PharmaSUG2015 - Paper AD06

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

Min Chen, Alkermes Inc., Waltham, MA

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

ABSTRACT

Both define.xml and a reviewer guide (define.pdf) are integral parts of any electronic FDA submission, in addition to SDTM and ADaM datasets. The standardized, well-defined, and detailed define files minimize the time needed for FDA reviewers to familiarize the data, and speed up the overall review process. The programming specification documentation serves as a part of a reviewer guide, as well as the documentation for programming validation. It is very crucial to ensure the consistency of the attributes of variables among datasets, define files, and programming specification. It is highly desirable to automate this process to ensure technical accuracy and operational efficiency. The automation of generating define-xml has been a challenge for statistical programming as Define-XML v2.0 released in March 2013 with significant enhancement from version 1.0 This paper introduces a metadata-driven SAS® macro-based tool that can automate the creation of Define.xml v2.0 from a CDISC-compliant ADaM Specifications for FDA electronic submission. ADaM specifications were designed for define-xml v2.0 new features. It avoids the waste of resources for manual creation and/or verification of the consistency at the later stage, facilitates regulatory submissions, and achieves high quality of submission.

INTRODUCTION

December 2011 CDER Common Data Standards Issues Document (Version 1.1) [1] states: “A critical component of data submission is the define file. A properly functioning define.xml file is an important part of the submission of standardized electronic datasets and should not be considered optional.”

“Additionally, sponsors should make certain that every data variable’s codelist, origin, and derivation is clearly and easily accessible from the define file. An insufficiently documented define file is a common deficiency that reviewers have noted.”

CDISC XML Technologies Team released the Define-XML v2.0 specification in March 2013. The define.xml v2.0 release package can be downloaded from the CDISC website (http://www.cdisc.org/define-xml). The distribution package includes: CDISC define-xml 2.0 specification, define-xml 2.0 schema, an SDTM based define-xml example and its html rendition, an ADaM based define-xml example and its html rendition, and XSL stylesheet referenced by the two define-xml examples.

Key changes from the new version include: support for CDISC/NCI controlled terminology, flexible definition of value level metadata, enhanced documentation of data origin or source, and improved handling of comments ADaM Define-XML provides metadata for describing data sets about:

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

• Domains (Dataset Name, Description, Structure, Dataset Location, ...)

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

• (Parameter) Value Level Metadata

• Controlled Terminology

• Computational Method (Analysis Derivations, for derived variables)

• Comments (for datasets and assigned variables)

• Supporting Documents (Analysis Data Reviewer’s Guide)

In order to create the define.xml file for ADaM datasets, we need different files to store the information listed above. Some developers create define-xml based on SAS submission dataset; and some developers use an Excel® Workbook to organize the metadata in different sheets, which are the inputs of a SAS program (macro) to create the define.xml. However the tedious and error-prone manual work from this process jeopardizes quality of submission, not to mention that resource is wasted from this process.

CDISC define-xml 2.0 specification describes an updated Define-XML 2.0 model that is used to describe CDISC SDTM, SEND and ADaM datasets for the purpose of submissions to the FDA. It is a very useful document for developing a tool (a SAS macro) to create define.xml file. This paper describes a metadata-driven method to pull metadata automatically from ADaM Specification in MS Word® format and use the metadata automatically to create Define-XML 2.0 for FDA submission. It also details how to develop ADaM Metadata (programming specification) for

automation purpose and how, illustrates different sections for ADaM Define-XML 2.0, and provides part of code for how to achieve the automation from Metadata to Define-XML 2.0.

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

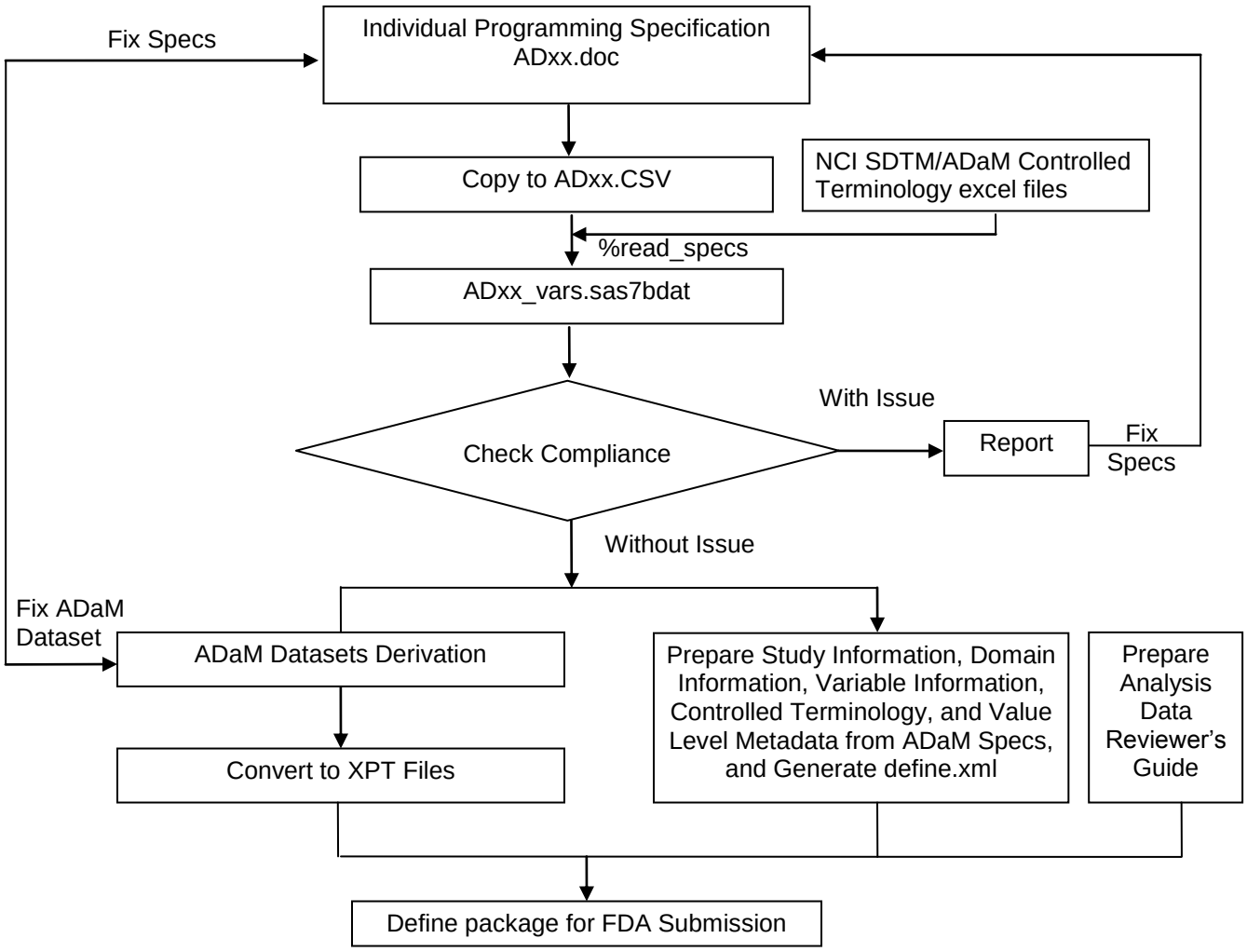


Figure 1. Overview of Process Flow

INTRODUCTION OF DEFINE-XML V2.0

Define-xml provides the specification for the data definitions for datasets. Define-xml v2.0 includes 8 key metadata components: Study and MetaDataVersion Section, links to supporting documents, dataset definitions, dataset variable definitions, controlled terminology definitions, value list definitions, computational method definitions, and comments definitions, to support the FDA submission [2] as shown in Appendix 1:
Compared with Define-xml v1.0, the new features of Define-xml v2.0 include:

(1) The definition of the source of ADaM variables is clearly clarified as Predecessor, Assigned, or Derived;

(2) Variable length is added

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

(4) The value level metadata section for PARCAT1 and PARAMCD is displayed in the controlled terminology section in Define-xml v2.0. However the new parameter value level metadata section is more flexibly defined by where clause shown in Appendix 1 (c), with which the variable value can be defined based on multiple variables with more comparators such as LT, GE, NOTIN, etc.

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

In define-xml v2.0, components MetaDataVersion Section and links to supporting documents contain the standard information about the metadata version information and standard external documentation such as reviewers' guide. The other 6 components include the detailed metadata information in a specific study. A CDISC compliant ADaM specification in MS Word® format is designed to provide all these metadata-related information needed in define-xml v2.0, which facilitates programmers and statisticians to review and communicate derivation rules among them, as well as to track the changes.

The ADaM specification consists of two blocks: **Domain** Information and **Variable** Information. The Domain Information block will provide the information related to Analysis Dataset in define-xml, and the Variable Information Block the information related to Dataset Variable, including the controlled terminology, and parameter value level metadata in define-xml as shown in Appendix 1. Well-design ADaM specifications are very crucial for the automation and generation of define-xml v2.0 as all study-related information will be retrieved from ADaM specifications. We will introduce the design in detail in the following sections.

In out macro-based methodology, the ADaM specifications will be copied to a comma delimited document CSV file in order to be read by a SAS macro. For automation purposes, a macro named **%read_specs** will read and reformat all the information based on the format of the individual specification into 4 SAS datasets domain_info, vars_info, ctlist_info, and value_info for the preparation of 6 major metadata components of define-xml v2.0.

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

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

Source (Row/Column) in ADaM Specs	MetaData Section in Define-xml	Define-XML Element	Reference	ODM Element Identifier
Domain Info.	Analysis Dataset Section	ItemGroupDef		
Domain Info. (Documentation)	Comments Section for Dataset	def:CommentDef	ItemGroupDef	def:CommentOID
Domain Info. (Dataset Name)	Dataset File	def:leaf xlink:href	ItemGroupDef	def:ArchiveLocationID
Variable Info.	Dataset Variable Section	ItemDef	ItemRef	ItemOID
Variable Info. (Controlled Terms or Format)	Controlled Terminology Section	CodeList	CodeListRef	CodeListOID
Variable Info: (Source/Derivation/Comment, Value Level Metadata)	Value List Definitions Section	def:ValueListDef	def:ValueListRef	ValueListOID
Variable Info: (where clause in Value Level Metadata)	WhereClause in Value List Definitions Section	def:WhereClause Def	def:WhereClauseRef	WhereClauseOID
Variable Info. (Source/Derivation/ Comment), for derived variables	Computational Method Definitions Section	MethodDef	ItemRef	MethodOID
Variable Info. (Source/Derivation/ Comment), for assigned variables	Comments Definitions Section	def:CommentDef	ItemDef	def:CommentOID
Variable Info: (Origin)	Origin	def:Origin	ItemDef	

Table 1. The format mapping from ADaM variables to define-xml element

As some characters have special meanings in XML, we need to replace 5 special characters with 5 pre-defined entity references in XML as below:

<	<
>	>
&	&
‘	'
“	"

DESIGN OF DOMAIN INFORMATION TABLE FOR DEFINE-XML V2.0

The Domain Information block in ADaM specifications contains information needed in Analysis Dataset Section of define-xml. The content of rows inside Domain Information block: **Dataset Name, Dataset Description, Class of Dataset, Dataset Structure, Key Variables of Dataset, and Documentation** will be displayed in this section as shown in Display 1.

