Paper PP02

Defining Standards: Challenges and Solutions in the Development of Define-XML for SDTM and ADaM

Sofia Vale, OCS Life Sciences, ‘s-Hertogenbosch, Netherlands

ABSTRACT

Define-XML is an integral part of any submission package of clinical studies to regulatory agencies, providing the metadata for all datasets being submitted. Therefore, it is essential that it is accurate and complete.
However, its development usually requires extensive manual input, making it time-consuming and prone to errors. There are solutions available on the market, but they are often expensive or incomplete.

Thankfully, due to its repetitive structure, it is possible to automate the development of Define-XML. This way, much of the manual input can be automated, and conventions can be implemented consistently across several studies. Our solution integrates several tools and information readily available, such as the datasets or shells already developed and the standards present in the CDISC Library, in a simple and intuitive manner for the end user. This turns the development of Define-XML into a fast, more accurate process, and greatly reduces the need for manual input.

Keywords: CDISC, Define-XML, Pinnacle 21, CDISC Library, SDTM, ADaM, metadata

INTRODUCTION

Define-XML is a CDISC metadata standard used to describe the content of tabular dataset structures, such as SEND, SDTM or ADaM. It contains all the metadata information at study-level, dataset-level, variable-level, and value-level, from data types and origins to codelists and algorithms used. It is an essential and required part of any submission package to regulatory authorities, as it provides the reviewer with valuable information about the origin, structure and content of the data being submitted, therefore making the familiarisation with the study data a fast and easy process.

Due to its key role in data submission and review, it is vital that the metadata present in the Define-XML is accurate, consistent, and complete. This makes it a challenging document to create, as it must describe all data being submitted, which is usually comprised of a large number of datasets and variables, at several levels and in high detail. Furthermore, conventions might be needed to ensure consistency, particularly when more than one person is involved in the creation and maintenance of both the data and the metadata.

For the above reasons, developing such a complex document by hand is a tedious and extremely time-consuming process, prone to errors and inconsistencies. Fortunately, due to its well-defined structure, it is possible to automate the derivation of much of the metadata information present in a Define-XML, based on the data available and the metadata of the CDISC standards used. Examples of such tools already exist in the market; for instance, Pinnacle 21® has a solution for the creation of Define-XML in the free version of their product, Pinnacle 21 Community. This tool makes use of the datasets already generated for submission and creates an Excel template, already holding some derived metadata information, that can be manually completed by the user and then fed back to the software to generate a full XML file. Although this is a good and user-friendly solution, it also lacks many functionalities, meaning most of the metadata still needs to be filled by hand. The paid version, Pinnacle 21 Enterprise, features the extra functionalities needed, but it can be expensive for small companies or short projects.

In this paper, we will discuss the Define-XML development tool we have built to overcome many of these obstacles. Our solution expands on the one provided by Pinnacle 21 Community, automating the derivation of metadata whenever possible, such as variable origins and derivation methods, and integrating more functionalities in order to keep the manual work to a minimum and the flexibility to a maximum, without compromising user-friendliness or increasing development time.

A FIRST LOOK

When designing the tool, we defined the following goals:

- Reduce the manual input needed in the creation of a Define-XML to a minimum.
- Guarantee the reliability of the automated metadata information.
- Make the tool flexible and adaptable to several standards and user requirements.
- Keep the tool as user-friendly as any other tool available.

As input, the tool uses the available data in the form of SAS datasets and the metadata defined in the CDISC standards, as well as other optional parameters as defined by the user, such as CRF annotations. This is currently set up in a SAS Studio task, due to its user-friendly interface, as shown in Figure 1**Figure 1:** An example of the SAS Studio task that can be used to create an automated SDTM Define-XML. The task derives metadata based on the input given and outputs an Excel file with all the information gathered.. Behind this task is a SAS program that performs several derivations to obtain all the metadata information that is possible to automate from the input given. In the end, it outputs an Excel file that follows a template compatible with Pinnacle 21 Community, where the user can manually complete any metadata missing. Once the file is complete, it can be run through a second tool or the Pinnacle 21 Community tool to generate a complete XML file.

