PhUSE 2013

Paper CD05

Define-XML v2 – What's New

Lex Jansen, SAS Institute Inc., Cary, NC, USA

ABSTRACT

In March 2013 the final version of the Define-XML 2.0.0 standard [1], formerly known as CRT-DDS (Case Report Tabulation Data Definition Specification) or "define.xml", as most people called it, was released by the CDISC XML Technologies team. Define-XML 2.0.0 is a major revision of the Define-XML standard for transmission of SDTM, SEND and ADaM metadata. Version 1.0.0 was released for implementation in February 2005 [2]. Define-XML has been a useful mechanism and critical component for providing Case Report Tabulation Data Definition Specifications (CRT-DDS) in an XML format for CDISC based electronic submissions to a regulatory authority such as the U.S. Food and Drug Administration (FDA). The Define-XML specification has been greatly improved with an increased clarity and reduced ambiguity. This paper gives an overview of changes between the 2 versions and the new features of Define-XML 2.0.0.

Keywords: CDISC, CRT-DDS, Define-XML, define.xml, metadata

INTRODUCTION

This paper describes an updated Define-XML version 2.0.0 model that is used to describe CDISC SDTM (Study Data Tabulation Model), SEND (Standard for Exchange of Nonclinical Data) and ADaM (Analysis Data Model) datasets for the purpose of submissions to the FDA, as well as any proprietary (non-CDISC) dataset structure. Define-XML version 2.0.0 can be used to transmit metadata for the following CDISC standards:

- SDTM Implementation Guide Versions 3.1.2 and higher
- ADaM Implementation Guide Versions 1.0 and higher
- SEND Implementation Guide Versions 3.0 and higher

Note: The official name for Define-XML version 1.0.0 is CRT-DDS (Case Report Tabulation Data Definition Specification) version 1.0.0. In this paper we will refer to this version as Define-XML version 1.

This paper assumes that the reader is familiar with some basic XML concepts, and also with Define-XML version 1.0.0. Reference [3] contains both a short overview of the XML needed to understand this paper, and also an overview of the structure of a define.xml file based on Define-XML version 1.0.0.

When submitting clinical study data in electronic format to the FDA, or other regulatory agencies, not only information from trials has to be submitted, but also information to help understand the data. Part of this information is a data definition file, which is the metadata describing the format and content of the submitted data sets. When submitting data in CDISC format it is required to submit the data definition file in the Case Report Tabulation Data Definition Specification format (define.xml) as prepared by the CDISC define.xml team [4]. In August 2013 the FDA has also started accepting Define-XML version 2.0 [5].

A define.xml file, or data definition specification, provides different kinds of metadata for:

- Datasets:
  - Name, label, class, structure, purpose, keys, location, comments, documentation
- Variables:
  - Name, label, type, length, codelist, origin, derivations, comments
- Variables under a condition:
  - Value level metadata or parameter value level metadata
- Controlled Terminology:
  - Standard or sponsor defined, valid values, decodes
- Derivations or Algorithms
- Links to submission files:
  - Annotated CRF, Reviewers' Guide, data sets,

OPERATIONAL DATA MODEL

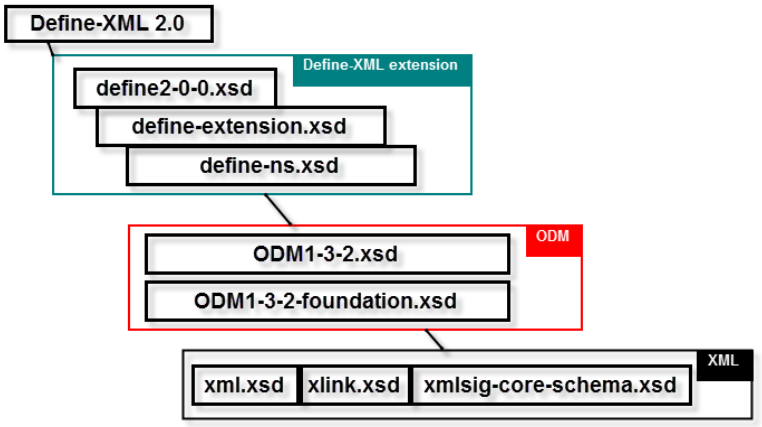
The Define-XML standard is based on the CDISC Operational Data Model (ODM) XML schema [6]. The new Define-XML version 2 takes full advantage of the latest ODM version 1.3.2.

The CDISC Operational Data Model is a vendor neutral, platform independent XML format for interchange and archival of clinical study data. The model represents study metadata, administrative metadata, reference data and subject data associated with a clinical trial. The ODM format is defined by an XML schema and a specification.

One of the features of the ODM is a standardized mechanism for defining schema extensions to provide functionality needed to support interchange requirements for specialized use cases.

To address the specific needs of data transmission in support of regulatory submissions, CDISC has developed the Define-XML model, which is implemented as an extension to the ODM foundation schema. These extensions follow the guidelines for Vendor Extensions provided in the ODM specification and comply with the W3C XML Schema 1.0 specification. The XML schema files for the Define-XML standard are available online [4]. Figure 1 depicts the extension mechanism.

Figure 1 Define-XML v2 as an ODM 1.3.2 extension



WHY A NEW VERSION?

Although Define-XML version 1 has been a successful standard since 2005, there were good reasons to publish a new version.

