PharmaSUG2011 - Paper CD01

A Cost-Effective SDTM Conversion for NDA Electronic Submission

Xiangchen (Bob) Cui, Vertex Pharmaceuticals, Cambridge, MA
Scott Moseley, Vertex Pharmaceuticals, Cambridge, MA
Min Chen, Vertex Pharmaceuticals, Cambridge, MA

ABSTRACT

When submitting clinical study data in electronic format to the FDA, sponsors are required to submit data definition tables (define.xml) and a reviewer guide (define.pdf), in addition to SDTM and ADaM datasets. The standardized, well-defined, and detailed define files minimize the time needed for FDA reviewers to familiarize the data, which can speed up the overall review process. The programming specification documentation serves as a part of a reviewer guide, as well as the documentation for programming validation. It is very crucial to ensure the consistency of the attribute of each variable among datasets, define files, and programming specification. It is highly desirable to automate this process to ensure technical accuracy and operational efficiency.

This paper describes a method that can streamline the process from SDTM/ADaM conversion to NDA electronic submission to achieve the goal, avoid the waste of the time and resources for verification of the consistency at the later stage, significantly reduce the time and work load to develop SAS® programs for SDTM/ADaM conversion and validation, and prepare define files.

INTRODUCTION

SDTM Conversion process for NDA Electronic Submission is composed of programming for SDTM conversion and its validation, preparing the Reviewer Guide documentation (define.pdf), and creating Data Definition Document (define.xml). A major part of define.xml is the “comment” column, which provides the detailed information for FDA reviewers to familiarize and understand the data. The programming specification documentation is a key part of the SDTM Conversion process, for it is used for programming and validation, the preparation of define.pdf, and the generation of the major parts of define.xml. A cost-effective way is very desirable, which can ensure SDTM Conversion meet the requirements described in SDTM Implementation Guide and in-house standards, and guarantee the consistency among SDTM datasets, programming specification, and define files. This paper introduces a methodology to achieve these goals. It was developed and used in our two Phase II and III studies for NDA submission. It turns out to be very successful.

We start with a Vertex SDTM Master Spreadsheet, an Excel® specification document, which follows CDISC SDTM Implementation Guide V3.1.1, and provides all variables for common domains. For each variable, the spreadsheet gives both the CDISC standard and Vertex standard attributes. The programming specification will be written in the “SDTM comment” column for each domain. The macro %cdiscspec is called to each spreadsheet and generates the programming specification in Rich Text Format (RTF) and a SAS dataset containing the variable attributes. The macro %mk0obs is called to the SAS dataset and generates the zero observation dataset. The calling of macro % sdtm_attrib to the zero observation dataset generates four global macro variables, which give the variables kept in the final domain and supplemental domain, and their attributes, respectively, and they will be used and facilitate SDTM conversion programming for both production and validation. Once each SDTM domain is fully developed and validated, the macros %all_spec and %write_vars_info are called to populate variable attributes and “SDTM comment” into define.xml at the final step for NDA submission. Since Excel® specification document is a unique source for SDTM datasets, programming specifications, and define files, consistency among them can be guaranteed. It avoids the waste of resources due to consistency checking. Further, it significantly reduces programming work load and error-prone manual processing to update all of three if there are some changes in SDTM dataset at any stage simply by updating the Excel® specification documents and rerunning all corresponding programs. This methodology is also applicable to the submission package for ADaM.

Figure 1 shows the process flow.

