Paper 279-2010

More Ways to Use SAS® to Manage, Monitor, and Control SAS® or the SAS® BI Server: Tools for the SAS User, Server Administrator, or Manager

LeRoy Bessler, Bessler Consulting and Research, Fox Point, WI

Abstract

This paper adds two tools to last year's kit.

The CPUmon tool sends emails to the SAS server user, with CC to the administrator or any other designee, whenever the user's SAS process exceeds a CPU time threshold and sends a further email to the user whenever a set CPU increment is exceeded.

The LogTimer SAS user macro puts in the SAS log the elapsed time and CPU time consumed between invocation with parameter Start and End. Without it, SAS provides only step-level numbers and a datetime for start of a Display Manager or SAS Enterprise Guide® session. Code resubmission does not change that datetime. SAS 9.2 system option FULLSTIMER adds a step-end datetime in the log. The LogTimer macro does not require the option and works in all current SAS versions.

The 2009 predecessor to this paper, whose content is included here, met other needs.

The 2009 tools answered questions for the administrator or manager of the server. Who is using SAS now and how much CPU time and memory? Since when? What is the last time each user was on the server? How heavily does each use the server in terms of frequency or CPU time?

The 2009 tools helped the SAS user with information and empowerment. What processes do I myself have running? I have a possible looping or hung process on the remote server, but my SAS Enterprise Guide session is hung or cancelled. How can I kill the process?

All of the tools, 2009 and 2010, were developed for a Windows BI server using SAS software. They can also be used for SAS on a stand-alone PC.

Introduction

Intended Audience

The intended audience for the user monitoring and CPU-time monitoring tools is SAS BI Server administrators and possibly their managers. UserMon and CPUmon can not be implemented or operated by users of the BI Server. If their data is made read-accessible to users, they, like an administrator, can run queries against the UserMon data, including data pertaining to other users, unless you filter it first. (Though not the intended purpose for which these monitoring tools were invented, they could be adapted to monitor any non-SAS process(es) as well or instead.)

The LogTimer, ShowProcessID, DisplayAllMySASprocesses, and TerminateProcess macros are intended for any SAS server user or PC SAS user.

The BI Server unrestricted user, presuming that she/he has administrator rights on the server, can deal with process display and termination for anyone's process on a server with DOS commands, rather than have to open SAS and run some macros.

Why I Did This

I got into the SAS Enterprise Guide client / remote SAS server architecture with SAS Enterprise Guide 2.5 and SAS 8.2 in 2003. When I first read about the SAS 9.1.3 BI Server, I was excited by the prospect of a "SAS Management Console".

I expected a tool that would enable me, as server administrator, to see who was using the server and their resource burden, and to find and, if needed, kill hung, looping, or abandoned processes. The real function of the Version 9.1.3 SAS Management Console is administration of a metadata repository to define BI server resources and control access to them. The SAS Management Console does not enable control or monitoring of the BI server operations.

So, I developed my own server monitoring and control tools. Of course, there are various vendor-provided monitors for Windows servers. You might have something that you like better than what I can show you. If you have personal control over the configuration of such a tool, over what logging it does, and over how the log data, if any, is managed and made available to you, perhaps you are satisfied. If not, then this paper can help you.

The UserMon monitor captured enough for my needs, and the configuration, logging, and data management were customizable to my personal preference, and were changeable by me at will.

Because it is not a vendor-provided, more limited tool, my CPUmon monitor has the ability to translate the user ID associated with any process that is over the limit into the user's email address and name. This information (and even the phone number) can be stored in the BI Server metadata database, or anywhere else that you prefer. In actual production use, my CPUmon did a lookup in a support table that was an extract from the metadata. Though I wanted to be notified about over-the-limit processes, I also wanted the process-owning user to be notified directly. I could not be continuously paying attention, but the user should be, and, in any case, is the responsible party. Of course, after every update by me to the BI server's user metadata, I needed to refresh the extract lookup table.

Scope and Structure of the Paper

The server operating system is Windows. I don't know whether UNIX analogues to all of these tools can be developed, but I can report that another user developed a UNIX tool to serve the function of my TerminateProcess macro. My tools specifically rely on Windows commands. The only SAS product needed is Base SAS. Macro language is used. ODS can be used to format reports of monitor data. Macro language and ODS are in Base SAS.

Though I present results of an example query against the UserMon data here, answering all of the questions listed in the Abstract is left as a programming exercise for the reader who is an interested user.

UserMon captures user ID (misleadingly called UserName by the Windows). More useful reports include the actual name of the user, which can, e.g., be obtained from an export of the user information sector of the BI Server metadata database or from any other lookup table maintained to provide the same information in a non-BI-server environment.

A Tool That Everyone Can Use: The LogTimer Macro

Q: When did my SAS program run, and what were its elapsed time and total CPU time?

A: Though the SAS log displays the real time (elapsed time) and the CPU time for each DATA step and each PROC step, it does not answer the questions of when your program ran, and, unless it was run in batch, what the elapsed time and CPU time were for the entire block of code submitted, which is usually multi-step.

The LogTimer macro is an easy-to-use tool that answers those questions. Without it, the only timing content in the SAS log, from SAS Display Manager or SAS Enterprise Guide, is of this form:

```
NOTE: DATA statement used (Total process time):
real time 0.41 seconds
CPU time 0.02 seconds
```

If your program runs in batch, step-level information like that above will be provided, but also the SAS log will end with a NOTE of this form:

```
NOTE: The SAS System used:
real time 16.13 seconds
CPU time 2.06 seconds
```

In SAS 9.2, if you turn on the system option FULLSTIMER, your SAS log will display the date/time at the end of each step. The LogTimer macro works in all versions of SAS, and does not require FULLSTIMER to be turned on. Before the SAS log content for the execution of the main body of your code, the LogTimer macro inserts something like this:

```
******************************************
Started at 4:42:02 on 27OCT2009
******************************************
```

and after the SAS log content for the execution of the main body of your code, the LogTimer macro inserts something like this:

```
******************************************
Ended at 4:42:04 on 27OCT2009
Elapsed Time (hours:minutes:seconds) = 0:00:02
CPU Time (hours:minutes:seconds) = 0:00:01
******************************************
```

Here is how to use the LogTimer macro:

```
%LogTimer(Start);
<your code here>
%LogTimer(End);
```

Invocation of the macro with “End” before “Start” or invocation of the macro with any other value will result in an appropriate error message.

The “End” invocation of the LogTimer macro can be used, if desired, at multiple points within a code submission, in which case it, of course, displays the cumulative Elapsed Time and CPU Time as processing progresses. Discrete Elapsed Time and CPU Time for each constituent DATA step or PROC step are provided by SAS itself in the usual manner.

