PharmaSUG China 2023 - Paper PO-131

Generate perfect define xml v2.1 with Analysis Results Metadata using python

Qiuping Fu, INNOVENT

ABSTRACT

The Define-XML document represents the metadata of clinical trial data (SDTM and ADaM), which includes Dataset-Level, Variable-Level, Value-Level Metadata, Code-Lists, Derivations, Comments, Supplemental Documents, and Analysis Results Metadata (ARM). According to ADaM v2.1, ARM is an optional component of ADaM metadata that facilitates the review process by providing traceability from Analysis result display to the source data and other documents, including but not limited to the Statistical Analysis Plan (SAP). In a regular submission, it is a best practice to include the ARM in a Define XML document. The Pharmaceuticals and Medical Devices Agency (PMDA) requires ARM in the submission package.

This article introduces a tool named I-ESUB, which was independently developed by the Biostatistics Department of Innovent using Python. I-ESUB can generate Define XML v2.1 as well as the ARM automatically, significantly reducing the working hours on document preparation and improving the submission package quality.

INTRODUCTION

Creating a Define XML file is a challenging and time-consuming task. Before creating a Define-XML, one needs to have a fundamental understanding of XML language and the structure of Define XML.

An XML file is an extensible markup language file , and it is used to structure data for storage and transport. The XML elements are the basic building block of the XML document. The first element of XML document is called root element. An XML element is everything from (including) the element's start tag to (including) the element's end tag, it can include other elements, which are called child elements, and each XML element can have one or more attributes.

The following example illustrates the basic syntax of an element in an XML document:

```
<element_name attribute1 attribute2 ... > Contents... </element_name>
```

Since its inception in 2005 with version 1.0, the Define-XML standard has evolved significantly, and version 2.1 has become an industry standard now. On January 30, 2015, CDISC released ARM v1.0, an extension to the Define-XML model for the purpose of submissions to regulatory agencies such as the United States Food and Drug Administration (FDA), as well as for the exchange of analysis datasets and key results between other parties.

Python is a high-level, dynamically typed, object-oriented, and interpreted programming language that provides many feasible solutions for creating XML files quickly.

This paper provide a comprehensive overview of the basic structure of the define XML document from a source code perspective, with a particular focus on the components of ARM. Additionally, it briefly explores a tool developed using the Python package minidom, which can rapidly generate define XML documents.

