PharmaSUG 2018 - Paper AD-29

Generating Define.xml from Pinnacle 21 Community

Pinky Anandani Dutta, Inclin, Inc

ABSTRACT

Define.xml is an XML document that describes the structure and contents (metadata definitions) of the data collected during the clinical trial process[1]. In December 2011, the CDER Common Data Standards Issues Document[2] stated that "a properly functioning define.xml file is an important part of the submission of standardized electronic datasets and should not be considered optional." While the metadata definitions of a clinical study may not be the most difficult deliverable to create, transforming those specifications to an xml file can be a daunting experience for many SAS programmers who lack skills in other programming languages. This paper details a simpler process that can be followed to generate the Define.xml document for a study without the need of any additional programming skills. Use of this procedure eliminates the additional time needed to transform study metadata into Define.xml at the end of the study. At the same time, this procedure also allows for easy updates and validation of this document simultaneously throughout the course of the study. The software packages used primarily in this procedure are Pinnacle 21 Community and Microsoft Excel.

INTRODUCTION

Define.xml is an essential part of a New Drug Application (NDA) submission package with the Food and Drug Administration (FDA)[1]. It is used to describe CDISC SDTM and ADaM datasets for the purpose of submissions to the FDA[1]. When included in an NDA submission package, this document increases the level of automation and improves the efficiency of the Regulatory Review process, making it highly desirable with every NDA submission[1].

In the past, creating a submission-ready Define.xml required a firm knowledge of the standards and mastery of XML[3]. Absence of such knowledge turned out to be a major setback for many clinical programmers who were working on an NDA submission. In the year 2014, Pinnacle 21 released OpenCDISC v2.0, which for the first time included the Define.xml generator feature. As Sirichenko *et al.* mentioned in their paper[3], Pinnacle 21's goal in creating the Define.xml Generator was to eliminate the need for prior knowledge of XML and lower the barrier to learning and becoming proficient with the Define.xml v2.0 standard. The Define.xml Generator is therefore based on a tool as simple as Excel, allowing a user to focus more on the metadata content of the study rather than a complex XML syntax[3].

Previously known as OpenCDISC, Pinnacle 21 Community is a free and user-friendly open source toolkit by Pinnacle 21. It is commonly used by CDISC programmers to validate SDTM and ADaM datasets. In addition to CDISC data conformance validation, Pinnacle 21 Community also features other tools such as the Define.xml generator, Data Converter and ClinicalTrials.gov Miner. This paper focuses on the Define.xml generator feature of the Pinnacle 21 Community software and details the steps involved in using this software to successfully generate a Define.xml.

The Define.xml generator of the Pinnacle 21 community software performs two functions. The first of the two functions is to create a Pinnacle 21 format spec. The second function is to generate a Define.xml file using a Pinnacle 21 format spec. The process described in this paper outlines using these two functions consecutively to generate a Define.xml. It should be noted that the process of generating a Define.xml document differs slightly when generating this document for SDTM datasets versus ADaM datasets and can only be done one at a time. We will look at the steps involved in generating a Define.xml for SDTM datasets in full length, and highlight the differences in creating the same for ADaM datasets along the way.

GENERATING THE PINNACLE 21 FORMAT DRAFT SPEC FILE

A Define.xml contains the metadata information of the NDA submission datasets and its variables. Different categories of the metadata information are organized as individual tabs in the Pinnacle 21 format specification. This information is categorized into the following ten categories:

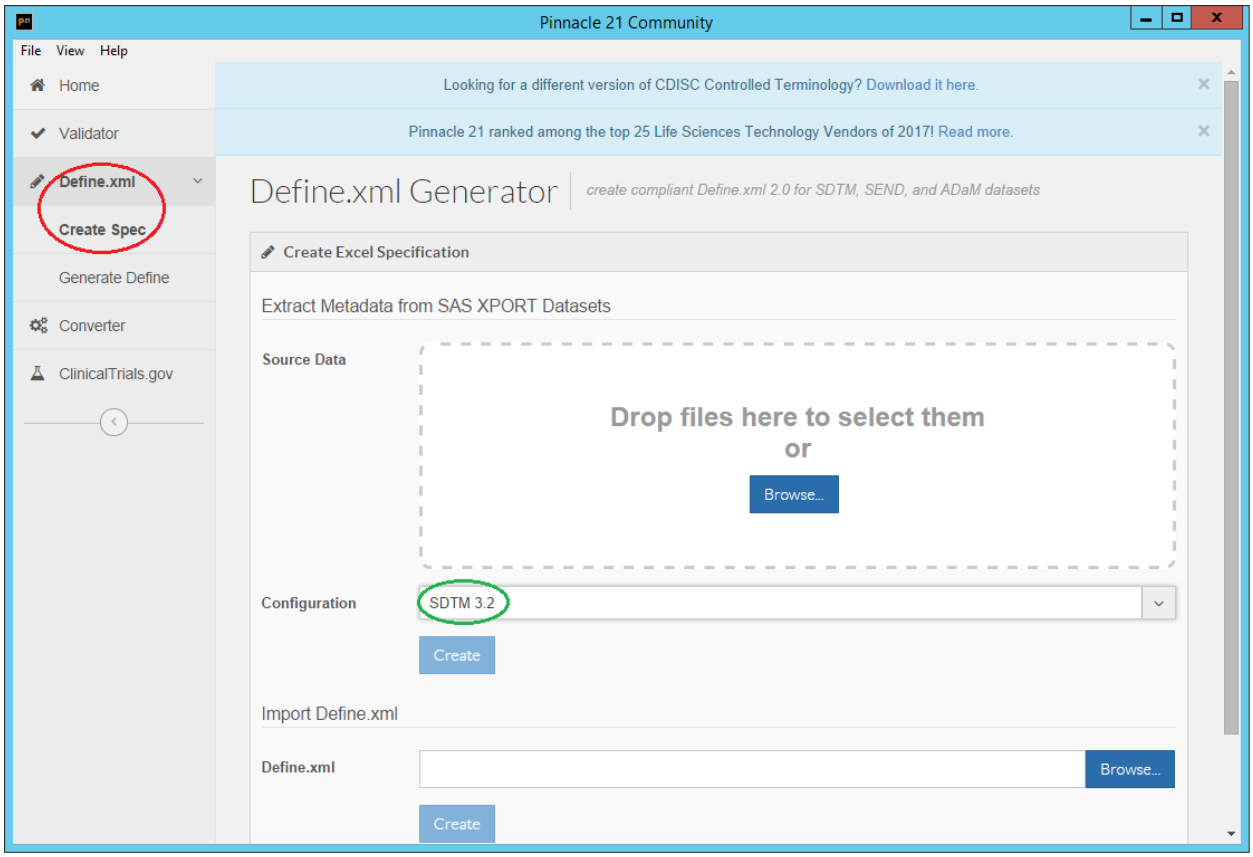
1. Study
2. Datasets
3. Variables
4. Value Level Metadata
5. Where Clauses for Parameter Value Level Metadata
6. Codelists
7. Dictionaries
8. Methods
9. Comments
10. Documents

