The SAS®9 Local Services Management Utility (SAS_lsm)

INTRODUCTION

Have you ever experienced any of the following issues?

- You try to restart SAS® servers to avoid loss of productivity, but the restart fails, causing delays.
- You have issues with and struggle to analyze the appropriate SAS logs as part of the troubleshooting process.

Or maybe you want some other processes developed to help you, such as these processes:

- You want to orchestrate a process to run after you start or stop SAS services.
- You would like SAS to notify you whether something in the system stopped working?

For both the problems you might have experienced or for processes that you might want to use, there is an answer. To address these issues, SAS Technical Support developed the SAS Local Services Management (SAS_lsm) utility.

First introduced at SAS Global Forum 2017, the SAS_lsm utility's sole purpose was to orchestrate the management of SAS services for a simple multi-tiered deployment (metadata, compute, and middle tiers) in UNIX or Linux operating environments. The utility works by using the sas.servers script that is deployed and configured with SAS installations. The utility enables the administrator to start, stop, restart essential SAS services and to determine the status of those services in their correct order across the tiers of a deployment. The utility is available via SAS Note 58231, "Utility that manages single- or multi-tiered SAS® related services for UNIX and Linux deployments."

Since the original SAS_lsm release, SAS Technical Support has added functionality to the utility. Here are some additional capabilities of SAS_lsm in the current release:

- metadata horizontal-clustering services
- robust parallel processing for SAS® LASR™ Analytic Services
- control of arbitrary, third-party services, based on user definitions
- deployment maintenance (target start from/stop at tier)
- centralized log collection and analysis
- streamlined input for the creation of Technical Support tracking entries
- robust pre- and post-control-action architecture for executing scripts (UserExits framework)
- operating-system integration with system software (if it is configured) for a graceful (controlled) exit of SAS services when the system powers off

UNDERSTANDING THE SAS_LSM SYSTEM ARCHITECTURE

SAS_lsm is designed to run from a single machine (the SAS_lsm controller), and it orchestrates the command processes to other machines that run SAS services via the secure shell (SSH) protocol. The configuration of a no-password SSH between the SAS_lsm controller and the accessed machines is required. This access is necessary only for the account that is used to start and stop SAS services.

You define machine names, control actions, and other settings in a SAS_lsm configuration file. A SAS_lsm controller can store an infinite number of SAS_lsm configuration files, enabling you to manage many SAS environments from a single machine. SAS_lsm reads the deployment-specific configuration file, which, in turn, drives the execution of tier-specific, services-management utilities in their proper order.

A SAS_lsm controller might, but is not required to, be a machine that is part of a SAS deployment. For example, many users opt to deploy SAS_lsm on their SAS® Metadata Server host because this machine is accessed first, traditionally, in a startup procedure. The SAS_lsm logic strips SSH from any locally running commands. Therefore, the execution steps to control that metadata server is run directly on the machine. Other users prefer a dedicated management host that is separate from any SAS software. Both methods are equally viable; organizational policy and personal preferences might dictate one method over the other.

System administrators who are familiar with Red Hat Ansible software might find it the SAS_lsm utility to be a good choice because the utility's functional architecture is very similar.

Secure Shell (SSH) Considerations

The SAS_lsm utility requires no-password SSH to be configured between the SAS_lsm controller and any hosts where control actions are to be executed. While you can execute the logic of SAS_lsm by entering a password each time an SSH action occurs, an average command action runs hundreds of queries over SSH. So, the time-saving aspect of SAS_lsm will be lost if the administrator must enter the password repeatedly!

Note that SSH functions differently when they are submitted in batch mode as compared to interactive mode. In batch mode, SSH sources only the user'' local shell profile; in interactive mode, both the system profile (for example, **`/etc/profile`**) and the local shell profile (for example, **`${HOME}/.bashrc`**) are sourced. SAS_lsm operates only in batch mode. Therefore, For any variables that are specific to SAS and that are located in a tier's system profile, ensure that they are replicated in the execution of your local shell profile when you use SAS_lsm.

The SAS_lsm configuration file contains a variable called SSHOPTIONS that is set in the miscellaneous section (typically at the top) of a configuration file. In this section, you can define any options that are normally provided to the SSH command when you run it interactively. Anything you define in that section is appended to the beginning of each SSH action that SAS_lsm runs. This behavior is useful if your system or systems have log-on messages, specific connection requirements, or other non-standard SSH functionalities that affect the SAS_lsm utility's ability to interpret command responses. For most users, the default command (`-q -o StrictHostKeyChecking=no -o PasswordAuthentication=no`) does not need to be modified.

