Paper 2620-2018

Troubleshooting your SAS® Grid Environment

Jason Hawkins, Amadeus Software Limited

ABSTRACT

A SAS® Grid environment provides a highly available and resilient environment for your business. The challenge is that the more complex these environments become, the harder it can be to troubleshoot incidents and proactively optimize the configuration. Efficient troubleshooting of the environment builds user confidence and enables organizations to get the most value out of their investment.

This paper focuses on the strategies that should be taken to identify and resolve issues within a SAS Grid environment. It provides a best-practice approach for making modifications to the default configuration files to understand what parts of the environment are in effect. The interpretation of this output can then be used to optimize or troubleshoot the environment.

This paper is for SAS administrators who wish to learn more about troubleshooting a SAS Grid environment. It covers both Microsoft Windows and UNIX topologies using both shared and individual configuration directories.

INTRODUCTION

As a SAS Grid Administrator, there are numerous challenges you may face when maintaining and administering the platform over and above what may be experienced when administering a standard, non-Grid platform. Knowing what to expect in certain situations and circumstances and being prepared for them is key, and will make your approach as a Grid Administrator that much more efficient.

In this paper, we first familiarize ourselves with the common architectures of a Grid environment. We then explore the common issues and challenges experienced by Grid Administrators, such as users being unable to submit work, the platform becoming unresponsive and administering and resolving issues whilst adhering to the high-availability policies and agreements in place.

We explore different methods for identifying issues, such as using specific LSF commands and turning up different logging levels. We look at resolving the issues whilst maintaining Grid functionality for end users, with a little downtime as possible.

Finally, we delve into different proactive steps which are designed to prepare your environment ahead of possible failures by enhancing log files and SAS autoexecs.

GRID ARCHITECTURES

In order to efficiently troubleshoot any platform, it is important to understand the topology of the environment and have a good understanding of the role that each SAS server plays. The following section will cover the basic components of a Grid environment.

STANDARD GRID TOPOLOGY

A SAS Grid Platform consists of three tiers, a Metadata tier, a Compute tier and a Middle tier:

- Metadata Tier - controls authentication and authorization, manages information shared by all components in a deployment.
- Compute Tier – manages and executes processing requests for work submitted to the Grid. Consists of multiple “Grid Nodes” and a “Master Grid Controller”.
- Middle Tier – Surfaces SAS Web Applications.

A Grid platform enables SAS sessions to be distributed across multiple Grid nodes. This provides the following key advantages:

- **Workload Distribution**; Work is split across the Grid nodes by a given set of parameters. For example, work may be distributed by the server's individual CPU utilization, or the number of jobs waiting for an execution slot on a Grid node. Workloads can be distributed to specific Grid nodes by using pre-defined queues. This guarantees an allocated amount of resources to high priority jobs.
- **High Availability**; A Grid environment has components which allow you to configure services that monitor others in such a way that the platform can be considered "highly available". Should a Grid service or server fail, another Grid server or service should be started to maintain the platform availability. This high availability extends to SAS jobs which, subject to the configuration, can be configured to resume on alternative Grid nodes following a failure. This ensures as little disruption to the end users as possible.

  This can be achieved using the Enterprise Grid Orchestrator (EGO), which monitors services and in case of a failure restarts them where possible. If unable to restart a service, EGO will bring up the service on another machine.
- **Scalability**; Additional Grid Nodes can be added to the platform so that it is able to adapt to cope with influxes of data or short term intensive analytics, ensuring that the end users have adequate power available. If we require extra resources in a SAS Grid Platform, additional Grid nodes can be added to the platform to facilitate this.

Grid infrastructure utilizes a third-party software component; IBM® Spectrum™ LSF®, which is the component responsible for performing the load balancing of SAS sessions across the Grid nodes. LSF consists of many services, but we focus on the five main services which allow LSF to work with a SAS Grid environment:

- Platform LIM (Load Information Manager) monitors the load on each server and records information about each of the processes sent for execution on the Grid. It includes statistics about the job, such as the time the job was sent for execution, the node it was sent to and whether it's running, pending or ended. This service runs on all Grid nodes.
- Platform RES (Remote Execution Server) accepts remote requests for job execution from the Grid Control Server. This service runs on all Grid nodes.
- Platform SBD (Slave Batch Daemon) receives requests to run jobs and manages the local execution of each job. This service runs on all Grid nodes.
- Platform MBD (Master Batch Daemon or "mbatchd") monitors all jobs across the system. This service runs on the Master Grid Controller only.
- Platform MBSCHD (Master Batch Scheduler Daemon) distributes work (both interactive and scheduled) across the Grid based on job requirements and resource information provided by LIM. This service runs on the Master Grid Controller only.

The following figure presents the location of the SAS components on a standard Grid Topology.

