Paper DH09

Early Detection and Prevention of Compliance Issues

Vara Prasad Reddy Sakampally, Syneos Health, Raliegh, USA

ABSTRACT

Creating the datasets that meets all compliance requirements is a challenging task and most companies do check for compliance and quality at the end of study. Identifying the issues at the later stage involves additional resources and risks the deliverable timelines as the issues are to be addressed. By adding a process in place that will check for compliance at the stages of specification creation and dataset development will minimize the quality issues at final delivery. This paper will propose various approaches and how to call the pinnacle application (which is considered as the gold standard in the industry) through a SAS® session to assess the quality of dataset during its development phase. By adopting such a process; majority of compliance issues can be addressed before passing to the validation.

INTRODUCTION

Targeting to create a dataset that is free of compliance issues before passing to the validation cycle is critical to avoid unexpected work during the last minute before delivery. Addressing compliance issues involves cost and staying within the timelines will be challenging. Most companies do implement various approaches to avoid compliance issues and are always a learning process. This paper proposes a simple process design during a dataset development lifecycle that makes the dataset free of most compliance issues in its initial development phase.

VERIFYING THE METADATA FOR COMPLIANCE DURING SPECIFICATION DEVELOPMENT

It is advisable to check for the compliance issues related to metadata at the initial specification development phase rather than doing at the end of a dataset development life cycle. A novel approach to this is to follow the CDISC® implementation guide when writing the specifications and checking for the compliance issues before creating the program.

One method to check for compliance is by reading the specification file into a SAS dataset and comparing it with the standard specification file dataset that is created off of the Implementation guide and is often referred to as a standard specification template. This approach is a complex process and often needs an update to the standard template when there are revisions proposed by CDISC. The other approach also involves reading in the specification file and creating SAS datasets with zero observations. These datasets are then converted to .xpt files and are passed through Pinnacle21® to check for the key issues like:

1) Order of variables should be as specified by CDISC standard (FDA Validator Rule ID - SD1079)
2) Identify any missing required or expected variables (SD1129, SD1044, SD1083, SD1076, SD0057, SD1073, SD1101, SD1293, SD1294, SD1102, SD1103, SD1104, SD1280, SD1281, SD1244, SD1245, SD1246, SD1282, SD1283, SD1285, SD1299, SD9999)
3) Compare dataset and variable metadata against the standard (SD0063)

The latter approach is preferred as the dummy datasets can be created using a simple code as provided below. These datasets are converted to .xpt files and are passed through Pinnacle21 in the first place and all the compliance issues related to metadata are identified which can be addressed before proceeding to programming.

```
/* Snippet of partial code to create dummy dataset*/
%do j=1 %to &cnt.;*&cnt represents the number of domains in the study;

/* spec file is imported and a dataset by name xlspec is created in work library*/
data &&dsin&j.._specs;
set xlspec(where=(upcase(domain)= "&&dsin&j.."));
run;

proc sql noprint; *get dataset label into macro variable from spec file;
select description into: dslabel from label where upcase(domain) eq "&&dsin&j..";
quit;
%put &dslabel.;

%let id=%sysfunc(open(&&dsin&j.._specs));
%let nobs=%sysfunc(attrn(&id,nobs));
```

```
%syscall set(id);

data &&dsin&j..(label="&dslabel.");
   %do i=1 %to &nobs;
       %let rc=%sysfunc(fetchobs(&id,&i));
       %if %upcase(&type)=CHAR %then %do;
              format &variable_name %sysfunc(compress($&format..));
              length &variable_name %sysfunc(compress($&length.));
       %end;
       %else %if %upcase(&type)=NUM %then %do;
           format &variable_name %sysfunc(compress(&length..));
       %end;
       label &variable_name= %sysfunc(compbl("&variable_label"));
       keep &variable_name;
%end;
if _n_ ge 1 then delete;
run;

%let id=%sysfunc(close(&id));
%end;
```

FORMATS AND CONTROL TERMINOLOGY

One of the annoying issues we encounter during compliance checks is related to the controlled terminology. As we have to adhere to controlled terminology, addressing these issues involves a lot of time and revisiting many programs. To overcome this, all the formats used in the study should be coming off of the study setup macro. This approach makes it easy for the study team as the programmers do not need to revisit the program when an issue related to controlled terminology is identified. To implement this, the programmers should be discouraged to use proc format in their programs.

