PharmaSUG 2018 - Paper AD-14

Automatically Configure SDTM Specifications Using SAS® and VBA

Xingxing Wu, Eli Lilly and Company, Indianapolis, IN

ABSTRACT

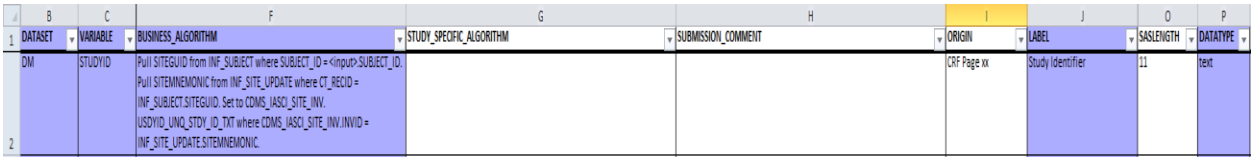
SDTM is becoming the standard for pharmaceutical companies to submit data to FDA. The SDTM specifications are the foundation to create SDTM domains. Usually a standard study specification template (SST) is available and the study team needs to configure it to meet the specific requirements of the study. Currently, the SDTM specifications are configured manually. This paper proposes an automatic and efficient approach to configure certain parts of SDTM specifications using SAS and Visual Basic for Applications (VBA) while maintaining the original specification formats or converting the formats to the desired ones. This approach can also be directly applied to other areas to dynamically update the excel files and satisfy the format requirements.

INTRODUCTION

SDTM provides a standard way to format and organize data. It is becoming one of the required standard to submit data to FDA because of its capability to improve the regulatory review and approval process. It is widely used in pharmaceutical companies. Writing SDTM specifications is the first step of SDTM development. To standardize the SDTM specifications, the company would usually provide a standard study specification template in excel file. The study team will then need to configure this template based on the specific requirements of the study. Usually, the configuration is implemented manually, even though, technically, certain parts of the specifications can be configured by SAS programs, such as the maximum variable lengths and the variable origins (Case Report Form (CRF) pages). The reason is that it is not easy to use SAS programs to maintain the formats of the original specification excel file. For example, to easy the configuration, the study specification template usually uses different colors to divide the excel sheets into different parts, such as using dark blue for non-configurable parts, and using light color for configurable parts. If we use SAS programs to directly update the contents of the specification excel file, all the original excel file formats, such as colors, fonts, will be lost. This might not be acceptable in certain situations.To address this issue, one possible solution is to use SAS Dynamic Data Exchange (DDE) to control excel. However, we can not run DDE programs submitted from SAS Enterprise Guide on SAS Grid. In order to use SAS programs to help the configuration of the SDTM specifications, another option is to use SAS programs to create a separate excel file without format requirement instead of directly updating the original SDTM specifications. For example, we can use SAS programs to calculate the maximum variable lengths in each domain, and find the related CRF pages of the CRF variables. All these results will be exported into separate excel files which will then be combined together with original SDTM specifications to generate the Define.xml submitted to FDA. The drawback of this approach is the potential inconsistence between Define.xml, XPT files, SAS datasets, and SDTM specifications. In order to overcome the issues of these approaches, this paper proposes an automatic and efficient approach to directly configure certain parts of the SDTM specifications using SAS and VBA while maintaining the original specification formats or even converting the formats to the required ones. The configured SDTM specifications can then be used to generate SDTM domains and Define.xml. This approach is not limited by the SAS running environment and has the advantage of maintaining the consistence between SDTM specifications, SAS datasets, XPT files, and Define.xml. Although this approach is developed to configure SDTM specifications, it can also be directly applied to other areas to automatically update the contents of the excel files based on the dynamically generated information while satisfying the format requirements. This paper will first briefly introduce the SDTM specifications, and then use two examples to demonstrate how to use SAS programs to generate the required information that will be used to configure the SDTM specifications. Finally, this paper will discuss how to use VBA to update the SDTM specifications based on the SAS-generated information.

SDTM SPECIFICATIONS

SDTM specification configuration is the first and very important step in the SDTM development. The SDTM domains will be created based on the algorithms defined in the SDTM specifications. The attributes of the variables in SDTM domains are also defined in the SDTM specifications. It will also be used as one of the inputs to create Define.xml submitted to FDA. In order to standardize SDTM specifications, the company will usually provide the standard study specification template. The study team just needs to configure this template based on study specific requirements. The following is an example of the SDTM specification template which might be different between companies:



	B	C	F	G	H	I	J	O	P
1	DATASET	VARIABLE	BUSINESS_ALGORITHM	STUDY_SPECIFIC_ALGORITHM	SUBMISSION_COMMENT	ORIGIN	LABEL	SASLENGTH	DATATYPE
2	DM	STUDYID	Pull SITEGUID from INF_SUBJECT where SUBJECT_ID = <input>.SUBJECT_ID. Pull SITEMNEMONIC from INF_SITE_UPDATE where CT_RECID = INF_SUBJECT.SITEGUID. Set to CDMS_IASCI_SITE_INV. USDYID_UNQ_STDY_ID_TXT where CDMS_IASCI_SITE_INV.INVID = INF_SITE_UPDATE.SITEMNEMONIC.			CRF Page xx	Study Identifier	11	text

Figure 1. SDTM Study Specification Template

In the example above, the blue parts are non-configurable. The study team just needs to configure the white parts, such as “STUDY SPECIFIC ALGORITHM”, “ORIGIN”, and “SASLENGTH”.

