

The Define.xml Designer 2022

User Manual for designing and developing Define.xml files

Last update: 2022-09-07

Introduction

This user manual describes the features for designing and developing define.xml 2.0 and 2.1 files using the "Define.xml Designer" software package. For information about designing ODM Study design files, please see the separate user manual for the "ODM Designer" software.

Background

CDISC SDTM, SEND and ADaM electronic submissions to regulatory authorities such as the US FDA, the Japanese PMDA and Chinese NMPA, require a "define.xml" file to accompany the submission files. The define.xml file contains the metadata (data about data) of the information in the submission files. As such, a well designed define.xml file is of utmost importance for explaining the reviewers about what the data is about and how it was generated.

The define.xml file is an XML file that is as well human-readable as well as machine-readable. The human readability is ensured by a so called "stylesheet" (XSLT stylesheet), that transforms the XML into information that is interpretable by any modern browser (HTML+CSS).

It is however essentially the define.xml file itself (i.e. the XML) that carries the information - what is seen in the browser is just a human understandable presentation of that data¹.

Although XML is very easy to learn, many people desire to use a good software tool to design or develop a define.xml file describing the contents of their electronic submission, which does not require them to write XML. The "Define.xml Designer 2022" is such a tool.

Ideally, the define.xml file is already developed before study start or in the early stages of a clinical study. The reason for this is that the most important goal of a (commercial) clinical study is to get a marketing authorization for the drug or therapy that is tested. So it is of utmost importance to already test whether the during the study captured data can be transformed to SDTM, SEND or ADaM (for the latter from SDTM). If it is found that this is not possible at the end of the study, then the whole submission is endangered.

Many pharma companies already develop their mapping to SDTM together with the corresponding define.xml even before study start. The ideal tool to do so is XML4Pharma's [SDTM-ETL](#) software, allowing s users to develop the mapping to SDTM starting from an ODM file with the study design. The SDTM-ETL software is also used a lot for SEND submissions. In that case, when no ODM file is available, it can be generated from "flat" files such as CSV files using the "[ODM Generator](#)" software.

There are however cases where such a synchronized (SDTM + define.xml) approach is not possible. This is especially the case for legacy studies where the SDTM files have already been generated and where the define.xml must be generated "post-SDTM". This often also applies to SEND and ADaM files.

¹A stylesheet could even be used to manipulate the information as seen in the browser. Therefore, the "truth" is what is in the define.xml file, not what is seen in the browser.

Another case is when the user wants to start from a specific SDTM-IG and corresponding define.xml template file, and then add/remove variables and add all other information, adapting the define.xml to the needs of a specific study or submission. For example, many sponsors already do develop a define.xml as a specification to their third party service provider, containing the information how the sponsor wants to obtain the SDTM data files for a specific study at the end of the study. A third use case is that the user obtains a define.xml that is incomplete, is not correct, or does not well explain the submission to the regulatory reviewer.

These use cases are covered by the "Define.xml Designer 2022". This includes the almost-automated generation of a define.xml file starting from a set of SAS Transport 5 ("XPT") files.

This manual describes how the "Define.xml Designer 2022" is used for these three and other use cases. No knowledge of XML is required. It is however recommended that the user has a basic understanding of the goals and structure of define.xml. This can be obtained by reading the define.xml specification (a copy is included with the distribution) and/or attending a one-day CDISC define.xml training. The latter will make you a Define-XML expert, allowing you (when using this software) to develop jewel define.xml files that very well explain the regulatory reviewers what your data is about, how it was generated, and how it is organized.

It is recommended to not use "black box" tools, i.e. tools that let you start e.g. from an Excel worksheet, and "magically" generate a define.xml file from your information in the worksheet. This kind of tools force you to re-generate the define.xml file each time something is not perfect in the result (as displayed by the browser!). Furthermore, they give you no insight into the define.xml that is generated, so essentially, you are not understanding what you are doing. This will usually lead to disaster...

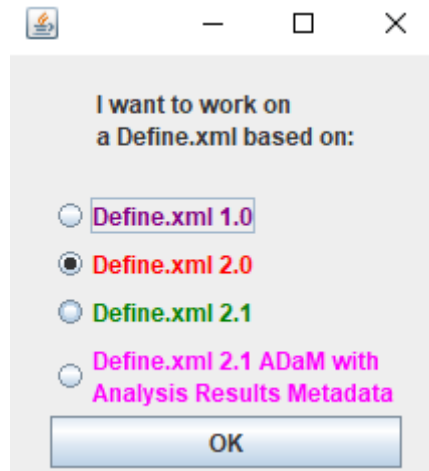
Table of Contents

Introduction.....	1
Background.....	1
Starting the software and main principles.....	4
Creating a new define.xml	13
Creating a define.xml starting from an SDTM/SEND/ADaM template.....	15
Creating a define.xml starting from an existing set of SAS-XPT files.....	26
Ordering the dataset definitions	44
Adding and subsetting CDISC Controlled Terminology	46
Adding and subsetting codelists from SAS-XPT files.....	59
Transforming an "EnumeratedItem" CodeList into a "CodeListItem" CodeList	67
Displaying your define.xml as HTML in a browser using a stylesheet.....	71
Creating and populating Sponsor-defined Codelists.....	72
Cleaning up your define.xml.....	79
Saving your work to file	81
Creating ValueLists.....	84
Automatically assigning WhereClauses to ValueList entries	98
Annotated CRF and Supplemental Documents	103
Assigning valuelists to SDTM/SEND/ADaM variables.....	107
Assigning an Origin to variables.....	110
Retrieving PDF Page Numbers from the annotated CRF	117

Starting the software and main principles

ODM Designer

If you are a windows user, double-click the "Define-XML_Designer.bat" icon or entry in your file explorer. If you are a Linux or MacIntosh user, use the Define-XML_Designer.sh" entry. First, license information will be displayed, followed by the dialog:

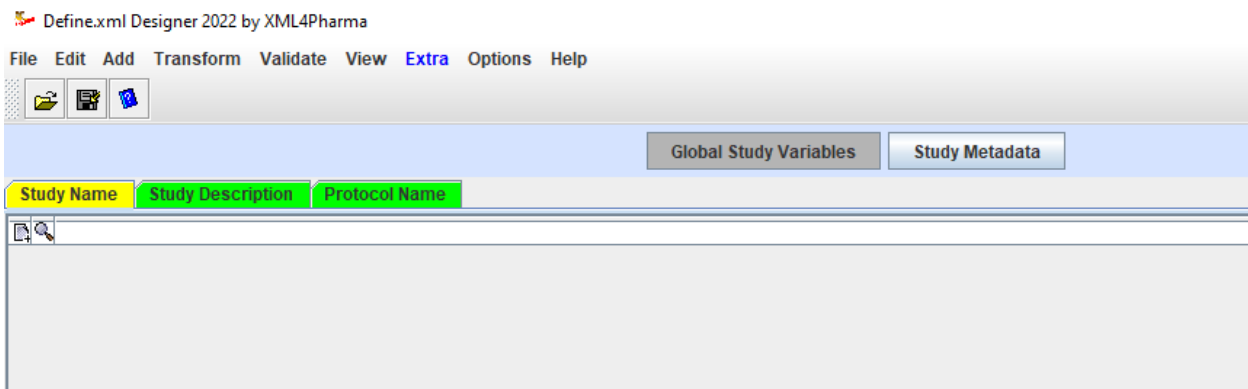


The software still supports define.xml 1.0, but it is **strongly recommended** to use define.xml 2.0 or 2.1 as this allows you to much better explain the contents and structure of your submission to the regulatory reviewers. Define.xml 1.0 is also not accepted by the FDA anymore.

Also, most of the new features have not been implemented for Define-XML 1.0 anymore.

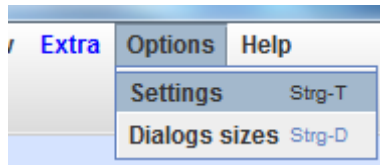
If you also want to implement the "Analysis Result Metadata" extension, select the raibutton "Define.xml 2.1 ADaM with Analysis Results Metadata". This will work with Define-XML 2.1.

After having clicked the "OK" button, the software starts reading the corresponding Define-XML XML-Schema, and generates the graphical user interface elements automatically from the XML-Schema. This e.g. results in:

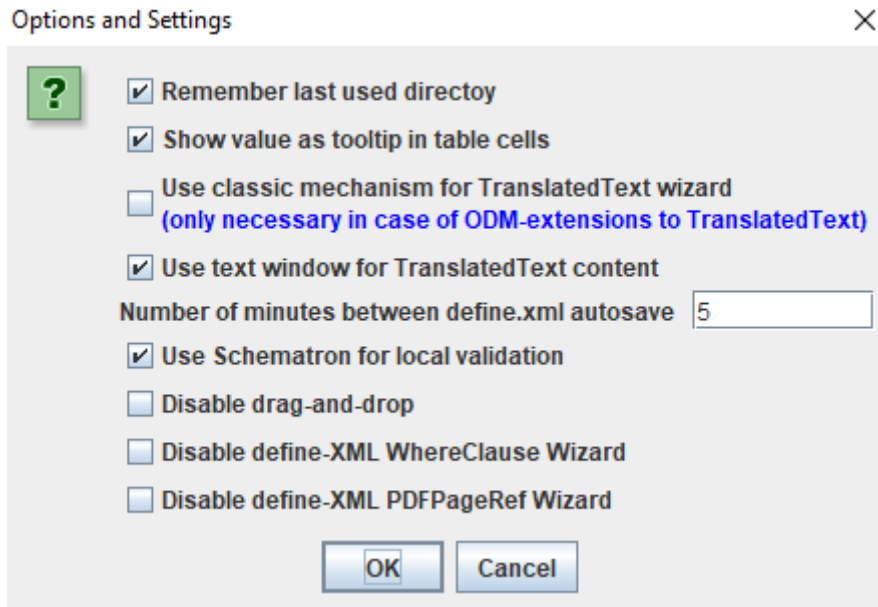


The "Study Name", "Study Description" and "Protocol Name" already allow you to add some mandatory information. Whereas the "Study Name" and "Study Description" items are self-explaining, the "Protocol Name" is expected the contain the "*The sponsor's internal name assigned to the Study*" which usually is the title of the protocol.

Let us however first have a look at the "Options". To do so, use the menu "Options - Settings".

















This opens a new dialog:



We will now discuss the most important options and settings.

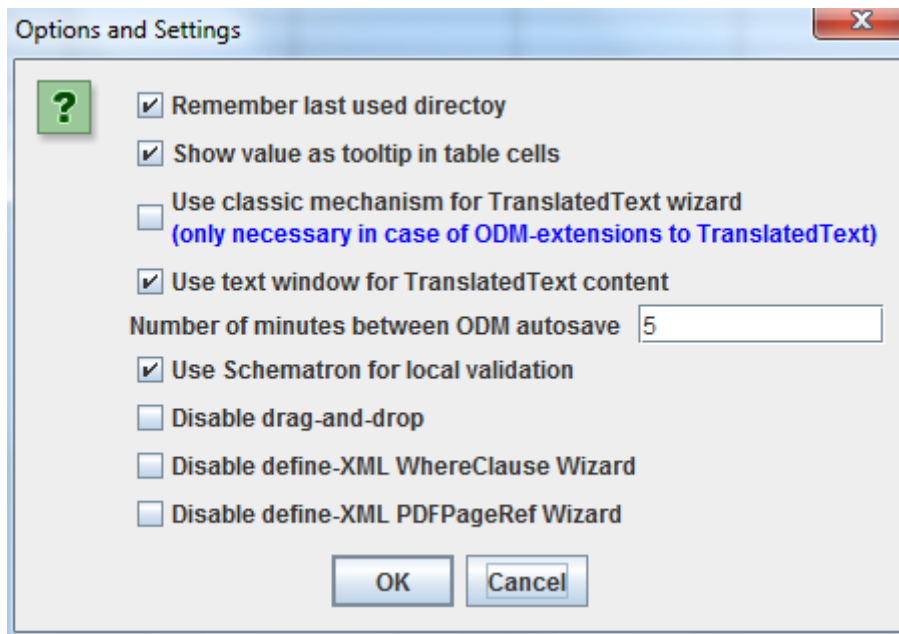
- Usually you will want to have the software remember the last used import (file read) or export (file write) directory. You can disable this feature by unchecking the checkbox "Remember last used directory"
- The feature "Show value as tooltip in table cells" is very useful when there are many columns, and you don't want to start dragging column borders to see the complete value. The feature is enabled by default, but can be switched off.
- The wizard for "TranslatedText" (i.e. for support of multiple languages) can be disabled using the checkbox "Use classic mechanism for TranslatedText wizard". You probably however want to use the wizard...
- When adding text that goes within "TranslatedText", a separate small text window shows up for adding the text. If the user does not want this, but wants to add the text directly in the table cell, then the checkbox "Use text window for TranslatedText content" should be unchecked.
- The following option is a very important one: the software will save the full study or define.xml design to an define.xml or ODM file each 5 minutes in the directory "autosave". This allows you to pick up an earlier version of your design. The files in this directory use a naming convention for allowing the user to see when it has been created. For example:

Name	Änderungsdatum	Typ	Größe
 ODM_2022_8_30_19-4-15.xml	30.08.2022 19:04	XML-Datei	6.858 KB
 ODM_2022_8_30_18-59-15.xml	30.08.2022 18:59	XML-Datei	6.858 KB
 ODM_2022_8_30_18-54-15.xml	30.08.2022 18:54	XML-Datei	6.858 KB
 ODM_2022_8_30_18-49-15.xml	30.08.2022 18:49	XML-Datei	6.858 KB
 ODM_2022_8_30_17-33-40.xml	30.08.2022 17:33	XML-Datei	6.858 KB
 ODM_2022_8_30_17-28-40.xml	30.08.2022 17:28	XML-Datei	6.858 KB
 ODM_2022_8_30_17-23-40.xml	30.08.2022 17:23	XML-Datei	6.858 KB
 ODM_2022_8_30_17-18-40.xml	30.08.2022 17:18	XML-Datei	6.858 KB
 ODM_2022_8_30_17-13-40.xml	30.08.2022 17:13	XML-Datei	6.858 KB
 ODM_2022_8_30_17-8-40.xml	30.08.2022 17:08	XML-Datei	6.858 KB
 ODM_2022_8_30_17-3-40.xml	30.08.2022 17:03	XML-Datei	6.858 KB
 ODM_2022_8_30_16-58-40.xml	30.08.2022 16:58	XML-Datei	6.858 KB
 ODM_2022_8_30_16-53-40.xml	30.08.2022 16:53	XML-Datei	6.858 KB
 ODM_2022_8_30_16-48-40.xml	30.08.2022 16:48	XML-Datei	6.858 KB

The time periods between an "autosave" can be changed by using the textfield after "Number of minutes between define.xml autosave". Only positive numbers can be entered. Setting the value to "0" means that no autosaving is done.

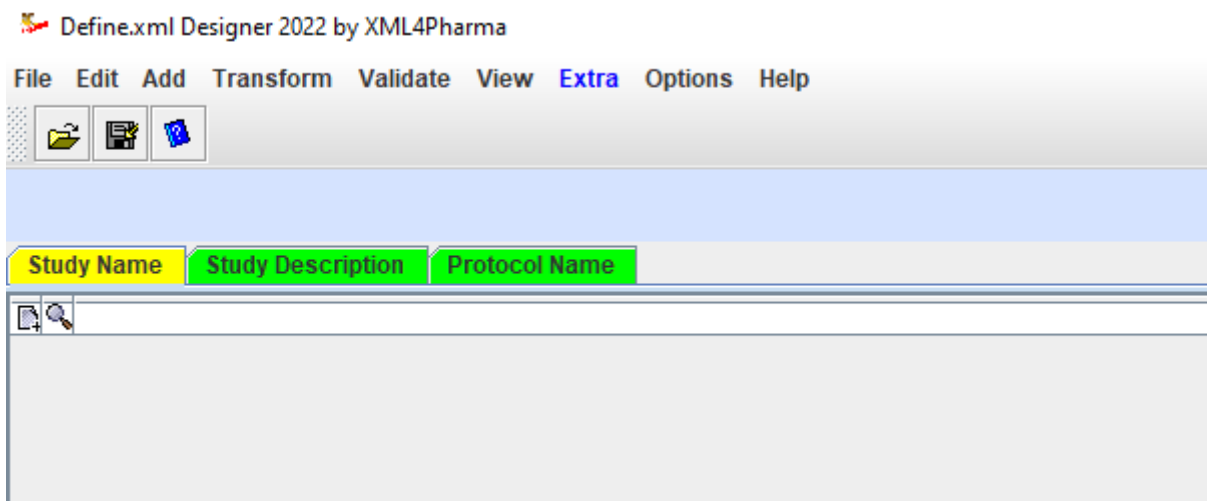
Please regularly have a look at the "autosave" directory and clean it up.

- The designer uses schematron technology during validation. This is the methodology of choice for validating XML files. The software comes with a set of schematron files for as well ODM as for define.xml. If you want to limit validation to XML-Schema validation and hardcoded rules when doing "local" validation (see further), uncheck the checkbox "Use schematron for local validation". You will also be able to make a choice when you start a "local" validation (explained further on).
- Most people like to use "drag and drop". There are however some users that don't. If you are one of these users, check the checkbox "disable drag-and-drop". The latter mechanism is then replaced by a "select and click" mechanism.
- The last two checkboxes enable you to disable the wizards for the define.xml "WhereClause" and "PDFPageRef". If you disable these, you will need to enter some information manually (i.e. by typing). This requires a very good knowledge of define.xml. Use with care!



Let us now go back to the main window that is displayed after the XML-Schema has been read and the graphical user elements have been generated.

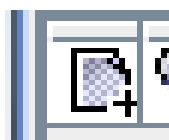
For example, for "Global Study Variables", you will find 3 tabs:



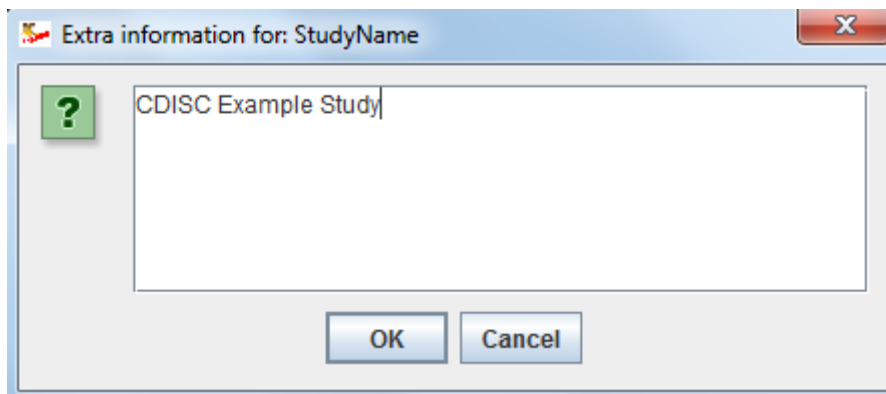
The currently selected tab is always colored yellow, a green tab means that it represents information that must be entered, i.e. is mandatory.

Usually, a table is displayed below the tab name. In this case, there is only 1 row, as only 1 "StudyName" element is allowed in the define.xml file.

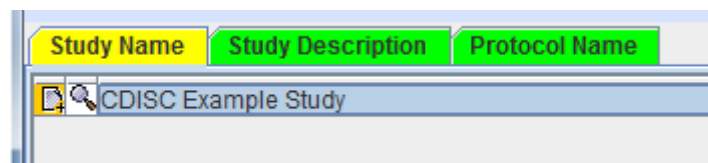
"Study Name" is a "text content only" element in XML. In order to enter that text, click the "+" icon (the most left one):



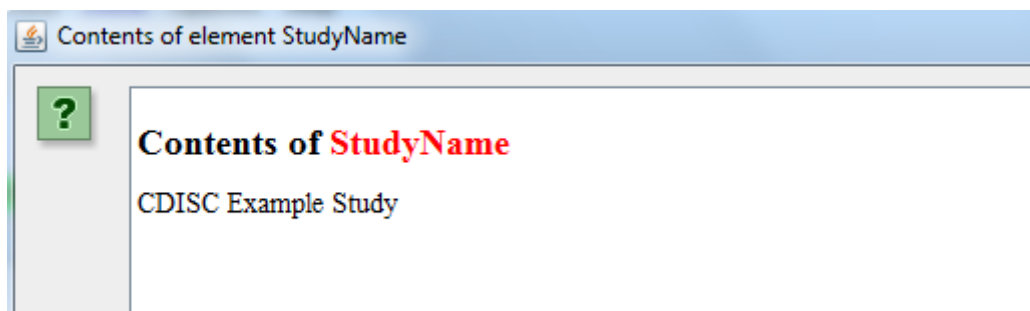
This will result in a new window being displayed, allowing you to enter the study name. For example:



After clicking "OK" (or hitting the Enter" key on the keyboard), the text also is displayed in the table:



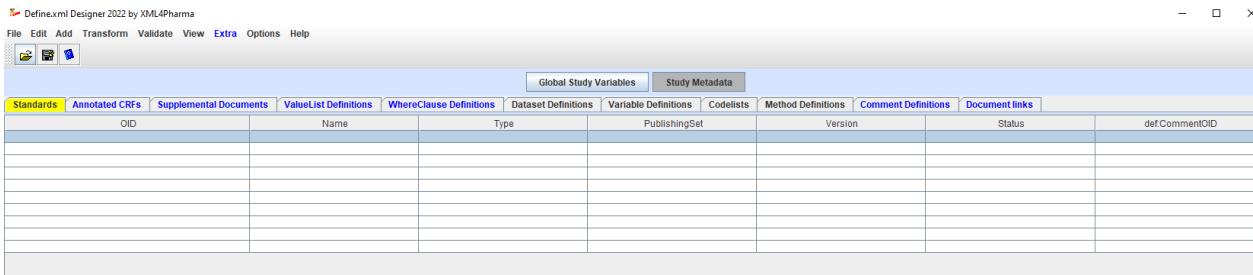
If the text is longer than the cell in the row allows, you can inspect the text by clicking on the "magnifying glass" icon (second icon). This will show a dialog window:



In more complicated cases, such as when the ODM or define.xml element has attributes and/or child elements, much more information will be displayed in this dialog window.

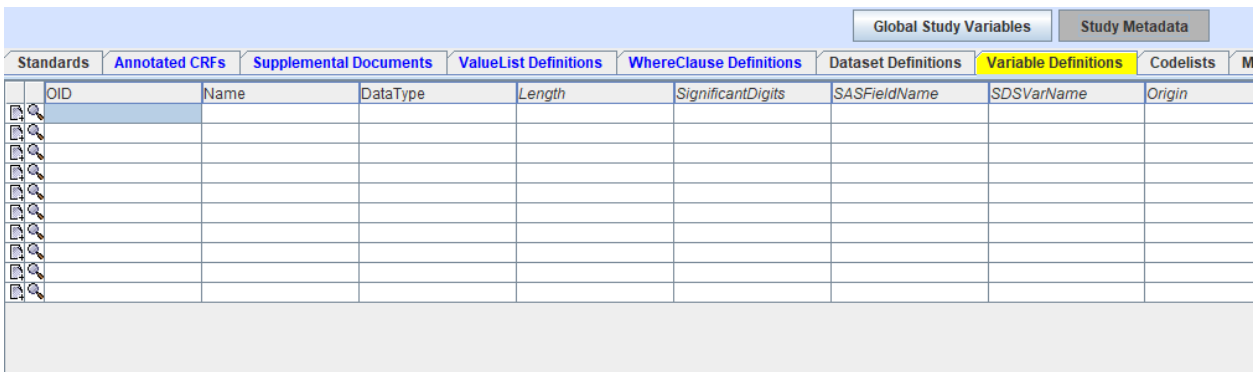
You can now add similar information for the tabs "Study Description" and "Protocol Name".

Let us now see how this works for an SDTM or SEND or ADaM variable. These, and all other metadata of the submission can be added/edited after clicking the "Study Metadata" button. This results in:



(here for Define-XML 2.1).

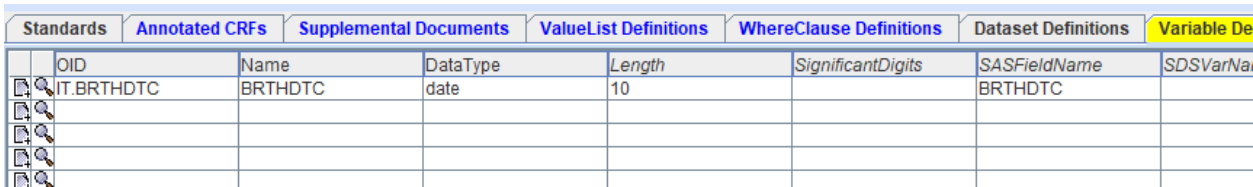
To add an SDTM/SEND/ADaM variable, click the tab "Variable Definitions". This then displays the following panel:



Initially, 10 rows are generated, but you can add additional rows (and you probably will) by clicking the "Add Row" button.

In the table, each row corresponds to a single SDTM/SEND/ADaM variable, and each column to an attribute for it. Adding child element information will be done by clicking the "+" icon on the left of each row, and inspecting all the contents can be done by clicking the "magnifying glass" icon (second from the left).

For a variable "BRTHDTC" (birthdate) in DM, a variable declaration will typically look like (for define.xml 2.1):



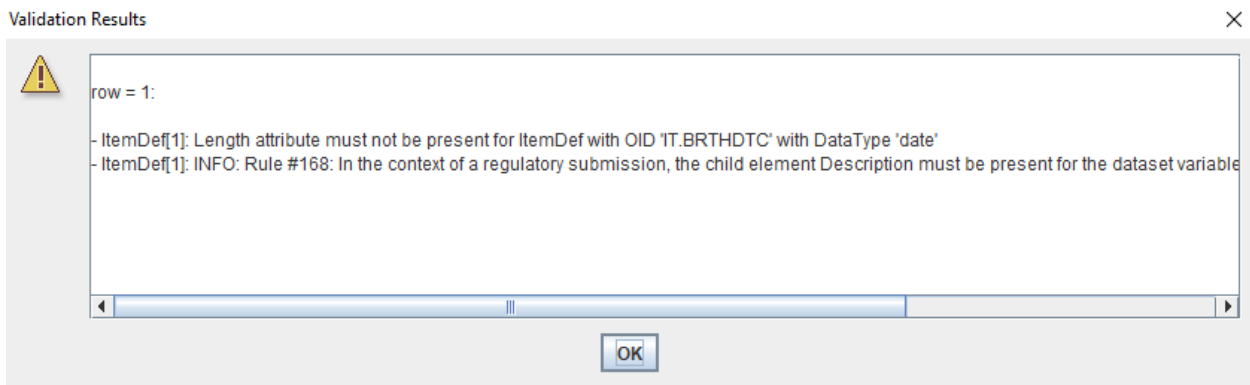
Remark that "SASFieldName" is optional, and does not make sense when one does not use SAS to generate the datasets.

We also deliberately added some invalid information, as a value for "Length" is not allowed when the datatype is "date".

The "DataType" value is enumerated, so when clicking that cell, a list appears from which one needs to select a single one from:

DataType	Length
date	
integer	
float	
date	
datetime	
time	
text	
string	
double	

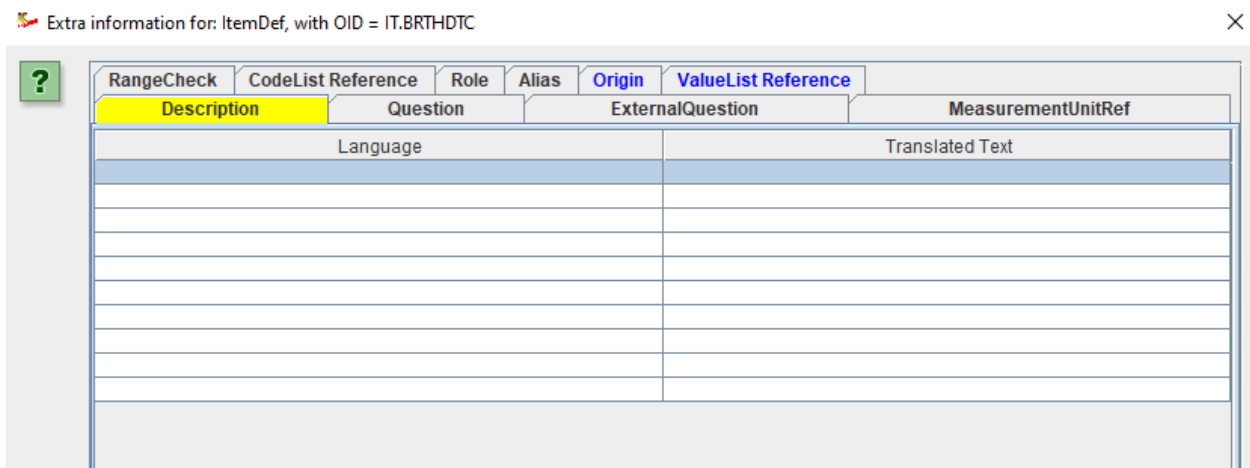
The "Validate" button (in the lower part of the panel, on the right) can then be used to validate the local contents. In our case this will lead to:



Two violations² of the define.xml standard (2.1) have been found:

- No "Description" child element was found. It represents the "label" of the variable
- The "Length" attribute is not allowed when the DataType has the value "date"

We just "empty" the "Length" cell by selecting its contents and using the "delete" key on the keyboard. The "Description", which is a child element of "ItemDef" (representing the SDTM/SEND/ADaM variable label) can then be added by using the "+" icon:



As one sees, this table has several rows, as a description in several languages can be added. For a

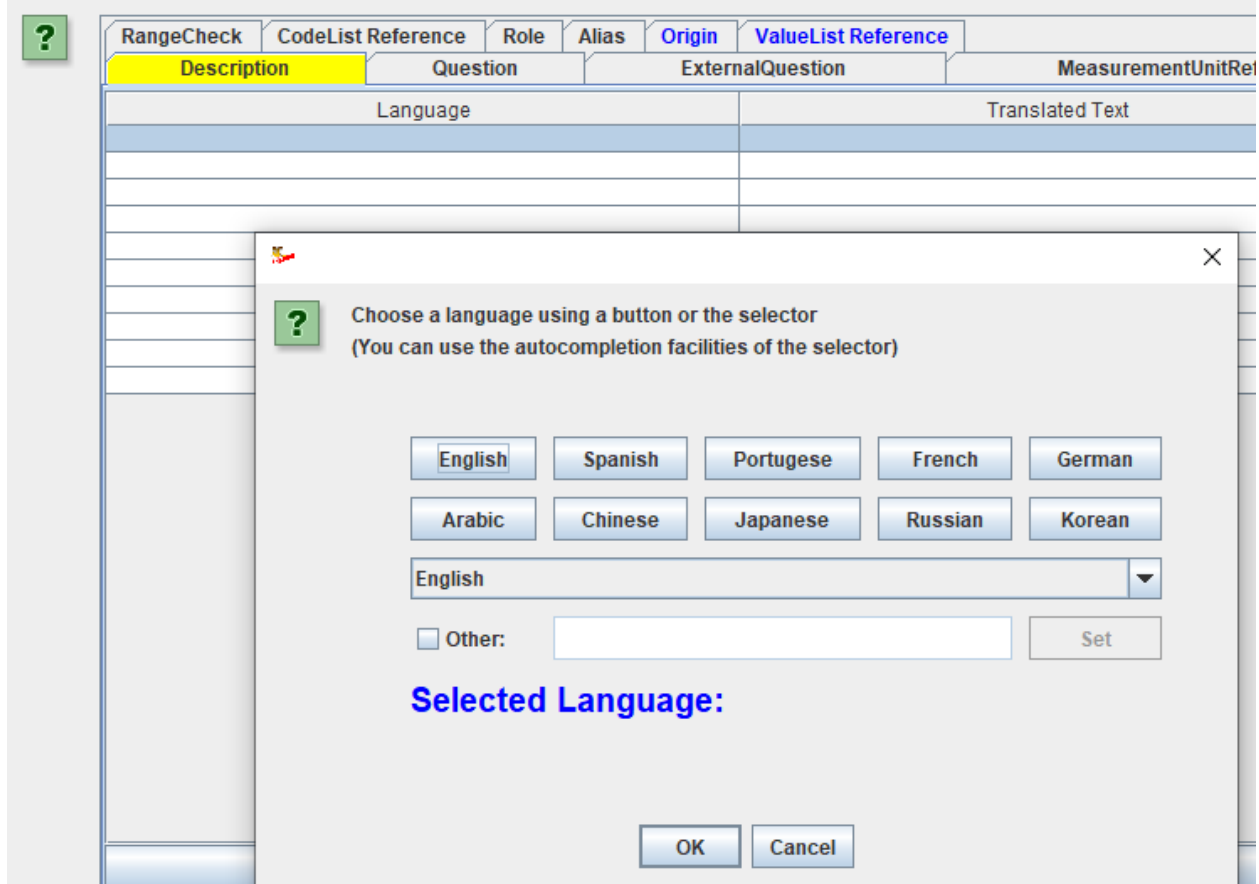
²Remark that we do not speak of "Errors" and "Warnings" as such an assignment is completely arbitrary. We will only speak about an "Error" when it is an XML-Schema error.

submission to the FDA, at least an English description must be provided, which must match³ the "label" from the SDTM-IG or SEND-IG or ADaM-IG which is "Date/Time of Birth" in this case.

Remark that this feature also allows to generate a define.xml with non-English labels, which may become a requirement by e.g. the PMDA and NMPA in future.

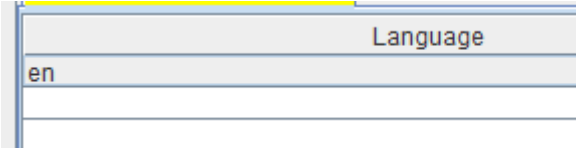
We first want to add the language. Clicking in the first "Language" cell pops up a dialog:

Extra information for: ItemDef, with OID = IT.BRTHDTC



Language	Translated Text

from we click the "English" button and then click OK⁴. Resulting in:

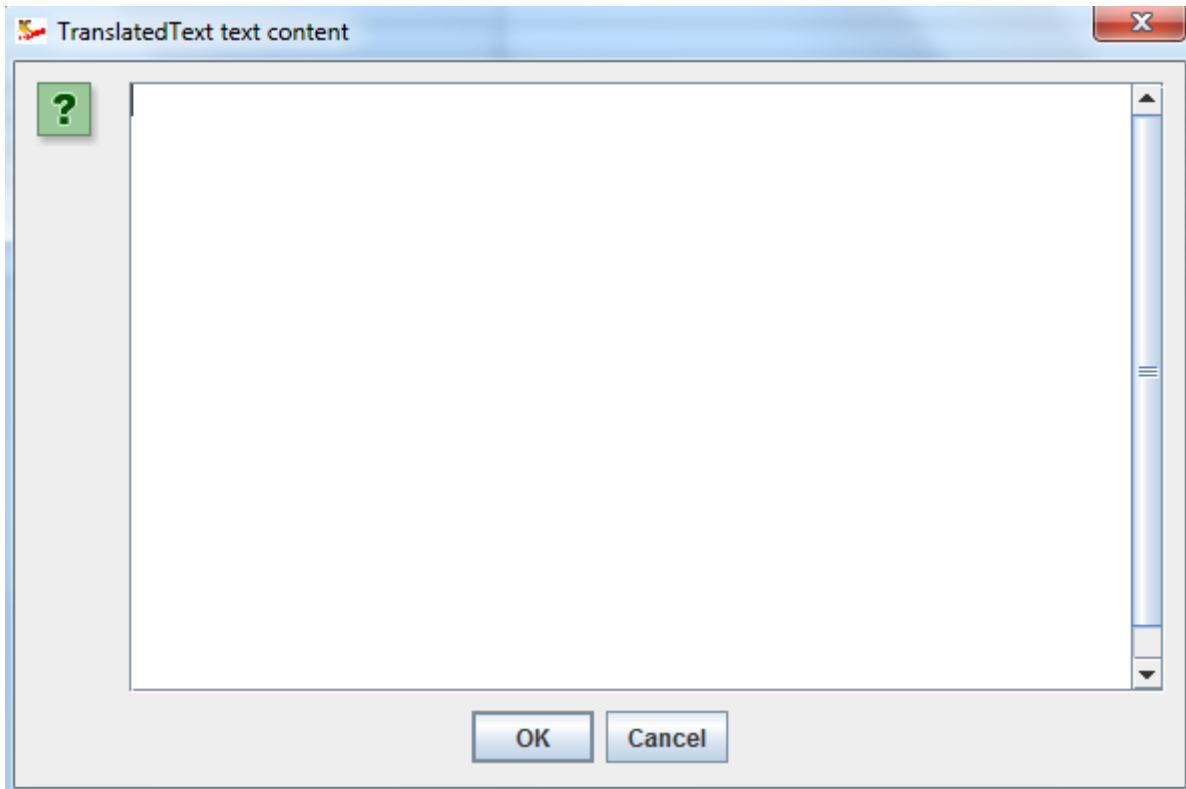


Language
en

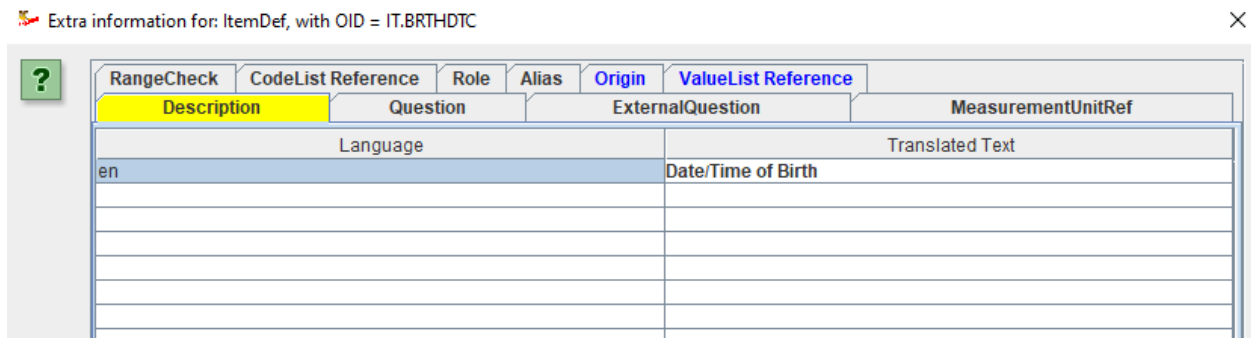
Then clicking the cell under "TranslatedText" for the same row displays a text entry dialog:

³In future, we intend an option that automatically looks up the variable label for a given variable name and SDS version, either locally (from a file that comes with the software), by a RESTful web service, or by a query to the [CDISC Library](#).

⁴This wizard was created so that users do not need to know the ISO language codes.



where we add the "SDTM label", in this case "Date/Time of Birth". After clicking "OK" leading to:

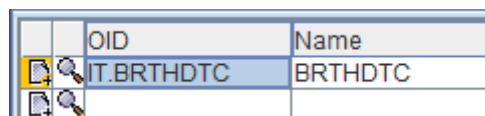


We could now add the description in other languages⁵, but this is usually not necessary in the case of submissions to the FDA.

By now, you will probably have asked yourself the question: "*will I really have to add all my variable definitions manually for my SDTM/SEND/ADaM submission?*".

The answer is "No", as we will later learn how to start from an SDTMIG, SENDIG or ADAMIG template to do all this for you.

After clicking the "OK" button:

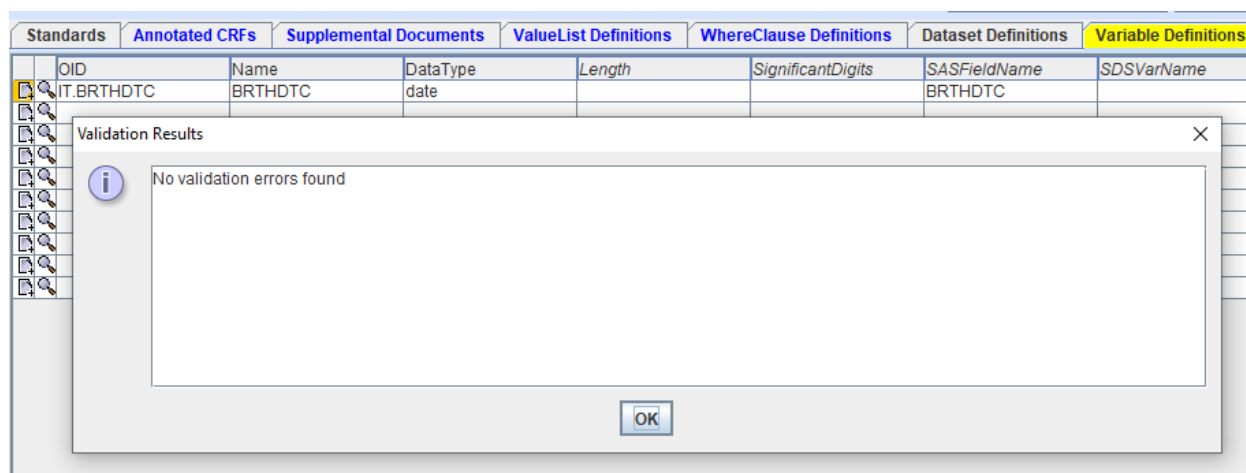


we see that the "+" icon is now with a yellow background color, meaning that additional

⁵This is the reason why "Description" is not a field in the table for "Variable Definition", as it would then only be possible to add a description in only one language.

information has been added. The background color may also become red after validation, meaning that the additionally added information does not conform to the Define-XML standard.

Clicking the "validate" button again leads to:



Essentially, this does not mean yet that everything is fully compliant. For example, at this moment, the software does not know yet whether the variable "BRTHDTC" is a value-level variable or not, so that for example the rule that the "Origin" must be declared either on the variable level or at the value-level, cannot be checked yet.

So, the essentials again:

- Variable attributes are added by clicking the appropriate cell. In case the value is enumerated, a "selection list" will be displayed. In some cases, a wizard will pop up.
- Additional information that is stored in child elements are added by clicking the "+" icon, selecting a tab, and adding the information just like in the main panel. In some cases, one will want to "drill down" into deeper levels and sub-dialogs. Examples will be provided later on.
- "Local" validation can be done using the "validate" button. A short report will then be displayed. In case a mandatory attribute is not populated, the field will be colored red.
- In case additional information has been added, the "+" icon is colored orange. It can turn to red after validation in case the added information violates the Define-XML standard

Creating a new define.xml

Everything has a beginning...

Though designing define.xml files from scratch is easily possible, one may in many cases either start from either an SDTM or SEND or even ADaM template⁶, or from a set of existing SAS-XPT SDTM, SEND or ADaM files. The ideal is of course to have a process in place to develop the mapping between operational data and SDTM/SEND data that is synchronized with the development of the define.xml such as in the [SDTM-ETL software](#).

In order to start the development of a new define.xml either from an SDTM or SEND template, or

⁶ In the case of ADaM, only tables for the basic datasets, such as ADSL, will be generated. The user can then add additional dataset definitions and variables in the usual way.

from a set of existing SAS-XPT files with SDTM or SEND data, use the menu "File - New define.xml". The following dialog is presented:

New Study Metadata [X]

Define-XML version: 2.1.0

I want to start from a CDISC SDTM/SEND/ADaM template

- define_template_ADaMIG_1_3.xml
- define_template_SDTMIG_3.1.2_SDTM_1.2.xml
- define_template_SDTMIG_3.1.2_SDTM_1.2_oncology_draft.xml
- define_template_SDTMIG_3.1.2_SDTM_1.2_PGx_new.xml
- define_template_SDTMIG_3.1.3_Med_Devices.xml
- define_template_SDTMIG_3.1.3_SDTM_1.3.xml
- define_template_SDTMIG_3.1.3_SDTM_1.3_Non_Subject_Data.xml
- define_template_SDTMIG_3.2_AssociatedPersons.xml

I want to start from a set of SAS-XPT files

SDTM [v] Browse SAS-XPT

I want to load by CDISC published Controlled Terminology

- ADaM_Terminology_2021-12-17.xml
- ADaM_Terminology_2022-06-24.xml
- COA_Terminology_2014-12-19.xml
- COA_Terminology_2015-03-27.xml
- QRS_Terminology_2015-06-26.xml
- QRS_Terminology_2015-09-25.xml
- QS-FT_Terminology_2014-06-26.xml
- QS-FT_Terminology_2014-09-26.xml

Generate Define-XML Variable DataType, Length and SignificantDigits from XPT content

Try to create subset CodeLists from XPT content and selected Controlled Terminology

Try to create sponsor-defined CodeLists from definitions in a 'sponsorcodelistvariables.dat' file

Try to create Valuelists for Supplemental Qualifier datasets from XPT content

Try to create Valuelists from definitions in a 'valuelistvariables.dat' file

Study OID (required)

Study Name (required)

Study Description (required)

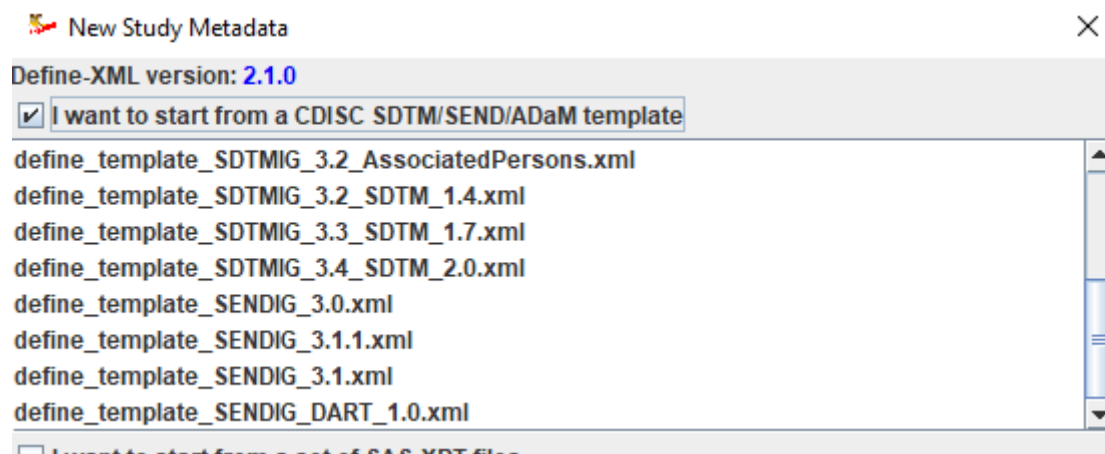
Protocol Name (required)

OK Cancel

Either click the checkbox "I want to start from a CDISC SDTM/SEND/ADaM template" or the checkbox "I want to start from a set of SAS-XPT files". Also create a study OID (this is the Define-XML identifier for the study, so one can use a "mnemonic" like "CES" for "CDISC Example Study", or use an identifier that is used within your company), a study short name, a study description and the protocol name or title.

Creating a define.xml starting from an SDTM/SEND/ADaM template

In case you checked the checkbox "I want to start from a CDISC SDTM/SEND/ADaM template", the following fields become visible and enabled:



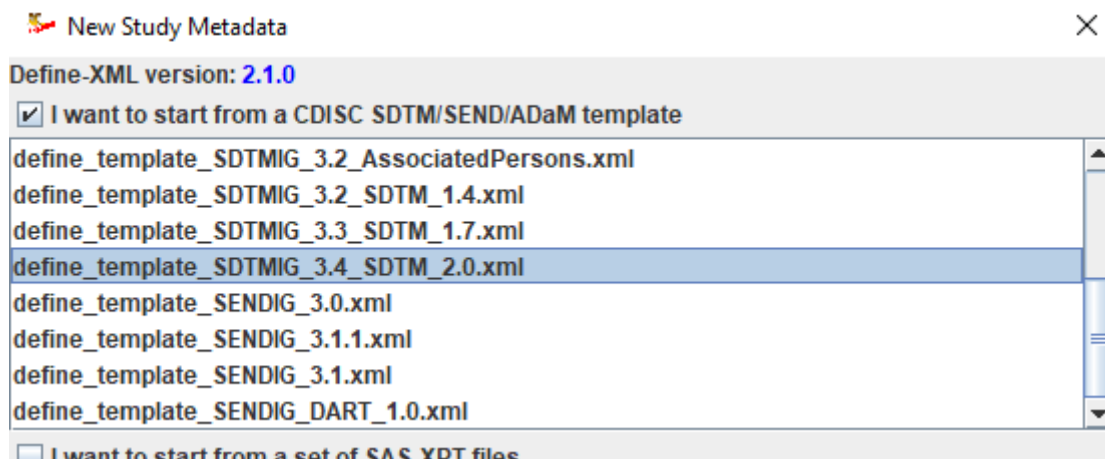
One can then select a template from the list. It contains all necessary information from the SDTM-IG or SEND-IG. One can later than still merge with another template. For example, if one has selected "define_template_SDTMIG_3.2.xml", then one can later still merge with domains from the "define_template_SDTM_3.2_AssociatedPersons" or even with templates for newly developed SDTM domains published by CDISC. It is also very easy to develop own templates and then merge them with CDISC-IG templates.

All these templates come as files with the software, and the content of the directory where they reside is inspected by the software. This also means that once CDISC publishes a new Implementation Guide, we will deliver the template file or files for it, and adding it to the directory will suffice to be able to use them - no software update will be necessary.

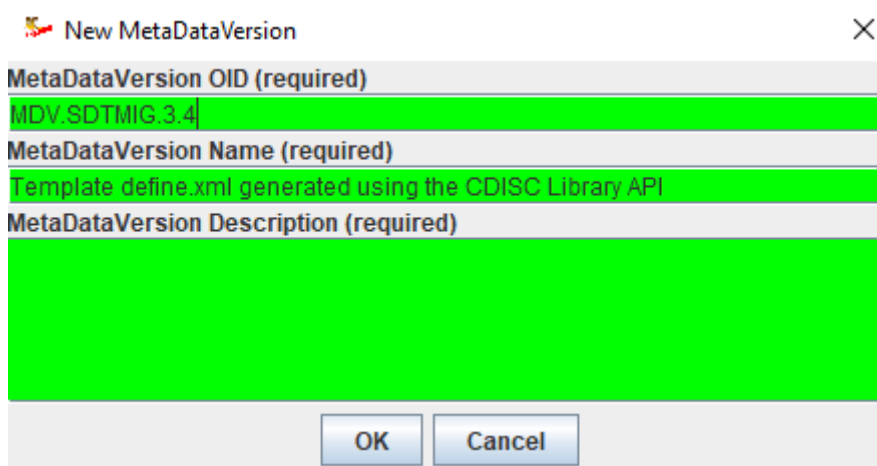
Suppose we would like to use the latest SDTM version, so we select "define_template_SDTMIG_3.4.xml". It is then also extremely useful to already add the latest by CDISC published controlled terminology⁷. In order to do so, check the checkbox "I want to load by CDISC published Controlled Terminology" and then select a version (probably the latest) from the list. Also here, when new controlled terminology is published by CDISC, we will provide it as a file, and adding it to the directory where the controlled terminology files reside will be sufficient to let appear it in the list in the software.

Additional CDISC controlled terminology can also be loaded separately (see further on) and merged later.

⁷In earlier versions of SDTM, the controlled terminology was coupled to the SDTM version. This is no longer the case, and one can select any more recent version of the controlled terminology. It is usually advised to use the latest published controlled terminology, and then to stick to that version during the generation of the SDTM and the define.xml.

