SAS® Open Model Manager 1.2: Administrator's Guide

Overview of SAS Open Model Manager Administration

This guide provides post-installation configuration tasks for SAS Open Model Manager 1.2, and explains both how to prepare SAS Open Model Manager for use and how to manage information that is associated with SAS Open Model Manager. Administrators can use the sas-admin command-line interface (CLI) to configure data libraries, manage user access and permissions, and manage content. The Model Publish API is used to configure publishing destinations.

Note: The Open Model Manager Resources GitHub repository contains examples of using Jupyter notebooks and Python scripts to submit REST API requests.

Here are the tasks that are included:

- Manage permissions
- Manage content
- Promote content
- Configure data libraries
- Configure the Docker daemon
- Configure publishing destinations
- Configure access to analytic store model files
- Configure support for Python code files

For information about deploying SAS Open Model Manager, see *SAS Open Model Manager for Containers: Deployment Guide*.

Managing Permissions

How to Manage Permissions

You use the sas-admin command-line interface (CLI) to manage identities and authorization for SAS Open Model Manager. You must have an administrator account in order to perform administrative tasks. If you did not specify an administrator in the sitedefault.yml file during deployment of SAS Open Model Manager, you must configure an administrator account. For more information, see "Configure an Administrator Account" on page 3.

Before you can use the sas-admin CLI you must complete the preliminary instructions for the command line interface. For more information, see "Command-Line Interface: Preliminary Instructions" in *SAS Viya Administration: Using the Command-Line Interfaces*.

Note: If your SAS Open Model Manager environment is enabled for Transport Layer Security (TLS), you must set the SSL_CERT_FILE environment variable to the path location of your site-signed certificate. Here is an example of how to set the environment variable in a Linux environment on the machine where the run_docker_container script was run:

```
export SSL_CERT_FILE= /<install-path>/runOpenMM/casigned.crt
```

The default permissions for SAS Open Model Manager are described in "Default Permissions" on page 2. You can change default settings:

- Modify permissions for specific folders or objects.
- Modify the existing groups or create new ones.
- Modify the existing rules or create new rules.

For more information, see the following documentation:

- "sas-admin CLI Permissions Example"
- "Folders: CLI Examples" in *SAS Viya Administration: Folders*
- "Identity Management: How To (CLI)" in *SAS Viya Administration: Identity Management*
- "General Authorization: How To (CLI)" in *SAS Viya Administration: General Authorization*

Default Permissions

Note: Only SAS administrators and other authorized users can create, update, or delete repository folders. In addition, authenticated users cannot initially access new custom repositories. A SAS administrator must grant access for a user or group to a new custom repository. Authorization for existing repositories is not modified during an upgrade.

By default, all authenticated users have permission to do the following tasks in the default repository and standard repositories:

- Read or view a list of all models and projects.
- Create a model or project.
- Copy a model from another project or a folder.
- Move a model from a folder to another folder or project version.
- Update and delete any model or project.
- Publish any model, including a project champion and its challenger models.
- Create a performance definition, run performance, and view performance results and history.
- Create a test definition for any model that they have access to read.
- View, update, or delete a test definition.
- Run a test and view the test results.

Note:

Specific permissions are required for the following tasks:

- To move a model, you must have the appropriate permissions for the source folder, object, and target folder. For more information, see "Folders: CLI Examples" in *SAS Viya Administration: Folders*.

Configure an Administrator Account

If you did not specify an administrator in the sitedefault.yml file during deployment of SAS Open Model Manager, you must configure an administrator account. Your SAS Open Model Manager environment is deployed with an initial administrator account that is named sasboot. The password for this account has expired by default, so you must reset the password before you can sign in.

To reset the password:

1. Run the following command to enter the Docker container:

   ```
   docker exec -it openmodelmanager bash
   ```

2. Locate the most recent log for the SAS Logon service in **/var/log/sas/viya/saslogon/default**.

3. Search the log for the account name **sasboot**.

   Note: Enter the command on a single line. Multiple lines are used here to improve readability.

