Paper 022-30

Performance Monitoring for SAS® Programs on Windows® XP

Rick Andrews, Centers for Medicare and Medicaid Services, Baltimore, MD
Sherry Dixon, Ph.D., National Center for Healthy Housing, Columbia, MD

ABSTRACT

The SAS System offers statisticians, programmers, and analysts a myriad of techniques, which can be used to sort, merge, manipulate, quantify and communicate data. The exact same result can be achieved in many different ways using SAS, though one method may finish much faster or use fewer resources than another. Although the difference in run time or resource allocation between analysis methods may be practically insignificant for many programs, it may be large for programs with extensive analyses. This paper will present a process that shall be called the "wrapper" program that will capture statistics about the performance of a workstation when using different analysis methods.

The process will demonstrate how to programmatically create performance measurements of a workstation while processes are executing. This will be done by examining CPU usage, memory allocation, paging, SAS work space and I/O utilization in a Microsoft® Windows XP environment.

INTRODUCTION

The Performance Monitor found on recent versions of the Windows operating system will collect various performance statistics relative to a local or remote computer. Logged counter data can be viewed using the System Monitor or exported into text files, binary files, or SQL Server database for further analysis and reporting. The application can be executed by typing PERFMON from a DOS prompt, which will give an interactive view of the statistics captured from the computer (Figure 1, Interactive Performance Monitor).

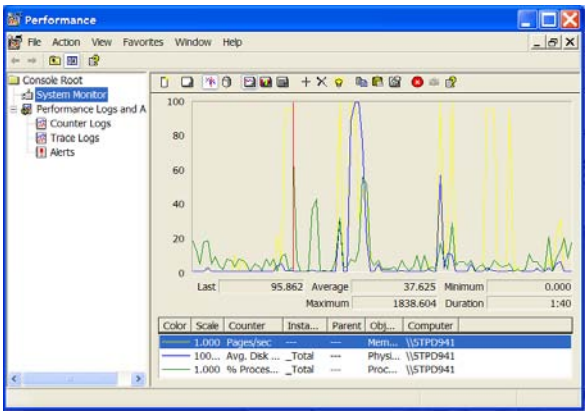

Figure 1: Interactive Performance Monitor

This paper provides a method of repeatedly capturing the performance data using an executable called the LOGMAN, which is new to Windows XP. The process will obtain various system-information such as processor type and clock speed, virtual memory, operating system and other data using a technology called Windows Management Instrumentation (WMI). This is a powerful set of system management features that will be accessed using Visual Basic™ scripting (VBScript) techniques. In order to fully utilize all of the features within this paper the computer must have the Windows XP operating system or greater installed, though WMI scripts can be run on Windows 98, NT, and 2000 with Active Directory Client Extensions installed (Microsoft Corporation, 2000).

WRAPPER PROGRAM

Assessing the performance of one computing method over another can be quite difficult. Evaluating the amount of CPU time, memory usage, or whether paging is occurring is daunting enough without trying to keep track of each particular program execution within a test plan. The wrapper program will provide a process that can be used to repetitively capture computer resource statistics and store the information into data sets for subsequent review using SAS procedures such as PROC REPORT and PLOT, which can be exported into a number of formats using the Output Delivery System (ODS).

The paper is meant to be a "how-to" guide containing the SAS code needed to create this process. We will be explaining these methods:

1) The **SysInfo** VBScript captures information about a computer.
2) The **SasWork** VBScript captures information about SAS WORK usage.
3) The **SasWorkTerm** VBScript shows how to terminate the SasWork process.
4) The **LogMan** execution captures the performance statistics of the computer.

The parameters found in Figure2, PerfMon Parameters, will allow the process to store information about each of the performance tests with the name of the SAS program, a version control number, and optionally the number of test records used. Keep in mind that the system administrator may configure the computer in such a way that will not allow these processes to be executed. Because of configuration differences adjustments may be required for portions of this process. If problems arise, execute each section individually, independent of the SAS System to determine the root cause.

SYSINFO SCRIPT

Ideally, one should examine the performance of SAS programs on different computers possessing varying levels of resources. To identify the available resources on a particular system this process will create a VBScript that programmatically exports system information to a text file, which can then be imported into SAS for reporting and analysis.