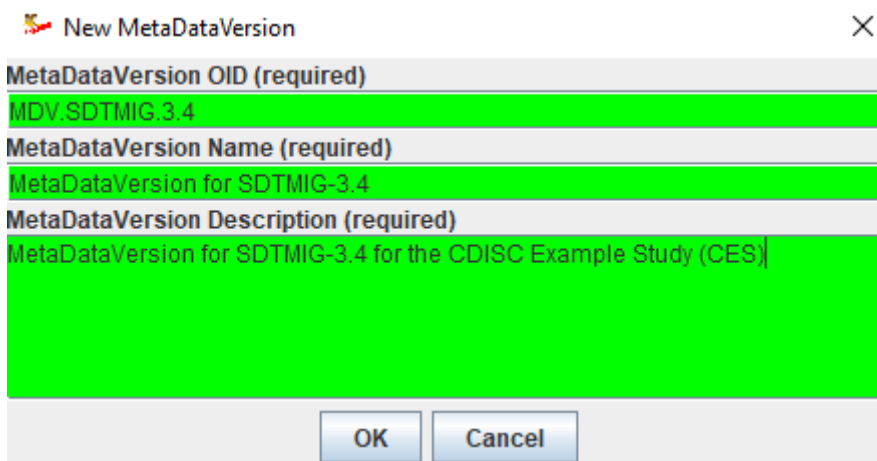


After clicking "OK", the system will start loading the templates and the controlled terminology, and will ask for additional information:



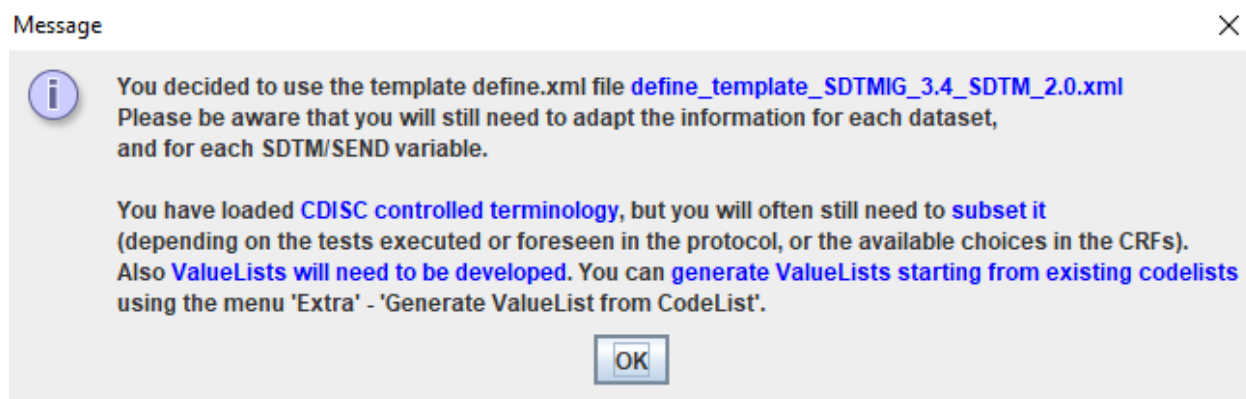
A proposal is made for the "MetaDataVersion OID" and the "MetaDataVersion Name". These are arbitrary - they are currently not really used as only one "MetaDataVersion" is currently allowed in a define.xml⁸.

For the "MetaDataVersion Description", we provide useful information, e.g.:



⁸This might change in future when the regulatory authorities allow to have updates on define.xml content.

After clicking OK, the templates and CDISC controlled terminology are loaded. The following message appears:



Remark that the whole CDISC controlled terminology has been loaded, and that you will need to subset many of the codelists. For example, the "LBTESTCD" contains over different 1,500 codes for lab tests, but you will probably have considerably less in your study. We will later learn how to subset a codelist.

Also, you will later have to develop the necessary "value-lists" (value-level metadata). The software has a wizard for this, allowing you to generate a "ValueList" from an existing codelist.

After clicking OK, and navigating to the "Variable Definitions", one gets:

Global Study Variables							Study Metadata	
Standards	Annotated CRFs	Supplemental Documents	ValueList Definitions	WhereClause Definitions	Dataset Definitions	Variable Definitions	Cod	
OID	Name	Data Type	Length	Significant Digits	SASFieldName	SDSVarName	Origin	
MH.MHENTPT	MHENTPT	text	80		MHENTPT			
BS.BSSEQ	BSSEQ	integer	8		BSSEQ			
BS.BSGRPID	BSGRPID	text	80		BSGRPID			
BS.BSREFID	BSREFID	text	80		BSREFID			
BS.BSSPID	BSSPID	text	80		BSSPID			
BS.BSTESTCD	BSTESTCD	text	8		BSTESTCD			
BS.BSTEST	BSTEST	text	40		BSTEST			
BS.BSCAT	BSCAT	text	80		BSCAT			
BS.BSSCAT	BSSCAT	text	80		BSSCAT			
BS.BSORRES	BSORRES	text	80		BSORRES			
BS.BSORRESU	BSORRESU	text	80		BSORRESU			
BS.BSSTRESC	BSSTRESC	text	80		BSSTRESC			
BS.BSSTRESN	BSSTRESN	float	8	2	BSSTRESN			
BS.BSSTRESU	BSSTRESU	text	80		BSSTRESU			
BS.BSSTAT	BSSTAT	text	8		BSSTAT			
BS.BSREASND	BSREASND	text	80		BSREASND			
BS.BSNAM	BSNAM	text	80		BSNAM			
BS.BSSPEC	BSSPEC	text	80		BSSPEC			
BS.BSANTREG	BSANTREG	text	80		BSANTREG			
BS.BSPCCND	BSPCCND	text	80		BSPCCND			
BS.BSMETHOD	BSMETHOD	text	80		BSMETHOD			
BS.BSBLFL	BSBLFL	text	1		BSBLFL			
BS.BSDTC	BSDTC	datetime			BSDTC			
BS.BSDY	BSDY	integer	8		BSDY			
BS.BSTPT	BSTPT	text	80		BSTPT			
BS.BSTPTNUM	BSTPTNUM	text	80		BSTPTNUM			
BS.BSELTM	BSELTM	text	80		BSELTM			
BS.BSTPTREF	BSTPTREF	text	80		BSTPTREF			
BS.BSRFTDTC	BSRFTDTC	datetime			BSRFTDTC			
CP.CPSEQ	CPSEQ	integer	8		CPSEQ			
CP.CPGRPID	CPGRPID	text	80		CPGRPID			
CP.CPREFID	CPREFID	text	80		CPREFID			
CP.CPSPID	CPSPID	text	80		CPSPID			

One sees that the table has been filled with information, and that for each variable, a datatype and a maximum length has been proposed. You will still have to adapt this information for your own study. For example, for RFSTDTC and similar variables (RFENDTC, RFXSTDTC, ...) you only have collected the date without a time part, you should change the datatype "datetime" into "date".

This can easily be achieved by clicking in the cell, and select "date" instead of "datetime". For example:

RFSTDTC	datetime	
RFENDTC	datetime	
RFXSTDTC	date	
RFXENDTC	datetime	
RFICDTC	time	
RFPENDTC	text	
DTHDTC	string	
DTHFL	double	2
SITEID	URI	80
INVID	URI	80
INVNAM	boolean	80

You will also need to adapt the value for "Length" to the maximum length the data can have for that variable in your SAS-XPT files. So if in your study "USUBJID" never has more than 20 characters, you should change the value of "60" into "20", and take care that in your SAS-XPT file the field length is exactly 20⁹. If you start developing your define.xml before the study is being finalized (which is a best practice), and do not know all the maximal lengths, no worry, you can still leave the "Length" with the proposed value, and do an automated adaption from the SAS-XPT file contents when you have all your SAS-XPT files available (see further on).

Remark that the "Length" attribute is only there due to the regulatory requirement to submit using SAS Transport, and does not have a meaning when using a modern transport format.

One also sees that all rows in the table have the "+" icon colored yellow, meaning that additional information is available. For example, when clicking on the "+" icon, the following dialog is displayed:

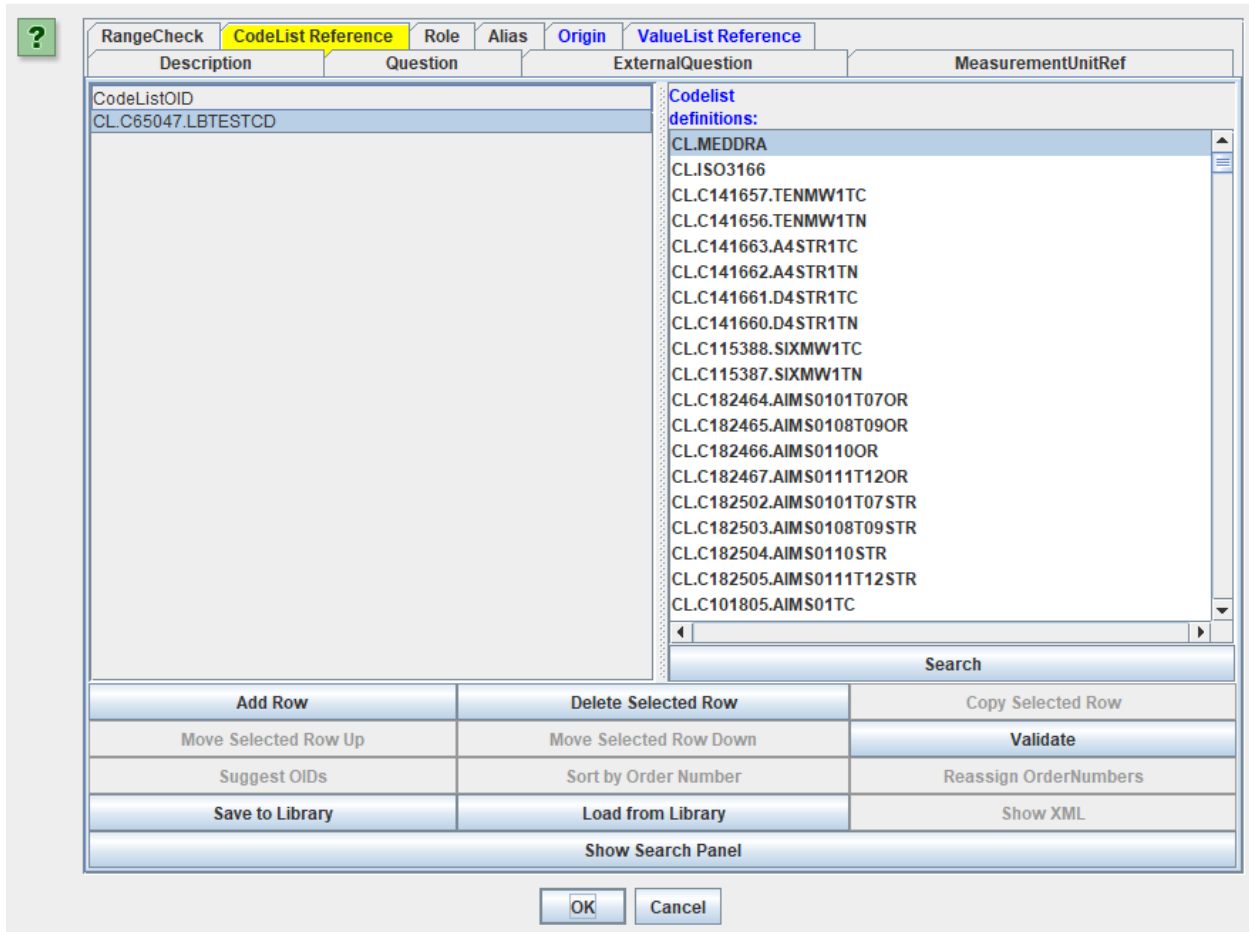
Extra information for: ItemDef, with OID = USUBJID

RangeCheck	CodeList Reference	Role	Alias	Origin	ValueList Reference
Description	Question	ExternalQuestion	Measur		
Language		Translated Text			
en	Unique Subject Identifier				

showing that the "label" has already been added, i.e. loaded from the template.

For values that are coded, e.g. for LBTESTCD, clicking the "+" icon and navigating to the "CodeList Reference" tab leads to:

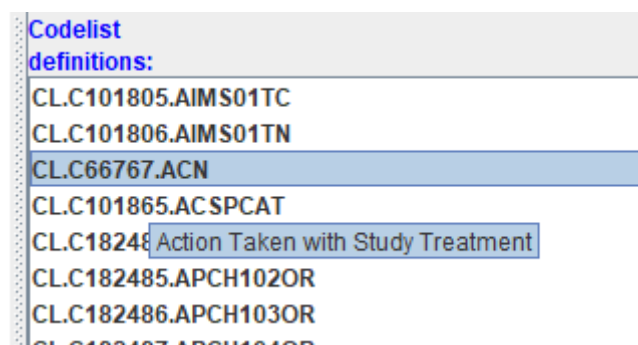
⁹This has to do with the fixed field and record length in SAS-XPT. Once the regulatory will start accepting Dataset-XML or Dataset-JSON files instead of outdated SAS-XPT, the requirement to provide a "Length" will probably be dropped.



The left side shows the pre-defined codelist (OID identifier) from the loaded template, the right side shows a list of all available codelists (including external ones like MedDRA), that can be used for drag-and-drop. Later, when having developed a subset codelist (for our study) with LBTESTCD codes, we will replace the reference to a new one for our subset codelist.

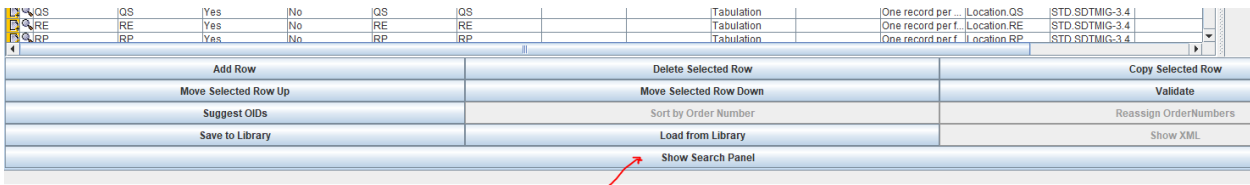
With the search button that appears under the list, it is very easy to find a specific codelist by a word in its name.

Each of the choices also has a tooltip showing the full name of the codelist, e.g.:

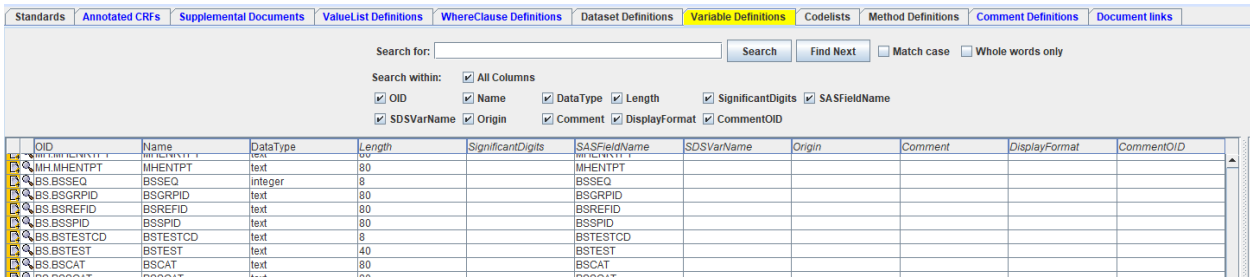


We will later also learn how to add information for "Origin" (required either on variable or value-level for each variable in the context of a regulatory submission), and other important information.

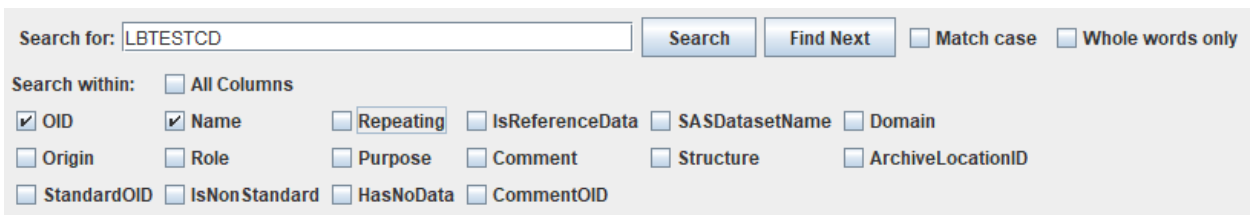
In the main window, we can easily search for a specific variable by clicking the "Show Search Panel" at the bottom of the screen:



leading to the appearance of a "Search" panel near the top of the screen:



allowing e.g. to find "LBTESTCD" in the list of variables:



and clicking the "Search" button will then select the entry for LBTESTCD:

LB.ISRFTDTC	ISRFTDTC	datetime		
LB.LBSEQ	LBSEQ	integer	8	
LB.LBGRPID	LBGRPID	text	80	
LB.LBREFID	LBREFID	text	80	
LB.LBSPID	LBSPID	text	80	
LB.LBTESTCD	LBTESTCD	text	8	
LB.LBTEST	LBTEST	text	40	
LB.LBTSTCND	LBTSTCND	text	80	
LB.LBBDAGNT	LBBAGNT	text	80	
LB.LBTSTOPO	LBTSTOPO	text	80	
LB.LBIRCAT	LBIRCAT	text	80	

clicking the "magnifying glass" icon on LBTESTCD leads to:

Contents of ItemDef with OID LB.LBTESTCD and with Name LBTESTCD

Attributes:

Name	Value
OID	LB.LBTESTCD
Name	LBTESTCD
DataType	text
Length	80
SignificantDigits	
SASFieldName	LBTESTCD
SDSVarName	
Origin	
Comment	
DisplayFormat	
CommentOID	

Content for Description

TranslatedText
Language: English
Text: Lab Test or Examination Short Name

OK Cancel

Now going back to the main window, and selecting the "Dataset Definitions", we find:

Define.xml Designer 2022 by XML4Pharma

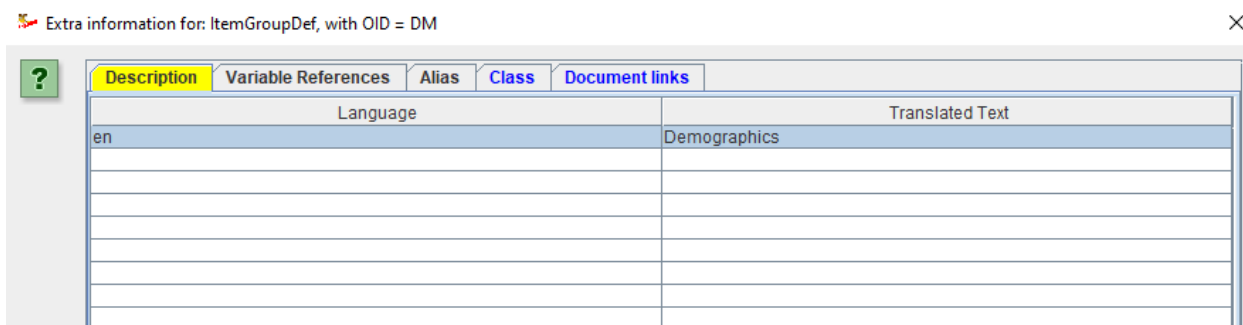
File Edit Add Transform Validate View Extra Options Help

Global Study Variables Study Metadata

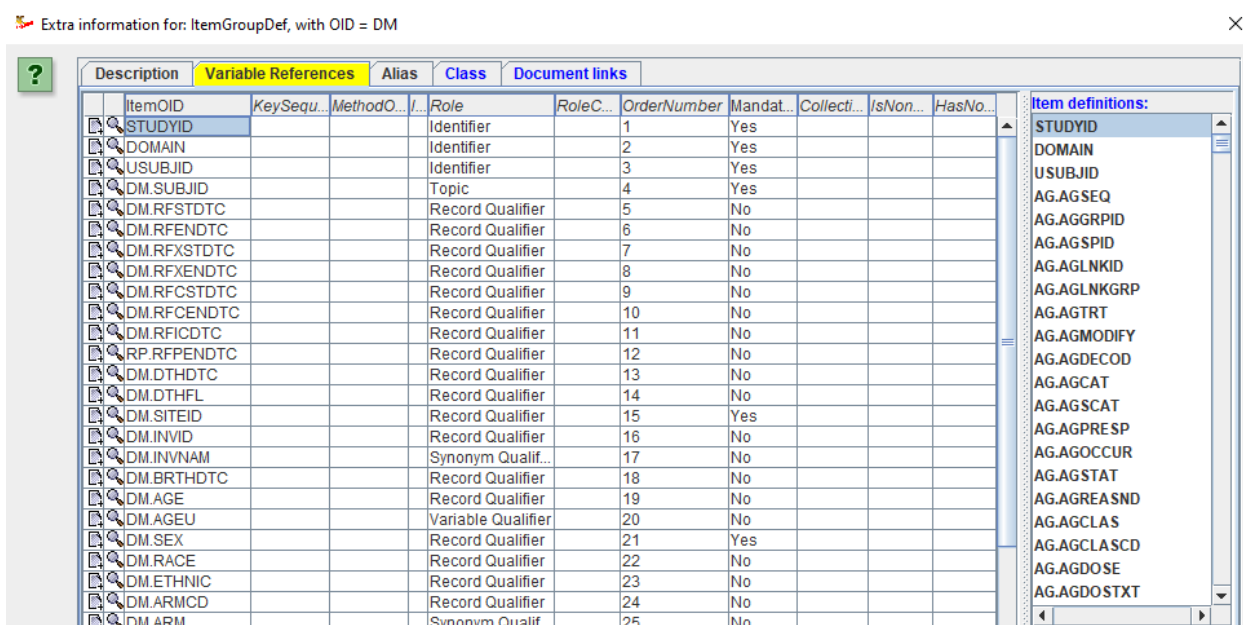
Standards	Annotated CRFs	Supplemental Documents	ValueList Definitions	WhereClause Definitions	Dataset Definitions	Variable Definitions	Codelists	Method Definitions	Comment Definitions	Document links			
OID	Name	Repeating	IsReferenceData	SASDatasetVa...	Domain	Origin	Role	Purpose	Comment	Structure	ArchiveLocation...	StandardOID	IsNo
CO	CO	Yes	No	CO	CO			Tabulation		One record per ...	Location.CO	STD_SDTMIG-3.4	
DM	DM	Yes	No	DM	DM			Tabulation		One record per ...	Location.DM	STD_SDTMIG-3.4	
SE	SE	Yes	No	SE	SE			Tabulation		One record per ...	Location.SE	STD_SDTMIG-3.4	
SM	SM	Yes	No	SM	SM			Tabulation		One record per ...	Location.SM	STD_SDTMIG-3.4	
SV	SV	Yes	No	SV	SV			Tabulation		One record per ...	Location.SV	STD_SDTMIG-3.4	
AG	AG	Yes	No	AG	AG			Tabulation		One record per ...	Location.AG	STD_SDTMIG-3.4	
CM	CM	Yes	No	CM	CM			Tabulation		One record per ...	Location.CM	STD_SDTMIG-3.4	
EC	EC	Yes	No	EC	EC			Tabulation		One record per ...	Location.EC	STD_SDTMIG-3.4	
EX	EX	Yes	No	EX	EX			Tabulation		One record per ...	Location.EX	STD_SDTMIG-3.4	
ML	ML	Yes	No	ML	ML			Tabulation		One record per f...	Location.ML	STD_SDTMIG-3.4	
PR	PR	Yes	No	PR	PR			Tabulation		One record per ...	Location.PR	STD_SDTMIG-3.4	
SU	SU	Yes	No	SU	SU			Tabulation		One record per ...	Location.SU	STD_SDTMIG-3.4	
AE	AE	Yes	No	AE	AE			Tabulation		One record per ...	Location.AE	STD_SDTMIG-3.4	
BE	BE	Yes	No	BE	BE			Tabulation		One record per l...	Location.BE	STD_SDTMIG-3.4	
CE	CE	Yes	No	CE	CE			Tabulation		One record per ...	Location.CE	STD_SDTMIG-3.4	
DS	DS	Yes	No	DS	DS			Tabulation		One record per ...	Location.DS	STD_SDTMIG-3.4	
DV	DV	Yes	No	DV	DV			Tabulation		One record per ...	Location.DV	STD_SDTMIG-3.4	
HO	HO	Yes	No	HO	HO			Tabulation		One record per ...	Location.HO	STD_SDTMIG-3.4	
MH	MH	Yes	No	MH	MH			Tabulation		One record per ...	Location.MH	STD_SDTMIG-3.4	
BS	BS	Yes	No	BS	BS			Tabulation		One record per ...	Location.BS	STD_SDTMIG-3.4	
CP	CP	Yes	No	CP	CP			Tabulation		One record per f...	Location.CP	STD_SDTMIG-3.4	
CV	CV	Yes	No	CV	CV			Tabulation		One record per f...	Location.CV	STD_SDTMIG-3.4	
DA	DA	Yes	No	DA	DA			Tabulation		One record per ...	Location.DA	STD_SDTMIG-3.4	
DD	DD	Yes	No	DD	DD			Tabulation		One record per f...	Location.DD	STD_SDTMIG-3.4	
EG	EG	Yes	No	EG	EG			Tabulation		One record per ...	Location.EG	STD_SDTMIG-3.4	
FT	FT	Yes	No	FT	FT			Tabulation		One record per f...	Location.FT	STD_SDTMIG-3.4	
GF	GF	Yes	No	GF	GF			Tabulation		One record per f... finding per subject	Location.GF	STD_SDTMIG-3.4	
IE	IE	Yes	No	IE	IE			Tabulation		One record per l...	Location.IE	STD_SDTMIG-3.4	
IS	IS	Yes	No	IS	IS			Tabulation		One record per L...	Location.IS	STD_SDTMIG-3.4	

with one row per domain.

Clicking on the "+" icon e.g. For "DM" opens a new window with:



I.e. the domain label is already present ("Demographics"). Selecting the "Variable References" leads to:



showing which variables belong to "Demographics" with a list of available variables on the right side. For other domains than DM, this will allow us later to e.g. add additional timing variables to any "Findings", "Event" or "Interventions" domain/dataset.

One also sees that the "Mandatory" field has been filled. Its value is "Yes" for the case that the SDTM variable is "required", and to "No" when the SDTM variable is "expected" or "permissible"¹⁰.

Remark that "KeySequence" is not populated, as the assignment of the table keys is a **user decision**. This is important to realize: it is up to you to decide how your SDTM/SEND/ADaM table will be organized and which combination of variables will guarantee uniqueness of the records. In the case of DM, the typical assignments for KeySequence is "1" for "STUDYID" and "2" for "USUBJID".

Going back to the main window and clicking the "magnifying glass" on e.g. "DM", the following window is displayed:

¹⁰Essentially, the designation "Expected" is an horror. A better designation would have been "conditionally required".

Contents of ItemGroupDef with OID DM and with Name DM

Attributes:

Name	Value
OID	DM
Name	DM
Repeating	Yes
IsReferenceData	No
SASDatasetName	DM
Domain	DM
Origin	
Role	
Purpose	Tabulation
Comment	
Structure	One record per subject
ArchiveLocationID	Location.DM
StandardOID	STD.SDTMIG-3.4
IsNonStandard	
HasNoData	
CommentOID	

Content for Description

TranslatedText
Language: English
Text: Demographics

Content for ItemRef

OK Cancel

and when scrolling to the bottom:

DM.ACTARMCD	ACTARMCD						Record Qualifier
DM.ACTARM	ACTARM						Synonym Qualifier
DM.ARMNRS	ARMNRS						Record Qualifier
DM.ACTARMUD	ACTARMUD						Record Qualifier
DM.COUNTRY	COUNTRY						Record Qualifier
DM.DMDTC	DMDTC						Timing
DM.DMDY	DMDY						Timing

Content for Alias
No information

Content for Class (ODM extension)

Name	SubClass
SPECIAL-PURPOSE	

Content for leaf (ODM extension)

ID	href	title
Location.DM	DM.xpt	DM.xpt

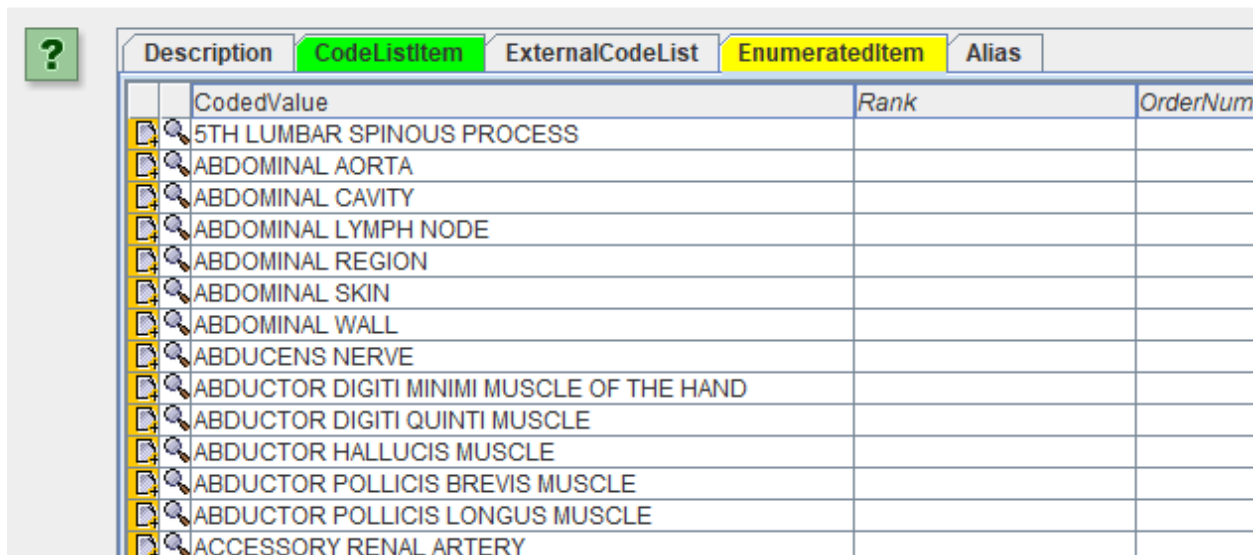
OK Cancel

Also let us have a look at the list of controlled terminology, by clicking the "CodeLists" tab:

Annotated CRFs	Supplemental Documents	ValueList Definitions	WhereClause Definitions	Dataset Definitions	Variable Definitions	CodeLists	Method D
		OID	Name			DataType	
		CL.MEDDRA	MedDRA Adverse Events Dictionary			text	
		CL.ISO3166	Country Codes			text	
		CL.C141657.TENMW1TC	10-Meter Walk/Run Functional Test Test Code			text	
		CL.C141656.TENMW1TN	10-Meter Walk/Run Functional Test Test Name			text	
		CL.C141663.A4STR1TC	4-Stair Ascend Functional Test Test Code			text	
		CL.C141662.A4STR1TN	4-Stair Ascend Functional Test Test Name			text	
		CL.C141661.D4STR1TC	4-Stair Descend Functional Test Test Code			text	
		CL.C141660.D4STR1TN	4-Stair Descend Functional Test Test Name			text	
		CL.C115388.SIXMW1TC	6 Minute Walk Functional Test Test Code			text	
		CL.C115387.SIXMW1TN	6 Minute Walk Functional Test Test Name			text	
		CL.C182464.AIMS010T07OR	Abnormal Involuntary Movement Scale Clinical Classification ORRES f...			text	
		CL.C182465.AIMS0108T09OR	Abnormal Involuntary Movement Scale Clinical Classification ORRES f...			text	
		CL.C182466.AIMS0110OR	Abnormal Involuntary Movement Scale Clinical Classification ORRES f...			text	
		CL.C182467.AIMS0111T12OR	Abnormal Involuntary Movement Scale Clinical Classification ORRES f...			text	
		CL.C182502.AIMS010T07STR	Abnormal Involuntary Movement Scale Clinical Classification STRESC f...			text	
		CL.C182503.AIMS0108T09STR	Abnormal Involuntary Movement Scale Clinical Classification STRESC f...			text	
		CL.C182504.AIMS0110STR	Abnormal Involuntary Movement Scale Clinical Classification STRESC f...			text	
		CL.C182505.AIMS0111T12STR	Abnormal Involuntary Movement Scale Clinical Classification STRESC f...			text	
		CL.C101805.AIMS01TC	Abnormal Involuntary Movement Scale Clinical Classification Test Code			text	
		CL.C101806.AIMS01TN	Abnormal Involuntary Movement Scale Clinical Classification Test Name			text	
		CL.C66767.ACIN	Action Taken with Study Treatment			text	
		CL.C101865.ACSPCAT	Acute Coronary Syndrome Presentation Category			text	
		CL.C182484.APCH101OR	Acute Physiology and Chronic Health Evaluation II Clinical Classificatio...			text	
		CL.C182485.APCH102OR	Acute Physiology and Chronic Health Evaluation II Clinical Classificatio...			text	
		CL.C182486.APCH103OR	Acute Physiology and Chronic Health Evaluation II Clinical Classificatio...			text	
		CL.C182487.APCH104OR	Acute Physiology and Chronic Health Evaluation II Clinical Classificatio...			text	
		CL.C182488.APCH105AOR	Acute Physiology and Chronic Health Evaluation II Clinical Classificatio...			text	
		CL.C182489.APCH105BOR	Acute Physiology and Chronic Health Evaluation II Clinical Classificatio...			text	
		CL.C182490.APCH106AOR	Acute Physiology and Chronic Health Evaluation II Clinical Classificatio...			text	
		CL.C182491.APCH106BOR	Acute Physiology and Chronic Health Evaluation II Clinical Classificatio...			text	
		CL.C182492.APCH107OR	Acute Physiology and Chronic Health Evaluation II Clinical Classificatio...			text	
		CL.C182493.APCH108OR	Acute Physiology and Chronic Health Evaluation II Clinical Classificatio...			text	

For each loaded codelist, it displays the identifier (the OID), the name of the codelist, and the datatype (usually "text"). Clicking the "+" icon e.g. for "CL.C74456.LOC" ("Anatomical Location") and then selecting the "CodeListItem" tab leads to:

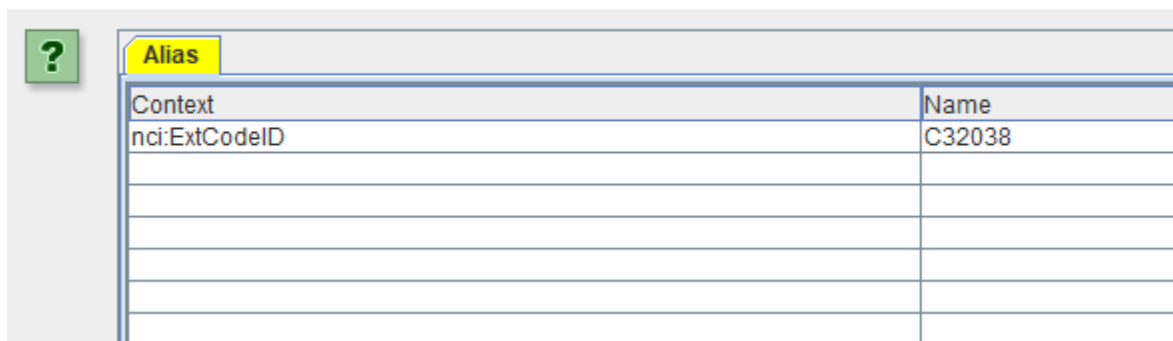
Extra information for: CodeList, with OID = CL.C74456.LOC



Description	CodeListItem	ExternalCodeList	EnumeratedItem	Alias
	CodedValue		Rank	OrderNum
+	5TH LUMBAR SPINOUS PROCESS			
+	ABDOMINAL AORTA			
+	ABDOMINAL CAVITY			
+	ABDOMINAL LYMPH NODE			
+	ABDOMINAL REGION			
+	ABDOMINAL SKIN			
+	ABDOMINAL WALL			
+	ABDUCENS NERVE			
+	ABDUCTOR DIGITI MINIMI MUSCLE OF THE HAND			
+	ABDUCTOR DIGITI QUINTI MUSCLE			
+	ABDUCTOR HALLUCIS MUSCLE			
+	ABDUCTOR POLLICIS BREVIS MUSCLE			
+	ABDUCTOR POLLICIS LONGUS MUSCLE			
+	ACCESSORY RENAL ARTERY			

One also sees that there is still underlying information for each of the coded values, as the "+" icon is colored yellow. Clicking on the "+" icon for e.g. "ABDOMINAL AORTA" leads to:

Extra information for: EnumeratedItem

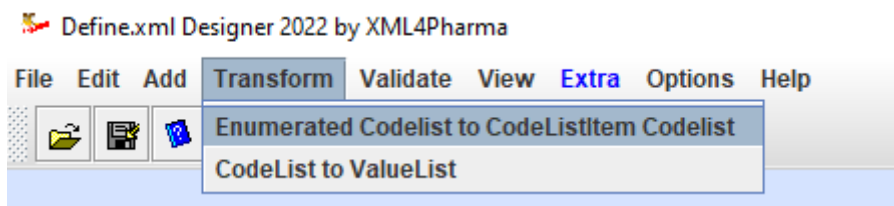


Context	Name
nci:ExtCodeID	C32038

also providing the NCI code.

Unfortunately, CDISC publishes most of its Controlled Terminology as "EnumeratedItem" codelists, so without "Decode" and "TranslatedText" possibilities, ignoring that there are other languages than English in the world.

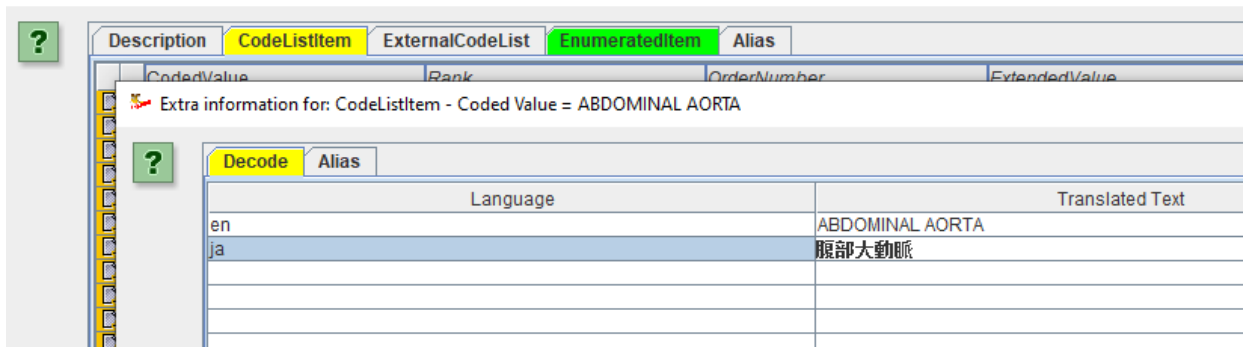
One can however transform an "EnumeratedItem" CodeList into a "CodeListItem" codelist (the latter allowing other languages) by using the menu "Transform - EnumeratedItem CodeList to CodeListItem CodeList"):



The system will then ask to select and codelist which is then transformed. The new OID is then the OID of the selected CodeList plus the suffix ".NEW". For our "Body Location" CodeList CL.C74456.LOC, we then get a new CodeList with OID

CL.C74556.LOC.NEW (one can of course later change the OID), which also has "Decodes". Editing it by clicking the "+" then allows us to e.g. add a Japanese term:

Extra information for: CodeList, with OID = CL.C74456.LOC.NEW

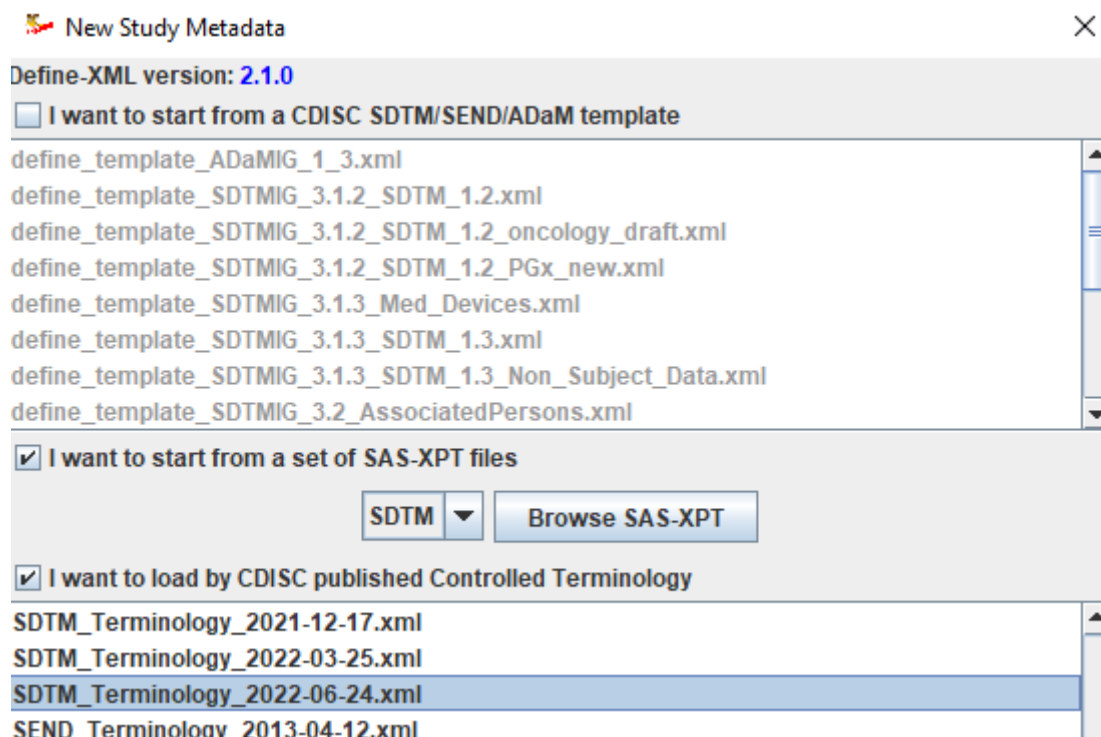


Before going into detail about other features, we will first look how to create a define.xml file from an existing set of SAS-XPT files.

Creating a define.xml starting from an existing set of SAS-XPT files

In some cases, sponsors need to generate the define.xml "post SDTM", "post SEND" or "post ADaM". Although this is not a best practice, it is an existing one, and the "Define.xml Designer" has a lot of features and wizards to bring this to a good end, this in contradiction to "black box" software that starts from an Excel worksheet and usually leads to disaster, as the users do not understand what they are doing.

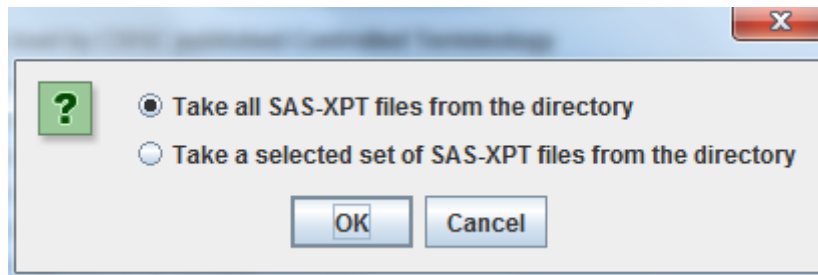
After using the "File - New Define.xml", select the checkbox "I want to start from a set of SAS-XPT files":



Then select the model, which can either be "SDTM" or "SEND" or "ADaM" or "Other". We will here demonstrate the functionality for the case of an SDTM define.xml.

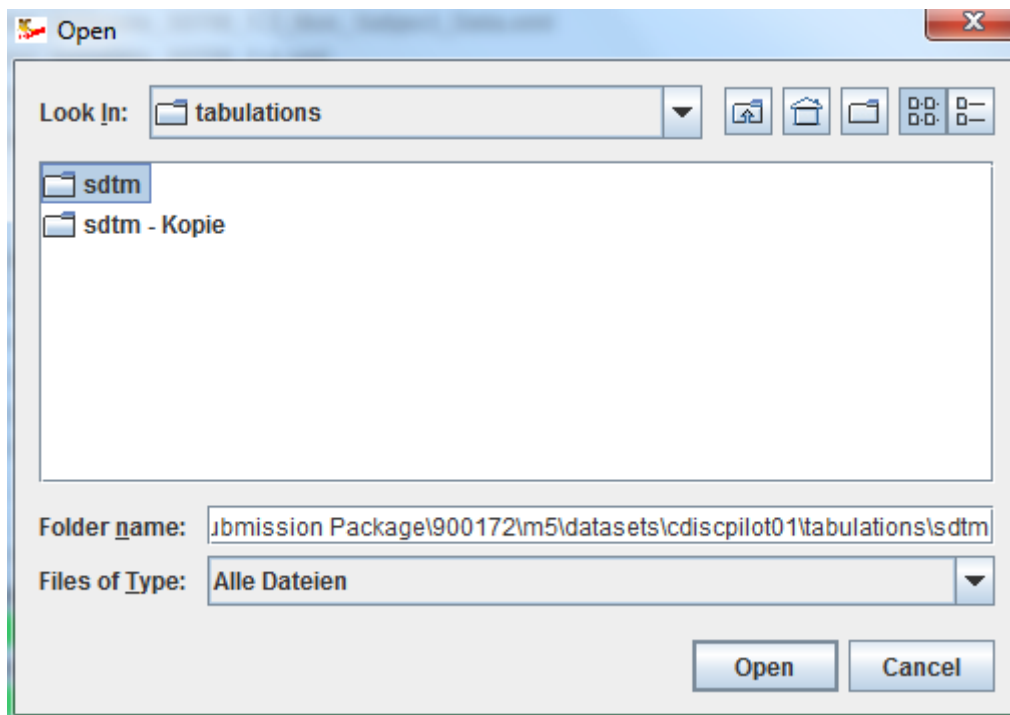
It is also very wise to then select a CDISC-CT version. This will allow the system to compare the distinct values of coded items in your SAS-XPT files with the CDISC codelists, and set up CDISC-compliant subset codelists.

When clicking the "Browse SAS-XPT" button, a dialog appears:

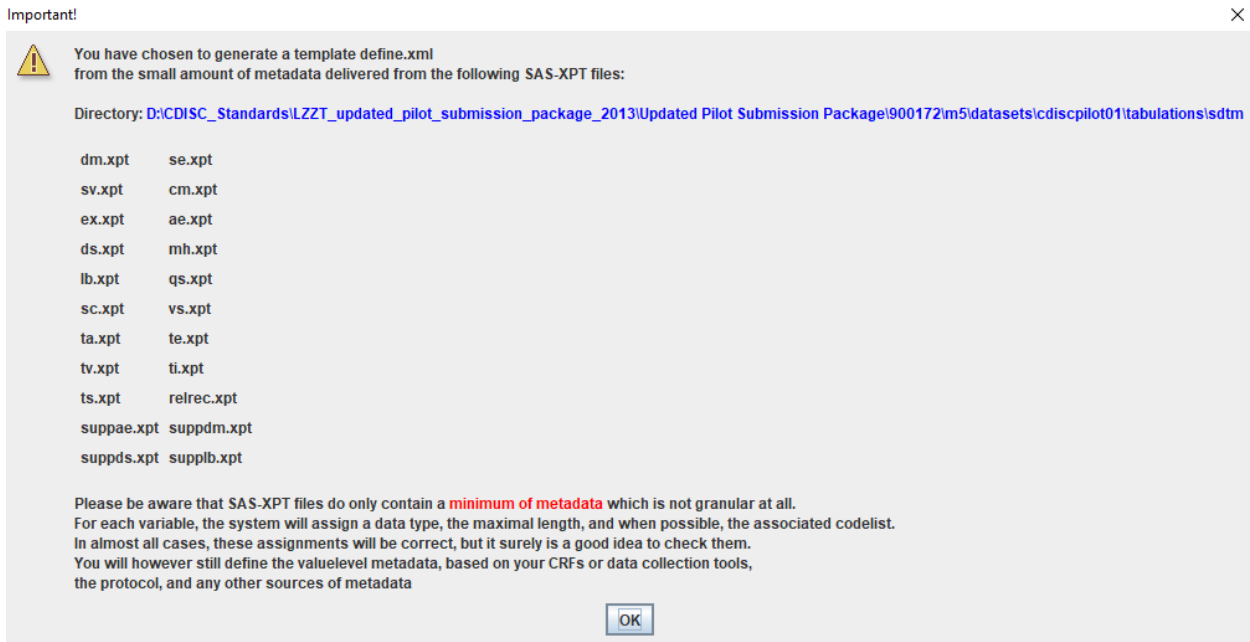


This allows you to select all SAS-XPT files from a single directory, or a selected set of SAS-XPT files. When using the first option, you will only be able to select a directory, when using the second option, you will need to select one or more SAS-XPT files from a directory.

For example for the first option:

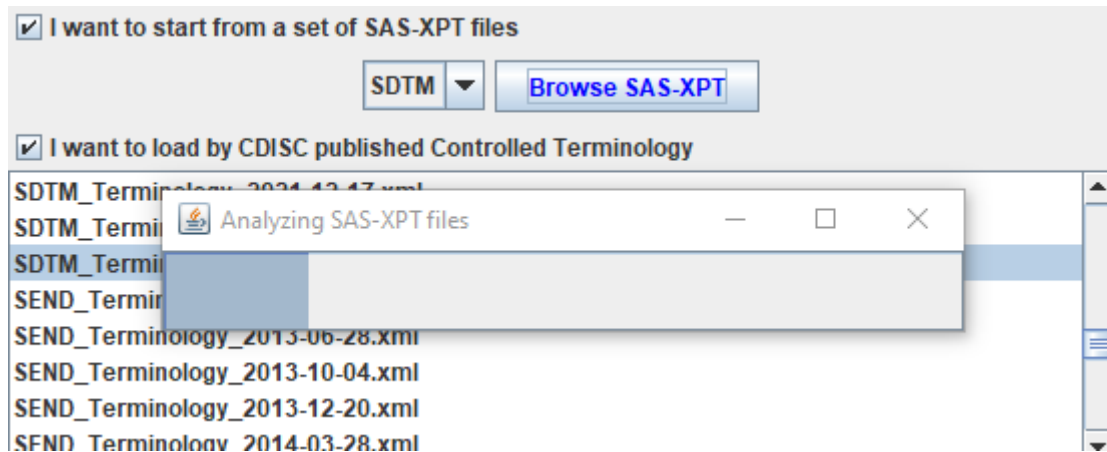


And after clicking "Open":



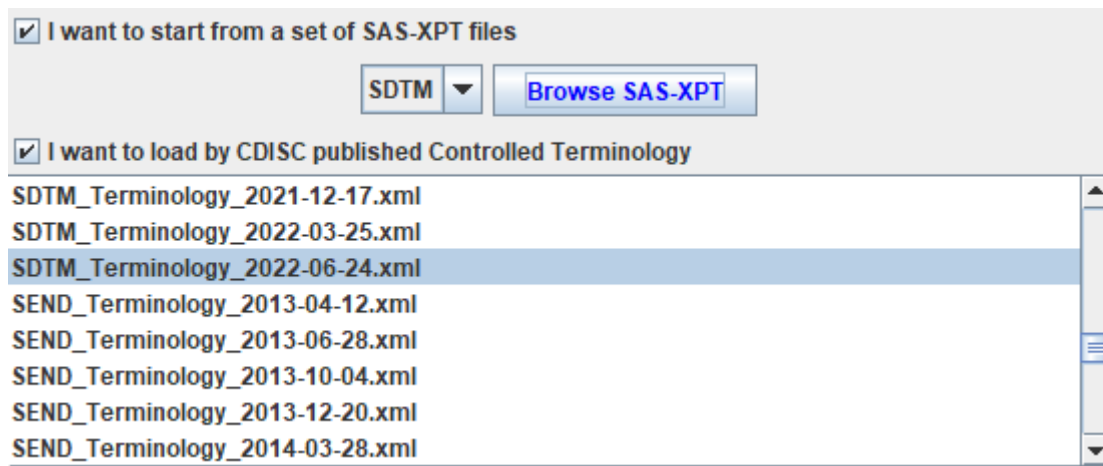
Explaining that the challenge of such an approach is that the SAS-XPT datasets only contain a minimum amount of metadata, and that you will need to get metadata from several sources, especially from the CRFs (in the case of SDTM) and from the source operational database.

