Paper 226-31

Measure the Performance of Your SAS/IntrNet Application Programs

Michael A. Raithel, Westat, Rockville MD

Abstract

When you are developing SAS/IntrNet applications, the question of performance naturally arises. How will the various programs that comprise your SAS/IntrNet application perform when initiated by users over the Internet? Will the response times from executing assorted SAS programs online be acceptable to users? Are there particular programs in your application that will provide unacceptable response times in an Internet environment? Web users are used to near-instantaneous response times and will not tolerate long waits. They may believe that your application is broken or unusable, and abandon it if they have to wait for long periods of time to get a report, chart, or graph that they have requested.

This paper presents a new methodology for measuring SAS/IntrNet application program response time. It uses SAS programs that read the SAS Application Server logs to measure the true response time of SAS/IntrNet programs. This information is stored in SAS data sets so that it can be available for systematic review over time. You can use this new methodology to measure and report the response time for SAS/IntrNet application programs in your organization. By doing so, you can tune your SAS/IntrNet programs so that your users achieve better turnaround on their requests for data over the web. This effort will strengthen their confidence in, and use of, your SAS/IntrNet applications.

Introduction

The typical SAS/IntrNet application is composed of four components. These components all work together to translate a request for information coming over the web, schedule a SAS program for execution, process the data, and the return a result set to the user over the web. It is important to have a basic understanding of these components in order to understand how we may measure response time. **Figure 1, SAS/IntrNet components**, provides a schematic overview of the four basic components.

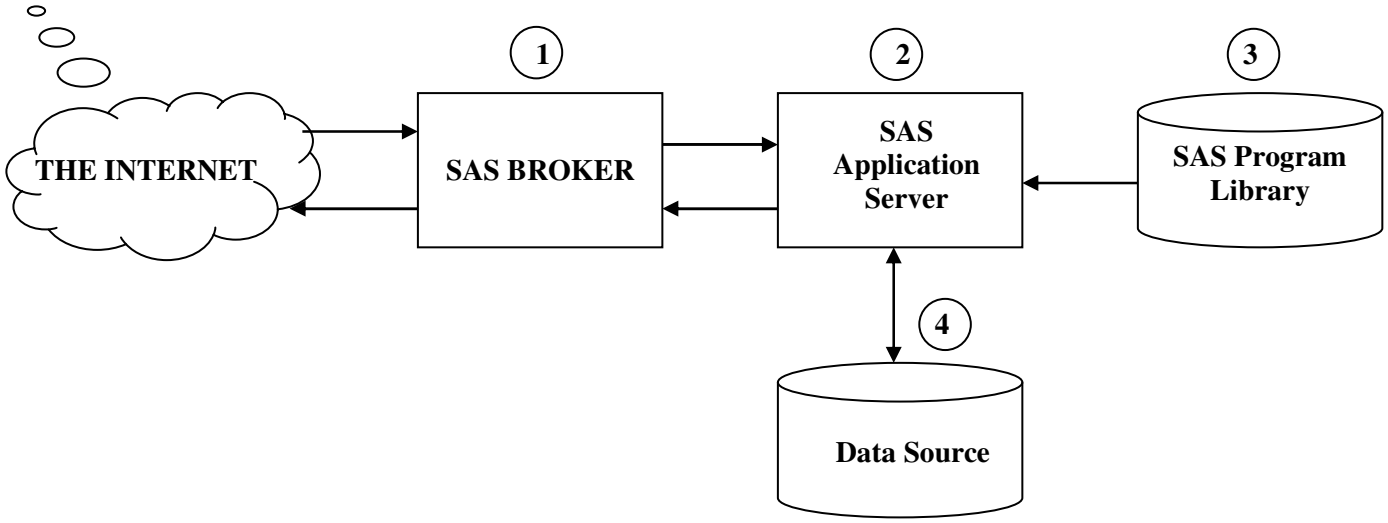

Figure 1. SAS/IntrNet components.

Here is a description of the four SAS/IntrNet components:

1. **The SAS Broker** – The SAS Broker is a CGI script that is placed on a web server in the *CGI-Bin* subdirectory of the *Inetpub* IIS directory. Its function is to accept incoming requests from the Internet, determine which SAS Application Server should receive the requests, and send the requests to that SAS Application Server. A *request* typically contains:
   - The name of the SAS Application Server that is to receive the request
   - The name of the SAS program that is to be executed by the SAS Application Server
   - Name/value pairs that are treated as SAS Macro variable names and values when input to the SAS program that is executed by the SAS Application Server.

2. **The SAS Application Server** – The SAS Application Server is a continuously running SAS session that waits for work to be sent to it by the SAS Broker. It receives a request from the SAS Broker to schedule a specific SAS program for execution. The SAS Application Server retrieves the specified program from the SAS Program Library, executes the program, cleans up after the program has completed, and then waits for the next request from the SAS Broker.

