Paper: 201-2013

A Practical Approach to Creating Define.XML by Using SDTM Specifications and Excel functions

Amos Shu, Endo Pharmaceuticals., Chadds Ford, PA

ABSTRACT

Define.xml (Case Report Tabulation Data Definition Specification) is a part of new drug submission required by the FDA. Clinical SAS® programmers usually use SAS programming [1, 2, 3, 4, 5] to generate the code of Define.xml as described in the CDISC Case Report Tabulation Data Definition Specification (define.xml) V1.0.0 [6]. This paper illustrates the process of using SDTM specifications and Excel functions to generate the code of Define.xml in an easy and straightforward way.

INTRODUCTION

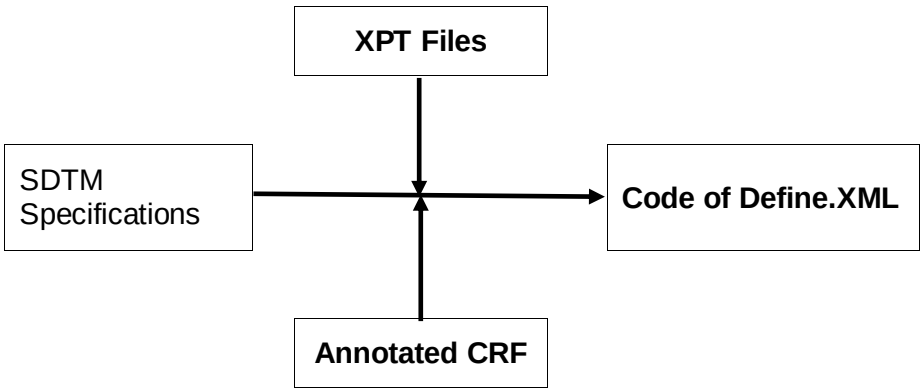
Define.xml (Case Report Tabulation Data Definition Specification) is a document that FDA required for drug submission. It describes the structure and contents of the data collected during the clinical trial process. Because Define.xml can increase the level of automation and improve the efficiency of the Regulatory Review process, FDA likes to have it with drug submission. The define.xml standard is based on the CDISC Operational Data Model (ODM), which is available at http://www.cdisc.org/standards/index.html.

To generate the code for Define.xml, there are three challenges [1] that average SAS programmers need to overcome:

1. Basic understanding of XML
2. Thorough understanding of the CDISC-specific XML structure of Define.xml
3. SAS expertise to generate the XML code

The first two challenges are fundamental; there are no alternatives or shortcuts to them. However, there are alternatives to the third one. Instead of SAS or XML tools, SDTM specifications and Microsoft Excel can be used to program Define.xml in a practical and efficient way.

PROCESS FLOW OF DEFINE.XML CODE GENERATION



STEP 1. XPT FILE GENERATION

Before generating the code for Define.xml, first transform the SDTM datasets into .xpt files. SAS XPORT engine is designed to do this type of job. Either DATA-SET step or PROC COPY can be used to do this [7]..

```
        LIBNAME  source 'SAS-data-library';
        LIBNAME  xportout xport 'transport-file';
        DATA xportout.xyz;
           SET source.xyz;
        RUN;
Or
        PROC COPY IN  = source
                  OUT = xportout memtype=data;
        RUN;
```

STEP 2. ANNOTATED CRF GENERATION

An annotated CRF is usually available in most clinical trials, which is prepared by the data management team for collecting clinical trial data. The issue is that many variable attributes are modified across all SDTM datasets based on the SDTM Specifications, which vary with the specific statistical analysis plan (SAP). Those changes need to be added to the annotated CRF for Define.xml.

STEP 3. USE SDTM SPECIFICATIONS TO GENERATE CODE OF DEFINE.XML

Define.xml has four sections in general: 1. Table of Contents (TOC, or Data Metadata), 2. Collection of Data Definition Tables (Variable Level Metadata), 3. Controlled Terminology, and 4. ODM XML Header, Study, and MetaDataVersion. The first two sections are the main part of Define.xml.

1. GENERATE THE TOC SECTION

The TOC lists all of the datasets (domains) included in the drug submission. It would be straightforward to create the following Excel sheet for TOC, based on the SDTM specifications and the SDTM IG[8].

