PhUSE 2012

Paper PP04

Define.xml content validation – SAS® based solution

Sam Tomioka, Sunovion Pharmaceuticals Inc, Fort Lee, USA
Steven Huang, Sunovion Pharmaceuticals Inc, Fort Lee, USA

ABSTRACT

Due to lack of commercially available tools for the validation of the contents of the *define.xml*, and the increasing submission activities, we at Sunovion had developed our in-house programs to address the common issues/mistakes of the define files. This article will illustrate the systematic approach of using SAS® programs to validate the contents of the define.xml. We will demonstrate how the source data are collected; explain the process of validation and present the most common findings from the validation process.

INTRODUCTION

Define.xml is a data definition file, formally known as CRT-DD, is used to facilitate the review of the study data submitted to the regulatory agency. A well prepared standardized metadata can minimize the time required for the reviewers to be familiarized with the data, which can speed up the overall review process.

In December 2011, the FDA published "CDER Common Data Standards Issues Document" to address the commonly observed issues in the standardized data submission. The document stated that "sponsors should make certain that every data variable's codelist, origin, and derivation is clearly and easily accessible from the define file". Furthermore, the FDA listed "Define Doesn't Validate" as one of the common errors that they have observed.

There are some validation tools which check for conformance to the XML schema and ODM specifications that address the issue of "Define Doesn't Validate". However these tools do not validate the contents of the define file, therefore, we have developed a tool to address the accuracy, consistency, and completeness of the define.xml, by using SAS programs to validate the contents of the define file such as dataset level metadata, variable level metadata including the Origin, value level metadata, and the controlled terminology.

VALIDATION TOOLS

We have evaluated two define file validation tools that were available to us in terms of XML conformance checks and contents validation.

SAS CLINICAL STANDARD TOOLKIT

SAS Clinical Standard Toolkit® Ver 1.4 (SAS/CST) validates the structure and syntax of the define.xml file against the XML schema i.e. define1-0-0.xsd for the CRT-DDS standard. We found that this tool did not directly check the contents of define.xml against the actual submission datasets, but rather checked against the CST work datasets, which were generated when define.xml was produced by SAS/CST or SAS Clinical Data Integration® (SAS/CDI). It checked to ensure that the referenced item actually exists, required variables are not missing or empty for an observation, and the character data conforms to a particular format etc.

OPENCDISC

OpenCDISC® uses default CDISC Define.xml 1.0 Validation Rules which performs similar validation checks as SAS/CST. This tool is very intuitive tool to use. The tool needs input define.xml file but it does not use actual submission datasets and the annotated CRF for validation. It has few additional enhanced checks that were not in SAS/CST. For example, we learned that it checks for ODM extensions which are not documented in the CRT-DDS specifications ver 1.0.0. It checks consistency of data types defined for variables, values, and codelist name. This tool was able to identify few errors that were not identified by SAS/CST.

Since both tools did not check the contents of define.xml against the actual submission datasets (XPT) and the annotated CRF, we have developed a tool to supplement the checks by SAS/CST and OpenCDISC®.

OVERALL FLOW OF SUNOVION'S DEFINE VALIDATOR

For usability and maintainability, our tool, **define_valid.sas** consists of several SAS macros and XML map files. Table1 illustrates the macros developed for our tool and Table 2 describes the purpose of the XML map files.

These macros and XML map files are used to:
1) Extract source data from define.xml, annotated CRFs, and submission datasets;
2) Execute series of define content checks using the extracted source metadata and our standard metadata specifications (DSP-SDTM); and
3) Compile the findings in user friendly report in Excel format.