Some of the early goals for Define-XML version 2 were:

- Put Define-XML on an ODM 1.3 platform, as to better enable support for CDISC standards like ADaM, and not just Case Report Tabulations (i.e. SDTM and SEND). ODM 1.3 introduced better support, in a machine readable way, for Controlled Terminology (e.g. EnumeratedItems), derivations, formal expressions (Methods) and comments.
- Improve the clarity of the specification by providing more examples and less ambiguity.
- More explicit support for document references.
- Improving the support for machine readable value level metadata or parameter level metadata. Although in Define-XML version 1 there was already support for value level metadata, it was incomplete and not explicit.
- Provide more metadata for Controlled Terminology enabling the designation of a codelist as CDISC Controlled Terminology.
- Implement best practices by SDTM and ADaM metadata experts.

The next sections will show more detail of the new Define-XML version 2 features.

DEPRECATED COMPONENTS

Some components of Define-XML version 1 have been deprecated to enable the utilization of existing ODM components. A list of the deprecated elements and attributes is provided in table 1.

Table 1 Deprecated components

Element(s)	Component	Comment
MetaDataVersion	def:ComputationalMethod	**Replaced by ODM 1.3 MetaDataVersion/MethodDef element.**
ItemGroupDef	def:Label	**Replaced by ODM 1.3 ItemGroupDef/Description element.**
ItemGroupDef	def:DomainKeys	**Replaced by ODM 1.3 ItemGroupDef/ItemRef/@KeySequence attribute.**
ItemDef	def:Label	**Replaced by ODM 1.3 ItemDef/Description element.**
ItemDef	def:ComputationMethodOID	**Replaced by ODM 1.3 ItemRef/@MethodOID attribute**
ItemDef	Origin	**Replaced by new ItemDef/def:Origin element.**
ItemDef	Comment	**Replaced by ItemDef/@def:CommentOID attribute and MetaDataVersion/def:CommentDef element**
CodeListItem	**def:Rank**	**Replaced by ODM 1.3 attribute Rank.**

Example 1 shows some of the changes mentioned in Table 1, specifically in a data set definition. Notice the Method reference on the ItemRef element in Define-XML version 2. In Define-XML version 1 Computational Methods were referenced from ItemDef elements.

Example 1 Deprecated components in a data set definition

```
<!-- Dataset Definition (DM)  (Define-XML version 1) -->
<ItemGroupDef OID="IG.DM" Name="DM" Repeating="No" IsReferenceData="No" Purpose="Tabulation"
  def:Label="Demographics" def:Structure="One record per event per subject"
  def:DomainKeys="STUDYID, USUBJID" def:Class="Special Purpose" def:ArchiveLocationID="LF.DM">
  <ItemRef ItemOID="STUDYID" OrderNumber="1" Mandatory="Yes" Role="Identifier"/>
  <ItemRef ItemOID="DOMAIN" OrderNumber="2" Mandatory="Yes" Role="Identifier"/>
  <ItemRef ItemOID="USUBJID" OrderNumber="3" Mandatory="Yes" Role="Identifier"/>
  <ItemRef ItemOID="SUBJID" OrderNumber="4" Mandatory="Yes" Role="Topic"/>


<!-- Dataset Definition (DM)  (Define-XML version 2) -->
<ItemGroupDef OID="IG.DM" Domain="DM" Name="DM" Repeating="No" IsReferenceData="No"
  SASDatasetName="DM" Purpose="Tabulation" def:Structure="One record per subject"
  def:Class="SPECIAL PURPOSE" def:CommentOID="COM.DOMAIN.DM" def:ArchiveLocationID="LF.DM">
  <Description>
    <TranslatedText xml:lang="en">Demographics</TranslatedText>
  </Description>
  <ItemRef ItemOID="IT.STUDYID" OrderNumber="1" Mandatory="Yes" KeySequence="1"/>
  <ItemRef ItemOID="IT.DM.DOMAIN" OrderNumber="2" Mandatory="Yes"/>
  <ItemRef ItemOID="IT.USUBJID" OrderNumber="3" Mandatory="Yes" KeySequence="2"
    MethodOID="MT.USUBJID"/>
  <ItemRef ItemOID="IT.DM.SUBJID" OrderNumber="4" Mandatory="Yes"/>
```

LINKING TO SUPPORTING DOCUMENTS

Define-XML version 2.0.0 adds support to explicitly link to external documents. These can be links to the Annotated Case Report Form, Supplemental Documents or other supporting documents, such as Data Guides. Page numbers, page ranges or named destinations within the PDF document can be specified in a machine readable way. Parsing of strings like “CRF Pages 23, 24 and 25” will no longer be needed!

These document links can be attached to various metadata elements:

- Data sets, as a reference in a comment
- Data set variables, as a reference in a comment or related to the origin (Annotated Case Report Form)
- Methods or derivations can include references to documents.

In Define-XML version 1 links to a document would be defined like the following example (example 2):

Example 2 **Document references in Define-XML version 1**
(or actually, the ***lack*** thereof ...)

```
<def:AnnotatedCRF>
  <def:DocumentRef leafID="blankcrf"/>
</def:AnnotatedCRF>
<def:SupplementalDoc>
  <def:DocumentRef leafID="ReviewersGuide"/>
</def:SupplementalDoc>

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


<ItemDef ... Origin="CRF Page 6, 7, 8"  ... />

<ItemDef OID="IT.DM.ARM" ...
   Comment="Derived from arm code in TRIAL ARM (TA) based on Randomization Number.
   See Note 2.1 in the Reviewers Guide" ... />
```

When viewing the define.xml file in a browser, the supporting XSL stylesheet was then expected to create links to the correct pages (6, 7 and 8) in the Annotated Case Report Form (blankcrf.pdf) or to a named destination (2.1) in a document (reviewersguide.pdf). Whether this would work would depend on undocumented and implicit conventions and a lot of programming in the XSL stylesheet. There is no machine readable connection between the source (Origin and Comment) and the target of the link.

