## Contents

**Chapter 1 • Introduction** ................................................................. 1  
  Overview ................................................................. 1  
  Prerequisites for Using Macros ........................................... 1  

**Chapter 2 • Model Management Macros** ........................................ 3  
  Overview of Model Management Macros ................................... 3  
  Dictionary ............................................................... 4  

**Chapter 3 • Performance Monitoring Macros** ................................... 31  
  Overview of Performance Monitoring Macros ............................ 31  
  Macro Variables .......................................................... 32  
  Set Alert and Warning Conditions ...................................... 37  
  Dictionary ............................................................... 38  

**Chapter 4 • Publish Destination Macros** ......................................... 45  
  Overview of Publish Destination Macros ................................... 45  
  Dictionary ............................................................... 45  

**Chapter 5 • Feature Contribution Index Macros** ............................... 57  
  Overview of FCI Macros .................................................... 57  
  How the Model Outcome Is Determined ................................... 58  
  Aggregation for Nominal Targets ........................................ 58  
  Dictionary ............................................................... 59  
  Example ................................................................. 64
Chapter 1
Introduction

Overview

SAS Model Manager provides a set of macros that you can use in your SAS programs to manage models that are within projects and folders. These macros are useful for handling many aspects of the model life cycle as well. You can create repositories, add folders, projects, and versions, and you can also import models into the SAS Model Manager common model repository. As part of the model life cycle, you can monitor performance and publish models to CAS, Hadoop, Teradata, or SAS Micro Analytic Service.

Here are the types of macros that are available for use:

- Model management macros
- Performance monitoring macros
- Publish destination macros

Prerequisites for Using Macros

The SAS Workspace Server does not automatically retrieve the authorization token when you call a macro in SAS Studio 4.4. You must specify the authorization token with the %MM_GET_TOKEN macro and the TOKEN= argument to call all macros in SAS Studio 4.4. You do not need to specify the authorization token or argument for SAS Studio 5.1 and later. For more information, see “%MM_GET_TOKEN” on page 4.

Note: Users must also have the appropriate permissions to run the SAS Model Manager macros. For more information, see “Managing Permissions” in SAS Model Manager: Administrator’s Guide.
Chapter 2
Model Management Macros

Overview of Model Management Macros

Use these macros in a SAS program to create repositories, folders, projects, and versions, as well as import models within the SAS Model Manager common model repository. You can import models into a repository, folder, or project that you have permissions to. You can also use these macros to retrieve objects by UUID or delete objects from the common model repository. You can use SAS Studio to run these macros.

Important: When a user’s SAS environment is in the locked-down state with limited file system access, all access validation to the host file system is done through the lockdown path list. In order to run macros such as %MM_IMPORT_MODEL, %MM_IMPORT_ASTORE_MODEL, and %MM_PUBLISH_MODEL that require access to the file system, your system administrator must add access for the HTTP access method to the lockdown whitelist. For more information, see “LOCKDOWN Statement” in SAS Viya Administration: Programming Run-Time Servers.
Note: Make sure that your user ID can be authenticated through the SAS Logon Manager and Identities Service and that you have the appropriate permissions for using these macros. If you cannot generate an authorization token, contact your SAS application administrator. If you are using SAS Studio 4.4, see “Prerequisites for Using Macros” on page 1.

When passing UUIDs in as arguments, it is a good practice to wrap %STR() around the UUID values or macro variables. The reason is that in the SAS language, a stand-alone string (for example, A1B2C3D4-E5F6-A7B8-C9D0-12345678) that contains a hyphen (-) can be interpreted by the parser as a mathematical operation.

Dictionary

%%MM_GET_TOKEN
Generates an authorization token to be used with the SAS Model Manager publish destination and model management macros.

Syntax

%%MM_GET_TOKEN (  
  BASEURL=host-name:<port>,  
  USER=user-ID,  
  PW=password,  
  TOKENNAME=authorization-token-name  
);

Required Arguments

BASEURL=host-name:<port>
specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

USER=user-ID
specifies a user ID that has permission to access database content.

PW=password
specifies the password for the user ID that is entered with the USER= parameter.

TOKENNAME=authorization-token-name
specifies the name of the generated token that is used for a secure connection when executing the macros. The value must be a valid SAS name. This authorization token name can be specified with the TOKEN= argument for the publish destination and model management macros.

Note: You must specify the authorization token name with the %%MM_GET_TOKEN macro and the TOKEN= argument to call all macros in SAS Studio 4.4.
Example

Example Code 1 Set Macro Variables

```sas
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
```

Example Code 2 Get Authorization Token

```sas
%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);
```

%MM_CREATE_REPOSITORY Macro

Adds a repository folder to the common model repository.

**Note:** Only SAS Administrators and other authorized users can create a repository. When a repository is created using this macro, an associated folder is created for it within the Folders service. However, the repository folder exists only in order to store metadata about the repository and should not be used for user content. It is recommended, that you create a subfolder within the repository folder to store content. This can be done using the %MM_CREATE_FOLDER macro.

**Syntax**

```sas
%MM_CREATE_REPOSITORY (  
    REPOSITORYNM =repository-name,  
    SERVERNM =host-name:<port>,  
    <REPOSITORYDESC =repository-description>,  
    <REPOSFOLDERID =repository-folder-ID>,  
    <TOKEN =%authorization-token>,  
    <REPOSID =repository-ID>,  
);
```

**Required Arguments**

- **REPOSITORYNM =repository-name**
  - specifies the name of the repository.

- **SERVERNM=host-name <port>**
  - specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

**Optional Arguments**

- **REPOSITORYDESC =repository-description**
  - specifies the name of the repository.
REPOSFOLDERID = repository-folder-ID-macro-variable
specifies the macro variable to assign to the repository folder ID. A repository folder ID is assigned by the Folders service when this macro is successfully run. If this argument is not included, the default macro variable name that is created is "_reposFolderID".

REPOSID = repository-ID-macro-variable
specifies the macro variable to assign to the ID for the repository. The repository ID is assigned when this macro is successfully run. If this argument is not included, the default macro variable name that is created is "_reposID".

TOKEN=%authorization-token
specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.

See “%MM_GET_TOKEN” on page 4

Example

```sas
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%mm_get_token(
   baseURL=&servernm,
   user=&userID,
   pw=&password,
   tokenname=myTokenName
);

%mm_create_repository(
   repositorynm   = MyRepository,
   repositorydesc = Description of MyRepository,
   reposfolderID  = repFolderID,
   servernm       = &servernm,
   token          = %myTokenName,
   reposID        = repID
);
```

%MM_CREATE_FOLDER Macro

Adds a folder to a repository.

Syntax

```sas
%MM_CREATE_FOLDER (  
   FOLDERNM = folder-name,  
   REPOSFOLDERID = repository-folder-ID,  
   SERVERNM = host-name:<port>,  
   <TOKEN =%authorization-token>,  
   <FOLDERID = folder-ID-macro-variable>,  
);
```
**Required Arguments**

**FOLDERNM** = *folder-name*

specifies the name of the folder.

**REPOSFOLDERID** = *repository-folder-ID*

specifies the folder ID for the repository. This is the UUID assigned by the Folders service.

**SERVERNM** = *host-name* <:<port>

specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

**Optional Arguments**

**TOKEN** = %authorization-token

specifies the authorization token that was generated with the `%MM_GET_TOKEN` macro.

*Note:* You must specify this argument to call all macros in SAS Studio 4.4.

See “%MM_GET_TOKEN” on page 4

**FOLDERID** = *folder-ID-macro-variable*

specifies the macro variable to assign to the folder ID. A folder ID is assigned by the Folders service when this macro is successfully run. If this argument is not included, the default macro variable name that is created is “_folderID”.

**Example**

```sas
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);
%mm_create_folder(
    foldernm       = MyFolder,
    reposfolderID  = %str(&repFolderID),
    servernm       = &servernm,
    token          = %myTokenName,
    folderID       = folderID
);
```

---

### %MM_CREATE_PROJECT Macro

Creates a project within a folder.

#### Syntax

```sas
%MM_CREATE_PROJECT (  ```
PROJECTNM =project-name,
FOLDERID =folder-ID,
<FUNCTION =model-function-name>,
SERVERNM =host-name:<port>,
<PROJECTID =project-ID-macro-variable>,
<PROJECTVERSIONID =project-version-ID-macro-variable>,
<TOKEN =%authorization-token>
);

Required Arguments
PROJECTNM =project-name
  specifies the name of the folder.

FOLDERID =folder-ID
  specifies the ID for the folder where the new project is to be created. This is the
  UUID assigned by the Folders service.

SERVERNM=host-name <:port>
  specifies the URL where SAS Model Manager is running. It includes the host name
  and port for the application server. The default port is 80.

Optional Arguments
PROJECTID =project-ID
  specifies the macro variable to assign to the project ID that is created. A UUID
  assigned by the Model Repository service. If this argument is not included, the
  default macro variable name that is created is "_projectID".

PROJECTVERSIONID =project-version-ID
  specifies the macro variable to assign to the project version ID that is created. A
  UUID assigned by the Model Repository service. If this argument is not included,
  the default macro variable name that is created is "_projectVersionID".

FUNCTION=function
  specifies the name of the model function for the project. For more information, see

TOKEN=%authorization-token
  specifies the authorization token that was generated with the %MM_GET_TOKEN
  macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.

See  “%MM_GET_TOKEN” on page 4

Example

%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token{
  baseURL=&servernm,
  user=&userID,
  pw=&password,
  tokenname=myTokenName
%MM_CREATE_PROJECTVERSION Macro

Creates a version within a project.

Syntax

%MM_CREATE_PROJECTVERSION (  
  PROJECTID =project-ID,  
  <PROJECTVERSIONNM =project-version-name>,  
  <PROJECTVERSIONDESC =project-version-description>,  
  SERVERNM =host-name:<port>,  
  <TOKEN =%authorization-token>,  
  <PROJECTVERSIONID =project-version-ID-macro-variable>  
);

Required Arguments

PROJECTID =project-ID  
  specifies the project ID of the project where the project version is to be created.

SERVERNM=host-name <:port>  
  specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

Optional Arguments

PROJECTVERSIONNM =project-version-name  
  specifies the name of the project version.

PROJECTVERSIONDESC =project-version-description  
  specifies the description of the project version.

PROJECTVERSIONID =project-version-ID-macro-variable  
  specifies the macro variable to assign to the project version ID that is created. A UUID assigned by the Model Repository service. If this argument is not included, the default macro variable name that is created is "_projectVersionID".

TOKEN=%authorization-token  
  specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.
Example

%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%m_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);

%m_create_projectVersion(
    projectID=%str(&projID),
    projectversionnm=myProjVerName,
    projectversiondesc=Description of myProjVerName,
    servernm=&servernm,
    token=myTokenName,
    projectversionID=newProjVerID
);

%MM_IMPORT_MODEL Macro

Imports a model into a project version or folder.

Syntax

%MM_IMPORT_MODEL ( 
    MODELNM=model-name,
    MODELLOC=model-location,
    <MODELID=model-ID-macro-variable>,
    <MODELDESC=model-description>,
    <MODELFUNC=model-function-name>,
    <FILETYPE=ZIP | SPK | PMML | SASAST>,
    <FILESIZEOVERRIDEm=filesize-override-flag>,
    <PROJECTID=project-ID>,
    <PROJECTVERSIONID=project-version-ID>,
    <FOLDERID=folder-ID>,
    IMPORTINTO=import-into-object,
    <PKGFolder=ZIP-package-folder>,
    SERVERNM=host-name:<port>,
    <TOKEN=%authorization-token>
);
Required Arguments

MODELNM =model-name
  specifies the name of the model.

MODELLOC= model-location
  specifies the location from which the model is to be imported. The location is a
  physical location on your local file system or a CAS library that SAS Studio can
  access when running the macro.

IMPORTINTO=import-into-object
  specifies the location to which the model is to be imported. The default is the project
  version when importing models into a project. Valid values are “project”, “folder”, or
  “projectVersion”. The values are not case sensitive.

SERVERNM=host-name [:port>
  specifies the URL where SAS Model Manager is running. It includes the host name
  and port for the application server. The default port is 80.

Optional Arguments

Either the folderID, projectID, or projectID and projectversionID arguments must be
provided. If just projectID is specified, the model is imported into the latest project
version.

MODELDESC =model-description
  specifies the description of the model.

MODELID =model-ID-macro-variable
  specifies the macro variable to assign to the model ID that is created. A UUID
  assigned by the Folders service. If this argument is not included, the default macro
  variable name that is created is "_modelID".

MODELFUNC=model-function-name
  specifies the name of the function for the model. For more information, see “Types

FILETYPE=ZIP | SPK | PMML | SASAST
  specifies the file type that contains the model content to be imported.

FILESIZEOVERRIDE=filesize-override-flag
  specifies whether to override the file size limitation for analytic store files when
  importing a model. Valid values are Y and N.

  The FILESIZEOVERRIDE argument is used only when importing locally stored
  SASAST files. The on-disk byte count is checked, and if it is more than 5MB
  (5242880 bytes), the import is canceled and a message is given to the user about the
  size of the file. The user can, however re-issue the same import, adding the
  FILESIZEOVERRIDE=Y argument to the invocation, and the import of the analytic
  store file proceeds as usual.

  Default  N

PROJECTID =project-ID
  specifies the project ID for the project to import the model into.

PROJECTVERSIONID =project-version-ID
  specifies the ID for the project version where the model is to be imported into.

  Default  LATEST
PKGFOLDER = ZIP-package-folder
specifies the location for the ZIP package that is built during the macro execution. The ZIP file contains all the supporting component files, such as JSON, XML, and SAS files.

FOLDERID = folder-ID
specifies the ID for the folder where the model is to be imported into.

TOKEN=%authorization-token
specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.

See “%MM_GET_TOKEN” on page 4

Examples

Example 1: Import a SAS Enterprise Miner SPK Model File into a Project
Either the folderID or projectID arguments must be provided. When using the projectID argument, you can also specify the projectversionID. If only the projectID is specified, the model is imported into the latest project version.

