Paper 356-2012

Another Way to Use SAS® to Monitor SAS or a SAS Server: A Tool for the User, Server Administrator, or Manager

LeRoy Bessler, Bessler Consulting and Research, Mequon, Milwaukee, WI
Victor Andruskevitch, Valence Health, Chicago, IL

ABSTRACT

The administrator or manager of a SAS server, whether BI or non-BI, has questions. Who is using SAS now and in the past, and how much of its resources (CPU, memory, or I/O)? How heavily does each use the server in terms of frequency or resources? The user, whether SAS is running on a remote server or on his/her own PC, has questions. What SAS processes do I have running? What resource consumption is making my process run so long?

By tapping into a no-additional-expense technology built into Microsoft Windows, this paper enhances the UserMon tool presented in 2009. The CPUmon tool presented in 2010 could also be enhanced to send real-time email alerts for overloads on resources other than processor time. *The tool presented here in 2012 can be adapted to also or to instead monitor any application other than SAS.*

INTRODUCTION

Every SAS administrator gets asked, “Why my jobs are running so slow?” Answering that question is not always simple. Elapsed time for a batch job or for interactive service request is impacted by data volumes, coding efficiency, and system resources. Coding efficiency and data volumes can be determined by reviewing the SAS log, but the SAS administrator does not have a clear picture of how competition for the shared system resources is impacting performance. This paper focuses on providing the SAS administrator some tools that can bring understanding at the system, user, and job level.

The objective of this monitor was to provide insight into a multi-server SAS configuration: which SAS applications, jobs, and users are responsible for how much resource use, whether CPU time, I/O volume, or memory. The tool described here meets that need.

The servers currently under monitoring are two non-BI SAS servers that serve Display Manager interactive users and Production Batch jobs and a SAS EBI server that serves Enterprise Guide, Web Report Studio, Stored Processes, and an Information Delivery Portal.

One of us (Bessler) had developed a simpler monitoring tool, both for post-use analysis and reporting, and for real-time alerts, that focusses exclusively on CPU time, and which has been reported in prior papers (References 1 and 2). That concept was enhanced here by supplementing the use of the DOS TaskList command with queries to the Windows Management Instrumentation, which comes at no additional charge with the Windows OS.

Windows Task Manager can be invoked with a DOS command, analogous to the situation with TaskList, but TaskMgr only can launch an interactive display, and cannot return a program-readable file like TaskList.

We stumbled across a paper (Reference 3) that used Windows Management Instrumentation (hereafter and herein abbreviated as WMI) to retrieve static system information about Windows XP workstation. A Google search turned up the full list (Reference 4) of static information and dynamic resource use information available in WMI, including the Command Line for the responsible executable consuming resources, its Process ID, Parent Process ID, Session ID, and Start Date Time. The one item not available from WMI is Session Name. So, the tool you will find here retains use of TaskList, in conjunction with WMI, to retrieve all information of interest.

COMPONENTS OF THE MONITORING PACKAGE

Continuous Data Collection

The heart of the package is a SAS program, UserMonViaWMI.sas, which runs indefinitely, waking up (as our choice) every six minutes to collect information about all resident SAS processes and to file it as a datetimesuffixed SAS data set.

The SAS program launches a VB script, Process_List.vbs, which first launches a bat file, TaskList.bat. The TaskList.bat retrieves process information and creates a CSV file for all sas.exe processes, which consists of merely their process IDs (PIDs) and Session Names. The CSV file is read back in by the VB script to drive the query from WMI of all of the other process-related information. That information, plus PID and Session Name, is used to create a Process_List.CSV file. Upon completion of the query loop and termination of the VB script, the SAS program reads the Process_List.CSV file to create a UserMonLog_dYYYYMMDD_tHHMMSS.sas7bdat SAS data set.

The interest in Session Name stemmed from the desire to be able to distinguish interactive versus batch processes, and Session Name was another way to make that inference, besides examining the Command Line provided by WMI. The most important value of Command Line is, in the case of batch, the ability to parse out the value of the SYSIN parameter passed in the Windows bat file, this value being the path and SAS program name for the application being run in batch. Such parsing is part of the initial phase of the daily reporting process, and not injected as overhead during data collection.