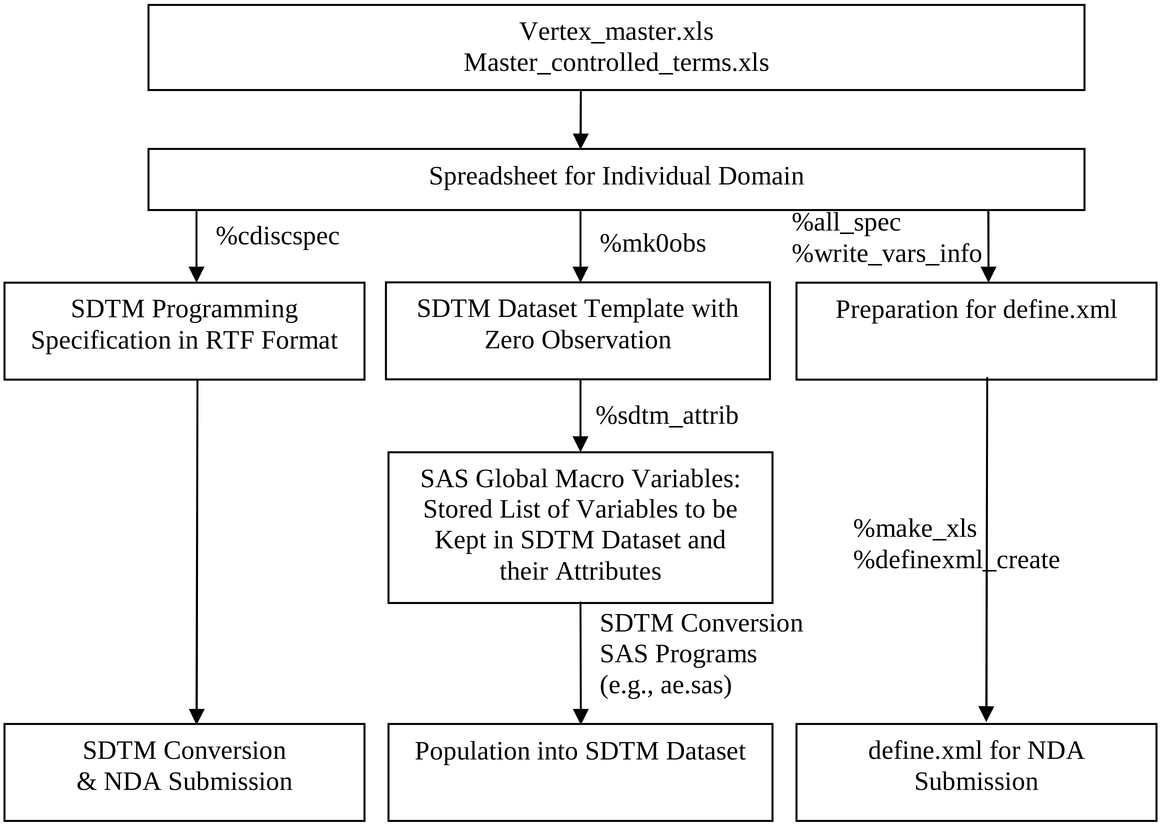


Figure 1 Overview of Process Flow

AN INTRODUCTION OF VERTEX MASTER SPREADSHEET FOR SDTM DOMAINS

Fig. 2 gives a sample specification document for AE domain filtered from the department tools library. The first part gives the information on the variable selection and the order of the variables, the second part gives the variable attributes from CDISC SDTM Implementation Guide standards, and the third part gives the variable attributes from Vertex SDTM Guide standards, which includes the comments for generating the variable.

Variable Selection and Order | Variable Attributes from CDISC SDTM | Variable Attributes from Vertex SDTM Guide

<table>
<tr><th></th><th>A</th><th>B</th><th>D</th><th>E</th><th>F</th><th>H</th><th>I</th><th>J</th><th>K</th><th>L</th><th>M</th><th>N</th><th>O</th><th>Q</th><th>R</th><th>S</th><th>AB</th><th>AC</th></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</td><td></td></tr>
<tr><td>2</td><td>Variable Selected</td><td>Vertex Default Variable</td><td>Seq. For Order</td><td>Observation Class</td><td>Domain Prefix</td><td>Variable</td><td>Variable Label</td><td>Type</td><td>Controlled Terms or Format</td><td>Origin</td><td>Role</td><td>CDISC Notes (for domains)</td><td>Core</td><td>Vertex Variable Status</td><td>Vertex Controlled Terminology</td><td>Vertex Variable Length</td><td>SDTM comment</td><td>Vertex Origin</td></tr>
<tr><td>3</td><td>Y</td><td>D</td><td>1</td><td>Events</td><td>AE</td><td>STUDYID</td><td>Study Identifier</td><td>Char</td><td></td><td>CRF</td><td>Identifier</td><td>Unique identifier for a study within</td><td>Req</td><td>Req</td><td></td><td>20</td><td>'ABC-xxx-xxx'</td><td>Sponsor Defined</td></tr>
<tr><td>4</td><td>Y</td><td>D</td><td>2</td><td>Events</td><td>AE</td><td>DOMAIN</td><td>Domain Abbreviation</td><td>Char</td><td>**AE</td><td>Derived</td><td>Identifier</td><td>Two-character abbreviation for</td><td>Req</td><td>Req</td><td>'AE'</td><td>2</td><td>'AE'</td><td>Derived</td></tr>
<tr><td>5</td><td>Y</td><td>D</td><td>3</td><td>Events</td><td>AE</td><td>USUBJID</td><td>Unique Subject Identifier</td><td>Char</td><td></td><td>Sponsor Defined</td><td>Identifier</td><td>Unique subject identifier within the submission.</td><td>Req</td><td>Req</td><td></td><td>40</td><td>'ABC-xxx-xxx' || '-' ||
substr(trim(left(RAW.AE.subjid)),1,3) || '-' ||
trim(left(RAW.AE.subjid))</td><td>Sponsor Defined</td></tr>
<tr><td>6</td><td>Y</td><td>D</td><td>4</td><td>Events</td><td>AE</td><td>AESEQ</td><td>Sequence Number</td><td>Num</td><td></td><td>CRF or Derived</td><td>Identifier</td><td>Sequence number given to ensure uniqueness within a dataset for a subject. Can be used to</td><td>Req</td><td>Req</td><td></td><td>8</td><td>Remove the empty records with RAW.AE.aeterm = ' '; ~
Sort RAW.AE by subjid aeseq; ~
SDTM.AE.aeseq = 1 for the first observation and increase by 1 for the subsequent</td><td>Derived</td></tr>
</table>

Figure 2 Sample Spreadsheet of Specification Implementation Guide

The Vertex controlled terminology is defined in the spreadsheets master_controlled_terms.xls from the department tools as shown in Fig. 3.

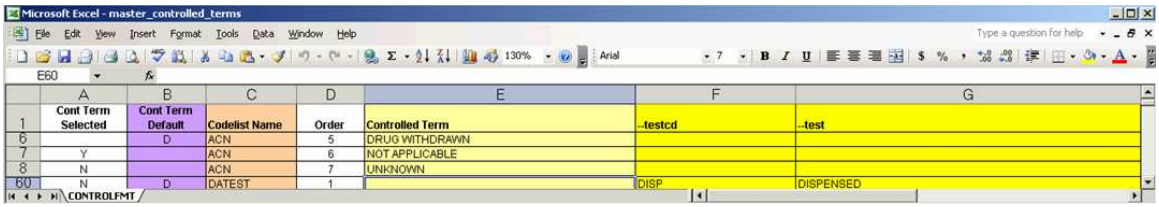


Figure 3 Spreadsheet of Controlled Terminology

If a specific domain exists in the master spreadsheet, it will be used to develop programming specification by selection of variables and addition of the mapping rules and/or derivation rules to the SDTM Comments column. Otherwise, one existing in the master spreadsheet can be used as a template for development of a new spreadsheet.

THE MACRO TO CREATE THE PROGRAMMING SPECIFICATION AND THE ZERO OBSERVATION DATASET

Each spreadsheet works for each domain. Hence the programming team can simultaneously work on different domains.