   ```
   grep 'sasboot'
   /var/log/sas/viya/saslogon/default/sas-saslogon_date-and-time-stamp.log
   ```

   Here is a typical message:

   ```
   Reset password for initial user sasboot using link: /SASLogon/reset_password?code=xxxxxx
   ```

4. Sign in from a URL with this format:

4

Note: Specify in the URL the protocol HTTP or HTTPS that was configured during deployment.

```
http://reverse-proxy-server/SASLogon/reset_password?code=xxxxxx
```

5 Follow the instructions on the displayed web page to reset the password.

Note: If the URL has expired, run the following command to restart the container:

```
docker restart openmodelmanager
```

Then go to the log and obtain the new URL. The URL expires 24 hours after the SAS Logon service restarts. For security purposes, the URL that is specified in a browser or in a text editor also expires, even if the password is not reset.

After you reset the password, SAS Open Model Manager automatically opens in your browser.

Note: You can now perform administrative tasks as the sasboot user. Use the sas-admin CLI identities plug-in to add members to the SAS Administrators group. For more information, see "Identity Management: How To (CLI)" in *SAS Viya Administration: Identity Management*.

sas-admin CLI Permissions Example

This is an example of using the sas-admin CLI to create a custom group and grant permissions to a custom repository folder that is located within the **Model Repositories** folder. For information about using the sas-admin CLI with SAS Open Model Manager, see "How to Manage Permissions" on page 2.

TIP You can create a custom repository using the SAS Open Model Manager web application or by submitting a request to the Model Repository API.

1 Open a bash shell.

```
bash
```

2 Change to the directory that contains sas-admin CLI.

```
cd /opt/sas/viya/home/bin
```

3 Create a default profile that points to this image.

```
sas-admin profile init
```

4 Respond to prompts, such as the service endpoint.

```
http://myserver.com/
```

5 Log in as an administrator user, such as **sasboot**.

```
sas-admin auth login
```

```
sasboot
```

```
password
```

6 Create a custom group.

```
sas-admin identities create-group --name "Modelers" --id "_modelers"
--description "Anyone who uses model functionality."
```

7 Add a user to the custom group.

```
./sas-admin identities add-member --user-member-id sasdemo --group-id _modelers
```

8 Authorize access for the custom group to the custom repository folder.

a List folders with the name "Model Repositories" to get the folder ID. Copy the folder ID into a text editor to use in the next step.

```
sas-admin folders list  --name "Model Repositories"
```

b List members of the Models Repositories folder by ID to get the URI for the custom repository folder that you previously created. Copy the URI into a text editor to use in the next two steps.

```
sas-admin folders list-members --id "a1ea3925-c6b4-4816-9ec8-ad7d09ce4193"
```

c Grant Read, Add, and Remove object permissions for the custom group to the custom repository folder.

```
./sas-admin authorization authorize --permissions Read,Add,Remove --group "_modelers"
--object-uri "/folders/folders/3e90256e-3ebd-4ab2-9f33-83f6f4588d9f"
```

d Grant Read, Update, Delete, Add, and Remove container permissions for the custom group to the custom repository folder.

```
./sas-admin authorization authorize --permissions Read,Update,Delete,Add,Remove --group "_modelers"
--container-uri "/folders/folders/3e90256e-3ebd-4ab2-9f33-83f6f45"
```

You can also administer user and group identities, and grant folder permissions by submitting REST API requests. For an example of administering user and group identities using Python code in a Jupyter notebook, see the **addons** directory in the Open Model Manager Resources GitHub repository. Information about using the REST APIs is available at https://developer.sas.com/apis/rest/.

Managing Content

Information that you or other users save is stored and organized in folders. A *folder* is a virtual container rather than a representation of a physical file system. A folder contains members that are URIs for other folders, SAS resources, or resources outside SAS.

The **Model Repositories** folder is a common model repository for SAS applications. The **Model Repositories** folder can contain one or more repository folders. A repository folder can contain folders, models, and projects. A folder within a repository folder can contain models or projects. A project contains project versions, and a project version can contain one or more models. When a user creates a model or project, both a folder and an object are created. The folder and the object have the same name, and the object appears within the folder.

CAUTION

Do not rename folders or objects. The name of the **Model Repositories** folder, as well as repository folders, project folders, project version folders, model folders, and objects within the **Model Repositories** folder should not be modified using the sas-admin CLI. However, user-defined folders within a repository folder can be renamed.

For more information, see the following documentation:

- “Managing Permissions”
- “Folders: CLI Examples” in *SAS Viya Administration: Folders*
- “General Authorization: How To (CLI)” in *SAS Viya Administration: General Authorization*

Promoting Content

About Promoting Content

Promotion is the process of capturing content and moving it to a different location. For SAS Open Model Manager 1.2, promotion is performed using the transfer plug-in to the admin command-line interface (CLI). You can promote content between SAS Open Model Manager and SAS Model Manager on SAS Viya.

