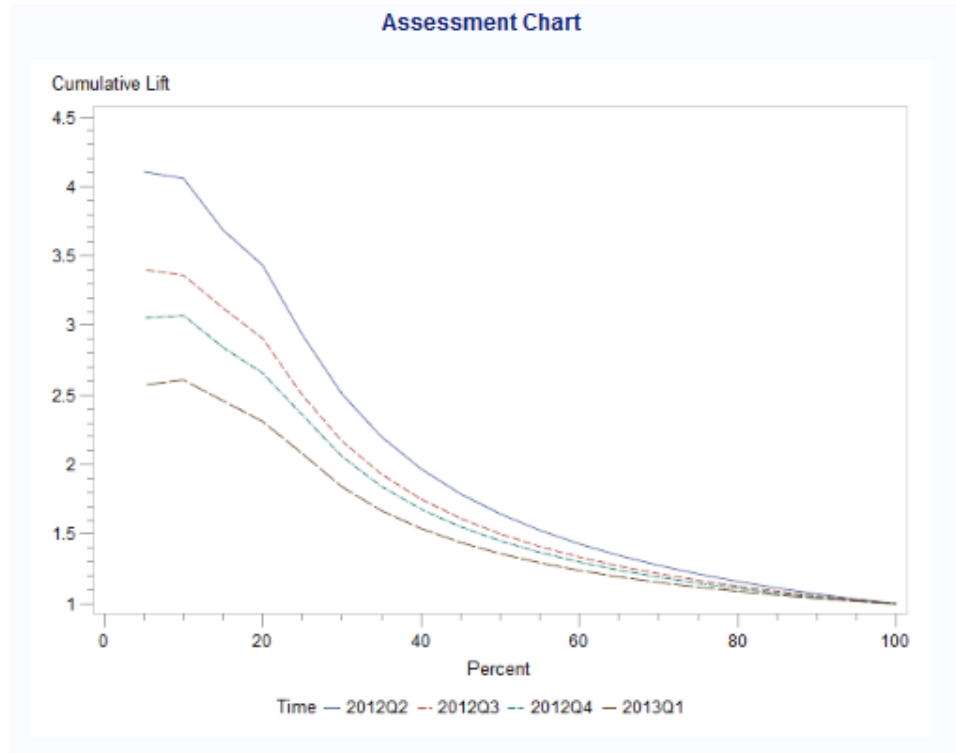


## Stability

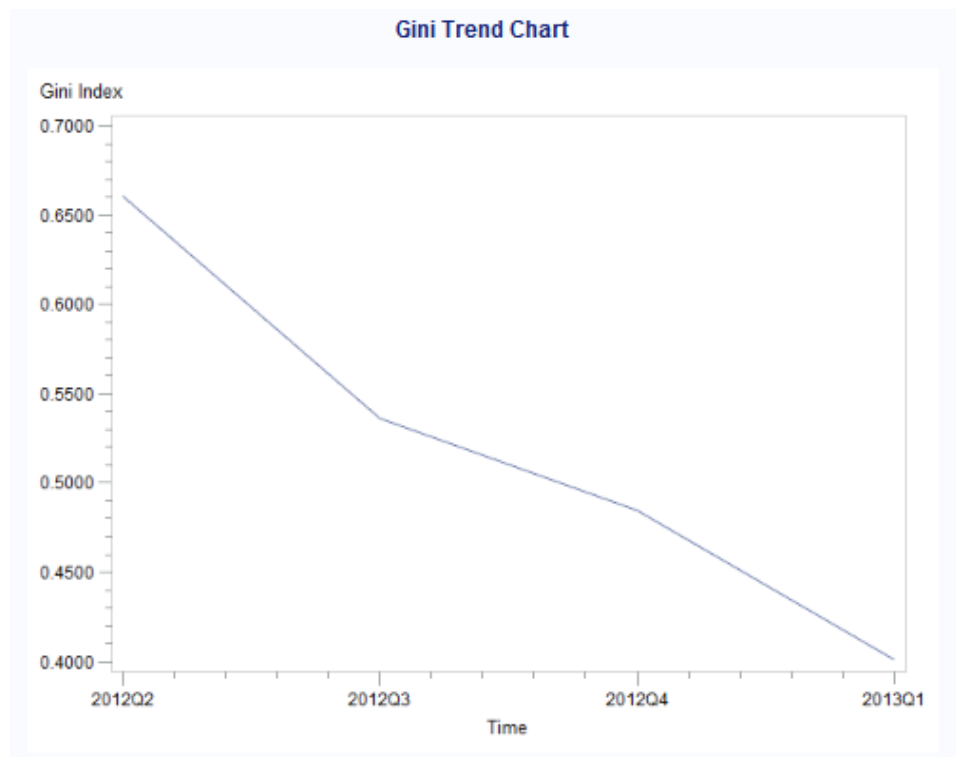
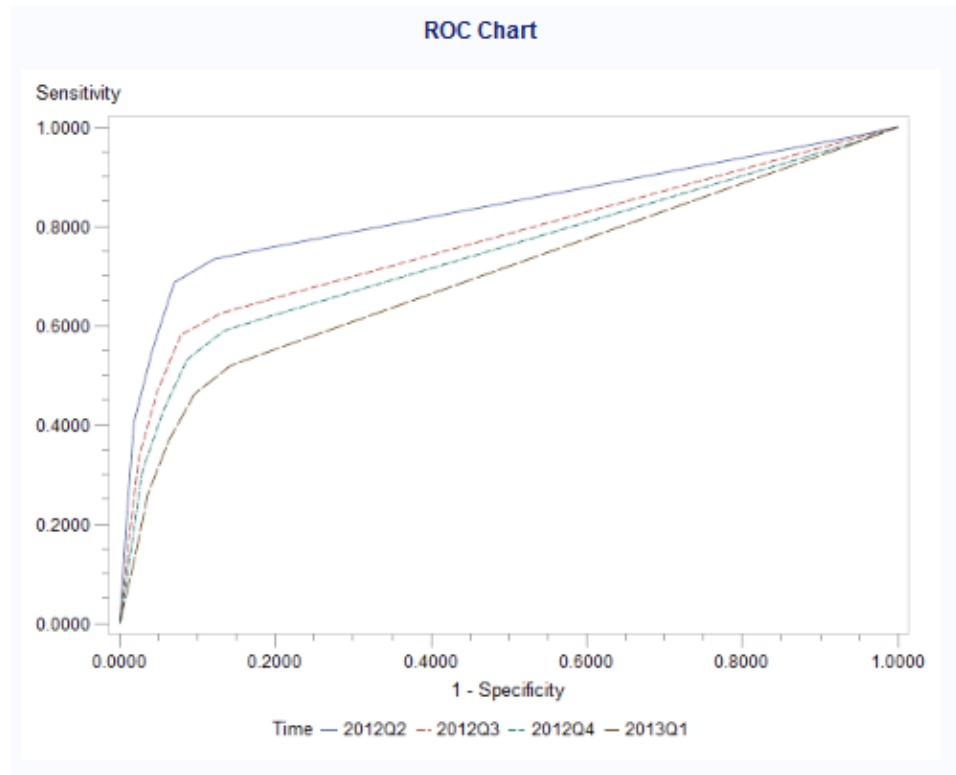