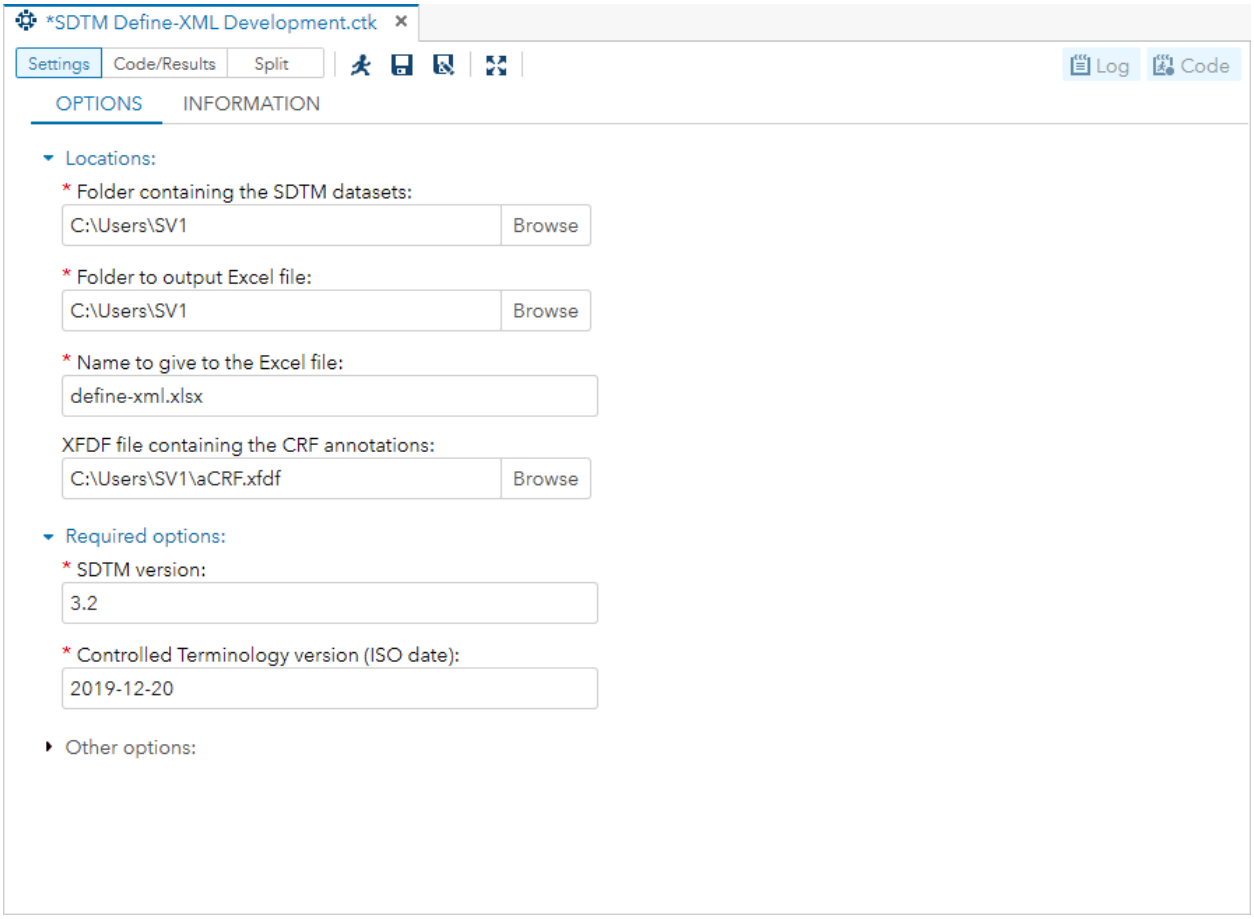


Figure 1: An example of the SAS Studio task that can be used to create an automated SDTM Define-XML. The task derives metadata based on the input given and outputs an Excel file with all the information gathered.

INPUT

As mentioned, our solution makes use of the available data and metadata to automate the information present in the Define-XML. This includes CDISC standards metadata, SAS datasets with the available study data, and other information that may be available, such as CRF annotations. This section discusses in more detail each of these inputs.

SAS DATASETS

This solution was first designed for the creation of Define-XML for legacy studies, after the submission datasets have been completed. This way, much of the metadata information can be automatically derived from the SAS datasets already created, thus not only reducing manual input but also the chances of discrepancies between the data and metadata. However, the tool can be easily adapted to work with dataset shells instead; that is, SAS datasets that contain little to no data, but already have the desired structure for the datasets to be submitted.

Independent on the data present in the SAS datasets, much of the metadata information is easy to gather or derive, such as:

- dataset and variable names and labels.
- key variables, based on the dataset sorting.
- default variable length.

- basic data types: text or integer, based on the variable type, or datetime, based on the variable name and/or label.

If data is available, this list can be expanded to:

- actual length, equal to the longest value available.
- data type and significant digits.
- codelist terms, for variables subjected to controlled terminology or for which a user-defined codelist has been identified.
- value-level metadata for variables such as --ORRES, QVAL and AVAL, including lengths and type for each group of values, as well as origin for each group of QVAL.
- determining predecessor variables for ADaM Define-XML, based on the data present in SDTM and ADaM datasets.

CDISC STANDARDS

One of the most important metadata resources is the CDISC standards metadata. This sort of metadata used to be available in a tabular structure in files that needed to be downloaded and then implemented in the program. This raised two main drawbacks: first, one file was needed for each standard and each version, which was especially problematic for controlled terminology, as there is at least one new version of controlled terminology every quarter; second, the structure of each file could change between versions, which forces the program to be adapted every time a new version is to be implemented.

Fortunately, a new source of CDISC standards metadata is now available: the CDISC Library. CDISC members can request an account, separate from other CDISC accounts, which will grant them access to the CDISC Library data standards browser (https://library.cdisc.org/browser/) and, more important, the CDISC Library API, which can be used to access all the information available in the CDISC Library from any program. To access the CDISC Library in SAS, the following code can be used (<username> and <password> must be replaced by the credentials to access the CDISC Library):

```
/*************************************************************
 Macro to access the CDISC Library API.
*************************************************************/
%macro cdisc_library(url=, outlib=);
  filename cdisclib temp;

  proc http url = "&url."
       auth_basic
       webusername = "<username>"
       webpassword = "<password>"
       out = cdisclib;
    headers "Accept" = "application/json";
  run;

  libname &outlib. json fileref = cdisclib;
%mend cdisc_library;

/*************************************************************
 Get the CDISC Library components and links to new hierarchies.
*************************************************************/
%cdisc_library(url = http://library.cdisc.org/api/mdr/products
              ,outlib = cdisc);
```

This way, it is possible to obtain the metadata information for any version of any standard present in the CDISC Library, always organised in the same structure and therefore avoiding adapting the program for each version.

