The Define.xml Designer 2026

User Manual for designing and developing Define.xml files in a modern way

Author: XML4Pharma
Last update: 2026-02-03

Table of Contents

Introduction

This user manual describes the features for designing and developing define.xml v.2.1 and 2.0 files.

All too often, define.xml files for regulatory submissions are generated after the CDISC SDTM, SEND or ADaM files have been generated, i.e. in a "post-process" step, mostly leading to low-quality define.xml files. Even in the cases that the define.xml is generated before the study starts, i.e. as a "requirements document" for the datasets to be developed,

this is based on setting up Excel tables, and using "black box" software[1] in a "trial-and-error" method.

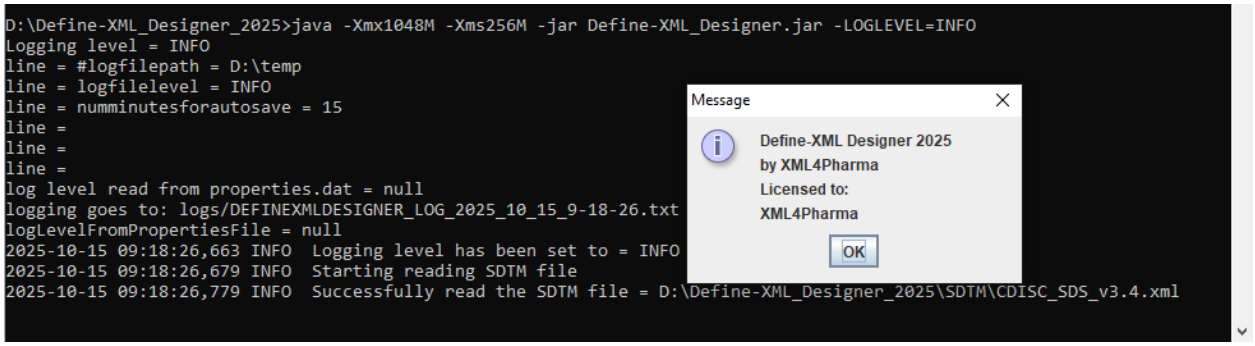
The "Define-XML Designer" software allows to develop and fine-tune define.xml files starting either from templates for the different SDTM, SEND and ADaM versions, from SAS-XPT datasets (and in future also from CDISC Dataset-JSON), or from an existing define.xml file, and this in a very user friendly "WYSIWYG" (What You See Is What You Get) way. No Excel is involved at all.

Installation requirements

The Define-XML Designer comes as a modern, GUI-based, Java software.
It requires Java 1.8 or higher being installed. If requested, the software can be delivered together with a Java installation. As such, the software can be run on either Windows or Linux systems. Testing on MacIntosh is currently in progress.
For a good display, a screen resolution of 1680x1050 or higher is recommended. As the software writes temporary files and log files, the included "temp" and "logs" files needs to be (made) writeable.
There is no special installation procedure: just copy the (unzipped) files from the distribution to the directory you want the software to run from.

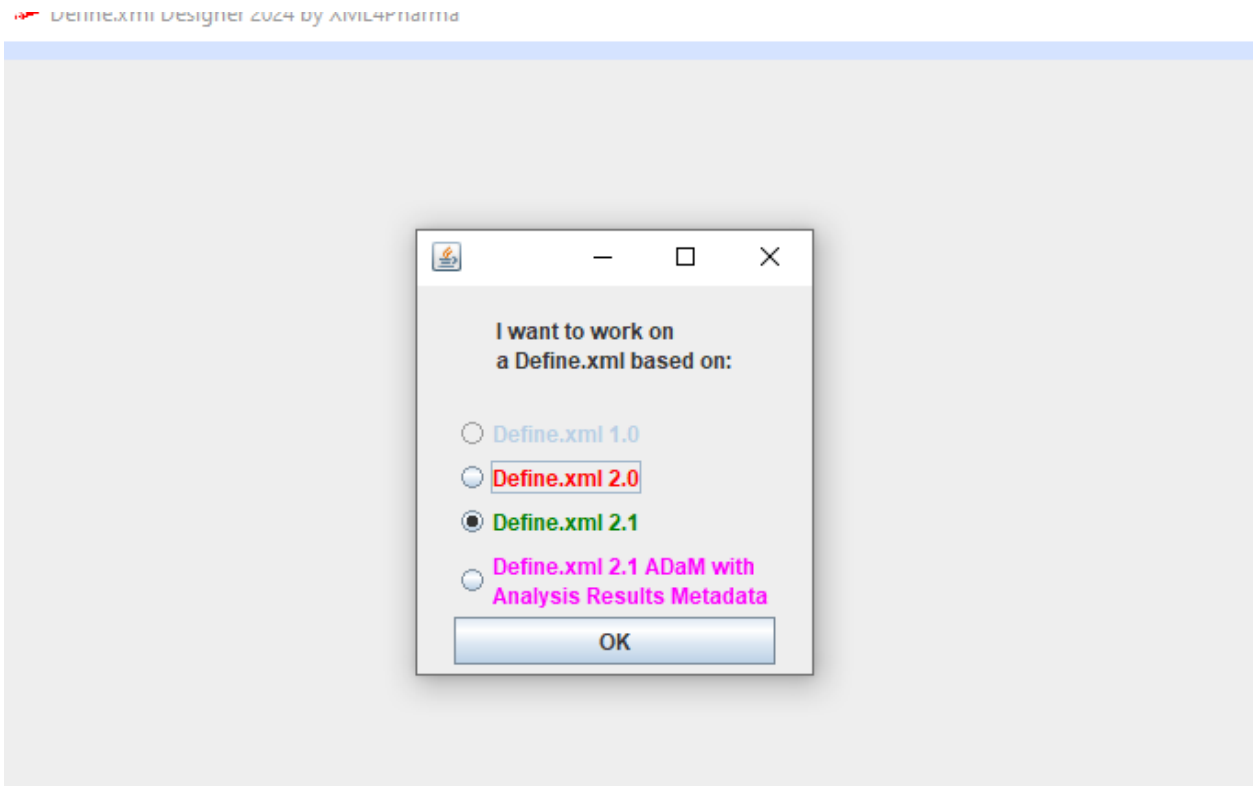
Starting the Define-XML Designer

On Windows , look for the file "SDTM-ETL.bat" in the main directory and double-click it. On Linux, use the file "SDTM-ETL.sh". In both cases, this results in:



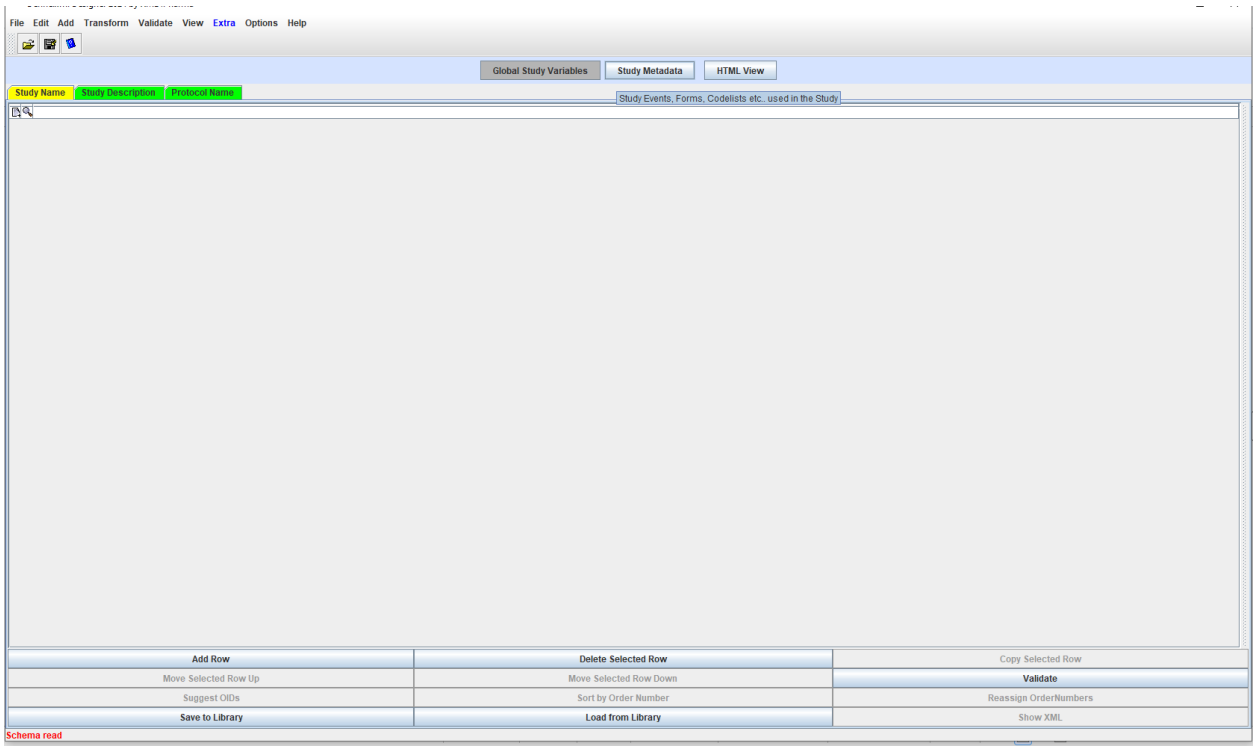
showing a license message, and with a separate "console": this is where the logging will be displayed, which is also written to a log file in the "logs" directory for traceability.
Clicking "OK" then leads to:

[1] Meant are especially "Pinnacle21 Community" and the extremely expensive "Pinnacle21 Enterprise" software packages.



asking the user whether he/she wants to work on either:
- a define.xml according to the old 2.0 standard (define.xml 1.0 is not supported anymore)
- a define.xml according to the modern 2.1 standard (recommended)
- a define.xml 2.1 implementing the "Analysis Results Metadata" (ARM) extension. This should only be used in the case of an ADaM requiring ARM.

In case "Define.xml 2.1" is selected, followed by "OK", the following window is displayed:



The upper part shows the menu bar, together with some "shortcut" image buttons for "Open File", "Save File" and "About" functionalities.

There are then 3 buttons "Global Study Variables"m "Study Metadata", and "HTML View". The latter will use the

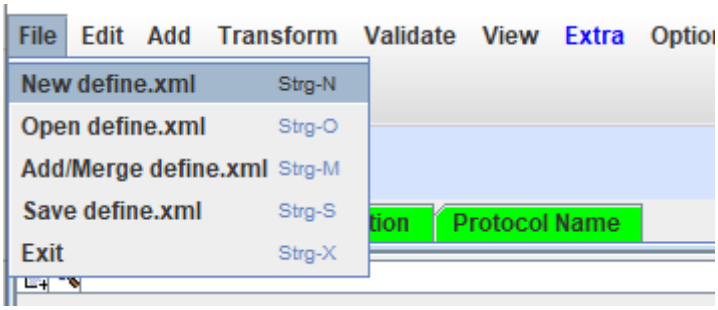
default or user's stylesheet to visualize the define.xml in a "human-friendly way" (e.g. a "browser view").