Another important usage of SDTM specifications is to automatically assign the attributes to the related variables in SDTM domains.

For this purpose, we can first convert the specification excel files into SAS datasets. This can be done using PROC IMPORT, or use XLSX engine:

```
libname specs xlsx <'physical-path and filename.xlsx'>
```

With this, the SAS datasets related to excel sheets (domains) in the specification file can be directly accessed using `specs.domain`, such as `specs.dm` and `sepcs.ae`.

After this, we can use SAS program to assign the attributes to variables based on SDTM specifications. The following is the main part of the implementation:

```
/*create the length variable*/
data domain_meta;
  set specs.domain;
  if datatype in ("text" "date" "datetime") then varlth=
     compress(variable)||" $"||compress(put(saslength, best.))||".";
  else varlth=compress(variable)||" "||compress(put(saslength, best.))||".";
run;

/*create macro variables of keep variables, length variables and labels */
proc sql noprint;
      select variable into: keepvars separated by " " from domain_meta;
      select varlth into: varslen separated by " " from domain_meta;
      select compress(variable)||'="'||strip(label)||'"' into: labels
             separated by " " from domain_meta;
quit;

/*assign attributes to the variables and only keep required variables */
```

```
data domain;
  format &varslen.;
  retain &keepvars.;
  set domain_in;
  keep &keepvars.;
  label &labels.;
run;
```

Another approach is to first create empty shells related to SDTM domains, and then set the shells and the related SAS datasets together. The shells will only have the required variables with assigned attributes, but have no record. The shells are also created based on the SDTM specification meta datasets generated before.

AUTOMATICALLY GENERATE CONFIGURATION INFORMATION USING SAS

Currently, the whole SDTM specifications are configured manually. From previous section, we know, technically, some parts of the specification template can be configured by SAS programs, such as maximum variable length (SASLENGTH) column, and ORIGIN column. We will use two examples to demonstrate how to use SAS programs to automatically generate the required information that will be used to configure SDTM specifications.

CALCULATE MAXIMUM VARIABLE LENGTHS USING SAS

We can use the following SAS code to calculate the actual maximum variable lengths for a given domain:

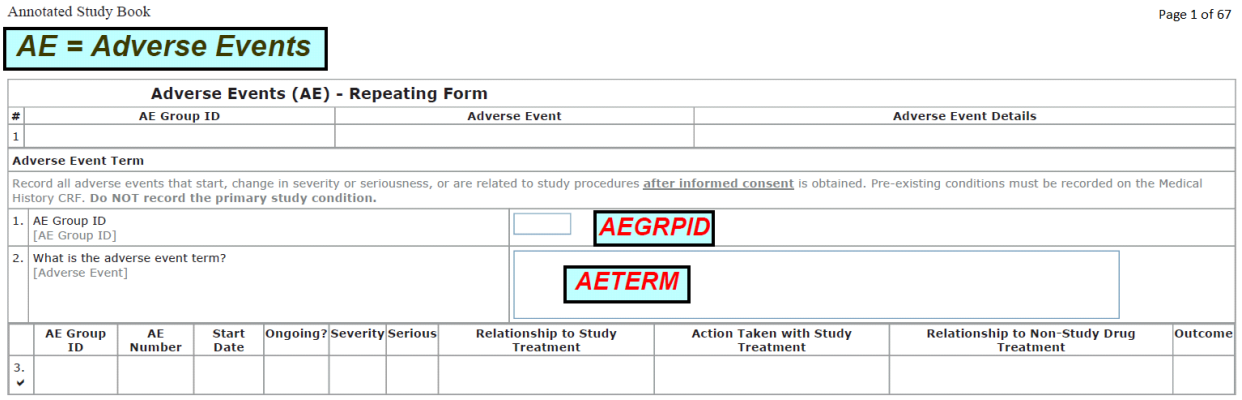
```
data domain;
  set sdtm.domain;
  array chars _character_;
  length variable $32;
  do over chars;
    variable = vname(chars);
    length = length(chars);
    output;
  end;
run;

proc sql;
  create table domain_maxlen as
  select variable, max(length) as maxlen
  from domain
  group by variable
  ;
quit;
```

The created SAS dataset `domain_maxlen` will contain the actual maximum lengths of the variables in the domains. This information can then be used to configure the SDTM specifications.

