PharmaSUG2016 - Paper SS08

A SAS® Macro Tool to Automate Generation of Define.xml V2.0 from SDTM Specification for FDA Submission

Min Chen, Alkermes Inc., Waltham, MA

Xiangchen (Bob) Cui, Alkermes Inc., Waltham, MA

ABSTRACT

A define-xml file is required to be submitted in FDA electronic submission, in addition to SDTM and ADaM datasets. An insufficiently documented define.xml frequently arouses a common complaint from FDA reviewers. Compared to define.xml Version 1.0, define.xml version 2.0 is a more powerful and user-friendly tool.

An SDTM programming specification spreadsheet provides the SDTM mapping rules and derivation rules for SDTM programming and QC, which can naturally serve as a define file. In this paper, the standard SDTM specification spreadsheets were re-designed for define.xml V2.0 new features, which provided complete details for derived variables. A metadata-driven SAS macro tool was developed to automate the creation of define.xml V2.0 from CDSIC SDTM specification spreadsheets, ensure the consistency of the two files, and achieve technical accuracy and operational efficiency. We hope the methodology and sample code provided in this paper can spare your resources and energies with FDA submission.

INTRODUCTION

FDA Center for Drug Evaluation and Research (CDER) and Center for Biologics Evaluation and Research (CBER) considered define files “critical component of data submission” and “a properly functioning define-xml file is an important part of the submission of standardized electronic datasets” [1]. They requested that “sponsors should make certain that every data variable’s codelist, origin, and derivation is clearly and easily accessible from the define file” as “an insufficiently documented define file is a common deficiency that reviewers have noted.”

CDISC XML Technologies Team released the Define-XML V2.0 specification in March 2013. The Define-XML V2.0 release package can be downloaded from the CDISC website (http://www.cdisc.org/define-xml). The distribution package includes: CDISC Define-XML Specification Version 2.0 [2], Define-XML V2.0 schema, an SDTM based define-xml example and its html rendition, an ADaM based define-xml example and its html rendition, and XSL stylesheet referenced by the two define-xml examples. CDISC Define-XML Specification Version 2.0 describes an updated Define-XML V2.0 model that is used to describe CDISC SDTM, SEND and ADaM datasets for the submission purpose. Our SAS Macro tool to create define-xml file was developed mainly based on this document.

SDTM Define-XML provides metadata for describing data sets about:

• Study (Study Name, Description, Protocol name, …)

• Datasets (Dataset Name, Description, Structure, Keys, Dataset Location, ...)

• Variables (Variable Name, Label, Key, Data Type, Length, controlled terminology, ...)

• Value Level Metadata

• Controlled Terminology

• Computational Method

• Comments (for datasets and assigned variables)

• Supporting Documents (Annotated Case Report, SDTM Reviewer’s Guide)

Key changes from the new version include: support for CDISC/NCI controlled terminology, flexible definition of value level metadata, enhanced documentation of data origin, and improved handling of Derivation/Comments.

In pharmaceuticals companies, programmers usually used customized Excel® Workbook from a sample CDISC SDTM Implementation Guide (SDTMIG) Metadata excel file as programming specifications, as well as the source to prepare define-xml file for SDTM datasets. However, the information of controlled terminology was generally generated from the real data, which cannot provide the complete set of allowable values for SDTM variables collected in the study. And the value level metadata was not included in the sample excel file and was often manually prepared. The tedious and error-prone manual work from this process jeopardizes quality of submission, not to mention that resource is wasted from this process.

This paper describes a metadata-driven method to pull metadata automatically from SDTM Specification in a customized Excel Workbook and use the metadata automatically to create Define-XML V2.0 for FDA submission. It also details how to develop SDTM Metadata (programming specification) for automation purpose, illustrates different

sections for SDTM Define-XML V2.0, and provides part of code for how to achieve the automation from Metadata to Define-XML V2.0.

Figure 1 shows the process flow from SDTM metadata to FDA submission package.

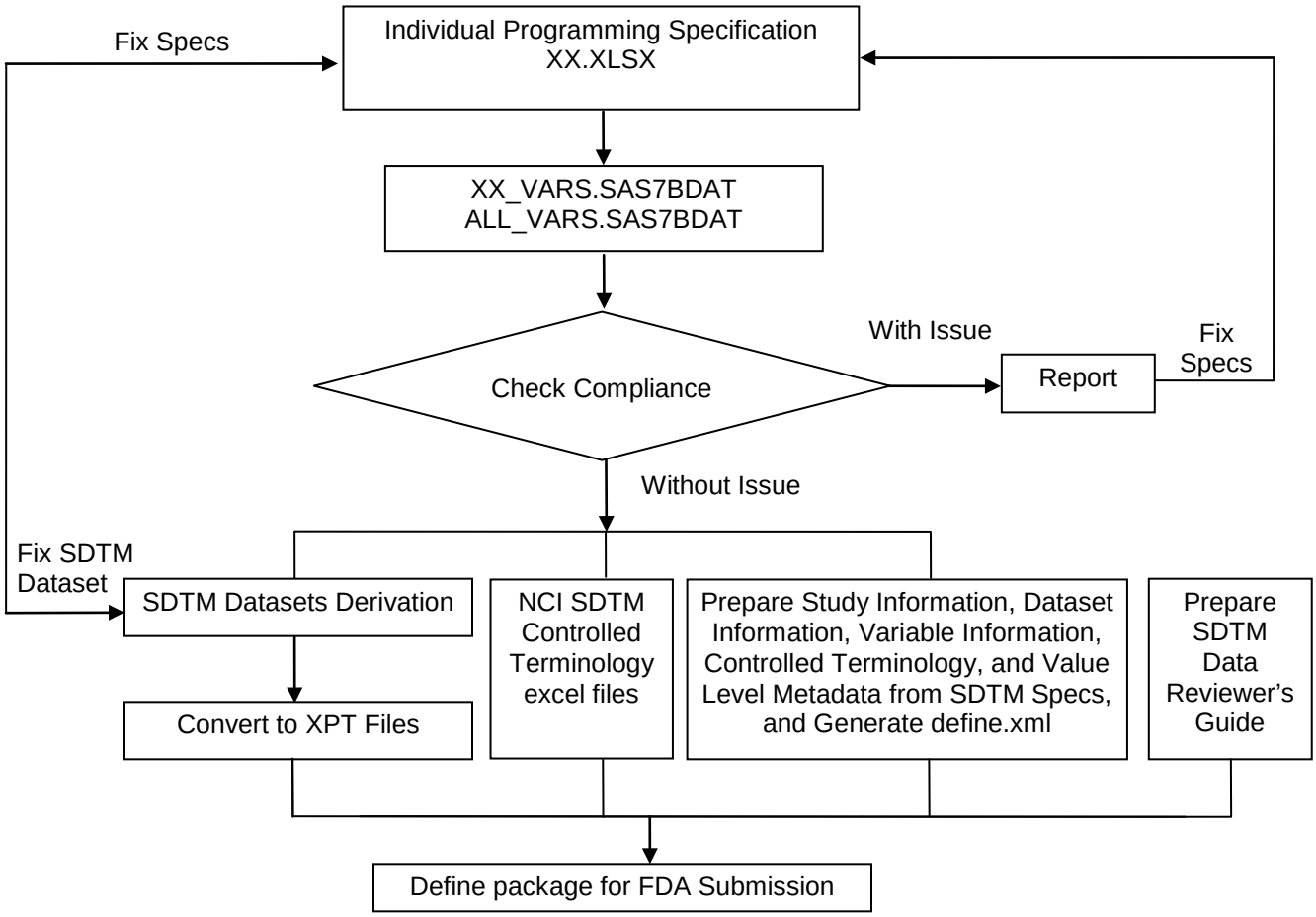


Figure 1. Overview of Process Flow

INTRODUCTION OF DEFINE-XML V2.0

Define-xml provides the specification for the data definitions for datasets. Define-XML V2.0 includes 8 key metadata components: Study and MetaDataVersion Section, links to supporting documents, dataset definitions, dataset variable definitions, controlled terminology definitions, value list definitions, computational method definitions, and comments definitions, to support the FDA submission as shown in Appendix 1:
Compared with Define-XML V1.0, the new features of Define-XML V2.0 include:

(1) Variable length is added

(2) There are two different sections for controlled terminology section. One is for the enumerated items (a list of allowed values), another is for code-decode pairs with 1:1 mapping between code variables and decode variables.

(3) The value level metadata section for XXTESTCD and QVAL is displayed in the controlled terminology section in Define-XML V2.0. However the new value level metadata section is more flexibly defined by where clause shown in Appendix 1 (c), with which the variable value can be defined based on multiple variables with more comparators such as LT, GE, NOTIN, etc.

DESIGN OF SDTM SPECIFICATION FOR AUTOMATICALLY GENERATING DEFINE-XML V2.0

In Define-XML V2.0, components MetaDataVersion Section and links to supporting documents contain the standard information about the metadata version information and standard external documentation such as reviewers' guide. The other 6 components include the detailed metadata information in a specific study. The standard CDISC compliant

SDTM specifications for individual domains were created as templates for the department to provide all the metadata-related information needed in Define-XML V2.0.

The standard SDTM specifications consist of three worksheets: Main Domain Worksheet, Supplemental Domain Worksheet, and Controlled Terminology Worksheet. The Main Domain Worksheet was modified based on CDISC SDTM IG Metadata Excel Workbook sdtmv1.3_sdtmigv3.1.3_metadata.xls with added information on variable length and key variables as shown in Figure 2(a), and provided the information related to Dataset Definitions and Dataset Variable Definitions for both Main Domain Dataset and Supplemental Domain Dataset if any in define-xml. The Supplemental Domain Worksheet provided the information of Value Level Metadata for QVAL for each QNAM in supplemental domains, as shown in Figure 2(b). Controlled Terminology Worksheet was designed for each individual domain for variables with controlled terminologies, as shown in Figure 2(c). These worksheets were well-designed for the automation and generation of Define-XML V2.0. We will introduce the design in detail in the following sections.

For a specific study, programmers only need to select/unselect the study-specific variables in Main Domain Worksheet, select/unselect QNAM in Supplemental Domain Worksheet, and select/unselect the controlled terminology defined in eCRF for an individual domain. Slightly update may be needed for Origin Column corresponding to CRF page number or Comment Column for derivation rules. If supplemental domain is not needed in a specific study, programmers need to unselect all the variables in supplemental domain in Main Domain Worksheet, and unselect all the QNAMs in Supplemental Domain Worksheet.

