**SAS111-2014**

# UNIX Utilities and What They Can Do for You

Jerry Pendergrass, SAS Institute Inc., Cary, NC

## ABSTRACT

The UNIX host group delivers many utilities that go unnoticed. What are these utilities, and what do they tell you about your SAS® System? Are you having authentication problems? Are you unable to obtain a result from a workspace server? What hot fixes have you applied? These are subjects that come up during a SAS Technical Support call. It would be good to have background information about these tools before you have to use them.

## INTRODUCTION

SAS is a very sophisticated system. The UNIX installation provides many tools and proxies that are used to support, control, and manage SAS. This paper discusses three major areas that enhance the management of SAS. These are areas are authentication, revision management, and installation management.

## LOCATION OF THE UNIX UTILITIES

When SAS is installed, its entire directory structure is located in a directory on your file system. This directory is referred to as **SASHOME**. Located as a subdirectory in SASHOME is the SAS Foundation directory, which is referred to as **SASROOT.** This directory can be found in **SASHOME/SASFoundation/9.x**, where **x** is the SAS release number. Any mention of !SASROOT in this paper is referring to the SASROOT directory installed within SASHOME.

### AUTHENTICATION

Authentication is used in SAS to validate that a client has the access rights to do the job requested and has the correct credentials to access data that the client wants to access. This identity can be used to in SAS to configure the environment to the requirements of that specific identity. An example is the different information that a sales manager would want access to versus a quality assurance manager. They log on to the same system, but the look and feel are different based on their requirements and configuration.

There are three proxies that are used to implement authentication in SAS:

1. sasauth
2. sasperm
3. elssrv

These three proxies share one important property, they are all setuid root processes. The reason for this requirement is described for each proxy. The SAS position of the use of setuid can be found in the “Related Reading” section at the end of this paper.

### SASAUTH

The sasauth utility is used by SAS 9 servers to perform authentication for connecting clients. The default method for this authentication is the host operating system authentication. Most sites that use host operating system authentication deploy a shadow password file setup. **In order to read the password entries containing the hashed password from the shadow password file, the calling user ID must be root.** If the sasauth executable is not setuid root, the system would not be able to authenticate the users accessing the servers.

The sasauth proxy is used primarily to authenticate access to a SAS server, such as the workspace server, SAS Metadata Server, SAS Stored Process Server, or SAS/SHARE® server.

The administrator is allowed to change the default authentication from host operating system to other methods of authentication such as PAM or LDAP. The file sasauth.conf is used by sasauth to manage the authentication requirements. Note that the sasauth.conf file must reside in the same directory as sasauth. In the sasauth.conf file there is a line:

```
methods=pw
```

This line directs the type of authentication to be user/password, using the operating system for validation. Other methods are:

1. `methods=pw` Use standard /etc/passwd - /etc/shadow authentication. On some hosts, this also includes protected password databases or OS-provided enhanced security.
2. `methods=pam` Use PAM for authentication. The password database is also used to determine the user's UID and GID. The pam.conf file must be configured properly for sasauth.
3. `methods=ldap` Use LDAP authentication. Be sure to define the LDAP parameters.
4. `methods=ext` Use a custom authentication mechanism. This mechanism is built using the authentication kit, available from SAS Technical Support.

## LOGGING SASAUTH OUTPUT

The sasauth.conf file allows the administrator to log sasauth operations. To turn on the logging, remove the '#' in front of the appropriate log variable and give a valid location for the log file.

The text below is taken from a sample sasauth.conf file.

```
#debugLog=/tmp/sasauth-debug.log
#accessLog=/tmp/sasauth-access.log
#errorLog=/tmp/sasauth-error.log
#logOwner=0
debugNoPasswords=true
```

**Log Definitions Taken from sasauth.conf**

This is a description of the available logs:

1. debugLog Path for the debugging log. This is intended for debugging only. Do not use debugLog regularly in a production environment, as the log can become very large.
2. accessLog Path for the access log. All authentication operations and the result (ok, expired, or error) are logged in this file.
3. errorLog Path for the errorLog. If unspecified, errors are sent to syslog.

Two variables that are important can be set in sasauth.conf:

1. `#logOwner=0` Define the owner UID of the debug log file. The default owner is root.
2. `debugNoPasswords=true` This controls whether passwords are written to the debug log.