### Monitoring Report

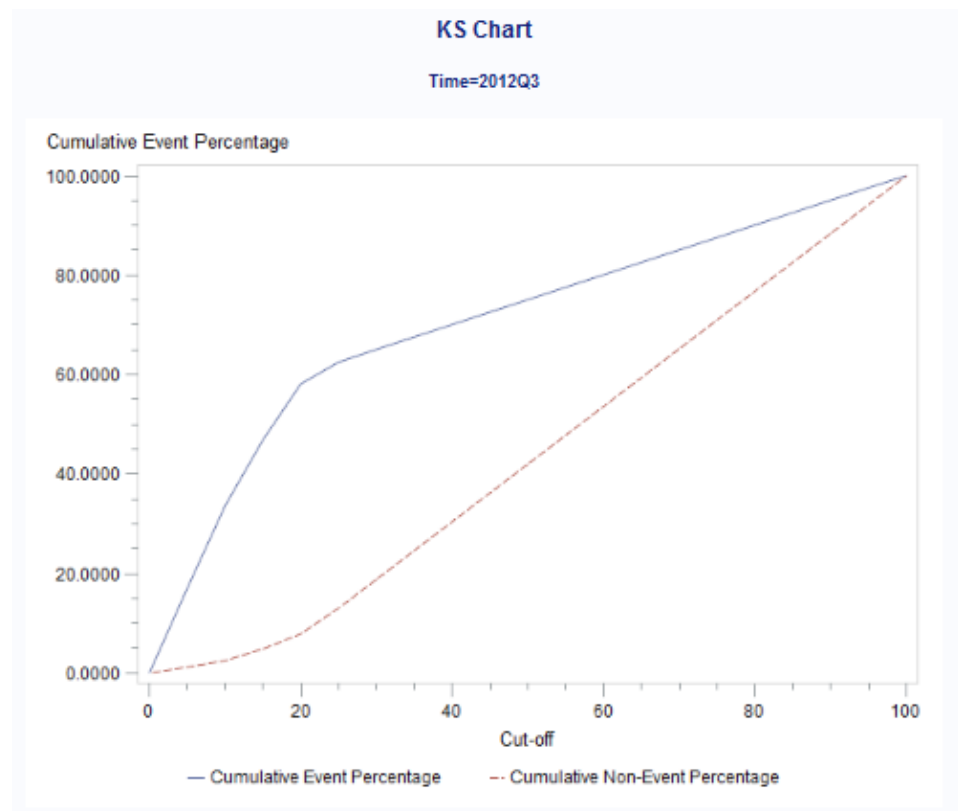
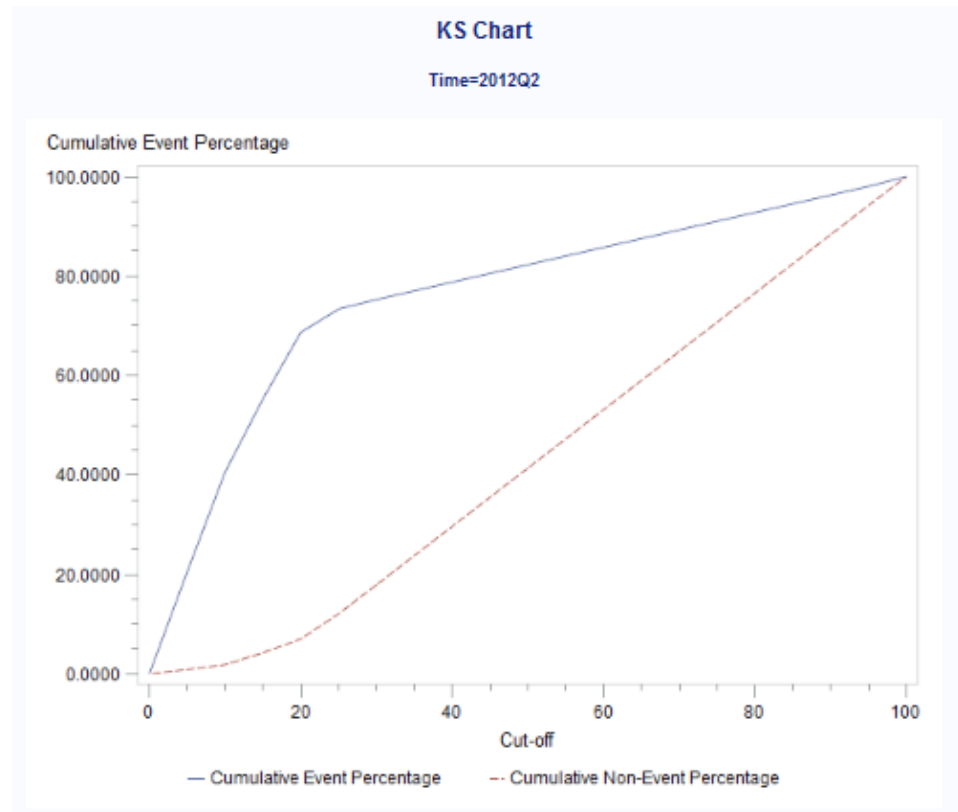
Here are several examples of the graphs that are displayed in the Monitoring Report for the HMEQ project.







A KS Chart for each time frame is displayed on the Monitoring Report. Here are examples for the second and third quarters of 2012.



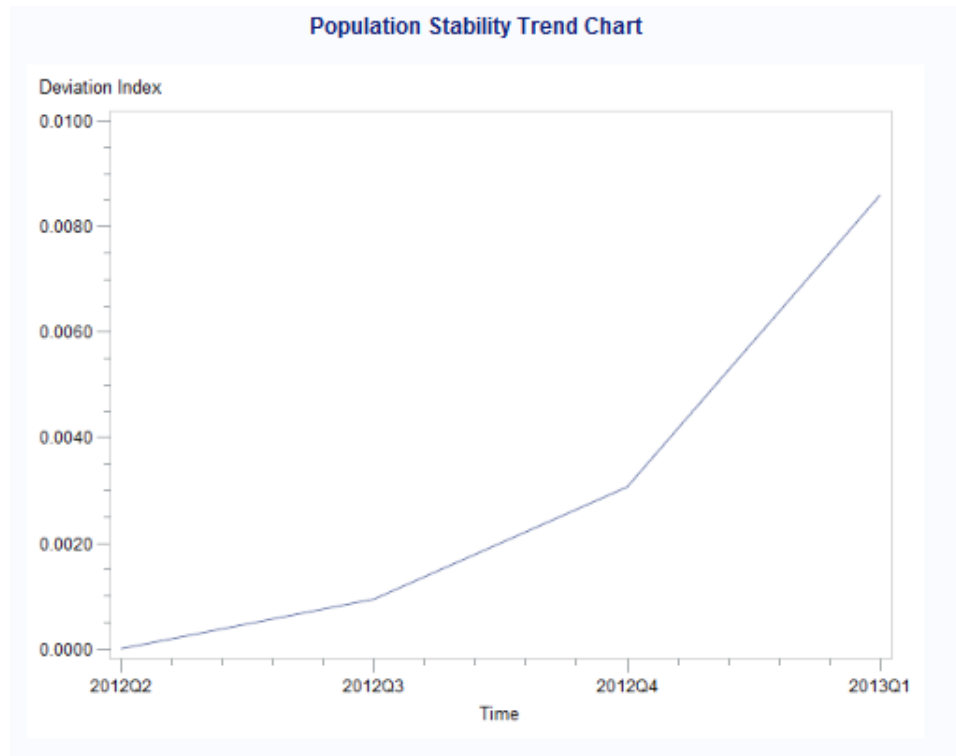
The following is a partial output of a Characteristic Deviation Index Table:

### Characteristic Deviation Index

Time	Variable Name	Deviation Index
2012Q2	CLAGE	0.0000
2012Q2	CLNO	0.0000
2012Q2	DEBTINC	0.0000
2012Q2	DELINQ	0.0000
2012Q2	DEROG	0.0000
2012Q2	JOB	0.0000
2012Q2	LOAN	0.0000
2012Q2	MORTDUE	0.0000
2012Q2	NINQ	0.0000
2012Q2	REASON	0.0000
2012Q2	VALUE	0.0000
2012Q2	YOJ	0.0000

### Stability Deviation Index

Time	Variable Name	Deviation Index
2012Q2	score	0.0000
2012Q3	score	0.0009
2012Q4	score	0.0031
2013Q1	score	0.0086