```sas
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token(
   baseURL=&servernm,
   user=&userID,
   pw=&password,
   tokenname=myTokenName
);
%let model1=/home/models/model1/miningResult.spk;
%let type=SPK;
%mm_import_model(
   modelnm          = MyModel&type,
   modeldesc        = Description of MyModel&type,
   modelfunc        = Classification,
   modelloc         = &model1,
   filesizeoverride = N,
   projectID        = %str(&projID),
   importinto       = project,
   servernm         = &serverNm,
   token            = %myTokenName,
   modelID          = myModelID
);
```

Example 2: Import a Model within a ZIP File into a Project

```sas
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token(
   baseURL=&servernm,
   user=&userID,
   pw=&password,
   tokenname=myTokenName
);
```
user=&userID,
pw=&password,
tokenname=myTokenName
);utch model2 = /home/models/model2/MyModel.zip;
%let type=ZIP;
%m_m_import_model(
    modelnm          = MyModel&type,
    modeldesc        = Description of MyModel&type,
    modelfunc        = Classification,
    modelloc         = &model2,
    projectID        = %str(&projID),
    importinto       = project,
    servernm         = &servernm,
    token            = %myTokenName,
    modelID          = myModelID
);

Example 3: Import a PMML XML File into a Project

%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%m_m_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);

%let model3 = /home/models/model3/neural.xml;
%m_m_import_model(
    modelnm          = Neural,
    modeldesc        = PMML model,
    modelfunc        = Classification,
    modelloc         = &model3,
    projectID        = %str(&projID),
    importinto       = project,
    servernm         = &servernm,
    token            = %myTokenName,
    modelID          = myModelID
);

Example 4: Import an Analytic Store File into a Project

Note: When importing an analytic store file, the file is copied to the Files service. The FILESIZEOVERRIDE argument is used for handling large files. For more information, see FILESIZEOVERRIDE on page 11.

%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%m_m_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
A SAS analytic store file can also be stored in a CAS library. Here is an example of importing an analytic store model from a CAS library.

```sas
libname rstore cas caslib="casuser";
%mm_import_model(
    modelnm = HMEQ SVM Analytic Store,
    modeldesc = Analytic store model,
    modelfunc = ,
    modelloc = rstore.hmeq_svm_output,
    filesizeoverride = N,
    projectID = %str(&projID),
    importinto = project,
    servernm = &servernm,
    token = %myTokenName,
    modelID = myModelID,
    pkgfolder = /home/models/pkgs
);
```

### %MM_IMPORT_ASTORE_MODEL Macro

Imports a single analytic store model into a project or folder.

**Interaction:** Users must have Read and Write permissions to the source file system directory path for the ModelStore caslib. For more information, see “Configuring Model Data Libraries” in SAS Viya Administration: Models.

**Note:** This macro performs the same import function for a SAS analytic store model as the %MM_IMPORT_MODEL macro, except that the analytic store file is copied to a CAS library by using the RSTORE argument.

### Syntax

```
%MM_IMPORT_ASTORE_MODEL (  
    LOCATIONID = location-ID,  
    MODELNAME = model-name,  
    RSTORE = astore-cas-table,  
    ISFOLDER = location-type,  
```

%MM_IMPORT_ASTORE_MODEL Macro

Required Arguments

LOCATIONID=location-ID
specifies the project ID or folder ID to which the model is to be imported into. The default is the latest project version when importing models into a project.

ISFOLDER=location-type
specifies whether the LOCATIONID is a folder ID or a project ID. The default is N.

MODELNAME =model-name
specifies the name of the model.

RSTORE=astore-cas-table
specifies a two-level SAS name for the analytic store CAS table.

Optional Arguments

ASTOREFILE =astore-file-path
specifies the file path to the analytic store file in the file system.

ASTOREFILEURI =astore-file-URI
specifies the file URI to the analytic store file in the Files service.

BASEURL=%str(host-name <:port> )
specifies the host name and port for the application server.

MODELDESC =model-description
specifies the description of the model.

TARGET =target-variable
specifies the target variable of the model.

TARGETLEVEL =target-level
specifies the target level of the model. Valid values are binary, interval, nominal, and ordinal.

PKG_FOLDER =ZIP-package-folder
specifies the location for the ZIP package that is built during the macro execution. The ZIP file contains all the supporting component files, such as JSON, XML, and SAS files.

MININGFUNCTION=model-function-name
specifies the name of the function for the model. For more information, see “Types of Model Functions” in SAS Model Manager: User’s Guide.
MININGALGORITHM=\textit{mining-algorithm}

specifies the name of the function for the model.

\textbf{PROJECTVERSION} =\textit{project-version}

specifies the project version where the model is to be imported into. Valid values are \texttt{NEW} and \texttt{LATEST}.

Default \texttt{LATEST}

\textbf{TOKEN}=\texttt{\%authorization-token}

specifies the authorization token that was generated with the \texttt{\%MM\_GET\_TOKEN} macro.

\textit{Note:} You must specify this argument to call all macros in SAS Studio 4.4.

See “\texttt{\%MM\_GET\_TOKEN}” on page 4

**Example: Import a Forest Analytic Store Model into a New Project Version**

```sas
   cas _CAS\_PUBLIC_;
   caslib _ALL_ assign;
   libname local '/home/sasdemo/MMLib';
   %mm\_loadData2CAS
   (sasData= local.hmeqperf\_1\_q1,
   outcaslib=public,
   options=replace)
;

   /* Create a Forest analytic store model and store the model in a CAS table. */
   proc forest data=public.hmeqperf\_1\_q1 maxdepth=50 numbin=20;
   target BAD / level=nominal;
   input LOAN MORTDUE VALUE YOJ DEROG DELINQ CLAGE CLNO DEBTINC / level=interval;
   input REASON JOB NINQ / level=nominal;
   savestate rstore=public.state;
   run;

   %mm\_import\_astore\_model(
   locationID=e62ea618-5911-4949-bd6e-18c6cf903f79,
   isfolder=N,    modelname=%nrstr(Forest Astore),
   modeldesc=Forest,
   projectVersion=%str(new),
   rstore=public.state,
   miningAlgorithm=%nrstr(forest),
   miningFunction=classification,    pkgfolder        = /home/sasdemo/models/pkgs);
```

\texttt{\%MM\_GET\_REPOSITORY\_ID}

Retrieves the repository ID and repository folder ID for the specified repository name.

\textbf{Syntax}

\texttt{\%MM\_GET\_REPOSITORY\_ID (}
REPOSITORYNM = repository-name,
IDVAR = repository-ID-macro-variable,
SERVERNM = host-name:<port>,
<FLDRIDVAR = repository-folder-ID-macro-variable>,
<TOKEN = %authorization-token>,
);

**Required Arguments**

REPOSITORYNM = repository-name
    specifies the name of the repository. The macro search for the name is not case
    sensitive.

IDVAR = repository-ID-macro-variable
    specifies the macro variable to assign to the ID for the repository.

SERVERNM = host-name <port>
    specifies the URL where SAS Model Manager is running. It includes the host name
    and port for the application server. The default port is 80.

**Optional Arguments**

FLDRIDVAR = repository-folder-ID-macro-variable
    specifies the macro variable to assign to the repository folder ID. If this argument is
    not included, the default macro variable name that is created is "_fldrID".

TOKEN = %authorization-token
    specifies the authorization token that was generated with the %MM_GET_TOKEN
    macro.

*Note:* You must specify this argument to call all macros in SAS Studio 4.4.

See “%MM_GET_TOKEN” on page 4

**Example**

```sas
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);

%mm_get_repository_id(
    repositorynm=MyRepository,
    idvar=myRepID,
    fldridvar=myRepFldrID,
    servernm=&servernm,
    token=%myTokenName
);
```
%MM_GET_FOLDER_ID
Retrieves the folder ID for the specified folder name.

Syntax
%MM_GET_FOLDER_ID (  
    FOLDERNM =folder-name,  
    IDVAR=folder-ID-macro-variable,  
    SERVERNM =host-name:<port>,  
    <TOKEN =%authorization-token>  
);  

Required Arguments
FOLDERNM =folder-name  
specifies the name of the folder. The macro search for the name is not case sensitive.

IDVAR =folder-ID-macro-variable  
specifies the macro variable to assign to the folder ID.

SERVERNM=host-name <:port>  
specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

Optional Argument
TOKEN=%authorization-token  
specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.

See “%MM_GET_TOKEN” on page 4

Example
%let servernm=http://myserver.com;  
%let userID=myUserID;  
%let password=myPassword;  
  
%mm_get_token(  
    baseURL=&servernm,  
    user=&userID,  
    pw=&password,  
    tokenname=myTokenName  
);  
  
%mm_get_folder_id(  
    foldernm=MyFolder,  
    idvar=myFldrID,  
    servernm=&servernm,  
    token=%myTokenName  
)
%MM_GET_PROJECT_ID

Retrieves the project ID for the specified project name.

Interaction: If more than one project with the same name exists in another model repository or folder, the macro cannot retrieve the project ID.

Syntax

%MM_GET_PROJECT_ID (  
   PROJECTNM = project-name,  
   <REPOSITORYNM = project-name>,  
   IDVAR = project-ID-macro-variable,  
   SERVERNM = host-name:<port>,  
   <TOKEN = %authorization-token>,  
);  

Required Arguments

PROJECTNM = project-name
   specifies the name of the project. The macro search for the name is not case sensitive.

IDVAR = project-ID-macro-variable
   specifies the macro variable to assign to the project ID.

Note: If more than one project with the same name exists in another model repository or folder, the macro cannot retrieve the project ID. Therefore the IDVAR argument would not have a value.

SERVERNM = host-name:<port>
   specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

Optional Arguments

REPOSITORYNM = repository-name
   specifies the name of the repository in which to search for the project. The macro search for the name is not case sensitive.

TOKEN = %authorization-token
   specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.

See “%MM_GET_TOKEN” on page 4

Example

%mm_get_project_id(  
   projectnm=MyProject,
%%MM_GET_PROJECTVERSION_ID

Retrieves the project version ID for the specified project version name.

---

**Syntax**