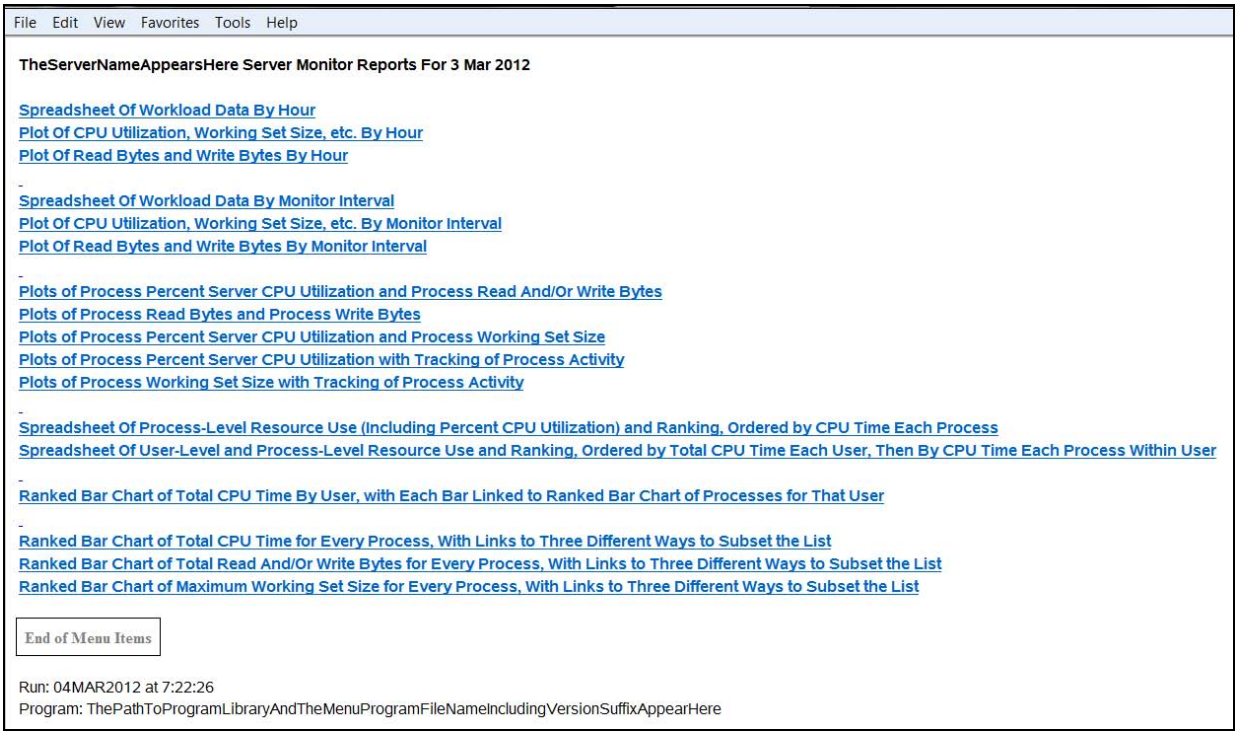
Data Concatenation into Daily History

Every day, after midnight, all of the UserMonLog_dYYYYMMDD_tHHMMSS.sas7bdat files found for the preceding day are concatenated into a daily history ConcatMonLogs_YYYYMMDD.sas7bdat SAS data set. The concatenator counts the number of interval SAS data sets. If the count falls below a predefined threshold, the program sends an email with the count and the total datetime range covered by the available data sets. If the datetime range spans midnight to midnight, a low count of interval data sets might reflect the fact that a very busy server took much longer than expected to complete many of its monitor processing events. It could also reflect that fact that a monitor outage occurred, but it was restarted. A datetime range shorter than midnight to midnight would reveal the extent of the monitor outage.

Data Analysis and Reporting

The SAS-based data analysis and reporting system uses yesterday's daily history as input.

Below is an excerpt from a screen print of the reports home page.



Its outputs are a web-deployed collection of spreadsheets and graphs, all of this menu-accessible, but also interlinked.

