

An Alternative Way to Create Define.XML for ADaM with SAS Macro Automation

Yiwen Li, Gilead Sciences, Foster City, CA

ABSTRACT

Define.XML for ADaM is required for most FDA submissions, as it describes the structure and contents of the ADaM data. It includes five sections: Data Metadata, Variable Metadata, Value Level Metadata, Computational Algorithm, and Controlled Terminology. Previously programmers used to create it with more support from filling out many Excel sheets as input. This paper provides a Linux SAS based simple method to extract almost all needed information from ADaM data with coding-logic-info-capture to create the whole Define.XML output. The only other importing source besides SAS is one Excel tab which stores description for variables derived with more complicated logic. It has been successfully implemented to HIV Phase1 study and significantly increased the efficiency of creating Define.XML by at least 50%.

Highlighted Strengths:

1. Build most columns on Define.XML from SAS Libraries
2. Build Origin/Comment columns with coding logic information capture

BACKGROUND

The highlighted strength has specific solid background support:

1. SAS Libraries contain information about all data and variables, which enables macrolization, such as: SASHELP.VCOLUMN, TASKTOOL.FORMATS
2. Naming Conventions mentioned a lot in ADaM Implementation Guide, which enables coding logic capture method, such as:
 - One-letter prefixes.
For an example of the problem, if * is Q, then a date *DT would be QDT; however, a starting date *SDT would be QSDT, which would potentially be confusing if the user intended QSDT to be something other than the numeric date version of the SDTM variable QSDT.
 - Two-letter prefixes, except when intentionally chosen to refer explicitly to a specific SDTM domain and its --DTC, --STDTC, and/or --ENDTC variables.
For an example of an appropriate intentional use of a two letter-prefix, if * is LB, then *DT is LBDT, the numeric date version of SDTM LBDT.

INTRODUCTION & IMPLEMENTATION

Define.XML is a web page based data definition file describing the formats and contents of the submitted data. The method introduced here is newly developed at Gilead Sciences to standardize and automate the creation of Define.XML for ADaM. The SAS Macro program captures information necessary for final display on different sessions of Define.XML web page from various strategies. Below is the introduction for how we implemented our two strengths to build every session on Define.XML.

Session1: Data Metadata

This is the data summary part with further hyperlink to each Variable Metadata of Session 2 and to .XPT file for each ADaM.

Columns: Dataset, Description, Class, Structure, Purpose, Keys, Location.
It is coded with several "PROC FORMAT" to control the overall data structure.

```
data tocs;
  Dataset="ADSL"; output;
  .....
run;
```

Session2: Variable Metadata

This is the detailed body part for each ADaM, with further hyperlink to Value Level Metadata session, Computational Algorithms session and Controlled Terminology session.

Columns: Variable, Label, Type, Controlled Terminology, Origin, Role, Comment.

Besides variable attributes pulled from SASHELP.VCOLUMN, the rest information needed is captured or derived based on coding logic. It needs the programmer to be familiar with data structure, naming convention, and the variable deriving process in order to better capture the hidden relationship between variables across ADaMs, and to transform the understanding to coding logic.

ADaM variables structure summary:

1. Variables first created in ADSL and carried over to other ADaMs
STUDYID, USUBJID
2. Variables origin=DERIVED and used in ADSL only
ENRLDT, LVISDT, LLSTDT
3. Variables otherwise
 - 3.1 variables coming from specific domain and used in only one ADaM
BRTHDT, LLABDT, AEXXDT
 - 3.2 variables used in more than one ADaM
ADT, PARCAT1, XXSEQ, AVAL, PARAMCD
 - 3.3 all other variables with ORIGIN=Domain.NAME

```
data specs(keep=sheetname name label type length origin ct role core comment ...);
  set sashelp.vcolumn (where=(libname='ADAMDATA'));
  if substr(name,1,2) in ('TR','AP') and (substr(name,length(name)-1) in ('DT','TM'...)
  or substr(name,length(name)) in ('P','A'))
  or ..... then do;
    if sheetname ne 'ADSL' then origin=strip('ADSL.'||strip(name));
    else do;
      .....
    end;
  end;
run;
.....
if strip(origin)='' then do;
  if strip(sheetname) ne 'ADSL' then
    origin=compress(substr(sheetname,3)||'.'||strip(name));
  else
    origin=compress(strip('DM')||'.'||strip(name));
end;
```

Session3: Value Level Metadata

This is the storage for value level display of various test codes derived in different ADaMs.

Columns: Source Variable, Value, Label, Type, Controlled Terminology, Origin, Role, Comment.