The center part then will display editable tables to add information to the define.xml. In many cases however, the users will want to use one of the many "wizards" to add or edit the information.

The bottom part contains a panel with a number of buttons for specific actions. Their functions will be explained later.

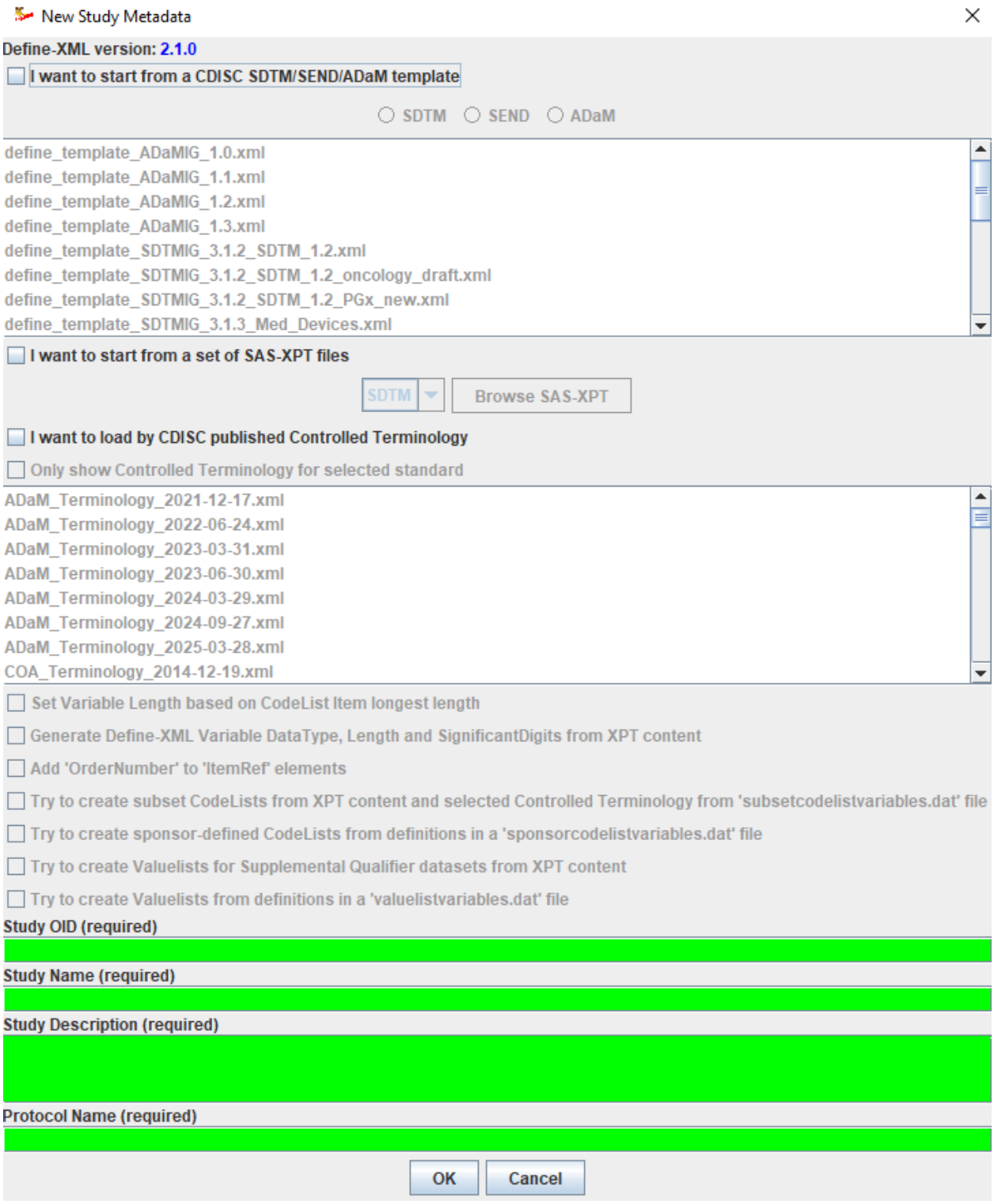
Starting a new define.xml

Use the menu "File - New define.xml" to start a new define.xml:



In case one wants to start from, or continue working on an existing define.xml, use the choice "Open define.xml". The "Add/Merge" define.xml can later be use to merge two define.xml files, for example as one person is working on the "Events" domains and another is working on the "Findings" domains.

When "New define.xml" is selected, this leads to the following dialog:



In the upper part, the checkbox "I want to start from a CDISC SDTM/SEND/ADaM template", it allows the user to start from a SDTM, SEND or ADaM template define.xml, to which one can then add the details. This is the preferred way to develop a define.xml even before the study starts, which can then be used e.g. as a "deliverables" or "requirements" define.xml for the submission. Essentially, this should be the preferred way of working.

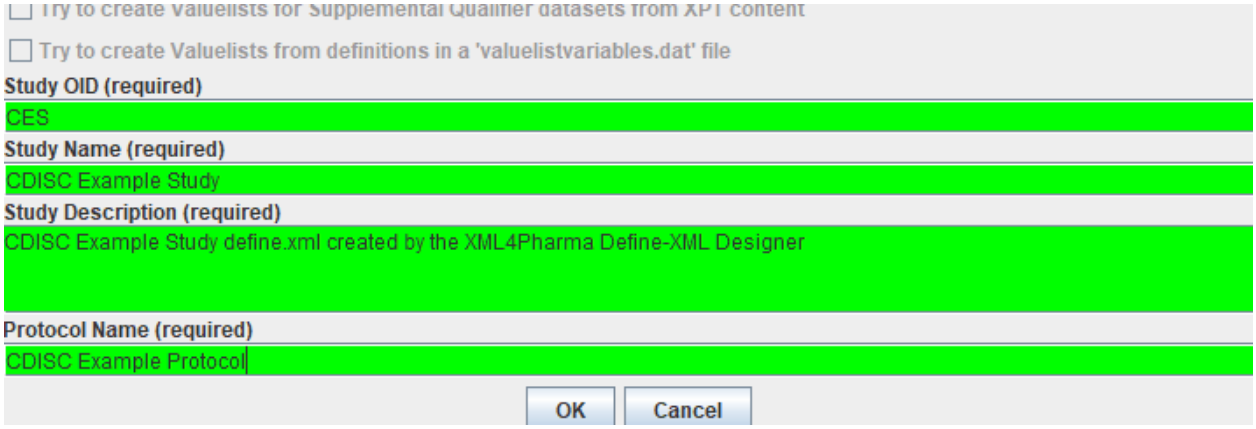
In still too many cases however, the define.xml only is generated after all the (SAS-XPT) datasets have been generated. This is of course viable for the case of "legacy" dataset submissions.
Also this use case is supported by the "Define-XML Designer".

In most cases, the user also wants to add CDISC Controlled Terminology (CDISC-CT), as this is required by the Define-XML standard. The software comes with all CDISC-CT published by CDISC in the last 10 years. When new

CT is published, it is made available from our website[2].

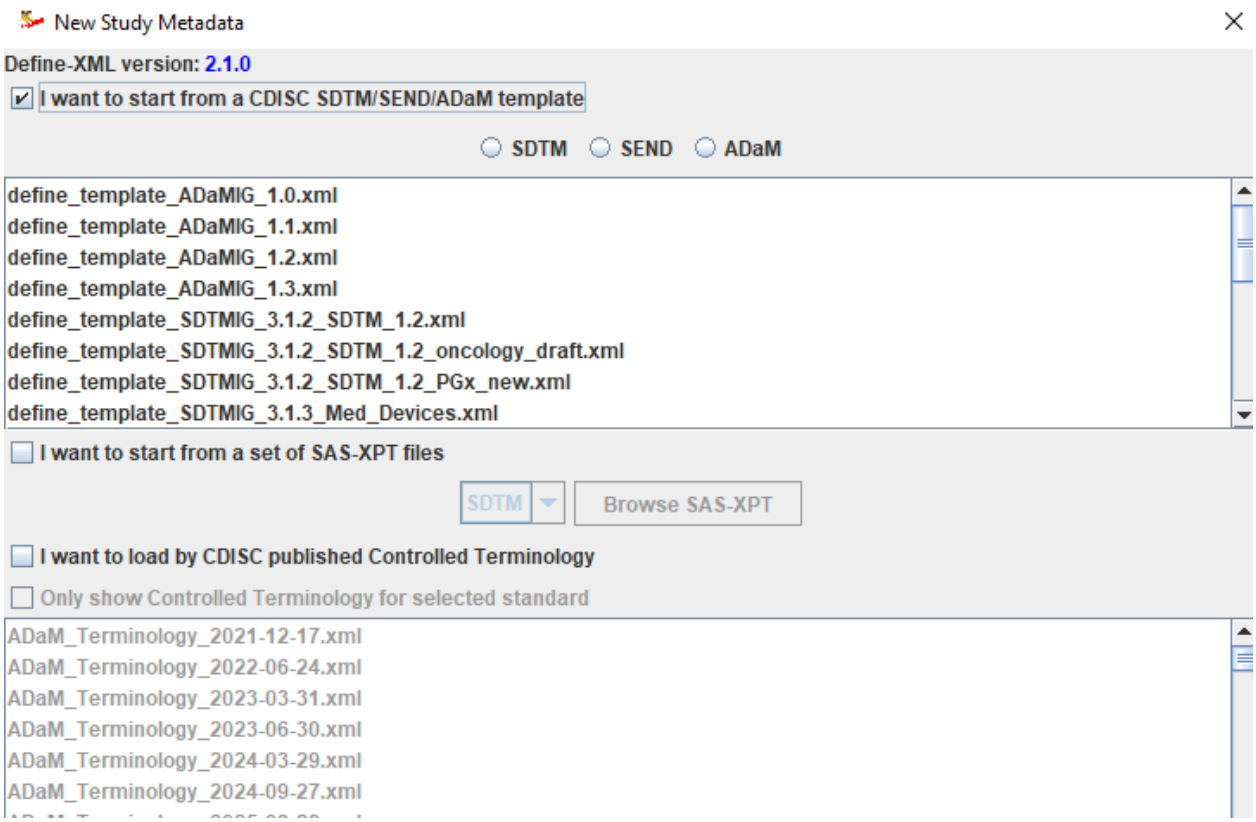
In the lower part, the information regarding the study-ID, the name of the study, a description of what the study is about, and the Protocol Name (which usually is the title of the protocol document) is to be added. The green color of the fields mean that this information is mandatory to be provided.
Remark that define.xml uses "OID" (Object Identifier) whereas SDTM/SEND/ADaM use "STUDYID". These are however the same thing. So, the first field needs to be filled with the value for the STUDYID.

