

# Modifying The LogParse PassInfo Macro to Provide a Link between Product Usage in Rtrace Log and Time Used in Job Log

Ronald J. Fehd, Centers for Disease Control and Prevention, Atlanta, GA, USA

## ABSTRACT

This paper examines the issues of building a SAS usage database by modifying macros used to read the job and rtrace logs to provide a common key in each log for joining the job times with the products used.

Expected audience is advanced users, macro programmers, site and server administrators. Readers are expected to be familiar with issues of configuration and running programs as a batch process.

**Keywords:** altlog log logparse passinfo rtrace rtraceloc .

## INTRODUCTION

Executing a SAS® program produces several outputs:

- the job log, which contains times of each step and the job
- the job listing, the desired output
- the product usage (rtrace) log

The job log can be read with the SAS Institute[FullStimer] logparse and passinfo macros. Using the LogParse suite of programs, we may extract these job times and build a SAS Time Usage transaction.

We could then answer questions such as: .....

In addition there is a little-known log file produced by the option pair rtrace and rtraceloc that directs the SAS File Resource Tracking System to write a product usage log which lists the resources that SAS uses: files referenced, opened and closed. From this log we may extract a list of the products that each job uses. Using the Rtrace suite of programs developed by Raithel[Rtrace] we may extract the list of products and build a SAS Product Usage transaction. We could then answer questions such as: .....

## Contents

Database Theory	2
The LogParse Suite	2
The Rtrace Suite	2
Accuracy of DateTime	3
Modifying SASv?.cfg	4
Modifying Autoexec.sas	5
Modifying PassInfo	5
Modifying LogParse	6
Modifying RTRACERD	7
A Demonstration Suite	7
Merging Time and Product Usage	10
Conclusion	10

- 
- How many users do we have?
  - Who are the least, intermediate, and most frequent users?
  - How much time is each group using, per time period (daily, weekly, monthly)?

- 
- How many users do we have using different products? e.g.: ETS, Graph, IML, STAT, etc.
  - Who are the least and most frequent users of each product?
  - How many times are the various products used, per time period?
-

This paper examines the issues of joining the time and product usage into one transaction so that we could then answer questions such as: .....

- How much time is spent using a specific product?
- How much time is spent by which users, using a product?
- Which programs are using a specific product?

## DATABASE THEORY

Kimball and Ross[1, pg. 18, 133], state that there are three types of fact tables in databases: transactions, and two kinds of snapshots: periodic and accumulating. Here I am focused on building a job transaction. I seek to combine two sets of job transaction facts: product usage and job times used, each in a separate log. The obvious primary key for the join is the job datetime.

In the following sections we examine the issues of how to produce one row from each job product and time log with a unique datetime.

Database Information	
fact tables:	transaction periodic snapshot accumulating snapshot

Transaction Table Information	
transaction keys	
has:	facts
keys are:	primary: unique row identifier datetime foreign: machine: SysHostName user: SysUserId
facts: time	job, cpu-user, -system, etc.
facts: product usage	product names: core (base), . . . , stat, etc. SAS/Access: . . . , etc.

## THE LOGPARSE SUITE

There are two routines in the logparse[FullStimer] suite: **passinfo**, which writes a header to the job log with various keys – including a date – and facts. And **logparse**, which reads the job time log and produces a data set with the times of the initialization, each step, and the job.

1. **passinfo**: write header to job log
2. **logparse**: read job log

## THE RTRACE SUITE

Raithel[Rtrace] has four routines in his rtrace suite. The first is a modification of the **SASv8.cfg** configuration file, in which the rtrace facility log is turned on, a default destination specified, and a site-wide autoexec.sas is provided. The second is the **autoexec.sas**, which provides a unique rtrace log destination for the job with a datetime in the rtrace log filename. Next is **rtracerd**, which reads the job product usage log. And last is **pcfldttm**, a subroutine of rtracerd, which reads the file-closed date and time and returns it to rtracerd which uses them to calculate the job time used.