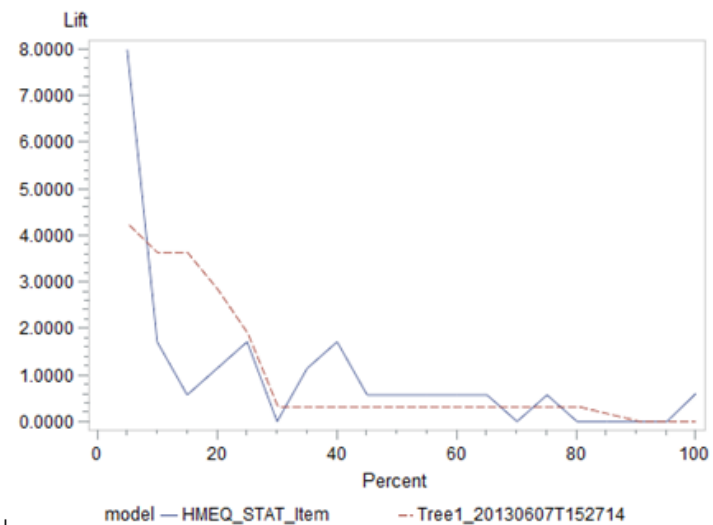


## Model Retrain Comparison Report Example

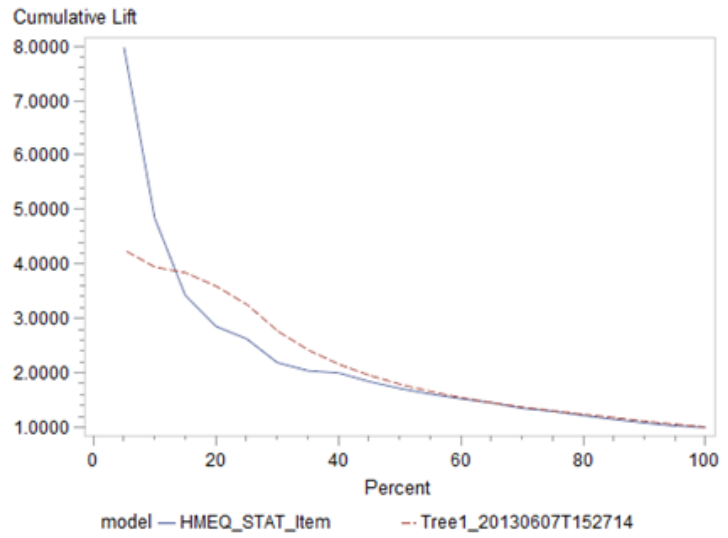
Here is an example of the model retrain comparison report for the retrained model Tree1\_20130607T152714 and the HMEQ\_STAT\_item model.