So we e.g. fill the fields with:



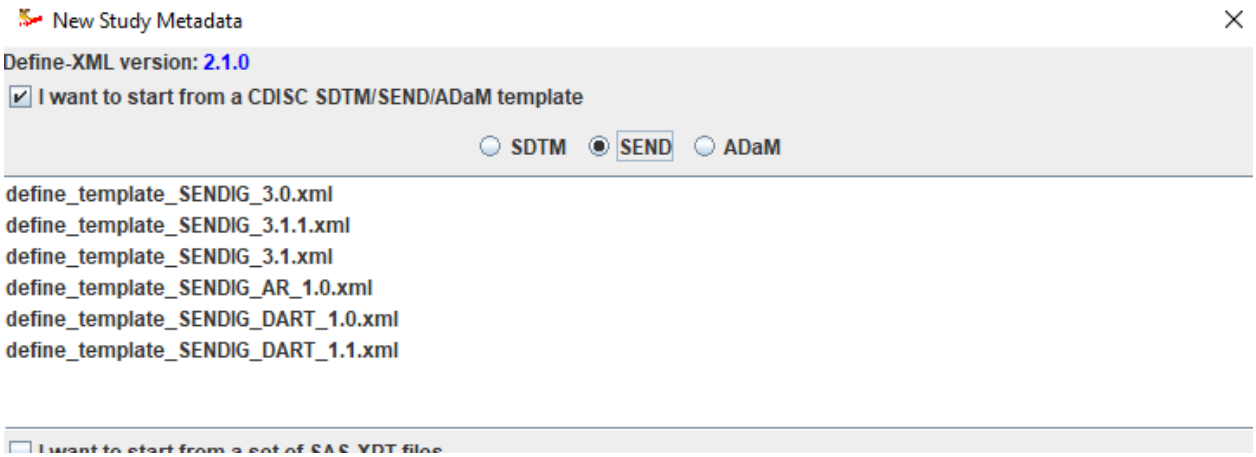
Starting from a define.xml template

The Define-XML Designer software comes with templates for all major CDISC submission standards. In this example, we will use SDTMIG v.3.4.
So we check the checkbox "I want to start from a CDISC SDTM/SEND/ADaM template", and then select "define_template_SDTMIG_3.4_SDTM_2.0":

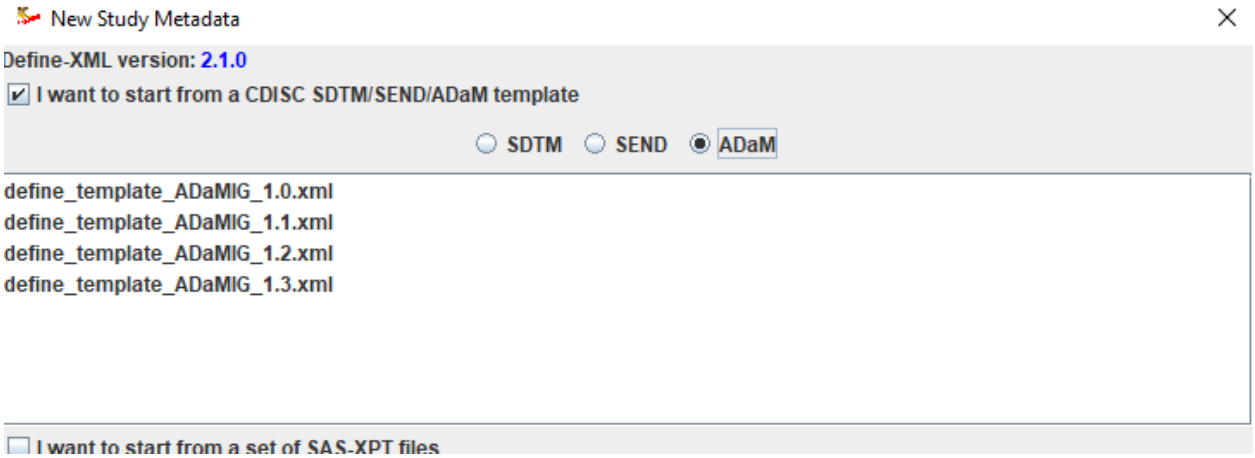


[2] free of charge of course ...

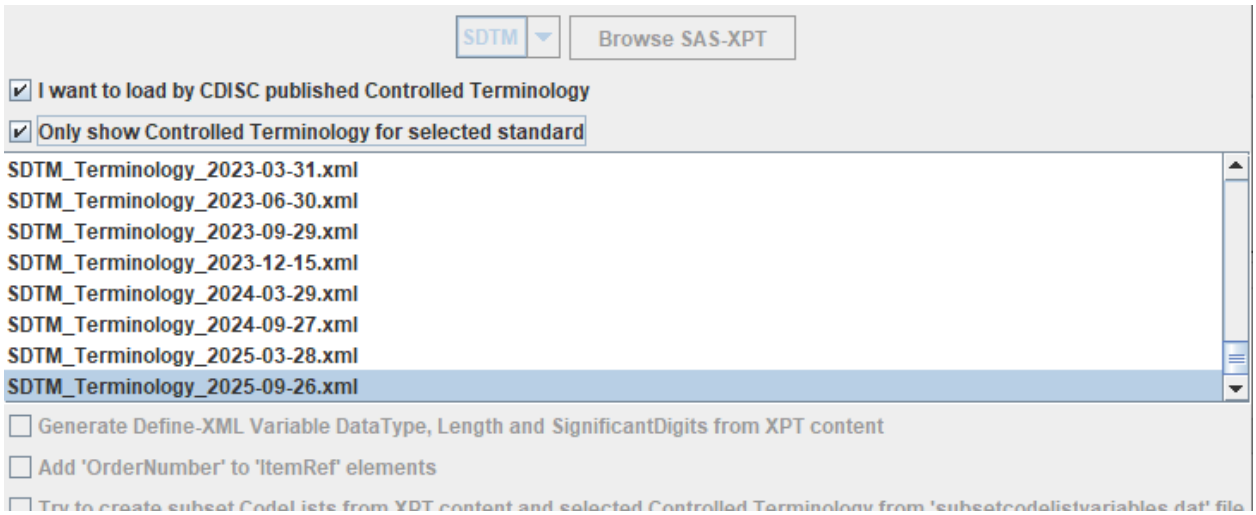
One can also limit the list to the templates for a specific standard by clicking one of the "SDTM", "SEND" or "ADaM" radiobuttons. For example for SEND:



or for ADaM:

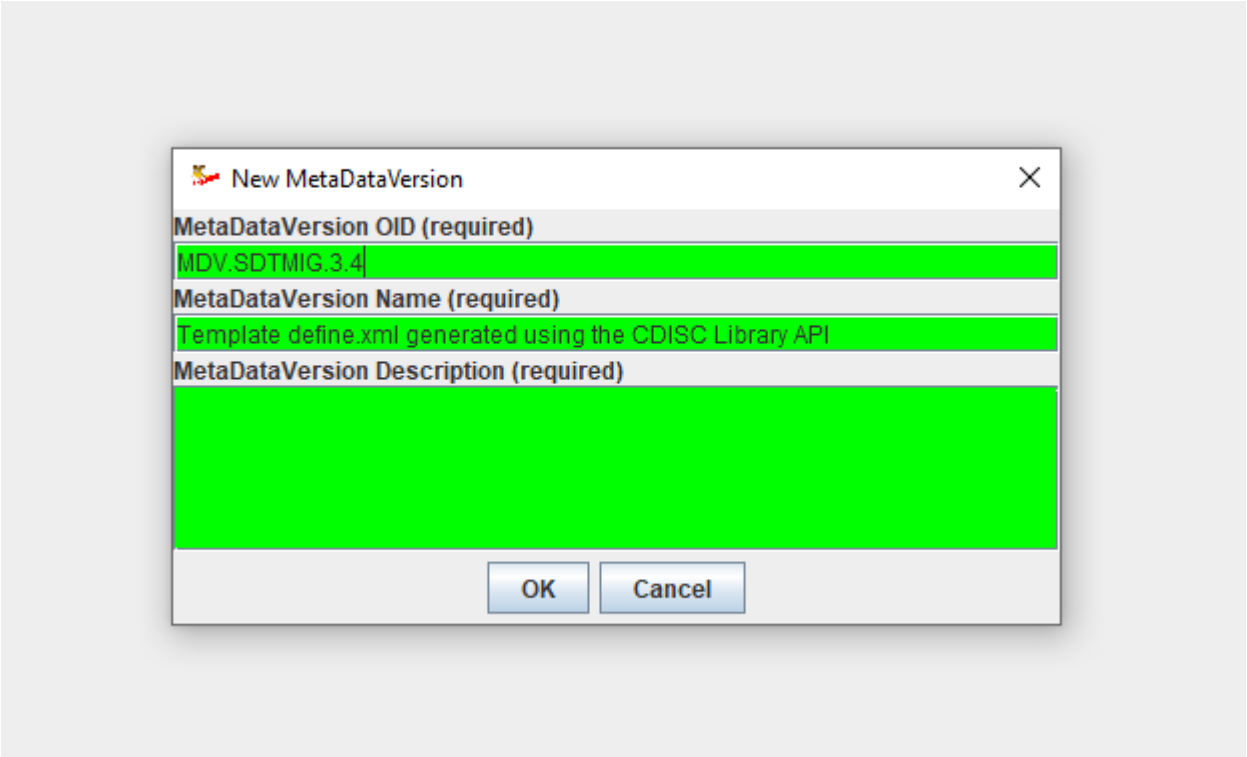


One can then also select a version of the SDTM-CT by first checking "I want to load by CDISC Controlled Terminology" and then "Only show Controlled Terminology for selected standard":

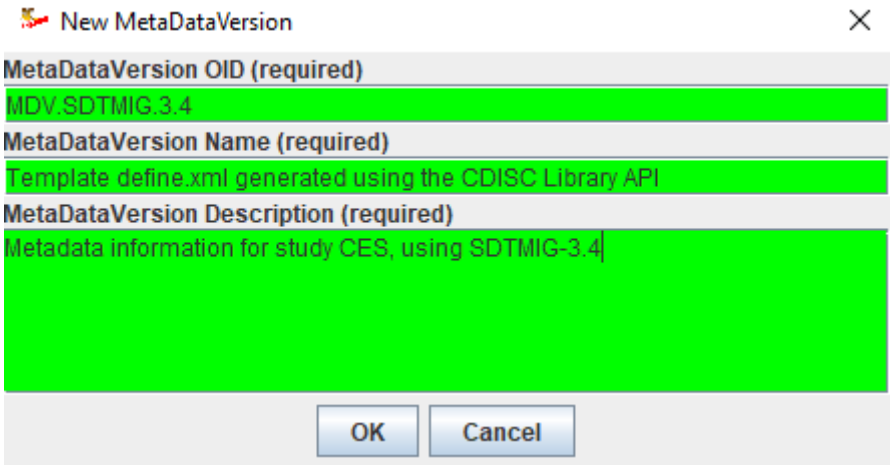


Remark that for ADaM, this also allows to load SDTM-CT as is often needed.