The macro %cdiscspec is used to generate the programming specification and a SAS dataset accommodating the variable attributes:

```
%macro cdiscspec(
       PATH      = ,  /* Path to all input files */
       INPUTFN   = ,  /* Filename of the Excel workbook
                         containing the variable specifications */
       SHEET     = ,  /* Name of the Excel(INPUTFN) worksheet
                         containing the variable specifications */
       AREA      = ,  /* Area of the worksheet containing the data */
       CTFN      = ,  /* Filename of the Excel workbook
                         containing the controlled terminology */
       CTSHEET   = ,  /* Name of the Excel worksheet
                         containing the controlled terms */
       OUTPUTFN  = ,  /* Filename of the Output RTF file */
       DATAOUT   = ,  /* Library name for output SAS datasets */
       VDATOUT   = ,  /* Output SAS datasets name
                         containing Variable Description */
       CTDATOUT  =    /* Output SAS datasets name
                         containing Controlled Terminology Information */
  );
```

Some samples of macro codes are shown as follows:

```
data spec(keep=f1-f29 memtype rename=(f1=var_sel f2=var_def f4=varnumc
          f6=memname f8=name f9=label f10=type f12=origin f13=role
          f14=note f15=core f17=ver_stat f18=ver_ct f19=lenc f28=rmap));
  set xlsin."&sheet$"n
  (dbSasType=( f1 =char100 f2 =char100  f3 =char100  f4 =char100  f5 =char100
               f6 =char100 f7 =char100  f8 =char100  f9 =char100  f10=char100
               f11=char100 f12=char100  f13=char100  f14=char100  f15=char100
               f16=char100 f17=char100  f18=char100  f19=char100  f20=char100
               f21=char100 f22=char100  f23=char100  f24=char100  f25=char100
               f26=char100 f27=char100  f28=char4000 f29=Char400)
  );
  if missing(f6) then delete;
  if f2="Vertex Default Variable" or missing (f8) then delete;
  if substr(f6,1,4)='SUPP' then memtype=2;
  else memtype=1;
  *** Generate Hard Return Sign in RTF ***;
  F28=tranwrd(f28,'~','~n');
  format _character_;
run;
...
*** Creates a single record per codelist with all domains including it;
Proc sort data=spec(where=(codelist ne ' ' and
     verify(codelist,'ABCDEFGHIJKLMNOPQRSTUVWXYZ_1234567890 ')=0))
     out=codes(keep=codelist name) nodupkey; by codelist name; run;
Proc transpose data=codes out=codes;
     by codelist;
     var name;
```

```
run;
Data codes;
     set codes;
     array col{*} $ col:;
     length vars $255;
     do i=1 to dim(col);
        if col{i} ne ' ' then vars = trim(vars) || ', ' || col{i};
     end;
     if length(vars) > 2 then vars = substr(vars,3);
  run;
...
**Create dataset containing information of TOC and Appendix of Control Terms;
Proc import datafile="&path&ctfn" out=ct;
     sheet="&ctsheet$";
     getnames=no;
run;
Data ct toc;
     set ct(rename=(f1=Cont_Term_Selected f2=Cont_Term_Defualt f3=codelist
                    f4=Varnum f5=_Controlled_Term  f6=_testcd f7=_test
                    f19=singmult) keep=f1-f7 f19);
     if (Cont_Term_Default ne ' ' or Cont_Term_Selected ne ' ')
         and index(upcase(Cont_Term_Selected),'N') = 0 then output ct;
     *** Identifies all domains & existed supplemental qualifier domains ***;
     if upcase(codelist) = 'DOMAIN' and _testcd ne ' ' then do;
        output toc;
        _testcd = 'SUPP' || _testcd;
        output toc;
     end;
run;
...
Data spec;set spec ct;run;
Proc sql;
     create table toc2 as
     select a.secnum, a.basemem, a.memtype, a.memname, toc._testcd, toc._test
     from (select distinct secnum, basemem, memtype, memname from spec) a
          left join toc on a.memname=toc._testcd
     order by basemem, memtype, _testcd;
quit;
Data toc;
     set toc2 toc(where=(_testcd='CONTTERM') in=cterm);
     if _testcd=' ' then _testcd = memname;
     if cterm then do;
        basemem = 'ZZZZ';
        secnum  = "See Section &CTSEC";
     end;
run;
```

The programming specification consists of three sections, including Table of Contents, is shown as follows:

3.1 Table of Contents

Section 3 -- Table of Contents of Target Vertex CDISC Datasets

Domain	Domain Description	Section
AE	Adverse Events	3.2.1
SUPPAE	Supplemental Qualifier for AE	3.2.2
CONTTERM	Controlled Terminology	See Section 3.3

3.2.1 Dataset: AE [One record per adverse event per subject]

Variable Ordinal #	Primary Key	Variable Name	Variable Label	Type/ Length	Codelist	Comments
1	Y	STUDYID	Study Identifier	Char/20		'ABC-xxx-xxx'
2	Y	DOMAIN	Domain Abbreviation	Char/2	AE	'AE'
3	Y	USUBJID	Unique Subject Identifier	Char/40		'ABC-xxx-xxx' || '-' || substr(trim(left(RAW.AE.subjid)),1,3) || '-' || trim(left(RAW.AE.subjid))
4	Y	AESEQ	Sequence Number	Num/8		Remove the empty records with RAW.AE.aeterm = ' '; Sort RAW.AE by subjid aeseq; SDTM.AE.aeseq = 1 for the first observation and increase by 1 for the subsequent observations per usubjid

3.3 Controlled Terminology

Codelist Name	Controlled Terminology Value
3.3.1 ACN Used by variables AEACN	
ACN	DOSE NOT CHANGED
	DOSE REDUCED
	DRUG INTERRUPTED
	DRUG WITHDRAWN
	NOT APPLICABLE

Figure 4 RTF Specification

The comments column guides programming activities of both SDTM conversion and validation by providing the mapping rules and derivation rules. A SAS dataset containing the attributes of each variable defined in each spreadsheet is output, and it is converted into a zero observation data set.

Figure 5 shows a dataset containing the attributes of the variables.