3. **SAS Program Library** – The SAS Program Library contains all of the SAS programs that can be executed by a particular SAS Application Server. Programs are generally configured to accept the name/value pairs (passed over the web to the SAS Broker and SAS Application Server) as SAS Macro variable names and their current values. The SAS Macro variable values are typically used to help the program make decisions about the type and quantity of data that must be processed. (For example, they may be used in WHERE statements to subset data.) The SAS programs usually process data, create reports, and then send the output back to the SAS Broker, which streams it back to the user.

4. **Data Sources** – The SAS program running in the SAS Application Server can access data stored in SAS data sets and flat files. It can also access data stored in database management systems if you have the correct SAS/Access product.

Though users may not be aware of it, the response time of a SAS/IntrNet transaction is actually composed of six events. Those events are numbered in **Figure 2, SAS/IntrNet response time events**.

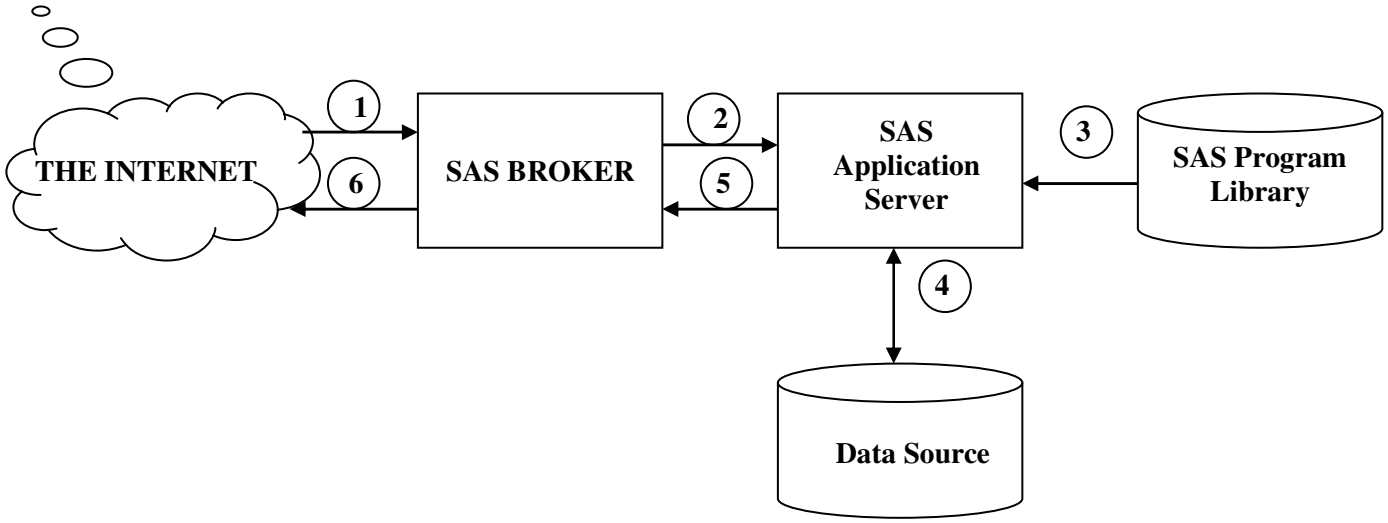

Figure 2. SAS/IntrNet response time events.

The events are:

1. The time it takes for the request to get from the user's web browser to the SAS Broker from over the Internet.
2. The time it takes for the request to travel between the SAS Broker and SAS Application Server
3. The time it takes for the SAS Application Server to schedule a SAS program
4. *The time it takes for the SAS application program to execute*
5. The time it takes to send the result set from the SAS Application Server to the SAS Broker
6. The time it takes to send the result set from the SAS Broker to the user's web browser over the Internet

The only response time event that you can measure via SAS programs is Event #4, *The time it takes for the SAS application program to execute*. That event is the focus of this paper. By understanding this particular response time event, you can tune your SAS programs for their best performance.

You do not have much, if any, control over the other events that make up the response time of a SAS/IntrNet transaction. Times #1 and #6, which have to do with transmission over the Internet, are variable and can be measured with various commercial software packages. Times #2 and #5 are often negligible, especially in cases where the SAS Broker is on the same server as the SAS Application Server. Time #3 can vary, depending upon the number of simultaneous requests and the configuration of the SAS Application Server.

SAS/IntrNet Logs

The key to determining SAS/IntrNet application program response time is harvesting information found in SAS/IntrNet logs. Each SAS Application Server has its own *logs* directory, in which seven log files are kept—one for each day of the week. All SAS Application Server transactions processed on Monday are stored in the *Mon_* log, those for Tuesday in the *Tue_* log, those for Wednesday in the *Wed_* log, and so forth. Each night, just after midnight, the SAS Application Server switches to writing transaction information to the log for the appropriate new day of the week. After seven days, the SAS Application Server writes over the seven-day-old log file (for the appropriate new day of the week), destroying the transaction information that was previously written to it.