NOTE: See Appendix 1 for the code for this macro. As stated in the macro’s internal comments, some parts of the code which captures CPU time are dependent on the specific version of Windows on which the macro is running.

Tools for What the SAS Enterprise Guide Facilities Do Not Help With

Purpose

The **TerminateProcess** macro is intended for when your process is in one of these situations:

Case 1. Your SAS Enterprise Guide session is no longer available, but the process is still running. This can happen if your PC/laptop has been rebooted or shut down without closing the SAS Enterprise Guide session, or if you used Windows Task Manager to kill SAS Enterprise Guide in desperation because it was frozen.

Case 2. Your SAS Enterprise Guide session is open, but the red square Stop button and the Task Status Window do not terminate your running process, and you can not wait for normal termination (or do not think normal termination will ever occur).

SAS provides no way to list all of your concurrent processes on the server.

You can use the **DisplayAllMySASprocesses** macro at any time.

E.g., you might want to know how much CPU time your puzzlingly long-running process has used up so far.

Is it using an excessive amount? I.e., is it possibly in a loop? Is it getting NO CPU time? I.e., is someone else's process consuming all of the resources, or is your process waiting for something to happen?

You can not display other people's processes. But a query to the UserMon data base can do that for you.

How To Use Them

These tools are intended to be used in the SAS Enterprise Guide Code Window. However, the ShowProcessID macro can also be used at the beginning of a batch job whose log you are monitoring during its execution.

1. Using the **ShowProcessID** Macro

In the Code Window, submit this statement:

%ShowProcessID;

Look in the Log Window for feedback. You will get a message of this form:

Process ID for this SAS Enterprise Guide session or SAS batch job is 6212

2. Using the **DisplayAllMySASprocesses** Macro

In the Code Window, submit this statement:

%DisplayAllMySASprocesses;

In the Log Window, look for a listing of your processes. Until this tool is enhanced, the listing will be inelegant, but useful. You can ignore any information of no interest in this context. Key information is the list of all Process IDs for your active SAS processes. (Other information includes the CPU time and Memory used by each process. Scroll to the right in the Log window to see it. CPU time is at the far right.)

3. Using the **TerminateProcess** Macro

(This must be used in an SAS Enterprise Guide session other than the one of the process that you want to terminate.)

In the Code Window, submit this statement:

%TerminateProcess(ProcessID=NNNNNN);

where **NNNNNN** is the Process ID (a.k.a., “PID”) for the process ID that you want to terminate. **NNNNNN** is typically a three- to six-digit number.

Look in the Log Window for feedback. If your request succeeded, you will get this message:

SUCCESS: Sent termination signal to the process with PID NNNNNN.

Note that it states that a termination signal was **Sent**.

To verify termination, resubmit **%DisplayAllMySASprocesses;**

If you inadvertently try to kill a process that is not yours, the action will fail, and you will get this message (where **ABCDEFGH** will be your user ID):

Process ID NNNNNN is not for User ID ABCDEFGH and will not be killed.

If you inadvertently try to kill a non-existent process, the action will fail, and you will get this message:

Process ID NNNNNN was not found.

NOTE: See Appendix 1 for the code for these macros.

Using SAS to Monitor SAS Users and Processes

The heart of these tools is a DOS tasklist command

```
tasklist /v /fi "imagename eq sas.exe"
```

issued from a SAS program running on the server with a user ID that has the credentials of a Windows System Administrator. The BI Server unrestricted user might have such credentials. /v stands for “verbose”.

This command could be used to find users and processes for any executable, not just SAS, by eliminating the filter:

```
/fi "imagename eq sas.exe"
```

The mechanism for data capture is a program that wakes up every six minutes (you can adjust this to any interval you prefer) to issue the command above with its output directed to a named pipe, followed by a DATA step that converts the output to a datetimestamped SAS data set that is written to permanent disk. The logic that parses the data returned in response to the tasklist command is Windows-version-dependent.

Each monitor runs as a Windows Scheduled Task, using a bat file of this form:

```
"C:\Program Files\SAS\SAS 9.1\sas.exe"  -sysin "PathToPgm\UserMon.sas"  -log
"PathToLog\UserMonSASlog.txt"  -print "PathToLst\UserMonSASlst.txt"  -work "PathToWork"
```

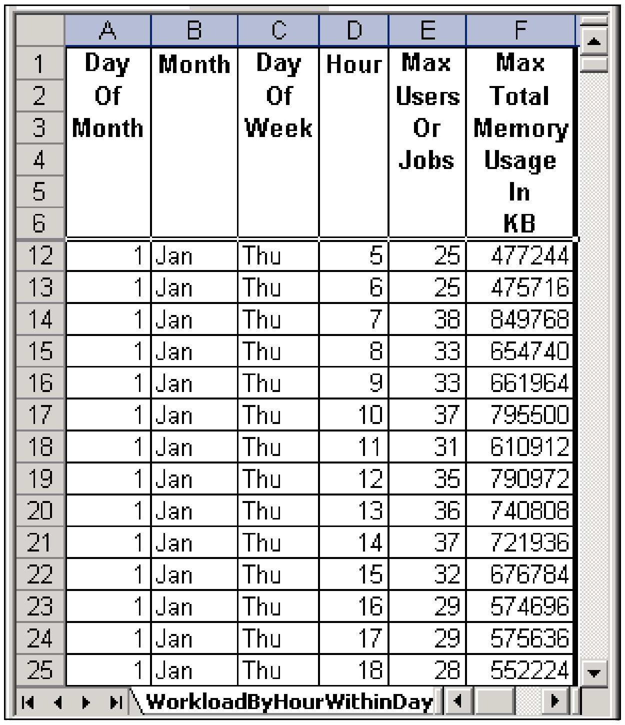
where PathToPgm, PathToLog, PathToLst, and PathToWork could all be assigned instead as the same Windows folder. Also, each monitor could run as a Windows service that starts automatically at reboot.

The monitors are not a big resource burden. Their CPU time and disk space needs are not significant.

You can run either monitor without running the other.