DEFINE-XML V2.1 STRUCTURE

A define file in 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 for some special elements.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="define2-1.xsl"?>
<ODM xmlns="http://www.cdisc.org/ns/odm/v1.3"
    xmlns:def="http://www.cdisc.org/ns/def/v2.1"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:arm="http://www.cdisc.org/ns/arm/v1.0" FileOID="CDISC-Sample" ODMVersion="1.3.2"
    FileType="Snapshot" CreationDateTime="2018-11-15T11:01:00" def:Context="Submission">
    <Study>
        <GlobalVariables>
            <StudyName/>
            <StudyDescription/>
            <ProtocolName/>
        </GlobalVariables>
        <MetaDataVersion>
            <def:Standards/>     <!--Standards definitions-->
            <def:AnnotatedCRF/>    <!--annotated case report forms-->
            <def:SupplementalDoc/>   <!--Supplemental Data Definitions-->
            <def:ValueListDef/>    <!--Value definitions-->
            <def:WhereClauseDef/>   <!-- Where Clause definitions-->
            <ItemGroupDef/>   <!--Dataset definitions -->
            <ItemDef/>   <!--Variable definitions-->
            <CodeList/>   <!--Controlled Terminology definitions-->
            <MethodDef/>   <!--Computational Method definitions -->
            <def:CommentDef/>   <!--Comment definitions-->
            <def:leaf/>   <!--External file Xlinks -->
            <arm:AnalysisResultDisplays>   <!--Analysis Results Metadata -->…
            </arm:AnalysisResultDisplays>
        </MetaDataVersion>
    </Study>
</ODM>
```

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

The XML header is a crucial component of any XML document file, including Define-XML files. It ensures that the file is properly recognized and decoded by the application, and helps to prevent any potential errors or issues during data processing and analysis, therefore the first line of a Define-XML document must be an XML header. Figure 1 shows an example of a Define-XML document using "UTF-8" character encoding.

An XSL stylesheet allows the define.xml file to be easily viewed in a web browser. If a stylesheet reference is provided then a browser can open the define.xml file and display it according to the stylesheet. This example references the define2-1 stylesheet in the red box that is located in the same location as the define.xml file.

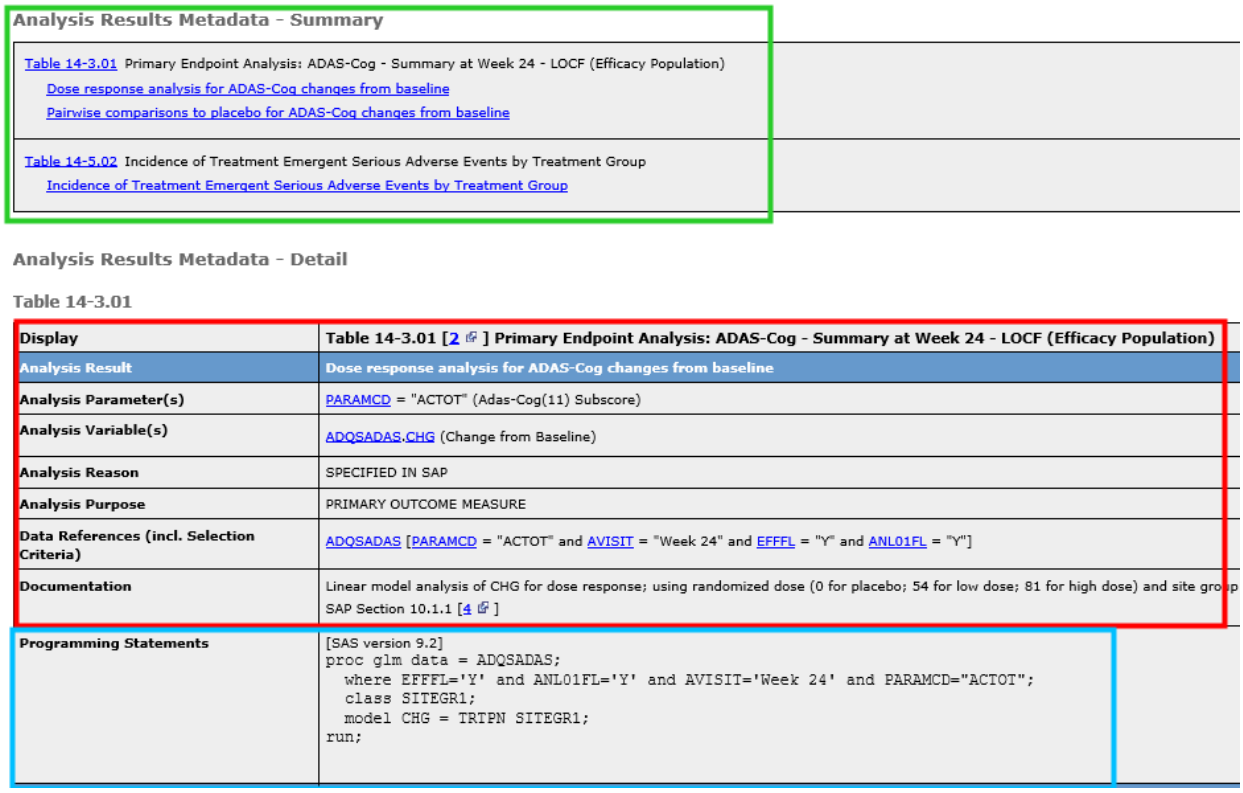
The MetaDataVersion element contains all the definitions related to the domains specified in the Define-XML. According to the figure, Blue text in Angle brackets represents the name of each element, and the grey italic text indicates information storing in each element. The “def:Standards” element is a new addition to version 2.1, which makes the use of CDISC standards and controlled terminology more flexible and efficient. Multiple CDISC standards and controlled teminology can now be referenced in a single define.xml document, eliminating the need for multiple documents. This new feature is particularly helpful in addressing data standardization issues in complex projects, such as those spanning multiple research centers or utilizing different data sources, for more information refer to <CDISC Define-XML Specification version 2.1> that available at “https://www.cdisc.org/standards/data-exchange/define-xml” .