After clicking OK:

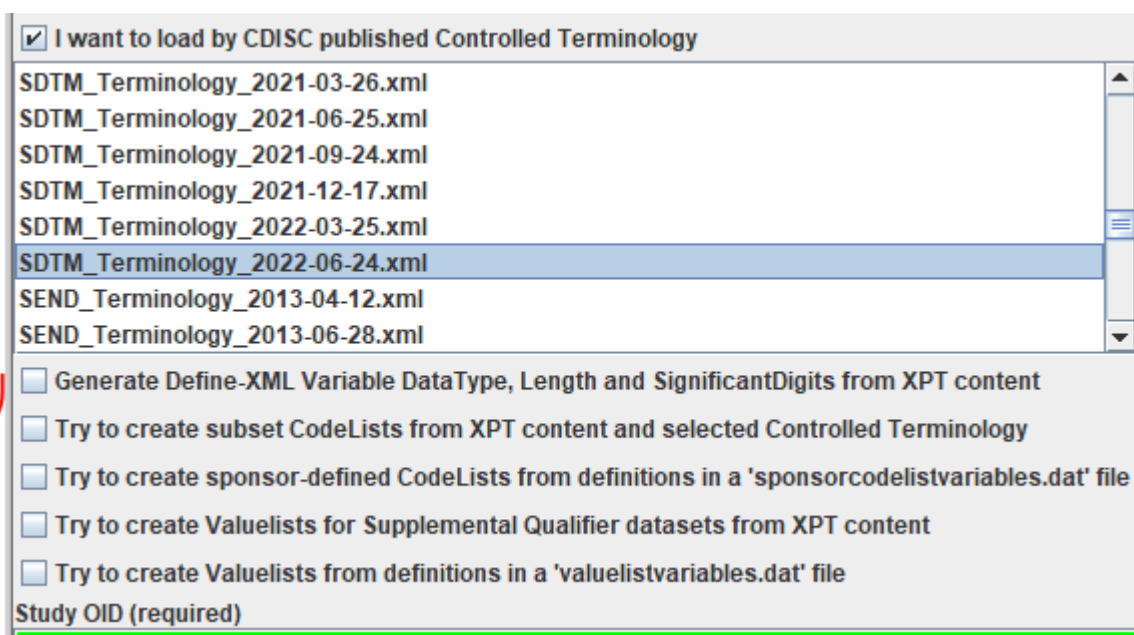


The system starts analyzing the SAS-XPT files¹¹ to generate a first primitive prototype of the define.xml. After that, it is a very good idea to select a version of the CDISC controlled terminology, and to add the general information that goes into the define.xml:

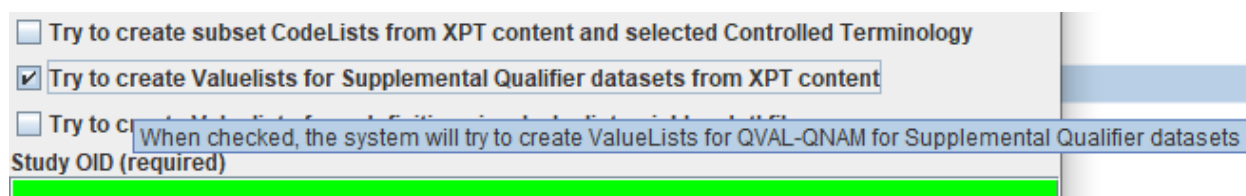
¹¹ At this stage, it only analyzes the "header" part of the SAS-XPT files, so not the complete contents. Therefore, it is fast, but the generated metadata is not very precise.



Below this part of the wizard, one will find some more checkboxes¹²:



We will explain them here in detail, although there is a tooltip for each of them:



The first one "**Generate Define-XML Variable Data Type**", "Length and SignificantDigits from XPT content", will, when checked, tell the software to generate more granular data types for the variable definitions, and adapt the values for "Length" and "SignificantDigits" (in case the data type is "float").

When this checkbox is not checked, the software will only inspect the header of the XPT files, and assign the datatype as "text" when the information in the header states that the variable is of type "char", and assign the datatype as "integer" when the information in the header states that the variable is of type "num" (numeric). It will also assign the lengths as indicated in the header (as 8 for numeric variables). So if the header says that the variable length for VSBLFL (baseline flag) is

¹² This is a new feature of the Define-XML Designer version 2022.

200, then 200 will be assigned¹³, even when the only value in that column is "Y".

However, only reading the header is very fast.

When the checkbox is checked, a thorough analysis of the contents is performed, and the longest string value for variable values determined, and set to the "Length" of the variable in the define.xml. So, even when the header states "200" for the length of e.g. VSBLFL, but the only value found is "Y", then the Length in the define.xml will be set to "1".

For numeric variables, it will be checked whether the value is a float or an integer, and the "Length" and "SignificantDigits" (the latter in the case of "float") as defined in the Define-XML specification, will be determined.

Analyzing the full contents of the SAS-XPT files is however computing intensive, especially for large XPT files. So this process can take a few minutes.

The second checkbox "**Try to create subset CodeLists from XPT content and selected Controlled Terminology**" does exactly what it says. For each variable to which there is controlled terminology according to the Implementation Guide (IG), the system will retrieve the distinct values from that column from the XPT file, and create a codelist for the variable. This codelist is then compared with the CDISC one, and additional information, like the NCI code is added. If a distinct value is not found in the CDISC codelist, it is marked as an extension to the codelist, using the `def:ExtendedValue="Yes"` attribute.

In case of Define-XML 2.1, also the attribute `def:StandardOID` is added. This will be explained further on.

The newly generated codelist is then assigned to the corresponding variable definition (ItemDef).

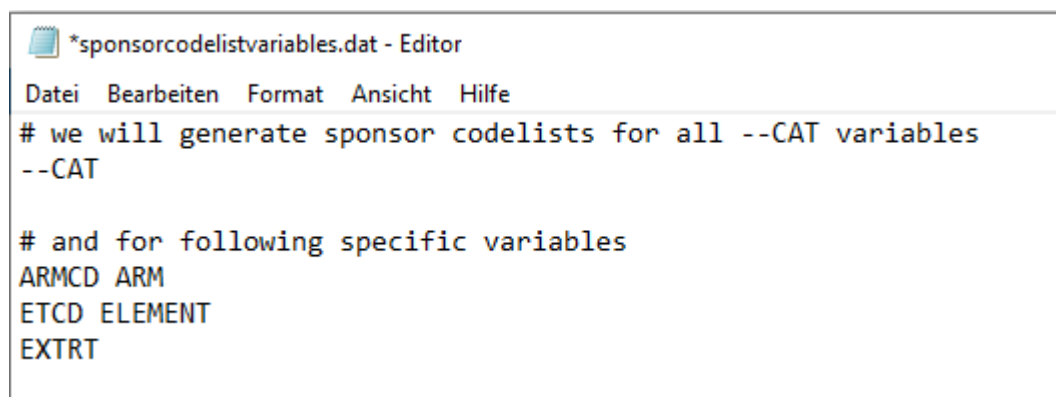
On one hand, this feature is very useful, allowing to obtain relative small codelists that reflect what has been captured, but at the other hand, it can be pretty tricky when does not think about the implications of this approach:

Imagine for example, that on the CRF, you had "Mild", "Moderate" and "Severe" for the Adverse Event Severity, but there was never a severe AE, so your XPT dataset will not have a value "SEVERE" for "AESEV". In that case, the generated codelist will only contain "MILD" and "MODERATE". The submission codelist must however reflect what was **planned**, in this case reflect the CRF, so also "SEVERE" must then be added to the codelist. As the system cannot know what exactly was planned, checking the generated codelist for completeness (and correctness) is the task of the user.

Also new is the checkbox "**Try to create sponsor-defined CodeLists from definitions in a 'sponsorcodelistvariables.dat' file**".

This file needs to reside in the folder where also the software is located.

Here is an example of the contents of the file:



```
*sponsorcodelistvariables.dat - Editor
Datei Bearbeiten Format Ansicht Hilfe
# we will generate sponsor codelists for all --CAT variables
--CAT

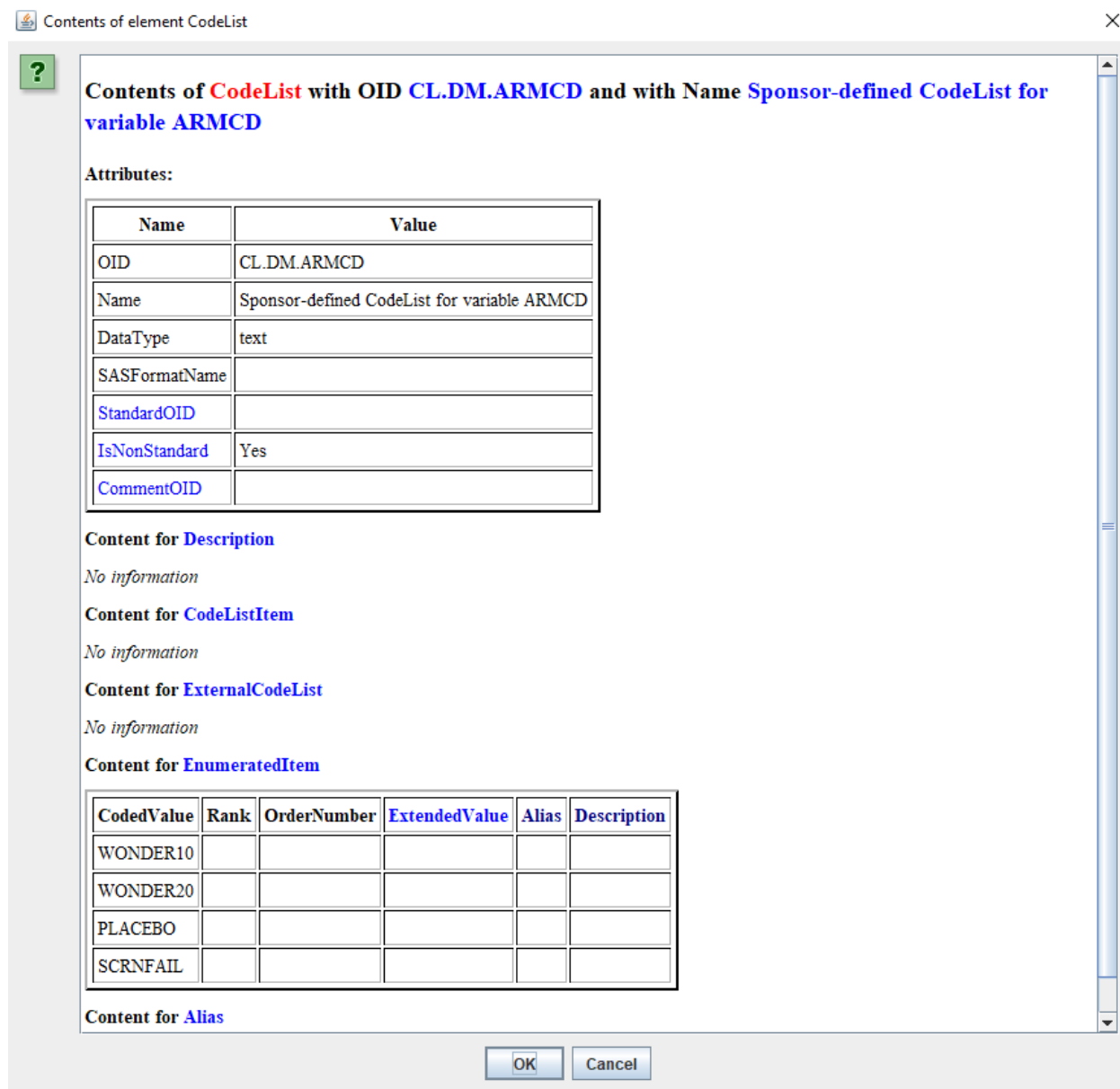
# and for following specific variables
ARMCD ARM
ETCD ELEMENT
EXTRT
```

¹³ The fixed-length format for variables is one of the major problems of the XPT format. It also means that if the string length is less than what is provided in the header, the remaining of the field for that variable is filled with blanks. This is why SAS Transport is such an inefficient format. Reason is that it is one step beyond the punch card ...

It states that it wants the software to generate sponsor-defined codelists for all --CAT variables, independent of the domain, and furthermore sponsor-defined codelists for ARMCD, ARM, ETC, ELEMENT, and EXTRT.

Remark that lines starting with a "#" are comment lines, which will be skipped by the software.

When this checkbox is checked, the result for e.g. ARMCD and ARM (in DM) will e.g. be:




One sees that for Define-XML 2.1, the "def:IsNonStandard" is set to "Yes", as it is a sponsor-defined codelist and not a CDISC codelist.

The fourth checkbox "**Try to create ValueLists for Supplemental Qualifier datasets from XPT content**", will inspect the SUPPxx datasets, and analyze the QNAM and QVAL values. For each distinct QNAM/QLABEL, it will retrieve the distinct values of QVAL, and assign a data type to the latter. For example for QNAM=FSYMPDAT with QLABEL="Date of first symptoms", it will only find dates for QVAL, and then assign the datatype "date" and use that for generating the

ValueList. When the assigned datatype is "text", it will generate a codelist with the distinct values of QVAL, and use that in the valuelist.

For example, for the file SUPPDM, this will lead to:

 Contents of element ValueListDef

?

Contents of ValueListDef with OID VL.SUPPDM.QVAL

Attributes:

Name	Value
OID	VL.SUPPDM.QVAL

Content for Description

No information

Content for ItemRef

ItemOID	Item Name	KeySequence	MethodOID	Method Name	ImputationMethodOID	ImputationMethod Name	Role	Role
IT.SUPPDM.QVAL.COMPLT24	COMPLT24							
IT.SUPPDM.QVAL.COMPLT8	COMPLT8							
IT.SUPPDM.QVAL.EFFICACY	EFFICACY							
IT.SUPPDM.QVAL.SAFETY	SAFETY							
IT.SUPPDM.QVAL.COMPLT16	COMPLT16							
IT.SUPPDM.QVAL.ITT	ITT							

and for example for the properties of "ITT":

Contents of ItemDef with OID IT.SUPPDM.QVAL.ITT and with Name ITT

Attributes:

Name	Value
OID	IT.SUPPDM.QVAL.ITT
Name	ITT
DataType	text
Length	1
SignificantDigits	
SASFieldName	
SDSVarName	
Origin	
Comment	
DisplayFormat	
CommentOID	

Content for Description

TranslatedText
Language: not assigned Text: Intent to Treat Population Flag

with its assigned codelist:

Contents of CodeList with OID CL.SUPPDM.QNAM.ITT.IDVARVAL and with Name CodeList for QVAL for QNAM = ITT in dataset SUPPDM

Attributes:

Name	Value
OID	CL.SUPPDM.QNAM.ITT.IDVARVAL
Name	CodeList for QVAL for QNAM = ITT in dataset SUPPDM
DataType	text
SASFormatName	
StandardOID	
IsNonStandard	Yes
CommentOID	

Content for Description
No information

Content for CodeListItem
No information

Content for ExternalCodeList
No information

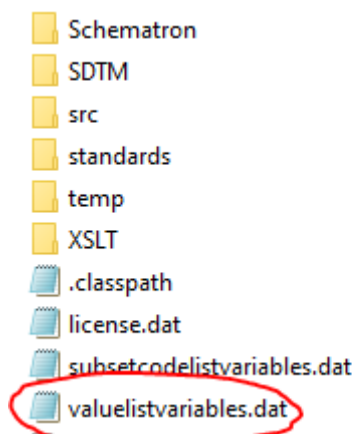
Content for EnumeratedItem

CodedValue	Rank	OrderNumber	ExtendedValue	Alias	Description
Y					

Content for Alias
No information

OK Cancel

The last checkbox "**Try to create ValueLists from definitions in a 'valuelistvariables.dat' file**" allows to have ValueLists created (at least, the system will try ...) from definitions, which need to be provided in the file "valuelistvariables.dat" in the folder where also the software resides:



The software comes with such a file, which one can edit, or even make different versions for different use cases, but when this feature is used, the definitions will be taken from the "valuelistvariables.dat" file.

Let us look at some example contents:

```
valuelistvariables.dat - Editor
Datei Bearbeiten Format Ansicht Hilfe
VSORRESU WHERE VSTESTCD EQ WEIGHT
VSORRESU WHERE VSTESTCD IN SYSBP,DIABP
#VSPOS WHERE VSTESTCD NE SYSBP,DIABP
VSPOS WHERE VSTESTCD IN SYSBP,DIABP
VSORRES WHERE VSTESTCD EQ FRMSIZE
VSORRES WHERE VSTESTCD NE FRMSIZE
# VSORRES WHERE VSTESTCD NOTIN SYSBP,DIABP,WEIGHT,HEIGHT,PULSE
# VSORRESU WHERE VSTESTCD NE HEIGHT
LBRORRES WHERE LBCAT NOTIN CHEMISTRY,HEMATOLOGY
LBSPEC WHERE LBCAT EQ HEMATOLOGY
```

Lines that start with a "#" are regarded as comments and will be skipped. This can very well be used for switching off some definitions.

The format is very similar to what one sees in the browser when displaying a define.xml in the section of "Value Level Metadata". It starts with the variable for which a valuelist will needed to be generated, the keyword "WHERE" followed by the selection condition.

For example, the first line states that a valuelist should be generated for VSORRESU with the condition that the value of VSTESTCD is "WEIGHT".

The second line states that a valuelist should be generated for VSORRESU with the condition that the value for VSTESTCD is either "SYSBP" or "DIABP".

The third line has been commented out, so will be skipped. This was done as the statement does not make much sense: it would generate a valuelist for VSPOS when the value of VSTESTCD is not one of SYSBP or DIABP. As VSPOS is usually only used for blood pressures, this would generate an empty codelist in the valuelist, which is not very useful.

The fourth line shows a classic use of valuelists, generating a valuelist for VSPOS for the case that VSTESTCD is either SYSBP or DIABP.

In the expressions, the following comparators are supported:

- EQ - meaning "equals"
- NE - meaning "does not equal"
- IN - meaning "is in the list of" - this requires that it is followed by a list of values, with the values separated by a comma
- NOTIN - meaning "is not in the list of" - this requires that it is followed by a list of values, with the values separated by a comma

Remark that the other Define-XML comparators "LT", "LE", "GT" and "GE" are not (yet) supported.

When applied, this will e.g. result in a valuelist with 2 "use cases" for VSORRESU:

Contents of ValueListDef with OID VL.VS.VSORRESU

Attributes:

Name	Value
OID	VL.VS.VSORRESU

Content for Description
No information

Content for ItemRef

ItemOID	Item Name	KeySequence	MethodOID	Method Name	ImputationMethodOID	Imputa
IT.VS.VSORRESU.VSTESTCD.WEIGHT.INCLUDE	VSORRESU					
IT.VS.VSORRESU.VSTESTCD.SYSBP_DIABP.INCLUDE	VSORRESU					

with the properties e.g. for the case of VSORRESU for "weight":

Contents of ItemDef with OID IT.VS.VSORRESU.VSTESTCD.WEIGHT.INCLUDE and with Name VSORRESU

Attributes:

Name	Value
OID	IT.VS.VSORRESU.VSTESTCD.WEIGHT.INCLUDE
Name	VSORRESU
DataType	text
Length	2
SignificantDigits	
SASFieldName	
SDSVarName	
Origin	
Comment	
DisplayFormat	
CommentOID	

and with the associated codelist:

CodedValue	Rank	OrderNumber	ExtendedValue	Alias		Description
LB				Attr.Name	Attr.Value	
				Context	nci:ExtCodeID	
				Attr.Name	Attr.Value	
				Name	C48531	
kg				Attr.Name	Attr.Value	
				Context	nci:ExtCodeID	
				Attr.Name	Attr.Value	
				Name	C28252	

Content for Alias

Context	Name
nci:ExtCodeID	C66770

and as depicted in the browser by the stylesheet:

Controlled Terms

Units for Vital Signs Results subset for ValueList [CL.VS.VSORRESU.VSTESTCD.WEIGHT.INCLUDE, C66770]

Permitted Value (Code)
LB [C48531]
kg [C28252]

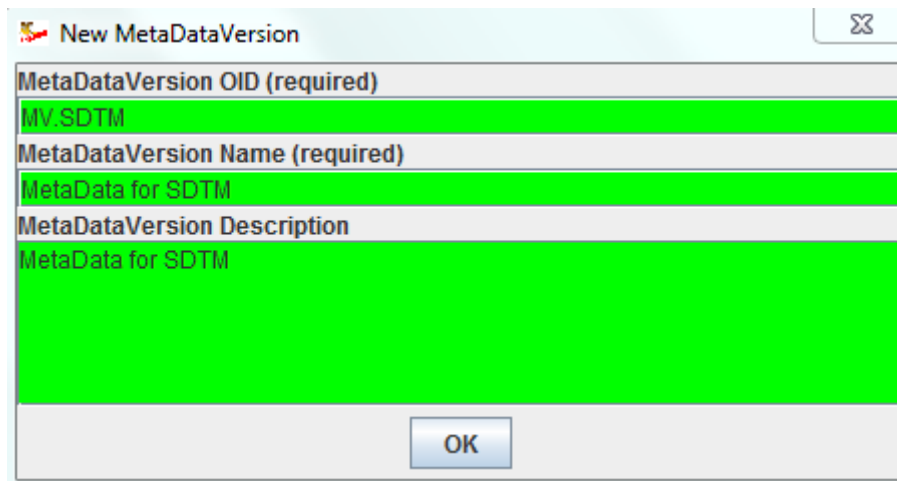
Units for Vital Signs Results subset for ValueList [CL.VS.VSORRESU.VSTESTCD.SYSBP_DIABP.INCLUDE, C66770]

Permitted Value (Code)
mmHg [C49670]

Also here, it is up to the user to carefully control the generated metadata. For example, if the CRF also had the unit choice "stones" for "weight", but it was never used, and thus is not present in the XPT file, it still must be added.

Remark that generating valuelists from XPT file contents is computing intensive, as each applicable XPT file must be analyzed. So, when one of the above features is used, you may want to go for a cup of tea or coffee.

After clicking OK in the "New Study MetaData" wizard, a proposal is made for the "MetaDataVersion OID", "Name" and Description":

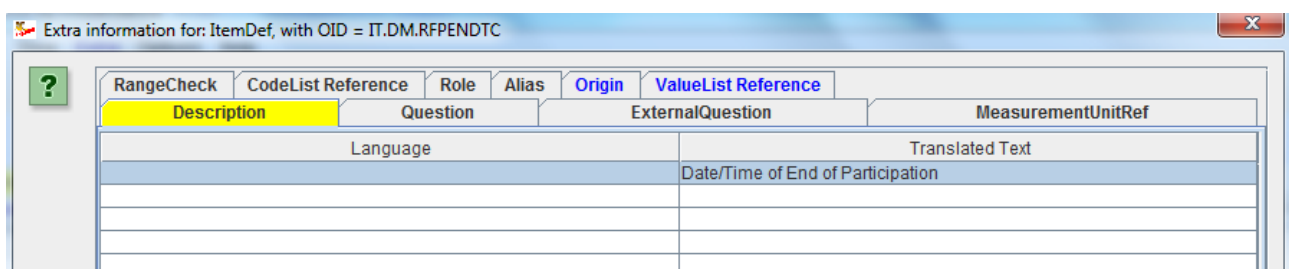


After clicking "OK", you may see some progress bars showing up, informing you about the progress made for each of the options selected. You may want to go for a cup of tea or coffee ...

When ready, this leads to a first proposal for the variable definitions:

Standards	Annotated CRFs	Supplemental Documents	ValueList Definitions	WhereClause Definitions	Dataset Definitions	Variable Definitions	C
OID	Name	Data Type	Length	Significant Digits	SASFieldName	SDSVan	
IT.DM.STUDYID	STUDYID	text	12		STUDYID		
IT.DM.DOMAIN	DOMAIN	text	2		DOMAIN		
IT.DM.USUBJID	USUBJID	text	11		USUBJID		
IT.DM.SUBJID	SUBJID	text	4		SUBJID		
IT.DM.RFSTDTC	RFSTDTC	datetime			RFSTDTC		
IT.DM.RFENDTC	RFENDTC	datetime			RFENDTC		
IT.DM.RFXSTDTC	RFXSTDTC	datetime			RFXSTDTC		
IT.DM.RFXENDTC	RFXENDTC	datetime			RFXENDTC		
IT.DM.RFICDTC	RFICDTC	datetime			RFICDTC		
IT.DM.RFPENDTC	RFPENDTC	datetime			RFPENDTC		
IT.DM.DTHDTC	DTHDTC	datetime			DTHDTC		
IT.DM.DTHFL	DTHFL	text	1		DTHFL		
IT.DM.SITEID	SITEID	text	3		SITEID		
IT.DM.AGE	AGE	integer	8		AGE		
IT.DM.AGEU	AGEU	text	6		AGEU		
IT.DM.SEX	SEX	text	1		SEX		
IT.DM.RACE	RACE	text	78		RACE		
IT.DM.ETHNIC	ETHNIC	text	25		ETHNIC		
IT.DM.ARMCD	ARMCD	text	8		ARMCD		
IT.DM.ARMCD	ARMCD	text	8		ARMCD		

When the checkbox "Generate Define-XML Variable Type, Lengths and Significant Digits from XPT contents" was not checked (the "fast way"), one sees that a first estimate of the datatype has been made for each variable, and that the maximum length has been taken from the field definitions within the header of the SAS-XPT files. Also the "label" has been taken from the SAS-XPT file, for example for "RFPENDTC":



Remark that no "language" needs to be defined, as the English language is considered as the default language by the Define-XML standard.

For the dataset definitions, we find:

Standards	Annotated CRFs	Supplemental Documents	ValueList Definitions	WhereClause Definitions	Dataset Definitions	Variable Definitions		
OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role	Purpose
IG.DM	DM	No	No	DM	DM			Tabulation
IG.SE	SE	Yes	No	SE	SE			Tabulation
IG.SV	SV	Yes	No	SV	SV			Tabulation
IG.CM	CM	Yes	No	CM	CM			Tabulation
IG.EX	EX	Yes	No	EX	EX			Tabulation
IG.AE	AE	Yes	No	AE	AE			Tabulation
IG.DS	DS	Yes	No	DS	DS			Tabulation
IG.MH	MH	Yes	No	MH	MH			Tabulation
IG.LB	LB	Yes	No	LB	LB			Tabulation
IG.QS	QS	Yes	No	QS	QS			Tabulation
IG.SC	SC	Yes	No	SC	SC			Tabulation
IG.VS	VS	Yes	No	VS	VS			Tabulation
IG.TA	TA	No	Yes	TA	TA			Tabulation
IG.TE	TE	No	Yes	TE	TE			Tabulation
IG.TV	TV	No	Yes	TV	TV			Tabulation
IG.TI	TI	No	Yes	TI	TI			Tabulation
IG.TS	TS	No	Yes	TS	TS			Tabulation
IG.RELREC	RELREC	Yes	No	RELREC	RELREC			Tabulation
IG.SUPPAE	SUPPAE	Yes	No	SUPPAE	SUPPAE			Tabulation
IG.SUPPDM	SUPPDM	Yes	No	SUPPDM	SUPPDM			Tabulation
IG.SUPPDS	SUPPDS	Yes	No	SUPPDS	SUPPDS			Tabulation
IG.SUPPLB	SUPPLB	Yes	No	SUPPLB	SUPPLB			Tabulation

And when "drilling down" into the details using the "+" icon, e.g. for "VS":

Extra information for: ItemGroupDef, with OID = IG.VS

Description	Variable References	Alias	Class	Document links	Item definitions:						
ItemOID	KeySeque...	MethodOID	Imputation...	Role	RoleCode...	OrderNum...	Mandatory	Collection...	IsNonSt...	HasNoD...	
IT.VS.STUDYID							Yes				IT.DM.STUDYID
IT.VS.DOMAIN							No				IT.DM.DOMAIN
IT.VS.USUBJID							Yes				IT.DM.USUBJID
IT.VS.VSSEQ							Yes				IT.DM.SUBJID
IT.VS.VSTESTCD							Yes				IT.DM.RFSTDTC
IT.VS.VSTEST							No				IT.DM.RFENDTC
IT.VS.VSPOS							No				IT.DM.RFXSTDTC
IT.VS.VSORRES							No				IT.DM.RFXENDTC
IT.VS.VSORRESU							No				IT.DM.RFCIDTC
IT.VS.VSSTRESC							No				IT.DM.RFPENDTC
IT.VS.VSSTRESN							No				IT.DM.DTHDTC
IT.VS.VSSTRESU							No				IT.DM.DTHFL
IT.VS.VSSTAT							No				IT.DM.SITEID
IT.VS.VSLOC							No				IT.DM.AGE
IT.VS.VSBLFL							No				IT.DM.AGEU
IT.VS.VSITNUM							No				IT.DM.SEX
IT.VS.VISIT							No				IT.DM.RACF
IT.VS.VISITDY							No				
IT.VS.VSDTC							No				

One sees that a good amount of information already has been added, based on both the SAS-XPT files and the knowledge about the SDTM or SEND standard that the system has. Furthermore, everything is transparent, and it is completely clear what the system has generated and what not, this is contradiction to "black box" tools that usually start from Excel worksheets, and that do not allow to inspect what one has been generated.

In case the checkbox **"Generate Define-XML Variable Type, Lengths and SignificantDigits from XPT contents"** was checked, a full analysis of the contents of each XPT dataset was performed (the "slow" way), and the metadata information is more granular and more precise, for example for LB variables:

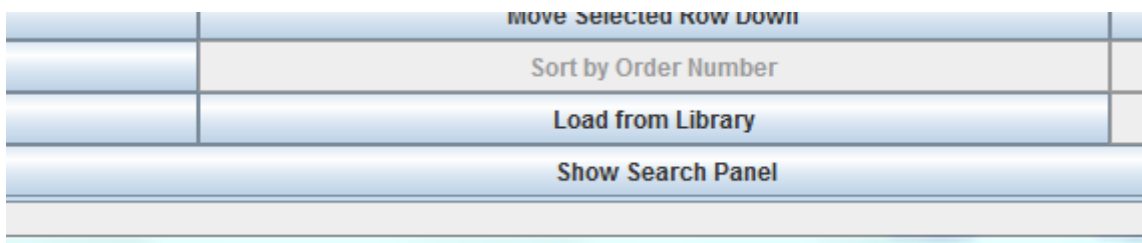
IT.LB.STUDYID	STUDYID	text	7	
IT.LB.DOMAIN	DOMAIN	text	2	
IT.LB.USUBJID	USUBJID	text	14	
IT.LB.LBSEQ	LBSEQ	integer	2	
IT.LB.LBREFID	LBREFID	text	7	
IT.LB.LBTESTCD	LBTESTCD	text	6	
IT.LB.LBTEST	LBTEST	text	19	
IT.LB.LBCAT	LBCAT	text	10	
IT.LB.LBORRES	LBORRES	text	8	
IT.LB.LBORRESU	LBORRESU	text	7	
IT.LB.LBORNRL0	LBORNRL0	text	4	
IT.LB.LBORNRHI	LBORNRHI	text	4	
IT.LB.LBSTRESC	LBSTRESC	text	8	
IT.LB.LBSTRESN	LBSTRESN	float	4	1
IT.LB.LBSTRESU	LBSTRESU	text	7	
IT.LB.LBSTNRLO	LBSTNRLO	float	3	2
IT.LB.LBSTNRHI	LBSTNRHI	float	3	2
IT.LB.LBSTNRC	LBSTNRC	text	19	
IT.LB.LBNRIND	LBNRIND	text	6	
IT.LB.LBSPEC	LBSPEC	text	5	
IT.LB.LBMETHOD	LBMETHOD	text	8	
IT.LB.LBBLFL	LBBLFL	text	1	

One sees that for LBBFL (baseline flag) the "Length" has been set to "1", as the only value occurring in the dataset is "Y", even when the SAS-XPT header e.g. defines "8" for the length for that variable.

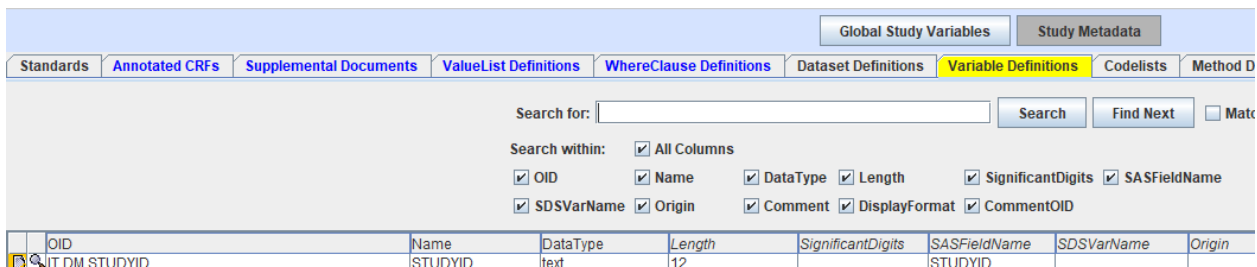
One also sees that for LBSTRESN, LBSTNRLO and LBSTNRHI, the system found out that the values are of type "float" (that information is not in the XPT header) with "Length" being "4" and "SignificantDigits". And indeed when looking into the XPT file, a typical value for LBSTRESN is "126.4".

Remark that one can also generate more exact and precise values for the "DataType", "Length" and "SignificantDigits" later, using the menu "Extra - Adapt Variable Length from SAS-XPT file contents". This is especially useful for "fine-graining" the define.xml once one has all XPT files as final.

Going back to the panel with the generated variables, one can use the "Show Search Panel" (near the bottom) to look for specific variables:



Clicking the "Show Search Panel" leads to:



allowing to search in the table with variables. Using the checkboxes one can limit the search to certain fields/columns.

As the amount of metadata in the XPT files is very limited, it is of utmost importance that one carefully inspects what has been generated in the define.xml, and extend, correct and adapt this to reflect as well the collected data as the study design.

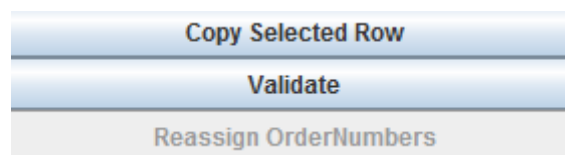
For example, the system can not retrieve the exact structure (how it was organized) for each of the XPT datasets¹⁴.

Therefore, we must add additional information at the "Dataset Definition" level:

OID	Name	Repeating	IsReference...	SASDataset...	Domain	O...	R...	Purpose	C...	Structure	Class	ArchiveLocationID	Con
IG.DM	DM	No	No	DM	DM			Tabulation			SPECIAL PURPOSE	LF.DM	
IG.SE	SE	Yes	No	SE	SE			Tabulation			SPECIAL PURPOSE	LF.SE	
IG.SV	SV	Yes	No	SV	SV			Tabulation			SPECIAL PURPOSE	LF.SV	
IG.CM	CM	Yes	No	CM	CM			Tabulation			INTERVENTIONS	LF.CM	
IG.EX	EX	Yes	No	EX	EX			Tabulation			INTERVENTIONS	LF.EX	
IG.AE	AE	Yes	No	AE	AE			Tabulation			EVENTS	LF.AE	
IG.DS	DS	Yes	No	DS	DS			Tabulation			EVENTS	LF.DS	
IG.MH	MH	Yes	No	MH	MH			Tabulation			EVENTS	LF.MH	
IG.LB	LB	Yes	No	LB	LB			Tabulation			FINDINGS	LF.LB	
IG.QS	QS	Yes	No	QS	QS			Tabulation			FINDINGS	LF.QS	
IG.SC	SC	Yes	No	SC	SC			Tabulation			FINDINGS	LF.SC	
IG.VS	VS	Yes	No	VS	VS			Tabulation			FINDINGS	LF.VS	
IG.TA	TA	No	Yes	TA	TA			Tabulation			TRIAL DESIGN	LF.TA	
IG.TE	TE	No	Yes	TE	TE			Tabulation			TRIAL DESIGN	LF.TE	
IG.TV	TV	No	Yes	TV	TV			Tabulation			TRIAL DESIGN	LF.TV	
IG.TI	TI	No	Yes	TI	TI			Tabulation			TRIAL DESIGN	LF.TI	
IG.TS	TS	No	Yes	TS	TS			Tabulation			TRIAL DESIGN	LF.TS	
IG.RELREC	RELREC	Yes	No	RELREC	RELREC			Tabulation			RELATIONSHIP	LF.RELREC	
IG.SUPPAE	SUPPAE	Yes	No	SUPPAE	SUPPAE			Tabulation			RELATIONSHIP	LF.SUPPAE	
IG.SUPPDM	SUPPDM	Yes	No	SUPPDM	SUPPDM			Tabulation			RELATIONSHIP	LF.SUPPDM	
IG.SUPPDS	SUPPDS	Yes	No	SUPPDS	SUPPDS			Tabulation			RELATIONSHIP	LF.SUPPDS	

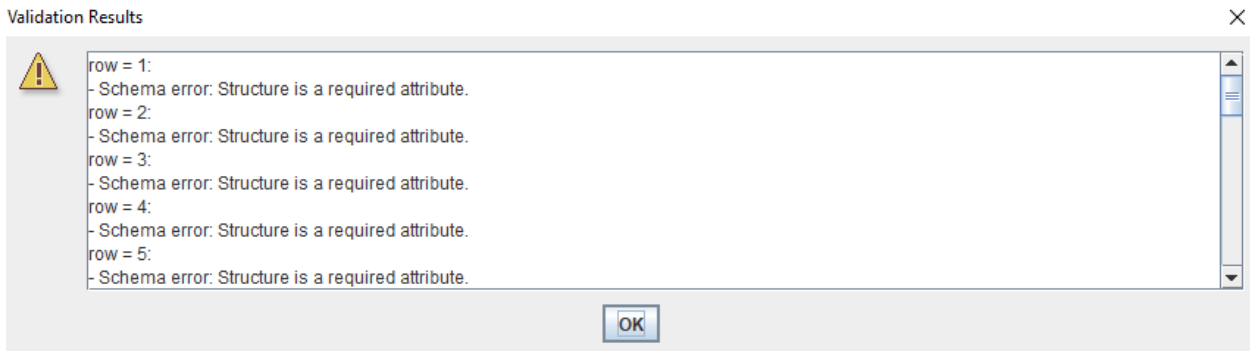
We see that for each SAS-XPT dataset, a row has been created, and a lot of information has been added, but some is still missing:

In order to find out what, click the "Validate" button in the buttons panel near the bottom:



The software does a quick "local" validation and soon reports:

¹⁴ This may maybe be possible in a future version of the software.



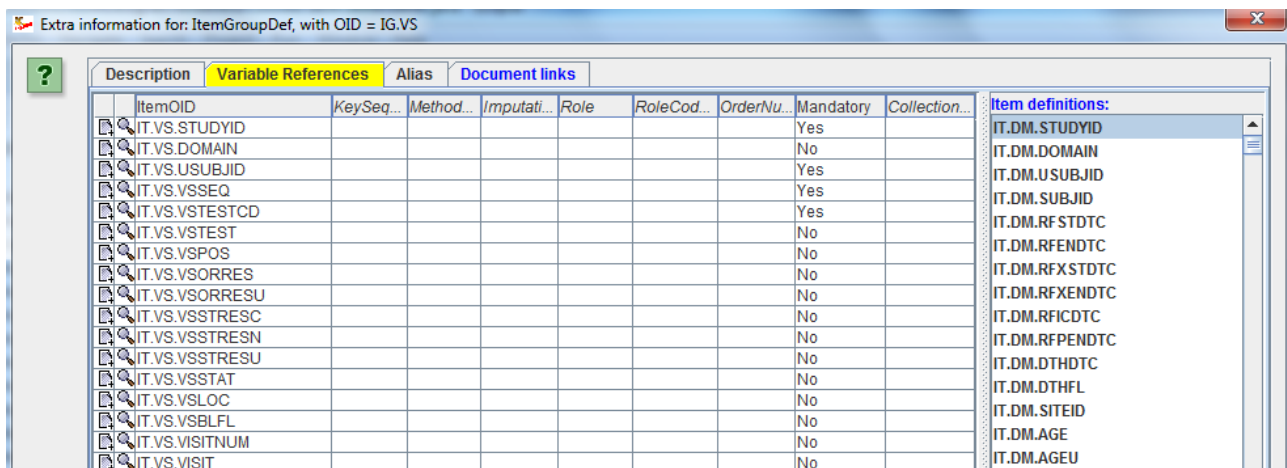
and in the table with the datasets itself, some cells are colored:

Purpose	C...	Structure	Class	ArchiveL...
Tabulation			SPECIAL PURPOSE	LF.DM
Tabulation			SPECIAL PURPOSE	LF.SE
Tabulation		Required attribute	SPECIAL PURPOSE	LF.SV
Tabulation			INTERVENTIONS	LF.CM
Tabulation			INTERVENTIONS	LF.EX
Tabulation			EVENTS	LF.AE

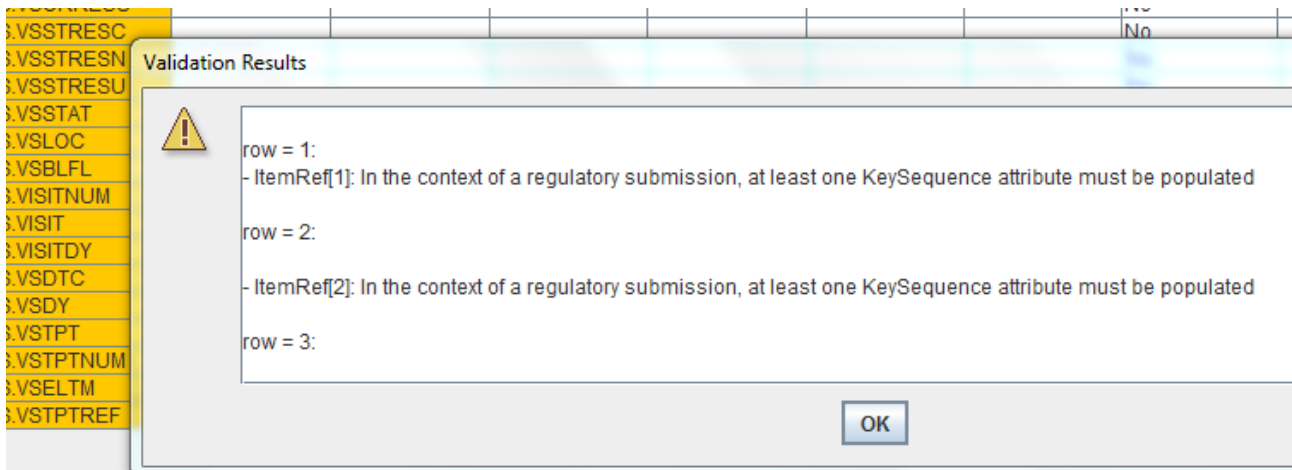
stating that "Structure" is a required attribute. A quick look in the Define-XML specification learns us that this field contains free text like "one record per vital sign per vital signs measurement per subject". This is of course information that cannot be retrieved from the SAS-XPT file and that needs to be added manually. If the "structure" is not clear from the organization of the SAS-XPT file, you will need to ask the person who generated the file, or look into the documentation of or the code that generated the SAS-XPT dataset. In most cases however, the "structure" can be deduced from the organization of the dataset itself. Remark that this is important information for the reviewer (and that is also why it is a required attribute).

Also remark that the "structure" can deviate considerably from the structure that is proposed in the SDTMIG or SENDIG. So do NOT copy the "structure" from the SDTM-IG or SEND-IG just like that: think yourself!

After having added the "structure" for each dataset, let us "drill down" into the details by clicking the "+" icon. For example for the "Variable References":



Clicking the "Validate" button near the bottom again leads to the message:



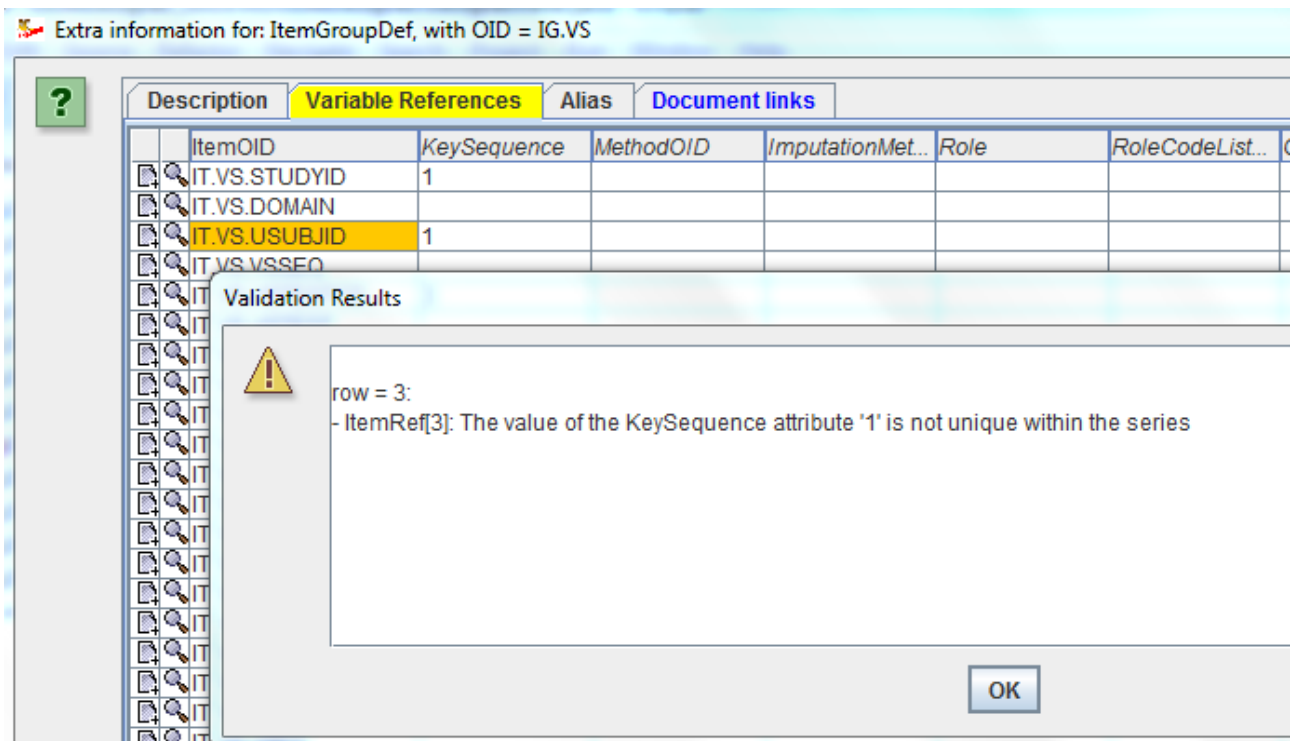
saying us that (in case the define.xml will be used in a regulatory submission), one must assign keys (and their sequence) to the variables. Examples about how this needs to be done are given in the Define-XML specification. An example for "Vital Signs" is:

Extra information for: ItemGroupDef, with OID = IG.VS

Description	Variable References	Alias	Document links
ItemOID	KeySeq...	Method...	Imputati... Role RoleCod... OrderN
IT.VS.STUDYID	1		
IT.VS.DOMAIN			
IT.VS.USUBJID	2		
IT.VS.VSSEQ			
IT.VS.VSTESTCD	3		
IT.VS.VSTEST			
IT.VS.VSPOS	6		
IT.VS.VSORRES			
IT.VS.VSORRESU			
IT.VS.VSSTRESC			
IT.VS.VSSTRESN			
IT.VS.VSSTRESU			
IT.VS.VSSTAT			
IT.VS.VSLOC			
IT.VS.VSBLFL			
IT.VS.VISITNUM	5		
IT.VS.VISIT			
IT.VS.VISITDY			
IT.VS.VSDTC	4		
IT.VS.VSDY			

Do not copy this example without knowing what you are doing! This is an example only! You will need to look into the organization of your clinical database or into the code that generated the dataset in order to find out which were the key variables, and in which order the keys apply.

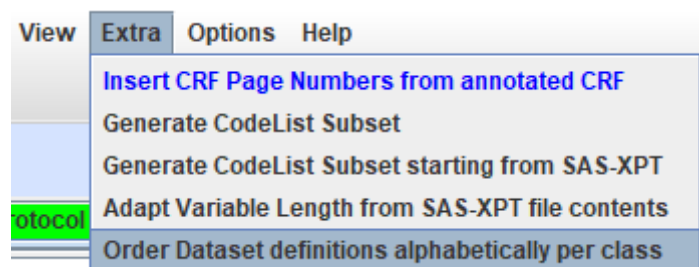
In case we accidentally assign the same key number twice, and validate again, the validation message becomes:



Ordering the dataset definitions

Especially when generating the define.xml starting from SAS-XPT files, the dataset names in the tab "Dataset Definitions" will probably not be in the order FDA reviewers want it to be¹⁵. For SDTM and SEND, the by the "[CDISC Metadata Submission Guide](#)" and FDA required order is per class (as in the "def:Class" attribute in the define.xml) and within the class, alphabetically. For ADaM, the suggested order is "Subject Level Analysis Datasets", followed by "Adverse Events Analysis Datasets", followed by "Basic Data Structure" and then "ADaM Other".

The dataset definitions can be ordered manually, using the buttons "Move Row Up" and "Move Row Down", but much more easy is to have it done by the program based on the above rules, using the menu "Extra – Order Dataset definitions alphabetically per class":



The ordering is done by the program. An example of "before" and "after" is given below:

¹⁵ Essentially, this is a bit ridiculous, as this is only for "ease of review" in the browser, which could easily be managed by a standardized FDA stylesheet. However, the FDA is not capable of developing stylesheets. For machine-readability, the order of the dataset definitions is completely irrelevant.

WhereClause Definitions		Dataset Definitions	Value
Annotated CRFs			
	OID	Name	Repeating
	IG.DM	DM	No
	IG.SE	SE	Yes
	IG.SV	SV	Yes
	IG.CM	CM	Yes
	IG.EX	EX	Yes
	IG.AE	AE	Yes
	IG.DS	DS	Yes
	IG.MH	MH	Yes
	IG.LB	LB	Yes
	IG.QS	QS	Yes
	IG.SC	SC	Yes
	IG.VS	VS	Yes
	IG.TA	TA	No
	IG.TE	TE	No
	IG.TV	TV	No
	IG.TI	TI	No
	IG.TS	TS	No
	IG.RELREC	RELREC	Yes
	IG.SUPPAE	SUPPAE	Yes
	IG.SUPPDM	SUPPDM	Yes
	IG.SUPPDS	SUPPDS	Yes
	IG.SUPPLB	SUPPLB	Yes

WhereClause Definitions		Dataset Definitions	Value
Annotated CRFs			
	OID	Name	Repeating
	IG.TA	TA	No
	IG.TE	TE	No
	IG.TI	TI	No
	IG.TS	TS	No
	IG.TV	TV	No
	IG.DM	DM	No
	IG.SE	SE	Yes
	IG.SV	SV	Yes
	IG.CM	CM	Yes
	IG.EX	EX	Yes
	IG.AE	AE	Yes
	IG.DS	DS	Yes
	IG.MH	MH	Yes
	IG.LB	LB	Yes
	IG.QS	QS	Yes
	IG.SC	SC	Yes
	IG.VS	VS	Yes
	IG.RELREC	RELREC	Yes
	IG.SUPPAE	SUPPAE	Yes
	IG.SUPPDM	SUPPDM	Yes
	IG.SUPPDS	SUPPDS	Yes
	IG.SUPPLB	SUPPLB	Yes

Adding and subsetting CDISC Controlled Terminology

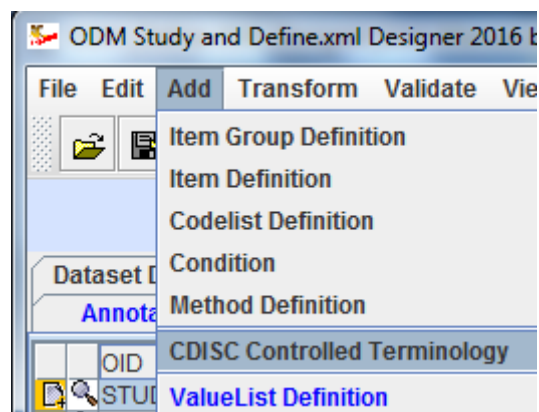
For most "Findings" domains, you will want to use CDISC controlled terminology. When starting a new define.xml you will be able to load all the CDISC controlled terminology that is needed. When new CDISC controlled terminology is published, you can simply copy the new XML file¹⁶ to the directory, and it will be automatically available.

However, you will very often need to subset the codelists that were provided by CDISC and adapt them for your own study. If you generate the define.xml before you generate your datasets (currently in SAS-XPT format), which is the best practice, you can subset loaded codelists using the menu "Extra - Generate CodeList Subset". If you create your define.xml after you created the SAS-XPT files (not such good practice, but usually the only way in case of legacy datasets), you can compare your SAS datasets with the published controlled terminology and generate a subset of an existing codelist using information from both. The latter is explained in detail in the section "Adding and Subsetting CodeLists from SAS-XPT files".

In case you create your define.xml before generating the SDTM/SEND/ADaM datasets, you will probably subset your codelists with information from the protocol, or from the CRFs.

In the example below, we will subset the codelists for VSTESTCD (Vital Signs Test Code) and VSTEST (Vital Signs Test Name).

