

SAS® Programming Tips

Using SAS® to Automate Third Party Reporting Software

Mark MacMullen

MBNA Hallmark Information Services

Abstract

Have you ever used third party software in the environment where your data sets, jobs or procedures are kept? Typically this information is kept in more than one environment, which requires the use of two or three different scan types.

Perhaps your data is kept in one environment but you have to run multiple scans to arrive at the end result. An example would be to determine what batch jobs in your application system use a particular Cobol program. First, you must plug that program name into the first scan to find the procedures which use the program and then into another scan to discover which jobs call those procedures. Using the SAS® language to assist in "gluing" together those scans will greatly reduce your keyboard time and in creating such a utility, others may also appreciate the "Joy of SAS®".

Introduction

The utilization of the SAS® language in this situation will allow the user to exercise the application of several SAS® tools. These will include: DATA step (This was easy!), INFILE, KEEP, FILE, INPUT, the END= option, IF - THEN - ELSE logic, the RETAIN statement, OUTPUT, 2 counters, (one of which is reset after 10 iterations), the TRIM function, the MOD option in the FILE statement, Concatenation, and the PUT statement.

We will look at the use of the SAS® language in the context of dynamically building the JCL within SAS® for an ISPF SuperC scan. I also use this same SAS® code, with satisfactory results, with CA's Librarian® Fairscan and Endeavor® scan with minor alterations.

General Considerations

Each of the scans mentioned have their own operational rules. It is assumed that the reader is

familiar with these rules. Suffice it to say that based upon those rules for setup in the JCL, the SAS® code will be in your SYSIN DD statement as instream data or can be set up to run from a cntlcard referenced by the SYSIN DD statement DSN name.

Extracting the "hits" from a scan report will be different for each of the scans that I've mentioned. The SAS® code however, changes only in the first DATA step in the INPUT statement and in the IF statement.

Finding The "HITS" From The Report Output

In slide #6 of the PowerPoint presentation you will see the phrase "&&ACTION ELEMENT" and immediately after that, on the same line, one of the "HITS" for this Endeavor scan.

In slide #7 you will see the phrase "STRING(S) FOUND" and immediately before that, on the same line, one of the "HITS" for this ISPF SuperC scan.

In slide #8 you will see the phrase "ACCEPTED FOR SEARCH" and out to the margin toward the left of the page, on the same line, one of the "HITS" for this Librarian scan.

This is a good representation of the varying methods each of the reporting software has of pointing out that we have a winner!

So Many Rules And So Little Time

I've chosen the ISPF SuperC scan to work with in this paper because it offers so many opportunities to use the vast untapped riches of the SAS® language.

There are two caveats in using this scan in ISPF version 4.0:

1. In the release of ISPF version 4.0 you could not use the usual job step designation of JSnnn. You instead had to use "SEARCH". That changed with the release of version 4.2

2. In version 4.0 you were only allowed 10 searches per step. This also changed with the release of version 4.2. The total number is higher than 10 but I have not tested for the ceiling.

Watch carefully when a new release of the scan with which you are working is placed on your operating system. Sometimes it will limit you, requiring some handy SAS® code change for the dynamic JCL build portion of the code and sometimes it will allow you to do more with less!

Working With The SAS® Code

We will use the output from the Endeavor scan as our input for the next scan to run in ISPF SuperC. Look at slide #12, containing a sample of the Endeavor output, and compare it with the SAS® code from slide #13. Refer to the sample JCL on slide #10 when necessary. In the first DATA step we use pointer control and character formatting in the INPUT statement. We identify the two most critical pieces of information from the output report (the "HITS"). The rest of the output report is garbage and can be deleted. Use of the KEEP statement allows the temporary use of the SAS® data set in conjunction with the OUTPUT statement. Otherwise, we would have to create a temporary data set in the JCL to be read later in the SAS® code.

The second DATA step, (slide #14), begins to form the first part of the JCL that will be built upon later in the third DATA step. Notice the single PUT statement which creates the JCL in the output file OUTDD1 as referenced in slide #10.

