Paper #216-30

Reporting SAS®9 License Usage

David Bosak, Comsys IT Partners, Kalamazoo, MI

ABSTRACT

The purpose of this paper is to describe a way of quantifying and reporting SAS product usage precisely. The paper will outline a method of capturing SAS session usage using the RTRACE configuration option. The data from RTRACE will be mapped to SAS products, and reports will be generated from SAS IT Resource Management (ITRM). The result will be a system for monitoring and analyzing SAS license usage from ITRM.

INTRODUCTION

How many times a month does your company use SAS Stat? What about SAS Graph? You may license SAS/Access to Oracle, but does anyone actually use it?

These questions and others like them are hard for most companies to answer. Every year we go through the ritual of renewing our SAS licenses. In the weeks leading up, there is always a period of scrambling around, trying to figure out which licenses to add, and which licenses to drop.

How one quantifies license usage depends largely on the size of the organization, and how many SAS users there are. In a small organization, you can just call everyone up and ask. In a large organization with hundreds or thousands of SAS users, license usage is an educated guess at best.

The purpose of this paper is to describe a way of quantifying and reporting SAS product usage precisely. The paper will outline a method of capturing SAS session usage using the RTRACE configuration option. The data from RTRACE will be mapped to SAS products, and reports will be generated from SAS IT Resource Management (ITRM). The result will be a system for monitoring and analyzing SAS license usage from ITRM.

The system will have the desirable characteristics of being automatic, invisible, useful, and relatively easy to implement. While the system could be developed on any platform, this paper will focus on the UNIX operating system. The paper will give an overall description of the architecture, and describe key technical points in detail. Further information will be provided by request to the author.

RTRACE BASICS

For SAS licensing purposes, it is important to know what SAS modules are used, by whom, and on what machine. This information can be gathered by turning on the –RTRACE option in SAS.

The RTRACE option was originally designed to help debug Base SAS. For each module Base SAS loads into memory, the -RTRACE option will write out the full path to the program log. These paths can be redirected to another file using -RTRACELOC. The command line syntax for both options is as follows:

sas –RTRACE ALL –RTRACELOC [filename]

When these two options are used, SAS will generate one RTRACE log for each SAS session, and the log will contain all the files referenced during that session. Here is a portion of an RTRACE log from the Windows environment:

```
File referenced: C:\Program Files\SAS Institute\SAS\V8\SASV8.CFG
File opened: C:\Program Files\SAS Institute\SAS\V8\SASV8.CFG
File closed: C:\Program Files\SAS Institute\SAS\V8\SASV8.CFG
File referenced: C:\Program Files\SAS Institute\SAS\V8\autoexec.sas
File referenced: C:\Program Files\SAS Institute\SAS\V8\core\sasmsg\msgdir.msg
File opened: C:\Program Files\SAS Institute\SAS\V8\core\sasmsg\msgdir.msg
File closed: C:\Program Files\SAS Institute\SAS\V8\core\sasmsg\msgdir.msg
File referenced: C:\Program Files\SAS Institute\SAS\V8\core\sasmsg\host.msg
File opened: C:\Program Files\SAS Institute\SAS\V8\core\sasmsg\host.msg
File referenced: C:\Program Files\SAS Institute\SAS\V8\core\sasmsg\core.msg
File opened: C:\Program Files\SAS Institute\SAS\V8\core\sasmsg\core.msg
File opened: C:\Program Files\SAS Institute\SAS\V8\sas.exe
```

You can imagine how useful this information would be if you were a developer on the Base SAS team. The SAS system is composed of hundreds of libraries. These libraries need to be loaded and executed in the proper order.

Moreover, it is important to know that each library is unloaded when it is no longer needed. Failure to unload unnecessary libraries will result in memory problems down the road. It is very convenient therefore to have a running list of libraries and files SAS loads and unloads from memory.

You can also imagine how this running list of file references can be used in another way: to determine what products SAS is using.

