# Proc SCAPROC – SAS Logging Unleashed

Andrew Beggs, Amadeus Software Limited


## ABSTRACT

Projects that utilise Base SAS® Software usually involve writing and maintaining groups of related SAS programs, with each one having a set of input and output files of various types. Often, the outputs from one program are the inputs to another, which means that when changes are made, they ripple through groups of programs, impacting on a number of other SAS jobs. Managing this will always require a disciplined approach, but with SAS 9.2 comes some help in the form of a new procedure called SCAPROC or the SAS Code Analyser, which is part of Base SAS Software. The procedure allows the programmer to generate an additional log file that specifically lists the inputs and outputs from each program step, allowing dependencies to be tracked. The Proc SCAPROC log files are structured and thorough in terms of the wide range of inputs and outputs that are recorded, which means that information can be programmatically gathered from them and used to help manage SAS source control code.

## INTRODUCTION

This paper describes the syntax of Proc SCAPROC, gives examples of the log files produced and demonstrates how thorough the procedure is at logging a wide spectrum of inputs and outputs. In addition, the paper compares SCAPROC logs to standard logs and demonstrates how a SAS program can be used to extract information from an SCAPROC log into a SAS data set. Although Proc SCAPROC can be used with the SAS grid job generator, this is outside the scope of the paper.

## SYNTAX OF THE PROCEDURE

The general layout and syntax of the procedure is as follows:

```
Proc SCAPROC;
  RECORD ext-fileref <ATTR><OPENTIMES>
                     <GRID ext-fileref <RESOURCE 'resource-name'>>;
run;

program code;
..;
..;

Proc SCAPROC;
  WRITE;
run;
```

**where:**

| | |
|---|---|
| ext-fileref | physical name or fileref to which SCAPROC output is to be written |
| ATTR | requests that additional attributes are output for data set variables |
| OPENTIMES | requests that access times and sizes of data sets and views are output. |
| GRID | specifies the name of a file to which output from the grid job generator is output. This is used when there is a requirement to split a job across multiple machines. |
| RESOURCE | specifies the name of the resource file to use when creating output for the grid job generator. |

## INVOKING IN BATCH MODE

Although Proc SCAPROC statements can be included at any point within a SAS program, it is neater to submit them via –INITSTMT and –TERMSTMT options on the command that is used to start SAS. The following example demonstrates this technique:

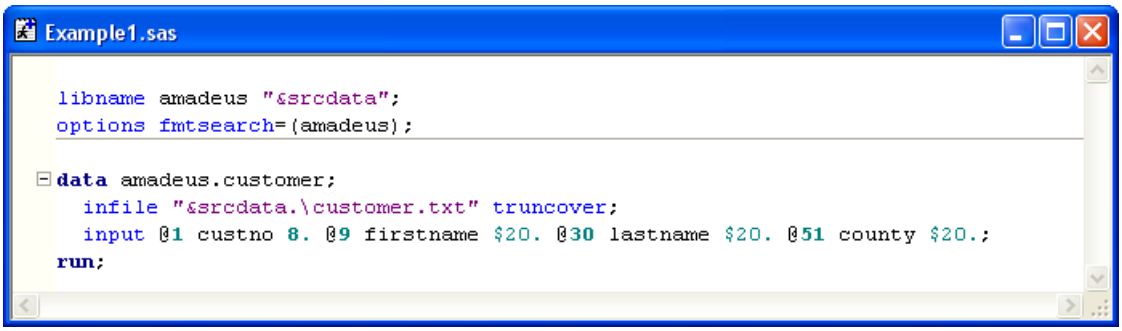
```
"C:\Program Files\SAS\SASFoundation\9.2\sas.exe"
-INITSTMT "proc scaproc; record 'c:\scaproc.log'; run;"
-TERMSTMT "proc scaproc; write; run;"
```