Figure 3, A typical SAS Application Server *logs* directory on a Windows server, shows the contents of a typical SAS Application Server *logs* directory.

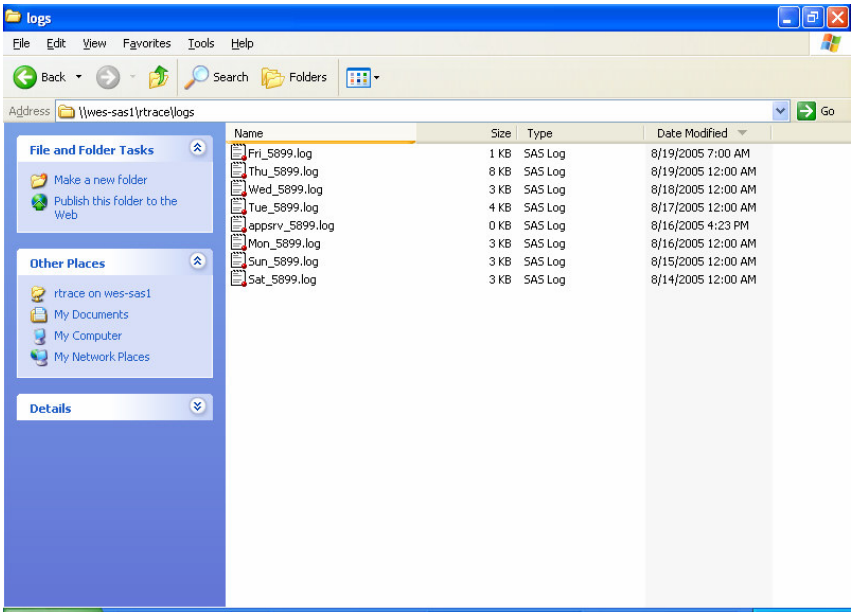

Figure 3 – A typical SAS Application Server *logs* directory on a Windows server.

The log files contain information about the processing of SAS programs within the SAS Application Server. This includes housekeeping events, program start times, the values of SAS/IntrNet-related macro variables, and program end times. **Figure 4**, below, presents some snippets from a typical SAS Application Server log file:

```
Mon, 27 Jun 2005 07:24:15.029 Request 8835 from 10.04.27.55 started.
Mon, 27 Jun 2005 07:24:15.091 Request 8835 Program is marprod.program1.sas
============= Request 8835 Symbols ===============
.     .     .     .     .     .     .     .     .
.     .     .     .     .     .     .     .     .
Mon, 27 Jun 2005 07:24:15.121 Request 8835 ended okay (0.09 seconds)
============= Request 8835 Log ===================
.     .     .     .     .     .     .     .     .
.     .     .     .     .     .     .     .     .
NOTE: request has completed
============= Request 8835 End ===================
Mon, 27 Jun 2005 07:27:01.854 Request 8836 from 10.04.27.55 started.
Mon, 27 Jun 2005 07:27:01.900 Request 8836 Program is marprod.program2.sas
============= Request 8836 Symbols ===============
```

Figure 4 – Snippets from a SAS/IntrNet log.

You can see from **Figure 4** that the beginning and ending times of individual SAS programs executed by the SAS Application Server are entered into the log. For example, *program1.sas* from the *marprod* SAS program library began executing at 7:24:15.029 on June 27th, 2005. It was request number 8835 from IP address 10.04.27.55. It took .09 seconds to execute. The same type of information for each program that was executed by this SAS Application Server on this particular day can be found in this log file.

By crafting a SAS program that reads through the log files and stores the information in a permanent SAS data set, you can keep track of which SAS programs your users are executing and the SAS program response times that they experience.

Getting SAS Program Execution Information From a SAS/IntrNet Log

Once you decide to measure the performance of your SAS/IntrNet applications, you need to put a program in place to do so. **Appendix A** contains the **Process_App_Server_Logs.sas** program which processes *yesterday's* SAS/IntrNet log file for a specified SAS Application Server. Here is the overall execution strategy for that program on a Windows SAS/IntrNet server:

1. Every morning, just after midnight, **Process_App_Server_Logs.sas** is executed via a .bat file. That .bat file is scheduled via the Windows server's Task Manager.

2. The **Process_App_Server_Logs.sas** program reads *yesterday's* SAS/IntrNet log file and creates observations that are stored in a permanent SAS data set.

3. Reporting of SAS/IntrNet program response times is done by ad-hoc (or scheduled) programs that access the permanent SAS/IntrNet program performance data set.

The execution strategy, above, can just as easily be implemented for SAS/IntrNet applications running on Unix, Linux, or z/OS servers.

