

Generate Define.xml & Define.pdf from Metadata Environment

Rohit Banga

ABSTRACT

The FDA requires a define file (CRT-DDS or define.xml) as a part of the Case Report Tabulation folder for eCTD submission. The define file can contain the metadata for both tabulation and analysis datasets. The XML schema used to define the expected structure is based on an extension to the CDISC Operational Data Model (ODM).

This paper describes the SAS functions which are used to add metadata from an external SDTM conversion specification sheet (e.g. excel or csv) into the metadata repository, extract metadata from that repository as an XML file, map the resulting XML file into SAS datasets and finally generate define.xml & define.pdf from the SAS datasets. These functions deploy SAS Open Metadata Architecture.

A centralized metadata repository facilitates exploitation of the metadata in the following ways:

- generating the define.xml or define.pdf
- comparing metadata across studies to ensure consistency
- metadata checks to ensure compliance to the SDTM standard

Creating the study definition upstream and utilizing metadata this way provides a powerful and robust method to achieve high quality and consistent SDTM datasets & define.xml.

INTRODUCTION

Define.xml (CRT-DDS)

Define.xml describes the content and structure of clinical data in a machine readable format. The XML schema used to define the expected structure of define.xml is based on an extension to the CDISC Operational Data Model (ODM). ODM is a specification of a standard XML schema for the interchange and archive of clinical trials data and metadata. ODM is vendor neutral and platform independent and is compatible with a wide range of current, legacy and emergent clinical trials systems.

Define.xml is specified in FDA's electronic submission (eSub) guidance and the electronic Common Technical Document (eCTD) documents. Therefore it is mandatory to include define.xml within the electronic submission of a clinical trial to FDA. The formal name of define.xml is **Case Report Tabulation Data Definition Specification (CRT-DDS)**. This data definition specification provides a list of the datasets included in the submission along with a detailed description of the contents of each dataset.

The data definition specification can also be provided in a printable format (define.pdf) for the ease of the reviewers.

This paper describes a methodology to generate both define.xml and define.pdf from a common and centralized metadata repository. The define.xml can contain metadata for a wide range of standards including CDISC Study Data Tabulation Model (SDTM), CDISC Analysis Dataset Model (ADaM), and Standard for Exchange of Non-clinical Data (SEND). However, the scope of this paper is limited to CDISC SDTM.

CONTENTS OF DEFINE

The define.xml describes metadata at 5 Levels

1. Table level metadata
The metadata at the table level consists of:
 - Name of the table
 - Description of the table
 - Class of the table
 - Structure of the table
 - Purpose of the table

PhUSE 2009

- Keys of the table
- Location of the included XPT files.

Datasets for Study						
Dataset	Description	Class	Structure	Purpose	Keys	Location
AE	Adverse Events	Events	One record per adverse event per subject	Tabulation	STUDYID, USUBJID, AETERM, AESTDTC	AE.xpt

EXAMPLE OF TABLE LEVEL METADATA

2. Variable level metadata

The metadata at the variable level consists of:

- Name of the variable
- Label of the variable
- Type of the variable
- Controlled Terminology/Formats attached to the variable
- Origin of the variable
- Role of the variable
- Comments for that variable (can also list the computational algorithm used to derive the variable)

Variable	Label	Type	Controlled Terminology	Origin	Role	Comment
STUDYID	Study Identifier	text		Protocol, CRF Page 1	IDENTIFIER	The STUDYID variable has a fixed format: 'XXXX-YYYY', where 'XXXX' indicates the 4-digit compound code and the 'YYYY' the 4-digit study code
DOMAIN	Domain Abbreviation	text	DOMAIN	Assigned	IDENTIFIER	
USUBJID	Unique Subject Identifier	text		Derived	IDENTIFIER	The USUBJID variable has a fixed format: 'XXXX-YYYY-ZZZZ', where 'XXXX' indicates the 4-digit compound code, 'YYYY' the 4-digit study code and 'ZZZZ' the 5-digit patient code

EXAMPLE OF VARIABLE LEVEL METADATA

3. Value level metadata

The metadata at the value level consists of:

- Name of the TESTCD/QNAM variable
- Value of the TESTCD/QNAM i.e. TEST/QLABEL
- Type of the value
- Controlled Terminology/Formats attached to the value
- Origin of the value
- Role of the value
- Comments for that value (can also list the computational algorithm used to derive the value)

Value Level Metadata (ValueList.VS.VSTESTCD)							
Source Variable	Value	Label	Type	Controlled Terminology	Origin	Role	Comment
VSTESTCD	BMI	BODY MASS INDEX	text		Derived		See Computational Method: COMPMETHOD.BMI
VSTESTCD	DIABP	DIASTOLIC BLOOD PRESSURE	text		CRF Page 24		
VSTESTCD	HEIGHT	HEIGHT	text		CRF Page 22		
VSTESTCD	PULSE	PULSE RATE	text		CRF Page 24		
VSTESTCD	SYSBP	SYSTOLIC BLOOD PRESSURE	text		CRF Page 24		
VSTESTCD	WEIGHT	WEIGHT	text		CRF Page 22		

EXAMPLE OF VALUE LEVEL METADATA

4. Controlled Terminology metadata
 The metadata at the controlled terminology level consists of:
- Controlled terminology name (Codelist/Format name)
 - Code value
 - Code text

SEX, Reference Name (SEX)	
Code Value	Code Text
F	FEMALE
M	MALE

EXAMPLE OF CONTROLLED TERMINOLOGY METADATA

5. Computational Algorithm metadata
 The metadata at computational algorithm level consists of:
- Computational algorithm name
 - Name of the computational algorithm
 - Description of the computation method

Computational Algorithms (COMPMETHOD.AGE)	
Reference Name	Computation Method
COMPMETHOD.AGE	Equals to (DM.RFSTDTC-DM.BRTHDTC)/365.25

EXAMPLE OF COMPUTATIONAL ALGORITHM METADATA

SAS OPEN METADATA ARCHITECTURE

The centralized metadata repository used to generate define.xml and define.pdf is based on SAS Open Metadata Architecture.

SAS Open Metadata Architecture is a general-purpose metadata management facility that provides common metadata services to SAS applications. The metadata architecture provides:

- SAS Metadata Server - provides a central location for storing metadata.
- SAS Open Metadata Interface - provides access to the server from a variety of programming environments, including Java, COM/DCOM, and SAS.
- SAS Metadata Model - provides a common, centralized method of searching and managing metadata.
- XML transport format and XML representation of metadata - makes it easy to transform the metadata to HTML and other standard XML representations, like the Object Management Group's Common Warehouse Metamodel (CWM).

The open metadata architecture enables users to perform metadata management tasks from a variety of programming environments such as Base SAS, SAS Data Integration Studio, and SAS Management Console. The PROC Metadata facility allows users to send XML requests to a metadata server to add, delete and update metadata. Users can also get metadata information in an XML format using PROC Metadata. The use of PROC Metadata to add/get metadata information is described in following sections.

