PharmaSUG 2014 – Paper AD02

Creating Define.xml v2 Using SAS® for FDA Submissions

Qinghua (Kathy) Chen, Exelixis, Inc. South San Francisco, CA
James Lenihan, Exelixis, Inc. South San Francisco, CA

ABSTRACT

When submitting clinical data to the Food and Drug Administration (FDA), we need to submit the trial results as well as information that help the FDA understand the data. The FDA has required the CDISC Case Report Tabulation Data Definition Specification (Define-XML), which is based on the CDISC Operational Data Model (ODM), for submissions using Study Data Tabulation Model (SDTM). Electronic submission to the FDA is therefore a process of following the guidelines from CDISC and FDA. This paper illustrates how to create an FDA guidance compliant define.xml v2 from metadata using SAS®.

INTRODUCTION

In 1999, The FDA standardized the submission of clinical data using the SAS version 5 Transport Format (XPT) and the submission of metadata using Portable Document Format (PDF), respectively. In 2005, the study data specifications published by the FDA included the recommendation that data definitions (metadata) be provided as a Define-XML file. The FDA has collaborated with the Clinical Data Interchange Standards Consortium (CDISC) ever since its founding in order to standardize the content and structure of clinical trials data for regulatory submission and now the FDA has included the CDISC Case Report Tabulation Data Definition Specification (define.xml), This content and structure is based on the CDISC ODM, as part of the eCTD Study Data Specifications for the eCTD for submissions using the SDTM. In March 2013 the final version of define.xml 2.0.0 was released by the CDISC XML Technologies team. The Define-XML specification has been improved from v1 with better clarifications and some new features. The define.xml v2.0 has improved the:

- Support for CDISC Controlled Terminology
- Definition of Value Level metadata
- The documentation of the data origin or source
- The handling of comments.

The initial implementation of define.xml can be a long complicated process for statistical programmers in the Pharmaceutical Industry. Because most SAS programmers may not have had any exposure to the XML language, this may affect their production/submission time; since they will be learning the language as they are writing the SAS code to generate the XML document.

What is XML? XML is an Extensible Markup Language, designed to transport or store data. It is self-descriptive as you may make your own elements, also known as tags. XML also simplifies the data sharing since the data is stored in plain text format, and is software and hardware independent.

XML documents form a tree structure that starts at "the root" and branches to "the leaves." An XML document contains XML Elements and an XML element is everything from the element's start tag to the element's end tag. An element can contain other elements, text, attributes or a mix of all of the above. Attributes provide additional information about an element. From the example below you can see that for a given pet, type is an element and name, nickname, color and age are the attributes of this element. If you have more than one element, a cat in this example, you would have the cat block repeated with the correct information filled in.

```
<pet>
  <type="CAT">
    <name>Spot</name>
    <nickname>Pig Monster</nickname>
    <color>black</color>
    <age>18</age>
  </type>
</pet>
```

In order to view the define.xml through a browser, we need the XSLT to transform an XML docuemtn into HTML. XSLT is the recommended style sheet language of XML and it is far more sophisticated than CSS. Define.xml v2 recommend by FDA came with define2-0-0.xsl for us to use. So all we need to do is to put define.xml together and the stylesheet will handle the display.