Comments Section for Datasets

Analysis Datasets for Study yyy (ADaM-IG 1.1)

Dataset	Description	Class	Structure	Purpose	Keys	Location	Documentation
ADSL	Subject-Level Analysis Dataset	SUBJECT LEVEL ANALYSIS DATASET (ADSL)	One record per subject	Analysis	USUBJID	ADSL.xpt	Includes all randomized subjects
ADAE	Adverse Event Analysis Dataset	Occurrence Data Structure (ODS)	One record per subject per each AE recorded in SDTM AE domain	Analysis	STUDYID, USUBJID, AEDECOD, ASTDT	ADAE.xpt	Includes records from randomized subjects
ADHAMA	HAM-A Questionnaire Analysis Dataset	BASIC DATA STRUCTURE (BDS)	One record per subject per parameter per baseline type per analysis time point	Analysis	STUDYID, USUBJID, BASETYPE, PARCAT1, PARAM, ADT, ASEQ	ADHAMA.xpt	Includes records from randomized subjects
ADCSSRS	Columbia Suicide Severity Rating Scale	BASIC DATA STRUCTURE (BDS)	One record per subject per parameter per baseline type per analysis time point	Analysis	STUDYID, USUBJID, PARCAT1, PARAM, ADT	ADCSSRS.xpt	
ADLB	Laboratory Analysis Dataset	BASIC DATA STRUCTURE (BDS)	One record per subject per parameter per baseline type per analysis time point	Analysis	STUDYID, USUBJID, PARCAT1, PARAM, ADTM	ADLB.xpt	Includes records from randomized subjects
ADVS	Vital Signs Analysis Dataset	BASIC DATA STRUCTURE (BDS)	One record per subject per parameter per baseline type per analysis time point	Analysis	STUDYID, USUBJID, PARAM, ADTM	ADVS.xpt	Includes records from randomized subjects

Go to the top of the define.xml

1.1.1. ADSL: Subject-Level Analysis Dataset

Dataset Name	**ADSL**
Dataset Description	Subject-Level Analysis Dataset
Class of Dataset	SUBJECT LEVEL ANALYSIS DATASET (ADSL)
Dataset Structure	One record per subject
Key Variables of Dataset	USUBJID
Input Datasets	DM, QS, DS, EX, DA, SV, VS, MH, SUPPDM
Documentation	Includes all randomized subjects
Notes	It is for XXX-yyy.

Comments Section for Datasets

Domain Information, Map to **Analysis Dataset Section**

Display 1. Domain Information Table in CDISC-Compliant ADaM (ADSL) Specification Designed for Define-xml v2.0

Domain information will be stored in SAS data **DOMAIN_INFO** for preparing Analysis Dataset Section, as shown in Display 2.

RUNORDER	DOMAIN	DESCRIPTION	STRUCTURE	KEYS	PURPOSE	CLASS	CLASSORD	REPEATING	ISREFERENCEDATA	DOCUMENTATION
1	ADSL	Subject-Level Analysis Dataset	One record per subject	USUBJID	Analysis	SUBJECT LEVEL ANALYSIS DATASET (ADSL)	1	No	No	Includes records from randomized subjects
2	ADAE	Adverse Event Analysis Dataset	One record per subject per each AE recorded in SDTM AE domain	STUDYID, USUBJID, AEDECOD, ASTDT	Analysis	Occurrence Data Structure (ODS)	2	Yes	No	Includes records from randomized subjects
3	ADHAMA	HAM-A Questionnaire Analysis Dataset	One record per subject per parameter per baseline type per analysis time point	STUDYID, USUBJID, BASETYPE, PARCAT1, PARAM, ADT, ASEQ	Analysis	BASIC DATA STRUCTURE (BDS)	3	Yes	No	Includes all randomized subjects
4	ADCSSRS	Columbia Suicide Severity Rating Scale	One record per subject per parameter per baseline type per analysis time point	STUDYID, USUBJID, PARCAT1, PARAM, ADT	Analysis	BASIC DATA STRUCTURE (BDS)	4	Yes	No	Includes records from randomized subjects
5	ADLB	Laboratory Analysis Dataset	One record per subject per parameter per baseline type per analysis time point	STUDYID, USUBJID, PARCAT1, PARAM, ADTM	Analysis	BASIC DATA STRUCTURE (BDS)	5	Yes	No	Includes records from randomized subjects
6	ADVS	Vital Signs Analysis Dataset	One record per subject per parameter per baseline type per analysis time point	STUDYID, USUBJID, PARAM, ADTM	Analysis	BASIC DATA STRUCTURE (BDS)	6	Yes	No	Includes records from randomized subjects

Display 2. SAS data DOMAIN_INFO, preparing for Analysis Dataset Section in define-xml v2.0

Analysis Dataset Section

An example of Analysis Dataset Section in define-xml v2.0 (shown in Display 1) is shown as below, and dataset metadata is described by an **ItemGroupDef** element in Define-XML.

```
<ItemGroupDef OID="IG.ADSL"  Name="ADSL" SASDatasetName="ADSL" Repeating="No"
IsReferenceData="No" Purpose="Analysis" def:Structure="One record per subject" def:Class="SUBJECT
LEVEL ANALYSIS DATASET (ADSL)" def:CommentOID="COM.ADSL" def:ArchiveLocationID="LF.ADSL">
    <Description>
      <TranslatedText xml:lang="en">Subject-Level Analysis Dataset</TranslatedText>
    </Description>
    <def:leaf ID="LF.ADSL" xlink:href="ADSL.xpt">
      <def:title>ADSL.xpt </def:title>
    </def:leaf>
</ItemGroupDef>
```

In above code, the keywords to define define-xml elements are shown in bold font, and the contents are shown in plain. Our macro-based tool will insert the variable value in SAS dataset domain_info as contents of define-xml elements into define-xml file. Therefore, the define-xml code can be automatically generated by SAS from domain_info dataset by the following SAS DATA step.

```
data datasets(keep=line order section);
  set domain_info;
  by classord domain;
  retain order 0;
  section = 40;  *** Dataset Section Number ***;
  *** Domain Information Table ***;
  line= ' <ItemGroupDef OID="IG.'||strip(domain)||'"  Name="'||strip(domain)||
        '" SASDatasetName="'||strip(domain)||'" Repeating="'||strip(repeating)||
        '" IsReferenceData="'||strip(isreferencedata)||'"Purpose="'||strip(Purpose)||
        '"def:Structure="'||strip(structure)||'"'||'def:Class="'||strip(class);order+1;output;
  if documentation ne '' then do; line='" def:CommentOID="COM.'||strip(domain);order+1;output;
  end;
  line= '" def:ArchiveLocationID="LF.'||strip(domain)||'">';order+1;output;
  line= '        <Description>';order+1;output;
  line= '          <TranslatedText xml:lang="en">'||strip(description)||'</TranslatedText>';
  order+1;output;
  line= '        </Description>';order+1;output;
  *** Generate Comments Section for Dataset ***;
  if documentation ne ' ' then do;
     section = 80;    *** Comments for Analysis Dataset Section Number ***;
     line= '<def:CommentDef OID="COM.'||strip(domain)||'">';order+1;output;
     line= '   <Description>';order+1;output;
     line= '     <TranslatedText>'||strip(documentation)||'</TranslatedText>';order+1;output;
     line= '   </Description>';order+1;output;
     line= '</def:CommentDef>';order+1;output;
     section = 40;
  end;
  line= '        <def:leaf ID="LF.'||strip(domain)||'"   xlink:href="'||
        strip(domain)||'.xpt">';order+1;output;   *** Link to ADaM data ***;
  line= '          <def:title>'||strip(domain)||'.xpt </def:title>';order+1;output;
  line= '        </def:leaf>';order+1;output;
  line= '      </ItemGroupDef>';order+1;output;
run;
```

If variable LINE generated above is output to an xml file sorting by SECTION and ORDER, the Analysis Dataset Section will be shown in Display 2.

Comments Definitions Section for Dataset

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

```
<def:CommentDef OID="COM.ADSL">
    <Description>
      <TranslatedText>Includes all randomized subjects</TranslatedText>
    </Description>
</def:CommentDef>
```

The comment section for analysis dataset in define-xml v2.0 is shown in Display 3.

Comments

CommentOID	Description
COM.ADSL	Includes all randomized subjects

Display 3. Comments Definitions Section for Analysis Dataset ADSL in define-xml v2.0

DESIGN OF VARIABLE INFORMATION TABLE FOR DEFINE-XML V2.0

The Variable Information block contains the information needed for sections of Dataset Variable Definitions, Controlled Terminology Definitions, and Value List Definitions in define-xml. The content of columns inside Variable Information block: **Variable Name**, **Variable Label**, **Type**, **Length/Display Format, and Source/Derivation/Comment** (except of the part for Value Level Metadata) will be displayed in Variable Information section of define.xml. Each variable is categorized as **'Predecessor', 'Assigned', and 'Derived'** in define.xml. For derived variables, the Derivation rules will also be displayed in computational method definitions; for assigned variables, the comments will also be shown in comments definitions section. Table 2 provides information to variable origins, how they are shown in define.xml, and how they are written in ADaM specifications.

