Paper #27-2018

Generating Variable Attributes for Define 2.0

Abhinav Srivastva, Gilead Sciences Inc.

ABSTRACT

CDISC Define-XML 2.0.0 was officially released in March 2013 by the CDISC-XML Technologies team to describe CDISC SDTM, SEND and ADaM datasets for the purpose of submissions to the FDA. This version presents major changes from the previous version 1.0.0 released in February 2005. The new version runs on the latest ODM v1.3.2. (Operational Data Model) which is a vendor neutral, platform independent format for the interchange and archival of clinical study data. Some of the key changes in this version include extended Controlled Terminology (CT), detailed Value Level Metadata (VLM) description allowing subsets of data with a where-clause, explicit linking to external documents and allowing more clarity to the specifications via Comments.

The paper focuses on elaborating the different variable data types that were introduced in v2.0.0 and utilize the guidelines to build an approach when creating Define-XML, especially those that are excel-driven in the back end. The attributes – data type, length, format, and significant digits of a variable are the key pieces that will be discussed in following sections. The discussion narrows down to a SAS® macro that is able to generate them as per the new guidelines.

INTRODUCTION

The data types in v2.0.0 is greatly enhanced to a richer set based on the CDISC ODM. Table 1 (below) represents the data types that are now available in this version.

Define-XML Data Type	Submission Data Type	Length	Considerations
text	Char	Maximum allowable length.	SAS Version 5 transport files restrict variable lengths to 200 characters.
integer	Num	The largest allowable integer width.	Use for numeric or equivalent variables that have discrete whole values (non-fractional). Can be positive, negative, or zero. ADaM date variables, are provided as integers.
float	Num	The largest allowable whole number width plus the maximum number of decimal digits.	Use for numeric variables that may contain a fractional component. It represents the set of all the decimal numbers with arbitrary lengths.
datetime	Char	N/A	Use if values for SDTM or Send variable represent Date Times (YYYY-MM-DDTHH:MM:SS.SS).
date	Char	N/A	Use if values for SDTM or SEND variable represent complete (YYYY-MM-DD) dates.
time	Char	N/A	Use if values for SDTM or SEND variable represent complete (HH:MM:SS.SS) times in ISO-8601 format.
partialDate	Char	N/A	Use when date values may be right truncated. That is, if the day or the day and month may be missing.
partialTime	Char	N/A	Use when time values may be right truncated. That is, if the seconds or seconds and minutes may be truncated.

partialDatetime	Char	N/A	Use when date or time values may be right truncated (see partialDate and partialTime). For example, use to represent a full date and time in HH:MM.
incompleteDatetime	Char	N/A	Use when date values may be missing a component but is not a partialdatetime.
durationDatetime	Char	N/A	Use to indicate values use the ISO8601 “P” notation to indicate a duration value.

Table 1: Data Types Define XML-2.0

From the above table, it can be seen that all the Timing variables’ data types are greatly expanded, i.e. data types for variables ending with --DTC, --ELTM, --DUR, --INT in SDTM. ADaM dates (ASTDT, ADT, etc) on the other hand tend to be stored as SAS internal numbers so they are reported as ‘integer’.

A broad classification of the Timing variables’ data type in SDTM is represented below (Table 2)

Root Variables	Label	Data Type	[Dataset].Variables
-- DTC	Date/time	date or time or datetime (including “partial” component)	ex: AE.AESTDTC, CM.CMSTDTC
-- ELTM	Planned Elapsed Time from Time Point Ref	durationDatetime	ex: EG.EGELTM, LB.LBELTM
-- DUR	Planned Duration	durationDatetime	ex: EX.EXDUR, PR.PRDUR
-- INT	Planned Interval	durationDatetime	ex: QS.QSEVLINT, PP.PPSTINT
TDSTOFF, TDTGTPAI, TDMINPAI, TDMAXPAI	SDTM Trial Disease (TD) Timing variables	durationDatetime	TD (Trial Disease Assessments)

Table 2: SDTM Timing variables data type

VARIABLE DATA TYPES IN DEFINE-XML

Based on the considerations in Table 1: Data Types Define XML-2.0, a tabular summary is presented below (Table 3) which gives some examples of data and the attributes resulting from it.