1. Create a pointer to the output file, SysInfo.txt.

2. Access the selected computer. Note that a period (.) points to the local computer.

3. Create a query that captures information from the WMI provider about the computer. Four of these provider classes are used in this paper.

   a. Win32_ComputerSystem
   b. Win32_OperatingSystem
   c. Win32_Processor
   d. Win32_LogicalDisk

SysInfo.vbs

```
1  Const ForWriting = 2
   Set objFSO = CreateObject("Scripting.FileSystemObject")
   Set objFile = objFSO.OpenTextFile("SysInfo.txt",ForWriting,True)

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

3  Set getValue = objWMIService.ExecQuery _
      ("Select * from Win32_ComputerSystem")

4  For Each objData in getValue
       objFile.WriteLine( "PhMemT ~ " & objData.TotalPhysicalMemory)
   Next

   objFile.Close
```

Each class contains a variety of objects that possess information about the computer, such as the make and model, the amount of memory available, the processor type, clock speed, and much more.

4. The `For` loop outputs the value within each object for each class. Multiple `WriteLine` functions can be created for all of the objects within a provider class in lieu of just one as shown here. The `GetSysInfo` macro was created to export them individually. (Figure 3: SysInfo Script Creation)

SASWORK SCRIPT

Create another VBScript called SasWork.vbs that will capture the SAS WORK space used at time interval from the `TestInt` macro variable. (Figure 4: SasWork Script)

1. Create a pointer to the output file, SasWork.txt.

2. Point to the SAS WORK folder and output its size to the text file at each time interval.

SasWork.vbs

```
1  Const ForWriting = 2
   Set objFSO = CreateObject("Scripting.FileSystemObject")
   Set objFile = objFSO.OpenTextFile("SasWork.txt", ForWriting, True)

2  Set objFolder = objFSO.GetFolder("C:\SAS Temporary Files\_TD1296")
   do
     objFile.WriteLine( "SASWORKSIZE: " & objFolder.Size )
     Wscript.Sleep 5000
   loop
```

SCRIPT TERMINATION

The SAS WORK space script will now be running in concert with the SAS program and will need to be terminated once the processing is complete. In order to determine which system process to terminate the program submits a query against the `Win32_Process` WMI class where the `CommandLine` value is `like '%cscript%SasWork.vbs%'`. (Figure 5: SasWork Script Termination)

SasWorkTerm.vbs

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

2  Set colProcessList = objWMIService.ExecQuery _
      ("Select * from Win32_Process Where CommandLine like
       '%cscript%SasWork.vbs%'")

3  For Each objProcess in colProcessList
       objProcess.Terminate()
   Next
```

1. Determine which computer to access. Note that a period (.) points to the local computer.

2. Identifies the SasWork script that is currently running on the local computer.

3. Terminate any process that contains the key phrases of "`cscript`" AND "`SasWork.vbs`".

This script assumes there is only **one** process running with the `CommandLine` value containing the key phrases of "`cscript`" AND "`SasWork.vbs`".

IMPORT SYSTEM INFORMATION

The process then imports the system information captured during the execution of the SysInfo.vbs script into SAS. This data will be used in calculations later in the program and for reporting purposes during the creation of the HTML web pages. (Figure 6: Import SysInfo Data).

GATHER SAS OPTIONS

Because it can be useful to change various SAS System options to improve performance the process captures and reports on some of these SAS options, as they exist during the execution of our program. (Figure 7: Capture SAS OPTIONS).

SysInfo.txt