SAS System Viewer - [ae_vars]

File Edit View Window Help

	memname	name	label	type	len	varnum
1	AE	STUDYID	Study Identifier	Char	20	1
2	AE	DOMAIN	Domain Abbreviation	Char	2	2
3	AE	USUBJID	Unique Subject Identifier	Char	40	3
4	AE	AESEQ	Sequence Number	Num	8	4
5	AE	AESPID	Sponsor-Defined Identifier	Char	8	7
6	AE	AETERM	Reported Term for the Adverse Event	Char	200	8
7	AE	AEDECOD	Dictionary-Derived Term	Char	200	10
8	AE	AEBODSYS	Body System or Organ Class	Char	200	14
9	AE	AESEV	Severity/Intensity	Char	8	16
10	AE	AESER	Serious Event	Char	2	17
11	AE	AEACN	Action Taken with Study Treatment	Char	40	18
12	AE	AEREL	Causality	Char	20	20
13	AE	AEOUT	Outcome of Adverse Event	Char	40	23
14	AE	AECONTRT	Concomitant or Additional Trtmnt Given	Char	2	32
15	AE	AESTDTC	Start Date/Time of Adverse Event	Char	20	34
16	AE	AEENDTC	End Date/Time of Adverse Event	Char	20	36
17	AE	AEENRF	End Relative to Reference Period	Char	20	40
18	SUPPAE	STUDYID	Study Identifier	Char	20	1
19	SUPPAE	RDOMAIN	Related Domain Abbreviation	Char	2	2
20	SUPPAE	USUBJID	Unique Subject Identifier	Char	40	3
21	SUPPAE	IDVAR	Identifying Variable	Char	8	4
22	SUPPAE	IDVARVAL	Identifying Variable Value	Char	8	5
23	SUPPAE	QNAM	Qualifier Variable Name	Char	8	6
24	SUPPAE	QLABEL	Qualifier Variable Label	Char	40	7
25	SUPPAE	QVAL	Data Value	Char	200	8
26	SUPPAE	QORIG	Origin	Char	20	9
27	SUPPAE	QEVAL	Evaluator	Char	40	10

Figure 5 Intermediate Dataset Containing Attributes of the Variables

The sample of zero observation data set is shown in Figure 6.

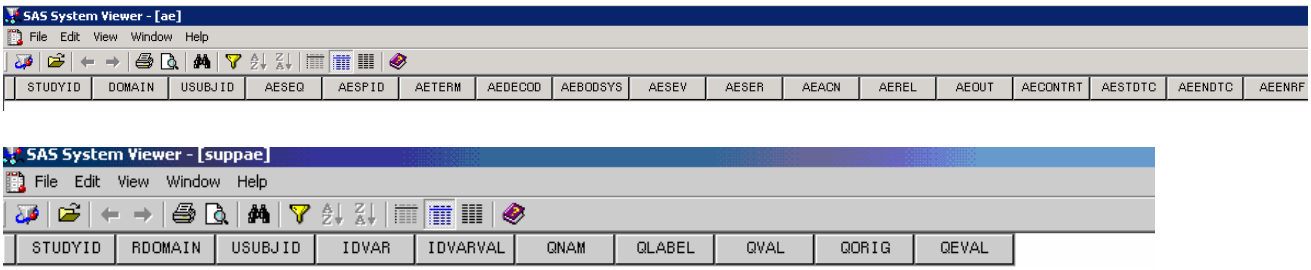


Figure 6 Zero Observation Data Set for SDTM AE and SUPPAE

The followings are the codes to generate the zero observation data set.

```
%macro mk0obs(input=,outlib=,showcontents=N);
  Proc sort data=&input out=vars;by memname varnum name;run;
  Data _null_;
       length t $1 type $8;
       set vars;
       by memname varnum name;
       if first.memname then call execute("data &outlib.." || left(memname)
          || '; ' || 'attrib ');
       if upcase(substr(type||'N',1,1)) = 'C' then t = '$';
       else t = ' ';
       if len = ' ' then do;
          if t = '$' then len = '1';
          else len = '8';
       end;
       call execute(name || ' length=' || compress(t||len) || ' label="' ||
            trim(left(translate(label,"'",'"'))) || ' ');
       if last.memname then call execute("; stop; run;");
  run;

  %if %upcase(%substr(&showcontents,1,1)) = Y %then %do;
    Proc contents data=&outlib.._all_;
    run;
  %end;
%mend;
%mk0obs(input=,outlib=,showcontents=N);
```

A MACRO TO CREATE GLOBAL MACRO VARIABLES FOR SELECTION OF VARIABLES AND THEIR ATTRIBUTES

After an individual RTF programming specification is created, the programmers append the RTF specification as the third part to the end of an individual Word specification, which contains the general information as the first part and the study specific information as the second part. The programming specification in MS Word format will be reviewed by the team. After the team's approval, the programmers can generate the final SDTM datasets following the mapping rules and derivation rules by using the zero-observation dataset.

The macro % sdtm_attrib generates four global macro variables: keep, keep_supp, attrib, and attrib_supp from a zero-observation dataset, which give the variables kept in the final domain and supplemental domain, and their attributes, respectively.

```
%macro sdtm_attrib(dsin     = ,
                   libin    = ,
                   template = ,
                   domain   = ,
                   suppqual =
                   );
  %local  domain suppqual;
  %global attrib keep attrib_supp keep_supp drop_qc drop_sys;
  *** Creates a datasets summarizing the content of all 0-obs datasets ***;
  Proc contents data=&template.._all_ out=_cont_template noprint;
  run;
  Data _vars; *** Extract Vertex SDTM specification for all datasets ***;
       set &libin..&dsin;
  run;
```

```
  %*** Get output dataset(s) variable attributes       ****;
  Proc sql noprint;
       select trim(left(name)) into: keep separated by ' '
       from _vars
       where upcase(memname) = (upcase("&domain"))
       order by varnum;
       select trim(left(name)) into: keep_supp separated by ' '
       from _cont_template
       where upcase(memname) = (upcase("&suppqual"))
       order by varnum;
       select trim(left(name)) into: drop_qc separated by ' '
       from _cont_template
       where upcase(memname) = (upcase("&suppqual"))
       and   upcase(name) not in ('STUDYID','USUBJID')
       order by varnum;
       select trim(left(name)) || " label='" || trim(left(label)) || "'
              length=" ||
              case when type = "Char" then '$'
              else ' '
              end || trim(left(len)) into: attrib separated by ' '
       from &libin..&dsin
       where upcase(memname) = (upcase("&domain"))
       order by varnum;
       select trim(left(upcase(name))) || " label='" || trim(left(label)) ||
              "' length=" ||
              case when type = 2 then '$'
              else ' '
              end || trim(left(put(length,best.))) into: attrib_supp
              separated by ' '
       from _cont_template
       where upcase(memname) = (upcase("&suppqual"))
       order by varnum;
  quit;
```