Table 1: List of macros and purpose used in our define_valid tool

Macros	Purpose
%xpt2sas	Extract SAS transport files.
%extrdef	Uses defxmlmap.map and libname XML engine to extract define.xml into SAS datasets.
%metads	Extract dataset metadata from all SAS transport files.
%metatab	Extract variable metadata from all SAS transport files.
%maxvarlen	Obtain the maximum length for variables and value levels from all SAS transport files.
%conttrm1	Compile the controlled terminologies from all SAS transport files.
%extranno	Extract annotations and page information from blankcrf.pdf. This macro is used when validating define.xml for SDTM
%getvlm	Generate a value level data from all SAS datasets.
%gtvlmtyp	Define the data type from the actual values in SAS datasets for the value level data.

Table 2: Purpose of SAS XML map file

XML map file	Purpose
defxmlmap.map	Used in %extrdef to extract metadata from define.xml.
annoxml.map	Used in % extranno to extract annotations from the XML Forms Data Format file exported from blankcrf.pdf.

SOURCE CONTENT EXTRACTION

DEFINE.XML

The contents within the define.xml come from the submission datasets, our global standards, and, SDTM annotated CRF. In our validation, the contents of the define file are extracted and stored in SAS datasets.

An XML map file, **defxmlmap.map**, was created using SAS XML Mapper® to provide instruction to the SAS Libname Engine on how to extract the metadata within the define.xml into SAS datasets. Below is the sample from **defxmlmap.map** file for creating metadata SAS dataset.

An element **'TABLE'** (line 1) is used to define the name of the resultant dataset. Next key element is **'TABLE-PATH'** (line 3) which defines the path (lines 21, 23, 25) within define.xml for the collection of records with a defined set of columns to be collected for the SAS dataset. **'COLUMN'** (lines 5 and 11) is an element that defines variable attributes. For instance, **COLUMN name** (lines 5 and 11) defines a variable name as def_oid, **'TYPE'** (line 8) defines a SAS data type as character, and **'LENGTH'** (line 10) defines a SAS variable length as 1000 character length. The length is defined as 1000 to avoid string truncation in the extracted SAS datasets. **'DATATYPE'** (line 9) defines the type of incoming data. **'PATH'** (line 6) is an element that specifies a location of the tag, which is OID in the example (line 21, 23, 25), within define.xml for the variable def_oid.

XML map file (Sample from defxmlmap.map)

```
1  <TABLE name="metadata">
2  <TABLE-DESCRIPTION>Metadata of variables</TABLE-DESCRIPTION>
3  <TABLE-PATH syntax="XPath">/ODM/Study/MetaDataVersion/ItemDef</TABLE-PATH>
4
5  <COLUMN name="def_oid">
6     <PATH syntax="XPath">/ODM/Study/MetaDataVersion/ItemDef/@OID</PATH>
7     <DESCRIPTION>Define OID</DESCRIPTION>
8     <TYPE>character</TYPE>
```

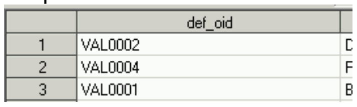
```
   <DATATYPE>string</DATATYPE>
   <LENGTH>1000</LENGTH>
</COLUMN>
```

Define.xml

```
     <ODM xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:def="http://www.cdisc.org/ns/def/v1.0"
xmlns="http://www.cdisc.org/ns/odm/v1.2" FileOID="1" CreationDateTime="2012-07-19T14:21:11-04:00"
AsOfDateTime="2012-07-19T14:20:17" Description="define.xml" FileType="Snapshot" Id="define.xml"
ODMVersion="1.0">
     <Study OID="1">
     …
     <MetaDataVersion OID="1" Name="CDISC-SDTM 3.1.2" Description="CDISC-SDTM 3.1.2"
def:DefineVersion="1.0.0" def:StandardName="CDISC SDTM" def:StandardVersion="3.1.2">
     …
     <ItemDef OID="VAL0002" Name="DISPAMT" DataType="integer" Length="8"
SASFieldName="DISPAMT" Origin="CRF Page 68" def:Label="Dispensed Amount"/>
     <ItemDef OID="VAL0004" Name="RETAMT" DataType="integer" Length="8"
SASFieldName="RETAMT" Origin="CRF Page 68" def:Label="Returned Amount"/>
     <ItemDef OID="VAL0001" Name="BLISTRET" DataType="text" Length="8"
SASFieldName="BLISTRET" Origin="CRF Page 68" def:Label="Was Blister Card Returned?"/>
```

Using this XML map file with SAS XML LIBNAME engine produces a SAS dataset called **metadata** shown below.