```
ComNam ~ MYCOMPUTER
ComMod ~ Inspiron 4000
ComMan ~ Dell Computer Corporation
PhMemT ~ 133660672
OprSys ~ Microsoft Windows XP Professional
VrMemA ~ 169076
PhMemA ~ 20700
VrMemT ~ 440524
CpuNam ~ Intel Celeron processor
DskCap ~ 10051256320
DrvLtr ~ C:
FreSpc ~ 5194539008
```

PERFORMANCE MEASUREMENTS

One of the most misunderstood measurements in evaluating performance of programs is Real-Time. Often used as the major factor in determining how effective one method of processing is over another when there are a host of factors involved in measuring the effectiveness of a process. Table 1 describes the various performance counters presented here. These descriptions were obtained from the Microsoft Management Console. (Figure 8: Create LOGMAN Collection).

Table 1: Performance Measurements

Counter	Description
Process\% Processor Time	% Processor Time is the percentage of elapsed time that the processor spends to execute a non-Idle thread. It is calculated by measuring the duration of the idle thread is active in the sample interval, and subtracting that time from interval duration.
\Memory\Available Bytes	Available Bytes is the amount of physical memory, in bytes, available to processes running on the computer. It is calculated by adding the amount of space on the Zeroed, Free, and Standby memory lists.
\Memory\Pages/sec	Pages/sec is the rate at which pages are read from or written to disk to resolve hard page faults. This counter is a primary indicator of the faults that cause system-wide delays. It is the sum of Memory\\Pages Input/sec and Memory\\Pages Output/sec.
\Memory\Page Reads/sec	Page Reads/sec is the rate at which the disk was read to resolve hard page faults. Hard page faults occurs when a process references a page in virtual memory that is not in working set or elsewhere in physical memory, and must be retrieved from disk.
\Memory\Page Faults/sec	Page Faults/sec is the average number of pages faulted per second. It is measured in number of pages faulted per second because only one page is faulted in each fault operation; hence this is also equal to the number of page fault operations.
\Memory\Cache Faults/sec	Cache Faults/sec is the rate at which faults occur when a page sought in the file system cache is not found and must be retrieved from elsewhere in memory (a soft fault) or from disk (a hard fault).
\PhysicalDisk(0)\% Disk Time	% Disk Time is the percentage of elapsed time that the selected disk drive was busy servicing read or write requests.
\Paging File(\??\C:\pagefile.sys)\% Usage	The percent of the Page File instance in use. See also Process\\Page File Bytes.

The “wrapper” program uses some of these measurements to determine what is happening on the computer as the SAS program is running. Papers have been written that discuss the use of various Memory counters to evaluate performance using different methods to determine when excessive paging is occurring (Brown, 2001; Walker, 1997). The statement developed within was derived from these various methods, intended to determine when excessive paging is occurring.

Measuring the performance of a computer can be quite involved. A baseline needs to created, a starting reference point of performance statistics in order to begin the interpretation process. The importance of multiple submissions of the same exact process to identify the outliers is the key to success. If consistent outliers occur from the baseline, this would indicate a problem. How to take action and fix the legitimate issues can often be as simple as adding more memory or obtaining a faster processor. When these are not options, more creative measures need to be tried and tested. Evaluating these statistics and enhancing the performance of SAS programs is beyond the scope of this paper.

LOGMAN BASICS

This application manages the "Performance Logs and Alerts" service for creating and managing Session and Performance logs on a Windows XP operating system (Microsoft, 2001). The program produces a collection of measurements that can be output to a text file, which in turn can be imported into a SAS data set for analysis.

The LOGMAN is an executable program on the computer that is submitted from a DOS prompt. The basic format for this program is as follows:

logman VERB <collection_name> [options]

- The **VERB** is the action the LOGMAN is asked to perform. Four actions are discussed in this paper.

  create counter - Create a counter collection.
  start - Start an existing collection.
  stop - Stop an existing collection.
  delete - Delete an existing collection.

- The **<collection_name>** is a user-supplied designation for the collection of performance measurements. Our process will use MY_PERF throughout as the name of collection.

- The **[options]** used by LOGMAN are numerous. We will only be using a very few in our discussion.

  -c Performance counters to collect.
  -o Path of the output log file.
  -si Sample interval for performance counters.
  -f Specifies the log format for the collection.
  -v Version control. nnnnnn | mmddhhmm
  -max Max size a log can become in megabytes.

CREATE LOGMAN COLLECTION

The collection of measurements is created using the `LogParms` macro variable. First check that the collection does not exist by executing the delete command to the operating system and then create and start the LOGMAN counter. Once started, our SAS program is included and when complete we stop our LOGMAN and `SasWork.vbs` and processes. (Figure 8: Create LOGMAN Collection)

IMPORT PERFORMANCE STATISTICS

Import the text file created during our LOGMAN execution. This file contains the performance statistics identified in the LogParms macro variable. Adding a new counter to LogParms will need to also be added to the input statement below in the same sequence in which it was passed to the performance monitor. (Figure 9: Import Performance Statistics)