The detail spreadsheets provided as drill-down targets are a convenient way to present all of the processes during a specific monitor event time, or to present all of the monitor events for a specific process. Other spreadsheets supply the data for time plots by hour or by interval. These spreadsheets are linked forwards and backwards with the corresponding time plots. And two other spreadsheets are ranked summaries of the day's data: (a) ordered by resource use impact of user and process within user; and (b) ordered by resource impact of process.

The graphs are plots over time, at hour summary level or monitor interval level, and horizontal bar charts. The horizontal bar charts are a convenient way to compare workload impact, for the day: either by user and, with a click, by process within user; or ranked descending on resource volume consumed and subsetted: (a) all processes (i.e., no subsetting); (b) top N processes; (c) all processes above some threshold; and (d) enough processes to account for the P% of the grand total resource consumption.

Examples of the outputs will be presented during the stand-up slide presentation. The code for the daily reporting program is site-specific, and will not be published here.

For how, in a non-site-specific context, to create time plots and supporting spreadsheets that are linked forwards and backwards, see Reference 5.

A general-purpose macro to create the four interlinked ranked and subsetted horizontal bar charts will be published separately by one of the authors (Bessler), and that reference can be requested from him.

The substructure for the daily reporting system was also adapted to be invoked for ad hoc queries, and, importantly, was provided a front-end query step to select any date-hour to date-hour range of monitor log data, including, if desired, today's data, except for that of the last monitor interval, for which the SAS data set could currently be locked in write mode.

Whether running in standard scheduled batch mode, or ad hoc Enterprise-Guide-launched mode, the code to analyze and report the monitor data writes the main body of its SAS log to a disk file with PROC PRINTTO.

An important, but perhaps not obvious as a needed implementation feature, was the provision of a SAS log parser for both the daily standard reporting and for the ad hoc query and reporting. Generating the collection of outputs produces a huge SAS log, which, if one runs the SAS code in Enterprise Guide, is beyond the capacity of Enterprise Guide to accept in the log window. Furthermore, there can be numerous potentially disquieting WARNING and NOTE messages, which can be safely disregarded. If one wants to be confident that a run was successful, an inspection of the SAS log is mandatory. However, it becomes impractical to visually scan the large SAS log, even with repeated FIND commands (for which you would need to maintain a checklist of To Do's), and, anyone but the program author might be uncertain as to which messages can be safely disregarded.

The log parser not only disregards, based on imbedded rules, certain messages, but also generates a convenient report on any anomalies found, with a listing of each distinct anomaly and a frequency distribution for any anomalies found. (This was also a very helpful tool during development.) If any anomalies are found, which is likely to be infrequent, an email is sent to support staff for the standard daily production run, and, in the case of an ad hoc run, to the user who submitted the run. The follow-up to an anomaly is to either fix the code, if needed, or to revise the log parser to disregard the new, but innocuous, previously unexpected message.

CONCLUSION

The tool presented here delivers user / process level insight into CPU, I/O, and memory resource use by SAS users and SAS applications. The machinery can be adapted to monitor resource use, if desired, by non-SAS users and applications. Also, just as the user monitor in Reference 1 was adapted to create to code used to provide real-time alerts for excessive resource use, so, too, the WMI-based user monitor could be similarly adapted. A foundation was built, but there is much more to do: setting up alerts, analyzing job performance history to identify consistently high resource burdens (i.e., targets for optimization), and building dashboards.

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*. Cary, NC: SAS Institute Inc. 2009. Find it on the web at http://support.sas.com/resources/papers/proceedings09/274-2009.pdf

2. Bessler, LeRoy. “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”, *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. 2010. Find it on the web at http://support.sas.com/resources/papers/proceedings10/279-2010.pdf

3. Andrews, Rick and Dixon, Sherry. "Performance Monitoring for SAS Programs on Windows XP", *Proceedings of the Thirtieth Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. 2005. Find it on the web at http://www2.sas.com/proceedings/sugi30/022-30.pdf

4. “Win32_Process class” (Context is Windows Dev Center > Desktop > Learn > Reference > System Administration > Windows Management Instrumentation > WMI Reference > WMI Classes > Win32 Classes). Microsoft Corp. 2012. Find it on the web at http://msdn.microsoft.com/en-us/library/windows/desktop/aa394372(v=vs.85).aspx

