## Contents

**Chapter 1 • Introduction** ................................................................. 1  
Overview ................................................................................. 1

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

**Chapter 3 • Performance Monitoring Macros** ........................................ 27  
Overview of Performance Monitoring Macros ................................ 27  
Macro Variables .......................................................................... 28  
Dictionary ................................................................................ 31

**Chapter 4 • Publish Destination Macros** ............................................. 35  
Overview of Publish Destination Macros ....................................... 35  
Dictionary ................................................................................ 35
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 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
Chapter 1 • Introduction
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 a common model repository. 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.

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 authentication token, contact your SAS application administrator.

TIP When passing UUIDs in as arguments, it is a good practice to wrap %STR() around the UUID values or macro variables.
Dictionary

%MM_GET_TOKEN

Generates an authentication 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=authentication-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=authentication-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 authentication token name must be specified with the TOKEN= parameter for the publish destination and model management macros.

Example

Example Code 2.1  Set Macro Variables

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

Example Code 2.2  Get Authorization Token

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

Adds a repository folder to the common model repository.

Syntax

%MM_CREATE_REPOSITORY (  
    REPOSITORYNM =repository-name,  
    <REPOSITORYDESC =repository-description>,  
    <REPOSFOLDERID =repository-folder-ID>,  
    SERVERNM =host-name:<port>,  
    TOKEN =token-name,  
    <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.

TOKEN=token_name  
specifies the name of the authentication token that was generated with the %MM_GET_TOKEN macro.

See “%MM_GET_TOKEN” on page 4

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 folder 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”.

Example

%let servernm=http://myserver.com;  
%let userID=myUserID;  
%let password=myPassword;  
%mm_get_token{
%MM_CREATE_FOLDER Macro

Adds a folder to a repository.

Syntax

%MM_CREATE_FOLDER (  
   FOLDERNM =folder-name,  
   REPOSFOLDERID =repository-folder-ID,  
   SERVERNM=host-name:<port>,  
   TOKEN=token-name,  
   <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 folder 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.

TOKEN=token_name
   specifies the name of the authentication token that was generated with the %MM_GET_TOKEN macro.

See “%MM_GET_TOKEN” on page 4

Optional Argument

FOLDERID =folder-ID-macro-variable
   specifies the macro variable to assign to the folder ID. A folder ID is assigned by the folder 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>,  
    TOKEN =token-name,  
    <PROJECTID =project-ID-macro-variable>,  
    <PROJECTVERSIONID =project-version-ID-macro-variable>,  
);
```

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 folder 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.

**TOKEN=token_name**

specifies the name of the authentication token that was generated with the %MM_GET_TOKEN macro.
Optional Arguments

**PROJECTID =** *project-ID*

specifies the macro variable to assign to the project ID that is created. A UUID assigned by the folder 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 folder 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 “Types of Model Functions” in *SAS Model Manager: User’s Guide*.

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_project(
    projectnm = MyProject,
    folderID = &folderID,
    function = Classification,
    servernm = &servernm,
    token = %myTokenname,
    projectID = projID,
    projectversionID = projVerID
);
```

---

**%MM_CREATE_PROJECTVERSION Macro**

Creates a version within a project.

**Syntax**

```sas
%MM_CREATE_PROJECTVERSION ( 
    PROJECTID = *project-ID*,
    <PROJECTVERSIONNM = *project-version-name*>,
    <PROJECTVERSIONDESC = *project-version-description*>,
    SERVERNM = *host-name:*<port>*,
    TOKEN = *token-name*,
```
<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.

**TOKEN = token_name**

specifies the name of the authentication token that was generated with the %MM_GET_TOKEN macro.

See “%MM_GET_TOKEN” on page 4

**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 folder service. If this argument is not included, the default macro variable name that is created is "_projectVersionID".

**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(
    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

```sas
%MIMPORT_MODEL (  
   MODELNM =model-name,  
   <MODELDESC =model-description>,  
   MODELFUNC =model-function-name,  
   MODELLOC =model-location,  
   <FILESIZEOVERRIDE=filesize-override-flag>,  
   <PROJECTID =project-ID>,  
   <PROJECTVERSIONID =project-version-ID>,  
   <FOLDERID =folder-ID>,  
   IMPORTINTO =import-into-object,  
   SERVERNM =host-name:<port>,  
   TOKEN =token-name,  
   <MODELID =model-ID-macro-variable>,  
 );
```

Required Arguments

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

- `MODELFUNC =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.

- `MODELLOC =model-location`
  specifies the location from which the model is to be imported.

- `IMPORTINTO =import-into-object`
  specifies the location to which the model is to be imported into. The default is the project version when importing models into a project.

- `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.