Category	Definition	Place in define.xml	How to write in ADaM specification
Predecessor	Variables are inherited from SDTM	Source/Derivation/ Comment Section	SDTM.Domain Name + '.' + Variable Name, e.g. LB.USUBJID, LB.LBSEQ, Note: No key word is needed!
Assigned	Coded variables for code-decode pairs	Source/Derivation/ Comment Section and Comments Definitions Section	Use key word 'Assigned:', e.g., ADAE.ASEVN (Analysis Severity/Intensity (N), code variable for ASEV) Assigned: 1 if ASEV = 'Mild' 2 if ASEV = 'Moderate' 3 if ASEV = 'Severe'
Derived	Other variables will be considered as derived variables.	Source/Derivation/ Comment Section and Computational Method Definition Section	Use key word 'Derived:' e.g., ADLB.ONTRTFL (On Treatment Record Flag) Derived: 'Y' if APHASEN in (1,2)

Table 2. How to Design Variable Origin Information in ADaM Specifications

Subject-Level Analysis Dataset (ADSL) [Location: ADSL.xpt]

Variable	Label	Type	Length / Display Format	Controlled Terms or Format	Source/Derivation/Comment
STUDYID	Study Identifier	text	20		Predecessor: DM.STUDYID
USUBJID	Unique Subject Identifier	text	40		Predecessor: DM.USUBJID
SUBJID	Subject Identifier for the Study	text	20		Predecessor: DM.SUBJID
SITEID	Study Site Identifier	text	8		Predecessor: DM.SITEID
BRTHDTC	Date/Time of Birth	text	20		Predecessor: DM. BRTHDTC
AGE	Age	integer	8		Predecessor: DM.AGE
AGEU	Age Units	text	8	["YEARS"] <ADSL.AGEU>	Predecessor: DM.AGEU
AGEGR1	Pooled Age Group 1	text	20	["<65", ">=65"] <AGEGR1>	Derived: If .<AGE<65 then AGEGR1='<65'; Else if AGE>=65 then AGEGR1='>=65'
AGEGR1N	Pooled Age Group 1 (N)	integer	8	["1" = "<65", "2" = ">=65"] <AGEGR1N>	Assigned: 1, if AGEGR1='<65'; 2, if AGEGR1='>=65'
BMI01BL	Basln Bdy Mass Index (kg/m^2) In Stage 1	float	5.2		Derived: It is derived from the last non-missing value when VS.VSTESTCD ='BMI' on or before the date of Visit 2

Computational Method

Comments

Variable Information ↑ Map to **Dataset Variable Definitions Section**

Variable Name	Variable Label	Type	Length/ Display Format	Controlled Terms or Format	Source/Derivation/Comment	Core
STUDYID	Study Identifier	text	20		DM.STUDYID	Req
USUBJID	Unique Subject Identifier	text	40		DM.USUBJID	Req
SUBJID	Subject Identifier for the Study	text	20		DM.SUBJID	Req
SITEID	Study Site Identifier	text	8		DM.SITEID	Req
BRTHDTC	Date/Time of Birth	text	20		DM. BRTHDTC	Req
AGE	Age	integer	8		DM.AGE	Req
AGEU	Age Units	text	8	AGEU: (1) YEARS	DM.AGEU	Req

Predecessor

Map to Controlled Terminology Definitions

AGEGR1	Pooled Age Group 1			Map to Computational Method Definitions Section	Derived: if .<AGE<65 then AGEGR1='<65' else if AGE>=65 then AGEGR1='>=65'	Perm
AGEGR1N	Pooled Age Group 1 (N)	integer	8	AGEGR1N (AGEGR1): (1) 1=<65 (2) 2=>=65	Assigned: 1, if AGEGR1='<65' 2, if AGEGR1='>=65'	Perm
BMI01BL	Basln Bdy Mass Index (kg/m^2) In Stage 1	float	8	5.2	Derived: It is derived from the last non-missing value when VS.VSTESTCD ='BMI' on or before the date of Visit 2.	Perm

Derived

Assigned

Map to Comments Definitions Section for Dataset Variables

Display 4. Variable Information Table in CDISC-Compliant ADaM (ADSL) Specs. Designed for Define-xml v2.0

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

DOMAIN	VAR SEQ	VARIABLE	LABEL	KEY SEQ	DATA TYPE	ORIGIN	COMMENT	LEN GTH	CODELIST	DISPLAY FORMAT	MANDA TORY	VALUE LIST	SIGNIFICANT DIGITS
ADSL	1	STUDYID	Study Identifier		text	Predecessor	DM.STUDYID	20			Yes		.
ADSL	2	USUBJID	Unique Subject Identifier	1	text	Predecessor	DM.USUBJID	40			Yes		.
ADSL	3	SUBJID	Subject Identifier for the Study		text	Predecessor	DM.SUBJID	20			Yes		.
ADSL	4	SITEID	Study Site Identifier		text	Predecessor	DM.SITEID	8			Yes		.
ADSL	5	BRTHDTC	Date/Time of Birth		text	Predecessor	DM. BRTHDTC	20			Yes		.
ADSL	6	AGE	Age		integer	Predecessor	DM.AGE	8			Yes		.
ADSL	7	AGEU	Age Units		text	Predecessor	DM.AGEU	8	AGEU		Yes		.
ADSL	8	AGEGR1	Pooled Age Group 1		text	Derived	If .<AGE<65 then AGEGR1='<65'; Else if AGE>=65 then AGEGR1='>=65'	20	AGEGR1		No		.
ADSL	9	AGEGR1N	Pooled Age Group 1 (N)		integer	Assigned	1, if AGEGR1='<65'; 2, if AGEGR1='>=65'	8	AGEGR1N		No		.
ADSL	93	BMI01BL	Basln Bdy Mass Index (kg/m^2) In Stage 1		float	Derived	It is derived from the last non-missing value when VS.VSTESTCD ='BMI' on or before the date of Visit 2	8		5.2	No		2

Display 5. SAS data VARS_INFO, preparing for Dataset Variables Section in define-xml v2.0

Dataset Variables Section

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

```
<ItemGroupDef OID="IG.ADSL"
…
  <ItemRef ItemOID="IT.ADSL.AGEGR1" OrderNumber="8" Mandatory="NO" MethodOID="MT.ADSL.AGEGR1"/>
  <ItemRef ItemOID="IT.ADSL.AGEGR1N" OrderNumber="9" Mandatory="NO" MethodOID="MT.ADSL.AGEGR1N"/>
  <ItemRef ItemOID="IT.ADSL.BMI01BL" OrderNumber="93" Mandatory="NO" MethodOID="MT.ADSL.BMI01BL"/>
…
<ItemDef OID="IT.ADSL.AGEGR1" Name="AGEGR1" SASFieldName="AGEGR1" DataType="text" Length="20">
  <Description>
     <TranslatedText xml:lang="en">Pooled Age Group 1</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.AGEGR1"/>
  <def:origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADSL.AGEGR1N" Name="AGEGR1N" SASFieldName="AGEGR1N" DataType="integer" Length="8"
def:CommentOID="COM.ADSL.AGEGR1N">
  <Description>
     <TranslatedText xml:lang="en">Pooled Age Group 1 (N)</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.AGEGR1N"/>
  <def:origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADSL.BMI01BL" Name="BMI01BL" SASFieldName="BMI01BL" DataType="float"
```

```
SignificantDigits="2" def:DisplayFormat="5.2" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Basln Bdy Mass Index (kg/m^2) In Stage 1</TranslatedText>
  </Description>
</ItemDef>
```

The code can be automatically generated by SAS with vars_info dataset.

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

  section=50; ** Item definition: define variables by OID, Dataset Variable Section Number ***;
  line= '<ItemDef OID="IT.'||strip(domain) ||'.'||strip(variable)||'" Name="'||strip(variable)||
        '" SASFieldName="'||strip(variable)||'" DataType="'||strip(datatype);
  if not missing(SignificantDigits) then line= trim(line)||'"
SignificantDigits="'||strip(put(significantdigits,best.));
  if not missing(DisplayFormat) then line= trim(line)||'"
def:DisplayFormat="'||strip(DisplayFormat);
  line= trim(line)||'" Length="'||strip(put(length, best.));
  if codelist ne ' ' then do;
     if substr(reverse(strip(codelist)),1,1) = '.' then line = trim(line)||'"
def:DisplayFormat="'||strip(codelist);
  end;
  if propcase(origin) = 'Assigned' then line = trim(line)||'"
def:CommentOID="COM.'||strip(domain)||'.'||strip(variable);
  line = trim(line)||'">'; order+1;output;
  line= '     <Description>';order+1;output;
  line= '          <TranslatedText
xml:lang="en">'||strip(label)||'</TranslatedText>';order+1;output;
  line= '     </Description>';order+1;output;

  if codelist ne ' ' and substr(reverse(strip(codelist)),1,1) ne '.' then do;
     line= '        <CodeListRef
CodeListOID="CL.'||strip(domain)||'.'||strip(codelist)||'"/>';order+1;output;
  end;
  if valuelist ne ' ' then do;
line = '        <def:ValueListRef ValueListOID="VL.'||strip(domain)||'.'||strip(variable)||'"/>';
order+1;output;
  end;

  line= '      <def:Origin Type="'||strip(origin)||'"/>';order+1;output;
  if propcase(origin) not in ('Derived','Assigned') then do;
     line= '        <Description>';order+1;output;
     line= '          <TranslatedText xml:lang="en">'||strip(comment)||'</TranslatedText>';
     order+1;output;
     line= '        </Description>';order+1;output;
     line= '      </def:Origin>';order+1;output;
  end;
  line= '    </ItemDef>';order+1;output;
run;
```

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

Computational Method Definitions Section for Variables

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

```
<MethodDef OID="MT.ADSL.AGEGR1" Name="CM.ADSL.AGEGR1" Type="Computation">
  <Description>
    <TranslatedText xml:lang="en"> If  .&lt;AGE&lt;65 then AGEGR1=&apos;&lt;65&apos;; Else if AGE&gt;=65
then AGEGR1=&apos;&gt;=65&apos;</TranslatedText>
  </Description>
</MethodDef>
```

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

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

An example of Computational Method Definition Section is shown in Display 6. As the algorithm has been linked to the variable ADSL.AGEGR1 when defining ADSL.AGEGR1 by MethodOID attribute in ItemRef element, the derivation rules defined here can also be shown in Source/Derivation/Comment Column of Dataset Variables Section in Display 4.

Analysis Derivations

Method	Type	Description
CM.ADSL.AGEGR1	Computation	If .<AGE<65 then AGEGR1='<65'; Else if AGE>=65 then AGEGR1='>=65'

Display 6. Computational Method Definitions Section for Dataset Variable ADSL.AGEGR1 in define-xml v2.0

Comments Definitions Section for Variables

For assigned variables, the comments at variable level are described by def:CommentDef element as below.