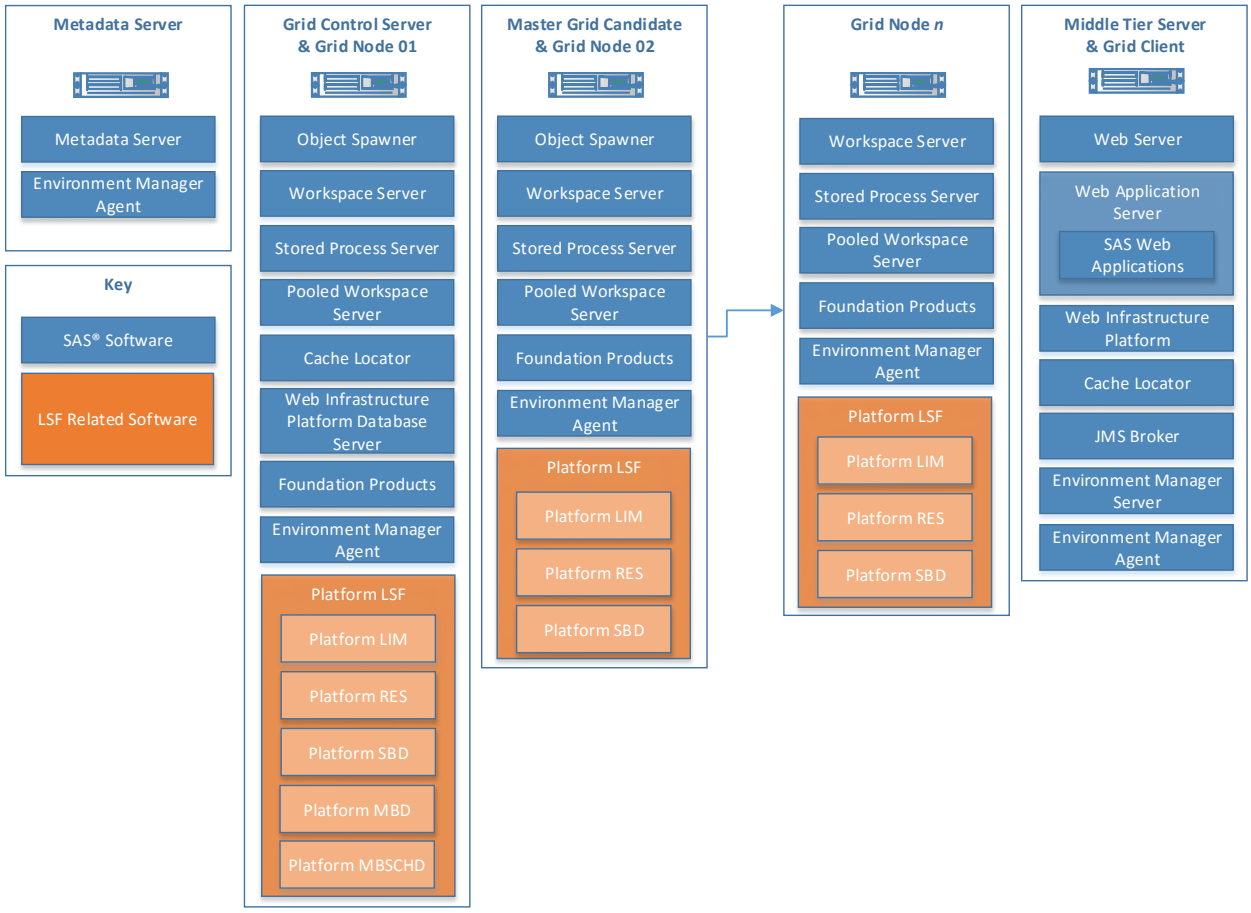

Figure 1. A Standard SAS Grid Platform

Each of the SAS Servers have specific jobs within the Grid:

- Grid Control Server. Monitors the load across the Grid and distributes jobs amongst the Grid Nodes. Does not execute work submitted to the Grid unless it's also configured as a Grid Node.
- Master Grid Candidate. Is capable of becoming the Grid Control Server in the event of Grid Control Server failure. There would generally be multiple SAS Grid Candidates on a Grid platform, but not necessarily every Grid Node. Each Master Grid Candidate needs to be capable of running an Object Spawner.
- Grid Node. Executes work submitted to the Grid.
- Grid Client. Is able to submit work to the Grid, but unable to execute work submitted to the Grid. Web Applications such as SAS Studio submit work to the Grid.

SEQUENCE OF EVENTS – GRID LAUNCHED WORKSPACE SERVER

Figure 2 shows the sequence of events that take place when an end user launches a Grid-launched workspace server through a client application such as SAS Enterprise Guide.

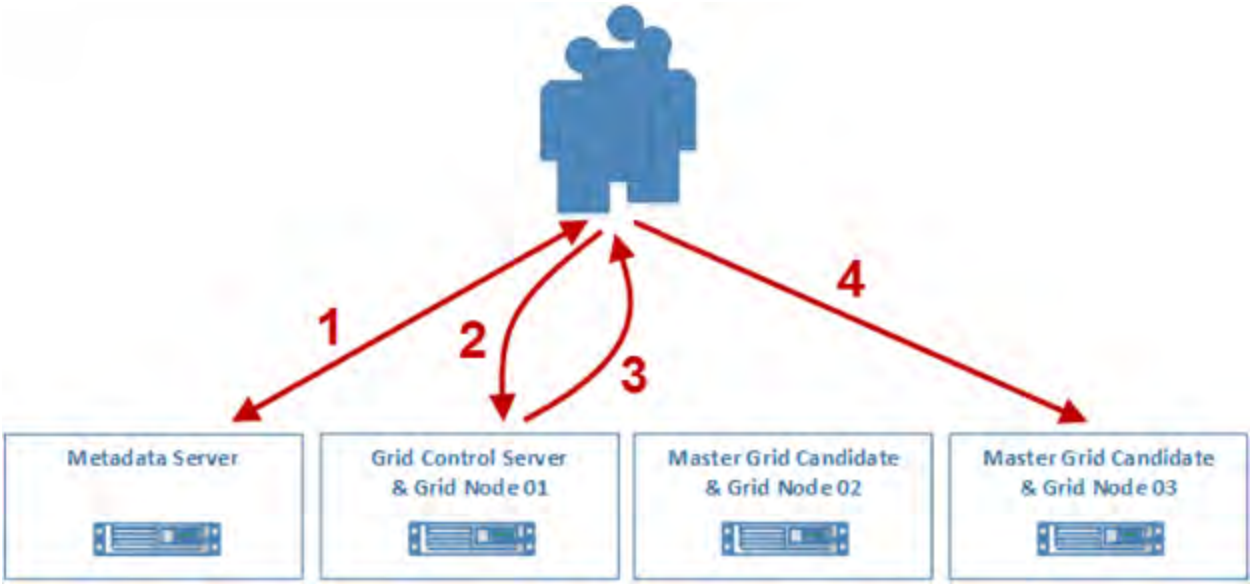

Figure 2 – Grid-Launched Workspace Server - Sequence of Events

The following is a summary of the events that take place:

1. The client application contacts the metadata server. It authenticates the user and retrieves the connection information for the object spawner.
2. The client application then contacts the server which is running the master object spawner. The master object spawner knows that the Workspace Server is configured for Grid load balancing and that it is using Grid-launched Workspace Servers. The master object spawner then:
    a) Contacts LSF to identify which Grid node to start a workspace server session on.
    b) LSF calculates which SAS Grid node is most suitable by using the metrics it collects and the parameters that are set, these include the queues that are set up and which method it has been configured to use to distribute jobs;
    c) On the selected Grid node, LSF begins a workspace server session which the command provided by the master object spawner. This workspace server reports back to the master object spawner to confirm it has started successfully.
3. The master spawner gives the client application the information about the Workspace Server session to which it has been assigned.
4. The client application sends the program for execution.