### Lift Charts

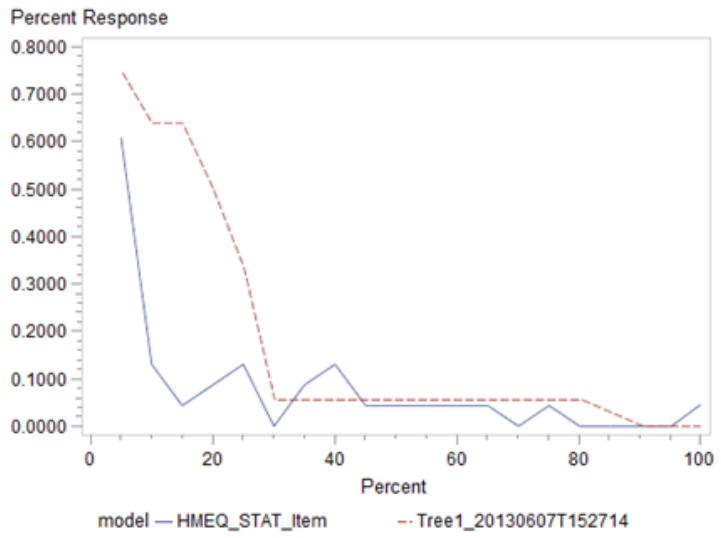
Lift Chart



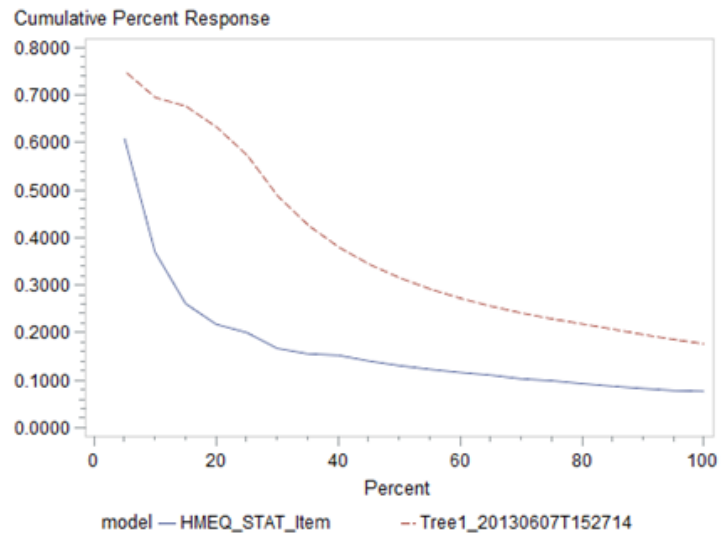
Lift Chart



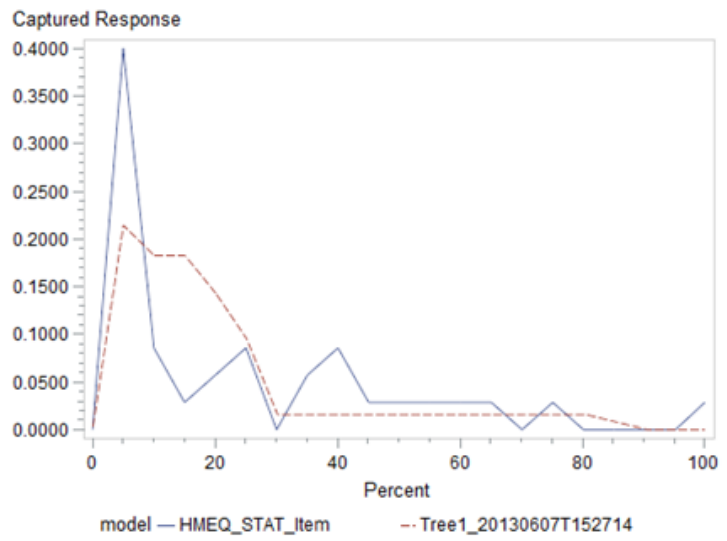
Lift Chart



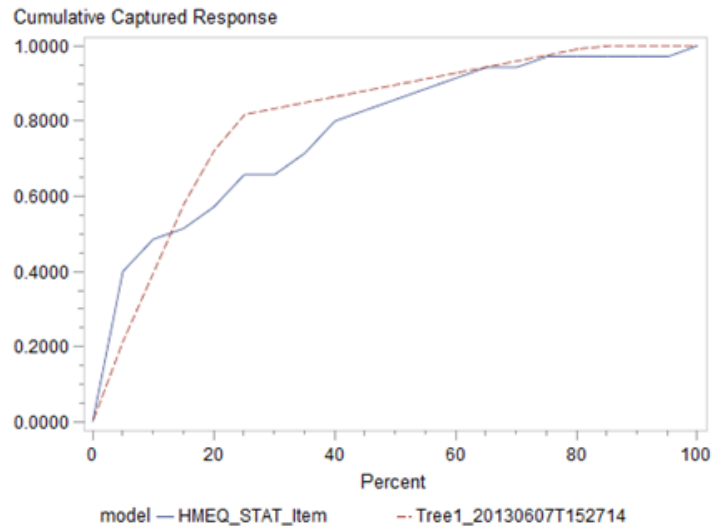
Lift Chart



Lift Chart



Lift Chart

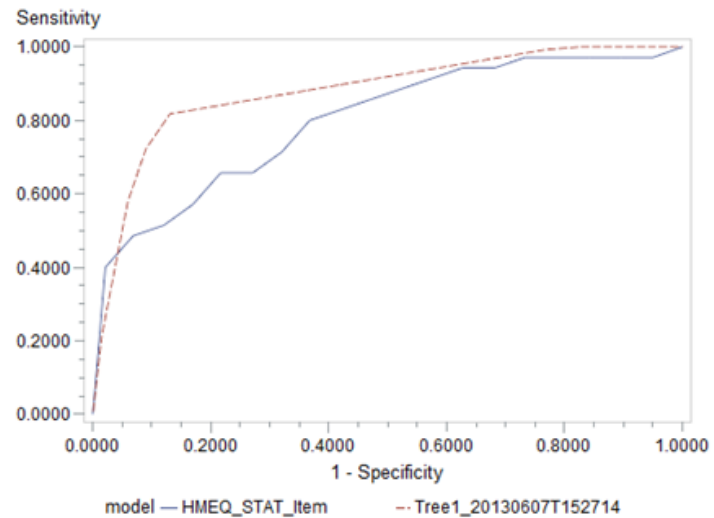


Lift Table

Obs	model	Event Count	Count	Percent	Cumulative Event Count	Cumulative Count	Lift	Cumulative Lift
1	Tree1_20130607T152714	0.0000	0	0	0.0000	0	.	.
2	Tree1_20130607T152714	27.0000	36	5	27.0000	36	4.2560	4.2560
3	Tree1_20130607T152714	23.0000	36	10	50.0000	72	3.6254	3.9407
4	Tree1_20130607T152714	23.0000	36	15	73.0000	108	3.6254	3.8356
5	Tree1_20130607T152714	18.0000	36	20	91.0000	144	2.8373	3.5860
6	Tree1_20130607T152714	12.0000	36	25	103.0000	180	1.8915	3.2471
7	Tree1_20130607T152714	2.0000	36	30	105.0000	216	0.3153	2.7585
8	Tree1_20130607T152714	2.0000	36	35	107.0000	252	0.3153	2.4095
9	Tree1_20130607T152714	2.0000	36	40	109.0000	288	0.3153	2.1477
10	Tree1_20130607T152714	2.0000	36	45	111.0000	324	0.3153	1.9441
11	Tree1_20130607T152714	2.0000	36	50	113.0000	360	0.3153	1.7812
12	Tree1_20130607T152714	2.0000	36	55	115.0000	396	0.3153	1.6479
13	Tree1_20130607T152714	2.0000	36	60	117.0000	432	0.3153	1.5369
14	Tree1_20130607T152714	2.0000	36	65	119.0000	468	0.3153	1.4429
15	Tree1_20130607T152714	2.0000	36	70	121.0000	504	0.3153	1.3624
16	Tree1_20130607T152714	2.0000	36	76	123.0000	540	0.3153	1.2925
17	Tree1_20130607T152714	2.0000	36	81	125.0000	576	0.3153	1.2315
18	Tree1_20130607T152714	1.0000	36	86	126.0000	612	0.1576	1.1683
19	Tree1_20130607T152714	0.0000	36	91	126.0000	648	0.0000	1.1034
20	Tree1_20130607T152714	0.0000	36	96	126.0000	684	0.0000	1.0453
21	Tree1_20130607T152714	0.0000	31	100	126.0000	715	0.0000	1.0000
22	HMEQ_STAT_Item	0.0000	0	0	0.0000	0	.	.
23	HMEQ_STAT_Item	14.0000	23	5	14.0000	23	7.9826	7.9826

## ROC Charts

ROC Chart



ROC Table

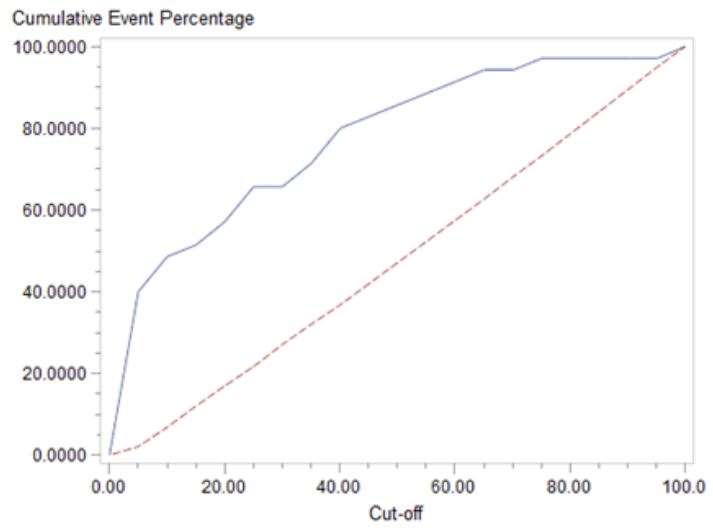
Obs	model	Sensitivity	1 - Specificity	Gini Index
1	Tree1_20130607T152714	0.0000	0.0000	0.7579
2	Tree1_20130607T152714	0.2143	0.0153	0.7579
3	Tree1_20130607T152714	0.3968	0.0374	0.7579
4	Tree1_20130607T152714	0.5794	0.0594	0.7579
5	Tree1_20130607T152714	0.7222	0.0900	0.7579
6	Tree1_20130607T152714	0.8175	0.1307	0.7579
7	Tree1_20130607T152714	0.8333	0.1885	0.7579
8	Tree1_20130607T152714	0.8492	0.2462	0.7579
9	Tree1_20130607T152714	0.8651	0.3039	0.7579
10	Tree1_20130607T152714	0.8810	0.3616	0.7579
11	Tree1_20130607T152714	0.8968	0.4194	0.7579
12	Tree1_20130607T152714	0.9127	0.4771	0.7579
13	Tree1_20130607T152714	0.9286	0.5348	0.7579
14	Tree1_20130607T152714	0.9444	0.5925	0.7579
15	Tree1_20130607T152714	0.9603	0.6503	0.7579
16	Tree1_20130607T152714	0.9762	0.7080	0.7579
17	Tree1_20130607T152714	0.9921	0.7657	0.7579
18	Tree1_20130607T152714	1.0000	0.8251	0.7579
19	Tree1_20130607T152714	1.0000	0.8862	0.7579
20	Tree1_20130607T152714	1.0000	0.9474	0.7579
21	Tree1_20130607T152714	1.0000	1.0000	0.7579
22	HMEQ_STAT_Item	0.0000	0.0000	0.5975
23	HMEQ_STAT_Item	0.4000	0.0212	0.5975
24	HMEQ_STAT_Item	0.4000	0.0212	0.5975