A SAMPLE OF SAS CODES FOR AE DOMAIN TO POPULATE VARIABLES AND THEIR ATTRIBUTES FROM GLOBAL MACRO VARIABLES

Four global macro variables: &keep, &keep_supp, &attrib, and &attrib_supp, generated from **%sdtm_attrib**, will be applied into SDTM mapping programs. A sample of SAS codes for AE domain below shows how these macro variables are used to generate SDTM variables specified in Vertex Master Spreadsheet. The methodology simplifies and facilitates the SAS programming for both production and validation, and results in significant reduction of programming work load and error-prone manual process to develop and validate the required variables and their attributes for SDTM.

```
%let domain     = AE;
%let suppdomain = suppae;
%let pre        = ae_;
*** Extract the attributes of the variables from 0-observation dataset ***;
%sdtm_attrib(dsin=&pre.vars, libin=sdtmspec, template=sdtmtmpl, domain=&domain.,
suppqual=&suppdomain.);
*** Create the SDTM domain ***;
Data &domain.;
     attrib &attrib.;
     retain &domain.seq 0;
     set rawdata(where=(&pre.aeterm ne '' ));
     by &pre.site &pre.subjid;
     domain  = upcase("&domain.");
     studyid = "&study_lbl.";
     usubjid = trim(left("&study_lbl.")) || "-" || substr(trim(left(&pre.subjid)),1,3) ||
               "-" || trim(left(&pre.subjid));
...
run;
...
*** Output SDTM datasets ***;
Data wsdtm.&domain(keep=&keep label='Adverse Events');
     attrib &attrib.;
     set &domain;
```

```
run;
data wsdtm.supp&domain(keep=&keep_supp label='Supplemental Qualifier for
                        AE');
     attrib &attrib_supp;
     set supp&domain;
run;
```

The resolutions of these four macro variables above inside AE.log are as follows.

```
%put &keep;
STUDYID DOMAIN USUBJID AESEQ AESPID AETERM AEMODIFY AEDECOD AEBODSYS AESEV AESER AEACN
AEREL AEOUT AECONTRT AESTDTC AEENDTC AEENRF

%put &attrib;
STUDYID label='Study Identifier' length=$20 DOMAIN label='Domain Abbreviation' length=$2
USUBJID label='Unique Subject Identifier' length=$40 AESEQ label='Sequence Number' length=
8 AESPID label='Sponsor-Defined Identifier' length=$8 AETERM label='Reported Term for the
Adverse Event' length=$120 AEMODIFY label='Modified Reported Term' length=$120 AEDECOD
label='Dictionary-Derived Term' length=$120 AEBODSYS label='Body System or Organ Class'
length=$120 AESEV label='Severity/Intensity' length=$8 AESER label='Serious Event'
length=$2 AEACN label='Action Taken with Study Treatment' length=$40 AEREL
label='Causality' length=$20 AEOUT label='Outcome of Adverse Event' length=$40 AECONTRT
label='Concomitant or Additional Trtmnt Given' length=$2 AESTDTC label='Start Date/Time of
Adverse Event' length=$20 AEENDTC label='End Date/Time of Adverse Event' length=$20 AEENRF
label='End Relative to Reference Period' length=$20

%put &keep_supp;
STUDYID RDOMAIN USUBJID IDVAR IDVARVAL QNAM QLABEL QVAL QORIG QEVAL

%put &attrib_supp;
STUDYID label='Study Identifier' length=$20 RDOMAIN label='Related Domain Abbreviation'
length=$2 USUBJID label='Unique Subject Identifier' length=$40 IDVAR label='Identifying
Variable' length=$8 IDVARVAL label='Identifying Variable Value' length=$8 QNAM
label='Qualifier Variable Name' length=$8 QLABEL label='Qualifier Variable Label'
length=$40 QVAL label='Data Value' length=$200 QORIG label='Origin' length=$20 QEVAL
label='Evaluator' length=$40
```

AUTOMATE CREATION OF SDTM PROGRAMMING SPECIFICATION FOR ALL DOMAINS

After each individual SDTM Programming Specifications is finalized, first a macro **%all_spec** is called to iteratively read and convert each Excel® specification document into a SAS data set, and combine these SAS data sets into one, which will be used to create one final RTF specifications by another macro %write_spec. The combined SAS dataset, called ALLSPECS.sas7bdat, will be used to prepare variable metadata for define.xml later.

%write_spec reads ALLSPECS.sas7bdat and spreadsheets for controlled terminologies, and generates TOC (table of contents), Programming Specifications, and controlled terminology for SDTM Programming Specifications in RTF format, shown in Figure 4.