In this paper I replace the calculation of job time used — the difference between job-start datetime and the file-closed datetime of the rtrace log — with the time read from the job log:

1. **SASv8.cfg**
  - (a) rtrace on
  - (b) rtrace log default destination
  - (c) site autoexec.sas
2. **autoexec.sas**:  
unique rtrace log destination
3. **rtracerd**: read rtrace log
4. **pcfldttm**: read pc file date time

NOTE: The SAS System used:  
real time 3.94 seconds

## ACCURACY OF DATETIME

Go to the Windows DOS prompt and type in: `time`; you see that your computer clock returns a time in format `time11.2`, i.e.: `hh:mm:ss.ss`, accurate to two decimal places.

```

_____ DOStime.txt _____
1 The current time is: 12:16:01.59
2 Enter the new time:

```

```

_____ DOSfile-datetime.txt _____
6 05/08/2006 12:16p          54 DOStime.txt
7                1 File(s)          54 bytes

```

The Alert Reader immediately recognizes the difference between the accuracy of the file-close datetime — left column, in minutes — and the datetime in the `rtraceloc` filename — right column, in seconds. For short jobs — lines 6–11 — the interval between job-start and file-close datetime produces an apparent negative job time!

```

_____ DOSfile-datetime-rtraceloc test.txt _____
6 05/19/2006 08:22a          0 z-19MAY2006-08-22-40-00.log
7 05/19/2006 08:22a          0 z-19MAY2006-08-22-50-00.log
8 05/19/2006 08:22a          0 z-19MAY2006-08-22-55-00.log
9 05/19/2006 08:22a          0 z-19MAY2006-08-22-56-00.log
10 05/19/2006 08:22a         0 z-19MAY2006-08-22-57-00.log
11 05/19/2006 08:22a         0 z-19MAY2006-08-22-58-00.log
12 05/19/2006 08:23a         0 z-19MAY2006-08-22-59-00.log
13 05/19/2006 08:23a         0 z-19MAY2006-08-23-01-00.log
14 05/19/2006 08:23a         0 z-19MAY2006-08-23-02-00.log
15 05/19/2006 08:23a         0 z-19MAY2006-08-23-03-00.log

```

Both Raithel and SAS Institute's `passinfo.sas` fetch a value from the `datetime` function and then convert into a user-readable form, accurate to whole seconds.

```

autoexec-Raithel:  datetime19.  09MAY2006:08:41:59
passinfo:          datetime16.  05may06:09:02:08

```

```

_____ autoexec-Raithel.sas _____
1 /*rtraceloc*****Do Not Remove*****/
2 %let  userid  = %sysget($USERNAME); *Win server evar;
3 %let  _utime  = %sysfunc(datetime(), datetime19.0);
4 %let  usertime = %sysfunc(compress(&_utime, ':'));
5 options rtraceloc = "F:\sastrace\&userid.&usertime.rtd";
6 /*rtraceloc*****/

```

```

_____ passinfo.sas snip 1 _____
19 temp=datetime();
20 temp2=lower(trim(left(put(temp,datetime16.))));
21 call symput('datetime', trim(temp2));

```

Note: Compare `autoexec-Raithel` filename in option `rtraceloc` with that in `passinfoX.sas`, snip 4, line 54, below.

The simplest way to pass a numeric value between routines is to use the simple computer numeric representation: hexadecimal.

Program `ViewDateTime.sas`, line 2, shows a method of assignment of a datetime value to a macro variable.

I use this method in `passinfoX.sas`; see snip 2.

```

_____ ViewDateTime.sas _____
1 options nosource;
2 %let DateTime = %sysfunc(datetime(), hex16.);
3 %put DateTime<&DateTime.> hex16;
4 %put DateTime<%sysfunc(putn(&DateTime.x, 14.3))> number;
5 %put DateTime<%sysfunc(putn(&DateTime.x,datetime21.2))> formatted;

```

```

_____ ViewDateTime.log _____
47 DateTime<41D5DAF407A7BE77> hex16
48 DateTime<1466683422.000> number
49 DateTime<23JUN2006:12:03:42.00> formatted

```

In the following sections, I review the modifications of various files necessary to set up a project so that a job writes its time and rtrace logs to separate folders, from which they can be read by macros `logparse` and `rtracerd`, and then joined.

