

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

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 logparse citeSASlogparse and passinfo macros. The rtrace log can be read with Raithel's rtracrd citeSASlogparse macro.

This paper examines the issues of building a SAS usage database by modifying the above macros to provide common keys 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 .

Contents

Database Theory	2
The LogParse Suite	2
The Rtrace Suite	2
Accuracy of DateTime	2
Modifying SASv?.cfg	3
Modifying Autoexec.sas	4
Modifying PassInfo	4
Modifying LogParse	5
Modifying RTRACERD	6
A Demonstration Suite	7
Merging Time and Product Usage	8
Conclusion	9

INTRODUCTION

Executing a SAS® program produces several outputs. The log file contains real and cpu time used of each step as well as the total time of the job. Using the LogParse suite of programs, we may extract these job times and build a SAS Time Usage Database. 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 citeRaither-Rtrace we may extract the list of products and build a SAS Product Usage Database. We could then answer questions such as:

This paper examines the issues of linking time and product usage so that we could then answer questions such as:

- 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?
- 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

In [1], the authors state that there are three types of fact tables in databases: transactions, and two kinds of snapshots: periodic and accumulating. We seek to combine two sets of job transaction facts: product usage and job times used. The obvious primary key for the join is the job datetime.

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

Database Table Information	
fact tables:	transaction periodic snapshot accumulating snapshot
transaction:	foreign keys unique row identifier facts
keys:	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.

THE LOGPARSE SUITE

There are two routines in the logparse cite?SAS-logparse? suite: passinfo, which writes a header to the job log with various keys 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. This information includes a datetime, which we will examine closely for accuracy.

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

THE RTRACE SUITE

Raithel[2] 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 rtracrd, which reads the job product usage log. And last is pcfldttm, a subroutine of rtracrd, which reads the file-closed date and time and returns it to rtracrd which uses them to calculate the job time used.

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

ACCURACY OF DATETIME

Go to the Windows DOS prompt and type in: time; you will 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:
```

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

```
autoexec-Raithel:  datetime19. 09MAY2006:08:41:59
passinfo:         datetime16. 05may06:09:02:08
accurate only to whole seconds.
```

```

----- autoexec-Raithel.sas -----
1  /*rtracbegin*****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  /*rtracend******/

```

Compare the filename of `rtraceloc` with `passinfoX.sas`, snip 4, line 54, below.

```

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

```

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

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

This is the method I will use 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 -----
54  DateTime<41D5CE7968978D50>      hex16
55  DateTime<1463412130.000>      number
56  DateTime<16MAY2006:15:22:10.00> formatted

```

In the following sections, I will 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 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.

```

1  -config          'C:\Program Files\SAS Institute\SAS\V8\SASv8.cfg'
2  -SET      ProjRoot 'C:\SASsite\LaTeX\ModPassInfo'
3  -autoexec      '!ProjRoot\sas\autoexec.sas'
4  -rtrace        all
5  -SET      rtraceloc '!ProjRoot\rtrace'
6  -rtraceloc      '!rtraceloc\00RtraceDefault.log'
7  -SASinitialFolder '!ProjRoot\sas'
8  -fullstimer /* show user- and system-cpu time */
9  -nosplash /* no display SAS banner at startup */
10 -noovp /* no overprint errors nor warnings */

```

This configuration file was written specifically for this test suite. Note that all folders (directory-specifications) — `autoexec`, line 3, `rtraceloc`, line 5, and `SASinitialFolder`, line 7 — are child folders of the parent `ProjRoot`.

To follow Raithel's suggestions of a site-wide `autoexec`, then change the `-autoexec` file-specification, line 3.

Note the environment variable `rtraceloc`, line 5. This is necessary so that the folder can be changed in the configuration file, and not hardcoded in `passinfoX.sas`; see snip 3, line 64.

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.

Macro `logparse` is designed to read time blocks written by option `fullstimer`. This option is discussed by Raithel [citeRaithel-logparse](#).

