PharmaSUG 2013 - Paper AD15

A Macro to Batch Submit a List of Programs with Real Time Feedback

Andrew E. Hansen, Quintiles, Columbia, MO

ABSTRACT

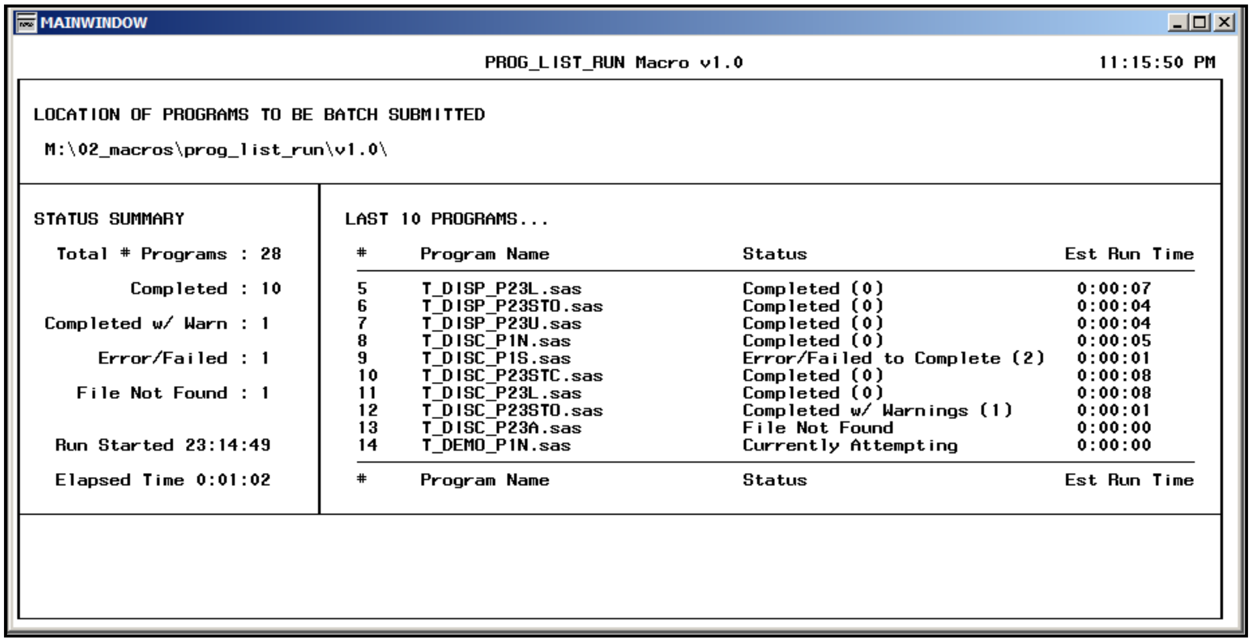
Once clinical projects are in the later stages of development or maintenance it is not uncommon to have the need to run a large group of programs due to new data or other updates. In recent years there have been multiple SAS papers authored by programmers in the pharmaceutical industry (Chen, Gilbert 2002; Shu 2006; Cawley, Prescod 2010; Sun, Wong 2010; Conover 2011) that describe methods and tools to automate this process. The purpose of this paper is to describe a macro tool for batch submitting a large number of SAS programs that is distinctly different from previous papers in that it provides real-time status feedback to the user as the programs are being executed.

INTRODUCTION

The tool described in this paper was first conceived during a large integration project that included more than 500 table programs which had to be rerun every time data from a new study was added. Certainly rerunning each of these programs individually would be too cumbersome and extremely inefficient, so some sort of batch process for mass submission had to be implemented.

Given the large number of programs and the length of time they took to complete (4+ hours total) a tool was desired which would also provide the user progress information about the batch run (which program is currently executing, how long it has been running, how many programs have already run, how many are left, etc.) in addition to simply running all the programs. Also needed was a way to flag programs which had encountered problems during the batch run so the issues could be corrected after the batch run was complete.