OBTAIN VARIABLE ORIGINS USING SAS

In the SDTM specification template, the ORIGIN column will have the values of “CRF Page xx” if the variable is from CRF. We can use SAS programs to extract the exact CRF page numbers from the SDTM annotated CRF file. Figure 2 is an example of the SDTM annotated CRF file. The “AE = Adverse Events” indicates the SDTM domain name, and AEGRPID and AETERM are the variable names in the AE domain. In fact, all these annotations are added to the related CRF pages as comments, and thus they can be extracted to an adobe acrobat forms document (.fdf) as shown in Figure 3. It is more important

that the ".fdf" file can be open as a text file and can then be converted to a SAS dataset as shown in Figure 4. From Figure 3 and 4, we can notice that the domain and variable names are displayed as "/Contents(domain name = domain description)/" and "/Contents(variable name)/". The page number will be displayed as "/Page xx/".



Annotated Study Book

Page 1 of 67

AE = Adverse Events

Adverse Events (AE) - Repeating Form

#	AE Group ID	Adverse Event	Adverse Event Details
1			

Adverse Event Term

Record all adverse events that start, change in severity or seriousness, or are related to study procedures **after informed consent** is obtained. Pre-existing conditions must be recorded on the Medical History CRF. **Do NOT record the primary study condition.**

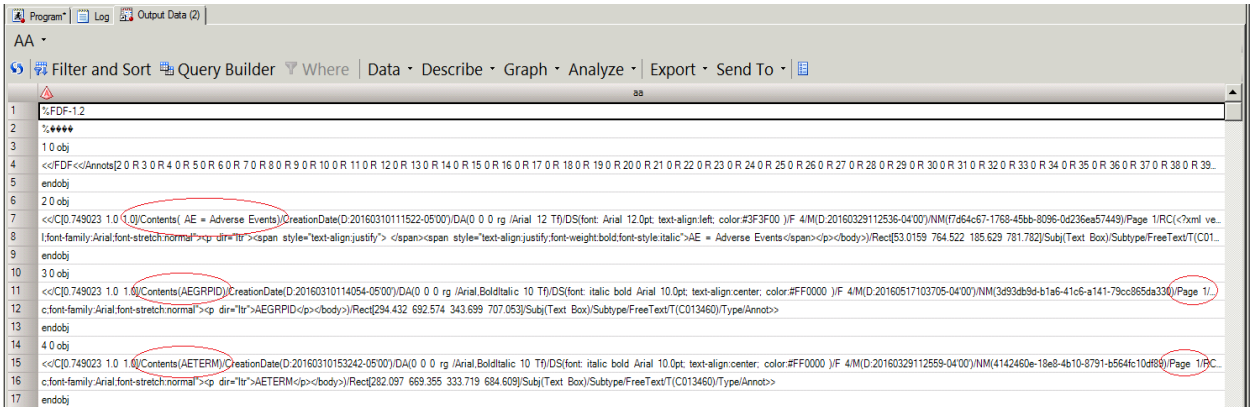
1.	AE Group ID [AE Group ID]	AEGRPID
2.	What is the adverse event term? [Adverse Event]	AETERM

	AE Group ID	AE Number	Start Date	Ongoing?	Severity	Serious	Relationship to Study Treatment	Action Taken with Study Treatment	Relationship to Non-Study Drug Treatment	Outcome
3.										

Figure 2. Annotated CRF

```
%FDF-1.2
%âãÏÓ
1 0 obj
<<FDF<</Annots[2 0 R 3 0 R 4 0 R 5 0 R 6 0 R 7 0 R 8 0 R 9 0 R 10 0 R 11 0 R 12 0 R 13 0 R 14 0 R 15 0 R 16 0 R 17 0 R 18 0 R 19 0 R 20 0 R 21 0 R 22 0 R 23 0 R
endobj
2 0 obj
<</C[0.749023 1.0 1.0]/Contents( AE = Adverse Events)/CreationDate(D:20160310111522-05'00')/DA(0 0 0 rg /Arial 12 Tf)/DS(font: Arial 12.0pt; text-align:left; col
l;font-family:Arial;font-stretch:normal"><p dir="ltr"><span style="text-align:justify"> </span><span style="text-align:justify;font-weight:bold;font-style:italic
endobj
3 0 obj
<</C[0.749023 1.0 1.0]/Contents(AEGRPID)/CreationDate(D:20160310114054-05'00')/DA(0 0 0 rg /Arial,BoldItalic 10 Tf)/DS(font: italic bold Arial 10.0pt; text-align
c;font-family:Arial;font-stretch:normal"><p dir="ltr">AEGRPID</p></body>)/Rect[294.432 692.574 343.699 707.053]/Subj(Text Box)/Subtype/FreeText/T(C013460)/Type/A
endobj
4 0 obj
<</C[0.749023 1.0 1.0]/Contents(AETERM)/CreationDate(D:20160310153242-05'00')/DA(0 0 0 rg /Arial,BoldItalic 10 Tf)/DS(font: italic bold Arial 10.0pt; text-align:
c;font-family:Arial;font-stretch:normal"><p dir="ltr">AETERM</p></body>)/Rect[282.097 669.355 333.719 684.609]/Subj(Text Box)/Subtype/FreeText/T(C013460)/Type/An
endobj
```

Figure 3. Adobe Acrobat Forms Document (.fdf)



Program* | Log | Output Data (2)

