PharmaSUG China 2021 - Paper AD-049

An Approach to Generate Define.xml for Submission in China Based on Python

Siyi Li, Sanofi

ABSTRACT

Data Definition File (Define) is one of the submission components of clinical trial data which is required by regulatory agencies. In July 2020, the Center for Drug Evaluation (CDE) released a guideline for the submission of clinical trial data. In the guideline, sponsor needs to provide clinical trial data including Data Definition File following certain specifications.

This paper presents a brief introduction about Data Definition File and the requirements of Data Definition File released by CDE. In addition, an approach to generate define.xml for China submission using Python is also introduced in the paper. Python is an object-oriented and open-source scripting programming language. It can be executed in any environment with Python installed. The define.xml complying CDE requirement can be easily generated with datasets and Chinese Metadata by Python. This paper gives a detailed description of developing and examples using this approach which may help you generate define.xml with little effort.

INTRODUCTION

Giving the fact that Python is more and more popular nowadays, we, statistical programmers should try to use python in our daily work. Using Python to generate define.xml is just an idea flashed into my mind, however it is a good way.

Before you start to generate define.xml for submission in China, you should (1) fully understand guideline from CDE, (2) be familiar with the Data Definition File and all of its components, and (3) get all related submission documents ready. Once you have completed the three tasks, you can write your own code to generate define.xml. I strongly recommend you read guideline of “CDISC Define-XML Specification Version 2.1” before you read this paper. The guideline will give you a comprehensive introduction about Define-XML. This information is also mentioned in this paper, but only limited part.

BACKGOUND

In July 2020, CDE released “Guideline on the Submission of Clinical Trial Data (for trial implementation)”. The principles in the guideline should be followed for critical clinical trials for registration and marketing. The guideline was effective from October 2020. In the guideline, the Data Definition File is required for submission. When submitting Data Definition File, at least the following content should be in Chinese: description/label and specification of each dataset in the database, description/label and derivation progress of variables in dataset and values or codes list of efficacy indicators.

Therefore, creating Data Definition File with part of the content in Chinese is necessary for sponsor if the study needs to be submitted in China.

DEFINE.XML AND ITS COMPONENTS

The Define-XML transmits metadata that describes any tabular dataset structure. It is to describe CDISC Study Data Tabulation Model (SDTM), Standard for Exchange of Nonclinical Data (SEND), and Analysis Data Model (ADaM) datasets for the purpose of submissions to regulatory authorities.

An XML file is an extensible markup language file, and it is used to structure data for storage and transport. In an XML file, there are both tags and text. The tags provide the structure to the data. The text with the information to store is surrounded by tags. An element is a logical document component that either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag. An element can include other elements, which are called child elements.

The following example is about an element which is used in define.xml:

```
<def:CommentDef OID="COM.VSSTRESU">
    <Description>
        <TranslatedText xml:lang="en">Standard units consistent with CDISC
    controlled terminology
        </TranslatedText>
    </Description>
</def:CommentDef>
```

The start-tag is "<def:CommentDef>", and the end-tag is "</def:CommentDef>" where the name of the element attribute is "OID", and its value is " COM.VSSTRESU". The element Description is a child element of element def:CommentDef. The text is " Standard units consistent with CDISC controlled terminology" in element TranslatedText.

Define file with XML format contains standard elements specified in "CDISC Define-XML Specification".

The elements are commonly used in SDTM Define, ADaM Define and SEND Define except some special elements. For instance, "def:AnnotatedCRF" is only used for SDTM Define.

Figure 1 shows the structure and some of the important elements in define.xml v2.1.

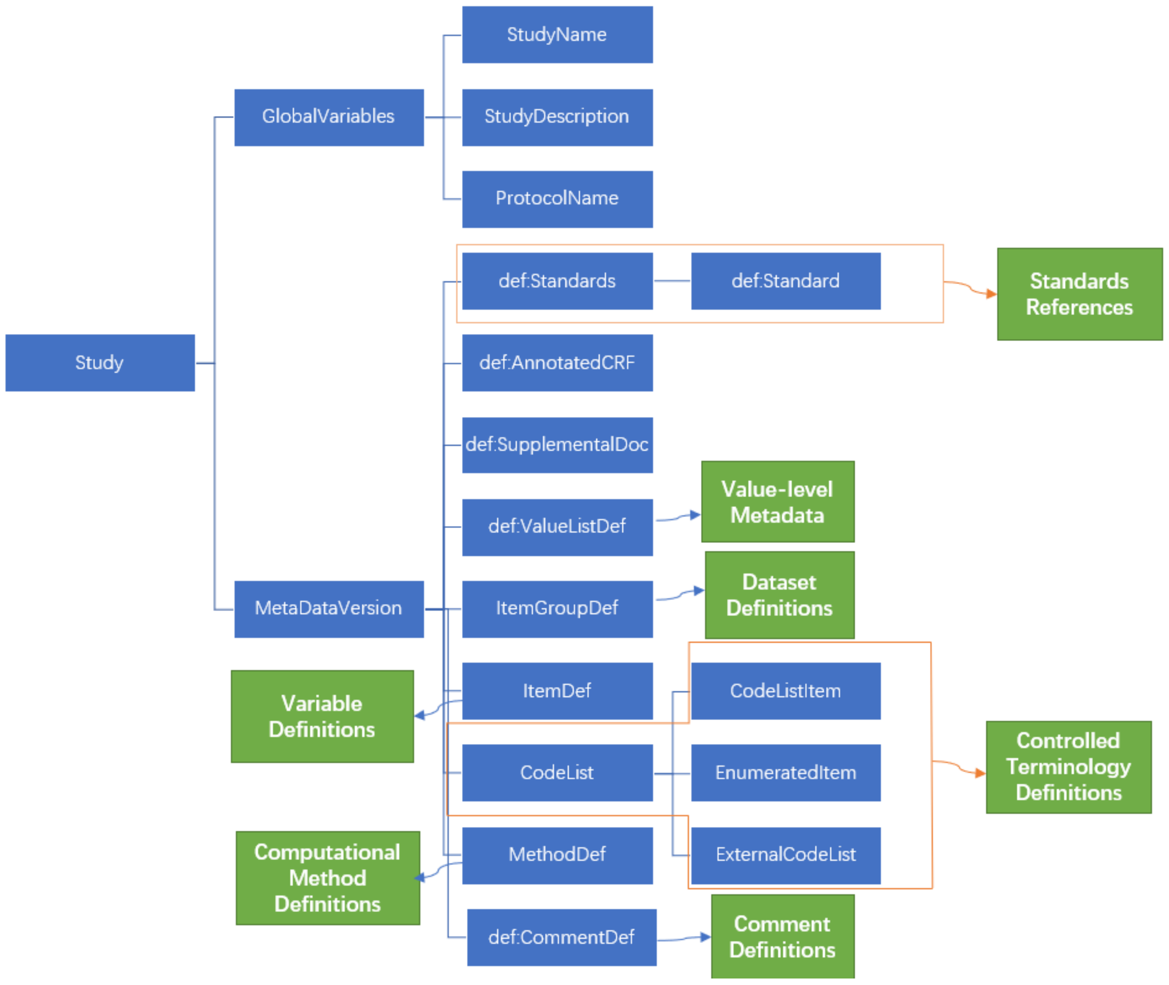


Figure 1. Structure and elements in define.xml v2.1

According to the figure, the blue box contains the name of each element, and the green box indicates information storing in each element. For some of the important elements, please see the following description for each of them.