- `TOKEN =token_name`
  specifies the name of the authentication token that was generated with the `%MM_GET_TOKEN` macro.

  See “%MM_GET_TOKEN” on page 4

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 folder service. If this argument is not included, the default macro variable name that is created is "_modelID".
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

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

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
);

%let model2 = /home/models/model2/MyModel.zip;
%let type=ZIP;
%mm_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**

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

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

%let model3 = /home/models/model3/neural.xml;
%mm_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**

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

%mm_get_token(
```
baseURL=&servernm,
user=&userID,
pw=&password,
tokenname=myTokenName
);
%let model4 = /home/models/model4/hmeq_svm_output.sasast;
%mm_import_model(
    modelnm          = HMEQ SVM Analytic Store,
    modeldesc        = Analytic store model,
    modelfunc        = ,
    modelloc         = &model4,
    filesizeoverride = N,
    projectID        = %str(&projID),
    importinto       = project,
    servernm         = &servernm,
    token            = %myTokenname,
    modelID          = myModelID
);

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

**Syntax**

%MM_GET_REPOSITORY_ID (  
    REPOSITORYNM =repository-name,
    IDVAR=repository-ID-macro-variable,
    <FLDRIDVAR =repository-folder-ID-macro-variable>,
    SERVERNM =host-name:<port>,
    TOKEN =token-name,
);

**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.

TOKEN=token_name  
    specifies the name of the authentication token that was generated with the %MM_GET_TOKEN macro.

See “%MM_GET_TOKEN” on page 4

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

**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".

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**

```sas
%MM_GET_FOLDER_ID (  
    FOLDERNM = *folder-name*,  
    IDV AR = *folder-ID-macro-variable*,  
    SERVERNM = *host-name:*<port> ,  
    TOKEN = *token-name*,  
);  
```

**Required Arguments**

**FOLDERNM** = *folder-name*

specifies the name of the folder.

**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.

**TOKEN** = *token_name*

specifies the name of the authentication token that was generated with the %MM_GET_TOKEN macro.
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_folder_id(
   foldernm=MyFolder,
   idvar=myFldrID,
   servernm=&servernm,
   token=%myTokenname
);
```

%**MM_GET_PROJECT_ID**

Retrieves the project ID for the specified project name.

**Syntax**

```sas
%MGET_PROJECT_ID (    PROJECTNM =project-name,    IDVAR=project-ID-macro-variable,    SERVERNM =host-name:<port>,    TOKEN =token-name,    );
```

**Required Arguments**

**PROJECTNM =project-name**

specifies the name of the project.

**IDVAR=project-ID-macro-variable**

specifies the macro variable to assign to the project 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.

**TOKEN=token_name**

specifies the name of the authentication token that was generated with the %**MM_GET_TOKEN** macro.

See “%**MM_GET_TOKEN**” on page 4
Example

```sas
%mm_get_project_id(
   projectnm=MyProject,
   idvar=myProjID,
   servernm=&servernm,
   token=%myTokenname
);
```

%MM_GET_PROJECTVERSION_ID

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

Syntax

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

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.

**TOKEN =token_name**  
specifies the name of the authentication token that was generated with the %MM_GET_TOKEN macro.

See “%MM_GET_TOKEN” on page 4

Optional Argument

**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.

Example

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

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

### 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.

- **TOKEN**: `token_name`
  - Specifies the name of the authentication token that was generated with the `%MM_GET_TOKEN` macro.

See “%MM_GET_TOKEN” on page 4

### Example

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

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

Syntax

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

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.

TOKEN=token_name
  specifies the name of the authentication token that was generated with the  
  %MM_GET_TOKEN macro.

See  “%MM_GET_TOKEN” on page 4

Example

%let servernm=http://myserver.com;  
%let userID=myUserID;  
%let password=myPassword;
%MM_DELETE_FOLDER
Deletes the folder that is associated with the specified folder name or ID.

Syntax

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

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.

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.

TOKEN = token_name
specifies the name of the authentication token that was generated with the  
%MM_GET_TOKEN macro.

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 =token-name,
  );

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.

TOKEN=token_name
    specifies the name of the authentication token that was generated with the
    %MM_GET_TOKEN macro.

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_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

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

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.

`TOKEN=token_name`

specifies the name of the authentication token that was generated with the `%MM_GET_TOKEN` macro.

See “%MM_GET_TOKEN” on page 4
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.