CHECKING LOG FOR UNWANTED MESSAGES

The other common issue we notice is the presence of unwanted messages in the log and are to be addressed in the first place by the authors of the program. Identifying these messages after finalizing all the datasets and addressing them involves cost, time and may result in inaccurate results. To address this issue, a log check macro should be called in at the end of every program to identify the issues and the author of a program can address them before acknowledging the programming as complete. A sample log check macro code is provided below that prints issues at the end of log and the code can be optimized based on client-specific needs.

```
%macro chklog;
dm log 'print file = "C:\SASPaper\logs\chklog_&pgm..log" replace';

data log(where=(type ne ""));
infile "C:\SASPaper\logs\chklog_&pgm..log" truncover lrecl=300;
length Message $500 type $20;
retain loc; input @1 Message $char500.;
if substr(Message,1,4) not in ("NOTE","WARN","ERRO","INFO") then
word=upcase(scan(Message,2));

if word in ("+DATA","+ DATA","DATA") then do;
       strt_txt=length(scan(Message,1))+1;
       loc=trim(left(substr(Message,strt_txt)));
end;
else if word in ("+PROC","+ PROC","PROC") then do;
       strt_txt=length(scan(Message,1))+1;
       loc=trim(left(substr(Message,strt_txt)));
end;

if substr(Message,1,5) eq "ERROR" then do;
       type="ERROR";
       output;
end;
else if substr(Message,1,7) eq "WARNING" then do;
       type="WARNING";
       output;
```

```
end;
else if substr(Message,1,4) in ("NOTE","INFO") then do;
      if prxmatch("m/stopped|Numeric values have been converted|Character values have
been converted|Uninitialized|Division|Invalid|Missing|Merge
statement|Mathematical|W.D|Overwritten/oi",Message) > 0 then type="NOTE";
      output;
end;
run;

data _null;
set log;
put "/*** ISSUE: " type= "location= " loc Message= "***/";
run;
%mend chklog;
```