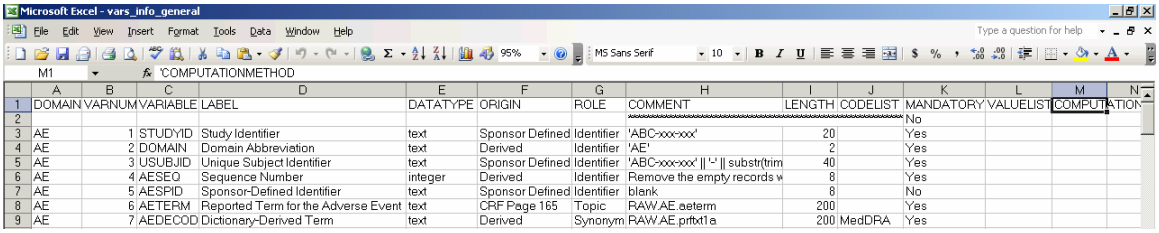
```
%let sheet   = V3.1.1 Domains;
%let allspecs = ae cm co ct dc dm ds eg ex fs hc hu ie lb mh pe qs sa sc se
                sq sv ta te ti ts tv vd vs xp;
%macro all_spec;
%let j=1;
%let onespec=%scan(&allspecs,&j," ");
%do %while (%quote(&onespec)^=());
  libname xlsin excel "E:\Final\data\sdtm\specs\&onespec._master_950108.xls"
          mixed=yes scantext=no dbmax_text=32000 getnames=no;
  data temp;
        set xlsin."&sheet$"n (dbSasType=(...));
        length driver $10;
        retain dnum &j;
        ...
        driver=upcase(strip("&onespec"));
        keep f1-f29 driver dnum memtype;
 run;
 proc append base=allspecs new=temp;run;
 %let j=%eval(&j + 1);
 %let onespec=%scan(&allspecs,&j," ");
%end;
```

```
%mend all_spec;
%all_spec;
```

PREPARATION FOR DEFINE.XML AND GENERATION OF VARIABLE LEVEL SPREADSHEET FROM AN EXCEL® SPECIFICATION DOCUMENT

Metadata file define.xml will be created by invoking macros %make_xls and %definexml_create, developed in-house. These macros need information including Study Level Spreadsheet, Domain Level Spreadsheet (Dataset Metadata), Variable Level Spreadsheet (Variable Metadata), Value Level Spreadsheet (Value Level Metadata), Computational Algorithm Spreadsheet, and Controlled Terminology/Format Spreadsheet,

The department tools library already provides five Excel file samples. All samples, except variable metadata, can be easily prepared. However vars_info_general.xls, as shown in Figure 7, is the most difficult to manually be prepared. There are large numbers of variables in clinical studies. For example, there are more than 550 variables in one of our studies for NDA submission. Another reason is that COMMENT column and ORIGIN column in the variable level spreadsheet are different from one study to another. They are study-specific. ALLSPECS.sas7bdat from Spreadsheet for SDTM Domains will be used to generate vars_info_general.xls by a macro %write_vars_info, shown as follows.


Microsoft Excel - vars_info_general

File Edit View Insert Format Tools Data Window Help

M1 'COMPUTATIONMETHOD

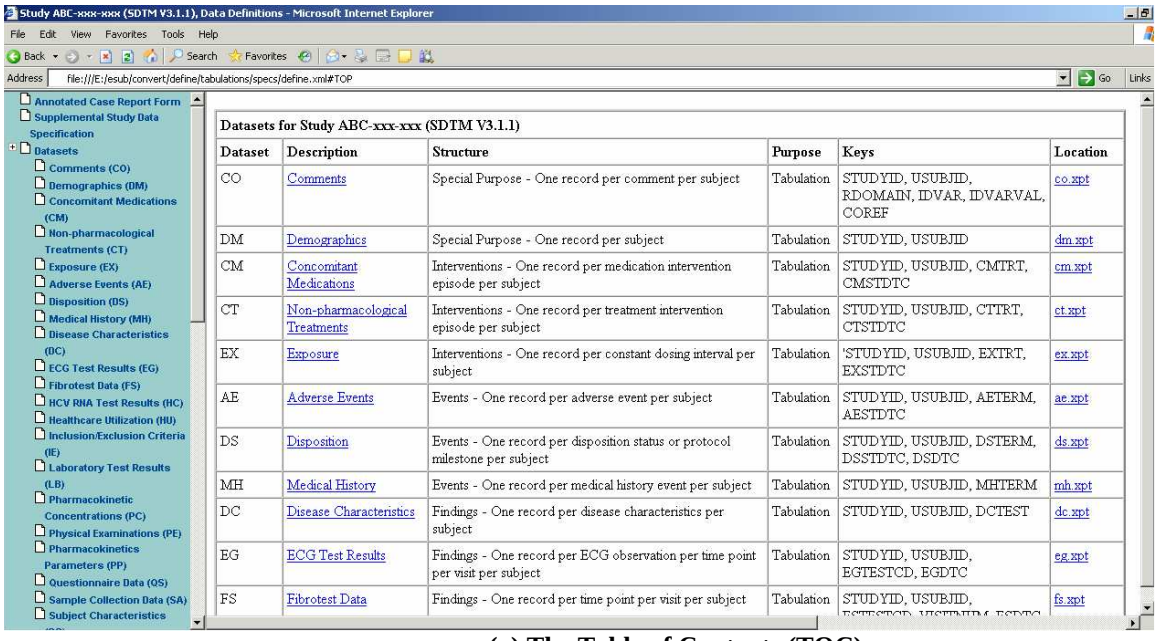
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	DOMAIN	VARNUM	VARIABLE	LABEL	DATATYPE	ORIGIN	ROLE	COMMENT	LENGTH	CODELIST	MANDATORY	VALUELIST	COMPUTATION	
2											No			
3	AE	1	STUDYID	Study Identifier	text	Sponsor Defined	Identifier	'ABC-xxx-xxx'	20		Yes			
4	AE	2	DOMAIN	Domain Abbreviation	text	Derived	Identifier	'AE'	2		Yes			
5	AE	3	USUBJID	Unique Subject Identifier	text	Sponsor Defined	Identifier	'ABC-xxx-xxx' || '-' || substr(trim	40		Yes			
6	AE	4	AESEQ	Sequence Number	integer	Derived	Identifier	Remove the empty records w	8		Yes			
7	AE	5	AESPID	Sponsor-Defined Identifier	text	Sponsor Defined	Identifier	blank	8		No			
8	AE	6	AETERM	Reported Term for the Adverse Event	text	CRF Page 165	Topic	RAW.AE.aeterm	200		Yes			
9	AE	7	AEDECOD	Dictionary-Derived Term	text	Derived	Synonym	RAW.AE.prftxt1a	200	MedDRA	Yes			

