CHAPTER

8

Enhancements for SAS Users under Windows NT

Overview

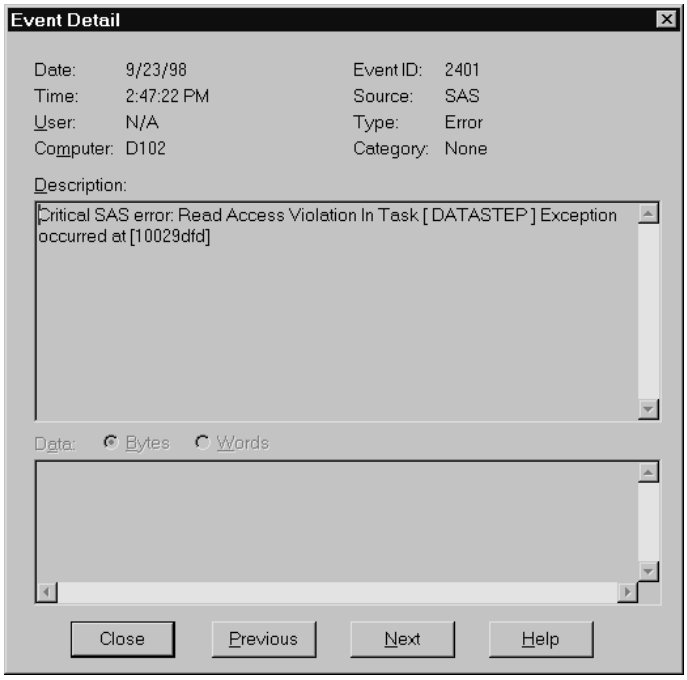
The SAS System provides some enhancements specific to Windows NT. These enhancements may be of interest to advanced users and system administrators in an enterprise environment. The SAS System now supports logging of error messages to the NT Event Log. Abnormal termination of SAS tasks (such as an access violation) can be viewed in the Event Log in addition to the SAS Log. Also, informational messages in SAS/CONNECT may be viewed in the Event Log. Another new feature is support for Windows NT Performance Monitor with the SAS System. Using NT Performance Monitor, users can monitor their SAS sessions (in either full-screen or batch) to obtain the information necessary to diagnose problems and tune their sessions.

NT Event Log

With event logging in Windows NT, applications, device drivers, services, and the operating system can record important hardware and software events in the NT Event Log.

After opening Event Viewer (found in the Administrative Tools folder or by starting EVENTVWR.EXE), you can view the Application Log by selecting **`Application`** from the **`Log`** menu. If a SAS task terminates abnormally, information regarding the task is placed into the Application Log. The **`Source`** column shows “SAS” as the event source. Messages from SAS/CONNECT display “SAS Job Spawner” as the event source. When you double-click on a SAS event, an Event Detail window similar to the one in Display 8.1 on page 178 opens with information about the event. This is the same error message you would see in your SAS Log file or in the SAS Log window.

Display 8.1 Event Detail Dialog Box



Sending Messages to the NT Event Log Using a User-Written Function

SAS System events can be sent to the NT Event Log using a user-written function in either SAS System code or SCL. Input to the function is a specific text string which corresponds to a type of event and the text string that will appear in the Event Viewer. Table 8.1 on page 178 lists the types of events available for the first parameter.

Table 8.1 Types of SAS System Events‘

Type of Event	First Parameter Value
Error	“ERROR”
Warning	“WARNING”
Information	“INFORMATION”
Success Audit	“SUCCESSAUDIT”
Failure Audit	“FAILUREAUDIT”

Although the first parameter values are displayed in upper case, mixed case is also allowed. The second parameter of the function is a string that will appear in the Event Viewer.