This was able to be accomplished by leveraging SAS %WINDOW and %DISPLAY macro statements and the SYSTASK COMMAND statement. Using these statements allowed us to create a macro tool which ran a list of programs and displayed real time progress and status information. Below is a screenshot of the macro in action.



```
MAINWINDOW

                         PROG_LIST_RUN Macro v1.0                          11:15:50 PM

LOCATION OF PROGRAMS TO BE BATCH SUBMITTED
 M:\02_macros\prog_list_run\v1.0\

STATUS SUMMARY            LAST 10 PROGRAMS...

  Total # Programs : 28   #    Program Name        Status                       Est Run Time

         Completed : 10   5    T_DISP_P23L.sas     Completed (0)                  0:00:07
                          6    T_DISP_P23STO.sas   Completed (0)                  0:00:04
 Completed w/ Warn : 1    7    T_DISP_P23U.sas     Completed (0)                  0:00:04
                          8    T_DISC_P1N.sas      Completed (0)                  0:00:05
      Error/Failed : 1    9    T_DISC_P1S.sas      Error/Failed to Complete (2)   0:00:01
                          10   T_DISC_P23STC.sas   Completed (0)                  0:00:08
    File Not Found : 1    11   T_DISC_P23L.sas     Completed (0)                  0:00:08
                          12   T_DISC_P23STO.sas   Completed w/ Warnings (1)      0:00:01
                          13   T_DISC_P23A.sas     File Not Found                 0:00:00
  Run Started 23:14:49    14   T_DEMO_P1N.sas      Currently Attempting           0:00:00

  Elapsed Time 0:01:02    #    Program Name        Status                       Est Run Time
```

Display 1. %PROG_LIST_RUN Status Window

Because of the complexity of the macro it is beyond the scope of a conference paper to describe the full macro, instead the focus will be on the few key steps that really drive this tool. The goal of this paper is to provide some insights that may help others create tools with similar functionality.

GENERAL OVERVIEW OF THE %PROG_LIST_RUN MACRO

Before using the %PROG_LIST_RUN macro, a SAS data set needs to be created containing a list of all the programs to be batch submitted. This could be done any number of ways depending on the structure and needs of the study. In practice these data sets have been created by importing a list from an Excel tracking document, using a macro that lists every SAS program in a directory, or even a simple list of datalines in a data step. All that is fundamentally required is that one of the columns in the data set is a list of program names.

To invoke the %PROG_LIST_RUN macro three simple parameters need to be passed to the macro: 1) the name of the data set with the list of programs, 2) the variable/column in the data set with the program names and 3) the folder/directory where the programs to be batch submitted are located.

```
%prog_list_run(dsn=work.proglist,
               progvar=progname,
               progdir=d:\programs)
```

Once the %PROG_LIST_RUN macro has been invoked a window pops up on the screen which displays the status of the current batch run. This status window is created using the %WINDOW and %DISPLAY macro statements since these are relatively easy to use and are part of Base SAS®. The use of the %WINDOW and %DISPLAY statements are not being detailed in this paper since they have been well-covered in the past. (Alden 2000, Mace 2002)

As the macro executes it cycles through the programs listed in the data set in order and attempts to batch submit each one. During this process the status screen is updated every five seconds or every time a batch process completes.

The outcomes of the batch submit attempts are saved in a SAS data set. This data set can then be used to create reports as needed to summarize the success or failure of the programs in the batch run.

Optional macro parameters can be used to change the behavior of the %PROG_LIST_RUN macro to do things such as playing a simple tune when the batch run completes, aborting the batch run if one of the programs fails to run/ends with errors, and automatically closing the status window when the batch run is complete.

CYCLING THROUGH THE LIST OF PROGRAMS

To cycle through the list of programs in the data set and pull the pull the program names into the macro environment the code below is used. This method uses SAS Component Language functions (OPEN, CLOSE, ATTRN, FETCHOBS, GETVARC, VARNUM) in conjunction with the %SYSFUNC macro function so they can be used in the macro language. (See papers by Yu 1998, Murphy 2007 and Nerren 2007 for similar examples.)

```
%let dsid = %sysfunc(open(&dsn(where=(&progvar ne " "))));
%if %eval(&dsid > 0) %then %do;
   %do i = 1 %to %sysfunc(attrn(&dsid,nlobsf));
      %let ob = %sysfunc(fetchobs(&dsid,&i));
      %if %eval(&ob = 0) %then %do;
         %let progname=%trim(%sysfunc(getvarc(&dsid,%sysfunc(varnum(&dsid,&progvar)))));

         %* {code to batch submit individual programs};

      %end;
   %end;
   %let rc = %sysfunc(close(&dsid));
%end;
```

STEPS WHEN SUBMITTING EACH INDIVIDUAL PROGRAM

Once a program name has been pulled from the program list data set and stored in the &progname macro variable the %PROG_LIST_RUN macro can begin the process of trying to batch submit the program. There are three basic steps in the process. First, the macro will verify that the program file actually exists in the specified directory. Next, the macro will attempt to batch submit the program using the SYSTASK COMMAND statement. Finally, the macro will wait for the batch process to complete before moving on to the next program in the list. However, while it is waiting the macro will continue to update the status window as to how long the current batch process has been running.

During these steps the outcome of the current batch submit attempt is stored in two macro variables: &status, which is a long text description of the outcome and &statcode, which is a corresponding short code.