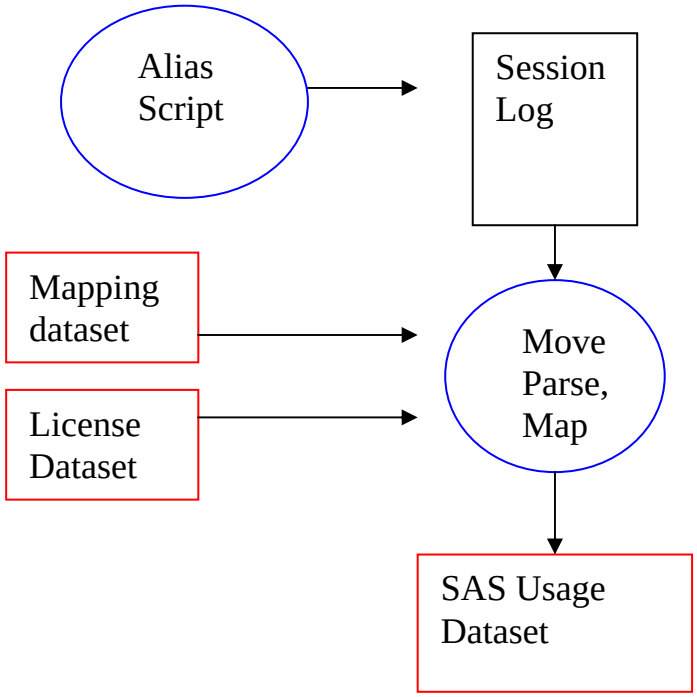
When you run a SAS product (like Graph), certain libraries are loaded into memory. Some of these libraries are unique to the SAS product you are executing, and not used by any other product.

For instance, let's suppose a library named *sasgplot* shows up in your RTRACE log. This is a library that is used only by SAS Graph procedures. The fact that *sasgplot* appeared in your RTRACE log means that you used SAS Graph at least once in your SAS Session.

The conclusion that you used SAS Graph at least once is valid. It is not valid to conclude that you used SAS Graph only once. The reason this conclusion is invalid is because SAS will load a module into memory once, but may use it several times during the course of a session. It cannot be determined from the RTRACE log how many times within a session you used a particular product. It can only be determined that the product was used at least once during that session.

It is important to keep this logical limitation in mind. You will not be able to count every time a license is used. What you will count is the number of sessions that used a particular license. This session count is adequate for most purposes.

SYSTEM OVERVIEW



The SAS usage tracking system architecture has 6 key components:

- **Alias script:** A script that executes when the user types "sas" at the command line. The user thinks they are starting SAS, but they are actually starting the alias script. The script will start SAS after appending the appropriate RTRACE options.
- **Session Log:** The session log is the log created by the RTRACE options. Each SAS session will have its own log. At the end of the day, many logs will be created by all the different interactive and batch SAS programs from all users in your company.

- **Log moving program:** Each day, all the session logs will have to be moved to a common location for processing. The log moving program will perform that function. It may have to move logs from several servers, depending on how many installations of SAS you have at your organization.
- **Log parsing program:** This program is where all the processing of the data happens. The log parsing program will transform the file names recorded in the session logs into useful information about SAS product usage. The result is the SAS usage dataset.
- **Mapping dataset:** The mapping dataset is a lookup table. It matches each SAS library to a SAS product.
- **License dataset:** The license dataset is merely a table of license information for your company. It resembles the information from a proc setinit. This table can help you by providing full product names, license type, expiration dates, and other information associated with licenses
- **SAS Usage dataset:** The SAS Usage dataset is the end result of the system. This dataset will contain information about each SAS session executed at your company, and what products were used in that session.

The alias script will kick off SAS with the RTRACE options turned on. The RTRACE options will generate a log for each session. The log will contain the names of all files referenced during that SAS session. It is convenient to name the log in the following manner:

[machine]_[user]_[datetime]_[processid].rtd

This log name will be created by the alias script, will fully identify the session, and will ensure that the log name is unique. It is impossible to have the same ProcessID generated on the same day at the same time by the same user on the same machine. Having a unique name is important if you wish to store all the logs in a common directory. The unique name will guarantee that no two SAS sessions write to the same log.

Once the logs are generated, a program or script must be written to collect them up. If your company is using multiple SAS servers, you will need a program to transfer the information from one machine to another. On UNIX, there are several tools available: *ftp*, *sftp*, and *scp* are the most common.

After the logs are collected, another program or script will parse the log. Frequently, the size of the logs will be between 1 and 3 megabytes, and will contain hundreds of file references. The parsing program will reduce this down to a few records in a dataset. The mapping and license lookup datasets will help the program do this.

