PhUSE US Connect 2019

Paper CT07

Automated compliance checks on programs using SAS

Hrideep Antony, Syneos Health, Cary, NC, USA
Aman Bahl, Syneos Health, Ontario, Canada

ABSTRACT

This paper introduces a SAS macro to automate SAS program header checking of single or several program files in a directory. Macro also generates a summary report of all the issue findings in the header such as inaccurate folder path, inaccurate validation program name, missing key information such as programmer/validator names, specification information, misspelled words in the header etc. This macro also provides the independent validation programmer limited access to just the content of the header while rest of the programming content is unveiled.

INTRODUCTION

Program header serves as the key documentation that typically has information about the purpose of the program, the outputs generated by the program, version history, specification documents used for the programming and so on. Maintaining header accuracy is one of the key challenges for programmers since the header checks are usually done manually and independent program validator may not have access to the source program to ensure its correctness.
Having inaccurate header information can also trigger traceability issues, especially if the program has the wrong directory path, program/output names etc.

Automating the header check process using this macro that is provided in the appendix section of this paper will improvise the overall submission quality by reducing the number of audit findings that are related to traceability imprecisions in the header.
Inaccurate header information such as wrong source/validation program names and folder location path, missing programmer/validator names etc could be identified easily using this macro. In addition, the validation programmer also will have the provision to check the header while the rest of the program content is secluded.

FUNCTIONALITY

One of the key functional elements of this macro is checking the legitimacy of all the external file references in the header using the PIPE device-type along with filename statement. By using the PIPE device type on a FILENAME statement, DOS DIR command is invoked. This command will, in turn, return a value “dir: cannot access ***” if the file location is not legitimate in that directory.
SAS string functions are used to segregate the texts in the header that are possibly related to the file path. This can be challenging since some of these references can be split into multiple lines. This macro is designed to identify such a split and combine them in order to generate an accurate file path.

PROCESS FLOW

Figure 1 shows the overall process flow of the “headercheck” macro that automates the header checking process. This macro consists of a series of steps that begins with checking the file names that are passed as input parameters to the macro and ends with displaying the summary of findings.

Below is a sample macro call to the headercheck macro. In this example, we are intending to check the header of a program named sample.sas. Printheader parameter is set to “Y” which indicates that we need to have the header extracted and printed in the final summary.
“Headerstart” and “headerend” parameters correspond to the delimiter that identifies the first and the last line of the header part in the program (sample.sas in this case).
“Key_sec” parameters are the sections in the header that are mandatory. Every organization follows a different header format. The key sections that are considered in this example can be found below.

```
%headercheck(ckprogram=%str(x:/programs/macros/backup/sample.sas),
printheader=Y,
headerstart=%str(*soh*) ,
headerend=%str(*eoh*)
key_sec=%str(DATAINPUT!CODENAME!PROJECTNAME!DESCRIPTION!SPECIFICATIONS!VALIDATIONTYPE!OUTPUT);
);
```