Examples of Using the User-Written Function to Write to the NT Event Log

The following is an example of some SAS System code in which the existence of a semaphore file is checked before lengthy processing:

```
%macro pdata(file);
    %let cmdstr = "dir &file";
    options noxwait;
    data _null_;
       call system(&cmdstr);
    run;
    %put &sysrc = sysrc;
    %put &file;
    %if &sysrc=0 %then %do;
       filename indata "&file";
       /* Your data step code for this file.  */
       DATA a;
          infile indata length=linelen;
          length line $ 200;
          input @1 line $ varying200. linelen;
       PROC print;
       run;
    %end;
    %else %do;
       /*  Log an NT Event of type Error.  */
       %let cmdstr = %str("The file &file did not exist
                           so no data step ran.");
       %put &cmdstr;
       DATA _null_;
          x=ntlog(''INFORMATION'',&cmdstr);
       run;
    %end;
 %mend;

 %pdata(c:\config.syss)
```

The following is SCL code to write to the NT Event Log:

```
/* Build a frame and add a pushbutton.  Change the Attribute
Name ''name'' to ''object1''.  In the Source window, add the
following code. */
object1:

x=ntlog("INFORMATION", "This is an INFORMATION event.");
x=ntlog("WARNING", "This is a WARNING event.");
x=ntlog("ERROR", "This is an ERROR event.");
x=ntlog("SUCCESSAUDIT", "This is a SUCCESSAUDIT event.");
x=ntlog("FAILUREAUDIT", "This is a FAILUREAUDIT event.");

return;
```

Sending Messages to the NT Event Log Using LOGEVENT.EXE

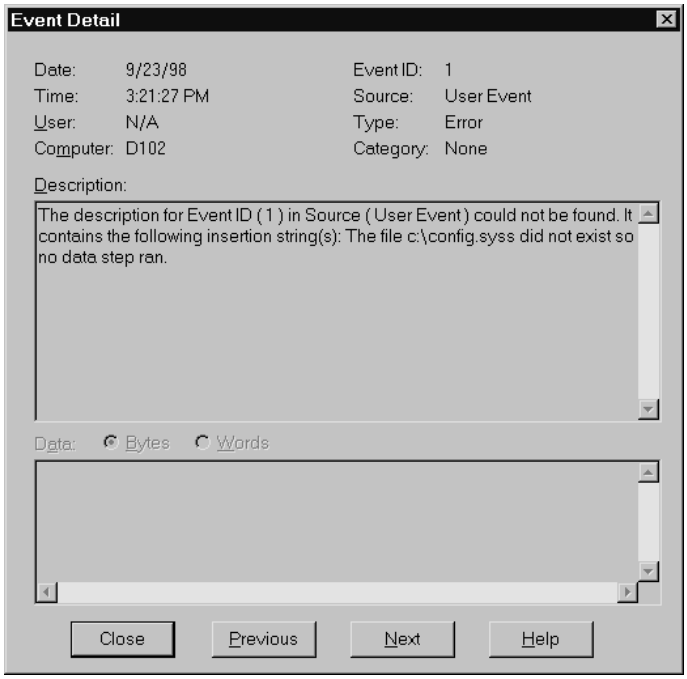
Using the LOGEVENT.EXE utility provided by the NT Resource Kit, SAS users can send their own information messages to the Event Log from within SAS code. The following is an example of some SAS code to do this.

In this example, the existence of a semaphore file is checked before SAS performs some lengthy processing.

```
%macro pdata(file);
   %local cmdstr;
   %let cmdstr = "dir &file";
   options noxwait;
   DATA _null_;
      call system(&cmdstr);
   run;
   %if &sysrc=0 %then %do;
      filename indata "&file";
      /* Your data step code for this file.  */
      DATA a;
         infile indata length=linelen;
         length line $ 200;
         input @1 line $ varying200. linelen;
      PROC print;
      run;
   %end;
   %else %do;
      /*  Log an NT Event of type Error.  */
      %let cmdstr = %bquote(c:\logevent.exe -s E
      "The file &file did not exist so no data
step ran.");
      DATA _null_;
         %sysexec &cmdstr;
      run;
   %end;
%mend;

%pdata(c:\config.syss)
```

When you double-click on the event in the Application Log, the Event Detail dialog box will give you the message displayed in Display 8.2 on page 181.

Display 8.2 Displaying a User Message in The Event Detail Dialog Box



NT Performance Monitor

Windows NT Performance Monitor (PERFMON.EXE) may be invoked from the Administrative Tools folder. Using Performance Monitor, you can monitor many of the Windows NT internal processes as well as information from third-party applications. Each piece of information monitored is known as a counter. Each counter can be found in logically grouped objects. For example, Windows NT has a Memory object that contains counters such as Available Bytes, Committed Bytes, and Page Faults/sec. The Processor object has counter such as %Processor Time and %User Time. By observing various system counters and application-defined counters, it is possible to determine performance problems. You can search for bottlenecks in your system and isolate them to areas such as hardware, system software, or your application. For more information on NT Performance Monitor, refer to the Windows NT Resource Kit.

