Paper 091-2010

Documenting SAS® Macros on a Server

Sumner Williams, CareOregon Inc., Portland, OR

ABSTRACT

Once a SAS macro is placed on a server for all users, the source code disappears and the comments which would help to determine how to use the macro also disappear. I propose writing two macros for each macro and calling a third macro. The first macro is the macro that a programmer would write no matter what. The second macro contains the documentation for the first macro. Finally, the third macro is a call to a macro which will deliver the source code to the programmer. This method of programming is already employed in open source projects as can be seen by calling the *man pages* for linux programs. More than likely, the company owns the source code and it does not matter that their employees can see the source code. It does matter that their employees do not know how to use the macro. This paper is intended for the macro programmer and does contain some more advanced concepts.

INTRODUCTION (HEADER 1)

This paper is actually a subset of the ideas presented at PDXSUG 2009 and PNWSUG 2009. Both of those papers expanded on how I get the documentation into the macro using the PERL language. This paper will simply focus on the benefits of having *callable documentation* enabled on macros.

Several macros are provided with each SAS license, but knowing how to use those macros depends on the programmer's ability to find the documentation. The documentation does not come with the macro, as far as I can tell. To alleviate this search, a second macro should be written named *<macroname>_usage*. The *<macroname>_usage* program will have documentation such as: programmer, creation date, modifiation date(s) and modifications, usage, purpose, variables, default values, and whatever else the programmer thinks is needed for documentation.

DOCUMENTING A SAS MACRO

The %example macro will demonstrate how I call for help when I forget how to use the macro.

```
%macro %example(help); ❶
        %if &help = ? | %upcase(&help) = HELP %then %do;❷
                %put ERROR: Help was requested.  ❸
                %put %example_usage; ❹
                %goto exit; ❺
        %end;

        SAS Code goes here

        %exit: ❻
%mend example;
```

It is best to give several options for calling help. ❶ is a positional, optional parameter. In this case, it is used to indicate if help will be called. The user is allowed to give a "?" or some case independent version of "help." ❷ tests for whether help was called. A programmer might wish to add in other required macro variables to this list. ❸ is some helpful text that will be printed to the log. The programmer can and should modify this to satisfy the end user. "ERROR:" will force the macro to bring up the log window when the user exits. ❹ calls a second macro that is the first macros name with "_usage" appended. ❺ uses a %goto statement to assign a macro variable telling SAS where to go next. **Goto** statements were popular in FORTRAN, but are not generally recommended in typical programming. This is one case where a **goto** is acceptable since it determines the end of the program. A line containing %exit: states where the code should **goto**. In this case the %exit: is added at the end of the macro. In the previously mentioned papers, I went into detail about using PERL scripts writing this section of the code.

EXTRA MACRO CODING

A second macro needs to be generated and it contains the documentation. The usage macro should include the programmer(s), dates of modification, purpose of the code, the manner in which it can be used, the variables, and the important output. Any additional requirements of the company should also be added, i.e. project identifiers, requestors, or other important information. When PERL is used to create the utilization macro, the columns line up and every line ends at the same point.

```
%macro example_usage;
%put ************************************************************;
%put *              Programmer: Sumner Williams                 *;
%put *              Modified: Created (6/5/2008)                *;
%put *              Macrotized (6/10/2008)                      *;
%put *              Added callable documentation                *;
%put *              Purpose: Demonstrate creation of SAS Macro with *;
%put *                       documentation                      *;
%put *              Usage: %nrbquote(%)example(help)            *;
%put *              Variables: help - option positional parameter. *;
%put ************************************************************;
%mend example_usage;
```

RESULTS

Instead of having this code:

```
data Example;
        infile "C:\Testing Folder\Example.csv" dsd;
        input claim \$ paid;
run;
proc sort data=Example;
        by claim;
run;
```

the programmer would have this code:

```
%macro Example_usage;
%put ************************************************************;
%put *          PROGRAMMER: Sumner Williams                     *;
%put *          MODIFIED: 20090504 Created                      *;
%put *          TRACK-IT: 0                                     *;
%put *          PURPOSE: This is a demonstration of creating    *;
%put *                   callable  documentation.               *;
%put *          USAGE: %nrbquote(%)Example%str(;)               *;
%put *          REQUESTNUMBER: 123456789-pi                     *;
%put ************************************************************;
%mend Example_usage;
```

```
%macro Example(help);

%if &help = ? | %upcase(&help) = HELP %then %do;
        %put ERROR:;
        %put Help has been requested.;
        %put Get help directly by running %nrbquote(%)Example_usage;
        %put Check the log for the help
        %Example_usage;
        %goto autoexit;
%end;

data Example;
        infile "C:\Testing Folder\Example.csv" dsd;
        input claim \$ paid;
run;
proc sort data=Example;
        by claim;
run;

%autoexit:
%mend Example;
```

CONCLUSION

Using the second macro allows for the documentation to be attached to the original macro and it allows for the same format to be used across macro documentation. I wrote a PERL program that generates the macro code so that all I have to do is write the program and the documentation. The PERL code puts in the %macro and %mend statements for both the documentation macro and the original macro, it neatly arranges all of the lines of code in the documentation macro, it puts in the code that tests for help, and it I puts in the goto statement. I only need to write the code and give the PERL code the text file containing the documentation.

ACKNOWLEDGMENTS

Ron Fehd “Macro Maven” has been helpful in his lectures and emails.

CONTACT INFORMATION

Contact the author:

Author: Sumner Williams
Company: CareOregon, Inc.
Address: 315 SW 5th Ave. Suite 900
Portland, OR 97204
Phone: (503)416-5710
work email: williamsu@careoregon.org
home email: shwilliams4@gmail.com