```latex
%%MM_GET_PROJECTVERSION_ID (  
  PROJECTID = project-ID,  
  <PROJECTVERSIONNM = project-version-name>,  
  IDVAR = project-version-ID-macro-variable,  
  SERVERNM = host-name:<port>,  
  <TOKEN = %authorization-token>  
);  
```

**Required Arguments**

- **PROJECTID = project-ID**
  specifies the project ID for the project where the project version is located.

- **IDVAR = project-version-ID-macro-variable**
  specifies the macro variable to assign to the project version ID.

- **SERVERNM = host-name:<port>**
  specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

**Optional Arguments**

- **PROJECTVERSIONNM = project-version-name**
  specifies the name of the project version. If a value is not specified, the name of the latest version is the default. The macro search for the name is not case sensitive.

- **TOKEN = %authorization-token**
  specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

*Note:* You must specify this argument to call all macros in SAS Studio 4.4.

See **“%MM_GET_TOKEN” on page 4**

**Example**

```latex
%let servernm=http://myserver.com;  
%let userID=myUserID;  
%let password=myPassword;  
%mm_get_token (  
  baseURL=&servernm,  
  user=&userID,  
  password=&password,  
  servernm=&servernm,  
  token=&myTokenName  
);  
```
pw=&password,
tokenname=myTokenName
);

%mm_get_projectVersion_id(
    projectID=%str(&projID.),
    projectVersionNm=myProjVerName,
    idvar=projVerID,
    servernm=&servernm,
    token=%myTokenName
);

%MM_GET_MODEL_ID
Retrieves the model that is associated with the specified model name.

Note:  Model names are not required to be unique. If one or more models are found with
the same name, all of the model names are returned in the results that are displayed
to the user.

Syntax

%MM_GET_MODEL_ID (  
    MODELNM =model-name,  
    IDVAR=model-ID-macro-variable,  
    SERVERNM =host-name:<port>,  
    <TOKEN =%authorization-token>  
);

Required Arguments

MODELNM =model-name  
specifies the name of the model.

IDVAR =model-ID-macro-variable  
specifies the macro variable to assign to the model ID.

SERVERNM =host-name:<port>  
specifies the URL where SAS Model Manager is running. It includes the host name
and port for the application server. The default port is 80.

Optional Argument

TOKEN =%authorization-token  
specifies the authorization token that was generated with the %MM_GET_TOKEN
macro.

Note:  You must specify this argument to call all macros in SAS Studio 4.4.

See  “%MM_GET_TOKEN” on page 4

Example

%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);

%mm_get_model_id(
    modelnm=myModel,
    idvar=myModelID,
    servernm=&servernm,
    token=%myTokenName
);

%MM_DELETE_REPOSITORY

Deletes the repository that is associated with the specified repository name or ID.

Note: Only SAS Administrators and other authorized users can delete a repository.

Syntax

%MM_DELETE_REPOSITORY (REPOSITORYNM =repository-name,
    REPOSITORYID=repository-ID,
    SERVERNM =host-name:<port>,
    <TOKEN =%authorization-token>);

Required Arguments

REPOSITORYNM =repository-name
    specifies the name of the repository. Either the REPOSITORYNM argument or the
    REPOSITORYID argument are required, but not both. If both are provided, the
    REPOSITORYNM argument takes precedence.

REPOSITORYID =repository-ID
    specifies the ID for the repository. Either the REPOSITORYNM argument or the
    REPOSITORYID argument are required, but not both. If both are provided, the
    REPOSITORYNM argument takes precedence.

SERVERNM =host-name:<port>
    specifies the URL where SAS Model Manager is running. It includes the host name
    and port for the application server. The default port is 80.

Optional Argument

TOKEN=%authorization-token
    specifies the authorization token that was generated with the %MM_GET_TOKEN
    macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.
See “%MM_GET_TOKEN” on page 4

Example

```text
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);

%mm_delete_repository(
    /* Either repositorynm or repositoryID can be used. */
    repositorynm=MyRepository,
    /* repositoryID=%str(&myRepID),*/
    servernm=&servernm,
    token=%myTokenName
);
```

%MM_DELETE_FOLDER

Deletes the folder that is associated with the specified folder name or ID.

Syntax

```text
%MM_DELETE_FOLDER (    
    FOLDERNM =folder-name,
    FOLDERID=folder-ID,
    SERVERNM =host-name:<port>,
    <TOKEN =%authorization-token>
);
```

Required Arguments

**FOLDERNM =folder-name**

specifies the name of the folder. Either the FOLDERNM argument or the FOLDERID argument are required, but not both. If both are provided, the FOLDERNM argument takes precedence.

*Important:* It is recommended that you use the FOLDERID argument, because there other folders with the same name could exist within the common model repository.

**FOLDERID =folder-ID**

specifies the ID for the folder. Either the FOLDERNM argument or the FOLDERID argument are required, but not both. If both are provided, the FOLDERNM argument takes precedence.
SERVERNM=host-name <:port>
specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

Optional Argument

TOKEN=%authorization-token
specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.

See “%MM_GET_TOKEN” on page 4

Example

%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%mm_get_token(
  baseURL=&servernm,
  user=&userID,
  pw=&password,
  tokenname=myTokenName
);

%mm_delete_folder(
  /* Either foldernm or folderID can be used. */
  foldernm=MyFolder,
  /* folderID=%str(&myFldrID), */
  servernm=&servernm,
  token=myTokenName
);

%MM_DELETE_PROJECT

Deletes the project that is associated with the specified project name or ID.

Syntax

%MM_DELETE_PROJECT (  
  PROJECTNM =project-name,  
  PROJECTID=project-ID,  
  SERVERNM =host-name::<port>,  
  <TOKEN =%authorization-token>
);
Required Arguments

PROJECTNM = project-name
specifies the name of the project. Either the PROJECTNM argument or the PROJECTID argument are required, but not both. If both are provided, the PROJECTNM argument takes precedence.

PROJECTID = project-ID
specifies the ID for the project. Either the PROJECTNM argument or the PROJECTID argument are required, but not both. If both are provided, the PROJECTNM argument takes precedence.

SERVERNM= host-name <:port>
specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

Optional Argument

TOKEN=%authorization-token
specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.
See “%MM_GET_TOKEN” on page 4

Example

%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);
%mm_delete_project(
    /* Either projectnm or projectID can be used.*/
    projectnm=MyProject,
    /* projectID=%str(&projID), */
    servernm=&servernm,
    token=%myTokenName
);

%MM_DELETE_PROJECTVERSION
Deletes the project version that is associated with the specified project version name or ID. If only one project version exists, it is not deleted.

Syntax

%MM_DELETE_PROJECTVERSION (
PROJECTID ="project-ID",
SERVERNM ="host-name:<port>",
<PROJECTVERSIONNM ="project-version-name",
<PROJECTVERSIONID ="project-version-ID-macro-variable",
<TOKEN ="%authorization-token"
);

Required Arguments

PROJECTID ="project-ID"
specifies the project ID of the project where the project version is to be created.

SERVERNM=host-name <:port>
specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

Optional Arguments

PROJECTVERSIONNM ="project-version-name"
specifies the name of the project version. Either the PROJECTVERSIONNM argument or the PROJECTVERSIONID argument are required, but not both. If both are provided, the PROJECTVERSIONNM argument takes precedence.

PROJECTVERSIONID ="project-version-ID-macro-variable"
specifies the project version ID. Either the PROJECTVERSIONNM argument or the PROJECTVERSIONID argument are required, but not both. If both are provided, the PROJECTVERSIONNM argument takes precedence.

TOKEN= %authorization-token
specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.

See “%MM_GET_TOKEN” on page 4

Example

%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);

%mm_delete_projectversion(
    projectID=%str(&projID.),
    /* If the projectversionnm argument is omitted, */
    /* the LATEST version name is the default. */
    /* Either projectversionnm or projectversionID can be used. */
    projectversionnm=myProjVerName,
    /* projectversionID=%str(&projVerID), */
    servernm=&servernm,
%MM_DELETE_MODEL

Deletes a model from the common model repository.

Note: If one or more models are found with the same name, a listing that contains all of the model names is returned. No models are deleted.

Syntax

%MM_DELETE_MODEL ( 
    MODELNM = model-name,
    MODELID = model-ID,
    SERVERNM = host-name:<port>,
    <TOKEN =%authorization-token>
);

Required Arguments

MODELNM = model-name
    specifies the name of the model. Either the MODELNM argument or the MODELID argument are required, but not both. If both are provided, the MODELNM argument takes precedence.

MODELID = model-ID
    specifies the model ID. Either the MODELNM argument or the MODELID argument are required, but not both. If both are provided, the MODELNM argument takes precedence.

SERVERNM = host-name:<port>
    specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

Optional Argument

TOKEN = %authorization-token
    specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

Note: You must specify this argument to call all macros in SAS Studio 4.4.

See “%MM_GET_TOKEN” on page 4

Example

%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token(
    baseURL=&servernm,
    user=&userID,
%MM_PUBLISH_MODEL

Publishes a model to CAS, Hadoop, SAS Micro Analytic Service, or Teradata.

Interaction:
If a value is specified for both the MODELNM and MODELID arguments, then the publish request is based on the model that is found with the specified ID, not the model name.

Syntax

%MM_PUBLISH_MODEL (  
  MODELNM =model-name,  
  MODELID=model-ID,  
  <MODELNOTES=model-publish-notes>,  
  <MODELINFOFILEREF=model-info-fileref>,  
  <PUBLISHNM=published-model-name>,  
  SCORECODETYPE=model-score-code-type,  
  <SCORECODE=score-code>,  
  <SCORECODEFILEREF=score-code-fileref>,  
  PUBLISHDESTINATION=publish-destination-name,  
  SERVERNM=host-name:<port>,  
  <TOKEN =%authorization-token>  
);

Required Arguments

MODELNM =model-name
specifies the name of the model. Either the MODELNM argument or the MODELINFOFILEREF argument are required, but not both.

MODELID=model-ID
specifies the model ID. If your score code is already included as part of the model in SAS Model Manager, provide the model ID and the macro retrieves the score code from the Files service.

Note: A value for this argument is required only if publishing a model that is registered in the SAS Model Manager common model repository. When this argument is specified, if you provide a value for the SCORECODE or SCORECODEFILEREF arguments, they are ignored by the macro. If you are publishing a model that is not in the SAS Model Manager common model repository, this argument is not required. However, if this argument is not
specified, you must provide a location for the score code in the SCORECODEFILEREF argument, or inline code within the SCORECODE argument.

SCORECODETYPE = model-score-code-type
specifies the score code type for the model. A value must be specified for this argument, unless a value is provided for the MODELINFOFILEREF argument.

Valid values are ds2EmbeddedProcess, ds2Package, sasProgram, pmml, dataStep, and analyticStore.

PUBLISHDESTINATION = publish-destination-name
specifies the name of the publish destination. You can use the %MM_PRINTPUBLISHDESTINATION macro to retrieve a list of the publish destination names.

SERVERNM = host-name <;port>
specifies the URL where SAS Model Manager is running. It includes the host name and port for the application server. The default port is 80.

Optional Arguments

MODELNOTES = model-publish-notes
specifies the publish notes for a model.

**TIP**
If there are any special characters in the description value, such as commas, ampersands, or quotes you must enclose the value in %NRSTR().

MODELINFOFILEREF = model-info-fileref
specifies the pre-defined fileref that contains the user-defined publish information JSON content.

*Note:* When this argument is specified, if you provide a value for the SCORECODE, SCORECODEFILEREF, or MODELID arguments, they are ignored by the macro.

PUBLISHNM = published-model-name
specifies the published name for the model. The default value is the model name (MODELNM), if not specified.

SCORECODE = model-score-code
specifies inline score code. A value must be specified for this argument, unless a value is provided for the MODELINFOFILEREF argument.

**TIP**
If there are any commas, you must enclose the value in %STR().

SCORECODEFILEREF = model-score-code-fileref
specifies a predefined SAS fileref that points to your locally store score code. If a value is specified for this argument, the SCORECODE argument is ignored by the macro.

**TIP**
If there are any commas, you must enclose the value in %STR().

TOKEN = %authorization-token
specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

*Note:* You must specify this argument to call all macros in SAS Studio 4.4.

See “%MM_GET_TOKEN” on page 4
Example

Note: To get a list of the publish destinations, see “Example 2: Print Multiple Publish Destinations” on page 54.