In Define-XML version 2 this is much more explicit, as the following example shows:

Example 3 **Explicit document references in Define-XML version 2**

```
<ItemDef
  ...
  <def:Origin Type="CRF">
    <def:DocumentRef leafID="LF.blankcrf">
      <def:PDFPageRef PageRefs="6 7 8" Type="PhysicalRef"/>
    </def:DocumentRef>
  </def:Origin>
</ItemDef>

<ItemDef OID="IT.DM.ARM" ... CommentOID="COM.ARM"
  ...
</ItemDef>


<def:CommentDef OID="COM.ARM">
  <Description>
    <TranslatedText xml:lang="en">Derived from arm code in TRIAL ARM (TA) based on
Randomization Number. See Note 2.1</TranslatedText>
  </Description>
  <def:DocumentRef leafID="LF.ReviewersGuide">
    <def:PDFPageRef PageRefs="section2.1" Type="NamedDestination"/>
  </def:DocumentRef>
</def:CommentDef>

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

In Define-XML version 2 the XSL stylesheet will no longer have to guess as what to link to. It is expected, for Named Destinations to work, that the PDF file will have the Named Destination defined. It is the responsibility of the XSL stylesheet to convert this XML into HTML hyperlinks. This might look like the following:

```
CRF Pages <a href="blankcrf.pdf#page=6">6</a>
<a href="blankcrf.pdf#page=7">7</a> <a href="blankcrf.pdf#page=8">8</a>

<a href="reviewersguide.pdf#secttion2.1">Derived from arm code in TRIAL ARM
(TA) based on Randomization Number. See Note 2.1</a>
```

These document references can play an important role where the define.xml document does not adequately document SDTM mapping decisions, sponsor-defined domains, sponsor-defined controlled terminology, and sponsor extensions to CDISC controlled terminology, as stated in the Study Data Reviewers Guide [8]. Another use case is when a context for analysis datasets and terminology that benefit from additional explanation beyond the Statistical Analysis Plan (SAP) and the Data Definitions document (define.xml) needs to be provided [9].

VALUE LEVEL METADATA

The normalized data structure used by datasets based on the SDTM, SEND and ADaM models (generally one record per subject per test code or per parameters per visit or observation) provides an efficient method for transmitting information. However, there are cases where the dataset variable metadata does not provide sufficient detail to support data review and analysis. For example, the Controlled Terminology for a Vital Signs test may depend on the particular kind of test (height, weight, size).

In these cases Value Level Metadata should be provided in the Define-XML document. Value Level Metadata enables the specification of the metadata of a variable under conditions involving one or more other dataset variables. The definition of a variable for a specific condition is known as Value Level Metadata.
Value Level Metadata should be provided when there is a need to describe differing metadata attributes for subsets of cells within a column. It is most often used on SDTM Findings domains to provide definitions for Variables such as --ORRES, --ORRESU, --STRES, --STRESU that are specific to each test code (value of --TESTCD). It is not required for Findings domains where the results have the same characteristics in all records, such as IE domains. In ADaM, value level metadata often describes AVAL or AVALC in BDS data structures based on values of PARAMCD.
Although support for Value Level Metadata has always been part of the Define-XML standard, there were shortcomings. In Define-XML version 1 Value Level Metadata was typically attached to –TESTCD variables or QNAM variables. It was defined by convention which data set variable was being described by the Value Level Metadata (e.g. –ORRES or QVAL).

In Define-XML version 2 *where clauses* are being used to explicitly describe – in a machine-readable form – under which condition the Value Level definitions apply. Value Level Metadata can be provided whenever there is a need to describe differing metadata attributes for values or subsets of values of a variable. In Define-XML version 2 valuelists should be attached to the variable being defined so that any number of variables can be defined in detail.

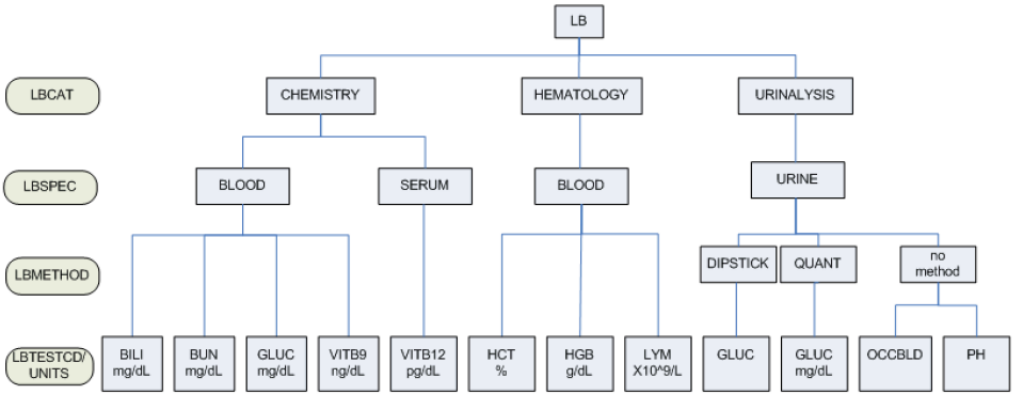
The following example shows Value Level Metadata for Vital Signs in Define-XML version 1. In the example we want to describe Value Level Metadata for the original test result (VSORRES). If heart rate, weight, and frame size are collected for a vital signs dataset, heart rate is typically collected as a numeric integer value, weight is often collected with fractional numeric values (data type =float), and frame size may be collected as a coded character field (SMALL, MEDIUM, LARGE).

Example 4 Value Level Metadata in Define-XML version 1