Figure 7 Preparation of define.xml – Variable Level Spreadsheet (vars_info_general.xls)

```
%let excel_path  = E:\esub\convert\define\tabulations;
libname sdtm "E:\esub\source\data\sdtm";
libname allspecs "&excel_path.";
%macro write_vars_info(libname=,dsin=);
proc sort data=&libname..&dsin.(rename=(f6=memname f8=name f9=label
          f10=type)) out=allspecs;
  by memname name;
  where not missing(memname) and
        (upcase(f1)='Y' or (upcase(f2)='D' and upcase(f1) ne 'N'));
run;
*** Generate a line with characters long enough to avoid the truncation ***;
*** When reading the generated vars_info excel by the department macros ***;
data longline;
  length longline $4000 memname $100 name $100;
  name    = ' ';memname = ' ';longline = repeat("*",4000);
run;
data allvars(rename=(name=variable));
  merge allspecs (in=a) longline (in=b);
  by memname name;
  length DOMAIN $10 DATATYPE $20 ORIGIN $300 ROLE $20 CODELIST $20 MANDATORY
         $10 COMMENT $4000 VALUELIST $20 COMPUTATIONMETHOD $20;
  retain varnum;
  if first.memname and not missing(memname) then varnum = 0;
  else if not missing(memname) then varnum + 1;
  ** The line with characters long enough should be output in the first row;
  if a then sort=2;
  if b then sort=1;
  if index(upcase(type), "NUM")  then datatype="float";
  if index(upcase(type), "CHAR") then datatype="text";
  *** Use Vertex Origin, if it is missing, use CDISC SDTM Origin ***;
  if not missing(f29) then origin = strip(f29);
  else origin = strip(f12);
  role = strip(f13);
  codelist = strip(f18);
  length = input(strip(f19),best.);
  if b then comment  = strip(longline);
  else comment = strip(f28);
  domain   = strip(memname);
```

```
    if upcase(f17) = 'REQ' then mandatory = 'Yes';
    else mandatory = 'No';
    if memname ne ' ' then do;
       if substr(strip(name),3) in ('ENRF','STRF','ENDY','STDY','DY') and
          strip(name) not in ('MHENRF')
       then ComputationMethod = substr(strip(name),3);
       else ComputationMethod = ' ';
       if substr(strip(name),3) in ('TESTCD') and strip(name) ne 'IETESTCD'
       then do;
          valuelist = 'Yes';
          codelist  = ' ';
       end;
       else if strip(name) = 'QNAM' then valuelist = 'Yes';
       else ValueList = ' ';
       if substr(strip(name),3) in ('SEQ','DY') or
          strip(name) in ('VISITNUM','VISITDY','TAETORD')
       then datatype = "integer";
    end;
    comment = strip(tranwrd(comment,'~',''));
    *** Define codelist ***;
    if strip(upcase(codelist)) not in ('SEX','NY','NYNULL') and
       strip(upcase(scan(codelist,1,'.'))) not in ('MEDDRA','WHODD')
    then codelist = ' ';
    keep domain varnum name label datatype origin role comment length codelist
         mandatory valuelist ComputationMethod sort;
run;
proc sort data=allvars;by sort domain varnum;run;
%mend;
%write_vars_info(libname=allspecs,dsin=allspecs);
```

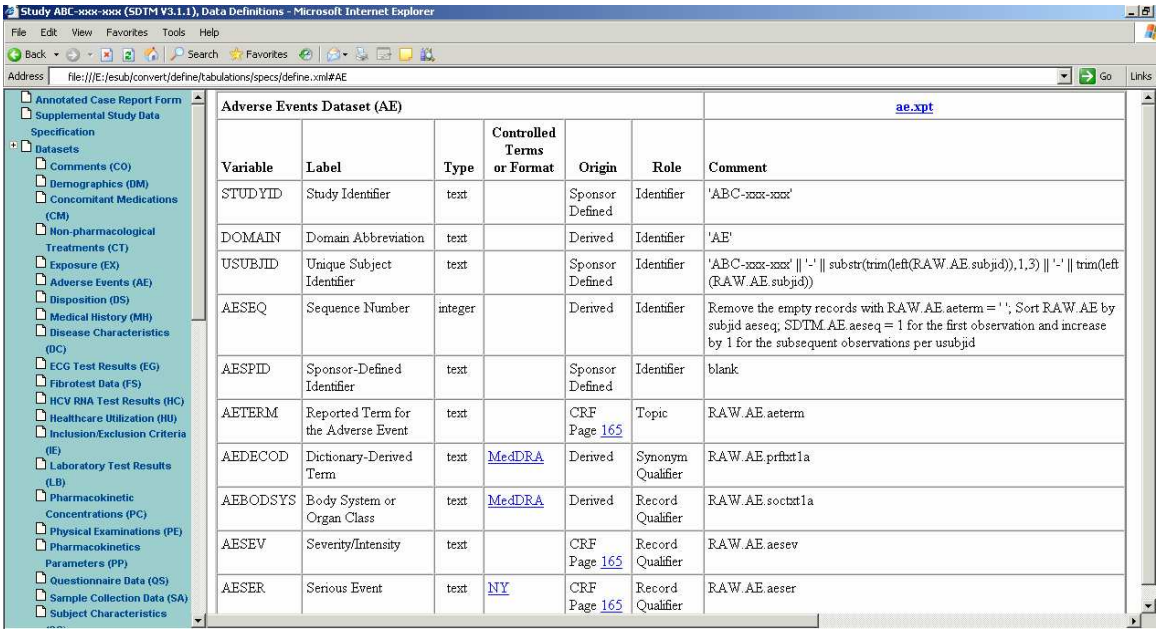
Define.xml is generated by calling %definexml_create. Figure 8 (a) shows an example of TOC for define.xml. The detailed mapping rules, derivation rules, and origin information in the programming specification, in addition to variable attributes, are shown in Figure 8 (b).