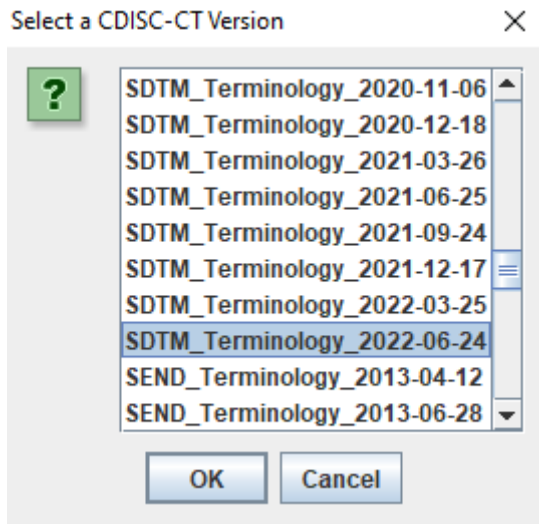
If you have not loaded any controlled terminology yet, you can now do so using the menu "Add - CDISC Controlled Terminology":



This menu can also be used to replace the controlled terminology by a newer version, or to add one or more (e.g. new) codelists.

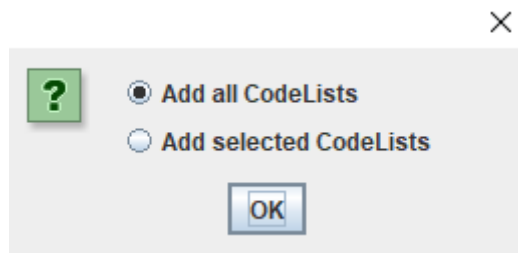
All CDISC controlled terminology is then presented as a list:

¹⁶ A download page for new published CDISC Controlled Terminology is made available by XML4Pharma

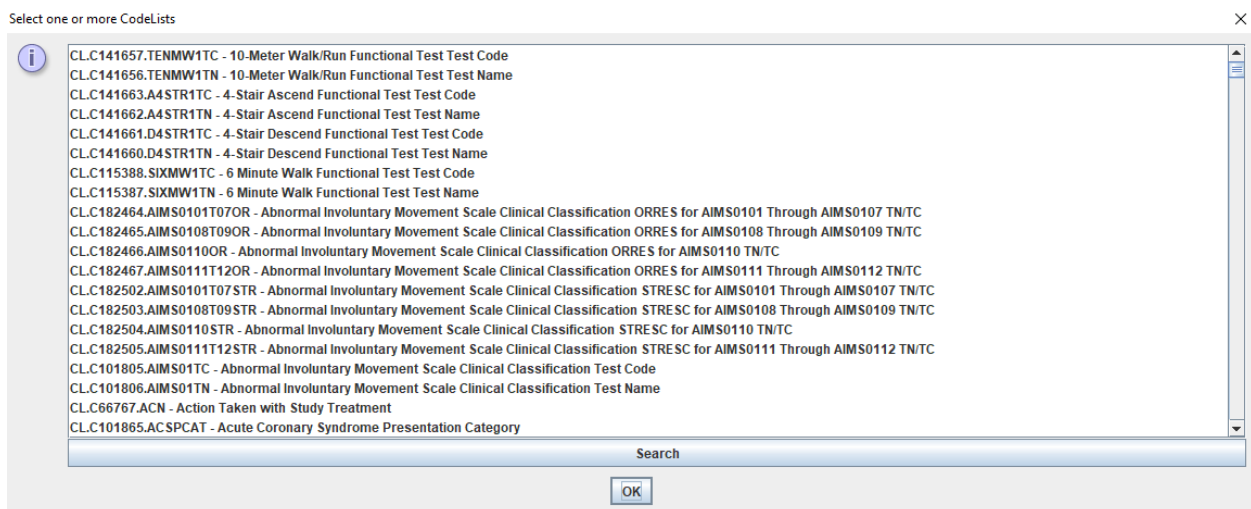


We choose e.g. the SDTM controlled terminology version 2022-06-24.

If controlled terminology (e.g. an older version) was already loaded before, the system will ask:

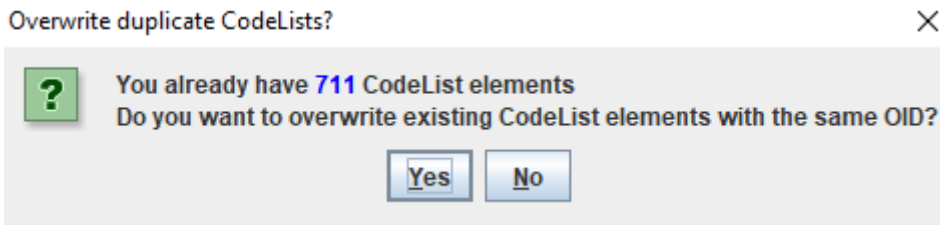


And when "Add selected CodeLists" is selected, it will ask to select one or more, e.g.:

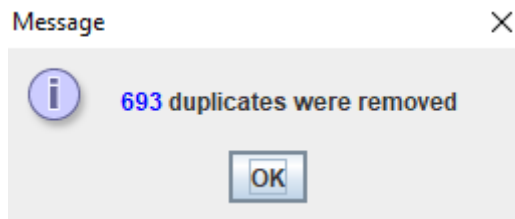


Once again, one can use the "Search" button for finding a specific codelist.

When one has then selected "all codelists" or "selected codelists", and one already has codelists loaded, e.g. an older version, the system will ask whether these earlier codelists may be overwritten when there is one with the same OID in the selected set:



If one clicks "No", there is of course the risk of duplicates ...
 So one will usually use "Yes", leading to:



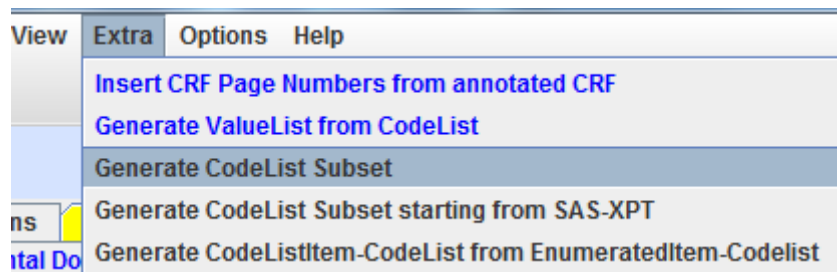
This may sometimes lead to surprises, as some codelists have been deleted or deprecated by CDISC in the course of time, but in the case of Define-XML, this will be clearly visible by the value in the "StandardOID" column. For example:

	OID	Name	DataType	SASFo	StandardOID	IsNonStanda
	CL.C100134.BPR01TC	Brief Psychiatric Rating Scale-A Questionnaire Test Code	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C100133.BPR01TN	Brief Psychiatric Rating Scale-A Questionnaire Test Name	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C135684.CGGUY1TC	Clinical Global Impression Questionnaire Test Code	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C135683.CGGUY1TN	Clinical Global Impression Questionnaire Test Name	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C66787.TDIGRP	Diagnosis Group Response	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C111346.FAQ02TC	Functional Activities Questionnaire-NACC Version Questionnaire Test Code	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C111345.FAQ02TN	Functional Activities Questionnaire-NACC Version Questionnaire Test Name	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C103329.GNRLOBSC	General Observation Class	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C132317.GENRTYP	Genetic Region of Interest Type	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C100170.KPSSTC	Karnofsky Performance Status Scale Questionnaire Test Code	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C100169.KPSSTN	Karnofsky Performance Status Scale Questionnaire Test Name	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C114119.PBSTMT	Pharmacogenomics Biomarker Medical Statement	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C116106.PFTESTCD	Pharmacogenomics Findings Test Code	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C116105.PFTEST	Pharmacogenomics Findings Test Name	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C135010.PGTSTCD	Pharmacogenomics/Genetics Methods and Supporting Information Test Code	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C135011.PGTST	Pharmacogenomics/Genetics Methods and Supporting Information Test Name	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C116111.SPCIES	SDTM Species	text		STD.SDTM.CDISC-NCI_2018-03-30	
	CL.C141657.TENMW1TC	10-Meter Walk/Run Functional Test Test Code	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C141656.TENMW1TN	10-Meter Walk/Run Functional Test Test Name	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C141663.A4STR1TC	4-Stair Ascend Functional Test Test Code	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C141662.A4STR1TN	4-Stair Ascend Functional Test Test Name	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C141661.D4STR1TC	4-Stair Descend Functional Test Test Code	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C141660.D4STR1TN	4-Stair Descend Functional Test Test Name	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C115388.SIXMW1TC	6 Minute Walk Functional Test Test Code	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C115387.SIXMW1TN	6 Minute Walk Functional Test Test Name	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C182464.AIMS0101T07OR	Abnormal Involuntary Movement Scale Clinical Classification ORRES for AIMS...	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C182465.AIMS0108T09OR	Abnormal Involuntary Movement Scale Clinical Classification ORRES for AIMS...	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C182466.AIMS0110OR	Abnormal Involuntary Movement Scale Clinical Classification ORRES for AIMS...	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C182467.AIMS0111T12OR	Abnormal Involuntary Movement Scale Clinical Classification ORRES for AIMS...	text		STD.SDTM.CDISC-NCI_2022-06-24	
	CL.C182502.AIMS0101T07STR	Abnormal Involuntary Movement Scale Clinical Classification STRESC for AIM...	text		STD.SDTM.CDISC-NCI_2022-06-24	

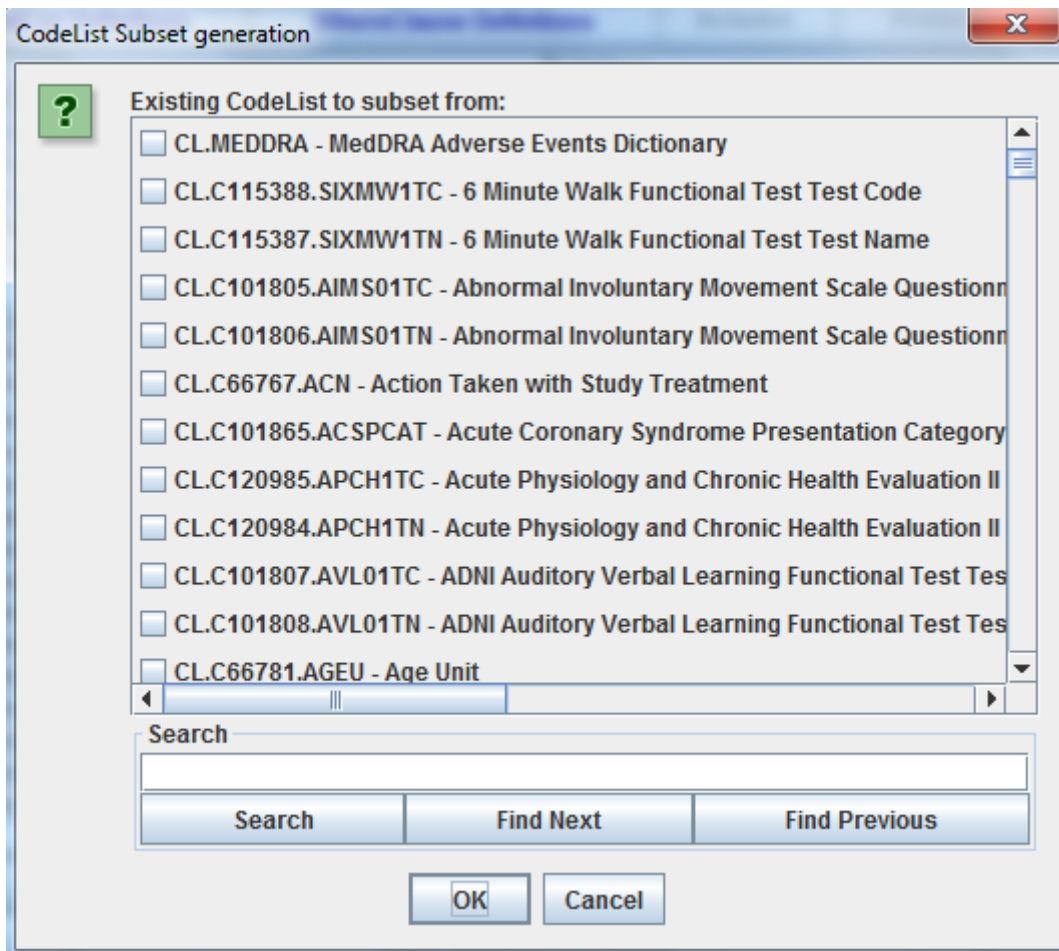
where one sees that for 17 codelists from the 2018-03-30 version, there is no newer 2022-06-24 version.

One can now of course already remove codelists that are not needed, but one can also do this later in an automated way using the menu "Edit - Clean" (see later).

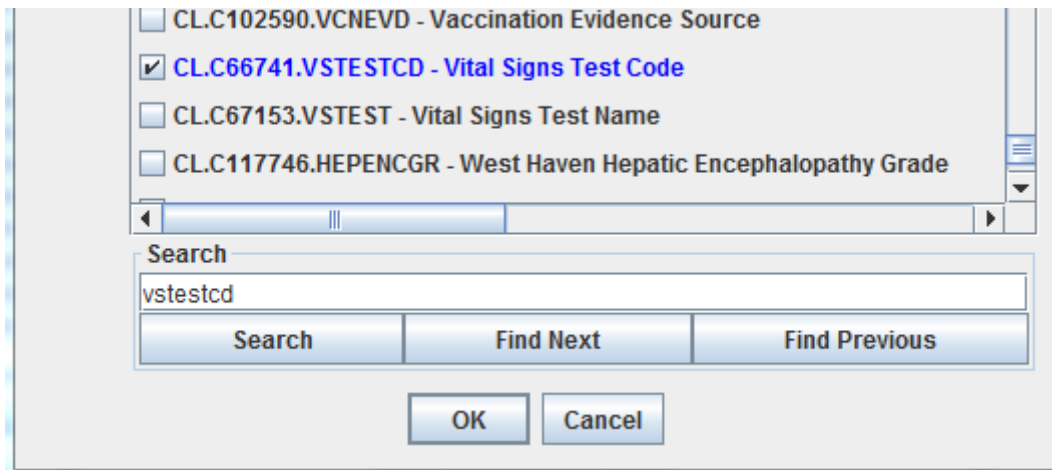
Once loaded, each of the loaded codelists can be **subsetting** by using the menu "Extra - Generate CodeList Subset":



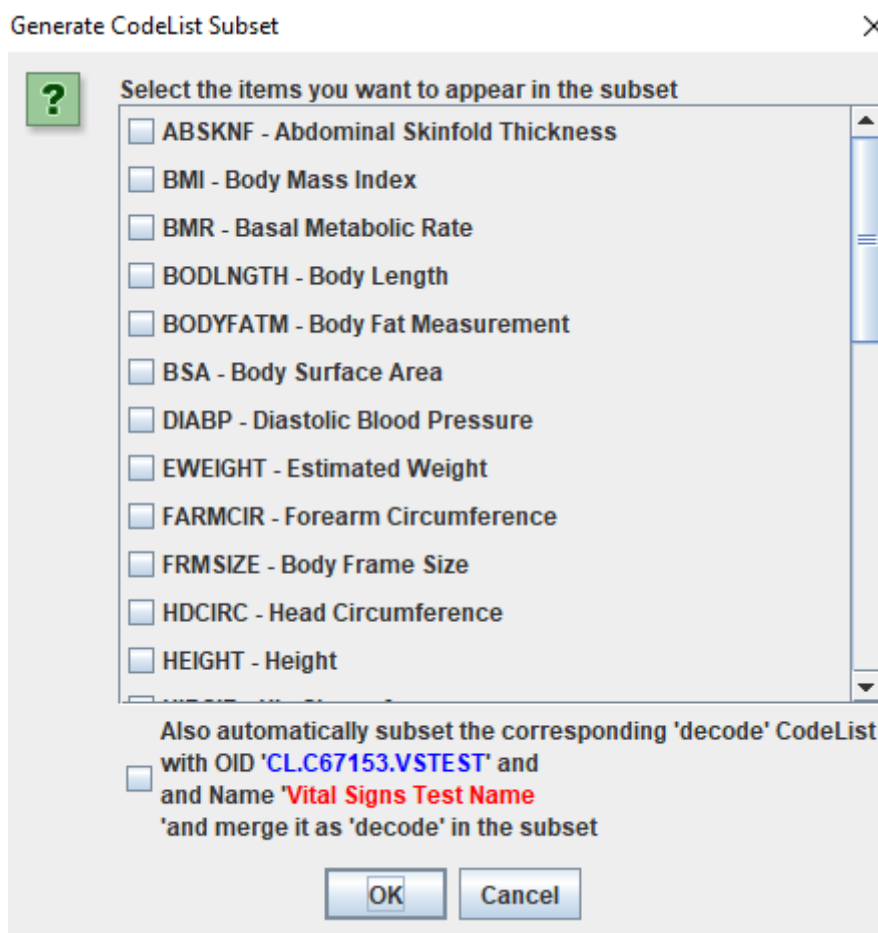
A new dialog is displayed, allowing you to select for which code list you want to generate a subset:



Searching for the right code list is made easy by the "Search" field and buttons. One can search on the code list OID (CDISC identifier) or the code list name. So for "vital signs test code" one easily finds:



Clicking "OK" leads to a new dialog:



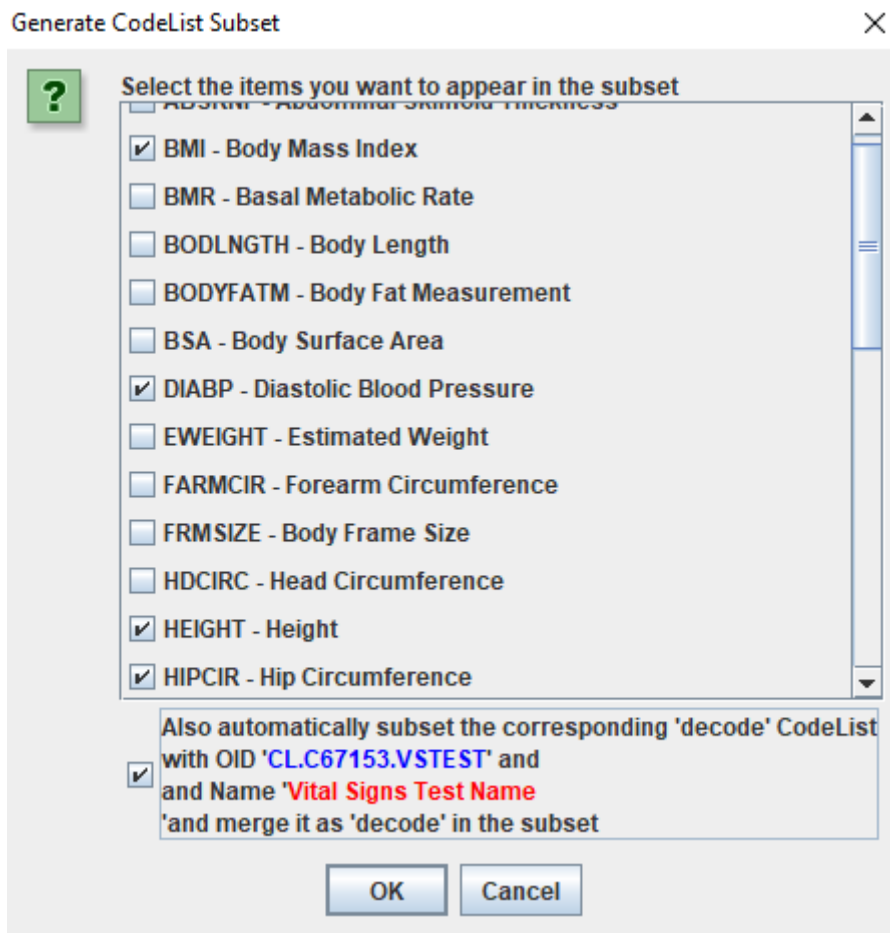
from which the codes to appear in the subset can be selected. Unfortunately, the CDISC Controlled Terminology Team still publishes codelist with test codes without any information about what the code means, i.e. without "decodes". The latter can only be found by taking the NCI code of the code, and then do a lookup in another codelist with the decodes). The software will always try to do this, so for the "VSTESTCD" codelist, although it may not contain the "decodes" like "Abdominal Skinfold Thickness", the software will pick them up from the "VSTEST" codelist, and display them. Also, the SDTM and SEND standards requires us to both deliver the test code (--TESTCD) and the test name (--TEST) as two variables (two columns in the dataset), whereas this is completely unnecessary when using modern technologies, as there is a 1:1 relation between test code and test

name (for more information, see here). So when creating subsets, we will both need to create a subset for --TESTCD as well as for --TEST.

For ease of use, the checkbox "Also automatically subset ...", takes the "decodes" (the "test names") and merges them as "decode" in the subset VSTESTCD codelist.

This then also automatically will create the corresponding subset codelist for VSTEST¹⁷.

Suppose that we want to create a vital signs test code subset consisting of "DIABP" (diastolic blood pressure), "SYSBP" (systolic blood pressure), "HEIGHT" (body height), "WEIGHT" (body weight), "BMI" (body mass index) and "HIPCIR" (hip circumference). We tick the boxes for those codes that we want to appear in the subset:



and then click the OK button. The software now proposes a new OID (identifier) and name for the subset codelist:

¹⁷ This also works for CDISC codelists that end with "Test Code" in the name, like the codelist with OID "CL.C141657.TENMW1TC" with Name "10-Meter Walk/Run Functional Test Test Code". If this codelist is taken, and the checkbox "Also automatically subset the corresponding 'decode' CodeList ..." is checked, also the corresponding subset codelist "10-Meter Walk/Run Functional Test Test Name" (subset codelist of codelist with OID "CL.C141656.TENMW1TN") will be generated.

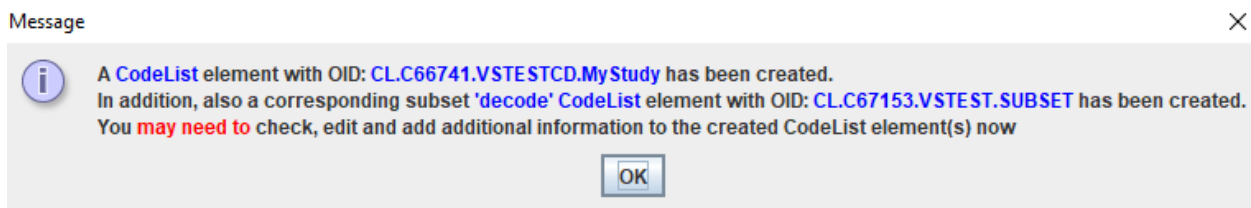


We can of course change these, for example:



If also the checkbox "Also automatically subset the corresponding 'decode' CodeList" is checked, also a subset codelist "CL.C67153.VSTEST.SUBSET" will be created.

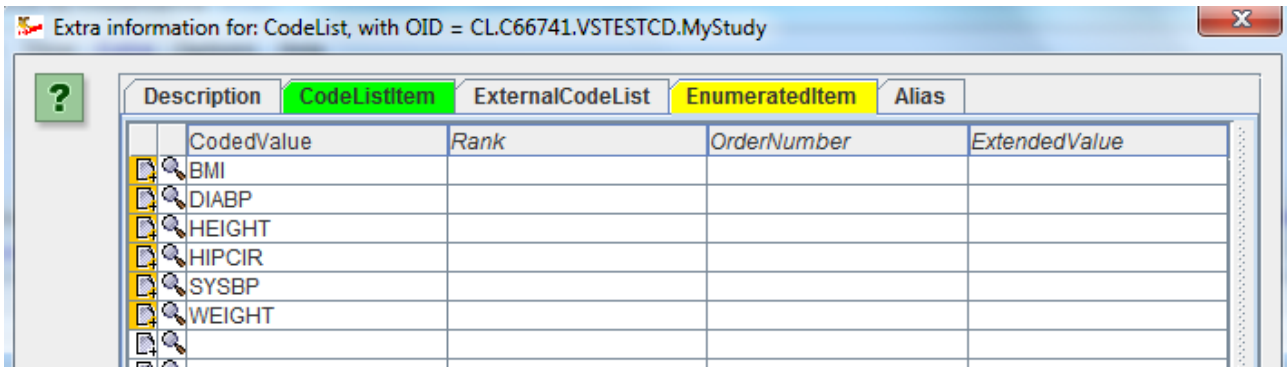
After clicking "OK", a message is shown:



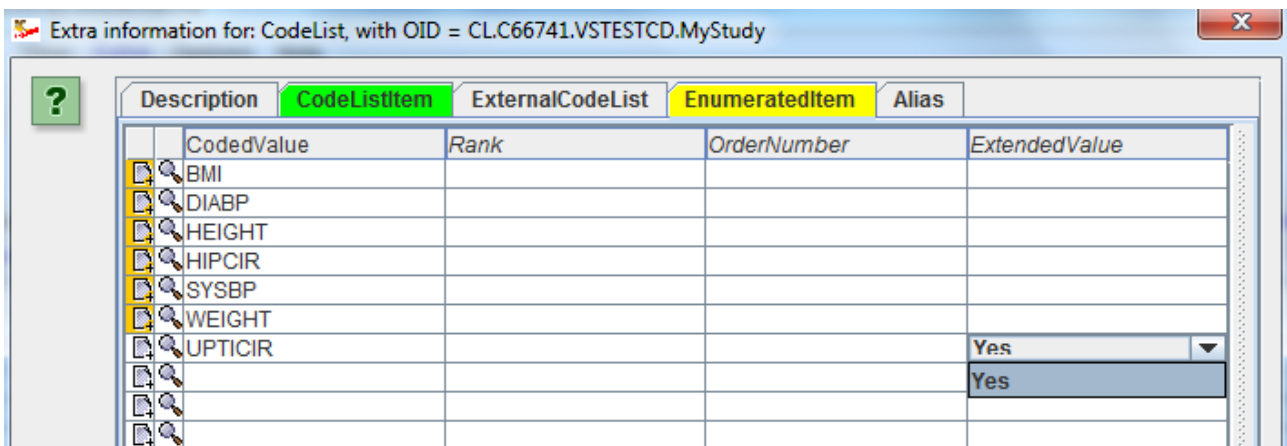
and the new codelist(s) appears in the list of all codelists:

	CL.C130270.WD01TN	World Health Organization Disabil...	text
	CL.C130273.WD7TC	World Health Organization Disabil...	text
	CL.C130272.WD7TN	World Health Organization Disabil...	text
	CL.C66741.VSTESTCD.MyStudy	Vital Signs Test Code for my study	text
	CL.C67153.VSTEST.SUBSET	Vital Signs Test Name subset	text

One can then of course still change the subset codelist, change the value for OID or Name, add new terms. For example, if one wants to extend this subset vital signs codelist with the term "Upper Tight Circumference", with a code "UPTICIR", just click the "add information" icon ("+"), leading to:

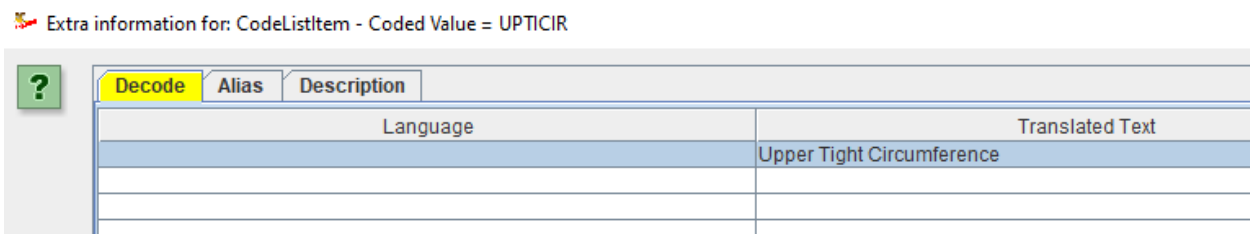


and add "UPTICIR" in a new row. As this is an extension to the CDISC codelist, you should also set the value of "ExtendedValue" to "Yes":



Do however NOT assign an NCI code to such a new entry that is not a CDISC term.

In case the codelist is an "CodeListItem" codelist, one should also add the "decode" by clicking the "+" icon and adding the information:

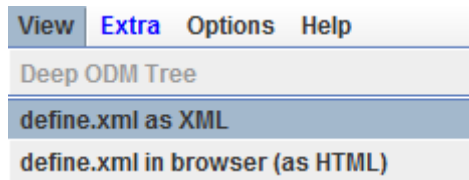


After clicking "OK", the subset codelist is extended and updated.

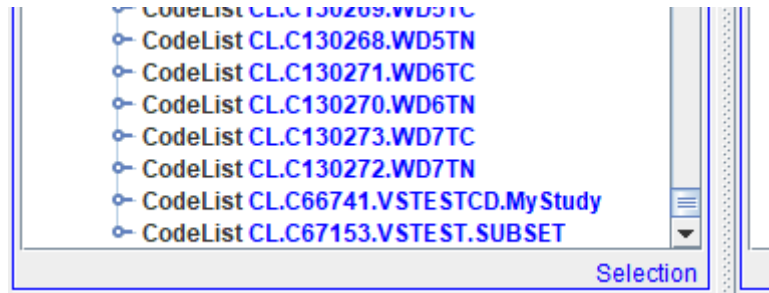
You might also now do similar for the subset codelist for VSTEST, adding "Upper Tight Circumference" as an extension.

Let us now have a quick look to the generated XML in the define.xml. This is never a bad idea if one wants to understand what one has done.

In order to do so, use the menu "View - Define.xml as XML":



and select the last "CodeList" in the list of tree nodes:



The XML is then shown on the right side:

```

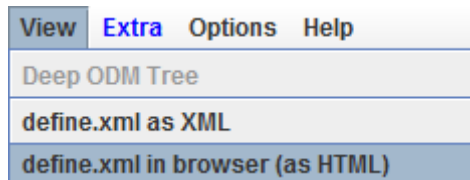
<CodeList xmlns="http://www.cdisc.org/ns/odm/v1.3"
  xmlns:def="http://www.cdisc.org/ns/def/v2.1"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  DataType="text"
  Name="Vital Signs Test Code subset"
  OID="CL.C66741.VSTESTCD.MyStudy">
  <CodeListItem CodedValue="BMI">
    <Decode>
      <TranslatedText>Body Mass Index</TranslatedText>
    </Decode>
    <Alias Context="nci:ExtCodeID" Name="C16358" />
  </CodeListItem>
  <CodeListItem CodedValue="DIABP">
    <Decode>
      <TranslatedText>Diastolic Blood Pressure</TranslatedText>
    </Decode>
    <Alias Context="nci:ExtCodeID" Name="C25299" />
  </CodeListItem>
  <CodeListItem CodedValue="HEIGHT">
    <Decode>
      <TranslatedText>Height</TranslatedText>
    </Decode>
    <Alias Context="nci:ExtCodeID" Name="C25347" />
  </CodeListItem>
  <CodeListItem CodedValue="HIPCIR">
    <Decode>
      <TranslatedText>Hip Circumference</TranslatedText>
    </Decode>
    <Alias Context="nci:ExtCodeID" Name="C100947" />
  </CodeListItem>
  <CodeListItem CodedValue="SYSBP">
    <Decode>
      <TranslatedText>Systolic Blood Pressure</TranslatedText>
    </Decode>
    <Alias Context="nci:ExtCodeID" Name="C25298" />
  </CodeListItem>
  <CodeListItem CodedValue="WEIGHT">
    <Decode>
      <TranslatedText>Weight</TranslatedText>
    </Decode>
    <Alias Context="nci:ExtCodeID" Name="C25208" />
  </CodeListItem>
  <CodeListItem CodedValue="UPTICIR" def:ExtendedValue="Yes">
    <Decode>
      <TranslatedText>Upper Tight Circumference</TranslatedText>
    </Decode>
  </CodeListItem>
  <Alias Context="nci:ExtCodeID" Name="C66741" />
</CodeList>

```

Where we see that there is an NCI code (using the "Alias" element) for each "standard" vital signs test code, whereas the extension code "UPTICIR" does not have an NCI code, but has the def:ExtendedValue="Yes" attribute. We also see that the codelist as a whole has an NCI code (C66741) from the last "Alias" element.

This is also a requirement from the CDISC standards, i.e. that when one subsets a codelist from an existing CDISC codelist, one must keep the NCI code of the codelist itself.

Or when inspected in the browser using the stylesheet (menu "View - define.xml in browser):



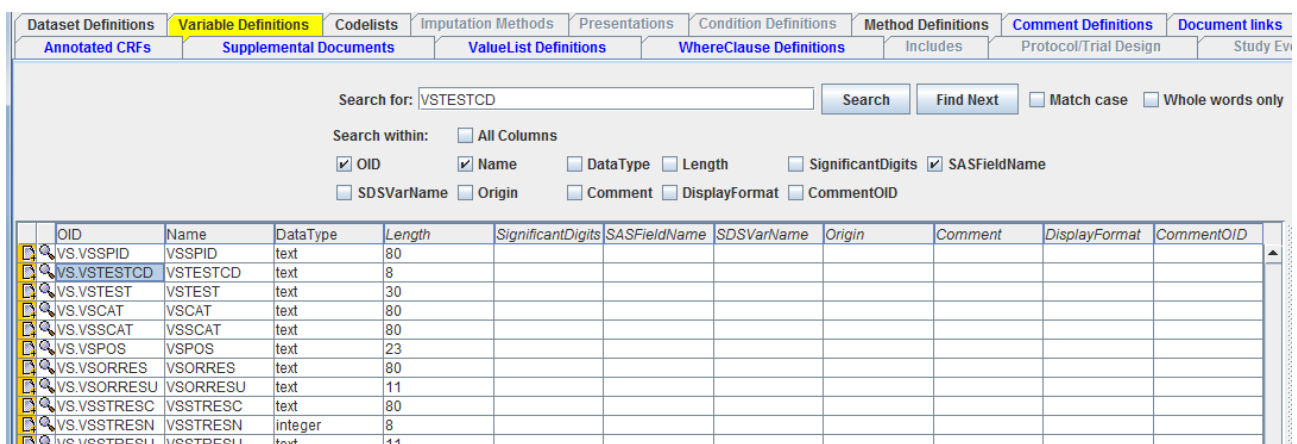
which will result that the "view" on the define.xml will be opened in your standard (favorite) browser:

Vital Signs Test Code subset [C66741]

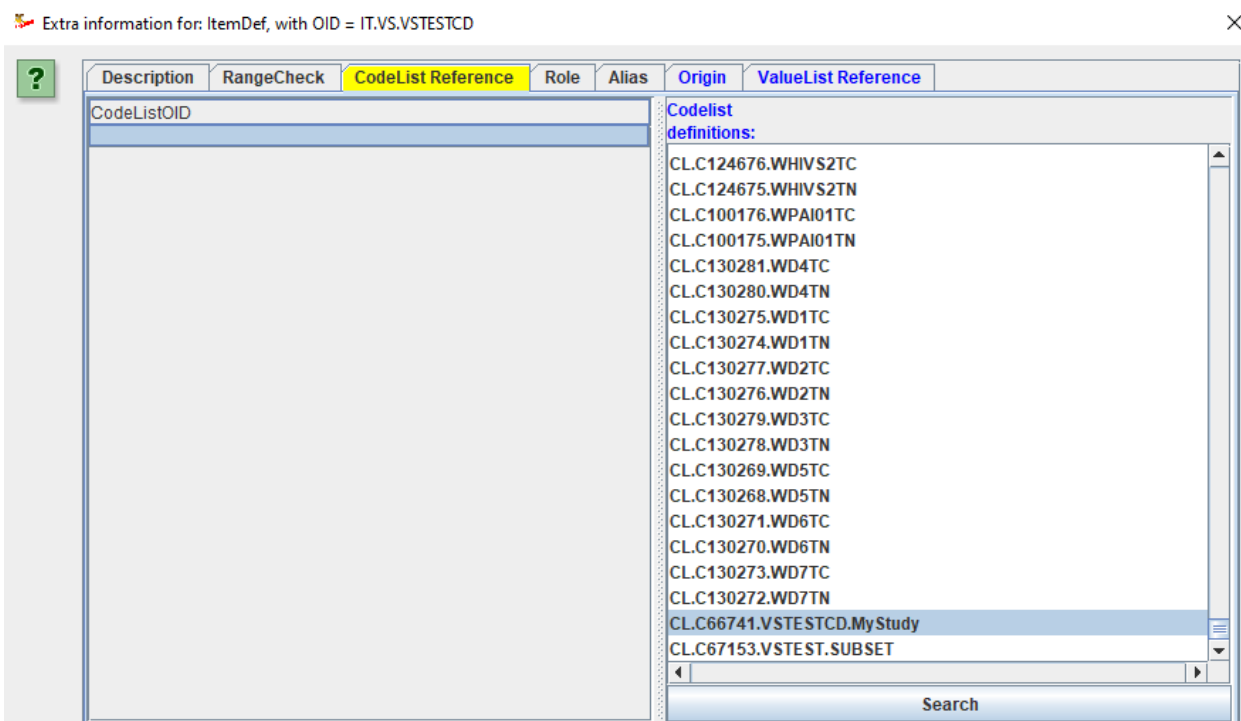
Permitted Value (Code)	Display Value (Decode)
BMI [C16358]	Body Mass Index
DIABP [C25299]	Diastolic Blood Pressure
HEIGHT [C25347]	Height
HIPCIR [C100947]	Hip Circumference
SYSBP [C25298]	Systolic Blood Pressure
WEIGHT [C25208]	Weight
UPTICIR [*]	Upper Tight Circumference

* Extended Value

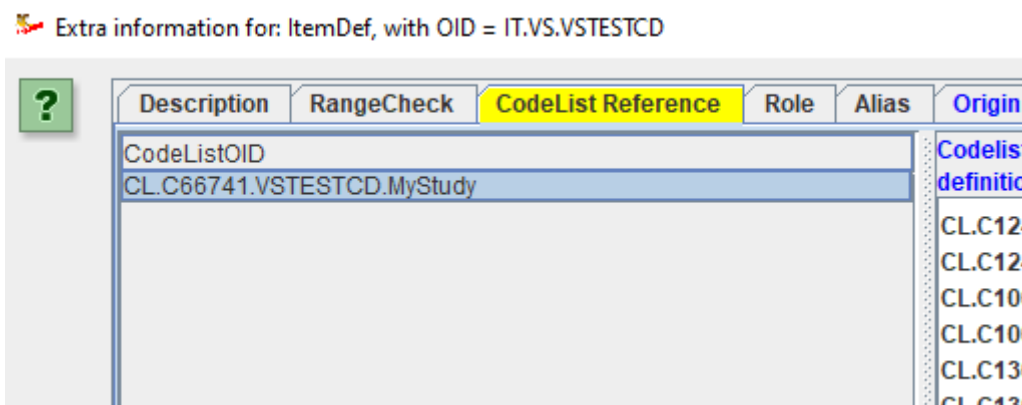
The only thing we still need to do is to assign our new subset codelist to the VSTESTCD variable. This can easily be done by selecting the "Variable Definitions" tab, selecting the VSTESTCD variable:



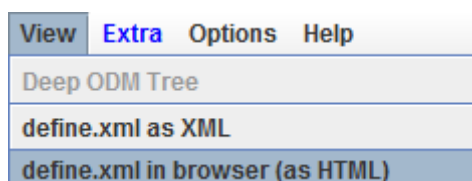
and then click the "Add Information" icon ("+") to add additional information. When the selecting the "CodeList Reference" tab:



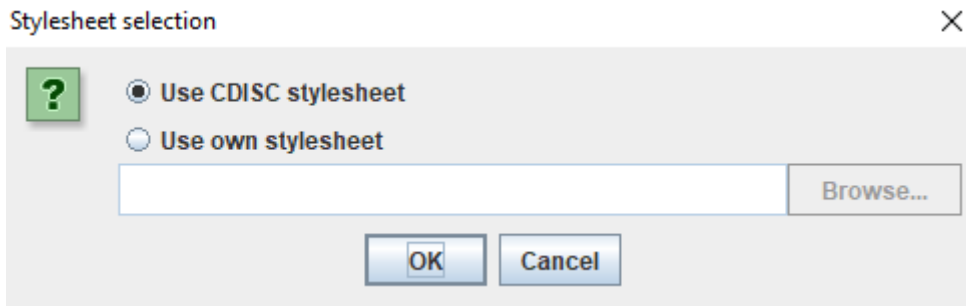
Remark that one can easily find a specific codelist by using the "Search" button.
 Now just drag-and-drop "CL.C66741.VSTESTCD.MyStudy" to the empty cell on the right:



After clicking "OK", the subset codelist "CL.C66741.VSTESTCD.MyStudy" is assigned to VSTESTCD.
 We can inspect this using the "browser view" using the menu "View - define.xml in browser",



With a "stylesheet selection" dialog showing up:



You can choose between the "standard" CDISC stylesheet (the one that comes with the [CDISC Define-XML standard package](#)), and your own or any other one. In the latter case, the "Browse" button will become available, so that you can select your own preferred stylesheet, which will then be applied.

When the "CDISC stylesheet" is selected, this is leading to:

Variable	Label	Key	Type	Length	Controlled Terms or Format
STUDYID	Study Identifier		text	40	
DOMAIN	Domain Abbreviation		text	2	
USUBJID	Unique Subject Identifier		text	60	
VSSEQ	Sequence Number		integer	8	
VSGRPID	Group ID		text	80	
VSSPID	Sponsor-Defined Identifier		text	80	
VSTESTCD	Vital Signs Test Short Name		text	8	Vital Signs Test Code for my study

and when clicking on "Vital Signs Test Code for my study":

Vital Signs Test Code for my study [CL.C66741.VSTESTCD.MyStudy, C66741]

Permitted Value (Code)
BMI [C16358]
DIABP [C25299]
HEIGHT [C25347]
HIPCIR [C100947]
SYSBP [C25298]
WEIGHT [C25208]
UPTICIR [*]

* Extended Value

with for each "Standard" code, the NCI code displayed and for the "UPTICIR" code, it being marked as a "extended" value.

In case we checked the checkbox "Also automatically subset the corresponding 'decode' CodeList ..." also the corresponding subset codelist with VSTEST terms has been created, and the only thing

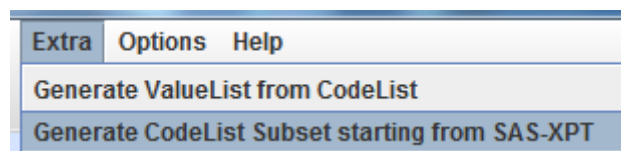
we then still need to do is to assign it to the VSTEST variable.
If we did not check the checkbox, no subset codelist with VSTEST terms has been created, but we can then do this separately, again using the menu "Extra - Generate CodeList Subset".

Adding and subsetting codelists from SAS-XPT files

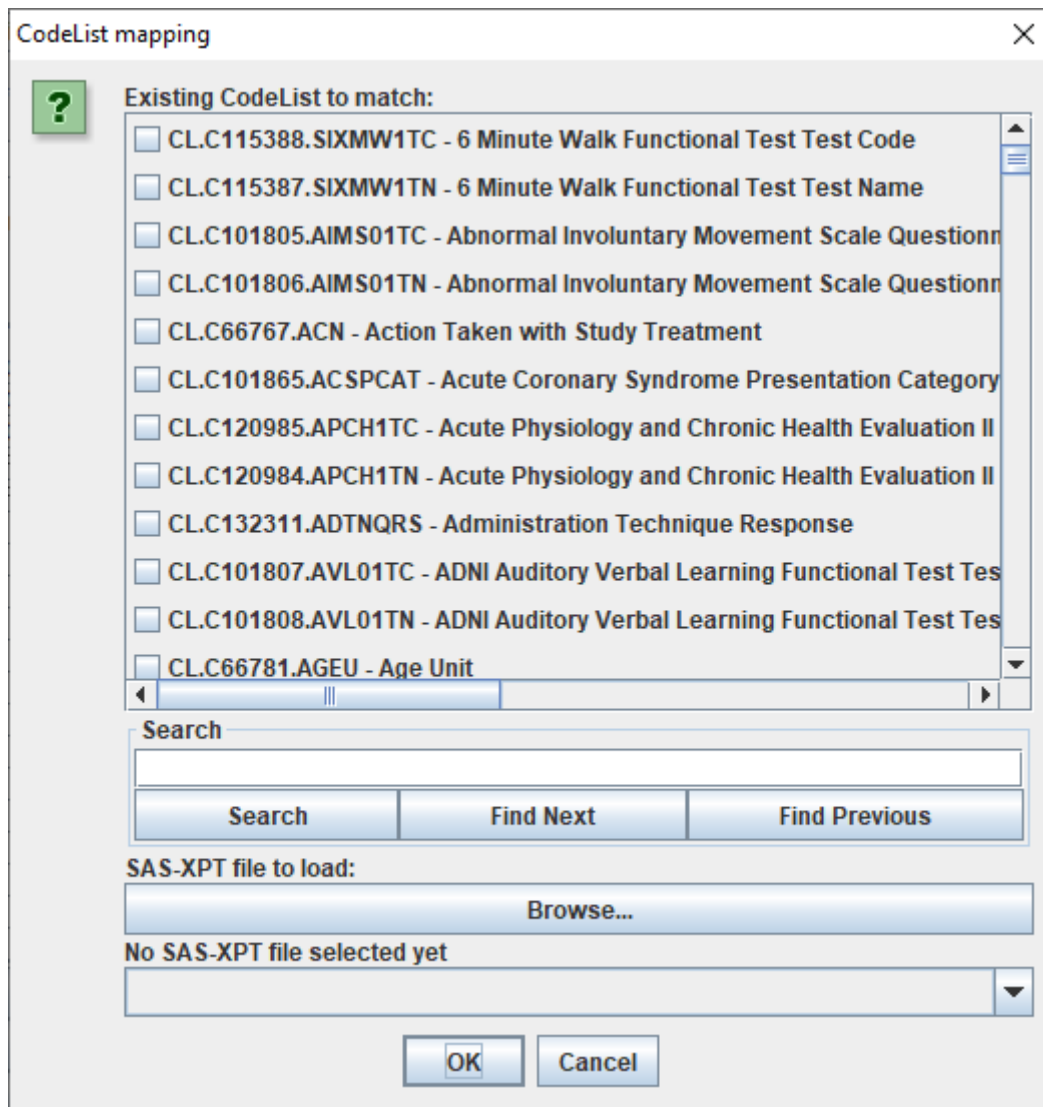
When generating a "prototype" from a set of SAS-XPT files, one has the option to have the system generate subset codelists from the information in the XPT files (see section "Creating a define.xml starting from an existing set of SAS-XPT files"). One should however not need to wait until all XPT files to start generating a define.xml file!

Also, when generating the define.xml starting from a template for a specific SDTM, SEND or ADaM version (see section "Creating a define.xml starting from an SDTM/SEND/ADaM template"), which is the better practice, it will not always be possible to generate all the subset codelists until the XPT files are final. Of course, one can often already use the information from the CRFs to generate such subset codelists, but also this is not always possible.

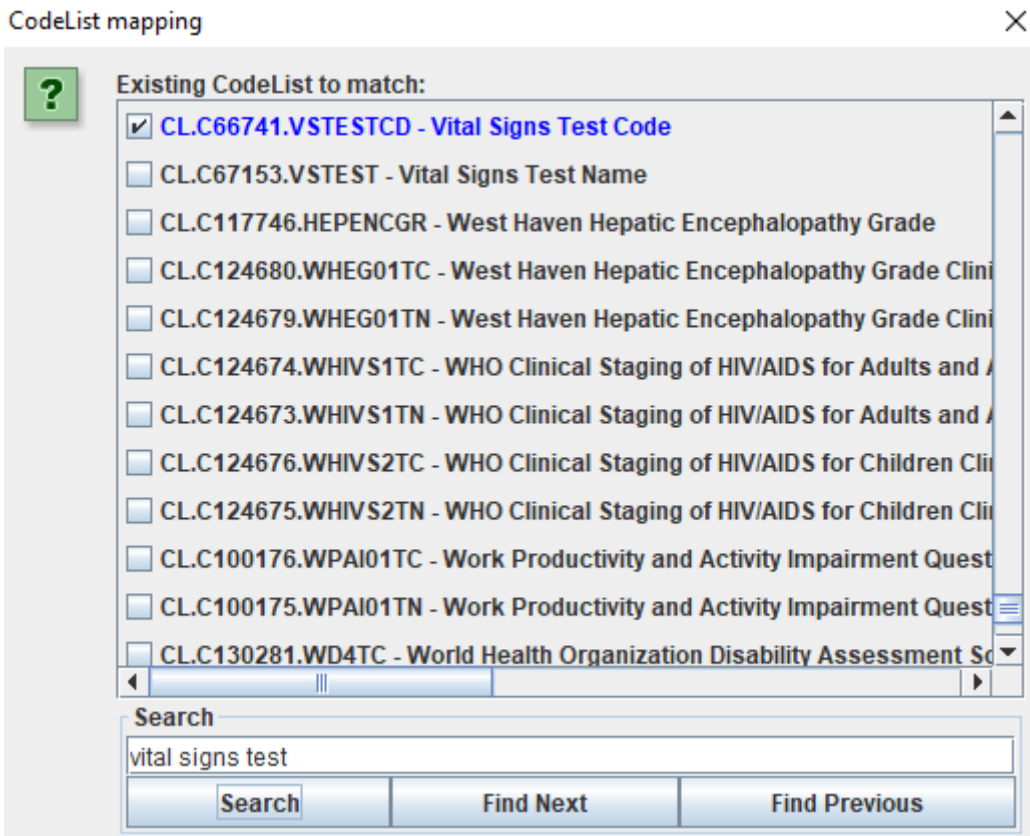
In case one wants to create a subset codelists from the contents of one or more XPT files in a later stage, one can always do so using the menu "Extra - Generate CodeList Subset starting from SAS-XPT":



The following dialog is presented:

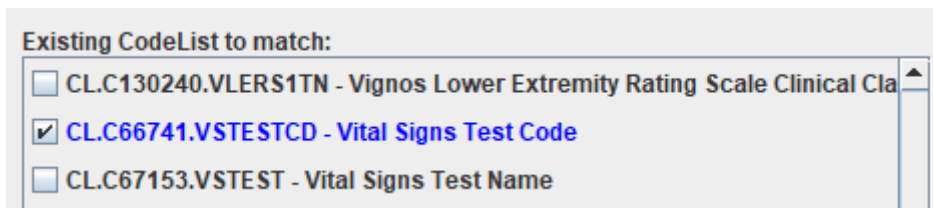


Now select a codelist you wish to subset. If one does not know the exact name, one can use the "Search" panel, for example for "vital signs test code":

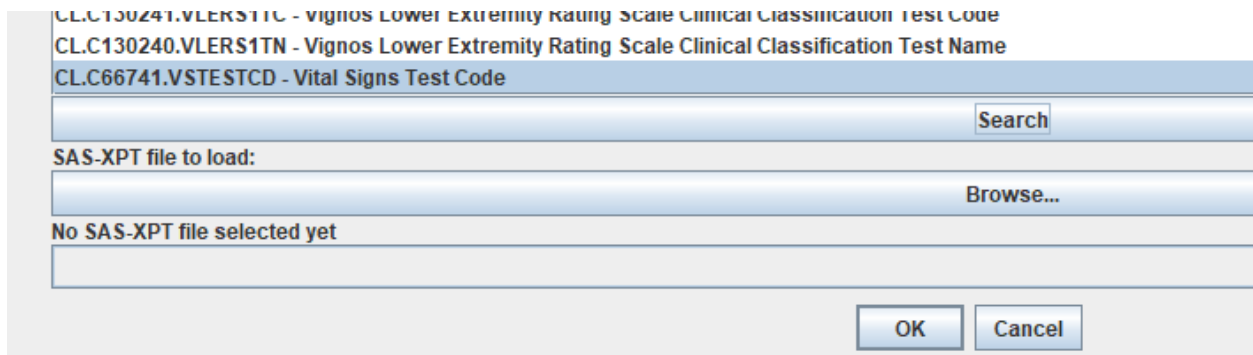


In our example, we will generate a subset for the VSTESTCD codelist based on the contents of our vs.xpt dataset:

We select the item: CL.C66741.VSTESTCD - Vital Signs Test Code

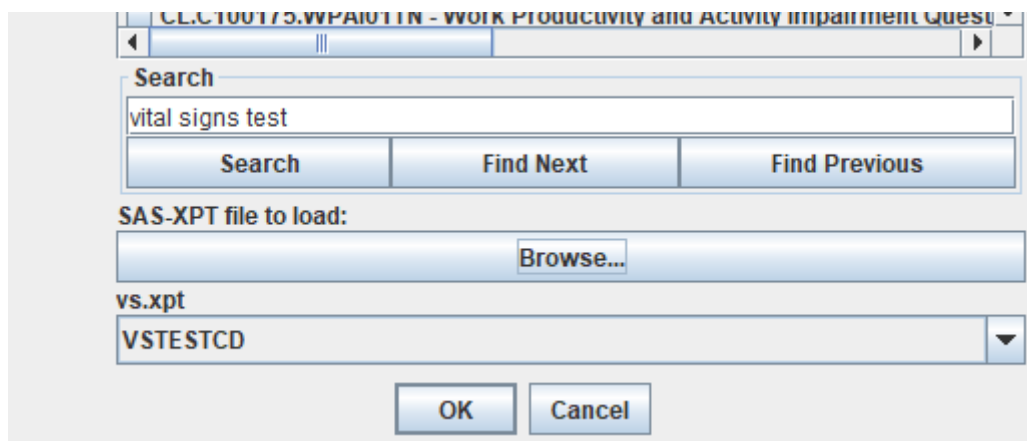


Then select the SAS-XPT file you want to get the values from, using the "Browse" button. Obviously this is the vs.xpt file in this case.