Variable Selection

Dataset Section

Dataset Variable Section

Mapping Rule, Not Shown in define.xml

Selected	Class	Domain	Variable	Label	Key	Type	Length	Controlled Terminology	Origin	Core	Comment	Form.Item
Y	Events	AE	STUDYID	Study Identifier	1	Char	20		CRF Page 1	Req		TrialNo
D	Events	AE	DOMAIN	Domain Abbreviation		Char	2	DOMAIN	Assigned	Req	AE; **See Reviewer's Guide, Section 2.2 Adverse Event**	Constant: 'AE'
D	Events	AE	USUBJID	Unique Subject Identifier	2	Char	40		Derived	Req	STUDYID+ '-' + SUBJID	TrialNo PatientNo
D	Events	AE	AESEQ	Sequence Number		Num	8		Derived	Req	Sort by STUDYID, USUBJID, AETERM, AESTDTC then assign value. Start at 1 for each subject. No duplicates allowed within a subject within a domain.	Sequence
D	Events	AE	AETERM	Reported Term for the Adverse Event		Char	200		CRF Page 8	Req		AE.AETERM
D	Events	AE	AEDECOD	Dictionary-Derived Term	3	Char	100	MedDRA	CRF Page 9	Req	MedDRA dictionary assigned.	AE.AEDECOD
D	Events	AE	AEBODSYS	Body System or Organ Class		Char	100	MedDRA	CRF Page 9	Exp	MedDRA dictionary assigned.	AE.SOC
Y	Events	AE	AESEV	Severity/Intensity		Char	8	AESEV	CRF Page 8	Perm		AE.AESEV
D	Events	AE	AESER	Serious Event		Char	1	NY	CRF Page 8	Exp		AE.AESER
D	Events	AE	AEACN	Action Taken with Study Treatment		Char	40	ACN	CRF Page 8	Exp		AE.ACN1
			…									
Y	Relationship	SUPPQUAL	STUDYID	Study Identifier	1	Char	20		CRF Page 1	Req		
D	Relationship	SUPPQUAL	RDOMAIN	Related Domain Abbreviation	2	Char	2	DOMAIN	Assigned	Req		
D	Relationship	SUPPQUAL	USUBJID	Unique Subject Identifier	3	Char	40		Derived	Req		
D	Relationship	SUPPQUAL	IDVAR	Identifying Variable	4	Char	8		Assigned	Exp		
D	Relationship	SUPPQUAL	IDVARVAL	Identifying Variable Value	5	Char	8		Assigned	Exp		
D	Relationship	SUPPQUAL	QNAM	Qualifier Variable Name	6	Char	8		CRF Page 8, 9	Req		
D	Relationship	SUPPQUAL	QLABEL	Qualifier Variable Label		Char	40		Assigned	Req		
D	Relationship	SUPPQUAL	QVAL	Data Value		Char	200		CRF Page 8, 9	Req		
			…									

Dataset Section – Documentation

(a) Design of Main Domain Worksheet for Define-XML V2.0

QNAM Selection

Value Level Metadata Section for SUPPXX.QVAL

Mapping Rule, Not Shown in define.xml

Selected	Domain	QNAM	QLABEL	Controlled Terminology	Origin	Comment	Form.Item
Y	SUPPAE	VMEDDRA	MedDRA Version Number		Assigned	MedDRA dictionary assigned.	
Y	SUPPAE	AEDESC1	Subject Description of Adverse Event		CRF Page 4		AE.AEDESC11 AE.AEDESC12
Y	SUPPAE	AEDESCYN	Any comments-Invest. Or Medical review?	NY	CRF Page 4		AE.AEDESCYN
Y	SUPPAE	AEDESC21	Description - 1		CRF Page 4		AE.AEDESC21

(b) Design of Supplemental Domain Worksheet for Define-XML V2.0

Controlled Terminology Selection

Controlled Terminology Section – Enumerated Items

Controlled Terminology Section – Codelist

Dataset Section – Structure

Selected	Variable	Codelist	Order	Controlled Term	TESTCD	TEST
Y	DOMAIN	DOMAIN	1	**One record per adverse event per subject**	AE	Adverse Events
Y	RDOMAIN	DOMAIN	1		AE	Adverse Events
N	AECAT	AECAT	2	ADVERSE EVENTS		
N	AECAT	AECAT	1	BASELINE SIGNS AND SYMPTOMS		
Y	AESEV	AESEV	1	MILD		
Y	AESEV	AESEV	2	MODERATE		
Y	AESEV	AESEV	3	SEVERE		
Y	AEREL	REL	1	DEFINITELY NOT RELATED		
Y	AEREL	REL	2	DEFINITELY RELATED		
Y	AEREL	REL	3	POSSIBLY RELATED		
Y	AEREL	REL	4	PROBABLY NOT RELATED		
Y	AEREL	REL	5	PROBABLY RELATED		
N	AEOUT	OUT	1	FATAL		
Y	AEOUT	OUT	2	NOT RECOVERED/NOT RESOLVED		
Y	AEOUT	OUT	3	RECOVERED/RESOLVED		
Y	AEOUT	OUT	4	RECOVERED/RESOLVED WITH SEQUELAE		
N	AEOUT	OUT	5	RECOVERING/RESOLVING		
N	AEOUT	OUT	6	UNKNOWN		
	…					
N	QVAL.AEDESCYN	NY	4	A		
Y	QVAL.AEDESCYN	NY	1	N		
N	QVAL.AEDESCYN	NY	3	U		
Y	QVAL.AEDESCYN	NY	2	Y		

(c) Design of Controlled Terminology Worksheet for Define-XML V2.0

Figure 2. SDTM Specification Designed for Define-XML V2.0

A SAS macro named **%get_sdtm_specs** reads and reformats all the information from eachindividual specification into 4 SAS datasets **DOMAIN_INFO, VARS_INFO, CTLIST_INFO, and VALUE_INFO** to prepare six major metadata components of Define-XML V2.0.

AUTOMATICALLY GENERATING DEFINE-XML V2.0 FROM SDTM PROGRAMMING SPECIFICATION

The automation of define-xml generation is a process to convert SDTM metadata defined in SAS datasets into metadata in xml format. 1:1 reformatting from SAS datasets to define-xml file can be shown in table 1. We will illustrate the mapping process in detail in the following sections.

Source Worksheet/ (Column) in SDTM Specs	MetaData Section in Define-XML	Define-XML ODM Element	Reference	ODM Element Identifier
Main Domain Worksheet (Domain, Class, Variable)	Dataset Section	ItemGroupDef		
Main Domain Worksheet (Domain)	Dataset File	def:leaf xlink:href	ItemGroupDef	def:ArchiveLocationID
Main Domain Worksheet (Origin)	Origin	def:Origin	ItemDef	
Main Domain Worksheet (Other Columns)	Dataset Variable Section	ItemDef	ItemRef	ItemOID
Main Domain Worksheet, Comment Column: (where clause in Value Level Metadata)	WhereClause in Value List Definitions Section	def:WhereClauseDef	def:WhereClauseRef	WhereClauseOID
Supplemental Domain Worksheet	WhereClause in Value List Definitions Section	def:WhereClauseDef	def:WhereClauseRef	WhereClauseOID

Controlled Terminology Sheet	Controlled Terminology Section	CodeList	CodeListRef	CodeListOID
Main Domain Worksheet, Comment Column for Variable Domain	Comment Section for Dataset	def:CommentDef	ItemDef	def:CommentOID
Main Domain Worksheet (Comment), for derived variables	Computational Method Definitions Section	MethodDef	ItemRef	MethodOID
Main Domain Worksheet (Comment), for assigned/CRF/eDT/Protocol variables	Comments Definitions Section	def:CommentDef	ItemDef	def:CommentOID

Table 1. The format mapping from SDTM variables to Define-XML element

The macro replaces 5 special characters in our specs with 5 pre-defined entity references because they have special meanings in XML.

Special Character	Pre-Defined Entity Reference
<	<
>	>
&	&
‘	'
“	"

DESIGN OF SDTM SPECIFICATIONS FOR DATASET DEFINITION SECTION IN DEFINE-XML V2.0

The Column **DOMAIN, CLASS, KEY,** and the second part of Column **COMMENT** for Variable **DOMAIN** in Main Domain Worksheet and the Column **CONTROLLED_TERM**, **TESTCD**, and **TEST** for Codelist DOMAIN in Controlled Terminology Worksheet provides information for Dataset Section of Define-XML. These columns provide Dataset Information: **DOMAIN, CLASS** and corresponding **CLASSORD, KEYS, DOCUMENTATION, STRUCTURE,** and **DESCRIPTION,** separately, as shown in Display 1.

Selected	Class	Domain	Variable	Key	Comment
Y	Events	AE	STUDYID	1	
D	Events	AE	DOMAIN		AE; **See Reviewer’s Guide, Section 2.2 Adverse Event**
D	Events	AE	USUBJID	2	STUDYID+ '-' + SUBJID

Main Domain Worksheet: AE

Selected	Variable	Codelist	Order	Controlled Term	TESTCD	TEST
Y	DOMAIN	DOMAIN	1	**One record per adverse event per subject**	AE	**Adverse Events**

Controlled Terminology Worksheet: AECT

Tabulation Datasets for Study XXX-zzz (SDTM-IG 3.2)

Dataset	Description	Class	Structure	Purpose	Keys	Location	Documentation
AE	Adverse Events	EVENTS	One record per adverse event per subject	Tabulation	STUDYID, USUBJID, AESTDTC, AETERM	AE.xpt	See Reviewer's Guide, Section 2.2 Adverse Event SDTM Data Reviewer's Guide

Display 1. Mapping of Domain Information from Specifications of individual domains to Define-XML V2.0

A SAS data **DOMAIN_INFO** is saved for Dataset Section.

DOMAIN	DESCRIPTION	CLASSORD	CLASS	STRUCTURE	PURPOSE	KEYS	DOCUMENTATION	REPEATING	ISREFERENCEDATA
AE	Adverse Events	4	EVENTS	One record per adverse event per subject	Tabulation	STUDYID, USUBJID, AESTDTC, AETERM	See Reviewer’s Guide, Section 2.2 Adverse Event	Yes	No

Display 2. SAS data DOMAIN_INFO is saved for Dataset Section in Define-XML V2.0

Dataset Section

An example of ODM source code for Dataset Section in Define-XML V2.0 (shown in Display 1) is shown as below, and dataset metadata is described by an **ItemGroupDef** element in Define-XML.

```
<ItemGroupDef OID="IG.AE"  Domain="AE" Name="AE" SASDatasetName="AE" Repeating="Yes"
IsReferenceData="No" Purpose="Tabulation" def:Structure="One record per adverse event per subject"
def:Class="EVENTS" def:CommentOID="COM.DOMAIN.AE" def:ArchiveLocationID="LF.AE">
```

```
    <Description>
      <TranslatedText xml:lang="en">Adverse Events</TranslatedText>
    </Description>
    ...
    <def:leaf ID="LF.AE" xlink:href="AE.xpt">
      <def:title>AE.xpt</def:title>
    </def:leaf>
</ItemGroupDef>
```

In above source code, the keywords for define-xml ODM elements are shown in bold font, and the contents are shown in plain. **DOMAIN_INFO** provides the contents of define-xml ODM elements. Therefore, the define-xml source code can be automatically generated from DOMAIN_INFO dataset by the following SAS DATA step.

```
data datasets(keep=line order section);
  set DOMAIN_INFO;
  by classord dataset;
  retain order 0;
  section = 40;  *** Dataset Section Number ***;
  *** Dataset Information Table ***;
  line= ' <ItemGroupDef OID="IG.'||strip(dataset)||'"  Domain="'||strip(dataset)||'" Name="'||
        strip(dataset)||'" SASDatasetName="'||strip(dataset)||'" Repeating="'||strip(repeating)||
        '" IsReferenceData="'||strip(isreferencedata)||'"Purpose="'||strip(Purpose)||
        '"def:Structure="'||strip(structure)||'"'||'def:Class="'||strip(class);order+1;output;
  if documentation ne '' then do; line='" def:CommentOID="COM.DOMAIN.'||strip(dataset);
     order+1; output;
  end;
  line= '" def:ArchiveLocationID="LF.'||strip(dataset)||'">';order+1;output;
  line= '        <Description>';order+1;output;
  line= '          <TranslatedText xml:lang="en">'||strip(description)||'</TranslatedText>';
  order+1;output;
  line= '        </Description>';order+1;output;
  *** Generate Comments Section for Dataset ***;
  line= '        <def:leaf ID="LF.'||strip(dataset)||'" xlink:href="'||
        strip(dataset)||'.xpt">';order+1;output;   *** Link to SDTM data ***;
  line= '          <def:title>'||strip(dataset)||'.xpt </def:title>';order+1;output;
  line= '        </def:leaf>';order+1;output;
  line= '      </ItemGroupDef>';order+1;output;
run;
```

If variable LINE generated above is output to an xml file sorting by SECTION and ORDER, Dataset Section is shown in Display 1.