Datasets for Study ABC-xxx-xxx (SDTM V3.1.1)

Dataset	Description	Structure	Purpose	Keys	Location
CO	Comments	Special Purpose - One record per comment per subject	Tabulation	STUDYID, USUBJID, RDOMAIN, IDVAR, IDVARVAL, COREF	co.xpt
DM	Demographics	Special Purpose - One record per subject	Tabulation	STUDYID, USUBJID	dm.xpt
CM	Concomitant Medications	Interventions - One record per medication intervention episode per subject	Tabulation	STUDYID, USUBJID, CMTRT, CMSTDTC	cm.xpt
CT	Non-pharmacological Treatments	Interventions - One record per treatment intervention episode per subject	Tabulation	STUDYID, USUBJID, CTTRT, CTSTDTC	ct.xpt
EX	Exposure	Interventions - One record per constant dosing interval per subject	Tabulation	'STUDYID, USUBJID, EXTRT, EXSTDTC	ex.xpt
AE	Adverse Events	Events - One record per adverse event per subject	Tabulation	STUDYID, USUBJID, AETERM, AESTDTC	ae.xpt
DS	Disposition	Events - One record per disposition status or protocol milestone per subject	Tabulation	STUDYID, USUBJID, DSTERM, DSSTDTC, DSDTC	ds.xpt
MH	Medical History	Events - One record per medical history event per subject	Tabulation	STUDYID, USUBJID, MHTERM	mh.xpt
DC	Disease Characteristics	Findings - One record per disease characteristics per subject	Tabulation	STUDYID, USUBJID, DCTEST	dc.xpt
EG	ECG Test Results	Findings - One record per ECG observation per time point per visit per subject	Tabulation	STUDYID, USUBJID, EGTESTCD, EGDTC	eg.xpt
FS	Fibrotest Data	Findings - One record per time point per visit per subject	Tabulation	STUDYID, USUBJID,	fs.xpt

(a) The Table of Contents (TOC)



Adverse Events Dataset (AE) — ae.xpt

Variable	Label	Type	Controlled Terms or Format	Origin	Role	Comment
STUDYID	Study Identifier	text		Sponsor Defined	Identifier	'ABC-xxx-xxx'
DOMAIN	Domain Abbreviation	text		Derived	Identifier	'AE'
USUBJID	Unique Subject Identifier	text		Sponsor Defined	Identifier	'ABC-xxx-xxx' || '-' || substr(trim(left(RAW.AE.subjid)),1,3) || '-' || trim(left(RAW.AE.subjid))
AESEQ	Sequence Number	integer		Derived	Identifier	Remove the empty records with RAW.AE.aeterm = ' '; Sort RAW.AE by subjid aeseq; SDTM.AE.aeseq = 1 for the first observation and increase by 1 for the subsequent observations per usubjid
AESPID	Sponsor-Defined Identifier	text		Sponsor Defined	Identifier	blank
AETERM	Reported Term for the Adverse Event	text		CRF Page 165	Topic	RAW.AE.aeterm
AEDECOD	Dictionary-Derived Term	text	MedDRA	Derived	Synonym Qualifier	RAW.AE.prftxt1a
AEBODSYS	Body System or Organ Class	text	MedDRA	Derived	Record Qualifier	RAW.AE.soctxt1a
AESEV	Severity/Intensity	text		CRF Page 165	Record Qualifier	RAW.AE.aesev
AESER	Serious Event	text	NY	CRF Page 165	Record Qualifier	RAW.AE.aeser

(b) The Data Definition Table

Figure 8 Sample of define.xml

CONCLUSION

Since the SDTM dataset structure, the programming specification, and define.xml are all generated from the Excel® specification documents, the methodology used in our NDA submission ensures the consistency in the entire study from SDTM Conversion to NDA Electronic Delivery, and achieve the high quality of submission. When any revision and/or adding new domains are needed in the late stage, updating and/or adding a new Excel® specification for domains, and rerunning all corresponding SAS programs can accomplish the tasks.

This paper introduced a streamline process to generate SDTM dataset programming specifications for SDTM Conversion and QC activities, and prepare information for define.xml from Vertex SDTM Master Spreadsheet. The methodology guarantees the consistency among the specifications, SDTM dataset, and SDTM metadata file define.xml, enhances the submission quality, and achieves the cost-effectiveness and the efficiency. We hope the methodology and the SAS codes provided in this paper can assist you in saving your time and resources for clinical study reporting, especially for NDA submission.

REFERENCES

1. CDISC Submission Data Standards Team. “Study Data Tabulation Model Implementation Guide: Human Clinical Trials”, August 2005.
http://www.cdisc.org/content1605

2. Ellen Xiao. (2010). “SDTM Attribute Checking Tool”. SAS Global Forum, April 2010.

3. Alan Meier. (2009) ” Implementation Plan for CDISC SDTM & ADaM Standards at MedImmune”, PharmaSUG, June 2009.

4. Misha Rittmann. (2010) “Automating the Link between Metadata and Analysis Datasets”, PharmaSUG, May 2010.

ACKNOWLEDGEMENTS

Appreciation goes to SDTM Working Group for SDTM Master Spreadsheet, and Dean Gittleman and Kelly Blackburn for their review and comments.

CONTACT INFORMATION

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

Name: Xiangchen (Bob) Cui, Ph.D.
Enterprise: Vertex Pharmaceuticals, Inc.
Address: 88 Sidney Street
City, State ZIP: Cambridge MA, 02139
Work Phone: 617-444-6069
Fax: 617-460-8060

E-mail: xiangchen_cui@vrtx.com

Name: Scott Moseley, M.S.
Enterprise: Vertex Pharmaceuticals, Inc.
Address: 88 Sidney Street
City, State ZIP: Cambridge MA, 02139
Work Phone: 617-444-6162
Fax: 617-460-8060
E-mail: scott_moseley@vrtx.com

Name: Min Chen**,** Ph.D.
Enterprise: Vertex Pharmaceuticals, Inc.
Address: 88 Sidney Street
City, State ZIP: Cambridge MA, 02139
Work Phone: 617-444-7134
Fax: 617-460-8060
E-mail: min_chen@vrtx.com