VERIFY EXISTENCE OF PROGRAM

The first step for each batch submit attempt is to verify that the current program exists. If the program cannot be found then the macro stores a ".M" in the &statcode macro variable and it skips the next two steps.

```
%let rc=%sysfunc(fileexist(&progdir.&progname..sas));
%if &rc = 0 %then %do;
   %let status = File Not Found;
   %let statcode = .M;
%end;
```

BATCH SUBMIT THE PROGRAM

Each of the programs is batch submitted using the SYSTASK COMMAND statement similar to the following…

```
systask command "'d:\SAS92\SASFoundation\9.2(32-bit)\sas.exe'
        -sysin &progdir.&progname..sas
        -log &progdir.&progname..log
        -print &progdir.&progname..lst
        -noterminal -rsasuser -nosplash -nologo"
        status=taskrc ;
```

To ensure that the command itself executed properly the value of the %sysrc automatic macro variable is checked immediately after the attempt. It this has a value of anything other than zero it means that the attempt to batch submit the program failed and a return code will not populate the %taskrc macro variable.
If %sysrc is not zero, then the macro stores a ".S" in the &statcode macro variable to indicate a failure of the systask command and it skips the next step.

```
%if %eval(&sysrc ne 0) %then %do;
   %let status = Systask Error (&sysrc);
   %let statcode = .S;
%end;
```

One of the advantages of using the SYSTASK COMMAND statement to batch submit the program is the ability to capture a return code when it completes. The return code is stored in a user defined macro variable specified using the status option. In the example code above this is the macro variable &taskrc. Similar to the other outcomes this return code is stored by the macro in the &statcode variable.

The *SAS® 9.2 Companion for Windows* defines these returns codes as follows…

Condition	Severity	Return Code Value
All steps terminated normally	SUCCESS	0
SAS issued warning(s)	WARNING	1
SAS issued error(s)	ERROR	2
User issued the ABORT statement	INFORMATIONAL	3
User issued the ABORT RETURN statement	FATAL	4
User issued the ABORT ABEND statement	FATAL	5
SAS internal error	INFORMATIONAL	6
User issued the ABORT ABEND *nnn* statement	FATAL	*nnn*

Table 1. Return Codes on a Windows System

It is worth noting that it is also possible to have large negative numbers as return code. During the development of this macro a return code of -1073741819 was encountered, and neither the log or output listing files were modified in any way. Negative return codes are not mentioned in the documentation, so this was a good reminder to create code that was robust enough to handle the unexpected.

WAIT FOR RETURN CODE

By default the SYSTASK COMMAND statement is executed asynchronously. Even though the programs in the list are being batch submitting one after the other, we still need the asynchronous execution so our main program can continue processing statements updating our status window while that batch submitted program is running.

The code mechanism used to accomplish this is the loop below. The main program remains in this loop until it receives a return code back from the submitted batch process. The sleep function is included in the loop to prevent the main program from constantly using CPU resources checking for the return code. Instead the sleep function limits this to checking approximately once a second, which is more than sufficient.

```
%do %until("&taskrc" ne "");
   %* {code to update status window} ;
   %let rc = %sysfunc(sleep(1,1));
%end;
```

It is the ability to update the status window in the middle of this loop that allows the macro to provide us with nearly real time status information, such as how long the current batch process has been running.

An early version of the macro had the status window being updated with every pass through the loop, but because there is a slight flicker when the window is updated this was visually unappealing. The current version of the macro instead updates the status window every fifth pass through the loop, or approximately every five seconds.

After the return code (&taskrc) has been obtained it is it is collapsed into 4 categories and the &status and &statcode macro variables are populated accordingly. When combined with the values from the first two steps there wwill be the values in Table 2 for &status and &statcode at the end of the batch submit attempt.

&statcode Value	&status Value	Description
.M	File Not Found	Program file could not be found
.S	Systask Error (*&sysrc value*)	SYSTASK COMMAND statement failed to execute successfully / &sysrc ≠ 0
0	Completed (*return code*)	Program completed normally / &taskrc return code = 0
1	Completed w/ Warnings (*return code*)	Program completed, but with warning messages / &taskrc return code = 1
2	Error/Failed to Complete (*return code*)	Program ended with an error / &taskrc return code of 2+
3	Undefined Error (*return code*)	Other error

Table 2. &statcode and &status Values After Batch Submit Attempt

At this point the status information about this batch attempt is written out to our results data set, and the macro proceeds to the next program in the program list data set.

