PharmaSUG 2016 – Paper MS04

QA and Compliance Insights Using the SCAPROC Procedure

Ben Bocchicchio, SAS Institute, Cary NC
Sandeep Juneja, SAS Institute, Cary NC

ABSTRACT

The SCAPROC procedure is a new procedure that runs a *SAS Code Analyzer*. This SAS Code Analyzer captures metadata information about the contents of the SAS code that is run. It collects information on files that are used in the code's input, the code's output, and macro variables used **while the code is running**.

How does this procedure relate to quality assurance (QA) and compliance? Just imagine if you could collect all the metadata about all the SAS code that is run to generate output for a FDA submission. You could programmatically prove that all the data referenced for input is consistently used, all macros called in the programs are consistent (without worrying about calling a generic macro when a project-specific macro is required), and determine that all designated output is saved to the correct location. Would this make you feel more confident about your submission? This presentation reveals these uses of SCAPROC. In addition, this presentation discusses other potential uses of this information.

INTRODUCTION

The SCAPROC procedure implements the SAS Code Analyzer, which captures information about input, output, and the use of macro symbols from a SAS program while it is running. The SAS Code Analyzer can write this information and the information that is in the original SAS file to a file that you specify. When developing SAS code for clinical trial analysis, ensuring the correct data sets are used as input is paramount to creating consistent output. The output can be either the submission data sets or the supporting Figures, Tables, and Listings. SAS programmers have developed various process to ensure that the correct data is chosen; however, issues can still arise. The PROC SCAPROC procedure can be another tool to help verify that SAS programs are producing consistent results. The output can be used to perform impact analyses. The PROC SCAPROC procedure first appeared in Release 9.2 of Base SAS® Software.

PROC SCAPROC SETUP

To produce the PROC SCAPROC output, a simple macro can be developed that only needs three inputs, the SAS program to analyze, the output Log location, and the location to store the SCAPROC results as shown in Figure 1.

Figure 1: Simple Macro for PROC SCAPROC Output

```
%macro sca_outp (programPath=, LogFile=, scaProcOutput=);

   proc printto log="&LogFile"; run;

   filename scafile "&scaProcOutput" mod;

   proc scaproc; write; run;

   proc scaproc; record scafile EXPANDMACROS; run;

   %include "&programPath";

   proc scaproc; write; run;

   proc printto log=log; run;

%mend sca_outp
```

This macro can be called from list of all SAS programs for a project or protocol.

PROC SCAPROC RESULTS

The results from running the PROC SCAPROC on a simple PROC PRINT SAS program generates the following information captured in Table 1.

Table 1. Textual Output from Using PROC SCAPROC

```
/* JOBSPLIT: JOBSTARTTIME 22MAR2016:07:23:02.21 */
/* JOBSPLIT: JOBSTARTTIME 30MAR2016:08:30:10.29 */
/* JOBSPLIT: TASKSTARTTIME 30MAR2016:08:30:10.31 */
/* JOBSPLIT: FILE INPUT C:\pharmaSUG2016\sca_work\program\test1.sas */
/* JOBSPLIT: DATASET INPUT SEQ NIC1.DEMO.DATA */
/* JOBSPLIT: LIBNAME NIC1 "C:\SDD_training\NICSAH_Study_001\Data" */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.DEMO1.DATA */
/* JOBSPLIT: LIBNAME WORK V9 'C:\Users\bebocc\AppData\Local\Temp\SAS Temporary
Files\_TD15384_L75022_' */
/* JOBSPLIT: SYMBOL GET PROGRAMPATH */
/* JOBSPLIT: SYMBOL GET SYSSORTDETAILS */
/* JOBSPLIT: SYMBOL GET SYSSORTTRACELEVEL */
/* JOBSPLIT: SYMBOL GET SYSSORTTRACE */
/* JOBSPLIT: ELAPSED 13  */
/* JOBSPLIT: SYSSCP WIN */
/* JOBSPLIT: PROCNAME SORT */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

   %include "&programPath";

/* JOBSPLIT: TASKSTARTTIME 30MAR2016:08:30:10.32 */
/* JOBSPLIT: DATASET INPUT SEQ NIC2.DEMO.DATA */
/* JOBSPLIT: LIBNAME NIC2 "C:\SDD_training\NICSAH_Study_002\Data" */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.DEMO2.DATA */
/* JOBSPLIT: LIBNAME WORK V9 'C:\Users\bebocc\AppData\Local\Temp\SAS Temporary
Files\_TD15384_L75022_' */
/* JOBSPLIT: SYMBOL GET SYSSORTDETAILS */
/* JOBSPLIT: SYMBOL GET SYSSORTTRACELEVEL */
/* JOBSPLIT: SYMBOL GET SYSSORTTRACE */
/* JOBSPLIT: ELAPSED 13  */
/* JOBSPLIT: PROCNAME SORT */
/* JOBSPLIT: STEP SOURCE FOLLOWS */


/* JOBSPLIT: TASKSTARTTIME 30MAR2016:08:30:10.33 */
/* JOBSPLIT: DATASET INPUT SEQ NIC1.ADVERSE.DATA */
/* JOBSPLIT: LIBNAME NIC1 "C:\SDD_training\NICSAH_Study_001\Data" */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.AE1.DATA */
/* JOBSPLIT: LIBNAME WORK V9 'C:\Users\bebocc\AppData\Local\Temp\SAS Temporary
Files\_TD15384_L75022_' */
/* JOBSPLIT: SYMBOL GET SYSSORTDETAILS */
/* JOBSPLIT: SYMBOL GET SYSSORTTRACELEVEL */
/* JOBSPLIT: SYMBOL GET SYSSORTTRACE */
/* JOBSPLIT: ELAPSED 12  */
/* JOBSPLIT: PROCNAME SORT */
/* JOBSPLIT: STEP SOURCE FOLLOWS *//* JOBSPLIT: FILE INPUT C:\Windows\Fonts\times.ttf */
ETC..
```

