# SAS® Viya® 3.5 Administration: Programming Run-Time Servers

## Programming Run-Time Servers: Overview
- SAS Viya Programming Run-Time Servers (Full Deployment) .................................................. 2
- SAS Viya Programming Run-Time Servers (Programming-Only Deployment) .......................... 3

## SAS Compute Server and Compute Service
- Overview ................................................. 4
- Operate the Compute Service (Linux) .......................................................... 4
- Operate the Compute Service (Windows) .................................................................. 5
- Add a Compute Server ......................................................................................... 6
- Lock Down the SAS Compute Server ................................................................... 7
- CLI Examples ........................................................................................................ 8
- Concepts ............................................................................................................... 8
- Log Files ............................................................................................................. 10
- Input Parameters ................................................................................................. 10
- Global Scope Macro Variables ........................................................................... 11

## SAS Launcher Server and Launcher Service
- Overview ................................................. 12
- Operate the Launcher Service (Linux) .................................................................. 12
- Operate the Launcher Service (Windows) ......................................................... 13
- View Launcher Server Properties ....................................................................... 14
- CLI Examples ........................................................................................................ 15
- Concepts ............................................................................................................... 15
- Troubleshooting .................................................................................................. 15
- Log Files ............................................................................................................... 16

## SAS Workspace Server and SAS Object Spawner
- Overview ................................................. 16
- How To ...................................................... 16
- Concepts ............................................................................................................... 22

## Embedded Web Application Server
- Overview .................................................. 23
- How To ...................................................... 23

## SAS/CONNECT Server and SAS/CONNECT Spawner
- Overview ................................................. 26
Programming Run-Time Servers: Overview

A programming run-time environment includes several SAS Viya servers. The following table lists the servers (and services, where applicable) and indicates which are available in a programming-only deployment:

<table>
<thead>
<tr>
<th>Server</th>
<th>Full deployment</th>
<th>Programming-only deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>“SAS Compute Server and Compute Service”</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>“SAS Launcher Server and Launcher Service”</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>“SAS Workspace Server and SAS Object Spawner”</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>“Embedded Web Application Server”</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>“SAS/CONNECT Server and SAS/CONNECT Spawner”</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

In the following diagram, the highlighted box shows the relationship of the programming run-time servers to other components in the SAS Viya environment in a full deployment:
In the following diagram, the highlighted box shows the relationship of the programming run-time servers to other components in the SAS Viya environment in a programming-only deployment:

**Figure 2  SAS Viya Programming Run-Time Servers (Programming-Only Deployment)**

*SAS Embedded Process is not required for serial connections.*
SAS Compute Server and Compute Service

Overview

The Compute service enables clients to submit SAS programs and stored procedures in the form of jobs for processing. The SAS Compute Server implements the Compute service. For more information, see "Concepts" on page 8.

Operate the Compute Service (Linux)

SAS Viya provides a script in /etc/init.d that you use to stop, start, restart, and check the status of the compute service. The script is named, sas-viya-compute-default.

Syntax

How you run sas-viya-compute-default depends on your operating system:

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  ```
  sudo systemctl status | stop | start | restart sas-viya-compute-default
  ```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
  ```
  sudo service sas-viya-compute-default status | stop | start | restart
  ```

Usage Notes and Tips

- You must be logged on to the machine where the compute service resides. Also, you must have sudo privileges to run this script.

- On multi-tenant SAS Viya systems, the script is named sas-tenant-ID-sas-viya-compute-default.

- There is another script that you can use to manage and view the running state of all SAS Viya services. For more information, see "Start and Stop All Servers and Services" in SAS Viya Administration: General Servers and Services.

  Note: There is a sequence for starting and stopping SAS Viya servers and services. You must follow this sequence to avoid operational issues. For more information, see "Read This First: Start and Stop Servers and Services" in SAS Viya Administration: General Servers and Services.

- On Linux systems that support systemd, use the systemctl command when running sas-viya-compute-default. The systemctl command maintains a record of service status that the service command and a direct call does not use.

  CAUTION
  
  On Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x, do not mix System V init and systemd commands. Mixing the System V init (service command) with the systemd (systemctl command) causes several issues. The systemctl
command knows nothing about a SAS Viya service started with the service command. If you start sas-viya-compute-default on Red Hat Enterprise Linux 7.x with the service command, and later attempt to shut down the compute service using the systemctl command, the compute service stops responding and does not shut down.

**Examples**

- To check status of the compute service on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

  ```
  sudo systemctl status sas-viya-compute-default
  ```

- To stop the compute service on Red Hat Enterprise Linux 6.x (or an equivalent distribution):

  ```
  sudo service sas-viya-compute-default stop
  ```

- To start the compute service on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

  ```
  sudo systemctl start sas-viya-compute-default
  ```

- To restart the compute service on Red Hat Enterprise Linux 6.x (or an equivalent distribution):

  ```
  sudo service sas-viya-compute-default restart
  ```

**Operate the Compute Service (Windows)**

Using the Microsoft Management Console (MCC) Services Snap-In, you can start, stop, and restart SAS Compute Service.
Add a Compute Server

For single machine deployment:

1. Stop all SAS Viya services. See “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services for more information.

2. Edit the original deployment inventory.ini file:
   - Review the comments for [ComputeServer] for shared file system and ensure that the infrastructure is correctly provisioned.
   - Add deploy targets to the [ComputeServer] host group.
   - Consider implications for [ComputeServices] and [StudioViya] if additional deploy targets are needed for them.

3. Direct Ansible to execute the site.yml playbook:
# ansible-playbook -i inventory.ini site.yml

4 Validate:

- Launch SAS Studio 5.x sessions and confirm the /opt/sas/spre/home/SASFoundation/utilities/bin/compsrv process is running on the new deploy target hosts.

---

Lock Down the SAS Compute Server

Using the “LOCKDOWN System Option” on page 39 and the “LOCKDOWN Statement” on page 40, you can limit access to files and to specific SAS features in a SAS Compute Server session in a multi-tenant environment.

To lock down a compute server:

Note: XCMD must be disabled to use the LOCKDOWN option.

1 With administrator privileges, log on to the machine that contains the compute server.

2 By default, SAS adds certain predefined paths from the SAS configuration file to the lockdown path list. (A whitelist that contains all the paths that are accessible to the compute server). To add more paths to the lockdown whitelist, go to /opt/sas/viya/config/etc/compsrv/default/autoexec_usermods.sas and add the lockdown path statements. For more information, see LOCKDOWN Statement Details on page 42.

Note: For Windows, add the lockdown path list to %ProgramData%\SAS\Viya\etc\compsrv\default\autoexec_usermods.sas.

Note: A path that is declared in the whitelist does not mean that an arbitrary user can read any file in that path. Host permissions on physical files and directories always take precedence over the whitelist.

Changes to the autoexec_usermods.sas file are automatically included when the compute server scripts run. Your changes will take effect the next time SAS starts a compute server session.

**TIP** For a suggestion about how to implement the whitelist, see “Example 2: Hiding the Whitelist By Locating the Path outside the Whitelist” on page 43.

3 To enable lockdown, set the environment variable COMPUTESERVER_LOCKDOWN_ENABLE to 1 in the sysconfig file /opt/sas/viya/config/etc/sysconfig/compsrv/default/sas-compsrv.

Setting this variable enables the -lockdown option in the start-up script.

Note:

For Windows, uncomment the environment variable COMPUTESERVER_LOCKDOWN_ENABLE in the configuration file that is located in the server's configuration directory %ProgramData%\SAS\Viya\etc\compsrv\default\compsrv_start_usermods.cmd.
Note: For both the operating systems, administrators can change the base path during installation and install to a different root folder.

4 If your site uses SAS Studio, set webdms.showSystemRoot=false.
   For more information, see “Update SAS Studio Configuration Properties” in SAS Viya Administration: Configuration Properties.

CLI Examples

The following examples assume that you have already signed in to SAS Viya at the command line. See “Command-Line Interface: Preliminary Instructions” in SAS Viya Administration: Using the Command-Line Interfaces.

**Example:** List the compute contexts.
```
sas-admin compute contexts list
```

**Example:** Validate the compute context session with the specified ID.
```
sas-admin compute contexts validate --id 389fee7a-e164-4e45-b836-a301638e9945
```

**Example:** Delete the compute context session with the specified name.
```
sas-admin compute contexts delete --name "SAS Job Execution compute context"
```

**Example:** List the launcher contexts.
```
sas-admin compute launchers list
```

**Example:** Delete the launcher context with the specified ID.
```
sas-admin compute launchers delete --id 8fbdd5f8-a2ee-42a5-a228-8737a0cf778f
```

**Example:** List the compute sessions.
```
sas-admin compute sessions list
```

See Also

“Command-Line Interface: Overview” in SAS Viya Administration: Using the Command-Line Interfaces

Concepts

SAS Compute Server

The SAS Compute Server enables clients to submit SAS programs and stored procedures in the form of jobs for processing using the SAS language. For every job that is processed, the compute server writes a logging message to a SAS log. If the job produces ODS results, output data sets, files, and so on, the output is associated with the job.

Compute servers are launched by a SAS Launcher Server.
Compute Service

The Compute service is a SAS Viya microservice that provides API endpoints for requesting a SAS Compute Server session. The compute service also provides API endpoints for creating and managing compute contexts, specifications that contain all the information that is needed to run a compute server.

The launcher service provides a specification to the launcher server called a launcher context, that enables the SAS administrator to apply constraints for how the launcher server starts a compute server.

How It Works

The following figure describes how a SAS client submits code to the SAS Compute Server.

*Figure 4  How a Client Submits Code to the SAS Compute Server*

1. A client gets an OAuth token from SASLogon. The client uses the token when submitting requests to the compute service.

2. The Compute service locates a compute context for the user. The compute context has a launcher context associated with it.

3. Using the contexts, the Compute service calls the Launcher service, which in turn calls the Launcher server. The Launcher server starts a Compute server.

4. The Compute service requests a session from the Compute server for the client user.

5. The user submits SAS code, and the Compute service routes the request to the correct Compute server session.
Fault Tolerance

You are able to deploy SAS Compute Servers for fault tolerance. You can deploy multiple SAS Launcher Servers on multiple compute server machines, and the Launcher service randomly routes client requests among the registered Launcher servers.

Only machine-level fault tolerance is supported. If a machine goes down, and you have other machines running Launcher and Compute servers, then fault tolerance is applied. If an individual Launcher or Compute Server process abnormally terminates, then no fault tolerance is applied.

Log Files

**Compute service**
Log files for the compute service are located in `/opt/sas/viya/config/var/log/compute/default`.

On multi-tenant systems, log files for the compute service are located in `/opt/sas/tenant-ID/config/var/log/compute/default`.

In Windows, the log files for the compute service are located in `ProgramData\SAS\Viya\var\log\compute\default`.

**Compute Server**
Compute servers and their logs are located where the launcher servers are running. Each compute server generates its own log. Log files are owned by the account under which the server was launched. This is useful in locating the file for a specific user.

Log files for the compute server are located in `/opt/sas/viya/config/var/log/compsrv/default`.

On multi-tenant systems, log files for the compute server are located in `/opt/sas/tenant-ID/config/var/log/compsrv/default`.

In Windows, the log files for the compute server are located in `ProgramData\SAS\Viya\var\log\compsrv\default`.

Input Parameters

Users can pass input parameters to the compute server. These parameters become global macro variables that can be used in SAS code. Therefore, you should be diligent when you process the input values as they can potentially cause a malicious code injection.

Here is an example to show the importance of where the input parameter is specified:

```sas
data=; proc datasets lib=work; run; quit;/*
where DATASET is the input parameter.

Here is an example of how the preceding data set is used in SAS code:

```sas
proc print data=; where sex eq "M" and age gt 14;
run;
quit;
```
Normally, the user specifies only the name of a data set to be used in the PROC PRINT statement. However, in this case, the user is being potentially malicious by specifying additional SAS code with the data set name. The code runs without errors, but provides unexpected results:

- The PROC PRINT statement runs without the WHERE statement.
- The PROC DATASETS statement is executed.
- The /* in the parameter value prevents execution of any SAS code that follows PROC DATASETS.

Situations like this one can pose a major security vulnerability. To avoid the vulnerability, the Compute server processes every incoming variable value to handle "special" characters. By default, the semicolon (;) character is replaced with a blank character.

The following characters are masked using a macro quoting function:

- Single quotation mark (‘)
- Double quotation mark (")
- Ampersand (&)
- Percent sign (%)

You can modify the list of "special" characters by changing the unsafeJobCharacters attribute in the compute context. For example, the list can have the following special characters: ' % " ; & < > . See Compute Server API documentation for more information.

For information about using input parameters in your code, see COMPSRV_OVAL Function and "COMPSRV_UNQUOTE2 Function" in SAS Functions and CALL Routines: Reference.

Global Scope Macro Variables

The Compute server creates global scope variables in the GLOBAL space. Here is one way to view them:

```$put _GLOBAL;```

**Note:** You can also view global scope variables using PROC PRINT and PROC SQL.

Here are the global scope variables:

- **SYS_COMPUTE_JOB_ID**
  Displays the compute server job ID of the current session.

- **SYS_COMPUTE_SESSION_ID**
  Displays the compute server session ID of the current session.

- **SYS_COMPUTE_SESSION_OWNER**
  Displays the owner of the current compute server session.

- **SYS_JES_JOB_URI**
  Provides a reference to the job execution object.

  For more information, see **SYS_JES_JOB_URI**.
SAS Launcher Server and Launcher Service

Overview
The SAS Launcher Server runs processes in a SAS Viya environment. The Launcher service is a SAS Viya microservice that provides API endpoints for how the launcher server runs a process.

Operate the Launcher Service (Linux)
SAS Viya provides a script in /etc/init.d that you use to stop, start, restart, and check the status of the launcher service. The script is named, sas-viya-launcher-default.

Syntax
How you run sas-viya-launcher-default depends on your operating system:

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
sudo systemctl status | stop | start | restart sas-viya-launcher-default

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
sudo service sas-viya-launcher-default status | stop | start | restart

Usage Notes and Tips
- You must be logged on to the machine where the launcher service resides. Also, you must have sudo privileges to run this script.
- On multi-tenant SAS Viya systems, the script is named sas-tenant-ID-sas-viya-launcher-default.
- There is another script that you can use to manage and view the running state of all SAS Viya services. For more information, see "Start and Stop All Servers and Services" in SAS Viya Administration: General Servers and Services.

Note: There is a sequence for starting and stopping SAS Viya servers and services. You must follow this sequence to avoid operational issues. For more information, see "Read This First: Start and Stop Servers and Services" in SAS Viya Administration: General Servers and Services.

- On Linux systems that support systemd, use the systemctl command when running sas-viya-launcher-default. The systemctl command maintains a record of service status that the service command and a direct call does not use.

CAUTION
On Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x, do not mix System V init and systemd commands. Mixing the System V init (service command) with the systemd (systemctl command) causes several issues. The systemctl command knows nothing about a SAS Viya service started with the service command. If you start
sas-viya-launcher-default on Red Hat Enterprise Linux 7.x with the service command, and later attempt to shut down the launcher server using the systemctl command, the launcher server stops responding and does not shut down.

The launcher server and launcher service support Kerberos on Linux. For more information, see “Configure Kerberos for SAS Launcher Server” in SAS Viya Administration: Authentication.

Examples

- To check status of the launcher service on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  ```bash
  sudo systemctl status sas-viya-launcher-default
  ```

- To stop the launcher service on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
  ```bash
  sudo service sas-viya-launcher-default stop
  ```

- To start the launcher service on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  ```bash
  sudo systemctl start sas-viya-launcher-default
  ```

- To restart the launcher service on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
  ```bash
  sudo service sas-viya-launcher-default restart
  ```

Operate the Launcher Service (Windows)

Using the Microsoft Management Console (MCC) Services Snap-In, you can start, stop, and restart SAS Launcher Service.
Figure 5  SAS Launcher Service in the Services Snap-In

See “General Servers and Services: Operate (Windows)” in SAS Viya Administration: General Servers and Services for details.

View Launcher Server Properties

To access the Servers window from SAS Environment Manager:

1. In the applications menu (≡), under Administration, select Manage Environment.
2. In the vertical navigation bar, click ⌜.
3. Select the server whose properties you want to view, and then click ⌜.
CLI Examples

The following examples assume that you have already signed in to SAS Viya using the command line. See “Command-Line Interface: Preliminary Instructions” in SAS Viya Administration: Using the Command-Line Interfaces.

**Example:** List all launcher contexts.

```bash
sas-admin launcher contexts list --all
```

**Example:** Show detailed information about the SAS Launcher context with the specified ID.

```bash
sas-admin launcher contexts show --id context-id
```

**Example:** List the SAS Launcher options set mappings.