5. Bessler, LeRoy. “ The Most Communication-Effective and Most Usable Information Delivery”, *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. 2010. Find it on the web at http://support.sas.com/resources/papers/proceedings10/231-2010.pdf

CONTACT INFORMATION

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

LeRoy Bessler PhD
Bessler Consulting and Research
PO Box 96
Milwaukee, WI 53201-0096
262-242-1099
Le_Roy_Bessler@wi.rr.com

Victor Andruskevitch
Valence Health
600 W. Jackson Suite 800
Chicago, IL 60661
Phone: 312.277.4801
Fax: 312.277.0330
E-mail: vandruskevitch@valencehealth.com
Web: http://valencehealth.com

APPENDIX 1. COMPONENTS FOR MONITOR DATA COLLECTION

NOTE 1: Throughout the code provided here, the string “Path” is used to remove all site specificity.
NOTE 2: The bat files, the VB script, and the SAS program in this appendix all retrieve the host name where they are running so that the same code can be used to monitor any server by being run on that server.
NOTE 3: The line breaks in the two bat files and the VB script file in this Appendix could cause problems if the code is copied and pasted into your files without then removing the line breaks.

A. TaskList.bat (to retrieve PID and SESSION NAME for every SAS process)

```
TASKLIST /FO "CSV" /NH /FI "IMAGENAME EQ SAS.EXE" >
Path\%computername%\Work\SAS_PID.CSV
```

B. ProcessListToWorkFolder.vbs (to retrieve WMI process data for every SAS process identified by TaskList.bat)

```
Set oShell = CreateObject("WScript.shell")
sCmd = "Path\JobLib\TASKLIST.BAT"
oShell.Run sCmd, 0, True

Const ForReading = 1
Const ForWriting = 2

strComputer = "."

Set wshShell = WScript.CreateObject( "WScript.Shell" )

strComputerName = wshShell.ExpandEnvironmentStrings( "%COMPUTERNAME%" )

Set objFSO = CreateObject("Scripting.FileSystemObject")

if objFSO.FileExists("Path" & strComputerName & "\Work\SAS_PID.CSV") then
  Set objFile = objFSO.OpenTextFile("Path" & strComputerName &
"\Work\SAS_PID.CSV",ForReading)
end if

Set objFSO = CreateObject("Scripting.FileSystemObject")

Set objLogFile = objFSO.CreateTextFile("Path" & strComputerName &
"\Work\ProcessList.csv",ForWriting)

  objLogFile.Writeline
"ImageName,CSName,MonDateTime,Process_Start,Process_Owner,PID,Parent_PID,Session_Name,
CPUtime_In_Seconds,Handles,Threads,Working_Set_Size_In_KB,Page_File_Size,Page_Faults,R
ead_Bytes,Write_Bytes,Other_Bytes,CommandLine"

Do Until objFile.AtEndOfStream
  strLine = objFile.ReadLine
  arrFields = Split(strLine, ",")
  intPID=arrFields(1)
  strLoginType=arrFields(2)

  Set objWMIService = GetObject("winmgmts:" _
    & "{impersonationLevel=impersonate}!\\" _
    & strComputer & "\root\cimv2")

  Set colProcesses = objWMIService.ExecQuery ("Select * from Win32_Process WHERE
ProcessID = " & intPID & "")

  On Error Resume Next

  For Each objProcess in colProcesses

    sngProcessTime = (CSng(objProcess.KernelModeTime) + CSng(objProcess.UserModeTime))
/ 10000000

    colProperties = objProcess.GetOwner(strNameOfUser,strUserDomain)

  objLogFile.Writeline objProcess.Name _
    & "," & objProcess.CSName _
    & "," & now() _
    & "," & objProcess.CreationDate _
    & "," & strUserDomain & "\" & strNameOfUser _
    & "," & objProcess.ProcessID _
    & "," & objProcess.ParentProcessID _
    & "," & strLoginType _
```

```
    & "," & sngProcessTime _
    & "," & objProcess.HandleCount _
    & "," & objProcess.ThreadCount _
    & "," & objProcess.WorkingSetSize _
    & "," & objProcess.PageFileUsage _
    & "," & objProcess.PageFaults _
    & "," & objProcess.ReadTransferCount _
    & "," & objProcess.WriteTransferCount _
    & "," & objProcess.OtherTransferCount _
    & "," & objProcess.CommandLine

  Next

Loop

objLogFile.Close
```