## KS Charts

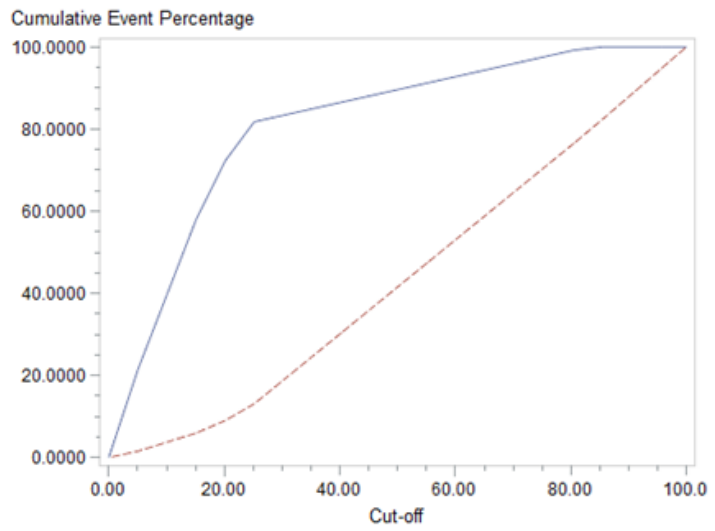
### KS Chart

model=HMEQ\_STAT\_Item



### KS Chart

model=Tree1\_20130607T152714



## KS Table

Obs	model	Maximum Cut-off	KS Statistic	Cut-off	Cumulative Event Percentage	Cumulative Non-Event Percentage
1	HMEQ_STAT_Item	0.11	0.4402	0.00	0.0000	0.0000
2	HMEQ_STAT_Item	0.11	0.4402	5.01	40.0000	2.1226
3	HMEQ_STAT_Item	0.11	0.4402	10.02	48.5714	6.8396
4	HMEQ_STAT_Item	0.11	0.4402	15.03	51.4286	12.0283
5	HMEQ_STAT_Item	0.11	0.4402	20.04	57.1429	16.9811
6	HMEQ_STAT_Item	0.11	0.4402	25.05	65.7143	21.6981
7	HMEQ_STAT_Item	0.11	0.4402	30.07	65.7143	27.1226
8	HMEQ_STAT_Item	0.11	0.4402	35.08	71.4286	32.0755
9	HMEQ_STAT_Item	0.11	0.4402	40.09	80.0000	36.7925
10	HMEQ_STAT_Item	0.11	0.4402	45.10	82.8571	41.9811
11	HMEQ_STAT_Item	0.11	0.4402	50.11	85.7143	47.1698
12	HMEQ_STAT_Item	0.11	0.4402	55.12	88.5714	52.3585
13	HMEQ_STAT_Item	0.11	0.4402	60.13	91.4286	57.5472
14	HMEQ_STAT_Item	0.11	0.4402	65.14	94.2857	62.7358
15	HMEQ_STAT_Item	0.11	0.4402	70.15	94.2857	68.1604
16	HMEQ_STAT_Item	0.11	0.4402	75.16	97.1429	73.3491
17	HMEQ_STAT_Item	0.11	0.4402	80.17	97.1429	78.7736
18	HMEQ_STAT_Item	0.11	0.4402	85.19	97.1429	84.1981
19	HMEQ_STAT_Item	0.11	0.4402	90.20	97.1429	89.6226
20	HMEQ_STAT_Item	0.11	0.4402	95.21	97.1429	95.0472
21	HMEQ_STAT_Item	0.11	0.4402	100.0	100.0000	100.0000
22	Tree1_20130607T152714	0.25	0.6867	0.00	0.0000	0.0000
23	Tree1_20130607T152714	0.25	0.6867	5.03	21.4286	1.5280

---

## 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. 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:

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** or **Prediction**, and create a version. You can skip this step if you have already created a project and version.

3. If it is a project that has a model function property value of **Classification**, 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**
- Output Event Probability Variable (for example, **score**)

If it is a project that has a model function property value of **Prediction**, verify that the following project properties are set:

- Training Target Variable (for example, *lgd*)
- Class Target Level as **Interval**
- Output Prediction Variable (for example, **p\_lgd**)

4. In the Project Tree, navigate to the project's version.

**MMRoot** ⇒ *organizational folder* ⇒ *project folder* ⇒ *version folder*

5. Right-click **Models** 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 144](#).

6. Navigate to the folder on your computer that contains the component files for your model.
7. Select a classification or prediction 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 133](#) or [“Model Template Properties” on page 152](#).

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. Click the list in the **Models Variables** column and select the model output variable. Click **OK**.
13. Right-click the champion model and select **Set as Champion**. For more information, see [“Ensure That the Champion Model Is Set or That the Challenger Model Is Flagged” on page 266](#).
14. Before defining a performance task, verify that the performance data set is registered in SAS Management Console or that a libref has been defined for the performance data set library using the Edit Start-up Code window. Make sure that the data set 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. Also, be sure to clear the **Run model score code** option for the **Data Processing Method** section of the **Define Performance Task** wizard. For more information, see [“Run the Define Performance Task Wizard” on page 268](#).

# Glossary

---

**activity**

See task

**activity status**

See task status

**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 is 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.

**characteristic report**

a report that detects and quantifies shifts in the distribution of input variables over time in data that is used to create predictive models.

**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 file**

a file that defines 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 object**

an object that holds the business data that is required to execute workflow tasks.

**data set**

See SAS data set

**data source**

a table, view, or file from which you will extract information. Sources can be in any format that SAS can access, on any supported hardware platform. The metadata for a source is typically an input to a job.

**DATA step**

in a SAS program, a group of statements that begins with a DATA statement and that ends with either a RUN statement, another DATA statement, a PROC statement, or the end of the job. The DATA step enables you to read raw data or other SAS data sets and to create SAS data sets.

**DATA step fragment**

a block of SAS code that does not begin with a DATA statement. In SAS Model Manager, all SAS Enterprise Miner models use DATA step fragments in their score code.

**delta report**

a report that compares the input and output variable attributes for each of the variables that are used to score two candidate models.

**dynamic lift report**

a graphical report that plots the sequential lift performance of one or more models over time, against test data.

**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.

**holdout data**

a portion of the historical data that is set aside during model development. Holdout 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 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 input variable report**

reports the frequencies that input variables are used in the models for an organizational folder, a project, or a version.

**model profile report**

reports the profile data that is associated with the model input variables, output variables, and target variables.

**model scoring**

the process of applying a model to new data in order to compute outputs.

**model target variable report**

a report that indicates the frequency in which target variables are used in the models that exist in the selected folder.

**neural network**

any of a class of models that usually consist of a large number of neurons, interconnected in complex ways and organized into layers. Examples are flexible nonlinear regression models, discriminant models, data reduction models, and nonlinear dynamic systems.

**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**

a user, group, or role that is assigned to a task. These users, groups, and roles are defined in SAS metadata and are mapped to standard roles for the workflow.

**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. It enables the sharing and deployment of data mining results between applications and across data management systems. Short form: PMML.

**process flow diagram**

a graphical sequence of interconnected symbols that represent an ordered set of steps or tasks that, when combined, form a workflow designed to yield an analytical result.

**production models aging report**

reports the number and the aging distribution of champion models.

**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 used in signal detection theory to plot the sensitivity, or true positive rate, against the false positive rate ( $1 - \text{specificity}$ , or  $1 - \text{true negative rate}$ ) of binary data values. An 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 format**

a type of SAS language element that is used to write or display data values according to the data type: numeric, character, date, time, or timestamp. Short form: format.

**SAS informat**

a type of SAS language element that is used to read data values according to the data's type: numeric, character, date, time, or timestamp. Short form: informat.

**SAS Metadata Repository**

a container for metadata that is managed by the SAS Metadata Server.

**SAS Model Manager repository**

a location in the SAS Content Server where SAS Model Manager data is stored, organized, and maintained.

**SAS package file**

a container for data that has been generated or collected for delivery to consumers by the SAS Publishing Framework. Packages can contain SAS files, binary files, HTML files, URLs, text files, viewer files, and metadata.

**SAS publication channel**

See publication channel

**SAS variable**

a column in a SAS data set or in a SAS data view. The data values for each variable describe a single characteristic for all observations (rows).

**scoring**

See model scoring

**scoring function**

a user-defined function that is created by the SAS Scoring Accelerator from a scoring model and that is deployed inside the database.

**scoring task**

a workflow that executes a model's score code.

**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

**stability report**

a graphical report that detects and quantifies shifts in the distribution of output variables over time in data that is produced by a model.

**swimlane**

a workflow diagram element that enables you to group tasks that are assigned to the same participant.

**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.

**task**

a workflow element that associates executable logic with an event such as a status change or timer event.

**task status**

the outcome of a task in a workflow. The status of a task (for example, Started, Canceled, Accepted) is typically used to trigger the next task.

**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.

**user-defined report**

a customized report. The customized report is a SAS program and its auxiliary files and is stored on the workspace server that is used by SAS Model manager. You access a user defined report by using the New Reports wizard.

**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 series of tasks, together with the participants and the logic that is required to execute the tasks. A workflow includes policies, status values, and data objects.

**workflow definition**

a workflow template that has been uploaded to the server and activated. Workflow definitions are used by the SAS Workflow Engine to create new workflow instances.

**workflow instance**

a workflow that is running in the SAS Workflow Engine. After a workflow template is uploaded to the server and activated, client applications can use the template to create and run a new copy of the workflow definition. Each new copy is a workflow instance.