Based on this metadata, there is a large amount of information that can be automatically derived for a Define-XML, such as:

- dataset class (for non-custom domains).
- mandatory variables.
- variables subjected to controlled terminology and some external dictionaries.
- codelist codes, term codes and decoded values from controlled terminology.
- standardised external dictionaries.

CRF ANNOTATIONS

If an annotated CRF (aCRF) is ready, it is possible to use the annotations to determine the variables originating from the aCRF. To do so, the annotations must first be extracted to an XFDF file, which can easily be done using Adobe® Acrobat® Reader (Comments > Export All To Data File…). The XFDF file will contain all the information about annotations in the aCRF. Each annotation will be wrapped in XML code similar to the following:

```
<freetext color="#FFFFAA" creationdate="D:00000000000000Z" flags="print"
date="D:20091109121708-05'00'" name="31ec1e26-bd9f-44f4-8ab0-fcae416fe7c4" page="5"
rect="34.595300,119.954000,101.629000,136.895000" subject="DS">
  <contents-richtext>
    <body xmlns="http://www.w3.org/1999/xhtml"
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/" xfa:APIVersion="Acrobat:7.0.8"
xfa:spec="2.0.2" style="font-size:12.0pt;font-weight:bold;font-style:italic;font-
family:Arial,sans-serif">
      <p dir="ltr">DSTERM</p>
    </body>
  </contents-richtext>
  <defaultappearance>0 0 0 rg /Arial,BoldItalic 12 Tf</defaultappearance>
  <defaultstyle>font: italic bold Arial,sans-serif 12.0pt; text-align:left;
color:#000000 </defaultstyle>
</freetext>
```

In bold and underlined is the information important in the development of Define-XML: the annotation content and the page where the annotation is present. This information can be extracted from the XFDF file by a SAS program to create an overview of all variables mentioned in the aCRF annotations and the pages where they appear.

When deriving the origin of variables based on CRF annotations, it is important to take into consideration any conventions used in the creation of such annotations. For instance, an annotation may read “IEORRES when IETESTCD = EXCL01”, meaning that the field in the aCRF is collecting the value of IEORRES for a given IETESTCD; in the Define-XML, this annotation is important for IEORRES, but not so much for IETESTCD. Therefore, when designing the SAS program, these examples and conventions must be considered.

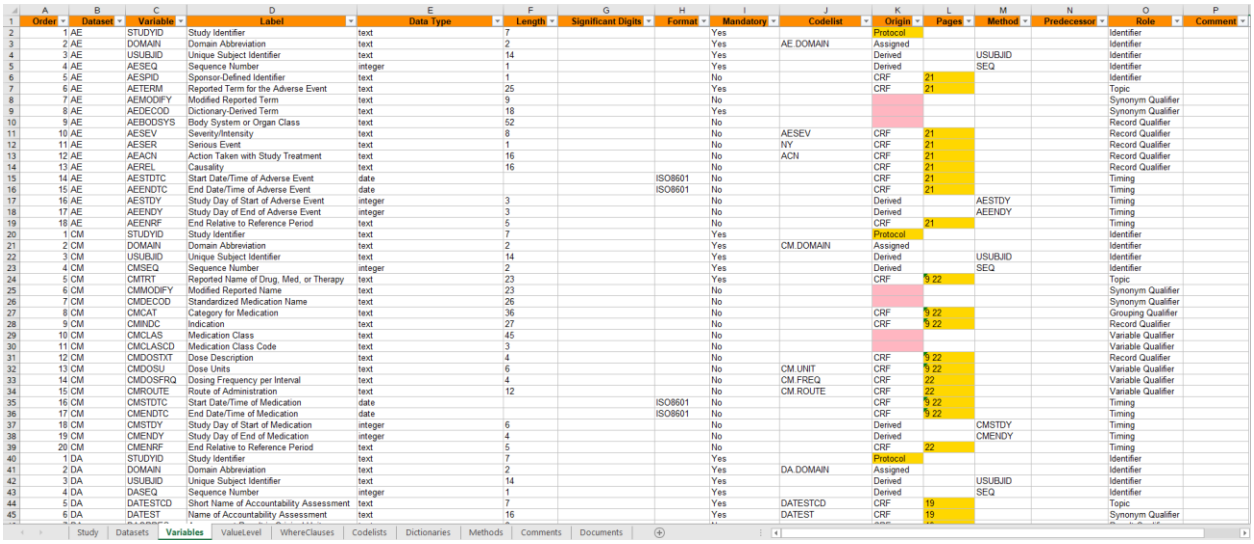
OUTPUT

Once all metadata information has been derived, the tool outputs it to an Excel file, in the same fashion as Pinnacle 21 Community (Figure 2):

- The metadata is grouped in different sheets, depending on its type: Study, Datasets, Variables, Value Level and Where Clauses, Codelists, Dictionaries, Methods, Comments, and Documents.
- Each sheet contains a table with all the related metadata, making it easy to complete and review before creating the final XML file.