Output metadata SAS dataset



	def_oid	
1	VAL0002	D
2	VAL0004	F
3	VAL0001	B

This concept was used to extract metadata from define.xml into following five SAS datasets.

Table 3: List of SAS datasets from defxmlmap.map

SAS dataset	Description
metadata	This dataset contains variable metadata
table_meta	This dataset contains dataset metadata
var_ref	This dataset contains additional variable metadata and variable references
codelist	This dataset contains value level and controlled terminology codelist data including coded values and translated values
annot	This dataset contains origin values such as CRF page number, derived, assigned etc.

The extraction of define.xml using **defxmlmap.map** file in SAS XML Libname Engine was compiled as a macro called ***%extrdef***.

SDTM ANNOTATED CRF:

Annotations and form data within a SDTM annotated CRF i.e. blankcrf.pdf was extracted as XML Forms Data Format (XFDF) using Adobe Acrobat 9 Pro ® version. The content of XFDF file can be extracted into SAS datasets using the same methodology to extract define.xml.

Below is a sample from the XFDF file extracted from the blankcrf.pdf. This sample contains one annotation “SV.SVSTDTC” on page 3 of blankcrf.pdf. The text of the annotation is contained in the **‘contents-richtext’** element (lines 5 and 11) which is the second child of the **‘annots’** element (line 1) followed by **‘freetext’** element (lines 2 and 14) in the example below. The actual text are enclosed by **<p></p>** (line 9), but when style is defined for the annotation text, the text is enclosed by **<span></span>** within **‘p’** tag. As a result, two different tags **‘p’** and **‘span’** had to be taken into consideration in extraction of actual annotations when XML map file was developed.

The page number of the form is contained in the **‘page’** tag (line 3) within the **‘freetext’** element (lines 2 and 14). However, the numbering starts from zero instead of one in the XFDF file generated from Adobe Acrobat 9 Pro version. The sample below indicates the page number is ‘2” but it is page 3 on blankcrf.pdf.

XFDF file

```
1   <annots>
2     <freetext width="1.5" color="#FFFFFF" creationdate="D:00000000000000Z" flags="print"
3      date="D:20101206102547-05'00'" name="e2085b08-e2f8-4c01-a2b7-f86230ec3743" page="2"
4      justification="centered" rect="164.093994,565.987976,223.391006,580.072998" title="T">
5       <contents-richtext>
6         <body xmlns="http://www.w3.org/1999/xhtml" xmlns:xfa=http://www.xfa.org/schema/xfa-data/1.0/
7          xfa:APIVersion="Acrobat:9.2.0" xfa:spec="2.0.2" style="font-size:8.0pt;text-align:left;color:#0000FF;font-
8          weight:normal;font-style:normal;font-family:Arial;font-stretch:normal">
9             <p dir="ltr">SV.SVSTDTC</p>
10        </body>
11      </contents-richtext>
12      <defaultappearance>0 0 1 rg /ArialMT 8 Tf</defaultappearance>
13      <defaultstyle>font: Arial 8.0pt; text-align:left; color:#0000FF </defaultstyle>
14    </freetext>
```

In the XML map file, two columns were generated for annotations to accommodate the two different tags used in the XFDF file as discussed earlier. The first one is the column annotation1 (line 17) which obtains values from tag **‘p’** and the second one is annotation2 (line 23) which obtains values from tag **‘span’** as shown below. These parts were written in the editor and pasted to the draft XML map file created by SAS XML Mapper®.

