SESUG Paper270-2018

Synchronized tracking of dataset versions with programs and logs using Proc PRINTTO and other SAS® tricks

Laxminarayana Ganapathi
Center for Health Data Analytics, RTI International, RTP NC 27709.

ABSTRACT

It is common to generate several versions of output data in a production environment. Quite often, the most important information, the program and its version that generated the data is documented manually. We describe an automated process that tracks each data set with a specific version of the code as well as the log file generated in the specific run. We will also demonstrate other uses of proc PRINTTO in tracking the versions of the production runs to make meaningful documentation of code and data in a production environment. Use of automatic macro variables generated by the system increases the reliability of versioning and audit trails. The process is suited for tracking minor changes implemented during a production run. It implicitly takes care of documenting any changes in the inputs to the program. Hence changes affected by running the same code in a changed environment can be dynamically documented. We will demonstrate the use of the process in a Linux as well as windows environment.

INTRODUCTION

Literature has several examples of using PROC PRINTTO for generating customized reports and outputs [1 -3]. Hughes [3] has published a thorough review of the importance and techniques in parsing the logfiles. These works have emphasized the need to go beyond the default outputs and log files we get from SAS. SAS generated reports and logs have been shown to be more meaningful and useful when properly customized for a specific need or task at hand. This further renders any work to be more accountable and reliable. Though one is familiar about the regulatory requirements in pharmaceutical industry, Quality Assurance in general, is being demanded in all work domains. In today's data driven environment, where data is considered a priced commodity, a good documentation and accountability of data is a must. SAS is a choice of several researchers across all domains for data analytics. Analytics typically involves program execution with incremental modifications in the code to test hypotheses or explore new insights. Such progressive development of SAS programs falls outside the realm of production software and norms. A fully version-controlled software development and maintenance may be too expensive and unproductive especially in a research environment. Here, we explore the possibility synchronizing key metadata and program logs with each SAS dataset dynamically. PROC PRINTTO is used to synchronize dataset generation and the accompanying code, listing and logs. Further we recommend some dynamic techniques for selecting the filenames and labels that reduce the errors due to manual documentation.

DETERMINE THE SAS ENVIRONMENT

SAS is available on multiple platforms like Linux and Windows. Further, the licensing arrangement that is applicable to an organization will guide the use of multiple user interfaces like, SAS Studio and SAS Enterprise Guide. Some of the key macro variables available for manipulation, depend on the platform on which the SAS is running. Hence a dynamic control of the filenames requires the knowledge of the SAS environment. The behavior of batch mode and Linux environment is considered the same. This is reasonable as most of the batch jobs are run on environments with SAS installed on Linux servers or a

SAS Grid running on Linux clusters. For current discussion we have chosen to determine the SAS environment among the four most popular variations:

1. SAS Studio
2. SAS Enterprise Guide
3. DMA Process (Display Manager or PC SAS)
4. Batch Mode (Linux)

Table 1 depicts the values of various macro variables useful for our purposes.

Macro Variable	SAS Studio	SAS Enterprise Guide	DMS	Batch/Linux
_CLIENTAPP	**SAS Studio**	**SAS Enterprise Guide**		
SYSPROCESSNAME	Object Server	Object Server	DMS Process	**Program 'yourProgram.sas'**
_SASPROGRAMFILE	**Full Program Path**	**Mapped Program Path**	-	-
_CLIENTTASKLABEL	-	Prefix of the program	-	-

Table 1. Value of Various macro variables in different modes.

The '_CLIENTapp' variable returns a value in case of SAS Enterprise Guide and SAS Studio. In the other two modes we get a return value from the 'SYSPROCESSNAME' variable. Combining the two sets, we have developed a macro to determine the SAS operating environment and store the value in a global macro variable called 'SASenv'. We also derive a filename string, '*LogFile*' unique to the time of execution of the program with a precision up to a minute. These two macro variables can be used anywhere in the program to customize the programming requirements. Herein we have derived the name of two filenames, one each for log and output based on a single macro variable *&logfile* dynamically based on the SAS environment.