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 [citeCarpenter](#) and [citeFehd2004](#).

```

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:

```

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

```

Macro `passinfo` can be simplified in several ways. The first is to replace the datetime conversion with a macro variable allocation. DateTime is the key which will be 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
38  PASS HEADER BEGIN
39  PASS HEADER os=WIN
40  PASS HEADER os2=WIN_PRO
41  PASS HEADER host=RONALD-VHQPUEVI
42  PASS HEADER ver=8.02.02M0P012301
43  PASS HEADER datetime=41D5CE79684F1AA0
44  PASS HEADER parm=
45  PASS HEADER memsize=0
46  PASS HEADER sumsize=0
47  PASS HEADER sortsize=2097152
48  PASS HEADER SysUserId=Administrator
49  PASS HEADER END 16MAY2006:15:22:09.00

```

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.

```

_____ logparseX.sas snip 1 _____
61      attrib UserId length = $ 16;                                %*logparseX;
62      length logfile $ 200 stepname $ 20

```

Add the variable UserId to the retain list.

```

_____ logparseX.sas snip 2 _____
128      retain UserId                                %*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 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 Print data = Library.Rtrace(keep = DateTime Product);'
12 ; stop; run;
```

```
_____ RtraceRdXtestcall.sas _____
1 %RTRACERDx(PRODFILE = !ProjRoot\sas7b
2 ,RTRACEFL = !RtraceLoc\Administrator_41D5CE7968C84189.log );
3 Proc Print data = Library.Rtrace(keep = DateTime 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	41D5CE7968C84189	core

Compare the hex16 DateTime in RtraceRdXtestcall.lst, line 6, with test.log snip 1, line 43.

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 %logparseX(saslog = test.log
2 , outds = Library.LogParseXtest);
3
4 PROC Print data = Library.LogParseXtest
5 (where = (StepName eq 'SAS')) noobs;
6 var DateTime RealTime UserTime SysTime Memused;

test.log snip 1

43 PASS HEADER datetime=41D5CE7968C84189
44 PASS HEADER parm=
45 PASS HEADER memsize=0
46 PASS HEADER sumsize=0
47 PASS HEADER sortsize=2097152
48 PASS HEADER SysUserId=Administrator
49 PASS HEADER END 16MAY2006:15:22:11.00

Compare the DateTime-stamp in test.log snip 1, line 49, with logparseXtest.lst, which is rounded to seconds.

test.log snip 2

112 NOTE: The SAS System used:
113 real time 1.32 seconds
114 user cpu time 0.14 seconds
115 system cpu time 0.72 seconds
116 Memory 2846k

Listing logparseXtest.lst shows the time log transaction.

logparseXtest.lst

	datetime	realtime	usertime	sysTime	memused
4					
5					
6	16MAY06:15:22:11	0:00:01.320	0:00:00.140	0:00:00.720	2846

MERGING TIME AND PRODUCT USAGE

Now I have two data sets each containing the transaction values from the two logs: time and product usage, and the single key, DateTime.

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 Print data = MergeTimeRtrace;

MergeTimeRtrace.lst

Obs	realtime	datetime	product	
4				
5				
6	1	0:00:01.320	16MAY06:15:22:11	core

CONCLUSION

Program `MergeTimeRtrace` demonstrates that the join is possible. However there are several problems 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 problems yet to solve but I have demonstrated the proof of the concept.

REFERENCES

- [1] Ralph Kimball, Margy Ross, (2002). *The Data Warehouse Toolkit, The Complete Guide to Dimensional Modeling, Second Edition*, John Wiley & Sons, Inc., New York.
- [2] Michael A. Raithel, (2004), *Measuring SAS® Software Usage on Shared Servers With the RTRACE Facility*, www2.sas.com/proceedings/sugi29/215-29.pdf
Proceedings of the 29th Annual Conference of the SAS Users Group, Int'l, 2004.
- [3] Michael A. Raithel, (2005), *Programmatically Measure SAS® Application Performance On Any Computer Platform With the New LOGPARSE SAS Macro*, www2.sas.com/proceedings/sugi30/219-30.pdf
Proceedings of the 30th Annual Conference of the SAS Users Group, Int'l, 2005.
- [4] 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**

Document Production: This paper was typeset in $\text{\LaTeX}$. For further information about using $\text{\LaTeX}$ 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

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.

about the author:

education:	B.S. Computer Science, U/Hawaii,	1986
	SUGI attendee	since 1988
	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