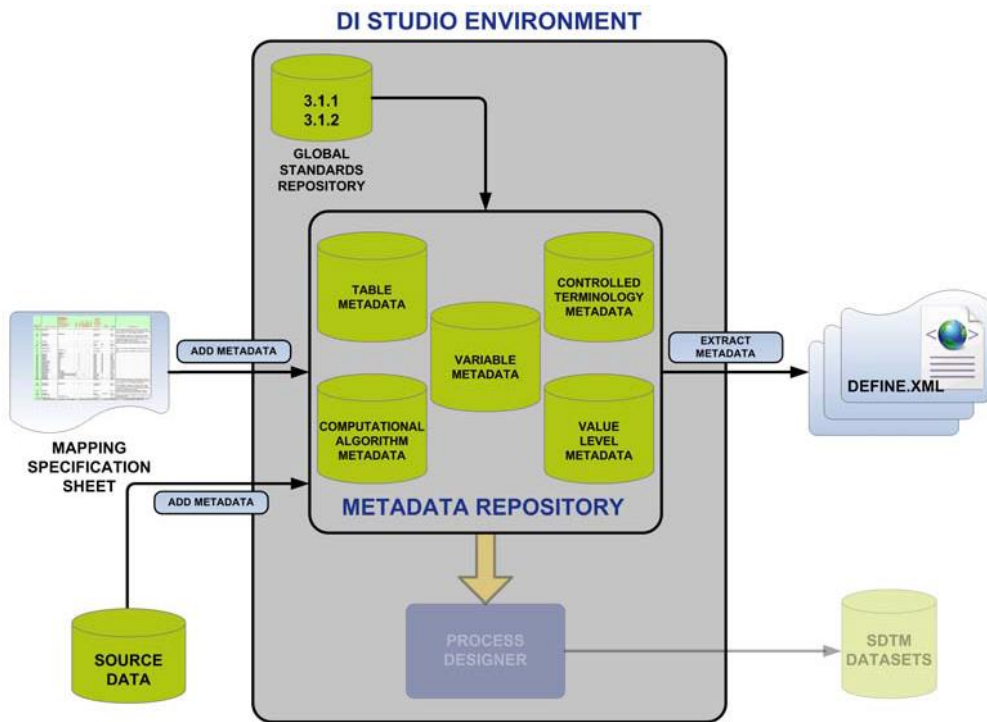


SAS OPEN METADATA ARCHITECTURE

GENERATE DEFINE.XML

GENERATING DEFINE.XML – METADATA PROCESS

The following diagram describes the process of creating define.xml and define.pdf from a centralized metadata repository. The metadata for a particular clinical trial is described in a metadata specification sheet. The metadata is registered in the central metadata repository using the “<AddMetadata>” method. The centralized metadata repository stores metadata of all clinical trials related to a particular compound at one location. The metadata is extracted from this repository using the “<GetMetadataObjects>” method. The result is then converted into SAS datasets using SAS XML Mapper. These SAS datasets are used to generate define.xml and define.pdf using PROC Template and PROC Report respectively.



GENERATE DEFINE FROM CENTRALIZED METADATA REPOSITORY – METADATA PROCESS

The individual elements of the process are described as follows:

METADATA SPECIFICATION TABLE

The metadata specification table is the entry point of the process. The five levels of metadata – table metadata, variable metadata, value level metadata, controlled terminology and computational algorithms, as discussed earlier are defined in a mapping specification sheet. The metadata specification sheet can be in excel, text, CSV or any other format which SAS is able to import.