LOCAL VS SHARED BINARIES AND CONFIGURATION

There is more to a Grid platform than just extra components to consider! A Grid platform can have shared binaries and configuration files, or files local to each server. The location of these SAS binaries and SAS configuration files will determine how you go about troubleshooting a Grid environment. Table 1 shows the advantages and disadvantages of having a local file system when troubleshooting the environment:

Local File System	
Advantages	**Disadvantages**
Hot Fixing can be done on individual Grid nodes without requiring downtime for the platform; causing less downtime for end users.	Log files can be time consuming to locate, with administrators often having to log onto or search multiple servers to find the correct log file. Any configuration changes must be done manually on each Grid node. There is a greater overhead in adding extra Grid nodes to the platform as each additional server needs to be installed and configured individually.

Table 1. Advantages and Disadvantages of Local Installation and Configuration Files

Table 2 shows the advantages and disadvantages of having shared installation and configuration files.

Shared File System	
Advantages	**Disadvantages**
All logs are automatically placed in the central, shared storage location making it less time-consuming to find relevant log files. Platform maintenance is centrally managed – a change to a configuration file will automatically be disseminated amongst all machines. Minimal overhead in adding extra Grid nodes to the platform as the servers can use the existing files.	Hot fixing can only be done to all binaries at once and therefore downtime is required. A shared storage location needs to be available to all Grid nodes which can sustain a high level of throughput, as well as being able to cope with high levels of concurrency.

Table 2. Advantages and Disadvantages of Shared Installation and Configuration Files

Although both topologies are perfectly valid, it is important to be aware of which is in use, as the troubleshooting approach will change slightly.

COMMON APPROACH - TROUBLESHOOTING

With any Enterprise Platform, hardware and software can fail. Although it is not practical to prepare for every possible eventuality, by being aware of the most common failures you can place yourself in a better position to resolve them. Here are some of the most common errors/failures which you might come across in a Grid environment:

- Users unable to launch SAS sessions. For example, users are not able to connect to SAS through SAS Studio or SAS Enterprise Guide;
- SAS jobs being unevenly distributed;
- Jobs unable to be scheduled;
- Resources running out on a specific server;
- Services failing to start;
- Errors in user code.

Depending on the type of error or failure, you might want to start troubleshooting by checking the log files produced or use LSF commands if there are specific issues with the job distribution.

CHECKING LOG FILES

There are numerous log files to be aware of when troubleshooting a SAS Grid environment. LSF logs should be checked if end users were unable to start SAS sessions. In these circumstances, administrators can assume that LSF is struggling to launch the SAS process.

Table 3 shows the LSF logs, the components they relate to, a summary of the information they contain and in what circumstances they would be checked.

Component	Log File	Contents	Checked Under What Circumstance
Platform Process Manager (jfd)	<share>*[1]/pm/log/ jfd.log.<servername>*[2]	Process Manager failover and error logs.	Jobs are unable to be scheduled through Schedule Manager or Flow Manager.
LSF – Events	<share>/lsf/work/<cluster _name>*[3]/logdir/lsb.event s	Keeps track of the state of all current jobs across the Grid.	Running jobs are not executing as expected.
LSF – Previous Events	<share>/lsf/work/<cluster _name>/logdir/lsb.events. *n*	The state of old jobs across the Grid.	Certain jobs have failed prior to starting or during execution. Also used to identify the LSF Job ID. The BHIST command can be used to query these log entries.
Platform LSF – LIM	<share>/lsf/log/lim.log.<se rvername>	Static and dynamic attribute information about individual hosts.	When LIM and other daemons working with LIM (PEM, VEMKD) are having issues starting.
Platform LSF – RES	<share>/lsf/log/res.log.<s ervername>	Status of remote execution requests to the local Grid node.	When sessions will not start across the Grid. For example, a user is unable to launch a Workspace Server session.
Platform LSF – SBD (sbatchd)	<share>/lsf/log/sbatchd.lo g.<servername>	Status of job execution on the local Grid node.	When sessions will not start across the Grid. For example, a user is unable to launch a Workspace Server session. Issues could relate to any of these components. It's recommended to check them in the order shown.
Platform LSF – PEM	<share>/lsf/log/pem.log.< servername>	Information about remote operations of services, for example the starting and stopping of services.	
Platform LSF – MBD (mbatchd)	<share>/lsf/log/mbatchd.l og.<servername>	Status of services on other machines.	
Platform LSF – MBSD (mbschd)	<share>/lsf/log/mbschd.lo g.<servername>	Status of the mbschd service.	
Platform LSF – VEMKD	<share>/lsf/log/vemkd.log. <servername>	Aggregated host information about the state of individual resources and status of allocation requests.	To check the availability of a server. Contains information about servers becoming "Closed".

Component	Log File	Contents	Checked Under What Circumstance
Platform LSF – EGOSC	<share>/lsf/log/esc.log.<servername>	Information about service failures and service instance restarts.	EGO is unable to bring up services which have been configured for High Availability.

Table 3. LSF Log Contents.

*[1] "<share>" resolves to the file share which presents the shared LSF and PM directories.

*[2] "<servername>" resolves to the server on which the process is running.

*[3] "<cluster_name>" resolves to the chosen cluster name.

Table 4 shows the common SAS logs and a summary of the information they contain. These logs would be checked if the users are able to launch SAS sessions but then experiences errors during execution. Note that this is not a complete list, but instead contains the "typical" components.

Component	Contents	Checked Under What Circumstance
SAS Metadata Server	Security events, Metadata server backups, changes to metadata, metadata administration events, all client connections.	User is unable to access metadata objects that they expect to be able to see. Metadata has changed unexpectedly. Metadata backups fail.
SAS Object Spawner	Details of client connections and requests to all SAS servers.	User is unable to launch a SAS session. User is unable to launch Stored Processes.
SAS Stored Process Server	Information about client connections to the Stored Process server and SAS code ran on the stored process server.	Stored Processes not executing as expected.
SAS Workspace Server	Everything that the user has done in their SAS session (turned off by default).	User is not able to view libraries/items set up in Autoexec's that they believe they should. SAS session is not executing as expected.
SAS Web	Information about the individual Web Applications.	Web Applications are unavailable. Web Applications are not executing as expected.
SAS WebAppServer	Information about the individual Web Application servers.	Web Applications are unavailable.
SAS WebServer	Information about the Web Server.	Web Applications are unavailable.

Table 4. SAS Log Locations

If using the Windows Operating System, the Event Viewer can also be useful when troubleshooting issues. Specifically, the “Application” tab listed under “Windows Logs”.

