SESUG 2020 Paper 142

Getting a Handle on All of Your SAS® 9.4 Usage

Stephen Sloan, Accenture

ABSTRACT

SAS is very popular, versatile, and easy to use, so it proliferates rapidly through an organization. It can handle systems integration, data movement, and advanced statistics and AI, has links to a large amount of file types (Oracle, Excel, text, and others), and has functionality to meet almost every need. In addition to SAS Base, there are a large number of specialized and special-purpose SAS products, some of the most popular of which are SAS STAT, SAS OR, SAS Graph, SAS Enterprise Miner, and SAS Forecast Server. SAS EG allows for quick creation of useful artifacts and then facilitates saving the generated code for later use as part or all of another program.

As a result of the above, sometimes it is difficult to track all of the SAS programs and artifacts being used across an organization, and economies of scale can be overlooked and repetition and "reinventing the wheel" sometimes take place. Programs and macros developed in one area can be useful in other areas and, as I have found when I share my programs, the programs and macros can be improved by this internal crowd-sourcing.

Understanding all of the places where SAS is used is also important when upgrading a system that makes heavy use of SAS, or when upgrading SAS itself to a new version like Viya. It can also help an organization identify which SAS products it is using and how much use these products are getting.

To accomplish the above, we've developed a set of programs to search a Unix server or a Windows server or machine to find, catalog, and identify the SAS usage on the machine.

INTRODUCTION

SAS is a very dynamic language and is continually improving its products. Therefore there may be PROCs, functions, and capabilities that are not currently available in SAS 9.4M6 in early 2020 that might have become available in a later release.

The purpose of this paper is to describe a mechanism for identifying SAS programs, classifying their structure, identifying their complexity, and compiling information about them. If that is the desire of the organization this paper will provide a road map. If the organization then desires to rationalize the use of the SAS software or prepare systems upgrades or conversion to SAS Viya, the next steps would be to dig into the programs for further analysis on an individual basis. The actual treatment of individual programs would proceed after the steps in this paper are completed.

CREATE A CATALOG THAT CLASSIFIES THE SAS PROGRAMS AND FUNCTIONS

First, the organization needs to decide how to classify its SAS usage. Some usage categories might be:

- Select, sort, and total
- Move data between systems

- Generate complex reports
- Advanced statistics and AI

Next, the organization will need to create a catalog that classifies the SAS PROCs and functions into the categories it has defined. I have included spread sheets at the back of this paper that list of all the SAS 9.4 PROCs and functions and identify the SAS-identified categories of the functions.

We then need to use the categories that the organization has defined and classify the PROCs and functions into those categories. We also need to determine how to classify programs that contain PROCs and functions in more than one category. It's often useful, although not required, to take the most complex category identified in a program and use that to classify the program. For example, if the program contains both basic PROCs and advanced High-Performance Function (HPF) PROCs, we should probably classify it according to the classification assigned to the HPF PROCs.

Sorting the programs into categories will be helpful when identifying resources to implement the conversions. The required skill sets can vary depending on the types of programs being converted.

NOW FIND THE SAS PROCS AND FUNCTIONS THAT ARE USED THROUGHOUT THE ORGANIZATION

Next we need to search through all the directories and sub-directories on the servers or drives and look for SAS programs. This can be done by using SAS to parse the programs, logs, macros, and other objects, and using Unix or Windows commands to locate the objects. The program code in the programs or logs will be the input to the SAS programs that we run. Through using the wide variety of text functions in SAS, the PROCs, functions, macro calls, and references to external files and programs can be identified and placed into categories.

The logs can be useful as a supplement to the programs because they identify the amount of space used and the amount CPU and real time taken by the programs. If the options like MACROGEN, MLOGIC, SYMBOLGEN, and MPRINT have been turned on the log can provide detailed information that may be useful in tracing the program's progress.

SAS PROGRAMS

The SAS programs will be of the form program_name.sas and the logs produced by these programs will be of the form program_name.log.