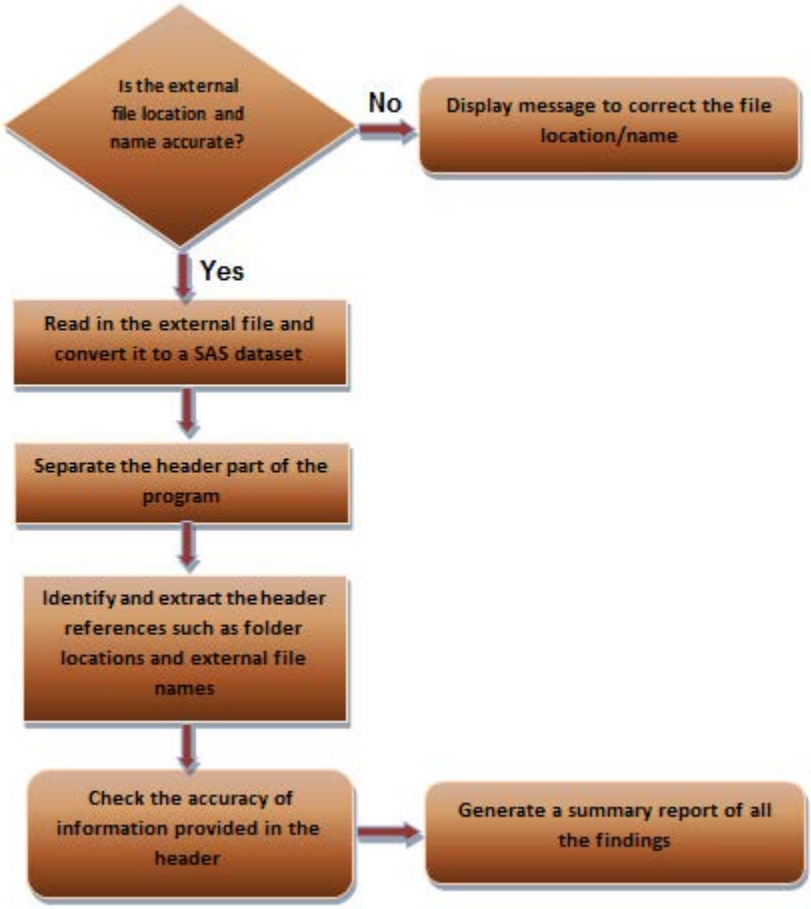

Figure1: Process flow chart

Let's take a closer looks at the overall process flow steps.

Step 1: Check if the external SAS program file name passed as “ckprogram” parameter exists.
SAS PIPE option is used with filename statement to check the legitimacy of the file path as shown in the code below.

```
/*Initialize the proceed_flg to 0 at the begining of execution */
%let proceed_flg=0;
 Filename Filename   pipe "dir  &ckprogram ";
Data _file_exist (keep=message)  ;
  Infile Filename truncover;
  Input filename $200.;
  Put filename=;
  if filename=:"dir: cannot access" then fileok=0;
   else fileok=1;
  if fileok=0 then Message=" NOTE: Unable to open file. Please check the  File path:  &ckprogram " ;
      if fileok=1 then Message=" NOTE: Success File path:  &ckprogram  is accurate" ;
  call symput ("fileok",fileok);
 Run;
```

The PIPE option along with “dir” and file path will return a value ‘ dir: cannot access.. <reference file path> ‘, if the file cannot be accessed in the specified location.
If the file location is not accurate then a macro variable (fileok) is assigned with a value 0 and “message” variable is assigned with corresponding text as shown the code above.

Step 2: Read in all the programs or a single program from the folder and separate the header part of the program header using the “Infile” and “Filename” statements as shown below.

```
data my_header;
      infile  "&ckprogram" truncover;
 input org_prgm $200.;
 if index(upcase(org_prgm) ,upcase("&headerend") ) then endofheader_flg=1;
 run;
```

“Headerend” variable needs to be assigned with a text that identifies the end of header. “endofheader_flg” variable will be missing for all the header related contents.

Step 3: Once the header is separated from the main program search each line of the code for possible directory/folder references and keywords such as “.sas, .rtf, .xls, .xlsx, .sas7bdat” etc. for possible file name references mentioned in the headers and check each of these references to test the legitimacy of these file names. If any of the key components that are expected in the header is missing those errors are also summarized along with the other findings as shown in the Header compliance report output as shown below.

Header compliance report
<Program Line #13> dir: cannot access C:/cdommon/macros/: No such file or directory
<Program Line #14> dir: cannot access C:/cdommon/programs/macros/: No such file or directory
<Program Line #19>Description (Required) : Cannot be missing
<Program Line #22> Cannot find files in the location(C:/cdommon/programs/output/shared/tfl/) sample2.sas"
<Program Line #25>Codename (Required) : Cannot be missing

CONCLUSION

Having an automated SAS header check macro really saves a lot of programmer's time and effort in ensuring the accuracy of the header. Since most of the organizations do not follow a standard template for the header, it is impossible to have a universal macro that can do a foolproof compliance check. However, with some effort, it is very much possible to standardize this key element of a program and automate them.
This macro could be an inspiration for those individuals or organizations that are trying to implement a better-standardized header documentation system process.