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,
  token=%myTokenname
);
```

%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 =token-name, );```
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.

TOKEN=token_name
specifies the name of the authentication token that was generated with the %MM_GET_TOKEN macro.

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_model{
   /* Either projectversionnm or projectversionID can be used.*/
   modelnm=MyModel&type,
   /* modelID=%str(&myModelID), */
   serverNm=&servernm,
   token=%myTokenname
};

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

Syntax

%MM_PUBLISH_MODEL (  
   MODELNM =model-name,  
   <MODELNOTES=model-publish-notes>,  
   <MODELID=model-ID>,  
   <MODELINFOFILEREF=model-info-filerref>,  

Required Arguments

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

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.

TOKEN = token_name
  specifies the name of the authentication token that was generated with the
  %MM_GET_TOKEN macro.

See “%MM_GET_TOKEN” on page 4

Optional Arguments

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

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

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 File service.

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

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()`.

### Example

**Note:** To get a list of the publish destinations, see “Example 2: Print Multiple Publish Destinations” on page 44.

```sas
%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. It monitors the performance of a model from three categories. A user can use the following assignment statements to set warning and alert conditions.

Here are the three categories:

**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:

```text
%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:

```text
%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);

### 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_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>
</tbody>
</table>
| _MM_MODELFLAG       | 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. | 0 |
<p>| _MM_PREDICTEDVAR    | The predicted variable, when the model TARGET level is interval. | P_PRICE |
| _MM_EVENTPROBVAR    | The posterior probability variable of the event, when the model TARGET level is binary. | P_BAD1 |</p>
<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Description</th>
<th>Example Value</th>
</tr>
</thead>
<tbody>
<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.</td>
<td>0</td>
</tr>
<tr>
<td>_MM_SCORECODETYPE</td>
<td>The type of model score code. Valid values are DATASTEP, DS2EP, ANALYTICSTORE, and SASPROGRAM.</td>
<td>DATASET STEP</td>
</tr>
<tr>
<td>_MM_PERFOUTCASLIB</td>
<td>The CASLIB and libref for the performance monitoring results.</td>
<td>mm_mart</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>hmeq_perf_</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>Macro Variable Name</td>
<td>Description</td>
<td>Example Value</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</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>_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.</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_CHARACTERISTICALER</td>
<td>The alert criterion for characteristic analysis. The default value is \texttt{char_p1&gt;2}.</td>
<td>char_p1&gt;2</td>
</tr>
<tr>
<td>_MM_CHARACTERISTRICWAR</td>
<td>The warning criterion for characteristic analysis. The default value is \texttt{char_p1&gt;5 or char_p25&gt;0}.</td>
<td>char_p1&gt;5 or char_p25&gt;0</td>
</tr>
<tr>
<td>_MM_STABILITYALER</td>
<td>The alert criterion for stability analysis. The default value is \texttt{stab_p1 &gt; 1}.</td>
<td>stab_p1 &gt; 1</td>
</tr>
<tr>
<td>_MM_STABILITYWARNING</td>
<td>The warning criterion for stability analysis. The default value is \texttt{stab_p1 &gt; 2}.</td>
<td>stab_p1 &gt; 2</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_MODELASSESSMENTALERT</td>
<td>The alert criterion for model assessment. The default value is $(\text{lift5Decay} &gt; 0.15 \text{ and lift10Decay} &gt; 0.12) \text{ or } (\text{giniDecay} &gt; 0.1 \text{ or } \text{ksDecay} &gt; 0.1)$.</td>
<td>$(\text{lift5Decay} &gt; 0.15 \text{ and lift10Decay} &gt; 0.12) \text{ or } (\text{giniDecay} &gt; 0.1 \text{ or } \text{ksDecay} &gt; 0.1)$.</td>
</tr>
<tr>
<td>_MM_MODELASSESSMENTWARNING</td>
<td>The warning criterion for model assessment. The default value is $\text{lift5Decay} &gt; 0.05$.</td>
<td>$\text{lift5Decay} &gt; 0.05$.</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 source in the CASLIB “&amp;_MM_PERFOUTCASLIB”.</td>
<td>N or Y</td>
</tr>
<tr>
<td>_MM_ASTORELOCATION</td>
<td>A string that indicates where the analytic store files are located. The file locations are separated by a blank space. caslib1.astore caslib1.astore2 In this example, two analytic store files $\text{astore.sashdat}$ and $\text{astore2.sashdat}$ are stored in the Caslib “caslib1”.</td>
<td>caslib1.astore caslib1.astore2</td>
</tr>
<tr>
<td>_MM_FORCERUNALLDATA</td>
<td>Indicates to force performance monitoring to run against all performance input tables. The default value is N.</td>
<td>N or Y</td>
</tr>
<tr>
<td>_MM_LOADPERFRESULT</td>
<td>Loads the performance monitoring results if the CAS server is restarted. The default value is N. Note: Make sure the performance monitor results are saved before setting the value to Y.</td>
<td>N or Y</td>
</tr>
</tbody>
</table>

**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,
   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.

- **MM_MART=monitoring-output-caslib**
  specifies the CASLIB for the performance monitor results.