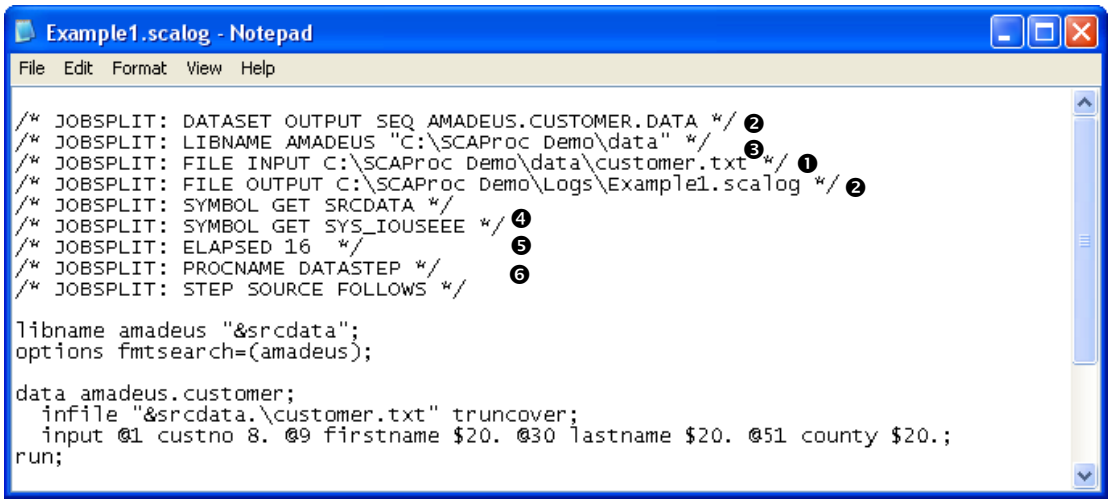
Statements specified with the -INITSTMT option are submitted after any that are present in an autoexec file and before any that exist in a batch file.

Statements specified with the –TERSTMT option are submitted when the SAS session ends.

## SAMPLES OF SCAPROC LOGS

The following screenshots show snippets of SAS code, together with the SCAPROC log that is generated when the code is run:




The SCAPROC log lists the source statements that have been executed, but precedes them with comments containing the JOBSPLIT keyword, which provide useful information on a range of items that include the following:

1. Names of Input files (external and SAS) ;
2. Names of Output files (external and SAS);
3. Paths relating to Libraries;
4. Macro variables used;
5. Elapsed time;
6. Step names.

The log generated by Proc SCAPROC contains comprehensive information about the SAS session's external interactions. It is important to note that the SCAPROC log is not intended to replace the standard log, but to complement it with additional information.

For example, with Proc GLM both the input data set ❶ and the output data set ❷ are listed:

Example2.sas

```
proc glm data=amadeus.music;
   model value=costprice;
   output out=amadeus.music_anlaysis stdp=stdpq uclm=uclm;
run;
```

Example2.scalog - Notepad

File Edit Format View Help

```
/* JOBSPLIT: DATASET INPUT MULTI AMADEUS.MUSIC.DATA */    ❶
/* JOBSPLIT: LIBNAME AMADEUS V9 'C:\SCAProc Demo\data' */
/* JOBSPLIT: DATASET OUTPUT SEQ AMADEUS.MUSIC_ANLAYSIS.DATA */ ❷
/* JOBSPLIT: LIBNAME AMADEUS V9 'C:\SCAProc Demo\data' */
/* JOBSPLIT: ITEMSTORE INPUT SASUSER.TEMPLAT */
/* JOBSPLIT: ITEMSTORE INPUT SASHELP.TMPLMST */ ❸
/* JOBSPLIT: DATASET UPDATE MULTI WORK._TF0006.UTILITY */ ❹
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: DATASET OUTPUT MULTI WORK._TF0007.UTILITY */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: DATASET OUTPUT MULTI WORK._TF0008.UTILITY */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: DATASET OUTPUT MULTI WORK._TF0009.UTILITY */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: FILE OUTPUT C:\SCAProc Demo\Logs\Example2.scalog */
/* JOBSPLIT: SYMBOL GET SYS_IOUSEEE */
/* JOBSPLIT: ELAPSED 1093  */
/* JOBSPLIT: PROCNAME DATASTEP */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

proc glm data=amadeus.music;
  model value=costprice;
  output out=amadeus.music_anlaysis stdp=stdpq uclm=uclm;
run;
```

The SCAPROC log also lists the item stores that have been accessed ❸, together with any temporary utility files ❹.