GLOBALVARIABLES

The element GlobalVariables contains child elements that capture high-level study information. The child element StudyName, StudyDescription and ProtocolName contains information about the study name, description of the contents of the study and protocol name or number respectively. And for study submitted in China, you may prepare study description in Chinese.

The following elements are child elements of element MetaDataVersion.

DEF:STANDARDS AND DEF:STANDARD

The element def:Standards contains all standards used in study. And each standard is described in element def:Standard. For example, Implementation Guide (IG) version used for tabular dataset and Controlled Terminology version should be specified in element def:Standard.

DEF:ANNOTATEDCRF AND DEF:SUPPLEMENTALDOC

The information which these two elements contain can be seen from their names. The element def:AnnotatedCRF link to annotated Case Report Form (aCRF) for SDTM Define. And the element def:SupplementalDoc is used when there are supplemental documents need to be submitted such as Study Data Reviewers Guide (cSDRG) for a SDTM submission, Analysis Data Reviewers' Guide (ADRG) for an ADaM submission and computational algorithm.

ITEMGROUPDEF

The element ItemGroupDef and its child elements are used to describe the dataset metadata. The dataset-level metadata specifies the description, class, structure, purpose, documentations, domain level keys and other attribute of each dataset. Besides, the child elements of element ItemGroupDef are to link the variables including in each dataset and corresponding dataset SAS Xport Transport (XPT) files.

According to CDE’s guideline, the description and documentations for each dataset have to be Chinese.

ITEMDEF

The ItemDef element is used to represent variable metadata. The ItemDef element describes the properties of the variable, such as type, description, origin, length. For variable in SDTM dataset and collected from Case Report Form (CRF), the page number of aCRF related to variable needs to specify in child element. Also, the child elements of element ItemDef are to link corresponding value-level metadata, computational method, comments, controlled terminology, the supported document such as Data Reviewer’s Guide ang complex algorithms.

According to CDE’s guideline, the description and computational method for each variable have to be Chinese.

DEF:VALUELISTDEF

The def:ValueListDef element is used to describe value-level metadata. Value-level metadata should be provided in the define.xml document when the variable metadata does not provide sufficient detail to support data review and analysis. So, value-level metadata is always a specialization of variable metadata. The value-level metadata can be used in SDTM, ADaM and SEND domains. It is most often used within SDTM Findings domains and Trial Summary (TS) domain to provide definitions for variables. For example, in Vital Signs (VS) domain, the type of variable VSORRES is different between different tests. The test Temperature was collected as a floating point and the test Heart Rate was collected as an integer value. In this case, use variable-level metadata cannot provide detail information for different tests. So, the value-level metadata should be used in this situation. In ADaM, value-level metadata often describes AVAL or AVALC in Basic Data Structure (BDS) dataset based on values of PARAMCD. The def:ValueListDef element has similar structure with ItemGroupDef element. The child element of element def:ValueListDef can link the corresponding value definitions.

CODELIST

For each controlled terminology referenced by variable or valuelist, a CodeList element with the definition of the controlled terminology must be provided. If you use Medical Dictionary for Regulatory Activities

(MedDRA) or WHO Drug Dictionary (WHO-DD) to coding events or medications, you need to use the child element ExternalCodeList to describe controlled terminology.

The values or codes list of efficacy indicators should be provided in Chinese for China submission. The Chinese version dictionary of MedDRA and WHO-DD are released twice a year by official institutions. And the Chinses dictionary can be used to translate terms.

METHODDEF

A MethodDef element provides details about a computational algorithm that is used as part of a variable or value definition. And it can link to external documents when the derivation method is much complex, or the derivation rule is too long.

The derivation method for variables should be in Chinese in define.xml.

DEF:COMMENTDEF

The def:CommentDef element provides short comments that are contained in the Define-XML document or long comments referenced in external documents.

The comments need to be in Chinese when generating define.xml for submission.

RELATED DOCUMENTS PREPARATION AND STEPS ABOUT GENERATING DEFINE.XML

The below chart can help you simply understand the required documents and steps for generating define.xml:

1. Datasets	2. Supported documents	3. Metadata	4. Run Python code	5. Define.xml

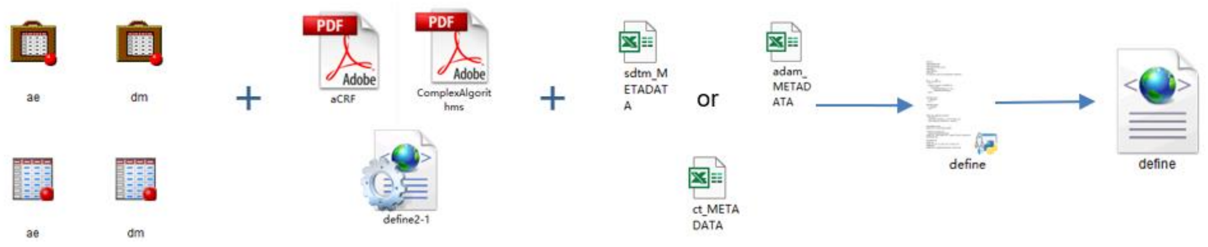


Figure 2. Flow chart about steps

Let’s focus on an example of SDTM Define generation. The following example using a test study to demonstrate how to generate define.xml of v2.1. The test study including DM domain, AE domain, EX domain and FAAE domain. After understanding this example, you will draw inferences about other cases from one instance to create SDTM Define for more domains and ADaM Define.

The detail steps are:

Prepare Datasets

The datasets need to be XPT format in submission package, but Chinese characters appears garbled in XPT format due to SAS Universal Viewer does not support any other encodings than ASCII for XPT files and files cannot be read in Python. So, datasets with SAS Data Set format (SAS7BDAT) are also needed.

Get Supplemental Files for Submission Ready

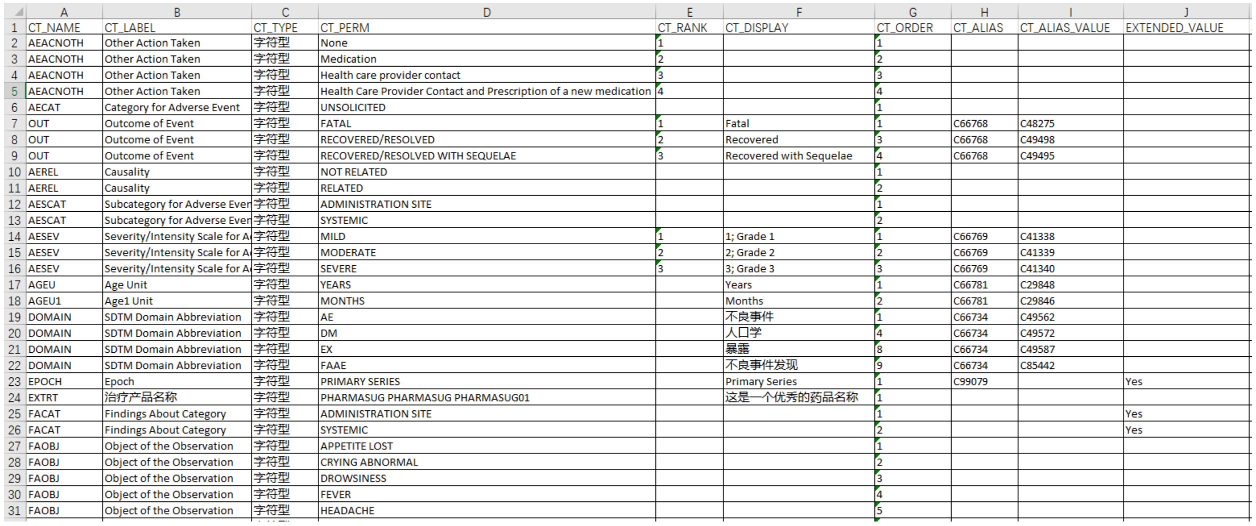
We have aCRF, cSDRG and Complex Algorithm (CA) for test study. These documents should be PDF format in submission package. All files should be in Chinese.