## MODIFYING SASV?.CFG

The project root folder contains:

```
rtrace  rtrace logs
sas     programs
sas7b   data sets: sas7bdat
```

```

----- projroot.txt -----
4  Directory of C:\SASsite\LaTeX\ModPassInfo
5
6  05/18/2006  08:47p      <DIR>      .
7  05/18/2006  08:47p      <DIR>      ..
8  05/16/2006  05:43p      <DIR>      rtrace
9  05/18/2006  08:47p      <DIR>      sas
10 05/16/2006  03:22p      <DIR>      sas7b
```

The project configuration file is most important to this task since several of the options can only be assigned at start up (command line or batch file), or in a configuration file.

Note: `-SET`, allocates (names and assigns a value) to an environment variable. Environment variables may then be referenced using the bang (!, exclamation mark) as a prefix. For practical purposes, I refer to them as project constants.

Note: `SASv9.cfg` is similar.

```

----- SASv8.cfg -----
1  -config          'C:\Program Files\SAS Institute\SAS\V8\SASv8.cfg'
2  -SET    ProjRoot 'C:\SASsite\LaTeX\ModPassInfo'
3  -SET    rtraceloc '!ProjRoot\rtrace'
4  -autoexec      '!ProjRoot\sas\autoexec.sas'
5  -fullstimer    /* show user- and system-cpu time for logparse */
6  -rtrace      all          /* turn on rtrace log production */
7  -rtraceloc   '!rtraceloc\00RtraceDefault.log'
8  -SASinitialFolder '!ProjRoot\sas'
```

This configuration file was written specifically for this test suite. Note that all folders (directory-specifications) — `autoexec`, line 4, `rtraceloc`, line 7, and `SASinitialFolder`, line 8 — are child folders of the parent `ProjRoot`.

To follow Raithe's suggestions of a site-wide `autoexec`, then change the `-autoexec` file-specification.

Note the environment variable `rtraceloc`, line 3. This is necessary so that the folder can be changed in this configuration file, and not hardcoded in `passinfoX.sas`; see snip 3, line 64.

Macro `logparse` is designed to read time blocks written by option `fullstimer`, line 5. This option is discussed in Raithe[LogParse].

## MODIFYING AUTOEXEC.SAS

This demonstration suite has all the programs containing both original macros and their modifications in the project folder.

Macro `logparse` requires that a header record be written to each program log with the macro `passinfo`. This utility is accomplished by adding a call to `passinfoX` in the `autoexec` by enabling the autocall facility to search the project folder, here named `SiteMacr`. A production version would change this directory-specification from the project folder to a site-wide macro folder.

Options `SASautos` and `mautosource` are the pair needed for the autocall facility: SAS searches for called macros in the named filerefs. For more information about the autocall facility, see Carpenter[ch. 12][Guide-2e] and Fehd[SASautos-Comp].

```
autoexec.sas
1  FileName          SiteMacr "%sysfunc(getoption(SASinitialFolder))";
2  Options  SASautos = (SiteMacr SASautos) mautosource; %PassInfoX;
3
4  LibName  Library  "!ProjRoot\sas7b";
```

Note the `logparse` output data set is written to libref `Library`. Compare to `autoexec-Raithel.sas`, above; the overwrite of option `rtraceloc` is in `passinfoX.sas`, snip 4, line 64, below.