The standard log that results from running Proc GLM only gives details about the output data set:

Log - (Untitled)

```
73
74   proc glm data=amadeus.music;
75     model value=costprice;
76     output out=amadeus.music_anlaysis stdp=stdpq uclm=uclm;
77   run;

78

NOTE: The data set AMADEUS.MUSIC_ANLAYSIS has 682 observations and 17 variables.
NOTE: PROCEDURE GLM used (Total process time):
      real time           0.90 seconds
      cpu time            0.04 seconds
```

Note that with Base SAS procedures, the standard log will list both input and output data sets.

Even when an external file specification is derived, it is correctly reported on in the SCAPROC log ❶:

Example3.sas

```
data electric_sort;
   file "%sysfunc(pathname(amadeus))\electric.txt";
   set amadeus.electric_sort;
   cost=cost*1.15;
   put _all_;
run;
```

Example3.scalog - Notepad

File Edit Format View Help

```
/* JOBSPLIT: DATASET INPUT SEQ AMADEUS.ELECTRIC_SORT.DATA */
/* JOBSPLIT: LIBNAME AMADEUS "C:\SCAProc Demo\data" */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.ELECTRIC_SORT.DATA */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: FILE OUTPUT C:\SCAProc Demo\data\electric.txt */ ❶
/* JOBSPLIT: FILE OUTPUT C:\SCAProc Demo\Logs\Example3.scalog */
/* JOBSPLIT: SYMBOL GET SRCDATA */
/* JOBSPLIT: SYMBOL GET SYS_IOUSEEE */
/* JOBSPLIT: ELAPSED 266  */
/* JOBSPLIT: PROCNAME DATASTEP */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

libname amadeus "&srcdata";
options fmtsearch=(amadeus);

data electric_sort;
  file "%sysfunc(pathname(amadeus))\electric.txt";
  set amadeus.electric_sort;
  cost=cost*1.15;
  put _all_;
run;
```

When code is included, the name of the input macro file is shown in the SCAPROC log ❶, along with the name of the output catalog entry to which the compiled version has been written ❷:

Example4.sas

```
%include "&root.\plotvars.sas";

%plotvars(dsname=amadeus.music,v1=store,v2=value,outfile=music);
%plotvars(dsname=amadeus.electric,v1=month,v2=units,outfile=electric);
```

Example4.scalog - Notepad

File Edit Format View Help

```
/* JOBSPLIT: FILE INPUT C:\SCAPROC Demo\plotvars.sas */ ❶
/* JOBSPLIT: CATALOG OUTPUT WORK.SASMACR.PLOTVARS.MACRO */ ❷
```

In the above example, the included macro contains the following statements:

PlotVars.sas

```
%macro plotvars(dsname=, v1=, v2=, outfile=);
   ods listing close;
   ods html file ="&outpath.\&outfile..html";
   goptions device=jpeg;

   proc gplot data=&dsname;
     plot &v1 * &v2;
   run;
   quit;

   ods html close;
   ods listing ;
%mend;
```

The macro outputs plots to ODS destinations, which are listed in the SCAPROC log ❶, together with the names of the output GRSEG entries ❷:

Example4.scalog - Notepad

File Edit Format View Help

```
/* JOBSPLIT: FILE OUTPUT C:\SCAPROC Demo\Output\music.html */ ❶
/* JOBSPLIT: DATASET INPUT MULTI AMADEUS.MUSIC.DATA */ ❸
/* JOBSPLIT: LIBNAME AMADEUS V9 'C:\SCAProc Demo\data' */
/* JOBSPLIT: CATALOG INPUT #C00001.DEVICES.JPEG.DEV */
/* JOBSPLIT: LIBNAME #C00001 V9 'C:\Program
Files\SAS\SASFoundation\9.2\core\sashelp' */
/* JOBSPLIT: FILE INPUT C:\WINDOWS\Fonts\saswcur.ttf */ ❹
/* JOBSPLIT: FILE INPUT C:\WINDOWS\Fonts\saswalr.ttf */ ❹
/* JOBSPLIT: CATALOG UPDATE WORK.GSEG.GMASTER.GRSEG */ ❷
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: FILE INPUT C:\WINDOWS\Fonts\arialbd.ttf */ ❹
/* JOBSPLIT: FILE INPUT C:\WINDOWS\Fonts\arial.ttf */ ❹
/* JOBSPLIT: CATALOG OUTPUT WORK.GSEG.G3AGCD8O.GRSEG */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: CATALOG UPDATE WORK.GSEG.G3AGCD8O.GRSEG */ ❷
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: CATALOG OUTPUT WORK.GSEG.GMASTER.GRSEG */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: CATALOG INPUT WORK.GSEG.G3AGCD8O.GRSEG */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: CATALOG INPUT WORK.GSEG.GMASTER.GRSEG */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
Temporary Files\_TD3224' */
/* JOBSPLIT: FILE OUTPUT C:\SCAPROC Demo\Output\electric.html */ ❶
/* JOBSPLIT: DATASET INPUT MULTI AMADEUS.ELECTRIC.DATA */ ❸
/* JOBSPLIT: LIBNAME AMADEUS V9 'C:\SCAProc Demo\data' */
/* JOBSPLIT: CATALOG INPUT AMADEUS.FORMATS.MONTHFMT.FORMAT */ ❺
/* JOBSPLIT: LIBNAME AMADEUS V9 'C:\SCAProc Demo\data' */
```

Names of input data sets ❸, true type fonts ❹ and format entries ❺ are also shown.

## COMPARING SCAPROC LOGS TO STANDARD LOGS

As previously mentioned, the SCAPROC log is not designed to replace the standard SAS log. Instead it is intended to provide more structured and comprehensive information about the inputs and outputs of SAS programs, at a level of detail that is not available in a standard log. For example, a standard SAS log does not report on the physical file locations used when creating a report, but instead lists files as they are seen by the SAS session in terms of librefs and filerefs. There are also certain procedures such as GLM, for which the standard log does not show the name of the input data set.

Because the emphasis of SCAPROC logs is about recording information on inputs and outputs, the standard SAS log is still the key source of information relating to syntax, performance and all other aspects of program execution.

The following is a list of the broad types of input and output that are tracked in a Proc SCAPROC log:

- Library references (from SAS 9.2 TS2M0);
- Input and output data sets from DATA steps and procedures;
- Input and output external files and devices defined through various mechanisms including:
  - Statements such as INFILE, FILE and FILENAME
  - Functions such as PATHNAME
  - Code referenced via %INCLUDE statements.
- Macro variables
- Catalog entries for Informats, Formats, Macros and Graphics files
- ODS destinations
- Item stores
- Temporary and utility files

