Paper 179-2019

Pipe it! Extract it! Build it!

Smitha Krishnamurthy, Roche Molecular Systems, Pleasanton, CA

Sofia Shamas, MaxisIT Inc., Metuchen, NJ

ABSTRACT

A process improvement tool developed using FILENAME statement with PIPE option in SAS to create study related validation tracker with key information captured from individual SAS program headers within a folder. The validation tracker is an excel document with hyperlinks to the actual RTF output and serves as a guideline for the developer and the validator during validation. Additionally, this macro creates a validation issue document in excel where we verify accuracy of the program header by comparing details of the RTF outputs listed against the actual outputs created. This macro does require the user to follow a standard header structure across all codes and execute it as a batch process. The use of PIPE option limits the code use with SAS Enterprise Guide. Macro was run in SAS 9.4.

INTRODUCTION

Validation is a time and resource intensive process and enormous amount of deliverables need to be completed. Keeping track of all the deliverables and key details related to a user who either develops or validates a code is an ominous task for the lead programmer of the study. This macro is an automated, process improvement tool developed to help reduce time and effort by creating documents in excel with pre-populated columns from the header of SAS codes under validation.

In order to understand the functionality of the macro presented in this paper, user needs to be familiar with SAS PIPE option in the FILENAME statement and follow a standard header in every SAS program. PIPE option in SAS FILENAME statement is used to link a parent process with a child process. Using this feature current SAS session which executes the macro becomes the parent process and it reads data from the external SAS code which is now the child process. Another feature of PIPE option is to obtain a list of files within a folder and macro uses this functionality to read the SAS code names and the RTF file names.

This macro is written with the assumption that a standard header is present for every SAS code and has input, output and the author information listed as is shown in an example below.

Example of a standard header –

```
/**************************************************************************
Program:
Purpose:
Input:
Output:
Created by:
Modified by:
Notes:
**************************************************************************/
```

Macro is designed to handle a varied folder structure. To verify the accuracy of the header for the RTF outputs the RTF output location can be same as the SAS code or it can be saved in a different folder.

MACRO CALL

STANDARD MACRO CALL

```
%vallist (analyte=XXX, protocol=YYY, fname=ZZZ, tname=in-text, pgmloc=, rtfloc= outfile=YYY1);
```

SPECIAL CASE MACRO CALL

For cases where RTF output location is different from the SAS code location.

%***vallist*** (analyte=XXX, protocol=YYY, fname=ZZZ, tname=in-text, pgmloc=programs, rtfloc=in-text, outfile=YYY1);

PROCESS FLOW

For the process flow details please refer to the macro code attached as an appendix.

STEP 1: Using the macro input parameters create the Study Path and RTF Path.

The study structure is identified with macro input parameters which create the study path and RTF path. Study path refers to the location of SAS codes which usually is the same as RTF output location (rtf path). A different rtf or SAS code location is specified with optional macro parameters.

STEP2: Save as a SAS dataset list of RTF outputs.

Using filename statement with the PIPE option we create a fileref which points to the location where all rtf outputs are saved and then with an infile statement a SAS dataset is created with all rtf output name. Macro will compare this list of RTF output with the output names mentioned in the code headers and will point out issues if any.

```
filename rtflist pipe "dir ""%unquote(&rtfpath)"" /b /s " lrecl=32767;
```

STEP3: Save as a SAS dataset list of the SAS codes under validation.

SAS PIPE option is used to read the name of all the files at the location specified by the study path, we then subset to keep the SAS code files only. A macro variable is created with a count of the total number of SAS codes which would be a limit for running the do loop iterations.

STEP4: Convert SAS code into a SAS dataset

Using SAS PIPE option, the current SAS session will link to each individual SAS codes and read the entire code into a SAS dataset. As we follow a standard header it is possible to separate the header of the code from the rest of the code.

STEP5a: Extract information from the header

Using keywords such as ‘CREATED BY’ and ‘OUTPUT’ we extract from the header of the code key information such as the author and the name and number of outputs created and save it as macro variables. These macro variables map into columns of the final dataset created per code.

STEP5b: Stack to create the final dataset

Individual datasets per code are stacked to create the final dataset which is used as an input to create documents for validation.

STEP6: Compare the actual RTF output with RTF filenames specified in the header - Validation Issues Sheet

List of RTF outputs read from the output folder get compared with the RTF output names read from the final SAS dataset. All mismatches get saved in the validation issues excel sheet. Source of the mismatch is also identified, whether the code was missing the output name or if we cannot link an output to a SAS code.