SAS ENTERPRISE GUIDE (EG) OBJECTS

The SAS EG outputs will require some iterative programming to locate.

- Look for .EGP files.
- Unzip the .EGP files
- Go through directories under .EGP files recursively to find SAS programs and logs

SAS ENTERPRISE MINER OBJECTS

SAS Enterprise Miner programs can be found as follows:

- Look for directories containing .EMP files. When someone uses Enterprise Miner, a file named PROJECT.EMP is produced.
- SAS programs and logs in the same directories or in subdirectories of that directory are related to Enterprise Miner calls. They will be under subdirectories whose name

starts with EMWS, with one EMWS subdirectory for each node on the Enterprise Miner analysis.

SAS MACROS

At this point, we might want to see programs or macros called by other programs in order to get frequency of use and relative importance. We can do this by looking for %INCLUDE commands in the programs, logs, and macros to find SAS code called by other programs, looking for %MACRO statements to identify macros, and then looking for the programs in which the macros are called.

- The name of the SAS program will follow the %INCLUDE statement
    - %INCLUDE program_name.sas;
- The macros used can be identified as follows:
    - %MACRO macro_name; identifies the macro being defined.
    - %MEND; identifies the end of the macro.
    - %macro_name is the call to the macro
- Also check for autocalls to macros. The autocall libraries can be identified in the MAUTOSOURCE option in an OPTIONS statement.

If we want to find all macro executions, we can look for words in the program that start with the % sign. However, there are also SAS commands that start with %. Therefore, we need to exclude those commands from our search. The SAS commands and functions that start with % are listed in the appendix. The autocall macros that start with % are not listed in the appendix because they contain logic that might need to be examined.

EXTERNAL REFERENCES

It will be valuable to our effort to identify the external files being read, updated, or created by the SAS programs. This will help pinpoint areas where the programs might be missed if not modified and the organizational importance of the files being written can help set priorities.

External databases can be identified by the following statements:

- LIBNAME statements can identify SAS data sets, Oracle tables, and other types of data bases
- CONNECT statements can identify non-SAS data sets like Oracle, Teradata, and Netezza tables
- FILENAME statements can identify sequential files and outputs like FTP and Email as destinations for the sequential files
- PROC IMPORT and PROC EXPORT can be used to read and write external files like Excel spread sheets
- INFILE and FILE can be used to read sequential files into at DATA step and to write sequential files from a DATA step.

We should now look through the programs and logs and identify all PROCs, functions, macro creations, macro usages, and references to external programs and files.

REFERENCES TO SAS DATA SETS

- The SET and MERGE statements can be used to bring SAS data sets into a DATA step.
- A DATA statement, which is the beginning of the DATA step, will identify one or more SAS data sets being created by the DATA step.

- A PROC step can create or update a SAS data set if it contains an OUT= parameter.

CLASSIFICATION

The final step is to use the categorizations mentioned above to classify the programs by what categories of PROCs and functions they used, both internally and through external references. This can be especially helpful if we want to rationalize the use of the programs and macros by having people use the same programs to accomplish the same ends. In these cases, we need to check the external references to make sure that we're not eliminating any valuable information.

Another attribute that helps reveal complexity, and perhaps repetitiveness, is the length of the programs and macros and being used and/or called by %INCLUDE statements. Longer programs can be parsed to see if they have pieces matching parts of other programs. The program length can be calculated either as the number of lines in a program or the number of words.

The size of the data sets being created can also be a classification variable if the programs that produce or modify larger data sets are deemed to be more crucial, or more in need of study when being replaced.

Now we can use the classifications defined by the organization to classify the PROCs and functions in each program. Depending on the algorithm that makes the most sense for the organization, the programs can then be classified and counts of the different types of programs can be compiled. Also, since the PROCs are identified by the SAS product, usage of the individual SAS products can be tracked.

