PharmaSUG 2022 – Paper QT-08

Programmed Value-Level Metadata for Define XML 2.0

Abhinav Srivastva, Exelixis Inc.

ABSTRACT

Value-level metadata in Define-XML 2.0 has been greatly enhanced to list a unique set of values (typically for a result variable) in combination with one or more categorical parameters of interest using a WHERE subset clause. A given slice of data using a WHERE subset clause can also be based on multiple datasets to create complex subsets as per the intent of the sponsor. The paper introduces a SAS® macro which can help create a Value-level metadata in XML format that can be used as part of the larger Define XML file along with preparing other components like Control Terminology (CT), Comments, etc that go with the Define document.

INTRODUCTION

Value-level metadata describes the data attributes like type, length, format, controlled terminology, origin, derivation method or comments associated with a subsetted value created with a WHERE clause. Display 1 exhibits these attributes associated with a sample case of EGORRES variable with the subset of *where* EGTESTCD = INTP.

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
EGORRES	EGTESTCD EQ INTP (Interpretation)	text	8	["ABNORMAL", "NORMAL"] <Interpretation: Original Results>	CRF Page 12	

Display 1: Value-level metadata example for Variable EGORRES

Underlying XML code to create the above display is as below. It's divided into 3 parts- *ValueListDef*, *WhereClauseDef* and *ItemDef*.

```
<!-- ***********************-->
<!-- Value-List Definition -->
<!-- ********************** -->

<def:ValueListDef OID="VL.EG.EGORRES">
   <ItemRef ItemOID="IT.EG.EGORRES.INTP" OrderNumber="1" Mandatory="No">
   <def:WhereClauseRef WhereClauseOID="WC.EG.EGTESTCD.INTP"/>
   </ItemRef>
</def:ValueListDef>
```

```
<!-- ***********************-->
<!-- WhereClause Definition -->
<!-- ********************** -->

<def:WhereClauseDef OID="WC.EG.EGTESTCD.INTP">
   <RangeCheck SoftHard="Soft" def:ItemOID="IT.EG.EGTESTCD" Comparator="EQ">
   <CheckValue>INTP</CheckValue>
   </RangeCheck>
</def:WhereClauseDef>
```

```
<!-- ******************-->
<!-- ItemDef Definition -->
<!-- ******************-->

<ItemDef OID="IT.EG.EGORRES.INTP" Name="EG.INTP.ORRES" DataType="text"
         Length="8" SASFieldName="INTPOR">
   <Description>
     <TranslatedText xml:lang="en">Interpretation: Original
         Results</TranslatedText>
   </Description>
   <CodeListRef CodeListOID="CL.NABCLIN"/>
   <def:Origin Type="CRF">
      <def:DocumentRef leafID="LF.blankcrf">
        <def:PDFPageRef PageRefs="12" Type="PhysicalRef"/>
      </def:DocumentRef>
   </def:Origin>
</ItemDef>
```

Display 2 and Display 3 are other examples of complex WHERE clauses which can be created for Value-level metadata.

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
LBORRES	LBTESTCD EQ BILI (Bilirubin) AND LBCAT EQ CHEMISTRY AND LBSPEC EQ BLOOD	float	3		eDT	

Display 2: Example of a subset based on multiple variables in the same dataset (LB)

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
VSORRESU	VSTESTCD EQ HEIGHT (Height) AND COUNTRY IN ("CAN" , "MEX")	text	5	["cm" = "Centimeter"] <Unit (UH_MC)>	CRF Page 11	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. The data submitted only includes subjects in the USA since other sites did not enroll any subjects.

Display 3: Example of a subset based on variables from two separate datasets (VS and DM)

VALUE-LEVEL LIST MACRO

Please see ***%vlm*** macro definition below. SAS® code is provided in Appendix section.

```
%macro vlm (inlib = , inds = /* Input Library and dataset name (SDTM.LB) */
           ,vvar     = /* Name of the Variable to create value-list (eg.
                          EGORRES) */
           ,wherevar = /* Name of the Variable for WHERE-clause (eg.
                          EGTESTCD) */
           ,xmlloc   = /* Location of the outputted XML file */
           ,wantxls  = %str(N) /* Do you need an excel output (Y/N) */
           );
```

The macro generates XML code relevant to the value-level metadata sections of the Define document. Getting '*Data Type*', '*Length*' and '*Significant Digits*' are some of the key aspects which needs to be evaluated at each slice of the WHERE clause. In addition to the XML code, the macro also creates an excel file per Pinnacle 21 template with necessary attributes for value-level metadata which can be used as part of Define XML creation for sponsors which uses excel as an input to create Define document. There are 2 tabs created in the excel output as below for a sample case of EGORRES.