**workflow template**

a model of a workflow that has been saved to an XML file.

# Index

---

## Special Characters

%AA\_Model\_Register macro 454  
 %mdlmgr\_AddFolder macro 464  
 %mdlmgr\_AddProject macro 465  
 %mdlmgr\_AddVersion macro 467  
 %mdlmgr\_SetProperty macro 468  
 %MM\_AddModelFile macro 407  
 %MM\_CreateModelDataset macro 437  
 %MM\_GetModel macro 295  
 %MM\_GetModelFile macro 410  
 %MM\_GetModels macro 294, 295  
 %MM\_GetURL macro 414  
 %MM\_RegisterByFolder macro 432  
 %MM\_RunReports macro 300  
     macro variables used by 299

## A

access macros  
     %MM\_AddModelFile 407  
     %MM\_CreateModelDataset 437  
     %MM\_GetModelFile 410  
     %MM\_GetURL 414  
     %MM\_RegisterByFolder 432  
 accessing 404  
 global macro variables and 402  
 identifying files used by 405  
 identifying model repository objects  
     404  
     required global macro variables 402  
     required tables 405  
 ad hoc reports 199, 201  
     compared with user-defined reports 200  
     creating 202  
     example 202  
 add a Project Tree node  
     folder 464  
     project 465  
     version 467  
 aggregate report  
     create 340

    delete 342  
     edit 341  
 aggregate reports 339  
     view 341  
 alert notifications 259  
 analytical model  
     query 397  
     specifying for model import 152  
     template 132  
 Approvers  
     groups as, for life cycle templates 79  
     life cycle template participants 79  
     Model Manager Example Life Cycle  
         Approvers 20  
 assessing models  
     tasks by user groups 23  
 Assessment Charts 305  
 Assignees  
     groups as, for life cycle templates 79  
     life cycle template participants 79  
     Model Manager Example Life Cycle  
         Assignees 20  
 associating documents 49  
 attaching documents 51

## B

Basel II reports 191  
 batch performance reports 279  
     accessing performance data set 300  
     copying example batch programs 281  
     creating folder structure for 279  
     defining report folders and data sets  
         298  
     defining specifications 283  
     e-mail recipient specifications 286  
     encoding passwords 300  
     example code 302  
     export channel for 281  
     extracting champion model from a  
         channel 294

- job scheduling specifications 290
  - librefs for running 298
  - performance data for 281
  - prerequisites for running 279
  - project specifications 284
  - publishing champion model from
    - project folder 279
  - report output in production mode 283
  - report output in test mode 282
  - report specifications 287, 291
  - SAS code for running 297
  - user ID and password for 282
  - browse templates 140
  - Browse Templates window 80, 133
- C**
- category views 10
    - Data Sources 13
    - Life Cycle 12
    - Projects 10
  - challenger models 219
  - Champion and Challenger report
    - output files 310
  - champion models 216
    - clearing 218
    - deploying 216
    - extracting from a channel 294
    - monitoring performance tasks by user
      - groups 25
    - monitoring process 260
    - performance monitoring 251
    - publishing 229
    - publishing for batch performance
      - reports 279
    - replacing/retiring 247
    - requirements for 217
    - setting 217
    - setting status 66
  - channels
    - export channel for batch performance
      - reports 281
    - extracting champion model from 294
    - publishing models to 224
  - Characteristic reports 252, 254, 255
    - example 256
    - overview 254
    - performance index warnings and alerts
      - 259
  - classification model
    - query 397
    - specifying for model import 152
    - template 132
  - clustering model
    - query 397
    - specifying for model import 152
  - columns
    - managing 349
  - comparison reports
    - See [model comparison reports](#)
  - component files
    - model templates 133
  - content files, scoring task 175
  - create aggregate report 340
  - current.sas7bdat data set 296
- D**
- dashboard reports 313
    - define 314
    - delete report definition 325
    - edit indicators 324
    - generate 319
    - manage 325
    - viewing 320
  - Data Composition reports 252, 254
    - Characteristic report 252, 254, 255
    - Stability report 252, 254, 255
  - data sets 6
    - containing model information 437
    - current.sas7bdat 296
    - performance data sets 264, 300
    - performance tables 36
    - project control tables 33
    - project input tables 34
    - project output tables 34
    - scoring task input tables 35
  - data source tables
    - local or network drive 42
  - data sources 31
    - performance table 40
    - project input table 37
    - project output table 37
    - registering 32
    - scoring task input tables 39
    - scoring task output tables 39
    - tables 4, 32
    - test table 40
  - Data Sources category view 10, 13
  - DATA step
    - accessing performance data set 300
  - database
    - prerequisites for publishing 234
  - Define Performance Task wizard
    - naming performance tables for use with
      - 41
    - prerequisites for running 266
    - running 268
  - degradation of models 253
  - delete aggregate report 342
  - delete dashboard report definition 325
  - delete schedule 172

- delivering models
    - tasks by user groups 24
  - Delta reports 184
  - deploying models 215
    - champion models 216
    - freezing models 220
    - tasks by user groups 24
  - documents
    - associating with folders 49
    - attaching to folders 51
    - saving 51
    - showing versions 51
    - subfolder 49
    - viewing 51
  - Documents folder
    - associating documents with 49
  - Dynamic Lift reports 186
    - creating 188
    - creating test tables 40
    - verifying model properties 187
    - verifying project properties 187
- E**
- edit aggregate report 341
  - Edit menu 18
  - edit schedule 170
  - encoding passwords 300
  - environment, operational 5
  - export channel
    - for batch performance reports 281
  - extracting
    - champion model from a channel 294
    - published models 228
- F**
- File menu 18
  - folders
    - associating documents with 49
    - attaching documents to 51
    - creating organizational folders 48
    - creating project control group 103
    - project folder organization 56
    - project folder tasks 57
    - structure for batch performance reports 279
    - version folder organization 72
    - version folder tasks 73
  - freezing
    - models 220
    - versions 221
- G**
- general properties 503
  - general tasks 26
  - Gini plots 257
  - Gini Trend Chart 306, 308
  - global macro variables 402
  - graphing scoring task results 172
  - groups 4, 20
- H**
- Help menu 19
- I**
- importing models 125
    - from metadata repository 127
    - mapping model variables to project variables 146
    - model templates 131
    - package files from SAS Enterprise Miner 128
    - partial models 144
    - PMML models 143
    - R model 142
    - SAS code models 130, 140
    - setting model properties 145
    - tasks by user groups 23
    - user-defined model templates 148
  - In-Database Scoring 231
  - indexes
    - warnings and alerts 259
  - input data variable distribution shifts 255
  - input variables 61, 64
  - Interval Target Variable report 189
- J**
- job scheduling specifications
    - batch performance reports 290
- K**
- Kolmogorov-Smirnov (KS) plots 257
  - KS Chart 306, 308
  - KS reports 257
  - KS Trend Chart 306, 308
- L**
- libref 42
  - librefs
    - %MM\_RunReports macro 300
    - access batch performance data sets 300
    - batch performance reports 283
    - R model 520
    - running batch performance reports 298