Comments Definitions Section for Dataset

The above SAS code can also automatically generate comments section for dataset as shown below at the same time if variable DOCUMENTATION is not missing.

```
<def:CommentDef OID="COM.DOMAIN.AE">
    <Description>
      <TranslatedText xml:lang>See Reviewer&amp;apos;s Guide, Section 2.2 Adverse Event</TranslatedText>
    </Description>
    <def:DocumentRef leafID="LF.ReviewersGuide">
      <def:PDFPageRef PageRefs="section2.2" Type="NamedDestination"/>
    </def:DocumentRef>
</def:CommentDef>
```

The comment section for dataset in Define-XML V2.0 is shown in Display 3.

Comments

CommentOID	Description
COM.DOMAIN.AE	See Reviewers Guide, Section 2.2 Adverse Event SDTM Data Reviewer's Guide (define.pdf)

Display 3. Comments Definitions Section for Dataset AE in Define-XML V2.0

DESIGN OF SDTM SPECIFICATIONS FOR DATASET VARIABLE SECTION IN DEFINE-XML V2.0

The Column **Variable**, **Label**, **Type**, **Controlled Terminology**, **Origin**, **Core**, and **Comment** in Main Domain Worksheet provide the variable information for sections of Dataset Variable Definitions and Value List Definitions in define-xml. The content of columns except of the part for Value Level Metadata will be displayed in Dataset Variable Section of define-xml. The origin of each variable is categorized as **'CRF', 'PROTOCOL', 'eDT', 'Assigned', and 'Derived'** in define-xml. For derived variables, the derivation rules will also be displayed in computational method

definitions section; for other variables, the comments are also shown in comments definitions section. Table 2 provides information to variable origins, and how they are shown in define-xml.

Origin	Definition	Column 'Comment' in define.xml
CRF	Data Collected from CRF	Comments Section
Protocol	Data Defined by Trial Design Preparation in Protocol	Comments Section
eDT	Data Received via an Electronic Data Transfer	Comments Section
Assigned	Variables which values are determined by an evaluator, exp. code variable for code-decode pairs or coded terms (--DECOD)	Comments Section
Derived	Variables calculated by an algorithm or a rule defined by the sponsor, which is dependent upon other data values.	Computational Method Definitions Section (Computational Algorithms)

Table 2. Where Comment Column Was Shown in SDTM Define-XML V2.0 for Variable with Different Origin

Demographics (DM) [Location: DM.xpt]

Variable	Label	Key	Type	Length	Controlled Terms or Format	Origin	Derivation/Comment
STUDYID	Study Identifier	1	text	20		CRF Page 1	
DOMAIN	Domain Abbreviation		text	2	["DM" = "Demographics"] <Domain Abbreviation (DM)>	Assigned	DM Comments
USUBJID	Unique Subject Identifier	2	text	40		Derived	STUDYID + '-' + SUBJID Computational Method
SUBJID	Subject Identifier for the Study		text	2		CRF Page 2	
RFSTDTC	Subject Reference Start Date/Time		datetime	20	ISO8601	CRF Page 23	First dose date/time for all dosed subjects, otherwise blank. Comments

Link to Controlled Terminology Definitions Section

Map to **Dataset Variable Definitions Section**

Variable	Label	Key	Type	Length	Controlled Terminology	Origin	Core	Comment
STUDYID	Study Identifier	1	Char	20		CRF Page 1	Req	Map to Comments
DOMAIN	Domain Abbreviation		Char	2	DOMAIN	Assigned	Req	DM Definitions Section for Dataset Variables
USUBJID	Unique Subject Identifier	2	Char	40		Derived	Req	TRNO+ '-' + PatientNo Map to Computational Method Definitions Section
SUBJID	Subject Identifier for the Study		Char	20		CRF Page 2	Req	
RFSTDTC	Subject Reference Start Date/Time		Char	20	ISO 8601	CRF Pages 23	Exp	First dose date/time for all dosed subjects, otherwise blank.

Map to Comments Definitions Section for Dataset Variables

Display 4. Mapping of Variable Information from DM Specification to Define-XML V2.0

Variable information will be stored in SAS data VARS_INFO for preparing Dataset Variables Section, as shown in Display 5.

DOMAIN	VARSEQ	VARIABLE	LABEL	KEY	DATA TYPE	LENGTH	SIGNIFICANT DIGITS	DISPLAY FORMAT	ORIGIN	CODELIST	CORE	COMMENT	MANDATORY
DM	1	STUDYID	Study Identifier	1	text	20	.		CRF Page 1		Req		Yes
DM	2	DOMAIN	Domain Abbreviation	.	text	2	.		Assigned	DOMAIN	Req	DM	Yes
DM	3	USUBJID	Unique Subject Identifier	2	text	40	.		Derived		Req	STUDYID+ '-' + SUBJID	Yes
DM	4	SUBJID	Subject Identifier for the Study	.	text	20	.		CRF Page 2		Req		Yes
DM	5	RFSTDTC	Subject Reference Start Date/Time	.	integer	20	.	datetime	CRF Pages 23	ISO 8601	Exp	First dose date/time for all dosed subjects, otherwise blank.	No

Display 5. SAS data VARS_INFO, preparing for Dataset Variables Section in Define-XML V2.0

Dataset Variables Section

Each variable is defined by an **ItemDef** element outside of **ItemGroupDef** element, which will be referenced by an **ItemRef** element within an **ItemGroupDef** element with same ODM element identifier (**OID**).

```
<ItemGroupDef OID="IG.DM"
…
  <ItemRef ItemOID="IT.DM.DOMAIN" OrderNumber="2" Mandatory="Yes"/>
  <ItemRef ItemOID="IT.DM.USUBJID" OrderNumber="3" Mandatory="Yes" KeySequence="2"
MethodOID="MT.DM.USUBJID"/>
…
  <ItemRef ItemOID="IT.DM.RFSTDTC" OrderNumber="5" Mandatory="No"/>
…
</ItemGroupDef>
…
<ItemDef OID="IT.DM.DOMAIN" Name="DOMAIN" SASFieldName="DOMAIN" DataType="text" Length="2"
def:CommentOID="COM.DM.DOMAIN">
  <Description>
    <TranslatedText xml:lang="en">Domain Abbreviation</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.DM.DOMAIN"/>
  <def:origin Type="Assigned"/>
</ItemDef>
<ItemDef OID="IT.DM.USUBJID" Name="USUBJID" SASFieldName="USUBJID" DataType="text" Length="40">
  <Description>
    <TranslatedText xml:lang="en">Unique Subject Identifier</TranslatedText>
  </Description>
  <def:origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.DM.RFSTDTC" Name="RFSTDTC" SASFieldName="RFSTDTC" DataType="datetime" Length="20"
def:CommentOID="COM.DM.RFSTDTC">
  <Description>
    <TranslatedText xml:lang="en">Subject Reference Start Date/Time</TranslatedText>
  </Description>
  <def:origin Type="CRF">
    <def:DocumentRef leafID="LF.blankcrf">
      <def:PDFPageRef PageRefs="23" Type="PhysicalRef"/>
    </def:DocumentRef>
  </def:Origin>
</ItemDef>
```