In the end, the only information you care about is what products were used for each session, by whom, when, and where. The end result – the SAS Usage dataset - will look something like this:

Machine	User	Datetime	PID	Product
Dilbert	Dbosak	8/3/2004 10:16:47	105	BASE
Dilbert	Dbosak	8/3/2004 10:16:47	105	GRAPH
Felix	Rsmith	8/3/2004 10:18:52	13	BASE
Dilbert	Jhanson	8/3/2004 10:21:28	203	BASE
Dilbert	Jhanson	8/3/2004 10:21:28	203	GRAPH
Dilbert	Jhanson	8/3/2004 10:21:28	203	STAT
Dilbert	Dbosak	8/3/2004 10:32:05	222	BASE

ALIAS SCRIPT: MAKING SAS OUTPUT THE DATA

To my knowledge, the first published account of using the RTRACE facility to track SAS license usage belongs to Michael Raithel. In his paper Measuring SAS Software Usage on Shared Servers with the RTRACE Facility (SUGI Proceedings 215-29), Michael turns on the RTRACE options in the configuration file of each user. This approach works well in a Windows environment. The UNIX environment provides an opportunity for a different approach: an alias script.

On UNIX, SAS is typically executed on the command line from a link in /usr/bin or /usr/local/bin. All users share this same link. One way to turn on the RTRACE options is to replace this link with a perl or shell script. The script will execute SAS, append the options, and perform any other tasks as necessary.

The alias script approach is highly recommended, because there are indeed quite a few tasks that must be performed. Let's looks at the steps in detail. Here is an example of what happens when a user launches SAS using an RTRACE alias script:

1) User types "sas" at the command line.
2) The operating system looks for "sas" in the directories specified in the PATH environment variable, and finds the alias script in /usr/bin.

3) The alias script executes.
4) Script gets the machine name.
5) Script gets the user login name.
6) Script gets the date and time.
7) Script gets the process ID of the currently executing process.
8) Script generates a name for the RTRACE session log.
9) Script checks for available space in the target directory.
10) Script strips out any user options that would interfere (such as a second RTRACELOC)
11) Script preserves any user options that don't interfere
12) Script launches SAS executable with appropriate user and RTRACE options.
13) Script handles any aberrant conditions in an appropriate manner

The alias script allows you to perform the above tasks with a flexible and powerful language like perl, rather than the more limited SAS config commands.

Further, by maintaining a normal link to SAS in the /usr/bin directory, the alias script gives you a simple way to bypass your tracking system if necessary. Bypassing the system is sometimes desirable for (a) testing, (b) for regularly scheduled jobs at close intervals, or (c) for very large jobs that access thousands of files. These large jobs can generate huge RTRACE files that may clog up your system. For these special cases, it is nice to have an easy way to run SAS without generating an RTRACE log.

A sample alias script is provided in **Appendix A** of this document.

CREATING A SAS USAGE DATASET

Once the logs are gathered and moved to a common location, they are ready to be parsed. The parsing program will perform the following tasks:

1) Get the text data from the log into a SAS dataset
2) Eliminate duplicates
3) Eliminate uninteresting file names
4) Add in the information from the log file name

Once these tasks have been accomplished, you will have a clean SAS dataset full of file names. The remaining task is to map file names to products.

As mentioned previously, there are some libraries that are unique to a product. These unique libraries can be used to identify which product was used. On Unix, finding a library that is used by one and only one product is important because all SAS libraries are placed into the same directory: sasexe.

On Windows, the situation is somewhat easier. The libraries for each product are stored in separate product folders. Therefore if you want to know what product a library belongs to, you just parse out the name of the folder where the library is located.

You can leverage the PC SAS library organization to help compile the lookup list for Unix. The library names are almost identical from PC to Unix, except for the .dll extension. This method is likely the easiest way to figure out which libraries go with which product. Outside the above approach, I have not found comprehensive documentation concerning this mapping - either from SAS or elsewhere.

When the mapping is complete, you should have a product associated with every SAS library. Here is a portion of a mapping table:

Library	**Product**
sasabld	BASE
sasacces	BASE
sasacecl	STAT
sasadmin	CONNECT
sasadxr	QC
sasadxz	QC
Sasaf	BASE
sasafbrg	BASE
sasafe	BASE

sasafs	BASE
sasaimdb	BASE
sasalora	ORCL
sasalsyb	SYBASE
sasamgis	GRAPH
sasammap	GRAPH
sasanom	QC

In addition to a library lookup table, it is helpful to have a list of SAS products you license. This can be easily accomplished with a proc setinit. The product dataset will help you generate reports, and can contain additional information about that license, such as the type of license, servers on which it is installed, etc.