Here is an overview of the **Process_App_Server_Logs.sas** program:

- The **Process_App_Server_Logs.sas** program is a SAS macro that accepts three parameters:
    - **LOGDATA** – This is the full path name of the directory in which the SAS/IntrNet logs for a particular Application Server can be found.
    - **SASFILE** – Is the full path name of the directory in which the permanent SAS/IntrNet program performance data set can be found. This is used in a LIBNAME statement to allocate the SAS/IntrNet program performance data set in update mode.
    - **APPSERVR** – The APPSERVR port number is a part of the SAS/IntrNet log name. So, the port number must be specified.
- The **Process_App_Server_Logs.sas** program starts off by building several formats ($DAYNUM, NUMDAY, and $MONTHNO) that are used in the program.
- Then, **Process_App_Server_Logs.sas** determines which day was *yesterday* in a DATA _NULL_ step. It uses that information and the APPSERVR macro variable to create the value of the LOGNAME macro variable. The LOGNAME macro variable contains the specific name of the SAS/IntrNet log file that was in effect yesterday for the given SAS Application Server.
- The next step is to allocate:
    - The SAS/IntrNet Application Server log file for *yesterday* via a FILENAME statement
    - The permanent SAS/IntrNet program performance data set via a LIBNAME statement
    - A temporary flat file that is used to temporarily store SAS/IntrNet log records during processing via a FILENAME statement.

- A DATA _NULL_ step parses the SAS/IntrNet log file looking for lines with specific keywords in them. When those lines are found, they are written to the temporary file (TEMPFILE) so that they can be processed in the next DATA step. This is done in two steps because:

  1. Only a few of the many thousands of lines in the SAS/IntrNet log file are of interest.

  2. Two or three individual lines in the log file give the complete picture of how a particular SAS/IntrNet program executed. They must be processed in sequence to build the complete observation that is stored in the SAS/IntrNet program performance data set. For example, in **Figure 4 – Snippets from a SAS/IntrNet log**, three separate lines provide all of the information needed for request 8835. All three of those lines are placed into the temporary file so that they may be further processed by the main data step.

     Figure 5 – The contents of the TEMPFILE temporary file, below, shows what the contents of TEMPFILE might look like.

```
Mon, 27 Jun 2005 07:33:07.017 Request 8840 Program is prodapp1.autocoll_2a.sas
Mon, 27 Jun 2005 07:33:07.033 Request 8840 ended okay (0.08 seconds)
Mon, 27 Jun 2005 09:09:17.490 Request 8841 from 10.13.9.30 started.
Mon, 27 Jun 2005 09:09:17.538 Request 8841 Program is prodapp1.n_allds.sas
Mon, 27 Jun 2005 09:09:18.194 Request 8841 ended okay (0.70 seconds)
Mon, 27 Jun 2005 09:35:00.398 Request 8842 from 10.13.9.30 started.
Mon, 27 Jun 2005 09:35:00.446 Request 8842 Program is prodapp1.n_allfreq.sas
Mon, 27 Jun 2005 09:35:00.946 Request 8842 ended okay (0.55 seconds)
.        .          .          .    .    .    .    .    .
.        .          .          .    .    .    .    .    .
```

     Figure 5 – The contents of the TEMPFILE temporary file.

- A DATA step parses the temporary file (TEMPFILE), turning the selected lines of SAS/IntrNet request information into observations. Here are the highlights of the DATA step:

  1. The DATA step reads a record and determines which type of record it is. Normally, there are three records for each program request (see **Figure 4 – Snippets from a SAS/IntrNet log**):

     - One stating that the request has started
     - One stating the name of the program to be executed
     - One stating that the request has ended

     The DATA step keeps track of how many records it has received for an individual request via the FLAG variable.

  2. When the DATA step reads a *request started* record, it collects the start date/time (STARTDTTM), the request number (REQUEST), and the IP address (IPADDRESS).

  3. When the DATA step reads a *name of the program* record, it collects the program name (PROGRAM). Occasionally, this line may not be present in a log. If it is not present, the DATA step detects this and sets PROGRAM equal to "*** Missing Program Name ***".

4. When the DATA step reads a *request has ended* record, it collects the end date/time (ENDDTTM) and the elapsed time (ELAPSED).

5. Occasionally, a request will have a third line noting that it "ended with errors". If that is found, then ERRORFLAG is set to one.

6. Sometimes, SAS/IntrNet will "kill" a non-responding program and flush it from the Application Server. When it does so, it puts a "Killing program … " message in the log. The DATA step catches these messages when they occur and sets KILLFLAG equal to one.

The final result of this data step is the LOGDATA SAS data set containing one observation for each SAS/IntrNet program that was executed by the specified SAS Application Server on a particular day.

- The LOGDATA SAS data set is sorted by STARTDTTM, APPSERVR, REQUEST, and PROGRAM. Then it is used to update the permanent SAS/IntrNet program performance data set in a MODIFY statement.