```
<def:ValueListDef OID="VL.VS.VSTESTCD">
  <ItemRef ItemOID="IT.VS.VSTESTCD.FRMSIZE" OrderNumber="1" Mandatory="No"/>
  <ItemRef ItemOID="IT.VS.VSTESTCD.HEIGHT" OrderNumber="2" Mandatory="No"/>
  <ItemRef ItemOID="IT.VS.VSTESTCD.WEIGHT" OrderNumber="3" Mandatory="No"/>
  <ItemRef ItemOID="IT.VS.VSTESTCD.PULSE" OrderNumber="4" Mandatory="No"/>
</def:ValueListDef>


<ItemGroupDef OID="IG.VS" Name="VS" def:Label="Vital Signs" ...
  def:Class="Findings" def:ArchiveLocationID="Location.VS">
  <ItemRef ItemOID="IT.VS.VSTESTCD" OrderNumber="5" Mandatory="Yes" ... />
  ...
  <ItemRef ItemOID="IT.VS.VS.VSORRES" OrderNumber="8" Mandatory="No" ... />
  ...
</ItemGroupDef>


<ItemDef OID="IT.VS.VSTESTCD" Name="VSTESTCD" DataType="text" Length="8"
  Origin="Assigned" def:Label="Vital Signs Test Short Name">
  <CodeListRef CodeListOID="CL.VSTESTCD"/>
  <def:ValueListRef ValueListOID="VL.VS.VSTESTCD"/>
</ItemDef>

<ItemDef OID="IT.VS.VSORRES" Name="VSORRES" DataType="text" Length="30"
  Origin="CRF Page 11" def:Label="Result or Finding in Original Units"/>

<ItemDef OID="IT.VS.VSTESTCD.FRMSIZE" Name="FRMSIZE" DataType="text" Length="6"
  Origin="CRF Page 11" def:Label="Body Frame Size">
  <CodeListRef CodeListOID="CL.SIZE"/>
</ItemDef>
<ItemDef OID="IT.VS.VSTESTCD.HEIGHT" Name="HEIGHT" DataType="float"
  SignificantDigits="1" Length="5" Origin="CRF Page 11" def:Label="Height"/>
<ItemDef OID="IT.VS.VSTESTCD.WEIGHT" Name="WEIGHT" DataType="float"
  SignificantDigits="1" Length="4" Origin="CRF Page 11" def:Label="Weight"/>
<ItemDef OID="IT.VS.VSTESTCD.PULSE" Name="PULSE" DataType="integer"
  Length="2" Origin="CRF Page 11" def:Label="Pulse Rate"/>
```

Notice that nowhere in this example there is metadata explaining that the Value Level Metadata is being defined for the Original Result (VSORRES). Also, since the ValueList is attached to the single variable that defines the condition (VSTESTCD), it gets complicated and unwieldy to define metadata that is based on conditions involving more than a single VSTESTCD value. The Study Data Tabulation Model Metadata Submission Guidelines (SDTM-MSG) describes an example when using the CDISC standard laboratory test code dictionary, it was necessary to nest on lab specimen and method in order to provide appropriate descriptions of different types of results that share the same test code. For example, serum glucose, qualitative urine glucose, and quantitative 24-hour urine glucose all share the same LBTESTCD value (GLUC), but have different attributes. This resulted in a very complicated nesting of valuelists as can be seen in Figure 2.

Figure 2 Scheme of the nested valuelists for the LB domain.



In Define-XML version 2 Where Clauses are used to describe the conditions under which the definition of a Value applies in a machine-readable form. Each Value definition may have a Where Clause attached to it to describe when that Value applies. Where Clauses define a condition by using one or more Range Checks. Where there are multiple Range Checks the condition is defined by the logical AND of all the Range Checks. Below is an example of a compound Where Clause where a condition is based on a Vital Signs test code (VSTESTCD=”SYSBP”) and a Vital Signs position (VSPOS=”SITTING”).

Example 5 Where Clause definition in Define-XML version 2

```
<def:WhereClauseDef OID="WC.VS.VSTESTCD.SYSBP.VS.VSPOS.SITTING">
  <RangeCheck SoftHard="Soft" def:ItemOID="IT.VS.VSTESTCD" Comparator="EQ">
    <CheckValue>SYSBP</CheckValue>
  </RangeCheck>
  <RangeCheck SoftHard="Soft" def:ItemOID="IT.VS.VSPOS" Comparator="EQ">
    <CheckValue>SITTING</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
```

Example 6 shows the Value Level Metadata from example 4 (Define-XML version 1), but then implemented in Define-XML version 2. The value list is now attached to VSORRES with conditions defined based on VSTESTCD.

Example 6 Value Level Metadata in Define-XML version 2

```
<def:ValueListDef OID="VL.VS.VSORRES">
  <ItemRef ItemOID="IT.VS.VSORRES.FRMSIZE" OrderNumber="1" Mandatory="No">
    <def:WhereClauseRef WhereClauseOID="WC.VS.VSTESTCD.FRMSIZE"/>
  </ItemRef>
  <ItemRef ItemOID="IT.VS.VSORRES.HEIGHT" OrderNumber="2" Mandatory="No">
    <def:WhereClauseRef WhereClauseOID="WC.VS.VSTESTCD.HEIGHT"/>
  </ItemRef>
  ...
</def:ValueListDef>