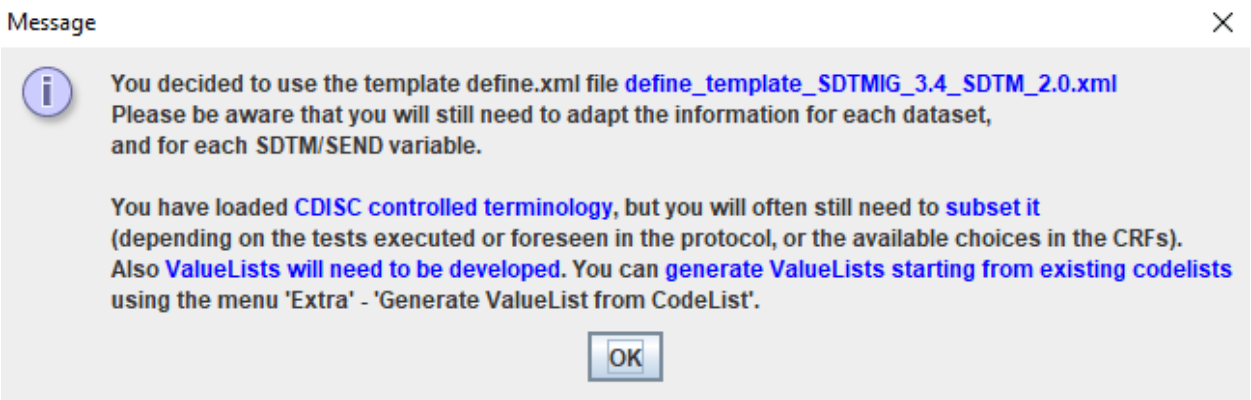
Clicking "OK" at the bottom of the dialog then starts loading the requested template and the selected CDISC-CT, leading to another shorter dialog:



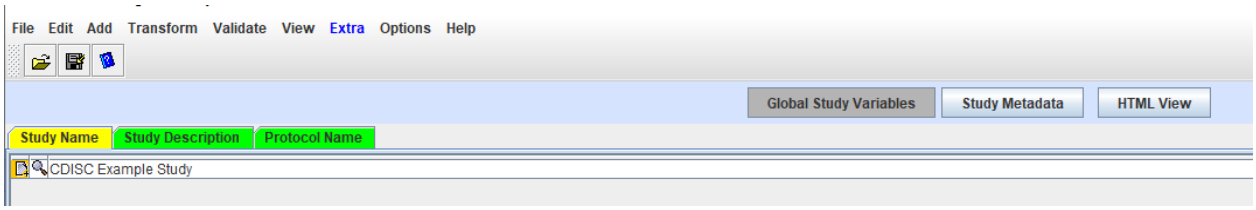
It also states that originally the template was generated from the CDISC Library using the API. This is a good thing, as the CDISC Library is "the CDISC truth".
The only thing we then still need to do is to add some text for "MetaDataVersion Description".
This information will later also be displayed in the browser through the CDISC stylesheet. So we e.g. add:



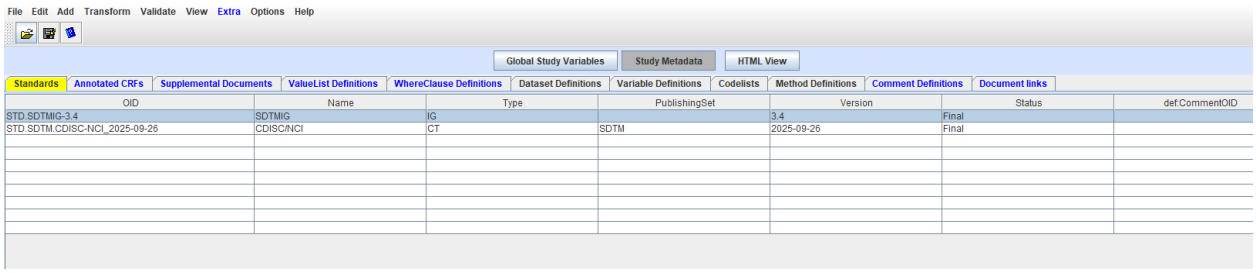
and then click "OK", leading to an information message:



Especially important here is that all the CDISC-CT has been loaded, and the user will probably need to subset some of it. For example, the CDISC-CT for "VSTESTCD" currently has 75 terms, from ABI (Ankle-Brachial Index) to WTHTPCTL (Weight-for-Height Percentile), but it is expected that only those entries are submitted for the tests that are actually planned. Later we will learn how to subset codelists.
Often, one will also want to add ValueLists, e.g. in SDTM defining the units (--ORRESU/--STRESU) as function of the test, such as "mmHg" for blood pressure tests.
Also this will be explained in detail further on, e.g. how to generate a ValueList starting from a CodeList.
After clicking "OK", the tables with metadata begin to fill:

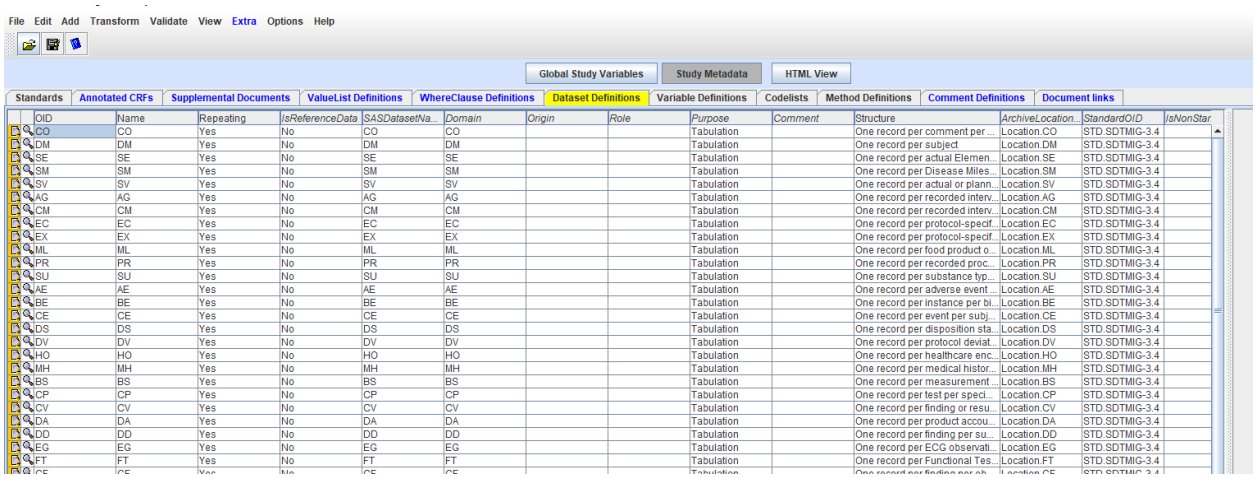


This becomes more obvious when one clicks the "Study Metadata" button:



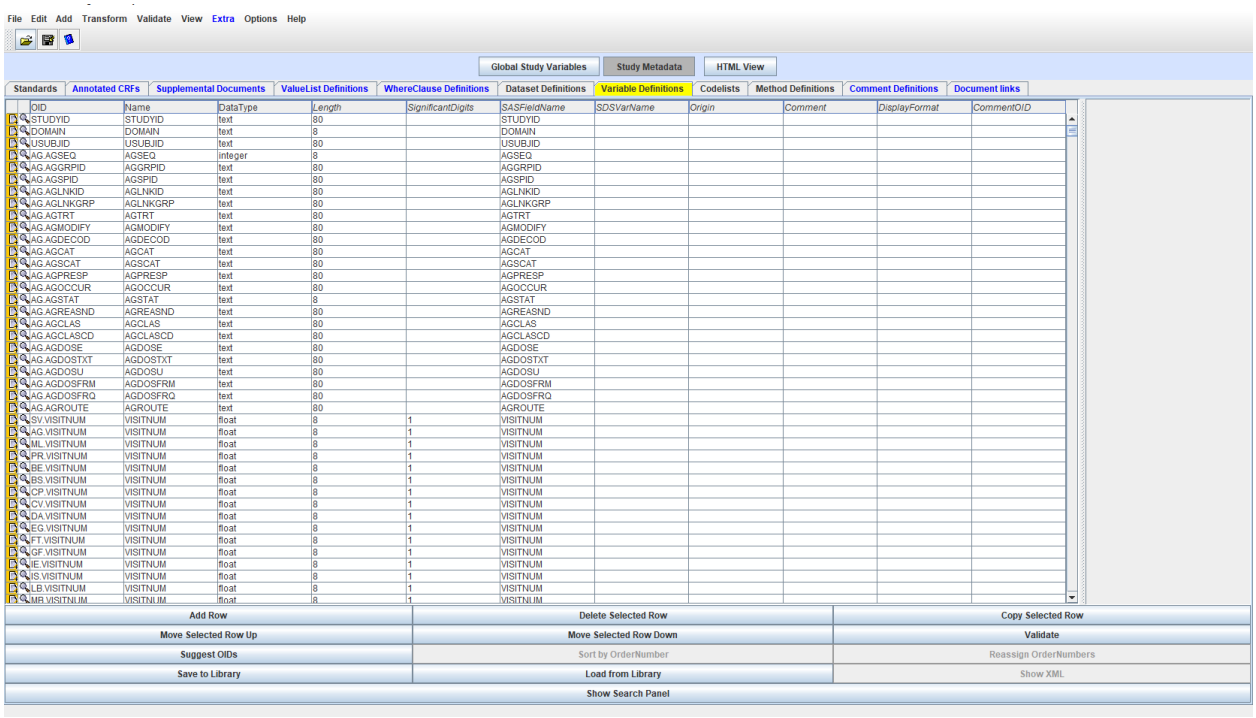
The most-left tab already filled in the information which standards and versions are used.