- SAS Model Manager access macros
    - 405
    - user-defined report 207
  - Life Cycle category view 10
  - life cycle templates 75
    - accessing 80
    - adding milestones 83
    - creating 75, 82
    - creating from a sample 81
    - deleting 80
    - groups as Assignees and Approvers 79
    - milestone properties 86
    - modifying 84
    - participants 78
    - properties 86
    - SAS Model Manager Template Editor
      - window 76
    - saving 80
    - selecting participants 79
    - task properties 87
    - template properties 86
    - viewing 94
  - life cycles 92
    - definition 92
    - milestone organization 92
    - properties 96
    - searching for tasks assigned to users
      - 398
    - tasks 93
    - updating milestone status 95
  - Life Cycles category view 12
  - life style templates
    - adding tasks 84
  - Lift Trend chart 306, 308
  - Local Files method
    - importing R models 142
    - importing SAS code models 130
  - logs
    - publishing scoring functions 244
    - publishing scoring model files 244
  - Loss Given Default (LGD) report 193
- M**
- macro variables
    - %MM\_RunReports macro 299
    - ad hoc report 202
    - defining for user-defined reports 205
    - description of 0
    - global 207, 402
    - optional for report monitoring 299
    - required for access macros 402
    - to define report folders and data sets
      - 298
  - macros
    - accessing report macros 295, 298
    - accessing Project Tree and property
      - macros 460
    - adding Project Tree nodes 464
    - create property table 461
  - manage
    - workflow 367
  - manage dashboard reports 325
  - managing
    - columns 349
  - mapping variables
    - model variables to project variables 146
    - scoring task output variables 167
  - menus 10, 18
  - metadata
    - project metadata 67
  - metadata repository
    - importing models from 127
    - viewing MiningResult objects in 230
  - metadata tables
    - publishing scoring functions 244
  - milestones
    - adding to life cycle template 82
    - organization of life cycle milestones 92
    - specific properties for 512
    - updating status 95
  - MiningResult objects 228
    - viewing in metadata repository 230
  - Model Assessment reports
    - performance index warnings and alerts
      - 259
  - model comparison reports 180
    - Delta 184
    - Dynamic Lift 186
    - input files 181
    - Model Profile 183
    - output files 182
    - viewing 198
  - model component files 130, 407
    - create SAS Package file 449
    - import partial models 144
    - importing R model 142
    - model templates 131
    - naming for model template 148
    - project planning 59, 100
    - R model 522
    - SAS package file 139
    - specifying for model import 140
    - used by access macros 405
  - model delivery
    - publishing models 228
    - publishing to a database 231
  - model function types 70, 507
  - Model Input Variable Report 254
  - model management process 6
  - Model Manager Example Life Cycle
    - Approvers 20



- Model Manager Example Life Cycle
    - Assignees 20
  - Model Monitoring reports 252, 256
    - KS reports 257
    - monitoring Gini (ROC and Trend) reports 257
    - monitoring Lift reports 256
  - Model Profile reports 183
    - creating 183
  - model repository objects 404
  - model scoring files
    - steps for publishing 239
  - Model Target Variable Report 253
  - model templates 131, 148
    - component files 133
    - creating 148
    - File List properties 153
    - modifying 150
    - properties 152
    - system and user properties 153
    - template properties 152
    - user model templates 133
    - user-defined 148
  - model variables
    - mapping to project variables 146
  - models
    - See also* [champion models](#)
    - assessing tasks by user group 23
    - data sets containing model information 437
    - degradation of 253
    - deploying 215
    - deploying and delivering tasks by user groups 24
    - extracting published models 228
    - freezing 220
    - function types 60
    - importing 125
    - importing tasks by user groups 23
    - managing 3
    - mapping model variables to project variables 146
    - process of managing 6
    - publishing 223, 228
    - publishing to a channel 224
    - registering 432
    - searching for 394
    - specific properties for 154, 514
    - unfreezing 220
    - validating with model comparison reports 180
    - validating with user reports 199
    - verifying publish 230
  - monitoring champion model performance 251
    - tasks by user groups 25
  - monitoring Lift reports 256
  - monitoring performance 117
  - Monitoring reports 305
    - creating 117, 307, 309
    - output files 310
  - monitoring ROC & Gini reports 257
- N**
- naming performance tables 41
- O**
- objects
    - deleting in Project Tree 52
  - opening
    - objects 357
  - operational environment 5
  - organization-specific user-defined properties 508
  - organizational folders 48
  - output data variable distribution shifts 255
  - output variables 61, 64
- P**
- package files 128
    - creating 128
    - importing from SAS Enterprise Miner 128
    - publishing models 224
  - Partial Model Import utility 144
  - partial models
    - importing 144
  - participants
    - selecting for life cycle templates 79
  - passwords
    - encoding 300
    - for batch performance reports 282
  - performance
    - data for batch performance reports 281
    - monitoring champion model performance by user groups 25
  - performance data sets
    - creating performance reports 264
    - DATA step for accessing 300
  - performance indexes
    - warnings and alerts 259
  - performance indicator
    - See* [dashboard reports](#)
  - Performance Monitor
    - schedule properties 275
  - performance monitoring 251
    - Data Composition reports 254
    - delete data sets 276

- performance index warnings and alerts 259
- prerequisites for running Define Performance Task wizard 266
- Summary results 253
- performance monitoring reports
  - See also* batch performance reports
  - Data Composition reports 252
  - formatting 305
  - Model Monitoring reports 252, 256
  - Monitoring reports 305
  - SAS programs for creating 278
  - Summary reports 252
  - types of 252
  - viewing 311
- performance reports 263
  - See also* batch performance reports
  - creating 117
  - creating with performance tasks 263
  - performance data sets and 264
  - running Define Performance Task wizard 268
- performance tables 32, 36, 40
  - creating 40
  - naming for use with Define Performance Task wizard 41
- performance tasks
  - creating reports with 263
  - prerequisites for running Define Performance Task wizard 266
  - running Define Performance Task wizard 268
  - scheduling 273
- planning projects 58
- PMML models
  - importing 143
- prediction model
  - query 397
  - specifying for model import 152
  - template 132
- preferences
  - Workflow Console 355
- Probability of Default report 195
- production mode
  - batch performance reports 283
- Production Models Aging Report 253
- profile data 183
- project control group 99, 103
  - add a new version 113
  - add an input variable 114
  - create projects 104
  - prerequisites for creating 101
  - publishing champion models 115
- project control tables 32, 33
- project folders
  - organization of 56
- publishing champion model from 279
  - tasks 57
- project input tables 32, 34
  - creating 37
- project metadata 67
- project output tables 32, 34
  - creating 38
- project properties 67
  - setting programatically 468
- project tables 33
  - performance tables 36
  - project control tables 33
  - project input tables 34
  - project output tables 34
  - scoring task input tables 35
  - scoring task output tables 35
  - test tables 36
  - train tables 36
- Project Tree 10
  - associating documents with folders 49
  - creating organizational folders 48
  - deleting objects in 52
  - organizing 47
  - project control group 103
- project variables
  - mapping to model variables 146
  - planning 59, 100
- projects 55
  - controlling access to versions 220
  - creating 61
  - input variables 61, 64
  - lock metadata 66
  - model function types 60
  - modify 64
  - output variables 61, 64
  - planning 58
  - prerequisites for creating 60
  - properties 60
  - publishing champion models 229
  - publishing models to a database 231
  - retiring 248
  - setting champion model status 66
  - setup tasks by user groups 22
  - specific properties for 505
  - unlock metadata 66
- Projects category view 10
- properties
  - Basel II 191
  - Dynamic Lift reports 187
  - general 503
  - life cycle properties 96
  - life cycle templates 86
  - Loss Given Default (LGD) 191
  - model function types 70, 507
  - model template properties 152
  - organization-specific user-defined 508