XML map file (Sample from annoxml.map)

```
15  <TABLE name="annotation">
16        <TABLE-PATH syntax="XPath">/xfdf/annots/freetext/contents-richtext</TABLE-PATH>

17        <COLUMN name="annotation1">
18          <PATH syntax="XPath">/xfdf/annots/freetext/contents-richtext/body/p</PATH>
19          <TYPE>character</TYPE>
20          <DATATYPE>string</DATATYPE>
21          <LENGTH>1000</LENGTH>
22        </COLUMN>
23        <COLUMN name="annotation2">
24          <PATH syntax="XPath">/xfdf/annots/freetext/contents-richtext/body/p/span</PATH>
25          <TYPE>character</TYPE>
26          <DATATYPE>string</DATATYPE>
27          <LENGTH>1000</LENGTH>
28        </COLUMN>
29        <COLUMN name="page" retain="YES">
30          <PATH syntax="XPath">/xfdf/annots/freetext/@page</PATH>
31          <TYPE>numeric</TYPE>
32          <DATATYPE>integer</DATATYPE>
33        </COLUMN>
```

The above XML map file, ***annoxml.map***, generates a dataset called **‘annotation’** based on the **TABLE** element (line 15). This dataset consists of three variables: ‘annotation1’ (line 17), ‘annotation2’ (line 23), and ‘page’ (line 29). The page numbers obtained from the XFDF file were incremented by one to address the page number issue discussed above.

With Sunovion’s rules for annotating SDTM variables, controlled terminology, and conditional annotations on the blankcrf.pdf, two key variables domain and variable name were derived from annotations1 and annotations2 variables. These two derived variables were used as key variables when pages in the XFDF file and the Origin column in define.xml were compared.

The extraction of annotations from the XFDF file using **annoxml.map** file in SAS XML Libname Engine and restructuring of the extracted SAS dataset, **‘annotation’** was compiled as a macro called ***%extranno***.

SAS TRANSPORT FILES:

First, **%xpt2sas** macro was used to convert submission datasets in SAS transport file format into SAS datasets. Then the DICTIONARY SAS data views were used as a source for the dataset metadata and variable metadata from all SAS datasets.

Second, DOMAIN dataset which contains dataset metadata from SAS datasets was generated using **%metads** macro.

Third, **%metatab** macro was used to create COLUMN dataset which contains variable metadata.
The forth step was to generate value level metadata from the SAS datasets. In this step, **%getvlm** macro was used to generate value level metadata for –TESTCD variables and QNAM variables. The output from this macro contains the values and related values for each variable specified in the macro. Another macro used in this step was **%gtvlmtyp** macro. It was used to identify the data type since the data type may be different between parent variable level and value levels. For instance, the data type of VSORRES may be float, but the data type can be integer when VSTESTCD='SYSBP'. Similarly, macro **%maxvarlen** was used to derive the length of variables as well as the length at value levels.

Last step was to obtain the controlled terminology used in the SAS datasets. Our standard metadata specifications (DSP-SDTM) define the variables that use controlled terminology. **%conttrm1** macro uses DSP-SDTM to generate a list of variables that use controlled terminology then extract controlled terminology used in each SAS dataset.

VALIDATION AND VERIFICATION PROCESS:

Our content validation checks can be categorized as follows, 1) Dataset Metadata, 2) Variable Metadata, 3) Controlled Terminology and 3) Value Level Metadata. All of these checks are included in **define_valid.sas**.

VALIDATION OF DATASET METADATA

Standard dataset metadata from Sunovion's defined standard (DSP-SDTM) was compared against the 'table_meta' dataset from **%extrdef** macro for the following items.

a) Domain Purpose (standard vs define.xml)
b) Domain Structure (standard vs define.xml)
c) Reference Data (standard vs define.xml)
d) Dataset Descriptions (standard vs define.xml)
e) Repeating Data (standard vs define.xml)
f) Class Data (standard vs define.xml)

Furthermore, the '**domain'** dataset from **%metads** was compared against the '**metadata'** dataset from **%extrdef** for dataset descriptions.