Order	Dataset	Variable	Where Clause	Data Type	Length	Significant Digits	Format
1	EG	EGORRES	EG.EGORRES.INTP	text	8		

Mandatory	Codelist	Origin	Pages	Method	Predecessor	Value Level Comment	Join Comment

Display 4: ValueLevel tab in excel output

ID	Dataset	Variable	Comparator	Value
EG.EGTESTCD.INTP	EG	EGTESTCD	EQ	INTP

Display 5: WhereClause tab in excel output

Assumptions of the Macro

1. The macro subsets data based on *only* "EQ" (=) operator (eg. EGTESTCD 'EQ' INTP) among other options available like GT (>), LT (<) etc which can be customized outside the macro as per the sponsor specification.
2. The macro does not handle cases as explained in Display 2 and Display 3 which are very specific to the intent of the sponsor.
3. Derivation Method, Comment, Origin, Codelist as applicable at each value of the Value-level metadata is very unique to itself which can only be handled outside the macro through manual review.

CONCLUSION

The paper presented a programmatic approach to create value-level metadata XML code per Define 2.0 requirement. For users with basic XML knowledge, some additional coding can be done in the resultant XML file from the macro to upgrade to sponsor-specific requirements as mentioned in Assumptions of the Macro section. The customization can also be made in the excel output from the macro when doing excel-based Define generation.

REFERENCES

CDSIC Define-XML v2.0 Standards

(https://www.cdisc.org/standards/data-exchange/define-xml)

CONTACT INFORMATION

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

Name : Abhinav Srivastva
Enterprise : Exelixis Inc
E-mail : asrivastva@exelixis.com

APPENDIX

```
%macro vlm(inlib   = , inds =  /* Library & Name of the input dataset */
          ,vvar    = /* Name of the Variable for value-list (eg. EGORRES) */
          ,wherevar= /* Name of the Variable for WHERE clause(eg EGTESTCD)*/
          ,xmlloc  = /* Location of the outputted XML file */
          ,wantxls = %str(N) /* Do you need an excel output (Y/N) */
          );

/* Determine Data Type of the Value-List Variable (eg. EGORRES, AVAL) */
%let dsid=%sysfunc(open(&inlib..&inds.));
  %let vpos=%sysfunc(varnum(&dsid,&vvar.));
  %let vtype=%sysfunc(vartype(&dsid,&vpos.));
%let rc=%sysfunc(close(&dsid));

/* Assign attributes (TYPE, LENGTH, SIGNIFICANT DIGITS) of the Input Var. */
/* For Numeric results, set LENGTH=8, TYPE is either INTEGER or FLOAT     */
/* For Character results, LENGTH= max(character value), TYPE=TEXT         */

%if &vtype=N %then %do;
 data ___srcds (keep= &wherevar. dtype dlen sigdec);
  set &inlib..&inds. (where=(&wherevar. ne ' '));
   dlen=8;
   if not missng(&vvar.) then do;
    if index(strip(put(&vvar.,best32. -l)),'.')>0 then /* float */
     do;
       dtype=2; sigdec=lengthn(scan(strip(put(&vvar.,best32. -l)),2,'.'));
     end;
       else do; dtype=1; sigdec=.; end; * integer;
   end;
%end;

%else %if &vtype=C %then %do;
 data ___srcds (keep= &wherevar. dtype dlen sigdec);
  set &inlib..&inds. (where=(&wherevar. ne ' '));
   if not missing(&vvar.) then do;
     if anydigit(&vvar.,1)=0 then do; *text;
      dtype=3; dlen=length(strip(&vvar.));
     end;
      else do;
       dlen=8;
       if index(strip(put(&vvar.,best32. -l)),'.')>0 then do; *float;
          dtype=2; sigdec=lengthn(scan(strip(put(&vvar.,best32. -l)),2,'.'));
       end;
        else do; dtype=1; sigdec=.; end; * integer;
      end;
    end;
 run;

%end;

proc sort data= ___srcds;
  by &wherevar. descending dtype descending dlen descending sigdec;
run;
```

```
data ___attr (where=(dtype ne .));
  set ___srcds;
  length dfmt $5;
   by &wherevar. descending dtype descending dlen descending sigdec;
   if dtype=2 and n(dlen,sigdec)=2 then
      dfmt=catx('.',put(dlen, best32. -l),put(sigdec, best32. -l));
   if first.&wherevar. then output;
run;

/* Sometimes Value-List can have many values than actually needed. For example,
Lab dataset can have 100+ test codes (LBTESTCD) but only a few are meaningful
for analysis. In that case, please write some custom code to exclude extra Test
Codes */

<<<... Some SAS Code to exclude certain value-list if Needed ...>>

/* If Excel report is Needed, set up all variables here */
data ___report (drop= _dtype);
  set ___attr (rename=(dtype=_dtype));
  length dtype $10 whrcls VarID $50;
  array othvars{*} $ mandatory codelist origin pages method predecessor com
        joincom;
   order+1;
   dataset =%sysfunc(upcase("&inds."));
   variable=%sysfunc(upcase("&vvar."));
   whrcls=catx('.',dataset,%sysfunc(upcase("&wherevar.")),&wherevar.);
   VarID=catx('.',dataset,variable,&wherevar.);
          if _dtype=1 then dtype='text';
     else if _dtype=2 then dtype='float';
     else if _dtype=3 then dtype='integer';
  comparator='EQ';
run;

<<<... Insert Some SAS Code to output to EXCEL, such as ODS EXCEL ...>>

/* XML code for ValueListDef below, try for WhereClauseDef & ItemDef! */
data _null_;
 set ___report end=eof;
 by &wherevar.;
 file "./%upcase(&vvar)-vlist.xml" ls=1000;
 retain sp -1;
 if _n_=1 then
  put @3 '<def:ValueListDef OID="VL.' "%upcase(&inds.).%upcase(&vvar.)" '">';
 if first.&wherevar. then do;
  put @5 '<ItemRef ItemOID="IT.' VarID +sp '" OrderNumber="' order +sp '"
Mandatory="No">';
  put @5 '<def:WhereClauseRef WhereClauseOID="WC.' whrcls +sp '"/>';
 end;
 if last.&wherevar. then do;
  put @5 '</ItemRef>';
 end;
 if eof then put @3 '</def:ValueListDef>';
run;
%mend;

/* Sample Call on VS.VSSTRESC */
%vlm (inlib= work, inds=vs, vvar=vsstresc, wherevar=vstestcd);
```