Additional Considerations

SAS recommends that you start and stop services with the SAS installer account that you use when you originally deploy SAS. While it is not always a requirement that this be done, problems can occur if file permissions are not met for the user who starts SAS services. By using the SAS installer account, the files should be owned by that account, thereby avoiding such problems.

Note: You can determine the SAS installer account by checking the ownership of a deployed SAS file such as ***SAS-configuration-directory*/Lev*N*/sas.servers**.

SAS_lsm Tier Types

SAS_lsm tier definitions can be one of three types:

- MDCN – metadata cluster node
- MDCV – metadata cluster validation
- Standard – Any other tier or service

The MDCN tier type is used in SAS deployments with a clustered SAS® Metadata Server environment. When SAS_lsm manages more than one metadata server, the metadata-server process is started independently of any other services that might reside on the metadata-server machine. This independent startup occurs to ensure that potential dependencies are always met. You can identify an MDCN easily in a configuration file because the called script is MetadataServer.sh instead of the sas.servers script that is used commonly on standard-type tiers.

The MDCV tier type is used in SAS deployments with a clustered metadata environment. The MDCV tier is a non-functional tier that is used to check and maintain quorum status. While no additional setup is required to enable an MDCV tier, it is critical that you include one as the first tier *AFTER* each MDCN tier is defined. The MDCV tier is defined, typically, to point to the first-deployed MDCN machine. During a SAS_lsm action, the MDCV query reaches out to the defined machine and internally determines whether the metadata servers are in quorum. A failure to perform this quorum check results in SAS_lsm reporting a failure.

When you configure SAS_lsm on a clustered metadata environment with *n m*etadata-server machines, number the MDCN and MDCV tiers through *n*+1, where the additional tier **(+1**) is the MDCV tier for quorum validation. For example, consider an environment with three metadata-server machines. Because the metadata server is always started as the first step in a SAS deployment, you need to number these metadata-server machines as tiers 1, 2, and 3. Then, the MDCV tier is defined as tier 4. As a result, the quorum of tiers 1-3 can be checked after they are started (or before they are stopped).

In environments that use only a single metadata-server machine (and, thus, a single metadata-server tier), it is not necessary to use the MDCN and MDCV tier types. In single metadata-server environments, define the metadata-server tier as a Standard tier.

The Standard tier type is used in all SAS deployments. The Standard tier type is used, typically, for machines where services are controlled by using the sas.servers script. However, you can use the Standard tier for any control script call that is NOT controlling a clustered metadata-server tier. SAS compute servers and SAS middle-tier servers are defined as Standard tiers. If you define third-party services to be controlled using SAS_lsm, you should define them also as Standard tiers.

SAS_lsm Miscellaneous Variables

The top portion of a SAS_lsm configuration file contains a number of miscellaneous variables. Generally, these settings are expected to remain the same across all tiers. So, you define them once, and you do not redefine them on a per-tier basis. If, for some reason, one or more of these variables needs to be modified for a tier, simply redefine that variable at the top of a tier-definition stanza.

The table below shows the typical miscellaneous variables:

Variable	Definition	Example Value
CONFIGDIR	Root directory for the deployment's tier-specific service management scripts	**/opt/sas94/dev/config/Lev1**
INSTID	User ID (instance owner) that executes the deployment's tier-specific service management scripts	**sasinst**
MAXTIERS	Number of tiers in the deployment	**12**
STATUSROOT	Root directory for all reports that are generated by SAS_lsm and log files on the server where SAS_lsm is installed	**/usr/local/etc/SAS/reports**
SASADMIN	Email address(es) for your deployment administrators, space delimited	**sasadmlsm@sas.com**, **sas-admins@sas.com**, **vipusers@sas.com**
LOCALSSH	Optional. Default undefined. Forces use of SSH tunnel for commands executing on host where SAS_lsm is running. This is used to facade as another user in the event ownership/permissions are not consistent. Not recommended unless necessary.	Enabled: **LOCALSSH=override** Disabled *[Default]:* **LOCALSSH=bypass** If this variable is undefined, the default state (disabled) is used.