## MODIFYING PASSINFO

Copy `passinfo.sas` and change the macro name:

```
passinfoX.sas snip 1
11
12 %macro passinfoX;
```

Macro `passinfo` can be simplified in several ways. The first is to replace the `datetime` conversion with a macro variable allocation. `Date-Time` is the primary key which is used to join the two transactions.

```
passinfoX.sas snip 2
21 %* call symput('datetime', trim(temp2)); %*PassInfo ;
22 %let DateTime = %sysfunc(datetime(),hex16.); %*PassInfoX;
```

Next: align the name of the information with the value: `datetime`.

```
passinfoX.sas snip 3
51 %*put PASS HEADER date=&datetime; %*PassInfo ;
52 %*put PASS HEADER datetime=&datetime; %*PassInfoX;
```

Last is a series of improvements:

- reduce the size of the `passinfo` header by replacing the `proc options` calls, lines 54–57, with macro function calls, lines 59–62
- overwrite `rtraceloc`, line 64 compare the filename with `autoexec-Raithel.sas` above
- add one variable: `SysUserid`, line 65
- prettify with a readable `datetime`-stamp, line 66

```

_____ passinfoX.sas snip 4 _____
54  /*PassInfo : 27 lines in log *****
55  proc options option=MEMSIZE; run;          /*PassInfo ;
56  proc options option=SUMSIZE; run;         /*PassInfo ;
57  proc options option=SORTSIZE; run;       /*PassInfo ;
58
59  /*PassInfoX: 3 lines in log ***** */
60  %put PASS HEADER memsize=%sysfunc(getoption(memsize)); /*PassInfoX;
61  %put PASS HEADER sumsize=%sysfunc(getoption(sumsize)); /*PassInfoX;
62  %put PASS HEADER sortsize=%sysfunc(getoption(sortsize)); /*PassInfoX;
63
64  Options rtraceloc = "!RtraceLoc\&SysUserid.&datetime..log";/*PassInfoX;
65  %put PASS HEADER SysUserId=&SysUserId.; /*PassInfoX;
66  %Put PASS HEADER END %sysfunc(putn(&DateTime.x,datetime21.2));

```

The result of the new and improved passinfoX is:

```

_____ passinfoXtest.log _____
28  PASS HEADER BEGIN
29  PASS HEADER os=WIN
30  PASS HEADER os2=WIN_PRO
31  PASS HEADER host=RONALD-VHQPUEVI
32  PASS HEADER ver=8.02.02MOP012301
33  PASS HEADER datetime=41D5DAF40765C28F
34  PASS HEADER parm=
35  PASS HEADER memsize=0
36  PASS HEADER sumsize=0
37  PASS HEADER sortsize=2097152
38  PASS HEADER SysUserId=Administrator
39  PASS HEADER END 23JUN2006:12:03:41.00

```

## MODIFYING LOGPARSE

Macro `passinfoX` is writing more information into the header block.

Macro `logparseX` must be changed to modify the data structure, and read the additional variables.

Add the variable `UserId` to the data structure. Consolidate `datetime` attributes.

```

_____ logparseX.sas snip 1 _____
61  attrib UserId length = $ 16 label = 'UserId' /*logparseX;
62  datetime length = 8 format = datetime. /*logparse;
63  label = 'Run Date/Time';/*logparse;

```

Add the variable `UserId` to the retain list.

```

_____ logparseX.sas snip 2 _____
128 retain UserId '.' datetime /*logparseX;
129 stepcnt 0

```

Replace the variable `Date` with `DateTime`, and add the variable `UserId`.

```

_____ logparseX.sas snip 3 _____
273  /*when (keyword = 'DATE') do; /*logparse ;
274  /* datetime = input(scan(upcase_Line,2,"="),datetime.);
275  /*end;
276  when (keyword = 'DATETIME') /*logparseX;
277  datetime = input(scan(upcase_Line,2,"="),hex16.);
278  /* add additional variable value scans here */
279  when (keyword = 'SYSUSERID') /*logparseX;
280  UserId = scan(upcase_Line,2,"=");

```

## MODIFYING RTRACERD

Macro RtraceRd — read rtrace log — requires a few adjustments.