<def:WhereClauseDef OID="WC.VS.VSTESTCD.FRMSIZE">
  <RangeCheck SoftHard="Soft" def:ItemOID="IT.VS.VSTESTCD" Comparator="EQ">
    <CheckValue>FRMSIZE</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.VS.VSTESTCD.HEIGHT">
  <RangeCheck SoftHard="Soft" def:ItemOID="IT.VS.VSTESTCD" Comparator="EQ">
    <CheckValue>HEIGHT</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
...
<ItemGroupDef OID="IG.VS" Name="VS" Domain="VS" ... SASDatasetName="VS"
  def:Class="Findings" def:ArchiveLocationID="Location.VS">
  <Description>
    <TranslatedText xml:lang="en">Vital Signs</TranslatedText>
  </Description>

  <ItemRef ItemOID="IT.VS.VSTESTCD" OrderNumber="5" Mandatory="Yes" ... />
  ...
  <ItemRef ItemOID="IT.VS.VS.VSORRES" OrderNumber="8" Mandatory="No" ... />
  ...
</ItemGroupDef>

<ItemDef OID="IT.VS.VSTESTCD" Name="VSTESTCD" DataType="text" Length="8"
  SASFieldName="VSTESTCD">
  <Description>
    <TranslatedText xml:lang="en">Vital Signs Test Short Name</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.VSTESTCD"/>
  <def:Origin Type="Assigned"/>
</ItemDef>

<ItemDef OID="IT.VS.VSORRES" Name="VSORRES" DataType="text" Length="30"
  SASFieldName="VSORRES">
  <Description>
    <TranslatedText xml:lang="en">Result or Finding in Original Units</TranslatedText>
  </Description>
  <def:Origin Type="CRF">
    <def:DocumentRef leafID="LF.blankcrf">
      <def:PDFPageRef PageRefs="11" Type="PhysicalRef"/>
    </def:DocumentRef>
  </def:Origin>
  <def:ValueListRef ValueListOID="VL.VS.VSORRES"/>
</ItemDef>

<ItemDef OID="IT.VS.VSORRES.FRMSIZE" Name="VS.FRMSIZE.ORRES" DataType="text" Length="6"
  SASFieldName=" FRMSIZE">
  <Description>
    <TranslatedText xml:lang="en">Body Frame Size</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.SIZE"/>
  <def:Origin Type="CRF">
    <def:DocumentRef leafID="LF.blankcrf">
      <def:PDFPageRef PageRefs="11" Type="PhysicalRef"/>
    </def:DocumentRef>
  </def:Origin>
</ItemDef>

...
```

Example 7 demonstrates a Where Clause with references to:

1. Variables in the current dataset (VSTESTCD in the VS data set)
2. Variables in other subject-level datasets within the same metadata version (COUNTRY variable in DM)

In case #2, the implied dataset join should be documented in a comment attached to the WhereClauseDef.

In the current specification, there is no mechanism for machine-readable join specifications. Therefore, the Comment functionality is intended to document the implied join.

Example 7 Value Level Metadata in Define-XML version 2

```
<def:WhereClauseDef OID="WC.VS.VSTESTCD.WEIGHT.[DM].COUNTRY.CMETRIC"
  def:CommentOID="COM.SUBJECTDATA-JOIN-DM">
  <RangeCheck SoftHard="Soft" def:ItemOID="IT.VS.VSTESTCD" Comparator="EQ">
    <CheckValue>WEIGHT</CheckValue>
  </RangeCheck>
  <RangeCheck SoftHard="Soft" def:ItemOID="IT.DM.COUNTRY" Comparator="IN">
    <CheckValue>CAN</CheckValue>
    <CheckValue>MEX</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>

<ItemDef OID="IT.DM.COUNTRY" Name="COUNTRY" DataType="text" Length="3"
SASFieldName="COUNTRY">
  <Description>
    <TranslatedText xml:lang="en">Country</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.ISO3166"/>
  <def:Origin Type="Assigned"/>
</ItemDef>

<def:CommentDef OID="COM.SUBJECTDATA-JOIN-DM">
  <Description>
    <TranslatedText xml:lang="en">Join any Subject Level dataset with the Demographics
dataset based on [IG.datasetname]IT.USUBJID = [IG.DM]IT.USUBJID, assuming
'IG.datasetname' is the OID of the ItemGroupDef that defines the subject-level dataset
to be joined with the Demographics dataset.</TranslatedText>
  </Description>
```

Note: It may seem clear from the def:ItemOID definition (“IT.DM.COUNTRY”) that the COUNTRY variable in this WhereClause definition is part of the DM data set. However, this is just by convention. An application cannot depend on these conventions. OIDs are only intended as a mechanism for unambiguously linking between a definition of an object and references to it. The value of an OID attribute has no intrinsic meaning by itself (although some may find convenience in applying a pre-defined convention for assigning values, as is illustrated in these examples), but its significance comes in its matching with values of other OIDs.

Example 8, where the def:ItemOID attributes have been replaced by Universal Unique Identifiers is semantically identical to Example 7. Applications, including XSL stylesheets should not behave differently when identifiers change, as long as the associations between defections and references are kept intact.

Example 8 Value Level Metadata in Define-XML version 2