ARCHIVEDLOGS	Optional. Default undefined. Retains a record of start/stop actions with date-time stamps and output of SAS_lsm run. When enabled, output is saved under **$STATUSROOT/*name-of-cfg-file*/archive/**	Enabled: **ARCHIVEDLOGS=enabled** Disabled *[Default]:* **ARCHIVEDLOGS=disabled** If this variable is undefined, the default state (disabled) is used.

SAS_lsm Variables and Tier Definitions

The defined variables for a tier differ based on the type of tier that is defined. The following variables are always defined, regardless of tier type:

Variable	Description	Notes
TIERNAME[*N*]	Descriptive name for the tier	Requires user input
TIERINST[*N*]	User ID (instance owner) that executes scripts on the tier	Absolute value set
TIERHOST[*N*]	FQDN server host name	Requires user input
TIERSTAR[*N*]	Full path to the tier-start script	Absolute value set
TIERSTOP[*N*]	Full path to the tier-stop script	Absolute value set
TIERREST[*N*]	Full path to the tier-restart script	Absolute value set
TIERSTAT[*N*]	Full path to the tier-status script	Absolute value set
TIERSTAS[*N*]	Defines the search token to ensure that tier services are started	Absolute value set
TIERSTOS[*N*]	Defines the search token to ensure that tier services are stopped	Absolute value set
TIERLOGD[*N*]	Defines the root value for the tier's service-log directory	Absolute value set
TIERTYPE[*N*]	Defines tier type	Absolute value set
TIEREXIT[*N*]	Path to the tier's UserExit scripts directory	Optional – ignored if the value is missing

DEPLOYING SAS_LSM

Download SAS_lsm

To obtain the latest version of SAS_lsm, see the SAS_lsm GitHub project page and follow these steps:

1. Under the top-listed (newest) release, locate the **Assets** list at the bottom of the release notes. Click the file SAS_lsm-v*SAS-lsm-version-number*.tgz to download the release package.
2. Transfer the downloaded file onto the machine where you intend to run SAS_lsm.
3. Extract the files in your desired location on the machine using this command:

```
tar -xvf SAS_lsm-vSAS-lsmversion-number.tgz
```

Generate the Base Configuration File

SAS_lsm is a configuration-file driven process, meaning that most setup for the utility is done in the configuration file. To make this process easier, the utility includes a configuration-file wizard that assists in generating the base configuration file. The wizard asks questions about your deployment and helps to set the required variables in your SAS_lsm configuration file that are needed for the utility to run successfully.

To launch the configuration-file wizard and generate your base configuration file, execute the lsm_config wizard script in the BASH shell. This script is located in the SAS_lsm download package in the directory **`.../utilities/lsm_configWizard`** .

Note: Execute the configuration wizard as the SAS installer account.

The questions asked by the wizard might vary based on the information that you provide to previous questions. You need to determine the following information in advance, or open a second terminal while you run the configuration wizard so that you can locate the following values:

Value Required	Example Answer	Notes
Destination file path and file name of the SAS_lsm configuration file	**`/opt/sas/utilities/SAS_lsm.cfg`**	Provides the full directory path **AND** file name.
The SAS configuration directory	**`/opt/sas/config/Lev1`**	Include **`Lev`** #.
SAS Installation User	**`sasinstall`**	See "Additional Considerations."
The number of machines in the SAS deployment	**`6`**	Defines the MAXTIERS variable in the output configuration file.
The host name for each machine	**`demohost.sas.com`**	The fully qualified data-set name (FQDN) is recommended. An alias or a short name is acceptable if machines can resolve them.
Tier-identification name for each machine	**`Metadata Server 1`**	It is recommended that you set this value to a human-readable description of machine's role in deployment.

After the wizard generates your base configuration file, you should open the file and view the contents to check for accuracy. Take this time to familiarize yourself with the general layout of the SAS_lsm configuration file. Notice the miscellaneous variables at the top, followed by tier stanzas. A *tier stanza* contains all the definitions that are needed to generate a tier. Typically, tier stanzas start with TIERNAME[*N*] , where *N* increments by one each tier, from the value **`1`** to the value set as MAXTIERS.

For most standard deployment types, you can use this base configuration file without modification. If you have unique requirements or you want to configure extended SAS_lsm functionalities see the next section, "Modifying Your Configuration File." Configuring extended functionalities includes, for example, configuring a different CONFIGDIR or INSTID value per tier, user exits, SAS LASR Analytic Server control, third-party services control, or other customizations.