Below is the output from two authentication attempts. The first succeeds, but the second is an unknown user.

```
20140202-09:44:40 Authenticated user user1 (pw).
20140202-09:44:54 Unknown user mickey.
```

**Sample Output for sasauth-access.log (Note that this log is readable by anyone.)**

The sasauth-debug.log shown below is readable only by root unless LOG_OWNER=<uid> is specified.

```
20140202-09:44:40 Adding auth method pw
20140202-09:44:40 Initialized 1 methods.
20140202-09:44:40 Authenticating user user1 via pw
20140202-09:44:40 Authenticating user user1 via password database
20140202-09:44:40 Using crypt()/bigcrypt()/crypt16() encryption.
20140202-09:44:40 Checking for expiration.
20140202-09:44:40 Getting user's group memberships
20140202-09:44:40 User sasiom1 in 2 groups.
20140202-09:44:40 Authenticated user user1 (pw).
```

```
20140202-09:44:54 Authenticating user mickey via pw
20140202-09:44:54 Authenticating user mickey via password database
20140202-09:44:54 User not in /etc/passwd.
20140202-09:44:54 Unknown user mickey.
20140202-09:44:54 Request failed: 'Unknown user.'
```

**Sample Output from sasauth-debug.log**

## SASPERM

The sasperm utility performs host authorization checks against files on disk in the SAS/SHARE Server and SAS Stored Process Server. This process uses a combination of the stat() system call and the access() system call to determine if a user should have access to given file. The utility must switch identity to the requesting client to perform these calls as the user requesting the access. Therefore, these calls must be run as root.

## ELSSRV

The object spawner uses a setuid root utility called elssrv to launch processes under the identity of the requesting client (a standard workspace server) or a multi-user credential (a load-balanced stored process server and a pooled workspace server). **The user ID must be root in order to switch identity to another user.** In the standard workspace server case, the client provides host credentials for the user requesting the SAS process, such as a query or an ETL process, to the spawner. The spawner host authenticates the client and receives confirmation of valid credentials from sasauth. In addition, sasauth returns the UNIX user ID and list of groups. The elssrv utility launches the workspace server under this identity so that the process runs with the host authority of the requesting client. Note that the elssrv process also requires the credentials to do a second validation of the user authority to prevent misuse of the elssrv process.

In the case of a SAS Stored Process Server or a pooled workspace server, the spawner uses elssrv to launch processes under a chosen credential that is stored in metadata and is associated with the server. For a SAS Stored Process Server, clients are authenticated by the host before being allowed to run a SAS process on one of these servers. The pooled workspace servers do not require host authentication because processes that run on these servers are in a much more controlled environment. The SAS Stored Process Server host authenticates the connecting clients using sasauth and obtains the clients user ID and groups.

Note that the elssrv process will not show up in a `ps -ef` command as elssrv. It shows up as sasels if the process is not a SAS server or as saslesrv if the process is a SAS server. The path of execution is not shown.

### ELSSRV LOGGING

The elssrv process allows environment variables to be used to turn on elssrv debugging. This is very useful when a SAS BI Client installation is having problems launching a SAS BI Server or when the server is exiting or timing out for no known reason. The els log provides detailed information about what the servers are requesting and about the exit status of these servers.

### ELSDEBUG

ELSDEBUG is an environment variable that instructs the elssrv process to log its activity. This is very useful in debugging systask commands, launching a SAS server, or using the X command or filename pipe.

```
        setenv ELSDEBUG 1          (csh)
or
        export ELSDEBUG=1          (ksh)
```

**Syntax for Setting the ELSDEBUG Environment Variable**

### ELSLOG <path>

The ELSLOG environment variable allows you to specify the path for the output for logging. The name of the file is appended with ".els.pid". If you specify your path as /tmp/techsupport, the output will appear in /tmp/techsupport.els.2456. Where 2456 is the process ID for the server.

The server communicates the exit status of processes through a socket connection. By default, the port number is obtained from the operating system. If the administrator wants to limit the range of the port values that can be selected, the –MINPORT and –MAXPORT options can be used to set a range for the selection of the port used by elssrv.

```
-minport #
```

```
-maxport #
```

The –MINPORT option sets the lower range and the –MAXPORT option sets the high range for the port values that can be used by elssrv. If a port cannot be found in this range, then tkels will fail to load, which will cause SAS to fail to start. The maximum port number is 64*1024-1.