For more information, see the following documentation:

- “Promotion within SAS Viya: Tasks” in *SAS Viya Administration: Promotion (Import and Export)*
- “Import SAS Viya Resources” in *SAS Viya Administration: Promotion (Import and Export)*
- “Promotion: Import Using the Command-Line Interface” in *SAS Viya Administration: Promotion (Import and Export)*

Folder and Object Types

You can export the following folder and object types using the sas-admin command-line interface.

Here are a few points to be aware of:

- The **Model Repositories** folder, a repository folder, a folder or subfolder, and a project folder can be transferred. The transfer package contains all of the associated folders and objects within the exported folder.
- When a project object is exported, it does not contain the associated model folders and model objects in the transfer package. The model folder or model object URIs must be included in the transfer package.
- The project version folder does not contain the project object or model objects. Only the folders are transferred. Therefore, it is not recommended to promote content at the project version folder level.

Table 1 *Examples of Folder and Object Types*

Folder or Object Type	Location Example	URI Example
Model Repositories folder	/Model Repositories	/folders/folders/02c3ede2-0a41-4d56-997b-703aee31b329

Folder or Object Type	Location Example	URI Example
repository folder	/Model Repositories/ *Custom_Repository*	/folders/folders/ 3b709148-29c0-4be5-a1f1-4109b9b040c6
folder	/Model Repositories/ *Custom_Repository*/*sasdemo*	/folders/folders/ 870af9c9-9074-4ee0-84a6-bc3cd4954cf2
project folder	/Model Repositories/ *Custom_Repository*/*sasdemo*/ *QS_HMEQ*	/folders/folders/ de9c31c8-881f-40c0-86ef-029256c5f0a5
project version folder	/Model Repositories/ *Custom_Repository*/*sasdemo*/ *QS_HMEQ*/Version 1	/folders/folders/aa948017-7fe2-405f-abfc-1c7245f29e9f
project object	/Model Repositories/ *Custom_Repository*/*sasdemo*/ *QS_HMEQ*/*QS_HMEQ*	/modelRepository/projects/ a0792aac-1f20-4f70-8a0c-72baeb29523c
model folder	/Model Repositories/ *Custom_Repository*/*sasdemo*/ *QS_HMEQ*/*Version 1*/*QS_Tree1*	/folders/folders/3a7b1268-f9a4-4e7b-86f9-64df9acd4f99
model object	/Model Repositories/ *Custom_Repository*/*sasdemo*/ *QS_HMEQ*/*Version 1*/*QS_Tree1*/ *QS_Tree1*	/modelRepository/models/ 79b160f3-5fc4-4b66-9a82-baa98b2b3060

Important Considerations

Here are a few points to be aware of:

- When transferring content using the project object, model folder, or model object, you should import the projects first, followed by the models.
- When you are transferring content using the project object or project version folder, not all of the content is included in the transfer package. Therefore, it is not recommended to transfer content at these levels. You should transfer content from the project folder level instead.
- When project folders or project objects are transferred, the scoring tests, test results, published models, performance definition, and performance results are not included. Scoring tests can be transferred using the sas-admin command-line interface with the score definition URI and the score execution URI. However, it is recommended that you re-create and rerun the test in the target environment rather than transfer the test information.
- When you are transferring analytic store models, you must manually copy the content that is stored in the file system from the source system to the target system.

  Note: The analytic store files are located in the `/cas/data/modelStore` directory.

For more information, see “Configuring Access to Analytic Store Model Files”.

- When objects are transferred, the following properties change based on who performed the transfer and when it was done:
  - Created by
  - Modified by
  - Date created
  - Date modified
- Transferring data from the Model Publish service and Model Management service is not supported.

Transfer Request Body Examples

Folders and objects can be transferred at different levels from the Model Repository service using the transfer plug-in to the sas-admin command-line interface. Here are some examples of the request body that would appear within the JSON files that are used to transfer folder and object content using the sas-admin command-line interface.