Examples	Data Format	Data Type	Length	Decimals
AAA BBB CCC		Text	11	
2006-02-12	YYYY-MM-DD	Date		
2006-02	YYYY-MM	partialDate		
12:10:10.10	HH:MM:SS.SS	Time		
12:10	HH:MM	partialTime		
2006-02-12T12:10:10	YYYY-MM-DDTHH:MM:SS	Datetime		
2006-02-12T12:12	YYYY-MM-DDTHH:MM	partialDatetime		
2006-02-12T12:-:10	YYYY-MM-DDTHH:-:SS	incompleteDatetime		
P2Y	PnYnMnDTnHnMnS / PnW	durationDatetime		
100		integer	3	
100.5		Float	5	1

Table 3: Variable attributes

These attributes in-turn fit into the CDISC ODM as explained by a sample XML code for an SDTM variable, say, VSSTRESU:

```
<!-- Item Definition for VSSTRESU (1)                (3)          (4)
<ItemDef OID="IT.VS.VSSTRESU" Name="VSSTRESU" DataType="text" Length="9" SASFieldName="VSSTRESU"
  def:CommentOID="COM.VSSTRESU">(7)
  <Description>
    <TranslatedText xml:lang="en">Standard Units</TranslatedText> (2)
  </Description>
  <CodeListRef CodeListOID="CL.VSRESU"/> (5)
  <def:Origin Type="Assigned"/> (6)
</ItemDef>
```

(1) Variable	(2) Label	Key	(3) Type	(4) Length	(5) Controlled Terms or Format	(6) Origin	(7) Derivation/Comment
VSSTRESU	Standard Units		text	9	["BEATS/MIN" = "Beats per Minute", "cm" = "Centimeter", "kg" = "Kilogram", "mmHg" = "Millimeter of Mercury"] <Units for Vital Signs Results>	Assigned	Standard units consistent with CDISC controlled terminology

Display 1: Sample ItemDef for SDTM Vital Signs variable – VSSTRESU

Please see corresponding markers to relate the attribute with its respective place in the XML code.

The ItemDef element utilizes the following business rules for data type, length and significant digits as:

Attribute	Usage	Allowable Values	Description
DataType	Required	See Table 1: Data Types Define XML-2.0	The data type of the variable or value.
Length	Conditional Required if DataType is “text”, “integer”, or “float”.	Integer	The variable length. Business Rule: Length should be defined as the maximum expected variable length. Should only be present for DataType equal to “text”, “integer”, or “float”.
SignificantDigits	Conditional Required if DataType is “float”.	Integer	The number of digits following the decimal point in a floating point number. Business Rule: When DataType is float both Length and SignificantDigits must be provided.

Table 4: ItemDef Element rules for Data Type, Length and Significant Digits

ATTRIBUTE MACRO

The paper introduces a SAS macro ***%gen_var_att*** that can extract the attributes of a variable from the data and report in the form as demonstrated in the below example (Table 5).

Example: Consider a sample lab dataset as:

USUBJID	LBTESTCD	LBTEST	LBSTRESN	LBDTC	LBELTM
ABC-123	ALB	Albumin	5.1	2017-01-01	P3D
ABC-123	ALP	Alkaline Phosphatase	50	2017-01-01	P3D
ABC-123	ALT	Alanine Aminotransferase	17	2017-01-01	P3D
ABC-123	AST	Aspartate Aminotransferase	23	2017-01-01	P3D

Table 5: Sample Lab dataset for *%gen_var_att* demonstration

A sample macro call generates the following summary table and XML code as:

```
%gen_var_att (indsn = , /* Input dataset */
              outdsn=   /* Output dataset */
             )
```

The attributes from the above snippet dataset (Table 5) will be read and generated by the ***%gen_var_att*** macro in a tabular format as:

Variable	Label	Data Type	Length	Decimals
USUBJID	Unique Subject Identifier	Text	7	
LBTESTCD	Lab Test or Examination Short Name	Text	3	
LBTEST	Lab Test or Examination Name	Text	26	
LBSTRESN	Numeric Result/Finding in Standard Units	float	3	1
LBDTC	Date/Time of Specimen Collection	date		
LBELTM	Planned Elapsed Time from Time Point Ref	durationDatetime		

Table 6: Output from *%gen_var_att* macro

XML code resulting from ***%gen_var_att*** for one of the variables-USUBJID is shown in below display:

```
<ItemDef OID="IT.LB.USUBJID" Name="USUBJID" DataType="text" Length="7" SASFieldName="USUBJID">
    <Description>
       <TranslatedText xml:lang="en">Unique Subject Identifier</TranslatedText>
    </Description>
</ItemDef>

....
....
< more lines>
```

Display 2: Partial XML code from *%gen_var_att* macro

Please see Appendix for complete SAS code.

APPLICATION OF THE MACRO

While there are a number of ways that Define-XML is being created by organizations, Excel-based still remains one the popular choices as it is user-friendly and allows more flexibility to the end user. Schematically the process for creating excel-based Define-XML can be represented as Figure 1 below:

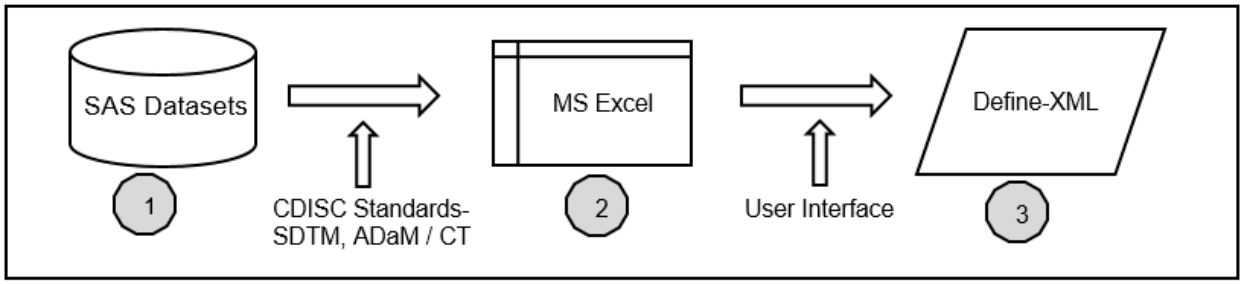

Figure 1: Excel-based Define Process flow

1. Metadata from SAS datasets are read and converted using vendor software application/ tool. Generally, the input needed will be the CDISC Standard name/version that the datasets are being based on.
2. The excel file generated from the software application/tool can need minor to major updates by the user depending on the functionality of the tool.
3. The updated excel file is in-turn imported back to the tool to produce the XML file which uses stylesheets and schema files to render for web browsing.

The macro as discussed can be used to generate the attributes of the variables across all datasets and output in an excel format with a layout similar to the excel file generated by the tool, thereby saving the time and effort to do it manually if the tool is lacking this capability. It can also be used to validate the attributes if one is already being generated by the tool. The macro can be further expanded to generate the attributes of the resultant data arising from slicing/sub-setting the variables for the Value-Level Metadata (VLM) section of the Define.

CONCLUSION

The paper presented an approach to extract Variable attributes in up-versioned Define 2.0 using Base SAS codes. It can be expanded to assemble other pieces like Codelists, Value-level metadata, etc to put together a submission-ready Define XML package.