As discussed earlier, the first step in generating a Define.xml document from the Pinnacle 21 community software is to create a Pinnacle 21 format spec file. In order to generate the Pinnacle 21 format spec file, access to the Pinnacle 21 Community software is required. This software can be downloaded from https://www.pinnacle21.com/downloads. Alternatively, OpenCDISC Community v2.0 or higher could be used. This can be can be downloaded from https://www.pinnacle21.com/news/introducing-opencdisc-community-20.

As part of the spec creation process, the Define.xml generator first scans an existing set of SAS® XPORT datasets to obtain the metadata information only for the Datasets and Variables categories as listed above. This metadata is then used to create and populate an excel format Pinnacle 21 format draft spec file[3]. Before proceeding with the spec creation, it is recommended to designate a folder for the Define.xml work and place the study's SAS® XPORT datasets in this folder.

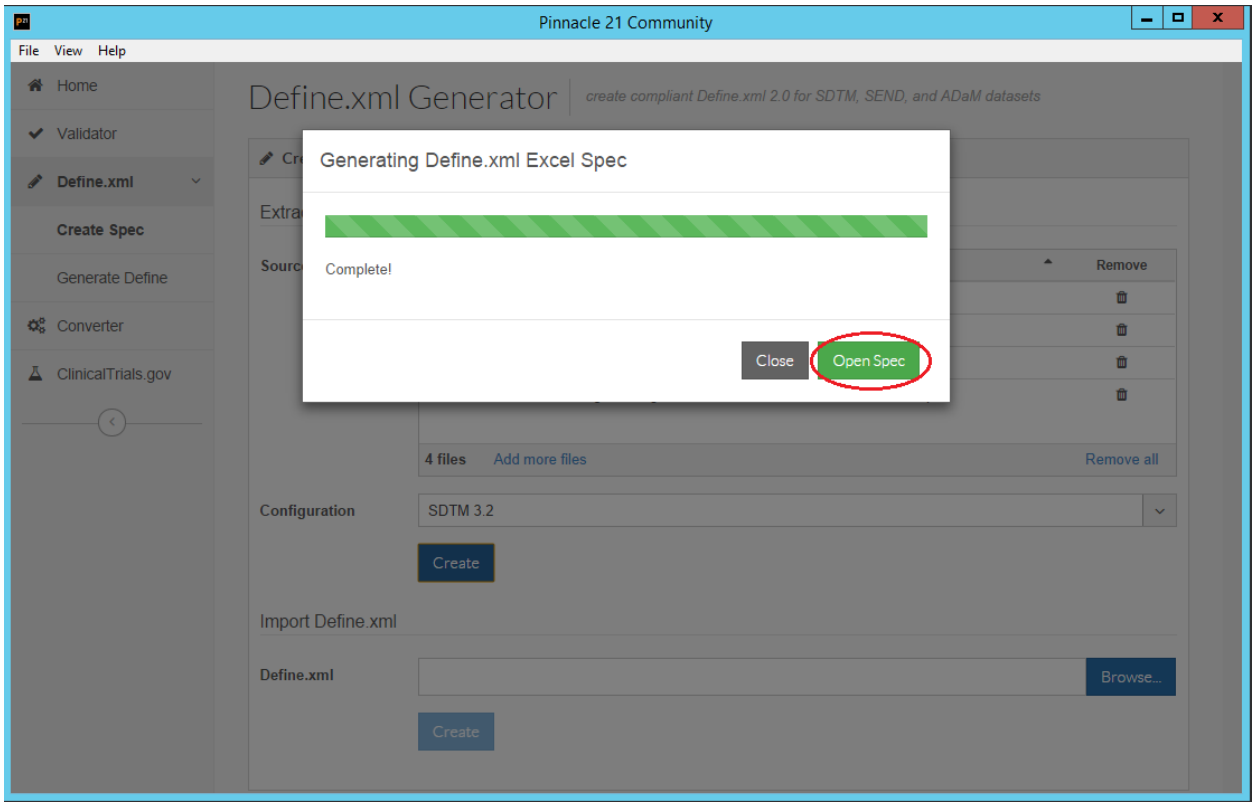
To create the Pinnacle 21 format draft spec file, follow these steps:

1. Double click on the Pinnacle 21 community software icon to initiate the software.
2. Click on the Define.xml menu from the tab menu at the upper left corner of the software's interface. This should unhide the functions under the Define.xml menu.
3. From the unhidden menu, select "Create Spec". The spec creation menu will open up on the right side of the window and will look as shown in Display 1.