STEP7: Save the final dataset – Validation Assignment Sheet

The final dataset from the headers of the code has all the information. We remove issues found in the header after printing them in validation issue excel file and save the clean records in the validation assignment tracker. Hyperlinks are added to make it easier to go to the actual location of the code.

LIMITATIONS AND CONCLUSION

The macro though flexible has a few limitations

• All SAS codes need to follow a standard header

• Macro cannot be executed in SAS Enterprise Guide

Efficient time saving tools are a great help to SAS programmers during the process of validation. The macro is written with the intent to make it easy for a programmer to create documents that track the validation progress and identify a few issues related to the code headers. It is a starting step and the code can be enhanced and a lot of other information can be extracted to be used by the programmer.

ACKNOWLEDGMENTS

We would like to thank Roche leadership team for their encouragement and their valuable insight in the development of the macro.

CONTACT INFORMATION

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

Smitha Krishnamurthy
Roche Molecular Systems.
925-730-8313
smitha.krishnamurthy.sk1@roche.com

Sofia Shamas
MaxisIT Inc.
925-730-8206
sofia.shamas@contractors.roche.com

APPENDIX

```
%macro vallist(analyte=, protocol=, fname=, tname=, pgmloc=, rtfloc=, outfile=);

*STEP1a: Error Message for Key macro variables;
%if %length(&analyte.)=0  %then %put "ERROR: Missing folder name";
%if %length(&protocol.)=0 %then %put "ERROR: Missing Protocol name";
%if %length(&fname.)=0    %then %put "ERROR: Missing folder name";
%if %length(&tname.)=0    %then %put "ERROR: Missing Task folder name";

*STEP1b: Derive Full Path;
%if %length(&pgmloc.)=0 and %length(&rtfloc.)=0 %then %do;
  %let path=\\server1\biowork\&analyte\&protocol\&fname\&tname;
  %let rtfpath=\\server1\biowork\&analyte\&protocol\&fname\&tname;
%end;
%else %do;
  %let path=\\server1\biowork\&analyte\&protocol\&fname\&tname\&pgmloc;
  %let rtfpath=\\server1\biowork\&analyte\&protocol\&fname\&tname\&rtfloc;
%end;

*STEP2: List of rtf output read from the rtf location saved in a SAS dataset;
filename rtflist pipe "dir ""%unquote(&rtfpath)"" /b /s " lrecl=32767;
data rtflist1;
  infile rtflist truncover;
  input rtfname $char1000.;
  put rtfname=;
run;
proc sort data=rtflist1; by rtfname; run;

*STEP3: Read the SAS Code names from the location specified by the &path macro variable;
filename tasklist pipe "dir ""%unquote(&path)"" /b /s " lrecl=32767;
data filelist;
  infile tasklist truncover;
  input filename $char1000.;
  put filename=;
run;
proc sort data=filelist;by filename;run;

data templist; *Subset for SAS code files with extension .sas to read code header;
  set filelist;
  if upcase(scan(filename,-1,'.')) in ("SAS") ;
  pgmpath=substr(filename, 1 ,length(filename)-length(scan(filename,-1,'\'))-1);
  extention = upcase(scan(filename,-1,'.')) ;
run;
proc sort data=templist sortseq=linguistic; by pgmpath pgnm; run;

data finallist; *cnttot has the number of SAS codes;
  set templist;
  cnt=_n_;
  call symput('cnttot', left(_n_));
run;

*STEP4a: Read SAS codes and save as SAS dataset;
%macro varglobal;*making all macro related to filenames global *;
  %do i = 1 %to &cnttot; filenm&I %end;
```

```
%mend;
%global % varglobal;

proc sql noprint; *save each sas code name into a macro variable using cnttot;
  select PGNM into :filenm1 through :filenm&cnttot
  from finallist;
quit;

%do j=1 %to &cnttot; *Iterate for each SAS Code;
  filename refsascd pipe "dir /b /s ""&path\&&filenm&j...sas""" ; *Read the entire code;
  data _all_code&j;
    length  file_name $ 100 string $ 200;
    infile refsascd truncover;
    input file_name $100.;
    put file_name=;
    infile dummy filevar=file_name end=done truncover;