```
<def:WhereClauseDef OID="WC.VS.VSTESTCD.WEIGHT.[DM].COUNTRY.CMETRIC"
  def:CommentOID="COM.SUBJECTDATA-JOIN-DM">
  <RangeCheck SoftHard="Soft" def:ItemOID="fd47e3f5-4853-439f-930e-b8329a50f649"
   Comparator="EQ">
    <CheckValue>WEIGHT</CheckValue>
  </RangeCheck>
  <RangeCheck SoftHard="Soft" def:ItemOID="a368f22b-68a4-4396-949e-eed0a8653ad8"
    Comparator="IN">
    <CheckValue>CAN</CheckValue>
    <CheckValue>MEX</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>

<ItemDef OID="a368f22b-68a4-4396-949e-eed0a8653ad8" Name="COUNTRY" DataType="text"
  Length="3" SASFieldName="COUNTRY">
  <Description>
    <TranslatedText xml:lang="en">Country</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.ISO3166"/>
  <def:Origin Type="Assigned"/>
</ItemDef>

<def:CommentDef OID="COM.SUBJECTDATA-JOIN-DM">
  <Description>
    <TranslatedText xml:lang="en">Join any Subject Level dataset with the Demographics
dataset based on [IG.datasetname]IT.USUBJID = [IG.DM]IT.USUBJID, assuming
'IG.datasetname' is the OID of the ItemGroupDef that defines the subject-level dataset
to be joined with the Demographics dataset.</TranslatedText>
  </Description>
```

CONTROLLED TERMINOLOGY

The term “Controlled Terminology” in the context of a study refers to the set of all allowable values across all variables that have finite sets of allowable values in the study. A “Codelist” is a unique subset of the controlled terminology to which one or more variables are subject. Beginning with SDTM Version 1.2, the SDTM-IG requires controlled terminology for many SDTM variables. For some variables, sponsor-specific controlled terminology is recommended. All controlled terminology used in a study must be provided within the Define-XML document. Each codelist referenced by a study item shall be represented in the Define-XML document using a CodeList element.

The CodeList element can define either an internal or external codelist. Internal codelists include a list of allowable codes and, if applicable, their corresponding decodes. In most cases where controlled terminology is just an enumeration of allowed values, the EnumeratedItem element can be used to define the list of values. In cases where it is useful to provide decodes for the coded values, the CodeListItem element can be used.

In Define-XML version 1 there was only one way to define internal codelists in the define.xml file. You always had to specify both coded values and decode values, as can be seen in example 9.

Example 9 Controlled Terminology in Define-XML version 1

```
<CodeList OID="CL.ACN" Name="Action Taken with Study Treatment" DataType="text">
  <CodeListItem CodedValue="DOSE NOT CHANGED">
    <Decode><TranslatedText>DOSE NOT CHANGED</TranslatedText></Decode>
  </CodeListItem>
  <CodeListItem CodedValue="DOSE REDUCED">
    <Decode><TranslatedText>DOSE REDUCED</TranslatedText></Decode>
  </CodeListItem>
  <CodeListItem CodedValue="DRUG INTERRUPTED">
    <Decode><TranslatedText>DRUG INTERRUPTED</TranslatedText></Decode>
  </CodeListItem>
  <CodeListItem CodedValue="DRUG WITHDRAWN">
    <Decode><TranslatedText>DRUG WITHDRAWN</TranslatedText></Decode>
  </CodeListItem>
</CodeList>
```

Define-XML version 2 adds extended support for Controlled Terminology. The CodeList element can define either an internal or external CodeList. Internal CodeLists include a list of allowable codes and, if applicable, their corresponding decodes. In most cases where controlled terminology is just an enumeration of allowed values, the EnumeratedItem element can be used to define the list of values. In cases where it is useful to provide decodes for the coded values, the CodeListItem element can be used. Both of these CodeLists items now can have an attribute that defines whether the item is an extension to an extensible CodeList.

For CodeList elements that define the contents of a CDISC Controlled Terminology an identification of the C-codes has been implemented that is used to identify Controlled Terminology Codelists and Coded Terms within the National Cancer Institute's Enterprise Vocabulary System.

In Define-XML version 2 example 9 can be defined by utilizing EnumeratedItem elements (example 10).

Example 10 Controlled Terminology in Define-XML version 2

```
<CodeList OID="CL.ACN" Name="Action Taken with Study Treatment" DataType="text">
  <EnumeratedItem CodedValue="DOSE NOT CHANGED">
    <Alias Name="C49504" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="DOSE REDUCED">
    <Alias Name="C49505" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="DRUG INTERRUPTED">
    <Alias Name="C49501" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="DRUG WITHDRAWN">
    <Alias Name="C49502" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
  <Alias Name="C66767" Context="nci:ExtCodeID"/>
</CodeList>
```

For codelists with entries that have a numeric significance, this can be described using the Rank attribute. For example, a list of enumerated text values including "Low", "Medium", and "High" might include a Rank attribute on each with values 1, 2, and 3 respectively. This can be used to override the normal lexical ordering of the entries with one based on numeric significance. The Rank attribute should not be used to define a display order.

The OrderNumber attribute can be used to define a display order for the items in a codelist.

CodeLists should never include a blank coded value. If an item does not require a value then the Mandatory attribute in the ItemRef element must be set to "No". Required items have Mandatory set to "Yes" and cannot have blank values.

CDISC codelists can be defined as extensible, which means controlled terms may be added to the codelist. Sponsors may add to extensible codelists as long as they are not adding duplicates or synonyms of existing terms. If a CDISC codelist is not extensible, the expectation is that sponsors will use only the published list of terms. If a Define-XML CodeList element includes definitions of terms that are not included in the published list, then the def:ExtendedValue attribute should be set to "Y" regardless of whether or not the codelist is extensible.

A codelist definition may include one or more Alias elements in order to facilitate the identification of the codelist components in an external system. For CodeList elements that define the contents of a CDISC codelist, an Alias element is used to provide the C-codes that are used to identify codelists and coded terms within the National Cancer Institute's Enterprise Vocabulary System. By convention the Alias Context attribute is set to "nci:ExtCodeId" in this scenario.

In example 11, a CodeListItem ("SUBJINIT") has been added by the sponsor to extend external controlled terminology. Note that the last *CodeListItem* element does not include an *Alias* element since it represents a value added to an extensible CDISC Controlled Terminology. The final *Alias* element identifies the CDISC Controlled Terminology for "Subject Characteristic Test Code (SCTESTCD)".

Example 11 Controlled Terminology in Define-XML version 2