GENERATING ENVIRONMENT SPECIFIC STRINGS FOR VARIABLE NAMES

One of the key concepts in our approach is the generation of valid filenames with time sensitive alphanumeric strings embedded. Automatic macro variables like *SYSDate* and *SYSDate9* have some limitations as they always return the string representing the start time of the current SAS session and hence will be misleading if used to differentiate multiple program executions during a long SAS session.

GENERATING TIME SENSITIVE UNIQUE STRING

We have used PROC FORMAT and picture option [4, 5] to generate a unique string which is able to discriminate time with a resolution of a minute. Table2 shows the various directives and their effect on the appearance of the string.

Directive	Description
%a	Abbreviated weekday name

%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%d	**Day of the month as a number (1-31)**
%H	Hour (24-hour clock) as a number (0-23)
%I	**Hour (12-hour clock) as a number (1-12)**
%j	Day of the year as a number (1-366)
%m	**Month as a number (1-12)**
%M	**Minute as a number (0-59)**
%p	**either AM or PM**
%S	Second as a number (0-59)
%U	Week number of the year (0,53)
%w	Weekday as a number (1= Sunday, 7)
%y	Year without century as a number (0-99)
%Y	**Year with century as a number**
%%	%

Table 2. Various directives and their description for datetime

We have used the directives %Y, %0m, %0d, %0I, %0M, and %p in combination to generate a macro variable *x* which can discriminate every minute:

```
proc format;
picture mydtfmt
  low-high = '%Y%0m%0d_%0I%0M%p' (datatype=datetime);
run;
%let x=%unquote(%sysfunc(datetime(),mydtfmt.));
```

An example of the string generated by the above code segment is given below:
201808**15_05**21**PM**
where directives %Y generated 2018, %om generated 08 for month August, %0d generated 15 for 15th day of the month, %0I generated 05 for the 5th hour in the 12-hour format, %0M generated 21 for the 21st minute and PM was generated by the directive %p.

MACRO TO DYNAMICALLY DERIVE APPROPRIATE FILENAMES

We have modified Tyndal's approach [6] to generate environment dependent global macro variables. Following is the macro called ***%sysCheck*** used to generate appropriate file names to synchronize the data and related files:

```
%global SASenv Logfile;

%macro sysCheck;
      /* macro to check the SAS environment */
      %if %symexist(_clientapp) %then
            %do;
            %if &_clientapp = SAS Studio %then
                  %do;
```

```
                    %Let SASenv = 1;/* Running SAS Studio;*/
                    %let logfile =
%sysfunc(compress("&_SASPROGRAMFILE._&x", ": '",""));
                %end;
                %else %if &_clientapp= 'SAS Enterprise Guide' %then
                %do;
                    %let SASenv = 2; /*SAS Enterprise Guide; */
                    %let logfile =
%sysfunc(compress("&outpath.&_CLIENTTASKLABEL._&x", ": '",""));
                %end;
            %end;
      %else %if %index(&sysprocessname,DMS) %then
            %do;
                %Let SASenv = 3; /* Running in Display Manager;*/
            %end;
      %else %if %index(&sysprocessname,Program) %then
            %do;
                %let prog=%qscan(%superq(sysprocessname),2,%str( ));
                %Let SASenv = 4; /* Running in batch Program &prog;*/
                %let logfile =
%sysfunc(compress("&outpath.&SYSPROCESSNAME._&x", ": '",""));
            %end;
%mend sysCheck;
```

After the above macro call, the global variables ***SASenv*** and ***Logfile*** are populated and available for use. In the following statement we create a macro variable '*createdby*' which can be used to set the label attribute of a data set:

```
%let createdby = %str(label=&logfile..log);
```

The proc printto step below directs the Log and the output of the SAS run to dynamically customized destinations:

```
proc printto log="&logfile..log" print="&logfile..txt";
run;
```

Table 3 shows the 3 different sets of file names generated in the three different SAS modes by the same program at three different times.