g) Dataset Descriptions (submission datasets vs define.xml)

VALIDATION OF VARIABLE METADATA

Datasets '**metadata'**, '**var_ref**', and '**annot'** from **%extrdef** macro were compared against the outputs from **%metatab** and **%extranno** macros for the following items.

a) Data Type (define.xml vs submission datasets)
b) Variable Label (define.xml vs submission datasets)
c) Consistency of variables (define.xml vs submission datasets) - Identifies variables missing in define.xml or XPT
d) Variable Length (define.xml vs submission datasets) - Checks for maximum length defined in Define.xml with actual maximum length in XPT
e) Variable Order (define.xml vs submission datasets)
f) Origin (define.xml vs annotated CRF)

VALIDATION OF CONTROLLED TERMINOLOGY

An output from **%conttrm1** macro was compared against '**codelist'** dataset from **%extrdef** macro for the following items.

a) Checks for duplication (define.xml)
b) Consistency checks (define.xml vs submission datasets)
c) Missing controlled terminologies in define.xml or datasets

Additionally, '**codelist'** dataset was checked against our standard Value Level metadata to check for missing codelist from Value Level metadata.

d) Missing codelist (standard vs define.xml)

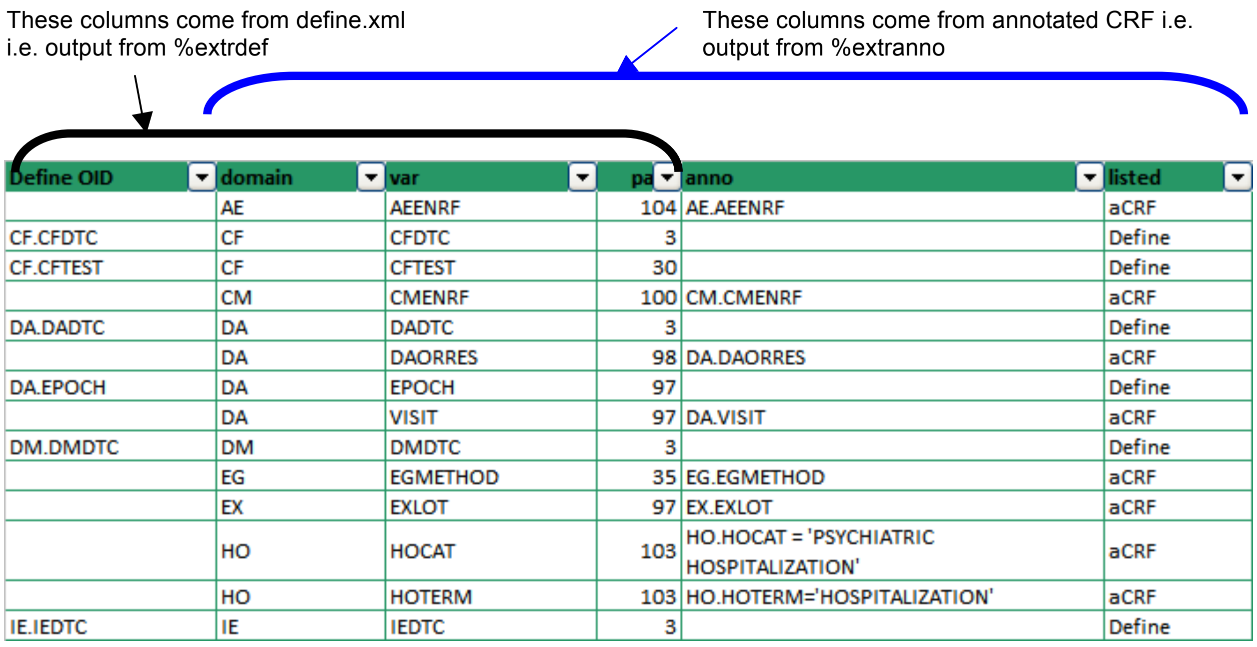
VALIDATION OF VALUE LEVEL METADATA

An output from %conttrm1 was compared against 'codelist' dataset from %getvlm for the following items.