Modify Your Configuration File

You can edit your SAS_lsm configuration file manually by using your favorite text editor, such as the UNIX vi editor or the GNU nano editor. Detailed sections below explain modification tasks that require further configuration (such as LASR control).

A common and simple change is to set a unique value for the CONFIGDIR variable for a different tier. This action might be required if you deployed multiple tiers on one machine, such as a shared compute-and-middle-tier server. To make this change, simply define the CONFIGDIR value at the top of each tier-stanza, as necessary.

Note that the configuration file is read for each tier-stanza at the beginning of the tier. **Consider this example:** Tiers 1, 2, 3, and 5 all use the same CONFIGDIR value, but tier 4 is different.

- To handle this scenario, define the CONFIGDIR variable on tier 1, tier 4, and tier 5.
    - Tiers 2 and 3 retain the value that you set on tier 1.
    - Change to the new value at tier 4.
    - Change again (back to the old value) on tier 5.
- If you define the CONFIGDIR variable on tier 4 and you DO NOT redefine it on tier 5, then tier 5 attempts to use the setting from tier 4.

This example is the general basis for most simple changes. Setting a unique-per-tier INSTID, for example, is done the same way.

The SAS_lsm utility attempts to validate the configuration file before each utility run. Be aware that the validation might not catch all errors that are made in the configuration file. However, DO NOT rely on this validation as a safety-net for mistakes! If you are unsure about how to set up something in your SAS_lsm configuration file, contact SAS Technical Support.

Enable Control of SAS® LASR™ Analytic Server

There are two known methods for controlling LASR servers via SAS-lsm. The current recommended option is to use the LASROps utility, which allows control via the LASR API. SAS-lsm previously offered its own custom LASR control method – this method is still functional for MPP (multi-machine) LASR deployments but is much more involved to configure.

Information on both methods is retained in this document.

Control LASR via LASROps Utility (recommended)

Download and configure the LASROps Utility from its GitHub project page.

Instructions on how to configure the utility are included in the README for the project.

- https://github.com/ajforeman/LASROps

This utility is designed to run as a standalone script. Test the utility outside of SAS-lsm to verify it can start, stop, and check the status of your LASR servers.

Once you have verified that the LASROps utility is working correctly, add it to your SAS-lsm configuration file as a normal tier stanza. *(outline example at right)*

```
INSTID=sas
LASRHD=/opt/sas/config/Utilities/LASROps
TIERNAME[N]='LASR via API'
TIERINST[N]=${INSTID}
TIERHOST[N]=TRCv156.trc.sas.com
TIERSTAR[N]="${LASRHD}/LASROps.sh start"
TIERSTOP[N]="${LASRHD}/LASROps.sh stop"
TIERREST[N]="\"${LASRHD}/LASROps.sh stop && ${LASRHD}/LASROps.sh start\""
TIERSTAT[N]="${LASRHD}/LASROps.sh status"
TIERSTAS[N]='is DOWN'
TIERSTOS[N]='is UP'
TIERLOGD[N]=${LASRHD}
TIERTYPE[N]='standard'
```

Control LASR via SAS-lsm Custom Method (legacy – only for MPP LASR)

To enable SAS_lsm control of SAS Analytic Server, it is necessary to generate .sas programs that contain the code to execute SAS LASR Analytic Server start-up and shut-down commands with the appropriate credentials.

Ensure that you have the following items configured:

- logon credentials for the SAS LASR Analytic Server instance owner--This account is any account that is registered in SAS® Metadata Server with appropriate permissions to control SAS LASR Analytic Server.
    - no-password SSH for the host operating-system account that starts and stops SAS LASR Analytic Server0
    - connection from the SAS_lsm control machine into the machine where SAS LASR Analytic Server runs

(list continued)

- a directory (on the SAS LASR Analytic Server machine) where you can store and execute .sas programs.
- ability to execute XCMD (the X statement) in SAS sessions (see "XCMD SAS System Option" in the *SAS Companion* for your operating environment).