Version 8 of the SAS System for Windows includes the following application-defined counters in the SAS object:

Virtual Alloc'ed Memory
: specifies the amount of committed virtual memory allocated by SAS through the VirtualAlloc() API.

Disk ReadFile Bytes Read Total
: specifies the total number of bytes read from disk files by SAS through the ReadFile() API.

Disk ReadFile Bytes Read/Sec
: specifies the number of bytes read per second from disk files by SAS through the ReadFile() API.

Disk WriteFile Bytes Written Total
: specifies the total number of bytes written to disk files by SAS through the WriteFile() API.

Disk WriteFile Bytes Written/Sec
: specifies the number of bytes written per second to disk files by SAS through the WriteFile() API.

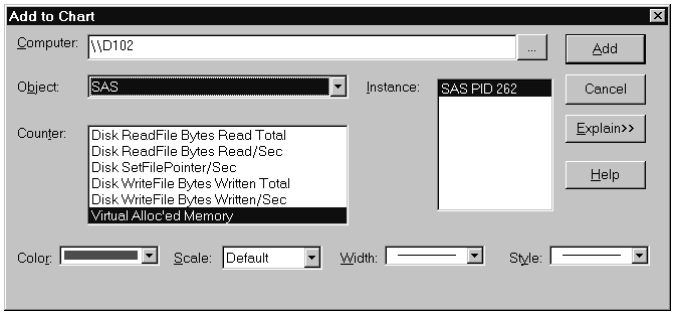
Disk SetFilePointer/Sec
: specifies the number of times per second that the SetFilePointer() API has been successfully called by SAS on disk files.

PSE36 Current Usage K
: specifies the amount of Extended Server Memory currently in use.

PSE36 Peak Usage K
: specifies the maximum amount of Extended Server Memory used in the current SAS session.

To monitor SAS counters, start SAS and then start NT Performance Monitor by selecting **`Performance Monitor`** from the Administrative Tools folder. Then click on the **`Add Counter`** (⊞) button to open the Add to Chart dialog box. After selecting **`SAS`** from the **`Object`** combo box, the Add to Chart dialog box looks like Display 8.3 on page 182.

Display 8.3 Add to Chart Dialog Box



For a description of each counter, click on Explain. You can now select counters from the list box, clicking on Add after each selection. After selecting the counters you want to monitor, click on Done. Performance Monitor immediately collects and display information about the counters you selected.

You may see multiple instances monitored, where each instance is a separate SAS process. SAS instances are listed in the form "SAS PID *number*". The PID *number* is the process identifier of the SAS session. You can see a list of all processes using NT Task Manager.

Performance Monitor is useful for tuning and diagnosing your application or computer system. By correlating the information from the SAS counters with other NT counters, you can more easily troubleshoot performance problems. An example would be a case of a user whose SAS job appears not to be working. Perhaps it is performing a long and complicated data step that generates a very large data set on a network volume. The user can be certain that the job is still working by monitoring the Disk WriteFile Bytes Written/Sec and Disk WriteFile Bytes Written Total counters.

Examples of Monitoring SAS Performance

The examples in this section show how the system performs when SAS process the DATA, PROC SORT, and PROC SQL steps. By entering the example SAS code in these examples, you can see how SAS uses the various system resources.

You set up all examples as follows:

1. invoke SAS and the Performance Monitor
2. open the Add Chart window and select the SAS object
3. add these SAS counters:
   - Disk ReadFile Bytes Read/Sec
   - Disk WriteFile Bytes Written/Sec
   - Disk SetFilePointer/Sec
4. select the Process object and select 'sas' from the Instances list box
5. add these Process counters:
   - %Processor Time
   - %User Time
   - %Privileged Time
6. click on Done.