```
<def:CommentDef OID="COM.ADSL.AGEGR1N">
  <Description>
    <TranslatedText>1, if AGEGR1=&apos;&lt;65&apos;; 2, if AGEGR1=&apos;&gt;=65&apos;</TranslatedText>
  </Description>
</def:CommentDef>
```

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

```
data datasets(keep=line order section);
  set vars_info;
  if propcase(origin) = 'Assigned' then do;
     section = 82;        *** Comments for Dataset Variables Section Number ***;
     line= '<def:CommentDef OID="COM.'||strip(domain)||'.'||strip(name)||'">';order+1;output;
     line= '  <Description>';order+1;output;
     line= '     <TranslatedText>'||strip(comment)||'</TranslatedText>';order+1;output;
     line= '  </Description>';order+1;output;
     line= '</def:CommentDef>';order+1;output;
     section = 50;
  end;
run;
```

An example of Comment Definitions Section is shown in Display 7. As the comment has been linked to the variable ADSL.AGEGR1N when defined it by def:CommentOID attribute in ItemRef element, the comments can also be shown in Source/Derivation/Comment Column of Dataset Variables Section in Display 4.

Comments

CommentOID	Description
COM.ADSL.AGEGR1N	1, if AGEGR1='<65'; 2, if AGEGR1='>=65'

Display 7. Comments Definitions Section for Dataset Variable ADSL.AGEGR1N in define-xml v2.0

DESIGN OF CONTROLLED TERMINOLOGY IN ADAM SPECIFICATION FOR DEFINE-XML V2.0

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.

In order to retrieves the controlled terminology information from specification and automatically populate them into Define.XML, the controlled terminology is written in the particular ways, explained below, in "Controlled Terms or Format" column.

Classification of Controlled Terminology

There are three types of controlled terminology in format.

1. **Codelist for decoding purpose**

   Generally, ADaM datasets have code-decode variable pairs, e.g., sponsor-defined controlled terminology AVISITN in Display 8 defines 1:1 mapping of variables **AVISIT** and **AVISITN** for reporting analysis visit windows in the ADaM BDS dataset **ADCSSRS**, where the former variable stores code values and the latter stores decoded values. The code variables in code-decode variable pairs are usually numeric or codes, which are used as a sorting key in SAS programs for tables, figures, and listings (TFLs).

   Fill in 'Controlled Terms or Formats' column for the coded variable only, leave it blank for the decoded variable, write "**codelist name (decoded variable):**" at the beginning, followed by code value, **'='** to link code value and code text, and code text preceded by an ordering number '(#)'. Display 8(a) shows an example of paired variables AVISIT and AVISITN.

   Display 8(b) shows how codelist is displayed in the Controlled Terminology Section of Define-XML for AVISITN-AVISIT pair. The codelists are shown for both coded variable with codelist name 'AVISITN' and decode variable with codelist name 'AVISIT'.

Variable Name	Variable Label	Type	Length/ Display Format	Controlled Terms or Format	Source/Derivation/Comment	Core
AVISIT	Analysis Visit	text	60		Derived: If ABLFL= 'Y' then AVISIT='Baseline'; Else if VISITNUM=4 then AVISIT='V4-Day 15'; Else if VISITNUM=7 then AVISIT='V7-Day 36'; For subjects who did enter Stage 2: If VISITNUM=9 then AVISIT='V9-Day 50'; Else If VISITNUM=12 then AVISIT='V12-Day 71'; Else if VISITNUM=13 and ADSL.EOP02STT='Completed' then AVISIT='V13-Day 78'; Else if VISITNUM=13 and ADSL.EOP02STT='Discontinued' then AVISIT is set to the next scheduled visit; For subjects who did not enter Stage 2: if VISITNUM=13 then AVISIT is set to the next scheduled visit	Cond
AVISITN	Analysis Visit (N)	integer	8	AVISITN **(AVISIT)**: (1) 0 = Baseline (2) 15= V4-Day 15 (3) 36 = V7-Day 36 (4) 50 = V9-Day 50 (5) 71 = V12-Day 71 (6) 78 = V13-Day 78	Assigned: If AVISIT='Baseline' then AVISITN=0; Else if AVISIT='V4-Day 15' then AVISITN=15; Else if AVISIT='V7-Day 36' then AVISITN=36; Else if AVISIT='V9-Day 50' then AVISITN=50; Else if AVISIT='V12-Day 71' then AVISITN=71; Else if AVISIT='V13-Day 78' then AVISITN=78	Perm

(a) An Example of Codelist for Sponsor-defined Controlled Terminology of AVISITN-AVISIT Pair in ADCSSRS

ADCSSRS.AVISIT [CL.ADCSSRS.AVISIT]

Permitted Value (Code)
Baseline
V4-Day 15
V7-Day 36
V9-Day 50
V12-Day 71
V13-Day 78

ADCSSRS.AVISITN [CL.ADCSSRS.AVISITN]

Permitted Value (Code)	Display Value (Decode)
0	Baseline
15	V4-Day 15
36	V7-Day 36
50	V9-Day 50
71	V12-Day 71
78	V13-Day 78

(b) define-xml for sponsor-defined controlled terminology of AVISITN-AVISIT pair with 1:1 mapping

Display 8. An Example of Codelist for Code-Decode purpose – From ADaM specs to define-xml

A special example of codelist for decoding purpose is the codelist required for PARAMCD and PARAM pair in all ADaM BDS Datasets, which describes a set of analysis parameters, helps to determine the unique analysis parameter values in a dataset, and serves as an analysis parameter index and identifiers.

2. Enumerated Codelist

If the controlled terminology of a variable in an ADaM dataset is inherited from an SDTM domain or defined by FDA, and is not used in TFLs SAS programs, there is no need to create a corresponding code variable for it. When writing Controlled Terms or Formats Column, provide the **code list name** with colon sign (:), followed by the individual controlled terms (i.e., code value) which is preceded by a number '(#)'. If no code list name is provided, then use the variable name for code list name. Display 9 shows an example of CDISC code list inherited from an SDTM Domain. In define.xml, all possible values of the variable AEACN in ADAE are listed as **Enumerated items**.

Variable Name	Variable Label	Type	Length/Display Format	Controlled Terms or Format	Source/Derivation/ Comment	Core
AEACN	Action Taken with Study Treatment	text	40	ACN: (1) DOSE NOT CHANGED (2) DRUG WITHDRAWN (3) NOT APPLICABLE	AE.AEACN	

ADAE.ACN [CL.ADAE.ACN]

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

Display 9. An Example of Enumerated Codelist for AEACN – From ADaM specs to define-xml

A special example of enumerated codelist is codelist required for PARCAT1 in all ADaM BDS Datasets, which describes a set of analysis parameters.

3. External Codelist: MedDRA and WHODDE

The sponsor is expected to provide a subsection for external code list references in define.xml, like dictionary name and version, to be used to map the terms. The MedDRA or WHO Drug dictionaries name will be provided in 'Controlled Terms or Format' Column. Display 10 shows an example of the external published source, MedDRA dictionaries from ADaM specs to define.xml.

Variable Name	Variable Label	Type	Length/Display Format	Controlled Terms or Format	Source/Derivation /Comment	Core
AEDECOD	Dictionary-Derived Term	text	100	MedDRA	AE. AEDECOD	Req
AEPTCD	Preferred Term Code	integer	8	MedDRA	AE. AEPTCD	Perm

External Dictionaries

Reference Name	External Dictionary	Dictionary Version
MedDRA (CL.MedDRA)	MedDRA	17.1

Display 10. An Example of External CodeList, MedDRA Dictionary – from ADaM Specs to define-xml

Controlled Terminology Definitions will be stored in SAS data CTLIST_INFO for preparing Controlled Terminology Definitions Section, as shown in Display 11.

SOURCEDATASET	CODELIST	CTORDER	CODEVAL	DECODEVAL	WARNING	DATATYPE	PAIREDVAR	CODE
	MedDRA		MedDRA	17.1		text		
ADAE	ACN	1	DOSE NOT CHANGED			text		C49504
ADAE	ACN	2	DRUG WITHDRAWN			text		C49502
ADAE	ACN	3	NOT APPLICABLE			text		C48660
ADCSSRS	AVISIT	0	Baseline			text	AVISIT	
ADCSSRS	AVISIT	15	V4-Day 15			text	AVISIT	
ADCSSRS	AVISIT	36	V7-Day 36			text	AVISIT	
ADCSSRS	AVISIT	50	V9-Day 50			text	AVISIT	
ADCSSRS	AVISIT	71	V12-Day 71			text	AVISIT	
ADCSSRS	AVISIT	78	V13-Day 78			text	AVISIT	
ADCSSRS	AVISITN	0	0	Baseline		integer	AVISIT	
ADCSSRS	AVISITN	15	15	V4-Day 15		integer	AVISIT	
ADCSSRS	AVISITN	36	36	V7-Day 36		integer	AVISIT	
ADCSSRS	AVISITN	50	50	V9-Day 50		integer	AVISIT	
ADCSSRS	AVISITN	71	71	V12-Day 71		integer	AVISIT	
ADCSSRS	AVISITN	78	78	V13-Day 78		integer	AVISIT	

Display 11. SAS data CTLIST_INFO, Preparing for Controlled Terminology Section in define-xml v2.0

Controlled Terminology Definitions Section

Each code list is defined by a CodeList element outside of ItemDef element, which will be referenced by CodeListRef element with same ODM element identifier (OID).

```
<CodeList OID="CL.ADCSSRS.AVISIT" Name="ADCSSRS.AVISIT" DataType="text">
  <EnumeratedItem CodedValue="Baseline" Rank="0"/>
  <EnumeratedItem CodedValue="V4-Day 15" Rank="15"/>
  <EnumeratedItem CodedValue="V7-Day 36" Rank="36"/>
  <EnumeratedItem CodedValue="V9-Day 50" Rank="50"/>
  <EnumeratedItem CodedValue="V12-Day 71" Rank="71"/>
  <EnumeratedItem CodedValue="V13-Day 78" Rank="78"/>
</CodeList>
<CodeList OID="CL.ADCSSRS.AVISITN" Name="ADCSSRS.AVISITN" DataType="integer">
  <CodeListItem CodedValue="0" OrderNumber="0">
    <Decode>
      <TranslatedText xml:lang="en">Baseline</TranslatedText>
    </Decode>
  </CodeListItem>
  <CodeListItem CodedValue="15" OrderNumber="15">
    <Decode>
      <TranslatedText xml:lang="en">V4-Day 15</TranslatedText>
    </Decode>
  </CodeListItem>
  <CodeListItem CodedValue="36" OrderNumber="36">
    <Decode>
      <TranslatedText xml:lang="en">V7-Day 36</TranslatedText>
    </Decode>
  </CodeListItem>
  <CodeListItem CodedValue="50" OrderNumber="50">
    <Decode>
      <TranslatedText xml:lang="en">V9-Day 50</TranslatedText>
    </Decode>
  </CodeListItem>
  <CodeListItem CodedValue="71" OrderNumber="71">
    <Decode>
      <TranslatedText xml:lang="en">V12-Day 71</TranslatedText>
    </Decode>
  </CodeListItem>
  <CodeListItem CodedValue="78" OrderNumber="78">
    <Decode>
      <TranslatedText xml:lang="en">V13-Day 78</TranslatedText>
    </Decode>
  </CodeListItem>
</CodeList>

<CodeList OID="CL.ADAE.ACN" Name="ADAE.ACN" DataType="text">
  <EnumeratedItem CodedValue="DOSE NOT CHANGED" OrderNumber="1">
    <Alias Name="C49504" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="DRUG WITHDRAWN" OrderNumber="2">
    <Alias Name="C49502" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="NOT APPLICABLE" OrderNumber="3">
    <Alias Name="C48660" Context="nci:ExtCodeID"/>
  </EnumeratedItem>
</CodeList>

<CodeList OID="CL.MedDRA" Name="MedDRA" DataType="text">
  <ExternalCodeList Dictionary="MedDRA" Version="17.1"/>
</CodeList>
```

The code can be automatically generated by SAS with ctlist_info dataset.