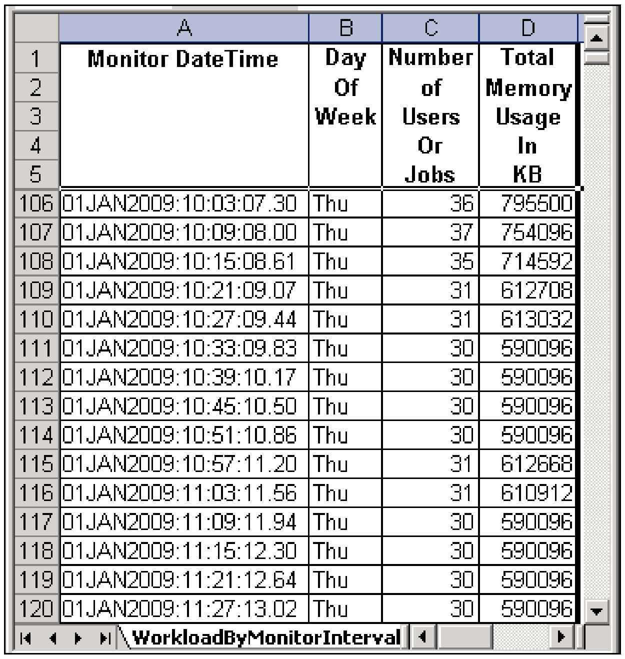
UserMon

Below are excerpts from two of many reports that can be created from the UserMon data.



	A	B	C	D	E	F
1-6	Day Of Month	Month	Day Of Week	Hour	Max Users Or Jobs	Max Total Memory Usage In KB
12	1	Jan	Thu	5	25	477244
13	1	Jan	Thu	6	25	475716
14	1	Jan	Thu	7	38	849768
15	1	Jan	Thu	8	33	654740
16	1	Jan	Thu	9	33	661964
17	1	Jan	Thu	10	37	795500
18	1	Jan	Thu	11	31	610912
19	1	Jan	Thu	12	35	790972
20	1	Jan	Thu	13	36	740808
21	1	Jan	Thu	14	37	721936
22	1	Jan	Thu	15	32	676784
23	1	Jan	Thu	16	29	574696
24	1	Jan	Thu	17	29	575636
25	1	Jan	Thu	18	28	552224

WorkloadByHourWithinDay



	A	B	C	D
1-5	Monitor DateTime	Day Of Week	Number of Users Or Jobs	Total Memory Usage In KB
106	01JAN2009:10:03:07.30	Thu	36	795500
107	01JAN2009:10:09:08.00	Thu	37	754096
108	01JAN2009:10:15:08.61	Thu	35	714592
109	01JAN2009:10:21:09.07	Thu	31	612708
110	01JAN2009:10:27:09.44	Thu	31	613032
111	01JAN2009:10:33:09.83	Thu	30	590096
112	01JAN2009:10:39:10.17	Thu	30	590096
113	01JAN2009:10:45:10.50	Thu	30	590096
114	01JAN2009:10:51:10.86	Thu	30	590096
115	01JAN2009:10:57:11.20	Thu	31	612668
116	01JAN2009:11:03:11.56	Thu	31	610912
117	01JAN2009:11:09:11.94	Thu	30	590096
118	01JAN2009:11:15:12.30	Thu	30	590096
119	01JAN2009:11:21:12.64	Thu	30	590096
120	01JAN2009:11:27:13.02	Thu	30	590096

WorkloadByMonitorInterval

For each monitor interval, UserMon creates a datetime-suffixed SAS data set of records, one record for each Process ID, along with the User ID, Cumulative CPU seconds, Current Memory Usage, and Monitor DateTime for that Process ID. To work with the monitor data, you need to concatenate all of the monitor-datetime-specific SAS data sets into one cumulative data set, optionally subject to start and end day filtering. If you do not cut off data concatenation at a day earlier than the concatenation run day, it is important to avoid the latest data set since the SAS user monitor might be awake and actively writing to it.

NOTE: See Appendix 1 for code used for the monitor, to concatenate monitor logs, and to create these reports.

CPUmon

At each monitor interval, CPUmon looks for any SAS processes that have consumed more than two CPU hours (by default), or whatever has been set as the cumulative CPU time threshold. The UserID, Process ID (a.k.a. “PID”), and CPU time are written to a reference table for any alert. Whenever such a process is detected, the table is checked for a pre-existing record. If no record is found, the alert is sent, and an alert record is created in the table. If there is a pre-existing record, and the current cumulative CPU time is over the previously alerted CPU time by a defined increment (one hour by default), another alert is sent, and the new CPU time is recorded in that row of the table. The increment trigger for additional alerts, like the threshold trigger for the initial alert, is a macro invocation parameter. Any PIDs that are no longer active at the monitor instant are deleted from the table. Alerts are sent directly to the user, with CC to an administrator email address. If no email address is available for the user, both TO and CC use the administrator email address. The administrator email address is a macro invocation parameter, but, of course, with a code change to the macro it could be accessed from a lookup file instead.

As a point of programming information, the reference table actually is not updated in place. It is rebuilt during every monitor interval, retaining only fresh alerts, old alerts still running, or old alerts still running for which a renewed alert has been sent.

The macro supplied in this paper does include optional code that can be turned on to permit inspection of the response from the tasklist command. You need to customize it to write the diagnostic information to a location where the monitor will, in fact, have WRITE authorization.

The layout of the response can vary by version of Windows, and the code needs to be revised to suit the environment. Also, structure and length of user ID will vary from site to site. And each site will have its own way of wanting to support the translation of user ID into email address.

Finally, the macro includes the capability to send a quite verbose email message. It is designed to be sent to users of a server. (It *could* be used on a standalone PC.) By modifying the macro, the message can be customized to fit the actual characteristics of the installed site and to suit the preferences of the site SAS Administrator. The approach taken in the code for the email message here is that a maximally concise message, especially the first time received, could be confusing to a user who is not aware of the implications of using a lot of CPU time. Also, in some cases, using a lot of CPU time is expected and harmless.

NOTE: See Appendix 1 for the CPUmon macro and an example of its invocation. If adapting it to your site, you need to verify the parsing of the TaskList response, customize the email message, customize the email address lookup, possibly remove the diagnostic code, and probably simplify or remove the ReportTimeUnits handling (which was designed to support a variety of testing scenarios).