REFERENCES

Create Directory on Windows Without the Fleeting DOS Window, Available at URL:
www.lexjansen.com/pharmasug/2002/proceed/Coders/cc14.pdf

Using Unnamed Pipes :: SAS(R) 9.3 Companion for Windows, Available at URL:
support.sas.com/documentation/cdl/en/.../n16puwsro9pakqn1jamy1vwyaqx6.html

ACKNOWLEDGMENTS

Thanks to our Syneos Health Director Steve Benjamin for his leadership, constant support, encouragement and valuable assistance in reviewing this paper.

CONTACT INFORMATION

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

Hrideep Antony
Principal Statistical Programmer, Clinical Division, Syneos Health
Work Phone: +1- 984-459-4785
Email: hrideep.antony@syneoshealth.com
Web: http:/www.syneoshealth.com

Aman Bahl
Senior Manager, Statistical Programming, Clinical Division, Syneos Health
Work Phone: +1-289-313-3014
Email: aman.bahl@syneoshealth.com
Web: http:/www.syneoshealth.com

APPENDIX

```
%macro headercheck (ckprogram=,
  printheader=N,
 headerstart=%str(*soh*) ,
 headerend=%str(*eoh*))
 ;

 %let proceed_flg=0;
 /*************************************************************************************************************************************/
 /*  check if the file path is correct*/
 /*************************************************************************************************************************************/;
     Filename Filename   pipe "dir  &ckprogram ";
     Data _file_exist(keep=message) ;
      Infile Filename truncover;
      Input filename $200.;
      Put filename=;
      if filename=:"dir: cannot access" then fileok=0;
       else fileok=1;
    /*If the file path is not accurate assign 0 value to fileok macro variable and display the message*/;
      if fileok=0 then Message=" NOTE: Unable to open file. Please check the  File path:  &ckprogram " ;
      call symput ("fileok",fileok);
     Run;

  %if &fileok=0 %then %do;
     proc print data= _file_exist nobs;run;
    %put NOTE: Unable to open file. Please check the  File path:  &ckprogram  ;
  %end;
 /*************************************************************************************************************************************/
 /*   Read the external program file and convert it to a sas dataset*/
 /* Extract references to external file location from the header by using string functions to identify the file path*/
 /*************************************************************************************************************************************/;

 data my_header;
  infile  "&ckprogram" truncover;
 input org_prgm $200.;
 retain section stopcheck headerflg;
 length section $20;
 sort_ord=_n_;
 /*assign headerflg to 1 until it gets a value of 0 when the end of header line is identified*/;
  if _n_=1 then do;
   stopcheck=0;headerflg=1;
  end;

  upcase_org_prgm=compress(upcase(org_prgm));
  if index(upcase_org_prgm ,upcase("&headerstart" )) >0)  and headerflg then section='Header';
  if index(upcase_org_prgm ,upcase("&headerend") ) >0 then do; section='Program';headerflg=0;end;
   /*key_sec macro variable has the mandatory sections in the Header that needs to be checked */
   string="&key_sec";
   /*count the number of "!" that corresponds to the total number of sections that needs to be checked*/
  count_loop=count(string,'!');


  if section="Header" then do;
    do i= 1 to count_loop+1;
    /*trigger2 variable is flagged when any of the key sections are identified*/
     if index(substr(upcase_org_prgm,1,25),scan(string,i,'!'))>0 then trigger2=1;
    end;
 /*look for valid locations in the string*/

 if index(upcase_org_prgm,'C:/') then valid_location=1;
   if valid_location then  do;