```
data ctlist;
  set ctlist_info;
  by sourcedataset codelist ctorder;
```

```
  length line $4000 code $8;
  retain order 0;
  section = 60; *** Section Number for Controlled Terminology except External Codelist ***;
  if first.codelist and sourcedataset = ' ' then do;
    if index(upcase(codelist),"MEDDRA")>0 or index(upcase(codelist),"WHODD")>0  then do;
      section = 65;  *** Section Number for External Codelist ***;
      line= '<CodeList OID="CL.'||strip(codelist)||'" Name="'||strip(codelist)||
            '" DataType="'||strip(datatype)||'">'; order+1;output;
      section = 60;
    end;
    else do;
      line= '<CodeList OID="CL.'||strip(codelist)||'" Name="'||strip(codelist)||
            '" DataType="'||strip(datatype)||'">'; order+1;output;
    end;
  end;
  else if first.codelist and sourcedataset ne ' ' then do;
    line= '<CodeList OID="CL.'||strip(sourcedataset)||'.'||strip(codelist)||'" Name="'||
          strip(sourcedataset)||'.'||strip(codelist)||'" DataType="'||strip(datatype)||'">';
          order+1;output;
  end;

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

  if last.codelist then do;
    if index(upcase(codelist),"MEDDRA")>0 or index(upcase(codelist),"WHODD")>0 then section=65;
    line='</CodeList>';order+1;output;
    section = 60;
  end;
run;
```

If variable LINE generated above is output to an xml file sorting by SECTION and ORDER, the Controlled Terminology Definitions Section will be shown in Display 8 – 10.

DESIGN OF VALUE LEVEL METADATA IN ADAM SPECIFICATION FOR DEFINE-XML V2.0

The definition of a variable for a specific condition is known as Parameter Value Level Metadata, which defines metadata for a dataset variable under a specific condition. In Define-XML v2.0, Parameter Value Level Metadata provides sufficient detail to support data review and analysis, where the dataset variable metadata does not. Usually it is not required to provide Value Level Metadata if **DataType** and **Length** attributes of each variable value are the same, unless it is desired to provide the metadata. There are typically four cases: **AVAL, PARAMTYP, DTYPE**, and **ASEQ**. We will illustrate how to write the ADaM specification to achieve the automation of generating Parameter Value Level Metadata in Define.XML for these variables.

Parameter Value Level Metadata for AVAL

In ADaM, parameter value level metadata often describes AVAL or AVALC in BDS data structures based on values of PARAMCD. In Define-XML v2.0 Parameter Value Level Metadata is viewed as value lists, which describes all of the possible definitions of the contents of a variable when different conditions hold true, for example definitions and

different data types of AVAL when PARAMCD = “BMI” or when PARAMCD = “DIABP”. Another example is different Display Formats of AVAL when PARAMCD = “BMI” or when PARAMCD = “WEIGHT”. Display 12(a) shows the specification of AVAL for ADVS. The key word: Value Level Metadata in column Source/Derivation/Comment divides the cell into two parts. The first part will be presented in AVAL Variable Section in Define-XML v2.0, shown in Display 12(b). The second part will be displayed in Parameter Value Level Metadata Section in Define-XML v2.0, shown in Display 12(c). The Value Level Metadata is shown as Value Lists.

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

Write “Value Level Metadata:” at the beginning, followed by each individual values of PARAMCD and corresponding PARAM preceded by an ordering number ‘(#)’, which specifies the ordering in Parameter Value Level Metadata for AVAL in Define-XML.

Variable Name	Variable Label	Type	Length/ Display Format	Controlled Terms or Format	Source/Derivation/Comment	Core
AVAL	Analysis Value	float	8	8.1 Value List Definitions Section	Derived: Equals to VS.VSSTRESN **Value Level Metadata:** (1) BMI= Body Mass Index (kg/m^2): float 5.2CTDerived$VS.VSSTRESN, populated for every visit where WEIGHT is collected; (2) DIABP= Diastolic Blood Pressure (mmHg): integerCTDerived$VS.VSSTRESN; (3) HEIGHT = Height (cm): float 5.1CTDerived$VS.VSSTRESN; (4) HR = Heart Rate (BEATS/MIN): integerCTDerived$VS.VSSTRESN; (5) RESP = Respiratory Rate (BREATHS/MIN): integerCTDerived$VS.VSSTRESN; (6) SYSBP = Systolic Blood Pressure (mmHg): integerCTDerived$VS.VSSTRESN; (7) TEMP = Temperature (C): float 4.1CTDerived$VS.VSSTRESN; (8) WEIGHT = Weight (kg): float 5.1CTDerived$VS.VSSTRESN.	Req

(a) An example of AVAL in ADVS specs designed for Parameter Value Level Metadata define-xml v2.0

Vital Signs Analysis Dataset (ADVS) [Location: ADVS.xpt]

Variable	Label	Type	Length / Display Format	Controlled Terms or Format	Source/Derivation/Comment
AVAL	Analysis Value	float	8.1		Derived: Equals to VS.VSSTRESN;

(b) Dataset Variable Section for ADVS.AVAL in Define.XML v2.0 (First Part of Source/Derivation/ Comment).

Parameter Value List - ADVS [AVAL]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
AVAL	PARAMCD EQ BMI (Body Mass Index (kg/m^2))	float	5.2		Derived	VS.VSSTRESN, populated for every visit where WEIGHT is collected;
AVAL	PARAMCD EQ DIABP (Diastolic Blood Pressure (mmHg))	integer	8		Derived	VS.VSSTRESN;
AVAL	PARAMCD EQ HEIGHT (Height (cm))	float	5.1		Derived	VS.VSSTRESN;
AVAL	PARAMCD EQ HR (Heart Rate (BEATS/MIN))	integer	8		Derived	VS.VSSTRESN;
AVAL	PARAMCD EQ RESP (Respiratory Rate (BREATHS/MIN))	integer	8		Derived	VS.VSSTRESN;
AVAL	PARAMCD EQ SYSBP (Systolic Blood Pressure (mmHg))	integer	8		Derived	VS.VSSTRESN;
AVAL	PARAMCD EQ TEMP (Temperature (C))	float	4.1		Derived	VS.VSSTRESN;
AVAL	PARAMCD EQ WEIGHT (Weight (kg))	float	5.1		Derived	VS.VSSTRESN.

(c) Parameter Value List for ADVS.AVAL in Define.XML v2.0 (Second Part of Source/Derivation/Comment).

Display 12. Value List Definitions Section in CDISC-Compliant ADaM (ADVS.AVAL) Specifications Designed for Define-xml v2.0

Table 3 illustrates the syntax for the Value Level Metadata for AVAL by an example of “BMI= Body Mass Index (kg/m^2): float 5.2CTDerived$VS.VSSTRESN, populated for every visit where WEIGHT is collected;”

Condition Used to Retrieved Words	Retrieved Words	The Syntax Created by SAS Macro Tool	Column Displayed in DEFINE.XML
:	BMI=BODY MASS INDEX (kg/m2)	PARAMCD EQ BMI (BODY MASS INDEX (kg/m2))	Where
first $, first word	float	float	Type
first $, second word	5.2	5.2	Length/Display Format
Second $	CT		Controlled Terms or Format
Third $	Derived	Derived	Origin
Words after last $	VSSTRESN, populated for every visit where WEIGHT is collected;	VSSTRESN, populated for every visit where WEIGHT is collected;	Derivation/Comment

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

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

Another way to write the specification for AVAL, which makes it possible to automatically retrieve elements for Value Level Metadata, is shown below. In order to facilitate the data manipulation**, if – then clause** will be used for generating Value Level Metadata. The variable written after if clause should be a variable in ADaM metadata based on the value of which AVAL is defined, and comparator after the variable should be 'EQ', 'IN', 'NOTIN', 'NE','LT','LE','GT',or 'GE'. The mapping from ADaM specification ADHAMA.AVAL to define-xml is shown in Display 13, in which ADHAMA (HAM-A Questionnaire Analysis Dataset) is an ADaM dataset for a standard questionnaire that is typically used in clinical trials to provide an assessment of anxiety symptoms.

HAM-A Questionnaire Analysis Dataset (ADHAMA) [Location: ADHAMA.xpt]

Variable	Label	Type	Length / Display Format	Controlled Terms or Format	Source/Derivation/Comment
AVAL	Analysis Value	integer	8		Derived: For the records from QS, AVAL=QS.QSSTRESN; For the total score, a missed post-baseline record will be imputed by LOCF within each stage.

Variable Name	Variable Label	Type	Length/ Display Format	Controlled Terms or Format	Source/Derivation/Comment	Core
AVAL	Analysis Value	integer	8		Derived: For the records from QS, AVAL=QS.QSSTRESN For the total, a missed post-baseline record will be imputed by LOCF within each stage. **Value Level Metadata:** (1) if PARAMCD NE 'TOTHAMA' then AVAL = QS.QSSTRESN; (2) if PARAMCD = 'TOTHAMA' then AVAL is the summation of the 14 items within each visit if neither of all 14 items is missing; AVAL is (the summation of the 13 non-missing items)*14/13 within each visit if 13 of 14 items are non-missing; AVAL is set to missing if more than one of the items are missing	Req

Parameter Value List - ADHAMA [AVAL]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
AVAL	PARAMCD NE TOTHAMA		8		Derived	AVAL = QS.QSSTRESN;
AVAL	PARAMCD EQ TOTHAMA (Re-calculated HAMA1-Total Score)		8		Derived	AVAL is the summation of the 14 items within each visit if neither of all 14 items is missing; AVAL is (the summation of the 13 non-missing items)*14/13 within each visit if 13 of 14 items are non-missing; AVAL is set to missing if more than one of the items are missing

order=1

order=2

Display 13. Mapping from Specification to define-xml for AVAL in ADHAMA.