The third DATA step, (slide #15) begins the fun part of this code. First we SET the temporary DATA set PRINTLIB and tag it with the END= option so that we will know when we reach the end of the input. This will come in handy later when we end the dynamic JCL in a manner that is syntactically correct for JCL. I've named the variable EOF, (End Of File). The MOD statement on the FILE statement will permit us to build on the OUTDD1 file which we have already started in the previous DATA step, without damaging any of the previous JCL code already written.

We will use two counters so that we are able to control the build of the JCL code. The first counter CNTR keeps track of the number of iterations of JNAME so that we do not go over

ten per step. The second counter increments the job steps by ten so that once the JCL is built we can view how many total "HITS" there were in the scan. Remember, IBM only allows 255 steps in JCL. Don't laugh, it's been done. The RETAIN statement allows us to keep the value of the counters and increment by 1 with each iteration until we choose to reset them. Keep in mind that SAS® automatically sets variables assigned values by an INPUT or assignment statement to "missing" before each iteration of the DATA step. Now, for the first iteration we set both counters to 1.

The "B" variable will give us the ability to append an apostrophe to the end of each JNAME variable so that we continue to play with SuperC according to its operational rules. (Each search string must be enclosed in apostrophes).

The next task is to create a new variable to rename the JNAME variable with the apostrophe attached. Keep in mind that the JNAME variable has a character format of eight but, sometimes the actual variable name is shorter than eight so we use the TRIM function to remove any trailing blanks before we concatenate the apostrophe to the end of the JNAME variable. The new variable name for the JNAME variable is now DSN. Remember, we've already built the JCL for the first step in the previous DATA step, up to the point of actually adding the DSN variable with the appropriate syntax for SuperC to recognize in the instream JCL. The PUT statement puts first things first by using pointer control. The literal string only SuperC understands (SRCHFOR and the first apostrophe) and the new variable DSN. *Viola!* That's one.

Recall if you will that the SAS® System executes programming statements in the exact order they are encountered. Now is the accepted time for us to end this dynamic JCL build if there is no more input,

ELSE we add each input JNAME until we reach the counter(CNTR) threshold of 10, ELSE if the counter is equal to 10, (slide #16), we reset the DSN variable counter to zero and assign the value of the other job step counter(JCLCTR) to a new variable name.

It's at this point that we continue to dynamically build the JCL with another PUT statement. Look carefully, don't confuse the JCL double hash marks with the SAS® hash mark in the PUT statement which commands the pointer to the next

line. Please notice the clever way this SAS® code increments the SEARCH steps by ten with the appropriate counter each time a new step is created in the JCL.

Slide #17 shows a sample of the resulting output, but running this bad boy is even more fun than looking at it.

Conclusion

This tutorial presents only one method of overcoming what at first appears to be a repetitive and mundane job of using third party software. Any SAS® programmer whether a beginner or expert can explore the untapped resources of ones imagination and the SAS® language in just such circumstances as these. Think of the utilities you can build! The joy of SAS® spread far and wide throughout the land.

Trademark Information

SAS is a trademark of SAS Institute Inc. in the United States and other countries. ® indicates USA registration.

To Contact The Author

Mark MacMullen
MBNA Hallmark Information Services
P.O. Box 809051
Dallas, Texas 75380-9051
972-701-6308
mmacmullen@juno.com

References:

SAS Institute Inc., *SAS® Language: Reference*, Version 6, First Edition, Cary, NC: SAS Institute Inc. 1990. 1042 pp.

SAS Institute Inc., *SAS® Procedures Guide*, Version 6, Third Edition, Cary, NC: SAS Institute Inc. 1990. 705 pp.

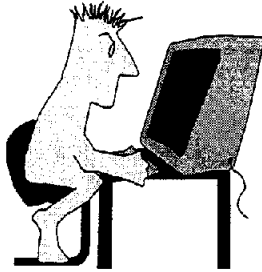
Cody, Ronald P., and Pass, Raymond, *SAS® Programming by Example*, Cary, NC: SAS Institute Inc., 1995. 337 pp.