The ODM source code can be automatically generated by SAS with VARS_INFO dataset.

```
data datasets(keep=line order section);
  set vars_info;
  by dataset varseq;
  retain dsorder order 0;
  if first.dataset then dsorder = 0;
  *** Variable Information Table ***;
  dsorder = dsorder+1;
  section=40;  *** Dataset Section Number, Item Reference ***;
  line = '  <ItemRef ItemOID="IT.'||strip(dataset) ||'.'||strip(variable)||
         '" OrderNumber="'||strip(put(dsorder, best.))||'" Mandatory="'||strip(mandatory)||'"';
  if key > 0 then line = trim(line)||' KeySequence="'||strip(put(key,best.))||'"';
  if propcase(origin) = 'Derived' then line = trim(line)||' MethodOID="MT.'|| strip(dataset) ||
                                              '.'||strip(variable)||'"';
  line = trim(line) || '/>';
  order+1;
  output;

  section=50; ** Item definition: define variables by OID, Dataset Variable Section Number ***;
  line= '<ItemDef OID="IT.'||strip(dataset) ||'.'||strip(variable)||'" Name="'||strip(variable)||
        '" SASFieldName="'||strip(variable)||'" DataType="'||strip(datatype);
  if not missing(SignificantDigits) then line= trim(line)||'" SignificantDigits="'||
                                               strip(put(significantdigits,best.));
  if not missing(DisplayFormat) then line= trim(line)||'" def:DisplayFormat="'||
                                          strip(DisplayFormat);
  line= trim(line)||'" Length="'||strip(put(length, best.));
  if codelist ne ' ' then do;
     if substr(reverse(strip(codelist)),1,1)='.' then line = trim(line)||'" def:DisplayFormat="'
```

```
                                                              ||strip(codelist);
  end;
  if propcase(origin) not in ('Derived') and not missing(comment) then line=trim(line)||
                        '" def:CommentOID="COM.'||strip(dataset)||'.'||strip(variable);
  line = trim(line)||'">'; order+1;output;
  line= '     <Description>';order+1;output;
  line= '          <TranslatedText xml:lang="en">'||strip(label)||'</TranslatedText>';
  order+1;output;
  line= '     </Description>';order+1;output;

  if codelist not in (' ','ISO8601', 'ISO 8601')and substr(reverse(strip(codelist)),1,1) ne '.'
then do;
     if not missing(coding) then do;
        line='  <CodeListRef CodeListOID="CL.'||strip(coding)||'.'||strip(codelist)||'"/>';
        order+1;output;
     end;
     else if not missing(sourcedataset) and not missing(sourcevariable) then do;
        line='  <CodeListRef CodeListOID="CL.'||strip(sourcedataset)||'.'||strip(sourcevariable)
             ||'.'||strip(codelist)||'"/>';order+1;output;
     end;
     else if not missing(sourcedataset) then do;
        line='  <CodeListRef CodeListOID="CL.'||strip(sourcedataset)||'.'||strip(codelist)||
             '"/>';order+1;output;
     end;
     else if not missing(sourcevariable) then do;
        line='  <CodeListRef CodeListOID="CL.'||strip(sourcevariable)||'.'||strip(codelist)||
             '"/>';order+1;output;
     end;
     else do;
        line='  <CodeListRef CodeListOID="CL.'||strip(codelist)||'"/>';order+1;output;
     end;
  end;
  if valuelist ne ' ' then do;
     line = '  <def:ValueListRef ValueListOID="VL.'||strip(dataset)||'.'||strip(variable)||'"/>';
     order+1;output;
  end;

  line= '  <def:Origin Type="'||strip(origin)||'"/>';order+1;output;

  if upcase(origin)=:'CRF' and crfnum ne '' then do;
     line= '  <def:Origin Type="CRF">';order+1;output;
     line= '    <def:DocumentRef leafID="LF.blankcrf">';order+1;output;
     line= '      <def:PDFPageRef PageRefs="'||strip(crfnum)||'" Type="PhysicalRef"/>';
     order+1;output;
     line= '    </def:DocumentRef>';order+1;output;
     line= '  </def:Origin>';order+1;output;
  end;
  if upcase(origin) =: 'CRF' and crffnum ne '' and crflnum ne '' then do;
     line= '  <def:Origin Type="CRF">';order+1;output;
     line= '    <def:DocumentRef leafID="LF.blankcrf">';order+1;output;
     line= '      <def:PDFPageRef FirstPage="'||strip(crffnum)||'" LastPage="'||strip(crflnum)||
           '" Type="PhysicalRef"/>';order+1;output;
     line= '    </def:DocumentRef>';order+1;output;
     line= '  </def:Origin>';order+1;output;
  end;
  else do;
     line= '  <def:Origin Type="'||strip(origin)||'"/>';order+1;output;
  end;
  line= '</ItemDef>';order+1;output;
run;
```

If variable LINE generated above is output to an xml file sorting by SECTION and ORDER, Dataset Variables Section will be shown in Display 4.

Computational Method Definitions Section for Variables

For derived variables, the algorithms for derivation are described by ODM **MethodDef** element as below. The algorithm comes from SAS data VARS_INFO.

```
<MethodDef OID="MT.DM.USUBJID" Name="CM.DM.USUBJID" Type="Computation">
  <Description>
    <TranslatedText xml:lang="en">STUDYID + &apos;-&apos; + SUBJID</TranslatedText>
  </Description>
</MethodDef>
```

SAS code which automatically generate computational Method Definitions Section for Variables is shown below.

```
data datasets(keep=line order section);
  set vars_info;
  if origin = 'Derived' and not missing(comment) then do;
    section = 70;         *** Computational Method for Variable Section Number ***;
    line= '<MethodDef OID="MT.'||strip(dataset)||'.'||strip(variable)||'" Name="CM.'||
          strip(dataset)||'.'||strip(variable)||'" Type="Computation">';order+1;output;
    line= '  <Description>';order+1;output;
    line= '    <TranslatedText xml:lang="en">'||strip(comment)||'</TranslatedText>';
    order+1;output;
    line= '  </Description>';order+1;output;
    line= '</MethodDef>';order+1;output;
    section = 50;
  end;
run;
```

An example of Computational Method Definitions Section is shown in Display 6. In **ItemRef** element the algorithm has been referenced to the variable DM.USUBJID defined by **MethodOID** attribute, the derivation rules will also be shown in Derivation/Comment Column of Dataset Variables Section in Display 4.

Computational Algorithms

Method	Type	Description
CM.DM.USUBJID	Computation	STUDYID + '-' + SUBJID

Display 6. Computational Method Definitions Section for Dataset Variable DM.USUBJID in Define-XML V2.0

Comments Definitions Section for Variables

If the origin of the variables is 'Assigned', 'CRF', 'Protocol', or 'eDT', the comments at variable level are described by **def:CommentDef** element as below.

```
<def:CommentDef OID="COM.DM.RFSTDTC">
  <Description>
    <TranslatedText>First dose date/time for all dosed subjects, otherwise blank.</TranslatedText>
  </Description>
</def:CommentDef>
```

SAS code to automatically generate Comment Definitions Section for Variables is shown below.

```
data datasets(keep=line order section);
  set vars_info;
  if propcase(origin) ne 'Derived' and not missing(comment) then do;
     section = 82;       *** Comments for Dataset Variables Section Number ***;
     line= '<def:CommentDef OID="COM.'||strip(dataset)||'.'||strip(name)||'">';order+1;output;
     line= '  <Description>';order+1;output;
     line= '     <TranslatedText>'||strip(comment)||'</TranslatedText>';order+1;output;
     line= '  </Description>';order+1;output;
     line= '</def:CommentDef>';order+1;output;
     section = 50;
  end;
run;
```

An example of Comment Definitions Section is shown in Display 7. In **ItemRef** element the comment has been referenced to the variable DM.RFSTDTC defined by **def:CommentOID** attribute, it will also be shown in Derivation/Comment Column of Dataset Variables Section in Display 4.

Comments

CommentOID	Description
COM.DM.RFSTDTC	First dose date/time for all dosed subjects, otherwise blank.

Display 7. Comments Definitions Section for Dataset Variable DM.RFSTDTC in Define-XML V2.0

DESIGN OF SDTM SPECIFICATIONS FOR CONTROLLED TERMINOLOGY SECTION IN DEFINE-XML V2.0

Controlled Terminology was defined as the finite set of allowable values used in a clinical trial across all variables. "A 'Codelist' is a unique subset of the controlled terminology to which one or more variables are subject [2]". In SDTM Implementation Guide, standard variables, such as AEACN, UNIT, and etc., are required to assign code values in NCI/CDISC codelist, and some other variables, such as are ARMCD, LBCAT, are subject to sponsor defined controlled terminology. All controlled terminologies used in a study must be provided within the Define-XML document using a **CodeList** element.

In SDTM datasets, some different variables may share the same codelist name. For example, Codelist UNIT is used for both LBSTRESU and EGSTRESU. In a clinical trial, usually there are hundreds of lab tests collected in a study, and therefore hundreds of code values were selected as codelist UNIT for variable LBSTRESU. It will be very hard for programmers and reviewers to tell the allowable values for EGSTRESU apart from those for LBSTRESU. In order to make it clear which allowable values belong to a specific variable, a controlled terminology worksheet was created for each individual domain as shown in Figure 2 (c) for each variable specified with a codelist name. The same codelist name was assigned to that variable at Column 'Controlled Terminology' in Main Domain Worksheet. It is more convenient to select the code values in a subset of controlled terminology for each variable in an individual domain than select in whole NCI/CDISC controlled terminology spreadsheet. Another benefit is that a customized controlled terminology worksheet can easily incorporate sponsor-defined controlled terminology other than NCI/CDISC codelist. The information in Controlled Terminology Worksheet will be shown in Controlled Terminology Section of Define-XML.

Classification of Controlled Terminology

There are three types of controlled terminology in format.

1. **Codelist for Code-Decode Variables**

   Generally, SDTM finding domains have 1:1 mapping of a pair of variables –TESTCD and –TEST, where the former variable stores code of the test and the latter stores the description of the test. Both variables are subject to controlled terminology and are not free text.

   For this kind of controlled terminology, the controlled terminology worksheet will define codelist name –TEST for Variable –TESTCD with 1:1 mapping between Variables –TESTCD and –TEST. The controlled terms were filled in Columns 'TESTCD' and 'TEST' in controlled terminology worksheet separately for Variable –TESTCD and –TEST. The column 'Controlled Term' will be blank. The programmer will select/unselect the tests collected in the study as shown in Display 8 (a). Display 8 (b) shows how codelist is displayed in the Controlled Terminology Section of Define-XML for VSTESTCD-VSTEST. The codelists are shown for both variable VSTESTCD with CodeList name VSTESTCD and variable VSTEST with CodeList VSTEST, in which CodeList VSTEST uses EnumeratedItems to store the list of possible values, and VSTESTCD CodeList uses CodeListItems to reflect the association with VSTEST items.

Selected	Variable	Codelist	Order	Controlled Term	TESTCD	TEST
Y	VSTESTCD	VSTEST	1		VSALL	VS Data
Y	VSTESTCD	VSTEST	2		WEIGHT	Weight
Y	VSTESTCD	VSTEST	3		HEIGHT	Height
Y	VSTESTCD	VSTEST	4		HR	Heart Rate
N	VSTESTCD	VSTEST	5		PULSE	Pulse Rate
Y	VSTESTCD	VSTEST	6		SYSBP	Systolic Blood Pressure

(a) An Example of Codelist for Controlled Terminology of VSTESTCD-VSTEST Pair in SDTM.VS

VSTEST [CL.VSTEST] Codelist Name for VSTEST

Permitted Value (Code)
Weight [*C25208*]
Height [*C25347*]
Heart Rate [*C49677*]
Systolic Blood Pressure [*C25298*]
Diastolic Blood Pressure [*C25299*]
Respiratory Rate [*C49678*]
Temperature [*C25206*]
Body Mass Index [*C16358*]
VS Data

VSTESTCD [CL.VSTESTCD] Codelist Name for VSTESTCD

Permitted Value (Code)	Display Value (Decode)
VSALL	VS Data
WEIGHT [*C25208*]	Weight
HEIGHT [*C25347*]	Height
HR [*C49677*]	Heart Rate
SYSBP [*C25298*]	Systolic Blood Pressure
DIABP [*C25299*]	Diastolic Blood Pressure
RESP [*C49678*]	Respiratory Rate
TEMP [*C25206*]	Temperature
BMI [*C16358*]	Body Mass Index

(b) define-xml for controlled terminology of VSTESTCD-VSTEST with 1:1 mapping

Display 8. An Example of Codelist for Code-Decode Variables – From SDTM specs to Define-XML V2.0

Other examples of codelist include a pair of variables ARMCD-ARM and a pair of variable TSPARMCD-TSPARM.

2. **Enumerated Codelist**

The Enumerated Codelist defines Code Values in a Controlled Terminology. An example of enumerated codelist is Codelist ACN for variable AEACN. For this kind of controlled terminology, the controlled terminology worksheet will define codelist name ACN for Variable AEACN at Column 'Controlled Term', and the Columns 'TESTCD' and 'TEST' will be blank. The programmer will select/unselect the code values collected in the study as shown in Display 9 (a). Display 9 (b) shows how codelist is displayed in the Controlled Terminology Section of Define-XML for AEACN. In define-xml, all possible values of the variable AEACN in AE are listed as **Enumerated items**.

