Pipe it! Extract it! Build it!

Smitha Krishnamurthy, Roche Molecular Systems Inc., 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 the 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 the accuracy of the program header by comparing the 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

In this macro we explore how we can use the PIPE option to read SAS code and create study related documents.

Depending upon the nature of the study folder setup the SAS RTF output location can be same as the SAS code or it can be saved in a different folder. Macro is designed to handle both scenarios.

SAS code standard header should have the input, output and the author information.

Example of a standard header –

```
/*************************************************************************
Program:

Purpose:

Input:

Output:

Created by:

Modified by:

Notes:

*************************************************************************/
```

PROCESS FLOW

- Using the macro input parameters create the Study Path and RTF Path
- Read RTF files from the actual RTF path using system commands

- Read and search individual program headers for key information
- Create a dataset per code with information read from the header
- Stack the header datasets
- Compare the actual RTF with RTF specified in the header - Validation Issues Sheet
- Compare the stacked data with the issues and remove records with issues – Validation Assignment Sheet

MACRO CALL

- **Standard Macro Call**

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

- **Special Case Macro Call**

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

Explanation of the macro parameters and macro details to follow the call

OUTPUT GENERATED

Validation Tracker

Name of Program	Location of Program	Name of Output	Type of Program	Location of rtf output	Developer Name	Developer Status	Pre-Validation Programmer	Pre-Validation Status	Validator Name	Validator Status

Validation Issues

Comment	Name of the SAS Program	Output Name	Actual path of the RTF output

LIMITATIONS AND CHALLENGES

Challenges –

- Using SAS to read a SAS code
- Parsing and sorting of alphanumeric strings
- System commands
- Create Hyperlink in SAS

Limitations -

- All SAS codes need to follow a standard header
- Macro cannot be executed in SAS Enterprise Guide

