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:
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 3.

Operate the Compute Service

SAS Viya uses the operating system’s default init system or the systemd command to launch scripts that can stop, start, restart, and check the status of the compute service. The script `sas-viya-compute-default` resides in `/etc/init.d`.

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

To operate the compute service, run the following command, as appropriate:

`sas-viya-compute-default status` | `stop` | `start` | `restart`

Note: You can use a script to operate and view the running state of all SAS Viya servers and services. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.

Here are a few examples of how to operate the compute service:
To check status of the compute service using a direct call:

```bash
sudo /etc/init.d/sas-viya-compute-default status
```

To stop the compute service using the Red Hat Linux version 6 init system command:

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

To start the compute service using the Red Hat Linux version 7 systemd command:

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

To restart the compute service using a direct call:

```bash
sudo /etc/init.d/sas-viya-compute-default restart
```

**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. The job produces results when output is created.

Note: The SAS Compute Server does not support X commands.

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, which 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.
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

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.
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
SAS Viya uses the operating system’s default init system or the systemd command to start, restart, and check the status of the launcher service. The script, `sas-viya-launcher-default` resides in `/etc/init.d`.

Note: You must be logged on to the machine where the launcher service resides, and you must have sudo privileges to run this script.

To operate the launcher service, run the following command, as appropriate:

```
sas-viya-launcher-default status | stop | start | restart
```

Note: You can use a script to operate and view the running state of all SAS Viya servers and services. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.

Here are a few examples of how to operate the launcher service:

- To check status of the launcher service using a direct call:
  ```
  sudo /etc/init.d/sas-viya-launcher-default status
  ```

- To stop the launcher service using the Red Hat Linux version 6 init system command:
  ```
  sudo service sas-viya-launcher-default stop
  ```

- To start the launcher service using the Red Hat Linux version 7 systemd command:
  ```
  sudo systemctl start sas-viya-launcher-default
  ```

- To restart the launcher service using a direct call:
  ```
  sudo /etc/init.d/sas-viya-launcher-default restart
  ```

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:
- 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.

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.

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
SAS Viya uses the operating system’s default init system command or the systemd command to launch a script that can stop, start, restart, and check the status of the SAS Object Spawner. This script, sas-viya-spawner-default, resides in /etc/init.d.

Note: You must be logged on to the machine where the object spawner resides, and you must have sudo privileges to run this script.

To the operate the object spawner, run the following command, as appropriate:
sas-viya-spawner-default status | stop | start | restart

Note: You can use a script to operate and view the running state of all SAS Viya servers and services. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.

Here are a few examples of how to operate the object spawner:
- To check status of the object spawner using a direct call:
sudo /etc/init.d/sas-viya-spawner-default status

To stop the object spawner using the Red Hat Linux version 6 init system command:

```bash
sudo service sas-viya-spawner-default stop
```

To start the object spawner using the Red Hat Linux version 7 systemd command:

```bash
sudo systemctl start sas-viya-spawner-default
```

To restart the object spawner using a direct call:

```bash
sudo /etc/init.d/sas-viya-spawner-default restart
```

Enable X Commands

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

To control umask and ulimit sessions for the SAS Workspace Server, modify the workspaceserver_usermods.sh file. To affect umask and ulimit sessions for the SAS/CONNECT Server, modify the connectserver_usermods.sh file. To set umask and ulimit settings for the SAS Workspace Server, the SAS/CONNECT server, and all SAS instances, modify sasenv_local.

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:
     ```bash
     /opt/sas/viya/config/etc/workspaceserver/default/workspaceserver_usermods.sh
     ```
   - For the SAS/CONNECT Server:
     ```bash
     /opt/sas/viya/config/etc/connectserver/default/connectserver_usermods.sh
     ```
   - For the SAS Workspace Server, the SAS/CONNECT Server, and all SAS instances:
     ```bash
     /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):

```bash
umask 022
```

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

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

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

```bash
# 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
elif [ "$GP" = sasuser ]
then
  MAXSIZE=6144
else
  MAXSIZE=8192
fi

export MAXSIZE

ulimit -f $MAXSIZE
```

Lock Down SAS Workspace Servers

Using the **LOCKDOWN** system option and the **LOCKDOWN** statement, 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-environment deployment.

To lock down one or more workspace servers, follow these steps:

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:**
   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**.
3 Add the following line to `/opt/sas/viya/config/etc/workspaceserver/default/sasv9_usermods.cfg`:
   
   -lockdown

   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.

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.

5 If your site uses SAS/CONNECT, see “Lock Down the SAS/CONNECT Server” on page 13.

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 in the normal configuration file format.

Group Restrictions

Create the `/opt/sas/spre/home/SASFoundation/misc/rstropts/groups/groupname_rsasv9.cfg` file and add options to this file in the normal configuration file format.

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 in the normal configuration file format.

For example, for user smith, the filename would be `smith_rsasv9.cfg`.