This example illustrates how to define metadata for AVAL based on the value of the PARAMCD variable in an ADaM Basic Data Structure dataset.

Parameter Value Level Metadata for PARAMTYP

PARAMTYP (Parameter Type) is used to indicate that an entire parameter is derived as a function of one or more other parameters in programming ADaM BDS datasets for traceability. The CDISC/NCI Controlled Terminology for PARAMTYP is shown in Appendix 2. Its CDISC submission value has unique one: "DERIVED". The way to write the ADaM specification for PARAMTYP and its display in ADaM Define.XML are demonstrated below.

ADHAMA analysis dataset has 2 data blocks. The first one is inherited from SDTM QS domain which includes all HAM-A questionnaires. Records in this data block establish traceability for derivation of at each analysis visit. Hence the value for variable PARAMTYP for these records is set to blank. The second block consists of the summation of the 14 HAM-A items within each visit. Its value for variable PARAMCD is set to TOTHAMA, labeled as "Re-calculated HAMA1-Total Score". The specification for PARAMTYP is written as below. The mapping from ADaM specification to define-xml is shown in Display 14.

Variable Name	Variable Label	Type	Length/Display Format	Controlled Terms or Format	Source/Derivation/Comment	Core
PARAMTYP	Parameter Type	Text	10	(1) DERIVED	Assigned: PARAMTYP = 'DERIVED' for the records with PARAMCD = 'TOTHAMA' **Value Level Metadata:** (1) if PARAMCD = 'TOTHAMA' then PARAMTYP = 'DERIVED'; (2) if PARAMCD NE ('TOTHAMA') then PARAMTYP = null	Perm

Parameter Value List - ADHAMA [PARAMTYP]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
PARAMTYP	PARAMCD EQ TOTHAMA (Re-calculated HAMA1-Total Score)	text	10	["DERIVED"] <ADHAMA.PARAMTYP>	Assigned	PARAMTYP = 'DERIVED';
PARAMTYP	PARAMCD NE TOTHAMA	text	10		Assigned	PARAMTYP = null

Display 14. Mapping from ADHAMA Specification to Define-xml for PARAMTYP.

Parameter Value Level Metadata for DTYPE

DTYPE should be used to indicate records that are derived within a given value of PARAM in ADaM Basic Data Structure (BDS). DTYPE specifies the method used to derive the added records, and indicates that those records were derived. Appendix 3 shows CDISC /NCI Controlled Terminology for DTYPE. Display 15 provides an example of writing ADLB specification for DTYPE in order to automatically generate value level metadata for DTYPE in Define.XML.

Variable Name	Variable Label	Type	Length/ Display Format	Controlled Terms or Format	Source/Derivation/Comment	Core
DTYPE	Derivation Type	text	10	(1) LOV	Assigned: DTYPE = 'LOV' for the records with AVISIT in ('Last Scheduled Post-baseline Observation during Stage 1 Safety Period', 'Last Scheduled Post-baseline Observation during Stage 2 Safety Period') Value Level Metadata: (1) if AVISITN not in ('Last Scheduled Post-baseline Observation during Stage 1 Safety Period', 'Last Scheduled Post-baseline Observation during Stage 2 Safety Period') then DTYPE=null; (2) if AVISIT in ('Last Scheduled Post-baseline Observation during Stage 1 Safety Period', 'Last Scheduled Post-baseline Observation during Stage 2 Safety Period') then DTYPE='LOV'	Cond

Parameter Value List - ADVS [DTYPE]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
DTYPE	AVISITN NOTIN ("Last Scheduled Post-baseline Observation during Stage 1 Safety Period" , "Last Scheduled Post-baseline Observation during Stage 2 Safety Period")	text	10		Assigned	DTYPE=null;
DTYPE	AVISIT IN ("Last Scheduled Post-baseline Observation during Stage 1 Safety Period" , "Last Scheduled Post-baseline Observation during Stage 2 Safety Period")	text	10	["LOV"] <ADVS.DTYPE>	Assigned	DTYPE='LOV'

Display 15. Mapping from ADLB Specification to Define-xml for DTYPE.

Parameter Value Level Metadata for ASEQ

When new records are added to the ADaM datasets in the ADaM Basic Data Structure (BDS), variable DTYPE and/or PARAMTYP are added into the dataset. Another variable **named ASEQ** (Analysis Sequence Number) is also

derived for traceability and further easy **identification of different blocks** (inherited from SDTM and newly derived), and it also serves as a sorting key when combined with USUBJID.

ADCSSRS (Columbia Suicide Severity Rating Scale Analysis Dataset) is an ADaM dataset for safety. It follows the BDS model and includes all CSSRS assessments from SDTM QS domain. There are 3 data blocks in ADCSSRS. One is collected raw values inherited from SDTM QS domain with PARAMCD begin with CSS; the second one, Preparatory Acts Toward Imminent Suicidal Behavior, is derived values with PARAMCD CSS0219, CSS0217, and CSS0215; and the third one, Suicidal Ideation, was derived by PARAMCD CSS0201, CSS0202, CSS0203, CSS0204, and CSS0205.

For traceability and easy identification of each of these blocks, variable ASEQ (Analysis Sequence Number) is created in addition to QS.QSSEQ.

Display 16 shows the specification of these two variables, and how the variables to be shown in define-xml.

Variable Name	Variable Label	Type	Length/ Display Format	Controlled Terms or Format	Comments	Core
ASEQ	Analysis Sequence Number	integer	8		Derived: Unique within USUBJID. **Value Level Metadata:** If PARAMCD ne ('CSS0288', 'CSS0299') then ASEQ = QS.QSSEQ If PARAMCD='CSS0288' then ASEQ = QS.QSSEQ + 10000 If PARAMCD='CSS0299' then ASEQ = QS.QSSEQ + 20000	Perm
QSSEQ	Sequence Number	integer	8		QS.QSSEQ	Perm

Parameter Value List - ADCSSRS [ASEQ]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
ASEQ	PARAMCD NOTIN ("CSS0288" (Preparatory Acts Toward Imminent Suicidal Behavior) , "CSS0299" (Suicidal Ideation))	integer	8		Derived	ASEQ = QS.QSSEQ;
ASEQ	PARAMCD EQ CSS0288 (Preparatory Acts Toward Imminent Suicidal Behavior)	integer	8		Derived	ASEQ = QS.QSSEQ + 10000;
ASEQ	PARAMCD EQ CSS0299 (Suicidal Ideation)	integer	8		Derived	ASEQ = QS.QSSEQ + 20000

Display 16. Mapping from ADCSSRS Specification to Define-xml for ASEQ.

The ranges of these two variables are displayed in table 4.

PARAMCD	ASEQ Value Range	QSSEQ Value Range
CSS0201, CSS0201A, CSS0202, CSS0202A, CSS0203, CSS0203A, CSS0204, CSS0204A, CSS0205, CSS0205A, CSS0206, CSS0206A, CSS0207, CSS0208, CSS0209, CSS0210, CSS0211, CSS0212, CSS0213, CSS0213A, CSS0214, CSS0215, CSS0216, CSS0216A, CSS0217, CSS0218, CSS0218A, CSS0219, CSS0219A, CSS0220, CSS0221, CSS0222A, CSS0222B, CSS0222C	1-1063	1-1063
CSS0288	10090-11051	Missing
CSS0299	20069-21030	Missing

Table 4. An Example of Comparison of ASEQ and QSSEQ for ADCSSRS Dataset.

The value of ASEQ can tell which data block the record shall belong to. The new sorting keys for ADCSSRS are USUBJID ASEQ.

Parameter Value List will be stored in SAS data VALUE_INFO for preparing Parameter Value Level Definitions Section, as shown in Display 17.

SOURCE DATASET	SOURCE VARIABLE	SOURCE LABEL	VORDER	WHEREVARIABLE	WHERECOMPARATOR	WHERECHECKVALUE	DATA TYPE	ORIGIN	COMMENT	LENGTH	CODE LIST	DISPLAYFORMAT	SIGNIFICANTDIGITS	MANDATORY
ADCSSRS	ASEQ	Analysis Sequence Number	1	PARAMCD	NOTIN	CSS0288	integer	Derived	ASEQ = QS.QSSEQ;	8			.	No
ADCSSRS	ASEQ	Analysis Sequence Number	1	PARAMCD	NOTIN	CSS0299	integer	Derived	ASEQ = QS.QSSEQ;	8			.	No
ADCSSRS	ASEQ	Analysis Sequence Number	2	PARAMCD	EQ	CSS0288	integer	Derived	ASEQ = QS.QSSEQ + 10000;	8			.	No
ADCSSRS	ASEQ	Analysis Sequence Number	3	PARAMCD	EQ	CSS0299	integer	Derived	ASEQ = QS.QSSEQ + 20000	8			.	No
ADVS	AVAL	Analysis Value	1	PARAMCD	EQ	BMI	float	Derived	VS.VSSTRESN, populated for every visit where WEIGHT is collected;	8		5.2	2	Yes
ADVS	AVAL	Analysis Value	2	PARAMCD	EQ	DIABP	integer	Derived	VS.VSSTRESN;	8			.	Yes
ADVS	AVAL	Analysis Value	3	PARAMCD	EQ	HEIGHT	float	Derived	VS.VSSTRESN;	8		5.1	1	Yes
ADVS	AVAL	Analysis Value	4	PARAMCD	EQ	HR	integer	Derived	VS.VSSTRESN;	8			.	Yes
ADVS	AVAL	Analysis Value	5	PARAMCD	EQ	RESP	integer	Derived	VS.VSSTRESN;	8			.	Yes
ADVS	AVAL	Analysis Value	6	PARAMCD	EQ	SYSBP	integer	Derived	VS.VSSTRESN;	8			.	Yes
ADVS	AVAL	Analysis Value	7	PARAMCD	EQ	TEMP	float	Derived	VS.VSSTRESN;	8		4.1	1	Yes
ADVS	AVAL	Analysis Value	8	PARAMCD	EQ	WEIGHT	float	Derived	VS.VSSTRESN.	8		5.1	1	Yes

Display 17. SAS data value_info, preparing for Value Level Metadata Definitions Section in define-xml v2.0

Value Level Metadata Definitions Section

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

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

```
<!-- Value List Definitions Section.                                    -->
<def:ValueListDef OID="VL.ADCSSRS.ASEQ">
  <ItemRef ItemOID="IT.ADCSSRS.ASEQ.VAL1" Mandatory="No" MethodOID="MT.ADCSSRS.ASEQ.VAL1">
    <def:WhereClauseRef WhereClauseOID="WC.ADCSSRS.ASEQ.VAL1"/>
  </ItemRef>
  <ItemRef ItemOID="IT.ADCSSRS.ASEQ.VAL2" Mandatory="No" MethodOID="MT.ADCSSRS.ASEQ.VAL2">
    <def:WhereClauseRef WhereClauseOID="WC.ADCSSRS.ASEQ.VAL2"/>
  </ItemRef>
  <ItemRef ItemOID="IT.ADCSSRS.ASEQ.VAL3" Mandatory="No" MethodOID="MT.ADCSSRS.ASEQ.VAL3">
    <def:WhereClauseRef WhereClauseOID="WC.ADCSSRS.ASEQ.VAL3"/>
  </ItemRef>
</def:ValueListDef>