Selected	Variable	Codelist	Order	Controlled Term	TESTCD	TEST
N	AEACN	ACN	1	DOSE INCREASED		
Y	AEACN	ACN	2	DOSE NOT CHANGED		
N	AEACN	ACN	3	DOSE REDUCED		
Y	AEACN	ACN	4	DRUG INTERRUPTED		
Y	AEACN	ACN	5	DRUG WITHDRAWN		
Y	AEACN	ACN	6	NOT APPLICABLE		

(a) An Example of Enumerated Codelist ACN for variable AEACN in SDTM.AE

Action Taken with Study Treatment [CL.ACN]

Permitted Value (Code)
DOSE NOT CHANGED [*C49504*]
DRUG INTERRUPTED [*C49501*]
DRUG WITHDRAWN [*C49502*]
NOT APPLICABLE [*C48660*]

(b) define-xml for controlled terminology of Enumerated Codelist ACN

Display 9. An Example of Enumerated Codelist for AEACN – From SDTM specs to Define-XML V2.0

A special example of enumerated codelist is codelist with description of the code values, such as AESEV. The description is provided at Column '**CDISC Synonym(s)**' in NCI/CDISC controlled terminology spreadsheet as shown in Display 10 (a). The programmer will select/unselect the code values collected in the study in Controlled Terminology Worksheet as shown in Display 10 (b). The codelist in define.xml can be seen from Display 10 (c).

Code	Codelist Code	Codelist Extensible (Yes/No)	Codelist Name	CDISC Submission Value	CDISC Synonym(s)
C66769		No	Severity/Intensity Scale for Adverse Events	AESEV	Severity/Intensity Scale for Adverse Events
C41338	C66769		Severity/Intensity Scale for Adverse Events	MILD	1; Grade 1
C41339	C66769		Severity/Intensity Scale for Adverse Events	MODERATE	2; Grade 2
C41340	C66769		Severity/Intensity Scale for Adverse Events	SEVERE	3; Grade 3

(a) An Example of Enumerated Codelist AESEV in NCI/CDISC Controlled Terminology Spreadsheet

Selected	Variable	Codelist	Order	Controlled Term	TESTCD	TEST
N	AESEV	AESEV	1	MILD		
Y	AESEV	AESEV	2	MODERATE		
N	AESEV	AESEV	3	SEVERE		

(b) An Example of Enumerated Codelist AESEV in Controlled Terminology Worksheet for Variable AESEV

Severity/Intensity Scale for Adverse Events [CL.AESEV]

Permitted Value (Code)	Display Value (Decode)
MILD [*C41338*]	1; Grade 1
MODERATE [*C41339*]	2; Grade 2
SEVERE [*C41340*]	3; Grade 3

(c) define-xml for controlled terminology of Enumerated Codelist AESEV

Display 10. An Example of Enumerated Codelist for AESEV – From SDTM specs to Define-XML V2.0

3. External Codelist: MedDRA and WHODDE

The sponsor is expected to provide a subsection for external code list references in define-xml, like dictionary name and version, to be used to map the terms. The MedDRA or WHO Drug dictionaries name will be provided as Global Macro Variables &MedDRA and &WHODRUG in programming setup. In Main Domain Worksheet, if any variable with

Controlled Terminology MedDRA or WHO Drug is selected, the external published source, MedDRA dictionaries, will be shown in define-xml.

Selected	Domain	Variable	Label	Key	Type	Length	Controlled Terminology
N	AE	AEMODIFY	Modified Reported Term		Char	200	MedDRA
D	AE	AELLT	Lowest Level Term		Char	100	MedDRA
D	AE	AELLTCD	Lowest Level Term Code		Num		
D	AE	AEDECOD	Dictionary-Derived Term		Char	100	MedDRA

(a) An Example of External Codelist: MedDRA in Main Domain Worksheet

External Dictionaries

Reference Name	External Dictionary	Dictionary Version
Adverse Event Dictionary (CL.MedDRA)	MedDRA	18.0

(b) An Example of External Codelist: MedDRA in define-xml

Display 11. An Example of External CodeList, MedDRA Dictionary – from SDTM Specs to Define-XML V2.0

Controlled Terminology Definitions will be stored in SAS data CTLIST_INFO for preparing Controlled Terminology Definitions Section, as shown in Display 12. In this data, Variable CODE stores the NCI/CDISC code for NCI/CDISC controlled terms; and Variable External flags codelist for 'External Dictionary' as 'Y'.

SOURCE DATASET	SOURCE VARIABLE	CODELIST	CTORDER	CODEVAL	DECODEVAL	WARNING	DATATYPE	CODE	External
		MedDRA		MedDRA	18.0		text		Y
		ACN	1	DOSE NOT CHANGED			text	C49504	
		ACN	2	DRUG INTERRUPTED			text	C49501	
		ACN	3	DRUG WITHDRAWN			text	C49502	
		ACN	4	NOT APPLICABLE			text	C48660	
AE		DOMAIN	1	AE	Adverse Events		text	C49562	
		VSTEST	1	VS Data			text		
		VSTEST	2	Weight			text	C25208	
		VSTEST	3	Height			text	C25347	
		VSTEST	4	Heart Rate			text	C49677	
		VSTEST	5	Systolic Blood Pressure			text	C25298	
		VSTESTCD	1	VSALL	VS Data		text	C25208	
		VSTESTCD	2	WEIGHT	Weight		text	C25347	
		VSTESTCD	3	HEIGHT	Height		text	C49677	
		VSTESTCD	4	HR	Heart Rate		text	C25298	
		VSTESTCD	5	SYSBP	Systolic Blood Pressure		text	C25299	

Display 12. SAS data CTLIST_INFO, Preparing for Controlled Terminology Section in Define-XML V2.0

Controlled Terminology Definitions Section

Each code list is defined by a **CodeList** element outside of **ItemDef** element, which will be referenced by **CodeListRef** element with same ODM element identifier (**OID**). If the codelist names or controlled terms were in NCI/CDISC controlled terminology, Alias Name will be referenced for NCI/CDISC code.

```
<CodeList OID="CL.VSTEST" Name="Vital Signs Test Name" DataType="text">
  <EnumeratedItem CodedValue=" VS Data" OrderNumber="1" def:ExtendedValue="Yes"/>
  <EnumeratedItem CodedValue="Weight" OrderNumber="2">
    <Alias Name="C25208" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="Height" OrderNumber="3">
    <Alias Name="C25347" Context="nci:ExtCodeID"/>
</EnumeratedItem>
  <EnumeratedItem CodedValue="Heart Rate" OrderNumber="4">
    <Alias Name="C49677" Context="nci:ExtCodeID"/>
</EnumeratedItem>
  <EnumeratedItem CodedValue="Systolic Blood Pressure" OrderNumber="5">
    <Alias Name="C25298" Context="nci:ExtCodeID"/>
</EnumeratedItem>
  <EnumeratedItem CodedValue="Diastolic Blood Pressure" OrderNumber="6">
    <Alias Name="C25299" Context="nci:ExtCodeID"/>
</EnumeratedItem>
  …
<Alias Name="C67153" Context="nci:ExtCodeID"/>
</CodeList>
<CodeList OID="CL.VSTESTCD" Name="Vital Signs Test Code" DataType="text">
  <CodeListItem CodedValue="VSALL" OrderNumber="1" def:ExtendedValue="Yes">
```

```
    <Decode>
      <TranslatedText xml:lang="en">VS Data</TranslatedText>
    </Decode>
  </CodeListItem>
  <CodeListItem CodedValue="WEIGHT" OrderNumber="2">
    <Decode>
      <TranslatedText xml:lang="en">Weight</TranslatedText>
    </Decode>
    <Alias Name="C49504" Context="nci:ExtCodeID"/>
  </CodeListItem>
  <CodeListItem CodedValue="HEIGHT" OrderNumber="3">
    <Decode>
      <TranslatedText xml:lang="en">Height</TranslatedText>
    </Decode>
    <Alias Name="C49504" Context="nci:ExtCodeID"/>
  </CodeListItem>
  <CodeListItem CodedValue="HR" OrderNumber="4">
    <Decode>
      <TranslatedText xml:lang="en">Heart Rate</TranslatedText>
    </Decode>
    <Alias Name="C49504" Context="nci:ExtCodeID"/>
  </CodeListItem>
  <CodeListItem CodedValue="SYSBP" OrderNumber="5">
    <Decode>
      <TranslatedText xml:lang="en">Systolic Blood Pressure</TranslatedText>
    </Decode>
    <Alias Name="C49504" Context="nci:ExtCodeID"/>
  </CodeListItem>
  <CodeListItem CodedValue="DIABP" OrderNumber="6">
    <Decode>
      <TranslatedText xml:lang="en">Diatolic Blood Pressure</TranslatedText>
    </Decode>
    <Alias Name="C49504" Context="nci:ExtCodeID"/>
  </CodeListItem>
<Alias Name="C66741" Context="nci:ExtCodeID"/>
</CodeList>
<CodeList OID="CL.ACN" Name="ACN" DataType="text">
  <EnumeratedItem CodedValue="DOSE NOT CHANGED" OrderNumber="1">
    <Alias Name="C49504" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="DRUG WITHDRAWN" OrderNumber="2">
    <Alias Name="C49502" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="NOT APPLICABLE" OrderNumber="3">
    <Alias Name="C48660" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
<Alias Name="C66767" Context="nci:ExtCodeID"/>
</CodeList>
<CodeList OID="CL.MedDRA" Name="MedDRA" DataType="text">
  <ExternalCodeList Dictionary="MedDRA" Version="18.0"/>
</CodeList>
```