Here is what Define.XML looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<ODM xmlns="http://www.cdisc.org/ns/odm/v1.3"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:def="http://www.cdisc.org/ns/def/v2.0"
  FileType="Snapshot" ODMVersion="1.3.2"
  FileOID="Studycdisc01-Define-XML_2.0.0"
  CreationDateTime="2012-06-21T11:07:23-05:00"
  Originator="CDISC XML Technologies Team">
  <Study OID="cdisc01">
    <GlobalVariables>
      <StudyName>CDISC01</StudyName>
      <StudyDescription>CDISC Test Study</StudyDescription>
      <ProtocolName>CDISC 01</ProtocolName>
    </GlobalVariables>
    <MetaDataVersion OID="MDV.CDISC01.SDTMIG.3.1.2.SDTM.1.2"
      Name="Study CDISC01, Data Definitions"
      Description="Study CDISC01, Data Definitions"
      def:DefineVersion="2.0.0"
      def:StandardName="CDISC SDTM"
      def:StandardVersion="3.1.2">
            < Annotated Case Report Forms (def:AnnotatedCRF) >
            < Supplemental Data Definitions (def:SupplementalDoc) >
            < Value Level Metadata (def:ValueListDef) >
            < Where Clause Definitions (def:WhereClauseDef) >
            < Domain Level Metadata (ItemGroupDef) >
            < Variable Level Metadata (ItemDef) >
            < Controlled Terminology Metadata (CodeList) >
            < Computational Algorithms (MethodDef) >
            < Comments (def:CommentDef) >
            < Referenced Documents (def:leaf) >
    </MetaDataVersion>
  </Study>
</ODM>
```

You can see define.xml v2 consists of different components. The first 20 lines should be included in each XML header. The elements in the gray box should be implemented as needed. Annotated Case Report Form, Reviewers Guide and Complex Algorithms are standalone documents while Datasets tabulation, Value Level Metadata, Controlled Terminology, Computational Algorithms and Comments are metadata embedded. The metadata for define.xml is kept in an Excel spreadsheet, partially derived from SDTM data specifications when possible, with some critical information added manually. The last three lines are the closing tags for this define file.

An Annotated Case Report Form is a CRF with SDTM annotation on it. The Reviewers Guide is a document provided to the FDA reviewer to help them understand the clinical data included and any other information that will help speed up the review process. For better readability, complex algorithms are used when derivation of certain variables gets too long to be displayed in the method cell and all the comments can be centralized in one place.

There are five steps to create a define file listed below and illustrated in the “Road Map” in Figure 1. They are:

1. Create the metadata spreadsheet from the SDTM data spec
2. Create define.xml components using SAS
3. Construct all the define.xml supporting documents
4. Create .xpt files for submitted SAS datasets
5. Construct define.xml

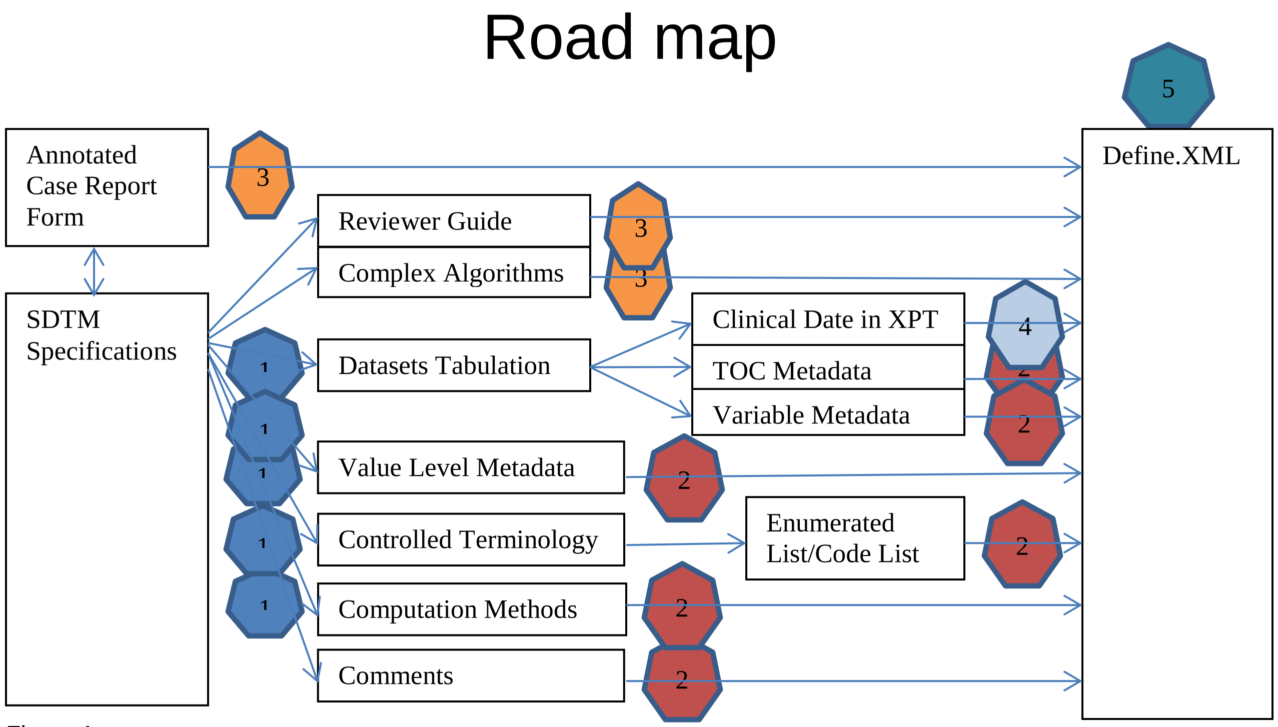


Figure 1.

CREATING METADATA SPREADSHEET FROM SDTM SPECIFICATION

In order to improve the programming efficiency, it makes more sense to have one program generate the define.xml rather than doing it manually. So it is very important to create the metadata spreadsheet for each of the sections consistently across studies.

CREATING METADATA SPREADSHEET FROM THE SDTM DATA SPEC:

The following are the metadata we need to create define.xml:

1. Header definitions
2. Datasets (TOC) definitions
3. Dataset variable definitions
4. Value level definitions
5. Controlled terminology definitions and code list

6. Computation Method
7. Comments

Header Metadata:
Header metadata contains the information about the study, SDTM version and the version of the style sheet used.

FILEOID	STUDYOID	STUDYNAME	STUDYDESCRIPTION	PROTOCOLNAME	STANDARD	VERSION	SCHEMALOCATION	STYLESHEET
XYZ123	123	XYZ123	A PHASE IIB, DOUBLE-BLIND, MULTI-CENTER, PLACEBO CONTROLLED, PARALLEL GROUP TRIAL OF ANALGEZIA HCL FOR THE TREATMENT OF CHRONIC PAIN	XYZ123	SDTM	3.1.2	http://www.cdisc.org/ns/odm/v1.3 define2-0-0.xsd	define2-0-0.xsl

TOC metadata:
The TOC metadata should contain definitions of all the datasets sent to FDA. The **ORDID** variable is used to determine the order of domains in the display.

NAME	REPEATING	ISREFERENCEDATA	PURPOSE	LABEL	STRUCTURE	DOMAINKEYS	CLASS	ARCHIVELOCATIONID	DOCUMENTATION	ORDID
TA	Yes	Yes	Tabulation	Trial Arms	One record per planned Element per Arm	STUDYID, ARMCD, TAETORD	Trial Design	../export/ta		1

***Variable Level Metadata*:**
The variable level metadata is used to describe the variables within each domain. The **KEYSEQUENCE** variable is used to provide a sequential number of keys variables so the stylesheet can use it to populate the **DOMAINKEYS** in the TOC session in define file automatically.

DOMAIN	VARNUM	VARIABLE	TYPE	LENGTH	LABEL	SIGNIFICANTDIGITS	ORIGIN	KEYSEQUENCE	DISPLAYFORMAT	COMPUTATIONMETHOD	CODELISTNAME	MANDATORY	ROLE	ROLECODELIST	VALUELISTOID
TA	1	STUDYID	text	15	Study Identifier		Derived	1				Yes	Identifier	ROLECODE	
TA	3	ARMCD	text	8	Planned Arm Code		Derived				ARMCD	Yes	Topic	ROLECODE	

Value level metadata:
The value level metadata in define.xml v2 has changed significantly from v1. With whereclause included, it not only allows us to show the condition(s) to use to subset the data, it also provides link(s) back to its source domain so you can refer to it when needed. The **VALUELISTOID** shows you which variable the value level metadata is created. The **ITEMOID** is used to define the record within that **VALUELISTOID** and **WHERECLAUSEOID** is used to display/link the subset condition(s). When a **DESCRIPTION** is needed for one particular parameter in **SUPPXX**, you can populate it with the source variable label and then it will show after the variable in the display within parentheses.

VALUELIST	VALUEO	VALUENAME	ITEMOID	*WHERECLAUSEOID*	TYPE	LEN	ORIGIN	COMPUTATION	CODELISTNA	SIGNIFICANT	DISPLAYFORMAT	MANDAT	DESCRIP

OID	RDER					GTH		METHODOID	ME	DIGITS		ORY	TION
VL.LB.LBORRES	1	LBORRES	IT.LB.LBORRES.BILI.LBCAT.CHEMISTRY.LBSPEC.BLOOD	*WC.LB.LBTESTCD.BILI.LBCAT.CHEMISTRY.LBSPEC.BLOOD*	float	3	eDT			4.2		YES	
VL.SUPPDM.QVAL	1	QVAL	IT.SUPPDM.QVAL.RACE1	*WC.SUPPDM.QNAM.RACE1*	text	5	CRF Page 6		RACE			NO	Race 1

Here is how the label is shown in Define.xml.

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
QVAL	QNAM EQ RACE1 (Race 1)	text	5	["BLACK OR AFRICAN AMERICAN", "OTHER", "WHITE"] <RACE>	CRF Page 6	

Computational Methods:
The Computational Methods section is used to display all the derivation rules used together in datasets submitted. **COMPUTATIONMETHODOID** is used to link to Derivation/Comment and Computational Method session. When derivation of a variable can't fit into a table cell nicely it should be saved in an external file called **Complex Algorithms.pdf**. You should note that this metadata is built on prototyping data therefore the **DESCRIPTION** of the computation method contains the pseudo code. When you prepare for the actual filing, the actual text needs to be written.

COMPUTATIONMETHODOID	COMPUTATIONMETHODNAME	COMPUTATIONMETHODTYPE	DESCRIPTION
MT.AESTDY	Algorithm to derive AESTDY	Computation	AESTDY = AESTDTC - RFSTDTC+1 if AESTDTC is on or after RFSTDTC. AESTDTC - RFSTDTC if AESTDTC precedes RFSTDTC

Codelist:
Codelist is a unique subset of the SDTM controlled terminology. For CDISC codelist, aliases are included at both the **CodeListItem** and **EnumeratedItem** levels. When a description is needed for a coded term, the **DECODED** variable should be marked as 'YES' so both the coded term and the translated term will be in display. Rank is provided for each **EnumeratedItem** to indicate the temporal order in analysis computation.

CODELISTNAME	*CODELISTLABEL*	*RANK*	*CODEDVALUE*	*TRANSLATED*	*TYPE*	*CODELISTDICTIONARY*	*CODELISTVERSION*	*DECODED*	*sourcevariable*	*sourcevalue*	*sourcetype*
AE	***Domain Abbreviation (AE)***		***AE***	***Adverse Events***	***text***			***YES***			
DM	*Domain Abbreviation (DM)*		***DM***	*Demographics*	***text***			***YES***			
ACN	*Action Taken with Study Treatment*	*1*	*DOSE INCREASED*	*DOSE INCREASED*	***text***				*aeaction*	*3*	*number*
ACN	*Action Taken with Study Treatment*	*2*	*DOSE NOT CHANGED*	*DOSE NOT CHANGED*	***text***				*aeaction*	*4*	*number*

ACN	*Action Taken with Study Treatment*	*3*	*DOSE REDUCED*	*DOSE REDUCED*	***text***				*aeaction*	*2*	*number*

Comments:
When a comment or an external document is referenced, we should create a record with COMMENTOID and a description for it. Here is how it looks:

COMMENTOID	**DESCRIPTION**
COM.LBREFID	Accession number
COM.RELTYPE	All values are null since this is used only when identifying a dataset-level relationship.

As you can see, some part of this metadata specification can be populated programmatically depending on how your SDTM/ADaM specifications are constructed. Some information just has to be filled manually. You can find an example how we created this metadata specification in Appendix A. In this case, we use a macro to create variable level metadata from SAS datasets and SDTM specification, and read in the metadata template for all other tabs. The benefit of doing this is that when template changes, we don't have to go back and change the SAS program and thus makes our programs easier to maintain.

CREATE DEFNE.XML COMPONENTS USING SAS

Once metadata specification is completed, we can use it to populate the define file. In order to populate the define file correctly, we need to know how each variable is used in define file. The Define-XML-2-0-Specification section 5 describes how XML file is constructed, and how each element(s) and/or attribute(s) is defined/constructed. Based on that specification, we can map the metadata into an element or attribute of ODM to form the define file in the end.

READING IN METADATA

The creation of define.xml basically involves taking data from the metadata spreadsheet one tab at a time and writing it out in .XML format as a text file. The following code is used to read in the DEFINE_HEADER_METADATA sheet from SDTM_METADATA.xls and convert the metadata to a SAS dataset called define_header. We should repeat this code for all other sheets until we read in all the information needed for define.xml.