It is now possible to create the SAS Usage dataset shown above. Simply inner join the list of library names from the RTRACE log to the Mapping table and eliminate duplicates. Products and libraries that don't match will be eliminated from the join. The result will be a dataset containing the machine name, user, date, time, PID, and name of the SAS products used for each SAS session in your enterprise.

REPORTING FROM ITRM

The SAS IT Resource Management (ITRM) product provides a robust, mature environment for performance measurement and reporting. It has the ability to store and analyze data from a number of popular data-capturing products. ITRM can also be used to report off data collected from RTRACE.

To get the RTRACE data into ITRM, it must undergo some transformations. ITRM requires a particular format, and is expecting certain columns. Here is a table of what the ITRM dataset should look like.

Column	Length	Type	Comments
Datetime	6	Date	SAS Date variable in Datetime21.2 format. This variable is required, and used for a variety of purposes.
Hour	3	Number	The hour of the day (1 through 24)
Shift	1	String	The shift variable is used to differentiate working hours from after-hours, and weekdays from weekends. A typical shift schedule will look like this: 1 = Normal working hours 2 = Weekday evenings 3 = Weekends
Duration	8	Date	The difference in time between the current observation and the previous observation.
Month	2	Number	An integer Month designation, extracted from the datetime
Monthnm	10	String	A text Month designation, extracted from the datetime
Weekday	10	String	A text day of the week designation, extracted from datetime
Week	3	Number	A numeric week designation of 1 through 52, extracted from datetime
Machine	32	String	The name of the machine on which the SAS session executed.
User	32	String	The User Login that executed SAS.
Product	10	String	The SAS product name.
PID	6	Number	This is the Process ID that identifies the SAS session. This number can be linked to other performance data to give memory and cpu usage for the session.
Job	1	Number	A numeric field seeded with the value 1 for all observations. This column is used for summary job counts in ITRM.

This transformed dataset can be generated with data step or with a proc sql statement. This dataset will allow you to

use the ITRM Manage Reports tool to create charts and graphs that can be incorporated into your ITRM performance reporting web site.

EXAMPLE REPORTS

Here are some examples of the types of reports you can create with the information provided in this paper.

Figure 1: SAS usage by Product by Server

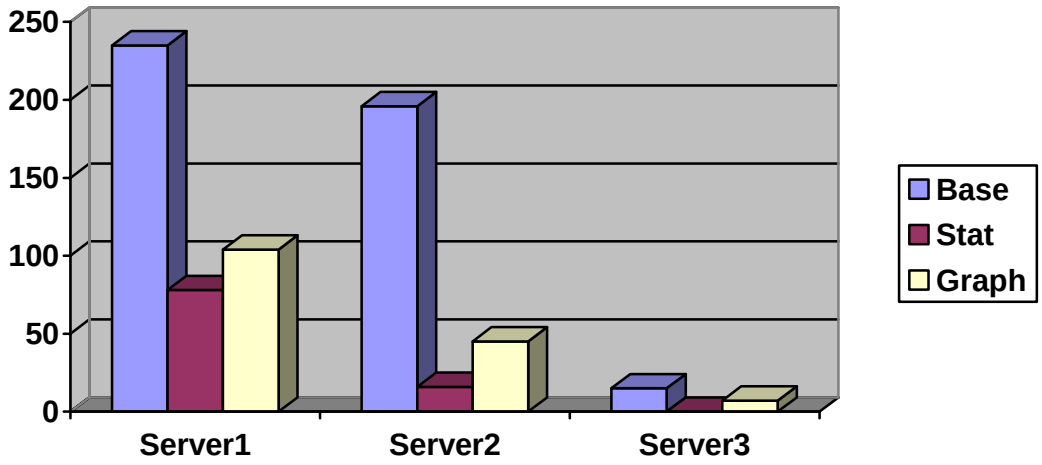


Figure 1 shows a basic bar chart of SAS product usage for three servers. Based on the information from a chart like this, you may determine that the SAS work that takes place on Server3, should be consolidated onto Server2. Consolidating the SAS work on Server2 would allow you to eliminate the Server3 licenses. Eliminating the Server3 licenses will not only save license costs, but will reduce the IT costs associated with maintaining and supporting the software.