Model Repositories Folder

If you specify the folder URI for the **`/Model Repositories`** directory, all of the repository folders, subfolders, authorization rules, projects, models, and model files are transferred as one package. Here is a sample request body:

```
{
    "name": "ModelRepositoriesFolderID",
    "items":
    [
            "/folders/folders/0fec5575-2ee2-4b18-ac3b-5afdd32b4412"
    ]
}
```

Repository Folder

If you specify the folder URL for a repository folder (for example, **`/Model Repositories/`***`myRepository`*), all subfolders, authorization rules, projects, models, and model files are transferred as one package. Here is a sample request body:

```
{
    "name": "RepositoryFolderID",
    "items":
    [
            "/folders/folders/b5d59f4c-2346-4de8-bb0d-e3714cdf5594"
    ]
}
```

Folder

If you specify a folder URI for a folder within a repository folder (for example, **`/Model Repositories/myRepository/myFolder`**), all subfolders, authorization rules, projects, models, and model files are transferred as one package. Here is a sample request body:

```
{
    "name": "FolderID",
    "items":
    [
            "/folders/folders/9c33b3dd-746e-4ebb-947a-c596f36d96d6"
    ]
}
```

Project Folder

If you specify the URIs for one or more projects, the specified projects and the associated models are transferred.

```
{
    "name": "Projects",
    "items":
    [
            "/folders/folders/de9c31c8-881f-40c0-86ef-029256c5f0a5",
            "/folders/folders/18f2c85c-68e3-418a-904e-a405aff8eb50"
    ]
}
```

Project Object

If you specify the URIs for one or more projects, the specified projects are transferred.

```
{
    "name": "Projects",
    "items":
    [
            "/modelRepository/projects/4f29e89c-bc93-42f5-8491-f338025d75e3",
            "/modelRepository/projects/15103b67-586c-4a3a-8ffd-b840c0921734"
    ]
}
```

Model Object

If you specify the URIs for one or more models, the specified models and their content are transferred.

```
{
    "name": "Models",
    "items":
    [
            "/modelRepository/models/c8f5694d-9717-45de-98cc-14d08edf7e10",
            "/modelRepository/models/85c66c5e-58ef-4ae6-9df3-3d709daf980f",
            "/modelRepository/models/d62ea249-5bea-4fc1-911d-966a387f4758"
```

```
    ]
}
```

sas-admin Command Line Interface Example

1. Create and initialize profiles for the source and target systems.

```
sas-admin --profile mySource profile init
```

```
sas-admin --profile myTarget profile init
```

2. Log in to the target and source systems.

```
sas-admin --profile mySource auth login
```

```
sas-admin --profile myTarget auth login
```

3. Export the source content to the transfer package.

```
/opt/sas/viya/home/bin/sas-admin --profile mySource transfer export
--request @request_myrepository.json
```

4. Download the transfer package from the source system. Replace the value of the *id* below with the *id* taken from the command output above in the console.

```
/opt/sas/viya/home/bin/sas-admin --profile mySource transfer download
--id 9b613274-ccf2-48b6-9567-85f46adb1430 --file myPackage_repository.json
```

5. Upload the transfer package to the target system.

```
/opt/sas/viya/home/bin/sas-admin --profile myTarget transfer upload
--file myPackage_repository.json
```

6. Replace the value of the *id* below with the *id* taken from the command output above in the console.

```
/opt/sas/viya/home/bin/sas-admin --profile myTarget transfer import
--request "{\"packageUri\":\"/transfer/packages/6b62ddfc-0a00-4b30-987e-1bb7c428e9a1\"}"
```

Configuring Data Libraries

About Configuring Data Libraries

During the deployment of SAS Open Model Manager, the ModelPerformanceData and ModelStore caslibs are created on the CAS Server (for example, cas-shared-default). The source type for the caslibs is a file system path. Users must have Read and Write permissions to the source file system directory paths. These directory paths are pre-configured as part of the SAS Open Model Manager deployment process. You can also create your own data libraries and then import tables or CSV formatted data.

Here is an example of creating a caslib using the sas-admin CLI:

```
sas-admin cas caslibs create path --name myCaslibA
--path /cas/data/myCasLibA
--server cas-shared-default
```

For more information, see "CLI Examples: CAS Administration" in *SAS Viya Administration: Using the Command-Line Interfaces*.

File System Directory Permissions

When you are defining a caslib where the source type is a file system directory path, you must grant the appropriate permissions. By default, CAS sessions run using the **cas** account. The CASHostAccountRequired custom group is a SAS Viya reserved group name. By default, all authenticated users are added to the CASHostAccountRequired custom group and members of this group automatically run their CAS sessions under their own host account. Analytic stores and performance results are created with group ownership by each user's primary group. For more information, see "File System Directory Permissions" in *SAS Viya Administration: Models*.