## THE SASUMGMT UTILITY

This sasumgmt utility is used to validate that a user name and password is valid and can be used for SAS authentication. This tool is used to debug any issues dealing with a user's ability to authenticate and is used during installation to validate the username/passwords that are entered for server configuration. If these passwords are not verified during the installation process, then the start-up of a server might fail. This results in a long process to determine the issue and then correct it. Validating the passwords up front makes sure the user can authenticate into the server once it is running.

The sasumgmt utility can be run at any time a user is experiencing problems with authentication. If a problem is found, then the debug logs for the sasauth authentication proxy can be used to isolate the authentication problem. See the section on debugging with sasauth for more information.

The sasumgmt utility can also be used to validate that a user has access to a specific file or directory. This is used during the installation process to make sure a user has access to files they need while running a server.

The syntax for running sasumgmt is given below. Note there are multiple ways to enter the password. It can be specified in plain text, through standard input, or given as a SAS encoded password. The –V option is useful when running to detect a problem with authentication. This –V option instructs the program to write any detected errors to standard output. The default behavior is to return any errors in the exit status of the process. This is the method used for installations.

### Execution:

The sasumgmt utility is installed in the SASROOT area of a SAS installation. !SASROOT represents the location of the SASROOT area.

```
        !SASROOT/utilities/bin/sasumgmt  -u username  -p plain_text_password
                                         -u username  -p {*}encoded_password
                                         -u username  -stdio
                                         -read     <pathname>
                                         -write    <pathname>
                                         -execute <filename>
                                         -re      <filename>
                                         -rw      <filename>
                                         -rwe     <filename>
```

Return codes for sasumgmt are listed below:

- The user was authenticated. User name and password are good
- Access denied. The user name and password do not authenticate.
- The Password has expired.
- The sasauth service is not SETUID root as required by the installation.
- The tksecure authentication service encountered a problem.
- The program cannot boot tk. It is likely that the program has not been patchnamed to the installation directory.
- A User name was not supplied.
- A Password was not supplied.
- Failure on reading the password from standard IN.
- Insufficient arguments to program.
- The user name or password could not be transcoded into UNICODE.
- Memory could not be allocated to perform the authentication.
- The password could not be decoded.

- The thread handle could not be obtained.
- The user does not have the desired access to the pathname.
- The filename was omitted from the argument list when specifying -read/-write.

Below is an example execution of sasumgmt using standard input to enter the password and requesting a verbose output.

It shows that the password was not valid.

```
%sasumgmt -u guest  -stdio -v
password
NOTE: [sasumgmt: SASUMGT_STATUS_ACCESS_DENIED]
NOTE: Access denied.
```

**Syntax for sasumgmt (Invalid Password)**

This invocation shows that the password is valid but the directory is not accessible by the user. A SAS002 password is used.

See the discussion of pwencode below for an explanation for the SAS002 encryption format.

```
%sasumgmt -u guest-pw SAS002}DBCC571245AD0B31433834F80BD2B99E16B3C969 –v -rw
/etc/password
NOTE: [sasumgmt: SASUMGT_STATUS_DIR_ACCESS]
```

**Syntax for sasumgmt Testing Access to /etc/password**

## PROC PWENCODE

You can use the PWENCODE procedure to encode a password to be given to SAS authentication. This provides multiple methods to create encrypted values for user passwords. This example shows the SAS002 algorithm, which is shipped with Base SAS.

```
  1?  proc pwencode in='my password' method=sas002; run;

{SAS002}DBCC571245AD0B31433834F80BD2B99E16B3C969
```

**Create an Encode Password Using PWENCODE**

The encoding methods supported include;

| | | |
|---|---|---|
| 1. | SAS001 | Uses base64 to encode passwords None |
| 2. | SAS002 (or sasenc) | Uses a 32-bit key to encode passwords default SAS Proprietary |
| 3. | SAS003 | Uses a 256-bit key to encode passwords AES (Advanced Encryption Standard), which is supported in SAS/SECURE®. |

SAS001 and SAS002 are supplied with Base SAS.