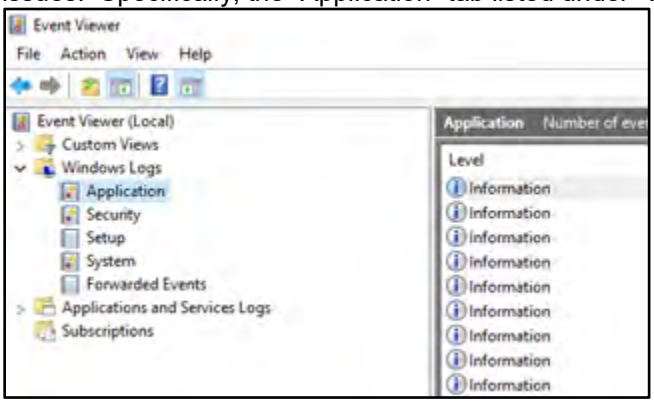

Figure 3 shows this view. Event Viewer is most commonly used to identify failures caused by interruptions to network connections. It can also be used to see errors relating to local services failing.

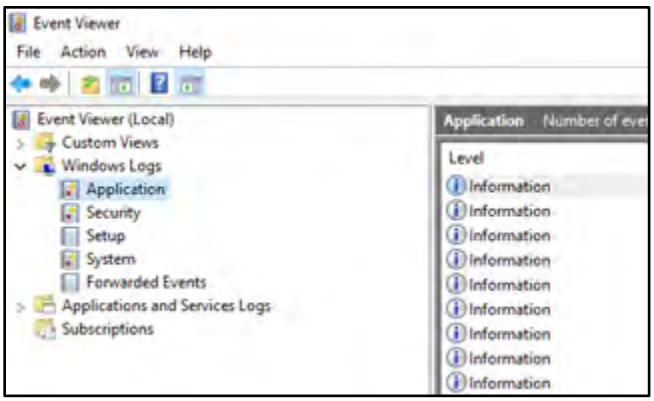

Figure 3. Windows Event Viewer

If using a Linux Operating System, there is no direct equivalent to Windows Event Viewer. LSF will write to the LSF logs where possible, but if it’s unable to write to them or unable to read or find the “lsf.conf” file, messages will be written to syslog.

TURNING UP LOGGING LEVELS – PLATFORM LSF

It is possible to increase or decrease the amount of information contained in the log files produced by LSF. By default, LSF produces logs to a level of “LOG_WARNING”; only messages that are warnings or more serious are recorded. This value is set using the “LSF_LOG_MASK” parameter in lsf.conf as shown in Figure 4.

```
# Daemon log messages
LSF_LOGDIR=/mnt/sas/lsf/log
LSF_LOG_MASK=LOG_WARNING
```

Figure 4. LSF Logging Parameter

When troubleshooting, it can be useful to change this logging level to increase the amount of information recorded in the log files. This would be done if the detail currently being generated in log files was not enough to troubleshoot the issue.

Level	Description
LOG_WARNING	Log only those messages that are warnings or more serious messages. This is the default level of debug information.
LOG_DEBUG	Extra messages appear in log files relating to DEBUG levels.

Table 5. Platform LSF Logging Levels

Changing the "LSF_LOG_MASK" value causes numerous logs created by Platform LSF to be affected.

When changes are made to "lsf.conf", Platform LSF needs to restart LIM and re-read configuration files in order for the changes to take effect. Table 6, below, shows these two commands.

Command	Description
lsadmin reconfig	Restarts Platform LIM
badmin reconfig	Reloads configuration files

Table 6. LSF Reconfiguration Commands

Note that the size of the log files being generated should be monitored to ensure that the disk space used does not become too large. The logging level should be returned to the default level once troubleshooting is complete.

Turning Up Logging Levels Temporarily - LSF

Logging can be turned up and down on an ad-hoc bases without needing to make changes to lsf.conf by using the "lsadmin" and "badmin" commands. This is particularly useful when you don't want to restart the system to alter the logging levels. The following commands set temporary message log level options for LIM, RES, mbatchd, sbatchd and mbschd respectively:

```
lsadmin limdebug [-l debug_level ] [-f logfile_name] [-o] [host_name]
lsadmin resdebug [-l debug_level ] [-f logfile_name] [-o] [host_name]
badmin mbddebug [-l debug_level ] [-f logfile_name] [-o]
badmin sbddebug [-l debug_level ] [-f logfile_name] [-o] [host_name]
badmin schddebug [-l debug_level ] [-f logfile_name] [-o]
```

Where the modifiers include:

- "debug_level" can be "1", "2" or "3" depending on the amount of information to be included. Level 3 contains the most debug information, whereas level 1 contains the least.
- "logfile_name" is the location and name of the log file which will contain the logging information. It is recommended that the name given is one of a new file that doesn't already exist, in order to appropriately isolate messages related to this period of additional logging.
- "host_name" is the host for which information will be logged.
- "-o" turns the temporary logging back off and logging resumes to normal.

This approach can be useful for capturing specific events, at a raised "debug" level in a separate log file. For example, the following command will collect DEBUG1 level information about LIM on host "AVSASLGN1" in a log file called "limdebug.lim.log.avsaslgn1":

```
lsadmin limdebug -l 1 -f ~/limdebug avsaslgn1
```

This command's successful execution can be seen in Figure 5.

```
[root@AVSASLGN1 ~]#
[root@AVSASLGN1 ~]# lsadmin limdebug -l 1 -f ~/limdebug avsaslgn1
Host avsaslgn1 set LIM debug done
[root@AVSASLGN1 ~]#
[root@AVSASLGN1 ~]#
```

Figure 5. Temporarily Turning up LIM Logging

The following figure shows the contents of the log file produced during this period of additional logging:

```
Feb 22 12:11:38 2018 20727 7 3.4.0 announceMaster: send to 1 hosts this period, cnt=2
Feb 22 12:11:38 2018 20727 7 3.4.0 announceMaster: announced to 0 hosts
Feb 22 12:11:43 2018 20727 7 3.4.0 announceMaster: send to 1 hosts this period, cnt=1
Feb 22 12:11:43 2018 20727 7 3.4.0 announceMaster: announcing mastership to host avsaslgn2
Feb 22 12:11:43 2018 20727 7 3.4.0 announceMaster: announcing mastership to host avsaslgn3
Feb 22 12:11:43 2018 20727 7 3.4.0 announceMaster: announced to 1 hosts
Feb 22 12:11:48 2018 20727 7 3.4.0 announceMaster: send to 1 hosts this period, cnt=2
Feb 22 12:11:48 2018 20727 7 3.4.0 announceMaster: announcing mastership to host avsaslgn2
Feb 22 12:11:48 2018 20727 7 3.4.0 announceMaster: announcing mastership to host avsaslgn3
Feb 22 12:11:48 2018 20727 7 3.4.0 announceMaster: announced to 0 hosts
```