```plaintext
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%mm_get_token{
   baseURL=&servernm,
   user=&userID,
   pw=&password,
   tokenname=myTokenName
};

%let pubdest=%str(CAS);

%mm_publish_model{
   modelnm=myModelName,
   modelnotes=%nrstr(Published model notes),
   scorecodetype=dataStep,
   publishnm=myPublishedModelName,
   modelID=%str(&myModelID),
   publishdestination=&pubdest,
   servernm=&servernm,
   token=%myTokenName
};
```
Overview of Performance Monitoring Macros

The performance monitor macro uses 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. It monitors the performance of a model from three categories.

Here are the three categories:

Characteristic Analysis
The performance indexes CHAR_P1 and CHAR_P25 represent the count of input variables with deviation index scores exceeding 0.1 and 0.25, respectively.

Stability Analysis
The output deviation index scores represent the deviation levels in the distribution of the model’s scored output variables.

Model Assessment
The Lift, Gini (ROC and Trend), and Kolmogorov-Smirnov (KS) reports, include the following decay statistics.

- \( \text{lift5Decay} \)
  is the lift performance decay based on the top 5% of the target population of interest from time A to time B.

- \( \text{lift10Decay} \)
  is the lift performance decay based on the top 10% of the target population of interest from time A to time B.

- \( \text{lift15Decay} \)
  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.

See Also


Macro Variables

It is helpful to set up macro variables for repetitive code. SAS Model Manager provides macro variables that can be used by the performance monitoring macros to create reports.

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Description</th>
<th>Example Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_MM_PERFEXECUTOR</td>
<td>Indicates where the macro is to be executed. A value of 0 should be specified when running the macro using SAS Studio. If the value is set to 0, the Model Management API is called and a record is inserted into the Job History table. The Job History table is displayed on the Performance tab of a project in the SAS Model Manager web application. If the value is set to 1, a record is not inserted into the Job History table.</td>
<td>0</td>
</tr>
<tr>
<td>_MM_PROJECTUUID</td>
<td>The UUID for the project.</td>
<td>40853758-953e-4d18-92b1-90eeb3f80b08</td>
</tr>
<tr>
<td>_MM_MODELID</td>
<td>The UUID of the model.</td>
<td>7d91298d-03fc-4e50-9fd1-8abdc48436e9</td>
</tr>
<tr>
<td>Macro Variable Name</td>
<td>Description</td>
<td>Example Value</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>_MM_MODELFLAG</td>
<td>The flag that indicates whether the model is a champion model, challenger model, or a candidate model. Here are the values for the different types of model roles: • 0 is for a champion model. • 1 is for a challenger model. • -1 is for a candidate model.</td>
<td>0</td>
</tr>
<tr>
<td>_MM_PREDICTEDVAR</td>
<td>The predicted variable, when the model TARGET level is interval.</td>
<td>P_PRICE</td>
</tr>
<tr>
<td>_MM_EVENTPROBVAR</td>
<td>The posterior probability variable of the event, when the model TARGET level is binary.</td>
<td>P_BAD1</td>
</tr>
<tr>
<td>_MM_NONEVENTPROBVAR</td>
<td>The non-event posterior probability variable, when the model TARGET level is binary. If a user does not specify a value, the macro determines its value based on the value of _MM_EVENTPROBVAR. If an incorrect value is specified, the result table MM_FITSTAT is not generated.</td>
<td>P_BAD0</td>
</tr>
<tr>
<td>_MM_TARGETNONEVENT</td>
<td>The target non-event value, when the model TARGET level is binary. If a user does not specify a value, the macro determines its value based on the values of _MM_TARGETVAR and _MM_TARGETLEVEL. If an incorrect value is specified, the result table MM_FITSTAT is not generated. <em>Note:</em> The value for this macro variable is case-sensitive. It must be the same as the value in the input data.</td>
<td>0</td>
</tr>
<tr>
<td>Macro Variable Name</td>
<td>Description</td>
<td>Example Value</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>_MM_SCORECODETYPE</td>
<td>The type of model score code. Valid values are DATASETP, DS2EP, ANALYTICSTORE, and SASPROGRAM.</td>
<td>DATASETP</td>
</tr>
<tr>
<td></td>
<td>Note: The score code type should be set to ANALYTICSTORE if only one analytic store file exists within a model. The score code type should be set to DS2EP if the model contains DS2 embedded process code and one or more analytic store files.</td>
<td></td>
</tr>
<tr>
<td>_MM_PERFOUTCASLIB</td>
<td>The CASLIB and libref for the performance monitoring results.</td>
<td>ModelPerformanceData</td>
</tr>
<tr>
<td>_MM_PERINCASLIB</td>
<td>The global CASLIB and libref of performance monitor input data.</td>
<td>public</td>
</tr>
<tr>
<td>_MM_PERF_INTABLEPREFIX</td>
<td>The prefix of the performance input table name.</td>
<td>hmeqperf</td>
</tr>
<tr>
<td></td>
<td>Note: The value for the prefix cannot contain underscores, and spaces are not recommended in the prefix name or the table name.</td>
<td></td>
</tr>
<tr>
<td>_MM_TABLENAMELEVEL</td>
<td>The format for the name of the data input table. See “Prepare Performance Data Sources”.</td>
<td>3</td>
</tr>
<tr>
<td>_MM_PERFSTATICTABLE</td>
<td></td>
<td>hmeqperf_1_q1</td>
</tr>
<tr>
<td>_MM_TARGETVAR</td>
<td>The model target variable. This value is case-sensitive. It must be as same as the value in the input data.</td>
<td>BAD</td>
</tr>
<tr>
<td>_MM_TARGETLEVEL</td>
<td>The level for the target variable.</td>
<td>INTERVAL or BINARY</td>
</tr>
<tr>
<td>_MM_TARGETEVENT</td>
<td>The target event value.</td>
<td>1</td>
</tr>
<tr>
<td>_MM_RUNSCORE</td>
<td>The flag that indicates whether scoring should be run or not. Y is for YES, and N is for NO. Set this value to N when the performance input table already contains the scoring output variables.</td>
<td>Y or N</td>
</tr>
<tr>
<td>Macro Variable Name</td>
<td>Description</td>
<td>Example Value</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>_MM_MAX_BINS</td>
<td>The global number of bins for all binning variables for characteristic analysis. The valid range is 1 ~ 1000. The default value is 10. Important: It is recommended that you use the same value for this macro variable for each time period, when running the %MM_PERFORMANCE_MONITOR macro.</td>
<td>10</td>
</tr>
<tr>
<td>_MM_CAKEEPVARS</td>
<td>The input variables for characteristic analysis. The variables are separated by a blank space.</td>
<td>Value Loan Job</td>
</tr>
<tr>
<td>_MM_KEEPVARS</td>
<td>Input variables for stability analysis. The variables are separated by a blank space. The value is the same as the _MM_PredictedVar or _MM_EVENTPROBVar macro variable.</td>
<td>P_BAD1</td>
</tr>
<tr>
<td>_MM_CHARACTERISTICALERT</td>
<td>The alert criterion for characteristic analysis. The default value is char_p1&gt;2. Important: It is recommended that you use the default value for this macro variable.</td>
<td>char_p1&gt;2</td>
</tr>
<tr>
<td>_MM_CHARACTERISTICWARNING</td>
<td>The warning criterion for characteristic analysis. The default value is char_p1&gt;5 or char_p25&gt;0. Important: It is recommended that you use the default value for this macro variable.</td>
<td>char_p1&gt;5 or char_p25&gt;0</td>
</tr>
<tr>
<td>_MM_STABILITYALERT</td>
<td>The alert criterion for stability analysis. The default value is stab_p1 &gt; 1. Important: It is recommended that you use the default value for this macro variable.</td>
<td>stab_p1 &gt; 1</td>
</tr>
<tr>
<td>Macro Variable Name</td>
<td>Description</td>
<td>Example Value</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>_MM_STABILITYWARNING</td>
<td>The warning criterion for stability analysis.</td>
<td>stab_p1 &gt; 2</td>
</tr>
<tr>
<td></td>
<td>The default value is <strong>stab_p1 &gt; 2</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Important:</em> It is recommended that you use the default value for this</td>
<td></td>
</tr>
<tr>
<td></td>
<td>macro variable.</td>
<td></td>
</tr>
<tr>
<td>_MM_MODELASSESSMENTALERT</td>
<td>The alert criterion for model assessment.</td>
<td>(lift5Decay &gt; 0.15 and</td>
</tr>
<tr>
<td></td>
<td>The default value is **(lift5Decay &gt; 0.15 and</td>
<td>lift10Decay&gt;0.12) or</td>
</tr>
<tr>
<td></td>
<td><em>Important:</em> It is recommended that you use the default value for this</td>
<td>(giniDecay &gt; 0.1 or</td>
</tr>
<tr>
<td></td>
<td>macro variable.</td>
<td>ksDecay &gt; 0.1)</td>
</tr>
<tr>
<td>_MM_MODELASSESSMENTWARNING</td>
<td>The warning criterion for model assessment.</td>
<td>lift5Decay &gt; 0.05</td>
</tr>
<tr>
<td></td>
<td>The default value is <strong>lift5Decay &gt; 0.05</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Important:</em> It is recommended that you use the default value for this</td>
<td></td>
</tr>
<tr>
<td></td>
<td>macro variable.</td>
<td></td>
</tr>
<tr>
<td>_MM_TRACE</td>
<td>Indicates whether to write a trace log.</td>
<td>ON or OFF</td>
</tr>
<tr>
<td>_MM_SAVEPERFRESULT</td>
<td>Indicates whether to save the performance monitoring results as a data</td>
<td>N or Y</td>
</tr>
<tr>
<td></td>
<td>source in the CASLIB “&amp;_MM_PERFOUTCASLIB”.</td>
<td></td>
</tr>
<tr>
<td>_MM_ASTORELOCATION</td>
<td>A string that indicates where the analytic store files are located. The</td>
<td>caslib1.astore caslib1.astore2</td>
</tr>
<tr>
<td></td>
<td>file locations are separated by a blank space.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In this example, two analytic store files <strong>astore.sashdat</strong> and</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>astore2.sashdat</strong> are stored in the Caslib “caslib1”.</td>
<td></td>
</tr>
<tr>
<td>_MM_FORCERUNALLDATA</td>
<td>Indicates to force performance monitoring to run against all performance</td>
<td>N or Y</td>
</tr>
<tr>
<td></td>
<td>input tables.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The default value is N.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Important:</em> It is recommended that you use the default value for this</td>
<td></td>
</tr>
<tr>
<td></td>
<td>macro variable.</td>
<td></td>
</tr>
</tbody>
</table>
### Set Alert and Warning Conditions

The performance monitor macro uses performance measurement thresholds to benchmark and gauge the performance of a predictive model. It is recommended that you use the default values for the alert and warning conditions when running the %MM_PERFORMANCE_MONITOR macro. However, you can use the following assignment statements if you need to set the alert and warning conditions to different values.

**Characteristic Analysis**

A user can configure the thresholds for the performance indexes CHAR_P1 and CHAR_P25. The CHAR and CHAR indexes represent the count of input variables with deviation index scores exceeding 0.1 and 0.25, respectively.

Here is an example of set alert and warning thresholds:

```markdown
%let _MM_CharacteristicAlert = %nrstr(char_p1>5 or char_p25>0);
%let _MM_CharacteristicWarning = %nrstr(char_p1>2);
```

**Stability Analysis**

A user 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 set alert and warning thresholds:

```markdown
%let _MM_StabilityAlert = %nrstr(stab_p1>1);
%let _MM_StabilityWarning = %nrstr(stab_p1>2);
```
Model Assessment reports
For the Lift, Gini (ROC and Trend), and Kolmogorov-Smirnov (KS) reports, a user can configure threshold values for the following decay statistics.

lift5Decay
is the lift performance decay based on the top 5% of the target population of interest from time A to time B.

lift10Decay
is the lift performance decay based on the top 10% of the target population of interest from time A to time B.

lift15Decay
is the lift performance decay based on the top 15% of the target population of interest from time A to time B.

lift20Decay
is the lift performance decay based on the top 20% of the target population of interest from time A to time B.

giniDecay
is the performance decay of the Gini index from time A to time B.

ksDecay
is the performance decay of the KS statistic from time A to time B.

Here is an example of alert and warning thresholds:

%let _MM_ModelAssessmentAlert = %nrstr((lift5Decay>0.15 and lift10Decay>0.12)
or giniDecay>0.1 or ksDecay>0.1 );
%let _MM_ModelAssessmentWarning = %nrstr(lift5Decay>0.05);

Dictionary

%MM_PERFORMANCE_MONITOR Macro
Defines and runs performance monitoring for a champion or challenger model.

Syntax

%MM_PERFORMANCE_MONITOR ( 
   PERFLIB=monitoring-input-caslib, 
   PERFDATANAMEPREFIX=input-data-prefix, 
   PERFDATASRCDS=single-input-cas-table, 
   MM_MART=monitoring-output-caslib, 
   ASTOREFREF=fileref-to-analytic-store-file, 
   SCORECODEFILEREF=fileref-to-score-code-file, 
   RUNSCORE=flag-for-scoring, 
   SAVERESULT=flag-for-saving-performance-results 
);
**Required Arguments**

**PERFLIB**=*monitoring-input-caslib*

specifies the CASLIB for the performance input data.