To ease the review of the document by the user, the Define-XML Development tool can identify cells that require the user’s attention, as exemplified in Figure 2. If the tool was unable to automatically derive a metadata value that must be submitted, the cell will be highlighted in pink. Likewise, if the tool was able to derive a metadata value but it cannot be certain of its reliability, the cell will be highlighted in yellow; this is the case with, for example, aCRF pages, as a variable name can appear unpredictably in an annotation that is related to another variable instead (see the section “CRF Annotations” for an example).

Once the document is completely filled and reviewed, the final XML file can be created. This can be done using the Pinnacle 21 Community tool, since the structure is the same. Likewise, a second program can be written to convert the document into an XML document, taking advantage of its tabular structure. This would allow for even more flexibility, as well as complete independence from external tools.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Order	Dataset	Variable	Label	Data Type	Length	Significant Digits	Format	Mandatory	Codelist	Origin	Pages	Method	Predecessor	Role	Comment
2	1	AE	STUDYID	Study Identifier	text	7			Yes		Protocol				Identifier	
3	2	AE	DOMAIN	Domain Abbreviation	text	2			Yes	AE.DOMAIN	Assigned				Identifier	
4	3	AE	USUBJID	Unique Subject Identifier	text	14			Yes		Derived		USUBJID		Identifier	
5	4	AE	AESEQ	Sequence Number	integer	1			Yes		Derived		SEQ		Identifier	
6	5	AE	AESPID	Sponsor-Defined Identifier	text	1			No		CRF	21			Identifier	
7	6	AE	AETERM	Reported Term for the Adverse Event	text	25			Yes		CRF	21			Topic	
8	7	AE	AEMODIFY	Modified Reported Term	text	9			No						Synonym Qualifier	
9	8	AE	AEDECOD	Dictionary-Derived Term	text	18			Yes						Synonym Qualifier	
10	9	AE	AEBODSYS	Body System or Organ Class	text	52			No						Record Qualifier	
11	10	AE	AESEV	Severity/Intensity	text	8			No	AESEV	CRF	21			Record Qualifier	
12	11	AE	AESER	Serious Event	text	1			No	NY	CRF	21			Record Qualifier	
13	12	AE	AEACN	Action Taken with Study Treatment	text	16			No	ACN	CRF	21			Record Qualifier	
14	13	AE	AEREL	Causality	text	16			No		CRF	21			Record Qualifier	
15	14	AE	AESTDTC	Start Date/Time of Adverse Event	date			ISO8601	No		CRF	21			Timing	
16	15	AE	AEENDTC	End Date/Time of Adverse Event	date			ISO8601	No		CRF	21			Timing	
17	16	AE	AESTDY	Study Day of Start of Adverse Event	integer	3			No		Derived		AESTDY		Timing	
18	17	AE	AEENDY	Study Day of End of Adverse Event	integer	3			No		Derived		AEENDY		Timing	
19	18	AE	AEENRF	End Relative to Reference Period	text	5			No		CRF	21			Timing	
20	1	CM	STUDYID	Study Identifier	text	7			Yes		Protocol				Identifier	
21	2	CM	DOMAIN	Domain Abbreviation	text	2			Yes	CM.DOMAIN	Assigned				Identifier	
22	3	CM	USUBJID	Unique Subject Identifier	text	14			Yes		Derived		USUBJID		Identifier	
23	4	CM	CMSEQ	Sequence Number	integer	2			Yes		Derived		SEQ		Identifier	
24	5	CM	CMTRT	Reported Name of Drug, Med, or Therapy	text	23			Yes		CRF	22			Topic	
25	6	CM	CMMODIFY	Modified Reported Name	text	23			No						Synonym Qualifier	
26	7	CM	CMDECOD	Standardized Medication Name	text	26			No						Synonym Qualifier	
27	8	CM	CMCAT	Category for Medication	text	36			No		CRF	22			Grouping Qualifier	
28	9	CM	CMINDC	Indication	text	27			No		CRF	22			Record Qualifier	
29	10	CM	CMCLAS	Medication Class	text	45			No						Variable Qualifier	
30	11	CM	CMCLASCD	Medication Class Code	text	3			No						Variable Qualifier	
31	12	CM	CMDOSTXT	Dose Description	text	4			No		CRF	22			Record Qualifier	
32	13	CM	CMDOSU	Dose Units	text	6			No	CM.UNIT	CRF	22			Variable Qualifier	
33	14	CM	CMDOSFRQ	Dosing Frequency per Interval	text	4			No	CM.FREQ	CRF	22			Variable Qualifier	
34	15	CM	CMROUTE	Route of Administration	text	12			No	CM.ROUTE	CRF	22			Variable Qualifier	
35	16	CM	CMSTDTC	Start Date/Time of Medication	date			ISO8601	No		CRF	22			Timing	
36	17	CM	CMENDTC	End Date/Time of Medication	date			ISO8601	No		CRF	22			Timing	
37	18	CM	CMSTDY	Study Day of Start of Medication	integer	6			No		Derived		CMSTDY		Timing	
38	19	CM	CMENDY	Study Day of End of Medication	integer	4			No		Derived		CMENDY		Timing	
39	20	CM	CMENRF	End Relative to Reference Period	text	5			No		CRF	22			Timing	
40	1	DA	STUDYID	Study Identifier	text	7			Yes		Protocol				Identifier	
41	2	DA	DOMAIN	Domain Abbreviation	text	2			Yes	DA.DOMAIN	Assigned				Identifier	
42	3	DA	USUBJID	Unique Subject Identifier	text	14			Yes		Derived		USUBJID		Identifier	
43	4	DA	DASEQ	Sequence Number	integer	1			Yes		Derived		SEQ		Identifier	
44	5	DA	DATESTCD	Short Name of Accountability Assessment	text	7			Yes	DATESTCD	CRF	19			Topic	
45	6	DA	DATEST	Name of Accountability Assessment	text	16			Yes	DATEST	CRF	19			Synonym Qualifier	