SAS Mode	**Name of the Log File**
SAS Studio	**/sasdata/home/lganapathi/**SESUG18_publish2.sas**_20180815_0521PM.log**
BatchMode	**/home/lganapathi/**ProgramSESUG18_publish2.sas**_20180815_0404PM.log**
SAS EG	**/home/lganapathi/**SESUG18_publish2_**20180815_0353PM.log**
SAS Mode	**Name of the Output File**
SAS Studio	**/sasdata/home/lganapathi/**SESUG18_publish2.sas**_20180815_0521PM.txt**

BatchMode	**/home/lganapathi/**ProgramSESUG18_publish2.sas**_20180815_0404PM.txt**
SAS EG	**/home/lganapathi/**SESUG18_publish2_**20180815_0353PM.txt**

Table 3. List of Log File names and Output file names generated by the same code in 3 different environments

The program SESUG18_publish2.sas provides the common string **'SESUG18_publish2'** in all the environments. The macro variable ***'&x'*** provides the date time value like **20180815_0521PM** which clearly associates the files to a specific run within a minute of accuracy. Note that if needed one can use the %S directive to discriminate with the second-precision.

In Table 4 we have highlighted the Label attribute generated by the same program on sample data set generated. Note that the label clearly associates the data set with the log file from the specific run. This enables researchers to quickly go to the correct log file and investigate any nuances one may find due to running a code which presumably did not change.

Output from SAS Enterprise Guide submission			
Data Set Name	GANA.TEST	**Observations**	19
Member Type	DATA	**Variables**	5
Engine	V9	**Indexes**	0
Created	########	**Observation Length**	33
Last Modified	########	**Deleted Observations**	0
Protection		**Compressed**	CHAR
Data Set Type		**Reuse Space**	NO
Label	**/home/lganapathi/**SESUG18_publish2_**20180815_0353PM.log**	**Point to Observations**	YES
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64	**Sorted**	NO
Encoding	latin1 Western (ISO)		
Output from batch submission			
Label	**/home/lganapathi/**ProgramSESUG18_publish2.sas**_20180815_0404PM.log**	**Reuse Space**	NO
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64	**Point to Observations**	YES
Encoding	latin1 Western (ISO)	**Sorted**	NO
Output from SAS Studio submission			
Data Set Name	GANA.TEST	**Observations**	19
Member Type	DATA	**Variables**	5
Engine	V9	**Indexes**	0
Created	########	**Observation Length**	33

Last Modified	########	**Deleted Observations**	0
Protection		**Compressed**	CHAR
Data Set Type		**Reuse Space**	NO
Label	**/sasdata/home/lganapathi/**SESUG18_publish2.sas**_20180815_0521PM.log**	**Point to Observations**	YES
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64	**Sorted**	NO
Encoding	latin1 Western (ISO)		

Table 4. Proc Contents Output of a sample data set generated by the same code in 3 different environments

SYSTEM OPTIONS AFFECTING THE LOG

SAS log files are at the core of Quality Assurance, as a detailed log file can contain the code as well as the notes on errors and warnings. By default, SOURCE option is enabled and will write the SAS code lines into the log. One can enable SOURCE2 option to capture all the codes including those from any secondary files included. MPRINT and MLOGIC are good option to capture MACRO related details in the log file. One can customize the details one would like to capture in the log file for quality assurance.

CONCLUSION

Proc format with picture option is used to generate unique strings with minute level precision to distinguish individual SAS runs. A macro is developed to identify the SAS mode and accordingly generate unique filenames for log and output. By writing the logfile information into the Label attribute of the data set, the data set and the log file corresponding to it are synchronized during the runtime. As a result, the programmer, analyst or any researcher will be able to easily trace the causes of any unique result associated with a specific run of program.

REFERENCES

1] Haworth, Lauren, March 16-19, 1997, Reports Based On Sas' Output: Taking Advantage Of Proc Printto, Data Steps And Proc Gprint, SAS Conference Proceedings: SAS Users Group International 22. San Diego, California. Available at http://www2.sas.com/proceedings/sugi22/ADVTUTOR/PAPER43.PDF.