The **Process_App_Server_Logs.sas** program is the key to creating and populating the SAS/IntrNet program performance data set.

The SAS/IntrNet Program Performance Data Set

A CONTENTS Procedure listing of the SAS/IntrNet Program Performance data set can be found in **Appendix B**. There are only ten items of interest in the SAS/IntrNet program logs. Those metrics are:

- **APPSERVR** – This is the port number of the SAS Application Server that executed the program.

- **IPADDRESS** – This is the end-user's IP address.

- **ELAPSED –** This is the total elapsed time of the SAS program.

- **ENDDTTM –** This is the end date/time of the executing SAS program

- **ERRORFLAG –** The ERRORFLAG is set to one (1) when the SAS Application Server indicates that the program had an error in the log.

- **KILLFLAG –** On occasion, the SAS Application Server will *kill* (halt) a SAS program that it knows is having a problem. When this happens, it makes a note that the program is being killed in the log. When a program is killed, the KILLFLAG is set to 1.

- **PROGRAM –** This is the name of the SAS program[1] that was executed by SAS/IntrNet.

- **REQUEST –** Every program executed by SAS/IntrNet is given a unique request number. This helps SAS/IntrNet to keep track of the various programs. The request number of a program is saved in this variable.

[1] In the Process_App_Server_Logs.sas program in Appendix A, the length of the PROGRAM variable is 30 characters. That was suitable for the programs used in the author's organization. However, it is a potential "gotcha" if SAS/IntrNet programs have names longer than 30 characters in your own organization. You may have to adjust the size of this variable.

- **STARTDTTM –** This is the start date/time of the SAS program.

It is easy to see how the data items above can be used to report on the performance of your SAS/IntrNet application. You can determine which programs your users are actually invoking via your web application. You can see whether those programs are executing without error, and what the response times are for the programs. Based on this information, you may choose to eliminate unused programs, fix ones that consistently have errors, and tune long-running programs. And, you can research how your SAS programs are performing over various time periods. Consequently, the items in the SAS/IntrNet program performance data set provide an invaluable look into what is going on with your SAS/IntrNet web application.

Reporting SAS/IntrNet Program Response Time

Once you have collected performance data from your SAS/IntrNet applications, you can create reports that characterize the performance of individual SAS programs or the performance of the entire web application. You can do this for a specific hour, day, or look at the performance over a long period of time. Because the information is automatically collected, and because it is stored in SAS data sets, you can bring all of your SAS programming skills to bear when creating reports.

Appendix C contains the **Analyze_App_Server_Logs.sas** program. That program creates several reports that characterize the performance of SAS/IntrNet programs. You can use it as a springboard for creating your own programs. Here are some example reports created by **Analyze_App_Server_Logs.sas:**

Production SAS/IntrNet Report Application
Frequency of SAS/IntrNet Programs That Were Executed
on November 8, 2005

The FREQ Procedure

Program Name

progname	Frequency	Percent	Cumulative Frequency	Cumulative Percent
allcont.sas	12	11.88	12	11.88
alldiy.sas	4	3.96	16	15.84
allds.sas	16	15.84	32	31.68
allfreq.sas	11	10.89	43	42.57
allmeans.sas	2	1.98	45	44.55
allmore.sas	7	6.93	52	51.49
allord.sas	6	5.94	58	57.43
allrec.sas	3	2.97	61	60.40
allsums.sas	9	8.91	70	69.31
alltabs.sas	16	15.84	86	85.15
autocoll_2a.sas	6	5.94	92	91.09
iknow.sas	8	7.92	100	99.01
iknow_txt.sas	1	0.99	101	100.00

Production SAS/IntrNet Report Application
Minimum, Average, and Maximum Response Times
For Programs Executed on November 8, 2005
On All Application Servers

Minimum Elapsed Time	Average Elapsed Time	Maximum Elapsed Time	Number of

Program Name	(Sec.)	(Sec.)	(Sec.)	Executions
allcont.sas	0.69	0.81	1.00	12
alldiy.sas	0.75	0.79	0.84	4
allds.sas	0.69	1.53	11.02	16
allfreq.sas	0.44	0.55	0.66	11
allmeans.sas	0.55	0.56	0.56	2
allmore.sas	0.72	1.36	2.23	7
allord.sas	0.63	0.81	1.13	6
allrec.sas	0.55	0.55	0.56	3
allsums.sas	0.97	4.65	33.34	9
alltabs.sas	0.56	4.78	56.72	16
autocoll_2a.sas	0.02	0.06	0.09	6
iknow.sas	0.53	0.66	0.72	8
iknow_txt.sas	0.73	0.73	0.73	1

In the second report, you can see that two programs—ALLSUMS.SAS and ALLTABS.SAS—warrant further inspection. At least one occurrence of ALLSUMS.SAS had an elapsed time of 33.34 seconds; while at least one execution of ALLTABS.SAS took 56.72 seconds. Half-a-minute and one-minute are long times for a user to wait in the world of the Internet. Perhaps these were anomalous events; perhaps they are signs that the programs need to be tuned.