The “define2-1.xsl” is Extensible Stylesheet Language (XSL) file for define.xml. It should be put in the same folder with the define.xml when submission. You can download it from CDISC website. With this file, define.xml can be easily viewed in a specific style in the web browser. The version of XSL file should match the define.xml structure. If you choose stylesheet in version 2.1, the elements in define.xml should comply “CDISC Define-XML Specification Version 2.1” guideline. It should be pointed out that the stylesheet is more suitable for the English version of the Define rather than Chinese version. So, you may need to update some items after downloading it from CDISC. For example, if you want to display “ISO8601” in “Controlled Terms or ISO Format” column for timing variables in web browser for define.xml when the web browser displays “Type” of variables in Chinese, you need to add the following item marked in red:

```
<xsl:if
  test="$itemDef/@DataType='date' or
  …
  $itemDef/@DataType='日期型' or
  $itemDef/@DataType='日期时间型'">
  <xsl:text>ISO 8601</xsl:text>
</xsl:if>
```

Metadata is Required and Important for Creating Define.xml

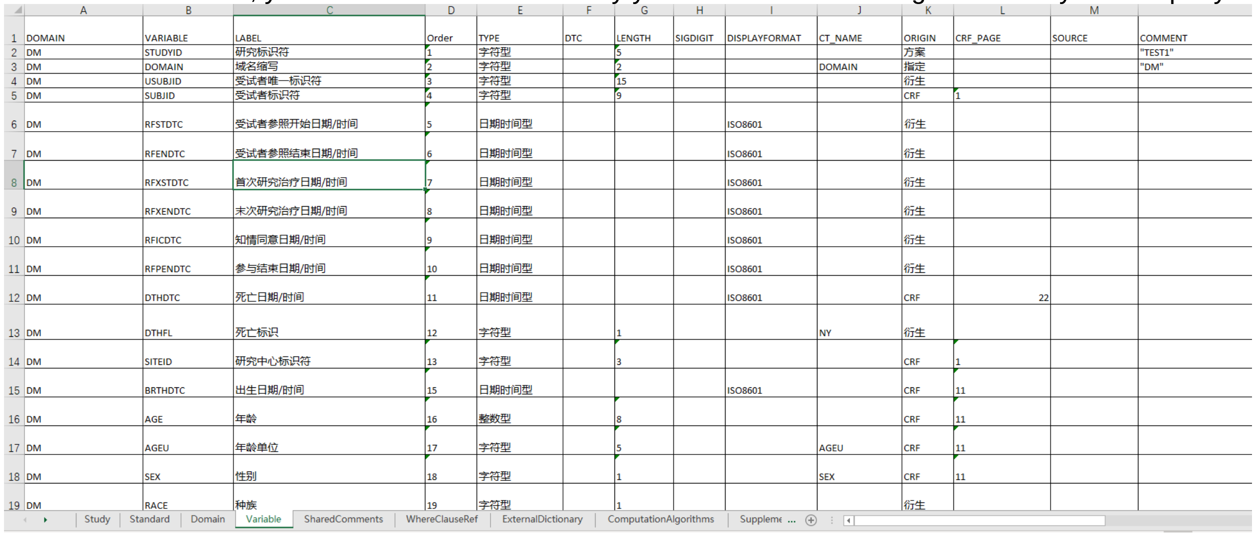
Control Terminology metadata is required for both SDTM Define and ADaM Define. Please pay attention to the efficacy indicators, they must be translated into Chinese for China submission.



	A	B	C	D	E	F	G	H	I	J
1	CT_NAME	CT_LABEL	CT_TYPE	CT_PERM	CT_RANK	CT_DISPLAY	CT_ORDER	CT_ALIAS	CT_ALIAS_VALUE	EXTENDED_VALUE
2	AEACNOTH	Other Action Taken	字符型	None	1		1			
3	AEACNOTH	Other Action Taken	字符型	Medication	2		2			
4	AEACNOTH	Other Action Taken	字符型	Health care provider contact	3		3			
5	AEACNOTH	Other Action Taken	字符型	Health Care Provider Contact and Prescription of a new medication	4		4			
6	AECAT	Category for Adverse Event	字符型	UNSOLICITED			1			
7	OUT	Outcome of Event	字符型	FATAL	1	Fatal	1	C66768	C48275	
8	OUT	Outcome of Event	字符型	RECOVERED/RESOLVED	2	Recovered	3	C66768	C49498	
9	OUT	Outcome of Event	字符型	RECOVERED/RESOLVED WITH SEQUELAE	3	Recovered with Sequelae	4	C66768	C49495	
10	AEREL	Causality	字符型	NOT RELATED			1			
11	AEREL	Causality	字符型	RELATED			2			
12	AESCAT	Subcategory for Adverse Even	字符型	ADMINISTRATION SITE			1			
13	AESCAT	Subcategory for Adverse Even	字符型	SYSTEMIC			2			
14	AESEV	Severity/Intensity Scale for A	字符型	MILD	1	1; Grade 1	1	C66769	C41338	
15	AESEV	Severity/Intensity Scale for A	字符型	MODERATE	2	2; Grade 2	2	C66769	C41339	
16	AESEV	Severity/Intensity Scale for A	字符型	SEVERE	3	3; Grade 3	3	C66769	C41340	
17	AGEU	Age Unit	字符型	YEARS		Years	1	C66781	C29848	
18	AGEU1	Age1 Unit	字符型	MONTHS		Months	2	C66781	C29846	
19	DOMAIN	SDTM Domain Abbreviation	字符型	AE		不良事件	1	C66734	C49562	
20	DOMAIN	SDTM Domain Abbreviation	字符型	DM		人口学	4	C66734	C49572	
21	DOMAIN	SDTM Domain Abbreviation	字符型	EX		暴露	8	C66734	C49587	
22	DOMAIN	SDTM Domain Abbreviation	字符型	FAAE		不良事件发现	9	C66734	C85442	
23	EPOCH	Epoch	字符型	PRIMARY SERIES		Primary Series	1	C99079		Yes
24	EXTRT	治疗产品名称	字符型	PHARMASUG PHARMASUG PHARMASUG01		这是一个优秀的药品名称	1			
25	FACAT	Findings About Category	字符型	ADMINISTRATION SITE			1			Yes
26	FACAT	Findings About Category	字符型	SYSTEMIC			2			Yes
27	FAOBJ	Object of the Observation	字符型	APPETITE LOST			1			
28	FAOBJ	Object of the Observation	字符型	CRYING ABNORMAL			2			
29	FAOBJ	Object of the Observation	字符型	DROWSINESS			3			
30	FAOBJ	Object of the Observation	字符型	FEVER			4			
31	FAOBJ	Object of the Observation	字符型	HEADACHE			5			

Display 1. Example of CT metadata

For test study, please see the following Display 2 of the structure for SDTM metadata. The Chinese dataset label and variable label can be found in Chinese version IG, but if you have some special domains or variables, you have to translate labels by yourself and reach an agreement in your company.