AA

Filter and Sort | Query Builder | Where | Data | Describe | Graph | Analyze | Export | Send To

	aa
1	%FDF-1.2
2	%ÿÿÿÿ
3	1 0 obj
4	<</FDF<</Annots[2 0 R 3 0 R 4 0 R 5 0 R 6 0 R 7 0 R 8 0 R 9 0 R 10 0 R 11 0 R 12 0 R 13 0 R 14 0 R 15 0 R 16 0 R 17 0 R 18 0 R 19 0 R 20 0 R 21 0 R 22 0 R 23 0 R 24 0 R 25 0 R 26 0 R 27 0 R 28 0 R 29 0 R 30 0 R 31 0 R 32 0 R 33 0 R 34 0 R 35 0 R 36 0 R 37 0 R 38 0 R 39...
5	endobj
6	2 0 obj
7	<</C[0.749023 1.0 1.0]/Contents(AE = Adverse Events)/CreationDate(D:20160310111522-05'00')/DA(0 0 0 rg /Arial 12 Tf)/DS(font: Arial 12.0pt; text-align:left; color:#3F3F00)/F 4/M(D:20160329112536-04'00')/NM(f7d64c67-1768-45bb-8096-0d236ea57449)/Page 1/RC(<?xml ve...
8	l;font-family:Arial;font-stretch:normal"> AE = Adverse Events </body>)/Rect[53.0159 764.522 185.629 781.782]/Subj(Text Box)/Subtype/FreeText/T(C01...
9	endobj
10	3 0 obj
11	<</C[0.749023 1.0 1.0]/Contents(AEGRPID)/CreationDate(D:20160310114054-05'00')/DA(0 0 0 rg /Arial,BoldItalic 10 Tf)/DS(font: italic bold Arial 10.0pt; text-align:center; color:#FF0000)/F 4/M(D:20160517103705-04'00')/NM(3d93db9d-b1a6-41c6-a141-79cc865da330)/Page 1/...
12	c;font-family:Arial;font-stretch:normal"> AEGRPID </body>)/Rect[294.432 692.574 343.699 707.053]/Subj(Text Box)/Subtype/FreeText/T(C013460)/Type/Annot>>
13	endobj
14	4 0 obj
15	<</C[0.749023 1.0 1.0]/Contents(AETERM)/CreationDate(D:20160310153242-05'00')/DA(0 0 0 rg /Arial,BoldItalic 10 Tf)/DS(font: italic bold Arial 10.0pt; text-align:center; color:#FF0000)/F 4/M(D:20160329112559-04'00')/NM(4142460e-18e8-4b10-8791-b564fc10df89)/Page 1/RC...
16	c;font-family:Arial;font-stretch:normal"> AETERM </body>)/Rect[282.097 669.355 333.719 684.609]/Subj(Text Box)/Subtype/FreeText/T(C013460)/Type/Annot>>
17	endobj

Figure 4. SAS Dataset of Adobe Acrobat Forms Document

The following is the SAS program to extract the CRF pages from the annotated CRF file:

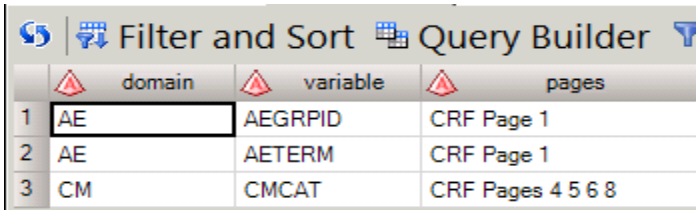
```
filename acrf "physical-path and filename of annotated CRF file";
```

```
/*domain list*/
%let domain=%str("CO" "DM" "SE" "SV" "CM" "EC" "EX" "PR" "XP" "AE" "CE" "DS"
"DV" "MH" "HO" "DA" "DD" "EG" "IE" "IS" "LB" "MB" "MI" "MO" "MS" "PC" "PP"
"PE" "QS" "RP" "RS" "SC" "SS" "TU" "TR" "VS" "FA" "SR");

/*read contents of annotated CRF*/
data acrf;
   length conts $ 32767;
   infile acrf lrecl=32767;
   input;
   conts=_infile_;
   if index(conts, 'Contents(') or index(conts, '/Page');
run;

/*use regular expression to extract the domain names, variable names, page
numbers, and background color*/
data acrf1;
  retain re1 re2 re3;
  length variable domain pageno tmpvar colvar $50;
  set acrf;
  if _N_=1 then do;
     re1 = prxparse("/(Contents\()((\w|\d|=| |'|,|\[|\]|\\)+)(\))/i");
     re2 = prxparse("/(\/Page +)(\d+)/i");
     re3 = prxparse("/^\<\<\/(C\[.+\])\/Contents/i");
  end;
  if prxmatch(re1,conts) then do;
     variable = prxposn(re1, 2, conts);
  end;
  if prxmatch(re2,conts) then do;
     pageno = prxposn(re2, 2, conts);
  end;
  if prxmatch(re3,conts) then do;
     colflg=1;
     colvar = prxposn(re3, 1, conts);
  end;
  variable=compbl(tranwrd(variable, '\r', ''));
  if index(variable, "=") then do;
     tmpvar=strip(scan(variable, 1, "="));
     if index(tmpvar, ' ')<length(tmpvar) and index(upcase(tmpvar), "WHEN")
then tmpvar=scan(tmpvar, 1, ' ');
     else if index(tmpvar, ' ')<length(tmpvar) then tmpvar=scan(tmpvar, -1, '
');
     if tmpvar in (&domain) then domain=tmpvar;
     else variable=tmpvar;
  end;

  if upcase(variable)='[NOT SUBMITTED]' or index(upcase(variable), 'SUPP') or
index(upcase(variable), 'RELREC') or index(upcase(variable), 'NOTE') or
variable='' then delete;
  keep domain variable pageno colvar;
run;

proc sort data=acrf1 out=acrf2 nodupkey;
   by pageno colvar domain variable;
run;
```

```
/*obtain domain name*/
data domain;
  set acrf2;
  where domain ne '';
run;

proc sort data=domain; by pageno colvar; run;

/*apply domain to related variables based on page and background color*/
data acrf3 (drop=domain_o);
  merge acrf2 (in=a rename=(domain=domain_o) where=(domain_o='')) domain
(in=b keep=pageno colvar domain);
  by pageno colvar;
  if a;
run;

proc sort data=acrf3; by colvar pageno; run;

/*handle the case when a domain has variables in more than one page */
data acrf4;
  retain domain;
  set acrf3 (rename=(domain=domain_o));
  by colvar pageno;
  if domain_o ne '' then domain=domain_o;
  keep domain variable pageno;
run;

/*handle the case with two or more variables separated by "/" */
data acrf5;
  length var1 var2 $50;
  set acrf4;
  if index(variable, '/')=0 then output;
  else do;
     var1=variable;
     i=1;
     var2=scan(var1, i, '/');
     do while (var2 ne ' ');
        variable=var2;
        output;
        i=i+1;
        var2=scan(var1, i, '/');
     end;
  end;
run;

proc sort data=acrf5; by domain variable pageno; run;


/*combine page numbers together if variables appear on more than one page*/
data acrf6;
  length pages $200;
  retain pages cnt;
  set acrf5;
  by domain variable pageno;
```

```
  if first.variable then do; pages=pageno; cnt=1; end;
  else do; pages=catx(' ', pages, pageno); cnt=cnt+1; end;

  if last.variable;
  if cnt=1 then pages=catx(' ', 'CRF Page', pages);
  else pages=catx(' ', 'CRF Pages', pages);
  keep domain variable pages;
run;
```

Figure 5 is an example of the extracted CRF pages of the variables in the SDTM domains.



	domain	variable	pages
1	AE	AEGRPID	CRF Page 1
2	AE	AETERM	CRF Page 1
3	CM	CMCAT	CRF Pages 4 5 6 8

Figure 5. CRF Pages of Variables in SDTM Domains

The code above uses regular regressions to extract the domain names, variable names, background colors, and related page numbers. The background colors can be used to handle the case when there are more than one domains in a page. The domain names and related variables will be matched if they have the same background colors. The code above can also handle the case when the variables in a domain spread over more than one pages. In order to demonstrate the main idea of this approach, the code above is a simplified version of the implementation, the uses can add more details into the code to fit their needs.

After the configuration information is automatically generated by SAS programs, we can then use it to configure the related parts of SDTM specifications by merging them together. The following is an example of SAS program to update the maximum variable lengths of the SDTM specifications:

```
libname specs xlsx <'physical-path and filename.xlsx'>

data specs.domain;
  merge specs.domain (in=a) domain_maxlen (in=b);
  by variable;
  if a;
  if b then saslength= maxlen;
run;
```

Even though the pre-defined contents of SASLENGTH column in the SDTM specification template can been updated by the actual maximum variable lengths which are automatically generated by SAS programs, we can also notice that the formats of the original specification template are also changed. In order to overcome this issue, we will propose a new approach to configure SDTM specifications in the next section.

	B	C	F	G	H	I	J	O	P
1	DATASET	VARIABLE	BUSINESS_ALGORITHM	STUDY_SPECIFIC_ALGORITHM	SUBMISSION_COMMENT	ORIGIN	LABEL	SASLENGTH	DATATYPE
2	DM	STUDYID	Pull SITEGUID from INF_SUBJECT where SUBJECT_ID =			CRF Page xx	Study Identifier	12	text

Figure 6. SDTM Specification Configured by SAS Only

UPDATE SDTM SPECIFICATIONS USING SAS AND VBA

From previous section, it is obvious that the format of Figure 6 is quite different from that of Figure 1. In some cases, the change of the formats might not be acceptable. This paper will propose a new approach to automatically configure the specification template while maintaining the original formats or even converting the formats to the required ones. This approach is implemented using SAS and VBA.