While numerous elements find common application in SDTM/ADaM Define, there exist certain distinctive elements, for instance, "def:AnnotatedCRF," which exclusively pertains to SDTM. This element facilitates the scrutiny and validation of source data to ascertain the precision and uniformity thereof. Conversely, the element "arm:AnalysisResultDisplays" is confined to utilization in ADaM.

ANALYSIS RESULTS METADATA (ARM)

The analysis results are generated through analyzing trial data, including statistical descriptions and inference of data, such as p-values or estimates of treatment effects. ARM provide a link between analysis results and the data used to generate them, ensuring that the reviewers can find important information such as the reasons for conducting the analysis, the data sets used, and the selection criteria through the links. This allows for a more comprehensive understanding of the data sources and processing procedures, ensuring the accuracy and reliability of data analysis.

Figure 2 shows Define-XML Analysis Results Metadata, which from CDISC Analysis Results Metadata (ARM) for Define-XML v2.1 package.



Analysis Results Metadata - Summary

Table 14-3.01 Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population) Dose response analysis for ADAS-Cog changes from baseline Pairwise comparisons to placebo for ADAS-Cog changes from baseline
Table 14-5.02 Incidence of Treatment Emergent Serious Adverse Events by Treatment Group Incidence of Treatment Emergent Serious Adverse Events by Treatment Group

Analysis Results Metadata - Detail

Table 14-3.01

Display	Table 14-3.01 [2] Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population)
Analysis Result	**Dose response analysis for ADAS-Cog changes from baseline**
Analysis Parameter(s)	PARAMCD = "ACTOT" (Adas-Cog(11) Subscore)
Analysis Variable(s)	ADQSADAS.CHG (Change from Baseline)
Analysis Reason	SPECIFIED IN SAP
Analysis Purpose	PRIMARY OUTCOME MEASURE
Data References (incl. Selection Criteria)	ADQSADAS [PARAMCD = "ACTOT" and AVISIT = "Week 24" and EFFFL = "Y" and ANL01FL = "Y"]
Documentation	Linear model analysis of CHG for dose response; using randomized dose (0 for placebo; 54 for low dose; 81 for high dose) and site gro SAP Section 10.1.1 [4]

Programming Statements	[SAS version 9.2]
	proc glm data = ADQSADAS;
	where EFFFL='Y' and ANL01FL='Y' and AVISIT='Week 24' and PARAMCD="ACTOT";
	class SITEGR1;
	model CHG = TRTPN SITEGR1;
	run;

Figure 2 Define-XML 2.1 Display of Analysis Results Metadata

The ”arm:AnalysisResultDisplays” element contains all the elements of ARM. Figure 3 is a sample source code. The key components in ADaM Analysis Results Metadata are:

- Analysis Display metadata definitions
  - Documentation
  - Analysis Result metadata definitions
    - Analysis parameter(s)
    - Analysis dataset(s)
      - Analysis variable(s)
      - Selection criteria

- Programming Statements Definitions