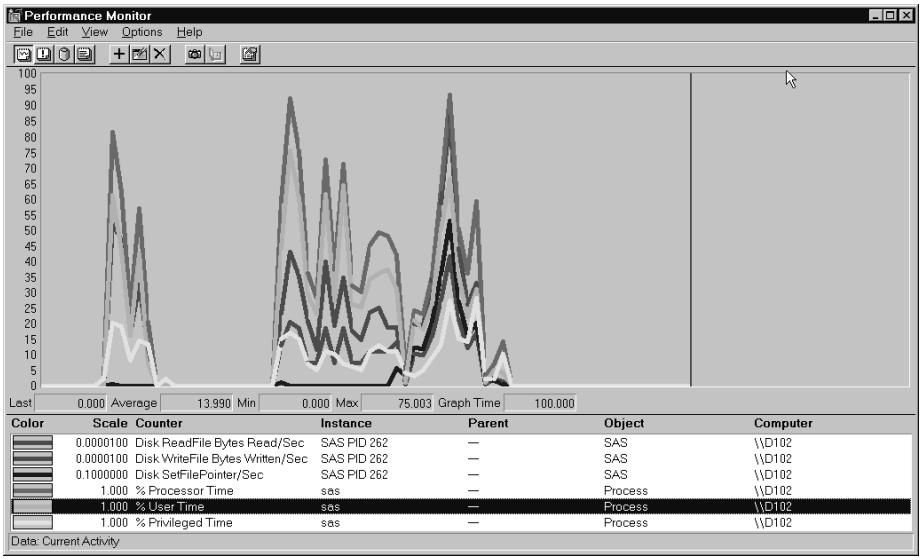
Examining the Performance Between the DATA and PROC SORT Steps

To see the difference in performance between the DATA step and the PROC step, submit this code:

```
options fullstimer;
  /* Create a test data set with some random data. */
DATA a (drop=s);
   do i = 1 to 500000;
      x = ranuni(i);
      y = x*2;
      z = exp(x*y);
      output;
   end;
   /* The sleep helps to delineate the subsequent  */
   /* sort in the Performance Monitor graph        */
   s = sleep(15);
run;
PROC sort data = a noduplicates;
   by z y x i;
run;
```

After submitting this code, the Performance Monitor graph should generate results like those in Display 8.4 on page 184. You may have to adjust the scale factor of the different counters.

Display 8.4 Performance of the DATA Step and the PROC SORT Step



You can see in the DATA step that there is very little activity from Disk ReadFile Bytes Read/Sec or Disk SetFilePointer/Sec. Notice that in the subsequent PROC SORT there is much more activity from these two counters. This indicates that the data set is being read (Disk Readfile Bytes Read/Sec) in order to be sorted, and that a certain amount of random I/O is performed by the sort (Disk SetFilePointer/Sec). The pause in the activity is caused by the SLEEP function that comes after the DATA step. The Disk WriteFile Bytes Written/Sec is active in both the DATA step and the PROC SORT. Lastly, you can correlate the counters from the Process object with the user and system CPU times in your SAS log.

Examining a PROC SQL Query

In this example you'll be submitting code to examine the performance of a PROC SQL query with and without using an index. Be sure that SAS and Performance Monitor are started.

To perform a PROC SQL with an index:

1. submit the code in Step 1 and Step 2. Step 2 creates an index.
2. clear the Performance Monitor graph by selecting **`Clear Display`** from the **`Edit`** menu
3. submit the code in Step 3 to see a graph like Display 8.5 on page 185.

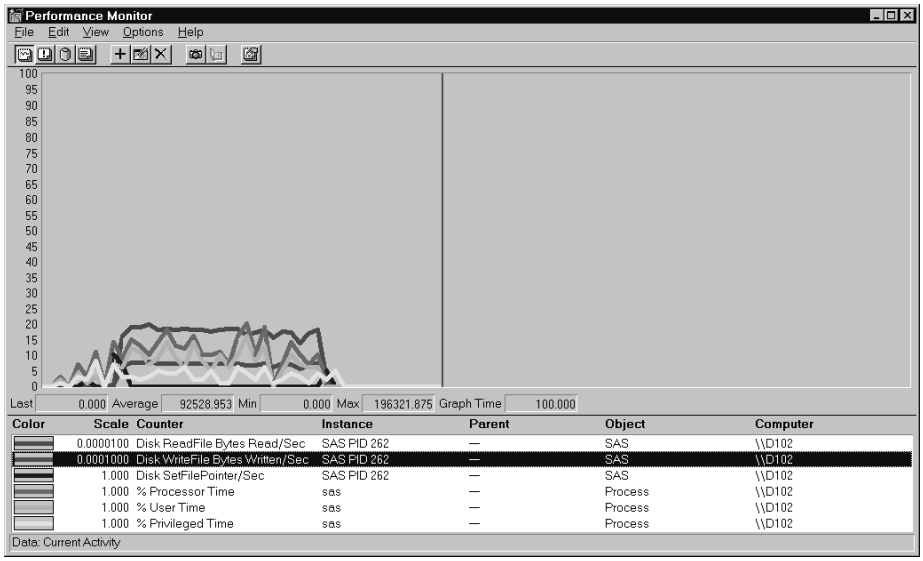
To perform a PROC SQL without an index:

1. resubmit Step 1
2. clear the Performance Monitor graph
3. resubmit Step 3 to see a graph like Display 8.6 on page 186.