When e.g. clicking the "Dataset Definitions" tab, one finds all SDTM domains from SDTMIG-3.4 listed:



We can now start editing the table, e.g. remove domains for which we do not plan to collect data and/or submit information. Deleting a row in the table can simply be done by selecting a cell in that row and click the "Delete row" button. If one deleted a row accidentally, no panic, as one can always revert to an earlier version - see the section "Autosave and logging"

For the tab "Variable Definitions", one will find:



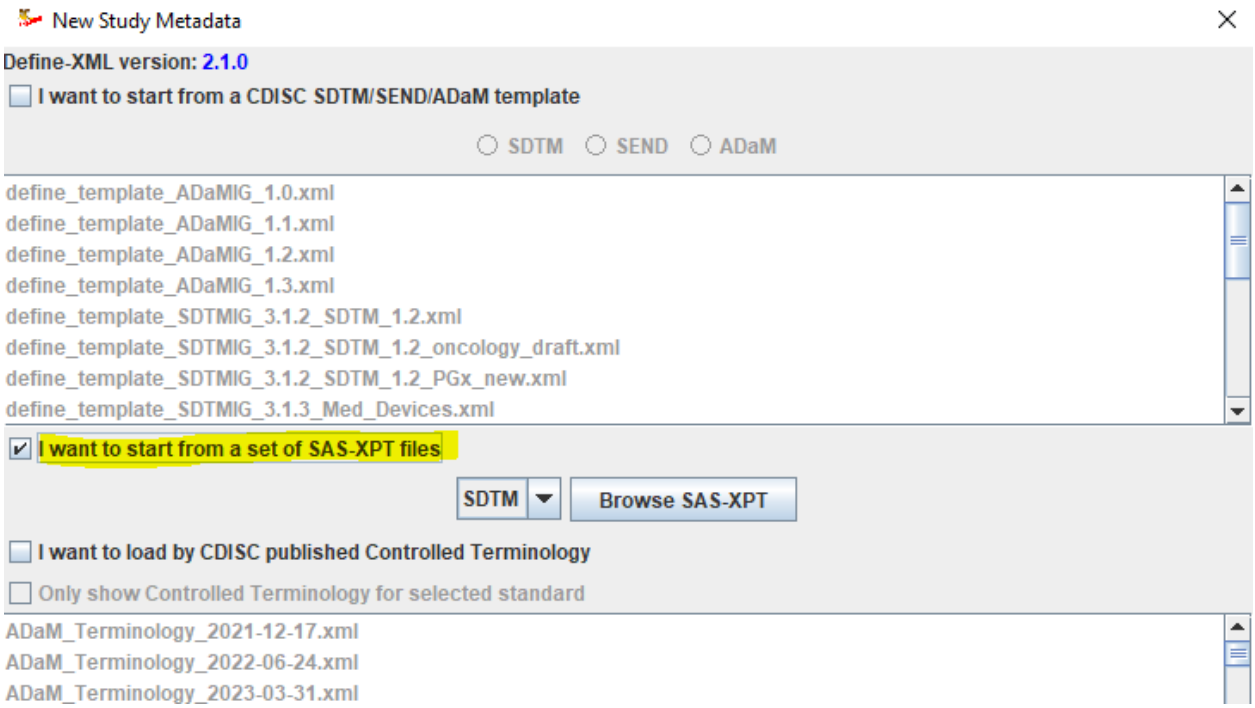
Before we will dig into how to further work with these tables, it will first be explained how to generate a "prototype" define.xml starting from a set of (SAS-XPT) submission files.

Starting from a set of submission files

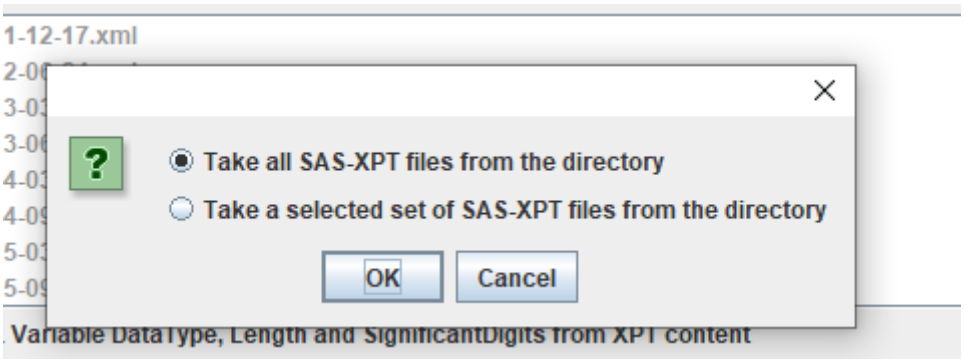
Starting from a set of existing submission files (currently only SAS-XPT format is allowed by regulatory authorities[3]) can be a viable solution in the case of xxxx data sets. It is considered bad practice in the case of recently generated SDTM, SEND of ADaM datasets, as the define.xml is **the** specification of the deliverables for the submission. So, essentially, the define.xml should be developed even before the study starts, in the case of SDTM from the protocol and the CRFs, and from the protocol and Statistical Analysis Plan (SAP) in the case of ADaM.
We do however recognize that this bad practice still exists in our industry, and decided to still provide support for this use case.

After starting up the Define-XML Designer, and using the menu "File - New define.xml", we now select "I want to start from a set of SAS-XPT files":

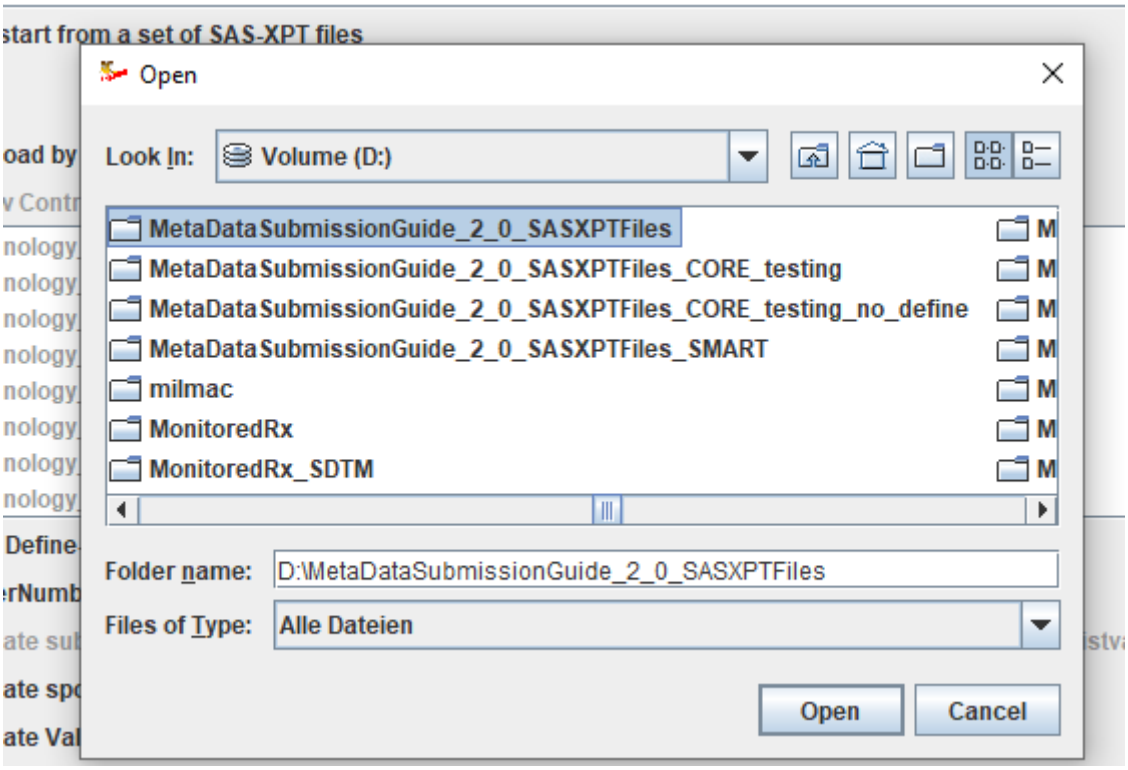
[3] This is expected to change soon, at least for the FDA, now that CDISC Dataset-JSON has been developed and is currently being piloted at the FDA. This modern JSON-based format has many advantages over outdated XPT.



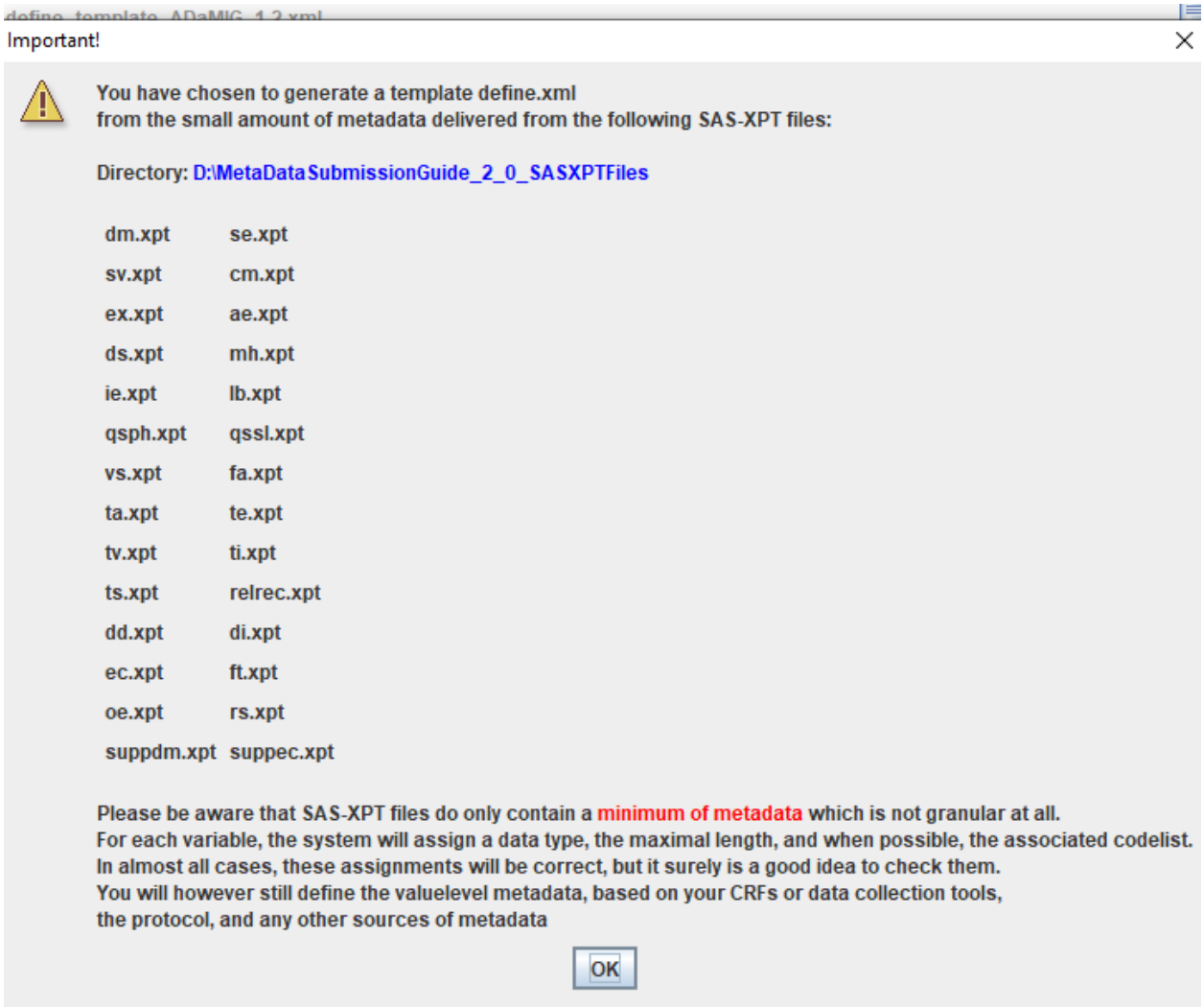
then select "SDTM", "SEND" or "ADaM" from the dropdown. Just for the example here, we will use SDTM.
We then select a set of XPT files using the button "Browse SAS-XPT", leading to:



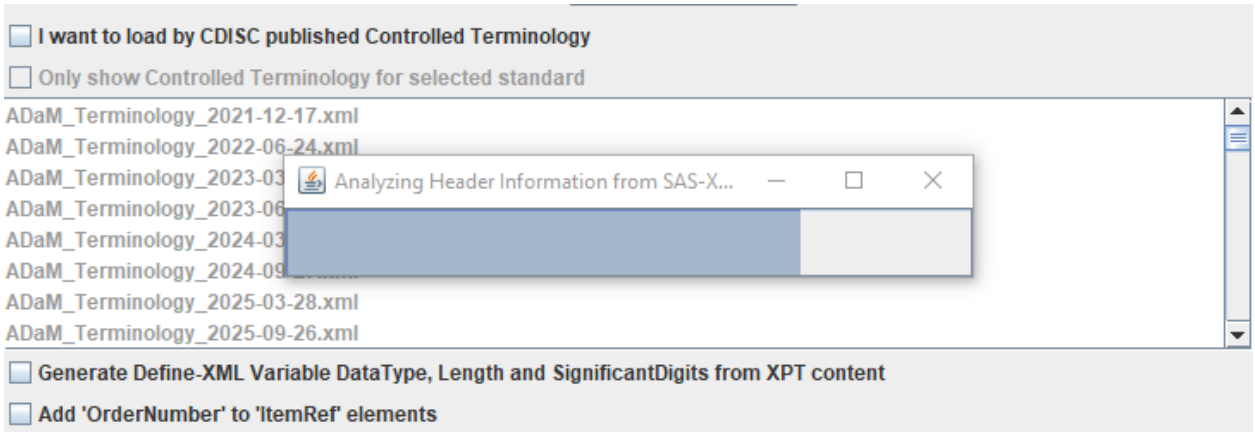
We will use all the XPT files from a directory - this is the most usual case. After clicking "OK", a filechooser appears, allowing us to select the directory where the XPT files are located:



and after clicking "Open", leading to:

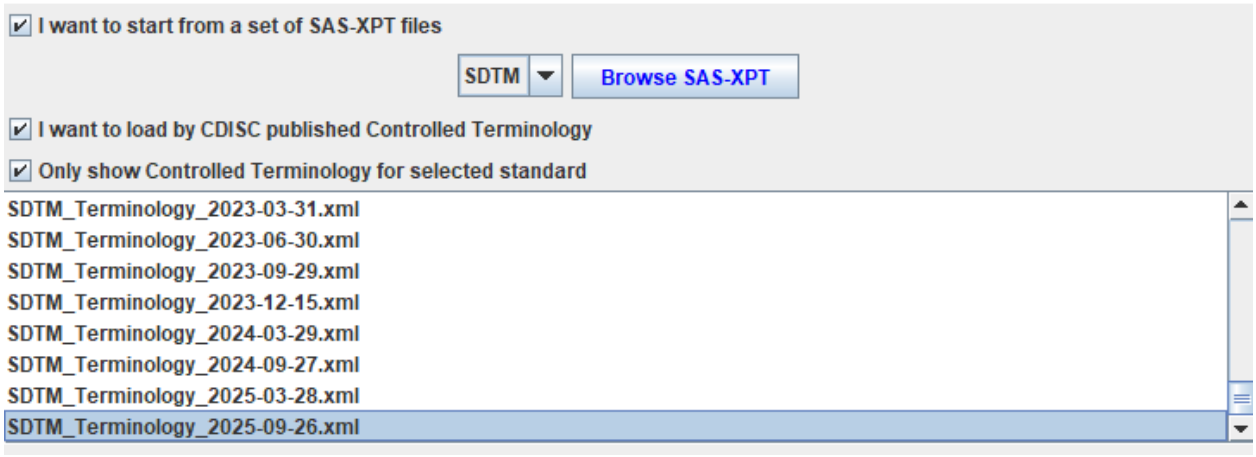


providing a list of the XPT datasets in the directory, and providing an explanation that SAS-XPT only contains a small amount of metadata, so that it is not a bad idea to check these, and that one will still need to add additional information like ValueLists, the data source and origin, etc..
After clicking "OK", the system starts analyzing the XPT data and extracting information:

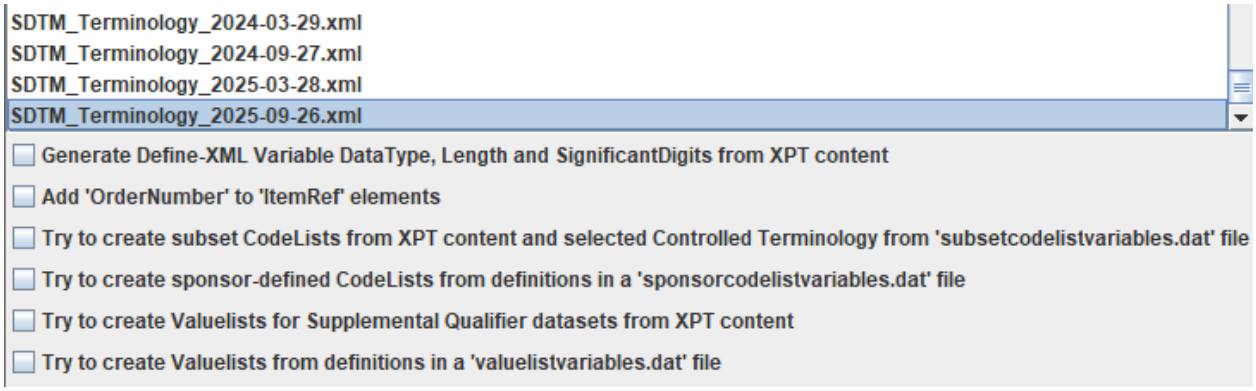


After this, it is always a good idea to select a version of the CDISC Controlled Terminology (CDISC-CT), as this will allow us to add new terms where needed, to align with what has been planned (XPT only delivers what has been done). For example, if there never was a "Severe" adverse event, the value "SEVERE" will not appear in the AE-XPT dataset, and the generated codelist from the XPT will not contain "SEVERE", although "Severe" was an option on the CRF.
The checkbox "Only show Controlled Terminology for selected standard" can be of help to limit the possible choices. It is not automatically checked, as for ADaM, one will often also want to load SDTM CDISC-CT.
Additional CDISC-CT can however also later be added.

So we e.g. select:



Underneath, there are a number of checkboxes:



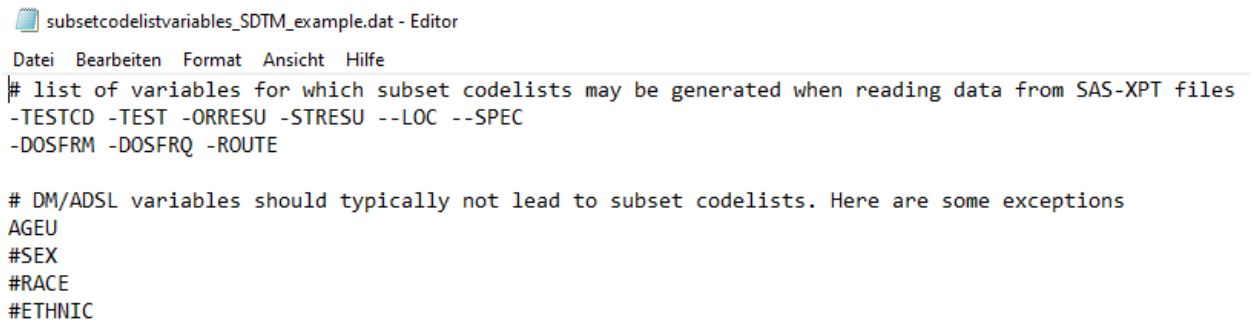
The first one allows the system to automatically generate variable data type, maximal length and, in the case of the

"float" datatype, define the "significant digits"[4] from the XPT content.
One will usually want to have this done in an automated way (the alternative is to add that information manually), so this checkbox is usually checked.

The second checkbox "Add 'OrderNumber' to 'ItemRef' elements" allows to automatically add the "OrderNumber" attribute and add a value for it automatically. Essentially, there is no need for this when the order in the file is also the "display order", but many companies want to have "OrderNumber" included[5].

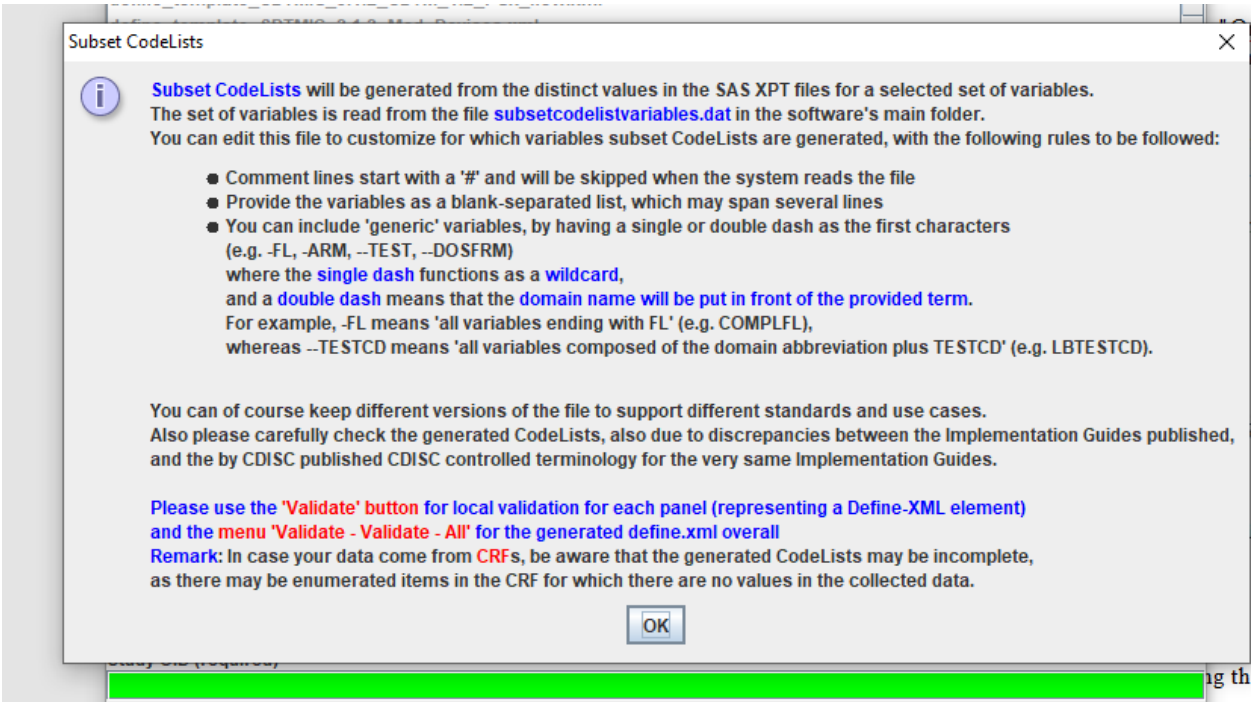
The third checkbox allows the user to have "subset codelists" generated from the information in an external file. This may be useful to have subset codelists generated automatically from the XPT content for specific variables. Reason is that we e.g. do not want to have the complete list of allowed LBTESTCD values (about 2,500 items), but only these that were actually in the lb.xpt file. Remark that we still then may need to extend this subset codelist when some tests were planned, but never done, as these will not be appear in the XPT file.