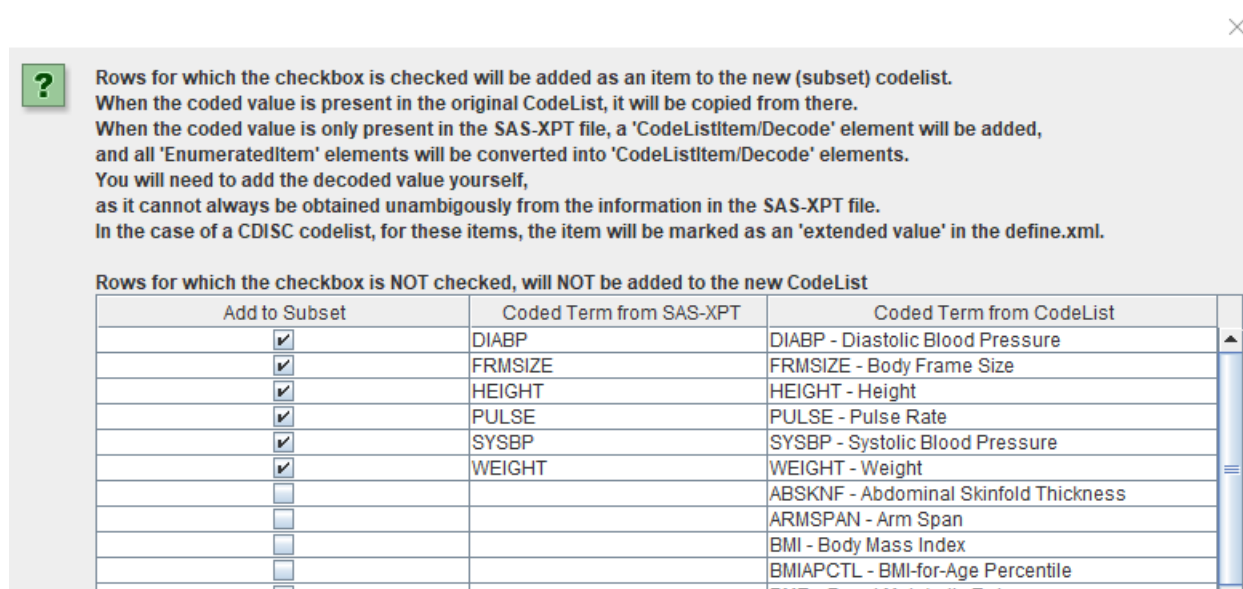
When an XPT file is selected, a dropdown to select the variable of interest will be displayed from which one can choose:

Once the file selected, the system analyzes its metadata (this takes a few seconds), and makes a proposal for the variable for which the unique values from the SAS-XPT file will be retrieved:



This choice can then still be changed using the dropdown.

When clicking "OK", the system takes all the VSTESTCD values from the SAS-XPT "vs.xpt" file and takes all the unique ones. It then tries to match them to the values from the codelist CL.C66741.VSTESTCD "Vital Signs Test Code". This leads to a new dialog:



The first column contains a checkbox, allowing the user to decide whether the coded term should be included in the subset.

The second column contains the term as found in the SAS-XPT file and the third column the coded term (and the decode when present) in the codelist. So if the term is present in as well the second as third column, there is a match between the SAS-XPT value and the already available codelist. In such a case, it is recommended to add the coded term to the subset codelist (so the checkbox is already checked).

When scrolling down:

Rows for which the checkbox is NOT checked, will NOT be added to the new CodeList			
Add to Subset	Coded Term from SAS-XPT	Coded Term from CodeList	
<input checked="" type="checkbox"/>	PULSE	PULSE - Pulse Rate	▲
<input checked="" type="checkbox"/>	SYSBP	SYSBP - Systolic Blood Pressure	
<input checked="" type="checkbox"/>	WEIGHT	WEIGHT - Weight	
<input type="checkbox"/>		ABSKNF - Abdominal Skinfold Thickness	
<input type="checkbox"/>		ARMSPAN - Arm Span	
<input type="checkbox"/>		BMI - Body Mass Index	
<input type="checkbox"/>		BMIAPCTL - BMI-for-Age Percentile	
<input type="checkbox"/>		BMR - Basal Metabolic Rate	
<input type="checkbox"/>		BODLNPTH - Body Length	
<input type="checkbox"/>		BODYFATM - Body Fat Measurement	
<input type="checkbox"/>		BIRTHWT - Birth Weight	

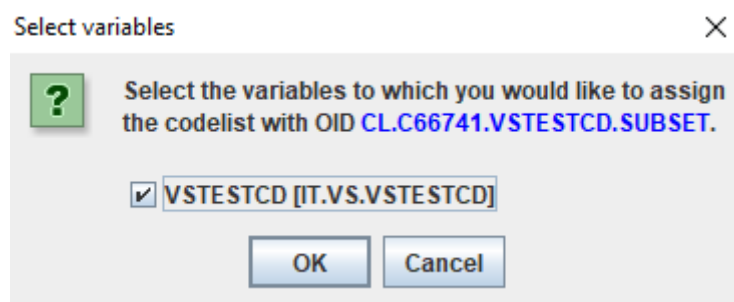
If one would find a value in the second column without a counterpart in the third column, this would mean that the value is NOT in the codelist, BUT it is in the SAS-XPT file. If the checkbox is then checked to indicate that it should be added to the subset codelist, it will then appear there as an "extended value" with an attribute 'def:ExtendedValue="Yes"'. Obviously, our dataset did not contain a value in the SAS-XPT file that was not in the codelist yet.

Suppose however that there was a field in the CRF for "BMI", but that there are no data for it (yet) in the SAS-XPT dataset. In such a case "BMI" must be added to the codelist, as it was on the CRF. So we do check the checkbox for "BMI":

Rows for which the checkbox is NOT checked, will NOT be added to the new CodeList			
Add to Subset	Coded Term from SAS-XPT	Coded Term from CodeList	
<input checked="" type="checkbox"/>	PULSE	PULSE - Pulse Rate	▲
<input checked="" type="checkbox"/>	SYSBP	SYSBP - Systolic Blood Pressure	
<input checked="" type="checkbox"/>	WEIGHT	WEIGHT - Weight	
<input type="checkbox"/>		ABSKNF - Abdominal Skinfold Thickness	
<input type="checkbox"/>		ARMSPAN - Arm Span	
<input checked="" type="checkbox"/>		BMI - Body Mass Index	
<input type="checkbox"/>		BMIAPCTL - BMI-for-Age Percentile	
<input type="checkbox"/>		BMR - Basal Metabolic Rate	
<input type="checkbox"/>		BODLNPTH - Body Length	
<input type="checkbox"/>		BODYFATM - Body Fat Measurement	

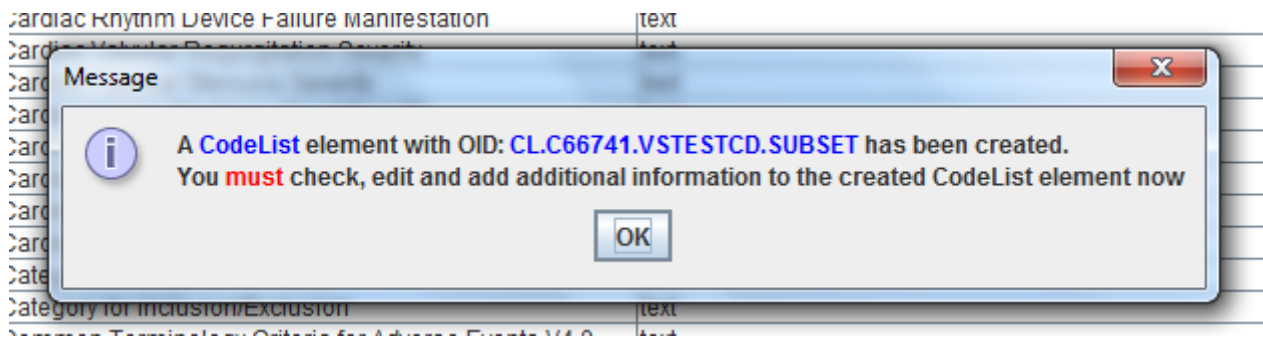
Clicking "OK" will then create a subset of the codelist. It automatically assigns the OID "CL.C66741.VSTESTCD.SUBSET". You will of course later be able to change the OID. The original codelist name is however retained.

The system then asks whether it should assign the newly generated subsetted codelist to the variable "VSTESTCD":



In some cases, a list of variables can appear here, and you can choose to which ones you want to have the newly generated codelist assigned.

This results in:



and when clicking "OK" again, the table with all the codelists will be displayed, containing a new row for the subset codelist:

CL.C66741.VSTESTCD	Vital Signs Test Code	text
CL.C67153.VSTEST	Vital Signs Test Name	text
CL.C117746.HEPENCGR	West Haven Hepatic Encephalopathy Grade	text
CL.C66741.VSTESTCD.SUBSET	Vital Signs Test Code	text

One can now still edit this codelist, add terms, remove terms, change the name, etc..

Clicking the "magnifying glass" icon gives an overview:

Contents of element CodeList

Contents of CodeList with OID CL.C66741.VSTESTCD.SUBSET and with Name Vital Signs Test Code

Attributes:

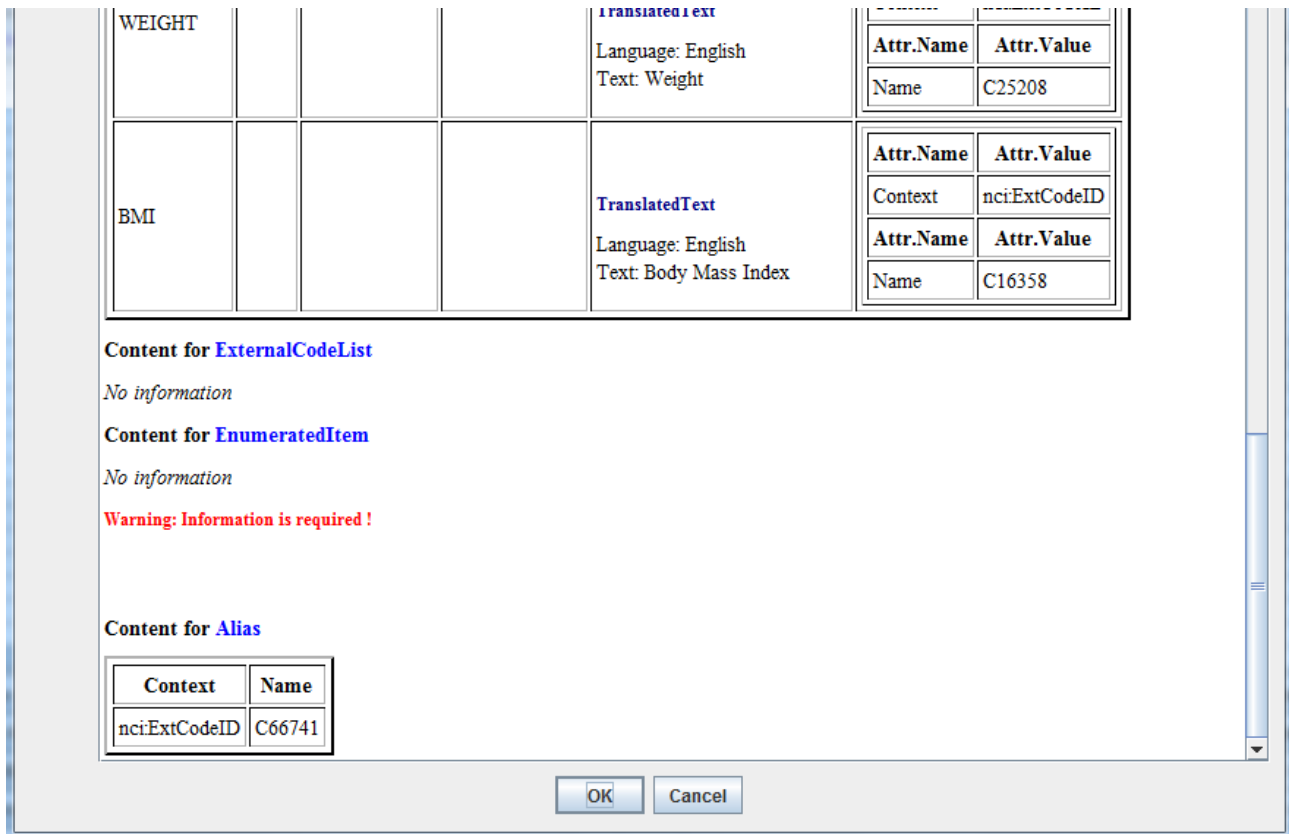
Name	Value
OID	CL.C66741.VSTESTCD.SUBSET
Name	Vital Signs Test Code
DataType	text
SASFormatName	

Content for Description
No information

Content for CodeListItem

CodedValue	Rank	OrderNumber	ExtendedValue	Decode	Alias								
DIABP				TranslatedText Language: English Text: Diastolic Blood Pressure	<table border="1"> <thead> <tr> <th>Attr.Name</th> <th>Attr.Value</th> </tr> </thead> <tbody> <tr> <td>Context</td> <td>nci:ExtCodeID</td> </tr> <tr> <th>Attr.Name</th> <th>Attr.Value</th> </tr> <tr> <td>Name</td> <td>C25299</td> </tr> </tbody> </table>	Attr.Name	Attr.Value	Context	nci:ExtCodeID	Attr.Name	Attr.Value	Name	C25299
Attr.Name	Attr.Value												
Context	nci:ExtCodeID												
Attr.Name	Attr.Value												
Name	C25299												

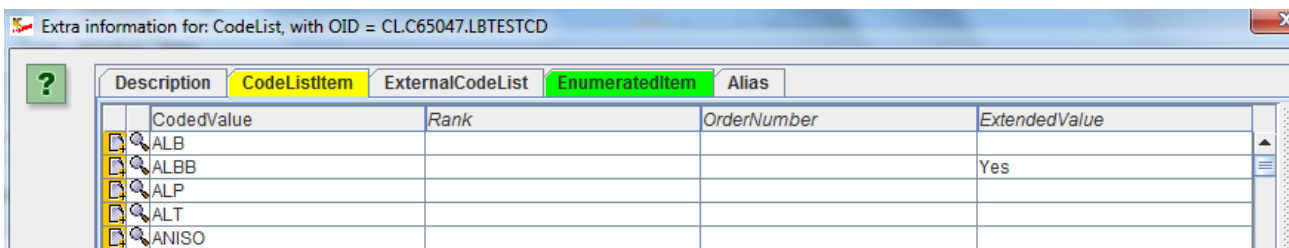
also containing "BMI":



Remark that also the "Alias" child elements, containing the NCI code, were copied from the original codelist.

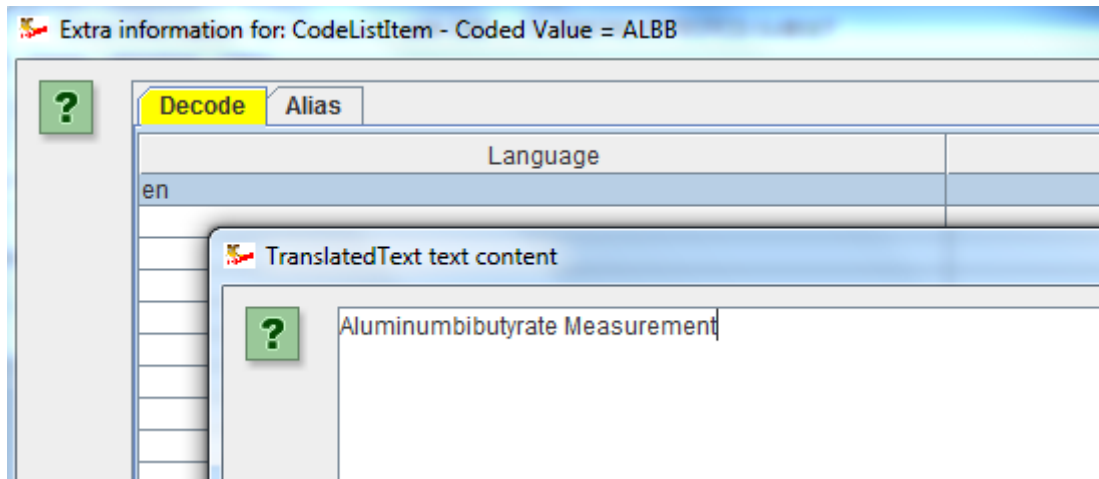
Suppose now that we also subsetting the "LBTESTCD" codelist, and that the SAS-XPT file contains an LBTESTCD value "ALBB". It might be that this is a mapping error: in that case one would need to re-generate the lb.xpt file.

If, however, it was not an error, but "ALBB" corresponds to "Aluminumbityrate Measurement", one would find the following list of coded items in the codelist:

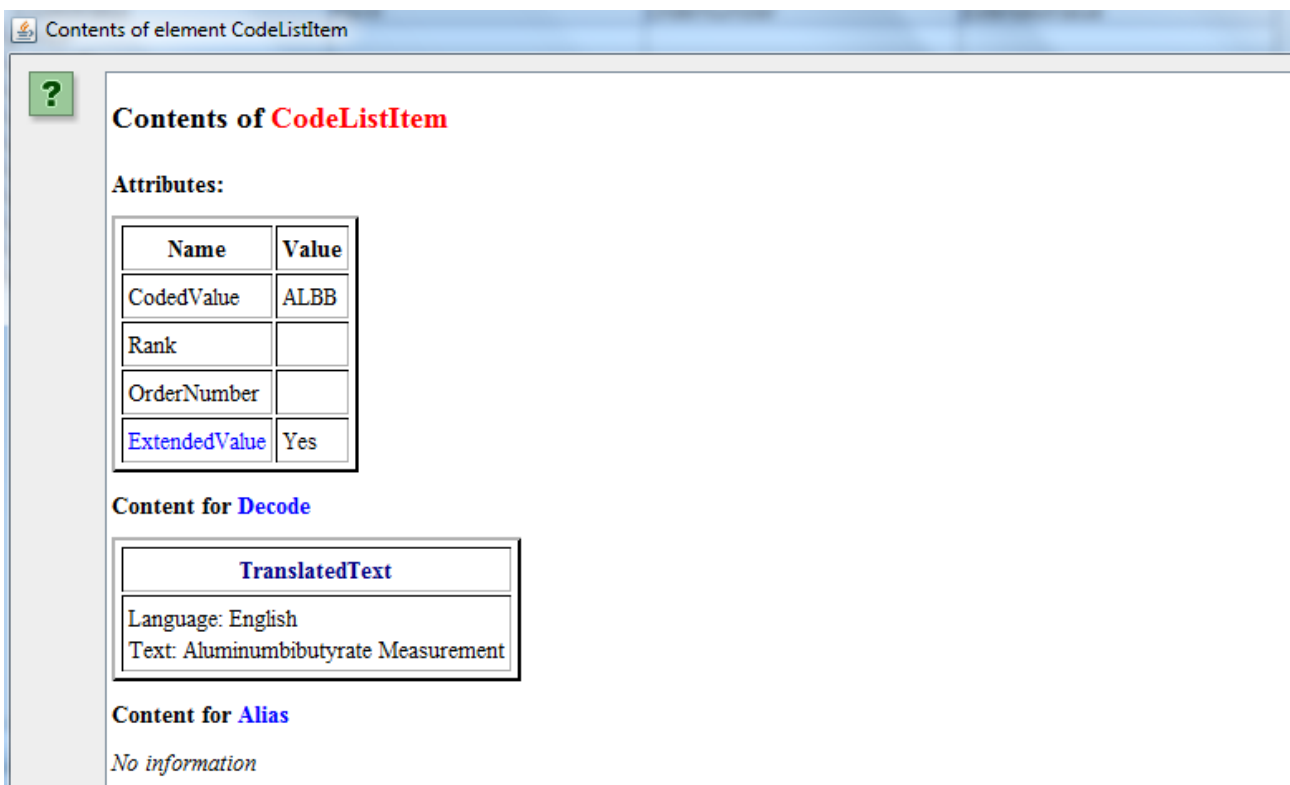


Remark that it is automatically marked as "extended".

One would still need to add the "decoded value". This can be done using the "+" icon, and adding the text to the "Decode" element:



Clicking the magnifying glass icon in the main table then displays:



Remark the "ExtendedValue" with value "Yes" in the table of attributes, and that there is no "Alias" element with "nci:ExtCodeID", as this term is not in the CDISC controlled terminology. You can of course always do a "new term request" to the CDISC "controlled terminology team", but this may take a few months until it is approved. When it is, you can then add the NCI code, and remove the "extended" marker.

Important here is that **the user is always in control**, and not the system, as in some other "black box" tools starting from Excel worksheets.

Looks good isn't it?

Transforming an "EnumeratedItem" CodeList into a "CodeListItem" CodeList

Many of the CDISC controlled terminology is published as an "Enumerated" CodeList, i.e. all values are published without description and without any translations into other languages. For example:

```
<CodeList OID="CL.C102578.DSSOUT" Name="Disease Outcome" DataType="text">
  <EnumeratedItem CodedValue="BACTERIOLOGICAL CURE">
    <Alias Context="nci:ExtCodeID" Name="C102600"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="BACTERIOLOGICAL FAILURE">
    <Alias Context="nci:ExtCodeID" Name="C102601"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="CLINICAL CURE">
    <Alias Context="nci:ExtCodeID" Name="C102607"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="CLINICAL FAILURE">
    <Alias Context="nci:ExtCodeID" Name="C102608"/>
  </EnumeratedItem>
  <EnumeratedItem CodedValue="RECURRENT DISEASE">
    <Alias Context="nci:ExtCodeID" Name="C38155"/>
  </EnumeratedItem>
  <Alias Context="nci:ExtCodeID" Name="C102578"/>
</CodeList>
```

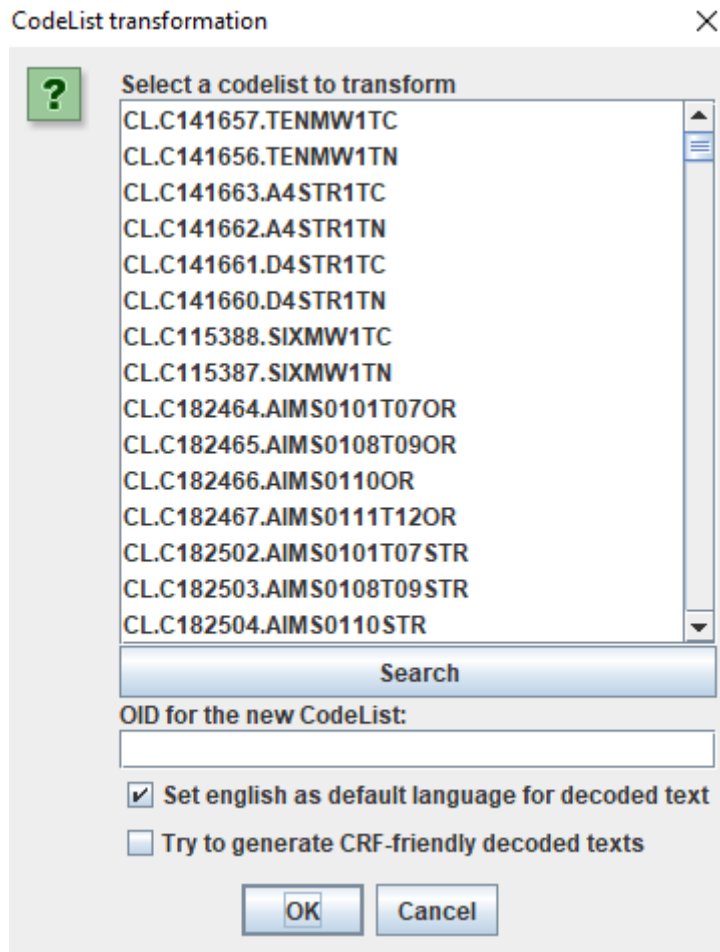
Usually, the "coded value" is written uppercase, which is not very useful for populating a CRF. Also, in the case of electronic submissions to regulatory authorities in non-English speaking countries (Japan, China), sponsors will usually submit the English term as well as the term in the local language.

So we will want to transform such an "Enumerated" CodeList into a classic CodeList with a "Decode" and "TranslatedText" elements, like in:

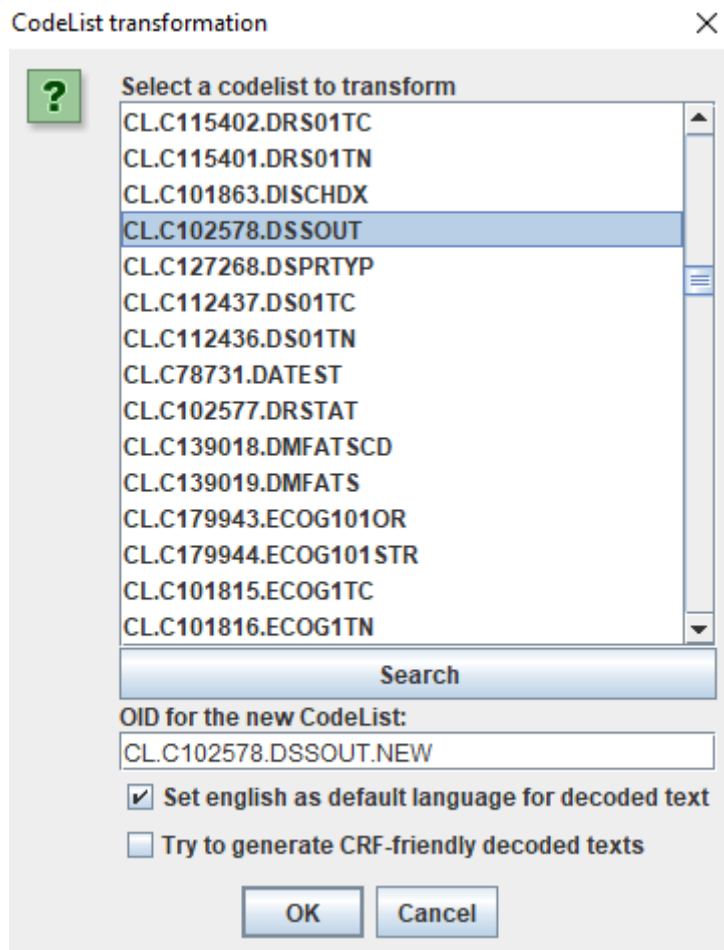
```
<CodeList OID="CL.C102578.DSSOUT" Name="Disease Outcome" DataType="text">
  <CodeListItem CodedValue="BACTERIOLOGICAL CURE">
    <Decode>
      <TranslatedText xml:lang="en">Bacteriological Cure</TranslatedText>
      <TranslatedText xml:lang="ja">細菌学的治癒</TranslatedText>
    </Decode>
    <Alias Context="nci:ExtCodeID" Name="C102600"/>
  </CodeListItem>
</CodeList>
```

Our system has a feature to do such a transformation automatically. The result will initially always have the value for Decode/TranslatedText being the same value as for "CodedValue", but the user will then be able to change this and add translations for additional languages.

In order to do so, use the menu "Extra - Generate CodeListItem-CodeList from EnumeratedItem-CodeList". This will then present the use with a list of all available codelists that have at least one "EnumeratedItem" child element:



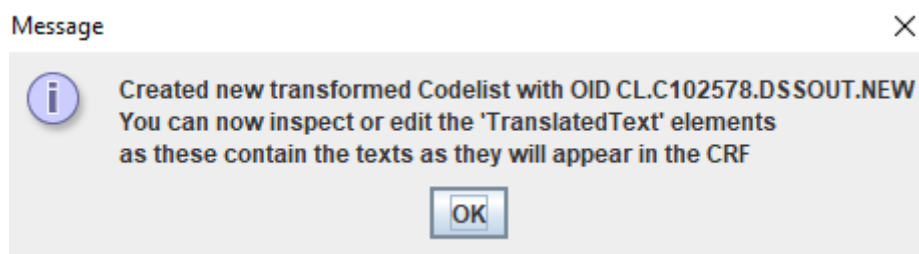
Then select the codelist for which you want to create Decode/TranslatedText elements, allowing you to change the way this appears on a CRF and/or you want to add translations. For example:



Checking "Set English as default language" will make it clearer later which language is meant when adding translations for other languages.

When checking "Try to generate CRF-friendly decoded texts" the system will try to generate camel-case "decoded" texts, which e.g. can be used in CRFs¹⁸.

After clicking the "OK" button, the following message is displayed:



and the codelist is added at the bottom to the table with the codelists.

If you do not want the old codelist anymore, but only want to work with the new one, you can delete the old one and rename the new one.

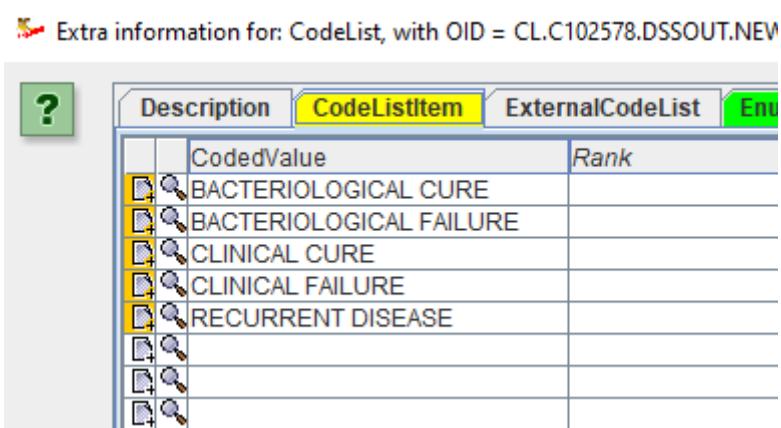
The table is displayed automatically:

¹⁸ One of the use cases of define.xml is to develop CRFs that are obeying the SDTM requirements of the sponsor.

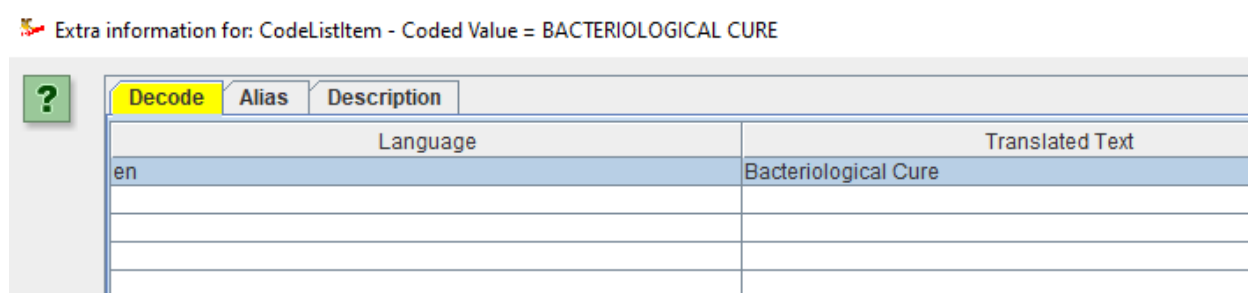
CL.C66741.VSTESTCD	Vital Signs Test Code	text
CL.C67153.VSTEST	Vital Signs Test Name	text
CL.C117746.HEPENCGR	West Haven Hepatic Encephalopathy Grade	text
CL.C102578.DSSOUT.NEW	Disease Outcome	text

Add Row	Delete Selected Row
Move Selected Row Up	Move Selected Row Down

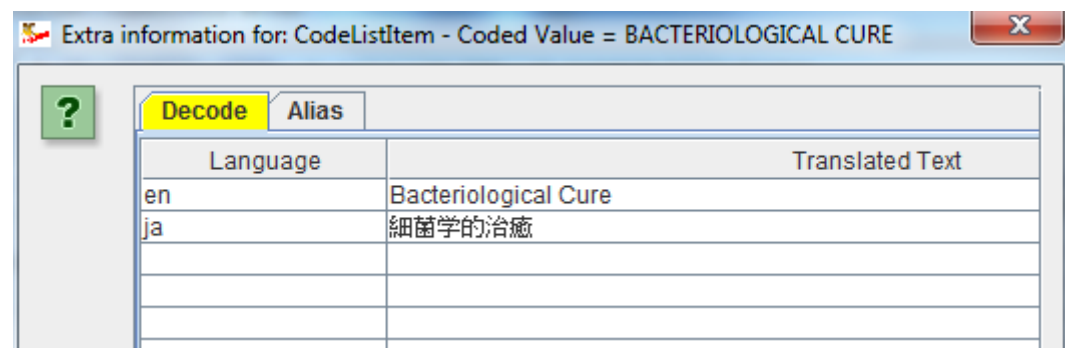
When clicking the "+" icon, one sees that the child elements are "CodeListItem" elements now instead of "EnumeratedItem" items:



and one can start adding translations into other languages. For example, for "BACTERIOLOGICAL CURE", we find for the "decode":



which is "camel case".
and e.g. add the Japanese translation, leading to:



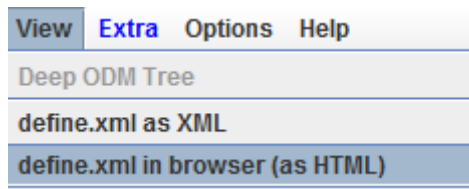
The "CodedValue" (asto be used in a database) however remains "BACTERIOLOGICAL CURE".

Displaying your define.xml as HTML in a browser using a stylesheet

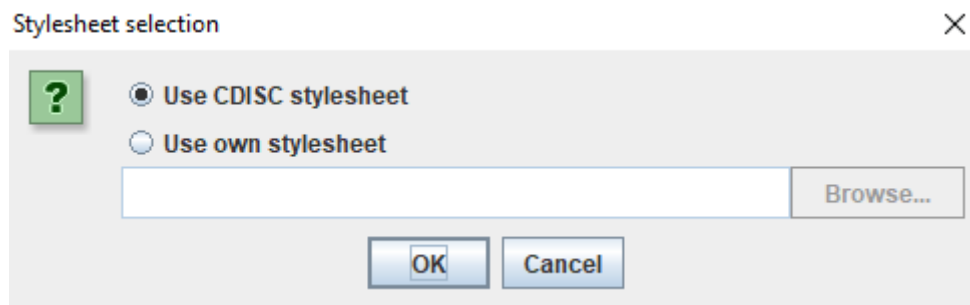
Many people (unfortunately) know define.xml only from what they see in the browser. What is seen in the browser is however only a particular view on the information in the define.xml provided by a stylesheet¹⁹. When submitting the define.xml to the FDA or the PMDA, it is the sponsors duty²⁰ to also provide a stylesheet so that the reviewers can see the information in a user-friendly way.

Most sponsors use the stylesheet that is provided by CDISC. Sponsors are however also free to develop their own stylesheet (or alter the one from CDISC). This however requires a good amount of XSLT knowledge.

Also from within our software, one can already obtain a view of the developed define.xml in the browser by applying the stylesheet. To do so, use the menu "View - define.xml in browser":



You will then be asked whether you want to use the default [CDISC stylesheet](#), or your own or other favorite one:



If you select "Use own stylesheet", you will need the "Browse" button to select a stylesheet file from your local file system.

Then click OK, and the define.xml will be displayed (as HTML of course) in your default²¹ (favorite) browser. For example:

¹⁹ Technically, stylesheets can even be used to manipulate the contents of the XML!

²⁰The FDA would better forbid that the stylesheet is provided by the sponsor, and use their own one. This would ensure that all studies are displayed in the same, sponsor-independent way.

²¹The system recognizes what your default browser is.

Study Name My own study
Study Description Description of my own study
Protocol Name Protocol name of my own study
Metadata Name MetaData for SDTMIG 3.4 for My own study
Metadata Description MetaData for SDTMIG 3.4 for My own study Version 1.0

Standards for Study My own study

Standard	Type	Status	Documentation
SDTMIG 3.4	IG	Final	
CDISC/NCI SDTM 2022-06-24	CT	Final	

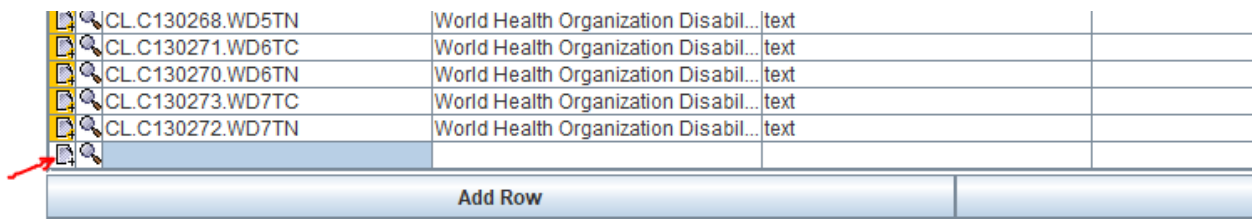
Datasets

Dataset	Description	Class	Structure	Purpose	Keys	Documentation	Location
DM [SDTMIG 3.4]	Demographics	SPECIAL PURPOSE		Tabulation			dm.xml
SE [SDTMIG 3.4]	Subject Elements	SPECIAL PURPOSE		Tabulation			se.xml
SV [SDTMIG 3.4]	Subject Visits	SPECIAL PURPOSE		Tabulation			sv.xml
CM [SDTMIG 3.4]	Concomitant/Prior Medications	INTERVENTIONS		Tabulation			cm.xml
EX [SDTMIG 3.4]	Exposure	INTERVENTIONS		Tabulation			ex.xml
AE [SDTMIG 3.4]	Adverse Events	EVENTS		Tabulation			ae.xml
DS [SDTMIG 3.4]	Disposition	EVENTS		Tabulation			ds.xml
MH [SDTMIG 3.4]	Medical History	EVENTS		Tabulation			mh.xml
EG [SDTMIG 3.4]	ECG Test Results	FINDINGS		Tabulation			eg.xml
IE [SDTMIG 3.4]	Inclusion/Exclusion Criteria Not Met	FINDINGS		Tabulation			ie.xml
LB [SDTMIG 3.4]	Laboratory Test Results	FINDINGS		Tabulation			lb.xml
PE [SDTMIG 3.4]	Physical Examination	FINDINGS		Tabulation			pe.xml
QSCG [SDTMIG 3.4]	Questionnaires	FINDINGS		Tabulation			qscg.xml
QSCS [SDTMIG 3.4]	Questionnaires	FINDINGS		Tabulation			qscs.xml
QSM [SDTMIG 3.4]	Questionnaires	FINDINGS		Tabulation			qsm.xml

Creating and populating Sponsor-defined Codelists

You probably also want to generate some sponsor-defined codelists. Sponsor-defined codelists in Define-XML are not different from CDISC codelists except that they do not have a CDISC-NCI code (as provided in an "Alias" element).

In order to generate a new codelist, either navigate to the "CodeLists" panel and click the button "Add Row", or use the menu "Add - CodeList Definition". This will add a new, empty row at the bottom of the CodeList panel:



Now assign an OID (identifier), and Name for the codelist, e.g.:



and select the value for the data type (usually "text") from the dropdown:

	CL.C130270.WD6TN	World Health Organization Disability A...	text	
	CL.C130273.WD7TC	World Health Organization Disability A...	text	
	CL.C130272.WD7TN	World Health Organization Disability A...	text	
	CL.ARMCD	Sponsor-defined codelist for ARMCD	text	
Add Row				

You can now adding some terms to your codelist by clicking on the "+" icon. This opens a new dialog:

Extra information for: CodeList, with OID = CL.ARMCD

?	Description		CodeListItem	ExternalCodeList	EnumeratedItem	Alias
	Language			Translated Text		

On the tab "Description", you can now add a description of the codelists, even in different languages. You will now also need to decide whether you just want to add a list of the codes without explanation ("enumerated items") or a list of terms with explanations ("codelist items" with "decodes"). In most cases you will want to add explanations to the coded values. In that case, select the tab "CodeListItem". This leads to:

Extra information for: CodeList, with OID = CL.ARMCD

?	Description		CodeListItem	ExternalCodeList	EnumeratedItem	Alias
		CodedValue	Rank	OrderNumber	ExtendedValue	
						
						
						
						
						

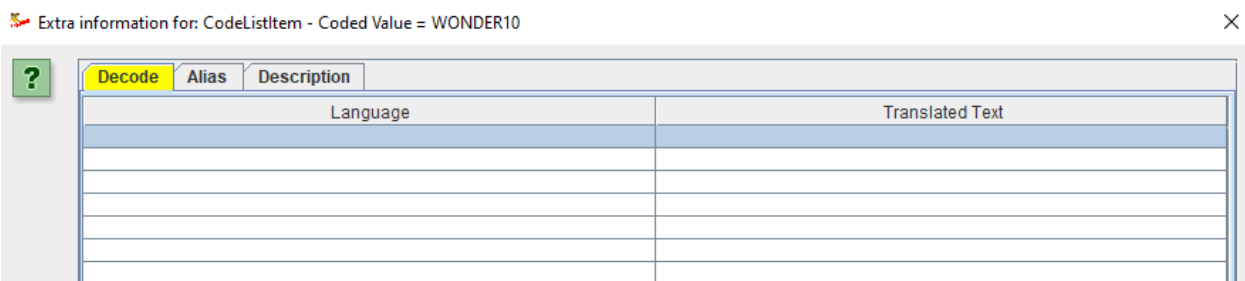
Remark that the selected tab is always displayed in yellow.

You can now add codes, in our case for "ARMCD" (trial arm code). Let us already add a few ones:

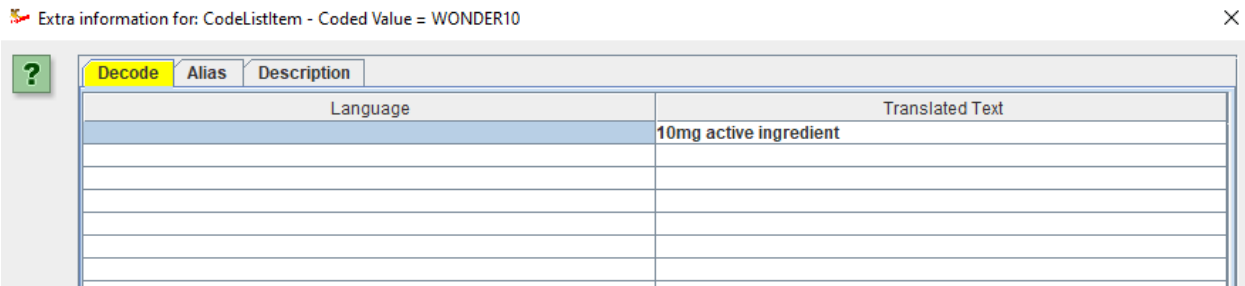
Description		CodeListItem	ExternalCodeList	EnumeratedItem	Alias
	PLACEBO				
	WONDER10				
	WONDER20				
					
					

But what do "WONDER10" and "WONDER20" mean? Best practice is to provide an explanation by using the "decode" of Define-XML.

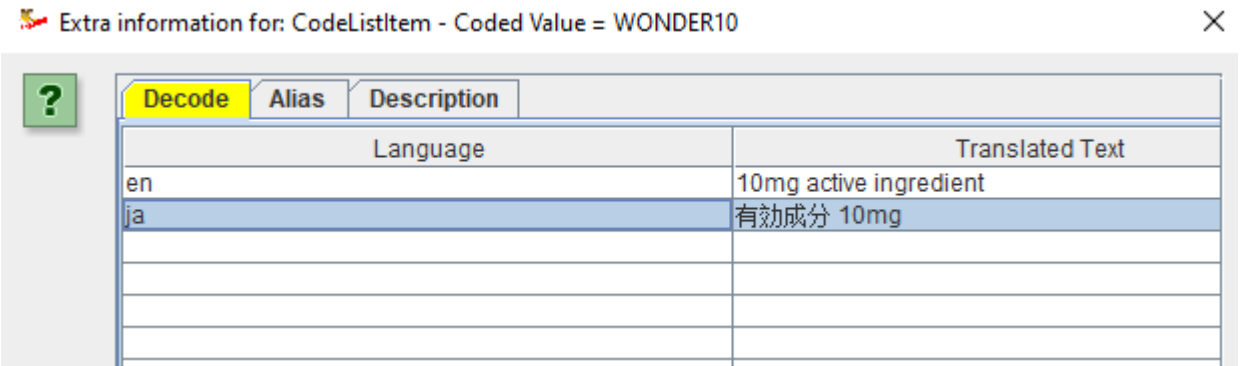
For doing so, click on the "+" icon, e.g. for the row with "WONDER10". This opens a new dialog:



Now add the explanation in the field "TranslatedText", e.g.:



You can do so in different languages, but then also need to provide the language. For example:



Then do also for "PLACEBO" and "WONDER20".

You will usually not need to populate "Rank", unless you want to add an order for importance, like in "mild - moderate - severe". Also remind that "OrderNumber" is only about the order of display, which is something completely. If you just want to have the display order the same as how you added the items, there is no need to populate "OrderNumber". In the case of a sponsor-defined codelist, also leave "ExtendedValue" untouched, as you do not extend a CDISC codelist here.

When inspecting the complete codelist (by clicking the "magnifying glass" icon), this may then look like:

Contents of CodeList with OID CL.ARMCD and with Name Sponsor-defined codelist for ARMCD

Attributes:

Name	Value
OID	CL.ARMCD
Name	Sponsor-defined codelist for ARMCD
DataType	text
SASFormatName	
StandardOID	
IsNonStandard	
CommentOID	

Content for Description
No information

Content for CodeListItem

CodedValue	Rank	OrderNumber	ExtendedValue	Decode	Alias	Description
PLACEBO				TranslatedText Language: English Text: Placebo TranslatedText Language: Japanese Text: プラセボ		
WONDER10				TranslatedText Language: English Text: 10mg active ingredient TranslatedText Language: Japanese Text: 有効成分 10mg		
WONDER20				TranslatedText Language: English Text: 20mg active ingredient TranslatedText		

OK Cancel

In the case of Define-XML 2.1, you will still want to state that this is a "non-standard" (i.e. non-CDISC codelist), which you can do by using the dropdown for "IsNonStandard":

OID	Name	DataType	SASFormatName	StandardOID	IsNonStandard	CommentOID
CL.ARMCD	Sponsor-defined codelist for ARMCD	text			Yes	
CLC154449.UHDR1TC	Unified Huntington's Disease Rating Scale '9...	text		STD.SDTM.CDISC-NCL_2022-06-24	Yes	
CLC154448.UHDR1TN	Unified Huntington's Disease Rating Scale '9...	text		STD.SDTM.CDISC-NCL_2022-06-24		

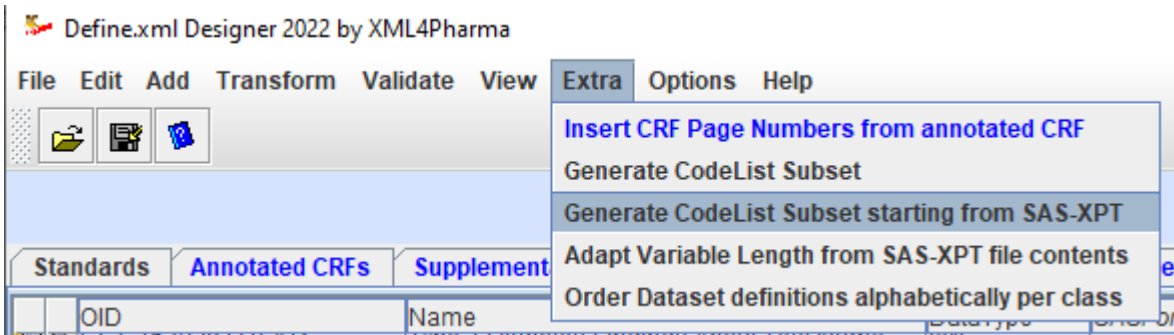
Remark that in Define-XML 2.1 "StandardOID" and "IsNonStandard" are "mutually exclusive", i.e. it is not allowed to have both set.

Suppose now that you have already created your SAS-XPT files, and want to have the sponsor-defined codelist populated with the values from a SAS-XPT file.

In order to do so, create the sponsor-defined codelist, but do now add any terms to it. For example:

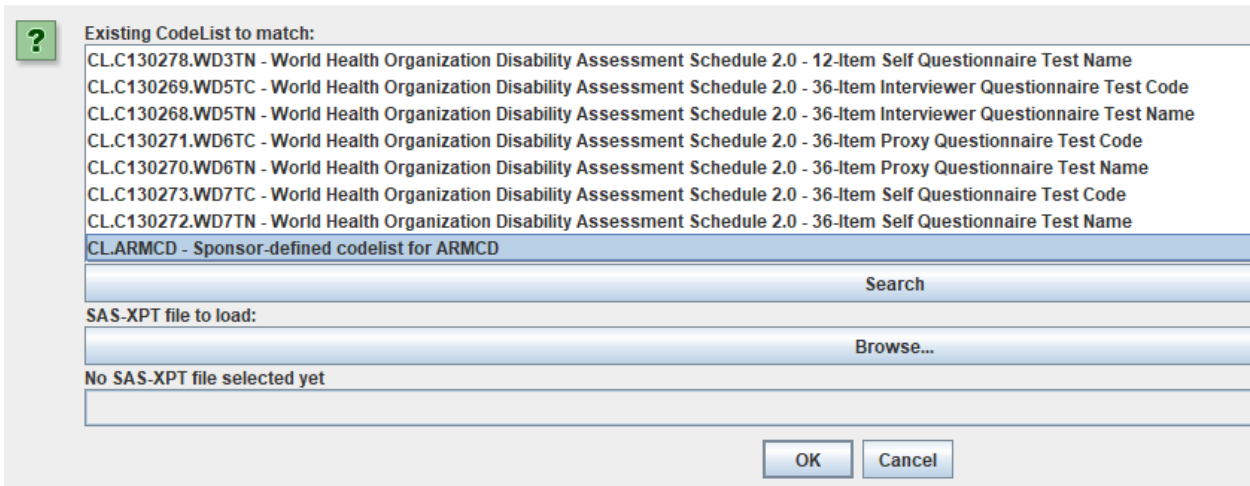
	CL.C130270.WD6TN	World Health Organization Disability A...	text	
	CL.C130273.WD7TC	World Health Organization Disability A...	text	
	CL.C130272.WD7TN	World Health Organization Disability A...	text	
	CL.ARMCD	Sponsor-defined codelist for ARMCD	text	
Add Row				

One sees that the "+" icon is not colored, meaning that there is no underlying information. If you e.g. want to populate the codelist from the values for ARMCD in your SAS-XPT file "dm.xpt", use the menu "Extra - Generate CodeList Subset starting from SAS-XPT":



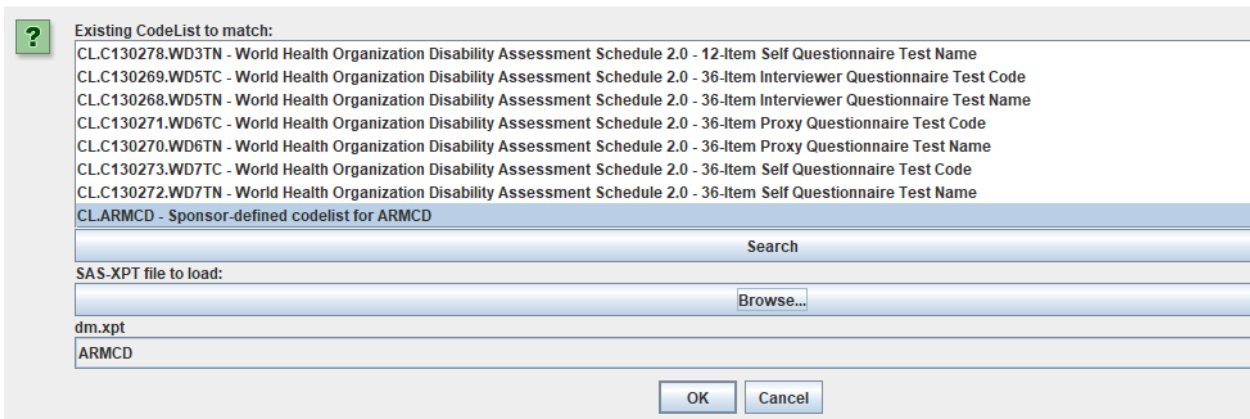
and in the dialog that is displayed, select your newly generated (but still empty) codelist:

CodeList mapping



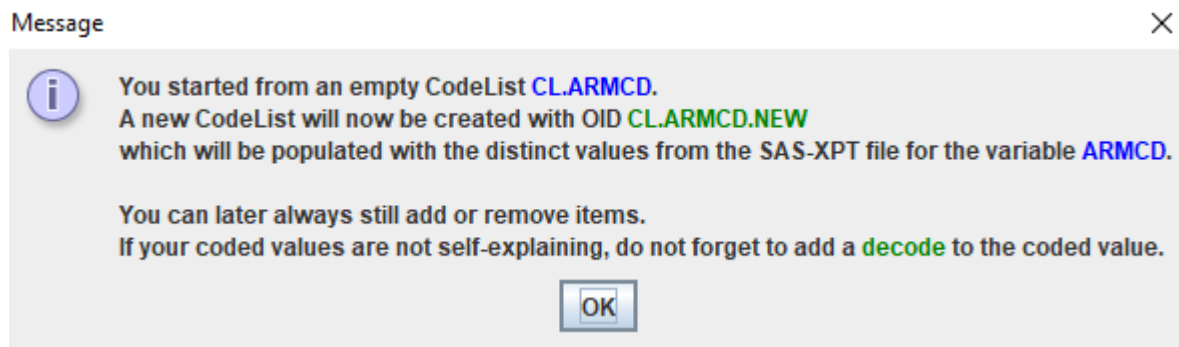
Then select your dm.xpt file using the "Browse.." button. For example leading to:

CodeList mapping



One sees that "ARMCD" is automatically selected as the SAS-XPT variable to retrieve from, but you can of course still change that by using the dropbox.