Display 1. Pinnacle 21 Define.xml Spec Creation Interface

4. Select the SDTM version being used from the drop down menu of the Configuration as circled in green in Display 1. For creating a Define.xml file for ADaM datasets, select the ADaM version instead from the same drop down menu.
5. Obtain the Source Data by using the Browse button in the Source data field and navigating to the location of the SDTM transport files. For creating a Define.xml file for ADaM datasets, navigate to the location of the ADaM transport files instead.
6. Use Ctrl+A to select all the transport files at once or manually click on each of the transport files to select them.
7. Once selected, press OK and the “Create” button under the Configuration field will unlock and turn dark blue in color.
8. Now, click the “Create” button and Pinnacle 21 will start to generate the spec file. Once done, a new window will pop up on the screen (Display 2) confirming the completion.
9. Click the “Open Spec” button to open the spec file.



Display 2. Confirmation of Define.xml Excel Spec Generation

10. Once the spec file is open, use the SAVE AS function from the File menu of this file to save it to the Study's Define folder.

POPULATING THE SPEC FILE CORRECTLY

There are several tabs in the spec file that need to be filled out correctly in order for the define.xml to reflect the right information about the study. While this section will go over each one of them one by one, here are a few things to be cautious of as we populate the spec file:

- Excel has a propensity for auto-correction. Ensure that entries into the spec file have not been altered by this propensity.
- An XML file can become unreadable if it involves any special characters. It is easy to introduce special characters into the spec file by Copy-Paste from another text source. To avoid this problem, use Paste Special instead of Paste.

STUDY TAB

The Study tab requires only three pieces of information: Study Name, Study Description and Protocol Name, all of which can be found in the Study Protocol. In most clinical studies, the Study Name and the Protocol Name are the same. To populate the Study Tab, locate this information in the Study Protocol and populate it in its respective cells of Column B of this tab.

DATASETS TAB

While Pinnacle 21 does a great job at populating this tab, not all of the pre-populated information is correct. It is a good idea to verify the pre-populated information, and populate what is missing. We will discuss the information that goes into each of the columns on this tab as follows:

Dataset column: This column shows the name of the datasets per the transport files fed to Pinnacle 21 during the Spec creation process.

Description, Class & Structure Columns: Metadata information as it relates to these three columns can be found in the SDTM Implementation Guide[6] v.3.2 Table 3.2.1. To complete these columns, first ensure that the columns are populated correctly for the datasets listed the Dataset column. If the information in any of these columns is missing, populate it accordingly.

Purpose column: All records of the Purpose column (Column E) should indicate "Tabulation". If that is not the case, then update to indicate "Tabulation". For ADaM datasets, this column should indicate "Analysis".

Repeating column: As for the Repeating column (Column F), one record per subject or one record per test code datasets such as DM, IE, TI, TS, etc. are to be marked as "No". These datasets fall under the CDISC SDTM classes of Special Purpose & Trial Design. Datasets that belong to classes other than the Special Purpose or Trial Design are considered repeating datasets and can be marked as "Yes".

Reference Data column: All datasets that fall under the Trial Design class are considered reference data and should be marked in the Reference Data column (Column H) as "Yes". Datasets that do not fall under the Trial Design class should be marked as "No" in this column.

Key Variables column: The Key Variables column (Column F) is usually populated inaccurately by Pinnacle 21. It needs to be populated with the key variables used to create the Sequence number variable (XXSEQ) for a particular dataset. Additionally, key variables that are populated in this column should be tested for uniqueness. If the key variables do not identify unique records in the dataset it is listed for, additional variables should be included in the sequence key and re-tested for uniqueness of the dataset by that sequence key.

Comment column: The Comment column (Column G) is not required to be populated. Hence, it is fine if there are no comments to mention. However, if there are comments to mention, here's how to use this column:

1. In the comment column, specify an ID for the comment. For example, the ID is COMMENT.AE.
2. Now, mention this ID in the ID column of the Comment tab (Column A of the Comments tab).
3. Finally, place the comment in the Description column of the Comments tab (Column B of the Comments tab).
4. In the Comments tab, there is a column for Document and Pages (Column C and D respectively). If there is a document that references or is associated to a comment (Column B of the Comments tab), then place the ID of that document in the Document column of the Comments tab. This ID should also be listed in the Documents tab, where all reference documents for the study are listed. The Documents tab will be discussed in further detail in the next section. Page numbers of the comment reference document can be indicated in Column D of the Comments tab. In the case of multiple page numbers, place all page numbers in Column D of the Comments tab separated by a single space and listed out in ascending order (e.g. "5 6 7 9 12").

DOCUMENTS TAB

This tab lists any documents we wish to submit along with the Define.xml. In order for a document's hyperlink to work correctly in the Define.xml file, documents listed on this tab must be present in the Study's Define folder with the same file name as listed in the Href column of this tab. While there aren't too many documents to submit, the Annotated CRF and Reviewers Guide are typical with every submission and included on this tab as follows:

ID	Title	Href
Blankcrf	Annotated Case Report Forms	blankcrf.pdf
ReviewersGuide	Reviewer's Guide	ReviewerNotes.pdf

Display 3. Screenshot of the Documents Tab of Pinnacle 21 Define.xml Spec File

DICTIONARIES TAB

This tab lists any dictionaries used in this study. While there aren't too many of them, the MedDRA and WHODRUG dictionaries are typical for most studies and are included on this tab as follows:

ID	Name	Data Type	Dictionary	Version
DRUGDICT	Drug Dictionary	Text	WHODRUG	201403
AEDICT	Adverse Event Dictionary	Text	MedDRA	17.0

Display 4. Screenshot of the Dictionaries tab of the Pinnacle 21 Define.xml Spec file.