A typical example of such a file with variables for which subset codelists need to be generated is:



Remark that lines starting with a "#" are "commented out". This allows a flexible way of managing the list with variables for which "subset-codelists" need to be automatically created.
One can generate different instances of such a dataset with variables (e.g. different ones for ADaM), but it is always the "subsetcodelistvariables.dat" file that will actually be used.

When the checkbox "Try to create subset Codelists from XPT content …" is checked, the following dialog is displayed:



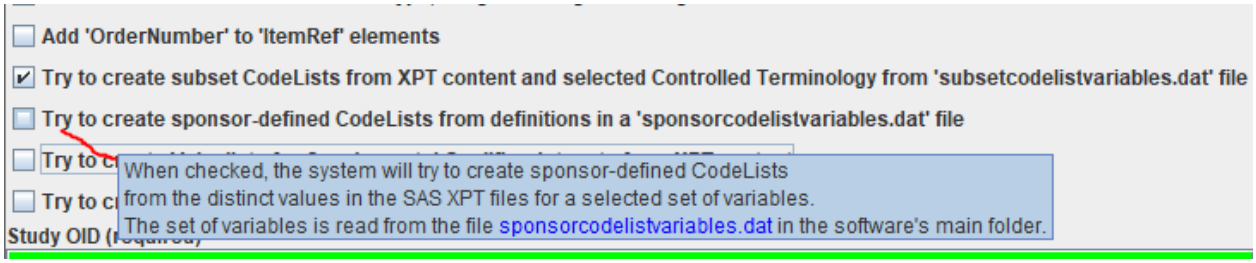
explaining how this works.

---

[4] "Significant Digits" is somewhat misleading here, as it is the number of characters after the decimal point.
[5] This is also related to some false positive messages generated by the P21 validation software in the past.

Remark that we discourage to allow subsetting codelists for variables like "AESEV" and "AESER", as this can lead to a define.xml that suggests that e.g. AESEV=SEVERE was not an possibility on the CRF. The define.xml should reflect what was planned, not what was finally obtained.

Similar is the checkbox "Try to create sponsor-defined codelists from definitions in a "sponsorcodelistvariables.dat" file:



An example of such a "sponsorcodelistvariables.dat" (in this case for SDTM) file is:

sponsorcodelistvariables_SDTM_example.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
```

where we define that "sponsor-defined" codelists need to be created for all --CAT variables, as well as for ARMCD and ARM in DM (Demographics), ETCD and ELEMENT in SE (Subject Elements), and EXTRT in EX (Exposure).

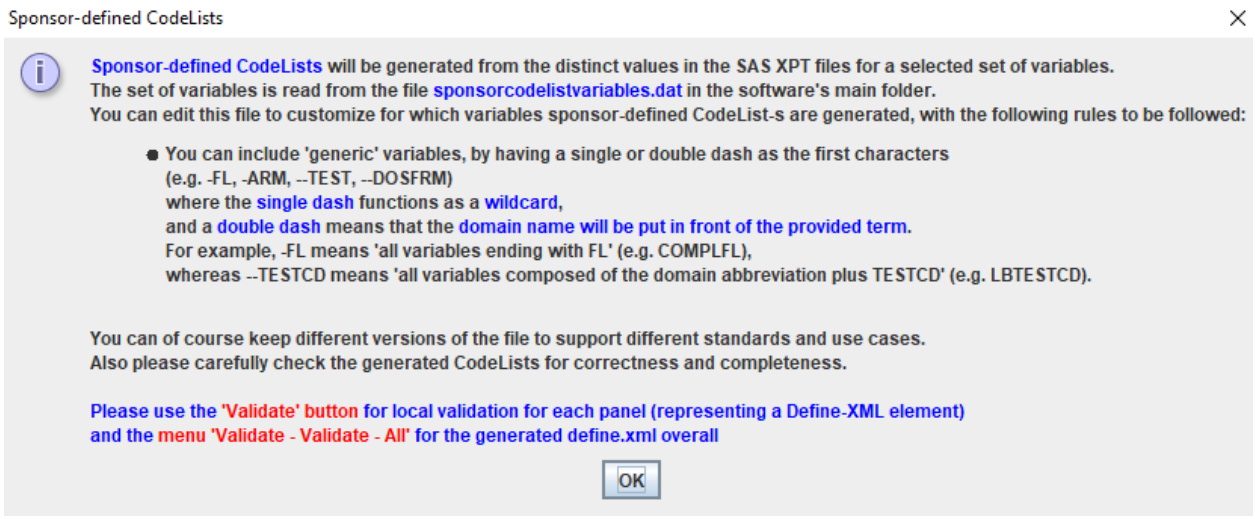
A typical example for ADaM can be:

sponsorcodelistvariables_ADaM_example.dat - Editor

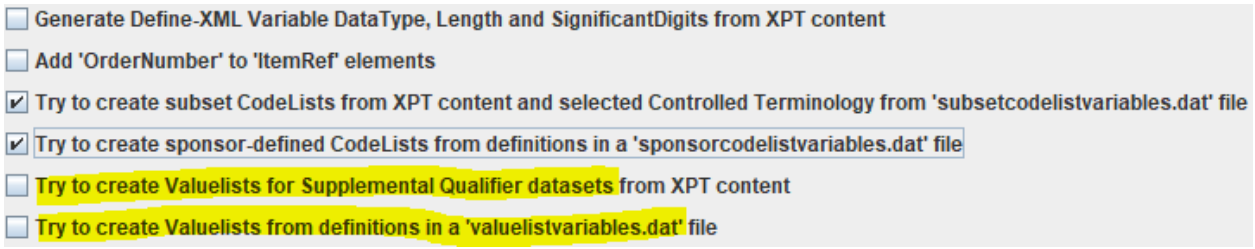
Datei Bearbeiten Format Ansicht Hilfe

```
# variables for which to automatically generate a codelist
PARAMCD
PARAM
AGEU
RACE
# ETHNIC
SEX
SITEID
# "generic" variables, single dash means "end-with"
# double dash means "replace by dataset name"
# using -FL will generate separate codelists for each individual "flag" variable
# which can be very time consuming
# The alternative is to add the "NY" codelist manually in the Define-XML Designer by drag-and-drop
# -FL
```

The dialog that appears when the checkbox "Try to create sponsor-defined codelists from definitions in a "sponsorcodelistvariables.dat" explains this very well:



The next two checkboxes are:



For the Supplemental Qualifiers, the Define-XML specification version 2.1 states:

5.3.9 def:ValueListDef Element

The table below specifies the XML structure for valuelist metadata.

Element Name	def:ValueListDef
Element XPath(s)	/ODM/Study/MetaDataVersion/def:ValueListDef
Element Textual Value	None
Usage	• Requirement: Conditional • Cardinality: Required for each unique value of the ValueListOID attribute within the MetaDataVersion • Business Rule: For SDTM SUPPQUAL datasets, a def:ValueListDef element must be provided to describe the QVAL variable. • Other Information: Contains ItemRef elements that reference ItemDef elements that provide the value-level metadata details
Attributes	OID
Child Elements	Description, ItemRef

An example is also provided:

4.5.2.3 Value-level Metadata for a SUPPQUAL Domain

This example illustrates Value-level definitions for variable QVAL in the SUPPLB and SUPPQS datasets as supplemental or non-standard variables for the Laboratory Test Results and Questionnaires domains.

Note: The definition of the valuelist is for the variable QVAL and not for the variable QNAM.

Example 4.5.2.3.1 Value-level Metadata: SUPPQUAL

```
<def:ValueListDef OID="VL.SUPPLB.QVAL">
    <ItemRef ItemOID="IT.SUPPLB.QVAL.LBCLSIG" OrderNumber="1" Mandatory="No"
MethodOID="MT.CLSIG" Role="Record Qualifier">
        <def:WhereClauseRef WhereClauseOID="WC.SUPPLB.QNAM.LBCLSIG"/>
    </ItemRef>
</def:ValueListDef>
<def:ValueListDef OID="VL.SUPPQS.QVAL">
    <ItemRef ItemOID="IT.SUPPQS.QVAL.RTRINIT" OrderNumber="1" Mandatory="No"
Role="Identifier">
        <def:WhereClauseRef WhereClauseOID="WC.SUPPQS.QNAM.RTRINIT"/>
    </ItemRef>
</def:ValueListDef>

<ItemDef OID="IT.SUPPLB.QVAL.LBCLSIG" Name="LBCLSIG" DataType="text" Length="1"
SASFieldName="LBCLSIG" Role="Record Qualifier">
    <Description>
        <TranslatedText xml:lang="en">Clinically Significant</TranslatedText>
    </Description>
    <CodeListRef CodeListOID="CL.NY"/>
    <def:Origin Type="Derived"/>
</ItemDef>