Ensure that you have the following required information:

- the SAS Metadata Server host name: For clustered metadata, specify the first metadata host.
- the SAS Metadata Server port: The default port is 8561.
- the SAS Metadata Server logon account for the SAS LASR Analytic Server instance owner: You can test these credentials by logging on to SAS® Management Console, SAS® Visual Analytics, or another SAS application.
- the password for the owner account of the SAS LASR Analytic Server instance , encoded in SAS002 format. To generate the SAS002-encoded password, submit the PWENCODE procedure in a SAS session. An example of this procedure, the log output, and the SAS002-encoded string are shown below:

```
proc pwencode in=my-password-to-encode123!';

run;
```

```
3     proc pwencode in=XXXXXXXXXXXXXXXXXXXXXXXX;
4     run;


{SAS002}B829B94107972BE127E998E028F3AF954AF4C10B5431B17A15631C7C4208120557A19B19

NOTE: PROCEDURE PWENCODE used (Total process time):
      real time           0.04 seconds
      cpu time            0.01 seconds
```

The SAS002-encoded string for *MyPasswordToEncode123!* is:
{SAS002}B829B94107972BE127E998E028F3AF954AF4C10B5431B17A15631C7C4208120557A19B19

Follow this procedure to generate the SAS LASR Analytic Server control files:

1. Log on to the SAS® LASR™ host machine as the owner of the SAS LASR instance.
2. Change directory to the location that you created to store SAS LASR Analytic Server control programs.
3. Copy the SAS_lsm_LASR.tar file archive to your current directory.
4. Extract the archive with the command **`tar -xvf ./SAS_lsm_LASR.tar`**

(list continued)

5. Edit the file meta_cfg.sas to provide the information that is needed to make a connection to the metadata server as the SAS LASR instance owner:

```
%let mdserverhost=your-metadata-server-host-name;

%let mdserverport= your-metadata-server-port; /* Default port:*/
                                              /* 8561;        */

%let mdserveruser=sasdemo;

%let mdserverpass={SAS002}your-password;
```

6. Run the SAS program SAS_lsm_LASR_config.sas to generate instance-specific SAS LASR Analytic Server configuration files by calling the SAS® Foundation executable binary file (sas) by submitting the following command:

```
/opt/sas94/dev/SASHome/SASFoundation/9.4/sas SAS_lsm_LASR_config.sas
```

7. If the metadata-server connection is successful, a file that is named mpp_cfg_*LASR-port-number*.sas file should be produced for each SAS LASR Analytic Server port and instance in the environment.

   If these files are not produced, verify that the metadata-server credentials that are provided are valid.

Implement the SAS LASR Analytic Server control scripts:

You are now ready to implement SAS LASR Analytic Server control via SAS_lsm by setting a call the default mpp_*action*.sas files. These files search for and detect the port-specific control files that are generated in the previous steps.

The following example shows two SAS LASR Analytic Server control tiers. The first SAS LASR Analytic Server (tier 9) is on port 10011; the second server is on port 10031. You can use the syntax that is shown below to call the SAS LASR host machine (sasapp-2) and to control those services.

```
## BEGIN OPTIONAL massively parallel processing SAS LASR services Tier Descriptor Template

INSTID=sasdemo
SASHD=/opt/sas94/dev//SASHome/SASFoundation/9.4
LASRHD=/usr/local/etc/SAS/LASR
LASRPORT=10011
TIERNAME[9]='Private LASR'
TIERINST[9]=${INSTID}
TIERHOST[9]=sasapp-2
TIERSTAR[9]="${SASHD}/sas ${LASRHD}/mpp_start.sas -log ${LASRHD}/logs/${LASRPORT} -sysparm ${LASRPORT}"
TIERSTOP[9]="${SASHD}/sas ${LASRHD}/mpp_stop.sas -log ${LASRHD}/logs/${LASRPORT} -sysparm ${LASRPORT}"
TIERREST[9]="${SASHD}/sas ${LASRHD}/mpp_restart.sas -log ${LASRHD}/logs/${LASRPORT} -sysparm ${LASRPORT}"
TIERSTAT[9]="${SASHD}/sas ${LASRHD}/mpp_checklasr.sas -log ${LASRHD}/logs/${LASRPORT} -sysparm ${LASRPORT}"
TIERSTAS[9]='LASR DOWN'
TIERSTOS[9]='LASR UP'
TIERLOGD[9]=${LASRHD}/${LASRPORT}
TIERTYPE[9]='standard'

LASRPORT=10031
TIERNAME[10]='Public LASR'
TIERINST[10]=${INSTID}
TIERHOST[10]=sasapp-2
TIERSTAR[10]="${SASHD}/sas ${LASRHD}/mpp_start.sas -log ${LASRHD}/logs/${LASRPORT} -sysparm ${LASRPORT}"
TIERSTOP[10]="${SASHD}/sas ${LASRHD}/mpp_stop.sas -log ${LASRHD}/logs/${LASRPORT} -sysparm ${LASRPORT}"
TIERREST[10]="${SASHD}/sas ${LASRHD}/mpp_restart.sas -log ${LASRHD}/logs/${LASRPORT} -sysparm ${LASRPORT}"
TIERSTAT[10]="${SASHD}/sas ${LASRHD}/mpp_checklasr.sas -log ${LASRHD}/logs/${LASRPORT} -sysparm ${LASRPORT}"
TIERSTAS[10]='LASR DOWN'
TIERSTOS[10]='LASR UP'
TIERLOGD[10]=${LASRHD}/${LASRPORT}
TIERTYPE[10]='standard'
```