```bash
sas-admin launcher options-set-mappings list --all
```

**Example:** Show detailed information about the SAS Launcher options that are set with the specified ID.

```bash
sas-admin launcher options-sets show --id options-set-ID
```

See Also

- “Command-Line Interface: Overview” in SAS Viya Administration: Using the Command-Line Interfaces
- “SAS Launcher Server and Launcher Service” on page 12

Concepts

**SAS Launcher Server**

The SAS Launcher Server starts processes, stops processes, and checks the status of processes in a SAS Viya environment.

For information about clustering, see “Fault Tolerance”.

**Launcher Service**

The launcher service is a SAS Viya microservice that provides API endpoints for how the launcher server runs a process. These API endpoints are used to create and manage launcher contexts.

Troubleshooting

**Failure to launch Compute server sessions**

**Explanation:**

Here are some reasons why a Compute server fails to launch:
- The user account under which the client is running does not have a home directory on the machine where the Compute server resides.
- Client users in a multi-tenant environment have to be a member of the sas group on the machine where the Compute server resides.
- Kerberos is present without valid credentials.

**Resolution:**
Check for the preceding issues in logs for the client application, Compute service, and Launcher service.

---

**Log Files**

Log files for the launcher service are located in `/opt/sas/viya/config/var/log/launcher/default`. On multi-tenant systems, log files for the launcher service are located in `/opt/sas/tenant-ID/config/var/log/launcher/default`.

For Windows, the log files for the launcher service are located in `\ProgramData\SAS\Viya\var\log\launcher\default`.

---

**SAS Workspace Server and SAS Object Spawner**

**Overview**

The SAS Workspace Server enables client programs to access SAS libraries, to perform tasks by using the SAS language, and to retrieve the results. One or more SAS Workspace Servers are initialized by the SAS Object Spawner.

---

**How To**

**Operate (Linux)**

SAS Viya provides a script in `/etc/init.d` that you use to stop, start, restart, and check the status of the SAS Object Spawner. The script is named, `sas-viya-spawner-default`.

**Syntax**

How you run `sas-viya-spawner-default` depends on your operating system:

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  ```
  sudo systemctl status | stop | start | restart sas-viya-spawner-default
  ```
Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-spawner-default status | stop | start | restart
```

**Usage Notes and Tips**

- You must be logged on to the machine where the object spawner resides. Also, you must have sudo privileges to run this script.
- On multi-tenant SAS Viya systems, the script is named `sas-tenant-ID-sas-viya-spawner-default`.
- There is another script that you can use to manage and view the running state of all SAS Viya services. For more information, see "Start and Stop All Servers and Services" in *SAS Viya Administration: General Servers and Services*.

---

**Note:** There is a sequence for starting and stopping SAS Viya servers and services. You must follow this sequence to avoid operational issues. For more information, see "Read This First: Start and Stop Servers and Services" in *SAS Viya Administration: General Servers and Services*.

---

**CAUTION**

On Red Hat Enterprise Server 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x, do not mix System V init and systemd commands. Mixing the System V init (`service` command) with the systemd (`systemctl` command) causes several issues. The `systemctl` command knows nothing about a SAS Viya service started with the `service` command. If you start `sas-viya-spawner-default` on Red Hat Enterprise Server 7.x with the `service` command, and later attempt to shut down the object spawner using the `systemctl` command, the object spawner stops responding and does not shut down.

---

**Examples**

- To check status of the object spawner on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  ```
sudo systemctl status sas-viya-spawner-default
  ```

- To stop the object spawner on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
  ```
sudo service sas-viya-spawner-default stop
  ```

- To start the object spawner on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  ```
sudo systemctl start sas-viya-spawner-default
  ```

- To restart the object spawner on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
  ```
sudo service sas-viya-spawner-default restart
  ```

---

**Operate (Windows)**

Using the Microsoft Management Console (MCC) Services Snap-In, you can start, stop, and restart SAS Object Spawner.
Enable X Commands (Linux)

Because clients can use host commands to perform potentially harmful operations such as file deletion, by default, X commands are disabled for the SAS Object Spawner. However, to enable X commands, follow these steps:

1. Log on to the machine on which the object spawner resides.
2. Using a text editor, open `/opt/sas/viya/config/etc/spawner/default/spawner_usermods.sh`.
3. Add the following line, save, and close the spawner_usermods.sh file:
   ```bash
   USERMODS="$JREOPTIONS -allowxcmd"
   ```
4. Restart the object spawner:
   ```bash
   sudo service sas-viya-spawner-default restart
   ```
Set umask or ulimit Values (Linux)

In many circumstances, it might be desirable to control the permissions of files created from SAS sessions, or to set process limits for SAS sessions. To set permissions on files created from SAS sessions on Linux, the umask command can be used. The ulimit command is used to set process limits. The location from which these commands are executed affect the scope of the settings.

1 Log on to the machine on which the SAS Workspace Server or the SAS/CONNECT Server resides. Log on as the SAS install user or log on with sudo privileges.

2 Using a text editor, open one of the following files, as appropriate:
   - For the SAS Workspace Server:
     /opt/sas/viya/config/etc/workspaceserver/default/workspaceserver_usermods.sh
   - For the SAS/CONNECT Server:
     /opt/sas/viya/config/etc/connectserver/default/connectserver_usermods.sh
   - For the SAS Workspace Server, the SAS/CONNECT Server, and all SAS instances:
     /opt/sas/spre/home/SASFoundation/bin/sasenv_local

3 Add your umask and ulimit values, and save the file.
   Your changes take effect the next time the server or servers are launched.

TIP The umask and ulimit settings can be set for all users (or values can be set conditionally for each user), for collections of users, or for all members of a given Linux group. For more information, see “Examples of umask and ulimit Settings”.

Examples of umask and ulimit Settings

In the following example, the umask command creates all files for all users with effective permissions of rw-r--r-- (owner:read and write; group:read; other:read):

```sh
umask 022
```

In the following example, umask is set for user joe00001 only:

```sh
if [ "$LOGNAME" = joe00001 ]
then
  umask 022
fi
```

In the following example, ulimits are set according to user ID or group membership.