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 : Gilead Sciences
E-mail : srivastvaabhinav@gmail.com

APPENDIX

```
/*=========================================*/
/* Get Variable attributes from SDTM datasets */
/*=========================================*/

options minoperator missing=' ';
libname src "&source”; /* Library where SDTM datasets are located */
%let path = &path.;    /* Path where outputs (excel,XML) from the macro will be stored */

%macro gen_var_att (indsn   = , /* Input dataset */
                    outdsn  = , /* Output dataset */
                    wantxlsx= Y,/* Excel output (Y/N), default=Y */
                    wantxml = N /* XML output (Y/N), default=N */
                    );

/* Check Input parameters */
%if %length(&indsn)=0 %then %do;
    %put !!!! STOP - No Input provided !!!!;
    %goto exit;
%end;

%if %sysfunc(exist(src.&indsn.)) NE 1 %then %do;
  %put !!!! STOP - Dataset src.&indsn. does not exist !!!!;
  %goto exit;
%end;

%if %length(&outdsn)=0 %then %do;
    %put Setting Ouput dataset name same as Input with suffix: xx_metadata;
    %let outdsn=&indsn._metadata;
%end;

/* Get Dataset contents */
proc contents data=src.&indsn. out=__&indsn.1 (keep= varnum memname name label type format
                                                     length rename=(length=length2)) noprint;
run;

/* Convert all numeric fields to character */
data _null_;
       set __&indsn.1 end=last;
       if _n_=1 then call execute("proc sql; create table __&indsn.2 as select ");
       if type=2 then call execute(name);
         else call execute('put('||strip(name)||',best32. -l) as '|| strip(name));
       if not last then call execute(',');
         else call execute(" from src.&indsn; quit; ");
run;

%let num_list=;
proc sql;
       select strip(name) into:num_list separated by " "
        from __&indsn.1
        where type=1;
quit;

%if &num_list. ne %then %do;

data __&indsn.2;
       set __&indsn.2;
       array allcols{*} &num_list;
       do i=1 to dim(allcols);
              if strip(allcols{i})='.' then allcols{i}=' ';
       end;
run;

%end;

%macro attrs(var= , type= );

       /* Identify all date --DTC variables */
       %let root=;
       %if "%substr(&var.,%length(&var.)-2)"="DTC" %then %do;
```

```
               %let root=date/time;
      %end;

/* Identify all --DUR, --INT, --ELTM and other duration variables from TD Domain */
%else %if
           "%substr(&var.,%length(&var.)-2)"="DUR"  OR "%substr(&var.,%length(&var.)-2)"="INT"
           OR
           "%substr(&var.,%length(&var.)-3)"="ELTM" OR "&var." IN ("TDSTOFF" "TDTGTPAI"
           "TDMINPAI" "TDMAXPAI") %then %do;
           %let root=duration;
%end;

      data __tmp1;
             set __&indsn.2 (keep = &var.);
             length root $10;
             ind_len=lengthn(strip(&var.));
             dtyp= &type.;

             if dtyp=1 then do;
                    root='Numbers';
                    if index(&var.,'.')>0 then set_pref=2;
                       else set_pref=1;
             end;
             else do;
                    %if "&root"="date/time" %then %do;
                           root='date/time';
                           if index(&var.,'T')>0 then set_pref=2;
                              else set_pref=1;
                    %end;
                    %else %if "&root"="duration" %then %do;
                           root='duration';
                           set_pref=1;
                    %end;
                    %else %do;
                      root='text';
                      set_pref=1;
                    %end;
             end;
      run;

      /* Calculate max length, max decimal precision */
      data __tmp2 (keep= dtyp max_len max_dec max_pref);
             set __tmp1 end=last;
             retain max_len 0;
             retain max_dec 0;
             retain max_pref 1;
             if ind_len > max_len then max_len=ind_len;
             if dtyp=1 and set_pref=2 then do;
                    if lengthn(strip(scan(&var.,2,'.'))) > max_dec then
                           max_dec=lengthn(strip(scan(&var.,2,'.')));
             end;
             if set_pref > max_pref then max_pref=set_pref;
             if last then output;
      run;

      data __tmp3 (drop= max_len max_dec);
             merge __tmp1 __tmp2;
              by dtyp;
              length Memname Name $32;
              Memname = %sysfunc(upcase("&indsn."));
              Name    = "&var.";
              Length  = max_len;
              Decimals= max_dec;