```
perfLib=mmlib
```

**PERFDATANAMEPREFIX**=*input-data-prefix*

specifies the prefix of the performance input table name. This argument should not be populated when there is a value specified for the PERFDATASRCDS argument.

**PERFDATASRCDS**=*input-cas-table*

specifies the name of a CAS input table. This argument should not be populated when there is a value specified for the PERFDATANAMEPREFIX argument.

**MM_MART**=*monitoring-output-caslib*

specifies the CASLIB for the performance monitor results.

**SCORECODEURI**=*score-code-file-URI*

specifies the URI for the model score code file.

**ASTOREURI**=*analytic-store-score-code-file-URI*

specifies the URL for the analytic store score code file.

**ASTOREFREF**=*fileref-to-analytic-store-file*

specifies the fileref for the model analytic store file.

**SCORECODEFILEREF**=*fileref-to-score-code-file*

specifies the fileref for the model score code file.

```
scoreCodeFref=ep
```

When the DS2 embedded process code contains a reference to an analytic store file, you can create a fileref for the DS2 EP code. You can then pass the fileref to the `%MM_PERFORMANCE_MONITOR` macro.

```
filename ep '/home/models/forest_ep.sas';
```

**RUNSCORE**=*flag-for-scoring*

specifies the flag that indicates whether scoring should be run or not. The values are Y or N.

**SAVERESULT**=*flag-for-saving-performance-results*

specifies to save the performance results out of CAS memory. The values are Y or N.

**Details**

**Prepare Performance Data Sources**

You can choose to provide your own scored input data sources or have the system score the data when running performance. If you choose to provide scored data sources, the data tables must contain the predicted values for the scored model. If you choose to have the system to score your data, the model score code is used to score the data before generating the performance results.

Use one of the following formats for the name of the data table that you use as a data source, or for the name of the data tables that are located in the performance input CAS library (caslib). Note that the value for the `prefix` cannot contain underscores, and spaces are not recommended in the prefix name or the table name. In addition, the sequence number and time label must be unique across all of the data table names.

- `prefix_sequenceNumber_timeLabel`

```
hmeqperf_1_q1
```
• prefix_sequenceNumber_timeLabel_uid
  hmeqPerfScored_1_q1_9d5b4c98-e9f5-4858-8913-07ce91885da0

• prefix_sequenceNumber_timeLabel_uid_champion
  hmeqPerfScored_1_q1_9d5b4c98-e9f5-4858-8913-07ce91885da0_champion

**Note:** When you provide your own scored input data source and you indicate to use a library that contains tables with a specified prefix, your data table names must contain the UUID of the model.

### How to Run This Macro

The `%MM_PERFORMANCE_MONITOR` macro can be run using SAS Studio. Here is the process to prepare for running the macro.

1. Connect to a SAS Server.
   ```
   options cashost='cas-server-hostname'
   casport=cas-port-number;
   cas _mmcas_;
   caslib _all_ assign;
   ```

   **Note:** The OPTIONS statement is needed only if you are running the macro using a CAS server other than the default CAS server.

2. Load performance data sources into a caslib. For more information, see “Making Data Available to CAS” in *SAS Data Explorer: User’s Guide*.

3. Upload the model score code to a directory path that is accessible from SAS Studio. Here is an example:
   ```
   ~/scorecode/score.sas
   ```

4. Set the performance monitor predefined macro variables. The macro variables are used to pass model related properties and control how the performance monitor task is executed. For more information, see “Macro Variables” on page 32.
   ```
   %let _MM_ProjectUUID=%nrstr(40853758-953e-4d18-92b1-90eeb3f80b08);
   %let _MM_ModelID=%nrstr(7d91298d-03fc-4e50-9fd1-8abdc48436c9);
   %let _MM_ModelFlag = 0;
   %let _MM_TargetVar=BAD;
   %let _MM_ScoreCodeType = %str(DATASTEP);
   %let _MM_TargetEvent=1;
   %let _MM_EventProbVar=P_BAD1;
   %let _MM_TargetLevel=BINARY;
   %let _MM_PredictedVar=
   %let _MM_KeepVars=P_BAD1;
   %let _MM_CAEKeepVars=YOJ MORTDUE DEROG VALUE CLNO LOAN CLAGE DELINQ NINQ;
   %let _MM_Trace = OFF;
   %let _MM_Max_Bins = 10;
   %let _MM_PerfOutCaslib= ModelPerformanceData;
   %let _MM_PerfInCaslib=public;
   %let _MM_Perf_InTablePrefix=hmeqperf;
   %let _MM_TableNameLevel = 3;
   %let _MM_RunScore=Y;
   %let _MM_SAVEPERFRESULT=Y;
   ```
Examples

Example 1: Code Example for Running Performance Monitoring for a DATA Step Model

Use SAS Studio to run the code in this example on the default CAS server.

cas _mmcas_;  
caslib _all_ assign;

%let _mm_projectuuid=%nrstr(40853758-953e-4d18-92b1-90eeb3f80b08);  
%let _mm_modelid=%nrstr(7d91298d-03fc-4e50-9fd1-8abdc48436c9);  
%let _mm_modelflag = 0;  
%let _mm_targetvar=BAD;  
%let _mm_scorecodetype = %str(DATASTEP);  
%let _mm_targetevent=1;  
%let _mm_eventprobvar=P_BAD1;  
%let _mm_targetnonevent=0;  
%let _mm_noneventprobvar=P_BAD0;  

%let _mm_targetlevel=BINARY;  
%let _mm_predictedvar=;  
%let _mm_keepvars=YOJ MORTDUE DEROG VALUE CLNO LOAN CLAGE DELINQ NINQ;  
%let _mm_trace = ON;  
%let _mm_max_bins = 10;  
%let _mm_perfoutcaslib=ModelPerformanceData;  
%let _mm_perfincaslib=public;  
%let _mm_perf_intableprefix=hmeqperf;  
%let _MM_TableNameLevel = 3;  
%let _mm_runscore=Y;  
%let _mm_saveperfresult=Y;

/* Create a score code fileref if set _mm_runscore=Y */  
filename scoreref '~/scorecode/score.sas';  

%mm_performance_monitor(  
  perflib=&_MM_PerfInCaslib,  
  perfdatanameprefix=&_MM_Perf_InTablePrefix,  
  mm_mart=&_MM_PerfOutCaslib,  
  scorecodefref=scoreref,  
  runscore=&_MM_RunScore  
);  

%put SYSERR = &syserr.;  
%put SYSCC = &syscc.;  

/* View the performance monitoring results. */  
libname mm_mart cas caslib="&_MM_PerfOutCaslib" tag="&_MM_ProjectUUID";

/* View a list of the MM_MART library tables. */  
proc datasets lib=mm_mart;  
run;
**Example 2: Code Example for Running Performance Monitoring for a DATA Step Model with a Single Scored Data Source**

Use SAS Studio to run the code in this example on the default CAS server.

```sas
cas _mmcas_;
caslib _all_ assign;

%let _mm_projectuuid=%nrstr(40853758-953e-4d18-92b1-90eeb3f80b08);
%let _mm_modelid=%nrstr(7d91298d-03fc-4e50-9fd1-8abdc48436c9);
%let _mm_modelflag = 0;
%let _mm_targetvar=BAD;
%let _mm_scorecodetype = %str(DATASTEP);
%let _mm_targetevent=1;
%let _mm_eventprobvar=P_BAD1;
%let _mm_targetnonevent=0;
%let _mm_noneventprobvar=P_BAD0;

%let _mm_targetlevel=BINARY;
%let _mm_predictedvar=;
%let _mm_keepvars=P_BAD1;
%let _mm_cakeepvars=YOJ MORTDUE DEROG VALUE CLNO LOAN CLAGE DELINQ NINQ;
%let _mm_trace = ON;
%let _mm_max_bins = 10;
%let _mm_perfoutcaslib=ModelPerformanceData;
%let _mm_perfincaslib=public;
%let _MM_TableNameLevel = 3;
%let _MM_PerfStaticTable = hmeqPerfScored_1_q1;
%let _mm_runscore=N;
%let _mm_saveperfresult=Y;

%mm_performance_monitor
 (perflib=&_MM_PerfInCaslib, perfdatasrcds=&_MM_PerfStaticTable, mm_mart=&_MM_PerfOutCaslib, runscore=&_MM_RunScore);

%put SYSERR = &syserr.;
%put SYSCC = &syscc.;

/* View the performance monitoring results. */
libname mm_mart cas caslib="&_MM_PerfOutCaslib" tag="&_MM_ProjectUUID";

/* View a list of the MM_MART library tables. */
proc datasets lib=mm_mart;
run;
```

**Example 3: Code Example for Running Performance Monitoring for an Analytic Store Model**

Use SAS Studio to run the code in this example on the default CAS server.

```sas
cas _mmcas_;
caslib _all_ assign;
```
%let _MM_PerfExecutor = 0;
%let _MM_ProjectUUID = %nrstr(e60ae98e-b347-4b9b-b28c-023ba689dcaf);
%let _MM_TargetVar = bad;
%let _MM_TargetLevel = BINARY;
%let _MM_PredictedVar = ;
%let _MM_TargetEvent = 1;
%let _MM_EventProbVar = P_BAD1;
%let _MM_KeepVars = P_BAD1;
%let _MM_CAKeepVars = REASON VALUE YOJ CLAGE CLNO DEBTINC DELINQ DBROG JOB LOAN MORTDUE NINQ;
%let _MM_Trace = OFF;
%let _MM_Max_Bins = 10;
%let _MM_PerfOutCaslib = ModelPerformanceData;
%let _MM_PerfInCaslib = Public;
%let _MM_Perf_InTablePrefix = hmeqperf;
%let _MM_TableNameLevel = 4;
%let _MM_PerfStaticTable = ;
%let _MM_ForceRunAllData = N;
%let _MM_RunScore = N;
%let _MM_SAVEPERFRESULT = Y;
%let _MM_JobID = %nrstr(6e80398b-556c-4f9d-8373-80a45161a570);
%let _MM_ModelID = %nrstr(a7864cfb-2c2c-45c9-9f98-3542ccac4f85);
%let _MM_ModelName = %nrstr(GB);
%let _MM_ModelFlag = 0;
%let _MM_ScoreCodeType = DS2EP;
%let _MM_ScoreCodeURI = ;
%let _MM_ScoreAstURI = ;
%let _MM_aStoreLocation=ModelStore._93034785EFAC7B4BFF6BA4E4_E_ast ;

%mm_performance_monitor
(  perflib=&_MM_PerfInCaslib,
    perfDataNamePrefix=&_MM_Perf_InTablePrefix,
    mm_mart=&_MM_PerfOutCaslib,
    runScore=&_MM_RunScore,
    scorecodeURI=&_MM_ScoreCodeURI
);

%put &syserr;
%put &syscc;

/* View the performance monitoring results. */
libname mm_mart cas caslib="&_MM_PerfOutCaslib" tag="&_MM_ProjectUUID";

/* View a list of the MM_MART library tables. */
proc datasets lib=mm_mart;
run;
Chapter 4
Publish Destination Macros

Overview of Publish Destination Macros

The publish destination macros enable you to define, delete, and update publish destinations, as well as print a list of published destinations that have already been defined. Before you can use these macros, you must first create a global caslib using SAS Environment Manager. For more information, see “Create a Global Caslib” in SAS Viya Administration: Publishing Destinations.

Note: You do not need a global caslib when you are configuring a publishing destination for SAS Micro Analytic Service.

Important: If you are using SAS Studio 4.4, see “Prerequisites for Using Macros” on page 1.

You can also manage publishing destinations using SAS Environment Manager. For more information, see SAS Viya Administration: Publishing Destinations

Dictionary

%MM_DEFINEPUBLISHDESTINATION Macro

Defines a publish destination for CAS, Hadoop, SAS Micro Analytic Service, or Teradata.

Note: The default SAS Micro Analytic Service destination is named maslocal. This default destination is configured automatically when SAS Model Manager is installed. If the default destination is deleted, you can use this macro to define a new publish destination for the SAS Micro Analytic service.
Syntax

```%MM_DEFINEPUBLISHDESTINATION (  
    BASEURL=%str(host-name:<port>),  
    DEFINITIONNAME=definition-name,  
    EXTTYPE=CAS | HADOOP | MICROANALYTICSERVICE | TERADATA,  
    CASSERVERNAME=CAS-server-name,  
    <CASLIB=CAS-library>,  
    <MODELTABLE=model-table>,  
    <DATABASECASLIB=external-database-caslib>,  
    <HOST=Teradata-host-name>,  
    <PORT=Teradata-port-number>,  
    <SCHEMA=Teradata-schema>,  
    <AUTHDOMAIN=authentication-domain>,  
    <HDFSDIR=HDFS-directory>,  
    <CONFIGDIR=Hadoop-configuration-directory>,  
    <USER=user-name>,  
    <TOKEN=%authorization-token>  
);```

Required Arguments

**BASEURL=%str(host-name:<port>)**

specifies the host name and port for the application server.