Aster, Rick, *Professional SAS® Programmer's Pocket Reference*, Cary, NC: SAS Institute Inc., 1993. 188 pp.

Karp, Andrew K., (1996). SAS® System Programming Tips and Efficiencies: A Tutorial *Proceedings of the Sixth Annual South-Central Regional SAS® User's Group Conference*. SCSUG '96 Austin, Texas. pp. 119 - 126

S. David Riba, (1997) Your Friend The Colon: *Proceedings of the Seventh Annual South-Central Regional SAS® User's Group Conference*. SCSUG '97 Houston, Texas. pp. 193 - 194

SAS® Programming Tips



Using SAS® to Automate
3rd Party Reporting
Software

Mark MacMullen
MBNA Hallmark Information Services

8/3/98

1

Abstract of Tutorial

- Usually start with a scan from one environment and go to another
- Usually a predefined table/column layout
- Use the SAS® System to glue multiple scan types together in the JCL and visually appear seamless
- Multiple levels are possible

8/3/98

2

Abstract of Tutorial - continued

- Start with output from 3rd party software
- Capture the results with SAS®
- Format to the rules of the next scan to run
- Build the JCL structure dynamically
- Finally - submit those results from the command line of the file created by SAS®

8/3/98

3

Introduction

- SAS® Language Tools -
 - Data step - Infile - File - Input - End= - If, Then, Else logic - Retain - 2 counters - Trim - Mod - Concatenation - Put - Keep - Output
 - Dynamically build execution JCL
- The 3rd party software scans -
 - Endeavor
 - ISPF's SuperC
 - Librarian's Fairscan

8/3/98

4

Introduction - continued

- Separate scans
- One for each environment
- Use SAS®
 - create JCL dynamically
 - achieve integration between environments
 - one job processing instead of multiple jobs
 - create a utility usable to other analysts
 - spread the joy of using the SAS® Language

8/3/98

5

Endeavor Output Report

```
000037 1 (C) 1987,1996 Computer Associates International, Inc.
000038 ENDEAVOR EXECUT
000039 REQUESTED BY: EUCMIM
000040
000041
000042 SET FROM ENVIRONMENT PROD SYSTEM STANDARD
000043 TYPE COBOL STAGE NUMBER 2.
000044
000045 &&ACTION ELEMENT S063095B VERSION 01 LEVEL 02
000046
000047
000048 89: * CALLED ROUTINES:
000049
000050
000051 * 91: * CMA007AD - CARDHOLDER PROFILE I/O
```

8/3/98

6

ISPF SuperC Output Report

```
000001 1 SUPER C - MVS/PDF FILE/LINE/WORD/BYTE/SFOR COMPARE -
000002 LINE-# SOURCE SECTION SRCH DSN: PROD.CNTLC
000003
000004
000005 SSO093D2 STRING(S) FOUND
000006
000007 2 PRODS.SB126P.REJTM.SS157220(0) -
000008
000009 1 SUPER C - MVS/PDF FILE/LINE/WORD/BYTE/SFOR COMPARE
000010 SEARCH-FOR SUMMARY SECTION SRCH DSN: PROD.C
000011
000012 LINES-FOUND LINES-PROC MEMBERS-W/LNS MEMBERS-WO/
000013 1 482478 1 28410 1:50
000014
```

E3/98

7

Librarian Output Report

```
000025 1
000026 MODULE PSWD.LANG. RECORD INFORMATION
000027
000028
000029 CWF003DS DVVJ JCL *** ACCEPTED FOR SEARCH ***
000030
000031
000032
000033
000034
000035
000036 1
000037 MODULE PSWD.LANG. RECORD INFORMATION
000038
000039 *** JOB STATISTICS ***
```