Note that it is possible also to implement these LASR control steps as a UserExit framework script. Continue to the next section for more information about UserExit framework control.

USEREXIT FRAMEWORK

Capabilities

The UserExit Framework is an exciting new addition to SAS_lsm 4.0 . Using this framework, you can define custom scripts and execute them at various points during a SAS_lsm action sequence. This behavior provides administrators with an easy way to perform additional functions during the start -up and stopping of SAS services. It also offers a simple way to connect to third-party tools without those tools needing to adhere to the SAS_lsm utility's tier-messaging expectations.

Using the UserExit framework is optional. If you want to run UserExit scripts on a tier, define the TIEREXIT[*N*] variable in that tier's definition stanza. If this variable is not defined, the UserExit framework simply skips processing on that tier.

You can define UserExit scripts to run at various times in relation to their tier. The available options are as shown here:

- **Pre Start:** Runs before the tier is started.
- **Post Stop:** Runs after the tier is started.
- **Pre Stop:** Runs before the tier is stopped.
- **Post Stop:** Runs after the tier is stopped.

Post-action UserExit scripts can react to their tier's status. This means that you can define post-action UserExit scripts to run dependent on whether SAS_lsm determines that their tier action experiences success or failure.

UserExit scripts can be marked as **required**, if you want, for their completion to impact the SAS_lsm process. A required UserExit script prevents SAS_lsm from starting and stopping additional tiers when a failure is reported. If the completion of a UserExit script is not critical and it is not marked as required, SAS_lsm continues the action sequence regardless of the UserExit script's status.

By using the UserExit timing options, tier-status reaction, and required flag, powerful new functionalities can be added to the SAS administrator's capabilities with SAS_lsm. Here are some examples of tasks that you can implement using the UserExits Framework:

- Before a tier is started, you can verify that required network ports are not in use.
- After a tier is started successfully, you can start the SAS Deployment Agent service on that tier.
- After a tier start action fails, you can create a list of configuration files that were edited in the past day.
- Before a tier is stopped, you can email users to notify them that SAS services are being restarted.
- After a tier is stopped successfully, you can make a backup of the **`SASHOME`** and SAS configuration directory.
- After a tier stop action fails, you can generate a list of any orphaned or zombie processes on the host.

Configuration

The SAS_lsm UserExits framework expects that all the UserExit scripts that should be run on a tier are stored in the same directory. This directory is defined by the TIEREXIT[*N*] property in the tier stanza, and it should exist on the tier on which the scripts are to be run. Ensure that this location is readable and executable by the TIERINST[*N]* user.

For example, suppose that you want to run UserExit scripts on your metadata-server host (which is defined as Tier 1). To do so, you set TIEREXIT[1] to a path on the metadata-server host where you store your UserExit scripts.

The SAS_lsm process determines the order and factors of when a UserExit script should run based on the filename. Therefore, it is critical that the proper naming conventions are followed. UserExit scripts are expected to be named by using this syntax:

*XX*_[PRE|POST]_[START|STOP|STATUS]_[SUCCESS|FAILURE]_[REQUIRED]_*identifying-name*

In this syntax:

- *XX* is a numeric identifier that defines the order in which UserExit scripts should be run, if multiple scripts are detected for a given action factor. If you have only one script for that action factor, use the value **`01`**. If you have multiple scripts, you use sequential numbers (**`01`**, **`02`**, **`03`**, and so on).