Notice the conversion of the `Page_File, Disk_IO, CPU,` and `Memory` variables into percentages to be used in the subsequent PROC PLOT. The calculations for this have been tested on various computers, though should tested on the current operating system to ensure accuracy. Also notice the creation of the "`_Test`" variables that indicate whether all seems OK or if whether it looks BAD. This is yet another area that should be evaluated for accuracy. Table 2: Performance Statistics shown below is an example of the data captured in this step.

Table 2: Performance Statistics

	Run_Date	Run_Time	Avail_Memory	Page_Per_Sec	Pg_Read_PSec	Page_Faults	Cache_Faults	Page_File	Disk_IO	CPU	Memory	Time_Interval	Paging_Test	Memory_Test	DiskIO_Test	CPU_Test
1	01/17/2005	4:38:50.6 PM	66,215,936	212	22	1,011	833	16.52%	100.0%	0.64%	75.24%	1	OK	OK	BAD	OK
2	01/17/2005	4:38:53.6 PM	60,760,064	307	23	1,249	827	16.82%	100.0%	5.05%	77.28%	2	OK	OK	BAD	OK
3	01/17/2005	4:38:56.6 PM	53,882,880	244	22	1,045	819	16.91%	98.96%	3.47%	79.85%	3	OK	OK	BAD	OK
4	01/17/2005	4:38:59.6 PM	46,243,840	212	22	976	795	16.91%	100.0%	1.33%	82.71%	4	OK	OK	BAD	OK
5	01/17/2005	4:39:02.6 PM	38,891,520	212	22	998	821	16.91%	100.0%	1.50%	85.46%	5	OK	OK	BAD	OK
6	01/17/2005	4:39:05.6 PM	28,979,200	226	30	1,296	992	16.90%	99.48%	8.51%	89.16%	6	OK	OK	BAD	OK
7	01/17/2005	4:39:08.6 PM	22,175,744	206	23	992	809	16.90%	100.0%	5.47%	91.71%	7	BAD	BAD	BAD	OK
8	01/17/2005	4:39:11.6 PM	19,132,416	136	35	2,350	1,974	17.18%	9.84%	35.74%	92.84%	8	BAD	BAD	OK	OK

IMPORT SASWORK INFORMATION

Note that the SAS WORK termination script and this subsequent import are performed after the LOGMAN portion of the process has been stopped. The data will be later plotted using SAS/GRAPH™ that will identify the amount of SAS WORK space used during our processing at each time interval specified in the `TestInt` macro variable. (Figure 10: Import SAS WORK Data)

CREATE PERFORMANCE REPORTS

The data are then used to create performance reports stored in web pages. Using the Output Delivery System (ODS) mark-up text and Portable Network Graphic (PNG) files are created. A myriad of output formats can be output using ODS such as Rich Text Files (RTF), Portable Document Format (PDF), and in our case Hyper Text Mark-up Language files (HTML), among others. The output created from this data is limited only by the imagination. (Figure 12: Create Performance Report)

CONCLUSION

The goal of this process, which is to programmatically capture performance statistics about a SAS routine, seems to be a viable one as long as all of the appropriate pieces are in place. Windows XP Professional or greater must be running on the computer and access to run the Performance Monitor on the workstation must be granted. The inability to execute the LOGMAN due to permissions issues may be circumvented by capturing the performance data via the WMI functionality. The disadvantage to using the objects within WMI is that Performance Monitor counters are calculated values as described in Table 1: Table of Counters. The correct objects would need to be identified and personalized counters created in order to mimic that which occurs in the LOGMAN process. Another possible option is the MODULE family of SAS CALL routines that invoke functions within Dynamic Link Libraries (DLL) on the computer. There are many more ways to get results using the SAS System.

REFERENCES

Susan Davis, Carl Ralston (2001). "Windows NT Server Configuration and Tuning for Optimal Server Performance" *Proceedings of the Twenty-sixth Annual SAS® Users Group International Conference,* 277-26.

Gary Mehler, (2000). "Taking Advantage of the SAS System on the Windows Platform"
" *Proceedings of the Twenty-fifth Annual SAS® Users Group International Conference,* 280-25.

Patrick Fay, Tracy Warren Carver (2003). "SAS® Performance Optimizations on Intel Architecture" *Proceedings of the Twenty-eigth Annual SAS® Users Group International Conference,* 286-28.

Tony Brown (2001). "A Practical Approach to Solving Performance Problems with the SAS System" Customer Technology Center. October 17, 2001. <http://support.sas.com/rnd/scalability/papers/solve_perf.pdf> (December 16, 2004).

Walker, Jamie. (1997) "Tuning Your NT Server for the SAS System", SAS Institute Inc. <http://www.sas.com/partners/enterprise/msinfo/tuning.html> (December 16, 2004).

Casey Thompson. (2002) "PC Performance and the SAS System" Technical Support TS-684. <http://support.sas.com/techsup/technote/ts684/ts684.html> (December 16, 2004).

Demand Technology Software. "Frequently Asked Questions about Memory" Demand Technology Online. September 09, 2004. <http://www.demandtech.com/FAQSmem.htm> (December 16, 2004).

Microsoft Corporation (2000). "Active Directory Client Extensions for Windows 95/98/NT" Windows Market Bulletins. February 2000, <http://www.microsoft.com/windows2000/server/evaluation/news/bulletins/adextension.asp> (January 1, 2005).

Microsoft Corporation (2004). "Windows Management Instrumentation \ Run-Time Requirements" MSDN Home, MSDN Library, Win32 and COM Development, Windows Management Instrumentation (WMI). October 2004. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/wmi_start_page.asp> (January 1, 2005).

Microsoft Corporation (2005). "The Portable Script Center" TechNet Script Center Sample Scripts. January 4, 2005 <http://www.microsoft.com/downloads/details.aspx?FamilyID=b4cb2678-dafb-4e30-b2da-b8814fe2da5a&displaylang=en> (January 15, 2005).

ACKNOWLEDGMENTS

The authors wish to thank the following IT staff at Computer Sciences Corporation, Robert Cho, Gilbert Goh, and Rex Quedit, our SAS friends, Fran Alfano, Brian Williams, and crew from CIGNA; as well as the SAS Help Desk for their tireless efforts to answer seemingly endless questions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Rick Andrews
Centers for Medicare and Medicaid Services
Office of Research, Development, and Information
7500 Security Boulevard
Baltimore, MD 21244
Phone: (410) 786-4088
E-mail: randrews@cms.hhs.gov

Sherry Dixon, PhD
National Center for Healthy Housing
10227 Wincopin Circle, Suite 100
Columbia, MD 21044
Phone: (410) 772-2773
E-mail: sdixon@centerforhealthyhousing.org