VARIABLES TAB

This tab consumes the most time in the entire spec population process. Maintenance of clean and up-to-date specifications during the programming phase can significantly reduce the amount of time spent on this tab. In order to populate this tab, we shall discuss them column by column, while discussing exceptions along the way.

Apparently, Columns A-I of this tab are pre-populated quite well by Pinnacle 21. While most of the information pre-populated here is correct, here are a few things to watch out for in these columns:

Order Column: The order of the variables in this column reflects the order of the variables in the actual dataset. Hence, if the order of the variables is found to be incorrect here, it must be corrected both here and in the source dataset.

Label Column: The variable labels displayed here reflect the variable label in the actual dataset. Hence, if the label of a variable is found to be incorrect here, it must be corrected both here and in the source dataset. Additionally, the labels for variables that repeat over multiple datasets, such as STUDYID, DOMAIN, USUBJID, XXSEQ, etc., should be the same across the different datasets. If that is not the case, the dataset will need to be corrected, wherever applicable, and the spec will need to be updated for the same. More information on SDTM variable labels can be found in the SDTM Implementation Guide[6].

Data Type Column: The Data Type is pre-populated correctly for the most part. However, sometimes it populates incorrectly for date variables. All SDTM date variables should be populated for Data Type as "datetime". If that is not the case, update to indicate "datetime". Other possible values for this column are text, integer, float, datetime, date, time, partialDate, partialTime, partialDatetime, incompleteDatetime, and durationDatetime. More details on which of these choices should be adequately used can be found in Section 4.2.1 of the CDISC Define-XML Specification v2.0 document [4].

Length Column: To complete this column, ensure that the variable length is populated (not missing) for all variables except those whose Data Type (Column E) equals datetime. Variables where Data Type equals datetime should be left blank here.

Format Column: For variables where Data Type (Column E) equals datetime, populate this column as ISO 8601. The rest of the pre-populations can be left as is.

Mandatory Column: For the most part, Pinnacle 21 populates this column completely. However, in cases of a custom domain, Pinnacle 21 may not populate this column at all. It is important to ensure that this column is populated completely and that there are no missing values. For SDTM variables that are Required (Core = Required) per the SDTM Implementation Guide[6], this column should be populated as "Yes". At the same time, for SDTM variables, whose core is Expected or Permissible, and for whom the sponsor does not require a more restrictive condition, this column should be populated as "No". The only permissible values for this column are Yes and No. In case of missing values, populate this column accordingly. It is important to note that variables where this column has been set to "Yes" must not have a null value in the dataset.

Columns J-P of this tab require a bit more work than Columns A-I as Columns J-P do not have any data pre-populated and will need to be populated by the study programmer. Let's take a look.

Codelist Column: The codelist column requires populating the unique ID of the codelist to which each variable is associated. While most variables do not have a codelist associated with them, many do. Many variables have a set NCI codelist specified in the SDTM Controlled Terminology[5], while others may have a custom codelist defined by the Sponsor. These codelists must be listed in the Codelist tab, which we will cover in more detail in the next section. The codelist ID (Column A of the Codelist tab) must be populated here to indicate its association with that variable.

Origin Column: The origin of a variable is populated per one of the following allowable values: CRF, Derived, Assigned, Protocol and eDT for SDTM. More details on the appropriate selection of these values can be found in Section 5.3.11.3 of the CDISC Define-XML Specification v2.0 document[4].

Pages Column: The pages column is to be populated only when the Origin column (Column K) indicates CRF. This column is used to list out the CRF page number(s) where the information for the variable in question originates from. A single page number may be listed here by itself, while a reference to multiple pages can be made by separating the page numbers with a single space (e.g. "6 7 12"). When referencing multiple page numbers, the page numbers must be listed in ascending order. This column must remain blank for an ADaM Define.xml file.

Method Column: This column is to be populated only when the Origin column (Column K) indicates "Derived". It requires populating the unique ID of the method a particular variable is associated with. These methods must be listed in the Methods tab, which we will cover in more detail in the Methods Tab section. This Unique ID will be listed in Column A of the Methods tab and must be populated here with the associated variable.

Predecessor Column: The Predecessor column is used only for ADaM Define.xml files. In an ADaM dataset, if a variable originates directly from an SDTM dataset, the predecessor column displays the domain abbreviation and the variable name of that SDTM dataset, separated by a period (e.g. DM.SUBJID). This column is to be left null for an SDTM Define.xml file.

Role Column: The Role column is pre-populated by Pinnacle 21 with the appropriate information from the SDTM Implementation Guide[6]. On occasion, this information may be missing, especially for custom domains. One may choose to populate the missing information in this column by referring to the SDTM Implementation Guide[6]. However, this is not a required component of the Define.xml output. Hence, completing any missing information in this column can be omitted.

Comment Column: The Comment column requires populating the unique ID of the comment a particular variable is associated with. These comments must be listed in the Comments tab, which we will cover in more detail in Comments Tab subsection. This Unique ID will be listed in Column A of the Comments tab and must be populated here with the associated variable. It is important to note that for any variable, either the Comments column (Column P) or the Method column (Column M) can be populated. Populating both of these columns will result in an override problem. For simplicity's sake, I recommend using the Methods tab and column exclusively for all methods and comments. Both of these appear in the same column on the Define.xml output. Hence, separating the methods and comments into two different tabs doesn't hold much value.