```
<CodeList OID="CL.SCTESTCD" Name="Subject Characteristic Code (SCTESTCD)"
  DataType="text" SASFormatName="$SCTESTC">
  <CodeListItem CodedValue="EDLEVEL">
    <Decode>
      <TranslatedText xml:lang="en">Education Level</TranslatedText>
    </Decode>
    <Alias Name="C17953" Context="nci:ExtCodeID"/>
  </CodeListItem>
  <CodeListItem CodedValue="MARISTAT">
    <Decode>
      <TranslatedText xml:lang="en">Marital Status</TranslatedText>
    </Decode>
    <Alias Name="C25188" Context="nci:ExtCodeID"/>
  </CodeListItem>
  <CodeListItem CodedValue="SUBJINIT" def:ExtendedValue="Yes">
    <Decode>
      <TranslatedText xml:lang="en">Subject Initials</TranslatedText>
    </Decode>
  </CodeListItem>
  <Alias Name="C74559" Context="nci:ExtCodeID"/>
</CodeList>
```

ALGORITHMS

Algorithms can now be defined by using the ODM MethodDef elements. For cases where the algorithm description is longer than a few lines the method can link to a section in a Supplemental Document containing the additional details. A formal expression can be provided that contains a machine-readable expression that implements the algorithm.

Examples 12 and 13 show an algorithm in Define-XML version 1 and in Define-XML version 2.

Example 12 Algorithm in Define-XML version 1

```
<def:ComputationMethod OID="MT.SESTDTC">If Element = SCREEN, derived from SVSTDTC where
      VISIT = SCREENING or from DS where DSDECOD = 'INFORMED CONSENT', whichever is
      earliest. If Element = EOS, derived from DS where DSCAT = DISPOSITION EVENT.
      For treatment Elements, derived from first EXSTDTC for the element.</ComputationMethod>
```

Example 13 Algorithm in Define-XML version 2

```
<MethodDef OID="MT.SESTDTC" Name="Algorithm to derive SESTDTC" Type="Computation">
  <Description>
    <TranslatedText xml:lang="en">If Element = SCREEN, derived from SVSTDTC where
      VISIT = SCREENING or from DS where DSDECOD = 'INFORMED CONSENT', whichever is
      earliest. If Element = EOS, derived from DS where DSCAT = DISPOSITION EVENT.
      For treatment Elements, derived from first EXSTDTC for the element.
    </TranslatedText>
  </Description>
</MethodDef>
```

Examples 12 and 13 are not fundamentally different. Example 14 shows the added functionality of Define-XML version 2. Example 14 would allow the XSL stylesheet to create a link to a named destination in the PDF file that further explains the algorithm.

Example 14 Algorithm in Define-XML version 2 with a document reference

```
<MethodDef OID="MT.EGDRVFL" Name="Algorithm to derive EGDRVFL" Type="Computation">
  <Description>
    <TranslatedText xml:lang="en">EGDRVFL = "Y" for derived EGTESTCDs QTCB and QTCF.
      Null otherwise. </TranslatedText>
  </Description>
  <def:DocumentRef leafID="LF.ComplexAlgorithms">
    <def:PDFPageRef PageRefs="EG" Type="NamedDestination"/>
  </def:DocumentRef>
</MethodDef>


<def:leaf ID="LF.ComplexAlgorithms" xlink:href="complexalgorithms.pdf">
  <def:title>Complex Algorithms</def:title>
</def:leaf>
```

Example 15 shows how to link to a file with SAS programming code.

Example 15 Algorithm in Define-XML version 2 with an external program reference

```
<MethodDef OID="MT.QTCB" Name="Algorithm to derive QTCB" Type="Computation">
  <Description>
    <TranslatedText xml:lang="en">QTcB = QT interval / square root of
      (60 / heart rate). For the complete algorithm see the referenced
       external document.
    </TranslatedText>
  </Description>
  <def:DocumentRef leafID="LF.CODE.001"/>
</MethodDef>

<def:leaf ID="LF.CODE.001" xlink:href="../programs/QTCB-computation-sas.txt">
  <def:title>QTcB-Bazett's Correction Formula</def:title>
</def:leaf>
```

COMMENTS

Define-XML version 1 had very limited possibilities for providing comments related to the various metadata objects in a clinical study (data sets, variables, derivations, ...). Define-XML allows the definition of Comments at both the dataset and the variable and value levels. The mechanism allows referencing short comments self-contained in the Define-XML document or long comments referenced in external documents. For comments in external documents, the reference can include specific pages within the document, or a Named Destination.

Example 16 A Data Set Comment related to Split Domains