Configuring the Docker Daemon

About Configuring the Docker Daemon

To enable publishing of Python and R models to container destinations and to validate the models within the container destinations, you must first configure the Docker daemon. Here are the requirements:

- Review the Docker container security information about seccomp profiles and running the Docker daemon in rootless mode. See "Docker Container Security Considerations" on page 11.
- Map the Docker daemon socket on the host machine to the Docker daemon socket in the Docker run-time container. See "Map Docker Daemon Socket" on page 12.
- If the Docker daemon is installed on a different system than the one on which the Model Publish API service resides, then the Docker host must be configured to go through the TCP port. If you are publishing models to a Private Docker container destination, you must also assign the TCP port to the Private Docker daemon. For more information, see the Daemon CLI (dockerd) documentation.

For more information, see Configure and troubleshoot the Docker daemon.

Docker Container Security Considerations

Here are several points to consider as you configure the level of security for your Docker container.

For more information about container security, see Application Container Security Guide.

Seccomp Security Profile

Secure computing mode (seccomp) is a Linux kernel feature. You can use it to restrict the actions that are available within the Docker container. The `seccomp()` system call operates on the seccomp state of the calling process. You can use this feature to restrict your application's access.

The default seccomp profile that is used for running containers, disables around 44 system calls out of 300+. It is moderately protective, while providing wide application compatibility. You can override the default seccomp profile with the `--security-opt` option.

For more information, see Seccomp security profiles for Docker.

Rootless Docker

By default, the Docker daemon process runs as a user with root privileges. As a result, a user who performs the deployment would require root or sudoers privileges in order to run the required Docker commands.

However, you can also enhance the run-time security of the Docker container by running the Docker daemon in rootless mode instead. Rootless mode enables you to run the Docker daemon and containers as a non-root user. This mitigates potential vulnerabilities in the daemon and the run-time container. It is recommended that you run the Docker daemon in rootless mode with a TCP port connection. For more information, see Run the Docker daemon as a non-root user (Rootless mode).

If the rootless Docker daemon is on the same machine as the Docker run-time container, then configure the TCP port on that machine. If the rootless Docker daemon is on a remote machine, then open the TCP port on the remote machine. For more information, see the Daemon CLI (dockerd) documentation.

Note: Although configuring a TCP port connection between the Docker daemon and the Docker run-time container is the recommended approach, you can also map the Docker daemon socket.

See Also

"User and Group Requirements" in *SAS Open Model Manager for Containers: Deployment Guide*

Map Docker Daemon Socket

IMPORTANT Before you add the mapping option for the Docker daemon socket, make sure that you review the Docker Engine security information and understand the security impact to your site. Also, make sure that only trusted users have permissions to access SAS Open Model Manager when using this option. For more information, see Protect the Docker daemon socket.

To map the Docker daemon socket that is on the host machine to the Docker daemon socket that is in the Docker run-time container, add the following line of code to your Docker container script (run_docker_container).

```
-v /var/run/docker.sock:/var/run/docker.sock
```

Configuring Publishing Destinations

About Publishing Destinations

You can publish models to publishing destinations on SAS Cloud Analytic Services (CAS) and SAS Micro Analytic Service, as well as container destinations such as Amazon Web Services and Private Docker. By default, a SAS Micro Analytic Service destination named **maslocal** and a CAS destination named **CAS_PUBLIC** are defined for you. You must configure all other publishing destinations.

Note: You must update the thread count for the default SAS Micro Analytic Service (maslocal) destination. For more information, see "Update Number of Threads for SAS Micro Analytic Service" on page 13.

Here are the prerequisites for configuring other publishing destinations:

- Configure the Docker daemon for container destinations. For more information, see "Configuring the Docker Daemon" on page 11.
- Create a caslib for a CAS publishing destination using the sas-admin command-line interface (CLI).

  Here is an example of creating a caslib using the sas-admin CLI:

  ```
  sas-admin cas caslibs create path --name myCaslibA
  --path /cas/data/myCaslibA
  --server cas-shared-default
  ```

  For more information, see "Manage Caslibs" in *SAS Viya Administration: Using the Command-Line Interfaces*.

In order to configure publishing destinations for CAS, Amazon Web Services, and Private Docker, you must submit a REST API request to the Model Publish API. For more information, see "Using the REST API to Define Publishing Destinations" on page 14.