Below is an example of the email alert sent to a fictitious SAS user. The recipient email address, User ID, Process ID, name of the user, server name, date, and time are bogus.

```
From: LeRoy Bessler [mailto:Le_Roy_Bessler@wi.rr.com]
Sent: Sunday, February 28, 2010 11:59 PM
To: NoSuchUser@NoneSuch.com
Cc: Le_Roy_Bessler@wi.rr.com
Subject: User ID CPUtestUserID Process 314159 Has Used 2.01 Hours of CPU
Time. Is it OK?
```

```
Dear FirstName LastName

Please read this long automated message carefully and in entirety.

As Of 28FEB2010:23:59:03.45
User ID CPUtestUserID Process 314159 Has Used 2.01 Hours of CPU Time.
Is it OK? If not, please terminate it.

For information on how to terminate a SAS process, go to
the URL for that part of the SAS Server Site User Documentation.
If you can not terminate it, call the SAS Server Administrator.
If the SAS Server Administrator is unavailable, call the Help Desk.
They need the Process ID 314159 and Server Name IamYourSASservant.

You will get a message every time current process 314159 uses another 1.00
Hours of CPU time.
The process ID 314159 was assigned when your SAS Enterprise Guide or Display
Manager session or batch job started.
It varies from session to session and from job to job.

CPU time is NOT elapsed time. It is active processing time.
The CPU waits idle for file reads, file writes, responses from Oracle, etc.
CPU time is usually less than elapsed time.

Using a lot of CPU time is not necessarily bad.
Heavy-duty analytics processes can use a lot of CPU time.
However, if your process is in a loop, it is wasting resources.
If your process is from an abandoned or frozen Enterprise Guide session,
you will be unaware of how much CPU time it is using
unless you investigate with the process management tools,
or it has used at least 2 Hours of CPU time and you get this message.
If you have any questions, call or send email to the SAS Server
Administrator.
```

Conclusion

SAS can be used to put more run-related information in its SAS log and to monitor and control the SAS BI server. To support the server, and to further empower server users, I have developed other tools, not all of which could be presented here. One of the tools not presented is EGbatch.

EGbatch allows users to submit SAS code from Enterprise Guide, but gives them the ability to monitor the SAS log on disk while their code is running. Via email, it sends the user a job start, a job end, and a job completion status message. The start and end messages include Process ID for the job. All three messages identify the SAS code file run by the job. In cases where the log size is not a concern for email, there is the option to have it attached to the job completion status email whenever there is an abnormal termination. Any report that the code might create can be emailed to a list of recipients. The ODS packaging options of normal SAS Enterprise Guide submissions are supported by EGbatch with no extra coding required in the submitted code itself. If interested, send me an email.

I always enjoy finding ways to make SAS software do something that it does not do on its own accord. Being a long-time SAS enthusiast, I like to say, “If you can’t do it with SAS software, maybe you don’t really need to do it.”

References

1. Bessler, LeRoy. “Using SAS to Manage, Monitor, and Control the SAS BI Server: User-Developed Custom Tools for the SAS Server Administrator, User, or Manager”, *Proceedings of the SAS Global Forum 2009 Conference*. Find it on the web at http://support.sas.com/resources/papers/proceedings09/274-2009.pdf.

Contact Information, Etc.

Your comments, questions, and suggestions are welcome. All code presented here is available in a zip file upon email request to the author. If you have any better ideas, or alternative ideas, as long as they do not require extra software and do not require non-SAS coding (other than DOS commands), I would be interested to see them.

LeRoy Bessler PhD
Le_Roy_Bessler@wi.rr.com

Strong Smart Systems™
Visual Communication Power™

Zum sehen geboren, zum schauen bestellt.
(Born to see, meant to look.)
—Goethe

A SAS user since 1978, Dr. LeRoy Bessler has shared his knowledge and experience with other users at conferences throughout the USA and in Montreal, London, Heidelberg, and Dublin. Though a SAS generalist with long experience in Base SAS, SAS macro language, and SAS tools for access to non-SAS data, his special interests include creation of unique tools to support the SAS BI server and its users, communication-effective visual communication and reporting, web information delivery, highly formatted Excel reporting, SAS/GRAPH, ODS, and Software-Intelligent Application Development for Reliability, Reusability, Extendibility, and Maintainability. He is a regular contributor to *VIEWS News*, the web newsletter of the VIEWS International SAS Programmer Community.

Appendix 1.