CONCLUSION

When an organization has a multitude of SAS users and programmers who use SAS in a variety of processes, it can be difficult to gain an overall understanding of how SAS is used throughout the organization. By using the scripting language on the servers containing SAS implementations, we can get a high-level view of the SAS usage and drill down where necessary to find individual sections that might be relevant to our analysis. We can also find overlapping uses to drive efficiencies, identify infrequently used SAS objects, and make creative solutions available more widely within an organization.

CONTACT INFORMATION

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

Stephen B. Sloan
Accenture
Data Science Senior Principal
Stephen.b.sloan@accenture.com

APPENDIX

SAS PROCs .xlsx SAS Functions.xlsx

WINDOWS FILE EXTENSIONS AND THEIR ASSOCIATED MEMBER TYPES
http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#introsasfiles.htm

V6 File Extension	V7 and Beyond File Extensions	Short Extensions	SAS Member Type	Description
.sas	.sas	none	none	SAS program
.ss2	.sas7bpgm	ss7	Program	stored program (DATA step)
	.cfg (Version 8 and beyond)	none	none	configuration file
.lst	.lst	none	none	output file
.log	.log	none	none	log file
none	.sas7baud	st7	Audit	audit file
.sd2	.sas7bdat	sd7	Data	data set
.sv2	.sas7bvew	sv7	View	data set view
.si2	.sas7bndx	si7	Index	data set index. Indexes are stored as separate files but are treated by SAS as integral parts of the SAS data file.
.sc2	.sas7bcat	sc7	Catalog	SAS catalog
.sa2	.sas7bacs	sa7	Access	access descriptor file
.sf2	.sas7bfdb	sf7	FDB	consolidation database file
.sm2	.sas7bmdb	sm7	MDDB	multi-dimensional database file
none	.sas7bdmd	s7m	DMDB	data mining database file
none	.sas7bitm	sr7	Itemstor	item store file
.su2	.sas7butl	su7	Utility	utility file
.sp2	.sas7bput	sp7	PUtility	permanent utility
.stx	None	none	none	transport file
none	.sas7bbak	none	none	back up file

COMMANDS BEGINNING WITH % THAT ARE NOT SAS MACRO CALLS

%*
%’
%”
%(
%)
%%
%/
%\

%ABEND	%IF	%OPEN	%SYMLOCAL
%ABORT	%INC	%PAUSE	%SYSCALL
%ACT	%INCLUDE	%PUT	%SYSEVALF
%ACTIVATE	%INDEX	%QKCMPRES	%SYSEXEC
%BQUOTE	%INFILE	%QKLEFT	%SYSFUNC
%BY	%INPUT	%QKSCAN	%SYSGET
%CLEAR	%KCMPRES	%QKSUBSTR	%SYSLPUT
%CLOSE	%KINDEX	%QKTRIM	%SYSMACDELETE
%CMS	%KLEFT	%QKUPCASE	%SYSMACEXEC
%COMANDR	%KLENGTH	%QSCAN	%SYSMACEXIST
%COPY	%KSCAN	%QSUBSTR	%SYSMEXECDEPTH
%DEACT	%KSUBSTR	%QSYSFUNC	%SYSMEXECNAME
%DEL	%KTRIM	%QUOTE	%SYSPROD
%DELETE	%KUPCASE	%QUPCASE	%SYSRPUT
%DISPLAY	%LENGTH	%RESOLVE	%THEN
%DMIDSPLY	%LET	%RETURN	%TO
%DMISPLIT	%LIST	%RUN	%TSO
%DO	%LISTM	%SAVE	%UNQUOTE
%EDIT	%LOCAL	%SCAN	%UNSTR
%ELSE	%MACRO	%STOP	%UNTIL
%END	%MEND	%STR	%UPCASE
%EVAL	%METASYM	%SUBSTR	%WHILE
%FILE	%NRBQUOTE	%SUPERQ	%WINDOW
%GLOBAL	%NRQUOTE	%SYMDEL	
%GO	%NRSTR	%SYMEXIST	
%GOTO	%ON	%SYMGLOBL	