```sh
# determine primary group membership of user
# GP=`groups $LOGNAME | awk '{ print $1 }'`
# assign new ulimit based on userid or group membership as desired
if [ "$LOGNAME" = joe00001 ]
then
  MAXSIZE=4096
  umask 022
elif [ "$LOGNAME" = fred0002 -o "$GP" = saspower ]
then
  MAXSIZE=8192
```
umask 077

eif [ "$GP" = sasuser ]
then
  MAXSIZE=6144
else
  MAXSIZE=8192
fi

export MAXSIZE

ulimit -f $MAXSIZE

Lock Down SAS Workspace Servers

Using the "LOCKDOWN System Option" on page 39 and the "LOCKDOWN Statement" on page 40, you can limit access to files and to specific SAS features in a SAS Workspace Server session that executes in a batch mode or a server processing mode in a multi-tenant environment.

To lock down one or more workspace servers:

1. With administrator privileges, log on to the machine that contains the workspace server.

2. Create a lockdown path list (a whitelist) that contains all the paths that are accessible to the server, and add it to /opt/sas/viya/config/etc/workspaceserver/default/autoexec_usermods.sas.

   Note: For Windows, add the lockdown path list to \ProgramData\SAS\Viya\etc\workspaceserver\default\autoexec_usermods.sas

   Note: A path that is declared in the whitelist does not mean that an arbitrary user can read any file in that path. Host permissions on physical files and directories always take precedence over the whitelist. SAS adds certain predefined paths from the SAS configuration file by default. For more information, see LOCKDOWN Statement Details on page 42.

   Changes to the autoexec_usermods.sas file are automatically included when the workspace server scripts run. Your changes will take effect the next time SAS starts a workspace server session.

   TIP For a suggestion about how to implement the whitelist, see “Example 2: Hiding the Whitelist By Locating the Path outside the Whitelist” on page 43.

3. To enable lockdown, set the environment variable WORKSPACESERVER_LOCKDOWN_ENABLE to 1 in the sysconfig file /opt/sas/viya/config/etc/sysconfig/workspaceserver/default/sas-workspaceserver.

   Setting this variable enables the -lockdown option in the start-up script.

   Note: For Windows, add the -lockdown system option to the configuration file that is located in the server’s configuration directory \ProgramData\SAS\Viya\etc\workspaceserver\default\sasv9_usermods.cfg.

4. If your site uses SAS Studio, set webdms.showSystemRoot=false.
Restricting SAS System Options

You can restrict SAS system options so that they cannot be changed by a user. An option can be restricted globally, by group, or by user.

Global Restrictions
Create the `/opt/sas/spre/home/SASFoundation/misc/rstropts/rsasv9.cfg` file and add options to this file.

Group Restrictions
Create the `/opt/sas/spre/home/SASFoundation/misc/rstropts/groups/groupname_rsasv9.cfg` file and add options to this file.
For example, for user smith in the group staff, the filename would be `staff_rsasv9.cfg`.

User Restrictions
Create the `/opt/sas/spre/home/SASFoundation/misc/rstropts/users/username_rsasv9.cfg` file and add options to this file.
For example, for user smith, the filename would be `smith_rsasv9.cfg`.

TLS Support for the SAS Object Spawner (Programming-Only Deployment)

To configure TLS on SAS Object Spawner, see “Configure TLS on the SAS Object Spawner ” in *Encryption in SAS Viya: Data in Motion*.

To configure the SAS Object Spawner to use TLS in a Linux programming-only deployment, see “Configure SAS Object Spawner to Use TLS and Custom Certificates (Linux Programming-Only Deployment)” in *Encryption in SAS Viya: Data in Motion*.

To configure the SAS Object Spawner to use TLS in a Windows deployment, see “Configure SAS Object Spawner to Use TLS and Custom Certificates (Windows)” in *Encryption in SAS Viya: Data in Motion*.
Concepts

SAS Workspace Server

The SAS Workspace Server enables client programs to access SAS libraries, to perform tasks by using the SAS language, and to retrieve results. Each workspace server process is owned by the client user that made the server request.

SAS Object Spawner

SAS Object Spawners interact with SAS by creating a server process for each client connection. SAS Workspace Servers are initialized by the SAS Object Spawner. An object spawner runs on the same machine as the workspace server, listens for requests, and launches the servers as necessary.

SAS Workspace Servers and SAS Cloud Analytic Services

In a SAS Viya environment, you can set up your autoexec.sas file to start a CAS session automatically. If you opt for automatic CAS session start-up, SAS uses that CAS session whenever it needs to communicate with SAS Cloud Analytic Services.

Many SAS procedures that are used in a SAS Viya deployment (such as PROC CARDINALITY and PROC NNET) use the CAS engine to communicate with CAS. The CAS engine uses the CAS session. In this context, the workspace server is used to interpret your SAS program and to determine how to run the lower-level actions in CAS.

Use of the SESSREF= DATA statement option in a SAS program is another method to inform the workspace server that CAS is being used. To run a DATA step in CAS, you must use a libref from the CAS engine, and you must specify the CAS session name in the SESSREF= option. When the workspace server interprets these language elements, it knows to run your DATA step in CAS.

In a SAS Viya environment, the workspace server is also used to do some work outside of CAS. Here are two examples:

- When creating graphics with procedures like PROC SGPLOT, although the data might be read from CAS with a CAS engine libref, the graphics are created with the workspace server.
- When processing data with the INFILE statement, the INPUT statement, and related DATA step statements and functions, the workspace server reads the contents of external files before the data can be transferred to CAS for analysis.

SAS Object Spawner Invocation

The SAS Object Spawner uses a suid root program, called elssrv, to launch processes under the identity of the requesting client. The user ID must be root in order to switch the identity to another user.

When launching a SAS Workspace Server, the client provides host credentials for the user who is requesting the SAS process (for example, a query or an ETL process) via the spawner. The spawner host authenticates the client and receives confirmation of valid credentials from sasauth. In addition, sasauth returns the UNIX uid and the list of groups. The suid root program launches the workspace server under this identity so that the process runs with the host authority of the requesting client.
Embedded Web Application Server

Overview

The embedded Apache Tomcat server that is used in all of the SAS Viya web applications provides the execution environment for SAS Studio.

How To

Operate (Linux)

SAS Viya provides a script in /etc/init.d that you use to stop, start, restart, and check the status of SAS Studio. The script is named, sas-viya-sasstudio-default.

Syntax

How you run sas-viya-sasstudio-default depends on your operating system:

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  sudo systemctl status | stop | start | restart sas-viya-sasstudio-default

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
  sudo service sas-viya-sasstudio-default status | stop | start | restart

Usage Notes and Tips

- You must be logged on to the machine where the embedded web application server resides. Also, you must have sudo privileges to run this script.

- On multi-tenant SAS Viya systems, the script is named sas-tenant-ID-sas-viya-sasstudio-default.

- There is another script that you can use to manage and view the running state of all SAS Viya services. For more information, see "Start and Stop All Servers and Services" in SAS Viya Administration: General Servers and Services.

Note: There is a sequence for starting and stopping SAS Viya servers and services. You must follow this sequence to avoid operational issues. For more information, see "Read This First: Start and Stop Servers and Services" in SAS Viya Administration: General Servers and Services.

- On Linux systems that support systemd, use the systemctl command when running sas-viya-sasstudio-default. The systemctl command maintains a record of service status that the service command and a direct call does not use.

CAUTION
On Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x, do not mix System V init and systemd commands. Mixing the System V init (service command) with the systemd (systemctl command) causes several issues. The systemctl command knows nothing about a SAS Viya service started with the service command. If you start sas-viya-sasstudio-default on Red Hat Enterprise Linux 7.x with the service command, and later attempt to shut down SAS Studio using the systemctl command, SAS Studio stops responding and does not shut down.

Examples

- To check status of SAS Studio on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  ```
sudo systemctl status sas-viya-sasstudio-default
  ```

- To stop SAS Studio on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  ```
sudo service sas-viya-sasstudio-default stop
  ```

- To start SAS Studio on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  ```
sudo systemctl start sas-viya-sasstudio-default
  ```