CREATE CONFIGURATION EXCEL FILE USING SAS

The first step of this approach is to use SAS to generate the excel file that contains the information that will be used to configure SDTM specifications. For example, if we want to replace SASLENGTH and ORIGIN columns with the actual maximum variable lengths and actual CRF page numbers in each domain, we can use the SAS code mentioned before to automatically obtain these values and store them to related SAS datasets. For the convenience of implementation, we can also name these datasets with their related domain names. After this step, we can then use the following code to export these datasets to related sheets in a excel file:

```
libname specs xlsx "physical-path and filename of SDTM specification";

/*get the names of all domains from the TABLES tab of the specification and
put in a macro variable*/
proc sql noprint;
      select dataset into: domains separated by " " from specs.tables;
quit;

libname specs clear;

libname update xlsx "physical-path and filename of configuration info file";


%macro loop();

/*get the name of each domain from the macro variable domains*/
%let cnt=1;
%let db=%sysfunc(scan(&domains, &cnt, %str( )));
%put db=&db;

%do %while (&db ne );
   /*output spec update information &db dataset into sheet &db*/
   %if %sysfunc(exist(&db)) %then %do;
      data update.&db;
         set &db;
      run;
   %end;

   %let cnt=%eval(&cnt+1);
   %let db=%sysfunc(scan(&domains, &cnt, %str( )));
%end;

%mend loop;

%loop;

libname update clear;
```

In the code above, all the domain names are obtained from the TABLES tab of the SDTM specification. A

macro variable is created to hold all these domain names. A macro function is used to extract each domain name from this macro variable, and check the existence of the related SAS configuration dataset. If the dataset is found, it will then be exported to the related sheet in the configuration excel file. In addition, the configuration excel file will be created if it does not exist before. Otherwise, the contents of this configuration excel file will be updated accordingly.

CONFIGURE SDTM SPECIFICATIONS USING VBA

The second part of this approach is to use VAB to configure the SDTM specifications based on the information contained in the configuration excel file generated before. The contents of the SDTM specifications will be updated while the original formats can be maintained or be converted to other required formats.

Create the User Interface of SDTM Specification Configuration

For the convenience of usage, a user interface of this configuration tool is constructed within Microsoft Excel. The user interface consists of 3 input text boxes and 3 buttons as shown in Figure 7. The user can input the file path and name of the SDTM specification into “primary excel file” input box, or click “browse” button next to it to locate the SDTM specification. Similarly, the user can use “secondary excel file” input box to input or browse the excel file containing the configuration information generated by SAS before. The user can also use this interface to select the columns in the SDTM specification to be configured. The default column name is set to SASLENGTH. The user can input other column name, or choose more than one columns to be configured, such as SASLENGTH ORIGIN. In addition, the use can leave it blank or input “ALL” to choose all the columns in the SDTM specification. After all these inputs, the user can then click the “configure” button to start the SDTM specification configuration.

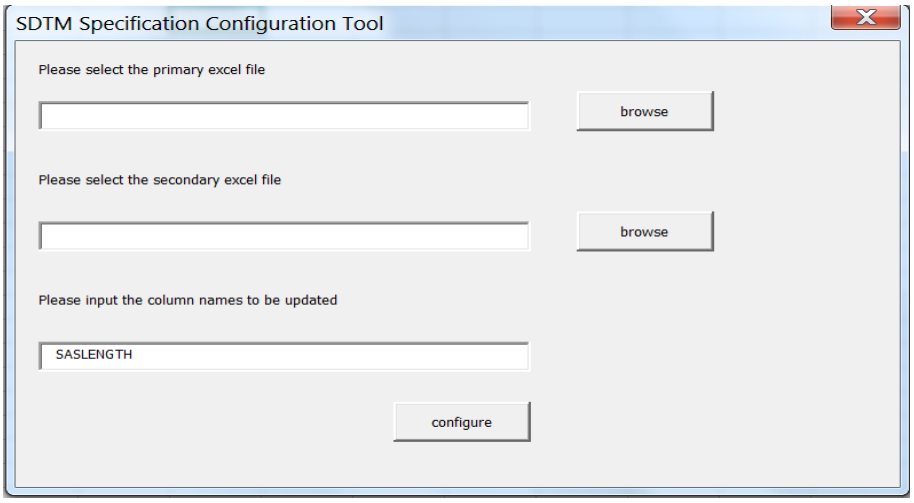


Figure 7. SDTM Specification Configuration User Interface 1

When the configuration is done, the Figure 8 window will be displayed. The user can decide whether to continue or stop the configuration.

