SESUG Paper 108-2023

# How to Prepare Precise and Intact Software Program for Submission

Linping Li, Merck & Co., Inc., Upper Gwynedd, PA, USA
Shunbing Zhao, Merck & Co., Inc., Rahway, NJ, USA

## ABSTRACT

Software programs are one of the components in the analysis package for e-Submissions submitted to regulatory agencies. Submission examples include Investigational New Drug Application (IND), New Drug Application (NDA), Abbreviated New Drug Application (ANDA), and Biologics License Application (BLA). The FDA Study Data Technical Conformance Guide states that “Sponsors should provide the software programs used to create all ADaM datasets and generate tables and figures associated with primary and secondary efficacy analyses”, and “The main purpose of requesting the submission of these programs is to understand the process by which the variables for the respective analyses were created and to confirm the analysis algorithms and results.”

In accordance with FDA’s Guidance, pharmaceutical companies need to submit software programs that do not contain any macro languages to ensure the precision and integrity of the submission. To meet this requirement, most pharmaceutical companies have a standard macro library that usually has the option to display the SAS statements from macro execution and route them to a new external file, thus, to generate the non-macro SAS program. However, if in a SAS program, some pre-processed SAS codes are added before calling the standard macro, integrating these codes with the non-macro program and ensuring that the combined program is both intact and accurate can be challenging. This paper will demonstrate how to use additional programs to address these issues.

## INTRODUCTION

In pharmaceutical companies and contract research organizations (CROs), it is very common to use company standard macro programs to generate tables, listings, and graphs. In these macro programs, SAS system functions MPRINT and MFILE, along with FILENAME statement, are used to display the SAS statements of the resolved macro codes and to route the SAS statements to an external file with .sas as the file extension. This way, the executable SAS program that does not contain any macro language is generated. However, there may be instances where additional SAS codes are required to pre-process the data before calling the standard macro program. As the pre-processed SAS code is not part of the standard macro, it will not be included in the external .sas file after executing the SAS program. Consequently, the resulting SAS program is incomplete and cannot be submitted to the regulatory agencies. This paper will introduce a solution to address these issues and ensure that a complete SAS program is generated and an accurate result is produced.

## PROCESS

### 1. Prepare the pre-processed code

Add system options MPRINT, MFILE, and FILENAME statement in the pre-processed code, and assign a temporary folder as the file path.

As those options and statement have been written in the standard macro already, two macro-free SAS programs are generated and outputted to the specified location after running the whole program. One contains the pre-processed codes, and another one contains the codes from executing the standard macro.

```
%let outloc = %str(C:\outlist);          ** Specified folder **;
%let outtmp = %str(C:\outlist\temp);     ** Temporary folder **;
```

```
*** Write codes to pre-process data, add MPRINT/MFILE options, and
output/write the SAS codes to a temporary folder ***;
%macro pre_tdemo;
   filename mprint "&outtmp\tdemo_pre.sas" lrecl=2048;
   options mprint mfile;

   data adsl;
     set adam.adsl;
     if country ne 'USA' then bygry = 'Non-USA';
     else if country eq 'USA' then bygrp = 'USA';
   run;

   options nomprint nomfile;
%mend pre_tdemo;
%pre_tdemo;

 ** Call the standard macro which creates the final statistical result and
 the table, and output/write the SAS codes to the temporary folder **;
%std_tdemo (indata=adsl,
            bygroup=bygrp,
            more macro parameters=,
            out_tab_name=tdemo,
            out_sas_file=&outtmp\tdemo_std.sas);
```

## 2. Create the macro-free SAS program

Firstly, we use INFILE and INPUT statements to read in the above two SAS programs separately from the temporary folder and output them as two temporary SAS datasets. In each dataset, define a SAS variable "sascode" as shown in codes below and assign the entire SAS program as its value.

```
** Read in the pre-processed SAS codes **;
data tdemo_pre;
  length sascode $5767;
  infile "&outtmp/tdemo_pre.sas" lrecl=8192 end=eof;
  input ;
  sascode=_infile_;
  output tdemo_pre;
run;

** Read in the SAS codes from execution of the standard macro **;
data tdemo_std;
  length sascode $5767;
  infile "&outtmp/tdemo_std.sas" lrecl=8192 end=eof;
  input ;
  sascode=_infile_;
  output tdemo_std;
run;
```

Then, we can combine the two temporary SAS datasets, and use FILE/PUT statements to route the combined SAS dataset to the specified folder with the desired SAS program name. By doing this, a SAS program containing both parts are created. Since the pre-processed code is set on top of the resolved code of the standard macro, the codes are in the order as we wanted without any sorting.

```
data _null_;
  file "&outloc/tdemo_sub.sas" lrecl=5767 nopad;
```

```
    set tdemo_pre(keep=sascode)
        tdemo_std(keep=sascode);
    put sascode;
  run;
```

3. **Validate the result**

The to-be submitted SAS program that contains the pre-processed codes and standard macro outputs the final statistical results to a permanent dataset prior to generating the formal report. To validate the results generated by the macro-free SAS program, we can copy it to a temporary folder and run the program. Then we can compare the final statistical results generated from the two programs. If the comparison result is equal, that means the macro-free program can duplicate the results and the program is intact.

```
** Re-assign the libname, so the macro-free program can output the final
result to the temporary folder **;
libname proddata "&outtmp";
%include "&outtmp/tdemo_sub.sas";

** Compare the final result generated from both programs **;
libname vald "&outtmp";
libname prod "&outloc";

proc compare base=prod.tdemo comp=vald.tdemo
             out=vald.chk_tdemo outnoequal noprint;
run;
```

## CONCLUSION

The overall approach involves incorporating MPRINT/MFILE options into the pre-processed code, utilizing INFILE/INPUT statements to read each macro-free SAS program as input and generate a SAS dataset as output, combining these datasets, and then using FILE/PUT statements to generate a complete SAS program. Lastly, the results are compared to ensure accuracy.

Using this approach, the submission-ready programs can be generated concisely and precisely. This will enable agencies to comprehend the process and trace the steps taken to create the variables for the respective analyses. It will also help them confirm the analysis algorithms and results.

## DISCUSSION

1. As we know that the resolved SAS codes from MPRINT have no blank lines and no indentation. How to make this macro-free SAS program more “prettier” and more readable? There are different ways to do this, one easy way is using SAS Enterprise Guide. You can open the program in SAS Enterprise Guide, right-click in the Program Editor, select Format code from the popup window. You can see that extra blank lines are added to the program, and the SAS codes are indented automatically.
2. We all know that it is a good practice to insert the comments in the SAS program. While using the codes above, we noticed that MPRINT option can’t display the comment lines starting with “/” or “%” contained in the macro program. If you want to keep the comment lines in the macro-free SAS program, start the comment line with ‘*’ in the macro program.
3. The programs have been further developed into two macro programs. One macro is used to combine the pre-processed code with the standard macro. Another macro is used to compare the final results. So all similar scenarios can be processed in the same way.

## REFERENCES

Dalton, Maria. “Submission of Software Programs to Regulatory Agencies.” PhUSE US Connect 2019

https://www.lexjansen.com/phuse-us/2019/sa/SA04.pdf

https://www.fda.gov/media/143550/download

SAS Help Center: Programming Documentation for SAS 9.4 and SAS Viya 3.5

## CONTACT INFORMATION

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

Linping Li
Linping.li@merck.com

Shunbing Zhao
shunbing.zhao@merck.com