```
<ItemGroupDef OID="IG.QSCG" Domain="QS" Name="QSCG" Repeating="Yes" IsReferenceData="No"
   SASDatasetName="QSCG" Purpose="Tabulation"
   def:Structure="One record per questionnaire per question per visit per subject"
   def:Class="FINDINGS"
   def:CommentOID="COM.DOMAIN.QS" def:ArchiveLocationID="LF.QSCG">
…
<ItemGroupDef OID="IG.QSCS" Domain="QS" Name="QSCS" Repeating="Yes" IsReferenceData="No"
  SASDatasetName="QSCS" Purpose="Tabulation"
  def:Structure="One record per questionnaire per question per visit per subject"
  def:Class="FINDINGS"
  def:CommentOID="COM.DOMAIN.QS" def:ArchiveLocationID="LF.QSCS">
…
<!-- Comment Definition: Long Comment, included in a PDF file -->
<def:CommentDef OID="COM.DOMAIN.QS">
  <Description>
    <TranslatedText xml:lang="en"> QS is submitted as a split dataset. The split was done
      based on QSCAT as QSCG (CLINICAL GLOBAL IMPRESSIONS), QSCS (CORNELL SCALE
      FOR DEPRESSION INDEMENTIA) and QSMM (MINI MENTAL STATE EXAMINATION).
      See additional documentation in the Reviewer's Guide, Split Datasets Section.
    </TranslatedText>
  </Description>
  <def:DocumentRef leafID="LF.ReviewersGuide"/>
</def:CommentDef>

<def:leaf ID="LF.ReviewersGuide" xlink:href="reviewersguide.pdf">
  <def:title>Reviewers Guide</def:title>
</def:leaf>
```

Example 17 A Data Set Variable Comment

```
<ItemDef OID="IT.MH.MHBODSYS" Name="MHBODSYS" DataType="text" Length="23"
  SASFieldName="MHBODSYS" def:CommentOID="COM.MHBODSYS">
  <Description>
    <TranslatedText xml:lang="en">Body System or Organ Class</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.MHBODSYS"/>
  <def:Origin Type="Assigned"/>
</ItemDef>

<def:CommentDef OID="COM.MHBODSYS">
  <Description>
    <TranslatedText xml:lang="en">Assigned for Medical History but not Psychiatric
History</TranslatedText>
  </Description>
  <def:DocumentRef leafID="LF.ReviewersGuide">
    <def:PDFPageRef PageRefs="MH" Type="NamedDestination"/>
  </def:DocumentRef>
</def:CommentDef>

<def:leaf ID="LF.ReviewersGuide" xlink:href="reviewersguide.pdf">
  <def:title>Reviewers Guide</def:title>
</def:leaf>
```

CONCLUSION

Define-XML 2.0.0 has a lot of opportunities to improve Data Definition Specifications.

REFERENCES

1. CDISC Define-XML Specification, Version 2.0, March 5, 2013 (http://www.cdisc.org/define-xml)
2. Case Report Tabulation Data Definition Specification (define.xml), Version 1.0, February 9, 2005 (http://www.cdisc.org/define-xml)
3. Lex Jansen (2012). Using the SAS® Clinical Standards Toolkit for define.xml creation. Proceedings of the Pharmaceutical Industry SAS® Users Group (PharmaSUG 2012, San Francisco, CA) (http://www.lexjansen.com/pharmasug/2012/HW/PharmaSUG-2012-HW02-SAS.pdf)
4. U.S. Department of Health and Human Services Food and Drug Administration Center for Drug Evaluation and Research (CDER). Study Data Specifications, Version 2.0, July 20, 2012 (http://www.fda.gov/ForIndustry/DataStandards/StudyDataStandards/default.htm)
5. FDA Study Data Standards Catalog (Version 2.2; Effective 2013-08-15). Retrieved on September 7, 2013. (http://www.fda.gov/downloads/ForIndustry/DataStandards/StudyDataStandards/UCM340684.xls)
6. CDISC Operational Data Model (ODM), Version 1.3.2, March 2013 (http://www.cdisc.org/odm)
7. Study Data Tabulation Model Metadata Submission Guidelines (SDTM-MSG), Version 1.0, December 2011 (http://www.cdisc.org/sdtm)
8. Study Data Reviewers Guide (SDRG), Version 1.1, May 2013 (http://www.phusewiki.org/wiki/index.php?title=Study_Data_Reviewer%27s_Guide)
9. Analysis Data Reviewer's Guide (ADRG), work in progress, as of September 15, 2013 (http://www.phusewiki.org/wiki/index.php?title=Analysis_Data_Reviewer%27s_Guide)

ACKNOWLEDGMENTS

The author would like to thank and recognize his fellow members of the CDISC XML Technologies Define-XML development team: Sam Hume, Sally Cassells, Kevin Burgess, Marcelina Hungria and Mike Molter.

RECOMMENDED READING

- CDISC: http://www.cdisc.org
- FDA's Study Data Standards Resources: http://www.fda.gov/ForIndustry/DataStandards/StudyDataStandards/default.htm
- Controlled Terminology, on NCI-EVS: http://www.cancer.gov/cancertopics/cancerlibrary/terminologyresources/cdisc

CONTACT INFORMATION

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

Lex Jansen
SAS Institute Inc.
SAS Campus Drive
Cary, North Carolina, 27513, USA
Phone: 919-531-9860
Email: lex.jansen@sas.com