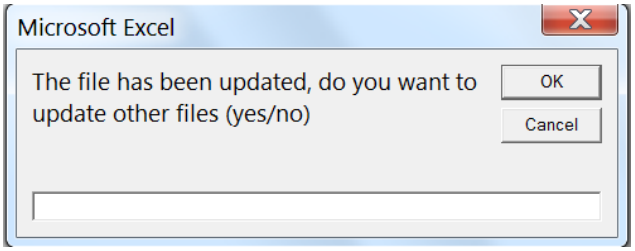


Figure 8. SDTM Specification Configuration User Interface 2

Select Excel Files

As mentioned before, the user can directly input the excel file path and name into the input text box. Also, the use can click the “browse” button to locate the excel files. The following is the VBA code that will be executed after the “browse” button is clicked:

```
Private Sub CommandButton_Click()

Application.DisplayAlerts = False

Dim strFileToOpen2 As String
Dim wkpathnm As String
Dim wkbook As Workbook

'open file browse window
strFileToOpen2 = Application.GetOpenFilename _
(Title:="Please choose a file to open", _
FileFilter:="Excel Files *.xls* (*.xls*),")
If strFileToOpen2 = "" Or strFileToOpen2 = "False" Then
MsgBox "No file selected.", vbExclamation, "Sorry!"
Exit Sub
Else
UserForm1.TextBox2.Value = strFileToOpen2

'check whether the excel file is already open
k = 0
For Each wkbook In Workbooks
   wkpathnm = wkbook.Path & "\" & wkbook.Name
   If wkpathnm = strFileToOpen2 Then
      k = 1
      Exit For
   End If
Next

'open the selected excel file
If k = 0 Then Workbooks.Open Filename:=strFileToOpen2

End If
End Sub
```

In the code above, the function `Application.GetOpenFilename` will pop up a window for the user to browse and select a file. The file filter will limit the file extensions to xls or xlsx. If the user does not select

any file, such as clicking “Cancel”, it will pop up a window to indicate “No file selected”. After the excel file is located, it will then check whether the selected excel file is already open or not. If not, the selected excel file will be open.

Configure SDTM Specifications

The main part of the implementation is to configure the selected SDTM specification based on the information stored in the configuration excel file. After clicking the “configure” button, the following VBA code will be executed to configure the SDTM specification automatically:

```
Private Sub Configure_Click()

Application.DisplayAlerts = False

'initialization
Dim wkbook As Workbook
Dim wksheet1 As Worksheet
Dim wksheet2 As Worksheet
Dim varselect()
Dim vartemp() As String
Dim varp()
Dim saslenp()
Dim vars()
Dim saslens()
Dim saslencolp As Single
Dim saslencols As Single
Dim varcolp As Single
Dim varcols As Single
Dim Rownum As Long
Dim colnum As Long
Dim Rownum1 As Long
Dim colnum1 As Long

Dim aa As String
Dim b1 As String
Dim b2 As String
Dim prim As String
Dim seco As String

b1 = UserForm1.TextBox1.Value
b2 = UserForm1.TextBox2.Value

'get the primary and secondary workbook names
For Each wkbook In Workbooks
   aa = wkbook.Name
   If b1 Like "*" & aa Then
      prim = aa
   ElseIf b2 Like "*" & aa Then
      seco = aa
   End If
Next

  updatefl = 0

  'loop through each work sheet of the primary file
  For Each wksheet1 In Workbooks(prim).Sheets
      Rownum = wksheet1.UsedRange.Rows.Count
      colnum = wksheet1.UsedRange.Columns.Count

      'get the columns to be configured
      If UserForm1.TextBox3.Value = "" Or UCase(UserForm1.TextBox3.Value) ="ALL" Then
         varselect = Application.WorksheetFunction.Transpose( _
```

```
        Application.WorksheetFunction.Transpose(wksheet1.Range(wksheet1.Cells(1, 1),
        wksheet1.Cells(1, colnum))))
      Else
        vartemp() = Split(UCase(UserForm1.TextBox3.Value))
        ReDim varselect(1 To UBound(vartemp) + 1)
        For i = 0 To UBound(vartemp)
            varselect(i + 1) = vartemp(i)
        Next
      End If

    'loop through each configuration column
    For Each varnm In varselect
     If varnm <> "" And UCase(varnm) <> "VARIABLE" Then

      update1fl = 0
      update11fl = 0
      update12fl = 0

      'check whether primary file has the configuration column
      For i = 1 To colnum
          If UCase(wksheet1.Cells(1, i).Value) = UCase(varnm) Then
             update11fl = 1
             saslencolp = i
          End If
          If UCase(wksheet1.Cells(1, i).Value) = "VARIABLE" Then
             update12fl = 1
             varcolp = i
          End If
          If update11fl = 1 And update12fl = 1 Then
             update1fl = 1
             Exit For
          End If
      Next

      If update1fl = 1 Then
         update2fl = 0

         'check whether secondary file has the required sheet (domain)
         For Each wksheet2 In Workbooks(seco).Sheets
             update3fl = 0
             If UCase(wksheet2.Name) = UCase(wksheet1.Name) Then
                update3fl = 1
             End If

             'check whether secondary file has the required column
             If update3fl = 1 Then
                Rownum1 = wksheet2.UsedRange.Rows.Count
                colnum1 = wksheet2.UsedRange.Columns.Count
                update21fl = 0
                update22fl = 0
                For i = 1 To colnum1
                    If UCase(wksheet2.Cells(1, i).Value) = UCase(varnm) Then
                       update21fl = 1
                       saslencols = i
                    End If
                    If UCase(wksheet2.Cells(1, i).Value) = "VARIABLE" Then
                       update22fl = 1
                       varcols = i
                    End If
                    If update21fl = 1 And update22fl = 1 Then
                       update2fl = 1
                       Exit For
```

```
                    End If
                Next
                'get the values of matched cells
                If update2fl = 1 Then
                   varp =
Application.WorksheetFunction.Transpose(wksheet1.Range(wksheet1.Cells(1, varcolp), _
                              wksheet1.Cells(Rownum, varcolp)))
                   saslenp =
Application.WorksheetFunction.Transpose(wksheet1.Range(wksheet1.Cells(1, saslencolp),
                              wksheet1.Cells(Rownum, saslencolp)))
                   vars =
Application.WorksheetFunction.Transpose(wksheet2.Range(wksheet2.Cells(1, varcols), _
                                wksheet2.Cells(Rownum1, varcols)))
                   saslens =
Application.WorksheetFunction.Transpose(wksheet2.Range(wksheet2.Cells(1, saslencols),
                                wksheet2.Cells(Rownum1, saslencols)))
                   'update the cell value with required format
                   For i1 = 2 To UBound(varp)
                       For i2 = 2 To UBound(vars)
                           If UCase(varp(i1)) = UCase(vars(i2)) Then
                              If saslenp(i1) <> saslens(i2) Then
                                 wksheet1.Cells(i1, saslencolp).Value = saslens(i2)
                                 wksheet1.Cells(i1, saslencolp).Font.Color =
                                                                      RGB(255, 0, 0)
                                 updatefl = 1
                              End If
                              Exit For
                           End If
                       Next
                   Next
                End If ' update2fl
             End If 'update3fl
          Next 'wksheet2
       End If 'update1fl
     End If 'check varnm
    Next 'varselect
  Next 'wksheet1

  'save the primary file if updated
  If updatefl = 1 Then Workbooks(prim).Save

  Workbooks(prim).Close
  Workbooks(seco).Close

  'check whether continue or stop the configuration
  dd = InputBox("The file has been updated, do you want to update other files
       (yes/no)")

  If UCase(dd) = "YES" Or UCase(dd) = "Y" Then
     Unload UserForm1
     UserForm1.Show
  Else
     Unload UserForm1
  End If
End Sub
```

