Paper 274-2009

Using SAS® to Manage, Monitor, and Control the SAS® BI Server: User-Developed Custom Tools for the SAS® Server Administrator, User, or Manager

LeRoy Bessler, Assurant Health, Milwaukee, Wisconsin, USA

Abstract

Who is using SAS now and how much CPU time and memory? Since when? When is the last time each user was on the server? How heavily does each use the server in terms of frequency or CPU time? 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? These are among the questions and needs that can be addressed by custom tools developed with SAS software itself and SAS-submitted Windows DOS commands.

Introduction

Intended Audience

The intended audience for the user monitoring tool is SAS BI Server administrators and possibly their managers. UserMon can not be implemented or operated by users of the BI Server. If its data is made read-accessible to users, they, like an administrator, can run queries against the data, including data pertaining to processes other than theirs.

The ShowProcessID, DisplayAllMySASprocesses, and TerminateProcess macros are intended for any server 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 the server with DOS commands, rather than have to open SAS Enterprise Guide and run some macros.

Why I Did This

I got into the SAS Enterprise Guide client / remote SAS server architecture with SAS EG 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. There are 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. The monitor I developed captures enough for my needs now, and the configuration, logging, and data management are customizable to my personal preference, and are changeable by me at will.

Scope of the Paper

The server operating system is Windows. I don't know whether UNIX analogues to these tools can be developed. This implementation specifically relies on Windows commands. The environment is Windows 2003 Advanced Server and Version 9.1.3 of SAS software. The only SAS component required is Base SAS. Macro language is required. ODS can be used to format reports of the 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.

The monitor captures user ID (misleadingly called UserName by the Windows). More useful reports include the actual name of the user, which could, e.g., be obtained from an export of the user information sector of the BI Server metadata database.

UserMon

The heart of this tool 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 probably Windows-version-dependent. The code supplied here assumes that the OS is Windows 2003 Advanced Server.

UserMon 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, UserMon could run as a Windows service that starts automatically at reboot.

The monitor is not a big resource burden. Its CPU time and disk space needs are not significant. For each monitor interval, it 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. Below are excerpts from two of many reports that can be created from the data.

	A	B	C	D	E	F
1	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	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

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

Tools for Users: What the SAS Enterprise Guide (SAS EG) Facilities Do Not Help With

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

Case 1. Your SAS EG 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 EG session, or if you used Windows Task Manager to kill SAS EG in desperation because it was frozen.

Case 2. Your EG 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 the Process Display and Termination Tools

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 EG 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 for the code for these macros.

Conclusion

SAS can be used to monitor and control the SAS BI server. To support the server, and to further empower server users, I have developed other tools. One of the tools not presented here 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 EG submissions are supported by EGbatch with no extra coding required in the submitted code itself. If interested, send me an email request for details.

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."

Contact Information

Your comments, questions, and suggestions are welcome. All code presented here, as well as code for some queries against UserMon data **not** 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 additional software and do not require non-SAS coding (other than DOS commands), I would be interested to see them.

LeRoy Bessler PhD
Email: Le_Roy_Bessler@wi.rr.com

Appendix 1.

UserMon.sas Code To Create UserMon Logs

```
options nosource nonotes noprint noprintnest 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;
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;
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 first to be able to distinguish your current EG 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 symput('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 symput('Found','N');
    put "Process ID &ProcessID is not for User ID &sysuserid and will not be killed.";
  end;
  else call symput('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;
```