Figure 2: SAS usage graph by Product by Month

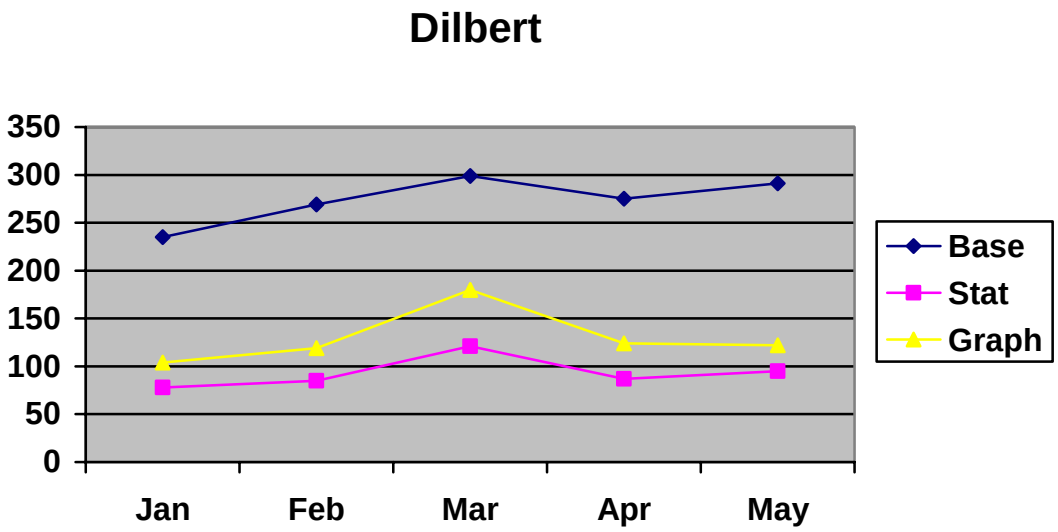


Figure 2 displays a trend line chart of SAS product usage. This chart will give you an indication of whether product usage is increasing or decreasing. This chart can contribute to your capacity planning analysis, and help you determine when it's time to upgrade your hardware, or add additional SAS licenses to a new server to help balance the load.

Figure 3: SAS usage by User and Product

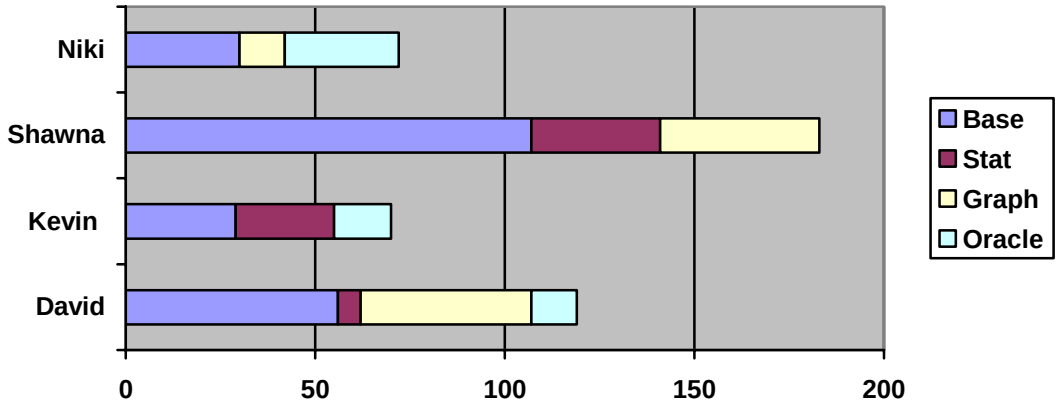


Figure 3 contains a stacked bar chart by user and product. This chart will allow you to identify your heaviest and lightest SAS users. For many companies, this information can by eye-opening and very useful. The data is most useful if you have a large user base, and do not oversee or associate with all your users. It can help you identify problem areas in your information architecture, and help determine if you need to change your licensing schemes. The data from this system could also be used as the basis for a charge-back system based on user or department.