Once all the metadata for the program(s) is generated, a second program simply parses this information and creates a data set with all dependencies, including the type of use (input or output) with the file's path and name.   This is important for impact analysis since examination of the results may show that during the program execution, an input source data set could have been incorrectly selected, or the output destination of the results incorrectly stored.

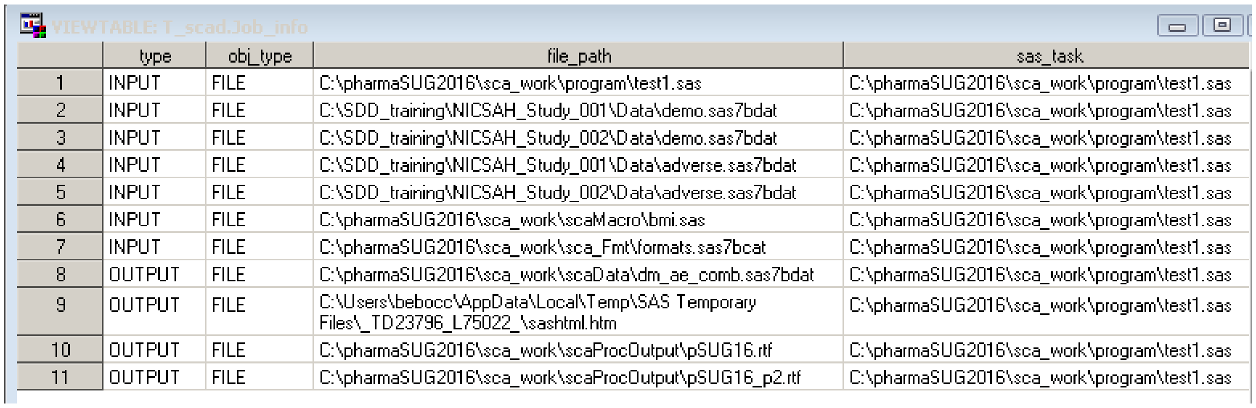
The parsing program is where the magic happens. It searches through the PROC SCAPROC output and extracts the necessary information for the impact analysis. The full parsing code is scheduled for SAS Technical Support review. Upon successful review, it is posted to the SAS Support site.

Table 2. Parsing Code Snippet

```
  . . .

  data mywork.scadata1;
        infile scafile lrecl=1000 length=linelength end=eof;
        input scaline $varying1000. linelength;

        * change / to \ for windows;
        * Ignoring SASTemp outputs;
        if index(scaline,"JOBSPLIT")>0 and index(scaline,"\sas_tmp\")= 0 then output;
      run;

      data mywork.scadata2;
        set mywork.scadata1;
        retain obj_type_pattern type_pattern lib_path_pattern xpt_libname_pattern
ttf_pattern;
        length type obj_type $200;

        if _n_=1 then do;
           type_pattern = prxparse("/(INPUT|OUTPUT|UPDATE|LIBNAME)/i");
           obj_type_pattern = prxparse("/:\s*\w+\s/i");
 * Change pattern to include all characters and spaces between quotes;
           lib_path_pattern = prxparse("/(\'|\"")+.*(\'|\"")+/i");
           xpt_libname_pattern = prxparse("/LIBNAME .* XPORT .*.xpt/i");
                    ttf_pattern = "/.*\.ttf/";
        end;


. . .
```

After parsing the PROC SCAPROC output, a SAS data set is produced. This data set contains details on the inputs to the code (even the code itself), macros used, format catalog used, and the outputs generated from the code. The resulting metadata data set is shown in Table 2 below:

Table 3. Resulting Data Set from Parsed PROC SCAPROC Output



	type	obj_type	file_path	sas_task
1	INPUT	FILE	C:\pharmaSUG2016\sca_work\program\test1.sas	C:\pharmaSUG2016\sca_work\program\test1.sas
2	INPUT	FILE	C:\SDD_training\NICSAH_Study_001\Data\demo.sas7bdat	C:\pharmaSUG2016\sca_work\program\test1.sas
3	INPUT	FILE	C:\SDD_training\NICSAH_Study_002\Data\demo.sas7bdat	C:\pharmaSUG2016\sca_work\program\test1.sas
4	INPUT	FILE	C:\SDD_training\NICSAH_Study_001\Data\adverse.sas7bdat	C:\pharmaSUG2016\sca_work\program\test1.sas
5	INPUT	FILE	C:\SDD_training\NICSAH_Study_002\Data\adverse.sas7bdat	C:\pharmaSUG2016\sca_work\program\test1.sas
6	INPUT	FILE	C:\pharmaSUG2016\sca_work\scaMacro\bmi.sas	C:\pharmaSUG2016\sca_work\program\test1.sas
7	INPUT	FILE	C:\pharmaSUG2016\sca_work\sca_Fmt\formats.sas7bcat	C:\pharmaSUG2016\sca_work\program\test1.sas
8	OUTPUT	FILE	C:\pharmaSUG2016\sca_work\scaData\dm_ae_comb.sas7bdat	C:\pharmaSUG2016\sca_work\program\test1.sas
9	OUTPUT	FILE	C:\Users\bebocc\AppData\Local\Temp\SAS Temporary Files_TD23796_L75022_\sashtml.htm	C:\pharmaSUG2016\sca_work\program\test1.sas
10	OUTPUT	FILE	C:\pharmaSUG2016\sca_work\scaProcOutput\pSUG16.rtf	C:\pharmaSUG2016\sca_work\program\test1.sas
11	OUTPUT	FILE	C:\pharmaSUG2016\sca_work\scaProcOutput\pSUG16_p2.rtf	C:\pharmaSUG2016\sca_work\program\test1.sas

Note: Usage Notes: SAS Itemstores, SAS Indexes, SAS database views (Oracle), web streams (PROC HTTPS, PROC SOAP), and inputs pulled from Java JAR files are not captured in PROC SCAPROC output.

IMPACT ANALYSIS

Several different impact analyses can be performed on the resulting data set. By sub setting the information in the 'type'' variable (input), and creating a new variables from the 'file_path' variable, you can easily produce a frequency of all data sets used in the protocol. The question of, "Have all the correct data sets been chosen?" can easily be answered. The same is true for the output files generated, you can answer the question of, "Has all the output been saved to the same, correct location".

You can perform other impact analyses to determine which programs must be re-run because of changes to individual data sets (due to a un-freeze of the data) or the updating of macros used in the code.

LIMITATIONS / CONSIDERATIONS

PROC SCAPROC has some limitations in the information it can gleam from *code analyzer* output. The procedure does not pull information about the following items that maybe used in SAS code:

- SAS Itemstores
- SAS Indexes,
- SAS database views (for example an oracle view),
- web streams (from Proc HTTPS or PROC SOAP),
- inputs pulled from Java JAR files.

To ensure all the output is collected correctly when using this procedure, consider running the SAS code in its own session. The justification is that any SAS macros used in your SAS code are copied into the Sasmacr catalog in your work library. If you use the same SAS session to run a program that uses the same macro, SAS pulls it from the work catalog and writes this information to the PROC SCAPROC output. The parsing code developed and tested for this paper drops all information from the work area, so knowledge of using this macro is lost. Therefore, it is recommended that this procedure should be run in Batch Submit.

CONCLUSIONS

In conclusion, the PROC SCAPROC is a very powerful procedure to conduct a real-time impact analysis that is easy to implement. Armed with this information, a wide array of analyses can be performed to ensure the correct data is used in the SAS code and the results are all stored in the correct location. Other insights can also be gleamed from this information such as macro and format catalog usage.

REFERENCES

Thies, Eric and Langston, Rick 2008 “Introducing the SAS® Code Analyzer” SAS *Global Forum 2008*

CONTACT INFORMATION

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

Name: Ben Bocchicchio
Enterprise: SAS Institute
Address: SAS Campus Drive
City, State ZIP: Cary, NC 27513
Work Phone: (919) 531-3704
E-mail: ben.bocchicchio@sas.com

Name: Sandeep Juneja
Enterprise: SAS Institute
Address: SAS Campus Drive
City, State ZIP: Cary, NC 27513
Work Phone: (919) 531-0541
E-mail: sandep.juneia@sas.com