<def:ValueListDef OID="VL.ADVS.AVAL">
  <ItemRef ItemOID="IT.ADVS.AVAL.VAL1" Mandatory="Yes" MethodOID="MT.ADVS.AVAL.VAL1">
    <def:WhereClauseRef WhereClauseOID="WC.ADVS.AVAL.VAL1"/>
  </ItemRef>
  <ItemRef ItemOID="IT.ADVS.AVAL.VAL2" Mandatory="Yes" MethodOID="MT.ADVS.AVAL.VAL2">
    <def:WhereClauseRef WhereClauseOID="WC.ADVS.AVAL.VAL2"/>
  </ItemRef>
  <ItemRef ItemOID="IT.ADVS.AVAL.VAL3" Mandatory="Yes" MethodOID="MT.ADVS.AVAL.VAL3">
    <def:WhereClauseRef WhereClauseOID="WC.ADVS.AVAL.VAL3"/>
  </ItemRef>
  <ItemRef ItemOID="IT.ADVS.AVAL.VAL4" Mandatory="Yes" MethodOID="MT.ADVS.AVAL.VAL4">
    <def:WhereClauseRef WhereClauseOID="WC.ADVS.AVAL.VAL4"/>
  </ItemRef>
  <ItemRef ItemOID="IT.ADVS.AVAL.VAL5" Mandatory="Yes" MethodOID="MT.ADVS.AVAL.VAL5">
    <def:WhereClauseRef WhereClauseOID="WC.ADVS.AVAL.VAL5"/>
  </ItemRef>
  <ItemRef ItemOID="IT.ADVS.AVAL.VAL6" Mandatory="Yes" MethodOID="MT.ADVS.AVAL.VAL6">
    <def:WhereClauseRef WhereClauseOID="WC.ADVS.AVAL.VAL6"/>
```

```
  </ItemRef>
  <ItemRef ItemOID="IT.ADVS.AVAL.VAL7" Mandatory="Yes" MethodOID="MT.ADVS.AVAL.VAL7">
    <def:WhereClauseRef WhereClauseOID="WC.ADVS.AVAL.VAL7"/>
  </ItemRef>
  <ItemRef ItemOID="IT.ADVS.AVAL.VAL8" Mandatory="Yes" MethodOID="MT.ADVS.AVAL.VAL8">
    <def:WhereClauseRef WhereClauseOID="WC.ADVS.AVAL.VAL8"/>
  </ItemRef>
</def:ValueListDef>

<!-- WhereClause Definitions Section -->
<def:WhereClauseDef OID="WC.ADCSSRS.ASEQ.VAL1">
  <RangeCheck Comparator="NOTIN" SoftHard="Soft" def:ItemOID="IT.ADCSSRS.PARAMCD">
    <CheckValue>CSS0288</CheckValue>
    <CheckValue>CSS0299</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.ADCSSRS.ASEQ.VAL2">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADCSSRS.PARAMCD">
    <CheckValue>CSS0288</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.ADCSSRS.ASEQ.VAL3">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADCSSRS.PARAMCD">
    <CheckValue>CSS0299</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>

<def:WhereClauseDef OID="WC.ADVS.AVAL.VAL1">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADVS.PARAMCD">
    <CheckValue>BMI</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.ADVS.AVAL.VAL2">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADVS.PARAMCD">
    <CheckValue>DIABP</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.ADVS.AVAL.VAL3">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADVS.PARAMCD">
    <CheckValue>HEIGHT</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.ADVS.AVAL.VAL4">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADVS.PARAMCD">
    <CheckValue>HR</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.ADVS.AVAL.VAL5">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADVS.PARAMCD">
    <CheckValue>RESP</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.ADVS.AVAL.VAL6">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADVS.PARAMCD">
    <CheckValue>SYSBP</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.ADVS.AVAL.VAL7">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADVS.PARAMCD">
    <CheckValue>TEMP</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>
<def:WhereClauseDef OID="WC.ADVS.AVAL.VAL8">
  <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADVS.PARAMCD">
    <CheckValue>WEIGHT</CheckValue>
```

```
  </RangeCheck>
</def:WhereClauseDef>

<ItemDef OID="IT.ADCSSRS.ASEQ.VAL1" Name="ASEQ" SASFieldName="ASEQ" DataType="integer" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Analysis Sequence Number</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADCSSRS.ASEQ.VAL2" Name="ASEQ" SASFieldName="ASEQ" DataType="integer" Length="8">
<Description>
    <TranslatedText xml:lang="en">Analysis Sequence Number</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADCSSRS.ASEQ.VAL3" Name="ASEQ" SASFieldName="ASEQ" DataType="integer" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Analysis Sequence Number</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>

<ItemDef OID="IT.ADVS.AVAL.VAL1" Name="AVAL" SASFieldName="AVAL" DataType="float" SignificantDigits="2"
def:DisplayFormat="5.2" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Analysis Value</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADVS.AVAL.VAL2" Name="AVAL" SASFieldName="AVAL" DataType="integer" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Analysis Value</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADVS.AVAL.VAL3" Name="AVAL" SASFieldName="AVAL" DataType="float" SignificantDigits="1"
def:DisplayFormat="5.1" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Analysis Value</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADVS.AVAL.VAL4" Name="AVAL" SASFieldName="AVAL" DataType="integer" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Analysis Value</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADVS.AVAL.VAL5" Name="AVAL" SASFieldName="AVAL" DataType="integer" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Analysis Value</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADVS.AVAL.VAL6" Name="AVAL" SASFieldName="AVAL" DataType="integer" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Analysis Value</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADVS.AVAL.VAL7" Name="AVAL" SASFieldName="AVAL" DataType="float" SignificantDigits="1"
def:DisplayFormat="4.1" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Analysis Value</TranslatedText>
  </Description>
```

```
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.ADVS.AVAL.VAL8" Name="AVAL" SASFieldName="AVAL" DataType="float" SignificantDigits="1"
def:DisplayFormat="5.1" Length="8">
  <Description>
    <TranslatedText xml:lang="en">Analysis Value</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
```

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

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

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

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

```
  if last.sourcevariable then do;
     line = '</def:ValueListDef>';order+1;output;
  end;
run;
```

If variable LINE generated above is output to an xml file sorting by SECTION and ORDER, the Value Level Metadata Definitions Section will be shown in Display 12 – 16.

Computational Method Definitions Section for Values

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

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

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

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

An example of Computational Method Definition Section for variable values is shown in Display 18. As the algorithm has been linked to the variable value ADVSL.AVAL.VAL1 when defining it by MethodOID attribute in ItemRef element, the derivation rules defined here can also be shown in Derivation/Comment Column of Value Level Definitions Section in Display 12-16.

Analysis Derivations

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

Display 18. Computational Method Section for Variable Value ADVSL.AVAL.VAL1 in define-xml v2.0

Comments Definitions Section for Values

For assigned variables, the comments at value level are described by def:CommentDef element as below.

```
<def:CommentDef OID="COM.ADHAMA.PARAMTYP.VAL1">
 <Description>
  <TranslatedText>PARAMTYP = &amp;apos;DERIVED&amp;apos;;</TranslatedText>
 </Description>
</def:CommentDef>
```

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

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