CONCLUSION

The above paragraphs described the key architectural components of a system to track SAS license usage. For a complete system, there are other important considerations not discussed above. Some of these additional considerations are:

- **Security:** To make this system work, a number of security issues need to be addressed. First, all users will need access to the directory designated to hold the RTRACE logs. This will require universal write and execute permissions for that directory. Also, collecting and moving files between machines will require some forethought. You want to give access to the programs and logs as needed, but not open up the system to vandalism or inadvertent damage.
- **Disk Space:** If you have a large user base, the system may generate a significant amount of RTRACE logs. These logs should be collected and processed daily to reduce the amount of disk space used. Further, the amount of workspace required to process the logs must be calculated based on average usage. You must also build error handling into the system to anticipate out of space conditions for working directories.
- **Scheduling:** The ideal collection system will run automatically and invisibly. You can use scheduling tools like *cron* or Control M to schedule daily, weekly, and monthly tasks. The user account that the scheduling software runs under must have sufficient permissions to access the files and directories related to the project.
- **Error handling and Notifications:** A system for error handling and notifications is a necessary aspect of a robust architecture. Your architecture should not only anticipate possible points of failure, but also provide email or other notifications to the appropriate people under certain failure conditions.
- **Logs:** You will need a system for logging activity, and a system for managing the logs. The logs may get quite large depending on what you record. Clearly, a system is needed to archive or compress these logs when certain size limits are reached.
- **Linking to other performance information:** You may wish to link your product usage data to other performance data like memory usage or CPU usage. The process id (PID) can help you do that. Performance Monitoring software like HP Measureware, BMC Patrol, and sar will capture data on individual processes - identified by PID. In this way, it is possible to join your product data to the performance data and find out how much memory that session used, how much CPU, etc.

REFERENCES

Raithel, Michael. Measuring SAS Software Usage on Shared Servers with the RTRACE Facility (SUGI Proceedings 215-29), 2004

ACKNOWLEDGMENTS

Thanks to Brian Varney and Kevin Kramer from Comsys IT Partners for providing all the important pieces of this system.

CONTACT INFORMATION

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

David Bosak
Principal Consultant
Comsys IT Partners
5278 Lovers Lane
Kalamazoo, MI 49002

Email: dbosak@comsys.com
Phone: 269-344-4100 ext 536

APPENDIX A: SIMPLIFIED ALIAS SCRIPT

```
#!/usr/bin/perl
#*********************************************************************
# TITLE:      SAS Alias script
# MODULE ID:  sas
# VERSION:    1.0.0
# PURPOSE:    Call SAS with rtrace turned on
#             with the current process ID.
#
# PROGRAMMER: (Venturi Partners)
#
# INPUT ASSERTIONS: Any sas parameters passed in at invocation.
#
# OUTPUT ASSERTIONS: SAS is started with rtrace on and the ouput file
# from rtrace set, plus any input assertions.
#
# REMARKS:  This script has been purposely simplified for this paper.
#           Error handling and checking have been stripped out.
#           Do not use this code as is.
#
#*********************************************************************

# Set the path to the SAS executable #
$SASDir = "/opt/sas82/sas";

# Set the output directory for the trace files #
$OutputDir = "/prod/home/lts/rtrace";

# Get the current processID #
$PID = $$;

# Get the user and server name #
$Host = $ENV{HOSTNAME};
$User = $ENV{USER};

# Get the current date and time #
($Sec, $Min, $Hour, $mDay, $Month, $Year, $wDay, $yDay, $isDst) =
localtime(time);

# Get the year #
$Year += 1900;

# Get the month #
$Month = $Month + 1;
if ($Month < 10 ) {
  $Month = "0" . $Month;
}

# Get the day #
if ($mDay < 10) {
  $mDay = "0" . $mDay;
}

# Get the hour #
if ($Hour < 10) {
  $Hour = "0" . $Hour;
}

# Get the minute #
if ($Min < 10) {
  $Min = "0" . $Min;
```

```
}

# Get the second #
if ($Sec < 10) {
  $Sec = "0" . $Sec;
}

# Create the timestamp variable #
$TimeStamp = $Year . $Month . $mDay . $Hour . $Min . $Sec;

# Create a unique filename for the trace log #
$FileName = $OutputDir . "/" . $Host . "_" . $User . "_" . $TimeStamp . "_" .
$PID . ".rtd";


# Exec SAS in the current process #
exec("$SASDir $sParm -RTRACE ALL -RTRACELOC $FileName");
```