DOMAIN	VARIABLE	LABEL	Order	TYPE	DTC	LENGTH	SIGDIGIT	DISPLAYFORMAT	CT_NAME	ORIGIN	CRF_PAGE	SOURCE	COMMENT
DM	STUDYID	研究标识符	1	字符型		5				方案			"TEST1"
DM	DOMAIN	域名缩写	2	字符型		2			DOMAIN	指定			"DM"
DM	USUBJID	受试者唯一标识符	3	字符型		15				衍生			
DM	SUBJID	受试者标识符	4	字符型		9				CRF	1		
DM	RFSTDTC	受试者参照开始日期/时间	5	日期时间型				ISO8601		衍生			
DM	RFENDTC	受试者参照结束日期/时间	6	日期时间型				ISO8601		衍生			
DM	RFXSTDTC	首次研究治疗日期/时间	7	日期时间型				ISO8601		衍生			
DM	RFXENDTC	末次研究治疗日期/时间	8	日期时间型				ISO8601		衍生			
DM	RFICDTC	知情同意日期/时间	9	日期时间型				ISO8601		衍生			
DM	RFPENDTC	参与结束日期/时间	10	日期时间型				ISO8601		衍生			
DM	DTHDTC	死亡日期/时间	11	日期时间型				ISO8601		CRF	22		
DM	DTHFL	死亡标识	12	字符型		1			NY	衍生			
DM	SITEID	研究中心标识符	13	字符型		3				CRF	1		
DM	BRTHDTC	出生日期/时间	15	日期时间型				ISO8601		CRF	11		
DM	AGE	年龄	16	整数型		8				CRF	11		
DM	AGEU	年龄单位	17	字符型		5			AGEU	CRF	11		
DM	SEX	性别	18	字符型		1			SEX	CRF	11		
DM	RACE	种族	19	字符型		1				衍生			

Display 2. Example of SDTM metadata

There are 9 different sheets in SDTM metadata. Each sheet contains information corresponds to the elements in define.xml. All these elements are the child elements of MetaDataVersion element.

Part 1

Sheet	Study	Standard	Domain	Variable	WhereClauseRef
Element	GlobalVariables	def:Standards	ItemGroupDef	ItemDef	def:ValueListDef

Part 2

Sheet	SharedComments	ExternalDictionary	ComputationAlgorithms	SupplementalDocuments
Element	def:CommentDef	ExternalCodeList	MethodDef	def:AnnotatedCRF & def:SupplementalDoc

Table 1. Sheet corresponding Element

Please fill in the metadata, check carefully your metadata and translate required columns into Chinese.

Use Python to Generate Define.xml

Import SAS Datasets and XLSX Files

You can import SAS datasets using Python Pandas read_sas function and import Microsoft Excel Worksheet using read_excel function. Pandas is a good tool to use for data analysis. One of the data structures named DataFrame is like dataset in SAS and is commonly used for data analysis.

The two metadata files and SAS datasets should be imported in Python:

```
import pandas as pd

# SDTM

o_metadata = pd.read_excel('sdtm_METADATA.xlsx',sheet_name=None)

sheet_name = []

for k,v in o_metadata.items():

     sheet_name.append(k)

     v = v.rename(columns=lambda x: x.upper())

     locals()['v_'+k[0:3].upper()]=v
```

```
# CT
ct_metadata = pd.read_excel('ct_METADATA.xlsx', sheet_name='Sheet1')
ct_metadata = ct_metadata.rename(columns=lambda x: x.upper())
# Load SAS datasets
itemdef_list = []
for item in sas_file:
    domain = item.replace('.sas7bdat', '').upper()
    try:
         locals()['d_'+domain]= doc = pd.read_sas(item, encoding="gbk")
    except BaseException as e:
        print(domain)
        print(e)
```

Data Preparation and Manipulation

You can crosscheck your metadata in different sheets and get the information needed.

For example, the values for “METHOD” column in Variable sheet and in WhereClauseRef sheet should also be in “METHOD” column of ComputationAlgorithms sheet. Usually, the unmapped values will be deleted from DataFrame. The pd.merge function can merge two DataFrames together:

```
mt_all=list(mt_var|mt_val)
mt_in_frame = pd.DataFrame({'METHOD': mt_all, 'IncludeXPT': 'Y'})
o_metadata_MT = pd.merge(v_COM, mt_in_frame, on=['METHOD'], how='inner')
```

Also, the actual length for variables and values should be got. The length of variable should be displayed in define.xml. Use len function can get the maximum length of variable easily. The following example is about getting maximum length for value-level variables:

```
for ii in dataset_vl_part.index:
        vl_var = (dataset_vl_part.loc[ii, 'VARIABLE'])
        vl_wherevar = (dataset_vl_part.loc[ii, 'WHERE_VARIABLE'])
        vl_value = (dataset_vl_part.loc[ii, 'CHECK_VALUE'])
        doc_v = doc[doc[vl_wherevar] == vl_value]
        d_col = doc_v[vl_var]
        a = d_col.apply(length_col).tolist()
        if a:
# Get max length for value-level variables
            max_a = max(a)
        else:
            max_a = 100000
…
# Remove value-level metadata for uncollected data
o_metadata_VL = o_metadata_VL[o_metadata_VL['LENGTH_XPT'] != 100000]
```

The data may not be collected sometimes, so for supplemental datasets, some pre-defined qualifier variable will not

in datasets. The value-level definitions for these uncollected data should be remove from metadata.
After crosscheck and get variable lengths, the final DataFrames can be used for generating elements in define.xml.

o_metadata_CT	DataFrame	(83, 14)	Column names: CT_NAME, CT_LABEL, CT_TYPE, CT_PERM, CT_RANK, CT_DISPLAY ...
o_metadata_EX	DataFrame	(2, 5)	Column names: DICT_ID, EXTERNAL_DICTIONARY_NAME, VERSION, TYPE, CODELI ...
o_metadata_IG	DataFrame	(6, 16)	Column names: DOMAIN, IncludeXPT, PARENT, ORDER, PARENT_LABEL, LABEL, ...
o_metadata_IN	DataFrame	(1, 7)	Column names: STUDY_NAME, STUDY_DESCRIPTION, PROTOCOL_NAME, METADATA_O ...
o_metadata_IT	DataFrame	(120, 26)	Column names: DOMAIN, VARIABLE, LENGTH_XPT, ORDER_XPT, LABEL, ORDER, T ...
o_metadata_MT	DataFrame	(14, 8)	Column names: METHOD, MTHDNAME, MTHDTYPE, TEXT, FORMAL, LEAFID, PAGE_R ...
o_metadata_SC	DataFrame	(2, 8)	Column names: DOMAIN, VALUELEVEL, VARIABLE, WHERE_ID, TEXT, LEAFID, PG ...
o_metadata_SD	DataFrame	(2, 5)	Column names: STANDARD_NAME, STANDARD_VERSION, STANDARD_TYPE, PUBLISHI ...
o_metadata_SU	DataFrame	(3, 3)	Column names: LEAFID, XLK_RF, TITLE
o_metadata_VL	DataFrame	(20, 26)	Column names: DOMAIN, VARIABLE, LENGTH_XPT, WHERE_VARIABLE, CHECK_VALU ...