Dataset	Description	Class	Structure	Purpose	Keys	Location
AE	Adverse Events Dataset	Events	One record per adverse event per subject	Tabulation	STUDYID, USUBJID, AEDECOD, AESTDTC	ae.xpt
CM	Concomitant Medications Dataset	Interventions	One record per recorded medication occurrence per subject	Tabulation	STUDYID, USUBJID, CMTRT, CMSTDTC	cm.xpt
…	…	…	…	…	…	…

The last column will generate a hyperlink with the XPT files created earlier. Based on this sheet, you can use an ODM (Operational Data Model) element – **ItemGroupDef** to generate XML code for the TOC section. The following is an example of the code for AE domain:

```
<ItemGroupDef  OID="AE"
               Name="AE"
               Repeating="Yes"
               IsReferenceData="No"
               Purpose=" Tabulation"
               def:Label="Adverse Events "
               def:Structure="One record per adverse event per subject"
               def:DomainKeys="STUDYID, USUBJID, AEDECOD, AESTDTC"
               def:Class="Events"
               def:ArchiveLocationID="LOCATION.AE">

               … …

               <def:leaf ID="LOCATION.AE"
                         xlink:href="ae.xpt">
                         <def:title>ae.xpt</def:title>
               </def:leaf>
 </ItemGroupDef >
```

The output of TOP looks like the following:

Dataset	Description	Structure	Purpose	Keys	Location
AE	Adverse Events	One record per adverse event per subject	Tabulation	STUDYID, USUBJID, AEDECOD, AESTDTC	ae.xpt

Two hyperlinks – Adverse Events and ae.xpt are created, which directly link to the corresponding variable level Metadata section and the xpt file of the specific domain, respectively.

2. GENERATE THE VARIABLE LEVEL METADATA SECTION

The ODM elements - **ItemRef** and **ItemDef** are used to create XML code for variable level Metadata section. Like the TOC section, it would not be difficult to use the SDTM specifications and follow the SDTM IG to create an Excel sheet like the following one for all domains:

Dataset Name	Dataset Label	Variable Number	Variable Name	Mandatory
AE	Adverse Events	1	STUDYID	Yes
AE	Adverse Events	2	DOMAIN	Yes
AE	Adverse Events	3	USUBJID	Yes
…	…	…	…	…

Depending on your submission preference, some optional items, such as 'Role', are not listed here. For detailed information, please refer to CDISC Define.xml on www.cdisc.org [6]. The 'Mandatory' column has a valid value of either 'Yes' or 'No', which indicates whether the clinical data for an instance of the containing item group is required or not.

The XML code for the first part of variable level metadata would look like this using **ItemRef** and Excel CONCATENATE function.

```
<ItemRef ItemOID="AE.STUDYID" OrderNumber ="1" Mandatory ="Yes" />
<ItemRef ItemOID="AE.DOMAIN"  OrderNumber ="2" Mandatory ="Yes" />
<ItemRef ItemOID="AE.USUBJID" OrderNumber ="3" Mandatory ="Yes" />
… …
```

The second Excel sheet looks like this:

Variable Name	Variable Label	Variable Type	Variable Length	Controlled Terms or Format	Origin	Comments	Computation
USUBJID	Unique Subject Identifier	text	18		Derived	STUDYID-SITEID-SUBJID	
SITEID	Study Site Identifier	text	3		CRF Page 1		
SEX	Sex	text	1	SEX	Derived		
AGE	Age	Integer	8		Derived		The number of (DEMODT-DOB)/365.25
BMICAT	Body Mass Index Category (kg/m^2)	text	15		Derived	BMI < 26, 26 => BMI <= 30, BMI > 30	
…	…	…	…	…	…	…	…

Most contents of this Excel sheet are from SDTM specifications. However, some special characters such as “&”, “>”, “<”, etc. must be treated specially following XML rules. Computational Algorithms, i. e. AGE = (DEMODT – DOB)/365.25, are usually included in the Computation column.

ItemDef and Excel functions such as CONCATENATE and IF are used to generate the XML code for the second part of variable level metadata:

```
<ItemDef OID="AE.USUBJID"
         Name="USUBJID"
         DataType ="text"
         Length ="18"
         Origin ="Derived"
         Comment =" STUDYID-SITEID-SUBJID "
         def:Label ="Unique Subject Identifier"
         def:DisplayFormat ="$18.">
</ItemDef>
```

If the variable has an item in the computation column, the ItemDef block should include the def:ComputationMethodOID element. For example, def:ComputationMethodOID ="AGE". Its corresponding computational algorithm will be coded as <def:ComputationMethod OID="AGE"> The number of (DEMODT - DOB)/365.25 </def:ComputationMethod>.

Controlled terminologies or Format will have the CodeListRef element, i.e. <CodeListRef CodeListOID="MedDRA"/>.

The output of variable level metadata looks like the following:

Variable	Label	Type	Controlled Terminology	Origin	Comment
STUDYID	Study Identifier	text		Derived	
DOMAIN	Domain Abbreviation	text	AE	Derived	
USUBJID	Unique Subject Identifier	text		Derived	STUDYID-SITEID-SUBJID
AESTDY	Study Day of Start of Adverse Event	Integer		Derived	See Computational Method: AESTDY
AETERM	Reported Term for the Adverse Event	text		CRF Page 53	
AEDECOD	Dictionary Derived Term	text	MedDRA	Derived	

Three types of hyperlinks will be generated by the ItemDef block. The first one is hyperlinks (i.e. MedDRA) to the Controlled terminology section, which will be discussed below. The second one is hyperlinks to the annotated CRF in the “Origin” column, which directly link to the corresponding pages on CRF where the value of the variable are collected (e.g. page 53). Hyperlinks to computational algorithms will be located in the “Comment” column following “See Computational Method:” e.g. See Computational Method: AESTDY.