**DEFINITIONNAME=definition-name**

specifies the name of the publish destination.

**CASSERVERNAME=CAS-server-name**

specifies the name of the CAS server.

Note Not required for defining a SAS Micro Analytic Service destination.

**CASLIB=CAS-library**

specifies the name of the CAS library.

Note Not required for defining a SAS Micro Analytic Service destination.

See “Create a Global Caslib” in *SAS Viya Administration: Publishing Destinations*

**MODELTABLE=model-table**

specifies the name of the CAS or Teradata model table.

Note Not required for defining a Hadoop or SAS Micro Analytic Service destination.

**EXTTYPE=external-database-type**

specifies the name of the external database type (CAS, HADOOP, MICROANALYTICSERVICE, or TERADATA) for the publish destination.

See “%MM_GET_TOKEN” on page 4
Optional Argument

TOKEN=authorization-token
specifies the authorization token that was generated with the %MM_GET_TOKEN macro.

See “%MM_GET_TOKEN” on page 4

Hadoop Arguments

AUTHDOMAIN=authentication-domain
specifies the authentication domain that is used to retrieve the Hadoop credentials.

See “Create a New Domain” in SAS Viya Administration: External Credentials

HDFSDIR=HDFS-directory
specifies the root HDFS folder where the model directory is to be created.

See “Destination Type Settings” in SAS Viya Administration: Publishing Destinations

CONFIGDIR=configuration-directory
specifies the Hadoop configuration and JAR file directories. Separate the two directory pathnames with a colon (:). These names must match the names that you specified when creating the Hadoop global caslib.

Here is an example:

CONFIGDIR=/hadoopjars/cdh54/prod:/config/clusters/cdh54d3

USER=user-name
specifies the user name for Hadoop.

Interaction This argument is not required if an authentication domain was created using SAS Environment Manager and the AUTHDOMAIN argument is specified in the macro code.

Teradata Arguments

DATABASECASLIB=external-database-caslib
specifies the caslib that contains the external database options.

See “Create a Global Caslib” in SAS Viya Administration: Publishing Destinations

HOST=host-name
specifies the server name for the Teradata database.

Interaction This argument is not required if a DATABASECASLIB argument is specified in the macro code.

PORT=port-number
specifies the port number for the database.

Interaction This argument is not required if a DATABASECASLIB argument is specified in the macro code.
SCHEMA=schema
specifies the connection option that names the Teradata database to use to qualify the Teradata tables.

Interaction This argument is not required if a DATABASECASLIB argument is specified in the macro code.

USER=user-name
specifies the user name for Teradata.

Interaction This argument is not required if a DATABASECASLIB argument is specified in the macro code.

AUTHDOMAIN=authentication-domain
specifies the authentication domain that is used to retrieve the Teradata database credentials.

Interaction This argument is not required if an authentication domain was created and specified as part of the global Teradata caslib using SAS Environment Manager.

See “Create a New Domain” in SAS Viya Administration: External Credentials

Examples

Example 1: Define a Publish Destination for CAS
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token(
   baseURL=&servernm,
   user=&userID,
   pw=&password,
   tokenname=myTokenName
);
%let defname=myDestinationName;
%mm_definepublishdestination(
   baseURL=%str(&servernm),
   definitionname=&defName,
   casservername=cas-shared-default,
   caslib=casuser,
   modeltable=mm_model_table,
   exttype=cas,
   token=%myTokenName
);

Example 2: Define a Publish Destination for Hadoop
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token(
   baseURL=&servernm,
Example 3: Define a Publish Destination for the SAS Micro Analytic Service

```sas
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%mm_get_token(
   baseURL=&servernm,
   user=&userID,
   pw=&password,
   tokenname=myTokenName
);

%let defname=myMAS;
%mm_definepublishdestination(
   baseURL=%str(&servernm),
   definitionname=&defName,
   casservername=cas-shared-default,
   caslib=Public,
   hdfsdir=%str(/tmp/mmtest),
   configdir=%str(/hadoopcfg:/hadoopjars),
   authdomain=myHadoopAuthDomain,
   exttype=hadoop,
   token=%myTokenName
);
```

Example 4: Define a Publish Destination for Teradata

```sas
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%mm_get_token(
   baseURL=&servernm,
   user=&userID,
   pw=&password,
   tokenname=myTokenName
);

%let defname=myTeradata;
%mm_definepublishdestination(
   baseURL=%str(&servernm),
   definitionname=&defName,
   casservername=cas-shared-default,
   caslib=casuser,
   exttype= microAnalyticService,
   token=%myTokenName
);
```
%MM_UPDATEPUBLISHDESTINATION Macro

Updates the publish destination for CAS, Hadoop, or Teradata.

Restriction: Only the authorization token can be updated for a SAS Micro Analytic Service publish destination. You must delete the definition and define a new publish destination if you want to make changes to the other destination arguments.

Syntax

%MM_UPDATEPUBLISHDESTINATION (  
BASEURL=%str(host-name::<port>),  
DEFINITIONNAME=definition-name,  
<TOKEN=%authorization-token>,  
<CASSERVERNAME=CAS-server-name>,  
<CASLIB=CAS-library>,  
<MODELTABLE=model-table>,  
<DATABASECASLIB=external-database-caslib>,  
<HOST=Teradata-host-name>,  
<PORT=Teradata-port-number>,  
<SCHEMA=Teradata-schema>,  
<AUTHDOMAIN=Teradata-authentication-domain>,  
<HDFSDIR=HDFS-directory>,  
<CONFIGDIR=Hadoop-configuration-directory>,  
<User=user-name>  
);  

Required Arguments

BASEURL=%str(host-name::<port>)  
specifies the host name and port for the application server.

DEFINITIONNAME=definition-name  
specifies the name of the publish destination.

CASSERVERNAME=CAS-server-name  
specifies the name of the CAS server.

Note Not required for defining a SAS Micro Analytic Service destination.

CASLIB=CAS-library  
specifies the name of the CAS library.

Note Not required for defining a SAS Micro Analytic Service destination.
See “Create a Global Caslib” in *SAS Viya Administration: Publishing Destinations*

**MODELTABLE=**<code>model-table</code>
specifies the name of the CAS or Teradata model table.

*Note* Not required for defining a Hadoop or SAS Micro Analytic Service destination.

### Optional Argument

**TOKEN=**<code>%authorization-token</code>
specifies the authorization token that was generated with the `%MM_GET_TOKEN` macro.

See “%MM_GET_TOKEN” on page 4

### Hadoop Arguments

**AUTHDOMAIN=**<code>authentication-domain</code>
specifies the authentication domain that is used to retrieve the Hadoop credentials.

See “Create a New Domain” in *SAS Viya Administration: External Credentials*

**HDFSDIR=**<code>HDFS-directory</code>
specifies the root HDFS folder where the model directory is to be created.

See “Destination Type Settings ” in *SAS Viya Administration: Publishing Destinations*

**CONFIGDIR=**<code>configuration-directory</code>
specifies the Hadoop configuration and JAR file directories. Separate the two directory pathnames with a colon (:). These names must match the names that you specified when creating the Hadoop global caslib.

Here is an example:

```
CONFIGDIR=/hadoopjars/cdh54/prod:/config/clusters/cdh54d3
```

**USER=**<code>user-name</code>
specifies the user name for Hadoop.

*Interaction* This argument is not required if an authentication domain was created using SAS Environment Manager and the AUTHDOMAIN argument is specified in the macro code.

See “Create a New Domain” in *SAS Viya Administration: External Credentials*

### Teradata Arguments

**DATABASECASLIB=**<code>external-database-caslib</code>
specifies the caslib that contains the external database options.

See “Create a Global Caslib” in *SAS Viya Administration: Publishing Destinations*
**HOST=host-name**  
specifies the server name for the Teradata database.  
**Interaction** This argument is not required if a DATABASECASLIB argument is specified in the macro code.  

**PORT=port-number**  
specifies the port number for the database.  
**Interaction** This argument is not required if a DATABASECASLIB argument is specified in the macro code.  

**SCHEMA=schema**  
specifies the connection option that names the Teradata database to use to qualify the Teradata tables.  
**Interaction** This argument is not required if a DATABASECASLIB argument is specified in the macro code.  

**USER=user-name**  
specifies the user name for Teradata.  
**Interaction** This argument is not required if a DATABASECASLIB argument is specified in the macro code.  

**AUTHDOMAIN=authentication-domain**  
specifies the authentication domain that is used to retrieve the Teradata database credentials.  
**Interaction** This argument is not required if an authentication domain was created and specified as part of the global Teradata caslib using SAS Environment Manager.  

**See**  
“Create a New Domain” in *SAS Viya Administration: External Credentials*  

### Examples

#### Example 1: Update the Definition for a CAS Publish Destination

```sas
%let servernm=http://myserver.com;  
%let userID=myUserID;  
%let password=myPassword;  

%mm_get_token(  
    baseURL=&servernm,  
    user=&userID,  
    pw=&password,  
    tokenname=myTokenName  
);  

%let defname=myCASServer;  
%mm_updatepublishdestination(  
    baseURL=%str(&servernm),  
    definitionname=&defName,  
    casservername=newcas,  
    caslib=casuser,  
    ...  
);  
```
Example 2: Update the Definition for a Hadoop Publish Destination

```sas
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);
%let defname=myHadoopServer;
%mm_updatepublishdestination(
    baseURL=%str(&servernm),
    definitionname=&defname,
    casservername=cas-shared-default,
    caslib=Public,
    hdfsdir=%str(/tmp/myHDFSdir),
    configdir=%str(/sasusr/u/hadoopcfg:
        /sasusr/u/hadoopjars),
    authdomain=myHadoopAuthDomain,
    token=%myTokenName
);
```

Example 3: Update the Definition for a Teradata Publish Destination

```sas
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);
%let defname=myTeradata;
%mm_updatepublishdestination(
    baseURL=%str(&servernm),
    definitionname=&defname,
    casservername=cas-shared-default,
    caslib= Teradata,
    modeltable=mm_model_table,
    databasecaslib=myTDLib,
    token=%myTokenName
);
```

%MM_PRINTPUBLISHDESTINATION Macro

Prints a list of publish destination definitions.
Syntax

```bash
%MM_PRINTPUBLISHDESTINATION(
    BASEURL=%str(host-name:<port>),
    DEFINITIONNAME=definition-name,
    LIMIT=limit
    <TOKEN=%authorization-token>
);
```

**Required Arguments**

- **BASEURL=%str(host-name:<port>)** specifies the host name and port for the application server.
- **DEFINITIONNAME=definition-name** specifies the name of the publish destination.
- **LIMIT=limit** specifies how many destination definitions to return. If you do not specify a value for the **DEFINITIONNAME** argument, the macro returns file destination definitions by default.

**Optional Argument**

- **TOKEN=%authorization-token** specifies the authorization token that was generated with the `%MM_GET_TOKEN` macro.