```
<arm:AnalysisResultDisplays>   <!--Analysis Results Metadata -->
    <arm:ResultDisplay>
        <Description><TranslatedText></TranslatedText></Description>
        <def:DocumentRef></def:DocumentRef>
        <arm:AnalysisResult>
            <Description><TranslatedText></TranslatedText></Description>
            <arm:AnalysisDatasets>
                <arm:AnalysisDataset>
                    <def:WhereClauseRef></def:WhereClauseRef>
                    <arm:AnalysisVariable></arm:AnalysisVariable>
                </arm:AnalysisDataset>
            </arm:AnalysisDatasets>
            <arm:Documentation>
                <Description><TranslatedText></TranslatedText></Description>
                <def:DocumentRef><def:PDFPageRef></def:PDFPageRef></def:DocumentRef>
            </arm:Documentation>
            <arm:ProgrammingCode>
                <arm:Code></arm:Code>
                <def:DocumentRef><def:PDFPageRef></def:PDFPageRef></def:DocumentRef>
            </arm:ProgrammingCode>
        </arm:AnalysisResult>
    </arm:ResultDisplay>
</arm:AnalysisResultDisplays>
```

Figure 3. Elements in ARM

The source code contained in the rectangular boxes of three colors in Figure 3 corresponds to the content in the rectangular boxes of the corresponding colors in Figure 2.

XLSX METADATA FOR ARM

The ARM metadata is an xlsx file containing 2 sheets, named AanalysisDisplays and AnalysisResults. These two sheets correspond to the “Analysis Results Metadata -summary” and “Analysis Results Metadata -detail” sections of Define-XML v2.1. These data cover various aspects of the analysis results, including descriptions of the analysis results, statistical methods used in the analysis, reasons for performing the analysis, datasets and variables used, selection criteria, and important information such as the code used for the analysis.

A	B	C	D
TableID	**Title**	**Document**	**Pages**
Table_14-3.01	Primary Endpoint Analysis: ADAS-Cog - Summary at Week 25 - LOCF (Efficacy Population)	CSR	2
Table_14-5.02	Incidence of Treatment Emergent Serious Adverse Events by Treatment Group	CSR	3

Figure 4. Analysis Displays template

According to the figure both “TableID” and “Title” are Required and can not be null.

Analysis Results Metadata - Summary

Table 14-3.01 Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population) **TableID + Title** Dose response analysis for ADAS-Cog changes from baseline Pairwise comparisons to placebo for ADAS-Cog changes from baseline
Table 14-5.02 Incidence of Treatment Emergent Serious Adverse Events by Treatment Group Incidence of Treatment Emergent Serious Adverse Events by Treatment Group

Analysis Results Metadata - Detail

Table 14-3.01

Document + Pages

Display	**Table 14-3.01 [2] Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population)**
Analysis Result	**Dose response analysis for ADAS-Cog changes from baseline**

Figure 5. Analysis Results Metadata – Summary

A	B	C	D	E	F	G	H	I
TableID	**AnalysisID**	**Dataset**	**Variables**	**Parameter**	**select_criteria**	**AnalysisReason**	**AnalysisPurpose**	**AnalysisDescription**
Table_14-3.01	R.1	ADQSADAS	CHG	PARAMCD	WC.Table_14-3.01.R.1.ADQSADAS	SPECIFIED IN SAP	PRIMARY OUTCOME MEASURE	Dose response analysis for ADAS-Cog changes from baseline
Table_14-3.01	R.2	ADQSADAS	CHG	PARAMCD	WC.Table_14-3.01.R.2.ADQSADAS	SPECIFIED IN SAP	PRIMARY OUTCOME MEASURE	Pairwise comparisons to placebo for ADAS-Cog changes from baseline
Table_14-5.02	R.1	ADAE	AEBODSYS, AEDECOD		WC.Table_14-5.02.R.1.ADSL	SPECIFIED IN SAP	PRIMARY OUTCOME MEASURE	

I	J	K	L	M	N
AnalysisDescription	**Documentation_Description**	**Document**	**Pages**	**ProgrammingCode**	**Programming_Document**

Dose response analysis for ADAS-Cog changes from baseline	Linear model analysis of CHG for dose response; using randomized dose (0 for placebo; 54 for low dose; 81 for high dose) and site group in model. Used PROC GLM in SAS to produce p-value (from Type III SS for treatment dose).	CSR	4	proc glm data = ADQSADAS;