Figure 6. DEBUG Logging Example

Turning off temporary logging will return the logging to its default state as soon in Figure 7.

```
[root@AVSASLGN1 ~]#
[root@AVSASLGN1 ~]#
[root@AVSASLGN1 ~]# lsadmin limdebug -o
Host avsaslgn1 set LIM debug done
[root@AVSASLGN1 ~]#
[root@AVSASLGN1 ~]#
```

Figure 7. Turning Off Temporary LIM Logging

TURNING UP LOGGING LEVELS - SAS

If information included in the SAS logs is not detailed enough to aid in resolving an issue, it may be necessary to turn up the SAS logging levels above the default settings.

When considering troubleshooting, SAS receives it's logging instructions from different XML files. This paper focuses on "logconfig.xml" and "logconfig.trace.xml":

- "logconfig.xml" is the default logging instruction, and using this, SAS will produce "INFO", "WARN" and "ERROR" messages. It does this for all SAS components apart from the Workspace server, whose "logconfig.xml" file instructs SAS to not save the logs to any location, in order to save on disk space due to the number of logs it would generate.
- "logconfig.trace.xml" contains instructions to produce the same log information as previously, but with additional "DEBUG" information. Where the Workspace server is considered, these new instructions also tell SAS to save the logs, with the DEBUG information added.

To instruct SAS to use the "logconfig.trace.xml" set of logging instructions, administrators can modify the sasv9_usermods.cfg file. This example uses the Workspace Server. The Workspace Server's sasv9_usermods.cfg file is located in the following location:

```
<config>/SASApp/WorkspaceServer/sasv9_usermods.cfg;
```

The logging level can be altered by using the "logconfigloc" option and specifying "logconfig.trace.xml" as follows:

```
-logconfigloc <config>/SASApp/WorkspaceServer/logconfig.trace.xml
```

When making changes to the “usermods” files, services require a restart in order to pick up the changes. This does not affect the Workspace server, as these are generated on a session-by-session basis.

Be warned that turning up logging levels should only be done when necessary and on a temporary basis. By turning up the logging level, more information is generated in the log, meaning that the space taken up by the log files is bigger. Workspace Server logs are a particular concern, as normally they would not be retained. With this in mind, remember to:

- Monitor the amount of space that the log files are using;
- Ensure that surplus logs are regularly deleted if the logs are turned up for an extended amount of time in hope of recording a certain event;
- Return the logging level to its original state once the issue is fixed;
- Delete remaining log files once they are no longer needed.

REDIRECTING THE LOG FILES – LOCAL CONFIGURATION ONLY

When using a local configuration, by default the log files are generated locally on each machine in its own configuration directory. This can be time consuming when searching for specific log files. If using a shared configuration, the log files are automatically written to a shared directory and hence, this is not an issue.

To redirect local log files to a centralized location, the relevant “logconfig.xml” files can be modified. Note that changes to configuration files such as this can be detrimental to the platform if done in error. It is recommended that backups of files are taken prior to making any changes.

Let’s look into the Object Spawner logging. The following line in the Object Spawners’ “logconfig.xml” file dictates the location where the log file is written:

```
<param name="FileNamePattern"
value="/opt/sas/Lev1/ObjectSpawner/Logs/ObjectSpawner_%d_%S{hostname}_%S{pi
d}.log"/>
```

Alter the location to point to a reserved, shared location. This example shows the file modified so that the log is written to the following location:

```
/sas_share/logs/Lev1/ObjectSpawner
```

Figure 8 shows these changes made in the logconfig.xml file to redirect the log files to another location.

Figure 8. Log Redirection

Remember that as each server has its own local version of the logconfig.xml file, this change should be made on each Grid node in each individual configuration. This results in a single location for SAS administrators to check for log files.

ENHANCING THE SAS LOG

One of the simplest, yet valuable ways to prepare ahead of any issues is to enhance the information included in the log, particularly in the Workspace Server log which will be used as an example. Workspace Server log files contain a copy of the code (and resulting log messages) produced during the sessions' execution. Some subtle additions to the Workspace Server's autoexec files will aid you when troubleshooting by making the sessions more identifiable. The following lines placed in the "autoexec_usermods.cfg" file will tell administrators who created the log, the operating system process ID, what server the program was executing on and importantly, the LSF job ID. Note that the code works in both UNIX and Windows topologies:

```
%put NOTE:  Session info: &sysuserid;
%put NOTE:  Process ID: &sysjobid;
%put NOTE:  Server Host: &syshostname;
%let lsfid = %sysget(LSB_JOBID);
%put NOTE:  LSF Job ID: &lsfid;
```

Figure 9 shows an excerpt of the log produced whilst the above alterations were in place.

```
27
28          %put NOTE:  Session info: &sysuserid;
NOTE:  Session info: hawkinsj
29          %put NOTE:  Process ID: &sysjobid;
NOTE:  Process ID: 64966
30          %put NOTE:  Server Host: &syshostname;
NOTE:  Server Host: AVSASLGN3
31          %let lsfid = %sysget(LSB_JOBID);
32          %put NOTE:  LSF Job ID: &lsfid;
NOTE:  LSF Job ID: 360
33
```

Figure 9. Log Output

The LSF Job ID is a unique identifier which enables administrators to investigate specific LSF jobs. It is different from the Job ID (UNIX) or the Process ID (Windows). This LSF Job ID is useful as it a unique identifier specific for LSF that allows administrators to track jobs from start to finish through the log files. It can be used in conjunction with the lsb.events.*n* files to check job history, as well as using the "BHIST" command as discussed in the section "Enhancing the SAS Log".

Having all of the information readily available in each log provides the SAS administrator (and end user) more information about the job or log file that they're investigating.

Note that Workspace server logging must be turned on in order to retain these logs and the information contained in them, as by default, Workspace Server does not save the created log files. These alterations can be made to other autoexec files (for example, Stored Process Server) to ensure that this information is captured for those components also. To benefit the most from these types of modifications, it is recommended that you consider making these changes to the following server's configuration files at a minimum:

- Workspace Server
- Batch Server
- Stored Process Server

IDENTIFYING THE ISSUE

Troubleshooting becomes much easier when using the information gained from the previous section. Knowing the server on which the SAS Session was executing allows us to check server-specific measures. For example, in Windows the correct server can be investigated and Event Viewer launched.