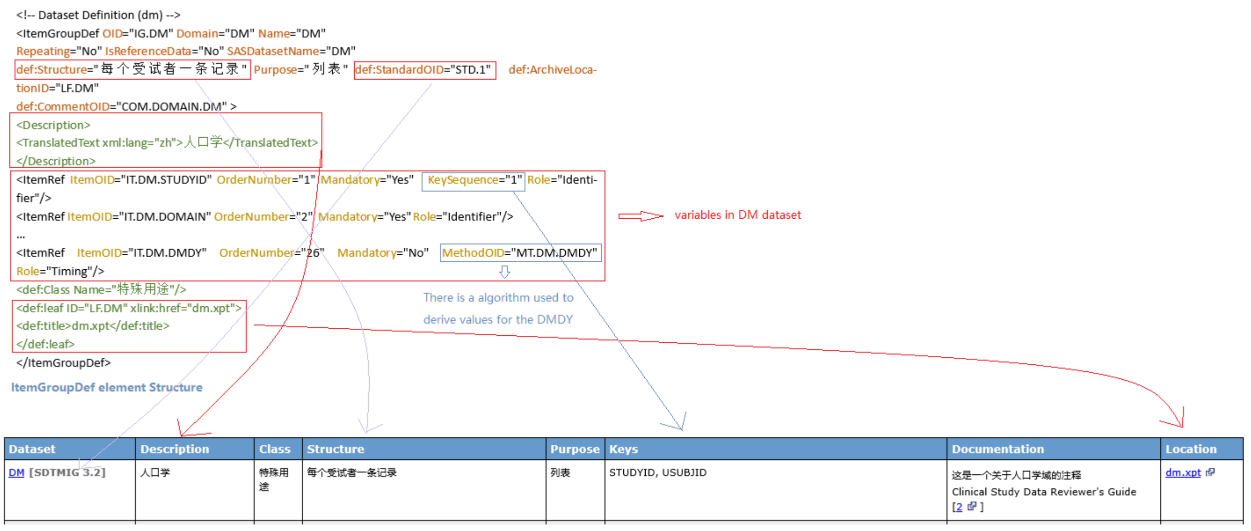
Display 3. Final DataFrames to generate Define elements

Create Elements

This step is the most boring and tedious step. You need to generate elements in order. The CDISC guideline specifies the order for the elements. Normally, the outermost element needs to be generated first including attribute and text. And then generate the child element. The first element of define.xml is ODM element. The ODM element has a child element named Study. The Study element contains all information about metadata for study. One of the child elements of Study element named MetaDataVersion containing all the definitions related to the domains. So, you can use the following order to create Define element:

1. header and start tag of ODM; 2. start tag of Study, GlobalVariables, StudyName, StudyDescription, and ProtocolName; 3. start tag of MetaDataVersion; 4. def:Standards and def:Standard; 5. def:AnnotatedCRF and def:SupplementalDoc; 6. def:ValueListDef; 7. def:WhereClauseDef; 8. ItemGroupDef; 9. ItemDef; 10. CodeList, EnumeratedItem ,CodeListItem and ExternalCodeList; 11. MethodDef; 12. def:CommentDef; 13. def:leaf; 14. end tag of MetaDataVersion, end tag of Study and end tag of ODM

Since each element has a standard structure. Generating elements is to concatenate strings in standard structure. Let's see an example of element ItemGroupDef.



```
<!-- Dataset Definition (dm) -->
<ItemGroupDef OID="IG.DM" Domain="DM" Name="DM"
Repeating="No" IsReferenceData="No" SASDatasetName="DM"
def:Structure="每个受试者一条记录" Purpose="列表" def:StandardOID="STD.1"   def:ArchiveLoca-
tionID="LF.DM"
def:CommentOID="COM.DOMAIN.DM" >
<Description>
<TranslatedText xml:lang="zh">人口学</TranslatedText>
</Description>
<ItemRef  ItemOID="IT.DM.STUDYID"  OrderNumber="1" Mandatory="Yes"   KeySequence="1" Role="Identi-
fier"/>
<ItemRef ItemOID="IT.DM.DOMAIN" OrderNumber="2" Mandatory="Yes" Role="Identifier"/>
...
<ItemRef   ItemOID="IT.DM.DMDY"   OrderNumber="26"   Mandatory="No"   MethodOID="MT.DM.DMDY"
Role="Timing"/>
<def:Class Name="特殊用途"/>
<def:leaf ID="LF.DM" xlink:href="dm.xpt">
<def:title>dm.xpt</def:title>
</def:leaf>
</ItemGroupDef>
```

ItemGroupDef element Structure

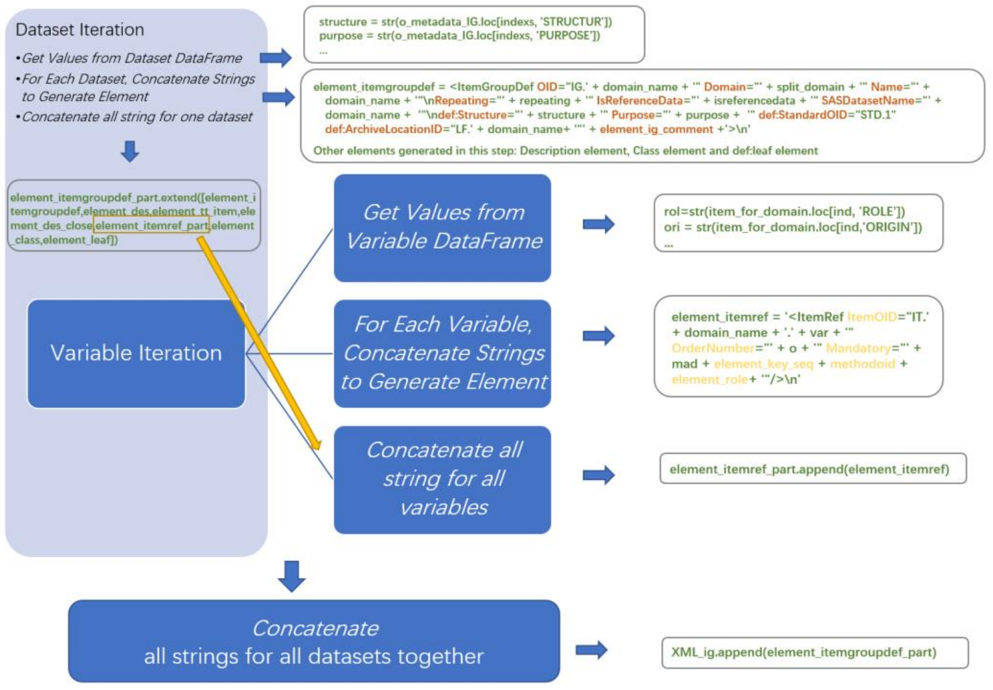
Dataset	Description	Class	Structure	Purpose	Keys	Documentation	Location
DM [SDTMIG 3.2]	人口学	特殊用途	每个受试者一条记录	列表	STUDYID, USUBJID	这是一个关于人口学域的注释 Clinical Study Data Reviewer's Guide [2]	dm.xpt

ItemGroupDef element displaying in browser

Figure 3. Example of ItemGroupDef for DM Domain

The metadata for DM Domain is split to several parts and be put in attribute values or child elements of ItemGroupDef. The structure, purpose, related standard and related comment is in values of corresponding attributes. And there is a child element named ItemRef. This element represents the variables including in DM Domain. The attributes of ItemRef describe the order, mandatory, related algorithm, the role for variables and if the variable is a key variable. Other variable definitions will be associated with another element named ItemDef. ItemDef element will be generated after ItemGroupDef element is generated. So, one ItemGroupDef element related to one dataset definition and several variable definitions. In order to generate all ItemGroupDef elements, two For loops need to be used.