See “%MM_GET_TOKEN” on page 4

**Examples**

**Example 1: Print One Publish Destination**

```bash
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;
%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);
%let defname=myDefinitionName;
%mm_printpublishdestination(
    baseURL=%str(&servernm),
    definitionname=%str(&defName),
    token=%myTokenName
);
```

**Example 2: Print Multiple Publish Destinations**

```bash
%let servernm=http://myserver.com;
```

%let userID=myUserID;
%let password=myPassword;

%mm_get_token(
    baseURL=&servernm,
    user=&userID,
    pw=&password,
    tokenname=myTokenName
);

%mm_printpublishdestination(
    baseURL=%str(&servernm),
    limit=8,
    token=%myTokenName,
    resp=pubsvrs,
    keepresp=Y
);

%mm_read_json(
    fref=pubsvrs,
    respType=GET_ITEMS,
    outds=dests,
    statusVar=_status
);

proc print data=dests noobs;
    var name destinationType destinationTable casServerName casLibrary;
run;

%MM_DELETEPUBLISHDESTINATION Macro

Deletes a publish destination definition.

Syntax

%MM_DELETEPUBLISHDESTINATION (  
    BASEURL=%str(host-name<:port>),  
    DEFINITIONNAME=definition-name,  
    <TOKEN=%authorization-token>
);

Required Arguments

BASEURL=%str(host-name<:port> )  
    specifies the host name and port for the application server.

DEFINITIONNAME=definition-name  
    specifies the name of the publish destination.

Optional Argument

TOKEN=%authorization-token  
    specifies the authorization token that was generated with the %MM_GET_TOKEN macro.
See “%MM_GET_TOKEN” on page 4

Example: Delete a Publish Destination Definition

```
%let servernm=http://myserver.com;
%let userID=myUserID;
%let password=myPassword;

%mm_get_token(
    baseURL=&servernm,
    user=& userID,
    pw=& password,
    tokenname=myTokenName
);

%let defname=myDefinitionName;
%mm_deletepublishdestination(
    baseURL=%str(&servernm),
    definitionname=%str(&defName),
    token=%myTokenName
);
```
Chapter 5
Feature Contribution Index Macros

Overview of FCI Macros

What Are FCI Macros?
The feature contribution index (FCI) measures the relationships between input and output variables. FCI macros enable you to compute the feature contribution indices for interval and nominal predictors, and return the indices in the output data. The output data is then used to create a report. The FCI macros are delivered with SAS Model Manager 15.3 on SAS Viya 3.5. The scoring data is stored in a CAS table. The feature contribution index macros use the CAS scoring data to compute the FCI in a CAS session, and then the output is returned in a CAS table. Before you use these macros, the model outcome must already be available. That is, the input data set must be scored using the model first, and then the model outcomes must be saved in the scoring data set. This document contains the syntax and argument descriptions, as well as examples for the macros.

Note: The data sets and code examples can be found in the SASViya_FCI_Macro_Examples.zip file. The ZIP file is available from SAS Model Manager Product Documentation page on support.sas.com.

Measuring Predictor Influence
When you train a model, you can evaluate the importance of predictors within that model. Some training algorithms (for example, decision trees) provide variable
importance indices. Alternatively, they can be calculated using statistics on predictors (for example, multiplying \(-1\) by the logarithm of a predictor’s significance value from a regression model). When you deploy a model, you might want to determine how much influence a predictor has on model outcomes. Because the distributions of predictors in a scoring data set might differ from those found in the training data set, the original variable importance metrics obtained from training the model might no longer be relevant.

**Scoring Details**

Therefore, the procedures used to compute the original variable importance metrics might not work for scoring because the observed target variables that are required by the procedures are not available in the scoring data set. If you face these constraints on the scoring data set, here are possible solutions to help you meet your users’ needs. When you follow these procedures, you do not need to wait for the observed target values to be available. Instead, you can use the predicted values for the interval target variable or the predicted probabilities for the nominal target variable. Instead of customizing procedures for different models, compute the FCI, a model-neutral procedure.

**How the Model Outcome Is Determined**

The contribution of a predictor or a feature of a model is defined as the aggregated influence of that predictor’s values on the spread of the model outcome. For classification models (nominal target variables), the model outcome consists of the predicted probabilities. For regression models (interval target variables), the model outcome is the predicted value. In both types of models, the model outcome consists of one or more numeric values. To measure the contribution, use the following procedure:

1. For each numeric value in the model outcome, build the main effect analysis of variance with each individual predictor.

2. Measure the contribution of a predictor by the R-squared statistic. (For nominal predictors, this is the full eta-squared statistic. For interval predictors, this is the squared Pearson correlation coefficient.)

**Note:** For nominal targets, the contribution indices are aggregated for each individual predicted probability.

The index is a numeric value between 0 and 1, inclusive. A value of 1 indicates that the variable contributes the most to the model outcomes, and it is probably the only variable needed for determining the model outcomes. A value of 0 indicates that the variable contributes the least to the model outcomes, and its absence would have little or no impact on the model.

**Aggregation for Nominal Targets**

There is one FCI for each predicted probability. In order to provide a single index to users, calculate a weighted sum of the individual FCIs.

Here are two common options for weights:

- Each weight is equivalent to the reciprocal of the number of predicted probabilities (uninformative).
Weights are equivalent to the observed relative frequencies (proportions) of the target categories in the training data set.

For a binary target variable, the choice of weight should not matter because the contribution of a predictor to either predicted probability is the same.

---

**Dictionary**

---

**%Compute_FCI_IntPred Macro**

Computes FCIs for a list of interval predictors. It appends FCIs to the output FCI data set. This macro primarily invokes CAS actions using the CAS procedure. It also uses the DATA step statement.

---

**Syntax**

```
%Compute_FCI_IntPred ( 
   InCASLibRef=\textit{input-CAS-library-reference}, 
   OutCASLibRef=\textit{output-CAS-library-reference}, 
   InData=\textit{scoring-dataset-name}, 
   DepVarList=\textit{list-of-variable-names}, 
   IntPredList=\textit{list-of-interval-predictors}, 
   OutFCIData=\textit{output-FCI-dataset}, 
   <Debug=Y | N>, 
); 
```

**Required Arguments**

- **InCASLibRef=\textit{input-CAS-library-reference}**
  
  specifies the name of the input CAS library reference.

- **OutputCASLibRef=\textit{output-CAS-library-reference}**
  
  specifies the name of the output CAS library reference.

- **InData=\textit{scoring-dataset-name}**
  
  specifies the name of the scoring data set.

- **DepVarList=\textit{list-of-variable-names}**
  
  contains a list of names of the numeric model output, separated by blanks.

- **IntPredList=\textit{list-of-interval-predictors}**
  
  contains a list of names of interval predictors, separated by blanks.

- **OutFCIData=\textit{output-FCI-dataset}**
  
  specifies the name of the output FCI data set.

**Optional Argument**

- **Debug**
  
  indicates whether to display debugging information. The default value is \texttt{N}.
%Compute_FCI_NomPred Macro

Computes FCIs for a list of nominal predictors. It appends FCIs to the output FCI data set. This macro primarily invokes CAS actions using the CAS procedure. It also uses the DATA step statement.

Syntax

```plaintext
%Compute_FCI_NomPred (  
  InCASLibRef=input-CAS-library-reference,  
  OutCASLibRef=output-CAS-library-reference,  
  InData=scoring-dataset-name,  
  DepVarList=list-of-variable-names,  
  NomPredList=list-of-nominal-predictors,  
  OutFCIData=output-FCI-dataset,  
  <qMissNom=Y | N>,  
  <Debug=Y | N>,  
);  
```

Required Arguments

- **InCASLibRef=** _input-CAS-library-reference_ specifies the name of the input CAS library reference.
- **OutputCASLibRef=** _output-CAS-library-reference_ specifies the name of the output CAS library reference.
- **InData=** _scoring-dataset-name_ specifies the name of the scoring data set.
- **DepVarList=** _list-of-variable-names_ lists the names of the numeric model output, separated by blanks.
- **NomPredList=** _list-of-nominal-predictors_ lists the names of nominal predictors, separated by blanks.
- **OutFCIData=** _output-FCI-dataset_ specifies the name of the output FCI data set.

Optional Arguments

- **qMissNom** indicates whether to include missing values in nominal predictors. The default value is `N`.
- **WorkLib=** _work-library-reference_ specifies the name of the working library reference. The default library is `WORK`.
- **Debug** indicates whether to display debugging information. The default value is `N`.
%Create_FCI_Report Macro

Calls the %Compute_FCI_NomPred and the %Compute_FCI_IntPred macros to compute the FCIs’ given input specifications. It overwrites the output FCI data set with the FCIs. Besides using a DATA step statement, this macro calls the CAS procedure.

Syntax

%Create_FCI_Report (  
   InCASLib=input-CAS-library,  
   OutCASLib=output-CAS-library,  
   MDataCASLib=monitoring-CAS-library,  
   MonitorDataSpec=monitoring-specification-dataset,  
   TargetSpec=monitoring-specification-dataset,  
   PredictorSpec=predictor-specification-dataset,  
   OutFCIData=output-FCI-dataset,  
   <PanelColumns=number-of-panel-columns>,  
   <PanelRows=number-of-panel-rows>,  
   <Debug=Y | N>,  
 );

Required Arguments

InCASLib=input-CAS-library-reference  
   specifies the name of the CAS library that contains the input specification data sets.

OutputCASLib=output-CAS-library  
   specifies the name of the CAS library that contains the output data sets.

MDataCASLib=monitoring-CAS-library  
   specifies the name of the CAS library that contains the monitoring data specification data set.

TargetSpec=monitoring-specification-dataset  
   specifies the name of the data set that contains the monitoring data specifications.

TargetSpec=target-specification-dataset  
   specifies the name of the data set that contains the target specifications.

PredictorSpec=predictor-specification-dataset  
   specifies the name of the data set that contains the predictor specifications.

OutFCIData=output-FCI-dataset  
   specifies the name of the output FCI data set.

Optional Arguments

PanelColumns=number-of-panel-columns  
   specifies the number of columns in the panel chart.

PanelRows=number-of-panel-rows  
   specifies the number of rows in the panel chart.
Debug
indicates whether to display debugging information. The default value is N.

Details
The following data sets are associated with the %Create_FCI_Report macro: monitoring specification, target specification, predictor specification, and output FCI.

The monitoring specification data set has as many rows as the number of monitoring data sets. The %Create_FCI_Report macro looks for the following variables in the data set.

Table 5.1 Monitoring Specification Data Set

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATANAME</td>
<td>String; maximum of 32 characters</td>
<td>SAS Name</td>
<td>Specifies the name of a CAS table that already contains the model outcomes.</td>
</tr>
<tr>
<td>DISPLAYNAME</td>
<td>String; maximum of 32 characters</td>
<td>String</td>
<td>Specifies the label of the monitoring sequence.</td>
</tr>
</tbody>
</table>

The target specification data set has as many rows as the number of model outcomes. The %Create_FCI_Report macro looks for the following variables in the data set.

Table 5.2 Target Specification Data Set

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>String; maximum of 32 characters</td>
<td>SAS Name</td>
<td>Specifies the name of a model outcome.</td>
</tr>
<tr>
<td>PRIOR</td>
<td>Numeric</td>
<td>Number</td>
<td>Specifies the weight for calculating the weighted sum of the individual FCIs.</td>
</tr>
</tbody>
</table>

The predictor specification data set has as many rows as the number of predictors. The %Create_FCI_Report macro looks for the following variables in the data set.

Table 5.3 Predictor Specification Data Set

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>String; maximum of 32 characters</td>
<td>SAS Name</td>
<td>Specifies the name of a predictor.</td>
</tr>
<tr>
<td>LEVEL</td>
<td>String; maximum of 8 characters</td>
<td>INTERVAL or NOMINAL</td>
<td>Specifies the level for the predictor.</td>
</tr>
</tbody>
</table>
QMISSNOM | String; maximum of 1 character | N or Y | Specifies whether to include missing values in nominal predictors. This value is ignored if LEVEL is INTERVAL.

NLEVEL | Numeric | Positive integer | Specifies the number of levels in nominal predictors. This value is ignored if LEVEL is INTERVAL.

The output FCI data set has as many rows as the number of predictors. The following variables are in this data set.