a) Check for incorrect value level metadata (submission dataset vs define.xml)
b) Check for consistency (submission dataset vs define.xml)
c) Missing Value Level Metadata in define.xml
d) Identify Value Level data that does not exist in the submission dataset
e) Data Type (submission dataset vs define.xml)
f) Variable Label (define.xml vs submission datasets)

REPORTING OF VALIDATION RESULTS:

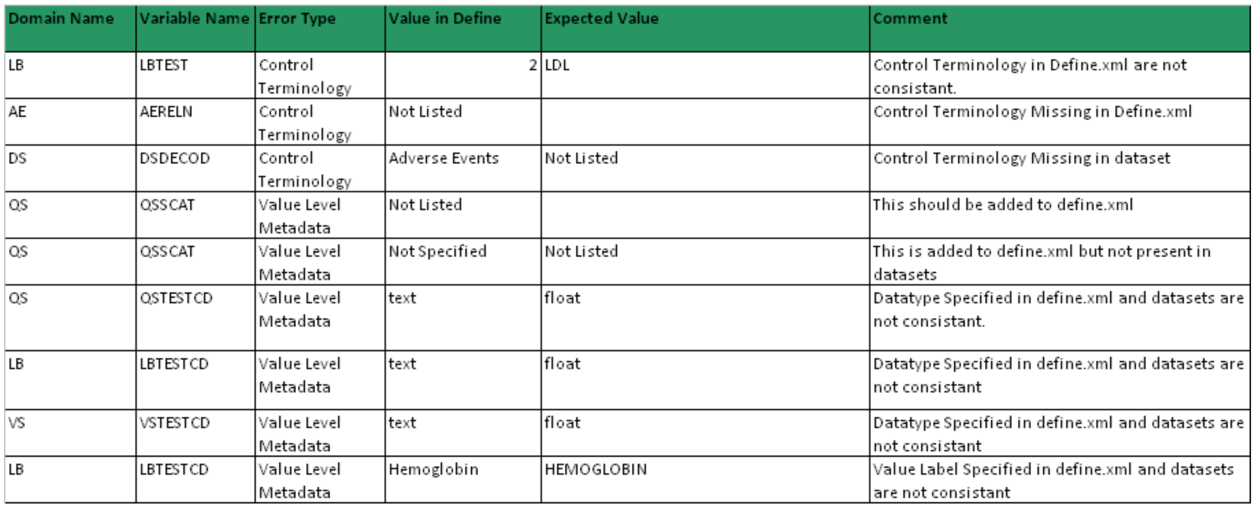
Results from series of checks discussed above were compiled into two parts. The first one is the report of discrepancies between annotated CRF and define.xml as shown below.


These columns come from define.xml i.e. output from %extrdef
These columns come from annotated CRF i.e. output from %extranno

Define OID	domain	var	pa	anno	listed
	AE	AEENRF	104	AE.AEENRF	aCRF
CF.CFDTC	CF	CFDTC	3		Define
CF.CFTEST	CF	CFTEST	30		Define
	CM	CMENRF	100	CM.CMENRF	aCRF
DA.DADTC	DA	DADTC	3		Define
	DA	DAORRES	98	DA.DAORRES	aCRF
DA.EPOCH	DA	EPOCH	97		Define
	DA	VISIT	97	DA.VISIT	aCRF
DM.DMDTC	DM	DMDTC	3		Define
	EG	EGMETHOD	35	EG.EGMETHOD	aCRF
	EX	EXLOT	97	EX.EXLOT	aCRF
	HO	HOCAT	103	HO.HOCAT = 'PSYCHIATRIC HOSPITALIZATION'	aCRF
	HO	HOTERM	103	HO.HOTERM='HOSPITALIZATION'	aCRF
IE.IEDTC	IE	IEDTC	3		Define

For instance, the first row indicates that a variable AEENRF is annotated on page 104 of blankcrf.pdf, however, the define OID is missing, therefore the Origin of variable in define.xml does not indicate that this variable came from page 104. The second row shows that define.xml defines the variable CFDTC is annotated on page 3 of blankcrf.pdf, however, page 3 of the blankcrf.pdf is missing the annotation.