One loop is about the dataset iteration to get the label, purpose, structure and other information for each dataset and generate string. Another loop is about the variable iteration which is a nested loop in dataset iteration to get the information about variables and generate string.


Dataset Iteration
•Get Values from Dataset DataFrame
•For Each Dataset, Concatenate Strings to Generate Element
•Concatenate all string for one dataset
Variable Iteration
Get Values from Variable DataFrame
For Each Variable, Concatenate Strings to Generate Element
Concatenate all string for all variables
Concatenate all strings for all datasets together

```
structure = str(o_metadata_IG.loc[indexs, 'STRUCTUR'])
purpose = str(o_metadata_IG.loc[indexs, 'PURPOSE'])
...
```

```
element_itemgroupdef = <ItemGroupDef OID="IG.' + domain_name + '" Domain="' + split_domain + '" Name="' +
  domain_name + '"\nRepeating="' + repeating + '" IsReferenceData="' + isreferencedata + '" SASDatasetName="' +
  domain_name +  '"\ndef:Structure="' + structure + '" Purpose="' + purpose +  '" def:StandardOID="STD.1"
  def:ArchiveLocationID="LF.' + domain_name+ '"' + element_ig_comment +'>\n'
```

Other elements generated in this step: Description element, Class element and def:leaf element

```
element_itemgroupdef_part.extend([element_itemgroupdef,element_des,element_tt_item,element_des_close,element_itemref_part,element_class,element_leaf])
```

```
rol=str(item_for_domain.loc[ind, 'ROLE'])
ori = str(item_for_domain.loc[ind,'ORIGIN'])
...
```

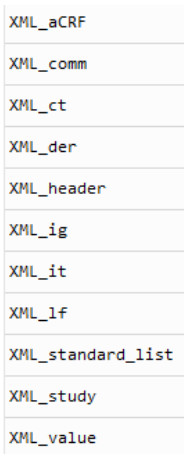
```
element_itemref = '<ItemRef ItemOID="IT.'
+ domain_name + '.' + var + '"
OrderNumber="' + o + '" Mandatory="' +
mad + element_key_seq + methodoid +
element_role+ '"/>\n'
```

```
element_itemref_part.append(element_itemref)
```

```
XML_ig.append(element_itemgroupdef_part)
```

Figure 4. Steps for generating ItemGroupDef Elements

Other elements are generated by For loop in the similar way. Finally, the following strings are generated. No complicated logic is needed to generate strings for elements, just be careful and more careful.



```
XML_aCRF
XML_comm
XML_ct
XML_der
XML_header
XML_ig
XML_it
XML_lf
XML_standard_list
XML_study
XML_value
```

Display 4. Elements in string

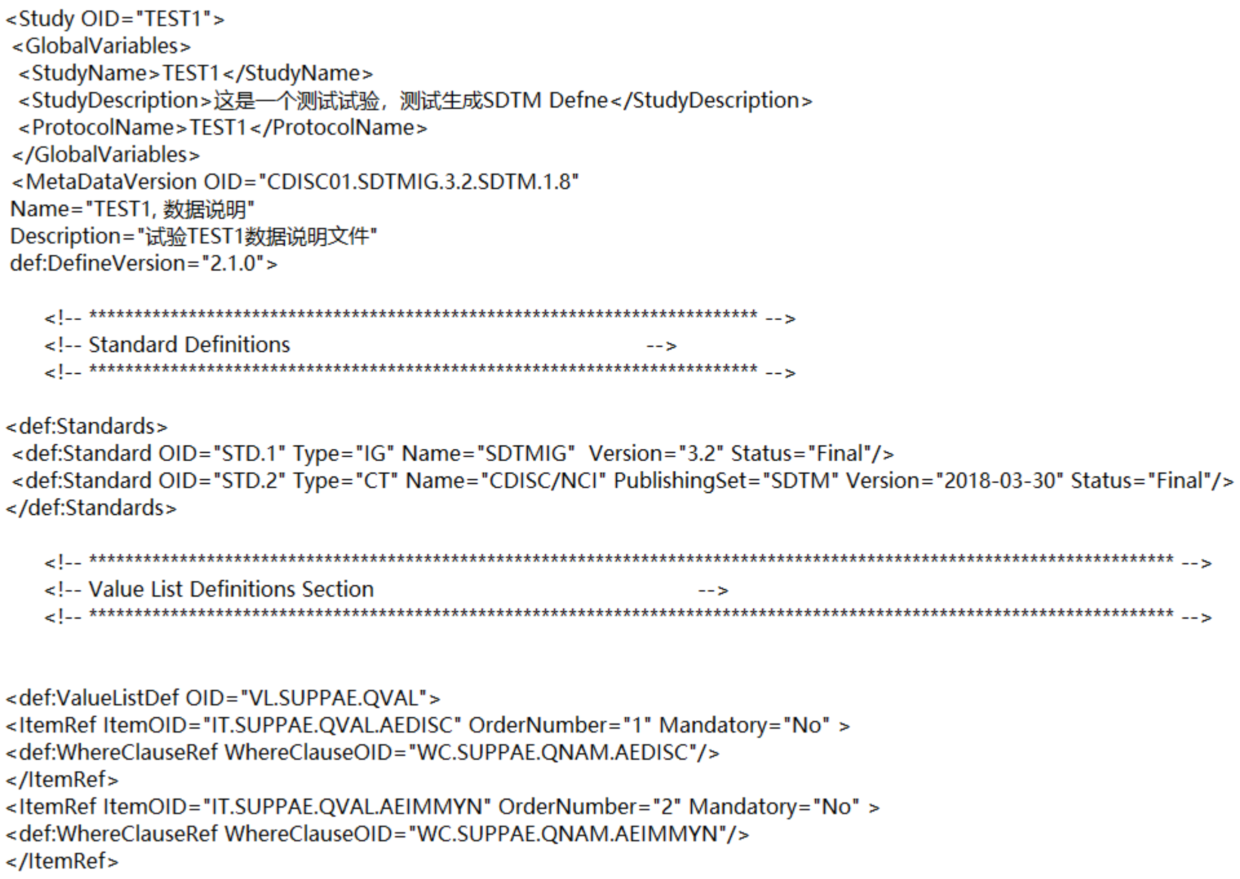
If you need, you can find code snippets to generate ItemGroupDef element in Appendix.

Export XML File

Use open function to create a new file, write strings in file and save.

```
file = open('define_new.xml', 'w', encoding='utf-8')
file.write(XML_header)
file.write(XML_study)
…
file.write(XML_lf)
file.write(element_final)
file.close()
```

The file will be saved in working directory. You can open the file with Notepad, and you will see the following display. The define.xml consists of elements which are mentioned before.