INVOKING PINNACLE THROUGH SAS

Pinnacle21 command line interface allows automation of validation jobs resulting in higher reliability of results and better performance of data preparation process[1]. The code to invoke Pinnacle21 using a command line depends on the Pinnacle21 type (Community or Enterprise) installed and under the assumption that Java is also installed on the system. Also, the code should be modified based on the version number of Pinnacle21 installed on the system. Additional details on optimizing the code for command line interface and for a specific task can be found at https://www.pinnacle21.com/projects/validator/community-cli. There are minor modifications needed for the code based on the dataset type (SDTM or ADaM). The sample code provided below assumes that the Pinnacle21 version 2.2.0 and Java is installed on the system and separate macros are provided for SDTM and ADaM datasets.

```
%macro chksdtm(ds=);
 %let ig=3.2; *Implementation guide version #;
 %let cldt=2019-09-27; *codelist date;
 %let ds=%upcase(&ds);
 %let src=C:\SASPaper\wip\; *.xpt file path;
 %let report=C:\SASPaper\wip\&ds._p21_report.xlsx; *Pinnacle report path;

 /* path of .jar file and name: file name depends on P21 version: validator-cli-
{version}.jar*/
 %let japplet=C:\SASPaper\pinnacle21-community\components\lib\validator-cli-2.1.5.jar;
 %let java=C:\SASPaper\pinnacle21-community\java64\bin\java.exe;
 %let config=C:\SASPaper\pinnacle21-community\components\config\SDTM &ig..xml;

 filename &ds "&src.&ds..xpt";
 filename supp&ds "&src.supp&ds..xpt";
 %if %sysfunc(fexist(&ds)) %then %do;
       %if %upcase(&ds.) ne DM %then %do;
          %if %sysfunc(fexist(supp&ds)) %then %do;
             %let jsource = %str(&src.dm.xpt;&src.&ds..xpt;&src.supp&ds..xpt);
          %end;
          %else %do;
             %let jsource = %str(&src.dm.xpt;&src.&ds..xpt);
          %end;
       %end;
       %else %do;
          %if %sysfunc(fexist(supp&ds)) %then %do;
             %let jsource = %str(&src.&ds..xpt;&src.supp&ds..xpt);
          %end;
          %else %do;
             %let jsource = %str(&src.&ds..xpt);
          %end;
       %end;
    options noxwait xsync;
    x &java -jar "&japplet"
    -type=SDTM
    -source="&jsource"
    -source:type=sas
    -config="&config"
    -config:cdisc=&cldt
    -report="&report"
    -report:type=excel
```

```
    -report:overwrite=yes;
 %end ;
 %else %put 'The xpt file is missing in specified location';
%mend chksdtm;

%macro chkadam(ads=);
 %let ig=1.0; *Implementation guide version #;
 %let cldt=2019-09-27; *codelist date;
 %let ads=%upcase(&ads);
 %let src=C:\SASPaper\wip\; *.xpt file path;
 %let report=C:\SASPaper\wip\&ads._p21_report.xlsx; *Pinnacle report path;

/* path of .jar file and name: file name depends on P21 version: validator-cli-
{version}.jar*/
 %let japplet=C:\SASPaper\pinnacle21-community\components\lib\validator-cli-2.1.5.jar;
 %let java=C:\SASPaper\pinnacle21-community\java64\bin\java.exe;
 %let config=C:\SASPaper\pinnacle21-community\components\config\ADaM &ig..xml;

 filename &ads "&src.&ads..xpt";
 %if %sysfunc(fexist(&ads)) %then %do;
        %if %upcase(&ads.) ne ADSL %then %do;
             %let jsource = %str(&src.adsl.xpt;&src.&ads..xpt) ;
        %end;
        %else %do;
             %let jsource = %str(&src.&ads..xpt);
        %end;
       options noxwait xsync;
    x &java -Dcli.log.level=debug -jar "&japplet"
    -type=ADaM
    -source="&jsource"
    -source:type=sas
    -config="&config"
    -config:cdisc=&cldt
    -report="&report"
    -report:type=excel
    -report:overwrite=yes;
 %end ;
 %else %put 'The xpt file is missing in specified location';
%mend chkadam;
```

The above macro calls when used for subject level dataset will run the checks on only a specified dataset whereas when called for other datasets, the checks will be performed along with the subject level dataset for cross-comparison of data. In other words, if the macro is called to check issues in ADVS, the macro will also check for ADSL and verify the data for interrelated issues between ADSL and ADVS.

FINAL CHECK THROUGH FREQUENCY DISTRIBUTION

Though the dataset has passed through the checks stated above, it is quite possible that the questionable values (extremely high or extremely low values which are not true to believe) are not picked up by Pinnacle21 or validation. These values can be identified by placing the edit check macros in place to scan the raw data for potential issues before proceeding to dataset development. These issues can also be identified by running a frequency distribution on the final dataset and reviewing the values on a high level before acknowledging the dataset programming is complete. The programmers should be encouraged to do a frequency check on dataset before finalizing the dataset.

CONCLUSION

I do agree that a dataset that meets all the compliance requirements is hard to create in the very first cycle of dataset development. But, most of the compliance issues can be addressed in the initial stage by adding additional checks as recommended above and avoid last-minute rush to address the issues. These last-minute changes will be the source of sliding issues that surface during the review of reports.

REFERENCES

[1] Garrett Amy, Vinokurov Aleskey “How to Automate Validation with Pinnacle 21 Command Line Interface and SAS” Proceedings of PharmaSUG 2018 Conference.

CONTACT INFORMATION

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

Vara Prasad Reddy Sakampally
Syneos Health
3201 Beechleaf Ct
Raleigh, NC 27604, USA
Work Phone: 270-282-5166
Email: varaprasadreddy.sakampally@syneoshealth.com