CPUmon Code To Detect Excessive CPU Usage and Sent Alert Emails

```
options nosource nonotes nomprint nomprintnest nosymbolgen nomlogic;
* Turn on one or more above options only for debugging. *;
* Since monitoring can run for days, weeks, or months,
  depending on how often the server is rebooted,
  the SAS log can get very large. *;
* options source;
* options notes;
* options mprint mprintnest;
* options symbolgen;
* options mlogic;
options linesize=max pagesize=max;

*****************************************************************************;
* Author: LeRoy Bessler PhD                                                 *;
* Date:   28 February 2010                                                  *;
* Purpose: Monitor CPU usage of server (or standalone PC) by SAS processes. *;
* Functions and Options:                                                    *;
* Monitoring uses ImageName, UserName (User ID), and Process ID (PID)       *;
* that are delivered in the response to a DOS TaskList command.             *;
* It uses filter "imagename EQ sas.exe".                                    *;
* (It could be adapted to monitor any other imagename instead, or as well.) *;
* Monitor events are separated by time interval WaitSeconds.                *;
* Optionally it limits the NumberOfMonitorEvents. Macro default is NoLimit. *;
* It sends an alert if it finds a SAS process that has consumed more than   *;
* the CPUthresholdInSeconds.                                                *;
* Each SAS process is identifiable by its PID, or Process ID.               *;
* The UserID, PID, & CPU time are written to a reference table for alerts.  *;
* Whenever such a process is detected,                                      *;
* the reference table is checked for a pre-existing record.                 *;
* If no record found, the alert is sent, and a reference record is created. *;
* If there is a pre-existing reference record, and the current CPU time     *;
* exceeds the previously logged cumulative CPU time by at least             *;
* the CPUincrementForAnotherAlert, then another alert is sent,              *;
* and the new CPU time is recorded in that row.                             *;
* (The CPUincrementForAnotherAlert must be assigned in seconds.)            *;
* Any PIDs that are no longer active are deleted from the reference table.  *;
* Alerts are sent directly to the user, with CC to Sender.                  *;
* If no email address is available for the user,                            *;
* the alert is sent to the DefaultEmailAddress.                             *;
* Optionally, every alert can also be sent to the AlwaysNotify address.     *;
* Email addresses for users are assumed to be in a SAS data set, where      *;
* User ID is up to 17 characters and Email Address is up to 80 characters.  *;
* The LookUp table could be different.                                      *;
* The SAS format built here at monitor start-up could have been pre-built,  *;
* and OPTIONS FMTSEARCH could be used to point to it.                       *;
*****************************************************************************;

%macro CPUmon_macro_ForSGFdemo(WaitSeconds=360,
  NumberOfMonitorEvents=NoLimit,
  MonLib=,
  ReportTimeUnits=Hours, /* these units are used for CPU times in the emails */
  CPUthresholdInSeconds=7200,
  CPUincrementForAnotherAlert=3600,
  PathToEmailAddressDataLib=C:\LeRBmonitors\EmailAddressLib,
  EmailAddressData=CPUusers,
  DefaultEmailAddress=Le_Roy_Bessler@wi.rr.com,
  DefaultExplain28Characters=UserID without Email Address,
  Sender=Le_Roy_Bessler@wi.rr.com,
  AlwaysNotify=);

libname EaddrLib "&PathToEmailAddressDataLib";

data ToFormat;
length Start $ 17 Label $ 80 HLO $ 1;
retain fmtname 'Eaddr' type 'C';
set EaddrLib.&EmailAddressData(rename=(UserID=Start)) end=LastOne;
Label = EmailAddress;
substr(Label,41,40) = FormattedUserName;
output;
if LastOne then do;
  HLO = 'O';
  Start = ' ';
  Label = "&DefaultEmailAddress";
  substr(Label,41,40) = "&DefaultExplain28Characters";
  output;
end;
run;

libname EaddrLib clear;

proc format lib=work CntlIn=ToFormat;
run;

%let MonitorCycleCount = 0;

libname MonLib "&MonLib";

%StartOfCycle:

data _null_;
DateTimeOfMon = datetime();
DateOfMon = datepart(DateTimeOfMon);
TimeOfMon = timepart(DateTimeOfMon);
call symput('MonDateTime','D'|| trim(left(put(DateOfMon,yymmddn8.))) || '_' ||
     'T'|| trim(left(put(input(compress(put(TimeOfMon,time8.),':'),6.),Z6.))));
run;
```

```
data _null_;
length TaskList $ 256;
TaskList = "'tasklist /v /fi " || '"imagename EQ sas.exe"' || "'";
call symput('TaskListCommand',trim(left(TaskList)));
run;

filename tasklist pipe &TaskListCommand;

data
MonLib.ActiveProcesses
(keep=UserID Process_ID     Cumulative_CPU_seconds     MonDT OrigID)
%if &MonitorCycleCount eq 0
%then %do;
MonLib.CPUmonRefTable
(keep=UserID Process_ID Ref_Cumulative_CPU_seconds Ref_MonDT)
%end;
                      ;
length OrigID UserID $ 17 Process_ID $ 6 Cumulative_CPU_seconds 8 MonDT $ 21;
infile tasklist lrecl=229 pad;
input @1 line $char229.;
if _N_ GE 4;
OrigID = substr(substr(line,89,50),17,17);
UserID = upcase(OrigID);
Process_ID = left(substr(line,27,6));
Cumulative_CPU_seconds = input(trim(left(substr(line,140,12))),hhmmss12.);
MonDT = put(datetime(),datetime21.2);
drop line;
 /* Site-customize and UnComment the section BELOW for testing, if desired.
file
%if &MonitorCycleCount eq 0
%then %do;
     "C:\LeRBmonitors\TasksFoundAtStartUp_&ReportTimeUnits..txt"
%end;
%else %do;
     "C:\LeRBmonitors\TasksFoundThisCycle_&ReportTimeUnits..txt"
%end;
                                               lrecl=229;
put @1 line $char229.;
    Site-customize and UnComment the section ABOVE for testing, if desired. */
output MonLib.ActiveProcesses;
%if &MonitorCycleCount eq 0
%then %do;
Ref_Cumulative_CPU_seconds = Cumulative_CPU_seconds;
Ref_MonDT                  = MonDT;
output MonLib.CPUmonRefTable;
%end;
run;

proc sort data=MonLib.CPUmonRefTable;
by Process_ID UserID;
run;

proc sort data=MonLib.ActiveProcesses;
by Process_ID UserID;
run;

data
Alerts               (keep=OrigID Process_ID     Cumulative_CPU_seconds     MonDT)
MonLib.CPUmonRefTable(keep=UserID Process_ID Ref_Cumulative_CPU_seconds Ref_MonDT);
merge MonLib.CPUmonRefTable(in=Ref) MonLib.ActiveProcesses(in=Active);
by Process_ID UserID;
if Active;
if Cumulative_CPU_Seconds GT &CPUthresholdInSeconds;
      if NOT Ref
      then do;
        output Alerts;
        Ref_Cumulative_CPU_seconds = Cumulative_CPU_seconds;
        Ref_MonDT                  = MonDT;
      end;
      else do;
%if &MonitorCycleCount eq 0
%then %do;
        output Alerts;
%end;
%else %do;
        if Cumulative_CPU_seconds GE
           Ref_Cumulative_CPU_seconds + &CPUincrementForAnotherAlert
        then do;
          output Alerts;
          Ref_Cumulative_CPU_seconds = Cumulative_CPU_seconds;
          Ref_MonDT                  = MonDT;
        end;
%end;
      end;
output MonLib.CPUmonRefTable;
run;

data _null_;
call symput('AlertsCount',ObsCount); stop;
set Alerts nobs=ObsCount;
run;

%if %eval(&AlertsCount GT 0)
%then %do;
data _null_;
length AddressAndName Address Name $ 80;
set Alerts;
AddressAndName = put(OrigID,$Eaddr.);
Address = substr(AddressAndName,01,40);
Name    = substr(AddressAndName,41,40);
if Name EQ "&DefaultExplain28Characters"
then Name = trim(left(OrigID)) || " is &DefaultExplain28Characters";
call symput('MonDT'||trim(left(_N_)),trim(left(MonDT)));
call symput('ID'   ||trim(left(_N_)),trim(left(OrigID)));
call symput('Addr' ||trim(left(_N_)),trim(left(Address)));
call symput('Name' ||trim(left(_N_)),trim(left(Name)));
call symput('PID'  ||trim(left(_N_)),trim(left(Process_ID)));
  %if %upcase(&ReportTimeUnits) = HOURS
  %then %do;
call symput('Hrs'  ||trim(left(_N_)),
            trim(left(put(Cumulative_CPU_seconds      /3600,6.2))));
call symput('MsgIncrementValue'     ,
            trim(left(put(&CPUincrementForAnotherAlert/3600,6.2))));
call symput('MsgTriggerValue'       ,
            trim(left(put(&CPUthresholdInSeconds      /3600,6.2))));
  %end;
  %else
  %if %upcase(&ReportTimeUnits) = MINUTES %then %do;
call symput('Mins' ||trim(left(_N_)),
            trim(left(put(Cumulative_CPU_seconds      /60,comma8.1))));
call symput('MsgIncrementValue'     ,
            trim(left(put(&CPUincrementForAnotherAlert/60,comma8.1))));
call symput('MsgTriggerValue'       ,
            trim(left(put(&CPUthresholdInSeconds      /60,comma8.1))));
  %end;
  %else %do;
call symput('Secs' ||trim(left(_N_)),trim(left(Cumulative_CPU_seconds      )));
call symput('MsgIncrementValue'     ,trim(left(&CPUincrementForAnotherAlert)));
call symput('MsgTriggerValue'       ,trim(left(&CPUthresholdInSecond       )));
  %end;
run;

  %do i = 1 %to &AlertsCount %by 1;

    %if &&PID&i NE &sysjobid
    %then %do;

%if %upcase(&ReportTimeUnits) = HOURS
%then %let ReportTimeValue = &&Hrs&i;
%else
%if %upcase(&ReportTimeUnits) = MINUTES
%then %let ReportTimeValue = &&Mins&i;
%else %let ReportTimeValue = &&Secs&i;
%let Msg =
User ID &&ID&i Process &&PID&i Has Used &ReportTimeValue &ReportTimeUnits of CPU
Time.;
%let IncrementMsg =
You will get a message every time process &&PID&i uses another &MsgIncrementValue
&ReportTimeUnits of CPU time.;
%let TriggerMsg =
or it has used at least &MsgTriggerValue &ReportTimeUnits of CPU time and you get this
message.;

filename AnyEmail EMAIL
FROM="&Sender"
SENDER="&Sender"
TO=("&&Addr&i")
CC=("&Sender"
%if %length(&AlwaysNotify) NE 0 %then %do;
    "&AlwaysNotify"
%end;
)
SUBJECT="&Msg Is it OK?";

data _null_;
file AnyEmail;
put "Dear &&Name&i";
put ' ';
put "Please read this long automated message carefully and in entirety.";
put ' ';
put "As Of &&MonDT&i";
put "&Msg";
put "Is it OK? If not, please terminate it.";
put ' ';
put "For information on how to terminate a SAS process, go to";
put "the URL for that part of the SAS Server Site User Documentation.";
put "If you can not terminate it, call the SAS Server Administrator.";
put "If the SAS Server Administrator is unavailable, call the Help Desk.";
put "They need the Process ID &&PID&i and Server Name %sysget(computername).";
put ' ';
put "&IncrementMsg";
put "The process ID &&PID&i was assigned when your SAS Enterprise Guide or Display
Manager session or batch job started.";
put "It varies from session to session and from job to job.";
put ' ';
put "CPU time is NOT elapsed time. It is active processing time.";
put "The CPU waits idle for file reads, file writes, responses from Oracle, etc.";
put "CPU time is usually less than elapsed time.";
put ' ';
put "Using a lot of CPU time is not necessarily bad.";
put "Heavy-duty analytics processes can use a lot of CPU time.";
put "However, if your process is in a loop, it is wasting resources.";
put "If your process is from an abandoned or frozen Enterprise Guide session,";
put "you will be unaware of how much CPU time it is using";
put "unless you investigate with the process management tools,";
put "&TriggerMsg";
put "If you have any questions, call or send email to the SAS Server Administrator.";
run;

    %end;

  %end;

%end;

%let MonitorCycleCount = %eval(&MonitorCycleCount + 1);

%put This monitor cycle &MonitorCycleCount ran at &MonDateTime;
 /* If there is an ERROR or WARNING message in the log,
    the above statement lets you estimate the time of that message,
    which is NOT datetimestamped by SAS software. */

%if %upcase(&NumberOfMonitorEvents) NE NOLIMIT %then %do;
  %if %eval(&MonitorCycleCount GE &NumberOfMonitorEvents)
  %then %GoTo EndOfMonitorSession;
%end;

data _null_;
x = sleep(&WaitSeconds);
run;

%GoTo StartOfCycle;

%EndOfMonitorSession:

%mend CPUmon_macro_ForSGFdemo;

%CPUmon_macro_ForSGFdemo(WaitSeconds=360,
              NumberOfMonitorEvents=NoLimit,
              MonLib=PathToFolderForOutputCPUmonLogDataSets,
              ReportTimeUnits=Hours,
              CPUthresholdInSeconds=7200,
              CPUincrementForAnotherAlert=3600,
              PathToEmailAddressDataLib=C:\LeRBmonitors\EmailAddressLib,
              EmailAddressData=CPUusers,
              DefaultEmailAddress=Le_Roy_Bessler@wi.rr.com,
              DefaultExplain28Characters=UserID without Email Address,
              Sender=Le_Roy_Bessler@wi.rr.com,
              AlwaysNotify=LeRB.BackUp@somewhere.com); /* using mostly the defaults */
```

