

A SAS® Macro to Automate the Process of Define.xml

Wenyu Hu, Merck Research Labs, Merck & Co., Inc., Upper Gwynedd, PA
Liping Zhang, Merck Research Labs, Merck & Co., Inc., Upper Gwynedd, PA

ABSTRACT

When submitting CDISC SDTM formatted data to the FDA, the data definition file define.xml, which describes the structure and contents of the data, is a mandatory deliverable. This file is also called the electronic case report tabulation data definitions (eCRT DD) and consists of the following sections: Data Metadata, Variable Metadata, Variable Value Level Metadata and Controlled Terminology/Code Lists. Since much of the metadata required for define.xml are already inherent in the datasets themselves, it is desirable to automate the process of creating define.xml. This paper provides a SAS®-based method to create Variable Value Level Metadata and Controlled Terminology/Code Lists with the help of in-house macros and SASHELP data dictionary views.

INTRODUCTION

A data definition file, which is the metadata describing the format and content of the submitted datasets, is necessary to facilitate the review of the data submitted to the regulatory agency. It consists of 2 parts: the metadata definition of the datasets or Tables of Contents (TOC) and the metadata definition of the variables within the datasets or Data Definition Table (DDT). Well defined, standardized metadata can minimize the time needed to become familiar with the data, which in turn can speed up the overall review process.

In order to standardize the submission process, Merck has created a series of in-house SAS based applications to ensure that the SDTM datasets submitted to FDA are CDISC compliant and to automate the creation of the define.xml. This streamlined process has greatly increased the accuracy and efficiency of the Electronic Common Technical Document (eCTD) submission.

The input to the custom built applications are SAS transport files and define.xls which includes SDTM metadata for domain datasets, SDTM metadata for domain content (including value level metadata) and SDTM metadata for controlled terminology/code list. Most of the standard domains' value level metadata and controlled terminology (code list) are already present in define.xls, however the study specific ones are not. The list could also contain controlled terms or value level metadata not used in this study. It is very time consuming to manually add new terms and remove the ones not used. Since much of information required to create metadata are in the datasets, we decided to automate the process of creating value level metadata and controlled terminology (code list) and thus further optimize the eCRT creation.

This paper will first introduce the basics of value level metadata and controlled terminology, and then describe the implementation steps.

VALUE LEVEL METADATA (Value List)

Case Report Tabulation datasets with normalized structure (i.e., one record per patient by test code per visit) provide an efficient method for information interchange, because the data structure does not change when new information is added. However, this structure requires additional metadata support in order to be useful for review and analysis.

For example, the CDISC SDTM defined Vital Sign (VS) domain has one record per vital sign measurement per visit per subject, and may contain records related to subjects' diastolic blood pressure, weight, pulse and systolic blood pressure, etc. Per the CDISC SDTM, the measurement parameter name is stored in a variable name ending with TESTCD, and the parameter value is stored in a variable ending with ORRES, STRESN or STRESC. Results of different test code could contain different data types, definition or display formats. To clearly communicate the definitions and attributes of these variable values, value level metadata

should be provided for each unique value of the test code and should be described using the CDISC variable metadata fields. The following is an example of value level metadata within the define.xml.

Value Level Metadata (ValueList.VS.VSTESTCD)

Source Variable	Value	Description	Type	Controlled Terminology	Origin	Comment
VSTESTCD	DIABP	Diastolic Blood Pressure	integer		CRF Page 8	
VSTESTCD	HEIGHT	Height	float		CRF Page 8	
VSTESTCD	PULSE	Pulse Rate	integer		CRF Page 8	
VSTESTCD	SYSBP	Systolic Blood Pressure	integer		CRF Page 8	
VSTESTCD	WEIGHT	Weight	float		CRF Page 8	

CONTROLLED TERMINOLOGY (Code Lists)

Many variables in a submission have a list of valid discrete values or controlled terms associated with them. The code list element defines a discrete set of permitted values for an item. The definition can be an explicit list of values or a reference to an externally defined code list such as MedDRA. The following is an example of a code list within the define.xml.