- **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 '/r/ge.unx.sas.com/vol/voll20/u12/scnkuj/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**

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.

   ```sas
   options cashost='cas-server-hostname'
      casport=cas-port-number;
   cas _mmcas_;
   caslib _all_ assign;
   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 monitor data into a caslib. For more information, see “Data Administration: How to (SAS Environment Manager)” in SAS Viya Administration: Data.

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 28.

```
%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_CakeepVars=YOJ MORTDUE DEROG VALUE CLNO LOAN CLAGE DELINQ NINQ;
%let _MM_Trace = OFF;
%let _MM_Max_Bins = 10;
%let _MM_PerfOutCaslib= CASUSER;
%let _MM_PerfInCaslib=public;
%let _MM_Perf_InTablePrefix=hmeq_perf_
%let _MM_RunScore=Y;
%let _MM_SAVEPERFRESULT=Y;
```

Example: 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_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=casuser;
%let _mm_perfincaslib=public;
%let _mm_perf_intableprefix=hmeq_perf_;
%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;
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 configure the publish destinations using SAS Environment Manager.

For more information, see “Configuring Publish Destinations” in SAS Model Manager: Administrator’s Guide.

The %MM_GET_TOKEN macro must be run to generate an authentication token to be used when executing the publish destination macros.

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,
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=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.

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

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

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.

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.
Hadoop Arguments

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

See “Configure a Publish Destination for Hadoop” in SAS Model Manager: Administrator’s Guide

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

See “Hadoop Data Source Settings” in SAS Model Manager: Administrator’s Guide

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

See “Hadoop Data Source Settings” in SAS Model Manager: Administrator’s Guide

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

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 “Configure a Publish Destination for Hadoop” in SAS Model Manager: Administrator’s Guide

Teradata Arguments

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

See “Configure a Publish Destination for Teradata” in SAS Model Manager: Administrator’s Guide

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 “Configure a Publish Destination for Teradata” in SAS Model Manager: Administrator’s Guide

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,
user=&userID,
pw=&password,
tokenname=myTokenName
);
%let defname=myHadoop;
%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 3: Define a Publish Destination for the SAS Micro Analytic Service**

%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,
    exttype= microAnalyticService,
    token=%myTokenName
);

**Example 4: Define a Publish Destination for Teradata**

%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,
%%MM_UPDATEPUBLISHDESTINATION Macro

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

Restriction: Only the authentication 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.

**TOKEN=authorization-token**

specifies the name of the authentication token that was generated with the %MM_GET_TOKEN macro.

See “%MM_GET_TOKEN” on page 4

**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.

**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.

---

**Hadoop Arguments**

**AUTHDOMAIN=** **authentication-domain**

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

See “Configure a Publish Destination for Hadoop” in *SAS Model Manager: Administrator’s Guide*

**HDFSDIR=** **HDFS-directory**

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

See “Hadoop Data Source Settings” in *SAS Model Manager: Administrator’s Guide*

**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

See “Hadoop Data Source Settings” in *SAS Model Manager: Administrator’s Guide*

**USER=** **user-name**

specifies the user name for Hadoop or Teradata.

*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 “Configure a Publish Destination for Hadoop” in *SAS Model Manager: Administrator’s Guide*

---

**Teradata Arguments**

**DATABASECASLIB=** **external-database-caslib**

specifies the caslib that contains the external database options.

See “Configure a Publish Destination for Teradata” in *SAS Model Manager: Administrator’s Guide*

**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 “Configure a Publish Destination for Teradata” in *SAS Model Manager: Administrator’s Guide*

### 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,
    modeltable=sas_model_table,
    token=%myTokenName
);
```

42 Chapter 4 • Publish Destination Macros
Example 2: Update the Definition for a Hadoop Publish Destination

%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

%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=casuser,
    modeltable=mm_model_table,
    databasecaslib=myTDLib,
    token=%myTokenName
);

/MMM_PRINTPUBLISHDESI TINATION Macro
Prints a list of publish destination definitions.

Syntax

%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.

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

See “%MM_GET_TOKEN” on page 4

**Examples**

**Example 1: Print One Publish Destination**

```&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**

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

**TOKEN=authorization-token**

specifies the name of the authentication 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(

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

%mm_read_json(  
   fref=pubsvrs,  
   respType=GET_ITEMS,  
   outds=dests,  
   statusVars_status  
 );
```

proc print data=dests noobs;  
   var name destinationType destinationTable casServerName casLibrary;  
run;
baseURL=&servernm,
user=&userID,
pw=&password,
tokenname=myTokenName
);

%let defname=myDefinitionName;
%mm_deletepublishdestination(
   baseURL=%str(&servernm),
   definitionname=%str(&defName),
   token=%myTokenName
);


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.