The ODM source code can be automatically generated by SAS with CTLIST_INFO dataset.

```
data ctlist;
  set ctlist_info;
  by sourcedataset codelist ctorder;
  length line $4000 code $8;
  retain order 0;
  section = 60; *** Section Number for Controlled Terminology except External Codelist ***;
  if first.codelist and sourcedataset = ' ' then do;
    if index(upcase(codelist),"MEDDRA")>0 or index(upcase(codelist),"WHODD")>0  then do;
      section = 65;  *** Section Number for External Codelist ***;
      line= '<CodeList OID="CL.'||strip(codelist)||'" Name="'||strip(codelist_name)||
            '" DataType="'||strip(datatype)||'">'; order+1;output;
      section = 60;
    end;
    else if codelist ne 'ISO8601' and not missing(codelist_name) then do;
```

```
      line= '<CodeList OID="CL.'||strip(codelist)||'" Name="'||strip(codelist_name)||
            '" DataType="'||strip(datatype)||'">'; order+1;output;
    end;
    else if codelist ne 'ISO8601' then do;
      line= '<CodeList OID="CL.'||strip(codelist)||'" Name="'||strip(codelist)||
            '" DataType="'||strip(datatype)||'">'; order+1;output;
    end;
  end;
  if index(upcase(codelist),"MEDDRA")>0 or index(upcase(codelist),"WHODD")>0 then do;
    section = 65;
    line= '  <ExternalCodeList Dictionary="'||strip(codeval)||'" Version="'||
          strip(decodeval)||'"/>'; order+1;output;
    section = 60;
  end;
  else if missing(decodeval) and missing(code) then do;
    line= '  <EnumeratedItem CodedValue="'||strip(codeval)||'" OrderNumber="'||
          strip(put(ctorder,best.))||'" def:ExtendedValue="Yes"/>';order+1;output;
  end;
  else if missing(decodeval) and code not in ('') then do;
    line= '  <EnumeratedItem CodedValue="'||strip(codeval)||'" OrderNumber="'||
          strip(put(ctorder,best.))||'">';order+1;output;
    line= '    <Alias Name="'||strip(code)||'" Context="nci:ExtCodeID"/>';order+1;output;
    line= '  </EnumeratedItem>';order+1;output;
  end;
  else do;
    line= '  <CodeListItem CodedValue="'||strip(codeval)||'" OrderNumber="'||
          strip(put(ctorder,best.))||'">';
    if code in ('') then line = trim(line)||'" def:ExtendedValue="Yes';
    line = trim(line) || '">';order+1;output;
    line= '    <Decode>';order+1;output;
    line= '      <TranslatedText xml:lang="en">'||strip(decodeval)||'</TranslatedText>';
    order+1;output;
    line= '    </Decode>';order+1;output;
    if not missing(code) then do;
      line= '      <Alias Name="'||strip(code)||'" Context="nci:ExtCodeID"/>';order+1;output;
    end;
    line= '  </CodeListItem>';order+1;output;
  end;

  if last.codelist then do;
    if index(upcase(codelist),"MEDDRA")>0 or index(upcase(codelist),"WHODD")>0 then section=65;
    else if not missing(codelist_code) then do;
       line= '  <Alias Name="'||strip(codelist_code)||'" Context="nci:ExtCodeID"/>';
       order+1; output;
    end;
    line='</CodeList>';order+1;output;
    section = 60;
  end;
run;
```

If variable LINE generated above is output to an xml file sorting by SECTION and ORDER, the Controlled Terminology Definitions Section will be shown in Display 8 – 11.

DESIGN OF SDTM SPECIFICATIONS FOR VALUE LEVEL METADATA IN DEFINE-XML V2.0

Value Level Metadata defines metadata for a dataset variable under a specific condition, and it provides detailed information that cannot be provided by dataset variable metadata if metadata attributes, such as **DataType**, **Length**, or **Derivation/Comment**, are different for subsets of variable values. Therefore in order to facilitate data review it is desirable to provide Value Level Metadata for better interpreting the study data. Value Level Metadata is typically used in SDTM findings domains when definition of the variables **--ORRES/--STRESC/ –STRESN/--ORRESU/--STRESU** are specific to corresponding test code (--TESTCD). Another case to use Value Level Metadata is defining the value list for variable QVAL in SDTM supplemental domains.

Value Level Metadata for Variables in Main Domain: –ORRES/--STRESC/--STRESN/--ORRESU/--STRESU

In SDTM findings domains, variables --ORRES/--STRESC/--STRESN/--ORRESU/--STRESU often contain different metadata attributes or mapping rules for different values of the test code. Value Level Metadata can describe those variables based on all of the possible subsets of the test code.

Take VSSTRESN as an example, derivation/mapping rules and data types of VSSTRESN are different when VSTESTCD=“BMI” from that when LBTESTCD=“DIABP”. Another example is different Display Formats of VSSTRESN when LBTESTCD=“BMI” or LBTESTCD=“WEIGHT”. Display 13 (a) shows the SDTM specification of VSSTRESN for VS. The key word: Value Level Metadata in column Comment divides the cell into two parts. The first

part will be presented in VSSTRESN Variable Section in Define-XML V2.0, shown in Display 13 (b). The second part will be displayed in Value Level Metadata Section in Define-XML V2.0, shown in Display 13 (c).

The way to write the second part for Value Level Metadata Section in Define-XML is explained below.

Write "Value Level Metadata:" at the beginning, followed by each individual values of VSTESTCD and corresponding VSTEST preceded by an ordering number '(#)', which specifies the ordering in Value Level Metadata for VSSTRESN in Define-XML.

Domain	Variable	Label	Type	Length	Controlled Terminology	Origin	Core	Comment
VS	VSSTRESN	Numeric Result/Finding in Standard Units	Num	8	8.1	Derived	Exp	Numeric value of VSORRES **Value Level Metadata:** (1) BMI=Body Mass Index: float 4.1CTDerived$populated for every visit where WEIGHT is collected (2) DIABP= Diastolic Blood Pressure: integerCTCRF Page 40 (3) HEIGHT = Height: float 5.1CTCRF Page 40 (4) HR = Heart Rate: integerCTCRF Page 40 (5) RESP = Respiratory Rate: integerCTCRF Page 40 (6) SYSBP = Systolic Blood Pressure: integerCTCRF Page 40 (7) TEMP = Temperature: float 4.1CTCRF Page 40 (8) WEIGHT = Weight: float 5.1CTCRF Page 40

Value List Definitions Section

(a) An example of VSSTRESN in VS specs designed for Value Level Metadata Define-XML V2.0

Vital Signs (VS) [Location: VS.xpt]

Variable	Label	Key	Type	Length	Controlled Terms or Format	Origin	Derivation/Comment
VSSTRESN	Numeric Result/Finding in Standard Units		float	8		Derived	Numeric value of VSORRES

(b) Dataset Variable Section for VS.VSSTRESN in Define-XML V2.0 (First Part of Column Comment).

Value Level Metadata - VS [VSSTRESN]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
VSSTRESN	VSTESTCD EQ BMI (Body Mass Index)	float	4.1		Derived	populated for every visit where WEIGHT is collected
VSSTRESN	VSTESTCD EQ DIABP (Diastolic Blood Pressure)	integer	8		CRF Page 40	

(c) Value List for VS.VSSTRESN in Define-XML V2.0 (Second Part of Column Comment).

Display 13. Value List Definitions Section in SDTM (VSSTRESN) Specifications Designed for Define-XML V2.0

Table 3 illustrates the syntax for the Value Level Metadata for VSSTRESN by an example of "BMI=Body Mass Index: float 4.1CTDerived$populated for every visit where WEIGHT is collected;"

Condition Used to Retrieved Words	Retrieved Words	The Syntax Created by SAS Macro Tool	Column Displayed in DEFINE.XML
:	BMI=BODY MASS INDEX (kg/m2)	LBTESTCD EQ BMI (BODY MASS INDEX (kg/m2))	Where
first $, first word	float	Float	Type
first $, second word	4.1	4.1	Length/Display Format

Second $	CT		Controlled Terms or Format
Third $	Derived	Derived	Origin
Words after last $	populated for every visit where WEIGHT is collected;	populated for every visit where WEIGHT is collected;	Derivation/Comment

Table 3. An Example of How SAS Macro Transforms the "Phrase" into the Different Columns in Define-XML.

Note: if the type is an integer and length is not provided, the default length will be 8 for Define-XML.

Value Level Metadata can also be used on other types of SDTM domains such as Events Domains. For example, in a DS domain, DSCAT is defined based on DSDECOD values as Value Level Metadata. Here we will introduce another way to write the specification for DSCAT, which makes it possible to automatically retrieve elements for Value Level Metadata. In order to facilitate the data manipulation**, if – then clause** will be used for generating Value Level Metadata. The variable written after if clause should be a variable in SDTM metadata based on the value of which DSCAT is defined, and comparator after the variable should be 'EQ', 'IN', 'NOTIN', 'NE','LT','LE','GT',or 'GE'. The mapping from SDTM specification DS.DSCAT to define-xml is shown in Display 14. This method applies the condition that derivation/mapping rules were different across the variable values and cannot redefine the attributes for subsets of variable values.

Disposition (DS) [Location: DS.xpt]

Variable	Label	Key	Type	Length	Controlled Terms or Format	Origin	Derivation/Comment
DSCAT	Category for Disposition Event	5	text	40	["PROTOCOL MILESTONE", "DISPOSITION EVENT"] <Category for Disposition Event>	Assigned	

Domain	Variable	Label	Type	Length	Controlled Terminology	Origin	Core	Comment — Value List Definitions Section
DS	DSCAT	Category for Disposition Event	Char	40	DSCAT	Assigned	Exp	**Value Level Metadata:** (1) if DSDECOD in ('INFORMED CONSENT OBTAINED', 'RANDOMIZE') then DSCAT = 'PROTOCOL MILESTONE' (2) if DSDECOD in ('ADVERSE EVENT', 'COMPLETED', 'LOST TO FOLLOW-UP', 'NON-COMPLIANCE WITH STUDY DRUG', 'OTHER', 'PREGNANCY', 'WITHDRAWAL BY SUBJECT', 'PROTOCOL DEVIATION', 'FAILURE TO MEET RANDOMIZATION CRITERIA') then DSCAT = 'DISPOSITION EVENT'

order

Value Level Metadata - DS [DSCAT]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
DSCAT	DSDECOD IN ("('INFORMED CONSENT OBTAINED'" , "'RANDOMIZED'")	text	40	["PROTOCOL MILESTONE", "DISPOSITION EVENT"] <Category for Disposition Event>	Assigned	DSCAT = 'PROTOCOL MILESTONE'
DSCAT	DSDECOD IN ("('ADVERSE EVENT'" , "'COMPLETED" , "'LOST TO FOLLOW-UP" , "'NON-COMPLIANCE WITH STUDY DRUG" , "'OTHER" , "'PREGNANCY" , "'WITHDRAWAL BY SUBJECT" , "'PROTOCOL DEVIATI "'FAILURE TO MEET RANOMIZATION'")		40	["PROTOCOL MILESTONE", "DISPOSITION EVENT"] <Category for Disposition Event>	Assigned	DSCAT = 'DISPOSITION EVENT'

order=1

order=2

Display 14. Mapping from DS Specification to Define-XML V2.0 for Value Level Metadata DS.DSCAT.

Value Level Metadata for QVAL

In SDTM supplemental domains, the values (QVAL) of each qualifier variable (QNAM) provide the supplemental information of non-standard variables for the main domains. These values may have different attributes, different controlled terminology, or different mapping/derivation rules. A separate worksheet, Supplemental Domain Worksheet, was created for the detailed mapping of these variables. Display 15 gives an example how QVAL is

mapped from SUPPDM Worksheet to define-xml V2.0 Value Level Metadata for SUPPDM.QVAL. For each QNAM, controlled terminology and the origin of QVAL are different. For example, when QNAM='CTYN', QVAL is subject to controlled terminology NY and mapping from CRF Page 35; while QVAL for QNAM='GROUP' is populated from external data.