C. UserMonViaWMI.sas (to wake up periodically and collect data for every SAS process)

```
%let UpcasedHostName = %upcase(&syshostname);

%put This monitor is running as Process ID &sysjobid on Host &UpcasedHostName for
UserName &sysuserid;

%let ProblemDescription = %str(); /* Null loading for a PUT of Problem Description
that might never run, but would be best compiled without WARNINGs. */

%let VBscript = VBscriptThatWillFail.vbs; /* use for sysrc NE 0 testing */

%let VBscript = ProcessListToWorkFolder.vbs; /* if last assignment, it prevails */

%let EmailFromAndCC = lbessler@valencehealth.com;
%let EmailTo        = %nrstr('vandruskevitch@valencehealth.com');
%let EmailBCC       = %nrstr('Le_Roy_Bessler@wi.rr.com');

%let SecondsBetweenMonitorEvents = 360;

%let NumberOfEvents = NOLIMIT; /* alternative value is any integer */

proc printto
log="Path\&UpcasedHostName.\LogLib\SASlogFor_UserMonViaWMI_&sysdate._%sysfunc(compress
(&systime,':')).txt";
run;

%put This monitor is running as Process ID &sysjobid on Host &UpcasedHostName for
UserName &sysuserid;

%let PathToVBS = Path\VBS; * where ProcessListToWorkFolder.vbs is located *;
%let PathToProcessList = Path\&UpcasedHostName.\Work; * this path must match
                         what is used in ProcessListToWorkFolder.vbs *;

options nosource nonotes noprint noprintnest nosymbolgen nomlogic nofullstimer;

* 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 source;
* options notes;
* options mprint mprintnest;
* options symbolgen;
* options mlogic;
```

```
options pagesize=max linesize=max;

%macro SASemail(From=,To=,BCC=,Subject=);

* provides filename AnyEmail EMAIL statement needed by DATA _null_ step that uses PUT
'any message text' statements to create email body content sent at end of step *;

filename AnyEmail EMAIL FROM="&From" SENDER="&From" TO=(&To) CC=("&From")
%if %length(&BCC) NE 0 %then %do;
        BCC=(&BCC)
%end;
        SUBJECT="&Subject";

%mend  SASemail;

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

%let SendEmailAtTermination = N; /* Program could be modified to send email
  at NORMAL termination, not just abnormal. This flag set to Y only for abnormal. */

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

%let SavedMonDT = 0;

%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;

options noxwait;
* Execute VB Script to list WMI-sourced information about processes *;
options xsync; * xsync will allow process to finish *;

%sysexec wscript &PathToVBS\&VBscript;

* script above puts process list csv into folder specified by the PathToProcessList
macro variable specified by percentLET statement at top of this program *;

%if &sysrc NE 0 %then %do;

  %let SendEmailAtTermination = Y;
  %let Subject = UserMonViaWMI Problem and Termination on Server &UpcasedHostName -
sysrc from run of VB script was &sysrc;
  %let ProblemDescription = On Server &UpcasedHostName sysrc from run of VB script
&PathToVBS\&VBscript was &sysrc;

  %put ********************************;
  %put &ProblemDescription;
  %put GoTo EndOfMonitorSession;
  %put ********************************;
```

```
  %GoTo EndOfMonitorSession;

%end;

libname MonLib "&MonLib";

data MonLib.UserMonLog_&LogDateTime(compress=NO
                                    /* compress=YES would make SAS data sets larger */
     drop=ImageNameShouldBeSASdotEXE PriorMonDTfromWMI
     Process_Start Start_SASdate Start_SAStime Process_Owner);

retain PriorMonDTfromWMI &SavedMonDT;

length
CSName $ 16
MonDateTime   $ 22