Appendix: SAS code

```
/************************************************************************
Program:       vallist.sas

Purpose:          A macro that generates the validation assignment sheet for a
clinical study (rtf outputs)

Input:

Output:               Excel File


Created by:           Smitha & Sofia

Modified by:

Notes:

************************************************************************/

%macro loop1(analyte=, protocol=, fname=, tname=, pgmloc=, rtfloc=, metafileout=);

        *STEP1: Error Message to Identify Key macro variables;
       *Key Macro Variables Need to be assigned a value, if missing study path cannot be defined;
        %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";

        %if %length(&metafileout)=0 %then
                %do;
                        %let metafileout=YES;
                %end;

        *default is Yes;
        *STEP2: Derive Full Path;

        *If a standard structure is followed then study path will not use pgmloc
        and rtfloc macro variables. For certain studies code location (pgmloc) and output
location
        (rtfloc) needs to be defined and maybe similar or different;
        %if %length(&pgmloc.)=0 and %length(&rtfloc.)=0 %then
                %do;
                        %let path=\\rpbmssasp1\biometrics\&analyte\&protocol\&fname\&tname;
                        %put &path;
                        %let rtfpath=\\rpbmssasp1\biometrics\&analyte\&protocol\&fname\&tname;
                        %put &rtfpath;
                %end;
        %else
                %do;
                       %let  path=\\rpbmssasp1\biometrics\&analyte\&protocol\&fname\&tname\&pgmloc;
                        %put &path;
                        %let
rtfpath=\\rpbmssasp1\biometrics\&analyte\&protocol\&fname\&tname\&rtfloc;
                        %put &rtfpath;
                %end;

        options nodate nonumber noxwait;
        %let path=%sysfunc(tranwrd(&path,/,\));
        %let rtfpath=%sysfunc(tranwrd(&rtfpath,/,\));
```

```
	*STEP3: List of rtf output from the rtf location;
	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;

	*clear if multiple studies are being run. Good practice;
	filename rtflist clear;

	data rtflist2;
		set rtflist1;

		if index(upcase(rtfname),"ARCHIVE") > 0 then
			delete;

		if upcase(scan(rtfname,-1,'.')) in ("RTF");
		rtfNM = scan(scan(rtfname,-1,'\'),1) || ".rtf";
	run;

	data rtflist3;
		set rtflist2;

		*actual rtfpath;
		artfpath= substr(rtfname, 1 , length(rtfname) - length(scan(rtfname,-1,'\'))-1);

		*analyte path;
		alytpath=substr(artfpath, index(artfpath,"&analyte"));
	run;

	*STEP4: Read the SAS Code and get the header information;
	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;

	*clear if multiple studies are being run. Good practice;
	filename tasklist clear;

	*look for the program with extension .sas to read code header;
	data templist_;
		set filelist;

		if upcase(scan(filename,-1,'.')) in ("SAS");
		PGMPATH  = substr(filename, 1 , length(filename) - length(scan(filename,-1,'\'))-
1);
		PGNM = scan(scan(filename,-1,'\'),1);
		extention = upcase(scan(filename,-1,'.'));
		alocfld = reverse(scan(reverse(upcase(pgmpath)),1,'\'));
	run;

	*Study Folder In-text has a template code, need to delete that;
	data templist;
		set templist_;

		if index(upcase(pgnm), 'TEMPLATE') then
			delete;
	run;
```

```
	proc sort data=templist sortseq=linguistic;
		by pgmpath pgnm;
	run;

	data templist1;
		set templist;

		%if %length(&pgmloc.)=0 %then
			%do;
				locpgfld=%upcase("&tname");
			%end;
		%else
			%do;
				locpgfld=%upcase("&pgmloc");
			%end;

		if locpgfld=alocfld;
	run;

	proc sort data=templist1 sortseq=linguistic;
		by pgmpath pgnm;
	run;

	*cnttot has the number of SAS codes;
	data finallist;
		set  templist1;

		if upcase(PGNM) ^= '';
		cnt=_n_;
		call symput('cnttot', left(_n_));
	run;

	*making all macro related to filenames global *;
%macro globuild;
	%do i = 1 %to &cnttot;
		filenm&i
	%end;
%mend;

%global %globuild;

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

*Iterate for each SAS Code;
%do j=1 %to &cnttot;
	%put &&filenm&j;

	*Create a reference to SAS code, Assume the sas code to be a text file and Read in the
text file;
	*Read the entire code;
	filename refsascd pipe "dir /b /s ""&path\&&filenm&j...sas""";

	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;

	*Read only the header part of the code upto 'Created By: ' statement;
```

```
	data _all_code&j._1;
		set _all_code&j;

		*initalize *;
		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;

	%put &linenum;

	data header&j;
		set _all_code&j._1;

		if _n_ <= &linenum;
	run;

	data _null_;
		set header&j;

		if index(upcase(string),'CREATED') >0;
		call symput("createdby", scan(string,2,':'));
	run;

	%put &createdby=;

	data _null_;
		set header&j;

		if index(upcase(string),'MODIF') >0;
		call symput("modifyby", scan(string,2,':'));
	run;

	%put &modifyby=;

	*Read starting point of the output;
	data header&j._2;
		set header&j;
		indxby=_n_;
		retain aoutpath;

		if index(upcase(string),'.RTF') >0;
		tmpstrng = catx('\',"&analyte","&protocol","&fname","&tname");

		* create output location from rtf output path from header;