LogTimer Macro

```
%MACRO LogTimer(StartOrEnd);

%LET StartOrEnd = %UPCASE(&StartOrEnd.);
%GLOBAL LogTimerWasStarted;
%GLOBAL SaveStartDateTime;
%GLOBAL SaveStartCPUtime;
```

```
DATA _NULL_;
DateTime = DATETIME();
CALL SYMPUT('LogTimerDate',TRIM(LEFT(PUT(DATEPART(DateTime), DATE9.))));
CALL SYMPUT('LogTimerTime',TRIM(LEFT(PUT(TIMEPART(DateTime), TIME8.))));
IF "&StartorEnd" = 'START'
THEN CALL SYMPUT('SaveStartDateTime',PUT(DateTime, 22.10));
 /* maximum precision for saved Start datetime
    to avoid possible negative elapsed time */
ELSE DO;
  ElapsedSeconds = DateTime - INPUT(SYMGET('SaveStartDateTime'), 22.10);
  CALL SYMPUT('LogTimerElapsedTime',TRIM(LEFT(PUT(ElapsedSeconds, TIME.))));
END;
RUN;

%LET TaskListCommand = %STR('tasklist /v');
FILENAME TaskList PIPE &TaskListCommand.;
DATA _NULL_;
INFILE TaskList LRECL = 224 PAD END = LastOne;
 /* LRECL varies by Windows version.
    224 is appropriate for Windows XP.
    229 is for Windows 2003 Advanced Server. */
INPUT @1 CommandResponse $CHAR224.;
IF _N_ GE 4;
IF INDEX(CommandResponse, "&sysjobID.") NE 0;
Cumulative_CPUtime = TRIM(LEFT(SUBSTR(CommandResponse, 140, 12)));
 /* offset of Cumulative_CPU_time varies by Windows version.
    140 is appropriate for Windows XP.
    145 is for Windows 2003 Advanced Server. */
Cumulative_CPU_seconds = INPUT(Cumulative_CPUtime, HHMMSS12.);
IF "&StartorEnd." = 'START'
THEN CALL SYMPUT('SaveStartCPUtime',TRIM(LEFT(Cumulative_CPU_seconds)));
ELSE DO;
  CPU_seconds = Cumulative_CPU_seconds - INPUT(SYMGET('SaveStartCPUtime'), 8.);
  CALL SYMPUT('LogTimerCPUtime',TRIM(LEFT(PUT(CPU_seconds, TIME.))));
END;
RUN;
%PUT *****************************************;
%IF &StartOrEnd. = START
%THEN %DO;
  %PUT Started at &LogTimerTime. On &LogTimerDate.;
  %LET LogTimerWasStarted = YES;
%END;
%ELSE
%IF &StartOrEnd. = END
%THEN %DO;
  %IF &LogTimerWasStarted. EQ YES
  %THEN %DO;
    %PUT Ended at &LogTimerTime. on &LogTimerDate.;
    %PUT Elapsed Time (hours:minutes:seconds) = &LogTimerElapsedTime.;
    %PUT CPU Time (hours:minutes:seconds) = &LogTimerCPUtime.;
  %END;
  %ELSE %DO;
    %PUT LogTimer Macro User ERROR: Invocation Value was &StartOrEnd.;
    %PUT But there was no prior invocation with Start;
  %END;
%END;
%ELSE %DO;
  %PUT LogTimer Macro User ERROR: Invocation Value was &StartOrEnd.;
  %PUT Must be Start or End;
%END;
%PUT *****************************************;

%MEND LogTimer;
```