E3/98

8

ISPF SuperC Caveat

```
//JS030 EXEC PGM=ISRSUPC,
// PARM=(SRCHCMP,
// 'ANYC')
//NEWDD DD DSN=EUCMIM.CNTLCARD,
// DISP=SHR
//OUTDD DD SYSOUT=L,OUTPUT=*.OUTZ
//SYSIN DD *
SRCHFOR 'AUTHHOLD'
SRCHFOR 'AUTHSHLD'
SRCHFOR 'CARDEXT1'
SRCHFOR 'CARDEXTS'
SRCHFOR 'CISLTRS'
SRCHFOR 'LETTEREX'
SRCHFOR 'LETTERS'
/*
//*
```

E3/98

9

JCL Example for In Stream SAS® Code

```
//JS010 EXEC SAS
//INDD1 DD DSN=TESTEU.EUCMIM.SCAN.ENDEAVOR.JOB,
// DISP=SHR
//OUTDD1 DD DSN=TESTEU.EUCMIM.JCLOUT,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,(1,1),RLSE),
// DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)
//SYSIN DD *
DATA PRINTLIB;
INFILE INDD1;
KEEP JNAME;
```

E3/98

10

- Our Example Scenario -

We are looking for all the jobs running in the Endeavor environment for our application area that will call a particular Cobol program.

The Cobol program in question is a subprogram and not used directly but called internally by other programs across all applications within our company.

We start our tutorial using an Endeavor scan focusing on our application area only. Using JCL supplied to us from the Endeavor Support personnel. The result of the Endeavor scan will be placed in a file and used in the next step by SAS® as an external file.

E3/98

11

Endeavor Output Report

```
000037 1 (C) 1987,1996 Computer Associates International, Inc.
000038 ENDEAVOR EXECUT I
000039 REQUESTED BY: EUCMIM
000040
000041
000042 SET FROM ENVIRONMENT PROD SYSTEM STANDARD
000043 TYPE COBOL STAGE NUMBER 2.
000044
000045 &&ACTION ELEMENT 5063095B VERSION 01 LEVEL 02
000046
000047
000048 39: * CALLED ROUTINES:
000049
000050
000051 * 91: * CMA007A0 - CARDHOLDER PROFILE I/O
```

E3/98

12

Reading the Endeavor file with SAS® - Step 1

```
DATA PRINTLIB;
INFILE INDD1;
KEEP JNAME;
INPUT @36 JNAME $8. @17 STRING $16.;
IF STRING = '&&ACTION ELEMENT' THEN
OUTPUT;
ELSE DELETE;
RUN;
```

1/3/98

13

The Dynamic JCL with SAS® - Step 2

Creating the first part of the Dynamic JCL in a file

```
DATA JCLREAD;
FILE OUTDD1;
PUT @1
"//SEARCH= EXEC PGM=ISRSUPC,PARM=(SRCHCMF,ANYCLPST) /*
"// CLASS=OMSGCLASS=OMSGLEVEL(1,1) /*
"// NOTIFY=SYSUD /*
"//ROUTE PRINT RMTG /*
"// /*
"//SEARCH= EXEC PGM=ISRSUPC,PARM=(SRCHCMF,ANYCLPST) /*
"//NEWDD DD DSN=PRODUCTION.PROCEDURE.LIB,DISP=SHR /*
"//OUTDD DD DSN=TESTEUCMIMTEMP.OUTPUT /*
"// DISP=(MOD,CATLG,DELETE) /*
"// SPACE=(CYL,(1,1),RLSE) /*
"// DCB=(LRECL=131,BLKSIZE=6,RECFM=FB) /*
"//SYSIN DD * /*
```

RUN;
1/3/98

14

The Dynamic JCL with SAS® - Step 3

Creating the second part of the Dynamic JCL in a file

```
DATA SETUP; SET PRINTLIB END=EOF;
FILE OUTDD1 MOD;
RETAIN CNTR 0 JCLCTR 0;
CNTR=CNTR+1;
JCLCTR=JCLCTR+1;
B="";
DSN=TRIM(JNAME)B;
PUT @1 "SRCHFOR " @11 DSN;
IF EOF THEN
PUT @1 " /*
"//";
ELSE
IF CNTR LT 10 THEN DO; RETURN; END;
```