```
*------------------------------------------------------------------------*;
* Program: perfmon.sas                                  SUGI 022-30      *;
* Purpose: Performance Monitoring for SAS(r) Programs on Windows(r) XP   *;
* Creator: Rick Andrews, Sherry Dixon                                    *;
* Created: 04/13/2005                                                    *;
* Remarks: This is the "Wrapper" program discussed in SAS User Group     *;
*          International (SUGI) paper 022-30.                            *;
*------------------------------------------------------------------------*;

options fullstimer missing=0 noxwait;
```

*Figure 2: PerfMon Parameters;

```
  *========================= P A R A M E T E R S ========================*;

  %let PgmLoc  = C:\SUGI30\Programs;   * Your Program Location *;
  %let TestPgm = test_using_set.sas;   * Your SAS Program      *;

  %let TestVer = 1;                    * Test Version          *;
  %let TestObs = 100;                  * Test Observations     *;
  %let TestInt = 00:03;                * Test Interval MM:SS   *;

  %let SasLoc  = C:\SUGI30\SasData;    * Test SasData Location *;
  %let OutLoc  = C:\SUGI30\Output;     * Test Output Location  *;
  %let SysLoc  = C:\SUGI30\SysInfo;    * System Info Location  *;

  * Ex. test_using_sql_1_1000 *;
  %let OutFile = %scan(&TestPgm,1,'.')_&TestVer._&TestObs;

  *========================= P A R A M E T E R S ========================*;

Libname PerfLib "&SasLoc";
```

* Figure 3: SysInfo Script Creation;

```
*-----------------------------------------*;
* Create VB Script to Capture System Info  *;
*-----------------------------------------*;
data _null_;
  file "&SysLoc\SysInfo.vbs";

  *** Create Pointer to Output File ***;

  put 'Const ForWriting = 2';         * 2=Overwrites text file *;
  put 'Set objFSO = CreateObject("Scripting.FileSystemObject")';
  put 'Set objFile = objFSO.OpenTextFile ("'"&SysLoc"'\SysInfo.txt",ForWriting,True)';

  *** Create Pointer to WMI Objects ***;

  put 'strComputer = "."';            * .=Identifies local computer *;
  put 'Set objWMIService = GetObject("winmgmts:" _ ';
  put '  & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")';

  *** Capture System Information    ***;

  %macro GetSysInfo(Short_Name, WMI_Provider, WMI_Object, Where_Clause);
    put 'Set getValue = objWMIService.ExecQuery _';
    put '   ( "Select * from ' "&WMI_Provider " "&Where_Clause" '"' ')';

    put 'For Each objData in getValue';
    put '    objFile.WriteLine( "' "&Short_Name" ' ~ " & objdata.' "&WMI_Object" ')';
    put 'Next';
  %mend GetSysInfo;

  %GetSysInfo(ComNam, Win32_ComputerSystem , Name                  , Where Name<>'' );
  %GetSysInfo(ComMod, Win32_ComputerSystem , Model                 , Where Name<>'' );
  %GetSysInfo(ComMan, Win32_ComputerSystem , Manufacturer          , Where Name<>'' );
  %GetSysInfo(PhMemT, Win32_ComputerSystem , TotalPhysicalMemory   , Where Name<>'' );
  %GetSysInfo(OprSys, Win32_OperatingSystem, Name                  , Where Name<>'' );
  %GetSysInfo(VrMemA, Win32_OperatingSystem, FreeVirtualMemory     , Where Name<>'' );
  %GetSysInfo(PhMemA, Win32_OperatingSystem, FreePhysicalMemory    , Where Name<>'' );
  %GetSysInfo(VrMemT, Win32_OperatingSystem, TotalVirtualMemorySize, Where Name<>'' );
  %GetSysInfo(CpuNam, Win32_Processor      , Name                  , Where Name<>'' );
  %GetSysInfo(DskCap, Win32_LogicalDisk    , Size                  , Where DriveType=3 );
  %GetSysInfo(DrvLtr, Win32_LogicalDisk    , DeviceID              , Where DriveType=3 );
  %GetSysInfo(FreSpc, Win32_LogicalDisk    , FreeSpace             , Where DriveType=3 );