UserMon Code To Create UserMon Logs

```
options nosource nonotes nomprint nomprintnest nosymbolgen nomlogic;
/* Turn on one or more above options only for debugging. Since monitoring can run for
   days, weeks, or months, the SAS log can get very large. */

options pagesize=max;

%macro UserMon(WaitSeconds=360,NumberOfMonitorEvents=NoLimit,MonLib=);

%let MonitorCycleCount = 0;
%let DateOfLog = 0;
%let TimeOfLog = 0;

libname MonLib "&MonLib";

%StartOfCycle:

data _null_;
DateOfLog = datepart(datetime());
TimeOfLog = timepart(datetime());
if DateOfLog GT &DateOfLog
  or
   (DateOfLog EQ &DateOfLog and TimeOfLog GT &TimeOfLog)
then do;
  call symput('DateOfLog',trim(left(DateOfLog)));
  call symput('TimeOfLog',trim(left(TimeOfLog)));
  call symput('LogDateTime','D'|| trim(left(put(DateOfLog,yymmddn8.))) || '_' ||
       'T'|| trim(left(put(input(compress(put(TimeOfLog,time8.),':'),6.),Z6.))));
end;
run;

data _null_;
length TaskList $ 256;
TaskList = "'tasklist /v /fi " || '"imagename EQ sas.exe"' || "'";
call symput('TaskListCommand',trim(left(TaskList)));
run;

filename tasklist pipe &TaskListCommand;

data MonLib.UserMonLog_&LogDateTime(drop=line Memory_Usage);
length MonDateTime $ 21 UserName $ 50 Process_ID $ 6 Cumulative_CPUtime $ 12
       Cumulative_CPU_seconds Current_Memory_Usage_In_KB MonDT 8;
retain MonDateTime '01JAN1960:00:00:01.00' MonDt 0;
infile tasklist lrecl=229 pad;
input @1 line $char229.;
if _N_ EQ 1 then do;
  MonDT = datetime();
  MonDateTime = put(MonDT,datetime21.2);
  call symput('MonDateTime',trim(left(MonDateTime)));
end;
if _N_ GE 4;
Process_ID   = left(substr(line,27,8));
Memory_Usage = substr(line,65,12);
Current_Memory_Usage_In_KB =
  input(substr(Memory_Usage,1,index(Memory_Usage,'K') - 2),comma10.);
UserName     = substr(line,94,50);
Cumulative_CPUtime = trim(left(substr(line,145,12)));
Cumulative_CPU_seconds = input(Cumulative_CPUtime,hhmmss12.);
run;

%let MonitorCycleCount = %eval(&MonitorCycleCount + 1);
```

```
%put This monitor cycle &MonitorCycleCount ran at &MonDateTime;
* If there is an ERROR or WARNING message in the log, the above statement lets you
estimate the time of that message, which is NOT datetimestamped by SAS software. *;

%if %upcase(&NumberOfMonitorEvents) NE NOLIMIT %then %do;
  %if %eval(&MonitorCycleCount EQ &NumberOfMonitorEvents)
  %then %GoTo EndOfMonitorSession;
%end;

data _null_;
x = sleep(&WaitSeconds);
run;

%GoTo StartOfCycle;

%EndOfMonitorSession:

%mend UserMon;

%UserMon(MonLib=PathToFolderForOutputUserMonLogDataSets);
```

ConcatMonLogs Macro and Its Invocation Code To Concatenate UserMon Logs

```
%macro ConcatMonLogs(MonLib=,OutLib=SASUSER,out=ConcatMonLogs,
StartYYYYMMDD=00000000,EndYYYYMMDD=99999999,OmitLastMonLog=YES);

/* OmitLastMonLog=YES captures logs through the second latest one in MonLib */
/* This prevents conflict if EndYYYYMMDD is left at macro default,
   but UserMon is still writing the latest Monitor Log in MonLib. */

%global StartDT EndDT; /* for reference by subsequent processing outside of macro */

libname MonLogs "&MonLib";

proc sql;
create table work.MonLogs as
(select libname, memname from dictionary.tables where libname EQ 'MONLOGS');
quit;

proc sort data=work.MonLogs; by memname; run;

data work.MonLogs;
set work.MonLogs(where=("&StartYYYYMMDD" LE substr(memname,12,8) LE "&EndYYYYMMDD"));
run;

data _null_;
call symput('HowMany',trim(left(HowMany)));
stop;
set work.MonLogs nobs=HowMany;
run;

%if &HowMany EQ 0 %then %do;
  %put No Logs in Date Range &StartYYYYMMDD through &EndYYYYMMDD;
  %goto MacExit;
%end;

data _null_;
set work.MonLogs end=LastOne;
if _N_ EQ 1 then call symput('StartDT',substr(memname,11,17));
if not LastOne then do;
  call symput('Log'||trim(left(_N_)),trim(left(memname)));
  call symput('EndDT',substr(memname,11,17));
end;
```

```
else do;
  flag = "&OmitLastMonLog";
  if upcase(flag) EQ 'YES'
  then call symput('LogCount',_N_ - 1);
  else do;
    call symput('Log'||trim(left(_N_)),trim(left(memname)));
    call symput('EndDT',substr(memname,11,17));
    call symput('LogCount',_N_);
  end;
end;
run;

proc datasets lib=&OutLib nodetails nolist;
delete &out; run; quit;

%do i = 1 %to &LogCount %by 1;
proc append base=&OutLib..&out data=MonLogs.&&Log&i force; run;
%end;

%MacExit:

libname MonLogs clear;

%mend  ConcatMonLogs;

%ConcatMonLogs(MonLib= PathToFolderForUserMonLogDataSets,
out=ConcatMonLogsJan2009,OmitLastMonLog=NO,
StartYYYYMMDD=20090101,EndYYYYMMDD=20090131);
```