After clicking "OK", the SAS file is being analyzed, and a message is displayed:

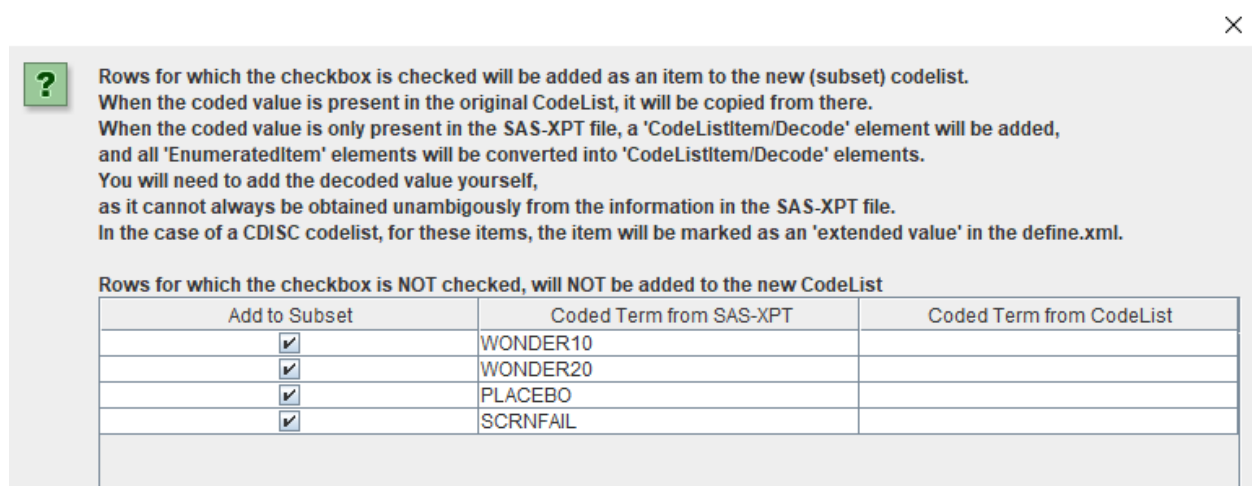


It states that a new codelist will be generated with the OID "CL.ARMCD.NEW" and which will be populated with the distinct values from the ARMCD variable in the dm.xpt file.

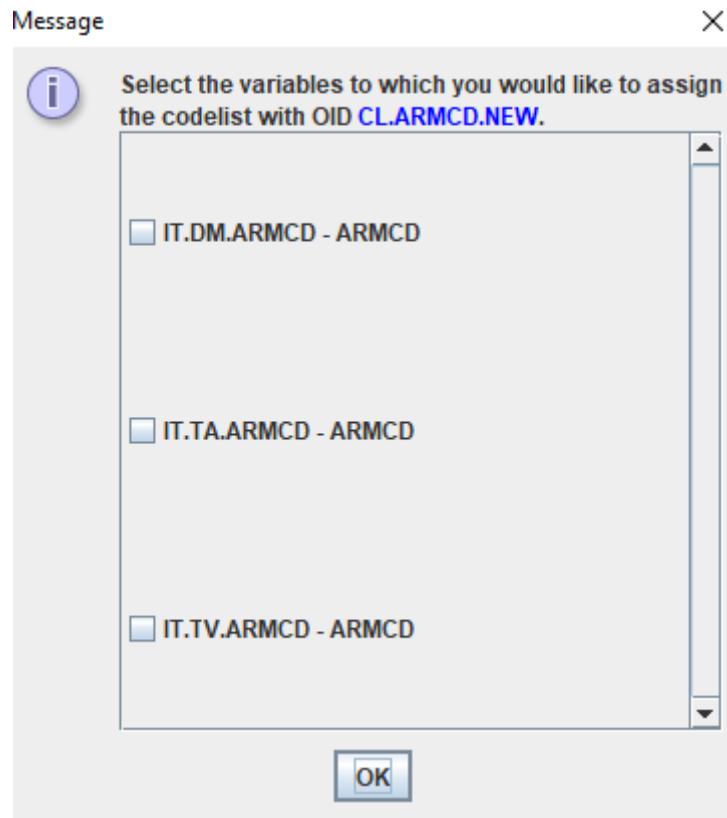
We decided to make this a new codelist, as when one is not satisfied with the results of the retrieval, one can still use the original codelist, and add values there.

Of course you can later always change the value for the OID (identifier) of the codelist. It is just an identifier and has no meaning by itself.

After clicking "OK", one is presented with:

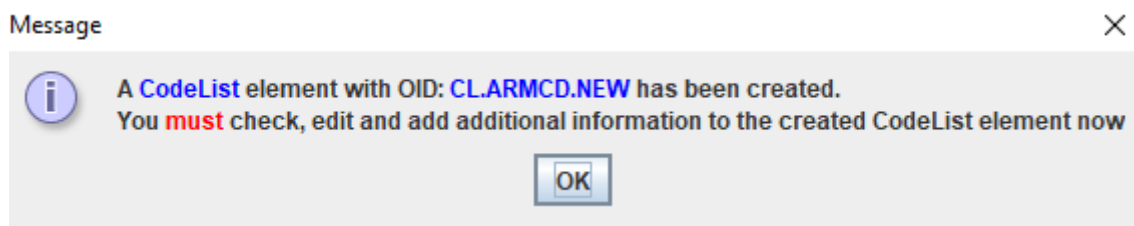


One can then still exclude some values by unchecking one or more checkboxes, e.g. due to regulatory requirements for the codelist (think about "OTHER", "MULTIPLE", "--ALL"). In this case, we accept everything by clicking "OK", leading to:



The system has now detected that the "ARMCD" variable is also used in the TA and TV dataset definition, for which the variable identifiers are "IT.DM.ARMCD" (from the dm.xpt file), "IT.TA.ARMCD" (for the TA dataset definition) and "IT.TV.ARMCD" (for the TV dataset definition) and asks whether the system should assign this new codelist to these 3 variables. You will usually want to have that, and thus check all three checkboxes.

After clicking "OK", one gets some additional advice ...



This is good! For example, when having generated the items in the codelist by retrieval from the SAS-XPT file, there is no information about the meaning of the generated codes like "WONDER10" and "WONDER20" as they were obtained:

Contents of element CodeList

Content for CodeListItem

CodedValue	Rank	OrderNumber	ExtendedValue	Decode	Alias	Description
WONDER10				TranslatedText Language: not assigned Text: WONDER10		
WONDER20				TranslatedText Language: not assigned Text: WONDER20		
PLACEBO				TranslatedText Language: not assigned Text: PLACEBO		
SCRNFAIL				TranslatedText Language: not assigned Text: SCRNFAIL		

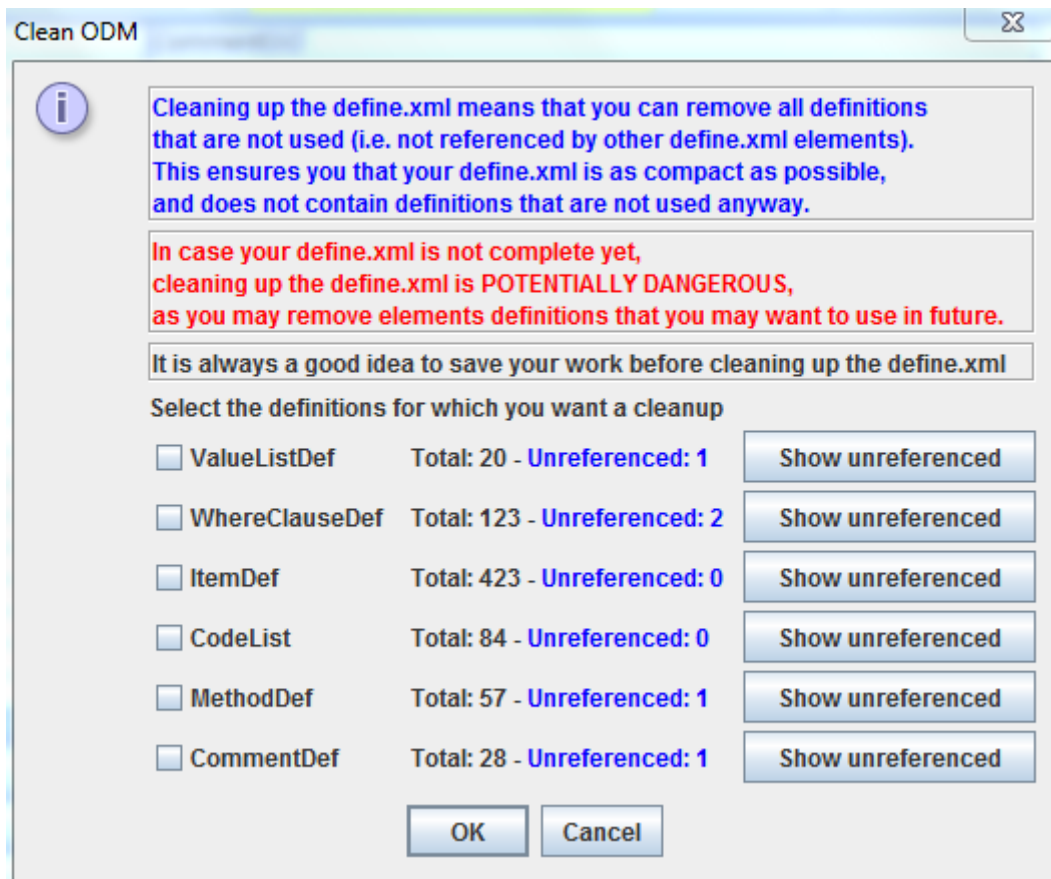
As the system only has retrieved "WONDER10", "WONDER20", "PLACEBO" and "SCRNFAIL", it has assigned the same values for the "decode". As we did in the prior case, we can then still change that by clicking on the "+" icon in the table with the codelists for "CL.ARMCD.NEW", and then "drill down" to the individual items.

Cleaning up your define.xml

After working some time on your define.xml, you will probably want to clean up some things. You might have some variables, valuelist definitions, comment and method definitions, and even "where clauses" that you once defined, but then finally decided not to use. Technically spoken, this means that you have "definitions" (e.g. "def:CommentDef") that are never reference. This might even happen to variable definitions ("ItemDef"), for example "permissible" variables that you loaded from the SDTM template, but that you do not use in any of your datasets.

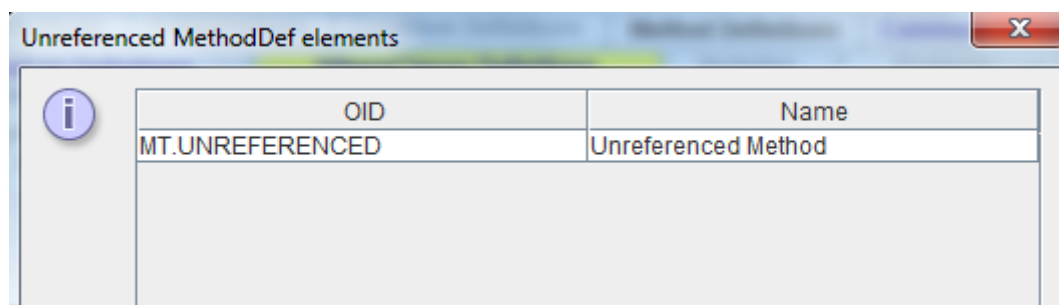
Having unreferenced definitions is in fact not a problem at all - they will even not show up in the HTML view that you get when displaying the define.xml using the stylesheet. However, the FDA does not like it.

In order to clean up your define.xml, use the menu "Edit - Clean". An inventory of all unreferenced "definitions" is then made and the result displayed:

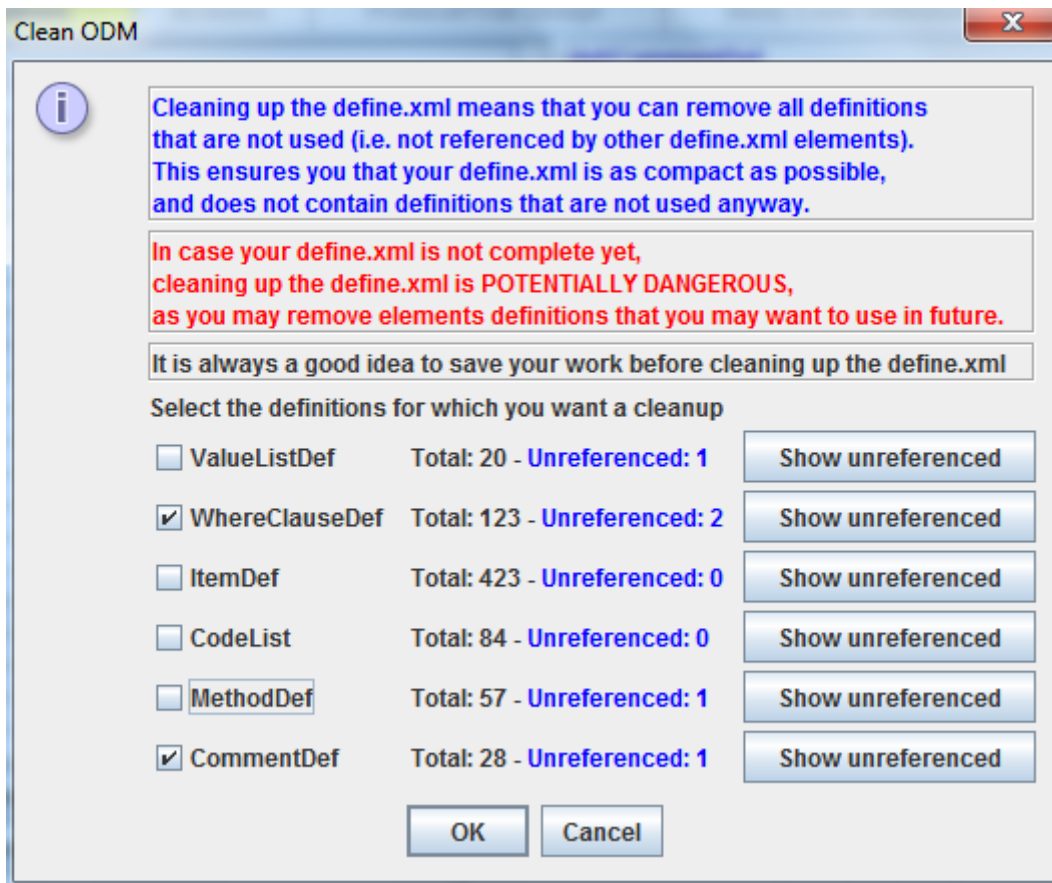


informing us that there is 1 unreferenced ValueList (def:ValueList element), 2 "where clauses" (def:WhereClauseDef", and 1 unreferenced method definition ("MethodDef") and 1 unreferenced comment definition ("def:CommentDef").

By clicking a "Show unreferenced" button, one can immediately see which definitions are unreferenced. For example for "MethodDef":



The user can now decide which type of unreferenced elements can be cleaned up, i.e. be removed from the define.xml. For example, if we do want to remove all unreferenced comments ("def:CommentDef") and "where clauses" ("def:WhereClauseDef"), but keep unreferenced "method definitions" (e.g. because you still want to use them later) and unreferenced "value list definitions", check the corresponding checkboxes:



and just click "OK" to start the cleanup action.

If you then use the menu "Edit - Clean" again, the result is:

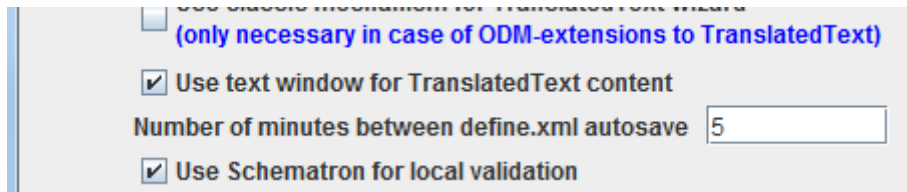


The effect can of course also immediately be seen by inspecting the tables for the "WhereClause" and for the "Comment" or by inspecting the define.xml XML itself using "View - define.xml as XML".

Saving your work to file

Independent on whether you want to clean your define.xml or not, you will want to save your work from time to time.

By default, the software already saves the define.xml file every 5 minutes to a file in the folder "autosave", which is a subfolder to the folder where you installed the software. You can change the "autosave" intervals by using the menu "Options - Settings" and look for the field "Number of minutes between define.xml autosave":



You will only be allowed to enter integer values. If you set "0" or a negative number, no autosaving will be done at all.

The files in the "autosave" directory have a name allowing you to find out when exact they were created. For example:

Name	Änderungsdatum	Typ
ODM_2019_10_19_16-52-35.xml	19.10.2019 16:52	XML-Datei
ODM_2019_10_19_16-57-35.xml	19.10.2019 16:57	XML-Datei
ODM_2019_10_19_17-2-35.xml	19.10.2019 17:02	XML-Datei
ODM_2019_10_19_17-23-28.xml	19.10.2019 17:23	XML-Datei

The file with name "ODM_2019_10_19_17-29-28.xml" was created on October 19nd at 17:23:28 local time (24 hour notation is used).

So, even if the software or your computer would crash, you can always recover your earlier work.

Remark that it is not a bad idea to regularly clean up the contents of the "autosave" directory.

To actively save your define.xml to file, use the menu "File - Save define.xml". The following dialog is displayed:

Save define.xml to file ×

CDISC ODM File OID

ODM File Description

Study OID (required)

Metadata Version OID (required)

Metadata Version Name (required)

Metadata Version Description

def:DefineVersion

Define-XML Context

Do not add a stylesheet reference (yet)
 Add CDISC stylesheet reference
 Add own stylesheet reference

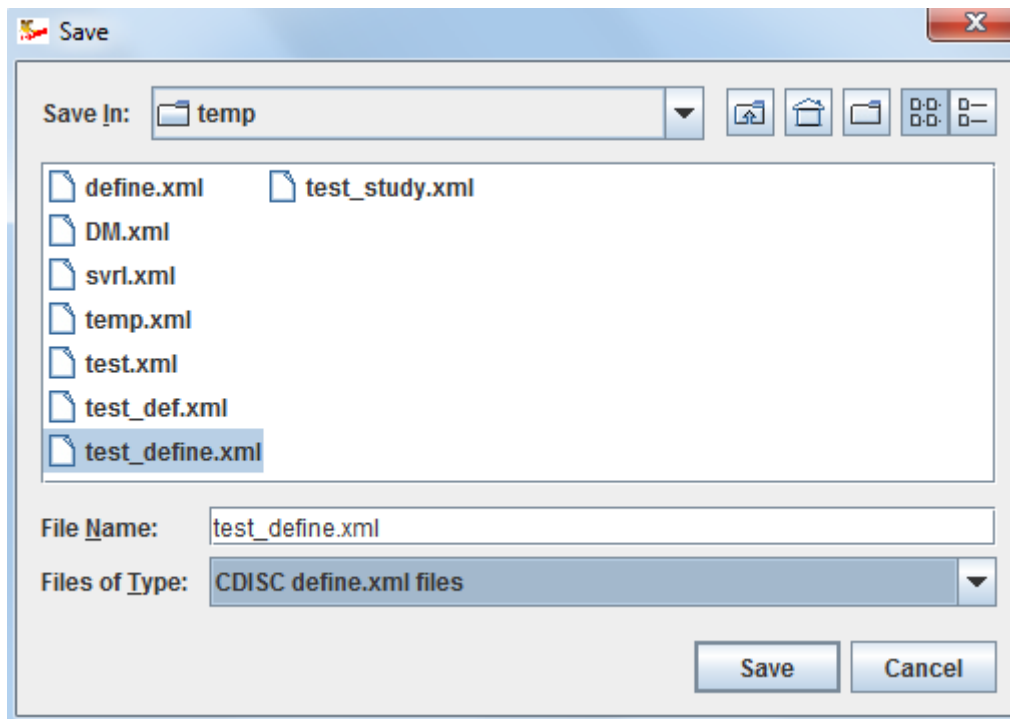
At this point, you can still change some of the metadata.

You can also select whether your output define.xml needs to have a reference to the stylesheet file. You can choose between not adding a reference at, adding a reference to the default stylesheet file delivered by CDISC, or to add a reference to a stylesheet that you developed yourself.

In the case of the latter choice, use the "Browse" button to browse for the name and location of that stylesheet file.

Remark that it remains your own responsibility to add (or copy) the stylesheet file to the directory where you save the file to. For the latest version of the CDISC define.xml stylesheet, please look in your define.xml CDISC distribution package or the CDISC website.

After clicking "OK", you will be asked where the file needs to be written to:



Please also remark that it is not always necessary to first save your work to file in order to display your define.xml in your favorite browser. As explained before, you can always use the menu "View - define.xml in browser".

Creating ValueLists

Creating ValueLists (value-level metadata) is often seen as one of the most challenging tasks when creating a define.xml. With our software however, it becomes very easy.

ValueLists are usually created when there are different "test codes", and the datatype, the maximal length, units and associated codelists for the results differ depending on the test codes. Typical examples are found in the "Findings" domains like VS (vital signs), LB (laboratory) and EC (electrocardiogram). For example, for vital signs, systolic and diastolic are integer number (with mmHg as unit), weight is a float, and "frame size" is text and governed by a codelist. For laboratory findings, the valuelists will usually be even more complicated.

Valuelists are mostly created based on --TESTCD and then assigned to --ORRES, and to --STRESC variables. They can also be assigned to --ORRESU and/or --STRESU variables. For supplemental qualifiers, a valuelist is very often assigned to QVAL, based on QNAM.

In this manual, we will show how to set up a valuelist for VS (vital signs)

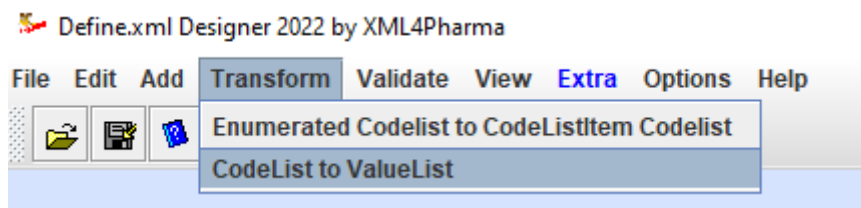
The best way to start generating a valuelist is to start from a codelist. Let us suppose that you have already loaded the CDISC-CT codelist for VSTESTCD:

Extra information for: CodeList, with OID = CL.C66741.VSTESTCD

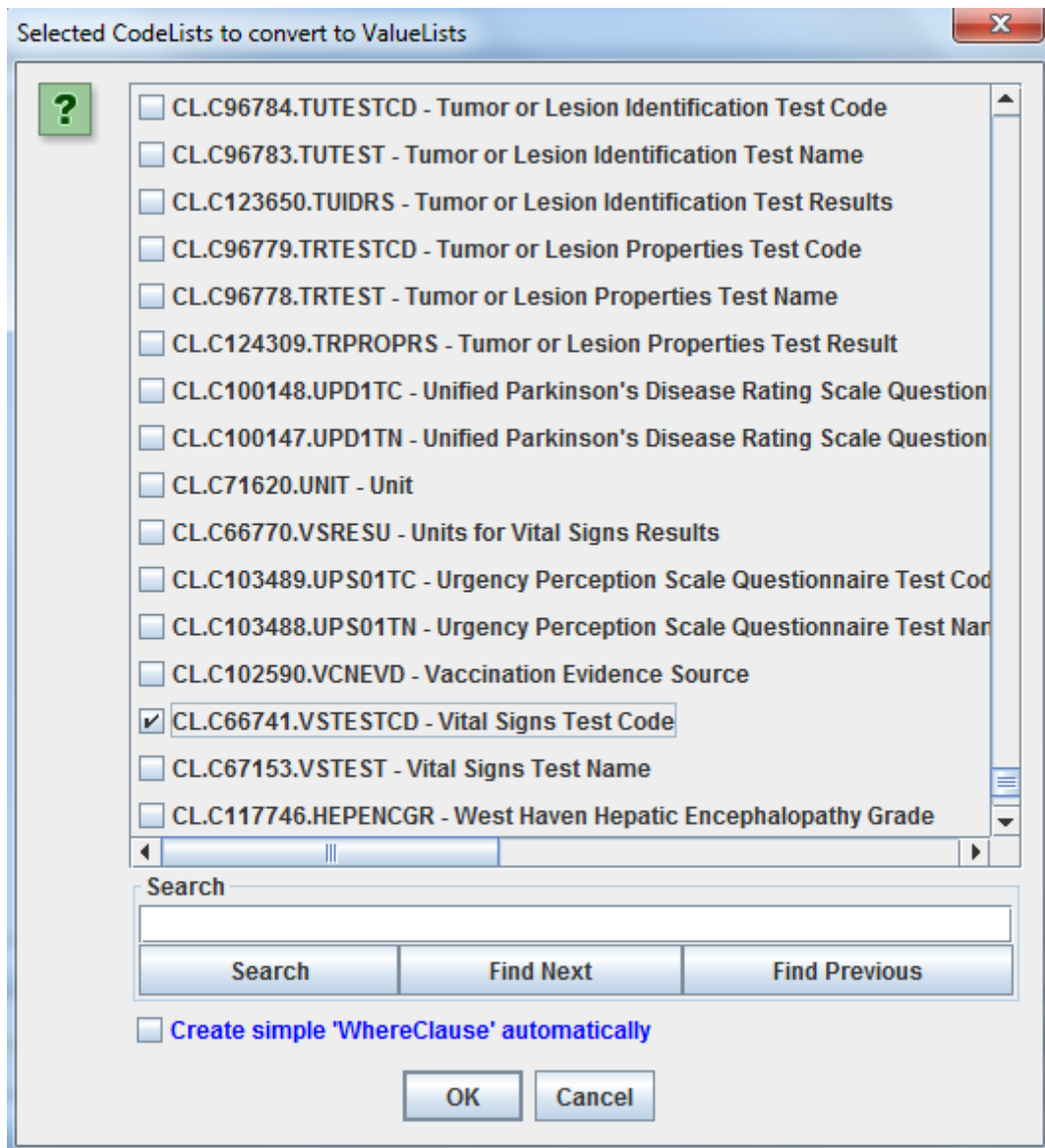
Description	CodeListItem	ExternalCodeList	EnumeratedItem	Alias
	CodedValue	Rank	OrderNumber	ExtendedValue
ABS	ABSKNF			
BMI	BMI			
BODLN	BODLNTH			
BODYF	BODYFATM			
BSA	BSA			
DIABP	DIABP			
FARMC	FARMCIR			
FRMSI	FRMSIZE			
HDCIR	HDCIRC			
HEIGHT	HEIGHT			
HIPCIR	HIPCIR			
HR	HR			
IDEAL	IDEALWT			
KNEE	KNEEHEEL			
LBM	LBM			
MAP	MAP			
OXYSAT	OXYSAT			
PULSE	PULSE			
PULSE	PULSEPR			
RESP	RESP			
SAD	SAD			
SSSKNF	SSSKNF			
SYSBP	SYSBP			
TBW	TBW			
TEMP	TEMP			

As one can see, there is a large number of tests for vital signs, and it might be that you have some more, or some less in your study, and have already extended or subsetted the codelist for VSTESTCD.

In order to start developing a new ValueList for vital signs tests, use the menu "Transform - CodeList to ValueList":



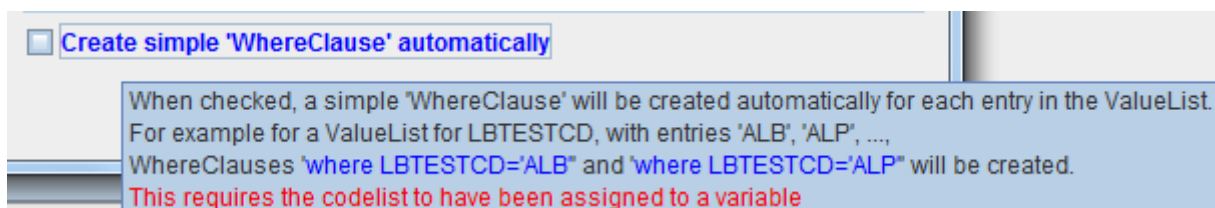
This results in:



and select the codelist "CL.C66741.VSTESTCD - Vital Signs Test Code" (you can search for a specific codelist using the "Search" field and buttons)

If you started from a set of SAS-XPT files, and have subsetted this codelist yet, you will of course select the subsetted codelist²².

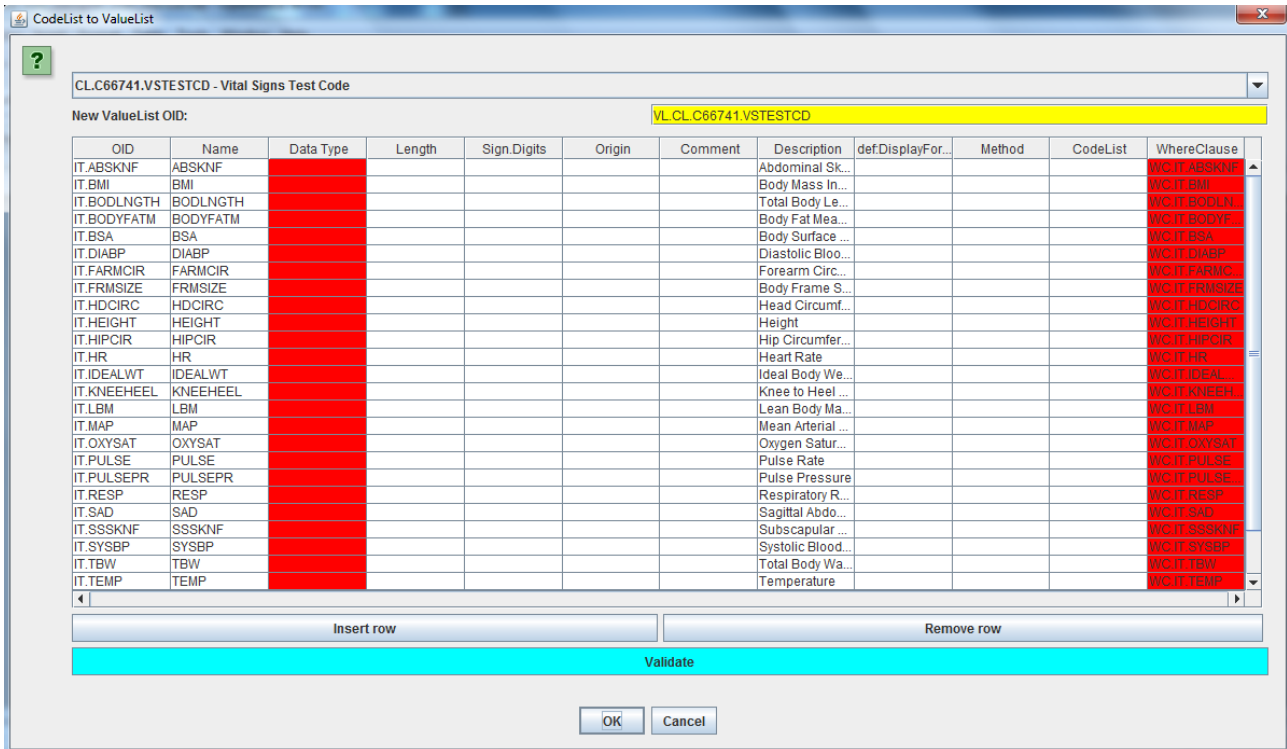
Remark the checkbox "Create simple 'WhereClause' automatically. When holding the mouse over it, more explanation is displayed:



²²In case you selected a codelist with many entries, the system will first provide a message that it might be useful to subset the codelist first. You can then still interrupt the process and do the subsetting first. For example, the by CDISC published codelists for LBTESTCD and LBTEST have several hundred entries, but you will not have used them all in your study, so you should then subset these codelists first.

We will explain this in detail in the next section "Automatically assigning WhereClauses to ValueList entries". We will however leave the checkbox unchecked for now.

After clicking "OK", this immediately leads to the following dialog and table:

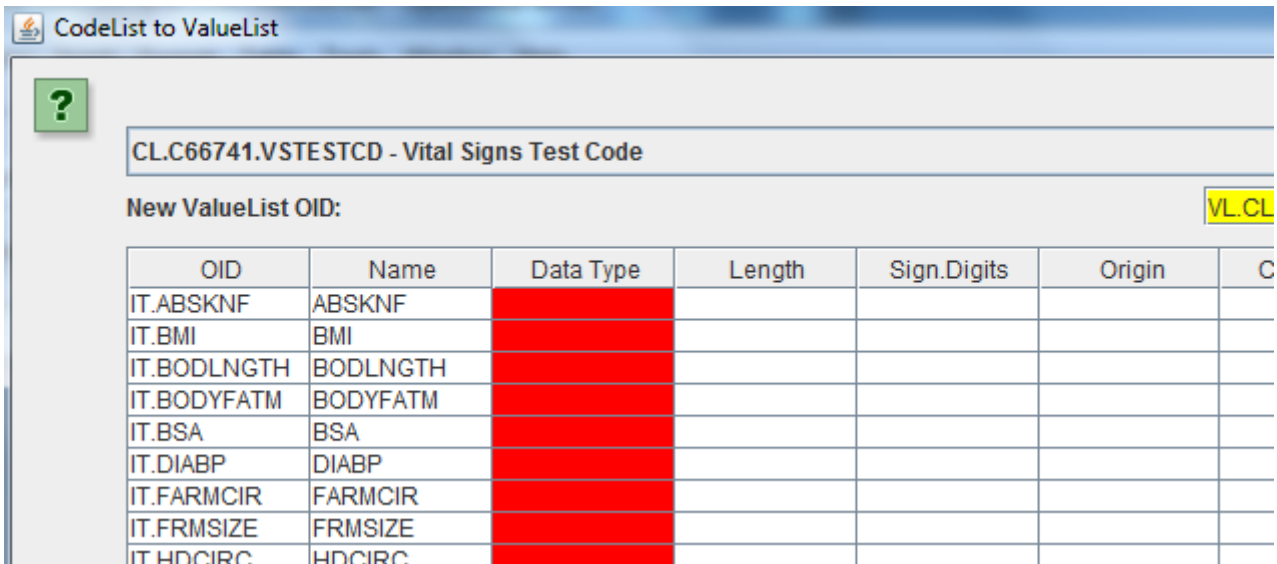


In the upper right part of the dialog, a new identifier (OID) is proposed for the ValueList, consisting of the prefix "VL." and the OID of the original codelist:



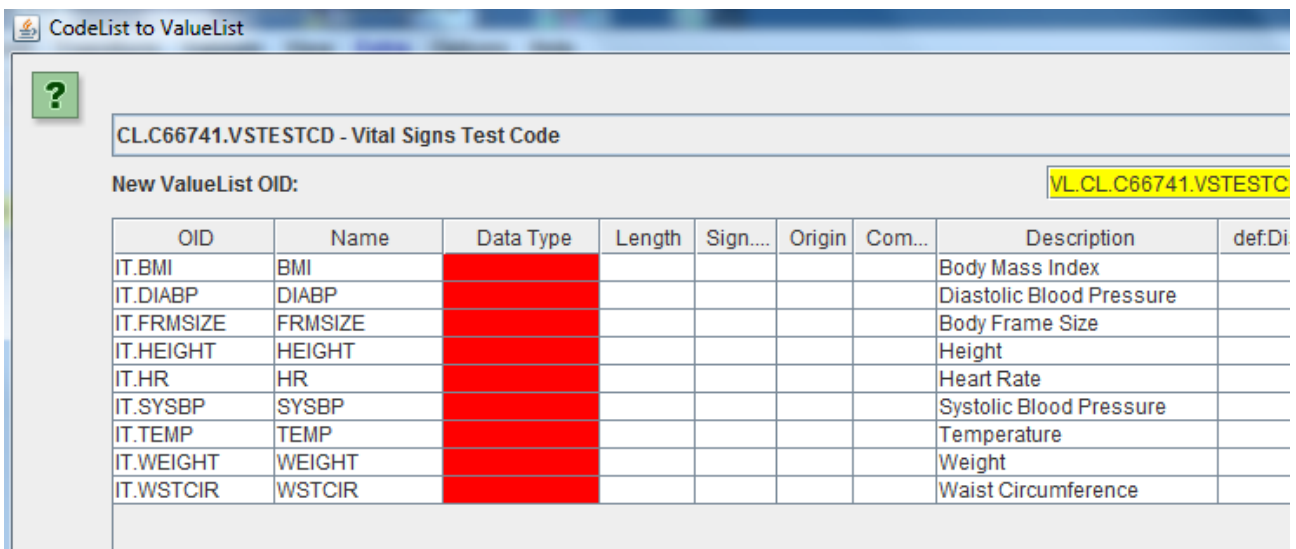
You can change it and assign another OID, but in most cases you will probably want to keep the proposed one, as it immediately shows that the ValueList was derived from the "VSTESTCD" codelist.

The table contains a list of all items in the codelist:



The red fields indicate that these are mandatory fields, and the you must provide a value for it.

We can now starting editing the table. In first instance, we want to delete those rows (test codes) that are not applicable to our study. Usually this will be easier if we already subsetted the vital signs codelist, and use that as a starting point for our valuelist. Deleting the unnecessary rows can e.g. lead to:



Remark that one can remove more than one row at the time by using the "Ctrl" key when selecting rows. Then use the "Remove Row" button to remove the selected rows.

Usually the field "Description" is very useful to understand the codes from the codelist. Use it!

For each remaining test, we now need to assign the appropriate data type. Usually, this is immediately clear by a look on the CRF. For example for BMI we expect a float, whereas for "Hearth Rate" we probably inspect an integer (depending on the unit that was used, mostly "beats per minute"). For "Body Frame Size" however, there will usually be an associated codelist (checkboxes on the CRF) with choices like "small", "large" etc..

For example:

IT.HR	HR	integer			
IT.SYSBP	SYSBP	integer			
IT.TEMP	TEMP	float			
IT.WEIGHT	WEIGHT	float			
IT.WSTCIR	WSTCIR	float			

integer ▲
 float
 text
 date
 partialdate
 time
 partialtime
 datetime ▼

It is a good idea now to click the button "Validate" (the cyan colored button near the bottom). This will lead to:

OID	Name	Data Type	Length	Sign....	Origin
IT.BMI	BMI	float			
IT.DIABP	DIABP	integer			
IT.FRMSIZE	FRMSIZE	text			
IT.HEIGHT	HEIGHT	float			
IT.HR	HR	integer			
IT.SYSBP	SYSBP	integer			
IT.TEMP	TEMP	float			
IT.WEIGHT	WEIGHT	float			
IT.WSTCIR	WSTCIR	integer			

I.e. we also need to provide the maximal length for each of the test results²³.

We fill the "Length" field, and then validate again using the "Validate" button:

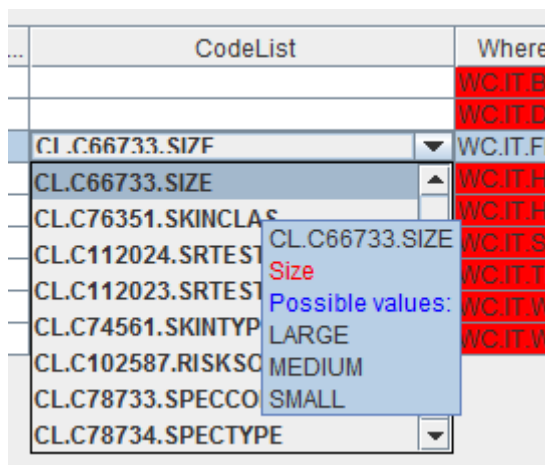
OID	Name	Data Type	Length	Sign....	Or...	Comment	Description	def.DisplayFor...	Method	CodeList	WhereClause
IT.BMI	BMI	float	5				Body Mass Index				WC.IT.BMI
IT.DIABP	DIABP	integer	3				Diastolic Blood...				WC.IT.DIABP
IT.FRMSIZE	FRMSIZE	text	10				Body Frame Size				WC.IT.FRMSIZE
IT.HEIGHT	HEIGHT	float	5				Height				WC.IT.HEIGHT
IT.HR	HR	integer	3				Heart Rate				WC.IT.HR
IT.SYSBP	SYSBP	integer	3				Systolic Blood ...				WC.IT.SYSBP
IT.TEMP	TEMP	float	4				Temperature				WC.IT.TEMP
IT.WEIGHT	WEIGHT	float	5				Weight				WC.IT.WEIGHT
IT.WSTCIR	WSTCIR	integer	3				Waist Circumfe...				WC.IT.WSTCIR

Everything looks all right, except that the "WhereClause" field is still colored red.

We also still need to assign a codelist to the item "Body Frame Size" (FRMSIZE) as the CRF shows checkboxes (choices) for this. It might be that we have already generated a separated codelist for this, but it might also be that we can use a CDISC codelist for "body frame size". For example,

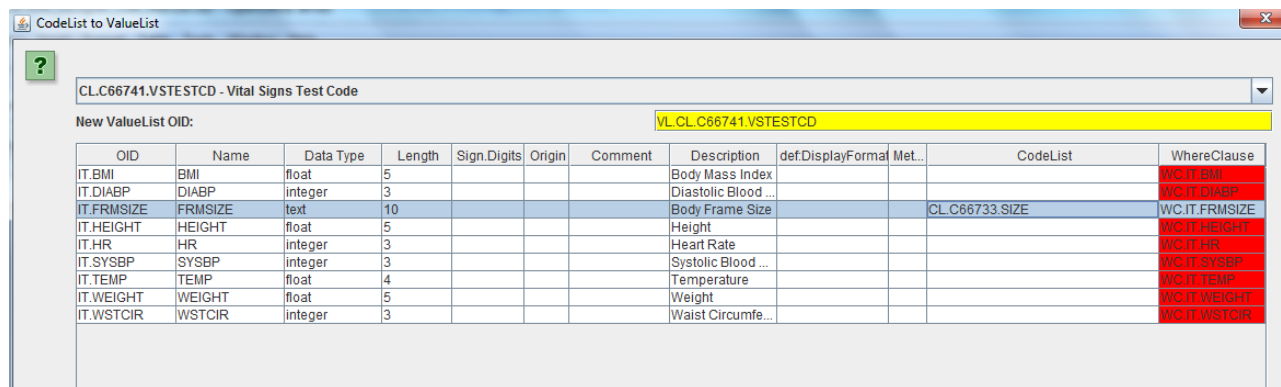
²³Remark that for datatype "Date", "Time" and similar data types, no maximal length needs to be provided.

clicking the "CodeList" field for "SIZE" leads to:



When making decisions about which codelist is applicable, holding the mouse over a selected item always shows the allowed values, which is very helpful. Also remark that "SIZE" is an extensible codelist, so if you have more checkboxes on your CRF for "body frame size" than "Small", "Medium" or "Large", you may want to extend that codelist (see further on).

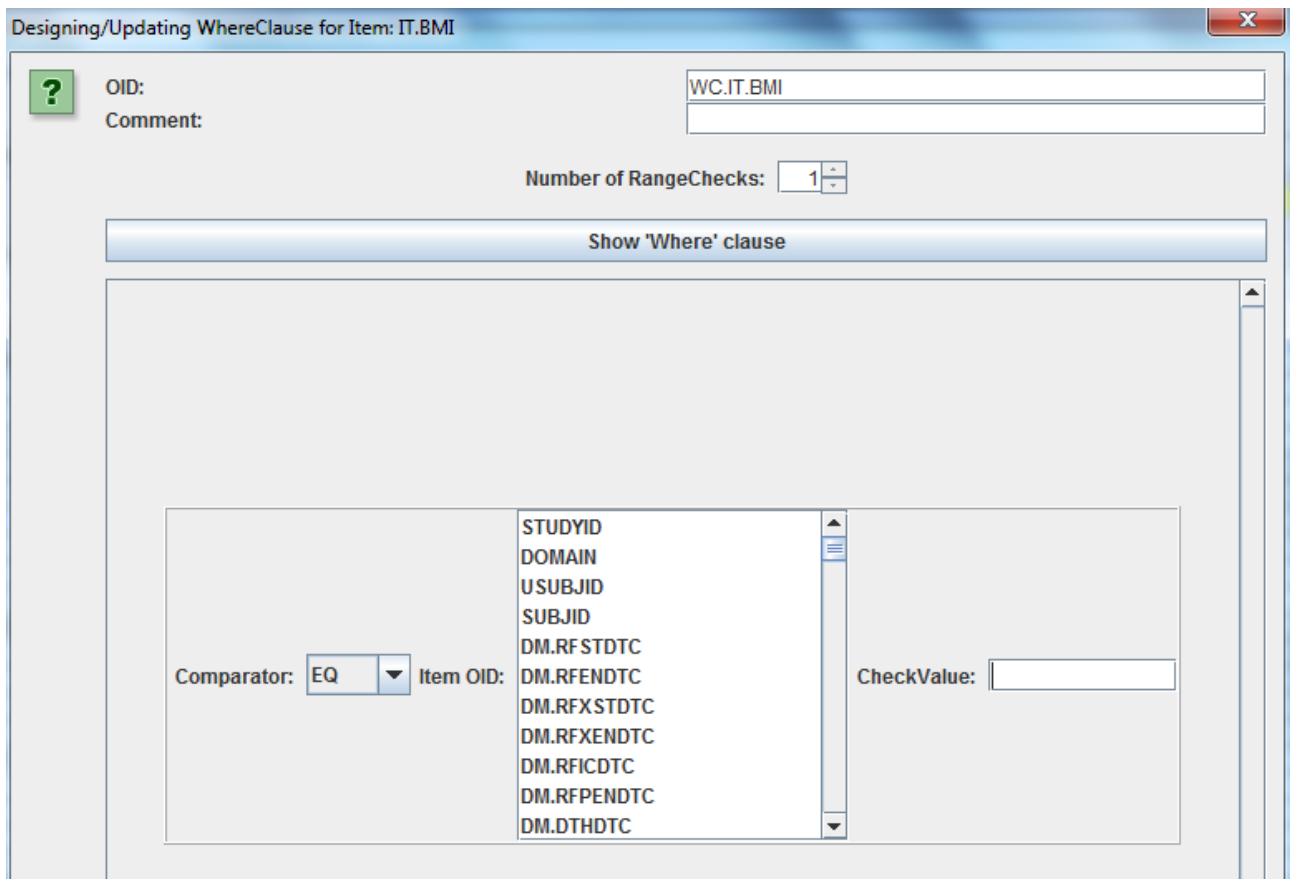
So we now have:



Working with the "Where Clause" is often seen as the most difficult part of define.xml 2.0. However with the wizards provided in our software, this becomes very easy. As of define.xml 2.0, a "Where Clause" must be provided for each item in a valuelist. It is a "machine-readable" as well as "human-readable" description about when the valuelist item applies. For example, "FRMSIZE" information (data type, length, associated codelist) will be used when ... the value of VSTESTCD is "FRMSIZE"... Looks logical isn't it?

The define.xml 2.0 specification (and SDTM sample file) however shows a more complicated example for VSORRESU (vital signs original result unit) where the value is "inches" when COUNTRY (in DM) is "USA" and "centimeter" when COUNTRY is either "MEX" (Mexico) or "CAN" (Canada).

Now click the most-right cell "WC.IT.BMI". This opens a wizard for setting up the "where clause":



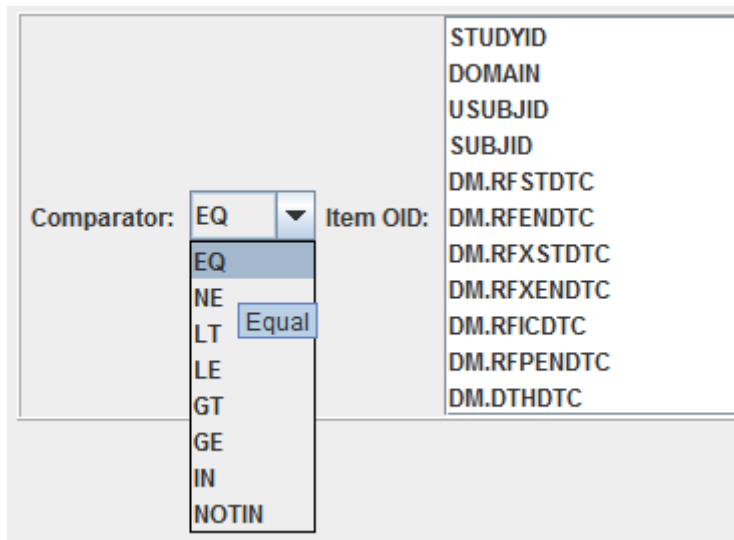
The identifier (the OID) has already been created, you can however change it (mostly you will want to keep the one that is suggested).

The "Comment" field only needs to be populated when your "where" contains variables from other domains. This is well explained in the define.xml 2.0 specification in example 4.2.2.2 about the metadata for VSORRESU (vital signs original result units) depending on the country (USA, Mexico or Canada).

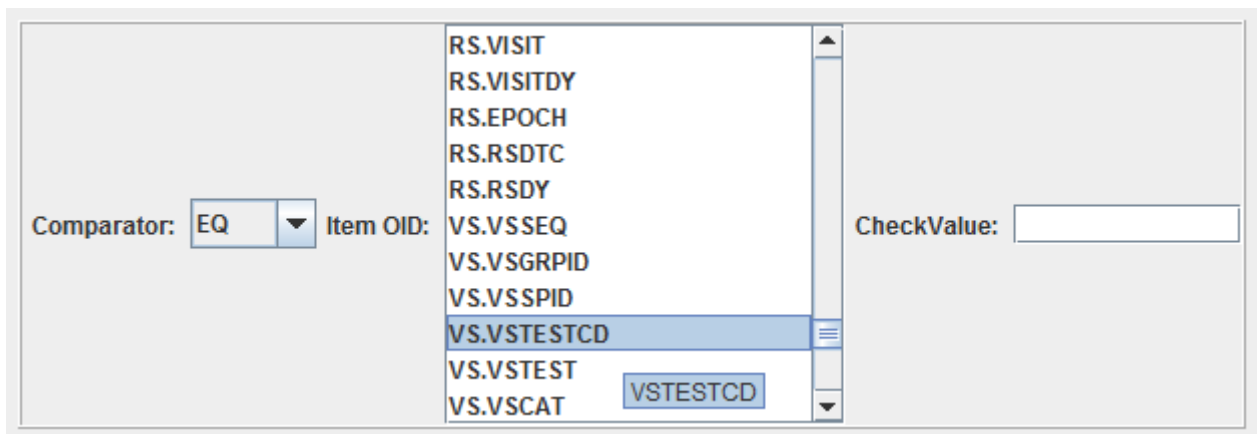
If you fill something into the "Comment" field, a "def:CommentDef" element will be generated and a reference to it is added in the "ItemRef" element within the ValueList.