Information is captured by coding logic from SAS Library. Extra test code record per CRF but not collected in data could be added with %addline

```
%macro paramcd(lib, data);
  data &data.paramcd;
    set &lib.&data (keep=paramcd param avalc);
    if index(avalc,'BQL') or index(avalc,'BLQ') then type='float';
    else do;
      if indexc(upcase(avalc),'+ABCDEFGHIJKLMNOPQRSTUVWXYZ():/') then type='text';
      else if .....
    end;
  run;
%mend paramcd;

*** add valuelist record for lab tests mentioned in Protocol but not seen in BDS ADaMs
***;
%macro addline(name=, key=, label=, type=, cat=%str(Value List));
```

```
name="&name.";
key="&key.";
label="&label.";
type="&type.";
origin="%substr(&name,3,2)"||'|.'||"%substr(&name,3,2)"||'|TESTCD';
avalc="";
comment="";
output;
%mend addline;

data valuelist;
  set valuelist end=eof;
  output;
  if eof then do;
    %addline(Name = %str(ADLB.PARAMCD),
             Key = %str(TSTCD),
             Label = %str(TSTNAM),
             Type = %str(testtype));
  end;
run;
```

Session4: Computational Algorithms

This is the storage for hard-to-derive variables in ADaMs.

Columns: Reference Name, Computation Method.
It is directly imported from one Excel tab.

Session5: Controlled Terminology

This is the storage for variable level display of those with valid discrete values.

Columns: Code Value, Code Text

Controlled Terminology is made up of 1. Code Lists; 2. External Dictionaries.
Information is captured either from TASKTOOL.FORMATS or coding logic.

---- Code Lists:

- RoleCodeList
(code in sas manually)
- PARAMCD and PARAM
(PROC SORT NODUPKEY, macrolized, %addline for adding extra record if need)
- Other variables from ADaM: ARMCD, LBMETHOD, ...
(PROC SORT NODUPKEY, analysis based study specific display, macrolized)
- Formats from TASKTOOL.FORMATS: DISCN, LBCATCD, NY, TRT,
(complete mapping display, if need)

---- External Dictionaries

```
data ct_codelist;
  set ct_rolecodelist ct_paramcd ct_param ct_othervar ct_fmtdata;
run;
data ct;
  set ct_codelist ct_exdict;
run;
```

NICE FEATURES

1. Intermediate Excel Output the Spec for review purpose

Sheetname	Name	Label	Type	Origin	CT	Role	Comment
ADAE	TRTP	Planned Treatment	text	DERIVED		TREATMENT	(TRTP)
ADAE	TRTPN	Planned Treatment (N)	integer	DERIVED		TREATMENT	(TRTPN)
ADAE	TRTA	Actual Treatment	text	DERIVED		TREATMENT	(TRTA)
ADAE	TRTAN	Actual Treatment (N)	integer	DERIVED		TREATMENT	(TRTAN)
ADAE	APERIOD	Period	text	DERIVED		TIMING	(APERIOD)
ADAE	APERIODC	Period (C)	text	DERIVED		TIMING	(APERIODC)
ADAE	TRTEMFL	Treatment-Emergent Flag	text	DERIVED	YNUL	TREATMENT	(ADAE.TRTEMFL)
ADAE	BASETYPE	Baseline Type	text	DERIVED		INDICATOR	(BASETYPE)
ADAE	SAFRFL	Safety Analysis Record-Level Flag	text	DERIVED	YNUL	INDICATOR	(SAFRFL)
ADAE	CRIT1FL	Criterion 1 Evaluation Result Flag	text	DERIVED	YNUL	INDICATOR	Equal to SAFRFL

2. Opencdisc checking automatically done when build Define.XML

```
%macro opencdisc();
  %if (%sysfunc(fileexist(&ocprg))=1) %then %do;
    x &ocprg &mydef >/dev/null;
  %end;
%mend;
```

3. Macro variable controls whether to show certain ADaM data or variable related information on Define.XML

```
%df_xml(study          = %str(123-4567),
        descr          = %str(A Phase 1 Study Evaluating.....),
        studynumber    = %str(1234567),
        adpp           = Y,
        cohort         = ,
        taskfmtvarlst  = %str('DISCN', 'LBCAT', .....),
        numordlst      = %str('AVISITN', 'DISCN'.....),
        export         = Y,
        adata          = Y,
        opencdisc      =
        );
```

CONCLUSION

It is a stable method with simple design.
 It is an opportunity for less manual typing and checking in Excel Tabs
 It is an opportunity for clearer mind of ADaM data structure, naming convention, variable deriving process.
 It is an opportunity for less time to build Define.XML for ADaM in the long run

REFERENCES

CDISC Metadata Submission Guidelines, www.cdisc.org/sdtm
 Analysis Data Model (ADaM) v2.1 and Implementation Guide v1.0, www.cdisc.org/adam
 FDA supports Define-XML v2 for CDER, CBER and CDRH, starting August 7, 2013, <http://www.fda.gov/ForIndustry/DataStandards/StudyDataStandards/default.htm>

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

Other brand and product names are trademarks of their respective companies

CONTACT INFORMATION

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

Yiwen Li
 Statistical Programming
 Gilead Sciences, Inc.
 300 Lakeside Drive, Foster City, CA 94404
 Office: 650-522-5780
 Email: Yiwen.Li@Gilead.com