```
%xls2sas( _datarow=2,
               _indir=../&path,
               _infile=SDTM_METADATA.xls,
               _labels=1,
               _outdata=define_header,
               _sheet=DEFINE_HEADER_METADATA,
               _subset=,
               _vars=1);
```

CREATING .TXT FILE ONE FOR EACH OF THE COMPONENTS

Define_header.txt contains the header information for the XML file as well as the links to each component of the define file. Variables studyoid, studyname, studydescription, protocolname, standard, version, schemalocation read in from METADATA tab were populated into Define.xml elements accordingly. In addition to that, the link to the external documents blankcrf.pdf and Reviewer's Guide are created in here by **def:AnnotatedCRF** and **def:SupplementalDoc.**

```
filename dheader "define_header.txt";
data define_header;
    set define_header;

    file dheader notitles;
```

```
       creationdate = compress(put(datetime(), IS8601DT.));

    put @1 '<?xml version="1.0" encoding="UTF-8" ?>' /
        @1 '<?xml-stylesheet type="text/xsl" href="' stylesheet +(-1) '"?>' /
        @1 '<!--
******************************************************************************* -->' /
        @1 '<!-- File: define2-0-0.xml
-->' /
        @1 "<!-- Date: &sysdate9.
-->" /
        @1 '<!-- Description: Define2-0-0.xml file for '   studyname +(-1) '
-->' /
        @1 '<!--
******************************************************************************* -->' /
        @1 '<ODM' /
        @3 'xmlns="http://www.cdisc.org/ns/odm/v1.3"' /
        @3 'xmlns:xlink="http://www.w3.org/1999/xlink"' /
        @3 'xmlns:def="http://www.cdisc.org/ns/def/v2.0"' /
        @3 'ODMVersion="1.3.2"' /
        @3 'FileOID="StudyXYZ123-Define-XML_2.0.0"' /
        @3 'FileType="Snapshot"' /
        @3 'CreationDateTime="' creationdate +(-1) '"' /
       @3 'Originator="CDISC XML Technologies Team"' /
       @3 'SourceSystem="MH-System">' /
       @3 'SourceSystemVersion="2.0.1">' /
        @3 '<Study OID="' studyoid +(-1) '">' /
        @3 '<GlobalVariables>' /
        @5 '<StudyName>' studyname +(-1) '</StudyName>' /
        @5 '<StudyDescription>' studydescription +(-1) '</StudyDescription>' /
        @5 '<ProtocolName>' protocolname +(-1) '</ProtocolName>' /
        @3 '</GlobalVariables>' /
        @3 '<MetaDataVersion OID="CDISC.' standard +(-1) '.' version +(-1) '"' /
        @5 'Name="' studyname +(-1) ',Data Definitions"' /
        @5 'Description="' studyname +(-1) ',Data Definitions"' /
        @5 'def:DefineVersion="2.0.0"' /
        @5 'def:StandardName="' standard +(-1) '-IG"' /
        @5 'def:StandardVersion="' version +(-1) '">' /
        %if "&standard" = "SDTM" %then
       %do;
            @5 '<def:AnnotatedCRF>' /
            @7 '<def:DocumentRef leafID="LF.blankcrf"/>' /
            @5 '</def:AnnotatedCRF>' /
        %end;
           @5 '<def:SupplementalDoc>' /
            @7 '<def:DocumentRef leafID="LF.ReviewersGuide"/>' /
            @7 '<def:DocumentRef leafID="LF.ComplexAlgorithms"/>' /
          @5 '</def:SupplementalDoc>' /;
run;
```

TOC OF DOMAINS

The Itemgroupdef.txt is used to create the TOC part of define.xml. **ItemGroupDef** is used to define the ItemGroup and **ItemRef** is used to refer to the key variables specified in variable metadata and any comments needed. The variables highlighted in bold black fonts below are read from TOC_METADATA AND VARIABLE_METADATA sheets.