Then you can set the number of condition "RangeChecks" that will be combined. For example, if you want to have a condition like "where the test code is BMI and the age is less than 70), then you will need two "RangeChecks" and need to set the "Number of RangeCheck" spinner to 2.

In our case however, we only need to state "where the vital signs test code is BMI", so we can leave the spinner to "1". Also the value for the "Comparator" can be left to "EQ" as it means "equals to" as is also shown by the tooltip:

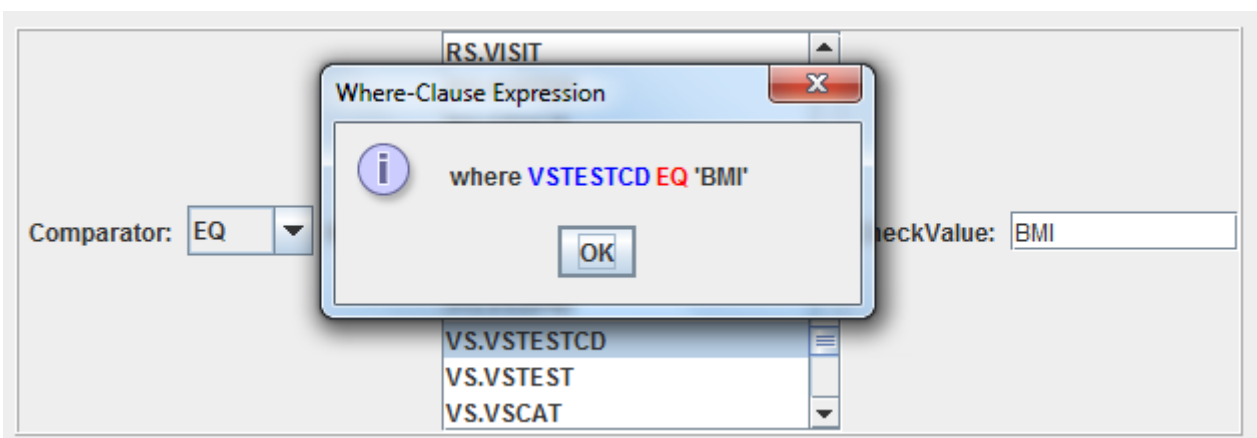


We then still need to assign the SDTM/SEND/ADaM variable in the condition definition which in our case is "VSTESTCD". It can quickly be found by typing the first characters "VS" and then selecting "VS.VSTESTCD":



The "CheckValue" then still has to be typed in. It is "BMI".

You can always check what you have developed as a condition by clicking the button "Show 'Where' Clause". The human readable condition is then displayed. In our case:



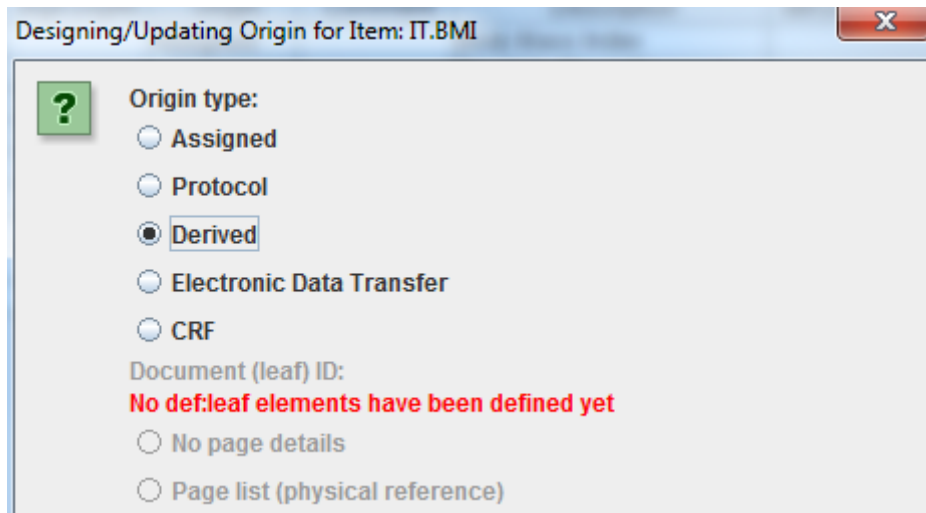
Now click OK until back in the table with all ValueList items.

Setting the "WhereClause" now needs to be repeated for all the items in the "ValueList" table. Regularly use the "Validate" button to ensure that (at least structurally) everything is correct. Finally, one would obtain a table without any red cells anymore:

OID	Name	Data Type	Length	Sign.Digits	Origin	Comment	Description	def.DisplayFor...	Method	CodeList	WhereClause
IT.BMI	BMI	float	5				Body Mass Index				WC.IT.BMI
IT.DIABP	DIABP	integer	3				Diastolic Blood Pressure				WC.IT.DIABP
IT.FRMSIZE	FRMSIZE	text	10				Body Frame Size			CL.C66733.SIZE	WC.IT.FRMSIZE
IT.HEIGHT	HEIGHT	float	5				Height				WC.IT.HEIGHT
IT.HR	HR	integer	3				Heart Rate				WC.IT.HR
IT.SYSBP	SYSBP	integer	3				Systolic Blood Pressure				WC.IT.SYSBP
IT.TEMP	TEMP	float	5				Temperature				WC.IT.TEMP
IT.WEIGHT	WEIGHT	float	5				Weight				WC.IT.WEIGHT
IT.WSTCIR	WSTCIR	integer	3				Waist Circumference				WC.IT.WSTCIR

In most cases, you will also need to add an "Origin" to each of the variables (you can still do it later too, but with the danger that it is forgotten). In order to add an "Origin", click in the "Origin" cell. For example, for "BMI":

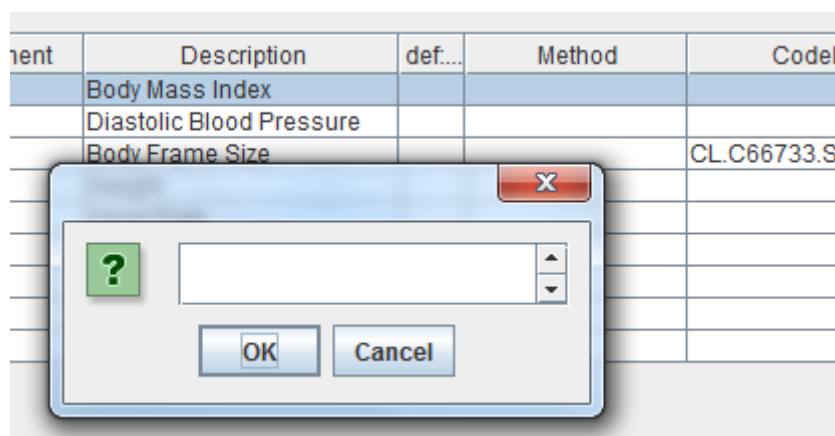
In this case, the "Origin" is derived, as the BMI was not captured on the CRF, but was calculated from "height" and "weight". So, in this case, one needs to set it to "Derived":



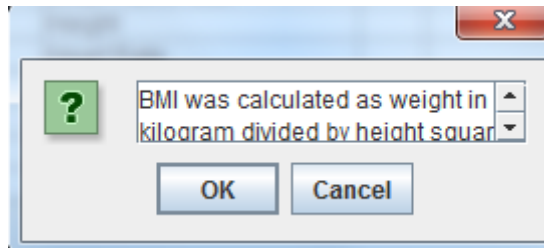
and after clicking "OK":

OID	Name	Data Type	Length	Sign.Digits	Origin	Comment	Description	def...	Method	CodeList
IT.BMI	BMI	float	5		Derived		Body Mass Index			
IT.DIABP	DIABP	integer	3				Diastolic Blood Pressure			
IT.FRMSIZE	FRMSIZE	text	10				Body Frame Size			CL.C66733.SIZE
IT.HEIGHT	HEIGHT	float	5				Height			
IT.HR	HR	integer	3				Heart Rate			
IT.SYSBP	SYSBP	integer	3				Systolic Blood Pressure			
IT.TEMP	TEMP	float	5				Temperature			
IT.WEIGHT	WEIGHT	float	5				Weight			
IT.WSTCIR	WSTCIR	integer	3				Waist Circumference			

Now we should also provide the "method" how the BMI was calculated. For this click in the "Method" cell. This results in:



and one can e.g. add:



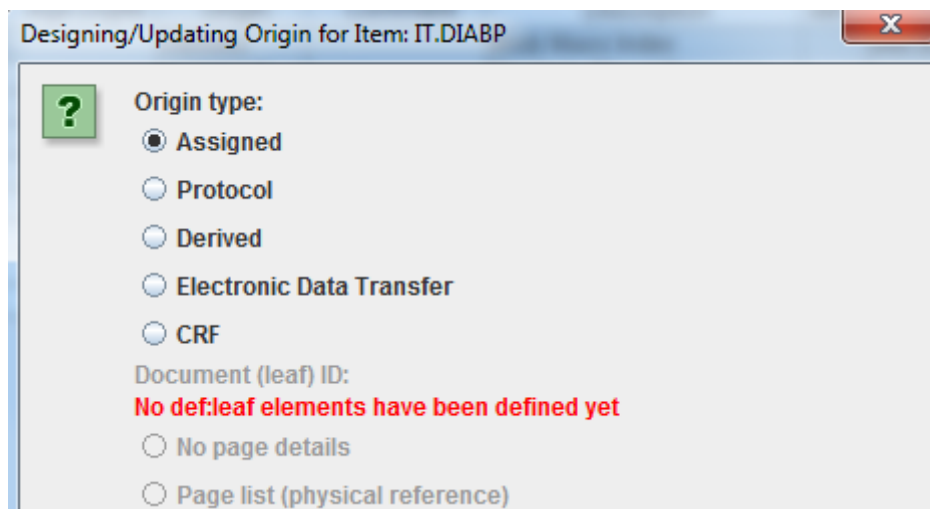
And after clicking "OK":

ent	Description	def...	Method	CodeList
	Body Mass Index		BMI was calculate...	
	Diastolic Blood Pressure			
	Body Frame Size			CL.C66733.SIZE
	Height			

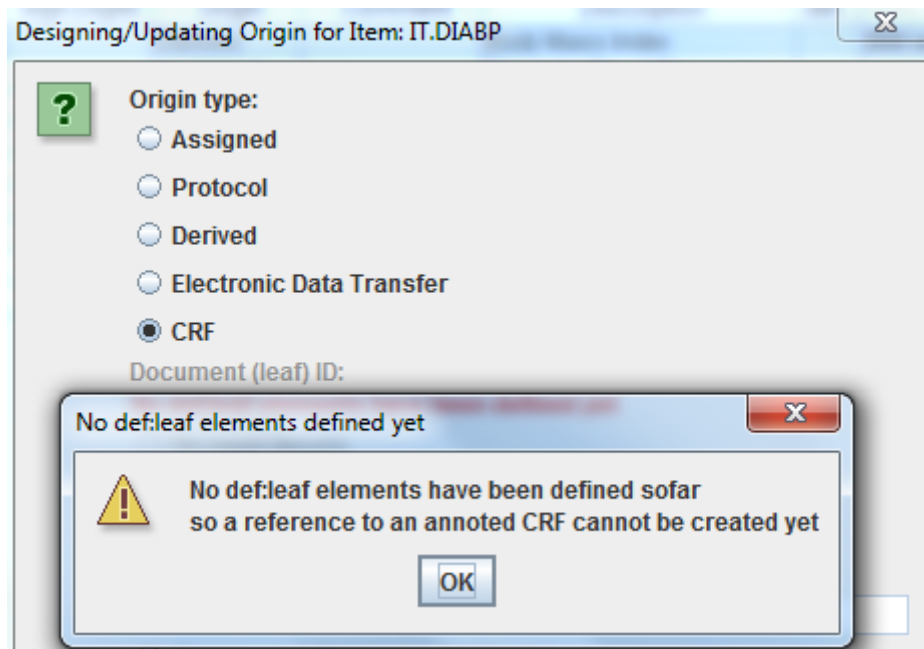
Adding an "Origin" and a "Method" will automatically create a "def:Origin" and a "MethodDef" element in the define.xml (with references), so you will later still be able to change the information using the regular methods.

In a similar way, one can always add a "Comment" by clicking in the "Comment" cell. This will generate a "def:CommentDef" with a reference to it, so that you can still later change the information.

Essentially, we should add an "Origin" to each of the newly created value-level variables, for example for "DIABP":

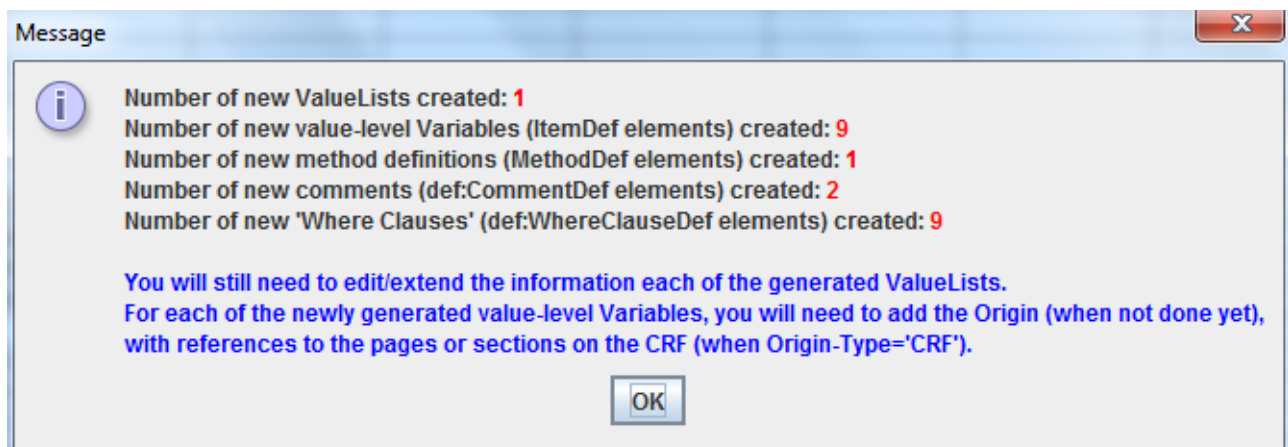


Now, the diastolic blood pressure was captured using the CRF, but we haven't added the information yet about where the annotated CRF resides. So, we are not allowed yet to add the page information yet. When "CRF" is clicked, the following message appears:



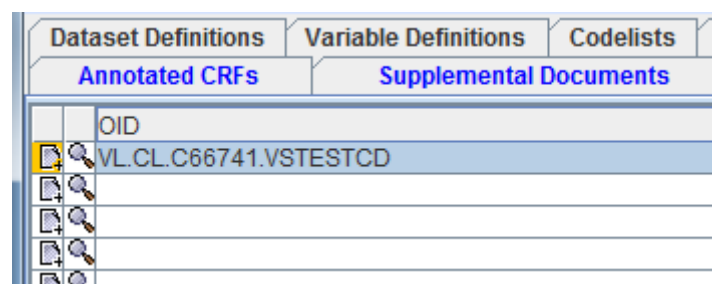
However, as we will have to add an "Origin" to each of the variables, domain variables as well as valuelist variables, we can do this later when assigning origins to all of them.

When done editing the table, then clicking "OK" leads to the following message:



showing a nice summary of what has been created.

You can now always edit the created valuelists, variable definitions, comments, method descriptions and "where clauses" by selecting the appropriate tab. For example, for "ValueLists":



or for "Where Clauses":

Dataset Definitions		Variable Definitions	Codelists	Imputation Methods	Presentations	Condition Definitions	Me
Annotated CRFs		Supplemental Documents		ValueList Definitions		WhereClause Definitions	
	OID					CommentOID	
	WC.IT.DAYS						
	WC.IT.BMI						
	WC.IT.DIABP					COM.WC.IT.DIABP	
	WC.IT.FRMSIZE						
	WC.IT.HEIGHT						
	WC.IT.HR						
	WC.IT.SYSBP						
	WC.IT.TEMP						
	WC.IT.WEIGHT						
	WC.IT.WSTCIR						

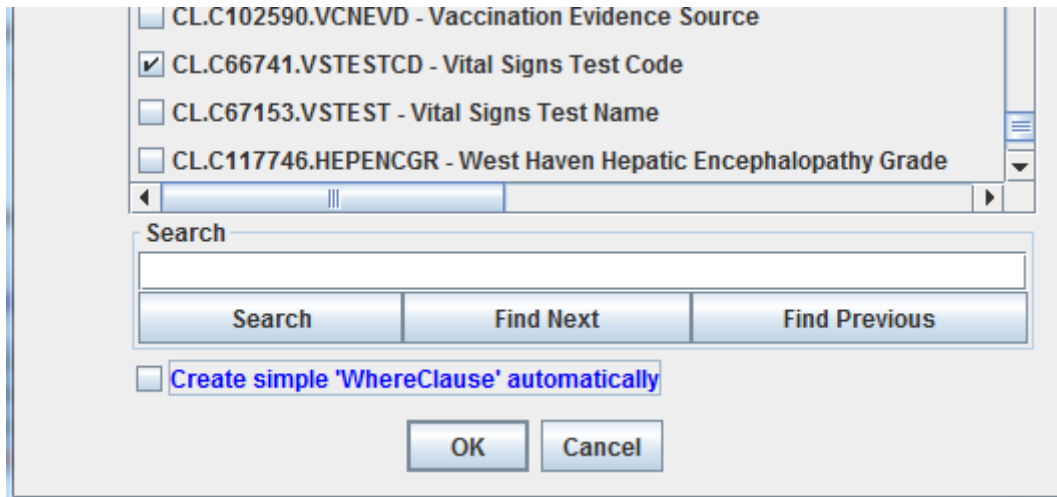
Also remark that new variables (at the "value" level) have been created. These appear in the "Variable Definitions" table:

Dataset Definitions		Variable Definitions		Codelists	Imputation Methods	Presentations	Condition Definitions
Annotated CRFs		Supplemental Documents		ValueList Definitions		WhereClause Definitions	
OID	Name	DataType	Length	SignificantDigits	SASFieldName	SDSVarNam	
TS.TSPARMCD	TSPARMCD	text	80				
TS.TSPARM	TSPARM	text	80				
TS.TSVAL	TSVAL	text	80				
TS.TSVALNF	TSVALNF	text	80				
TS.TSVALCD	TSVALCD	text	80				
TS.TSVCDREF	TSVCDREF	text	80				
TS.TSVCDVER	TSVCDVER	text	80				
IDVAR	IDVAR	text	80				
IDVARVAL	IDVARVAL	text	80				
RELTYPE	RELTYPE	text	80				
RELID	RELID	text	80				
QNAM	QNAM	text	80				
QLABEL	QLABEL	text	80				
QVAL	QVAL	text	80				
QORIG	QORIG	text	80				
QEVAL	QEVAL	text	80				
IT.BMI	BMI	float	5				
IT.DIABP	DIABP	integer	3				
IT.FRMSIZE	FRMSIZE	text	8				
IT.HEIGHT	HEIGHT	integer	3				
IT.HR	HR	integer	3				
IT.SYSBP	SYSBP	integer	3				
IT.TEMP	TEMP	float	5				
IT.WEIGHT	WEIGHT	float	5				
IT.WSTCIR	WSTCIR	integer	3				

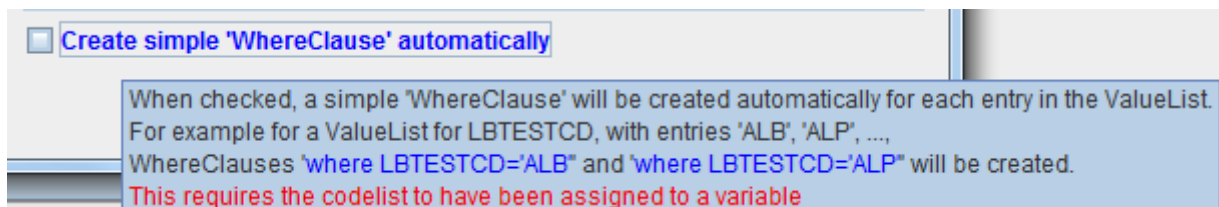
After you have defined where the annotated CRF resides, and how that file is named, you will be able to add the "Origin" to each of the variables, be it domain variables or valuelist variables.

Automatically assigning WhereClauses to ValueList entries

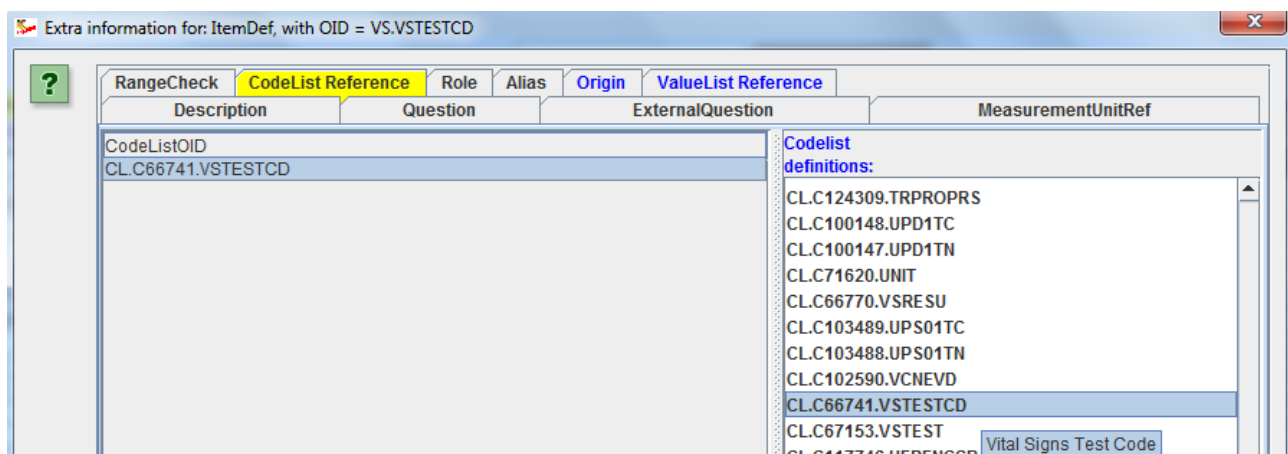
When creating a valuelist starting from a codelist, you will have noticed the checkbox "Create simple 'WhereClause' automatically":



When holding the mouse over it, more explanation is displayed:

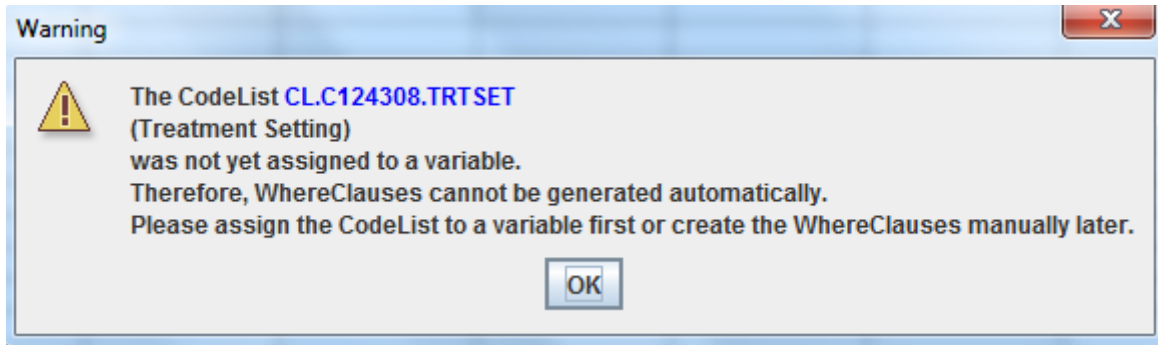


In order to generate such "WhereClauses" you will already need to have assigned the codelist to a variable. In the case of "Vital Signs Code" (CL.C66741.VSTESTCD), you will almost surely have assigned this codelist to the variable "VSTESTCD":



If the codelist was not assigned to any SDTM/SEND/ADaM variable, and the checkbox "Create

simple 'WhereClause' automatically" is checked, a warning message will be displayed:



If however, the codelist was already assigned (for example the VSTESTCD codelist to the VSTESTCD variable), all the "WhereClauses" will be generated automatically. These are simple "WhereClauses" with the structure:

"where *testcode* EQ '*valuelistvalue*'"

For example, for the valuelist variable "DIABP" from the codelist VSTESTCD, the "WhereClause" that is generated corresponds to:

"where *VSTESTCD* EQ '*DIABP*'"

So, let us try this out for the codelist VSTESTCD, and generate a ValueList from it. When the checkbox "Create simple 'WhereClause' automatically" is checked, this results (after a few seconds) in:

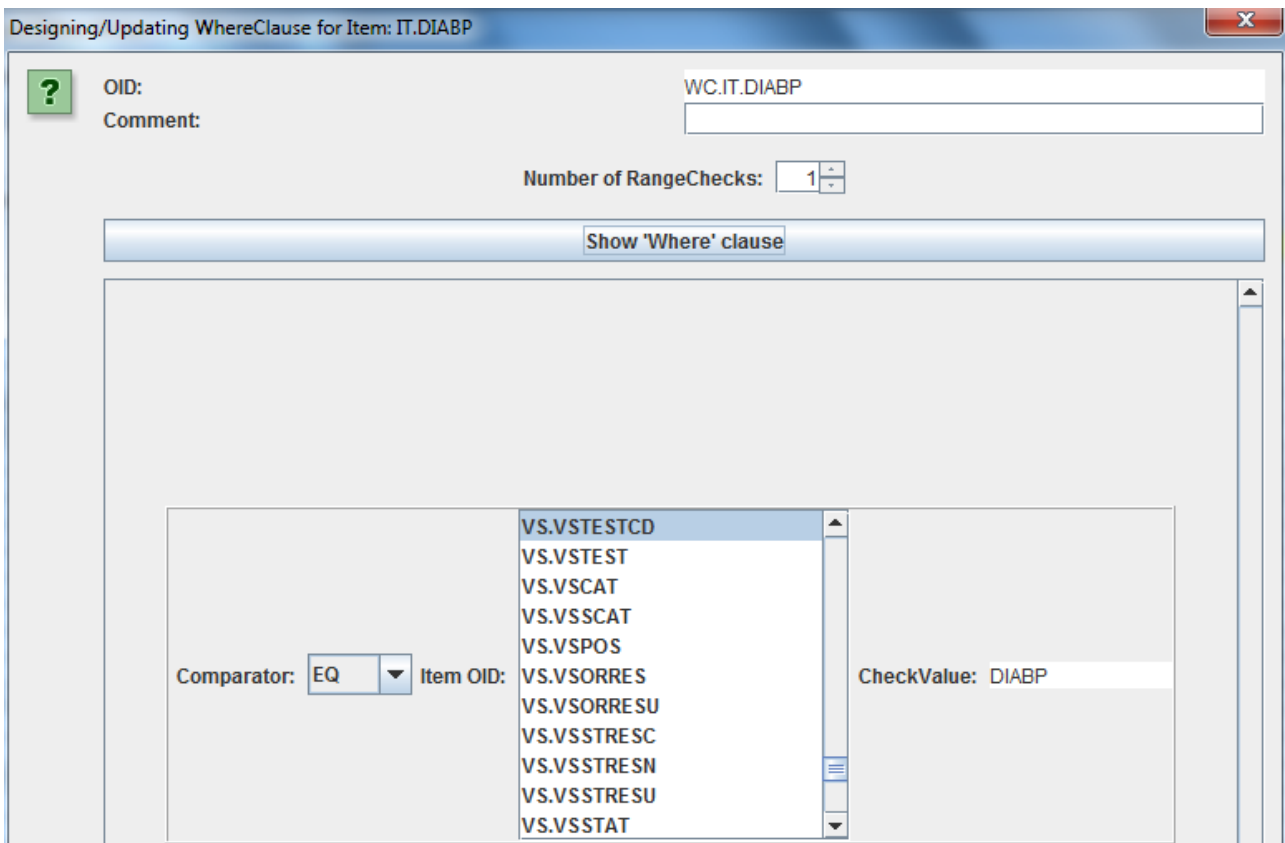
OID	Name	Data Type	Length	Sign.Digits	Origin	Comment	Description	def.DisplayFor...	Method	CodeList	WhereClause
IT.ABSKNF	ABSKNF						ABSKNF				WC.IT.ABSKNF
IT.BMI	BMI						BMI				WC.IT.BMI
IT.BMR	BMR						BMR				WC.IT.BMR
IT.BODLNTH	BODLNTH						BODLNTH				WC.IT.BODLN...
IT.BODYFATM	BODYFATM						BODYFATM				WC.IT.BODYF...
IT.BSA	BSA						BSA				WC.IT.BSA
IT.DIABP	DIABP						DIABP				WC.IT.DIABP
IT.FARMCIR	FARMCIR						FARMCIR				WC.IT.FARMC...
IT.FRMSIZE	FRMSIZE						FRMSIZE				WC.IT.FRMSIZE
IT.HDCIRC	HDCIRC						HDCIRC				WC.IT.HDCIRC
IT.HEIGHT	HEIGHT						HEIGHT				WC.IT.HEIGHT
IT.HIPCIR	HIPCIR						HIPCIR				WC.IT.HIPCIR
IT.HR	HR						HR				WC.IT.HR
IT.IDEALWT	IDEALWT						IDEALWT				WC.IT.IDEAL...
IT.KNEEHEEL	KNEEHEEL						KNEEHEEL				WC.IT.KNEEH...
IT.LBM	LBM						LBM				WC.IT.LBM
IT.MAP	MAP						MAP				WC.IT.MAP

One now notices that the "WhereClause" cell is not colored, i.e. the software generated a technically correct "WhereClause" for each valuelist entry. One can see the generated "WhereClause" by holding the mouse over the cell. For example:

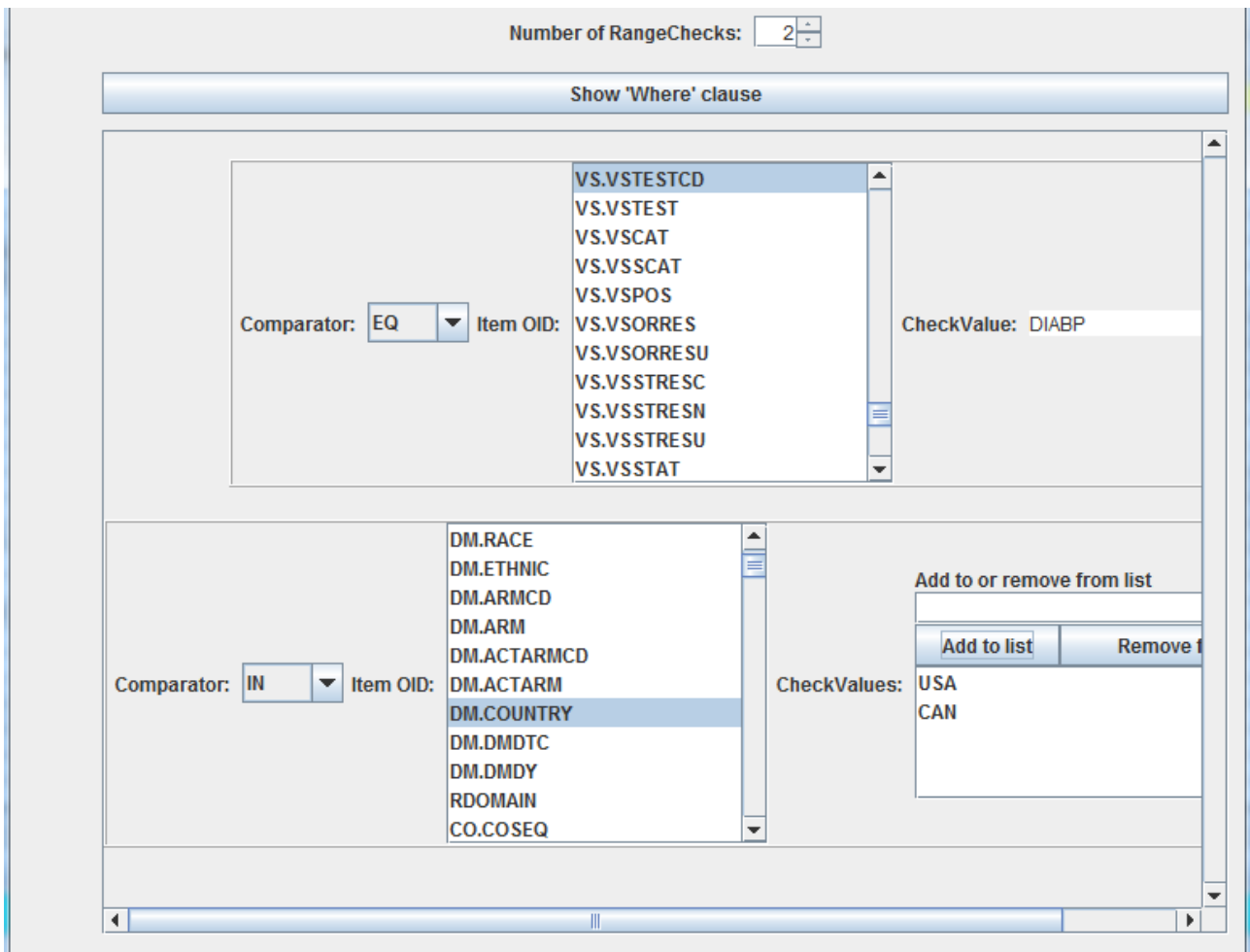
CodeList	WhereClause
	WC.IT.ABSKNF ▲
	WC.IT.BMI
	WC.IT.BMR
	where VSTESTCD EQ 'BMI'

for the cell "WC.IT.BMI"

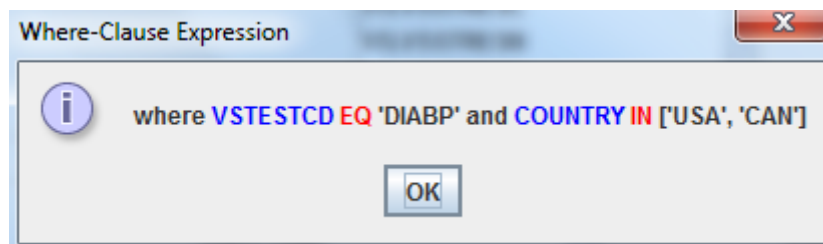
You can of course still refine the "WhereClause" by clicking on the cell. For example, for the "WhereClause" for the valuelevel variable DIABP:



For example, if you would have a "WhereClause" stating "where the VSTESTCD is 'DIABP' and the country of measurement is 'USA' or 'Canada'", you can easily do so using the wizard by adding an extra condition, like:



and when clicking the "Show 'WhereClause'" button, the "human-readable" expression is displayed:



You will probably be able to generate over 90% of the "WhereClauses" automatically in this way, but it is important to understand that checking them is necessary.

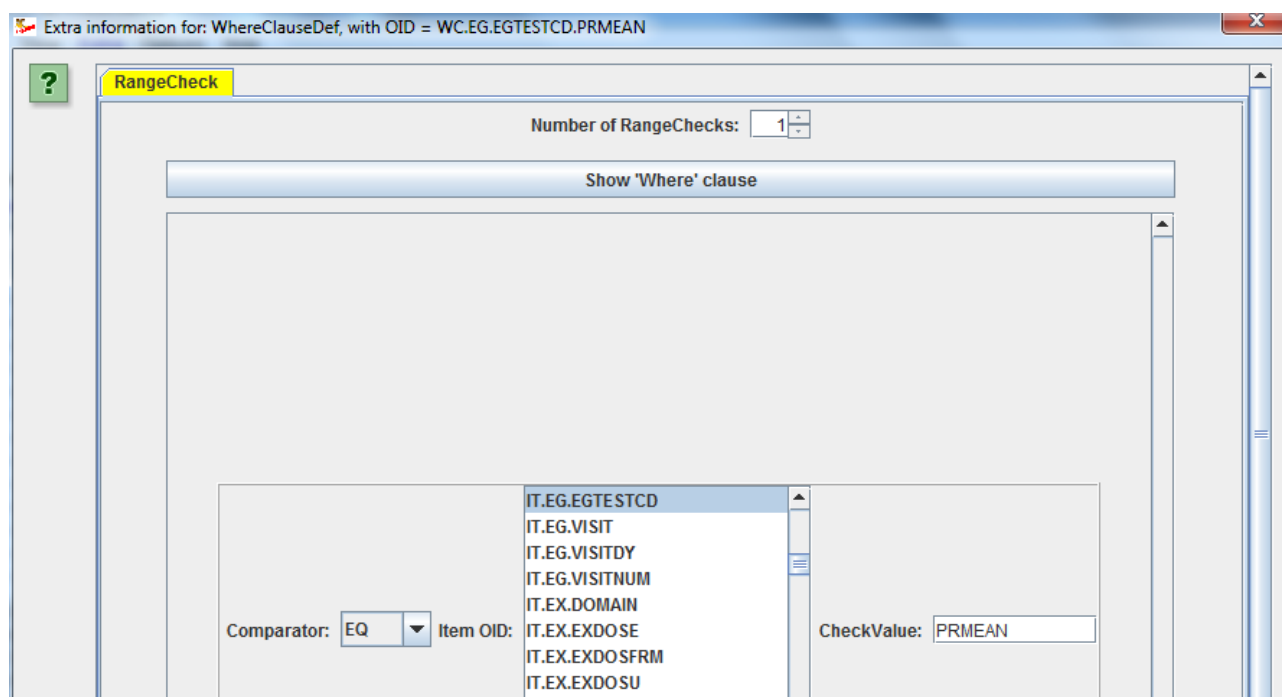
After clicking "OK" until one gets in the main window, and then selecting the tab "WhereClause Definition", one sees the newly generated "WhereClauses" in addition to any already generated:

Important remark: as generating valuelists from codelists is computing intensive, especially when also automatically generating the "WhereClauses" automatically, it is recommended to subset the codelist first to what appears in the CRFs when starting from the by CDISC published codelists. For example, the LBTESTCD codelist has several hundred entries, and it can take 5-10 minutes to convert it to a valuelist (containing the several hundred entries) when also generating the "WhereClauses" automatically. Also, as you will need to assign the datatype, lengths, codelists etc.

for each valuelist variable, it makes sense to subset the codelist LBTESTCD first to what is exactly needed, and then to create a valuelist from that subset.

Dataset Definitions	Variable Definitions	Codelists	Imputation Methods	Presentations	Condition Definitions	Method
Annotated CRFs	Supplemental Documents	ValueList Definitions	WhereClause Definitions			
		OID			CommentOID	
		WC.DA.DATESTCD.DISPAMT				
		WC.DA.DATESTCD.RETAMT				
		WC.EG.EGTESTCD.INTP				
		WC.EG.EGTESTCD.PRMEAN				
		WC.EG.EGTESTCD.QRSDUR				
		WC.EG.EGTESTCD.QTMEAN				
		WC.EG.EGTESTCD.VRMEAN				
		WC.EG.EGTESTCD.QTCB				
		WC.EG.EGTESTCD.QTCF				
		WC.IE.IETESTCD.EXCL01				
		WC.IE.IETESTCD.EXCL02				

Of course, you can then still edit any "WhereClause" here. For example for EGTESTCD "PRMEAN":

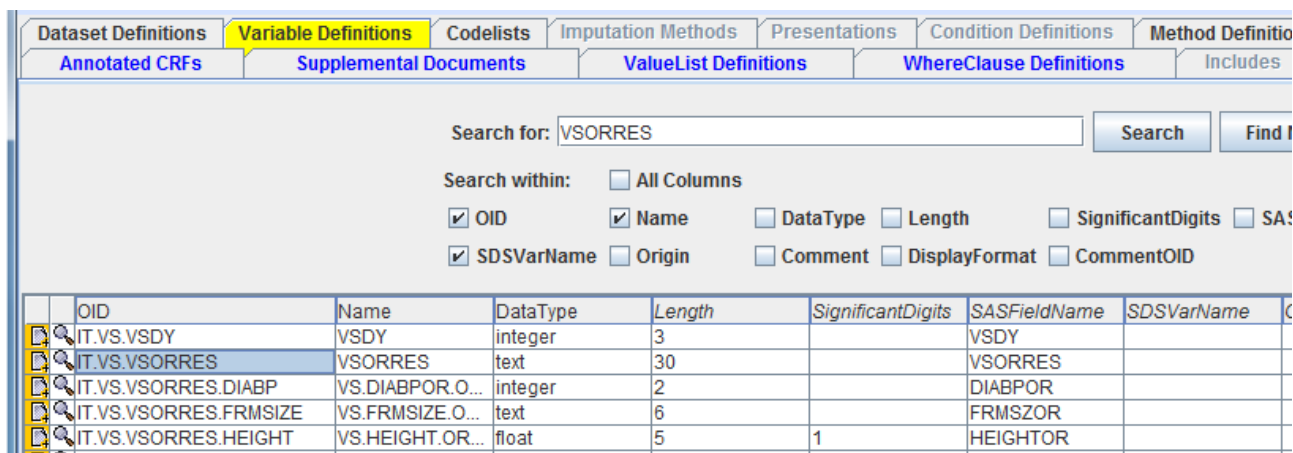


When clicking the "Edit" ("+") icon for WC.EG.EGTESTCD.PRMEAN, exactly the same wizard will show up as the one that was used during the generation of the "WhereClause". Of course one can also add new "WhereClauses" here, they do not need to be created from the "CodeList to ValueList" wizard at all.

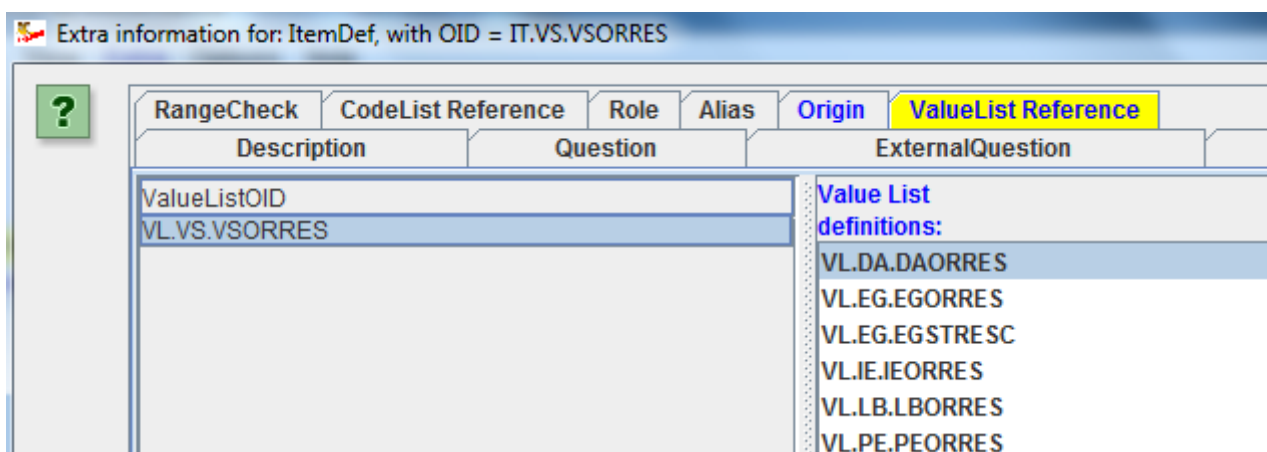
Also **do not forget** to assign valuelists to SDTM/SEND/ADaM variables themselves. For example, you will usually assign the valuelist with the different vital sign test codes to VSORRES, as this then describes the metadata of VSORRES (datatype, length, codelist) depending on the value of VESTESTCD.

If you want to assign a valuelist to VSORRESU (and other --ORRESU and --STRESU) variables, you will first need to subset the "VSRESU" (Units for Vital Signs Results) codelist or "UNIT" codelist, and then generate a ValueList from it.

For example, for assigning the previously created ValueList to VSORRES, go to the "Variable Definitions" tab, select VSORRES,



and then click the "Add Information" ("+") icon on the left of "VSORRES", and choose the tab "ValueList Reference":



where you can simply drag-and-drop from the list of existing ValueList definitions on the right to the cell on the left, or click the "ValueListOID" cell, after which a list to choose from is displayed. This is explained in detail in the section: "Assigning valuelists to SDTM/SEND/ADaM variables".

P.S. As ValueLists only have an OID (but no "Name" attribute), it may be useful to assign a meaningful OID to each entry in the "ValueList" table.

Annotated CRF and Supplemental Documents

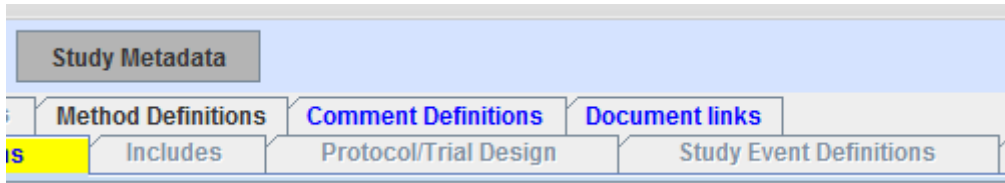
When doing a regulatory submission, you are still required to provide an "annotated CRF" in PDF format²⁴. Also, you will often want to supply supplemental documents, like a "reviewers guide", also in PDF format.

²⁴It would be much better if the requirements were that an ODM file with the study design must be delivered, where the questions are annotated with SDTM information. ODM is machine-readable, PDF isn't.

In define.xml, you can provide links to such documents, so that when the reviewer inspects your define.xml in the browser, a single click suffices to open the annotated CRF or supplemental document, ideally on the page of interest.

This is taken care of in define.xml documents by so-called "def:leaf" elements, allowing you to specify a reference ("href") to the document of interest.

In order to provide such a leaf, click the tab "Document Links":



The following table is displayed:

ID	href

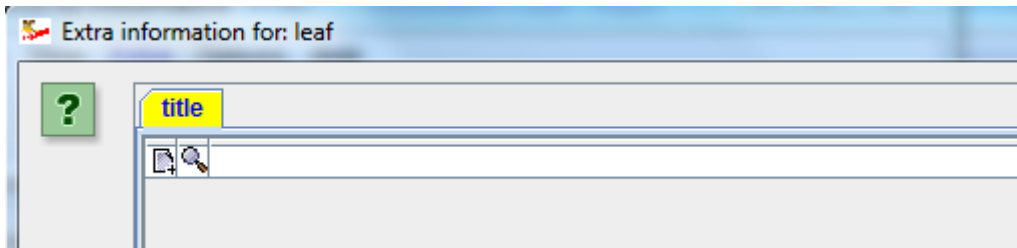
you can now provide an ID and a reference (usually just the file name) for each document. For example, to define a link to an annotated CRF and to a "reviewers guide":

ID	href
LF.aCRF	aCRF.pdf
LF.REFGUIDE	Reviewers_Guide.pdf

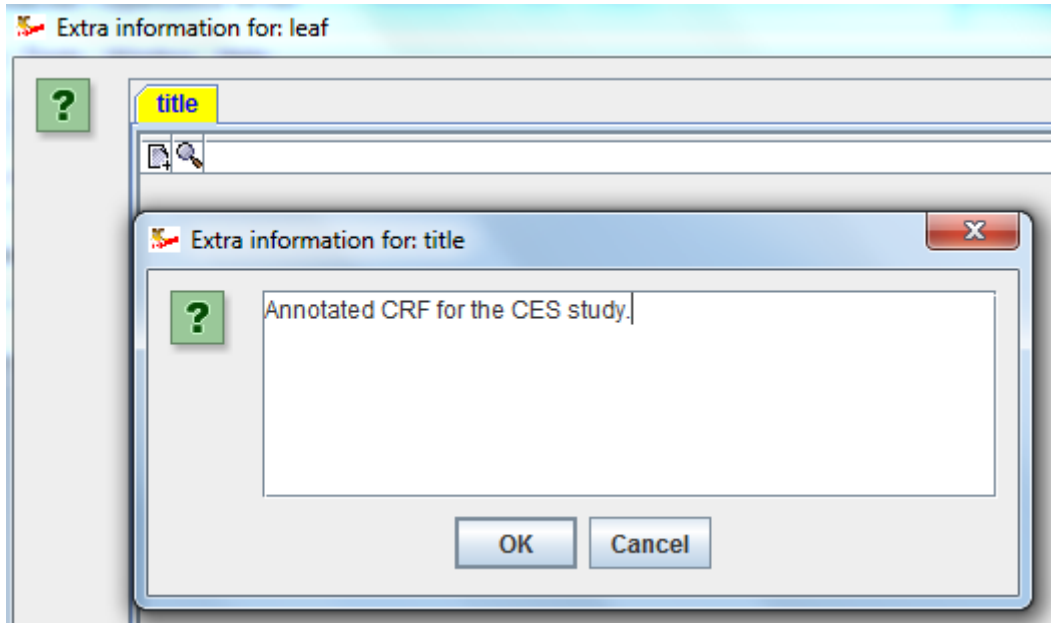
When you then click the "Validate" button, you will notice that some information is still missing:

ID	href
LF.aCRF	aCRF.pdf
LF.REFGUIDE	Reviewers_Guide.pdf
Invalid content	

So you will need to provide additional content. This can be provided by clicking on the "+" icon, leading to:



prompting you to provide a title by clicking the "+" icon again:



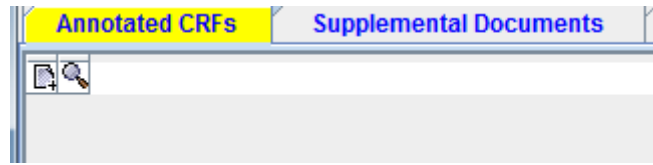
The same should then be done for the "reviewers guide".
Revalidating then leads to:

ID	href
LF.aCRF	aCRF.pdf
LF.REFGUIDE	Reviewers_Guide.pdf

Both "+" icons turned orange, meaning that additional information has been added and there are no validation errors.

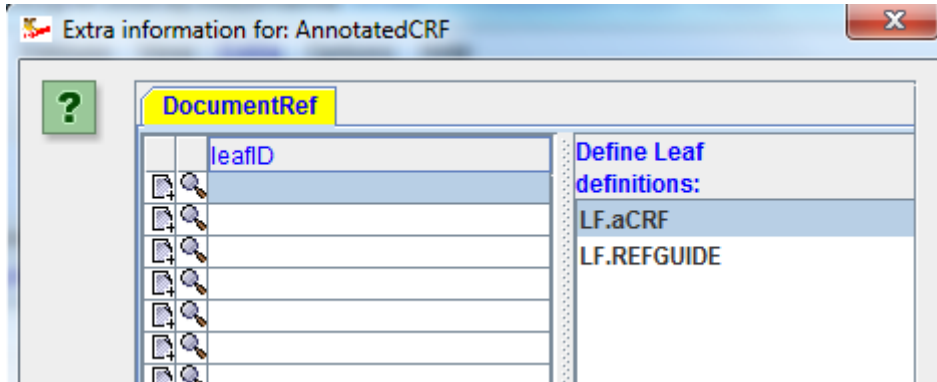
We now only defined the "leafs", i.e. the hyperlinks and a title for each of the documents, but the define.xml still requires us to state explicitly which of these is the "annotated CRF" (define.xml is machine-readable and machines should not rely on naming conventions only), and which documents belong to the "Supplemental Documents".

In order to do so, click the tab "Annotated CRFs":

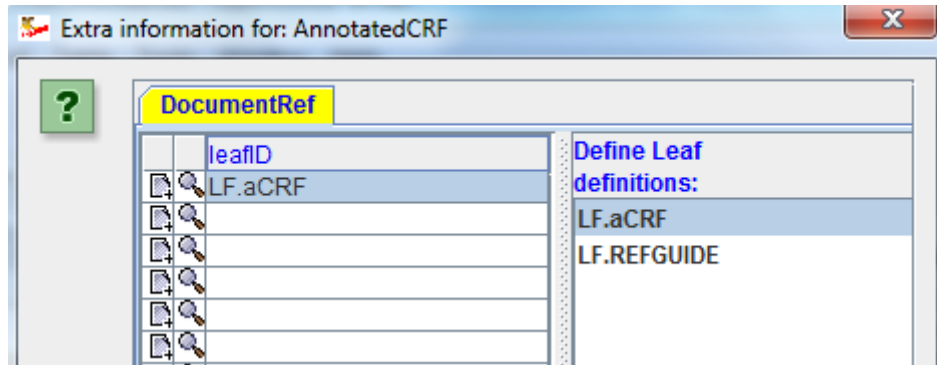


Currently, it is only allowed to provide a single PDF with all annotated CRFs in it (separate documents, e.g. one for each CRF, is not allowed).