Study | Datasets | Variables | ValueLevel | WhereClauses | Codelists | Dictionaries | Methods | Comments | Documents

Figure 2: Screenshot of the Excel file created with the Define-XML Development tool. Note that some cells have been highlighted in the sheet: pink cells must be filled manually, while yellow cells have been automatically filled but need to be manually reviewed.

WHAT CAN BE AUTOMATED?

There are several metadata fields that can be automatically filled based on the information already available in the input for this tool. This section aims to highlight several of the metadata fields present in the Define-XML and how we derive this information when possible.

This section is not a comprehensive list of all metadata fields, and will not contain information about certain types, such as containers. Likewise, some metadata fields, such as Study ID, can be automatically derived when creating the final XML file without having to be explicitly input in the intermediate Excel file; this sort of metadata information is not in scope for this paper, and will therefore not be included in this section. For more information on the Define-XML basics and structure, please refer to the recommended reading.

STUDY-LEVEL

Study-level metadata includes the identification of the study and general characteristics of the metadata. The values of most of these metadata fields can be provided in the Study sheet of the Excel file created, as shown in Figure 3.

	A	B	C	D
1	**Attribute**	**Value**		
2	StudyName	CDISC01		
3	StudyDescription	CDISC Test Study		
4	ProtocolName	CDISC01		
5	StandardName	SDTM-IG		
6	StandardVersion	3.2		
7	Language	en		
8				
9				
10				
11				
12				
13				

Study | Datasets | Variables | ValueLevel | WhereClaus ...

Figure 3. Example of the structure of the Study sheet and the metadata information it contains.

The following metadata fields describe the study:

- StudyName: this field identifies the study and may have the same value as the Protocol name; for our internal purposes, this is set to be equal to STUDYID.
- StudyDescription: this field is the study title from the protocol, which can be obtained from TS if available.

- ProtocolName: this field is the identifier of the study as specified in the protocol; it should be reviewed if automatically set to STUDYID, for example, to make sure that it matches the protocol.

Additionally, there are metadata fields to characterise the metadata, such as DefineVersion, StandardName and StandardVersion. These can easily be derived from the task parameters (see Figure 1).

DATASET-LEVEL

The Define-XML must contain the metadata for all datasets that are part of the submission package. The metadata information at dataset-level includes:

- Dataset: the name of the domain to which the datasets belongs to, particularly useful to identify the main domain of split datasets and supplemental qualifiers, if applicable.
- Description: the dataset label, which can be found in the dataset's metadata in SAS.
- Class: the domain class, restricted to non-extensible controlled terminology, and which can be found in the standard's metadata in the CDISC Library.
- Structure: a description of the general record structure, normally based on the key variables of the datasets. Although this is a free-text field, a default structure is present in the CDISC Library, which can be used as-is or manually reviewed in the intermediate Excel file.
- Purpose: equal to either "Tabulation" for SEND and SDTM, or "Analysis" for ADaM, and can therefore be set automatically based on the standard.
- Repeating: indicates if the dataset has more than one record per subject, which can be set automatically based on the dataset and the standard.
- IsReferenceData: indicates if the dataset contains non-subject data, such as the Trial Design domains, which can be set automatically based on the dataset and the standard.
- Comment: free-text field; some comments, such as for split domains, can be created automatically.

The Datasets sheet in the intermediate Excel file also includes a column for the list of key variables, which can be automatically derived from the dataset's metadata in SAS if it contains the sorting information. This information is stored in the Define-XML in the variable-level attribute KeySequence, described below.