## REVISION MANAGEMENT TOOLS

SAS consists of thousands of executable images. The installation provides tools that can help manage the set of images installed at a customer site. This becomes very important when a customer has an issue with a product in the field. SAS Technical Support needs to know which images are installed and which release level they were built from. The current process of collecting all of the necessary images for a release requires obtaining images from many different development groups. The final product represents the work of these various groups.

Every image in an installation has a signature that contains the necessary information to determine which group was responsible for each image and which release it belongs with.

## THE VERCON UTILITY

The vercon utility is a stand-alone executable that can be run against SAS images for a given installation. This utility provides detailed information about those SAS images. The tslvl user-written function can be used in the DATASTEP to access the same information programmatically. SAS Technical Support uses both methods to access information about installed images. These tools can be used by anyone who wants to understand what releases, hotfixes, and maintenance images are installed.

**Execution:**

```
!SASROOT/utilities/bin/vercon {-l outputfile} -{ d f h i n p q s t v}  <images .....>
```

Arguments and Options:

The options can be in either case.

| | |
|---|---|
| -l <output file name> | Filename for output. The default is standard output. |
| -b | Print product:dvd/script/scriptname.bld. This can be used with -i also to print the information about the image. |
| -d | Print the time at which the image was built as a date and time. |
| -f | Do not format the output. |
| -h | Print hotfix information. |
| -i | Print information about the image. |
| -p | Print the product name. |
| -q | Do not print the image name. |
| -s | Print the version information. |
| -t | Print the track name. |
| -dir  <Directory> | Sometimes the number of images are too large to list.  So process in the specified directory. |
| <images> | A list of images to evaluate. |

The default is to print the most relevant information that is embedded for a given image. This is an example of the default output for vercon run on the executable tkmemtst.so.

```
   % vercon tkmemtst.so


   [  1] ./tkmemtst.so  PortDate  : 9.03.01B0D11092009
      Track            : dev/mva-v940
      Epoch Time       : 1257879451
      Date             : Tue Nov 10 13:57:31 2009
      Product Name     : test
      Script Name      : tkmemtst
      Script DVD       : tktest
```

The information given includes the date on which the image was built, the product name, the release track and the group that provided the image.

Additional executions can be seen below:

```
vercon -p tkmemtst.so
[  1] ./tkmemtst.so        PortDate          : 9.03.01B0D11092009


 vercon -bi tkmemtst.so
 [  1] ./tkmemtst.so        test:tktest/script/tkmemtst.bld:memory test extension


vercon -ptn tkmemtst.so   (display port date  and track with no titles)
```

```
 [  1] ./tkmemtst.so        9.03.01B0D11092009
                                                      dev/mva-v930


vercon -pin  tkmemtst.so   (display port name, no image , no titles)
9.03.01B0D11092009


vercon -d  *.so    (Display date)

   [  1] ./docnavipkg.so      Clock Time       : Wed Nov 11 09:07:18 2009
   [  2] ./docsearpkg.so      Clock Time       : Wed Nov 11 09:07:18 2009
   [  3] ./docutilpkg.so      Clock Time       : Wed Nov 11 09:07:18 2009
   [  4] ./docviewpkg.so      Clock Time       : Wed Nov 11 09:07:18 2009
   [  5] ./gendevio.so        Clock Time       : Wed Nov 11 09:07:19 2009
   [  6] ./generext.so        Clock Time       : Wed Nov 11 09:07:19 2009
        File [gridplat.so] Entry [exe_version] not available: [ld.so.1: vercon: fatal:
         exe_version: cannot find symbol]
   [  7] ./lcebin.so          Clock Time       : Wed Nov 11 09:07:19 2009
   [  8] ./lceda.so           Clock Time       : Wed Nov 11 09:07:19 2009
...
File [tkears.so] Could not be loaded: [ld.so.1: vercon: fatal: libdfintgccl-8.2.so:
open failed: No such file or directory ]
```

Note that if a file does not have extracted information, an error is reported.

## THE TSLVL USER-WRITTEN FUNCTION

In order to handle the variety of information to extract from the modules, the new TSLVL function will take a second argument that will be a list of quoted options. The syntax will be similar to how the SCAN Function takes a quoted string in the third argument, and these options are NOT case sensitive.