(list continued)

- The PRE or POST keywords define whether a script should run before (PRE) or after (POST) a tier's respective action. This specification is required and unique (that is, you can specify only one of these two keywords).
- The START, STOP, and STATUS keywords define whether a script should run on a SAS_lsm start action, a SAS_lsm stop action, or a SAS_lsm status action. This specification is required and unique (that is, you can specify only one of these three keywords for each userExit script).
- The **`success`** or **`failure`** keyword can be used on START/STOP **`post`** actions, to control if a userExit should only be run when the preceding action succeeds or fails. This keyword will be ignored on **`pre`** START|STOP and all STATUS userExit scripts. This specification is unique.
- The **`required`** keyword can be specified when a UserExit script affects the entire SAS_lsm action process. If a required UserExit script is determined to be unsuccessful, further tiers are not be controlled by that SAS_lsm action process. This keyword is optional, and a UserExit script is not considered to be required if it is not specified.
- The identifying name (specified as the user-supplied value ***`identifying-name`***) is any text that the administrator wants to use in order to identify that UserExit file. This identifying name does not need to meet any specific naming conventions. The script name is shown during a SAS_lsm action execution, so providing a clear identifying name can be helpful in understanding which process is being executed at a given time.

To assist in determining which UserExit scripts run in a specified action, use the userExit_tester.bash script that is provided with the SAS_lsm download in the **`utilities/`** subdirectory. The script is executed by providing the full path to the UserExits script directory, followed by the action that you want to test, as shown in this example:

```
./userExit_tester.bash /path/to/userExit-scripts/ post stop success
```

The output of this script displays the files located in the specified directory that matches the naming conventions for the **`post stop success`** action. If more than one matching file is located, the order in which they are executed by SAS_lsm is displayed, as shown in the following sample output:

```
1.   /SASScripts/testScripts/01_post_stop_success_maketmpfile.bash
2.   /SASScripts/testScripts/02_required_post_stop_success_make2ndTempFile.sh
3.   /SASScripts/testScripts/03_post_stop_success_make3rdtmpfile
```

Platform LSF "GRID" Control UserExit

Added in SAS_lsm 4.1.0 is the ability to easily configure and control Platform LSF GRID services via SAS_lsm UserExit functionality.

Locate the script **SAS_lsm_Grid_TEMPLATE.sh** in the **`utilities/`** subdirectory. Edit this file and supply the necessary configuration variables for your Platform LSF environment. Comments inside the script provide examples of locations and settings for these variables.

Custom Scripts

The successful (or unsuccessful) completion of a UserExit script is determined by reading the shell **exit code** of the script. A script that runs successfully and has no errors should return an exit code of **0**. For a script that does not perform its expected functions, the recommended code that should be returned is **5.**

- SAS_lsm considers the following exit codes as a successful completion of the UserExit script: **0**, **1**, **2**, **4**
    - The standard BASH exit code for an error is either **1** or **2**. SAS_lsm receives these exit codes only when the called script has issues when it runs. SAS expects that the administrator has tested and verified the functionality of the UserExit script before implementing it with SAS_lsm.
    - Exit code **4** is set to appear successful to SAS_lsm for instances where the administrator wants to handle specific failures in a required UserExit script without ending the SAS_lsm action process.
- Any other exit code s be considered a failure by SAS_lsm. SAS Technical Support recommends the use of exit code **5** because it is a non-reserved code in BASH.

At times, you might want to run third-party processes as a SAS_lsm UserExit script. However, bat script does not reside in the TIEREXIT[*N*] directory. Or, you might have a case where a script cannot provide the naming conventions or the expected exit-code conventions. In such cases, use the following translation wrapper-script examples:

Example 1: Call a script that resides in different directory or translate naming conventions

```
GOODOUTPUT="started successfully"
UECMD=$(/path/to/some/service/serviceagent.sh start)
if [[ $AGENTCMD != *"$GOODOUTPUT"* ]]
    then RETCODE=5 #if UECMD output does not contain the expected GOODOUTPUT
        exit ${RETCODE}
fi
```

Example 2: Translate exit codes

```
/path/to/some/service/serviceagent.sh start
if [ $? != 0 ]
   then RETCODE=5 #if the executed command did not return 0
      exit ${RETCODE}
fi
```

CENTRALIZED REPORT OUTPUT AND LOG COLLECTION

SAS_lsm stores both report and log files in the **STATUSROOT** directory that you define in your SAS_lsm configuration file. The following table shows the artifacts you can expect in this location:

Artifact	Description
report.status	Details of the most recent status request
report.start	Details of the most recent start request
report.stop	Details of the most recent stop request
logfiles.BOM	Report listing tier-specific log files in which an error was detected
logfiles/	Subdirectory containing the actual log files listed in logfiles.BOM *
tarfile.*YYYY-MM-DD-HH-MM*.gz	Compressed TAR archive of the logfiles/ subdirectory above *
logfiles.ANALYSIS	Report containing information useful for opening a track with SAS Technical Support *

*Created only when the **-e** option is invoked in the **SAS_lsm** command.