CODELISTS TAB

This tab contains all codelists for the study, including both the National Cancer Institute (NCI) codelists as well as the user-defined codelists (also known as sponsor-defined codelists). Let's take a look at the NCI codelists first.

NCI Codelists

NCI codelists are codelists that are defined by the National Cancer Institute and adapted by CDISC for SDTM controlled terminology. These codelists are available online at the CDISC website [5]. In order to demonstrate how to fill in the Codelist tab, let's work on the codelist of the variable LBNRIND from the LB

domain. Here's what the codelist looks like in the SDTM Controlled Terminology for NRIND (the codelist name for variable LBNRIND).

Code	Codelist Code	Codelist Extensible (Yes/No)	Codelist Name	CDISC Submission Value	CDISC Synonym(s)	CDISC Definition	NCI Preferred Term
C78736		Yes	Reference Range Indicator	NRIND	Reference Range Indicator	A description of the results relative to the normal or expected range of values.	CDISC SDTM Reference Range Indicator
C78802	C78736		Reference Range Indicator	ABNORMAL		Reported values outside the typical or expected range. (NCI)	Abnormal Reference Range
C78800	C78736		Reference Range Indicator	HIGH		Reported values above the typical or expected range. (NCI)	Value Above Reference Range
C78801	C78736		Reference Range Indicator	LOW		Reported values below the typical or expected range. (NCI)	Value Below Reference Range
C78727	C78736		Reference Range Indicator	NORMAL		Reported values within the typical or expected range. (NCI)	Normal Reference Range

Display 5. Screenshot of NRIND NCI Codelist

This information can be included in the Codelists tab as follows:

ID Column: From the CDISC Submission Value column in Display 5 above, the value of the cells highlighted in blue is to be filled in here.

Name Column: This column is to be filled out from the Codelist Name column in Display 5.

NCI Codelist Code Column: This column is to be filled out from the Codelist Code column as shown in Display 5.

Data Type Column: This column is to be filled out as the value of Column E of the Variables tab for the variable in question.

Order Column: This column shows the order in which the terms of a codelist are printed in the Define.xml file. The order of the codelist values must be listed here starting at 1. For the purpose of ordering, this column allows numeric values only.

Term Column: This column is to be filled out from the CDISC Submission Value column of Display 5 where the Codelist Code column in Display 5 is not missing.

NCI Term Code Column: This column is to be filled out with the values of the Code column in Display 5.

Decoded Value Column: This column is to be filled out with the values of the NCI Preferred Term column in Display 5.

User Defined Codelists

User Defined Codelists are custom codelists created based on study needs. With the following exceptions, they are placed into the codelist tab in a manner very similar to that of the NCI codelists:

ID Column: This column is to be filled out with a unique codelist ID that identifies the codelist. For consistency, one could concatenate the domain name and the variable name, separate them by a dot, and use the concatenation result for the unique codelist ID.

Name Column: This column is to be filled out with a user chosen name for the user defined codelist.

NCI Codelist Code & NCI Term Code Columns: These columns are to be left blank for user-defined codelists.

Term Column: This column is to be filled with the dataset values that comprises this codelist.

Decoded Value Column: If there is a decode available for the values in the Term Column (Column F), place it here.

Things to Note:

- If the Codelist Extensible column as shown in Display 5 indicates a "Yes" (in the cells highlighted in blue), it means that the Codelist is extensible and can hold more values than those defined in the SDTM controlled terminology codelist. Therefore, if the data has additional values for an NCI codelist, the additional values must be included in the Codelists tab under the same Codelist Code value (Column C). As such, the NCI Term Code (Column G) must be left blank.
- Codelist of variables that represent decoded values do not require the Decoded Value column (Column H) to be filled. Examples of such SDTM variables are VSTEST, LBTEST, DSDECOD, PKPARM, TSPARM, etc.

METHODS TAB

This tab stores all of the methods used for the derivation of the derived variables in the study datasets. It can be filled out as follows:

ID Column: This column is to be filled out with the unique Method ID specified in Column M of the Variables tab. An easy way to create this unique ID is to concatenate the domain name with the variable name (Column B and C of the Variables tab), separate them by a dot, and use the resulting string as the Method ID.

Name Column: This column is to be filled out as a concatenation of "Algorithm to derive " and the values of the ID column (Column A of the Methods Tab).

Type Column: This column is to be filled out with the word "Computation".

Description Column: This column is to be filled out with the derivation method.

Expression Content & Expression Code Columns: These columns can be left blank.

Document Column: This column is used to specify any documents that were referenced for the method in question. In order to specify a document here, the document must first be listed in the Documents tab, and its ID (Column A of the Documents tab) must be placed here (Column G of the Methods tab).

Pages Column: This column is used to specify page numbers of any documents specified in Column G. In order to specify the page numbers, simply type in the page numbers in ascending order, separated by a space (for multiple pages).

COMMENTS TAB:

This tab is very similar to the Methods tab. The ID, Description, Comments and Pages columns of the Comments tab are filled up exactly the same way as the ID, Description, Comments and Pages columns of the Methods tab respectively. For assistance on filling out the Comments tab, refer to the Methods tab section of this paper. However, to avoid unnecessary debugging, it is recommended to keep all comments and methods on the Methods tab exclusively. Issues with the Define.xml file's page width have been observed upon the usage of the Comments tab.

VALUELEVEL TAB

This tab allows us to provide the definition of a variable for a specific subset condition. Such information should be provided when there is a need to describe differing metadata attributes for subsets of data within a variable. Values of variables such as --ORRES, --ORRESU, --STRES, --STRESU, and QVAL are often specific to a test code (value of --TESTCD). Hence, value level metadata should be provided for these variables whenever necessary. In order to describe how to fill in the information in this tab, we will describe the Value level metadata for variable VSORRES as follows:

Order Column: The order of the value level metadata for any variable is to start at 1 and continue till the total number of test codes for that variable has been reached.