Dataset	Description	Class	Structure	Purpose	Keys	
AE	Adverse Events	Events	One record per adverse event per subject	Tabulation	STUDYID, USUBJID, AETERM, AESTDTC	
BA	Bleeding Event Assessment	Findings	One record bleeding event assessment per subject	Tabulation	STUDYID, USUBJID, BATESTCD, VISITNUM, BASPID	
BE	Bleeding Event	Events	One record per bleeding event per subject	Tabulation	STUDYID, USUBJID, BESPID	
BP	Diagnostic Assessments	Findings	One record per test code	Tabulation	STUDYID, USUBJID, BPTSTCD, VISITNUM, BPSPID, BPCAT	
BT	Blood Transfusion	Interventions	One record per blood transfusion per subject	Tabulation	STUDYID, USUBJID, BTTRT, BTSTDT, BTPSID	
CA	Central Assessments	Findings	One record per assessment per visit per subject	Tabulation	STUDYID, USUBJID, CATESTCD, VISITNUM, CACAT, CASCAT, CALOC, CASPID	
CM	Concomitant Medications	Interventions	One record per medication intervention episode per subject	Tabulation	STUDYID, USUBJID, CMTRT, CMSTDT, CMCAT, CMINDC	
CD	Comments	Special Purpose	One record per comment per subject	Tabulation	STUDYID, USUBJID, FDCOMANI, DVARVAL	
CF	Compliance	Findings	One record per test code	Tabulation	STUDYID, USUBJID, CPTSTCD, VISITNUM	
DM	Demographics	Special Purpose	One record per subject	Tabulation	STUDYID, USUBJID	
DX	Disposition	Events	One record per disposition status or protocol milestone per subject	Tabulation	STUDYID, USUBJID, DSSTDT, EPOCH	
ES	Exposure	Interventions	One record per constant dosing interval per subject	Tabulation	STUDYID, USUBJID, EXSTDT, EXTRT	
HO	Hospitalization	Events	One record per hospitalization event per subject	Tabulation	STUDYID, USUBJID, HOTERM	
IE	Inclusion/Exclusion - Exceptions	Findings	One record per Inclusion/Exclusion criteria exception per subject	Tabulation	STUDYID, USUBJID, IETESTCD	
IM	Intake of Investigational Medication	Interventions	One record per intake of investigational medication per subject	Tabulation	STUDYID, USUBJID, IMTRT, MSPID	
LB	Laboratory Test Results	Findings	One record per lab test result point per visit per subject	Tabulation	STUDYID, USUBJID, LBTESTCD, VISITNUM, LBSTC	
MH	Medical History	Events				
RELREC	Related Records	Special Purpose F				
PS	Pharmacokinetic/Pharmacodynamic Sampling	Findings				
SE	Subject Elements	Trial Design				
SR	Surgery	Events				
SU	Substance Use	Interventions				
SUPFAE	Supplemental Qualifiers	Special Purpose F	AE STUDYID	Protocol	Req	The STUDYID variable has a fixed format: '0000-YYYY', where '0000' indicates the 4-digit compound code and the 'YYYY' the 4-digit study code
SUPFBA	Supplemental Qualifiers	Special Purpose F	AE DOMAIN	Assigned	Req	
SUPFBE	Supplemental Qualifiers	Special Purpose F	AE USUBJID	Derived	Req	The USUBJID variable has a fixed format: '0000-YYYY-ZZZZZ', where '0000' indicates the 4-digit compound code, 'YYYY' the 4-digit study code and 'ZZZZZ' the 5-digit patient code
SUPFBP	Supplemental Qualifiers	Special Purpose F		Derived	Req	Sequence number (automatically generated) to ensure uniqueness within a dataset for a subject
SUPFBT	Supplemental Qualifiers	Special Purpose F		CRF	Req	
SUPFCA	Supplemental Qualifiers	Special Purpose F	AE ASEQ	Derived	Req	
SUPFDM	Supplemental Qualifiers	Special Purpose F	AE AESPID	CRF	Req	
SUPFDS	Supplemental Qualifiers	Special Purpose F	AE AETERM	CRF	Req	
SUPFEX	Supplemental Qualifiers	Special Purpose F	AE AEDECOD	Assigned	Req	
SUPFPM	Supplemental Qualifiers	Special Purpose F	AE AEBODSYS	AEDICT_F	Req	AE term decoded to MedDRA terminology version 11.1
SUPFSR	Supplemental Qualifiers	Special Purpose F	AE AEBODSYS	AEDICT_F	Req	AE term decoded to MedDRA terminology version 11.1
SUPFMH	Supplemental Qualifiers	Special Purpose F	AE AESV	AESV	Req	
SY	Subject Visits	Trial Design	AE AESER	NYS	Req	
TA	Trial Arms	Trial Design	AE AEACN	AEACN	Req	
TE	Trial Elements	Trial Design	AE AEREL	NV	Req	
TI	Trial Inclusion/Exclusion Criteria	Trial Design	AE AEDOUT	OUT	Req	
TS	Trial Summary	Trial Design	AE AESCORG	NV	Req	
TV	Trial Visits	Trial Design	AE AESDASAB	NV	Req	
VS	Vital Signs	Findings	AE AESDTH	NV	Req	
VA	Wound Assessment	Findings	AE AESHSP	NV	Req	
			AE AESMRE	NV	Req	
			AE AECOMTRT	NV	Req	
			AE AESDTC	ISO 8601	Req	
			AE AESTDTC	ISO 8601	Req	
			AE AESTDY	COMPNETHCD, AESTDY	Req	
			AE AESTDY	COMPNETHCD, AESTDY	Req	
			AE AEDUP	ISO 8601	Req	
			AE AEBURF	ISO 8601	Req	
			AE AEBURF	ISO 8601	Req	
			CM STUDYID	Protocol	Req	The STUDYID variable has a fixed format: '0000-YYYY', where '0000' indicates the 4-digit compound code and the 'YYYY' the 4-digit study code
			CM DOMAIN	Assigned	Req	
			CM USUBJID	Derived	Req	The USUBJID variable has a fixed format: '0000-YYYY-ZZZZZ', where '0000' indicates the 4-digit compound code, 'YYYY' the 4-digit study code and 'ZZZZZ' the 5-digit patient code
			CM CMSEQ	Derived	Req	Sequence number (automatically generated) to ensure uniqueness within a dataset for a subject
			CM CMTRT	CRF	Req	
			CM CMDECOD	Assigned	Req	CM term decoded to WHODD terminology version 11.1
			CM CMDECOD	DRUGDICT_F	Req	CM term decoded to WHODD terminology version 11.1