Selected	Domain	QNAM	QLABEL	Controlled Terminology	Origin	Comment
Y	SUPPDM	CTYN	Is sub participating psychotherapy?	NY	CRF Page 35	
Y	SUPPDM	GROUP	Collected BL HAM-D Total Score Group		eDT	In Randomization File

Value Level Metadata - SUPPDM [QVAL]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
QVAL	QNAM EQ CTYN (Is sub participating psychotherapy?)	text	200	["N" = "No", "Y" = "Yes"] <No Yes Response>	CRF Page 35	
QVAL	QNAM EQ GROUP (Collected BL HAM-D Total Score Group)	text	200		eDT	In Randomization File

Display 15. Mapping from Supplemental Domain Worksheet of SDTM Specification to Define-XML V2.0 for SUPPDM.QVAL

Value Level Metadata Definitions Section

Each value list is defined by a **def:ValueListDef** element outside of **ItemDef** element, which will be referenced by **def:ValueListRef** element with same ODM element identifier (**OID**).

Similarly, each where clause is defined by **def:WhereClauseDef** element outside of **def:ValueListDef** element, which will be referenced by **def:WhereClauseRef** element with same **WhereClauseOID.**

```
<!-- Value List Definitions Section.                                    -->
<def:ValueListDef OID="VL.VS.VSSTRESN">
  <ItemRef ItemOID="IT.VS.VSSTRESN.VAL1" Mandatory="No" MethodOID="MT.VS.VSSTRESN.VAL1">
    <def:WhereClauseRef WhereClauseOID="WC.VS.VSSTRESN.VAL1"/>
  </ItemRef>
  <ItemRef ItemOID="IT.VS.VSSTRESN.VAL2" Mandatory="No">
    <def:WhereClauseRef WhereClauseOID="WC.VS.VSSTRESN.VAL2"/>
  </ItemRef>
  …
</def:ValueListDef>

<def:ValueListDef OID="VL.SUPPDM.QVAL">
  <ItemRef ItemOID="IT.SUPPDM.QVAL.VAL1" Mandatory="Yes">
    <def:WhereClauseRef WhereClauseOID="WC.SUPPDM.QVAL.VAL1"/>
  </ItemRef>
  <ItemRef ItemOID="IT.SUPPDM.QVAL.VAL2" Mandatory="Yes">
    <def:WhereClauseRef WhereClauseOID="WC.SUPPDM.QVAL.VAL2"/>
  </ItemRef>
  …
</def:ValueListDef>

<!-- WhereClause Definitions Section -->
<def:WhereClauseDef OID="WC.VS.VSSTRESN.VAL1">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.VS.VSTESTCD">
    <CheckValue>BMI</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.VS.VSSTRESN.VAL2">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.VS.VSTESTCD">
    <CheckValue>DIABP</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
…
<def:WhereClauseDef OID="WC.SUPPDM.QVAL.VAL1">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.SUPPDM.QNAM">
```

```
    <CheckValue>CTYN</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.SUPPDM.QVAL.VAL2">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.SUPPDM.QNAM">
    <CheckValue>GROUP</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>

<ItemDef OID="IT.VS.VSSTRESN.VAL1" Name="VSSTRESN" SASFieldName="VSSTRESN" DataType="float"
SignificantDigits="1" def:DisplayFormat="4.1" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Numeric Result/Finding in Standard Units</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.VS.VSSTRESN.VAL2" Name="VSSTRESN" SASFieldName="VSSTRESN" DataType="integer"
Length="8">
  <Description>
    <TranslatedText xml:lang="en">Numeric Result/Finding in Standard Units</TranslatedText>
  </Description>
  <def:Origin Type="CRF">
    <def:DocumentRef leafID="LF.blankcrf">
      <def:PDFPageRef PageRefs="40" Type="PhysicalRef"/>
    </def:DocumentRef>
  </def:Origin>
</ItemDef>
…
<ItemDef OID="IT.SUPPDM.QVAL.VAL1" Name="QVAL" SASFieldName="QVAL" DataType="text" Length="200">
  <Description>
    <TranslatedText xml:lang="en">Is sub participating psychotherapy?</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.NY.NY"/>
  <def:Origin Type="CRF">
    <def:DocumentRef leafID="LF.blankcrf">
      <def:PDFPageRef PageRefs="36" Type="PhysicalRef"/>
    </def:DocumentRef>
  </def:Origin>
</ItemDef>
<ItemDef OID="IT.SUPPDM.QVAL.VAL2" Name="QVAL" SASFieldName="QVAL" DataType="text" Length="200"
def:CommentOID="COM.SUPPDM.QVAL.VAL2">
<Description>
    <TranslatedText xml:lang="en">Collected BL HAM-D Total Score Group</TranslatedText>
  </Description>
  <def:Origin Type="eDT"/>
</ItemDef>
```

The source code can be automatically generated by SAS with VALUE_INFO dataset.

```
data value;
  set value_info;
  by sourcedataset sourcevariable codelist vorder;
  length line $4000;
  retain order 0;
  section = 30;  *** Value List Definitions Section ***;

  if first.sourcevariable then do;
    line= '<def:ValueListDef OID="VL.'||strip(sourcedataset)||'.'||strip(sourcevariable)||'">';
    order+1;output;
    section = 30;
  end;

  if first.vorder then do;
    line = '  <ItemRef ItemOID="IT.'||strip(sourcedataset)||'.'||strip(sourcevariable)||'.VAL'||
           strip(put(vorder,best.))||'" Mandatory="'||strip(mandatory);
    if strip(upcase(origin)) = 'DERIVED' then line = strip(line)||'" MethodOID="MT.'||
      strip(sourcedataset)||'.'||strip(sourcevariable)||'.VAL'||strip(put(vorder,best.));
    line = strip(line)||'">'; order+1; output;
```

```
    line = '    <def:WhereClauseRef WhereClauseOID="WC.'||strip(sourcedataset)||'.'||
           strip(sourcevariable)||'.VAL'||strip(put(vorder,best.))||'"/>'; order+1; output;
    line = '  </ItemRef>';order+1;output;
    section = 35;   *** Where Clause Section ***;
    line = '<def:WhereClauseDef OID="WC.'||strip(sourcedataset)||'.'||strip(sourcevariable)||
           '.VAL'|| strip(put(vorder,best.))||'">'; order+1; output;
    line = '  <RangeCheck Comparator="'||strip(wherecomparator)||'" SoftHard="Soft"
     def:ItemOID="IT.'||strip(sourcedataset)||'.'||strip(wherevariable)||'">'; order+1; output;
    section = 30;
  end;
  section = 35;
  line = '    <CheckValue>'||strip(wherecheckvalue)||'</CheckValue>'; order+1; output;
  section = 30;
  if last.vorder then do;
    section = 35;
    line = '  </RangeCheck>'; order+1; output;
    line = '</def:WhereClauseDef>'; order+1; output;
    section = 30;
  end;
  if last.vorder then do;
    section = 30;
    line = '<ItemDef OID="IT.'||strip(sourcedataset)||'.'||strip(sourcevariable)||'.VAL'||
           strip(put(vorder,best.))||'" Name="'||strip(sourcevariable)||'" SASFieldName="'||
           strip(sourcevariable)||'" DataType="'||strip(datatype);
    if not missing(significantdigits) then line = trim(line)||'" SignificantDigits="'||
           strip(put(significantdigits,best.));
    if not missing(DisplayFormat) then line= trim(line)||'" def:DisplayFormat="'||
           strip(DisplayFormat);
    line = trim(line)||'" Length="'||strip(put(length,best.));
    if strip(upcase(origin)) ne 'DERIVED' then line = strip(line)||'" def:CommentOID="COM.'||
       strip(sourcedataset)||'.'||strip(sourcevariable)||'.VAL'||strip(put(vorder,best.));
    line = strip(line)||'">'; order+1; output;
    line = '  <Description>';order+1;output;
    line = '    <TranslatedText xml:lang="en">'||strip(sourcelabel)||'</TranslatedText>';
    order+1;output;
    line = '  </Description>';order+1;output;
    if codelist ne '' then do;
      line = '  <CodeListRef CodeListOID="CL.'||strip(sourcedataset)||'.'||strip(codelist)||
             '"/>';order+1;output;
    end;
  end;
  section = 30;
  if last.sourcevariable then do; line = '</def:ValueListDef>'; order+1; output; end;
run;
```

If variable LINE generated above is output to an xml file sorting by SECTION and ORDER, the Value Level Metadata Definitions Section will be shown in Display 13 – 15.

Computational Method Definitions Section for Values

Inside Parameter Level Metadata, for each variable value of a derived variable, the algorithm for derivation is described by ODM **MethodDef** element as below. The algorithm comes from SAS data VALUE_INFO.

```
<MethodDef OID="MT.VS.VSSTRESN.VAL1" Name="CM.VS.VSSTRESN.VAL1" Type="Computation">
 <Description>
  <TranslatedText xml:lang="en">populated for every visit where WEIGHT is collected</TranslatedText>
 </Description>
</MethodDef>
```

SAS code which automatically generate computational Method Definitions Section for Variable Values is shown below.

```
data value;
  set value_info;
  by sourcedataset sourcevariable codelist vorder;
  length line $4000;
  if last.vorder then do;
    if strip(upcase(origin)) = 'DERIVED' then do;
      section = 75; *** Computational Method for Variable Value Section Number ***;
      line = '<MethodDef OID="MT.'||strip(sourcedataset)||'.'||strip(sourcevariable)||'.VAL'||
             strip(put(vorder,best.))||'" Name="CM.'|| strip(sourcedataset)||'.'||
             strip(sourcevariable)|| '.VAL'|| strip(put(vorder,best.))||'" Type="Computation">';
      order+1;output;
      line = '  <Description>';order+1;output;
      line = '    <TranslatedText xml:lang="en">'||strip(comment)||'</TranslatedText>';
      order+1;output;
```

```
      line = '  </Description>';order+1;output;
      line = '</MethodDef>';order+1;output;
    end;
  end;
run;
```

An example of Computational Method Definitions Section for variable values is shown in Display 16. As the algorithm has been referenced to the variable value VS.VSSTRESN.VAL1 defined by **MethodOID** attribute in **ItemRef** element, the derivation rules can also be shown in Derivation/Comment Column of Value Level Definitions Section in Display 13-15.