1/3/98

15

The Dynamic JCL with SAS®

Step 3 continued

```
ELSE
IF CNTR EQ 10 THEN DO;
CNTR=0;
JCL=JCLCTR;
PUT @1 " /*
"// /*
"//SEARCH= @9 JCL
@13 "EXEC PGM=ISRSUPC,PARM=(SRCHCMF,ANYCLPST) /*
"//NEWDD DD DSN=PRODUCTION.PROCEDURE.LIB,DISP=SHR /*
"//OUTDD DD DSN=TESTEUCMIMTEMP.OUTPUT /*
"// DISP=(MOD,CATLG,DELETE) /*
"// SPACE=(CYL,(1,1),RLSE) /*
"// DCB=(LRECL=131,BLKSIZE=6,RECFM=FB) /*
"//SYSIN DD *; RETURN; END;
```

RUN;
1/3/98

16

Results of the Dynamic JCL build

```
//SEARCH= EXEC PGM=ISRSUPC,
// PARM=SRCHCMF,
// ANYCL
//NEWDD DD DSN=PRODUCTION.PROCEDURE.LIB,
// DISP=SHR
//OUTDD DD DSN=TESTEUCMIMTEMP.OUTPUT
//SYSIN DD *
SRCHFOR 'AUTHOLD'
SRCHFOR 'AUTHSHL'
SRCHFOR 'CARDI'
SRCHFOR 'CARD'
SRCHFOR 'CARDENT'
SRCHFOR 'CARDEN'
SRCHFOR 'CARDEF'
SRCHFOR 'CHLTR'
SRCHFOR 'LETTEREN'
SRCHFOR 'LETTER'
/*
//SEARCH= EXEC PGM=ISRSUPC,
// PARM=SRCHCMF,
// ANYCL
//NEWDD DD DSN=PRODUCTION.PROCEDURE.LIB,
// DISP=SHR
1/3/98
```

17

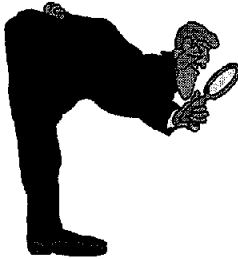
Conclusion

This tutorial presents only one method of overcoming what at first appears to be a repetitive and mundane job of using third party software. Multiple scans can be constructed in one job if necessary. We are limited only to our imagination and desire to learn.

1/3/98

18

QUESTIONS



ANSWERS
ANSWERS
ANSWERS
ANSWERS
ANSWERS
ANSWERS
ANSWERS

8/3/98

19

To Contact The Author

Mark MacMullen
MBNA Hallmark Information Services
P.O. Box 809051
Dallas, Texas 75380-9051
972-701-6308

8/3/98

20

Trademark Information

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries.

® indicates USA registration

8/3/98

21

REFERENCES

SAS Institute Inc., *SAS® Language Reference*,
Version 6, 1st Edition, Cary, NC
SAS Institute Inc 1990. 1042pp.

SAS Institute Inc., *SAS® Procedures Guide*,
Version 6, 3rd Edition, Cary NC
SAS Institute Inc 1990. 705pp.

8/3/98

22

Acknowledgments

Cody, Ronald P., and Pass, Raymond,
SAS® Programming by Example,
Cary, NC SAS Institute., 1995. 337pp.

Aster, Rick,
SAS® Programmer's Pocket Reference,
Cary, NC SAS Institute Inc., 1993. 188pp.

8/3/98

23

Acknowledgments

Karp, Andrew K., (1996) *SAS System
Programming Tips and Efficiencies: A Tutorial*
*Proceedings of the Sixth Annual South-Central Regional
SAS User's Group Conference*.
SCSUG '96 Austin, Texas. pp. 119 - 126

S. David Riba, (1997). *Your Friend The Colon:*
*Proceedings of the Seventh Annual South-Central
Regional SAS® User's Group Conference*.
SCSUG '97 Houston, Texas. pp. 193 - 194

8/3/98

24