VARIABLE-LEVEL

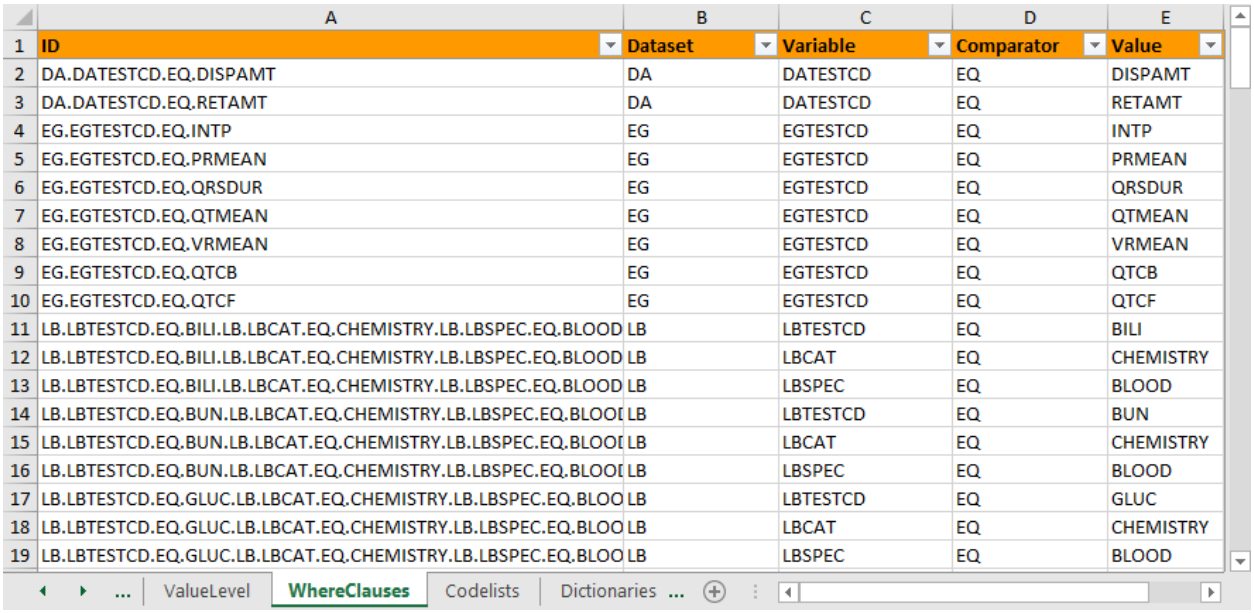
The metadata of each variable for each dataset being submitted must be described in the Define-XML. This includes, among others:

- Order: the order of the variable in the dataset, which can be obtained from the dataset independent of the data it contains.
- KeySequence: populated for key variables with their number in the sequential order of the key. This can be determined based on the sorting information of the dataset, if available.
- Mandatory: specifies whether null values are allowed for a particular variable. It can be automatically assigned for SDTM core variables, where Mandatory = "Yes" for all required variables, and it may be assigned "No" for all ADaM variables.
- Role: this field is not applicable for ADaM variables and is optional for SDTM variables, where the values provided in the implementation guide (and therefore in the CDISC Library) are used.
- DataType: indicates if the variable contains text, integer or float numbers, or datetime values. This can be derived automatically if there is data available by the time of the creation of Define-XML.
- Length and SignificantDigits: the maximum expected length of a value in the variable; significant digits are only applicable for float variables, and describe the number of possible digits for a value after the decimal point. A default value can be automatically derived for these two attributes based on the variable lengths and data, if provided, in the SDTM and/or ADaM datasets.
- DisplayFormat: formats are applicable for float variables, where it can be set as <Length>.<SignificantDigits>, and for datetime variables, where it describes the applicable date format (usually ISO8601 for SDTM datetime variables, or the variable format as represented in the SAS dataset metadata).
- Origin Type: the source of the variable; this can be automated for some specific variables, such as STUDYID and DOMAIN, coding variables, and derived variables like --DY and --SEQ, as well as variables whose values come from the aCRF.
- Codelist ID: the ID of the codelist or external dictionary that the variable is subjected to, if any. This can be determined for variables subjected to controlled terminology based on the CDISC standard metadata, as well as derived for specific variables such as flag variables.
- Method ID: the ID of the algorithm used to derive the variable, if applicable. This can be derived for specific variables, such as --SEQ and --DY variables.

VALUE-LEVEL

Value-level metadata is required for certain specific variables, such as QVAL from supplemental qualifiers, and for variables with metadata attributes, like the data origin, that differ between groups of observations. The metadata fields mirror those at variable-level and can be derived in a similar manner if there is data already available for a set of default variables, such as --ORRES, QVAL and TSVAL in SDTM or AVAL(C) in ADaM.

Moreover, value-level metadata contains an extra metadata field, the Where Clause that defines the condition to subset the variable into the correct groups of observations; this can be easily automated when deriving value-level metadata. For example, when creating value-level metadata for --ORRES, --TESTCD can be used to subset the variable into meaningful groups of observations; the where clause metadata is then organised in the intermediate Excel file as shown in Figure 4.