Clicking the "+" icon leads to:

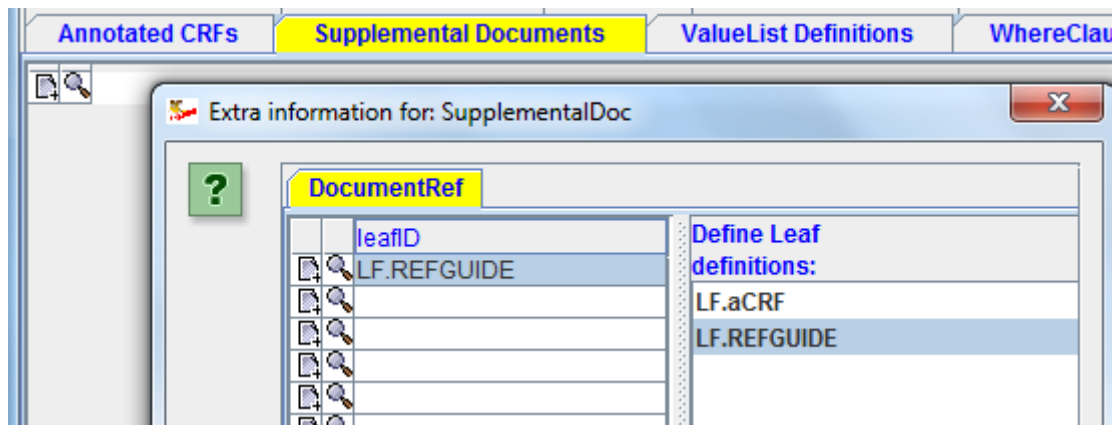


and one can now "drag-and-drop" a "leaf" from the right to the first cell on the left, with the result:



Click OK to finalize the process of assigning an "annotated CRF" to the study submission.

The same can then be repeated for any other documents like the "reviewers guide", by using the tab "Supplemental Documents":



Remark that you can add more than one supplemental document.

Assigning valuelists to SDTM/SEND/ADaM variables

ValueList are used for informing the reviewers about the metadata for individual groups of data. For example, for the Laboratory domain (LB), there can be hundreds of tests, and the information for each test can differ depending on the test code. For example, albumin measurements will have other units as glucose units.

Essentially, most findings domains have "hypervertical" structure, with a data model based on the "Entity - Attribute - Value" model (EAV model) This means that we have a single "entity" column (defining what the row "is about"). In the "Findings" domains, this is usually the --TESTCD column²⁵. There then are a number of attributes (like additional names and synonyms, but most importantly the --ORRESU (original result unit) and the --STRESU (standardized result unit). The "value" in EAV is delivered by the --ORRES column and by the --STRESC column and or the -STRESN column²⁶.

We already generated a valuelist "VL.CL.C66741.VSTESTCD" containing additional information (metadata) for 9 value-level vital signs variables. Currently, this valuelist is still "on its own", and we need to provide the information to which SDTM/SEND/ADaM variable it applies.

The define.xml 2.0 specification is very clear about this (section 4.4):

Value Level Metadata should be provided when there is a need to describe differing metadata attributes for subsets of cells within a column. It is most often used on SDTM Findings domains to provide definitions for Variables such as --ORRES, --ORRESU, --STRES, --STRESU that are specific to each test code (value of --TESTCD). It is not required for Findings domains where the results have the same characteristics in all records, such as IE domains. In ADaM, value level metadata often describes AVAL or AVALC in BDS data structures based on values of PARAMCD.

In our case, we provided information about what length of result can be expected, what datatype, and whether the result is coded, depending on the value of the VSTESTCD. So, we must assign the valuelist "VL.CL.C66741.VSTESTCD" to VSORRES.

If the value of the standardized result (VSSTRESC/VSSTRESN) is just copied from VSORRES, we can additionally also assign this valuelist to VSSTRESC and/or VSSTRESN.

²⁵A better choice would have been the --LOINC column, at least for LB and VS.

²⁶The reason that two columns are defined for the standardized result is due to the extremely old SAS-XPT format, in which columns can only either be "numeric" or "character".

In order to assign the valuelist "VL.CL.C66741.VSTESTCD" to VSORRES, select the tab "Variable Definitions" and look for "VSORRES" (one can use the "Show Search Panel"). One easily finds the row:

OID	Name	Data Type	Length	SignificantDi...	SASFieldNa...	SDSVarName	Origin
VS.VSORRES	VSORRES	text	80				
VS.VSORRE...	VSORRESU	text	11				
VS.VSSTRE...	VSSTRESC	text	80				

Depending on how the variable definitions were created, you will still want to assign another "Length". The above picture shows a maximal length of "80" coming from the template SDTM. If you generated the variable definitions from a SAS-XPT file, the maximal length will probably already be correct, unless of course that you generated your SAS-XPT files with fields that are "too broad", something that the FDA doesn't like at all²⁷.

Suppose that the longest result for VSORRES was 20 characters. In that case we need to set the "Length" to 20, and also take care that this field is not longer than 20 bytes in the SAS-XPT file:

OID	Name	Data Type	Length	SignificantDi...
VS.VSORRES	VSORRES	text	20	
VS.VSORRE...	VSORRESU	text	11	
VS.VSSTRE...	VSSTRESC	text	80	

Remark that max. Lengths have to be assigned to all variable definitions, except for when the data type is a date, time or datetime or an incomplete version of one of these three.

In order to assign the valuelist to VSORRES, click the "+" icon. This leads to:

Language	Translated Text
en	Result or Finding in Original Units

In this case, the "Label" was already added (in define.xml 2.0, it is added to the "Description" element). Be sure that the label for any domain variable is exactly identical (case sensitive!) to what is defined in the SDTM-IG, SEND-IG, or ADaM-IG.

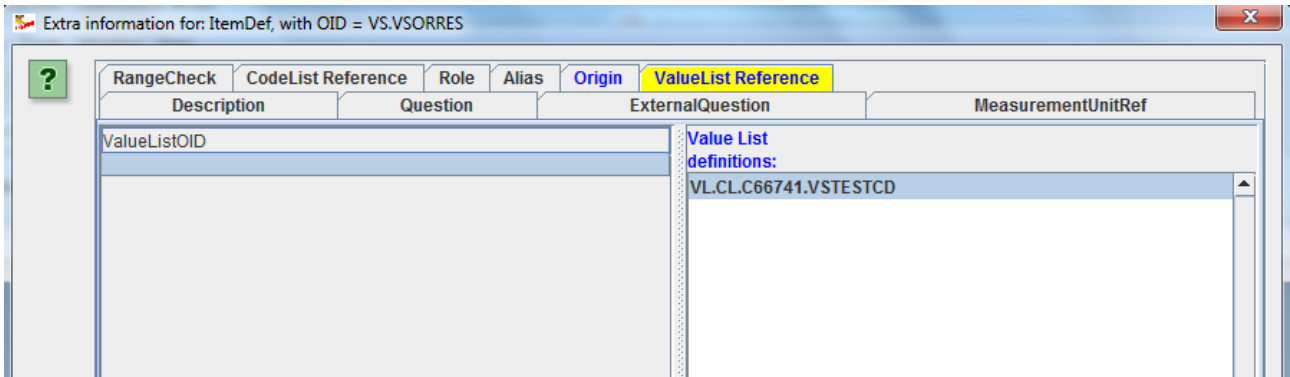
Unfortunately, there is no room for any deviation at all here, even if a slightly different label would better explain what the variable is about. See the article

²⁷This is another relict of the SAS-XPT era. When using XML instead of SAS-XPT, there would be no problem at all.

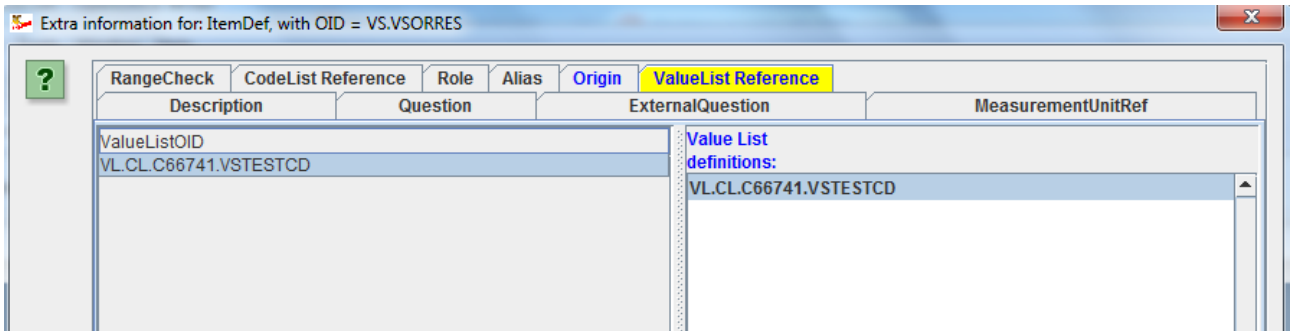
"<http://cdiscguru.blogspot.com/2015/12/sdtm-labels-freedom-or-slavery.html>" for further details.

Getting the label 100% correct can especially be challenging in case of additional variables that come from the model rather than from the implementation guide (like additional timing variables) and when coming from the SAS-XPT files themselves. In future, we will use the "CDISC Library API" to check labels against the published ones from CDISC.

Another tab in this dialog is the "ValueList Reference" tab. When clicked, the following appears:



Until now, we only defined one valuelist, so there is only one entry at the right. In order to state that the valuelist "VL.CL.C66741.VSTESTCD" applies to the variable "VSORRES", just drag-and-drop it from the right to the left, leading to:



and you are done...

Later, when inspecting the variable VSORRES in the browser, this will then appear as:

Value Level Metadata - VS [VSORRES]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
VSORRES	<u>VSTESTCD</u> = "BMI" (Body Mass Index) and =	float	5			Weight (kg) divided by height (m) squared
VSORRES	<u>VSTESTCD</u> = "DIABP" (Diastolic Blood Pressure) and =	integer	3			DIABP comment
VSORRES	<u>VSTESTCD</u> = "FRMSIZE" (Body Frame Size)	text	8	["LARGE", "MEDIUM", "SMALL"] <Size>		
VSORRES	<u>VSTESTCD</u> = "HEIGHT" (Height)	integer	3			
VSORRES	<u>VSTESTCD</u> = "HR" (Heart Rate)	integer	3			
VSORRES	<u>VSTESTCD</u> = "SYSBP" (Systolic Blood Pressure)	integer	3			
VSORRES	<u>VSTESTCD</u> = "TEMP" (Temperature)	float	5			
VSORRES	<u>VSTESTCD</u> = "WEIGHT" (Weight)	float	5			
VSORRES	<u>VSTESTCD</u> = "WSTCIR" (Waist Circumference)	integer	3			

Assigning an Origin to variables

Every data point in a submission has an origin. It is of utmost importance for traceability that this information is provided to the reviewer. It can be that the data point comes from the CRF, it can come from the protocol, or it can be assigned or derived (e.g. all --DY variables). It can also come from an electronic data transfer (e.g. some laboratory data that were not captured on a CRF), or (usually in case of ADaM), it can come from a "predecessor", e.g. when the data point was copied from SDTM into ADaM.

The origin can either be assigned on the "domain variable level" or on the "valuelist level". For example, "STUDYID" will always be identical and will probably come from the protocol. So we can assign it at the domain variable level.

On the other hand, "VSORRES" will need to be assigned on the "valuelist level", as there usually is no field "vital sign original result" on the CRF, but there are fields "systolic blood pressure", "systolic blood pressure", "height", "weight" and so on.

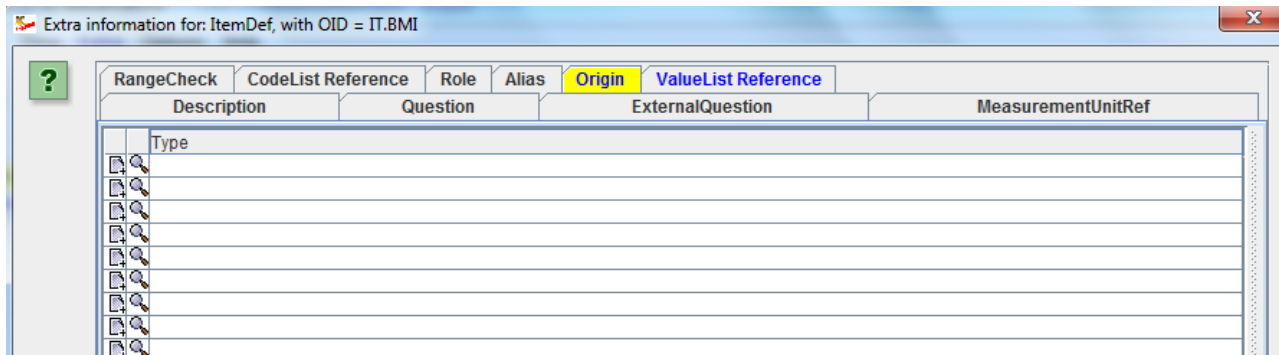
So it does not make sense (it even would probably be an error) to assign an Origin to VSORRES. However, as we assigned a valuelist to "VSORRES", this is already an indication that origins will be assigned on the valuelist level, i.e. for "DIABP", "SYSBP", "HEIGHT" etc..

So in our variable list, we will leave the value for "Origin" empty for VSORRES, but will assign origins for the valuelist variables "DIABP", "SYSBP", "HEIGHT" etc..

In order to assign an origin to e.g. the vital sign "BMI", look for it in the list of "Variable Definitions":

QNAM	QNAM	text	80		
QLABEL	QLABEL	text	80		
QVAL	QVAL	text	80		
QORIG	QORIG	text	80		
QEVAL	QEVAL	text	80		
IT.BMI	BMI	float	5		

Then click the "+" icon to add additional information, and select the "Origin" tab:

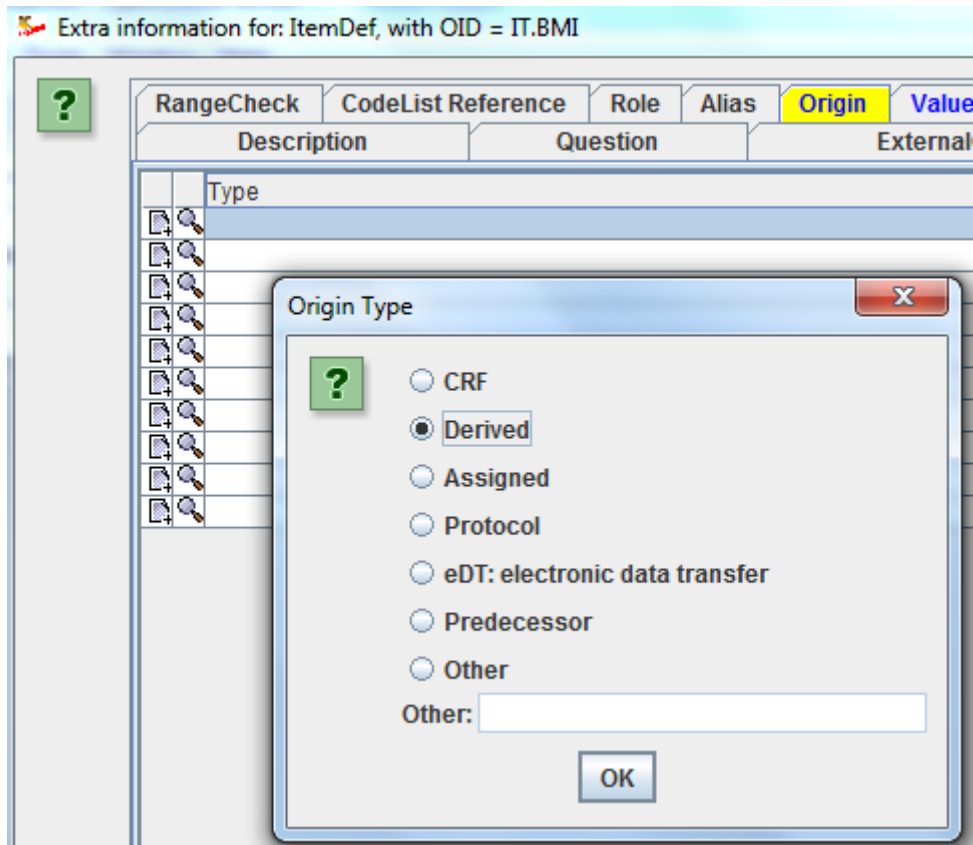


Although the define.xml XML-Schema allows you to add more than one origin, this is essentially not allowed. If you need to assign more than one "origin", this usually means that you did not well enough develop your valuelist.

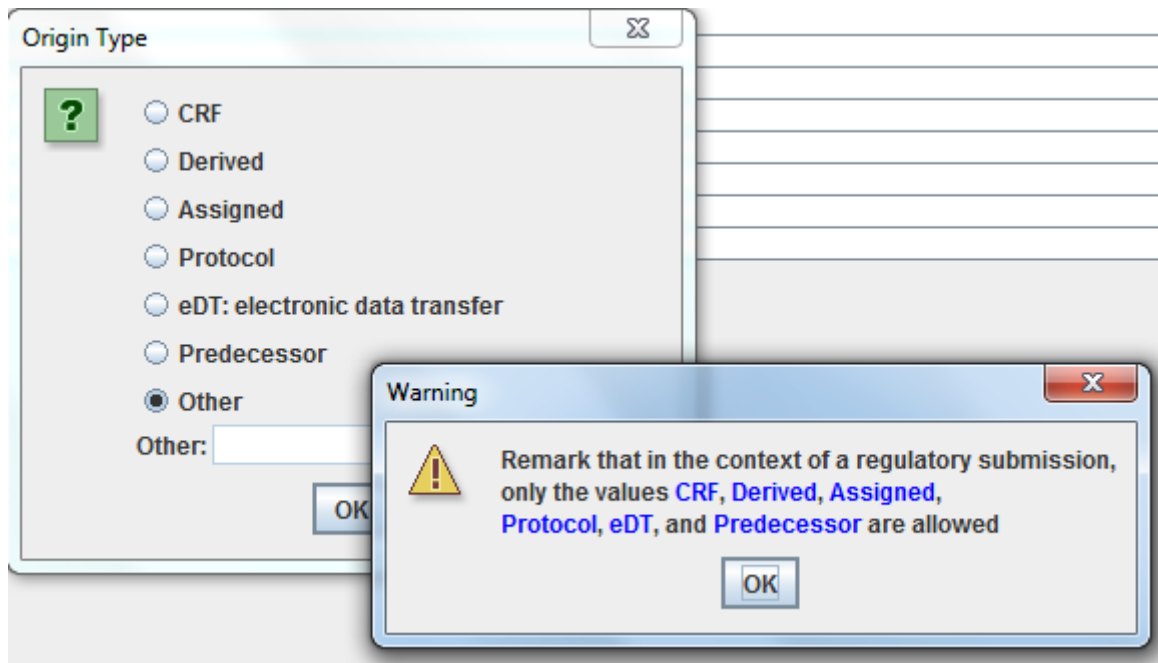
For example, if you have have different glucose values, and one set was delivered electronically by lab 1, and the other set was captured on the CRF, you may want to have two different variables "IT.GLUC.LAB1" and "IT.GLUC.LAB2" so that you can assign "electronic data transfer" to the former as being the origin, and "CRF" to the latter²⁸. In the "where clause" you will then need to differentiate e.g. by the "DM.SITE" or "LB.NAM" (laboratory name) variable.

But now back to our BMI valuelist variable. In this case the origin is "derived", as the value was calculated from the weight and height. So we assign "Derived" as the type, by clicking in the "Type" field which leads to the following dialog:

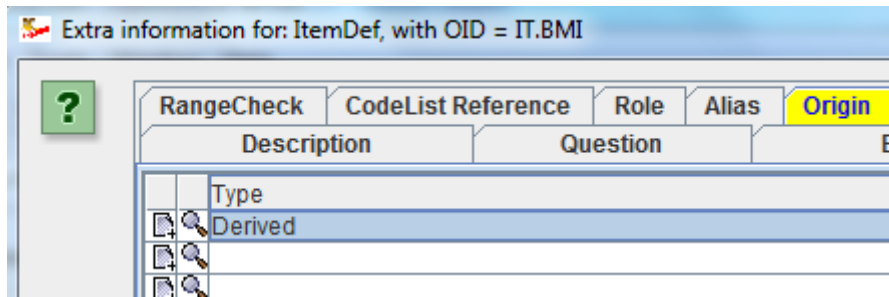
²⁸Unfortunately, the situation for lab values can be even much more complicated, as CDISC and the FDA still refuse to allow the LOINC code of the lab test as the unique identifier.



Remark that the option "Other" is essentially not allowed in the case of a regulatory submission. If one selects this option, the following warning is shown:

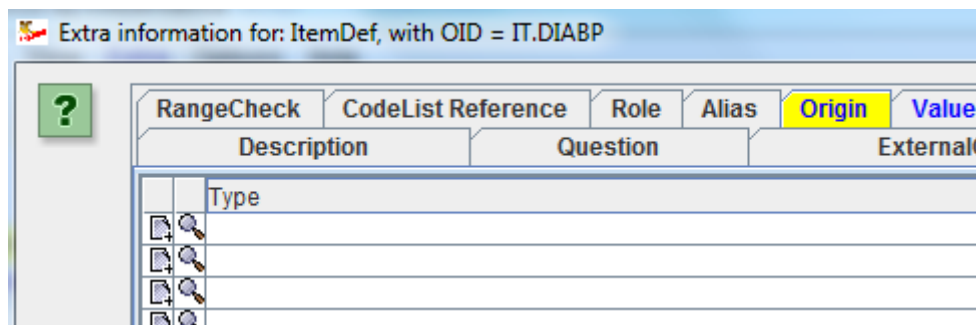


In our case however, we set the "Origin Type" to derived, leading to:

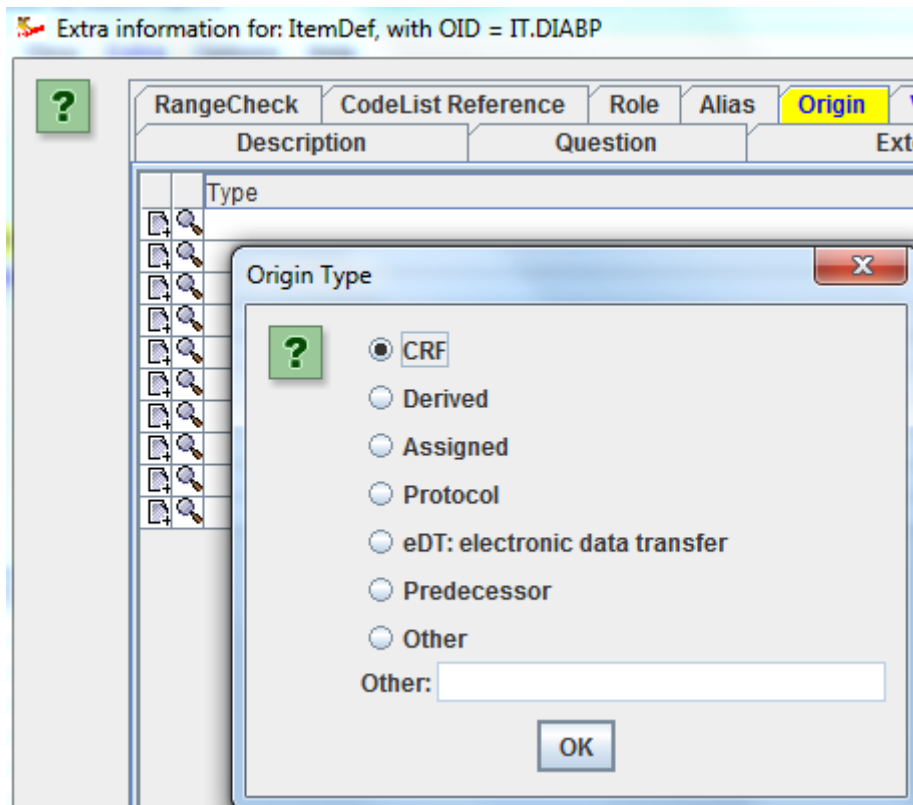


When the type is "Derived" we also need to add the derivation method. We already saw how that can be done when creating a "ValueList" from a "CodeList" and will revisit this topic later.

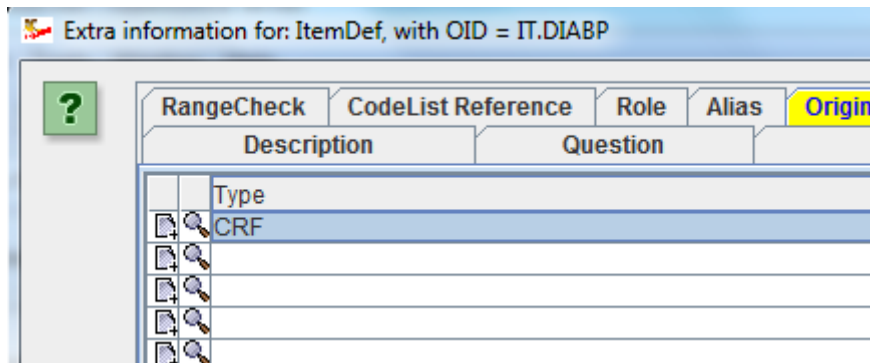
For the "diastolic blood pressure", the origin is the CRF. So, when drilling into the details for "DIABP" we get:



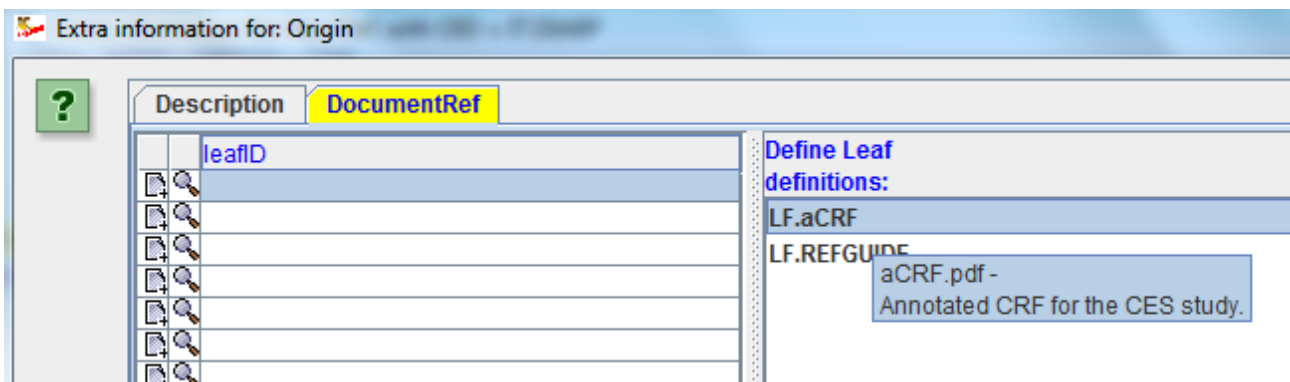
When then clicking the "Type" field, we select "CRF":



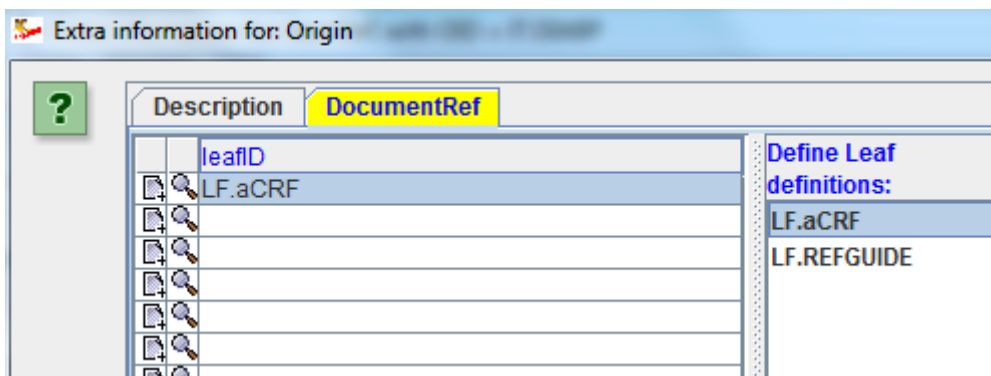
leading to:



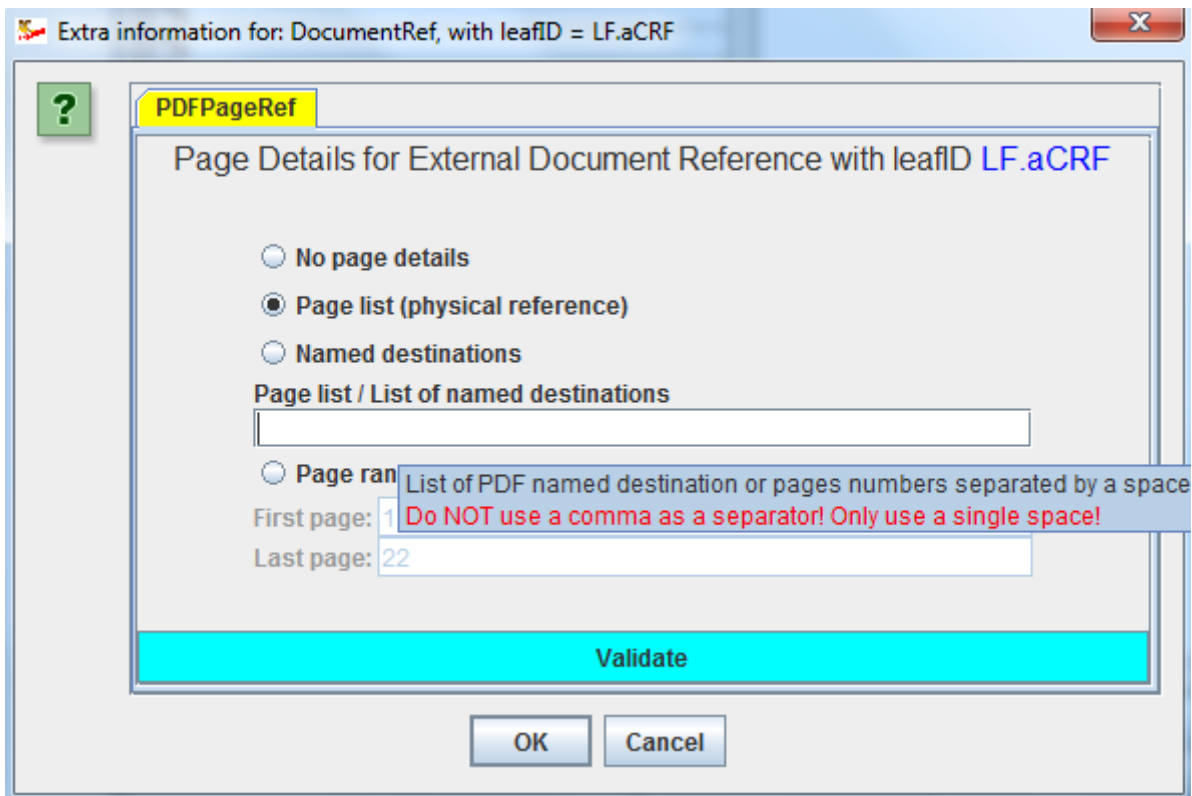
Clicking the "+" icon then allows to define where the field for "diastolic blood pressure" can be found in the annotated CRF:



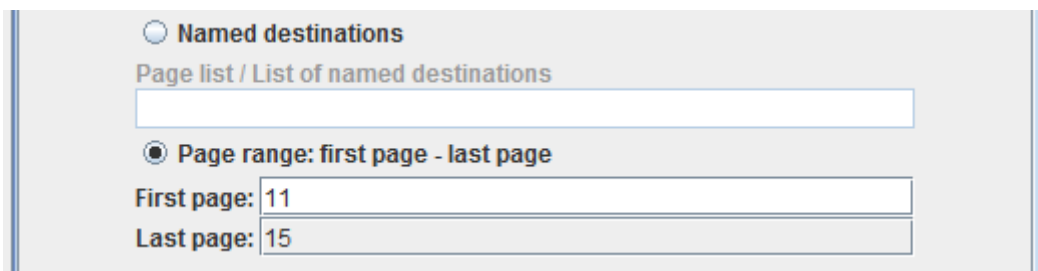
We state that the information can be found in the document defined by the "leaf" "LF.aCRF" by doing a drag-and-drop from the right to the left:



and the further details about "where" in the annotated CRF by clicking the "+" icon:

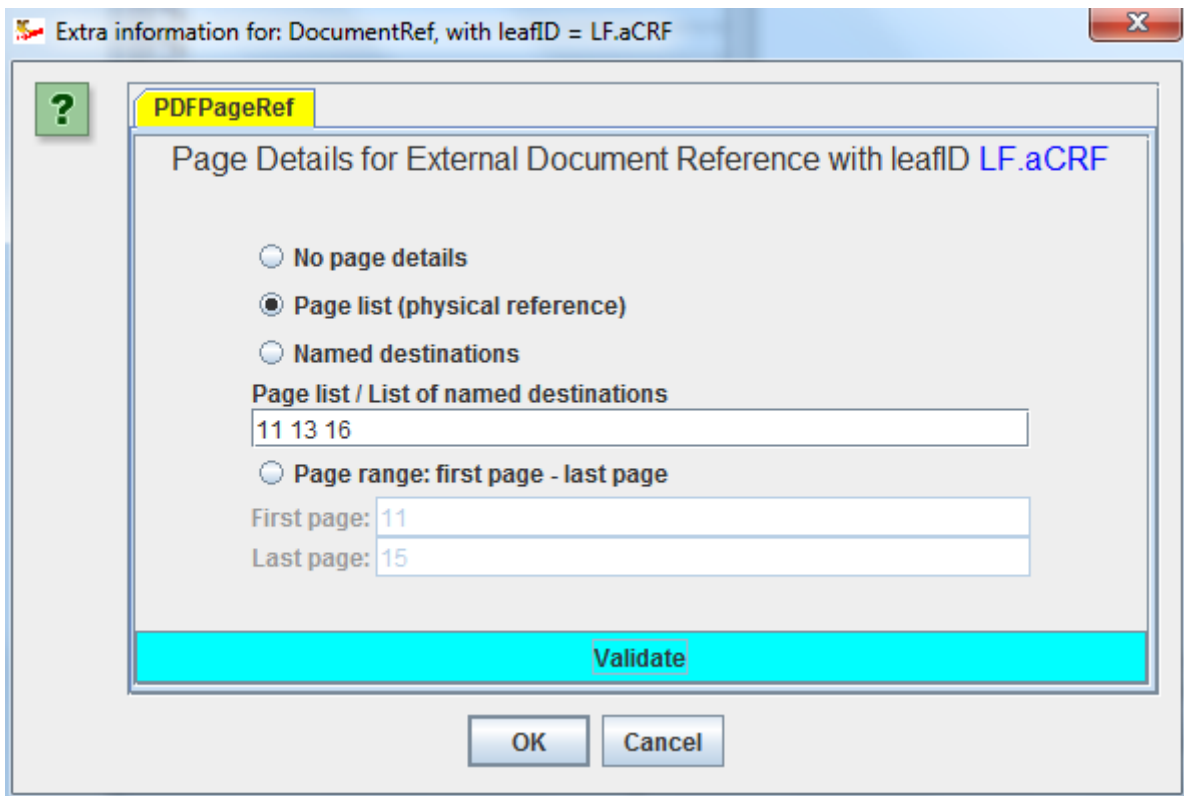


We can either provide no page details at all, or a page list (one or more page numbers separated by a space), or a page range:



The "Validate" button helps us avoiding making syntactic errors (like using comma-separated page numbers).

Suppose that the diastolic blood pressure was collected in different forms, which appears as pages 11, 13 and 16 in the "annotated CRF PDF". The selection would then be:



After clicking the OK button, the information is stored in the define.xml.

When viewing the define.xml in the browser, the result then is:

Value Level Metadata - VS [VSORRES]

Variable	Where	Type	Length / Display Format	Controlled Terms or Format	Origin	Derivation/Comment
VSORRES	VSTESTCD = "BMI" (Body Mass Index) and =	float	5		Derived	Weight (kg) divided by height (m) squared
VSORRES	VSTESTCD = "DIABP" (Diastolic Blood Pressure) and =	integer	3		CRF Pages 11 13 16	DIABP comment
VSORRES	VSTESTCD = "FRMSIZE" (Body Frame Size)	text	8	["LARGE", "MEDIUM", "SMALL"] <Size>		

When the user than e.g. clicks on "11" in "CRF Pages", then the browser should automatically open the "annotated CRF PDF" on page 11²⁹.

Like this, one should essentially assign an "Origin" to every variable in the define.xml. If the variable is a domain variable to which a valuelist has been assigned, one will usually assign the "Origin" to the value-level variables rather than on the parent domain level variable (as was the case for VSORRES).

For developers of define.xml, it is not always clear what "Origin" should be assigned to variables that do not represent data from a CRF (e.g. "STUDYID", "DOMAIN", "--CAT", "--DY"). Therefore, a list is provided in the appendix with typical "Origin" assignments for SDTM variables. Please remark that these assignments are suggestions only, and not binding at all.

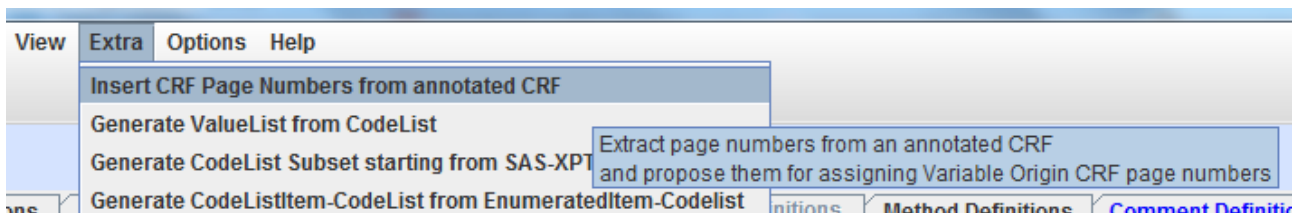
²⁹Remark that this functionality is (or must be) provided by the stylesheet. It is not programmed in the define.xml itself.

Retrieving PDF Page Numbers from the annotated CRF

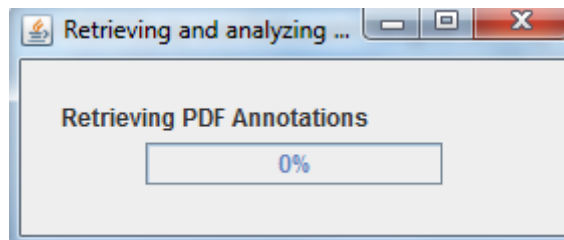
Getting the PDF page numbers from the annotated CRF into the define.xml can be a tedious task that is error prone. Therefore we developed a feature to extract annotations from the annotated CRF and compare them with the variable name definitions (ItemDefs) and the "WhereClause" definitions. This is not a "black box" feature as in some other software packages: before doing the PDF page assignments, the user can still inspect the suggestions, and select the ones to be implemented.

Of course, you will first need to add the location of the annotated CRF to the define.xml using the menus "Add - Document Link" and "Add - Annotated CRF".

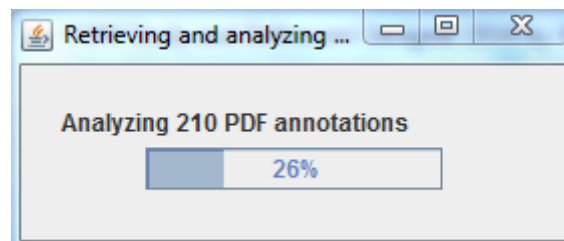
Once this link is present, use the menu "Extra - Insert CRF Page Numbers from annotated CRF":



The system will then look up the location of the annotated CRF you already provided and start retrieving all annotations:

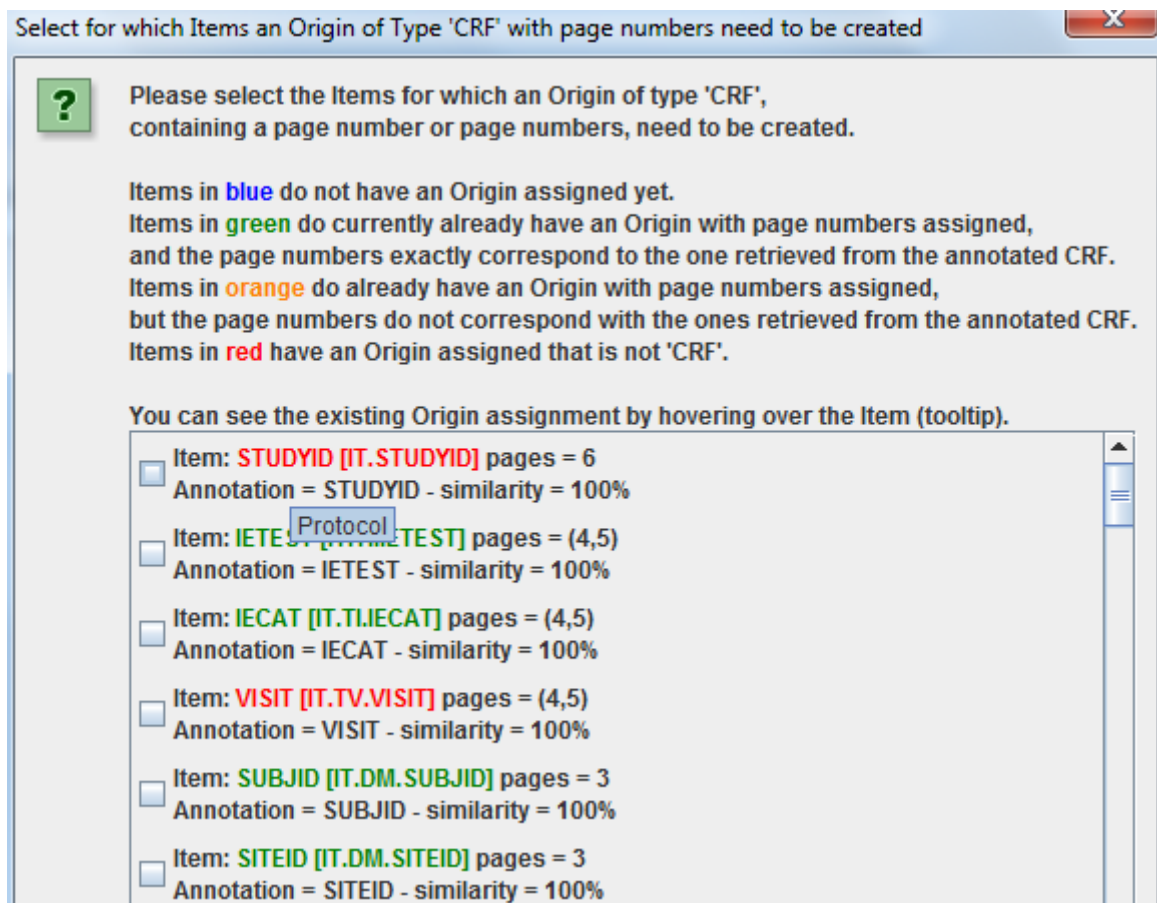


It will then analyze them, comparing them with all define.xml variable names that have been defined, but also with all "WhereClause" definitions.



For each, a "similarity" is calculated: if it is 100, meaning that the annotation exactly matches the variable name from the define.xml or the "WhereClause" definition (as human-readable text). For the latter, similarities of over 60-70% usually mean a "hit".

When the analysis is ready (this can take 1-2 minutes in the case of a large study - but you probably only need to do this once), the following dialog is presented:



For each annotation that could be matched to a variable (or to a set of variables that have the same "WhereClause"), a checkbox will appear in the dialog.

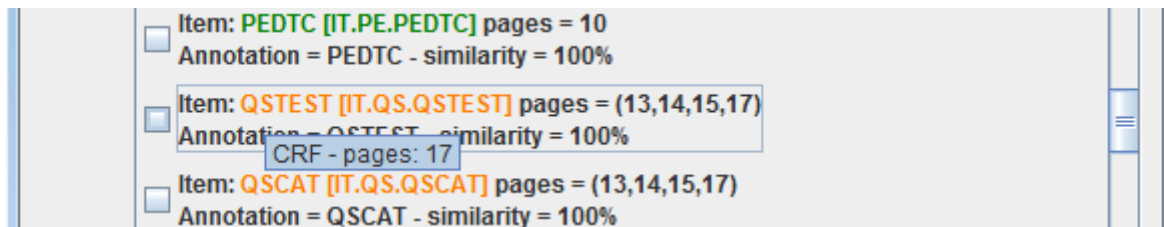
When the Item is colored green, this means that the Origin was already assigned in the define.xml as of type "CRF" and the page number(s) or the page range³⁰ exactly match with the page numbers retrieved from the annotated CRF. In this case you will probably not want to reassign the page number(s) in the define.xml.

In case the Item is colored red, this means that the "Origin" in the define.xml was not assigned to be "CRF", but something else. In the above example, the origin was assigned to be "Protocol", but a "STUDYID" annotation was found in the annotated CRF on page 6.

In case the Item is colored blue, this means that no origin was assigned yet in the define.xml. All items will be colored blue e.g. in case that you have not assigned any origins yet, and first want to automatically assign the ones that are of type "CRF" and are retrieved from the annotated CRF.

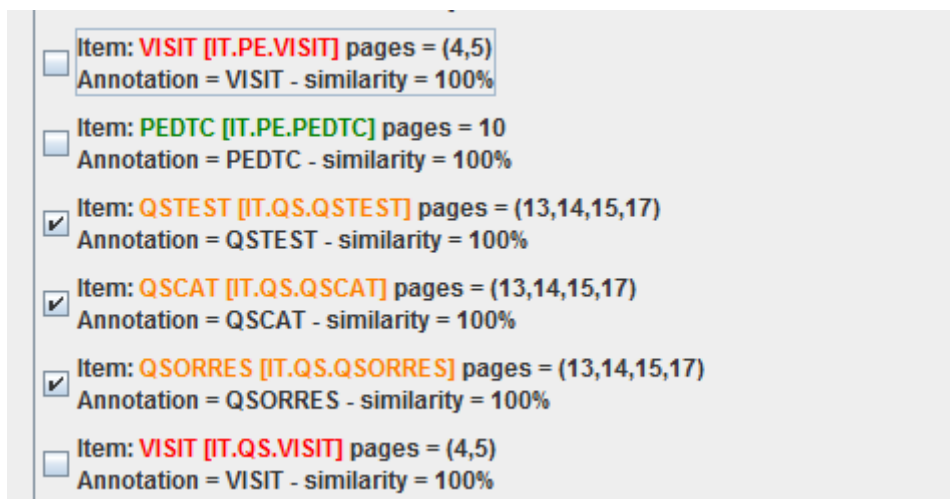
In case the Item is colored orange, this means that page numbers were retrieved from the annotated CRF, but these do not coincide with the ones that were already assigned. For example:

³⁰The algorithm takes into account that the define.xml def:Origin has a "FirstPage" and "LastPage" assigned. For example if in the define.xml one already FirstPage="6" and LastPage="10", and the page numbers from the annotated CRF are 6, 7, 8, 9, and 10, this is found to be a perfect match.



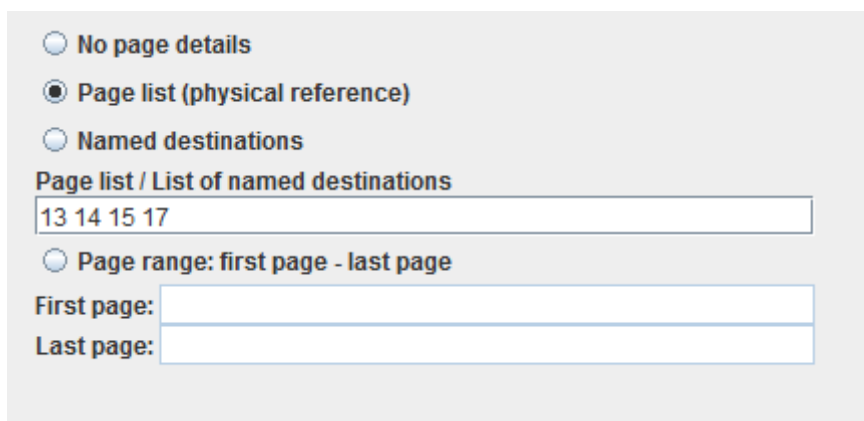
For QSTEST, an 100% matching annotation was found in the annotated CRF on pages 13, 14, 15 and 17, but in the define.xml, only page number 17 was assigned.

You can now check the checkboxes for the items for which you want to auto-generate the "def:Origin" page numbers in the define.xml, based on your considerations of what is correct and meaningful. For example:



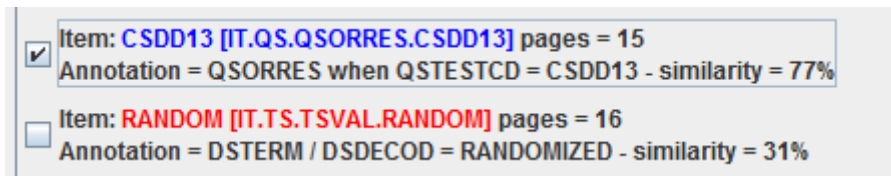
The "green" ones were already correctly assigned, so you don't want to change that anymore. For PE.VISIT and QS.VISIT you decide that the earlier assignment "derived" was correct, this although there as such annotations on pages 4 and 5. So you leave these unchecked too. Just for QSTEST, QSCAT and QSORRES, you decide to implement the automated generation of a "def:Origin" in the define.xml.

After clicking OK, the assignments are implemented. If one then does an "Origin" inspection for e.g. QSORRES, one will find:



with these page numbers extracted from the annotated CRF.

The same also works for "ValueList" variables, for example:

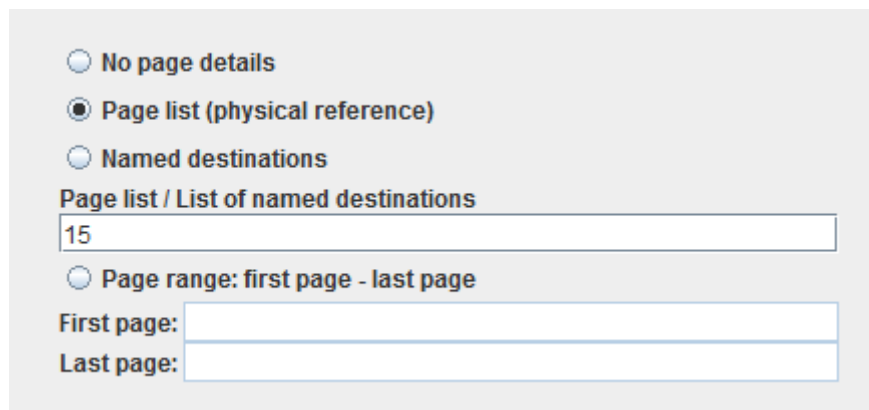


A "match" has been found for the CDSS13 "ValueList" variable with a high "similarity", for which

D. CYCLIC FUNCTIONS			
		<i>QSORRES when QSTESTCD = CSDD12</i>	
12. DIURNAL VARIATION OF MOOD Symptoms worse in the morning	0	1	2
		<i>QSORRES when QSTESTCD = CSDD13</i>	
13. DIFFICULTY FALLING ASLEEP Later than usual for this individual	0	1	2
		<i>QSORRES when QSTESTCD = CSDD14</i>	
14. MULTIPLE AWAKENINGS DURING SLEEP	0	1	2
		<i>QSORRES when QSTESTCD = CSDD15</i>	
15. EARLY MORNING AWAKENING Earlier than usual for this individual	0	1	2

the PDF annotation is "QSORRES when QSTESTCD = CSDD13".

In this case, no CRF pages have been assigned yet in the define.xml (the item is colored blue), so we would like to have this done, we do check the checkbox and click "OK"



The result upon inspection is:

Conclusion: this new feature in the ODM-Define-XML Designer allows you to extract page numbers from the annotated CRF semi-automatically, in such a way that the user still has full control of which page numbers are copied to the define.xml.

This feature will save you many hours of tedious and error prone work, for large studies maybe even days.