Rename the macro and add the parameter `testing`; this is used in snip 6 to not delete the log file.

```
_____ RTRACERDx.sas snip 1 _____
1  %MACRO RTRACERDx(PRODFILE,RTRACEFL,testing = 1);
2  *****;
```

Disable the call the PCFLDTTM: PC File Date Time.

```
_____ RTRACERDx.sas snip 2 _____
36  /** %PCFLDTTM(&RTRACEFL);                                %*RtraceRdX;
37  %If &Testing. %then %Put FILEDATE<&FileDate.> FILETIME <&FileTime.>;/**/
```

Add DateTime to the keep list and the data structure.

```
_____ RTRACERDx.sas snip 3 _____
52  data tracetmp(keep = DateTime                                %*RtraceRdX;
53  system userid strtdate strttime enddate endtime
54  product);
55  attrib DateTime length = 8 format = hex16.                %*RtraceRdX;
56  Hex16 length = $16;
```

Comment out the section calculating the start-date and -time,

```
_____ RTRACERDx.sas snip 4 _____
121
122  /* Obtain point of reference to decomp record ***** %*RtraceRdX;
```

... and replace it with:

```
_____ RTRACERDx.sas snip 5 _____
134  endtime = "&FILETIME"t;                                %* Get End Time from PC time *;
135  /*end decomp record ***** */ %*RtraceRdX;
136  %*e.g.: RJF2_41D5CCBB3087AE14.log;
137  Userid = upcase(scan(rtracefl,1,'_ '));
138  Hex16 = scan(rtracefl,2,'_ ');
139  DateTime = input(Hex16 ,hex16.);
140  strtdate = datepart(DateTime);
141  strttime = timepart(DateTime);                                %*RtraceRdX;
```

Note the rtraceloc filename was allocated in the configuration file and overwritten by `passinfoX`.

Delete the file only in production, not testing.

```
_____ RTRACERDx.sas snip 6 _____
224  data _null_; rc = .;                                %*RtraceRdX;
225  %If not &Testing. %then %do;                        %*RtraceRdX;
226  rc = fdelete('rtracefl');%end;                    %*RtraceRd, RtraceRdX;
```

## A DEMONSTRATION SUITE

I use this Windows batch file, `test.bat`, to run each program in the demonstration suite. Calls to `passinfoXtest` and `ViewDateTime` keep the times in their logs close to those of the demonstration log, `test.log`.

Before each test run I delete all SAS data sets in the library: `uDeleteSAS7b`.

```

test.bat
1 call SAS passinfoXtest
2 call SAS ViewDateTime
3 call uDeleteSAS7b
4 call SAS test
5 call SAS RtraceRdXtestcall
6 call SAS logparseXtest
7 call SAS MergeTimeRtrace

```

Program test.sas is supposed to produce a small log for reading with macro logparseX. In order to have a pair of time and rtrace logs from the same program execution, I added a step to write the program RtraceRdXtestcall.sas with the correct rtrace log filename.

```

test.sas
1 DATA TEST1;
2 set SAShelp.Class;
3 PROC Print data = Test1;
4 PROC Freq data = Test1;tables sex;
5
6 DATA _Null_;
7 file 'RtraceRdXtestcall.sas';
8 put '%RTRACERDx(PRODFILE = !ProjRoot\sas7b'
9 / ',RTRACEFL = '
10 "%sysfunc(getoption(rtraceloc)) );"
11 / 'Proc SQL; describe table Library.Rtrace; quit; '
12 / 'Proc Print data = Library.Rtrace(keep = DateTime Product); '
13 ; stop; run;

```

```

RtraceRdXtestcall.sas
1 %RTRACERDx(PRODFILE = !ProjRoot\sas7b
2 ,RTRACEFL = !RtraceLoc\Administrator_41D5DAF407DD9168.log );
3 Proc SQL; describe table Library.Rtrace; quit;
4 Proc Print data = Library.Rtrace(keep = DateTime Product);