	A	B	C	D	E
1	ID	Dataset	Variable	Comparator	Value
2	DA.DATESTCD.EQ.DISPAMT	DA	DATESTCD	EQ	DISPAMT
3	DA.DATESTCD.EQ.RETAMT	DA	DATESTCD	EQ	RETAMT
4	EG.EGTESTCD.EQ.INTP	EG	EGTESTCD	EQ	INTP
5	EG.EGTESTCD.EQ.PRMEAN	EG	EGTESTCD	EQ	PRMEAN
6	EG.EGTESTCD.EQ.QRSDUR	EG	EGTESTCD	EQ	QRSDUR
7	EG.EGTESTCD.EQ.QTMEAN	EG	EGTESTCD	EQ	QTMEAN
8	EG.EGTESTCD.EQ.VRMEAN	EG	EGTESTCD	EQ	VRMEAN
9	EG.EGTESTCD.EQ.QTCB	EG	EGTESTCD	EQ	QTCB
10	EG.EGTESTCD.EQ.QTCF	EG	EGTESTCD	EQ	QTCF
11	LB.LBTESTCD.EQ.BILI.LB.LBCAT.EQ.CHEMISTRY.LB.LBSPEC.EQ.BLOOD	LB	LBTESTCD	EQ	BILI
12	LB.LBTESTCD.EQ.BILI.LB.LBCAT.EQ.CHEMISTRY.LB.LBSPEC.EQ.BLOOD	LB	LBCAT	EQ	CHEMISTRY
13	LB.LBTESTCD.EQ.BILI.LB.LBCAT.EQ.CHEMISTRY.LB.LBSPEC.EQ.BLOOD	LB	LBSPEC	EQ	BLOOD
14	LB.LBTESTCD.EQ.BUN.LB.LBCAT.EQ.CHEMISTRY.LB.LBSPEC.EQ.BLOO	LB	LBTESTCD	EQ	BUN
15	LB.LBTESTCD.EQ.BUN.LB.LBCAT.EQ.CHEMISTRY.LB.LBSPEC.EQ.BLOO	LB	LBCAT	EQ	CHEMISTRY
16	LB.LBTESTCD.EQ.BUN.LB.LBCAT.EQ.CHEMISTRY.LB.LBSPEC.EQ.BLOO	LB	LBSPEC	EQ	BLOOD
17	LB.LBTESTCD.EQ.GLUC.LB.LBCAT.EQ.CHEMISTRY.LB.LBSPEC.EQ.BLOO	LB	LBTESTCD	EQ	GLUC
18	LB.LBTESTCD.EQ.GLUC.LB.LBCAT.EQ.CHEMISTRY.LB.LBSPEC.EQ.BLOO	LB	LBCAT	EQ	CHEMISTRY
19	LB.LBTESTCD.EQ.GLUC.LB.LBCAT.EQ.CHEMISTRY.LB.LBSPEC.EQ.BLOO	LB	LBSPEC	EQ	BLOOD

Figure 4. Example of the structure of the WhereClauses sheet and the metadata information it contains.

CODELISTS

This section of the Define-XML contains the metadata for all codelists, be it controlled terminology or user-defined codelists, as well as all external dictionaries, that are used in the study at either variable-level or value-level. It includes the following fields:

- Codelist Name: the name of the codelist, usually corresponding to the codelist name as presented in the controlled terminology. In case of external dictionaries, this is the full name of the dictionary presented as the Preferred Term in the CDISC Dictionary Name Terminology codelist.
- Dictionary Name: applicable only to external dictionaries, this corresponds to the short name of the dictionary presented as the Submission Value in the CDISC Dictionary Name Terminology codelist.
- Dictionary Version: applicable only to external dictionaries, this is the version of the external dictionary used in the study and must be manually checked.
- DataType: the type of data of the terms present in the codelists (text, float, or integer).
- CodedValue: the value of each term of the codelist used in the study. For controlled terminology, this corresponds to the Submission Value.
- Decoded Value: used when the term contains abbreviations or acronyms, to display the full term; for example, in a codelist for --TESTCD, this would be the corresponding value in --TEST. For controlled terminology, this corresponds to the Preferred Term.

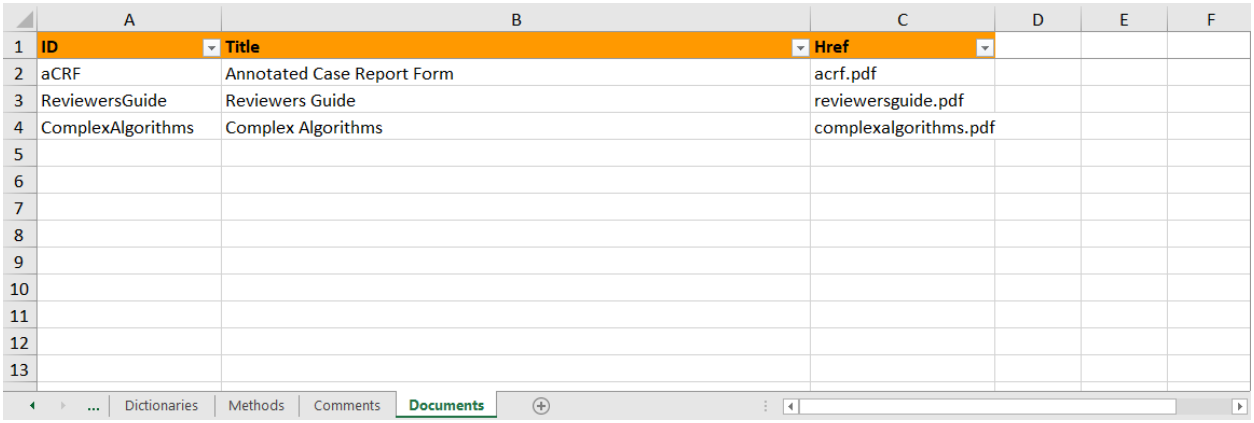
Aside from these attributes, the Define-XML metadata should also describe the NCI codes of any terms and codelists that follow controlled terminology. This can be automatically set using the CDISC Library API.