XML Output for VSSTRESC when VSTESTCD = DIABP, HEIGHT and PULSE:

==========

```
<def:ValueListDef OID="VL.VS.VSSTRESC">

  <ItemRef ItemOID="IT.VS.VSTESTCD.DIABP.VSSTRESC" Mandatory="No">
      <def:WhereClauseRef WhereClauseOID="WC.VS.VSTESTCD.DIABP.VSSTRESC"/>
  </ItemRef>
  <ItemRef ItemOID="IT.VS.VSTESTCD.HEIGHT.VSSTRESC" Mandatory="No">
      <def:WhereClauseRef WhereClauseOID="WC.VS.VSTESTCD.HEIGHT.VSSTRESC"/>
  </ItemRef>
  <ItemRef ItemOID="IT.VS.VSTESTCD.PULSE.VSSTRESC" Mandatory="No">
      <def:WhereClauseRef WhereClauseOID="WC.VS.VSTESTCD.PULSE.VSSTRESC"/>
  </ItemRef>

</def:ValueListDef>
```

```
<ItemDef OID="IT.VS.VSTESTCD.DIABP.VSSTRESC" Name="DIABP" DataType="integer"
         Length="8" SASFieldName="DIABP">
 <Description><TranslatedText xml:lang="en">Diastolic Blood
  Pressure</TranslatedText></Description>
      <def:Origin Type="Assigned"/>
</ItemDef>

<ItemDef OID="IT.VS.VSTESTCD.HEIGHT.VSSTRESC" Name="HEIGHT" DataType="float"
         Length="8" SASFieldName="HEIGHT" SignificantDigits="1">
 <Description><TranslatedText
  xml:lang="en">Height</TranslatedText></Description>
      <def:Origin Type="Assigned"/>
</ItemDef>

<ItemDef OID="IT.VS.VSTESTCD.PULSE.VSSTRESC" Name="PULSE" DataType="integer"
         Length="8" SASFieldName="PULSE">
 <Description><TranslatedText xml:lang="en">Pulse
   Rate</TranslatedText></Description>
      <def:Origin Type="Assigned"/>
</ItemDef>
```

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

<def:WhereClauseDef OID="WC.VS.VSTESTCD.HEIGHT.VSSTRESC">
  <RangeCheck SoftHard="Soft" def:ItemOID="IT.VS.VSTESTCD" Comparator="EQ">
      <CheckValue>HEIGHT</CheckValue>
  </RangeCheck>
</def:WhereClauseDef>

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