```

The log shows the data structure of Library.Rtrace.

```

RtraceRdXtestcall.log
141 create table LIBRARY.RTRACE( bufsize=8192 )
142 (
143   DateTime num format=HEX16.,
144   userid char(15) label='User ID',
145   strtdate num format=WORDDATE. label='Start Date',
146   enddate num format=WORDDATE. label='End Date',
147   strttime num format=TIME8. label='Start Time',
148   endtime num format=TIME8. label='End Time',
149   system char(3) label='System ID',
150   product char(8) label='SAS Product'

```

The listing from program RtraceRdXtestcall.sas shows that the transaction contains DateTime, shown in hex16., and the single product used by test.sas: core.

```

RtraceRdXtestcall.lst
4 Obs          DateTime      product
5
6 1           41D5DAF407DD9168    core

```

Compare the hex16 DateTime in RtraceRdXtestcall.lst, above, line 6, with test.log snip 1, line 33, below.

```

----- test.log snip 1 -----
33 PASS HEADER datetime=41D5DAF407DD9168
34 PASS HEADER parm=
35 PASS HEADER memsize=0
36 PASS HEADER sumsize=0
37 PASS HEADER sortsize=2097152
38 PASS HEADER SysUserId=Administrator
39 PASS HEADER END 23JUN2006:12:03:43.00

```

Program `logparseXtest.sas` calls macro `logparseX` which reads the `passinfoX` header and the fullstimer job time block in `test.log` and prints the transaction.

```

----- logparseXtest.sas -----
1 *options mprint;
2 %logparseX( saslog = test.log
3           , outds = Library.LogParseXtest);
4
5 Proc SQL; describe table Library.LogParseXtest;quit;
6
7 PROC Print data    = Library.LogParseXtest
8           (where  = (StepName eq 'SAS')) noobs;
9           var      DateTime RealTime UserTime SysTime Memused;

```

The log shows the data structure of `Library.LogParseXtest`.

```

----- logparseXtest.log -----
81 create table LIBRARY.LOGPARSEXTTEST( bufsize=16384 )
82 (
83   UserId char(16) label='UserId',
84   datetime num format=DATETIME. label='Run Date/Time',
85   logfile char(200) label='Log file name',
86   stepname char(20) label='Step Name',
87   portdate char(25) label='SAS Port Date',
88   platform char(50) label='Platform',
89   scp char(50) label='Operating System',
90   realtime num format=TIME12.3 label='Elapsed Time',
91   usertime num format=TIME12.3 label='User Time',
92   systime num format=TIME12.3 label='System Time',
93   cputime num format=TIME12.3 label='CPU Time',

```

Compare the `DateTime`-stamp in `test.log` snip 1, line 39, above, with `logparseXtest.lst`, below, which is rounded to seconds.

```

----- test.log snip 2 -----
106 NOTE: The SAS System used:
107     real time           1.81 seconds
108     user cpu time       0.17 seconds
109     system cpu time     0.73 seconds
110     Memory                2846k

```

Listing `logparseXtest.lst` shows the time log transaction.

```

----- logparseXtest.lst -----
4      datetime          realtime          usertime          systime          memused
5
6 23JUN06:12:03:43      0:00:01.810      0:00:00.170      0:00:00.730      2846

```

## MERGING TIME AND PRODUCT USAGE

Now I have two data sets, each contains their common primary key, DateTime and the facts from the two logs: time and product usage.

```
----- MergeTimeRtrace.sas -----
1 DATA MergeTimeRtrace (keep = DateTime %*key ;
2                               RealTime %*log ;
3                               Product %*rtrace;
4                               );
5 merge Library.logparseXtest(where = (StepName eq 'SAS'))
6       Library.Rtrace;
7 by   DateTime;
8
9 Proc SQL; describe table Work.MergeTimeRtrace; quit;
10
11 Proc Print data =           Work.MergeTimeRtrace;
```

```
----- MergeTimeRtrace.lst -----
4 Obs          datetime          realtime    product
5
6  1          23JUN06:12:03:43    0:00:01.810    core
```