Controlled Terminology (Code Lists) SEX, Reference Name (SEX)

Code value	Code Text
F	FEMALE
M	MALE
U	UNKNOWN

IMPLEMENTATION

STEP #1: Gather data sets information in the directory specified by user

SAS dictionary tables are metadata and are automatically available when a SAS session starts. SASHELP views are created from the dictionary table, so they replicate the information stored in the dictionary table. The SASHELP views can be used to easily identify the domains available in the session and observations for every variable available in the session. The following example shows how SASHELP is used to get dataset information:

```
%let inlib=%upcase(&inlib);

data tables(keep=memname);
    set sashelp.vtable(where=(libname="&inlib" and memtype="DATA"));
run;

proc sql noprint;
    select count(memname) into :tot
    from tables
    ;
```

```

quit;

%let tot=&tot;

proc sql noprint;
    select memname into :tabname1 - :tabname&tot
        from tables
        ;
quit;

data vcolumn(keep=memname name type length);
    set sashelp.vcolumn(where=(libname="&inlib" and memtype="DATA"));
run;

```

In the above step, SASHELP.VTABLE is used to gather the name of data sets in the directory defined by macro variable &inlib, and then the name is put into macro variables tabname1 to tabname&tot. Data set name, column name, length and width are retrieved by using SASHELP.VCOLUMN.

STEP #2: Read in define.xls to get variable level attributes, definitions, and usage information for each variable contained within each domain.

With the help of Excel LIBNAME statement, we can access multiple worksheets in an excel file in much the same way that we access SAS datasets in a SAS library. Each worksheet will appear as a dataset in the library and can be accessed through SAS procedures, using a two-level data set name.

```

libname inxls2 excel "&inxls";

%do _k=1 %to &tot;

    data xls(keep=domain name label list origin);
        set inxls2."&&tabname&k..$"n;

        .....

    run;

    proc contents data=&inlib."&&tabname&k out=sdtm(keep=name)
        noprint;
    run;

    .....

    data xls2;
        merge xls(in=a) sdtm(in=b);
        by name;
        if a and b;
    run;

    .....

%end;

```

Here the dataset names gathered above are used to access each individual work sheet, since the master define.xls may include other domains not used in this study. Note the worksheet name includes a \$ sign at the end, which means that in order to access the worksheets, the name literal must be used.

STEP #3: Get value level metadata from the data and define.xls. Since each possible TESTCD value is treated as a variable, attributes of each TESTCD need to be obtained.

```
%do _i=1 %to &valuetot;

    data &&tabname&_k;
        set &inlib..&&tabname&_k;
        length type $10;
        if &&tabname&_k..stresn ne . and
        index(&&tabname&_k..stresc, '.')>0 then do;
            type='float';
            id=2;
        end;

        .....

run;

proc sql;
    create table tmpvalue
        as select distinct &&value&_i as valname, "&&valuelis&_i"
        as code, &&tabname&_k..test as deflabel, "&&value&_i" as
        name, type, id
        from &&tabname&_k
        where &&value&_i ne ''
        order by valname, id
        ;
quit;

proc sort data=tmpvalue nodupkey;
    by valname;
    where id ne .;
run;

.....

%end;
```

STEP #4: Get code list from data and define.xls

```
%do _j=1 %to &codetot;
    proc sql;
        create table tmpcode
        as select distinct &&code&_j as code, "&&codelis&_j" as name
        from &inlib..&&tabname&_k
        where &&code&_j ne ''
        ;
    quit;

    .....

%end;
```

STEP #5: Output value list and code list to excel spreadsheets.

```
proc datasets library=outxls2;
    delete out_codeitem out_valueitem;
quit;

data outxls2.out_codeitem;
    set out_codeitem;
run;

data outxls2.out_valueitem;
    set out_valueitem;
run;

libname outxls2 clear;
```

Before outputting the worksheet, the existing one must be deleted since the EXCEL engine in the LIBNAME statement does not allow replacement of a worksheet. As a good practice, library reference should be cleared upon finishing writing to the spreadsheet.

SUMMARY

SASHELP views/dictionary tables and EXCEL LIBNAME engine can be used to collect most of metadata information directly from data. This saves much time from manual work, and also ensures the accuracy of the metadata updates.

REFERENCES

1. CDISC Metadata Submission Guidelines, Appendix to the Study Data Tabulation Model Implementation Guide 3.1.1, Draft version 0.9, July 25, 2007 (<http://www.cdisc.org/models/sdtm/v1.1/index.html>)
2. Case Report Tabulation Data Definition Specification (define.xml), Final Version 1.0, Feb 10, 2005 (<http://www.cdisc.org/models/def/v1.0/index.html>)

ACKNOWLEDGEMENTS

The authors would like to thank the Merck eSub committee team for its continued efforts to improve the submission process and the programming management team for their encouragement and review of the paper.

TRADEMARKS

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.

CONTACT INFORMATION

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

Wenyu Hu
UG 1D-88
Merck Research Labs
Merck Co., & Inc.
Upper Gwynedd, PA 19454
(267) 305-6847
wenyu_hu@merck.com

Liping Zhang
UG 1CD-44
Merck Research Labs
Merck Co., & Inc
Upper Gwynedd, PA 19454
(267) 305-7980
liping_zhang@merck.com