3. CONTROLLED TERMINOLOGY (CODE LISTS) SECTION

This section is for variables that have a discrete list of valid values or controlled terms associated with them. For example, the route of dosing in the dataset of concomitant medications (CM) would have the following values:

Name	Description
IV	Intravenous
IM	Intramuscular
PO	Per oral

SC	Subcutaneous
PR	Per rectal
SL	Sublingual
INH	Inhaled
TOP	Topical
TD	Transdermal
TB	Transbuccal
Other	Other, specify
NMT	Non-Medicinal Therapy
…	…

The ODM **CodeList** element is used to generate XML code for the controlled terminology section. Below is the code for displaying the above route information.

```
<CodeList OID="CL.ROUTE" Name="ROUTE" DataType="text">
        <CodeListItem CodedValue="IV" def:Rank="1">
                <Decode>
                        <TranslatedText xml:lang="en">Intravenous</TranslatedText>
                </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="IM" def:Rank="2">
                <Decode>
                        <TranslatedText xml:lang="en">Intramuscular</TranslatedText>
                </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="PO" def:Rank="3">
                <Decode>
                        <TranslatedText xml:lang="en">Per oral</TranslatedText>
                </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="SC" def:Rank="4">
                <Decode>
                        <TranslatedText xml:lang="en">Subcutaneous</TranslatedText>
                </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="PR" def:Rank="5">
                <Decode>
                        <TranslatedText xml:lang="en">Per rectal</TranslatedText>
                </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="SL" def:Rank="6">
                <Decode>
                        <TranslatedText xml:lang="en">Sublingual</TranslatedText>
                </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="INH" def:Rank="7">
                <Decode>
                        <TranslatedText xml:lang="en">Inhaled</TranslatedText>
                </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="TOP" def:Rank="8">
                <Decode>
                        <TranslatedText xml:lang="en">Topical</TranslatedText>
                </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="TD" def:Rank="9">
                <Decode>
                        <TranslatedText xml:lang="en">Transdermal</TranslatedText>
                </Decode>
        </CodeListItem>
```

```
        <CodeListItem CodedValue="TB" def:Rank="10">
                <Decode>
                        <TranslatedText xml:lang="en">Transbuccal</TranslatedText>
                </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="Other" def:Rank="11">
                <Decode>
                        <TranslatedText xml:lang="en">Other,specify</TranslatedText>
                </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="NMT" def:Rank="12">
                <Decode>
                        <TranslatedText xml:lang="en">Non-Medicinal Therapy</TranslatedText>
                </Decode>
        </CodeListItem>
</CodeList>
```

The output for this section is as follows:

ROUTE, Reference Name (CL.ROUTE)	
Code Value	**Code Text**
IV	Intravenous
IM	Intramuscular
PO	Per oral
SC	Subcutaneous
PR	Per rectal
SL	Sublingual
INH	Inhaled
TOP	Topical
TD	Transdermal
TB	Transbuccal
Other	Other,specify
NMT	Non-Medicinal Therapy

4. ODM XML HEADER, STUDY, and METADATAVERSION SECTION

The document of Define.xml[6] has good examples for creating this section. It can be easily completed by following the instructions.

CONCLUSION

The process of using SDTM specifications and Excel functions to generate the code of Defiune.xml is an easy, straightforward, and time-saving alternative with no additional cost. Average SAS programmers should not need extensive training to complete this task.

REFERENCES

[1]. Molter, Michael, *A SAS® Programmer's Guide to Generating Define.xml*, SAS Global Forum 2009
[2]. Kawohl, Monika, *A SAS based solution for define.xml*, PharmaSUG, 2007
[3]. Adams, John, etc., *Creating a define.xml file for ADaM and SDTM*, PharmaSUG, 2011
[4]. Banga, Rohit, Generate Define.xml & Define.pdf from Metadata Environment, PharmaSUG, 2009
[5]. Becker, Matt, etc., *SDTM, ADaM and define.xml with OpenCDISC*, PharmaSUG, 2011
[6]. CDISC Define.xml, V 1.0.0 (http://www.cdisc.org)

[7]. http://support.sas.com/documentation/cdl/en/movefile/59598/HTML/default/viewer.htm#creattrans.htm
[8]. http://www.cdisc.org/SDTM

CONTACT INFORMATION

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

Amos Shu
Endo Pharmaceuticals
1400 Atwater Dr.
Malvern, PA 19355
Email: shu.amos@endo.com

TRADEMARK INFORMATION