If when troubleshooting an issue reported by a specific user, searching for the relevant username in log files can aid in identifying the correct log file. If the log file itself doesn't include any indicators as to the cause of the error, the LSF Job ID along with some LSF commands can be used to gain an understanding of why the program may have failed.

There are numerous LSF commands available to SAS Administrators.

EXAMPLE OUTPUTS FOR USEFUL LSF COMMANDS

The following section shows example output produced by some of the most useful Platform LSF commands used when troubleshooting a SAS Grid Platform. We highlight specific output produced and explain what it shows.

BHIST

The BHIST command displays historical information about jobs such as the master Grid controller that submitted it, the queue to which it was submitted, the server it got sent to for execution, the user that submitted it, reasons for job termination as well as the date and time the job was submitted. It is used to find out job history, including any reasons for job failure. Particularly useful when jobs are scheduled. Enables administrators to find out the reasons for terminated jobs. It reads all of this information from the current "lsb.events" file. If a job has failed during execution and exited, the BHIST command will provide administrators with an exit code which can then be used to determine the issue. The BHIST command requires the LSF Job ID as an input.

Figure 10 shows the BHIST command searching for information about LSF Job ID 885. Specifically, it shows that:

1. SAS Enterprise Guide was used to submit the job;
2. User "cowheyj" submitted the job;
3. The job was submitted at 14:22:13 on Thursday, 22nd February.
4. "AVSASLGN1" was the Master Grid Controller at the time of job submission;
5. The job was submitted to "AVSASLGN3" for execution;
6. The UNIX Process ID was "7258";
7. The job ran successfully, finishing at 14:24:53 on Thursday 22nd February.

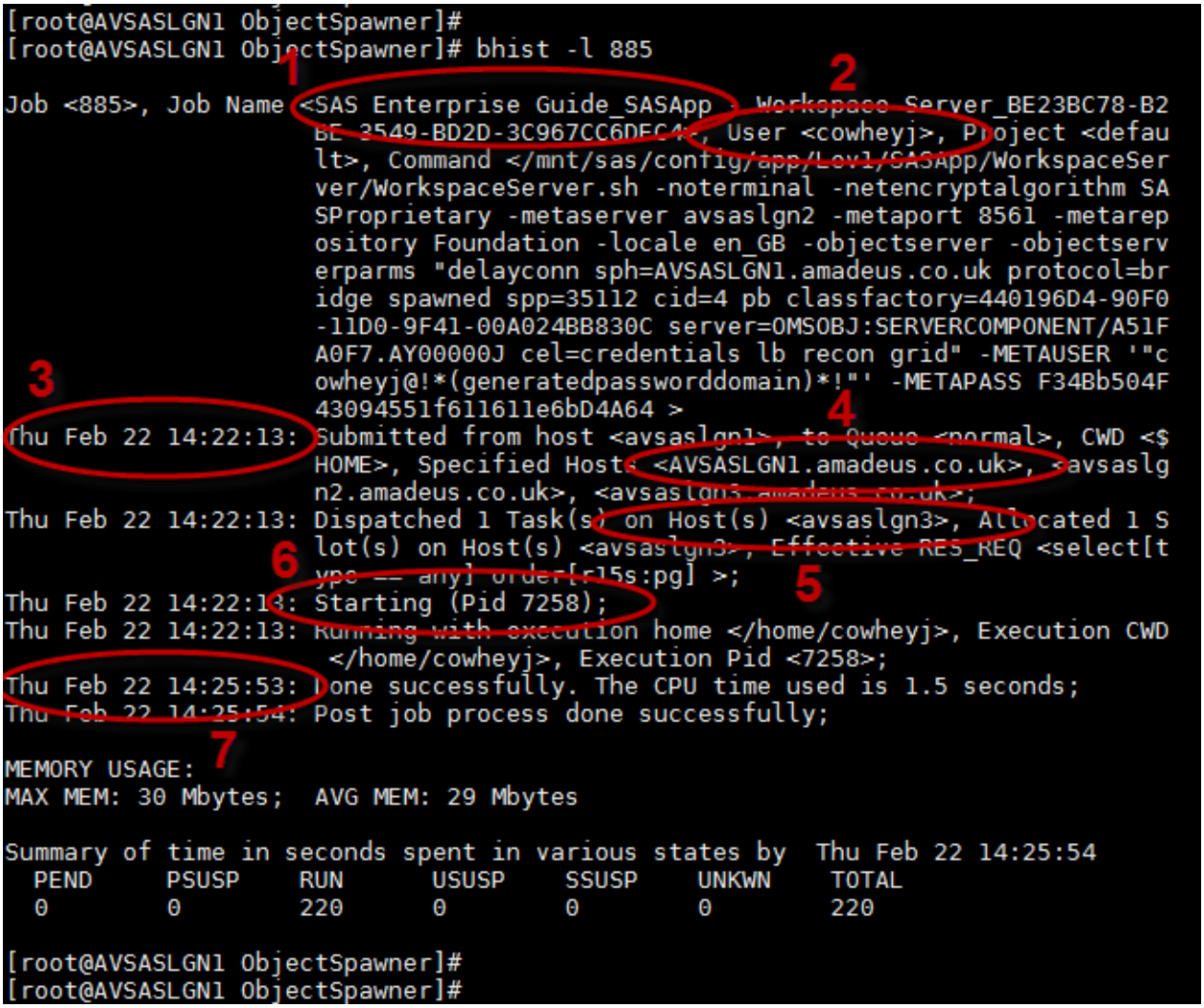

Figure 10. Example BHIST Output

BHOSTS

The BHOSTS command displays hosts and their static and dynamic resources. It can be used to check whether hosts are “open”, “closed” or “unavailable”, the number of jobs currently running across the Grid and number of job slots across the Grid. If the command times out, this usually means that Platform LIM is not started. It is useful when end users are reporting that work is unable to be submitted to the Grid, or that SAS Enterprise Guide sessions cannot be started. It could be because no nodes are available, because the job slots across the Grid are full, or because LIM isn’t up.