The second report includes all other findings from other checks included in **define_valid.sas**.. The report shows the identifiers, error type, expected value, and description of error as shown below.



Domain Name	Variable Name	Error Type	Value in Define	Expected Value	Comment
LB	LBTEST	Control Terminology	2	LDL	Control Terminology in Define.xml are not consistant.
AE	AERELN	Control Terminology	Not Listed		Control Terminology Missing in Define.xml
DS	DSDECOD	Control Terminology	Adverse Events	Not Listed	Control Terminology Missing in dataset
QS	QSSCAT	Value Level Metadata	Not Listed		This should be added to define.xml
QS	QSSCAT	Value Level Metadata	Not Specified	Not Listed	This is added to define.xml but not present in datasets
QS	QSTESTCD	Value Level Metadata	text	float	Datatype Specified in define.xml and datasets are not consistant.
LB	LBTESTCD	Value Level Metadata	text	float	Datatype Specified in define.xml and datasets are not consistant
VS	VSTESTCD	Value Level Metadata	text	float	Datatype Specified in define.xml and datasets are not consistant
LB	LBTESTCD	Value Level Metadata	Hemoglobin	HEMOGLOBIN	Value Label Specified in define.xml and datasets are not consistant

CONCLUSION

It is important to ensure the accuracy, consistency, and completeness of the define.xml prior to the submission. However it is a challenging process to check the define.xml against the source metadata or submission datasets with existing tools. With the use of our validation tool most of the tedious and time consuming checks such as the dataset level metadata, variable level metadata, value level metadata, and the controlled terminology in the define.xml against the actual submission datasets can now be accomplished quickly and precisely. Programmers with solid SAS and submission knowledge can easily understand and utilize the tool. Furthermore our simplified output format can clearly point out the discrepancy which will enable issues to be easily identified.

REFERENCES

CDER Common Data Standards Issues Document Version 1.1/December 2011
http://www.fda.gov/downloads/Drugs/DevelopmentApprovalProcess/FormsSubmissionRequirements/ElectronicSubmissions/UCM254113.pdf

Case Report Tabulation Data Definition Specification Final Version 1.0
http://www.cdisc.org/stuff/contentmgr/files/0/464923b10ea16b477151fcaa9f465166/misc/crt_ddspecification1_0_0.pdf
XML Schema Validation for Define.xml White Paper
http://www.cdisc.org/stuff/contentmgr/files/0/464923b10ea16b477151fcaa9f465166/misc/definereport_v1_0.pdf

SAS Clinical Standards Toolkit 1.4: User's Guide
http://support.sas.com/documentation/cdl/en/clinstdtktug/64439/PDF/default/clinstdtktug.pdf

SAS(R) 9.2 XML LIBNAME Engine: User's Guide, Second Edition
http://support.sas.com/documentation/cdl/en/engxml/62845/PDF/default/engxml.pdf

XML Forms Data Format Specification, version 3 August 2009
http://partners.adobe.com/public/developer/en/xml/XFDF_Spec_3.0.pdf

ACKNOWLEDGMENTS

We would like to thank our consultant Shiva Katepalli for development support.

CONTACT INFORMATION

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

Sam Tomioka
Associate Director, Programming, Data Science
Sunovion Pharmaceuticals Inc.
1 Bridge Plaza
Fort Lee, NJ
201-228-8039
Sam.tomioka@sunovion.com

Steven Huang
Associate Director, Programming, Data Science
Sunovion Pharmaceuticals Inc.
1 Bridge Plaza
Fort Lee, NJ
201-228-8039
Steven.Huang @sunovion.com