**Table 5.4 Output FCI Data Set**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>VARNAME</em></td>
<td>String; maximum of 32 characters</td>
<td>SAS Name</td>
<td>Specifies the name of a predictor.</td>
</tr>
<tr>
<td>User-supplied name</td>
<td>Numeric</td>
<td>Between 0 and 1, inclusive</td>
<td>Users supply a name for the aggregated (weighted sum) FCI.</td>
</tr>
</tbody>
</table>

Besides writing a data set of feature contribution indices, the %Create_FCI_Report macro also displays the following information in the report results:

- A table of the feature contribution indices. The rows are the predictors and the columns are the labels of the monitoring data sets.

- A stacked line chart of the indices. The indices are plotted against the predictors for each label of the monitoring data set. The chart shows if there are substantial changes.

- A panel chart of the indices includes the control limits. The indices and their control limits are plotted against the labels of the monitoring data sets for each predictor.

The control limits are constructed as the 95% confidence limits for the eta-squared statistics by applying the interval inversion method (Kromrey and Bell 2010 and Steiger 2004). The limits are computed based on the number of levels of the predictor (one for an interval predictor), the number of observations, and the eta-squared statistic. The charts are ordered in descending order of the indices of the reference monitoring data set. The tick marks on the vertical axis are rebased such that the indices of the reference monitoring data sets are shown as zeros. Attention should be paid to a predictor when there is a trend of systemic deviations of indices from that of the reference monitoring data set.

You can use the feature contribution indices to monitor the model performance of several data sets. The number of observations is computed for each monitoring data set, the number of levels of the predictor is read from the predictor specification table, and the eta-squared is fixed at that computed from the first monitoring data set, which is also referred to as the reference data set.
Example: Running FCI Macros Using SAS Studio

Details
The %Create_FCI_Report macro assists in calculating FCIs. In addition to specifying the data set on which a model is deployed, this macro requires a specification data set for the target variable and another for the predictors. The target specification data set should have two columns (NAME and PRIOR), and as many rows as the number of levels of the target variable (it is assumed that an interval target variable has a single level). The predictor specification data set should have three columns (NAME, LEVEL, and QMISSNOM), and as many rows as the number of predictors. For more information, see “%Create_FCI_Report Macro” on page 61.

To run this example:
1. Extract the data sets and code examples from the SASViya_FCI_Macro_Examples.zip file. The ZIP file is available at SAS Model Manager Product Documentation page on support.sas.com.
2. Sign in to SAS Studio. Here is an example of the URL:
   http://myserver.com/SASStudio
3. Import and load the FCI data tables into the PUBLIC caslib.
   For more information, see “Making Data Available to CAS” in SAS Data Explorer: User's Guide.
4. (Optional) Set the Debug argument to Y, to request the debugging information from the macros.

Compute FCIs and Create Report
This code example is from the file Policy_MonitorModel_CLAIM_FLAG_IN.sas file. The code examples are located in the /SASViya_FCI_Macro_Examples/FCI/Examples directory of the ZIP file that you extracted. The code examples use the same data sets.

```sas
%let FCIFolder = %str(~/SASViya/PCI);

cas _thisCAS;
caslib _ALL_ assign;

/* Requirements: */
1. A target variable.
2. Some interval and/or nominal predictors.
*/

/* Specify the target variable. */
%let TargetVar = %str(CLAIM_FLAG);

/* Specify the interval predictors. */
%macro IntPredList;
BLUEBOOK_1000
```
CUS'T__LOYALTY
HOMEKIDS
KIDSDRV
MVR_STS
TIF
TRAIVTIME
YOJ
%mend IntPredList;

%macro NomPredList;
AGE_BAND
CAR_USE
CREDIT_SCORE_BAND
REVOKED
URBANICITY
%mend NomPredList;

/* Create the scores from the CART model. */
proc cas;
  action table.dropTable / caslib = "CASUSER" name = "TargetSpec" quiet = True;
  run;

  action table.dropTable / caslib = "CASUSER" name = "PredictorSpec" quiet = True;
  run;
quit;

filename _MDLSCR_ "&FCIFolder./Examples/Policy_CART.sas";
%include "&FCIFolder./Examples/Build_DecoisionTree.sas";
%Build_DecoisionTree (  
  InCASLibRef = PUBLIC,
  OutCASLibRef = CASUSER,
  TrainData = policy_0_1995TO1997,
  TargetVar = &TargetVar.,
  TargetType = NOMINAL,
  IntPredList = %IntPredList,
  NomPredList = %NomPredList,
  qMissNom = Y,
  TargetSpec = TargetSpec,
  PredictorSpec = PredictorSpec,
  FRefScoreCode = _MDLSCR_,
  Debug = N);
proc cas;
  action table.fetch / 
    table = {caslib = "CASUSER" name = "TargetSpec"}
    from = 1 to = 2147483647 maxRows = 2147483647;
  run;

  action table.fetch / 
    table = {caslib = "CASUSER" name = "PredictorSpec"}
    from = 1 to = 2147483647 maxRows = 2147483647;
  run;
quit;

data CASUSER.policy_0_1995TO1997_score;
set PUBLIC.policy_0_1995TO1997;
%include _MDLSCR_;
run;

data CASUSER.policy_1_1998_score;
  set PUBLIC.policy_1_1998;
  %include _MDLSCR_;
run;

data CASUSER.policy_2_1999_score;
  set PUBLIC.policy_2_1999;
  %include _MDLSCR_;
run;

data CASUSER.policy_3_2000_score;
  set PUBLIC.policy_3_2000;
  %include _MDLSCR_;
run;

data CASUSER.policy_4_2001_score;
  set PUBLIC.policy_4_2001;
  %include _MDLSCR_;
run;

/* The policy_0_1995TO1997_score is the reference baseline, and the other data sets are compared with it. */
data CASUSER.MonitorDataSpec;
  length DataName $ 32;
  length DisplayName $ 32;

  DataName = 'policy_0_1995TO1997_score';
  DisplayName = '1995 to 1997';
  output;

  DataName = 'policy_1_1998_score';
  DisplayName = '1998';
  output;

  DataName = 'policy_2_1999_score';
  DisplayName = '1999';
  output;

  DataName = 'policy_3_2000_score';
  DisplayName = '2000';
  output;

  DataName = 'policy_4_2001_score';
  DisplayName = '2001';
  output;
run;

ods graphics / reset width = 10in height = 10in;

%Create_FCI_Report
{
InCASLib = CASUSER; /* CAS library of the input specification data sets. */
OutCASLib = CASUSER; /* CAS library of the output data sets. */
MDataCASLib = CASUSER; /* CAS library of the monitoring data sets. */
MonitorDataSpec = MonitorDataSpec; /* Monitoring data specification data set. */
TargetSpec = TargetSpec; /* Target specification data set. */
PredictorSpec = PredictorSpec; /* Predictor specification data set. */
OutFCIData = OutFCIData; /* Output feature contribution index data. */
Debug = N /* Display debugging information (Y/N)? */
);

proc cas;
  action table.fetch / table = {caslib = "CASUSER" name = "OutFCIData"};
  run;

  action table.dropTable / caslib = "CASUSER" name = "policy_0_1995TO1997_score" quiet = True;
  run;

  action table.dropTable / caslib = "CASUSER" name = "policy_1_1998_score" quiet = True;
  run;

  action table.dropTable / caslib = "CASUSER" name = "policy_2_1999_score" quiet = True;
  run;

  action table.dropTable / caslib = "CASUSER" name = "policy_3_2000_score" quiet = True;
  run;

  action table.dropTable / caslib = "CASUSER" name = "policy_4_2001_score" quiet = True;
  run;

quit;

Here are the output results. They include the observations for the predictor specification data set and for the target specification data set, the measurement level for the interval predictors and nominal predictors, and the calculated FCIs.

Output 5.1 Target Specification Data Set

<table>
<thead>
<tr>
<th><em>Index</em></th>
<th>NAME</th>
<th>PRIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PCLAIM_FLAG1</td>
<td>0.25220032569</td>
</tr>
<tr>
<td>2</td>
<td>PCLAIM_FLAG2</td>
<td>0.7487318458</td>
</tr>
</tbody>
</table>
Output 5.2  Predictor Specification Data Set

```
Selected Rows from Table PREDICTORSPEC

<table>
<thead>
<tr>
<th><em>index</em></th>
<th>NAME</th>
<th>LEVEL</th>
<th>QMISSMISS</th>
<th>NLEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLUEBOOK_1000</td>
<td>INTERVAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CUST_LOYALTY</td>
<td>INTERVAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HOMEXIDS</td>
<td>INTERVAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>KIDSDRIV</td>
<td>INTERVAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MVR_PTS</td>
<td>INTERVAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TIF</td>
<td>INTERVAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRAVTIME</td>
<td>INTERVAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YOJ</td>
<td>INTERVAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AGE_BAND</td>
<td>NOMINAL</td>
<td>Y</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>CAR_USE</td>
<td>NOMINAL</td>
<td>Y</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>CREDIT_SCORE_BAND</td>
<td>NOMINAL</td>
<td>Y</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>REVOKED</td>
<td>NOMINAL</td>
<td>Y</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>URBANITY</td>
<td>NOMINAL</td>
<td>Y</td>
<td>2</td>
</tr>
</tbody>
</table>
```

Output 5.3  FCI Measurement Levels for Internal and Nominal Predictors

```
Feature Contribution Indices

Results from table.fcteh

Selected Rows from Table FCIDATA_DISPLAY

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>INTERVAL</td>
<td>BLUEBOOK_1000</td>
<td>1.5%</td>
<td>2.0%</td>
<td>1.8%</td>
<td>1.6%</td>
</tr>
<tr>
<td></td>
<td>INTERVAL</td>
<td>CUST_LOYALTY</td>
<td>0.1%</td>
<td>0.0%</td>
<td>0.1%</td>
<td>0.4%</td>
</tr>
<tr>
<td></td>
<td>INTERVAL</td>
<td>HOMEXIDS</td>
<td>5.0%</td>
<td>5.4%</td>
<td>3.8%</td>
<td>5.4%</td>
</tr>
<tr>
<td></td>
<td>INTERVAL</td>
<td>KIDSDRIV</td>
<td>1.3%</td>
<td>0.4%</td>
<td>0.6%</td>
<td>0.7%</td>
</tr>
<tr>
<td></td>
<td>INTERVAL</td>
<td>MVR_PTS</td>
<td>17.1%</td>
<td>14.0%</td>
<td>16.1%</td>
<td>18.7%</td>
</tr>
<tr>
<td></td>
<td>INTERVAL</td>
<td>TIF</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td>INTERVAL</td>
<td>TRAVTIME</td>
<td>0.0%</td>
<td>0.1%</td>
<td>0.4%</td>
<td>0.1%</td>
</tr>
<tr>
<td></td>
<td>INTERVAL</td>
<td>YOJ</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.1%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td>NOMINAL</td>
<td>AGE_BAND</td>
<td>3.0%</td>
<td>3.0%</td>
<td>3.0%</td>
<td>3.2%</td>
</tr>
<tr>
<td></td>
<td>NOMINAL</td>
<td>CAR_USE</td>
<td>9.9%</td>
<td>0.5%</td>
<td>9.0%</td>
<td>10.0%</td>
</tr>
<tr>
<td></td>
<td>NOMINAL</td>
<td>CREDIT_SCORE_BAND</td>
<td>0.1%</td>
<td>0.8%</td>
<td>0.2%</td>
<td>0.5%</td>
</tr>
<tr>
<td></td>
<td>NOMINAL</td>
<td>REVOKED</td>
<td>3.7%</td>
<td>3.4%</td>
<td>3.6%</td>
<td>3.9%</td>
</tr>
<tr>
<td></td>
<td>NOMINAL</td>
<td>URBANITY</td>
<td>28.9%</td>
<td>31.9%</td>
<td>31.0%</td>
<td>31.0%</td>
</tr>
</tbody>
</table>
```
Output 5.4  FCI Plotted against Predictors for Each Monitoring Time
Output 5.5  FCI Plotted against the Monitoring Time for Each Predictor
Example 1: Running FCI Macros Using SAS Studio

Deviation From Reference Value (in Parentheses) Over Time with 95% Confidence Limits
(Variables Shown in Descending Level of First Indices)

Data Label
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore for additional books and resources.