Update Number of Threads for SAS Micro Analytic Service

To be able to publish Python models to the SAS Micro Analytic Service (maslocal) and to validate those models within the destination, you must first set the number of threads for the SAS Micro Analytic Service to 1. This setting helps avoid a deadlock issue in the SAS Micro Analytic Service core.

Note: The following instructions use the Docker command-line interface (CLI). If you are using Portainer or another user interface to manage your Docker environments, see that user interface documentation.

1 Download the updateMASThreads.sh script from the **addons** directory located in the Open Model Manager Resources GitHub repository. Place the file in the location where you plan to initiate the connection to the Docker container.

Note: The default Docker container name is **openmodelmanager**.

2 To execute the updateMASThreads.sh script, submit the following commands from the same directory where you placed the file:

```
sudo su -

chmod +x updateMASThreads.sh

docker cp updateMASThreads.sh <container-name>:/usr/bin/updateMASThreads.sh

docker exec -itu root <container-name> /usr/bin/updateMASThreads.sh
```

Note: If you are using Portainer or another user interface to manage your Docker environments, the commands for executing the script might be slightly different. For example, instead of `docker exec -itu root`, you might need to use `docker exec -it -u=root`. For more information, see the Docker user interface documentation.

3 In order for the changes to take effect, you must submit the following commands to stop and restart the Docker container:

```
docker stop <container-name>

docker rm <container-name>

./run_docker_container --container-name <container-name>
--image <registry URL>/<namespace>/<image>:<tag>
--order <SAS order> [--http-port <port>  | --https-port <port>]
```

Using the REST API to Define Publishing Destinations

You can define publishing destinations by submitting REST API requests to the Model Publish API.

Note: When defining a publishing destination for Amazon Web Services (AWS), you must create a credential domain in the SAS Credentials service and store the AWS access key information in the credentials. You must also specify the identity (user or group) that can use the credentials to score or validate a model within the container publishing destination.

Python code examples for defining CAS, AWS, and Private Docker publishing destinations, as well as the mmAuthorization module, are available at https://github.com/sassoftware/open-model-manager-resources/tree/master/addons.

For more information, see the Model Publish API documentation and the Model Publish API examples on GitHub.

Creating Container Base Images for Scoring Models

After you create a container publishing destination for AWS or Private Docker, you must also create a container base image. Base image creation is currently supported for Python 3 and R models. The base images are used for scoring and validating Python models and R models at run time that have been published from SAS Open Model Manager to a container destination.

Note: If you want to publish both Python and R models to the same container publishing destination, you must create two base images, one for R models and one for Python models.

You must specify the following information when creating base images:

- publishing destination name
- host server name and port
- SAS user account that is authorized to connect to the host server

Sample Python scripts and additional information about creating the container base images are available in the **addons** directory of the Open Model Manager Resources GitHub repository. The Python scripts are used to submit REST API requests to the Model Publish API when you are creating the base images.

For more information about creating base images, see the Model Containerization section of the README.md file in the **addons** directory.

Configuring Access to Analytic Store Model Files

In order to publish analytic store models to a SAS Micro Analytic Service destination, you must configure access to the location where the analytic store (ASTORE) files are located. Also, users who work with analytic store models must have both Read and Write access to analytic store directories. The ASTORE file system directory path is **/models/astores/viya**.

Note: The ASTORE directory and its permissions are automatically set during the installation and configuration process for SAS Open Model Manager.

For more information, see "Configuring Access to Analytic Store Model Files" in *SAS Viya Administration: Models*.

Configuring Support for Python Code Files

Here are the configurations for Python code files that are automatically set during installation:

- To support models that contain custom Python code files, the PyMAS package support is automatically configured during installation of the SAS Open Model Manager Docker container.

  Note: The file system directory path for Python code files is `/models/resources/viya`. This directory and its permissions are automatically set during the installation and configuration process for SAS Open Model Manager.

  For more information, see "Enabling PyMAS Package Support" in *SAS Micro Analytic Service: Programming and Administration Guide*.

- Python 3.6.9 or later and the following packages are installed with SAS Open Model Manager:
  - joblib
  - numpy
  - pandas
  - patsy
  - python-dateutil
  - pytz
  - scikit-learn
  - scipy
  - six
  - statsmodels
  - xgboost

  For more information, see Python Package Index (PyPI).