In the event that a start or stop action fails, SAS_lsm creates a Bill Of Materials (BOM) file that identifies tier-specific logs of interest. The utility also creates a BOM file when a status action detects failed tier services on a deployment that was previously started successfully. You can review the generated BOM file to determine what prevented the successful completion of the start or stop action.

When a tier failure is detected, the SAS_lsm utility identifies all the tier-specific log files that have been modified in the last hour. The files are listed in the BOM file (logfiles.BOM) in the **STATUSROOT** directory (as defined in the SAS_lsm configuration file).

If you invoke SAS_lsm with the **-e** option, the utility also extracts copies of the logs to the centralized location that is defined under **STATUSROOT**. Because SAS logs can be large, the utility trims the copied log to reflect only the previous hour(s) of activity (as defined by the TRIMOFFSET variable). These logs are analyzed for errors, and you are presented with an ordered list of these errors.

All collected and trimmed log files are also compressed into a TAR file. You can provide this TAR archive to SAS Technical Support if you cannot diagnose and resolve the issue on your own. Additional information is stored in an analysis file (logfiles.ANALYSIS) that can be used as input if you open a track with SAS Technical Support.

RUNNING SAS_lsm COMMANDS

For the most up-to-date guide about how to run SAS_lsm, and for a list of example commands, see the README.md file for the project on the SAS_lsm GitLab web page. This guide discusses common commands and options with a detailed explanation of what they mean and how they perform when they are executed.

Understanding the proper approach to run a SAS_lsm action depends on understanding the statement that each action makes. Only one action can be executed per SAS_lsm call.

The table below defines what the action options do:

Command Flag	Action Performed	Statement of Action
`-a [NUM]`	Start	Starts services, beginning at tier ***`tier-number`***
`-o [NUM]`	Stop	Stops services, down to, and including, tier ***`tier-number`***
`-s`	Status	Checks the current status of services on all tiers

The logic of SAS_lsm validates that tiers that are not involved in the action are in the proper state to support the action called. Consider an example where an administrator wants to start a deployment from tier 5. To perform this task, submit the following command:

```
SAS_lsm -a 5 -c path-to-SAS_lsm_configuration-file
```

To logically start a deployment at tier 5, you need to ensure that any dependencies on tiers 1-4 are in a started or running state. Therefore, SAS_lsm checks the status of tiers 1, 2, 3, and 4 to verify that they are running. If one of those tiers is detected as not running, SAS_lsm reports a failure and does not attempt to start tiers 5, 6, 7, and so on.

The table below shows some example commands and describes what they do:

Full Command	Statement of Action
`SAS_lsm -a 1 -c /path-to-configuration-file`	Starts services on all tiers. *(Start services beginning at tier 1)*
`SAS_lsm -a 5 -c /path-to-configuration-file`	Starts services beginning at tier 5.
`SAS_lsm -o 1 -c /path-to-configuration-file`	Stops services on all tiers. *(Stop services down to and including tier 1)*
`SAS_lsm -o 5 -c /path-to-configuration-file`	Stops services down to and including tier 5.
`SAS_lsm -s -c /path-to-configuration-file`	Checks current status of services on all tiers

SAS_lsm AND cron CONSIDERATIONS

When scheduling SAS_lsm to run status checks regularly as a cron task, it is possible to inadvertently execute a second instance of SAS_lsm while a task called by cron was already running. SAS_lsm always attempts to detect simultaneous process executions and prevent this, but a potential interaction between the BASH shell and cron may result in this situation being undetected.

In the case of using SAS_lsm via cron, it is recommended that a unique configuration file be used. This file may be a copy of the configuration file used for manual operations, but should have a different file name. The use of separate configuration files results in SAS_lsm generating a different location for its temporary files between the two processes, which prevents situations where the cron-initiated SAS_lsm task is not detected.