Options:

| | |
|---|---|
| A | Track Additional |
| D | Port Date (ex: 9.03.01B0D02282010) |
| E | Number of Seconds since Jan 1, 1970. |
| H | Hotfix |
| I | Description |
| M | Maintenance |
| P | Product Number |
| T | Track Look through |

Output will be returned with one character string with each type of information that is requested, separated with commas so that it can be easily parsed. The example below executes tslvl twice. The first instance gives the release number, in the case SAS 9.4 Technical Support level 1 Maintenance Level 0. (This was the initial release of 9.4.) The second instance asks for the track where the image was built, the date on which the image was built, and the UNIX time the image was built. So this image was built at the main development for SAS 9.4 development on June 21, 2012.

```
data x;
            y=tslvl('sasxkern');
            put y=;
            a=tslvl('sasxkern','tde');
            put a=;
            run;

y=9.04 TS1M0
a=dev/mva-v940,9.04.01M0D06212013,1371856718
NOTE: The data set WORK.X has 1 observations and 2 variables.
```

**Sample tslvl Execution**

For those modules that cannot load or have version information, there are 3 other possible text strings returned from the function:

- ImageNotLoaded - when the module is not loaded.
- VersionNotFound - version information could not be found for the module.
- NotAvailable - catch any other situation when having problems locating version information.

## INSTALL MANAGEMENT UTILITIES

The installation of the SAS product is a complex process. The installation process attempts to validate that the system is configured correctly for the installation. The sasumgmt tool described earlier in this paper is used to validate the username/password entries that are used to configure a server. It is best to catch an incorrect authentication at this point of the installation so that the username/password are correct by the time the server starts operation and tries to authenticate a client.

### ELSCONF

Another tool used during installation is elsconf. This tool is used to validate that the elssrv proxy mentioned earlier will operate correctly on your system. The syntax for elsconf is shown below.

```
cd !SASROOT/utilities/bin
./elsconf –v
```

**Syntax for elsconf**

The correct response should be

```
NOTE : Your Operating System is configured correctly to support the SAS Launch
Service.
```

**Correct Response to elsconf -v**

The different failures that can be detected are

```
NOTE:  The launch server ‘elssrv’ cannot be found
    :  The launch server ‘elssrv’ is not running as UID=ROOT, but is setuid ROOT.
       This may be due to SAS running in an NFS partion
    :  The launch server ‘elssv’ is not running UID=ROOT. This will prevent SAS
from
       Working as a Server.
    :
```

**Error responses to elsconf –v**

### PATCHNAME

The installation of a SAS foundation needs to record the location of !SASROOT and several paths that are used to locate executables. This information is stored in a file called tkdef.so. It is located in the same directory as the main executables. The foundation installation places tkdef.so in two places, one in !SASROOT/sasexe and one in !SASROOT/utilities/bin. The one in !SASROOT/utilities/bin is a soft link to the one in sasexe. This file has only one purpose and that is to contain the installation directory information.

This information is determined at run time. The information is stored in the tkdef.so file using the patchname utility. This is the utility that can also be used to add restricted user IDs to the elssrv proxy.

The patchname utility can also be used to display the information about the installation location.

Example execution syntax for the patchname utility is shown below.

```
!SASROOT/utilities/bin/patchname !SASROOT/sasexe/tkdefs.so
```

```
!SASROOT/utilities/bin/patchname !SASROOT/sasexe/tkdef.so  !SASROOT
!SASROOT/utilities/bin/patchname –tk !SASROOT/sasexe/tkdef.so !SASROOT/sasexe
!SASROOT/utilities/bin/patchname –blockuid  !SASROOT/utilities/bin/elssrv 0:612
```

**Syntax for the patchname Utility**

If you specify just the tkdef.so argument, patchname will display the value of !SASROOT.  If a second argument is given, it will be stored in tkdef.so as the !SASROOT. The -TK option displays and sets the location for the Threaded Kernel library used by the SAS foundation. The -BLOCKUID option is used to restrict the user IDs that are allowed to launch a SAS server as that user ID. Note that elssrv must be writable before running patchname. Then the process must be changed back to setuid root.