```
filename igdef "itemgroupdef.txt";
data itemgroupdef;
    set itemgroup;
    by ord oid;
    length label $ 40;
    file igdef notitles;
    if first.oid then do;
        put @5 "<!-- ******************************************* -->" /
            @5 "<!-- " oid   @25   "ItemGroupDef INFORMATION *** -->" /
            @5 "<!-- ******************************************* -->" /
            @5 '<ItemGroupDef OID="IG.' oid +(-1) '"' /
           @7 'Domain="' name +(-1) '"' /
            @7 'Name="' name +(-1) '"' /
            @7 'Repeating="' repeating +(-1) '"' /
            @7 'IsReferenceData="' isreferencedata +(-1) '"' /
            @7 'SASDatasetName="' name +(-1) '"' /
            @7 'Purpose="' purpose +(-1) '"' /
            @7 'def:Structure="' structure +(-1) '"' /
            @7 'def:Class="' class +(-1) '"' /
            @7 'def:CommentOID="' documentation +(-1) '"' /
            @7 'def:ArchiveLocationID="LF.' archivelocationid +(-1) '">' /;

        put @7 '<Description>' /
           @7 '<TranslatedText xml:lang="en">' label1 +(-1) '</TranslatedText>' /
            @7 '</Description>';
      end;

    put @7 '<ItemRef ItemOID="IT.' itemoid +(-1) '"' /
        @9 'OrderNumber="' varnum +(-1) '"' /
        @9 'Mandatory="' mandatory +(-1) '" / ;
        if keysequence ne ' ' then
           put @9 'KeySequence="' keysequence +(-1) '"' / ;
        if computationmethodoid ne ' ' then
           put @9 'MethodOID="' computationmethodoid +(-1) '"' / ;
    put '/>';

    if last.oid then
      put @7 "<!-- **************************************************** -->" /
          @7 "<!-- def:leaf details for hypertext linking the dataset   -->" /
          @7 "<!-- **************************************************** -->" /
          @7 '<def:leaf ID="LF.' archivelocationid +(-1) '" xlink:href="'
archivelocationid +(-1) '.xpt">' /
          @9 '<def:title>' archivelocationid +(-1)  '.xpt </def:title>' /
          @7 '</def:leaf>' /
          @5 '</ItemGroupDef>';
run;
```

VARIABLE LEVEL:

The Itemdef.txt is used to create the variable level part of define.xml. **ItemDef** is used to create the variable element. The **CodeListRef** and **def:ValueListRef** are used for codelist display and value level data drill down respectively. The variables used here are from VARIABLE_METADATA sheet.

```
filename idef "itemdef.txt";

data itemdef;
    set VARIABLE_METADATA end=eof;
    by oid;

    file idef notitles;

    if _n_ = 1 then
      put @5 "<!-- ************************************************************ -->" /
          @5 "<!-- The details of each variable is here for all domains         -->" /
          @5 "<!-- ************************************************************ -->" ;

    put @5 '<ItemDef OID="IT.' itemoid +(-1) '"' /
```

```
        @7 'Name="' variable +(-1) '"' /
        @7 'DataType="' type +(-1) '"' /
        @7 'Length="' length +(-1) '"' /
        @7 'SASFieldName="' variable +(-1) '"' ;

    if significantdigits ne '' then
       put @7 'SignificantDigitis="' significantdigits +(-1) '"';
    if displayformat ne '' then
       put @7 'def:DisplayFormat="' displayformat +(-1) '"';

       put @5 '>';

       put @7 '<Description>' /
           @7 '<TranslatedText xml:lang="en">' label +(-1) '</TranslatedText>' /
           @7 '</Description>';

    if codelistname ne '' then
       put @7 '<CodeListRef CodeListOID="CL.' codelistname +(-1) '"/>';

       put @7 '<def:Origin Type="' origint +(-1) '"' /;
       if origint ^= "CRF" then
          put  @7 '/>' ;
       else
          put @7 '>' /
             @9 '<def:DocumentRef leafID="LF.blankcrf">' /
             @11 '<def:PDFPageRef PageRefs=" ' originpn +(-1)'" Type="PhysicalRef"/>'
/
             @9 '</def:DocumentRef>' /
             @7 '</def:Origin>' /;

    if valuelistoid ne '' then
       put @7 '<def:ValueListRef ValueListOID="' valuelistoid +(-1) '"/>';

    put @5 '</ItemDef>';
run;
```

VALUE LEVEL:

Value Level Metadata should be provided when there is a need to describe different attributes for subsetting the data within a column. It can also be used on other types of SDTM domains to provide information useful for data interpretation but, the value level metadata approach is very complicated so it is recommended to use it only when it is absolutely necessary. The value level metadata consists of three parts: the itemdef_value, value list and the whereclause list. The Itemdef_value defines how the value level metadata is displayed, and the where clause shows the subset condition in define.xml and the value list is used to link to the whereclause definition. The **ItemDef** is used to create the valuelist item and **def:WhereClauseDef** and **def:ValueListDef** are used to create the whereclause and valuelist. Conditional **Description** is used for variables in SUPPXX domains that need a label. The variables read from valuelevel metadata are used here. As you can see from the code below, we need to parse the WhereClauseId so stylesheet can get the information it needs and display it correctly in the browser. Also it is worthwhile to mention that this code is for prototyping, and cannot handle all real-world situations as yet. You may find it fun to dig through the stylesheet and try to figure out how to make it work.