  *** Close System Info Text File ***;
  put 'objFile.Close';
 run;

*** Execute "System Info" VB Script ***;          *Note the use of xsync toallow process to finish;
options xsync;
%sysexec cscript &SysLoc\SysInfo.vbs ;
```

* Figure 4: SasWork Script;

```
*-------------------------------------------*;
* Create VB Script to Capture SASWORK Space  *;
*-------------------------------------------*;
data _null_;
  file "&SysLoc\SasWork.vbs";

  *** Determine Time for Sleep      ***;

  min=input(scan("&testint",1,':'),8.);
  sec=input(scan("&testint",2,':'),8.);
  if min ne 0 then min = min * 60;
  TotSec=(min+sec);
  call symput('TimeInt',TotSec);
  TotSec = TotSec * 1000;

  *** Create Pointer to Output File ***;

  put 'Const ForWriting = 2';
  put 'Set objFSO = CreateObject("Scripting.FileSystemObject")';
  put 'Set objFile = objFSO.OpenTextFile _ ';
  put '   ("'"&SysLoc"'\'"&OutFile._SasWork.txt"'", ForWriting, True)';

  *** Capture Sas Work Space        ***;

  SasWorkLoc = getoption("WORK");
  put 'Set objFolder = objFSO.GetFolder("' SasWorkLoc '")';
  put 'do';
  put '  objFile.WriteLine( "SASWORKSIZE: " & objFolder.Size )';
  put '  Wscript.Sleep ' TotSec;
  put 'loop';
run;

*** Execute SASWORK Script ***;
options noxsync;
%sysexec cmd /K cscript &SysLoc\SasWork.vbs;
```

* Figure 5: SasWork Script Termination;

```
*-----------------------------------------------------*;
* Create VB Script to Terminate SASWORK Space Script *;
*-----------------------------------------------------*;
data _null_;
  file "&SysLoc\SasWorkTerm.vbs";

  *** Create Pointer to WMI Objects ***;

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

  *** Capture SAS WORK Space        ***;

  put 'Set colProcessList = objWMIService.ExecQuery _';
  put '("Select * from Win32_Process Where CommandLine like '"'"'%cscript%SasWork.vbs%'"'"'")';

  put 'For Each objProcess in colProcessList';
  put '    objProcess.Terminate()';
  put 'Next';
run;
```

*Figure 6: Import SysInfo Data;

```
*-----------------------------------------------------*;
* Import System Information                           *;
*-----------------------------------------------------*;
data PerfLib.&OutFile._SysInfo (drop=TmpVar);
  length SysInfoNam $10. SysInfoVal $100.;
  infile "&SysLoc\SysInfo.txt" truncover;
  input TmpVar $100.;

  SysInfoNam = scan(TmpVar,1,'~');
  SysInfoVal = scan(left(
scan(TmpVar,2,'~')),1,'|');

  if SysInfoNam = 'PhMemT' then
    do;
      SysInfoNum = input(SysInfoVal,24.);
      SysInfoVal = trim(left(round((SysInfoVal / 1000000),1))) || ' MB';
      call symput('TotalPhyMemN',left(SysInfoNum));
    end;

  if SysInfoNam = 'DskCap'
  or SysInfoNam = 'FreSpc'
  then SysInfoVal = trim(left(round((SysInfoVal / 1000000),1))) || ' MB';
```

```
  if SysInfoNam = 'PhMemA'
  or SysInfoNam = 'VrMemA'
  or SysInfoNam = 'VrMemT'
  then SysInfoVal = trim(left(round((SysInfoVal / 1000),1))) || ' MB';

  if SysInfoNam = 'ComNam' then SortOrder = 1;
  if SysInfoNam = 'OprSys' then SortOrder = 2;
  if SysInfoNam = 'ComMan' then SortOrder = 3;
  if SysInfoNam = 'ComMod' then SortOrder = 4;
  if SysInfoNam = 'CpuNam' then SortOrder = 5;
  if SysInfoNam = 'PhMemT' then SortOrder = 6;
  if SysInfoNam = 'PhMemA' then SortOrder = 7;
  if SysInfoNam = 'VrMemT' then SortOrder = 8;
  if SysInfoNam = 'VrMemA' then SortOrder = 9;
  if SysInfoNam = 'DrvLtr' then SortOrder = 10;
  if SysInfoNam = 'DskCap' then SortOrder = 11;
  if SysInfoNam = 'FreSpc' then SortOrder = 12;
run;

proc sort data=PerfLib.&OutFile._SysInfo;
  by SortOrder;
run;
```

**Figure 7: Capture SAS OPTIONS;