Without having collected SAS/IntrNet log data information, we would never have known about the issue with the ALLSUMS.SAS and ALLTABS.SAS programs. That is the intrinsic value of the whole technique of harvesting SAS/IntrNet logs that was outlined in this paper. It gives you insights into SAS/IntrNet program performance.

Conclusions

You can measure the performance of your SAS/IntrNet application programs by processing SAS Application Server logs. The logs yield information about the time that it took for individual SAS programs to execute within SAS Application Servers. The collection of this information can be automated by scheduling the **Process_App_Server_Logs.sas** program to execute on a daily basis for each SAS Application Server *logs* directory. After storing this information in a SAS data set, you can examine the response times of your individual SAS/IntrNet programs. Doing so will allow you to drop unused programs, to discover which programs are most commonly executed, and to tune long-running programs. This effort should result in stronger SAS/IntrNet applications that are real assets to your users.

Disclaimer

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

References

SAS Institute Inc. 2004. **SAS OnlineDoc® 9.1.2**. Cary, NC: SAS Institute Inc.

Contact Information

Please feel free to contact me if you have any questions or comments about this paper. You may reach me at:

Michael A. Raithel
Westat
1650 Research Boulevard
Room RW4521
Rockville, Maryland 20850

michaelraithel@westat.com

Appendix A – Process_App_Server_Logs.sas

```
*******************************************************************************;
* Program:  Process_App_Server_Logs.sas                                        *;
*                                                                              *;
* Author:   Michael A. Raithel                                                 *;
*                                                                              *;
* Purpose:  This program reads SAS/IntrNet Application Server Logs, extracts   *;
*           performance data and stores it in a SAS data set.                  *;
*******************************************************************************;

*------------------------------------------------------------*;
* Options.                                                   *;
*------------------------------------------------------------*;
options bufno=10 cbufno=5 msglevel=I;

*------------------------------------------------------------*;
* Formats, Includes, etc.                                    *;
*------------------------------------------------------------*;
proc format;
     value $daynum
    'Sun' = 1
    'Mon' = 2
    'Tue' = 3
    'Wed' = 4
    'Thu' = 5
    'Fri' = 6
    'Sat' = 7
    ;
run;
proc format;
     value numday
    1 = 'Sun'
    2 = 'Mon'
    3 = 'Tue'
    4 = 'Wed'
    5 = 'Thu'
    6 = 'Fri'
    7 = 'Sat'
    ;
run;
proc format;
     value $monthno
    'Jan' = 1
    'Feb' = 2
    'Mar' = 3
    'Apr' = 4
    'May' = 5
    'Jun' = 6
    'Jul' = 7
    'Aug' = 8
    'Sep' = 9
    'Oct' = 10
    'Nov' = 11
    'Dec' = 12
    ;
run;
```

```
    /*************************************************************/
    /* Beginning of the APPLOGRD Macro. This Macro accepts three  */
    /* parameters:                                                */
    /*                                                            */
    /*                                                            */
    /* LOGDATA - Actual file name of the raw SAS Application      */
    /*           Server log.                                      */
    /*                                                            */
    /* SASFILE - The SAS data library where this data will be     */
    /*           saved.                                           */
    /*                                                            */
    /* APPSERVR - The port number of the SAS Application Server.  */
    /*                                                            */
    /*************************************************************/

%MACRO APPLOGRD(LOGDATA,SASFILE,APPSERVR);
************************************************************;
* Determine "yesterday" and create a macro variable of the  *;
* day name. This will be used to allocate the "yesterdays"  *;
* Application Server Log.                                   *;
************************************************************;
data _null_;

length dayabrev $3;

log = '.log';

date = today()-1;
x    = weekday(date);

dayabrev = put(x,numday.);

logname = '\' || trim(left(dayabrev)) || '_' || trim(left(&APPSERVR)) || log;

call symput('LOGNAME',trim(left(logname)));

run;

************************************************************;
* Allocate the Aplication Server log.                       *;
************************************************************;
filename logdata "&LOGDATA.&LOGNAME";

************************************************************;
* Allocate the SASFILE SAS data library.                    *;
************************************************************;
libname SASFILE "&SASFILE";

************************************************************;
* Allocate a temporary file to hold the 1st pass log data.  *;
************************************************************;
filename tempfile temp;

************************************************************;
* First pass to get STARTED, PROGRAM, and ENDED records.    *;