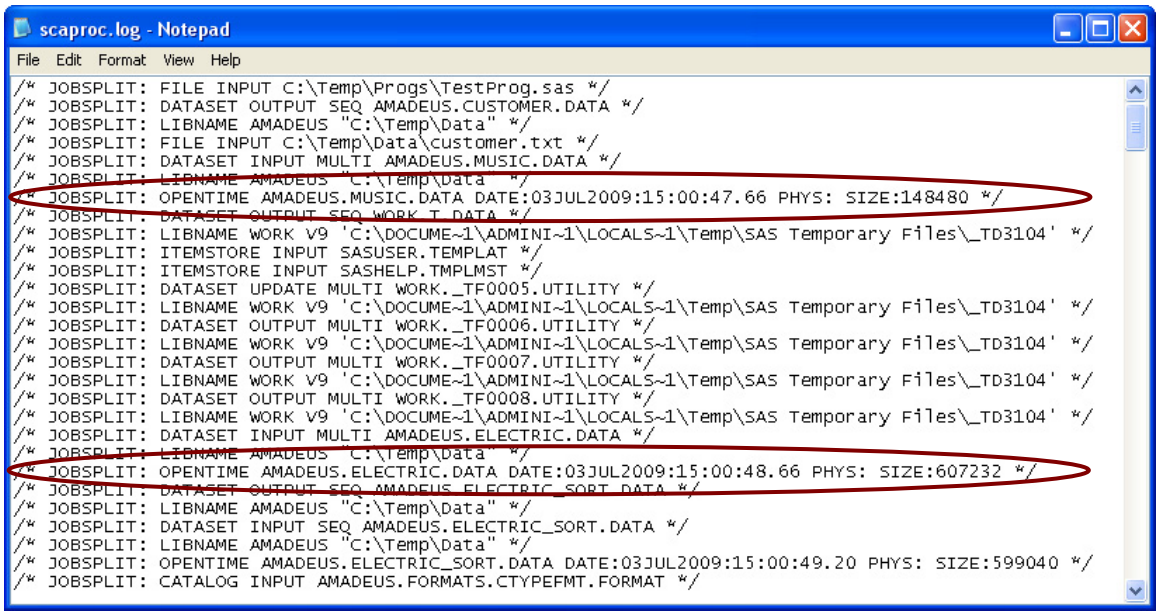
Other types of data source can also be tracked including FTP downloads, websites etc.

In a standard log, only some of the above are reported on and even then, it is not always consistent and provides far less detail.

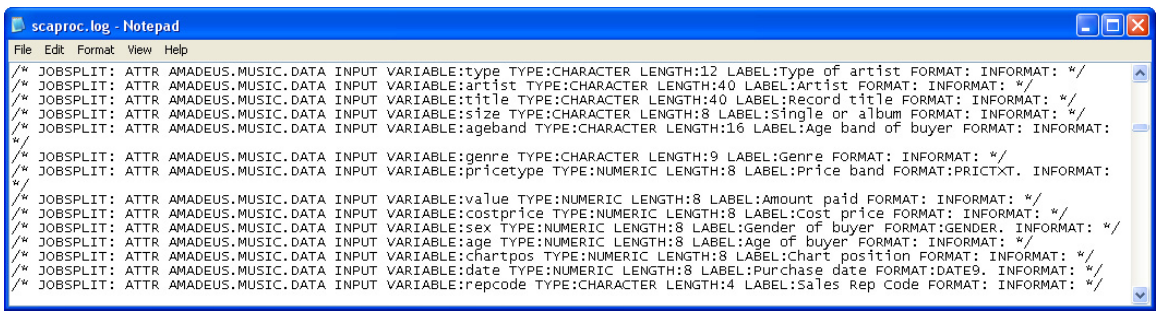
Having the ability to record the wide range inputs and outputs of a SAS job, means that it is easier to track the dependencies that exist between programs and therefore, be in a better position to manage the submission and updating of code. Combining this information with that from a standard log, gives the programmer a much fuller picture of what is happening with one or more SAS jobs.

## ADDING FURTHER DETAIL TO PROC SCAPROC LOGS

If the OPENTIMES option is used on the RECORD statement of Proc SCAPROC, then the last access time of a data set or view can be output in the SCAPROC log. If this information is programmatically read into a table, then there is the potential to write code which raises warning flags against a data set when it has been modified after the program that creates it has been run. The following excerpt from an SCAPROC log includes OPENTIME information:




Including the ATTR option on the RECORD statement outputs the attributes of each data set variable, which can be useful if there is a need to track changes to a data set in situations where it is updated or rewritten several times.



Although the SCAPROC log allows far more information to be gleaned about the inputs and outputs of a SAS job when compared with a standard log, it does not include details relating to processing and performance, such step timings, index usage, error, note and warning messages. For this, the standard log is still the best source and therefore, it's worth capturing both types of log.

## APPLICATION OF SCAPROC LOGS

SCAPROC logs provide the information that is needed in order to manage SAS source control code, in terms of the validity of programs, their inputs and outputs.