## SAS EXECUTION

The `sas` command that is executed as "`!SASROOT/sas`" is really a file  that is linked to a shell script that defines the default  language for this execution and prepares the system for execution. If that language is English, the `sas` command will link to bin/sas_en. This script initializes the shell environment to run SAS in English and initialize important environment variables such as LD_LIBRARY_PATH and JAVA_HOME. The administrator is allowed to set his own set of shell environment variables and settings by editing the sasenv_local file in the !SASROOT/bin directory. The administrator might want to add something similar to the following code.

```
LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/lib64:/thirdparty/syncsort/lib"
export LD_LIBRARY_PATH
```

**Example sasenv_local**

SAS Technical Support asks that you run the sas executable directly, bypassing the sas_en script. If you do this, then you will need to include the sasenv script in your shell before executing SAS. There are three versions of the sasenv script, one for SH, one for KSH, and one for CSH. This allows the user to use the correct version in the shell of her choice.

### Specification of SASROOT and -CONFIG

The specification of the -SET SASROOT <directory>, the -CONFIG <file> option, or both changes the initialization path for SAS start-up. It is important that we be able to test all possible configurations. This means that we need to be able to start SAS using the SASROOT patched into the image at link time. In addition, this allows the config file to be the default configuration file that is specified through the SASROOT value. The configuration file processing is a very complex algorithm and it is crucial to the operation of the SAS BI infrastructure. In the past we have not tested this path very much at the track level. This testing needs to be added to the standard testing for SAS 9.3.

Here is a description of how configuration files are handled in SAS on a UNIX environment:

1. Obtain the default SASROOT from the patched location in the sas executable. This will be used as SASROOT if a -SET SASROOT is not specified. This is the location for restricted option configuration files. They are found in the default SASROOT location even if there is a -SET SASROOT. If there are no restricted options in the default SASROOT, then the one at -SET SASROOT is used.
2. Search the SASV9_OPTIONS environment variable for a -SET SASROOT or a –CONFIG option. Remember the -SET SASROOT location and whether there is a –CONFIG option set.
3. Search the argv array for a -SET SASROOT or a –CONFIG option. Remember the -SET SASROOT location and whether there is a –CONFIG option set. This overrides the value in SASV9_OPTIONS.
4. Search the argv array for a -CONFIG option. Parse the configuration files if they are found.
5. Search the SASV9_OPTIONS for a -CONFIG option. Parse the configuration file if it is found. The –CONFIG option on the command line is processed before the –CONFIG option on SASV9_OPTIONS.
6. If there were no configuration files specified, then check to see if a SASV9_CONFIG environment variable was specified. If so, parse the specified configuration file.
7. If no configuration files have been parsed so far, then look at the SASCFGPATH environment variable. This is a list of configuration files. These replace the one in !SASROOT/sasv9.cfg. Read and parse them.
8. Now look for the default configuration files:

   !SASROOT/sasv9.cf  (unless SASCFGPATH) is set)

   ~/.sasv9.cfg
   ~/sasv9.cfg

```
./sasv9.cfg
```

Look for restricted options, first in the original default SASROOT and then, if not found, in the -SET SASROOT location.

```
!SASROOT/misc/rstropts/rsasv9.cfg
!SASROOT/misc/rstropts/group/_rsasv9.cfg
!SASROOT/misc/rstropts/user/_rsasv9.cfg
```

As you can see, setting the different options and environment variables changes the way SAS parses configuration files. We need to be able to force all of these paths to be taken.

## CONCLUSION

SAS is a very sophisticated system that requires administrators to understand how a user authenticates within a server and which executables are installed and whether they are being accessed correctly. The utilities in the SASROOT/utilities/bin directory help the administrator to manage these tasks. SAS Technical Support understands how to use these tools and can help with any further understanding.

## REFERENCES

- SAS Institute Inc. 2014. *SAS® 9.4 Companion for UNIX Environments, Second Edition*. Cary, NC: SAS Institute Inc.

- SAS Institute Inc. 2014. “SAS® modules that must have the setuid bit set to root in the UNIX environment.” Available at http://www.support.sas.com.