METHODS

This section of the Define-XML contains all derivation algorithms used to populate variables or groups of observations (value-level), identified with an origin of Derived. Some variables, like --SEQ and --DY, are always derived, and a standard method can be automatically created for this sort of variables, which ensures consistency across different Define-XML documents.

SUPPORTING DOCUMENTS

This section of the Define-XML describes all other documents included in the submission package aside from the datasets and the Define-XML itself, such as the aCRF) and the reviewer's guide. It describes each supporting document, as well as their location in the submission package. An example of how this metadata is inserted in the intermediate Excel file is shown in Figure 5.



	A	B	C	D	E	F
1	**ID**	**Title**	**Href**			
2	aCRF	Annotated Case Report Form	acrf.pdf			
3	ReviewersGuide	Reviewers Guide	reviewersguide.pdf			
4	ComplexAlgorithms	Complex Algorithms	complexalgorithms.pdf			
5						
6						
7						
8						
9						
10						
11						
12						
13						

Figure 5. Example of the structure of the Documents sheet and the metadata information it contains.

It includes the following metadata fields:

- ID: the identification of the document to be used in the Define-XML.
- Title: the label to be displayed for the hyperlink.
- Href: the relative path to the document location; usually the documents are placed in the same folder as the Define-XML, in which case the HREF can be just the name of the file, including the file extension.

This sort of metadata allows for the inclusion of links to specific documents in a comment or algorithm, or even to specific pages or sections, such as the links to pages of the aCRF when describing the origin of a variable or group of values.

FUTURE ENHANCEMENTS

As presented in this paper, the current version of our this tool already allows for the automation of the bigger part of the metadata information. However, there are more steps that can be taken towards a more flexible and customisable tool.

CUSTOM METHODS AND COMMENTS

Currently, there is a list of fixed methods and comments to be applied to specific variables, such as USUBJID, --SEQ and --DY variables. However, these are integrated directly in the program, and not part of the input, meaning that they cannot be easily modified before the program runs. Having instead a list of high-level methods and comments that can be used as input for the program will allow for them to be customisable according to the needs of each studies, without sacrificing consistency within and across studies.

DEFINE-XML CONTROLLED TERMINOLOGY

At the end of 2019, Define-XML controlled terminology was added, defining controlled terms for domain classes and subclasses, analysis result metadata (ARM), and dictionary names. It has since then been expanded to include terminology for origins and standard type and names.

Using this controlled terminology, it is possible to guarantee that Define-XML fields covered by it are consistent and correct. For example, before the release of the controlled terminology, all information for external dictionaries that could be used in submission data had to be hardcoded into the program; now, it is possible to obtain the dictionary title and name automatically from the controlled terminology, which removes the need to keep your own set of external dictionary metadata.

DEFINE-XML V2.1 AND ANALYSIS RESULT METADATA

The current structure of the output Excel file mirrors that of Pinnacle 21 Community, which only supports Define-XML v2.0 for SDTM, ADaM and SEND, and therefore lacks fields for the new metadata information in Define-XML v2.1 or for ARM. Consequentially, our tool does not currently support Define-XML v2.1 or ARM.

However, it is possible to adapt the current tool to be vendor-independent; namely, if a second program is used to convert the information in the Excel file into an XML file, rather than Pinnacle 21 Community, it is then possible to adapt the current structure in order to be able to create Define-XML v2.1 and ARM.

MULTI-TOOL INTEGRATION

As it can be seen in Figure 1, the current version of the tool has one separate task per CDISC standard. However, many of the derivation programs used behind the scenes are similar between standards: for example, the length of a variable or group of values is independent of the standard used. Integrating all tasks in a single master task has benefits both for the user and the programmer: from the user's perspective, there is only one program to learn how to use regardless of the standard(s) used in the submission(s); from the programmer's perspective, the code of the program is much easier to maintain and to upgrade if there is a single task rather than several.

CONCLUSION

The development of Define-XML is an important part of the creation of any submission package, and one must take care to make this document clear, accurate and complete. However, much of the information contained in a Define-XML can be automated based on the data and standards' metadata available, making this task much less daunting and time-consuming. The application of the methods and ideas shown in this paper can lead to a faster and consistent process of Define-XML development, accessible to anyone.

ACKNOWLEDGMENTS

I would like to thank my colleagues who have helped me along the way: Jules van der Zalm, for being there to help from the start to the end; Mylène van Doorn, Joske van Schijndel and Kai Wanke, for all the help provided in the development and validation of this tool; and Lieke Gijsbers and Fatima Kassim, for their valuable insight for this paper.

RECOMMENDED READING

For more information on Define-XML, please consult the CDISC Define-XML Specification and PHUSE Define-XML Version 2.0 Completion Guidelines.

We have also presented a sister tool to this paper during PHUSE EU Connect 2019, discussing our approach to Define-XML validation: "Work less – do more: An automated approach to Define-XML validation", by Kai Wanke.

CONTACT INFORMATION

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

Sofia Vale
OCS Life Sciences
Ruwekampweg 2G
5222 AT, 's-Hertogenbosch
The Netherlands
+31 (0)73 523 6000
sasquestions@ocs-consulting.com