     if index(upcase_org_prgm,':')>0 then _code_name=scan(upcase_org_prgm,2,':');
    else    _code_name= upcase_org_prgm ;
  end;
 /*extract the program path corresponding to CODENAME section*/
 if index(upcase_org_prgm,'CODENAME') then do;
     falg=1;
    _code_name=scan(upcase_org_prgm,2,':');
    _proj_name=scan(upcase_org_prgm,2,':');
 end;

   /*if the location has not started with '/' add '/' to have a valid location*/
  if _code_name=:'C:/' and valid_location then _code_name='/'!!_code_name;
  if _proj_name=:'C:/' and valid_location then _proj_name='/'!!_proj_name;
 end;
 /*if end of section is identified then stopcheck flag is set*/
 /*the next_section flag is only incremented for new sections */
  if _n_>10 and index(substr(upcase_org_prgm,1,25) , '------------')>0 then stopcheck=1;
 retain next_section;
 if _n_=1 then next_section=1;
  if   section='Header' and stopcheck=0 then do;
     if trigger2=1 then next_section+1;
     first_part=scan(upcase_org_prgm,1,':');
     second_part=scan(upcase_org_prgm,2,':');
  end;
 run;

 proc sort data= my_header;
   by next_section sort_ord;
 run;
```

```
data my_header2;
set my_header;
  by next_section sort_ord;
   lag_CODE_NAME=lag(_CODE_NAME);
   lag_next_section=lag(next_section);
 retain location_ck;
 if first.next_section   then location_ck=.;
 if first.next_section and VALID_LOCATION=1 then location_ck=1;
/*if the file name is mentioned with the datapath then use them*/
 if index(_CODE_NAME,'.') >0 or  index(_CODE_NAME,',')  >0 or index(_CODE_NAME,'/')   then do;
/*extract the file name from the whole path by reversing the text and using the scan function*/
    text=reverse(scan(reverse(_CODE_NAME),1,'/'));
end;

length reverse_code datanm1-datanm20 folder_location _folder_location $200;
array datanm(*)  datanm1--datanm20;

if lag_next_section=next_section and VALID_LOCATION=. and location_ck=1 then do;
  do i= 1 to dim(datanm);
    if lag_next_section =next_section and VALID_LOCATION=. then datanm(i)=scan(ORG_PRGM,i );
    if compress(upcase(datanm(i)))in('SAS' 'XLS' 'RTF') then datanm(i)='';
     if lastvar=. and datanm(i)='' then lastvar=i-1;
  end;
end;
/*sometimes more than one file is specified within the same file path the following code will separate such files */
if text ne '' and LOCATION_CK then do;
  do j= 1 to dim(datanm);
   if datanm(j)='' then datanm(j)=scan(text,j );
    if compress(upcase(datanm(j))) in('SAS' 'XLS' 'RTF')   then datanm(j)=' ';
     if lastvar=. and datanm(j )='' then lastvar=j-1;
  end;
end;

if VALID_LOCATION then do;
  reverse_code=reverse(compress(_CODE_NAME));
  folder_location=compress(reverse(substr(reverse_code,index(reverse_code,'/'))));
 end;

 if first.NEXT_SECTION then _folder_location='';
 retain _folder_location sub_cntr;;
   if folder_location ne '' then _folder_location=lowcase(folder_location);
 if first.NEXT_SECTION then sub_cntr= NEXT_SECTION ;
 if folder_location ne '' then sub_cntr+.01;
 merge_dummy=1;
run;
/*create macro variables for each of the file locations identified*/
/*create macro variables for each of the file locations identified*/
data macro_vars;
 set my_header2;
  by merge_dummy;
   _name_=_n_;
  where   folder_location ne '';
 call symput('path_'||left(_name_), compress(_folder_location));
 call symput('path_ct' ,  put(_name_,3.));
 call symput('path_ord_'||left(_name_) ,  put(sub_cntr,6.2));
run;
 /*/
/*************************************************************************************************************************/
/*  For each of the identified path check its accuracy**/
 /*Pathck macro will check all the identified path and final_reprot is created*/