EXAMPLE METADATA SPECIFICATION SHEET

ADD METADATA

The next step is to add the metadata defined in the metadata specification sheet to the centralized metadata repository. Since the repository is based on SAS open metadata architecture, it has an XML transport engine. Therefore to add metadata to the metadata repository, requests are sent to the server in a XML format. The XML string is constructed keeping in mind the rules of SAS metadata model.

This is a snippet of the program used to Add Metadata in the centralized repository.

```

%Macro Addtablemetadata (Extension =, Value=, TableId =);
data _null_;
File addmeta;
PUT '<AddMetadata>';
PUT '<Metadata>';
PUT "<Extension Id      = '#####'";
PUT "      Name = '#####&Extension'";
PUT "      Value = '#####&Value' ">";
PUT '<owningobject>';
PUT "  <PhysicalTable objref= '#####&TableId' />";
PUT '</owningobject>';
PUT '</Extension>';
PUT '</Metadata>';
PUT '<Reposid>$METAREPOSITORY</Reposid>';
PUT '  </NS>SAS</NS>';
    
```

PhUSE 2009

```
PUT '<Flags>268435456</Flags>';
PUT '<Options/>';
PUT '</AddMetadata>';
Run;

Proc Metadata Repos = "&REP_ID." In = addmeta;
Run;

%Mend Addtablemetadata;
```

The key points to note in this program are the <AddMetadata> method and PROC METADATA procedure. In the example, <AddMetadata> is the method name and <Metadata>, <Reposid>, <NS>, <Flags>, <Options> are AddMetadata parameters. The XML elements that are nested within the <Metadata></Metadata> tags comprise a metadata property string that defines the metadata object to be added.

The metadata of all clinical trials for a particular compound can be added to this central metadata repository. The <AddMetadata> call can be issued from a variety of programming environments as long as the programming environment is based on SAS Open Metadata Architecture.

GET METADATA

The next step is to extract desired metadata from the metadata repository. For extracting the metadata, <GetMetadataObjects> method is used. <XMLSelect> search string is used to extract the relevant metadata. The GetMetadata request is sent to the SAS metadata server using PROC Metadata.

Listed below is a snippet of the program used to get metadata information from the centralized repository:

```
FILENAME INPUT TEMP;
FILENAME OUTPUT "...\output.xml";

data _null_;
file Input;
  put '<GetMetadataObjects>';
  put '<Reposid>$METAREPOSITORY</Reposid>';
  put '<Type>PhysicalTable</Type>';
  put '<Objects/>';
  put '<NS>SAS</NS>';
  put '<Flags>388</Flags>';
  put '<Options>';
  put '<XMLSelect search="PhysicalTable[@SASTableName = '@;
  put "' " &NAME" "']/TablePackage/SASLibrary[@Libref = " @;
  put "' " &libref" "' " @;
  put ']]"/>';
  put '<Templates>';
  put '<PhysicalTable Id="" Name="" Desc="" SASTableName="" >';
  put '<Columns/>';
  put '<Extensions/>';
  put '<Notes/>';
  put '</PhysicalTable>';
  put '<Column ID="" Name="" Desc="" SASColumnLength="" SASColumnType=""';
  put '          SASColumnName="" SASFormat="" SASInformat="">';
  put '          <Extensions/> <Notes/> </Column>';
  put '<Extension ID="" Name="" Desc="" Value=""/>';
  put ' <TextStore ID="" Name="" TextType="" StoredText="" />';
  put ' </Templates>';
  put ' </Options>';
  put '</GetMetadataObjects>;run;

Proc Metadata Repos = "&Rep_ID." in=input out=results;
run;
```