```
*----------------------------------------*;
* Add SAS Option Info to SysInfo Table    *;
*----------------------------------------*;
data PerfLib.&OutFile._SysInfo;
  length SasInfoNam SasInfoVal $25.;
  set PerfLib.&OutFile._SysInfo;

  if _n_=1 then do; SasInfoNam="MEMSIZE";  SasInfoVal=getoption("MEMSIZE");  end;
  if _n_=2 then do; SasInfoNam="SORTSIZE"; SasInfoVal=getoption("SORTSIZE"); end;
  if _n_=3 then do; SasInfoNam="BUFSIZE";  SasInfoVal=getoption("BUFSIZE");  end;
  if _n_=4 then do; SasInfoNam="BUFNO";    SasInfoVal=getoption("BUFNO");    end;

  call symput('SasWorkLib',getoption("WORK"));
run;
```

*Figure 8: Create LOGMAN Collection;

```
*----------------------------------------*;
* Create LOGMAN Collection                *;
*----------------------------------------*;
%let LogParms = ;
%let LogParms = &LogParms -c ;
%let LogParms = &LogParms %str( "\Memory\Available Bytes");
%let LogParms = &LogParms %str( "\Memory\Pages/sec");
%let LogParms = &LogParms %str( "\Memory\Page Reads/sec");
%let LogParms = &LogParms %str( "\Memory\Page Faults/sec");
%let LogParms = &LogParms %str( "\Memory\Cache Faults/sec");
%let LogParms = &LogParms %str( "\Paging File(\??\C:\pagefile.sys)\% Usage");
%let LogParms = &LogParms %str( "\Processor(0)\% Processor Time");
%let LogParms = &LogParms %str( "\PhysicalDisk(_Total)\% Disk Time");
%let LogParms = &LogParms -o ;
%let LogParms = &LogParms %str( "&SysLoc\&OutFile..csv");
%let LogParms = &LogParms --v;
%let LogParms = &LogParms -max 100;
%let LogParms = &LogParms -si &TestInt;
%let LogParms = &LogParms -f csv;

%put &LogParms;

*** Execute LOGMAN Collection ***;
%sysexec logman delete MY_PERF;
%sysexec logman create counter MY_PERF &LogParms;
%sysexec logman start  MY_PERF;

*** SAS Process to be Tested  ***;
%include "&PgmLoc\&TestPgm";

*** Stop LOGMAN Collection    ***;
%sysexec logman stop MY_PERF;
%sysexec logman delete MY_PERF;

*** Terminate SAS WORK Script ***;
options xsync;
%sysexec cscript &SysLoc\SasWorkTerm.vbs;
```

*Figure 9: Import Performance Statistics;

```
*----------------------------------------*;
* Import Performance Statistics          *;
*----------------------------------------*;
data PerfLib.&OutFile;

  infile "&SysLoc\&OutFile..csv" dlm=','
         missover dsd lrecl=32767 firstobs=3;

    format Run_Date     mmddyy10.;      informat Run_Date     5.;
    format Run_Time     timeampm12.2;   informat Run_Time     5.;
    format Date_Time    $25.;           informat Date_Time    $25.;
    format Avail_Memory comma12.;       informat Avail_Memory 8.;
    format Page_Per_Sec comma8.;        informat Page_Per_Sec 3.;
    format Pg_Read_PSec comma8.;        informat Pg_Read_PSec 3.;
    format Page_Faults  comma8.;        informat Page_Faults  8.;
    format Cache_Faults comma8.;        informat Cache_Faults 8.;
    format Page_File    percent8.2;     informat Page_File    3.;
    format Disk_IO      percent8.2;     informat Disk_IO      3.;
    format CPU          percent8.2;     informat CPU          3.;
    format Memory       percent8.2;     informat Memory       3.;

  input Date_Time $     Avail_Memory
        Page_Per_Sec    Pg_Read_PSec
        Page_Faults     Cache_Faults
        Page_File       Disk_IO
        CPU       ;

  Run_Date   = input(substr(Date_Time,1,10),mmddyy10.);
  Run_Time   = input(substr(Date_Time,12,12),time12.);
  Page_File  = Page_File / 100;
  Disk_IO    = Disk_IO   / 100;
  CPU        = CPU       / 1000;
  Memory     = (&TotalPhyMemN - Avail_Memory) / &TotalPhyMemN;