/*************************************************************************************************************************/;

%macro pathck;
data final_reprot;
 length filename $200;
 sub_cntr=.;
run;
 /*Using the pipe statement check the legitimacy of the folder path for all the locations path*/
 %do i = 1 %to &path_ct;
  %let path=&&path_&i;
 %put *****&path ;
    Filename filelist   pipe "dir  &path ";

   Data _null_&i final_reprot_;
     Infile filelist truncover;
     Input filename $200.;
     Put filename=;
     sub_cntr=&&path_ord_&i;
     output;
    Run;

data final_reprot;
 set final_reprot final_reprot_;
 run;

 %end;

 %mend pathck;
%pathck;
```

```
proc sort data= final_reprot;
 by sub_cntr;
 run;

 data final_reprot2;
 set  final_reprot;
by sub_cntr;
retain combined;
length  combined $2000  filename $100. ;
if first.ref_no then combined=trim(FILENAME);
  else  combined=trim(combined)!!" % "!! trim(FILENAME);
  if last.sub_cntr then combined="SAS XLS XML XLSX"!!combined;
  if last.sub_cntr and sub_cntr ne .;

 run;

proc sort data= my_header2;
 by sub_cntr;
 run;
/*Generating issue reports*/
data my_header3;
 merge my_header2 final_reprot2;
 by sub_cntr;
if int(SUB_CNTR) ne (SUB_CNTR) then do;
  length notfnd1-notfnd20 $200 issue_reprot $1000 ;
  array datanm(*)  datanm1--datanm20 ;
  array notfnd(*)  notfnd1--notfnd20;
 if compress(filename)=:"dir:cannot" then issue_reprot="<Program Line #"!!compress(put(sort_ord,5.))!!'> '!!filename;
   do i = 1 to dim(datanm);
     if datanm(i) ne '' and index(lowcase(combined),compress(lowcase(datanm(i))))=0 then do;
      notfnd(i)= lowcase(datanm(i));
     if issue_reprot='' then do;
      issue_reprot="<Program Line #"!!compress(put(sort_ord,5.))!!'> Cannot find files in the location('!!compress(_FOLDER_LOCATION)!!') '!!compress(lowcase(datanm(i)));
     end;
    else do;
      issue_reprot=trim(issue_reprot)!!', '!!compress(lowcase(datanm(i)));
    end;
   end;
  end;
 end;
 run;
/*************************************************************************************************************************************************/
 /*Check if any of the key information is missing and generate corresponding issue statement*/
/*************************************************************************************************************************************************/

data my_header4;
 set   my_header3;;
 length  issue_reprot2 $1000 ;
 string="&key_sec";
 count_loop=count(string,'!');
 do i= 1 to count_loop+1;
   if index(substr(first_part,1,25),scan(string,i))>0 and stopcheck=0 then do;
     if compress(SECOND_PART)='' then cccc=1;
     if compress(SECOND_PART)='' then issue_reprot2="<Program Line #"!!compress(put(sort_ord,5.))!!'>'!!propcase(trim(ORG_PRGM))!!' Cannot be missing' ;
  end;
 end;
 run;

data final_report;
set my_header4 (in=a where=(issue_reprot ne '' ))my_header4(in=b where=(issue_reprot2 ne '' ));
  if a then final_reprot= issue_reprot;
  if b then  final_reprot= issue_reprot2;
run;
/*************************************************************************************************************************************************/
 /*Generating final statement*/
/*************************************************************************************************************************************************/
proc print data= final_report   label  ;
var final_reprot;
label final_reprot= "Header Compliance report" ;
run;

%if &printheader=Y %then %do;
proc print data=my_header4 nobs label ;
var SORT_ORD ORG_PRGM;
 label SORT_ORD= "Program Line Number" ORG_PRGM= "Program header";
  where SECTION="Header";
 run;
%end;

%mend headercheck;
```