```
filename idefvl "itemdef_value.txt";

data itemdefvalue;
    set valuelevel end=eof;
    by valuelistoid;

    file idefvl notitles;

    if _n_ = 1 then
      put @5 "<!-- ************************************************************ -->" /
          @5 "<!-- The details of value level items are here                  -->" /
          @5 "<!-- ************************************************************ -->" ;

    put @5 '<ItemDef OID="' itemoid +(-1) '"' /
        @7 'Name="' valuename +(-1) '"' /
        @7 'DataType="' type +(-1) '"' /
        @7 'Length="' length +(-1) '"';
    if significantdigits ne '' then
      put @7 'SignificantDigitis="' significantdigits +(-1) '"';
    if displayformat ne '' then
      put @7 'def:DisplayFormat="' displayformat +(-1) '"';

      put @5 '>';

    if description ne ' ' then
       put @7 '<Description>' /
           @7 '<TranslatedText xml:lang="en">' description +(-1) '</TranslatedText>' /
           @7 '</Description>';

    if codelistname ne '' then
       put @7 '<CodeListRef CodeListOID="CL.' codelistname +(-1) '"/>';

       put @7 '<def:Origin Type="' origint +(-1) '"' /;
       if origint ^= "CRF" then
          put  @7 '/>' ;
       else
          put @7 '>' /
              @9 '<def:DocumentRef leafID="LF.blankcrf">' /
              @11 '<def:PDFPageRef PageRefs=" ' originpn +(-1)'" Type="PhysicalRef"/>'
/
              @9 '</def:DocumentRef>' /
              @7 '</def:Origin>' /;
    put @5 '</ItemDef>';
run;


filename whrlist "wherelist.txt";
data valuelevel;
  set valuelevel;
    by valuelistoid itemoid;

    file whrlist notitles;

    if _n_ = 1 then
      put @5 "<!-- *************************************************** -->" /
          @5 "<!-- VALUE LEVEL WHERE CLAUSE DEFINITION INFORMATION  ** -->" /
          @5 "<!-- ***************************************************** -->";

    if first.itemoid then;
      put @5 '<def:WhereClauseDef OID="' whereclauseoid +(-1) '">';

    do i=3 to countw(whereclauseoid,'.')-1 by 2;
        varcat=compress('IT.' || scan(whereclauseoid,2,'.') || '.' ||
scan(whereclauseoid,i,'.'))
 ;
        varval=compress(scan(whereclauseoid,i+1));
        put @7 '<RangeCheck SoftHard="Soft" def:ItemOID="' varcat +(-1) '"
Comparator="EQ">' /
            @11 '<CheckValue>' varval +(-1) '</CheckValue>' /
            @7 '</RangeCheck>';
    end;

    if last.itemoid then
      put @5 '</def:WhereClauseDef>';
run;


filename vallist "valuelist.txt";
data valuelevel;
  set valuelevel;
    by valuelistoid;

    file vallist notitles;

    if _n_ = 1 then
      put @5 "<!-- ******************************************* -->" /
          @5 "<!-- VALUE LEVEL LIST DEFINITION INFORMATION  ** -->" /
          @5 "<!-- ******************************************* -->";

    if first.valuelistoid then
      put @5 '<def:ValueListDef OID="' valuelistoid +(-1) '">';

    put @7 '<ItemRef ItemOID="' itemoid +(-1) '"' /
        @9 'OrderNumber="' valueorder +(-1) '"' /
        @9 'Mandatory="' mandatory +(-1) '"' ;
        if computationmethodoid ne '' then
            put @9 'MethodOID="' computationmethodoid +(-1) '"';
    put @9 '>' /
        @9 '<def:WhereClauseRef WhereClauseOID="' whereclauseoid +(-1) '"/>' /
        @7 '</ItemRef>';

    if last.valuelistoid then
      put @5 '</def:ValueListDef>';
run;
```

It is worthwhile to mention that the way the stylesheet handles the variable label in the whereclause is different for the SUPPXX domains than others. For the value level metadata in case of supplemental qualifiers, the stylesheet will display the description. This is conditional on the availability of a description. (Such as RACE1, RACE2, .., RACEOTH will have "Race 1", "Race 2", ..., "Race, Other"). For other types of whereclauses the stylesheet will try to display the decoded value if controlled terminology is attached to the variable that is part of the whereclause.

CODELIST:

The Codelist section has changed dramatically from v1 as well. Not only did the Codelist CODE get added to the display, codelist has been separated into **EnumeratedItem** and **CodeListItem**. The Alias Context attribute refereneces “nci:ExtCodeID” to indicate the NCI/CDISC SDTM Terminology standards. When the coded value is the same as the translated value, only code value is shown. If the codelist is saved in an external dictionary, the separated section will be created just for that.

```
filename codes "codelist.txt";
data codelists;
    set codelists end=eof;
    by codelistname rank;
    length codelistname $200;

    file codes notitles;

    if _n_ = 1 then
      put @5 "<!-- ************************************************************ -->" /
          @5 "<!-- Codelists are presented below                                -->" /
          @5 "<!-- ************************************************************ -->" ;

    if first.codelistname then do;
      if index(upcase(ncipreferredterm),'DOMAIN') then
         put @5 '<CodeList OID="CL.' codelistname +(-1) '.DOMAIN"';
      else put @5 '<CodeList OID="CL.' codelistname +(-1) '"';
      put @7 'Name="' codelistlabel +(-1) '"' /
          @7 'DataType="text">';
    end;
    **** output codelists that are not external dictionaries;
    if codelistdictionary = '' then do;
       if decoded ne 'YES' then do;
           put @7  '<EnumeratedItem CodedValue="' codedvalue +(-1) '"' @;
           if rank ne . then
              put ' OrderNumber="' rank +(-1) '">';
           else
              put '>';
           if code ne ' ' then
              put  @7  '<Alias Name="' code +(-1) '" Context="nci:ExtCodeID"/>';
           put @7  '</EnumeratedItem>';
       end;
       else do;
           put @7  '<CodeListItem CodedValue="' codedvalue +(-1) '"' @;
           if rank ne . then
              put ' OrderNumber="' rank +(-1) '">';
           else
              put '>';
           put @9  '<Decode>' /
               @11 '<TranslatedText>' translated +(-1) '</TranslatedText>' /
               @9  '</Decode>' ;
           if code ne ' ' then
              put  @7  '<Alias Name="' code +(-1) '" Context="nci:ExtCodeID"/>';
           put @7  '</CodeListItem>';
       end;
       if codelist ne ' ' then
          put  @7  '<Alias Name="' codelist +(-1) '" Context="nci:ExtCodeID"/>';
    end;
    **** output codelists that are pointers to external codelists;
    else if codelistdictionary ne '' then
      put @7 '<ExternalCodeList Dictionary="' codelistdictionary +(-1)
             '" Version="' codelistversion +(-1) '"/>';

    if last.codelistname then
      put @5 '</CodeList>';

run;
```

COMPUTATIONAL METHODS:

MethodDef is used to display the computation method used.

```
filename comp "compmethod.txt";
data compmethods;
    set compmethod;

    file comp notitles;

    if _n_ = 1 then
    put @5 "<!-- ******************************************* -->" /
        @5 "<!-- COMPUTATIONAL METHOD INFORMATION        *** -->" /
        @5 "<!-- ******************************************* -->";
    put @5 '<MethodDef OID="' computationmethodoid +(-1) '"'/
       @5 'Name="' computationmethodname +(-1) '"' /
       @5 'Type="' computationmethodtype +(-1)'">' /
        @7 '<Description>' /
       @7 '<TranslatedText xml:lang="en">' description +(-1) '</TranslatedText>' /
        @7 '</Description>' /
       @5 '</MethodDef>';
run;
```

COMMENTS:

def:CommentDef is used to display the comments used.