PhUSE 2009

The <Reposid> element identifies the repository to query. The <type> element identifies the type of the metadata object to query. The metadata objects that were found are listed in the <Objects> element. The structure of the resulting xml file is determined by the <Templates> attribute. The <XMLSelect> element enables passing a search string to the server. In this example a physical table is queried. The result of the above program is returned in an xml file (filename "results").

The result.xml is as shown in the diagram below:

```
- <GetMetadataObjects>
  <Reposid>A0000001.ASTG5IXN</Reposid>
  <Type>PhysicalTable</Type>
- <Objects>
  - <PhysicalTable Id="ASTG5IXN.AR000001" Name="Adverse Events" Desc="SDTM 3.1.2 Global Standards Table" SASTableName="AE">
    - <Columns>
      - <Column Id="ASTG5IXN.AS000001" Name="STUDYID" Desc="Study Identifier" SASColumnLength="200" SASColumnType="C" SASColumnName="STUDYID"
        SASFormat="" SASInformat="">
        - <Extensions>
          <Extension Id="ASTG5IXN.AN0001G4" Name="CORE" Desc="" Value="Req" />
          <Extension Id="ASTG5IXN.AN0001G5" Name="ORIGIN" Desc="" Value="Protocol, CRF Page 1" />
          <Extension Id="ASTG5IXN.AN0001G6" Name="ROLE" Desc="" Value="Identifier" />
          <Extension Id="ASTG5IXN.AN0001G7" Name="TERM" Desc="" Value="" />
        </Extensions>
        - <Notes>
          <TextStore Id="ASTG5IXN.A900012A" Name="References" TextType="PlainText" StoredText="The STUDYID variable has a fixed format: 'XXXX-
            YYYY', where 'XXXX' indicates the 4-digit compound code and the 'YYYY' the 4-digit study code" />
        </Notes>
      </Column>
      - <Column Id="ASTG5IXN.AS000002" Name="DOMAIN" Desc="Domain Abbreviation" SASColumnLength="200" SASColumnType="C" SASColumnName="DOMAIN"
        SASFormat="" SASInformat="" MetadataCreated="27Aug2009:08:14:00" MetadataUpdated="27Aug2009:08:14:00">
        - <Extensions>
          <Extension Id="ASTG5IXN.AN0001G8" Name="CORE" Desc="" Value="Req" />
          <Extension Id="ASTG5IXN.AN0001G9" Name="ORIGIN" Desc="" Value="Assigned" />
          <Extension Id="ASTG5IXN.AN0001GA" Name="ROLE" Desc="" Value="Identifier" />
          <Extension Id="ASTG5IXN.AN0001GB" Name="TERM" Desc="" Value="DOMAIN" />
        </Extensions>
        - <Notes>
          <TextStore Id="ASTG5IXN.A900012B" Name="References" TextType="PlainText" StoredText="" />
        </Notes>
      </Column>
      - <Column Id="ASTG5IXN.AS000003" Name="USUBJID" Desc="Unique Subject Identifier" SASColumnLength="200" SASColumnType="C"
        SASColumnName="USUBJID" SASFormat="" SASInformat="" MetadataCreated="27Aug2009:08:14:00" MetadataUpdated="27Aug2009:08:14:00">
        - <Extensions>
          <Extension Id="ASTG5IXN.AN0001GC" Name="CORE" Desc="" Value="Req" />
          <Extension Id="ASTG5IXN.AN0001GD" Name="ORIGIN" Desc="" Value="Protocol" />
          <Extension Id="ASTG5IXN.AN0001GE" Name="ROLE" Desc="" Value="Identifier" />
          <Extension Id="ASTG5IXN.AN0001GF" Name="TERM" Desc="" Value="" />
        </Extensions>
        - <Notes>
          <TextStore Id="ASTG5IXN.A900012C" Name="References" TextType="PlainText" StoredText="The USUBJID variable has a fixed format: 'XXXX-
            YYYY-ZZZZ', where 'XXXX' indicates the 4-digit compound code, 'YYYY' the 4-digit study code and 'ZZZZZ' the 6-digit patient code" />
        </Notes>
      </Column>
    </Columns>
  </PhysicalTable>
</Objects>
```

RESULT.XML

The next step is to convert the result.xml output file into SAS Datasets.