Dataset Column: This column reflects the source dataset, which is "VS" for the purposes of this example.

Variable Column: This column reflects the variable whose value level metadata is being defined. For the purposes of this example, that would be "VSORRES".

Where Clause Column: This column reflects the specific condition for which the variable is being defined. For the variable VSORRES, there are seven test codes for this example namely BMI, DIABP, HEIGHT, PULSE, RESP, SYSBP, TEMP and WEIGHT. Hence, we have seven where clauses here that need to be specified in the Where Clause tab (described in the next section). For now, we need to fill in an ID for these clauses in this column space. In order to keep things simple, it is recommended to keep the name of the where clause ID the same as that of the test code.

Data Type Column: This column reflects the data type of the values of each of the specific conditions listed in the Where Clause column. It is to be filled out with allowable values as described in the Data Type Column of the Variables Tab section.

Length Column: This column reflects the maximum length of the variable's values for each of the subset groups. It is recommended to keep this length the same as the length of the variable, which can be obtained from Column F of the Variables tab.

Significant Digits Column: This column can be filled out using the information from Column G of the Variables tab.

Format Column: This column is used to specify the different formats applicable to different subsets of the value level data. If the format specified in Column H of the Variables tab is applicable to all subsets, then print that format here; else list the specific format for each specific condition here.

Mandatory Column: This column can be filled out with Column I of the Variables tab.

Codelist Column: This column is used to specify the different codelists applicable to different subsets of the value level data. If the codelist specified in Column J of the Variables tab is applicable to all subsets, then print that codelist here; else list the specific codelist for each specific condition here.

Origin Column: This column is used to specify the different origins applicable to different subsets of the value level data. If the origin specified in Column K of the Variables tab is applicable to all subsets, then print that origin here; else list the specific origin for each specific condition here.

Pages Column: If the Origin column (Column K) indicates "CRF", then the particular page of the CRF where the subset of data originates from must be specified here. All page numbers must be specified in ascending order and separated by a space when populated here.

Method Column: This column is used to specify the different methods by which the different subsets of data are derived or obtained. If the method specified in Column M of the Variables tab is applicable to all subsets, then print that Method's ID here. Otherwise, the ID for the method of each of the specific condition (subset) must be listed here. All of the new Method IDs, along with their derivations, must also be listed in the Methods tab.

Predecessor Column: This column can be left blank for all SDTM Define.xml purposes. In order to use this column for ADaM Define.xml, refer to the instructions in the Predecessor Column under the Variable Tab section.

Value Level Comment Column: If there are any comments to mention for specific conditions, it is best to include them as part of the Method column (Column M) of this tab.

Join Comment Column: This column can be left blank.

WHERECLAUSES TAB

As discussed in the previous section under the Where Clause Column, this tab is primarily used to list out the Where Clauses for each of the specific subset conditions listed out in the ValueLevel tab. We will now discuss how to fill out this tab for the example variable VSORRES and its subset conditions.

ID Column: This column contains the Where clause ID, which can be obtained from Column D of the ValueLevel tab.

Dataset Column: This column reflects the dataset on which the Where clause applies, and is usually the same as Column B of the ValueLevel tab.

Variable Column: This column is to be filled with the variable on which the Where clause is based. For example, if the variable is VSORRES, then the where clause is based on the variable VSTESTCD.

Comparator Column: This column is used to aid in specifying the condition of the where clause, which can be obtained by the use of a comparator. Allowable values of comparators for this column are LT, LE, GT, GE, EQ, NE, IN and NOT IN.

Value Column: This column is used to specify the value of the condition of the Where clause. For example, in the case of VSORRES, one of its where clauses would be WHERE VSTESTCD EQ DIABP. Hence, the value of the condition here is DIABP, while EQ is the comparator.