```
filename comt "comments.txt";
data comments;
    set comments;

    file comt notitles;

    if _n_ = 1 then
    put @5 "<!-- ******************************* -->" /
        @5 "<!-- COMMENTS INFORMATION        *** -->" /
        @5 "<!-- ******************************** -->";
    put @5 '<def:CommentDef OID="' commentoid +(-1) '">'/
        @7 '<Description>' /
       @7 '<TranslatedText xml:lang="en">' description +(-1) '</TranslatedText>' /
        @7 '</Description>' /
       @5 '</def:CommentDef>';
run;
```

LINKS FOR EXTERNAL DOCUMENTS:

def:leaf is used to link the external documents involved. It is very important to know that it is located after the comments in define file. The closing tag for define.xml is also included at the end for convenience.

```
filename leaves "leaves.txt";
data leaves;

    file leaves notitles;

    put @5 '<def:leaf ID="LF.blankcrf" xlink:href="blankcrf.pdf">' /
        @7 '<def:title>Annotated Case Report Form</def:title>' /
        @5 '</def:leaf>'/
        @5 '<def:leaf ID="LF.ReviewersGuide"
xlink:href="reviewersguide.pdf">' /
        @7 '<def:title>Reviewers Guide</def:title>' /
        @5 '</def:leaf>'/

        @5 '<def:leaf ID="LF.ComplexAlgorithms"
xlink:href="complexalgorithms.pdf">' /
        @7 '<def:title>Complex Algorithms</def:title>' /
        @5 '</def:leaf>'/;

    put @3 '</MetaDataVersion>' /
        @1 '</Study>' /
        @1 '</ODM>';
run;
```

CONSTRUCT ALL THE DEFINE.XML SUPPORTING DOCUMENTS

Create Annotated CRF based on SDTM specifications and put lengthy data derivation rules if any in Complex Algorithms. For anything else that will help the FDA to understand our data to speed up the review process, we should put that information in the Reviewer's Guide.

CREATING .XPT FILES FOR CLINICAL DATA

Since FDA standardized the submission data format in 1999, converting SAS datasets to .xpt files in version 5 is a must. There are different ways to convert SAS datasets to .xpt file, and the following SAS code provides one example:

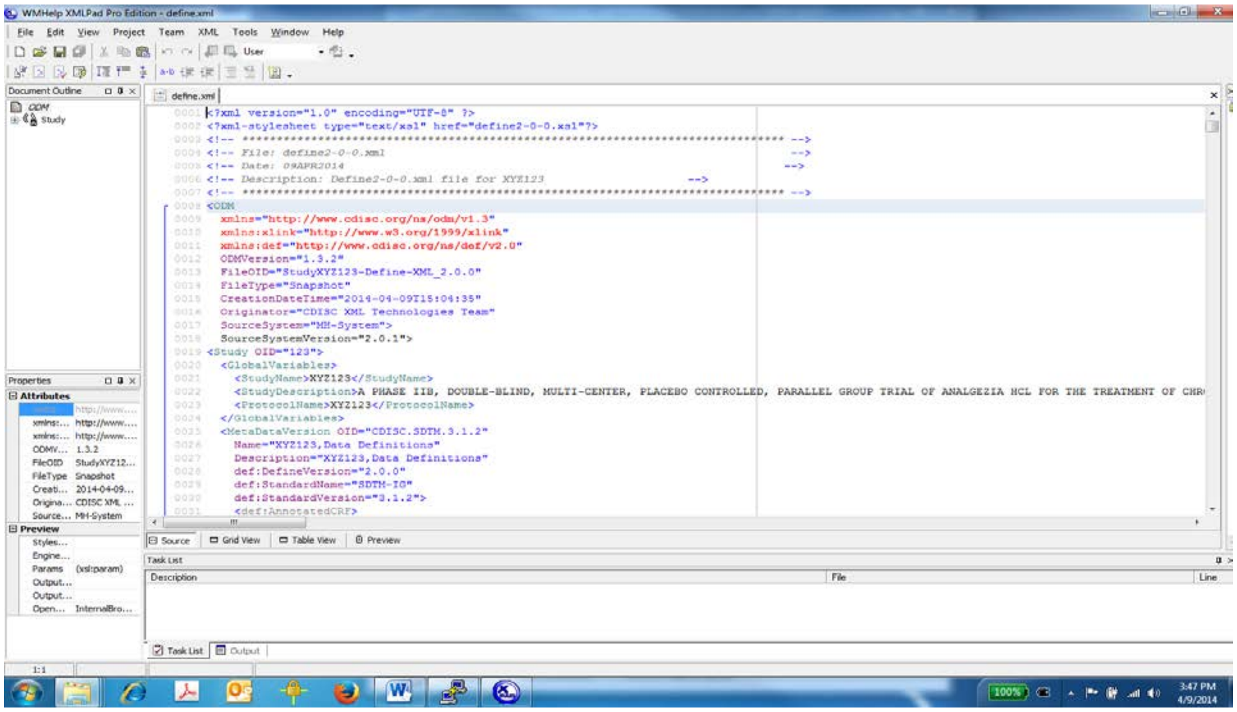
```
Libname source ‘SAS-data-library’;
Libname xportout xport ‘transport-file’;