- To restart SAS Studio on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  ```
sudo service sas-viya-sasstudio-default restart
  ```

Operate (Windows)

Using the Microsoft Management Console (MCC) Services Snap-In, you can start, stop, and restart SAS Studio.
Configure Mail

To use the email functionality in SAS Studio, an SMTP server and the following information is required:

- **fully-qualified-SMTP-server-name**
  
  The fully qualified host name of the SMTP server for the outbound mail (for example, my_mail_server.example.com).

- **SMTP-server-port**
  
  The port for the SMTP server (for example, 25).

- **site-administrator-email-address**
  
  The user name that accesses the SMTP server.

  This user name is not necessarily the person who is sending the mail.

- **site-administrator-password**

See "General Servers and Services: Operate (Windows)" in SAS Viya Administration: General Servers and Services for details.
The password for the user name that accesses the SMTP server.

- company-domain

The domain name for your site (for example, my_company.example.com).

To configure SAS Studio for SMTP email, follow these steps:

1. Log on to the machine on which the embedded web application server resides.

2. Using a text editor, open /opt/sas/viya/config/etc/sasstudio/default/init_usermods.properties.

3. Add the following lines, save, and close the init_usermods.properties file:
   ```
   webdms.SMTP.hostName=fully-qualified-SMTP-server-name
   webdms.SMTP.port=SMTP-server-port
   webdms.SMTP.user=site-administrator-email-address
   webdms.SMTP.password=site-administrator-password
   webdms.domain=company-domain
   ```

4. Restart the embedded web application server:
   ```
   sudo service sas-viya-sasstudio-default restart
   ```

When sending email, the sender address is derived from the user name that logged on to SAS Studio and the value of the webdms.domain property in the appserver_usermods.sh file. For example, if the user name is test, the sender address would be test@your-company.com.

---

SAS/CONNECT Server and SAS/CONNECT Spawner

### Overview

SAS/CONNECT software provides the essential tools for sharing data and processing power across multiple computing environments:

- For users of SAS 9.4 and earlier versions, SAS/CONNECT enables you to use SAS Viya functionality and features.
  
  For more information, see “SAS 9 and SAS Viya” in SAS Viya: Overview.

- For SAS Viya users who might also have SAS 9, SAS/CONNECT provides parallel processing for CAS procedures.
  
  For more information, see SAS/CONNECT for SAS Viya User’s Guide

In a full deployment of SAS Viya on Linux, SAS/CONNECT is secure by default. In a programming-only deployment on Linux and Windows, you must configure security using Transport Layer Security (TLS). See “Use SAS/CONNECT with TLS Enabled to Import Data” in Encryption in SAS Viya: Data in Motion.
How To

Operate (Linux)

SAS Viya provides a script in /etc/init.d that you use to stop, start, restart, and check the status of SAS/CONNECT Spawner. The script is named, sas-viya-consul-default.

Syntax

How you run sas-viya-connect-default depends on your operating system:

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  `sudo systemctl status | stop | start | restart sas-viya-connect-default`

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
  `sudo service sas-viya-connect-default status | stop | start | restart`

Usage Notes and Tips

- You must be logged on to the machine where the spawner resides. Also, you must have sudo privileges to run this script.

- On multi-tenant SAS Viya systems, the script is named sas-tenant-ID-sas-viya-connect-default.
  
  An example is sas-tenant1-connect-default.

- There is another script that you can use to manage and view the running state of all SAS Viya services. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.

Note: There is a sequence for starting and stopping SAS Viya servers and services. You must follow this sequence to avoid operational issues. For more information, see “Read This First: Start and Stop Servers and Services” in SAS Viya Administration: General Servers and Services.

---

**CAUTION**

On Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x, do not mix System V init and systemd commands. Mixing the System V init (service command) with the systemctl (systemctl command) causes several issues. The systemctl command knows nothing about a SAS Viya service started with the service command. If you start sas-viya-connect-default on Red Hat Enterprise Linux 7.x with the service command, and later attempt to shut down the spawner using the systemctl command, the configuration server stops responding and does not shut down.

---

Examples

- To check status of the spawner on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  `sudo systemctl status sas-viya-connect-default`
To stop the spawner on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
```
sudo service sas-viya-connect-default stop
```

To start the spawner on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
```
sudo systemctl start sas-viya-connect-default
```

To restart the spawner on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
```
sudo service sas-viya-connect-default restart
```

**Operate (Windows)**

Using the Microsoft Management Console (MCC) Services Snap-In, you can start, stop, and restart SAS Connect Spawner.

*Figure 8  SAS Connect Spawner in the Services Snap-In*

See “General Servers and Services: Operate (Windows)” in *SAS Viya Administration: General Servers and Services* for details.
Set Configuration Options

For Linux, you can use any of these methods to set options such as encryption options to invoke SAS/CONNECT spawner:

- `/opt/sas/viya/config/etc/connect/default/connect_usermods.sh`
- the SASCMD option
- `/opt/sas/viya/config/etc/sysconfig/connect/default/sas-connect`

For Windows, use `\ProgramData\SAS\Viya\etc\connect\default\connect_usermods.bat`

For Linux, you can use any of these methods to set options to invoke SAS/CONNECT server:

- `/opt/sas/viya/config/etc/connectserver/default/connectserver_usermods.sh`
- `/opt/sas/viya/config/etc/sysconfig/connectserver/default/sas-connectserver`

For Windows, you can use `\ProgramData\SAS\Viya\etc\connectserver\default\connectserver_usermods.bat`.

Your changes take effect the next time the SAS/CONNECT server or spawner is restarted.

Lock Down the SAS/CONNECT Server

Using the “LOCKDOWN System Option” on page 39 and the “LOCKDOWN Statement” on page 40, you can limit access to files and to specific SAS features in a SAS/CONNECT server session in a multi-tenant environment.

To lock down your SAS/CONNECT server:

1. With administrator privileges, log on to the machine that contains the SAS/CONNECT server.

2. If you have not done so already, create a lockdown path list (a whitelist) that contains all the paths that are accessible to the server, and add it to `/opt/sas/viya/config/etc/connectserver/default/autoexec_usermods.sas`.

   **Note:** For Windows, add the lockdown path list to `\ProgramData\SAS\Viya\etc\connectserver\default\autoexec_usermods.sas`.

Changes to the `autoexec_usermods.sas` file are automatically included when the SAS/CONNECT server scripts run. Your changes will take effect the next time SAS starts a SAS/CONNECT server session.

**TIP** If you have already locked down your SAS/CONNECT server, then this step is unnecessary.

**Note:** A path declared in the whitelist does not mean that an arbitrary user can read any file in that path. Host permissions on physical files and directories always take precedence over the whitelist. SAS adds certain predefined paths from the SAS configuration file by default. For more information, see `LOCKDOWN Statement Details on page 42`. 
To enable lockdown, set the environment variable `CONNECTSERVER_LOCKDOWN_ENABLE` to 1 in the `sysconfig` file `/opt/sas/viya/config/etc/sysconfig/connectserver/default/sas-connectserver`.

Setting this variable enables the `–lockdown` option in the start-up script.

Note: For Windows, add the `-lockdown` system option to the configuration file that is located in the server’s configuration directory `ProgramData\SAS\Viya\etc\connectserver\default\sasv9_usermods.cfg`.

Note: Do not start the SAS/CONNECT spawner using the `-SHELL` option. As long as the `-SHELL` option is not specified, the `-NOXCMD` option is added by default to the server’s invocation parameters. `-NOXCMD` prevents clients from executing X commands from their SAS sessions to access system files.

If your site uses SAS Studio, set `webdms.showSystemRoot=false`.

For more information, see “Update SAS Studio Configuration Properties” in SAS Viya Administration: Configuration Properties.

Concepts

SAS/CONNECT software provides the essential tools for sharing data and processing power across multiple computing environments.

Note: SAS/CONNECT is ordered and licensed separately from other SAS Viya products.

SAS code uses these tools to perform tasks such as the following:

- dividing time-consuming tasks into multiple units of work and executing these units in parallel
- moving data from a client machine to a server machine (including legacy data from SAS 9), or vice versa, so that the data is on the same machine as the code processing it.

Reference

System Options

`TCPPORTFIRST=<port-number> | TCPPORTLAST=<port-number>`

Restricts the range of TCP/IP ports that clients can use to remotely access servers. Within the range of 0 through 32767, assign a beginning value to `TCPPORTFIRST` and an ending value to `TCPPORTLAST`. To restrict the range of ports to only one port, set the values for `TCPPORTFIRST`
and TCP_PORT_LAST to the same number. Consult with your network administrator for advice about these settings.

When **-NOINHERITANCE** is on, you can set TCP_PORT_FIRST and TCP_PORT_LAST in a SAS start-up command or in the configuration file.

This applies in the noinheritance case only. When socket inheritance is enabled, the child SAS/CONNECT server does not start up as a listening port.

<table>
<thead>
<tr>
<th>Server</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>0–32767</td>
</tr>
</tbody>
</table>
| Example| In the example below, the server is restricted to the TCP/IP ports 4020 through 4050:  
  options tcpportfirst=4020;  
  options tcpportlast=4050; |

### Environment Variables

**TCPBINDADDR <IP_address>**

Specifies an IP address override for all listening ports to bind to.

It can be specified on SAS CONNECT client or CONNECT server invocation.

The order of precedence of the IP address bind override environment variables are: TCPBINDADDR, SAS_EXTERNAL_BIND_ADDR, SAS_BIND_ADDR. Therefore, if TCPBINDADDR is set, the CONNECT server or the client will use this value over other values. The CONNECT spawner passes the bind IP address override on to the launched CONNECT server. If the spawner uses its BINDADDR value, it passes this to the CONNECT server’s environment by setting the TCPBINDADDR environment variable for the launched session. Then the launched CONNECT server will use this IP address to bind all of its listening ports to.

**Default**

Listens on all IP addresses available for the host, if the variable is not specified.

**Example**

```
./sas -set TCPBINDADDR 127.0.0.1
signon t1 sascmd="!sascmd";
```

### Server Environment Variables

The following SAS/CONNECT Server environment variables are available for configuring your TCP/IP connections. Place them in the `/opt/sas/viya/config/etc/connectserver/default/connectserver_usermods.sh` script file. For information about configuring environment variables in a Linux environment, see [Defining Environment Variables in UNIX Environments](#).

**CONNECTWDWAIT=<seconds>**

Specify to limit the possibility that a client session disconnect might orphan a runaway DMR mode session. To ensure the responsiveness of the spawner, SAS starts a “watchdog” thread to monitor the connection. The default interval is five seconds. If a disconnect occurs, CONNECTWDWAIT checks 18 times and then terminates the DMR thread (for a default elapsed time of 90 seconds). Setting the CONNECTWDWAIT value to zero means that the process does not monitor the connection.

**Defaults**

interval: 5 seconds  
total elapsed time: 90 seconds
Examples

In the following example, the option is set to 10, so the process waits 180 seconds, and then terminates the thread:

```
set CONNECTWDWAIT=10
```

In the following example, the option is set to 0, so the process does not monitor the connection:

```
set CONNECTWDWAIT=0
```

**TCPLISTENTIME=<seconds> | <MIN> | <MAX>**

Specifies the amount of time that a SAS/CONNECT server listens for a SAS/CONNECT client to connect before terminating the server session. It enables you to control idle and unresponsive sign-on connections by specifying how long (in seconds) a server “listens” for a response from the client during sign-on before it exits automatically. The default value for a session time-out is 0 (meaning, no time limit). The maximum value is 600 seconds.

<table>
<thead>
<tr>
<th>Client</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defaults</td>
<td>0 (no time limit)</td>
</tr>
<tr>
<td></td>
<td>MIN (minimum value is 0)</td>
</tr>
</tbody>
</table>

**Examples**

```
TCPLISTENTIME=MIN
TCPLISTENTIME=1
TCPLISTENTIME=90
TCPLISTENTIME=MAX (maximum value is 600)
```

*Note:* If TCPLTO is already set, it will have precedence over TCPLISTENTIME.

**TCP_POLL_INTERVAL=<seconds>**

Specify to ensure responsiveness of SAS spawners and servers to various conditions outside of normal request processing. When idle, servers and spawners periodically awaken to check for requests. The interval in seconds for this check is governed by the TCP_POLL_INTERVAL environment variable. Generally, the default setting of 60 seconds should be acceptable.

A value of zero means the server remains idle and awakens for request processing only.

**Example**

In the following example, the option is set to 50, so the process checks every 50 seconds for a connection:

```
TCP_POLL_INTERVAL=50
```

**TCPPMSGLEN=<size>**

Specifies the size of the buffer (in bytes) that the TCP/IP access method uses for breaking up a message that it sends to or receives from the SAS/CONNECT application layer during a SAS/CONNECT session. The application layer uses a message size that is stored in the TBUFSIZE option that you can specify in the SIGNON statement or as a SAS option.

If TBUFSIZE is larger than TCPPMSGLEN, the TCP/IP access method breaks the message into a buffer whose size is defined by TCPPMSGLEN, and issues the number of send and receive messages that are necessary to complete the message transaction.

The value for TCPPMSGLEN must be set at both the client and the server. If the values that are set for TCPPMSGLEN at the client and at the server are different, the smaller value of the two is used during the SAS/CONNECT session. If the TCPPMSGLEN environment variable is not set, SAS uses the TCP stack’s default size and allows autotuning if implemented by the stack.
CONNECTKEEPALIVE=<seconds>
Prevents a SAS/CONNECT client connection to the SAS/CONNECT server from being terminated.

Setting this environment variable in the server session prevents firewalls from terminating a connection between a client and server when there are long periods of inactivity on the connection. A keepalive packet is sent from a thread that is started by the server session for the specified number of seconds.

Example
The value of 5 causes the keepalive packet to be sent every 5 seconds to prevent connection termination.
set CONNECTKEEPALIVE=5

Spawner General Options

-CLEARTEXT
Allows sign-ons from clients that do not support user ID and password encryption. This option allows clients that are running older releases (prior to SAS 6.09E and SAS 6.11 TS040, which do not support user ID and password encryption) to sign on to the spawner program. Use this option only when absolutely necessary because credentials are transmitted unencrypted. The default encodes all communications.

Default -NOCLEARTEXT

-DEBUG
Turns on debug level output.

-HELP
Specifies to print the Help message.

-LOG | -LOGFILE <filename>
Specifies the filename to use for spawner log output if you are not using the -LOGCONFIGLOC option. The -LOG option should not be used with the -LOGCONFIGLOC option. If both options are specified, then the -LOGCONFIGLOC option takes precedence.

You can specify the -DEBUG or -TRACE options with the -LOG <filename> option to cause the spawner to send detailed log messages to a log file.

Example
In this example, the following option is enclosed in double quotation marks and added after USERMODS= in /opt/sas/viya/config/etc/connect/default/connect_usermods.sh. When the spawner starts, it sends debug-level log messages to a file named sas-connect.log:
USERMODS="-log /var/log/sas/viya/connect/default/sas-connect.log"

-LOGCONFIGLOC <filename>
Enables the SAS logging facility for SAS servers and names the location of the configuration file that is used by the SAS logging facility to create spawner log output. The configuration file is an XML file that specifies and configures loggers and appenders for the SAS/CONNECT spawner.

The file specification that defines the location of the XML configuration file must be a valid filename or a path and filename for your operating environment. If the path contains spaces, enclose the file specification in quotation marks.
Note If LOGCONFIGLOC is specified, spawner messages are routed by default to the App.Connect.Spawner logger.

-NOINHERITANCE
Disables socket inheritance.

Socket inheritance enables SAS/CONNECT servers to use the socket connection that is established between the SAS/CONNECT client and the spawner. Socket inheritance saves resources and is easier to configure when clients connect to a server that is within a firewall.

Default Socket inheritance is on.

-NOSCRIPT
Prevents sign-on from clients that use scripts, and allows sign-on only from clients that do not use scripts.

-NOSCRIPT can be useful if you want to limit SAS start-up commands to the use of the -SASCMD option. Specifying -NOSCRIPT restricts clients from specifying additional options in SAS start-up commands or script files.

Requirement Must be used with -SASCMD

-SASCMD | -CMD <command>
Specifies the SAS command or a command file that starts a SAS session when you sign on without a script. If the client does not specify a script file at sign-on, the -SASCMD option must be specified when starting the spawner.

Example In this example, the following option is enclosed in double quotation marks and added after USERMODS= in /opt/sas/viya/config/etc/connect/default/connect_usermods.sh. When the spawner starts, it uses a command file named mystartup:

USERMODS="-sascmd '/u/username/mystartup'"

Here is a sample command file named mystartup:

#!/bin/ksh
#----------------------------------
# mystartup
#----------------------------------
. ~/.profile
sas -noterminal -nosyntaxcheck $*
#-------------------

The $* positional parameter enables you to specify additional SAS options when you invoke SAS. In addition, $* also allows the options that the spawner adds automatically, like -DMR, to be included in the server session.

-SASDAEMONSERVICE <service-name | port>
Specifies the service name or port number that the SAS/CONNECT spawner uses to listen for child SAS/CONNECT server process connections.

If you use a service, its name must be configured in the SERVICES file on the computer that the SAS/CONNECT server session runs on.

-SERVICE <service-name | port>
Specifies the service name or port number to use to listen for client connections.

The -SERVICE option values that are used to start the spawner determine what is used by the client to sign on.

Note If the -SERVICE option is not specified, the spawner listens on Telnet port (23).
Example

In this example, the following option is enclosed in double quotation marks and added after USERMODS= in `/opt/sas/viya/config/etc/connect/default/connect_usermods.sh`. When the spawner starts, it uses port 5020 for the -SERVICE option during spawner start-up:

```
USERMODS="-service 5020"
```

The client can then sign on by specifying the explicit port-number in the SIGNON statement:

```
%let myHost=<spawner-host> 5020;
signon myHost user='myuserid' password='mypassword';
```

-SHELL

Specifies that the started SAS/CONNECT servers allow X commands.

Without specifying the -SHELL option to the spawner, X command processing is disabled by default.

-SSPI

Identifies support for the Security Support Provider Interface for single sign-on connections to the spawner. To enable SSPI authentication, you must specify -SSPI in the spawner start-up command.

Default -NOSSPI

-TRACE | -VERBOSE

Turns on trace level output.

-BINDADDR <IP_address>

Specifies an IP address override for the CONNECT Spawner to bind all its listening ports to.

The CONNECT spawner passes the IP address bind override to the CONNECT server that it starts.

If the CONNECT spawner uses its BINDADDR value, then it passes it to the CONNECT server’s environment by setting the TCPBINDADDR environment variable for the launched session. The launched CONNECT server then will use this IP address to bind all of its listening ports to.

It is used in multi-NIC (Network Interface Card) environment.

Spawner Security Options

SAS/CONNECT Spawner uses the “SAS System Options for Encryption” in Encryption in SAS Viya: Data in Motion.

Server Configuration Files

Configuration Home Directory

The SAS Viya deployment process creates a configuration home directory for each server instance. For Linux, the location for all services:

```
/opt/sas/viya/config/etc/compsrv/default
```
For Windows, the location for all services:

```
\ProgramData\SAS\Viya\etc\compsrv\default
\ProgramData\SAS\Viya\etc\workspaceserver\default
\ProgramData\SAS\Viya\etc\spawner\default
\ProgramData\SAS\Viya\etc\connectserver\default
\ProgramData\SAS\Viya\etc\connect\default
```

Note: Linux and Windows support the same services. The last directory in the path, `default`, is the deployment instance for the server.

---

### Server Configuration Files

Each of the following SAS Viya programming run-time servers uses one or more server configuration files, as appropriate:

- SAS Compute Server
- SAS Workspace Server
- SAS Object Spawner
- SAS/CONNECT Server
- SAS/CONNECT Spawner

**Table 1  Server Configuration Files**

<table>
<thead>
<tr>
<th>Standard File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoexec.sas</td>
<td>Contains SAS statements that are executed immediately after SAS initializes all components of the SAS Application Server. Do not modify this file. If you need to make changes, modify the appserver_autoexec_usermods.sas file that is in the same directory.</td>
</tr>
<tr>
<td>autoexec_deployment.sas</td>
<td>Contains server configuration settings that are created during deployment by Ansible from vars.yml. During updates, user configuration settings are overwritten. Do not modify this file. If you need to make changes, modify the sasv9_usermods.cfg file that is in the same directory.</td>
</tr>
<tr>
<td>autoexec_usermods.sas</td>
<td>Contains modifications made by the SAS administrator. Using autoexec_usermods.sas ensures that your modifications are not overwritten when you update SAS Viya.</td>
</tr>
<tr>
<td>Standard File Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sasv9.cfg</td>
<td>Specifies start-up options for the server and contains calls to other files that are listed in this table. Do not modify this file. If you need to make changes, modify the sasv9_usermods.cfg file that is in the same directory.</td>
</tr>
<tr>
<td>sasv9_deployment.cfg</td>
<td>Specifies start-up options for the server and contains calls to other files that are listed in this table that are created during deployment by Ansible from vars.yml. Do not modify this file. If you need to make changes, modify the sasv9_usermods.cfg file that is in the same directory.</td>
</tr>
<tr>
<td>sasv9_usermods.cfg</td>
<td>Contains modifications made by the SAS administrator. Using sasv9_usermods.cfg ensures that your modifications are not overwritten when you update SAS Viya.</td>
</tr>
<tr>
<td>logconfig.xml</td>
<td>Specifies the logging configuration for the server or the spawner.</td>
</tr>
<tr>
<td>logconfig.trace.xml</td>
<td>Contains alternative logging configuration settings for high-level logging messages (for example, DEBUG and TRACE messages) that can be used by SAS Technical Support to help resolve server issues. The messages are written to the server or spawner rolling log file.</td>
</tr>
<tr>
<td>logconfig.arm.xml</td>
<td>Specifies logging configuration files for servers that include specifications for collecting Application Response Measurement (ARM) log information and sending it to a log file. ARM logging is used to collect performance-related events.</td>
</tr>
<tr>
<td>logconfig.trace.arm.xml</td>
<td>Contains alternative ARM log configuration settings for high-level logging information. The messages are written to the server rolling log file.</td>
</tr>
<tr>
<td>sasenv_deployment</td>
<td>Contains server environmental variable settings that are created during deployment by Ansible from vars.yml. During updates, user configuration settings are overwritten. Do not modify this file. Add local environmental variable settings in the sasenv_local file in the /opt/sas/viya/config/etc/server/default directory.</td>
</tr>
<tr>
<td>connect.sh</td>
<td>The connect.sh and spawner.sh scripts start the SAS/CONNECT server and the SAS Workspace Server, respectively. The spawners accept connections from the clients to start SAS servers. Do not modify these files. If you need to make changes, modify the server-spawner_usermods.sh file that is in the same directory.</td>
</tr>
<tr>
<td>connectserver.sh</td>
<td></td>
</tr>
<tr>
<td>spawner.sh</td>
<td></td>
</tr>
<tr>
<td>workspaceserver.sh</td>
<td></td>
</tr>
<tr>
<td>connect_usermods.sh</td>
<td>Contain modifications made by the SAS administrator to the configurations for these application servers. Using server-spawner_usermods.sh ensures that your modifications are not overwritten when you update SAS Viya. Note: For the Compute Server, the file that is used for modifications is sas-compsrv, which resides in the sysconfig directory.</td>
</tr>
<tr>
<td>connectserver_usermods.sh</td>
<td></td>
</tr>
<tr>
<td>spawner_usermods.sh</td>
<td></td>
</tr>
<tr>
<td>workspaceserver_usermods.sh</td>
<td></td>
</tr>
<tr>
<td>sas-compsrv</td>
<td></td>
</tr>
</tbody>
</table>
Configuring SAS to Run External Languages

Configuring SAS to Run Python

You can enable Python code to run in SAS in lockdown mode. You can also configure SAS to run Python code using PROC FCMP.

For more information, see the following resources:

- Enabling Python Code While in Lockdown Mode on page 44

Python Requirements

These requirements must be met before PROC FCMP can be used to run Python code.

1. Install Python. Python version v2.7 or later is recommended for use with PROC FCMP.

2. Set the MAS_M2PATH environment variable to specify the absolute path to the mas2py.py file. The mas2py.py file is used to execute Python code within a Python process that is launched by SAS Micro Analytic Service. Here is an example:

   - UNIX:
     ```
     export MAS_M2PATH="/opt/sas/spre/home/SASFoundation/misc/embscoreeng/mas2py.py"
     ```

   - Windows:
     ```
     set MAS_M2PATH="C:\Program Files\SAS\SPRE\SASFoundation\misc\embscoreeng\mas2py.py"
     ```

3. Set the MAS_PYPATH environment variable to specify the absolute path to the Python executable. Here is an example:

   - UNIX:
     ```
     export MAS_PYPATH="/bin/python"
     ```

   - Windows:
     ```
     set MAS_PYPATH="c:\python\python.exe"
     ```
LOCKDOWN System Option

Enables the ability to limit access to files and to specific SAS Viya features for a SAS Viya session that is executing in batch or server processing mode.

**LOCKDOWN**
enables the ability to limit access to files and to specific SAS features for a SAS session that is executing in batch mode or server processing mode.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>SAS 9.4: Configuration file, SAS invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SAS Viya: Configuration file, SAS invocation, SASV9_OPTIONS environment variable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Environment Control: Initialization and Operation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PROC OPTIONS GROUP=</th>
<th>EXECMODES</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Default</th>
<th>NOLOCKDOWN</th>
</tr>
</thead>
</table>

**Restriction**  
This version of the LOCKDOWN system option is for SAS Viya only.

**Requirement**  
XCMD must be disabled to use the LOCKDOWN option. Enabling LOCKDOWN option will not automatically disable XCMD. For a secure environment, enable both the LOCKDOWN option and NOXCMD.

**Note**  
This option can be restricted by a site administrator. For more information, see Restricted Options in SAS Intelligence Platform: Administration / Application Server Administration Guide.

**See**  
For XCMD, see XCMD System Option: UNIX in SAS 9.4 and SAS Viya Programming Documentation / SAS Companion for UNIX Environments.

For SAS 9.4, see Locked-Down Servers in SAS Intelligence Platform: Administration / Application Server Administration Guide.

For SAS Viya, see SAS Workspace Server and SAS Object Spawner: How to on page 16 and SAS Compute Server on page 4.

In addition to LOCKDOWN, security administrators also rely on the NOXCMD system option. For more information, see XCMD System Option: UNIX in SAS Companion for UNIX Environments.

When the LOCKDOWN option is specified for a SAS session, SAS enters a locked-down state at a lockdown point. A SAS session in the locked down state has following restrictions:

- limited file system access
All access to local files and directories is validated through the lockdown path list. The lockdown path list specifies which host file resources are available when a SAS session is in the locked-down state. This list includes the default system directories and files.

- limited SAS language features
  - The following SAS language features are disabled:
    - DATA step Java Object **javaobj**
    - PROC JAVAINFO
    - SAS functions: ADDR, ADDRLONG, PEEK, PEEKLONG, PEEKC, PEEKCLONG, POKE, POKELONG, and MODULE

LOCKDOWN does not take effect in a SAS session until after the lockdown point has been reached.

The lockdown point has been reached during SAS execution when the following tasks have been completed in order to establish a user’s SAS environment:

- SAS session initialization
- AUTOEXEC execution
- INITSTMT execution

During initialization of the user’s SAS environment, all paths and files are available and work as designed (for example, SASHELP, WORK, LOG, and so on). AUTOEXEC predefined libraries also work as designed. When initialization is complete, SAS is put in the locked-down state with limited file system access.

---

### LOCKDOWN Statement

Secures the SAS Viya workspace server or the SAS/CONNECT server or SAS Compute Server on SAS Viya by restricting access from within a server process to the host operating environment.

#### Summary

<table>
<thead>
<tr>
<th>Valid in</th>
<th>AUTOEXEC file or INITSTMT= system option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Control</td>
</tr>
<tr>
<td>Type</td>
<td>Executable</td>
</tr>
<tr>
<td>Restriction</td>
<td>This version of the LOCKDOWN statement is for SAS Viya only.</td>
</tr>
</tbody>
</table>

#### Syntax

- **LOCKDOWN ENABLE_AMS** = access-method-1 < ... access-method-n >;
- **LOCKDOWN FILE** = file-ref ;
- **LOCKDOWN PATH** = 'pathname-1' < ' ... pathname-n '>
- **LOCKDOWN LIST**;
Arguments

**ENABLE_AMS=access-method-1 < … access-method-n >;**

By default, when SAS is in a locked-down state, these access methods are not available:

- EMAIL
- FTP
- HADOOP
- HTTP
- PYTHON
- PYTHON_EMBED
- SOCKET
- TCPIP
- URL

Likewise, the HTTP and HADOOP procedures are also not available. These procedures are initially disabled, preventing users from potentially generating spam or accessing files on a server that is not configured for authentication.

ENABLE_AMS allows administrators to re-enable those access methods and procedures that are disabled by default in the locked-down state.

Access method names are case-insensitive. Duplicate names are ignored. Separate each name with a space, and do not use quotation marks.

URL/HTTP and SOCKET/TCPIP are aliased name pairs. If one is re-enabled, the other is also automatically re-enabled. For example, if URL is re-enabled, HTTP is also re-enabled.

If either URL or HTTP is re-enabled, PROC HTTP is also automatically re-enabled.

If HADOOP is re-enabled, PROC HADOOP is re-enabled.

**ENABLE_AMS=** can be included only in an AUTOEXEC file or an INITSTMT= option.

**FILE=** file-ref

Names a file that contains a list of valid directories and files to be added into the lockdown path list. SAS automatically adds subdirectories of any valid directories in the list to the lockdown path list.

The lockdown path list specifies which files and directories a SAS session can access when in a locked-down state. (This list is often referred to as a whitelist.)

A lockdown file can contain multiple lines. Each line specifies a path string. A path string on each line can be one pathname or a concatenation of pathname specifications. If a line contains only one pathname, the pathname can contain spaces or the host-supported pathname characters. This pathname does not need to be enclosed in quotation marks.

If a line contains a concatenation specification, enclose the entire group of concatenated path specifications in parentheses. Enclose each pathname in quotation marks. Use a space to separate each pathname specification. Do not use newline characters in a path string. For more information, see **PATH = ‘pathname’ on page 42**.

If DBCS characters are included in the path list in the lockdown text file, add the ENCODING option to the FILENAME statement. This addition ensures that the lockdown path list is transcoded properly to the SAS session encoding. For example, suppose you have a text file, lockdown-text-file.txt, that includes a path list with DBCS characters, add the following code to the server's AUTOEXEC file appserver_autoexec.sas:
PATH=`pathname`

Specifies one or multiple paths to be added to the lockdown path list. Enclose pathnames in quotation marks. (Single quotation marks are preferable to avoid conflicts with SAS macro variables.) Use a space to separate multiple pathnames.

The lockdown path list specifies which files and directories a SAS session can access when in a locked-down state. (This list is often referred to as a whitelist.) SAS automatically adds subdirectories of any valid directories in the list to the lockdown path list.

A pathname can be relative or absolute. A pathname can also include operating system environment variables, SAS environment variables such as !SASROOT, the current working directory (.), and other host-specific character substitutions that are typically allowed in pathnames. For example, a UNIX path accepts ~/.

Note: SAS environment variables can be specified only at the beginning of a pathname.

LIST

Prints the current valid paths in the lockdown path list to the SAS log.

This list is an optimized path list. Subdirectories and duplicate pathnames are not shown.

Details

The LOCKDOWN statement enables you to limit access to local files and to specific SAS features for a SAS session that executes in a server session or in batch processing mode.

To use the LOCKDOWN statement, you must specify the LOCKDOWN system option in the sasv9_usermods.cfg for the specific server. For more information, see “LOCKDOWN System Option” on page 39.

A SAS server in the locked-down state validates all access to the host file system through the lockdown path list.

The lockdown path list contains all the paths that are accessible to a particular server. (This list is often referred to as a whitelist.) SAS does not verify the existence of any path in the lockdown path list. The operating system permissions on directories that are included in the lockdown whitelist are still in effect.

A path that is declared in the whitelist does not mean that an arbitrary user can read any file in that path. When the LOCKDOWN statement is not used, host permissions on physical files and directories always take precedence over the whitelist.

For details about implementing the whitelist, see Hiding the Whitelist By Locating the Path outside the Whitelist on page 43.

The lockdown path list is established during SAS initialization and finalized at the lockdown point.

There are two types of paths in the lockdown path list: default lockdown directories (paths from SAS configuration files) and user directories and files, which are added using LOCKDOWN statements in a server autoexec file.

Any modifications made to the whitelist (including defining a new stored process repository) do not affect servers that are currently running. After making changes, you can stop or quiesce any currently running processes on the affected pooled workspace server or stored process server. Your changes take effect the next time a server session is started.
For more information, see "Lock Down SAS Workspace Servers" on page 20, "Lock Down the SAS/CONNECT Server" on page 29, and "Lock Down the SAS Compute Server" on page 7.

By default, SAS adds the following predefined paths from the SAS configuration file and the SAS server metadata definition to the whitelist:

- stored process repository paths
- SASROOT path
- current working directory (not the User home directory)
- SASAUTOS option path or environment variable path
- UTILLOC option path
- LOGPARM defined LOG file path
- MAPS option path
- TEXTURELOC option path
- FONTSLOC option path
- JAVA_HOME/lib/fonts
- SASINITIALFOLDER (Windows only)
- "%SYSTEMROOT%\fonts" (Windows only) path

Note: Pre-assigned library paths are not automatically added to the lockdown list. Users can access pre-assigned libraries through librefs that are established through metadata or the AUTOEXEC files.

Other valid paths can be added in LOCKDOWN statements in the server’s AUTOEXEC files or the INITSTMT statement. Statements are typically added in the server autoexec_usermods.sas file.

Different types of SAS servers might require different lockdown path lists. For example, you might want a workspace server to have access only to an individual user’s home directory. Alternatively, a stored process server might require access to locations that require greater privileges. Librefs that are created by pre-assigned libraries in the metadata or autoexec file are also available to users.

Examples

Example 1: Enabling Access to a User’s Files via Servers
The servers can be any one of Workspace Server, SAS/CONNECT Server or Compute Server.

For a server that is launched under the client user’s personal account, an administrator can enable access to the user’s personal files but prohibit access to other users. It is not necessary to establish separate server definitions or implement special logic to submit a customized LOCKDOWN statement in the server’s autoexec file. Instead, an administrator can specify a special form of the lockdown path that refers to the home directory of the user under whose account the server was launched. Here are the declarations for host platforms that the server can run on.

- The declaration for Linux: `lockdown path= '~'`
- The declaration for Windows: `lockdown path= '\\POLDERID_Profile'`

Example 2: Hiding the Whitelist By Locating the Path outside the Whitelist
The SAS administrator can hide the contents of the whitelist by locating the whitelist file in a path that is not on the whitelist.

In your whitelist file, define all valid paths. For example, in whitelist.txt, add this code:

```
/valid/path1
```
Modify the server autoexec file by adding a `LOCKDOWN file=` statement to point to the whitelist file. In this example, `/opt/sas/viya/config/etc/lockdown` is not defined in the lockdown whitelist:

```plaintext
filename lkdn */opt/sas/viya/config/etc/lockdown/whitelist.txt;
lockdown file = lkdn;
```

**Example 3: Enabling a URL While in Lockdown Mode**

In this example, an administrator allows users access to certain URL sites while SAS is in LOCKDOWN mode. To do this, the administrator adds the following global statement to the server autoexec file:

```plaintext
lockdown enable_ams = URL;
```

**Example 4: Enabling Python Code While in Lockdown Mode**

To enable Python code to run in SAS in lockdown mode, add the following line to the server autoexec file:

```plaintext
lockdown enable_ams=PYTHON;
```

To enable Python code to run in SAS by directly submitting through a SUBMIT block, add the following line to the server autoexec file:

```plaintext
lockdown enable_ams=PYTHON_EMBED;
```

The access methods list is used for re-enabling those access methods that are usually restricted when SAS is in lockdown mode. PYTHON and PYTHON_EMBED are added to the existing list of access methods.

**See Also**

- Types of Sign-ons in SAS/CONNECT User’s Guide
- “LOCKDOWN System Option” on page 39