				where EFFFL='Y' and ANL01FL='Y' and AVISIT='Week 24' and PARAMCD="ACTOT";
				class SITEGR1;
				model CHG = TRTPN SITEGR1;
				run;	

Pairwise comparisons to placebo for ADAS-Cog changes from baseline	ANCOVA analysis of CHG performed to provide pairwise comparisons among treatment groups and adjusted means; using randomized treatment as class variable and site group as class variable in model and the baseline value as a covariate.	CSR	4	proc glm data = ADQSADAS;
				where EFFFL='Y' and ANL01FL='Y' and AVISIT='Week 24' and PARAMCD="ACTOT";
				class TRTPN SITEGR1;
				model CHG = TRTPN SITEGR2 BASE;
				means TRTPN;
				lsmeans TRTPN / OM STDERR PDIFF CL;
				run;	

	Unique count of subjects that experienced an Adverse Event by Preferred Term, System Organ Class, and Treatment Group and percentages based on the number of subjects in the safety population within each treatment group. The total number of times an event occurred was recorded by Preferred Term, System Organ Class, and Treatment Group. Fisher's exact test was used for treatment comparison of event rates.	csr	5		at14-5-02.sas

Figure 6. Analysis Results template

- Variables: If multiple variables are used in the same analysis result, different variables can be separated by commas.
- Select criterial: Directed connection to the where clause definitions section, used to indicate the filtering conditions of the analysis results. Usually equals TableID + AnalysisID+ Dataset.
- Analysis Reason: The rationale for performing this analysis. It indicates when the analysis was planned. If a specific analysis was prespecified in both protocol and SAP, specify "SPECIFIED IN SAP" as the analysis reason.
- Analysis Purpose: The purpose of the analysis within the body of evidence (e.g., section in the clinical study report).

Analysis Results Metadata - Detail

Table 14-3.01 TableID

Display	Table 14-3.01 [2] Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population)
Analysis Result	Dose response analysis for ADAS-Cog changes from baseline AnalysisDescription
Analysis Parameter(s)	PARAMCD = "ACTOT" (Adas-Cog(11) Subscore) parametera
Analysis Variable(s)	ADQSADAS.CHG (Change from Baseline) Dataset+Variables
Analysis Reason	SPECIFIED IN SAP AnalysisReason
Analysis Purpose	PRIMARY OUTCOME MEASURE AnalysisPurpose
Data References (incl. Selection Criteria)	ADQSADAS [PARAMCD = "ACTOT" and AVISIT = "Week 24" and EFFFL = "Y" and ANL01FL = "Y"] select criteria
Documentation	Linear model analysis of CHG for dose response; using randomized dose (0 for placebo; 54 for low dose; 81 for high dose) and site grou SAP Section 10.1.1 [4] Document + Pages + Documentation Description
Programming Statements	[SAS version 9.2] proc glm data = ADQSADAS; where EFFFL='Y' and ANL01FL='Y' and AVISIT='Week 24' and PARAMCD="ACTOT"; class SITEGR1; model CHG = TRTPN SITEGR1; ProgrammingCode + Programming Document run;

Figure 7. Analysis Results Metadata -Detail

PACKAGES USED

After understanding the structure of define xml, we can obtain the required data by reading the metadata (in xlsx format). Then, we can perform simple operations on this data, such as merging or splitting, to meet our needs. Finally, we will fill the processed data into the define xml according to the structure and meaning of the XML source code. To achieve automation, we uses the pandas library to read xlsx metadata, the minidom library to generate XML files, and combines the tkinter library to create a user interface. By using these packages for data processing and filling, users can quickly and accurately generate files that conform to the defined xml structure.

USE PANDAS TO READ METADATA.

Pandas is a powerful data processing tool that can easily read and manipulate various data formats, including xlsx format. We use the pandas library to read metadata, which provides convenient methods to parse xlsx files and convert the data into the format of a dataframe for further processing.