		* Next step is to match this outpath with rtfpath. and make sure they are same;
		outpath1= compress(substr(string, 1 , length(string) - length(scan(string,-
1,'\'))-1));

		*actual name of the output path*;
		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"));

		*actual name of the anlyte path;
```

```
        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);

        if aoutpath ne '' then
                tmpapath=aoutpath;

        if aoutpath='' then
                do;
                        aoutpath= tmpapath;
                end;
run;

data header&j._4;
        set header&j._3;
        aoutpath=catx('\', tmpapath,hrtfnm1);
run;

data header&j._5;
        set header&j._4;

        if scan(aoutpath,1,'\')="&protocol" then
                aoutpath=compress(catx('\',"&analyte",aoutpath));
        else aoutpath=compress(aoutpath);
run;

data header&j._6;
        set header&j._5;

        *Subset for rtfloc;
        if index(aoutpath,"&rtfloc")>0 then
                output;
run;

options symbolgen macrogen mprint;

data _null_;
        set header&j._6;
        call symput('hrtfcnt', left(_N_));
run;

proc sql noprint;
        select hrtfnm1 into: rtfnm1 - :rtfnm&hrtfcnt
                from header&j._6;
quit;

data exlout&j;
        attrib Program_path length =$100
                Output_location length =$200
                Program_Name length =$100
                Output_Name length =$100
                Developer length =$100
                Program_status length =$100
                Validator length =$100
                Validation_status length =$100;
        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=" ";
                Validator=" ";
                Validation_status=" ";
                keep Program_path Output_location Program_Name Output_Name Developer
Program_status Validator Validation_status;
        run;

%end;

*End for the j loop;
data final_(rename=( Output_Name= Output_Name_));
        set

                %do k=1 %to &cnttot;
                        exlout&k
                %end;
        ;
        *for set statement;
run;

data final;
        length output_name $100;
        set final_;
        Output_Name=left(scan(Output_Name_,1,','));
        drop  Output_Name_;
run;

*temp dataset: check rtf list from header *;
data chkfinal;
        set final;
        length col1 $100;
        col1=upcase(strip(output_name));
        keep col1 program_name;
run;

proc print data=chkfinal;
run;

data rtflist4(keep = col1 alytpath);
        length col1 $100;
        set rtflist3(rename=(rtfnm=Output_Name));
        col1=upcase(left(compress(output_name)));
run;

proc sort data=chkfinal out=chkfinal_  sortseq=linguistic(numeric_collation=on);
        by col1;
run;

proc sort data=rtflist4 out=rtflist4_  sortseq=linguistic(numeric_collation=on);
        by col1;
run;

proc contents data=chkfinal;
run;

proc contents data=rtflist4;
run;

proc compare base=chkfinal_
        comp=rtflist4_ listbase listcomp listall;
        var col1;
run;

*Output 1: Comparison between header and actual rtf;
data Validation_Issues;
        length comment $200;
        merge chkfinal_(in=A) rtflist4_(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 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;

*Output 2: Validation file;
*read from the actual folder to capture the actual rtf present in the folder;
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;

*Add alytepath to the clean assignment sheet;
data Validation_assignment_sheet(drop=col1);
        merge pre_Val_sheet_(in=a) rtflist4_(in=b);
        by col1;

        if a;
run;

%if %upcase(&metafileout)=YES %then
        %do;
                *Output 3: Metadata file;
                data metafile;
                        set Validation_assignment_sheet;
                        filelocation='\\RPBMSSASP1\Biometrics\'||alytpath;
                        listofintexttables=output_name;
                        runflag='No';

                        if substr(upcase(output_name),1,5) ='TABLE' then
                                MasterTag=scan(output_name,1,'_');
                        else masterTag='';
                        StartTag='Start_'||mastertag;
                        Endtag='End_'||mastertag;
                        deletetitles='No';
                        keep filelocation listofintexttables runflag MasterTag StartTag Endtag
deletetitles;
                run;

                proc sort data=metafile out=metafile_ sortseq=linguistic(numeric_collation=on);
                        by mastertag filelocation;
                run;

                *hyperlink to location and variable name that has multiple values;
                ods excel
file="\\rpbmssasp1\biometrics\HPV\HPV435\Final_6888\Checks\Smitha\metafile_HPV_PS25.xlsx";
```

```
                proc report data=metafile_ nowd;
                        columns  filelocation listofintexttables runflag MasterTag StartTag Endtag
deletetitles;

                        compute filelocation;

                                *href=trim(filelocation)||".html";
                                call define(_col_, "URL", filelocation);
                        endcomp;
                run;

                ods excel close;
        %end;
%mend loop1;
```