Once the log has been captured, SAS programs can be used to extract specific information from it. The following example shows a DATA step being used to read an SCAPROC log in order to output two data sets listing the input and output files respectively:

Reading an SCALOG.sas

```
filename scalog 'C:\SCAPROC Demo\Logs\Example4.scalog';

data input_files output_files;
  infile scalog truncover;
  input @1 logline $256.;
  keep filename;
  retain prx_input_file prx_output_file;
  if _n_=1 then do;
    prx_input_file =  prxparse("!\bJOBSPLIT: FILE INPUT (\b.*\b) \*/!");❶
    prx_output_file = prxparse("!\bJOBSPLIT: FILE OUTPUT (\b.*\b) \*/!");
  end;                                                    ❷     ❸
  if prxmatch(prx_input_file,logline) > 0 then do;❹
    filename=prxposn(prx_input_file,1,logline);❺
    put t=;
    output input_files;
  end;
  else if prxmatch(prx_output_file,logline) > 0 then do;
    filename=prxposn(prx_output_file,1,logline);
    output output_files;
  end;
run;
```

Perl regular expressions are defined in order to match log lines to the specific text strings which relate to input and output files ❶. In the two PRXPARSE expressions, the sub-expression that appears in parentheses ❷, matches the actual file name by specifying the longest string that exists with word boundaries at both ends, which lies between FILE INPUT or FILE OUTPUT and */. Note that the "*" has to be preceded by an escape character "\" ❸. This form of regular expression allows file names to include embedded blanks

The PRXMATCH function looks for a match between the regular expression and the line of the log file, returning a value greater than zero when a match is found, which corresponds to the position ❹.

The PRXPOSN function returns the component of the input string that matches the sub expression ❺. The value of '1' used for the second parameter refers to the specific part of the match that is to be returned, which in this case is the sub-expression element. If a value of zero is specified, then the whole string is returned.

The resulting data sets are as follows:

VIEWTABLE: Work.Input_files

| | filename |
|---|---|
| 1 | C:\SCAPROC Demo\plotvars.sas |
| 2 | C:\WINDOWS\Fonts\saswcur.ttf |
| 3 | C:\WINDOWS\Fonts\saswalr.ttf |
| 4 | C:\WINDOWS\Fonts\arialbd.ttf |
| 5 | C:\WINDOWS\Fonts\arial.ttf |

VIEWTABLE: Work.Output_files

| | filename |
|---|---|
| 1 | C:\SCAPROC Demo\Output\music.html |
| 2 | C:\SCAPROC Demo\Output\electric.html |
| 3 | C:\SCAProc Demo\Logs\Example4.scalog |


The above program can be extended in order to extract information on data sets, catalog entries, items stores etc. For further information on using Perl regular expressions, refer to the paper written by Bob Newman titled: “The Perl in the Crown – Regular Expressions in SAS”.

Having captured the information into data sets, reports can easily be generated, which should be of considerable help when it comes to managing groups of related SAS jobs.

## CONCLUSION

Proc SCAPROC is an effective means of tracking the inputs and outputs of SAS programs, which is a valuable asset when it comes to analysing dependencies between SAS jobs and auditing the inputs and outputs. The information output from Proc SCAPROC does not replace the standard SAS log, but complements it.

The OPENTIMES and ATTR options can be used to further enrich the SCAPROC log. Extracting the required information from the SCAPROC log can be achieved by using traditional SAS programming techniques, which can be made simpler and more robust through the use of string handling functions.

## ACKNOWLEDEGMENTS

The author wishes to thank David Shannon and Bob Newman of Amadeus Software for their help and advice in assembling material for this paper.

## CONTACT INFORMATION

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


| | |
|---|---|
| Author Name | Andrew Beggs |
| Company: | Amadeus Software Limited |
| Address: | Mulberry House, 9 Church Green, Witney, Oxon OX28 4AZ |
| Work Phone: | +44 (0) 1993 848010 |
| Email: | andrew.beggs@amadeus.co.uk |
| Web: | www.amadeus.co.uk |