2] Pahmer, Emmy, May12-15, 2013, Making the Log a Forethought Rather Than an Afterthought, Proceedings of PharmaSUG. Available at: https://www.pharmasug.org/proceedings/2013/TF/PharmaSUG-2013-TF05.pdf

3] Hughes, Troy Martin, Nov 5-7, 2017, Pinching Off Your SAS® Log: Adapting from Loquacious to Laconic Logs, To Facilitate Near-Real Time Log Parsing, Performance Analysis, and Dynamic, Data-Driven Design and Optimization, SAS Conference Proceedings: SouthEast SAS Users Group 2017, Cary, North Carolina. Available at: https://analytics.ncsu.edu/sesug/2017/SESUG2017_Paper-209_Final_PDF.pdf

4] Karp, Andrew H, March 26-29, 2006, Getting in to the Picture (Format), SAS Conference Proceedings: SAS Users Group International 31. Available at: http://www2.sas.com/proceedings/sugi31/243-31.pdf

5] Croghan, Carry W, Oct 31- Nov2, 2004, PICTURE Perfect: In depth look at the PICTURE format, SAS Conference Proceedings: SouthEast SAS Users Group, Nashville, Tennessee. Available at: https://analytics.ncsu.edu/sesug/2004/TU03-Croghan.pdf

6] Tyndall, Russ, Oct 21, 2016, Macro variables that provide information about your SAS® environment, SAS Blog. Available at: https://blogs.sas.com/content/sgf/2016/10/21/macro-variables-that-provide-information-about-your-sas-environment/

ACKNOWLEDGMENTS

I would like to thank Merry Rabb of RTI International for reviewing the document.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Laxminarayana Ganapathi
RTI International
lganapathi@rti.org

APPENDIX 1

Program ***SESUG18_publish2.SAS***:

```
options mprint fullstimer;



options mprint fullstimer;

libname gana '/home/lganapathi'; /* your path definition goes here */

%let outpath = /home/lganapathi/; /* your path for output goes here */
%global SASenv Logfile;
proc format;
picture mydtfmt
  low-high = '%Y%0m%0d_%0I%0M%p' (datatype=datetime);
run;
%let x=%unquote(%sysfunc(datetime(),mydtfmt.));

%macro sysCheck;
 /* macro to check the SAS environment. modified from ref [6] */
  %if %symexist(_clientapp) %then %do;
   %if &_clientapp = SAS Studio %then %do;
    %Let SASenv = 1;/* Running SAS Studio */
      %let logfile = %sysfunc(compress("&_SASPROGRAMFILE._&x", ":
'","""));

   %end;
   %else %if &_clientapp= 'SAS Enterprise Guide' %then %do;
```

```
    %let SASenv = 2; /* Running SAS Enterprise Guide; */
     %let logfile =
%sysfunc(compress("&outpath.&_CLIENTTASKLABEL._&x", ": '',""));

   %end;
  %end;

  %else %if %index(&sysprocessname,DMS) %then %do;
    %Let SASenv = 3; /* Running in Display Manager;*/
  %end;
  %else %if %index(&sysprocessname,Program) %then %do;
     %let prog=%qscan(%superq(sysprocessname),2,%str( ));
     %Let SASenv = 4; /* Running in batch and the program running is
&prog;*/
      %let logfile = %sysfunc(compress("&outpath.&SYSPROCESSNAME._&x",
": '',""));

  %end;
  %mend sysCheck;
 %sysCheck /* Capture the SAS mode / environment */


%let createdby = %str(label=&logfile..log);

proc printto log="&logfile..log" print="&logfile..txt";
run;

/* Begin Your code here in place of the following data step */
data gana.test(&createdby);
set sashelp.class;
run;


%put _all_;

 proc contents data=gana.test;
 run;

/* End of Your code */

proc printto;
run;
```