```
      line = '  <Description>';order+1;output;
      line = '    <TranslatedText>'||strip(comment)||'</TranslatedText>';order+1;output;
      line = '  </Description>';order+1;output;
      line = '</def:CommentDef>';order+1;output;
    end;
  end;
run;
```

An example of Comment Definitions Section for Variable Values is shown in Display 19. As the comment has been linked to the variable value ADHAMA.PARAMTYP.VAL1 when defined it by def:CommentOID attribute in ItemRef element, the comments can also be shown in Derivation/Comment Column of Value Level Definitions Section in Display 12-16.

Comments

CommentOID	Description
COM.ADHAMA.PARAMTYP.VAL1	PARAMTYP = 'DERIVED';

Display 19. Comments Definitions Section for Variable Value ADHAMA.PARAMTYP.VAL1 in define-xml v2.0

CONCLUSION

This paper introduces a SAS Macro Tool to automatically pull metadata from ADaM Specification in MS Word® format and automatically uses the metadata to create Define-XML 2.0 for FDA submission. This automation of creating define.xml from ADaM Specification, significantly reduces time and the resources needed for preparation, and guarantees the high quality of the submission. With this tool, you do not need to be an ODM expert for a define-xml. We hope the methodology and the SAS codes provided in this paper can help you in saving your time and resources for FDA submission.

REFERENCES

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

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

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

3. CDISC Analysis Data Model Team. “Analysis Data Model (ADaM) Implementation Guide”. December 2009. http://www.cdisc.org/adam

4. Xiangchen (Bob) Cui, Min Chen, and Tathabbai Pakalapati. “An Innovative ADaM Programming Tool for FDA Submission”, PharmaSUG, May 2012.

5. Min Chen, Xiangchen Cui, Scott Moseley. (2011) ”Automating the Process of Preparing Data Definition Document for NDA Electronic Submission from Programming Specification in Word Format”, PharmaSUG, May 2011.

6. Xiangchen (BoB) Cui, Min Chen. “Automatic Consistency Checking of Controlled Terminology and Value Level Metadata between ADaM Datasets and Define.xml”, SAS Global Forum, April 2012.

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

CONTACT INFORMATION

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

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

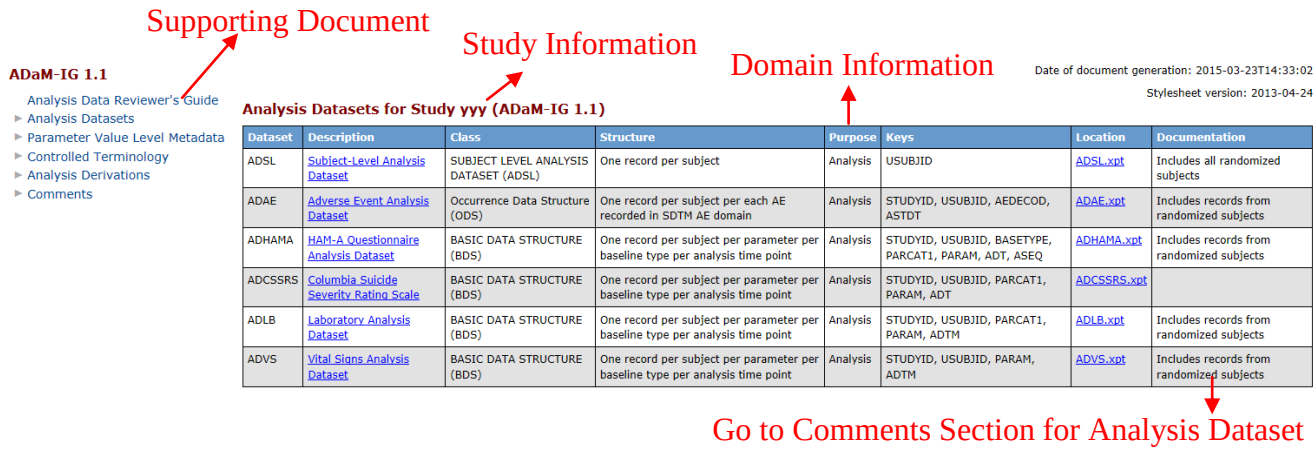
Name: Min Chen**,** Ph.D.
Enterprise: Alkermes, Inc.
Address: 852 Winter Street
City, State ZIP: Waltham, MA 02451
Work Phone: 781-609-6047
Fax: 781-609-5855
E-mail: min.chen@alkermes.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Appendix 1

An example of define-xml v2.0



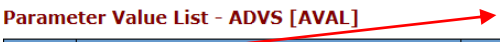
Dataset	Description	Class	Structure	Purpose	Keys	Location	Documentation
ADSL	Subject-Level Analysis Dataset	SUBJECT LEVEL ANALYSIS DATASET (ADSL)	One record per subject	Analysis	USUBJID	ADSL.xpt	Includes all randomized subjects
ADAE	Adverse Event Analysis Dataset	Occurrence Data Structure (ODS)	One record per subject per each AE recorded in SDTM AE domain	Analysis	STUDYID, USUBJID, AEDECOD, ASTDT	ADAE.xpt	Includes records from randomized subjects
ADHAMA	HAM-A Questionnaire Analysis Dataset	BASIC DATA STRUCTURE (BDS)	One record per subject per parameter per baseline type per analysis time point	Analysis	STUDYID, USUBJID, BASETYPE, PARCAT1, PARAM, ADT, ASEQ	ADHAMA.xpt	Includes records from randomized subjects
ADCSSRS	Columbia Suicide Severity Rating Scale	BASIC DATA STRUCTURE (BDS)	One record per subject per parameter per baseline type per analysis time point	Analysis	STUDYID, USUBJID, PARCAT1, PARAM, ADT	ADCSSRS.xpt	
ADLB	Laboratory Analysis Dataset	BASIC DATA STRUCTURE (BDS)	One record per subject per parameter per baseline type per analysis time point	Analysis	STUDYID, USUBJID, PARCAT1, PARAM, ADTM	ADLB.xpt	Includes records from randomized subjects
ADVS	Vital Signs Analysis Dataset	BASIC DATA STRUCTURE (BDS)	One record per subject per parameter per baseline type per analysis time point	Analysis	STUDYID, USUBJID, PARAM, ADTM	ADVS.xpt	Includes records from randomized subjects

(a) Study Information, Domain Information, and Supporting Document Section of define-xml v2.0

Subject-Level Analysis Dataset (ADSL) [Location: ADSL.xpt]

Variable	Label	Type	Length / Display Format	Controlled Terms or Format	Source/Derivation/Comment
STUDYID	Study Identifier	text	20		Predecessor: DM.STUDYID
USUBJID	Unique Subject Identifier	text	40		Predecessor: DM.USUBJID
SUBJID	Subject Identifier for the Study	text	20		Predecessor: DM.SUBJID
SITEID	Study Site Identifier	text	8		Predecessor: DM.SITEID
BRTHDTC	Date/Time of Birth	text	20		Predecessor: DM. BRTHDTC
AGE	Age	integer	8		Predecessor: DM.AGE
AGEU	Age Units	text	8	["YEARS"] <ADSL.AGEU>	Predecessor: DM.AGEU
AGEGR1	Pooled Age Group 1	text	20	["<65", ">=65"] <AGEGR1>	Derived: Go to Computational Method Section for Derived Variables If .<AGE<65 then AGEGR1='<65'; Else if AGE>=65 then AGEGR1='>=65'
AGEGR1N	Pooled Age Group 1 (N)	integer	8	["1" = "<65", "2" = ">=65"] <AGEGR1N>	Assigned: 1, if AGEGR1='<65'; 2, if AGEGR1='>=65' Go to Comments Section for Assigned Variables

(b) Variable Information Section of define-xml v2.0



Parameter Value List - ADVS [AVAL]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
AVAL	PARAMCD EQ BMI (Body Mass Index (kg/m^2))	float	5.2		Derived	VS.VSSTRESN, populated for every visit where WEIGHT is collected;
AVAL	PARAMCD EQ DIABP (Diastolic Blood Pressure (mmHg))	integer	8		Derived	VS.VSSTRESN;
AVAL	PARAMCD EQ HEIGHT (Height (cm))	float	5.1		Derived	VS.VSSTRESN;

(c) Parameter Value Lists Section of define-xml v2.0

Go to Computational Method Section for Variable Values (Value Level Metadata)

ADAE.OUT [CL.ADAE.OUT]

Enumerated items

Permitted Value (Code)
NOT RECOVERED/NOT RESOLVED [*C49494*]
RECOVERED/RESOLVED [*C49498*]
RECOVERED/RESOLVED WITH SEQUELAE [*C49495*] → NCI/CDISC Controlled Terminology Code
FATAL [*C48275*]

ADAE.OUTN [CL.ADAE.OUTN]

Code List

Permitted Value (Code)	Display Value (Decode)
1	NOT RECOVERED/NOT RESOLVED
2	RECOVERED/RESOLVED
3	RECOVERED/RESOLVED WITH SEQUELAE
4	FATAL

(d) Controlled Terminology Section of define-xml v2.0

Analysis Derivations

Method	Type	Description
CM.ADVS.AVAL	Computation	Equals to VS.VSSTRESN;
CM.ADVS.ADT	Computation	Input(substr(VS.VSDTC,
CM.ADVS.AVAL.VAL1	Computation	VS.VSSTRESN, populated for every visit where WEIGHT is collected;
CM.ADVS.AVAL.VAL2	Computation	VS.VSSTRESN;

Computational Method Definitions Section for Derived Variables

Computational Method Section for Value Level Metadata

(e) Computational Method Definitions Section of define-xml v2.0

Comments

CommentOID	Description
COM.ADAE	Includes records from randomized subjects
COM.ADHAMA.PARAMTYP	PARAMTYP = 'DERIVED' for the rec
COM.ADHAMA.PARAMTYP.VAL1	PARAMTYP = 'DERIVED';
COM.ADHAMA.PARAMTYP.VAL2	PARAMTYP = null

Comment Section for Domain

Comment Section for Assigned Variables

Comment Section for Assigned Variable Values

(f) Comments Section of define-xml v2.0

Display 20 An Example of define-xml v2.0

Appendix 2

CDISC / NCI Controlled Terminology for PARAMTYP

Code	Codelist Code	Codelist Extensible (Yes/No)	Codelist Name	CDISC Submission Value	CDISC Synonym(s)	CDISC Definition	NCI Preferred Term
C81225		No	Parameter Type	PARAMTYP	Parameter Type	Parameter Type: Indicates whether the parameter is derived as a function of one or more other parameters.	CDISC ADaM Parameter Type Terminology
C81197	C81225		Parameter Type	DERIVED		Derived: Indicates that a parameter is derived as a function of one or more other parameters.	Derived Flag

Appendix 3

CDISC / NCI Controlled Terminology for DTYPE

Code	**Codelist Code**	**Codelist Extensible (Yes/No)**	**Codelist Name**	**CDISC Submission Value**	**CDISC Definition**	**NCI Preferred Term**
C81224		Yes	Derivation Type	DTYPE	Derivation Type: Analysis value derivation method.	CDISC ADaM Derivation Type Terminology
C81209	C81224		Derivation Type	AVERAGE	Average: A data derivation technique which calculates a subject's average value over a defined set of records.	Average of Value Derivation Technique
C92225	C81224		Derivation Type	BC	Best Case: A data imputation technique which populates missing values with the best possible outcome.	Best Case Imputation Technique
C81201	C81224		Derivation Type	BLOCF	Baseline Observation Carried Forward: A data imputation technique which populates missing values with the subject's nonmissing baseline observation.	Baseline Observation Carried Forward Imputation Technique
C92226	C81224		Derivation Type	BOCF	Best Observation Carried Forward: A data imputation technique which populates missing values with the subject's best-case nonmissing value.	Best Observation Carried Forward Imputation Technique
C82866	C81224		Derivation Type	ENDPOINT	Endpoint: A data derivation technique which calculates a subject's analysis end point value.	Endpoint Value Derivation Technique
C81208	C81224		Derivation Type	INTERP	Interpolation: A method of imputation involving a missing value that is between known values and is estimated by a function of those known values.	Interpolation Imputation Technique
C81198	C81224		Derivation Type	LOCF	Last Observation Carried Forward: A data imputation technique which populates missing values with the subject's previous nonmissing value.	Last Observation Carried Forward Imputation Technique

C82868	C81224		Derivation Type	MAXIMUM	Maximum: A data derivation technique which calculates a subject's maximum value over a defined set of records.	Maximum Value Derivation Technique
C82867	C81224		Derivation Type	MINIMUM	Minimum: A data derivation technique which calculates a subject's minimum value over a defined set of records.	Minimum Value Derivation Technique
C53331	C81224		Derivation Type	ML	Maximum Likelihood: A data imputation technique which populates missing values with estimates that maximize the probability of observing what has in fact been observed.	Maximum Likelihood Estimation
C81204	C81224		Derivation Type	MOTH	Mean of Other Group: A data imputation technique which populates missing values with the mean value from a comparator or reference group.	Mean of Other Group Imputation Technique
C81207	C81224		Derivation Type	MOV	Mean Observed Value in a Group: A data imputation technique which populates missing values with the mean value observed in a group of subjects.	Mean Observed Value in a Group Imputation Technique
C81205	C81224		Derivation Type	POCF	Penultimate Observation Carried Forward: A data imputation technique which populates missing values with the subject's next-to-last nonmissing value.	Penultimate Observation Carried Forward Imputation Technique
C81200	C81224		Derivation Type	SOCF	Screening Observation Carried Forward: A data imputation technique which populates missing values with the subject's nonmissing screening observation.	Screening Observation Carried Forward Imputation Technique
C81203	C81224		Derivation Type	WC	Worst Case: A data imputation technique which populates missing values with the worst possible outcome.	Worst Case Imputation Technique
C81199	C81224		Derivation Type	WOCF	Worst Observation Carried Forward: A data imputation technique which populates missing values with the subject's worst-case nonmissing value.	Worst Observation Carried Forward Imputation Technique
C81206	C81224		Derivation Type	WOV	Worst Observed Value in a Group: A data imputation technique which populates missing values with the worst value observed in a group of subjects.	Worst Observed Value in a Group Imputation Technique