```
import pandas as pd
df1=pd.read_excel(medatafile, 'Variables')     #read Variables sheet from metadata
df2=pd.read_excel(medatafile,'Datasets')       #read Datasets sheet from metadata
df= df1.merge(df2,on=['Dataset'],how='left')   #merge dataframe 1,2 together
df.to_excel(outfile,"Variables")               #Write to outfile
```

USE MINIDOM TO GENERATE XML FILES.

Python's minidom library is one of the standard libraries for handling XML. We use the minidom library to create and edit XML files. By performing simple manipulations on the metadata, such as combining or splitting data, we can fill the data into an XML file according to the defined structure of the source code of the XML, in order to generate a compliant define xml file.

```
from xml.dom import minidom
newDoc = minidom.Document()
rootNode = newDoc.createElement('element_name')
newDoc.appendChild(rootNode)
firstNode = newDoc.createElement('subelement_name')
firstNode.setAttribute("rowid", "1")
```

```
firstNode.setAttribute("tableid", "100")
rootNode.appendChild(firstNode)
with open(rootDir + "/test.xml", "w", encoding="utf-8") as fo:
    newDoc.writexml(fo, indent='', addindent='\t', newl='\n', ncoding="utf-8")
    fo.close()
```

USE TKINTER TO CREATE A USER INTERFACE.

To facilitate users in using our generation tool, we use the tkinter library to create a user interface. This interface allows users to select the xlsx metadata file to be read and provides some simple options to customize the generated define xml file. With just a few button clicks, users can quickly generate a file that complies with the defined xml structure, without the need for manual complex operations.

Figure 8. Tkinter GUI application for python script

```
import tkinter as tk
from tkinter import *

root=Tk()
root.title(" I-ESUB")
root.geometry('600x200')
root.configure(background="#8c52ff")
Label(root,text='Define2.1 Generator ',bg='white',font=('Times New Roman ','10','italic
bold'),fg="red").grid(row=0, column=0,columnspan=5,ipady=30,sticky=E+W)

variable = tk.StringVar()
variable.set("Language")
menu=tk.OptionMenu(root, variable, "English", "Chinese")
menu.grid(row=0,column=0,sticky=W+S)
menu.config(bg="red")

Label(root,text='Excel Metadata').grid(row=1,column=0,sticky=E)
Entry(root,bg='white',width=40,font=20).grid(row=1,column=1,columnspan=3) # 0 行 1 列，跨 4 列
Button(root,text="Browse...",bg="red",height=1,width=10,command=ask_file).grid(row=1,column=4 )
```

```
Button(root,text="Create",bg="red",height=1,width=10,command=define_gen).grid(row=2,column=4,st
icky=E+S)

root.rowconfigure(0, weight=1)
root.rowconfigure(1, weight=4)
root.rowconfigure(2, weight=2)

root.mainloop()
```

CONCLUSION

Our method can effectively automate the generation of define xml 2.1 files. By using pandas to read xlsx metadata, we can easily load the data into memory and perform any necessary processing. Using the minidom library to generate XML files makes it simple and intuitive to fill the processed data into the define XML. With the user interface created using the tkinter library, users can conveniently select inputs and customize generation options, thereby improving efficiency. However, our method has some limitations. Currently, it only supports xlsx format metadata files, and additional processing may be required for other file formats. Future research directions include supporting the reading and processing of more data formats, and further optimizing the user interface to provide more customization options. Additionally, we can explore integrating the automated generation of define XML method with other clinical trial data management tools to further improve data management efficiency and accuracy.

REFERENCES

Trevor Mankus, ” Upgrading from Define-XML 2.0 to 2.1” PharmaSUG 2022.

Available at https://www.lexjansen.com/pharmasug/2022/DS/PharmaSUG-2022-DS-064.pdf

RECOMMENDED READING

- Define-XML v2.1
- Analysis Results Metadata v1.0 for Define-XML v2

CONTACT INFORMATION

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

Qiuping Fu

1451837459@qq.com