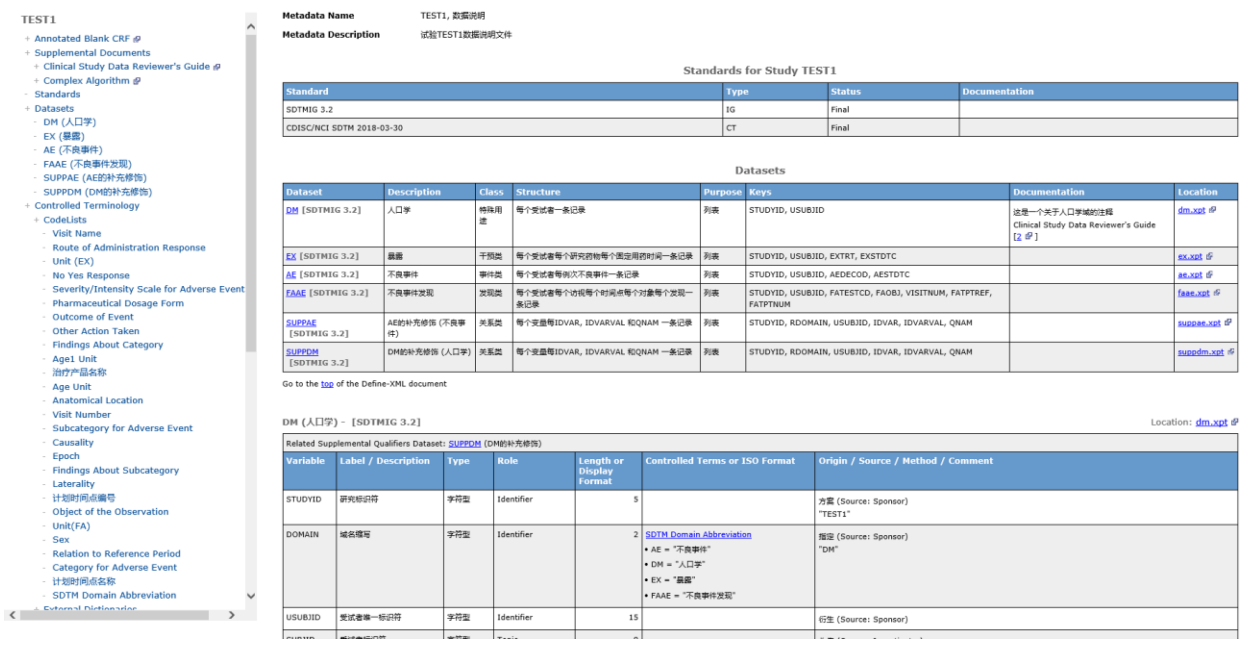
```
<Study OID="TEST1">
 <GlobalVariables>
  <StudyName>TEST1</StudyName>
  <StudyDescription>这是一个测试试验，测试生成SDTM Defne</StudyDescription>
  <ProtocolName>TEST1</ProtocolName>
 </GlobalVariables>
 <MetaDataVersion OID="CDISC01.SDTMIG.3.2.SDTM.1.8"
 Name="TEST1, 数据说明"
 Description="试验TEST1数据说明文件"
 def:DefineVersion="2.1.0">

     <!-- ************************************************************************** -->
     <!-- Standard Definitions                                                 -->
     <!-- ************************************************************************** -->

<def:Standards>
 <def:Standard OID="STD.1" Type="IG" Name="SDTMIG"  Version="3.2" Status="Final"/>
 <def:Standard OID="STD.2" Type="CT" Name="CDISC/NCI" PublishingSet="SDTM" Version="2018-03-30" Status="Final"/>
</def:Standards>

     <!-- ************************************************************************************************************************ -->
     <!-- Value List Definitions Section                                                   -->
     <!-- ************************************************************************************************************************ -->


<def:ValueListDef OID="VL.SUPPAE.QVAL">
<ItemRef ItemOID="IT.SUPPAE.QVAL.AEDISC" OrderNumber="1" Mandatory="No" >
<def:WhereClauseRef WhereClauseOID="WC.SUPPAE.QNAM.AEDISC"/>
</ItemRef>
<ItemRef ItemOID="IT.SUPPAE.QVAL.AEIMMYN" OrderNumber="2" Mandatory="No" >
<def:WhereClauseRef WhereClauseOID="WC.SUPPAE.QNAM.AEIMMYN"/>
</ItemRef>
```

Display 5. Define.xml fragment opening in Notepad

Follow-up

After running the Python script, the define.xml is generated in the folder. With the XSL file, it can be viewed better in web browser.

Display 6 is screen capture of define.xml fragment opening in the Internal Explorer (IE).



TEST1

- Annotated Blank CRF
- Supplemental Documents
  - Clinical Study Data Reviewer's Guide
  - Complex Algorithm
- Standards
- Datasets
  - DM (人口学)
  - EX (暴露)
  - AE (不良事件)
  - FAAE (不良事件发现)
  - SUPPAE (AE的补充修饰)
  - SUPPDM (DM的补充修饰)
- Controlled Terminology
  - CodeLists
    - Visit Name
    - Route of Administration Response
    - Unit (EX)
    - No Yes Response
    - Severity/Intensity Scale for Adverse Event
    - Pharmaceutical Dosage Form
    - Outcome of Event
    - Other Action Taken
    - Findings About Category
    - Age1 Unit
    - 治疗产品名称
    - Age Unit
    - Anatomical Location
    - Visit Number
    - Subcategory for Adverse Event
    - Causality
    - Epoch
    - Findings About Subcategory
    - Laterality
    - 计划时间点编号
    - Object of the Observation
    - Unit(FA)
    - Sex
    - Relation to Reference Period
    - Category for Adverse Event
    - 计划时间点名称
    - SDTM Domain Abbreviation
  - External Dictionaries

Metadata Name TEST1, 数据说明

Metadata Description 试验TEST1数据说明文件

Standards for Study TEST1

Standard	Type	Status	Documentation
SDTMIG 3.2	IG	Final	
CDISC/NCI SDTM 2018-03-30	CT	Final	

Datasets

Dataset	Description	Class	Structure	Purpose	Keys	Documentation	Location
DM [SDTMIG 3.2]	人口学	特殊用途	每个受试者一条记录	列表	STUDYID, USUBJID	这是一个关于人口学域的注释 Clinical Study Data Reviewer's Guide [2]	dm.xpt
EX [SDTMIG 3.2]	暴露	干预类	每个受试者每个研究药物每个固定用药时间一条记录	列表	STUDYID, USUBJID, EXTRT, EXSTDTC		ex.xpt
AE [SDTMIG 3.2]	不良事件	事件类	每个受试者每例次不良事件一条记录	列表	STUDYID, USUBJID, AEDECOD, AESTDTC		ae.xpt
FAAE [SDTMIG 3.2]	不良事件发现	发现类	每个受试者每个访视每个时间点每个对象每个发现一条记录	列表	STUDYID, USUBJID, FATESTCD, FAOBJ, VISITNUM, FATPTREF, FATPTNUM		faae.xpt
SUPPAE [SDTMIG 3.2]	AE的补充修饰 (不良事件)	关系类	每个变量每IDVAR, IDVARVAL 和QNAM 一条记录	列表	STUDYID, RDOMAIN, USUBJID, IDVAR, IDVARVAL, QNAM		suppae.xpt
SUPPDM [SDTMIG 3.2]	DM的补充修饰 (人口学)	关系类	每个变量每IDVAR, IDVARVAL 和QNAM 一条记录	列表	STUDYID, RDOMAIN, USUBJID, IDVAR, IDVARVAL, QNAM		suppdm.xpt