Computational Algorithms

Method	Type	Description
CM.VS.VSSTRESN.VAL1	Computation	populated for every visit where WEIGHT is collected

Display 16. Computational Method Section for Variable Value VS.VSSTRESN.VAL1 in Define-XML V2.0

Comments Definitions Section for Values

If the origin of the variable comes from CRF, Protocol, eDT, or Assigned, the comments at value level are described by **def:CommentDef** element as below.

```
<def:CommentDef OID="COM.SUPPDM.QVAL.VAL2">
  <Description>
    <TranslatedText>In Randomization File</TranslatedText>
  </Description>
</def:CommentDef>
```

SAS code to automatically generate Comment Definitions Section for Variable values is shown below.

```
data value;
  set value_info;
  by sourcedataset sourcevariable codelist vorder;
  length line $4000;
  if last.vorder then do;
    if strip(upcase(origin)) ne 'DERIVED' then do;
      section = 85; *** Comment for Variable Value Section Number ***;
      line = '<def:CommentDef OID="COM.'||strip(sourcedataset)||'.'||strip(sourcevariable) ||
             '.VAL'||strip(put(vorder,best.))||'">'; order+1;output;
      line = '  <Description>';order+1;output;
      line = '    <TranslatedText>'||strip(comment)||'</TranslatedText>';order+1;output;
      line = '  </Description>';order+1;output;
      line = '</def:CommentDef>';order+1;output;
    end;
  end;
run;
```

An example of Comment Definitions Section for Variable Values is shown in Display 17. As the comment has been referenced to the variable value SUPPDM.QVAL.VAL4 defined by **def:CommentOID** attribute in **ItemRef** element, the comments can also be shown in Derivation/Comment Column of Value Level Definitions Section in Display 13-15.

Comments

CommentOID	Description
COM.SUPPDM.QVAL.VAL2	In Randomization File

Display 17. Comments Definitions Section for Variable Value SUPPDM.QVAL.VAL4 in Define-XML V2.0

CONCLUSION

This paper introduced a customized SDTM spreadsheet specifications designed for SDTM programming and define-xml generating. A SAS Macro Tool was developed to automatically pull metadata from SDTM Specification and create Define-XML V2.0 for FDA submission. This automation significantly reduces time and the resources needed for preparation of define-xml, and guarantees the high quality of the submission. With this tool, you **DO NOT** need to be an ODM expert to create Define-XML V2.0. We hope the methodology and the SAS code provided in this paper can help you in saving your time and resources for FDA submission.

REFERENCES

1. “CDER Common Data Standards Issues Document Version 1.1”, December 2011

http://www.fda.gov/downloads/Drugs/DevelopmentApprovalProcess/FormsSubmissionRequirements/ElectronicSubmissions/UCM254113.pdf

2. CDISC Define-XML Team. “CDISC Define-XML Specification Version 2.0”, March 2013.

3. CDISC Submission Data Standards Team. “Study Data Tabulation Model Implementation Guide: Human Clinical Trials”. November 2013. http://www.cdisc.org/sdtm

4. Min Chen, Xiangchen (Bob) Cui. “A SAS® Macro Tool to Automate Generation of Define-XML V2.0 from ADaM Specification for FDA Submission”, PharmaSUG, May 2015.

7. John H. Adams, “Creating a define.xml file for ADaM and SDTM”, PharmaSUG, May 2011.

CONTACT INFORMATION

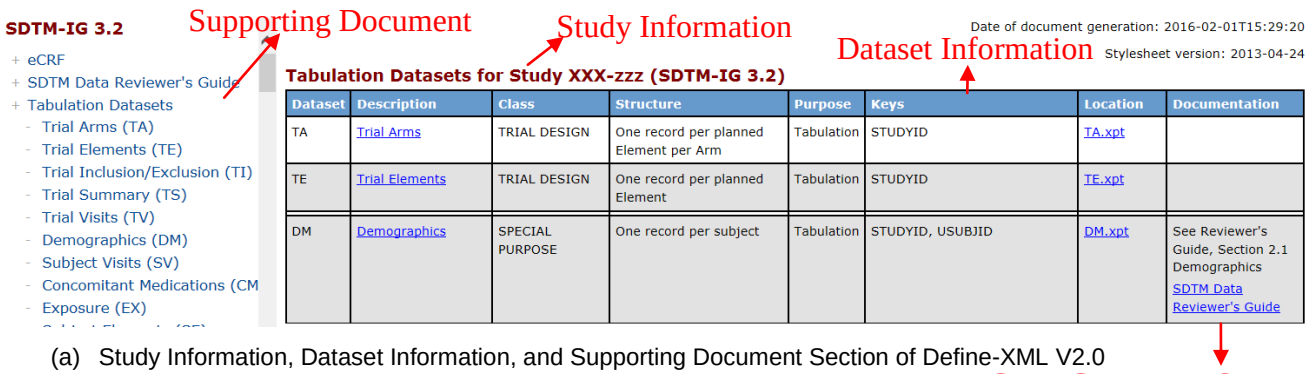
Your comments and questions are valued and encouraged. Contact the author at:

Name: Min Chen**,** Ph.D.
Enterprise: Alkermes, Inc.
Address: 852 Winter Street
City, State ZIP: Waltham, MA 02451
Work Phone: 781-609-6047
Fax: 781-609-5855
E-mail: min.chen@alkermes.com

Name: Xiangchen (Bob) Cui**,** Ph.D.
Enterprise: Alkermes, Inc.
Address: 852 Winter Street,
City, State ZIP: Waltham, MA 02451
Work Phone: 781-609-6038
Fax: 781-609-5855
E-mail:xiangchen.cui@alkermes.com

Appendix 1

An example of Define-XML V2.0



(a) Study Information, Dataset Information, and Supporting Document Section of Define-XML V2.0

Go to Comments Section for Dataset Definition

Variable Information

Vital Signs (VS) [Location: VS.xpt]

Variable	Label	Key	Type	Length	Controlled Terms or Format	Origin	Derivation/Comment
STUDYID	Study Identifier	1	text	20		CRF Page 1	
DOMAIN	Domain Abbreviation		text	2	["VS" = "Vital Signs"] <Domain Abbreviation (VS)>	Assigned	VS Go to Comments Section for Assigned Variables
USUBJID	Unique Subject Identifier	2	text	40		Derived	STUDYID + '-' + SUBJID
VSSEQ	Sequence Number		integer	8		Derived	Sort by STUDYID, USUBJID, VISITNUM, VSDTC, VSTESTCD then assign value. Start at 1 for each subject. No duplicates allowed within a subject.
VSSTRESC	Character Result/Finding in Std Format		text	20		CRF Page 39	VSORRES=VSSTRESC
VSSTRESN	Numeric Result/Finding in Standard Units		integer	8		Derived	Numeric value of VSORRES
VSSTRESU	Standard Units		text	40	Units for Vital Signs Results	CRF Page 39	

Go to Computational Method Section for Derived Variables

Go to Comments Section for Dataset Variables

(b) Variable Information Section of Define-XML V2.0

Value Level Metadata - VS [VSSTRESU]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
VSSTRESU	VSTESTCD IN ("DIABP" (Diastolic Blood Pressure) , "SYSBP '")	text	40	Units for Vital Signs Results	CRF Page 39	VSSTRESU = 'mmHg';
VSSTRESU	VSTESTCD IN ("HEIGHT" (Height))	text	40	Units for Vital Signs Results	CRF Page 39	VSSTRESU = 'cm';
VSSTRESU	VSTESTCD EQ HR (Heart Rate)	text	40	Units for Vital Signs Results	CRF Page 39	VSSTRESU = 'beats/min';
VSSTRESU	VSTESTCD EQ RESP (Respiratory Rate)	text	40	Units for Vital Signs Results	CRF Page 39	VSSTRESU = 'breaths/min';
VSSTRESU	VSTESTCD EQ TEMP (Temperature)	text	40	Units for Vital Signs Results	CRF Page 39	VSSTRESU = 'C';
VSSTRESU	VSTESTCD EQ WEIGHT (Weight)	text	40	Units for Vital Signs Results	CRF Page 39	VSSTRESU = 'kg';
VSSTRESU	VSTESTCD EQ BMI (Body Mass Index)	text	40	Units for Vital Signs Results	CRF Page 39	VSSTRESU = 'kg/m2';

Where Clause

(c) Value Lists Section of Define-XML V2.0

Go to Comments Section for Variable Values

Outcome of Event [CL.OUT]

Permitted Value (Code)
NOT RECOVERED/NOT RESOLVED [*C49494*]
RECOVERED/RESOLVED [*C49498*]
RECOVERED/RESOLVED WITH SEQUELAE [*C49495*]

Enumerated Items

NCI/CDISC Controlled Terminology Code

Domain Abbreviation (AE) [CL.AE.DOMAIN]

Permitted Value (Code)	Display Value (Decode)
AE [*C49562*]	Adverse Events

Code List

(d) Controlled Terminology Section of Define-XML V2.0

Computational Algorithms

Computational Method Definitions Section for Derived Variables

Method	Type	Description
CM.DM.USUBJID	Computation	STUDYID + '-' + SUBJID
CM.DM.DTHFL	Computation	Indicates the subject died. Valid values Y or null. Should be populated even when date of death is not known.
CM.SV.USUBJID	Computation	STUDYID + '-' + SUBJID
CM.SV.SVSTDY	Computation	Study day measured as integer days (cannot be 0). SVSTDY=(SVSTDTC - DM.RFSTDTC + x). If SVSTDTC < DM.RFSTDTC then x=0, else if SVSTDTC >= DM.RFSTDTC then x=1.
CM.SV.SVENDY	Computation	Study day measured as integer days (cannot be 0). SVENDY=(SVENDTC - DM.RFSTDTC + x). If SVENDTC < DM.RFSTDTC then x=0, else if SVENDTC >= DM.RFSTDTC then x=1.

(e) Computational Method Definitions Section of Define-XML V2.0

Comments

CommentOID	Description
COM.AE.DOMAIN	AE
COM.DA.EPOCH	BLINDED TREATMENT
COM.SUPPSV.QORIG	QORIG = "CRF" if collected on the CRF. QORIG = "ASSIGNED" for medical dictionary coding terms. QORIG = "eDT" if collected on external labs.
COM.SUPPAE.QORIG	QORIG = "CRF" if collected on the CRF. QORIG = "ASSIGNED" for medical dictionary coding terms. QORIG = "EDT" if collected on external labs.
COM.SUPPCM.QORIG	QORIG = "CRF" if collected on the CRF. QORIG = "ASSIGNED" for medical dictionary coding terms. QORIG = "EDT" if collected on external labs.
COM.SUPPDA.QORIG	QORIG = "CRF" if collected on the CRF. QORIG = "ASSIGNED" for medical dictionary coding terms. QORIG = "EDT" if collected on external labs.
COM.SUPPDS.QORIG	QORIG = "CRF" if collected on the CRF. QORIG = "ASSIGNED" for medical dictionary coding terms. QORIG = "EDT" if collected on external labs.
COM.VS.VSSTRESU.VAL1	VSSTRESU = 'mmHg';
COM.VS.VSSTRESU.VAL2	VSSTRESU = 'cm';
COM.VS.VSSTRESU.VAL3	VSSTRESU = 'beats/min';
COM.VS.VSSTRESU.VAL4	VSSTRESU = 'breaths/min';
COM.VS.VSSTRESU.VAL5	VSSTRESU = 'C';
COM.VS.VSSTRESU.VAL6	VSSTRESU = 'kg';
COM.VS.VSSTRESU.VAL7	VSSTRESU = 'kg/m2';

Comment Section for Dataset

Comment Section for Assigned Variables

Comment Section for Assigned Variable Values

(f) Comments Section of Define-XML V2.0

Display 18 An Example of Define-XML V2.0