MAP XML TO SAS DATASETS

The conversion of result.xml output file into SAS datasets is achieved with the help of a map file generated by SAS XML Mapper. Since the xml structure of result.xml is always the same (as determined by the <template> element in the previous step) the map file needs to be generated just once and it will always convert result.xml into the required SAS Datasets.

PhUSE 2009

```
<?xml version="1.0" encoding="windows-1252"?>
<!-- ##### -->
<!-- 2008-11-19T11:50:24 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XML Mapper, 9.1.0335.20070926.1036 -->
<!-- ##### -->
<SXLEMAP version="1.0">
  <!-- ##### -->
  <TABLE name="TABLE_METADATA">
    <TABLE_LABEL>Table Metadata</TABLE_LABEL>
    <TABLE_XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef</TABLE_XPATH>
    <COLUMN name="SASTableName">
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/@Name</XPATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>6</LENGTH>
    </COLUMN>
    <COLUMN name="Description">
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/@def:Label
    </XPATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>35</LENGTH>
    </COLUMN>
    <COLUMN name="Class">
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/@def:Class
    </XPATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>28</LENGTH>
    </COLUMN>
    <COLUMN name="Structure">
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/@def:Structure
    </XPATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>74</LENGTH>
    </COLUMN>
  </TABLE>
</SXLEMAP>
```

EXAMPLE MAP FILE GENERATED BY SAS XML MAPPER

Once the map file is generated the following piece of code converts the resulting xml file into SAS Datasets

```
filename myxmlmap "&define.\tempdata\tablemetadata.map";
libname tempdata xml xmlmap=myxmlmap xmlfileref=results;

Proc Copy in = tempdata out = work;
Run;
```

In the program, filename "myxmlmap" is given to the map file. An xml library "tempdata" is defined and the name of the map file and the output file "result.xml" is specified in the options. The PROC Copy procedure copies all the datasets from the XML library into the work library. These SAS Datasets can be manipulated to generate define.xml & define.pdf.

PROC TEMPLATE

Proc Template can be used to generate define.xml from the resulting SAS Datasets created in the previous step. The event functionality in PROC Template makes dynamic templates. Various events can be triggered based on conditions in the dataset. Then these dynamic templates write XML tags according to the data present.

Listed below is a snippet of the program used to generate define.xml from SAS Datasets

```
Proc template;
Define tagset TMP.ComputationMethod;

parent = tagsets.sasxmog;
Notes "SAS XML Engine tagsets for SDTM ComputationMethod";

/* define and initialize variables */
mvar markup
    OID
    ItemOID;
```


PhUSE 2009

```
define event row;
  start:
    break;
  finish:
    do /if cmp($markup, "FIRST");
      trigger markupFirst;
    else /if cmp ($markup, "COMMENT");
      trigger writeComment;
    done;
    break;
end;

/* Use SASColumn event to load values into variables*/
define event SASColumn;
  start:
    break;
  finish:
    do /if cmp(NAME, "MARKUP");
      set $markup VALUE;
      break;
    else /if cmp(NAME, "OID");
      set $OID VALUE;
      break;
    else /if cmp (NAME, "ITEMOID");
      set $ItemOID VALUE;
    done;
    break;
end;

/* Define Tag writing events */

define event writeStart;
  putq "<def:ComputationMethod OID=" $OID ">" CR;
break;
end;

define event writeEnd;
  put "</def:ComputationMethod>" CR;
break;
end;

define event writeMid;
  put $ItemOID CR;
  break;
end;

define event writeComment;
  put "<!--" $ItemOID "-->" CR;
  break;
end;

define event markupFirst;
  put NL;
  trigger writeStart;
  trigger writeMid;
  trigger writeEnd;
  break;
end;

end;
run;
```

For the five different levels of metadata, five different templates would be needed. Two additional templates would be needed for header & footer information. The PROC Template would then create seven different individual xml files. These seven xml files could be combined to generate a complete define.xml.