- Probability of Default (PD) 191
  - project properties 67
  - result set properties 177, 517
  - scoring task properties 176, 516
  - scoring task schedule 518
  - setting for imported models 145
  - specific for a model 154, 514
  - specific for a project 505
  - specific for a version 511
  - specific for milestones and tasks 512
  - system properties 504
  - user-defined 508
  - version properties 90
  - prototype tables 32
    - project input tables 34
    - project output tables 34
    - scoring task output tables 35, 39
  - publish model to a database
    - process flow 233
  - Publish Models to a Database window 235
  - publishing
    - champion model, for batch performance reports 279
    - extracting published models 228
    - models 223, 228
    - models to a channel 224
    - models to a database 231
    - project champion models 115, 229
    - remove models from a database 244
    - verifying model publish 230
    - view history 246
- Q**
- Query utility 393
    - searching for models 394
    - searching life cycles for tasks assigned to users 398
    - searching with UUID 396
- R**
- R model 142
  - R models
    - building 520
    - model component files 522
    - model template file 521
    - using in SAS Model Manager 519
  - registering
    - data sources 32
    - models 432
  - relational databases 6
  - report macros
    - accessing 295, 298
  - report templates
    - for user-defined reports 205
  - reports
    - See also performance reports*
    - ad hoc reports 199, 201
    - aggregate 339, 340
    - Basel II 191
    - batch performance reports 279
    - Characteristic reports 252, 254, 255
    - creating with performance tasks 263
    - dashboard 313
    - Data Composition reports 252, 254
    - delete aggregate report 342
    - Delta reports 184
    - Dynamic Lift reports 186
    - Interval Target Variable 189
    - KS reports 257
    - LGD 193
    - Loss Given Default (LGD) 191
    - Loss Given Default prerequisites 192
    - Model Assessment reports 259
    - model comparison reports 180
    - Model Input Variable Report 254
    - Model Monitoring reports 252, 256
    - Model Profile reports 183
    - Model Target Variable Report 253
    - monitoring Lift reports 256
    - Monitoring reports 305
    - monitoring ROC & Gini reports 257
    - performance monitoring 251
    - Probability of Default 195
    - Probability of Default (PD) 191
    - Probability of Default prerequisites 194
    - Production Models Aging Report 253
    - Stability reports 252, 254, 255
    - Summary of Reports 253
    - Summary reports 252
    - train table summary 196
    - user reports 199
    - user-defined reports 199, 204
  - repository
    - accessing files in 410
    - model repository objects 404
  - Resources folder
    - associating documents with 49
  - result set properties 177, 517
  - retiring
    - champion models 247
    - projects 248
  - retrain model 327
    - define task 328
    - execute 333
    - prerequisites 328
    - view comparison report 334
    - view models 334
  - ROC Chart 306, 308
  - ROC plots 257

- roles 20
  - life cycle template participant roles 78
- S**
- SAS code models
  - importing 130, 140
- SAS Content Server 6
- SAS Enterprise Miner
  - importing package files from 128
- SAS Foundation 6
- SAS Management Console 5
- SAS Metadata Repository
  - register SAS/STAT models 454
- SAS Metadata Server 6
- SAS Model Manager
  - interface 9
  - managing models 3
  - model management process 6
  - operational environment 5
  - services provided by 3
  - setup tasks by user groups 21
  - user groups 4
- SAS Model Manager Client 5
- SAS Model Manager Macros 6
- SAS Model Manager Middle Tier Server 6
- SAS Model Manager Template Editor
  - user-defined properties 509
- SAS Model Manager Template Editor window 76
- SAS Model Manager window 9
  - category views 10
  - layout 9
  - toolbar and menus 10
- SAS programs
  - creating performance monitoring reports 278
  - delete from SAS Content Server 208
  - edit on SAS Content Server 208
  - upload to SAS Content Server 205
  - user-defined report 204, 205
- SAS user-defined properties 510
- SAS Web Infrastructure Platform 6
- SAS Workspace Server 6
- SAS/STAT model
  - register using model components 449
- saving documents 51
- schedule
  - performance task 273
  - scoring task 170
- schedule properties 518
- scoring functions
  - log messages for publishing 244
  - metadata tables 244
  - steps for publishing 239
- scoring model files
  - log messages for publishing 244
- scoring models
  - publishing 231
- scoring output tables 162
  - creating 162
- scoring task content files 175
- scoring task input tables 32, 35, 39
  - creating 39
- scoring task output tables 32, 35, 39
  - adding to SAS Model Manager 35
  - creating 39
- scoring tasks 157
  - creating 164
  - creating scoring output tables 162
  - delete schedule 172
  - executing 168
  - generated content files 175
  - graphing results 172
  - mapping output variables 167
  - modifying 167
  - properties 176, 516
  - result set properties 177, 517
  - schedule 170
  - schedule properties 518
  - tabbed views 159
- searches
  - for life cycle tasks assigned to users 398
  - for models 394
  - with UUID 396
- searching
  - managing searches 354
  - retrieving and applying 354
  - save search criteria 352
- segmentation model
  - template 132
- setup
  - of projects and versions by user groups 22
- setup tasks by user groups
  - SAS Model Manager 21
- sorting
  - multiple columns 350
  - single columns 350
- SPK files
  - See package files
- Stability reports 252, 254, 255
  - example 256
  - overview 254
  - performance index warnings and alerts 259
- subfolders
  - associating documents with 49
  - attaching documents to 51
  - creating 50

- Summary of Reports 253
  - Summary reports 252
  - Summary results 253
  - system properties 504
- T**
- tabbed views
    - scoring tasks 159
  - tables
    - local or network drive 42
  - tasks
    - adding to life cycle template 82
    - creating reports using performance tasks 263
    - general, by user groups 26
    - life cycle tasks 93
    - project folder tasks 57
    - scoring task properties 176, 516
    - scoring tasks 157
    - searching for life cycle tasks assigned to users 398
    - specific properties for 512
    - version folder tasks 73
  - templates
    - See also* model templates
    - analytical model 132
    - classification model 132
    - creating from a sample 81
    - life cycle 75
    - prediction model 132
    - report templates 205
    - segmentation model 132
    - user model templates 133
    - user-defined model templates 148
    - viewing 140
  - test mode
    - for batch performance reports 282
  - test tables 32, 36
    - creating 40
  - thresholds
    - warnings and alerts 259
  - toolbar 10, 15
  - train table data sets 196
  - train table summary report 196
  - train tables 32, 36
- U**
- unfreezing
    - models 220
    - versions 221
  - URL 414
  - user groups 4, 20
  - user ID
    - for batch performance reports 282
  - user model templates 133
  - user reports 199
    - ad hoc 199, 201
    - output created by 200
    - user-defined 199, 204
    - validating models with 199
  - user-defined model templates 148
  - user-defined properties 508
    - organization-specific 508
    - SAS 510
  - user-defined reports 199, 204
    - compared with ad hoc reports 200
    - creating 204
    - example 209
    - macro variables 205
    - report template for 205
    - running 208
  - users
    - searching for life cycle tasks assigned to 398
  - UUIDs
    - searching with 396
    - translating to URL 414
- V**
- validating models
    - with model comparison reports 180
    - with user reports 199
  - VALIDVARNAME= system option 186
  - verifying model publish 230
  - version folders
    - organization of 72
    - tasks 73
  - version properties
    - setting programatically 468
  - versions 71
    - controlling access to project versions 220
    - creating 89, 113
    - creating life cycle templates 75
    - freezing 221
    - properties 90
    - SAS Model Manager functionality for 72
    - setup tasks by user groups 22
    - showing document versions 51
    - specific properties for 511
    - unfreezing 221
  - view aggregate report 341
  - View menu 19
  - viewing
    - documents 51
    - life cycle templates 94
    - model comparison reports 198
    - performance monitoring reports 311

**W**

- warning notifications 259
- workflow
  - add attachments 386
  - adding comments 363
  - assign participant 375
  - create 369
  - creating reports 386
  - edit 372
  - edit properties 374
  - importing models 380
  - model management components 379
  - publishing models 384
  - release activity 377
  - remove participant 376
  - replying to comments 364
  - searching comments 365
  - set champion and challenger models 383
  - terminate 378
  - viewing models 381
  - viewing performance 388
  - viewing reports 386
- workflow activity
  - adding comments 363
  - edit properties 361
  - release 377
  - replying to comments 364
  - searching comments 365
- workflow console
  - customizing category views 349
  - searching 352
- Workflow Console 346
  - activities 360
  - alert notifications 356
  - comments 362
  - create workflow 369
  - edit workflow 372
  - manage 367
  - navigation pane 348
  - opening objects 358
  - participants 374
  - rearranging objects 359
  - setting preferences 355
  - tile pane 357
  - user interface layout 346
  - view activities 359
  - view instances 370
  - view workflow definitions 368
  - window controls 348
- workflow participant
  - assign 375
  - remove 376
- workspaces
  - saving 359