## CONTACT INFORMATION

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

Jerry Pendergrass
Analytic Server Research and Development
SAS Campus Drive, S5124
Cary, NC 27513
Phone: 919-531-7766
Fax:     919-677-4444
jerry.pendergrass@sas.com

## APPENDIX  - SAMPLE ELSDEBUG OUTPUT

```
01242014 15:29:11  (13429) : ELSSRV
01242014 15:29:11  (13429) : Parent PID = 13428
01242014 15:29:11  (13429) : UID = 0  GID = 0
01242014 15:29:11  (13429) : send 7 recv 4 ioctl 8 log 10
01242014 15:29:11  (13429) : UID = 612  GID = 100
01242014 15:29:11  (13429) : Wait for Request
01242014 15:29:11  (13429) : RECV CMD  size(8)  command(8)
01242014 15:29:11  (13429) : Handle Request  8  [ELS_Logon_Command]
01242014 15:29:11  (13429) : LOGON

01242014 15:29:11  (13429) : Validate size 8 command 8
01242014 15:29:11  (13429) : Login Complete

01242014 15:29:11  (13429) : Wait for Request
01242014 15:29:11  (13429) : RECV CMD  size(0)  command(9)
01242014 15:29:11  (13429) : Handle Request  9  [ELS_WhoAreYou_Command]
01242014 15:29:11  (13429) : Wait for Request
01242014 15:29:11  (13429) : RECV CMD  size(0)  command(12)
01242014 15:29:11  (13429) : Handle Request 12  [ELS_Set_Port_Command]
01242014 15:29:11  (13429) : Set Child Termination Port 49690

01242014 15:29:11  (13429) : SEND OK rc = 0
01242014 15:29:11  (13429) : Wait for Request
01242014 15:29:11  (13429) : SEND OK rc = 0
01242014 15:29:11  (13429) : Wait for Request
01242014 15:29:11  (13429) : RECV CMD  size(40)  command(14)
01242014 15:29:11  (13429) : Handle Request 14  [ELS_SetEnv_Command]
01242014 15:29:11  (13429) : SETENV SASROOT=/sasgen/dev/mva-v9cas/SAS/laxnd

01242014 15:29:11  (13429) : SEND OK rc = 0
01242014 15:29:25  (13429) : RECV CMD  size(135)  command(6)
01242014 15:29:25  (13429) : Handle Request  6  [ELS_Exec_Command]
01242014 15:29:25  (13429) : Launch Arguments argc = 3  argv (nil) st 0x20

01242014 15:29:25  (13429)   : argc = 3

01242014 15:29:25  (13429)   : argv[0]  = /bin/csh
```

```
01242014 15:29:25  (13429)   : argv[1]  = -c
01242014 15:29:25  (13429)   : argv[2]  = ls

01242014 15:29:25  (13429)   : Number ports  = 0


01242014 15:29:25  (13429)   : number_environ_variables = 0

01242014 15:29:25  (13429) : SEND OK rc = 0
01242014 15:29:25  (13429) : EXEC Command Complete
01242014 15:29:25  (13429) : Wait for Request
01242014 15:29:25  (13429) : RECV CMD  size(0)  command(7)
01242014 15:29:25  (13429) : Handle Request  7  [ELS_Fork_Command]
01242014 15:29:25  (13429) : FORK Command

01242014 15:29:25  (13429) : Read Exec Pipe
01242014 15:29:25  (13464) : EXEC  /bin/csh

01242014 15:29:25  (13429) : Read Exec Pipe Complete 0
01242014 15:29:25  (13429) : FORK Complete [13464]
01242014 15:29:25  (13429) : FORK Return
01242014 15:29:25  (13429) : Wait for Request
01242014 15:29:25  (13429) : RECV CMD  size(0)  command(18)
01242014 15:29:25  (13429) : Handle Request 18  [ELS_CleanupFork_Command]
01242014 15:29:25  (13429) : FORK Cleanup
1242014 15:29:25  (13429) : SEND OK rc = 0
01242014 15:29:25  (13429) : Wait for Request
01242014 15:29:25  (13429) : Child Signal Handler.
01242014 15:29:25  (13429) : Child Death : Send pid(13464) Status(0) StatusM(0) Port(49690)
01242014 15:29:25  (13429) : Child Death: Connected fd = 3
01242014 15:29:26  (13429) : RECV CMD  size(0)  command(3)
01242014 15:29:26  (13429) : Handle Request  3  [ELS_Exit_Command]
01242014 15:29:26  (13429) : Exit
```