PROC REPORT

PROC Report is used to generate define.pdf from the resulting SAS Datasets. The compute block can be used to generate hyperlinks within the define.pdf. The ODS option PROCLABEL and PROC Report option PROC Contents enable good control over bookmarks. However, external pdf procedures could be used to gain complete control over bookmarks. Kindly see the references for further information on moving and splitting bookmarks.

Listed below is a snippet of the program used to generate define.pdf from SAS Datasets.

```
ODS PROCLABEL = "Datasets for Study &STUDY_ID.";
PROC REPORT DATA=Table_Metadata nowd noheader nocenter SPLIT=' ' CONTENTS="SDTM Domains"
style(report) = [OUTPUTWIDTH = 100%];

columns SASTableName DESCRIPTION CLASS STRUCTURE PURPOSE KEYS XPORT;

define SASTableName/display width=30 "Dataset";
define DESCRIPTION/display width=30 "Description";
define CLASS/display width=10 "Class" ;
define STRUCTURE/display width=40 "Structure";
define PURPOSE/display width=40 "Purpose" ;
define KEYS/display width=30 "Keys" ;
define XPORT/display "Location" ;

COMPUTE XPORT;
URLSTRING = STRIP(XPORT);
CALL DEFINE (_COL_, 'URL', URLSTRING);
ENDCOMP;

RUN;
```

To generate all five levels of metadata five separate PROC Report procedures are needed. The compute block can also be used to color headers differently and also to generate page numbers.

CONCLUSION

The key concept in the methodology presented above is a centralized metadata repository. The centralized metadata repository integrates all the metadata and stores it in one meta model schema that can be accessed easily. The following are the advantages of a centralized metadata repository:

- Efficiently generate define.xml & define.pdf.
- Run metadata validation checks on the repository to ensure high quality of metadata.
- Compare metadata to ensure consistency across all trials and facilitate cross study analysis.
- Create actual SDTM datasets based on the metadata definition provided in the central repository.
- Metadata repository populated before actual programming begins, so SDTM metadata available to statisticians without actual SDTM datasets.

Creating the study definition upstream and utilizing metadata this way provides a powerful and robust method to achieve high quality and consistent SDTM datasets & define.xml, ensuring a successful submission.

REFERENCES

1. U.S. Department of Health and Human Services, Food and Drug Administration, Center for Drug Evaluation and Research (CDER), information on Electronic Common Technical Document (eCTD)
<http://www.fda.gov/Drugs/DevelopmentApprovalProcess/FormsSubmissionRequirements/ElectronicSubmissions/ucm153574.htm>
2. Case Report Tabulation Data Definition Specification (CRT-DDS, also called define.xml)
Final Version 1.0 <http://cdisc.org/models/def/v1.0/index.html>
3. Operational Data Model Final Version 1.2.1 <http://www.cdisc.org/models/odm/v1.2.1/index.html>
4. SAS Institute Inc., (2004): SAS® 9.1.3 XML LIBNAME Engine User's Guide, Cary NC
(http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/base_xmlug_10166.pdf#page=99)
5. CDISC SDTM Implementation Guide V3.1.1, August 26, 2005
(<http://www.cdisc.org/models/sdtm/v1.1/index.html>)
6. Getting Started with SAS® 9.1.3 Open Metadata Interface, Second Edition
http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/omd_gs_9971.pdf
7. Creating and Modifying PDF Bookmarks, Tikiri Karunasundera, Allergan Inc., Irvine CA
http://www.lexjansen.com/wuss/2006/data_presentation_and_business_intelligence/DPR-Karunasundera.pdf
8. Using the SAS® XML Mapper and ODS to create a PDF representation of the define.xml ,Lex Jansen, Octagon Research Solutions <http://www.lexjansen.com/phuse/2008/cd/cd04.pdf>

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.