Proc copy in=source
               out=xportout memtype=data;
run;
```

CONSTRUCT DEFINE.XML

The following is the command that was used in Unix to concatenate the text files together to form the define.xml. Please note that the order of elements in define.xml is important when you do XML Schema Validation for define.xml. Hence, it is better to follow the Globel Element Ordering on Define-XML Specification page 93 when you start the program.

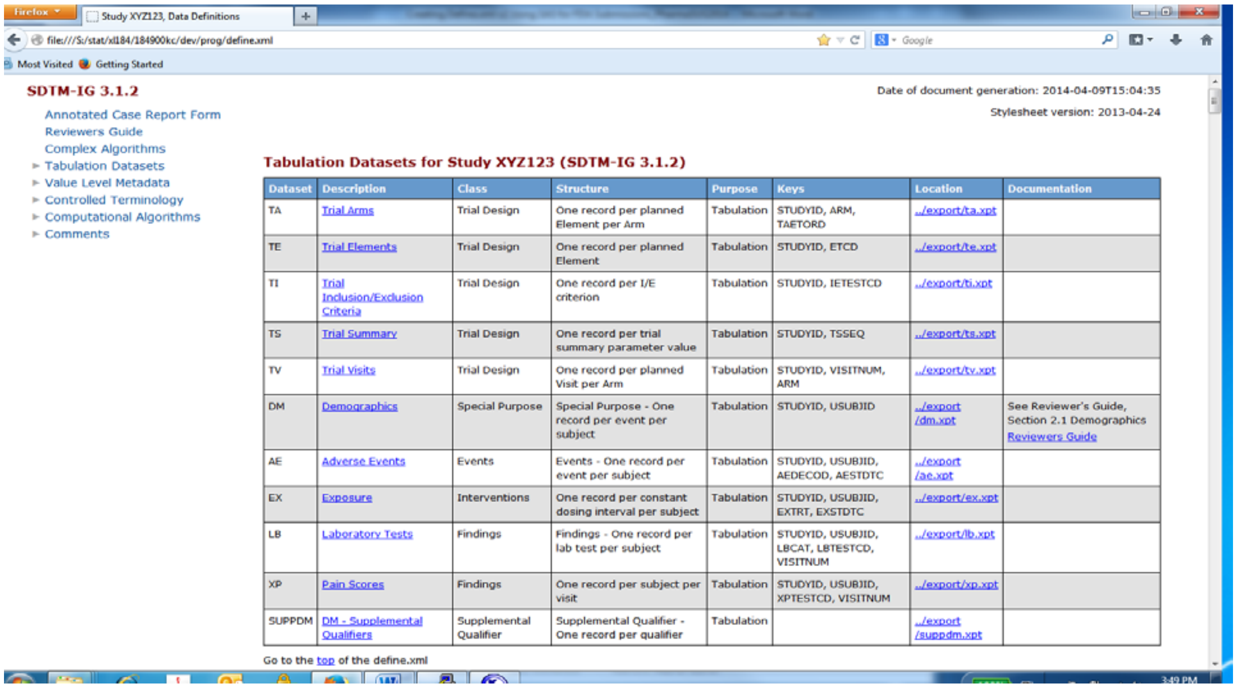
```
cat define_header.txt  valuelist.txt  wherelist.txt  itemgroupdef.txt  itemdef.txt  itemdef_value.txt  codelist.txt
compmethod.txt  comments.txt  leaves.txt  > define.xml
```

With the Stylesheet's help, the Define.xml v2 (Figure 2) can be viewed via IE or Firefox (Figure 3)



```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="define2-0-0.xsl"?>
<!-- ************************************************************************* -->
<!-- File: define2-0-0.xml                                                      -->
<!-- Date: 09APR2014                                                           -->
<!-- Description: Define2-0-0.xml file for XYZ123                  -->
<!-- ************************************************************************* -->
<ODM
  xmlns="http://www.cdisc.org/ns/odm/v1.3"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:def="http://www.cdisc.org/ns/def/v2.0"
  ODMVersion="1.3.2"
  FileOID="StudyXYZ123-Define-XML_2.0.0"
  FileType="Snapshot"
  CreationDateTime="2014-04-09T15:04:35"
  Originator="CDISC XML Technologies Team"
  SourceSystem="MH-System">
  SourceSystemVersion="2.0.1">
<Study OID="123">
  <GlobalVariables>
    <StudyName>XYZ123</StudyName>
    <StudyDescription>A PHASE IIB, DOUBLE-BLIND, MULTI-CENTER, PLACEBO CONTROLLED, PARALLEL GROUP TRIAL OF ANALGEZIA HCL FOR THE TREATMENT OF CHR
    <ProtocolName>XYZ123</ProtocolName>
  </GlobalVariables>
  <MetaDataVersion OID="CDISC.SDTM.3.1.2"
    Name="XYZ123,Data Definitions"
    Description="XYZ123,Data Definitions"
    def:DefineVersion="2.0.0"
    def:StandardName="SDTM-IG"
    def:StandardVersion="3.1.2">
    <def:AnnotatedCRF>
```

Figure 2.



SDTM-IG 3.1.2

Date of document generation: 2014-04-09T15:04:35

Stylesheet version: 2013-04-24

- Annotated Case Report Form
- Reviewers Guide
- Complex Algorithms
- Tabulation Datasets
- Value Level Metadata
- Controlled Terminology
- Computational Algorithms
- Comments

Tabulation Datasets for Study XYZ123 (SDTM-IG 3.1.2)

Dataset	Description	Class	Structure	Purpose	Keys	Location	Documentation
TA	Trial Arms	Trial Design	One record per planned Element per Arm	Tabulation	STUDYID, ARM, TAETORD	../export/ta.xpt	
TE	Trial Elements	Trial Design	One record per planned Element	Tabulation	STUDYID, ETCD	../export/te.xpt	
TI	Trial Inclusion/Exclusion Criteria	Trial Design	One record per I/E criterion	Tabulation	STUDYID, IETESTCD	../export/ti.xpt	
TS	Trial Summary	Trial Design	One record per trial summary parameter value	Tabulation	STUDYID, TSSEQ	../export/ts.xpt	
TV	Trial Visits	Trial Design	One record per planned Visit per Arm	Tabulation	STUDYID, VISITNUM, ARM	../export/tv.xpt	
DM	Demographics	Special Purpose	Special Purpose - One record per event per subject	Tabulation	STUDYID, USUBJID	../export /dm.xpt	See Reviewer's Guide, Section 2.1 Demographics Reviewers Guide
AE	Adverse Events	Events	Events - One record per event per subject	Tabulation	STUDYID, USUBJID, AEDECOD, AESTDTC	../export /ae.xpt	
EX	Exposure	Interventions	One record per constant dosing interval per subject	Tabulation	STUDYID, USUBJID, EXTRT, EXSTDTC	../export/ex.xpt	
LB	Laboratory Tests	Findings	Findings - One record per lab test per subject	Tabulation	STUDYID, USUBJID, LBCAT, LBTESTCD, VISITNUM	../export/lb.xpt	
XP	Pain Scores	Findings	One record per subject per visit	Tabulation	STUDYID, USUBJID, XPTESTCD, VISITNUM	../export/xp.xpt	
SUPPDM	DM - Supplemental Qualifiers	Supplemental Qualifier	Supplemental Qualifier - One record per qualifier	Tabulation		../export /suppdm.xpt	

Go to the top of the define.xml

Figure 3.

CONCLUSIONS

Creating define.xml is difficult especially if you don't have any knowledge about XML at the beginning. However, with the approach I have outlined above it is readily achievable.

REFERENCES

- http://support.sas.com/publishing/authors/holland_chris.html
- http://support.sas.com/resources/papers/proceedings13/201-2013.pdf
- http://support.sas.com/resources/papers/proceedings10/157-2010.pdf
- Define-XML v2 – What's New by Lex Jansen, SAS Institute Inc., Cary, NC, USA
- Implementing CDISC Using SAS: by Chris Holland and Jack Shostak
- Define-XML-2-0-Specification

ACKNOWLEDGEMENTS

Lex Jassen, SAS Institute.
Amos Shu, Medimmune Inc.

CONTACT INFORMATION

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

Qinghua (Kathy) Chen
Exelixis Inc
210 East Grand Ave,
South San Francisco
CA, 94080
Email: kchen@exelixis.com

James Lenihan
Exelixis Inc.
210 East Grand Ave,
South San Francisco
CA, 94080
Email: jlenihan@exelixis.com

APPENDIX A:

```
/***************************************************
*H*
*H* PROGRAM: mk_varmeta.sas
*H*
*H* USAGE:    create the SDTM variable_metadata sheet from the STDM specs
*H* REQUIRES anadata:
*H* REQUIRES rawdata:
*H* REQUIRES sdtmdata:
*H* REQUIRES export:    XLXXX_SDMM_Speciifcatins_V8.0.xls
*H* REQUIRES macros:    %xls2sas %sas2xls
*H* PRODUCES:
*H*
*H*
*H* REVISION HISTORY:
*H*     20131216 jkl Created.
*H*
*H* $Id:$
**************************************************/

options mlogic mprint symbolgen validvarname=upcase ;

*PN---------------------------------------------------------------*;
*PN---------------------------------------------------------------*;
*PN macro that
         1) converts an individual sheet from an xls file to SAS
         2) select and label variables
         3) creates the dataset "final" by concatinating the datasets
*PN---------------------------------------------------------------*;
*PN---------------------------------------------------------------*;

*PN-----------------------------------------------*;
*PN macro to convert xls sdtmdata sheet to sas data*;
*PN-----------------------------------------------*;
%xls2sas(_datarow = 2,
         _indir = ../import,
        _infile = Def_head.xls,
        _labels =1,
       _outdata = DEFINE_HEADER_METADATA,
          _sheet =sheet1  ,
          _vars =  )
        ;
 proc print data= DEFINE_HEADER_METADATA;
 title 'DEFINE_HEADER';
 run;


%macro doit(dsn=,sort=);

%xls2sas(_datarow  =  7,
           _indir  =  ,
          _infile  =  XL184307_SDTM_Specifications_V8.0.xls,
          _labels  =  6,
          _outdata =  ,
          _sheet   =  &dsn
         );

data &dsn;
     retain domain var16 var8 var10 var11 var9 var17 var15 var12 var14 var18 var19 ;
     set &dsn(drop=VAR1-VAR7 );

     *PN------------------------------------*;
     *PN crete the DOMAIN and SORT variables*;
     *PN------------------------------------*;
     domain = "&dsn";
     sort = "&sort";

     label   var8   = 'VARIABLE'
             var9   = 'LABEL'
             var10  = 'TYPE'
             var11  = 'LENGTH'
             var12  = 'DISPLAYFORMAT'
             var13  = 'FLAG'
             var14  = 'COMPUTATIONALMETHODOID'
             var15  = 'COMMENTS'
             var16  = 'VARNUM'
             var17  = 'ORIGIN'
             var18  = 'MANDATORY'
             var19  = 'ROLECODELIST'
             domain = 'DOMAIN'
     ;

     if var8= '' or var13 ne ''  then delete;

     drop var13 ;
run;

*PN-------------------------*;
*PN concatenate the datasets*;
*PN-------------------------*;
proc append base=final data=&dsn force;
run;

proc datasets library=work;
     delete &dsn;
quit;

%mend;

*PN--------------------------*;
*PN import from the SDTM spec*;
*PN--------------------------*;
%doit(dsn=AE,sort=A);
%doit(dsn=CD,sort=C);
%doit(dsn=CM,sort=D);
%doit(dsn=DA,sort=E);
%doit(dsn=DD,sort=F);
%doit(dsn=DM,sort=G);

proc contents data=final;
run;

*PN---------------------------------------------------------------------------------
-----*;
*PN------------------------  THIS SECTION IS FOR SUPP DATASETS ----------------------
-----*;
*PN---------------------------------------------------------------------------------
-----*;

%macro supp(dsn=,sort=);

*PN---------------------------*;
*PN get the contents datasets *;
*PN---------------------------*;
proc contents data=sdtmdata.&dsn out=&dsn noprint;
```

```
run;

proc sort data=&dsn;
     by varnum;
run;

proc print data=&dsn;
      title "supp &dsn";
run;

data &dsn;
     length mandatory $3 type $4;
     set &dsn(drop = type rename=(memname=domain name=variable));
     by varnum;

     *PN------------------------------------------------------*;
     *PN populate the Mandtory column in the variable_metadata*;
     *PN------------------------------------------------------*;
     if varnum in('4','5','10') then mandatory='No';
         else  mandatory='Yes';

     *PN----------------------------------------------*;
     *PN create the type column and the sort variable *;
     *PN----------------------------------------------*;
     type='text';
     sort = "&sort";

     keep domain varnum variable type length label mandatory sort;

     label domain='DOMAIN'
     ;
run;

data &dsn(rename=(temp11=var11  temp16=var16));
     retain domain var16 var8 var10 var11 var9 var18  ;
     length temp11 $3 temp16 $3;
     set &dsn(rename=(varnum   = var16
                      variable = var8
                      type     = var10
                      length   = var11
                      label    = var9
                      mandatory= var18))
     ;

     *PN-----------------------*;
     *PN make length and varnum *;
     *PN-----------------------*;
     temp11 = left(put(var11,4.0));
     temp16 = left(put(var16,2.0));

     drop var11 var16;

     label   var8    = 'VARIABLE'
             var9    = 'LABEL'
             var10   = 'TYPE'
             temp11  = 'LENGTH'
             temp16  = 'VARNUM'
             var18   = 'MANDATORY'
     ;
run;

*PN --------------------*;
*PN append the datasets *;
```

```
*PN---------------------*;
proc append base=final2 data=&dsn force;
run;

*PN------------------------------*;
*PN clean out the work directory *;
*PN------------------------------*;
proc datasets library=work;
     delete &dsn;
quit;

%mend;

*PN----------------------------------------*;
*PN import the supp contents from SDTM data*;
*PN----------------------------------------*;
%supp(dsn=SUPPAE,sort=B);
%supp(dsn=SUPPDM,sort=H);

proc print data=final2(obs=10);
run;

data final;
     length domain $6;
     set final final2;

     *PN----------------------------------------------------*;
     *PN need numeric to sort the variable numbers correctly*;
     *PN----------------------------------------------------*;
     sort_var = var16*1.0;
run;

proc sort data=final out=VARIABLE_METADATA(drop=sort sort_var);
     by sort sort_var ;
run;


*PN-------------------------------------------------------------*;
*PN----------------- export to a spread sheet -------------------*;
*PN-------------------------------------------------------------*;
%sas2xls(_inlib=work,
         _first=domain var16  ,
         _header=L,
         _files = DEFINE_HEADER_METADATA VARIABLE_METADATA ,
         _outfile = SDTM_METADATA);saz
```