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:

- Creating graphics with procedures like PROC SGPLOT does not run in CAS. The data might be read from CAS with a CAS engine libref, but the graphics are created with the workspace server.
- Some data processing (such as with the INFILE, INPUT, and related DATA step statements and functions) do not use CAS. These data processing statements are run in the workspace server session so that the contents of external files can be read before the data can be transferred to CAS for analysis.

SAS Object Spawner Invocation

The SAS Object Spawner uses an 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 that is requesting the SAS process to 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

SAS Viya uses the operating system’s default init system command or the systemd command to launch a script that can stop, start, restart, and check the status of SAS Studio. This script, `sas-viya-sasstudio-default`, resides in `/etc/init.d`.

Note: You must be logged on to the machine where SAS Studio resides, and you must have sudo privileges to run this script.
To operate SAS Studio, run the following command, as appropriate:

`sas-viya-sasstudio-default status | stop | start | restart`

Note: You can use a script to operate and view the running state of all SAS Viya servers and services. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.

Here are a few examples of how to operate SAS Studio:
- To check status of SAS Studio using a direct call:
  
  ```
  sudo /etc/init.d/sas-viya-sasstudio-default status
  ```
- To stop SAS Studio using the Red Hat Linux version 6 init system command:
  
  ```
  sudo service sas-viya-sasstudio-default stop
  ```
- To start SAS Studio using the Red Hat Linux version 7 systemd command:
  
  ```
  sudo systemctl start sas-viya-sasstudio-default
  ```
- To restart SAS Studio using a direct call:
  
  ```
  sudo /etc/init.d/sas-viya-sasstudio-default restart
  ```

**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**
  - 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
   ```
Restart the embedded web application server:

```bash
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 `apppserver_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 SAS 9 users, 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 http://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.3&docsetId=viyaconnref&docsetTarget=titlepage.htm

**How To**

**Operate**

SAS Viya uses the operating system’s default init system command or the systemd command to launch a script that can stop, start, restart, and check the status of SAS/CONNECT Spawner. This script, `sas-viya-connect-default`, resides in `/etc/init.d`.

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

To operate the spawner, run the following command, as appropriate:

```bash
sas-viya-connect-default status | stop | start | restart
```

Note: You can use a script to operate and view the running state of all SAS Viya servers and services. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.

Here are a few examples of how to operate the spawner:

- To check status of the spawner using a direct call:
  ```bash
  sudo /etc/init.d/sas-viya-connect-default status
  ```

- To stop the spawner using the Red Hat Linux version 6 init system command:
  ```bash
  sudo service sas-viya-connect-default stop
  ```

- To start the spawner using the Red Hat Linux version 7 systemd command:
  ```bash
  sudo systemctl start sas-viya-connect-default
  ```

- To restart the spawner using a direct call:
Set Configuration Options

Use `/opt/sas/viya/config/etc/connect/default/connect_usermods.sh` to set options such as encryption options, or you can use the SASCMD option in the SAS/CONNECT Spawner invocation.

Your changes take effect the next time the spawner is restarted.

Lock Down the SAS/CONNECT Server

Using the LOCKDOWN system option and the LOCKDOWN statement, you can limit access to files and to specific SAS features in a SAS/CONNECT server session in a multi-environment deployment.

To lock down your SAS/CONNECT server, follow these steps:

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`.

   **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.

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

3. Add the following line to `/opt/sas/viya/config/etc/connectserver/default/sasv9_usermods.cfg`:

   `-lockdown`

   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.

   **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.

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.

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

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

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

TCPMSGLEN=<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 TCPMSGLEN, the TCP/IP access method breaks the message into a buffer whose size is defined by TCPMSGLEN, and issues the number of send and receive messages that are necessary to complete the message transaction.

The value for TCPMSGLEN must be set at both the client and the server. If the values that are set for TCPMSGLEN at the client and at the server are different, the smaller value of the two is used during the SAS/CONNECT session. If the TCPMSGLEN environment variable is not set, SAS uses the TCP stack’s default size and allows autotuning if implemented by the stack.
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.

-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.

-NOCLERTEXT
Prevents sign-ons from clients that do not support user ID and password encryption. This option prevents 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) from signing on to the spawner program. However, the default permits both encrypted and plaintext user IDs and passwords.

-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.
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:

```ksh
#!/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.

Spawner Security Options
SAS/CONNECT Spawner uses the SAS System Options for encryption.

TCPPORTFIRST System Option
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 TCPPORTLAST to the same number. Consult with your network administrator for advice about these settings.

When -NOINHERITANCE is on, you can set TCPPORTFIRST and TCPPORTLAST 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;

Server Configuration Files

Configuration Home Directory
The SAS Viya deployment process creates a configuration home directory for each server instance.

Here is an example:

/opt/sas/viya/config/etc/spawner/default

The final 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.
<table>
<thead>
<tr>
<th>Standard Filename</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>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>Standard Filename</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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>Invoke sas with the default configuration for this SAS Application Server. It changes directories so that the SAS run-time environment is invoked from the root directory of this application server. 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.</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>
</tbody>
</table>

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © 2017, SAS Institute Inc., Cary, NC, USA. All Rights Reserved. February 2018 3.3-P1:calsvpgm