  Time_Interval + 1;
  call symput('TotTimeInt',Time_Interval);

  if  Memory       > .9
  and Pg_Read_PSec > 10
  and Page_Faults  > Cache_Faults
                   then Paging_Test = 'BAD'; else Paging_Test = 'OK ';
  if Memory   > .9  then Memory_Test = 'BAD'; else Memory_Test = 'OK ';
  if Disk_IO  > .85 then DiskIO_Test = 'BAD'; else DiskIO_Test = 'OK ';
  if CPU      > .9  then CPU_Test    = 'BAD'; else CPU_Test    = 'OK ';

  drop Date_Time;
run;
```

*Figure 10: Import SAS WORK Data;

```
*----------------------------------------*;
* Import SAS WORK Data                   *;
*----------------------------------------*;
data PerfLib.&OutFile._SasWork;
  infile "&SysLoc\&OutFile._SasWork.txt" dlm=','
         missover dsd lrecl=32767 firstobs=1;
  format SasWork comma12.;
  informat SasWork 8.;
  input @14 SasWork;
  Time_Interval + 1;
  if Time_Interval > &TotTimeInt then delete;
run;
```

*Figure 11: Format Values;

```
*----------------------------------------*;
* Create Format for Report               *;
*----------------------------------------*;
proc format;
  value $SYSINFO 'ComNam'='Computer Name'
                 'OprSys'='Operating System'
                 'ComMan'='Manufacturer'
                 'ComMod'='Model'
                 'CpuNam'='Processor Name'
                 'PhMemT'='Total Physical Memory'
                 'PhMemA'='Available Physical Memory'
                 'VrMemT'='Total Virtual Memory'
                 'VrMemA'='Available Virtual Memory'
                 'DrvLtr'='Drive Letter'
                 'DskCap'='Disk Capacity'
                 'FreSpc'='Free Disk Space';
run;
```

```
*----------------------------------------*;
* Set up SAS and ODS Options             *;
*----------------------------------------*;
options orientation=landscape;

goptions reset=all ftext="Arial" htext=15pt
device=png ymax=5in xmax=7in;

ods listing close;
ods html body="&OutFile..html"
         path="&OutLoc"
         style=sasdocPrinter;

*----------------------------------------*;
* Create System Info Report              *;
*----------------------------------------*;
title1 "&OutFile";
proc report
  data=PerfLib.&OutFile._SysInfo
    nowd headline;
  column SortOrder SysInfoNam SysInfoVal
         SasInfoNam SasInfoVal;
  define SortOrder / order noprint "";
  define SysInfoNam / display format=$sysinfo."
  define SysInfoVal / display "System Info";
  define SasInfoNam / display "";
  define SasInfoVal / display "";
run;

*----------------------------------------*;
* Create Performane Measurement Plot     *;
*----------------------------------------*;
symbol1 interpol=join line=1  color=black;
symbol2 interpol=join line=2  color=red;
symbol3 interpol=join line=46 color=blue;
symbol4 interpol=join line=26 color=green;
legend frame label=none;
axis1 label=('Percent') order=(0 to 1 by .1);
title; footnote;

title1 j=l "Performance Measures";
proc gplot data=PerfLib.&OutFile;
  plot (Disk_IO CPU Page_File
        Memory)*Time_Interval
       / overlay
         vaxis=axis1
         lvref=2 legend=legend;
run;quit;

*----------------------------------------*;
* Create Sas Work Space Plot             *;
*----------------------------------------*;
symbol1 interpol=join line=1  color=black;
legend frame label=none;
axis1 label=('Bytes'); * order=(0 to 1 by .1);
title; footnote;

title1 j=l "SASWORK";
proc gplot data=PerfLib.&OutFile._SasWork;
  plot SasWork*Time_Interval
       / overlay
         vaxis=axis1
         lvref=2 legend=legend;
run;quit;

ods html close;
ods listing;
```

*Figure 12: Create Performance Report;

Test_Using_SQL_1_1000.HTML

System Info

Computer Name	5TPD941	MEMSIZE	0
Operating System	Microsoft Windows XP Professional	SORTSIZE	2097152
Manufacturer	Dell Computer Corporation	BUFSIZE	0
Model	Inspiron 8600	BUFNO	1
Processor Name	Intel(R) Pentium(R) M processor 1600MHz		
Total Physical Memory	536 MB		
Available Physical Memory	276 MB		
Total Virtual Memory	1804 MB		
Available Virtual Memory	1350 MB		
Drive Letter	C		
Disk Capacity	60012 MB		
Free Disk Space	50364 MB