************************************************************;
data _null_;

infile logdata truncover length=varlen;
file tempfile;
```

```
input @;

input bigline $varying200. varlen;

if substr(bigline,1,3) not in('Sun' 'Mon' 'Tue' 'Wed' 'Thu' 'Fri' 'Sat') then delete;

if index(bigline,'Request') = 0 and index(bigline,'Killing') = 0 then delete;

if index(bigline,'Internal Request') > 0 then delete;

put bigline;

run;

************************************************************;
* Second pass to create observations.                      *;
************************************************************;
data logdata(keep=request program ipaddress startdttm enddttm appservr elapsed errorflag killflag);

infile tempfile truncover length=varlen;

informat dayabrev   $4.
         daynum      2.
         monabrev   $3.
         year        4.
         time       time12.3
         reqword    $7.
         inreques    3.
         word       $4.
         ipaddress  $15.
         is         $2.
         program    $30.
         okay       $4.
         ended      $16.
         errors     $8.
         ;

format time startime endtime time. startdttm enddttm datetime.;

length appservr $4;

label program   = 'Program Name'
      appservr  = 'Application Server Number'
      request   = 'Request Number'
      ipaddress = 'Requesting IP Address'
      startdttm = 'Program Start Date/Time'
      enddttm   = 'Program End Date/Time'
      elapsed   = 'Program Elapsed Time'
      errorflag = 'Program Error Flag'
      killflag  = 'Program Killed Flag'
      ;

retain appservr "&APPSERVR"
       flag     '3'
       killflag ' '
       startdttm
       request
       ipaddress
       program
       ;
```

```
input dayabrev
      daynum
      monabrev
      year
      time
      reqword
      @
      ;

if reqword not in ('Request' 'Killing') then delete;

if reqword eq 'Killing' then killflag = '1';  /* For Killed programs */
else do;

    input inreques
          word
          @
          ;

    select(word);
            when('from') do;
                if flag ne '3' then output;

                flag = '1';
                errorflag = '0';
                killflag = ' ';

                startime = time;

               startdttm =
               dhms(mdy(put(monabrev,$monthno.),daynum,year),hour(startime),minute(startime),second(startime));

                request = inreques;

                program = ' '; /* To reset program from previous request */

                input  ipaddress;

            end;
            when('Prog') do;
                    flag = '2';

                    input is
                          program
                          ;

            end;
            when('ende')do;
                    if flag ne '2' then program = '*** Missing Program Name ***';

                    flag = '3';

                    endtime = time;
               enddttm =
               dhms(mdy(put(monabrev,$monthno.),daynum,year),hour(endtime),minute(endtime),second(endtime));

                    input okay
                          @
                          ;

                    if okay = 'with' then do;
```

```
                        input errors
                              ended
                              ;

                        errorflag = '1';

                end;
                else input ended;

                substr(ended,1,1) = ' ';
                ended = left(ended);
                elapsed = input(ended,7.2);

                output;

        end;

    otherwise;
    end;

end;  /* End of Do statement for non-Killed programs */

run;

************************************************************;
* Sort the data.                                           *;
************************************************************;
proc sort data=logdata;
     by startdttm appservr request program;
run;

************************************************************;
* Update the permanent SAS/IntrNet performance data set.   *;
************************************************************;
%IF %SYSFUNC(exist(SASFILE.INTRNET)) %THEN %DO;
  data SASFILE.INTRNET;
    modify SASFILE.INTRNET logdata;
        by startdttm appservr request program;

         if _iorc_=0 then replace;
               else output;
  run;
%END;
%ELSE %DO;
  data SASFILE.INTRNET;
   set logdata;
        by startdttm appservr request program;
  run;
%END;

    /************************************************************/
    /* End of the APPLOGRD Macro.                               */
    /************************************************************/
%MEND APPLOGRD;

    /************************************************************/
    /* Invoke the APPLOGRD Macro.                               */
    /************************************************************/

%APPLOGRD(C:\TEMP\SASINTRNET,H:\MY DOCUMENTS\SUGI 31\Measure SAS IntrNet Performance Paper,5020);
```

Appendix B – CONTENTS listing of the SAS/IntrNet program performance data set

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
5	appservr	Char	4			Application Server Number
1	dns	Char	15		$15.	Requesting DNS
7	elapsed	Num	8			Program Elapsed Time
4	enddttm	Num	8	DATETIME.		Program End Date/Time
8	errorflag	Char	1			Program Error Flag
9	killflag	Char	1			Program Killed Flag
2	program	Char	30		$30.	Program Name
6	request	Num	8			Request Number
3	startdttm	Num	8	DATETIME.		Program Start Date/Time