If users are able to authenticate but then unable to launch a Workspace Session, it may well be because no Grid nodes are available, or because all available job slots are full. Figure 11 shows that:

1. There are 8 jobs currently running on the Grid, 2 on AVSASLGN1 and 3 on each of AVSASLGN2 and AVSASLGN3.
2. One job which has been sent to AVSASLGN1 for execution has been suspended by a user;
3. There are 48 job slots on each Grid node.

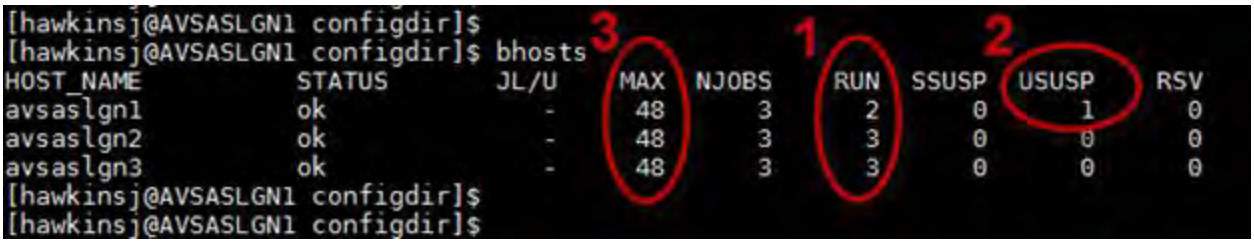

Figure 11. Example BHOSTS Output

BJOBS

The BJOBS command is used to check the status of jobs across the Grid. It is useful when you wish to know who is using the Grid and what Grid nodes their sessions are using. It will show you the state of all jobs across the Grid, whether jobs are running, suspended or pending. It provides details on the user who submitted the job, what client was used to submit the job (e.g. Enterprise Guide), which Grid node submitted the job (Master Grid Controller) and which Grid node the job is executing on. It is useful when you wish to know who is executing work on which server. It's also useful if you want to check what state jobs are in: whether they are pending, suspended or waiting for a job slot. You can query all jobs sent by a specific user by specifying their username (e.g. "bjobs -u cowheyj"). If you wish to see jobs submitted by all users across the Grid, the "-u all" switch must be used. Figure 12 shows:

1. "cowheyj" is the user whose job has been suspended;
2. The date and time the jobs are submitted;
3. "AVSASLGN1" was the Master Grid Controller at the time of submission;
4. All jobs have been sent to the normal queue;
5. The LSF Job ID's span from 363 to 371.

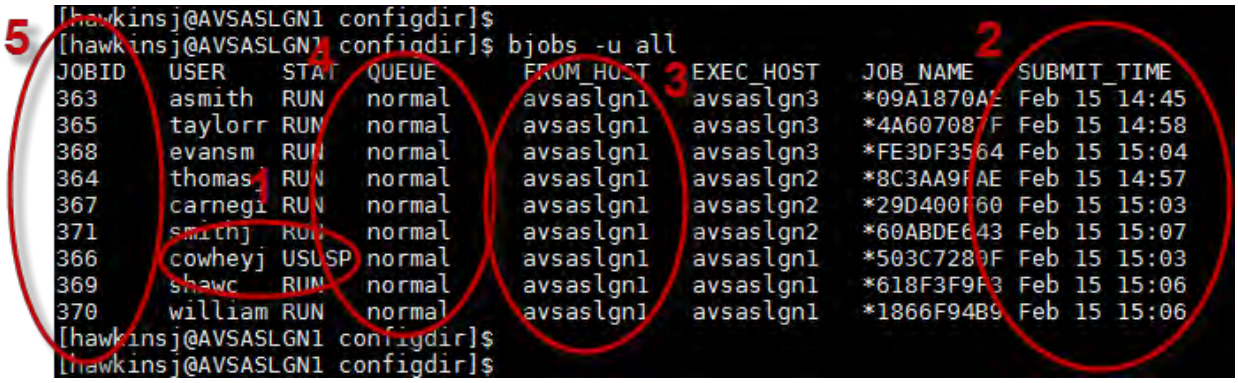

Figure 12. Example BJOBS Output

LSID

The LSID command displays the current LSF version number, the cluster name and the master host name. It is useful when you are unsure which Master Grid Candidate is currently the Master Grid Controller. Also, it will return "Host does not have a valid software license" if Platform LSF is not licensed. Figure 13 shows:

1. The LSF version is 10.1.0.3;
2. The cluster name is "sas_cluster", the Master Grid Controller is "AVSASLGN1".

Figure 13. Example LSID Output

Other Useful LSF Commands

Table 7**Error! Reference source not found.** shows some further LSF commands most commonly used when troubleshooting a Grid platform.

LSF Command	Usage	Aids in Resolving
lsload	The LSLOAD command displays load information for hosts, including CPU usage.	If users are experiencing performance issues, it could be because the CPU is over-utilized on a certain Grid node. The LSLOAD command enables administrators to view the CPU utilization across the Grid. The LSLOAD command is also useful if jobs seem to be being sent to the same node.
bsub e.g. bsub sleep 30	Submits a job to LSF.	If users are unable to launch jobs on the Grid, the bsub command can be used to launch jobs to quickly test the functionality and job distribution.
bstop / bresume e.g. bstop 273 bresume 273	Suspends unfinished jobs.	If a job is using a lot of resources which are required elsewhere, the job can be paused using the “bstop” command and specifying the LSF Job. This will force the job into a “USUSP” state – User Suspended. Jobs that have been paused using “bstop” can be resumed by issuing the “bresume” command.
bkill e.g. bkill 273	Sends signals to kill, suspend, or resume unfinished jobs.	If a job is running on the Grid which needs to be stopped, the “bkill” command can be issued and the LSF Job ID specified.

Table 7 - Commonly Used LSF Commands

METHODS FOR TROUBLESHOOTING SAS ISSUES

Once you’ve submitted the LSF commands listed in the previous section and the output/results are what you’d expect to see, if the issue persists you can assume the issue lies within SAS.

Assuming that all SAS services are running as expected, issues are most commonly caused by items in metadata or errors in configuration files. In order to troubleshoot these, you would first try launching SAS Enterprise Guide yourself. When launching Enterprise Guide, SAS will read all of the relevant configuration files in order to set the correct settings for your session. It will also query metadata and assign your access to different metadata objects based on authorizations set.

Assuming that workspace logging is already turned up to the “trace” level and that you have access to an account with identical OS permissions to the end user reporting the issue, you can launch

“workspaceserver.bat” (Windows) or “WorkspaceServer.sh” (UNIX) from command prompt/PowerShell (Windows) or bash (UNIX).