## CONCLUSION

Program `MergeTimeRtrace` demonstrates that the join is possible. However there are several issues yet to be solved for a production process:

1. Automating saving of job time logs similar to rtrace logs
2. Automating reading of job time logs
3. Modifying the rtrace data structure: at present, each product is in a separate row; multiple products must be in a single transaction row for the join; e.g.: product1 = core, product2 = graph, etc.
4. macro `logparse` reads the time blocks of every job step; a shorter version which read only the last time block would be better
5. macro `logparse` was designed to read logs run with option `fullstimer`, reading logs run with option `nofullstimer` would be appropriate

This paper has demonstrated much attention to the details of modifying many files: configuration, macros, and programs, to merge job times and product usage. There are many issues yet to solve but I have demonstrated the proof of the concept.

## ACKNOWLEDGEMENTS

Michael Raithel answered my probing questions in several interviews. Mark Tabladillo read over early drafts of this paper and provided commentary and critique.

## REFERENCES

- [Guide-2e] Art Carpenter, 2004. *Carpenter's Complete Guide to the SAS<sup>®</sup> Macro Language, Second Edition*, Cary, NC: SAS Institute Inc.  
[http://support.sas.com/publishing/bbu/companion\\_site/59224.html](http://support.sas.com/publishing/bbu/companion_site/59224.html)
- [SASautos-Comp] Ronald Fehd, *A SASAUTOS Companion: Reusing Macros*, Proceedings of the 30th Annual SAS<sup>®</sup> Users Group International Conference, 2005.  
<http://www2.sas.com/proceedings/sugi30/267-30.pdf>
- [1] Ralph Kimball, Margy Ross, (2002). *The Data Warehouse Toolkit, The Complete Guide to Dimensional Modeling, Second Edition*, John Wiley & Sons, Inc., New York.  
<http://www.kimballgroup.com/html/books.html>  
<http://www.amazon.com/exec/obidos/ASIN/0471200247/ralphkimballc-20/104-9414816-2399940>
- [Rtrace] Michael A. Raithel, *Measuring SAS<sup>®</sup> Software Usage on Shared Servers With the RTRACE Facility*, Proceedings of the 29th Annual SAS<sup>®</sup> Users Group International Conference, 2004.  
<http://www2.sas.com/proceedings/sugi29/215-29.pdf>
- [LogParse] Michael A. Raithel, *Programmatically Measure SAS<sup>®</sup> Application Performance On Any Computer Platform With the New LOGPARSE SAS Macro*, Proceedings of the 30th Annual SAS<sup>®</sup> Users Group International Conference, 2005.  
<http://www2.sas.com/proceedings/sugi30/219-30.pdf>
- [FullStimer] SAS Institute, (2005), *fullstimer SAS option*,  
<http://support.sas.com/rnd/scalability/tools/fullstim/fullstim.html>  
<http://support.sas.com/rnd/scalability/tools/fullstim/logparse.zip>

**Author: Ronald Fehd**      <mailto:Ronald.Fehd@cdc.hhs.gov>  
**Centers for Disease Control MS-G23**      **e-mail: RJF2@cdc.gov**  
**4770 Buford Hwy NE**  
**Atlanta GA 30341-3724**      **bus: 770/488-8221**

about the author:	
education:	B.S. Computer Science, U/Hawaii, 1986 SUGI attendee since 1989 SAS-L reader since 1994
experience:	programmer: 20+ years data manager at CDC, using SAS: 18+ years author: 10+ SUG papers
SAS-L:	author: 3,000+ messages to SAS-L since 1997 Most Valuable SAS-L contributor: 2001, 2003

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

**Document Production:** This paper was typeset in L<sup>A</sup>T<sub>E</sub>X. For further information about using L<sup>A</sup>T<sub>E</sub>X to write your SUG paper, consult the SAS-L archives:

<http://www.listserv.uga.edu/cgi-bin/wa?S1=sas-l>  
 Search for :  
 The subject is or contains: LaTeX  
 The author's address : RJF2  
 Since : 01 June 2003