Appendix C – The Analyze_App_Server_Logs.sas program.

```
*******************************************************************************;
* Program:  Analyze_App_Server_Logs.sas                                       *;
*                                                                             *;
* Author:   Michael A. Raithel                                                *;
*                                                                             *;
* Purpose:  This program creates a number of report for SAS Application Server logs.*;
*           It is a SAS Macro that accepts two parameters:                    *;
*                                                                             *;
*           SASLIB - The SAS Data Library that contains the Application Server      *;
*                    performance data.                                        *;
*           TITLEONE - The full text for title1 of the reports.               *;
*                                                                             *;
*******************************************************************************;


*************************************************************;
* Options.                                                  *;
*************************************************************;
options nodate nonumber msglevel=I pagesize=200 linesize=100;


  /*****************************************************************/
  /* Beginning of the ANALOGS SAS Macro.                           */
  /*****************************************************************/
%MACRO ANALOGS(SASLIB,TITLEONE);


*************************************************************;
* Allocate the saslib SAS data library.                     *;
*************************************************************;
libname saslib "&SASLIB";

*************************************************************;
* Determine current month and set Macro variable.           *;
*************************************************************;
data _null_;

date = today()-1;
call symput('REPTDATE',put(date,worddate.));

run;

*************************************************************;
* Extract yesterdays data for further analysis.             *;
*************************************************************;
data yesterday(drop=x);
set  SASLIB.INTRNET(where=(datepart(startdttm)=today()-1));

label progname = 'Program Name';

x = index(program,'.');

if x > 0 then progname = substr(program,x+1,30-x);
else delete;

hour = hour(timepart(startdttm));
```

```
run;

*************************************************************;
* Create report of all programs executed.                   *;
*************************************************************;
proc freq data=yesterday;
     table progname;
title1 "&TITLEONE";
title2 'Frequency of SAS/IntrNet Programs That Were Executed';
title3 "on &REPTDATE";
run;

*************************************************************;
* Create report of Application Server usage.                *;
*************************************************************;
proc freq data=yesterday;
     table appservr;
title1 "&TITLEONE";
title2 "Application Servers Usage on &REPTDATE";
run;

*************************************************************;
* Create report of Min, Avg, Max Program Response Time.     *;
*************************************************************;
proc summary nway data=yesterday;
     class progname;
     var elapsed;
output out=progelap(drop=_type_) min=minelap mean=meanelap max=maxelap;
run;

proc print noobs label data=progelap;
label minelap  = 'Minimum Elapsed Time (Sec.)'
      meanelap = 'Average Elapsed Time (Sec.)'
      maxelap  = 'Maximum Elapsed Time (Sec.)'
      _freq_   = 'Number of Executions'
      progname = 'Program Name'
      ;
format minelap meanelap maxelap  7.2;
var progname minelap meanelap maxelap _freq_;
title1 "&TITLEONE";
title2 'Minimum, Average, and Maximum Response Times';
title3 "For Programs Executed on &REPTDATE";
title4 'On All Application Servers';
run;

*************************************************************;
* Create report of Min, Avg, Max Program Response Time      *;
* By Application Server.                                    *;
*************************************************************;
proc summary nway data=yesterday;
     class appservr progname;
     var elapsed;
output out=progelap(drop=_type_) min=minelap mean=meanelap max=maxelap;
run;

proc print noobs label data=progelap;
label minelap  = 'Minimum Elapsed Time (Sec.)'
      meanelap = 'Average Elapsed Time (Sec.)'
      maxelap  = 'Maximum Elapsed Time (Sec.)'
      _freq_   = 'Number of Executions'
      progname = 'Program Name'
```

```
      appservr = 'Application Server Number'
      ;
format minelap meanelap maxelap  7.2;
var progname minelap meanelap maxelap _freq_;
sum _freq_;
by appservr;
id appservr;
title1 "&TITLEONE";
title2 'Minimum, Average, and Maximum Response Times';
title3 "For Programs Executed on &REPTDATE";
title4 'On All Application Servers';
run;

*************************************************************;
* Create report of Min, Avg, Max Program Response Time     *;
* By Application Server, by hour.                           *;
*************************************************************;
proc summary nway data=yesterday;
     class hour progname;
     var elapsed;
output out=progelap(drop=_type_) min=minelap mean=meanelap max=maxelap;
run;

proc print noobs label data=progelap;
label minelap      = 'Minimum Elapsed Time (Sec.)'
      meanelap     = 'Average Elapsed Time (Sec.)'
          maxelap  = 'Maximum Elapsed Time (Sec.)'
          _freq_   = 'Number of Executions'
          progname = 'Program Name'
          hour
          ;
format minelap meanelap maxelap  7.2;
var progname minelap meanelap maxelap _freq_;
sum _freq_;
by hour;
id hour;
title1 "&TITLEONE";
title2 'Minimum, Average, and Maximum Response Times';
title3 "For Programs Executed on &REPTDATE";
title4 'Sorted by Hour';
run;

  /*****************************************************************/
  /* End of the ANALOGS SAS Macro.                                 */
  /*****************************************************************/
%MEND ANALOGS;

  /*****************************************************************/
  /* Invoke the ANALOGS SAS Macro.                                 */
  /*****************************************************************/
%ANALOGS(H:\SASINTRNET\Production\Library,Production SAS/IntrNet Report Application);
```