Taking this step back and launching the workspace server manually allows administrators to isolate different settings which are normally picked up during the launching of an Enterprise Guide; configuration settings or access to Metadata objects. A traditional workspace server command is as follows:

```
<config>/Lev1/SASApp/WorkspaceServer/WorkspaceServer.sh
```

Launching this “WorkspaceServer.sh” has a domino effect as it contains its own set of instructions which in turn, call different files which make up the SAS configuration. Knowing this, administrators can dissect and omit specific commands. For example, administrators can launch a Workspace Server session without querying metadata by specifying a non-existent SAS Application Server context to the “metaautoresources” switch as follows:

```
<config>/Lev1/SASApp/WorkspaceServer/WorkspaceServer.sh –
metaautoresources none
```

If a session is started without errors, administrators can assume that the issue lies within metadata and can begin their investigation there, as metadata is the likely cause.

Conversely, administrators can opt to launch a workspace server session without picking up any specific autoexec settings by specifying the “-noautoexec” switch as follows:

```
<config>/Lev1/SASApp/WorkspaceServer/WorkspaceServer.sh -noautoexec
```

If this session starts successfully without errors, administrators can assume that the issue lies within settings specified within the autoexecs.

RESOLVING THE ISSUE

Once the issue has been identified, you can begin the task of resolving it. Sometimes this can be something as simple as altering a configuration file, which occasionally can be done immediately without the need to restart services.

However, if the problem lies with a specific Grid node (hardware, local files), it may be necessary to take that Grid node out of the Grid whilst the issue is resolved. There are multiple ways of doing this.

The most straight forward way of taking a node out of the Grid is to “close” the server itself from Platform LSF’s perspective. Once this is done, jobs will no longer be sent to that Grid node for execution.

To “close” the server from Platform LSF’s perspective, the following command should be issued from the server which the administrator wishes to close. The command is shown, alone with the expected output:

```
badmin hclose
Close <avsaslgn1> ...... done
```

The closing of this server can be checked by issuing the “bhosts” command. “closed” should appear under the “status” column next to the desired server:

```
[root@AVSASLGN3 ~]# bhosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
avsaslgn1          ok              -     48      0      0      0      0      0
avsaslgn2          ok              -     48      0      0      0      0      0
avsaslgn3          closed          -     48      0      0      0      0      0
[root@AVSASLGN3 ~]#
```

Figure 14 - Closed Grid Node

Once this has been done, no new sessions will be sent to the Grid node. However, there may be existing sessions which need to be stopped prior to carrying out any essential work. These can be identified using the “bjobs” command and specify information for all users using the “-u all” switch. Identified sessions can be stopped by using the “bkill” command available through LSF followed by the LSF Job ID:

```
bkill 273
```

Once the work has been carried out and the server functionality assured, the server should be reintroduced to the Grid. This is done by “opening” the server. The following command should be issued on the server the administrator wishes to open. The command is shown, along with the expected output:

```
badmin hopen
Open <avsaslgn1> ...... done
```

An LSF administrator is able to open and close servers from another server’s terminal by specifying the name of the server after the command, for example:

```
badmin hclose AVSASLGN2
Close <avsaslgn2> ...... done
```

Note that there are other methods for removing a server out of the Grid. If designed with high availability in mind, queues may be available for different host groups. It would then be possible to force users into specific queues and switch existing sessions to run in specific queues, freeing up specific hosts. For example, queues may be created which references the specific hosts, allowing the remaining hosts to be brought out of the Grid for maintenance.

PROACTIVE STEPS

The following section outlines steps which you should take in order to proactively monitor your Grid platform, ahead of any errors occurring.

SAS GRID MANAGER – SAS ENVIRONMENT MANAGER

When running enterprise scale software, it is best practice to monitor the platform whenever it is in use. This can be done for a SAS Grid Platform by utilizing the SAS Grid Manager plug-in for SAS Environment Manager which uses the Platform Web Services component.

Not only does the Grid Manager plug-in for SAS Environment Manager enable you to monitor the Grid, it also allows you to control certain aspects of the Grid through SAS Environment Manager, as opposed to doing this through the configuration files.

In addition, alerts can be set which will notify you if resources exceed certain thresholds.

ENTERPRISE GRID ORCHESTRATOR (EGO)

Platform LSF’s “EGO” service (Enterprise Grid Orchestrator) should be configured to monitor and restart services where possible. By default, EGO monitors the status of the Master Grid Controller and the Master Grid Candidates through the Platform LSF services only. It’s recommended that this component

is modified to monitor the status of the Object Spawner at a minimum, however it can be modified to monitor all SAS services.

EGO can actually be used to monitor (and commonly administer - start/stop) any service running on your Grid nodes – even services not related to SAS!

CONCLUSION

Once an error occurs within a Grid environment, it can be challenging to track down the relevant information without a logical methodology to follow.

Take time to familiarize yourself with the main components of the platform and have a methodology in mind for troubleshooting ahead of time. Knowledge of log locations and LSF commands is recommended; it ensures that troubleshooting can be performed in an efficient and safe manner.

Design and modify the Grid configuration in such a way that you are prepared for a failure. Preparing for this at its most simple level can be done by modifying configuration files. This ensures that should an issue arise, the information you require to resolve is easily and readily available. At a more advanced level, EGO can be configured to monitor services and maintain high availability. Alerts and thresholds can be set using SAS Environment Manager.

REFERENCES

IBM Knowledge Center. "IBM Platform LSF Command Reference." Accessed February 15, 2018. https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.2/lsf_kc_cmd_ref.html

IBM Knowledge Center. "Logging and Troubleshooting." Accessed February 15, 2018. https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_admin/lsf_ego_logs_troubleshoot.html

Platform Computing Inc. "Managing LSF on Platform EGO." Accessed February 15, 2018. http://capsella.ccs.yorku.ca/homepage/lsfhpc/admin/lsf_on_ego.html

IBM Knowledge Center. "Set Daemon Message Log to Debug Level." Accessed February 15, 2018. https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.2/lsf_admin/troubleshooting_daemon_debug_level_lsf.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jason Hawkins
Amadeus Software Limited
Mulberry House, 9 Church Green, Witney, Oxon OX28 4AZ
+44 (0) 1993 848010
Jason.Hawkins@amadeus.co.uk
www.amadeus.co.uk