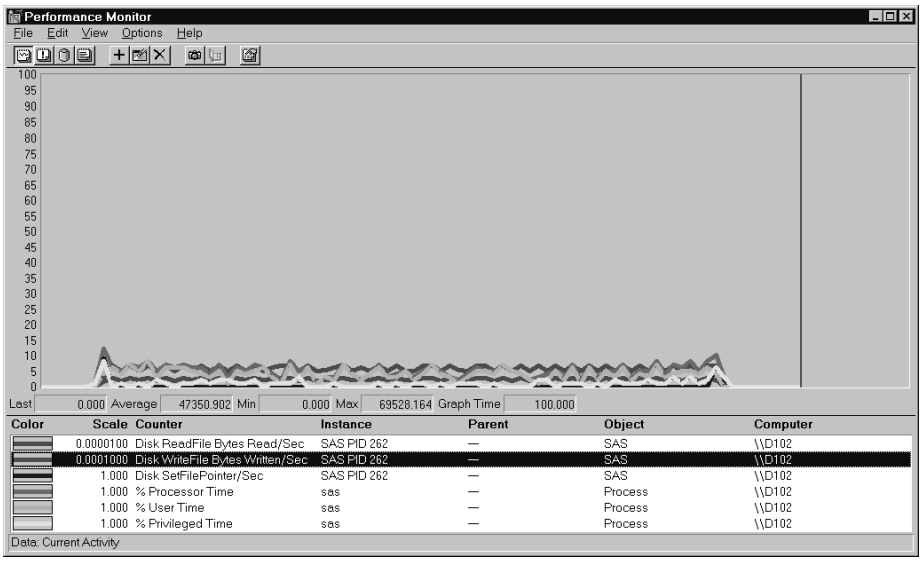
```
   /* Step 1                                          */
   /* Create a test data set with some random data. */
   /* Do this twice - once with Step 2 and once     */
   /* without Step 2.                                 */

libname sample 'c:\';
DATA sample.a;
   do i = 1 to 500000;
```

```
      x = ranuni(i);
      y = x*ranuni(i);
      z = exp(y);
      output;
   end;
run;

   /* Step 2                                              */
   /* Create a simple index on variable x.                */
   /* Submit this step once.                              */

PROC DATASETS library = sample;
   modify a;
   index create x;
   quit;

   /* Step 3                                              */
   /* Perform a query on the data.  Do this twice -       */
   /* once with an index and once without an index        */
   /* The query should select about 50% of the            */
   /* observations in the data set.                       */

PROC SQL;
   create table sample.yz as
   select y,z
      from sample.a
      where x > 0.5;
   quit;
```

Display 8.5 Performance of PROC SQL with an Index



Display 8.6 Performance of PROC SQL without an Index



In Display 8.6 on page 186, the counters averaged under 10% on the scale, whereas in Display 8.5 on page 185, several of the counters averaged well over 10%, and the Disk WriteFile Bytes Written/Sec counter rose over 25%. A higher value for these counters implies good overall throughput for the operation. Note that to make a valid comparison like this with Performance Monitor graph, you must make sure that the counters are using the same scale. You can double-check by observing the absolute values. The Average value for Disk WriteFile Bytes Written/Sec in Display 8.5 on page 185 was 92528.953. Contrast this with the same counter in Display 8.6 on page 186, in which the Average value was 47350.902. For this operation, bytes were written almost twice as fast when the data set was indexed.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Companion for the Microsoft Windows Environment, Version 8,* Cary, NC: SAS Institute Inc., 1999. pp.555.

SAS Companion for the Microsoft Windows Environment, Version 8

ISBN 1–58025–524–8

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, September 1999

The Institute is a private company devoted to the support and further development of its software and related services.