Example of a Query Macro to Analyze Monitor Logs

```
%macro WorkloadSummaryHistory(data=,RptPath=);

proc summary data=&data nway;
id MonDT;
class MonDateTime;
var Current_Memory_Usage_In_KB;
output out=ByMonitorInterval(rename=(_freq_=Number_of_Users_Or_Jobs))
sum=Total_Memory_Usage_In_KB;
run;

data ByMonitorInterval(drop=Day_Of_Week_Number);
length Day_Of_Week $ 9 Month $ 3 Day_Of_Month $ 2 Hour $ 2;
set ByMonitorInterval;
Hour = put(hour(MonDT),Z2.);
sasDate = datepart(MonDT);
Month = put(sasDate,monname3.);
Day_Of_Month = day(sasDate);
Day_Of_Week_Number = weekday(datepart(MonDT));
if Day_Of_Week_Number EQ 1
then Day_Of_Week = 'Sun';
else
if Day_Of_Week_Number EQ 2
then Day_Of_Week = 'Mon';
else
if Day_Of_Week_Number EQ 3
then Day_Of_Week = 'Tue';
else
if Day_Of_Week_Number EQ 4
then Day_Of_Week = 'Wed';
else
if Day_Of_Week_Number EQ 5
then Day_Of_Week = 'Thu';
```

```
else
if Day_Of_Week_Number EQ 6
then Day_Of_Week = 'Fri';
else Day_Of_Week = 'Sat';
run;

proc summary data=ByMonitorInterval nway;
id Day_Of_Week Month Day_Of_Month;
class sasDate Hour;
var Number_of_Users_Or_Jobs Total_Memory_Usage_In_KB;
output out=ByHourWithinDay(drop=_freq_) max=Max_Users_Or_Jobs
Max_Total_Memory_Usage_In_KB;
run;

 /* TITLE2 retrieves StartDT and EndDT passed by ConcatMonLogs macro */

title1 "Server SAS WorkLoad By Hour";
title2 "From &StartDT To &EndDT";
title3 "NOTE: Processes shorter than 6 minutes can be missed";
title1; /* If you turn on the titles, use HTML column spanning to prevent Column A
           being stretched to the width of the longest title line. */

ods html file="&RptPath.\WorkloadByHourWithinDay.xls" style=Minimal;
proc print data=ByHourWithinDay split='_' noobs;
var Day_Of_Month Month Day_Of_Week Hour Max_Users_Or_Jobs
Max_Total_Memory_Usage_In_KB;
run;
ods html close;

proc sort data=ByMonitorInterval;
by MonDT;
run;

title1 "Server SAS WorkLoad By Monitor Interval";
title2 "From &StartDT To &EndDT";
title3 "NOTE: Processes shorter than 6 minutes can be missed";
title1; /* If you turn on the titles, use HTML column spanning to prevent Column A
           being stretched to the width of the longest title line. */

ods html file="&RptPath.\WorkloadByMonitorInterval.xls" style=Minimal;
proc print data=ByMonitorInterval split='_' noobs;
* where '25Jan2009'd LE datepart(MonDT) LE '25Jan2009'd;
var MonDateTime Day_Of_Week Number_of_Users_Or_Jobs Total_Memory_Usage_In_KB;
label MonDateTime='Monitor DateTime';
run;
ods html close;

%mend WorkloadSummaryHistory;

%WorkloadSummaryHistory(data=SASUSER.ConcatMonLogsJan2009,RptPath=PathToReports);
```

ShowProcessID Macro

```
/* Run this macro at the start of your SAS Enterprise Guide session to be able to
distinguish your current SAS Enterprise Guide session’s process from the other one(s)
that you want to display and possibly terminate. */

%macro ShowProcessID;

%put Process ID for this SAS Enterprise Guide session or SAS batch job is &sysjobid;

%mend  ShowProcessID;
```

DisplayAllMySASprocesses Macro

```
%macro DisplayAllMySASprocesses;

%global SaveLineSize;
%let SaveLineSize = %sysfunc(getoption(LineSize));
options LineSize=229;
filename TaskList pipe 'tasklist /v';
data _null_;
infile TaskList lrecl=229 pad;
input @1 CommandResponse $char229.;
if _N_ LT 4 then put CommandResponse;
else
if index(CommandResponse,"&sysuserID") NE 0
  and
   CommandResponse =: 'sas.exe'
then put CommandResponse;
run;
options LineSize=&SaveLineSize;

%mend  DisplayAllMySASprocesses;
```

TerminateProcess Macro

```
%macro TerminateProcess(ProcessID=);

data _null_;
cmd = "'tasklist /v /fi " ||'"' || "PID EQ &ProcessID" || '"' || "'";
call symput('taskcmd',trim(left(cmd)));
run;
filename TaskList pipe &taskcmd;
data _null_;
infile TaskList lrecl=229 pad;
input @1 CommandResponse $char229.;
if CommandResponse EQ 'INFO: No tasks are running which match the specified criteria.'
then do;
  call sympput('Found','N');
  put "Process ID &ProcessID was not found.";
end;
else do;
  if _N_ GE 4;
  if index(CommandResponse,"&sysuserid") EQ 0 then do;
    call sympput('Found','N');
    put "Process ID &ProcessID is not for User ID &sysuserid and will not be killed.";
  end;
  else call sympput('Found','Y');
end;
run;

%if &Found EQ N %then %goto MacExit;

data _null_;
cmd = "'taskkill /fi " ||'"' || "PID EQ &ProcessID" || '"' || "'";
call symput('taskcmd',trim(left(cmd)));
run;
filename TaskKill pipe &taskcmd;
data _null_;
infile TaskKill lrecl=229 pad;
input @1 CommandResponse $char229.;
put CommandResponse;
run;

%MacExit:
%mend  TerminateProcess;
```