PUTTING IT ALL TOGETHER

Combining all these concepts together the macro is basically structured as the code below. This example is not a complete macro; notes in italics show where additional code would need to be added.

```
%macro prog_list_run(dsn=,
                     progvar=,
                     progdir=);

%* {insert code to validate parameter values};

%let dsid = %sysfunc(open(&dsn(where=(&progvar ne " "))));
%if %eval(&dsid > 0) %then %do;
   %do i = 1 %to %sysfunc(attrn(&dsid,nlobsf));
      %let ob = %sysfunc(fetchobs(&dsid,&i));
      %if %eval(&ob = 0) %then %do;
         %let progname = %trim(%sysfunc(getvarc(&dsid,%sysfunc(varnum(&dsid,&progvar)))));

         %* {insert code to display status window / update to change the current
             program to the value just read in from the program list data set};

         %let rc=%sysfunc(fileexist(&progdir.&progname..sas));
         %if &rc = 0 %then %do;
            %let status = File Not Found;
            %let statcode = .M;
            %goto output;
         %end;

         systask command "'d:\SAS92\SASFoundation\9.2(32-bit)\sas.exe'
              -sysin &progdir.&progname..sas
              -log &progdir.&progname..log
              -print &progdir.&progname..lst
              -noterminal -rsasuser -nosplash -nologo"
               status=taskrc ;

         %if %eval(&sysrc ne 0) %then %do;
            %let status = Systask Error (&sysrc);
            %let statcode = .S;
            %goto output;
         %end;

         %do %until("&taskrc" ne "");
            %* {insert code to update display window / update runtimes };
            %let rc = %sysfunc(sleep(1,1));
         %end;

         %if &taskrc = 0 %then %do;
            %let status = Completed (&taskrc);
            %let statcode = 0;
         %end;
         %else %if &taskrc = 1 %then %do;
            %let status = Completed w/ Warnings (&taskrc);
            %let statcode = 1;
         %end;
         %else %if &taskrc > 1 %then %do;
            %let status = Error/Failed to Complete (&taskrc);
            %let statcode = 2;
         %end;
         %else %do;
            %let status = Undefined Error (&taskrc);
            %let statcode = 3;
         %end;
         %output:
         {insert code to insert result of current
          batch attempt to result data set}
      %end;
   %end;
   %let rc = %sysfunc(close(&dsid));
   %put ;
%end;

%* {insert code to display final status window};

%mend;
```

CONCLUSION

Running a large number of programs manually can be very labor intensive, slow and inefficient. As a result it is common in the pharmaceutical industry to automate this process with some sort of tool. The concepts presented above can be used to create an efficient tool for Base SAS® that not only batch submits programs, but also displays real-time feedback for the user regarding the progress and success of the batch run.

REFERENCES

- Alden, Kay. 2000 "SAS' Best Kept Secret: Macro Windows® for Applications Development" Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference, paper 76.
- Cawley, Jim, Prescod Jillian 2010 "Need Reporting Deliverables the Painless and Easy Way? A Batch Submit Macro of Course!" Proceedings of the PharmaSUG 2010 Conference, paper CC10.
- Chen, Ling Y., Gilibet Steven A. 2002 "Run All Your SAS(R) Programs in One Program Automatically" Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference, paper 105.
- Conover, William 2011 "BAT Files: Run all Your Programs with One Click in PC SAS" Proceedings of the PharmaSUG 2011 Conference, paper PO01.
- Mace, Michael A. 2002 "%WINDOW: You Can Talk to the Users, and They Can Talk Back" Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference, paper 192.
- Murphy, Howard. 2007 "Changing Data Set Variables into Macro Variables" Proceedings of the 2007 SAS Global Forum, paper 050.
- Nerren, Nicholas. 2007 "Methods for Repeated Macro Calls" Proceedings of the PharmaSUG 2007 Conference, paper TT13.
- SAS Institute Inc. 2013 *SAS® 9.2 Companion for Windows* Cary, NC: SAS Institute Inc.
- Shu, Haibin 2006 "Highly Effective Batch Processing" Proceedings of the PharmaSUG 2006 Conference, paper AD09.
- Sun, Helen, Wong Cindy 2010 "A Macro to Create a Batch Submit SAS Program" Proceedings of the 2010 SAS Global Forum, paper 092.
- Yu, Hsiwei. 1998 "%SYSFUNC(BRIDGE TO EXTERNAL DATA)" Proceedings of the Northeast SAS User Group 1998 Conference, paper 35.

RECOMMENDED READING

- Cogswell, Denis. 2005 "More Than Batch – A Production SAS® Framework" Proceedings of the Thirtieth Annual SAS Users Group International Conference, paper 21.
- Furdal, Stanislaw. 2008 "Quick Windows Batches to Control SAS® Programs Running Under Windows and UNIX" Proceedings of the SAS Global Forum 2008, paper 17.

CONTACT INFORMATION

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

Andrew E. Hansen
Quintiles
6700 West 115th St
Overland Park, KS 66211
573.228.9140
573.234.6375
Andrew.Hansen@quintiles.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.