The code above will identify the configuration columns based on the user input. It will then go through each sheet (domain) in the SDTM specification to check whether the sheet contains the designated configuration column. If yes, it will then check whether the secondary excel file also contains the same sheet (domain) name. If the answer is yes, it will continue to check whether the secondary sheet also

contains the designated column. If all these are satisfied, then the code will start to perform the configuration. During this step, it will first try to match the cells between the primary and secondary excel files based on the "key" columns, such as "VARIABLE" column. When the cells are matched, it will first check whether the values of the matched cells are the same. If the values are different, the cell value of the primary excel file will be updated by that of the secondary excel file. In order to locate the updates quickly, the updated parts can be displayed by certain colors, such as red color. This process will continue until all selected configuration columns in all sheets of the designated SDTM specification have been updated.

Figure 9 is the configured SDTM specifications. We can notice that the SASLENGTH of STUDYID has been changed from 11 to 12 (in red). In addition, the code above will only update the selected parts of file, and thus, the original values and formats of the rest parts of the SDTM specification will be maintained.

	DATASET	VARIABLE	BUSINESS_ALGORITHM	STUDY_SPECIFIC_ALGORITHM	SUBMISSION_COMMENT	ORIGIN	LABEL	SASLENGTH	DATATYPE
1									
2	DM	STUDYID	Pull SITEGUID from INF_SUBJECT where SUBJECT_ID = <input>.SUBJECT_ID. Pull SITEMNEMONIC from INF_SITE_UPDATE where CT_RECID = INF_SUBJECT.SITEGUID. Set to CDMS_IASCI_SITE_INV. USDYID_UNQ_STDY_ID_TXT where CDMS_IASCI_SITE_INV.INVID = INF_SITE_UPDATE.SITEMNEMONIC.			CRF Page xx	Study Identifier	12	text

Figure 9. SDTM Specification Configured by SAS and VBA

CONCLUSION

This paper proposes an automatic and efficient approach to configure SDTM specifications using SAS and VBA. The SAS programs are used to automatically generate the configuration information, and the VBA programs are used to configure the SDTM specifications based on the configuration information generate by SAS programs. This approach can not only improve the efficiency of SDTM specification configuration, it can also maintain the original formats of the SDTM specifications or update the formats to the required ones. In addition, this approach is not limited by the SAS running environment. The user-friendly interface makes it straightforward to apply this tool. Although this approach is developed initially for the configuration of SDTM specifications, it can also be directly applied to other areas to dynamically update the excel files with additional format requirements.

REFERENCES

Alexander, M., Kusleika, D. 2016. *Excel 2016 Power Programming with VBA Advanced multiplicity adjustment methods in clinical trials*. Indianapolis, IN : John Wiley & Sons, Inc.

FDA. 2017. "Study Data Technical Conformance Guide v4.0." Accessed November 2, 2017. https://www.fda.gov.

CDISC. 2013. "Study Data Tabulation Model Implementation Guide (SDTMIG) v3.2." Accessed November 2, 2017. https://www.cdisc.org.

CONTACT INFORMATION

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

Xingxing Wu
Eli Lilly and Company
wu_xingxing@lilly.com