      run;

      /* Determine DataType for Character and Numeric fields */
      data __tmp4;
             set __tmp3(where= (set_pref=max_pref));
             if dtyp=2 and not missing(&var.) then do; *- datatype for character fields -*;
                    if root='date/time' and set_pref=2 then do; *- datetime patterns -*;
```

```
                              if (prxmatch('/\d{4}-\d\d-\d\dT\d\d:\d\d:\d\d/',&var.)>0 and
lengthn(strip(&var.))=19) or
                                 (prxmatch('/\d{4}-\d\d-\d\dT\d\d:\d\d:\d\d.\d\d/',&var.)>0 and
lengthn(strip(&var.))=22) then set_sub_pref=3;
                                else if (prxmatch('/\d{4}-\d\d-\d\dT\d\d/',&var.)>0 and
lengthn(strip(&var.))=13) or
                                        (prxmatch('/\d{4}-\d\d-\d\dT\d\d:\d\d/',&var.)>0 and
lengthn(strip(&var.))= 16) then set_sub_pref=2;
                                else set_sub_pref=1;
                          end;
                             else if root='date/time' and set_pref=1 then do;
                                if index(&var.,':')=0 and lengthn(strip(&var.))>=4 then do; *-
date patterns -*;
                                        date_time_pref=2;
                                  if prxmatch('/\d{4}-\d\d-\d\d/',&var.)>0 and
lengthn(strip(&var.))= 10 then set_sub_pref=3;
                                          else if (prxmatch('/\d{4}-\d\d/',&var.)>0 and
lengthn(strip(&var.))= 7) or
                                                  (prxmatch('/\d{4}/',&var.)>0 and
lengthn(strip(&var.))= 4) then set_sub_pref=2;
                                          else set_sub_pref=1;

                               end;
                                else do; *- time patterns -*;
                                        date_time_pref=1;
                                  if (prxmatch('/\d\d:\d\d:\d\d.\d\d/',&var.)>0 and
lengthn(strip(&var.))= 11) or
                                     (prxmatch('/\d\d:\d\d:\d\d/',&var.)>0 and
lengthn(strip(&var.))= 8) then set_sub_pref=3;
                                          else if (prxmatch('/\d\d:\d\d/',&var.)>0 and
lengthn(strip(&var.))= 5) or
                                (prxmatch('/\d\d',&var.)>0 and lengthn(strip(&var.))= 2)    then
set_sub_pref=2;
                                          else set_sub_pref=1;

                                end;
                             end;
                             else if root='duration' then do; *- duration with P notation -*;
                                if index(&var.,'P')=0 then set_sub_pref=2;
                                   else set_sub_pref=1;
                             end;
                             else if root='text' then set_sub_pref=1; *- all other text fields -*;
          end;
        run;

        proc sort data=__tmp4;
                by dtyp root set_pref date_time_pref set_sub_pref;
        run;

        data __tmp5;
                set __tmp4;
                 by dtyp root set_pref date_time_pref set_sub_pref;
                       if last.root then output;
        run;

        /* Assign DataTypes */
        data __tmp6 (keep= Memname Name dataType Length Decimals);
                set __tmp5;
                length dataType $20;
                if Length=0 then do;
                         Length=1;
                         if dtyp=1 then dataType='integer';
                            else dataType='text';
                end;
                    else do;
                      if dtyp=1 then do;
                          if set_pref=2 then dataType='float';
                             else dataType='integer';
                      end;
                         else do;
                              if root='date/time'     and set_pref=2 then do;
```

```
                                    if set_sub_pref=3 then dataType='datetime';
                                 else if set_sub_pref=2 then dataType='partialDatetime';
                                 else if set_sub_pref=1 then dataType='incompleteDatetime';
                                end;
                                 else if root='date/time'      and set_pref=1 then do;
                                       if date_time_pref=2 then do;
                                              if set_sub_pref=3 then dataType='date';
                                         else if set_sub_pref=2 then dataType='partialDate';
                                         else if set_sub_pref=1 then dataType='incompleteDatetime';

                                             end;