Go to the top of the Define-XML document

DM (人口学) - [SDTMIG 3.2] Location: dm.xpt

Related Supplemental Qualifiers Dataset: SUPPDM (DM的补充修饰)

Variable	Label / Description	Type	Role	Length or Display Format	Controlled Terms or ISO Format	Origin / Source / Method / Comment
STUDYID	研究标识符	字符型	Identifier	5		方案 (Source: Sponsor) "TEST1"
DOMAIN	域名缩写	字符型	Identifier	2	SDTM Domain Abbreviation • AE = "不良事件" • DM = "人口学" • EX = "暴露" • FAAE = "不良事件发现"	指定 (Source: Sponsor) "DM"
USUBJID	受试者唯一标识符	字符型	Identifier	15		衍生 (Source: Sponsor)

Display 6. Define.xml fragment displaying in IE

You can check if the hyperlinks in define.xml can be opened and run validation software. If you need the test datasets, test SDTM metadata and the define.xml generated. You can contact me by email, since the related files cannot be attached in this paper.

If the code is robust enough, and can be used for many studies, you can create an executable from Python script using Pyinstaller:

```
pip install pyinstaller

pyinstaller -F define.py
```

Also, you can design an interface using Python to develop a good application to generate define.xml.

CONCLUSION

To submit clinical trials data for drug applications, Data Definition File is required. It may cost a lot of time generating it, so in order to improve efficiency, developing a standard process to generate define.xml is necessary. This paper proposes an approach to generate define.xml using Python. Python is easy to learn and is friendly to the newbie. The Python script is easy to be packaged to an executable file, and people can run the executable even if they know nothing about Python. Therefore, using Python to generate define.xml is encourage to you if you want to develop an application with a friendly user interface to generate define.xml.

REFERENCES

CDE. “Guideline on the Submission of Clinical Trial Data (for trial implementation)”. July 20, 2020. Available at http://www.cde.org.cn/news.do?method=largeInfo&id=7a43c3abfde95950

CDISC Define-XML Team. “CDISC Define-XML Specification Version 2.1 (Final)”. July 6, 2021. Available at https://www.cdisc.org/standards/foundational/define-xml/define-xml-v2-1

RECOMMENDED READING

- *CDISC Define-XML Specification Version 2.1 https://www.cdisc.org/standards/foundational/define-xml/define-xml-v2-1*

CONTACT INFORMATION

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

Siyi Li
Sanofi
+862862228126
siyi2.li@sanofi.com

APPENDIX

```
# Loop for dataset
XML_ig=[]
# o_metadata_IG: DataFrame for dataset definitions
for indexs in o_metadata_IG.index:
# Get needed information from DataFrame
    element_itemgroupdef_part = []
    domain_name = str(o_metadata_IG.loc[indexs, 'PARENT'])
    split_domain = str(o_metadata_IG.loc[indexs, 'DOMAIN']).replace('SUPP','')
    repeating = str(o_metadata_IG.loc[indexs, 'REPEAT']).capitalize()
    isreferencedata = str(o_metadata_IG.loc[indexs, 'ISRFDATA']).capitalize()
    structure = str(o_metadata_IG.loc[indexs, 'STRUCTUR'])
    purpose = str(o_metadata_IG.loc[indexs, 'PURPOSE'])
    desc=str(o_metadata_IG.loc[indexs, 'PARENT_LABEL'])
    comm=str(o_metadata_IG.loc[indexs, 'COMMENT']).replace('nan', '')
    clas=str(o_metadata_IG.loc[indexs, 'CLASS'])
    if comm != '':
      element_ig_comment = '\ndef:CommentOID="COM.DOMAIN.' +domain_name+ '" '
    else:
      element_ig_comment = ''
# For each dataset, concatenate strings to generate element
    element_itemgroupdef = '<ItemGroupDef OID="IG.' + domain_name + '" Domain="' + split_domain + '"
Name="' + domain_name + '"\nRepeating="' + repeating + '" IsReferenceData="' + isreferencedata + '"
SASDatasetName="' + domain_name +  '"\ndef:Structure="' + structure + '" Purpose="' + purpose +  '"
def:StandardOID="STD.1"  def:ArchiveLocationID="LF.' + domain_name+ '"' + element_ig_comment +'>\n'
# Description element, Class element and def:leaf element generating in the same way
    element_des = '<Description>' + '\n'
    element_des_close = '</Description>' + '\n'
    element_tt = '<TranslatedText xml:lang="zh">'
    element_tt_close = '</TranslatedText>' + '\n'
    element_tt_item = element_tt + desc + element_tt_close
    element_class = '<def:Class Name="'+ clas +'"/>\n'
    element_leaf = '<def:leaf ID="LF.' + domain_name + '" xlink:href="' + domain_name.lower() + '.xpt">' +
'\n' + '<def:title>' + domain_name.lower() + '.xpt</def:title>\n</def:leaf>\n</ItemGroupDef>\n'
# Get variables corresponding to dataset in variable DataFrame
# o_metadata_IT: DataFrame for variable definitions
    item_for_domain = o_metadata_IT[o_metadata_IT.DOMAIN == domain_name]
# Loop for variable
    element_itemref_part = []
    for ind in item_for_domain.index:
# Get needed information from DataFrame
        key_seq = str(item_for_domain.loc[ind, 'KEYSEQ']).replace('.0', '')
        if key_seq == 'nan':
            element_key_seq = ''
        else:
            element_key_seq = '" KeySequence="' + key_seq + ''
        ori = str(item_for_domain.loc[ind,'ORIGIN'])
        mad = str(item_for_domain.loc[ind,'MANDAT']).capitalize()
        var=str(item_for_domain.loc[ind, 'VARIABLE'])
        rol=str(item_for_domain.loc[ind, 'ROLE'])
        o=str(item_for_domain.loc[ind, 'ORDER'])
        der = str(item_for_domain.loc[ind,'METHOD']).replace('nan','')
        s_der = str(item_for_domain.loc[ind,'METHOD_SHARE']).replace('nan','').upper()
        if der or s_der:
            methodoid = '" MethodOID="MT.' + s_der
            if der:
                  methodoid = '" MethodOID="MT.' +domain_name+'.'+ var
        else:
            methodoid = ''
# For each variable iteration, concatenate strings to generate element
        element_role = '" Role="'+rol
        element_itemref = '<ItemRef ItemOID="IT.' + domain_name + '.' + var + '" OrderNumber="' + o + '"
Mandatory="' + mad + element_key_seq + methodoid + element_role+ '"/>\n'
# For each variable iteration, save items to a list and transfer list to a string
        element_itemref_part.append(element_itemref)
        element_itemref_part = ''.join(element_itemref_part)
# Concatenate all strings for one dataset
    element_itemgroupdef_part.extend([element_itemgroupdef,element_des,element_tt_item,element_des_close,el
ement_itemref_part,element_class,element_leaf])
    element_itemgroupdef_part = ''.join(element_itemgroupdef_part)
# Concatenate all strings for all datasets
    XML_ig.append(element_itemgroupdef_part)
XML_ig = ''.join(XML_ig)
```