StartDateTime $ 22
Domain $ 32
UserName $ 32
PID $ 8
Parent_PID $ 8
Session_Name $ 32
CommandLine $ 512
CPUtime_In_Seconds 8
Handles 8
Threads 8
Working_Set_Size 8
Page_File_Size 8
Page_Faults 8
Read_Bytes 8
Write_Bytes 8
Other_Bytes 8
MonDT 8
StartDT 8
Process_Start $ 25
Process_Owner $ 64
;

infile "&PathToProcessList.\ProcessList.csv" DLM=',' lrecl=1024 pad end=LastOne;

input @1 ImageNameShouldBeSASdotEXE $7. @;

if ImageNameShouldBeSASdotEXE EQ 'sas.exe';

  input
  CSName $
  MonDateTime $
  Process_Start $
  Process_Owner $
  PID $
  Parent_PID $
  Session_Name $
  CPUtime_In_Seconds
  Handles
  Threads
  Working_Set_Size
  Page_File_Size
  Page_Faults
  Read_Bytes
  Write_Bytes
  Other_Bytes
  CommandLine $
```

```
  ;

  UserName =
    substr(Process_Owner,
           index(Process_Owner,'\') + 1,
           length(trim(left(Process_Owner))) - index(Process_Owner,'\'));
  Domain   = substr(Process_Owner,1,index(Process_Owner,'\') - 1);

  Start_SASdate = input(substr(Process_Start,1,8),yymmdd8.);
  Start_SAStime = hms(substr(Process_Start,9,2),
                      substr(Process_Start,11,2),
                      substr(Process_Start,13,2));
  StartDT = input(trim(left(put(Start_SASdate,date9.))) || ":" ||
                  trim(left(put(Start_SAStime,time8.))),datetime18.);
  StartDateTime = trim(left(put(month(Start_SASdate),2.))) || '/' ||
                  trim(left(put(day(Start_SASdate),2.)))   || '/' ||
                  trim(left(put(year(Start_SASdate),4.)))  || ' ' ||
                  trim(left(put(Start_SAStime,timeampm11.)));

  MonDT = input(MonDateTime,mdyampm22.);

  if NOT (MonDT GT PriorMonDTfromWMI) /* Comment out NOT
                                         to force a fake error situation */
  then do;
    call symput('SendEmailAtTermination','Y');
    call symput('Subject',"UserMonViaWMI Problem and Termination on Server
&UpcasedHostName - VB script could not return fresh information");
    call symput('ProblemDescription',"On Server &UpcasedHostName VB script
&PathToVBS\&VBscript could not return fresh information");
    put '**********************';
    put "&ProblemDescription";
    put '**********************';
    call symput('CSVfileRefreshed','N');
    stop;
  end;
  else do;
    call symput('CSVfileRefreshed','Y');
    output;
  end;

if LastOne;
call symput('SavedMonDT',MonDT);
call symput('MonDateTimeForPutToSASlog',trim(left(put(MonDT,datetime21.2))));

run;

%if &CSVfileRefreshed EQ N
%then %goto EndOfMonitorSession;

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

%put This monitor cycle &MonitorCycleCount ran at &MonDateTimeForPutToSASlog;
* 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:

%put Came To EndOfMonitorSession;
%if &SendEmailAtTermination EQ Y
%then %do;
  %SASemail(From=&EmailFromAndCC,To=&EmailTo,BCC=&EmailBCC,Subject=&Subject);
data _null_;
file AnyEmail;
put 'UserMonViaWMI Problem and Termination';
put "&ProblemDescription";
run;
%end;

%mend UserMonViaWMI;

%UserMonViaWMI(WaitSeconds=&SecondsBetweenMonitorEvents,
  NumberOfMonitorEvents=&NumberOfEvents,MonLib=Path\&UpcasedHostName.\MonLibForWMI);

proc printto;
run;
```

D. UserMonViaWMI.bat (to run the data collector SAS program)

```
"C:\Program Files\SAS\SASFoundation\9.2\sas.exe"
  -sysin Path\PgmLib\UserMonViaWMI.sas
  -log   Path\%computername%\LogLib\StartUpSASlogFor_UserMonViaWMI.txt
  -print Path\%computername%\Print\PrintFor_UserMonViaWMI.txt
  -work Path\%computername%\Work
  -icon
  -nosplash
```