                                        else if date_time_pref=1 then do;
                                              if set_sub_pref=3 then dataType='time';
                                         else if set_sub_pref=2 then dataType='partialTime';
                                         else if set_sub_pref=1 then dataType='incompleteDatetime';

                                             end;
                                 end;
                                else if root='duration' then do;
                                       if set_sub_pref=2 then dataType='text';
                                  else if set_sub_pref=1 then dataType='durationDatetime';
                                end;
                                else if root='text' then dataType='text';
                           end;
                  end;
        run;

        proc append base=metadata data=__tmp6;
        run;

%mend attrs;

   data _null_;
          set __&indsn.1;
          call execute('%attrs(var='||name||' , type='||put(type,best.)||')');
   run;

        data __tmp7;
               merge __&indsn.1 (in=a) metadata;
                by memname name;
                if a;
                *- Additional Rules from the Define 2.0 spec -*;
                if dataType ^='float' then decimals=.;                        *1) Decimals needed
                                                                                  only for float;
                if dataType not in ('integer','float','text') then length=.; *2) Lengths needed
                                                                        only for selected dataTypes;
                if dataType ='text' then length=length2;                      *3) Text length is
                                                                    set to column width in the dataset;
                if dataType ='float' then format=catx('.',length,decimals);  *4) Format float;
        run;

        /* Final Dataset */
        proc sort data=__tmp7 out= &outdsn.(drop= type length2 rename=(varnum=Order));
               by memname varnum;
        run;

        /* Need Excel Report (Y/N) */
        %if "%upcase(&wantxlsx)"="Y" %then %do;

       ods listing close;
       ods excel file="&path/&outdsn..xlsx"; *- specify a path here to send the report ;
       ods excel options (AUTOFILTER = 'ALL'
                          ABSOLUTE_COLUMN_WIDTH ='7,8,10,30,15,10,10,10'
                          FROZEN_HEADERS = 'ON'
                          SHEET_INTERVAL = 'NONE'
                          EMBEDDED_TITLES = 'ON'
                          ROW_REPEAT = '1-3'
                          ORIENTATION = 'LANDSCAPE'
                          TAB_COLOR = 'red'
                          SHEET_NAME = 'Variable');
```

```
     title j=l "Run date: %sysfunc(date(),date9.) T%sysfunc(time(),tod8.)";
      proc report data = &outdsn. nowd split='*' style(header)=[just=l];
        column Order Memname Name Label dataType Length Decimals Format;
        define Order    / order   'Order';
        define Memname  / display 'Dataset' ;
        define Name     / display 'Variable';
        define Label    / display 'Label';
        define dataType / display 'Data Type';
        define Length   / display 'Length' style(column)=[just=l];
        define Decimals / display 'Significant*Digits' style(column)=[just=l];
        define Format   / display 'Format';
      run;

     ods excel close;

  %end;

  /* Need XML (Y/N) */
  %if "%upcase(&wantxml)"="Y" %then %do;

  data _null_;
     set &outdsn.;
          file "&path./&outdsn..xml";
             retain sp -1;
            length ll dd ff ldf $200;

               if Length   ne .   then ll= 'Length=' || '"' ||strip(put(length,best.))||'"';
               if Decimals ne .   then dd= 'SignificantDigits=' || '"'
||strip(put(decimals,best.))||'"';
               if Format   ne ' ' then ff= 'def:DisplayFormat=' || '"' ||strip(format)||'"';
               ldf=catx(' ',ll,dd,ff);

      put '<ItemDef OID="IT.' memname +sp '.' name +sp '" Name="' name +sp '" DataType="'
dataType +sp '" ' ldf +sp
          ' SASFieldName="' name +sp '">';
      put @3 '<Description>';
      put @5 '<TranslatedText xml:lang="en">' label +sp '</TranslatedText>';
      put @3 '</Description>';
      put '</ItemDef>';
   run;

  %end;

  %exit:

  %put %str(NOTE: Exiting &sysmacroname. Macro);

%mend gen_var_att;

/* Sample Call */
%gen_var_att (indsn= lb, outdsn= lb_metadata, wantxlsx = Y, wantxml= Y );
```