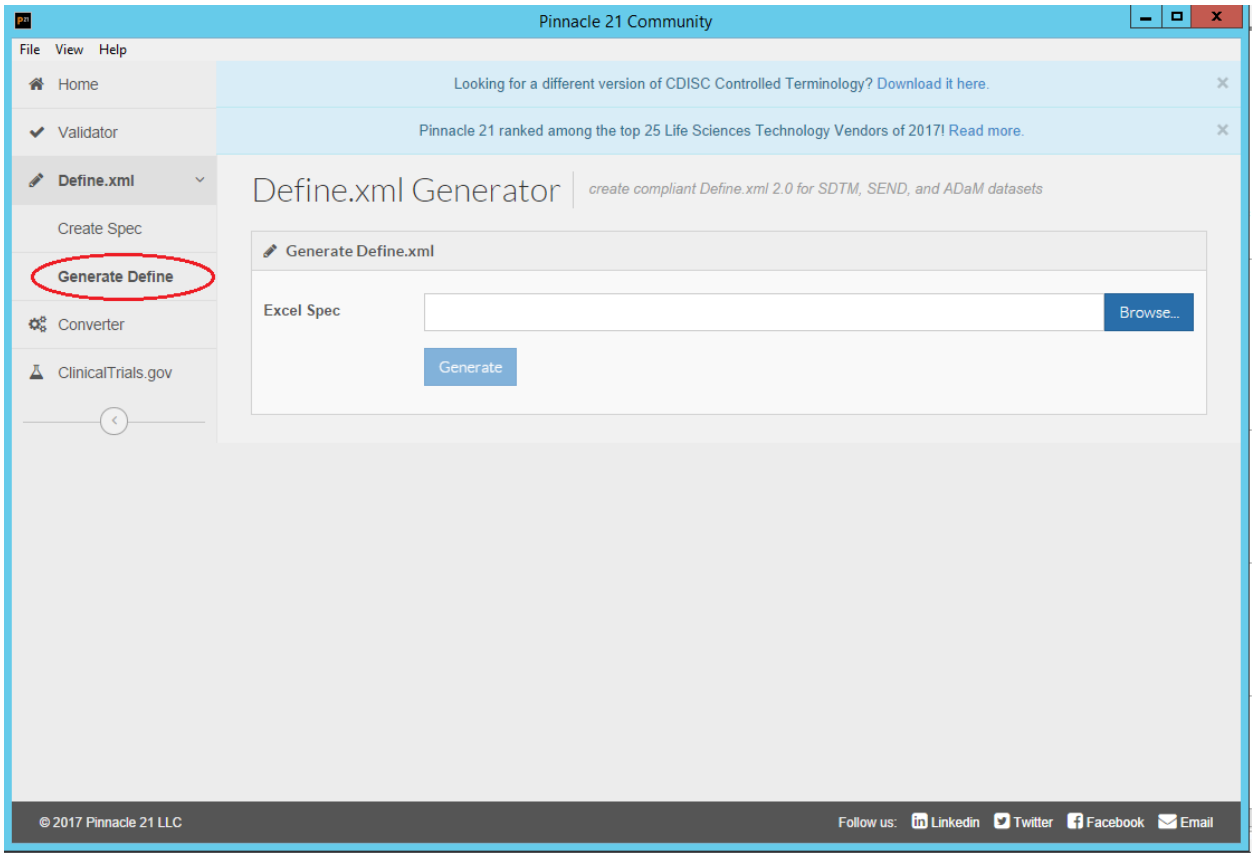
Things to Note:

- The Where Clause tab and ValueLevel tab are heavily dependent on each other. If a ValueLevel is filled out and the Where clause isn't filled out in the Where Clause tab, Pinnacle 21 will fail to generate the Define.xml file. If an issue like this is observed, check to ensure that the ValueLevel tab and WhereClauses tab are populated sufficiently to support each other.

GENERATING THE DEFINE.XML DOCUMENT

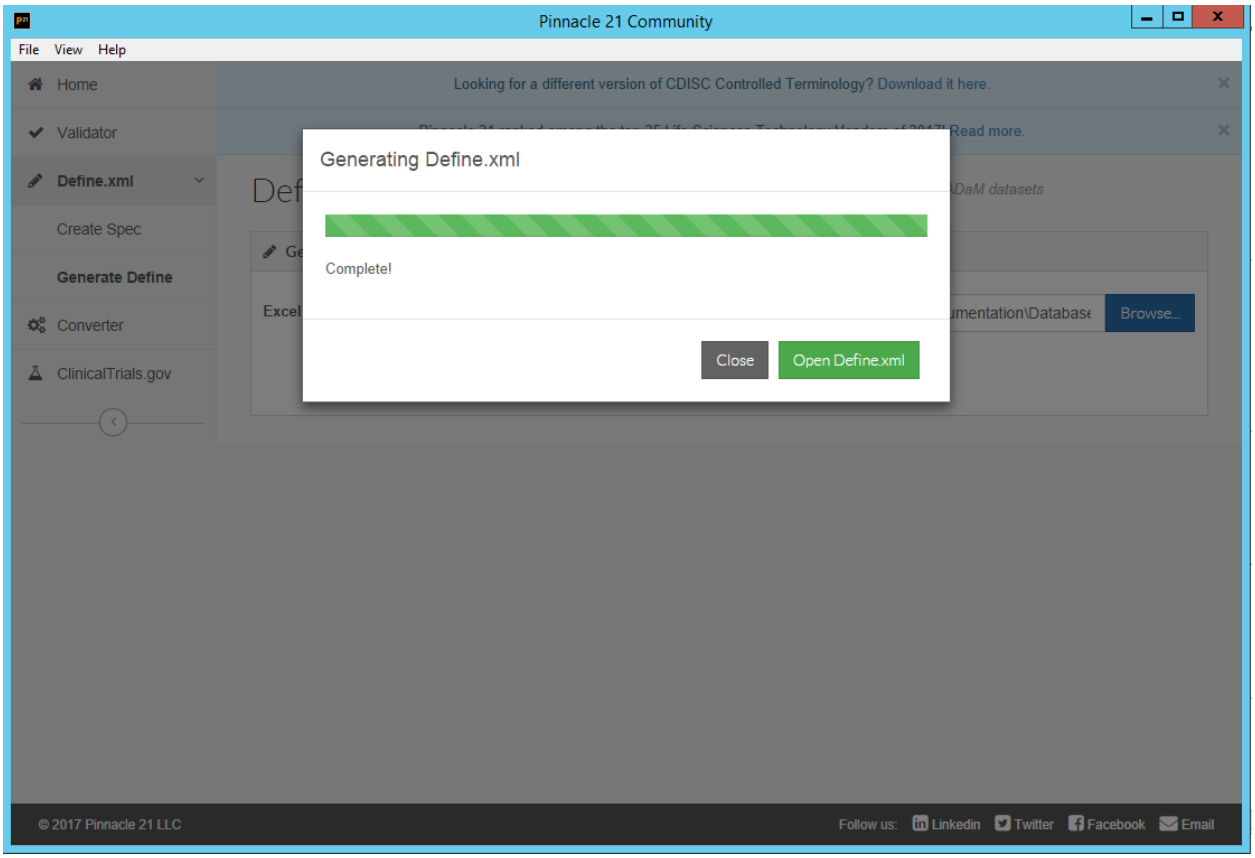
Congratulations! This concludes the most time consuming part of this project. Now that the Spec file has been populated, we are ready to generate the Define.xml file. To generate the Define.xml file, follow these steps:

1. Initiate the Pinnacle 21 software.
2. Click on the Define.xml menu from the tab menu at the upper left corner of the screen. This should unhide the options under the Define.xml menu.
3. From the unhidden menu, select "Generate Define". The Define.xml Generator interface will open up on the right side of the window as shown in Display 6 as follows:



Display 6. Screenshot of Define.xml Generator Interface

4. Click on the browse button located on the right side of the interface to locate the Pinnacle 21 format Spec file created in the previous section.
5. Once the Spec file has been located and the path has been displayed in the space provided in front of Excel Spec (as shown in Display 6), click the Generate button to generate the Define.xml file.
6. Upon successful execution by the software, a completion confirmation will pop up as shown in Display 7 as follows:



Display 7. Screenshot of Define.xml Generation Confirmation

7. Click on the green button of the prompt provided by Pinnacle 21 to view the Define.xml document.

TROUBLESHOOTING

As with any software, some limitations are expected. Being aware of those limitations can help us mitigate risks, minimize wastage of time and effort and enable us to identify standard workarounds. In another PharmaSUG proceeding[3], Sirichenko *et al.* have listed out some common issues encountered using Pinnacle 21 Community that could prevent a user from generating a valid Define.xml file. Additionally, they have also provided useful tips on overcoming the common issues listed in their paper. Review of their paper and familiarization with these issues is highly recommended.

SAVING THE DEFINE.XML DOCUMENT

Each Define.xml document generated by Pinnacle 21 Community is automatically saved in a folder that is part of the Pinnacle 21 software package. In order to save the Define.xml document to the Study's Define folder, follow these steps:

1. Navigate to: C:\\pinnacle21-community\components\reports. All Define.xml documents generated on a computer are automatically stored in this folder.
2. In this folder, find the latest generated Define.xml file and copy it to save to the Study Define folder.
3. In addition to the Define.xml file, the stylesheet file, define2-0-0.xsl must also be copied from this folder to save to the Study Define folder. This is the stylesheet of the Define.xml document, and is required to be in the Study Define folder in order for the Define.xml file to display its contents correctly.

VALIDATING THE DEFINE.XML DOCUMENT

Once the Define.xml is generated, the final step is to mitigate the corner cases and human errors. It is recommended that the Define.xml document is thoroughly reviewed and validated by another study programmer to ensure accuracy. Additionally, the Pinnacle 21 Community validator feature could also be explored and used to validate the Define.xml document either self-standing or in conjunction with study's SAS® transport files.

CONCLUSION

A Define.xml file is a critical component of an NDA submission. Generating a Define.xml file using the Pinnacle 21 community software is an easy and straightforward task that an average Clinical SAS programmer can complete without any prior knowledge of XML. If the Define.xml file is generated during the programming development phase of the study, and used in place of regular SDTM and ADaM specifications, it can be continuously validated by the study programmers for corner cases and human errors. This will also eliminate the time budgeted for transforming regular specifications into Define.xml at the end of the study. Unused budgeted time could in turn benefit an organization in other areas of process development.

REFERENCES

1. Shu, Amos. 2013. "A Practical Approach to Creating Define.XML by Using SDTM Specifications and Excel functions." *Proceedings of the SAS Global Forum 2013 Conference*, San Francisco, CA. Available at http://support.sas.com/resources/papers/proceedings13/201-2013.pdf
2. "CDER Common Data Standards Issues Document Version 1.1." Dec 2011. Available at https://www.fda.gov/downloads/drugs/developmentapprovalprocess/formssubmissionrequirements/electronicsubmissions/ucm254113.pdf
3. Sirichenko, S., DiGiantomasso, M., & Collopy, T. (2016) "Usage of Pinnacle 21 Community Toolset 2.1.1 for Clinical Programmers." *PharmaSUG 2016 Online Conference Proceedings*, Denver, CO. Available at https://www.lexjansen.com/pharmasug/2016/HT/PharmaSUG-2016-HT04.pdf
4. CDISC Define-XML Team, "CDISC Define-XML Specification." *CDISC*, Version 2.0, 24 Apr. 2014, CDISC. Available at http://www.cdisc.org/define-xml
5. "Controlled Terminology." *CDISC*, 22 Dec. 2017. Available at www.cdisc.org/standards/semantics/terminology
6. CDISC Submission Data Standards Team, "Study Data Tabulation Model Implementation Guide: Human Clinical Trials" *CDISC*, Version 3.2, 26 Nov. 2013, CDISC. Available at https://www.cdisc.org/standards/foundational/sdtmig

ACKNOWLEDGMENTS

The author would like to thank Hariprasath Narayanaswamy for his contribution in streamlining the process that led to this paper. The author is also very grateful to Gary E Moore for his valuable feedback and suggestions.

RECOMMENDED READING

CDISC Submission Data Standards Team, "Study Data Tabulation Model Implementation Guide: Human Clinical Trials" *CDISC*, Version 3.2, 26 Nov. 2013, CDISC.

CDISC Define-XML Team, "CDISC Define-XML Specification." *CDISC*, Version 2.0, 24 Apr. 2014,CDISC.

Sirichenko, S., DiGiantomasso, M., & Collopy, T. (2016) "Usage of Pinnacle 21 Community Toolset 2.1.1 for Clinical Programmers." *PharmaSUG 2016 Online Conference Proceedings*, Denver, CO

CONTACT INFORMATION

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

Pinky Anandani Dutta
Inclin, Inc
2655 Campus Dr Ste 100
San Mateo, CA 94403
panandani@inclin.com
www.inclin.com