   do while(not done); input string $char200.; output; end;
  run;

  data _all_code&j._1; *Get the endpoint of the code header;
    set _all_code&j;
    linenum=0;
    if index(upcase(string),'CREATED BY')>0 then linenum=_n;
  run;
  data templine_num;
    set _all_code&j._1;
    if linenum ne 0;
  run;
  data _null_;
    set templine_num;
    if _n_=1; call symput('linenum',linenum);
  run;

  *STEP4b: Subset to keep the header,extract key information;
  data header&j;
    set _all_code&j._1;
    if _n_ <= &linenum;
  run;

  *STEP5a: Extract key information from header;
  data _null_; *Code Author;
    set header&j;
    if index(upcase(string),'CREATED') >0;
    call symput("createdby", scan(string,2,':'));
  run;
  data header&j._2;*Output files from Code;
    set header&j;
    indxby=_n_;
    retain aoutpath;
    if index(upcase(string),'.RTF') >0;
    tmpstrng = catx('\',"&analyte","&protocol","&fname","&tname");
    outpath1= compress(substr(string, 1 , length(string) - length(scan(string,-1,'\'))-1));
    if index(upcase(string),'OUTPUT') >0 then outpath=outpath1;
    else if index(outpath1,'\')>0 then outpath=catx('',tmpstrng,outpath1);
    aoutpath=substr(outpath, index(outpath,"&analyte")); drop outpath1;
  run;
```

```
   proc sort data=header&j._2 out=header&j._3; by indxby; run;
   data header&j._3;
     retain  tmpapath;
     set header&j._3;
     by indxby;
     temphrtf=scan(string,-1,'\');
     hrtfnm1=compress(temphrtf,'09'X);
   run;
   data null;
     set header&j._3;
     call symput('hrtfcnt', left(_N_));
   run;
   proc sql noprint; select hrtfnm1 into: rtfnm1 - :rtfnm&hrtfcnt from header&j._3; quit;

   data exlout&j;
     set header&j._6;
     program_path="&path";
     if "&rtfloc" = "&pgmloc" then output_location="&tname";
     else output_location=catx('\',"&tname","&rtfloc") ;
     program_name="&&filenm&j...sas
     output_name=hrtfnm1;
     developer="&createdby";
     program_status=" ";
   run;
%end; *End of do loop;

*STEP5b: Stack datasets to create the final output;
data final;
   set %do k=1 %to &cnttot;
     exlout&k %end;
     ;*for set statement;
run;

*STEP6: Compare actual RTF outputs with the list created from the header of codes;
proc sort data=final sortseq=linguistic(numeric_collation=on);by col1;run;
proc sort data=rtflist sortseq=linguistic(numeric_collation=on);by col1;run;

data validation_issues; *Output 1: Validation Issues Excel;
   length comment $200;
   merge final(in=a) rtflist(in=b);
   by col1;
   if ^a then Comment='Header Isssue: RTF not listed in the header';
   if ^b then Comment='Output Issue: RTF not found in the folder';
   if (^a or ^b) then output Validation_Issues;
run;

proc export data=validation_issues label
         outfile="rtflocation\&outfile._Validation_Issues.xlsx" replace;
run;

*STEP7: Create the Validation Tracker Document;
proc sort data=validation_issues out=issues_  sortseq=linguistic(numeric_collation=on);
   by program_name;
run;
proc sort data= final out=final_1  sortseq=linguistic(numeric_collation=on);by program_name; run;
```

```
data pre_val_sheet(drop=comment) chk_val;
  merge final_1(in=A) Issues_(in=B rename=(col1=output_name));
  by program_name;
  length col1 $100.;
  col1=upcase(left(compress(output_name)));
  if B then output Chk_val;
  else output pre_Val_sheet;
run;
proc sort data=pre_val_sheet out=pre_val_sheet_  sortseq=linguistic(numeric_collation=on);
  by col1;
run;

data validation_assignment_sheet(drop=col1);
  merge pre_Val_sheet_(in=a) rtflist(in=b);
  by col1;
  if a;
run;

ods excel file="rtflocation\&outfile._Validation_Assignment_Sheet.xlsx";
proc report data=Validation_assignment_sheet nowd;
  compute program_path;
    call define(_col_, "URL", program_path);
  endcomp;
run;
ods excel close;

%mend vallist;
%vallist(analyte=XXX,protocol=XXX001,fname=final,tname=in-text,pgmloc=,rtfloc=,outfile=XXX001_);
```