<ItemDef OID="IT.SUPPQS.QVAL.RTRINIT" Name="RTRINIT" DataType="text" Length="3"
SASFieldName="RTRINIT" Role="Result Qualifier"
def:CommentOID="COM.SUPPQS.QVAL.RTRINIT">
    <Description>
        <TranslatedText xml:lang="en">Rater Initials</TranslatedText>
    </Description>
    <def:Origin Type="CRF">
        <def:DocumentRef leafID="LF.blankcrf">
```

where ValueLists are defined for the "Non-Standard Variable" (NSV - or "Supplemental Qualifier") "LBCLSIG" (Clinical Significant) in LB, and "RTRINIT" (Rater Initials) in QS.

When the checkbox "Try to create for Supplemental Qualifier datasets from XPT content" is checked, the system will try to generate these ValueLists automatcally.
Please be aware that such ValueLists are important to allow reviewers to "bring back" the NSVs to the parent domain in their review systems.

One can also have ValueList being created automatically for some of the variables by checking the checkbox "Try to create ValueLists from definitions in a "valuelistvariables.dat" file. An example of the contents of such a file (for SDTM) is:

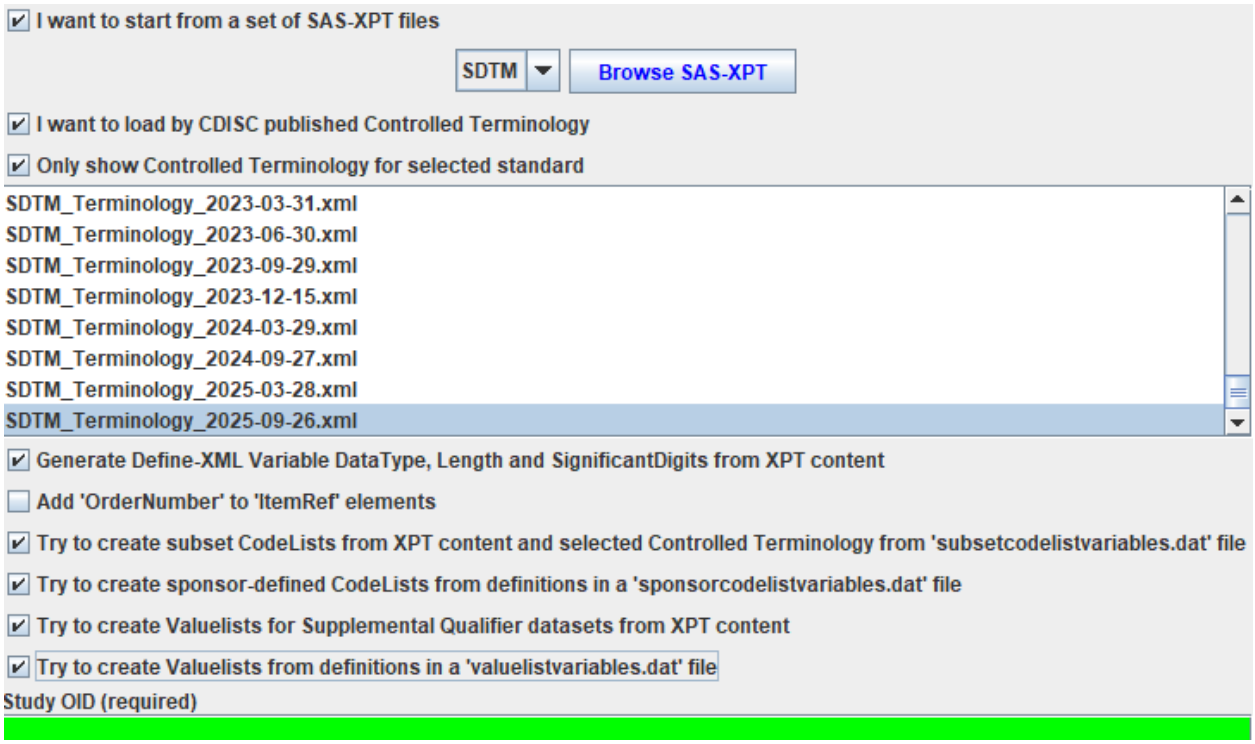
valuelistvariables.dat - Editor

Datei Bearbeiten Format Ansicht Hilfe

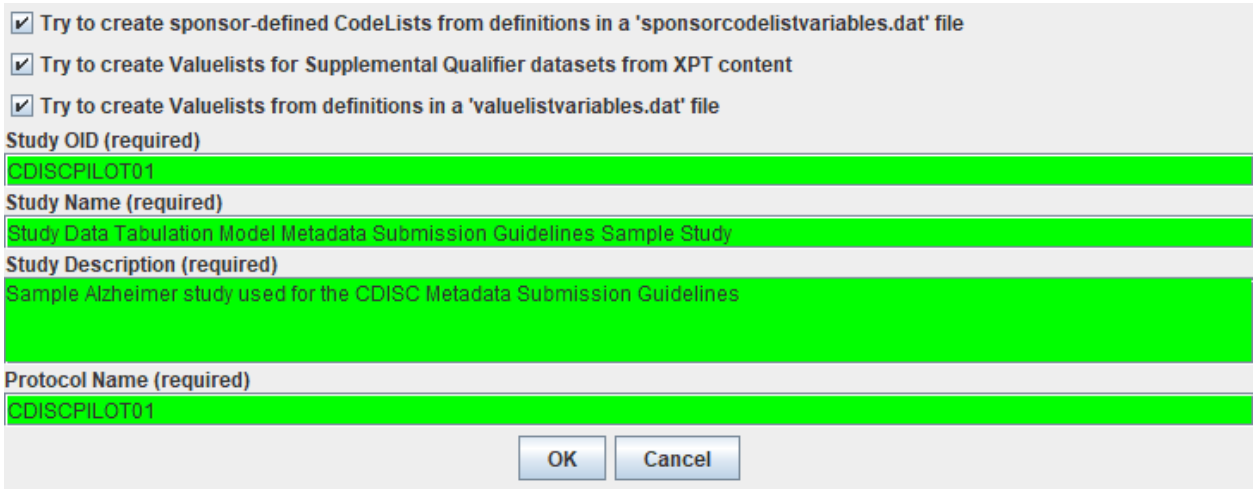
```
VSORRESU WHERE VSTESTCD EQ WEIGHT
VSORRESU WHERE VSTESTCD IN SYSBP,DIABP
#VSPOS WHERE VSTESTCD NE SYSBP,DIABP
VSPOS WHERE VSTESTCD IN SYSBP,DIABP
VSORRES WHERE VSTESTCD EQ FRMSIZE
#VSORRES WHERE VSTESTCD NE FRMSIZE
# VSORRES WHERE VSTESTCD NOTIN SYSBP,DIABP,WEIGHT,HEIGHT,PULSE
# VSORRESU WHERE VSTESTCD NE HEIGHT
#LBORRES WHERE LBCAT NOTIN CHEMISTRY,HEMATOLOGY
LBSPEC WHERE LBCAT EQ HEMATOLOGY
```

also here, lines starting with a "#" are "commented out.
For example, the last entry "LBSPEC WHERE LBCAT EQ HEMATOLOGY" will try to generate a ValueList on LBSPEC, with the selection criterion is "where LBCAT = HEMATOLOGY". We will later look at the result of this.

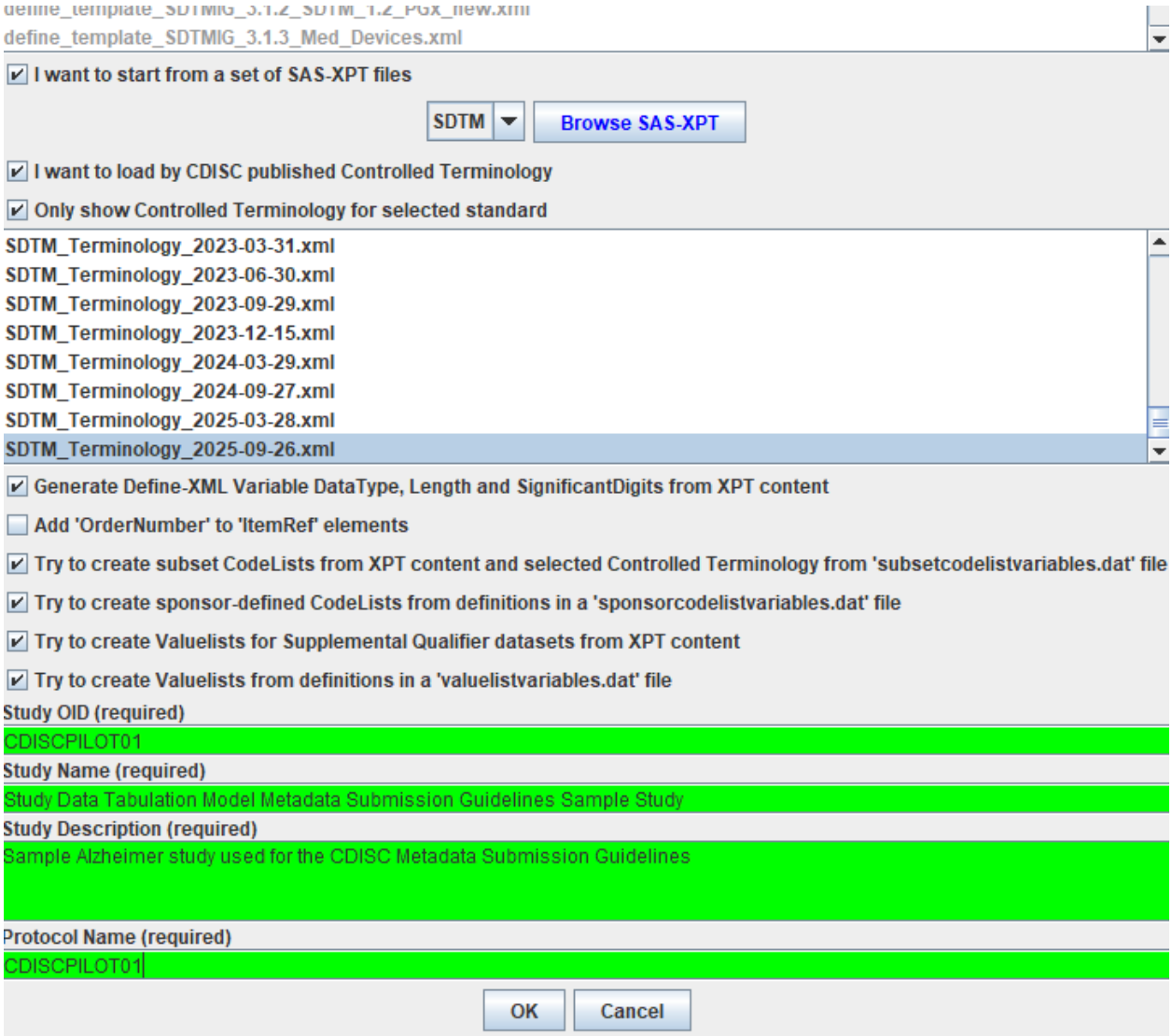
Suppose we use the following choices:



We then still need to provide some information, like the identifier (OID) of the study (this will usually be equal to the "STUDYID" in the XPT files), a "Study Name", "Study Description" (usually from the protocol) and "Protocol Name" (usually this is the title of the protocol document). For example:

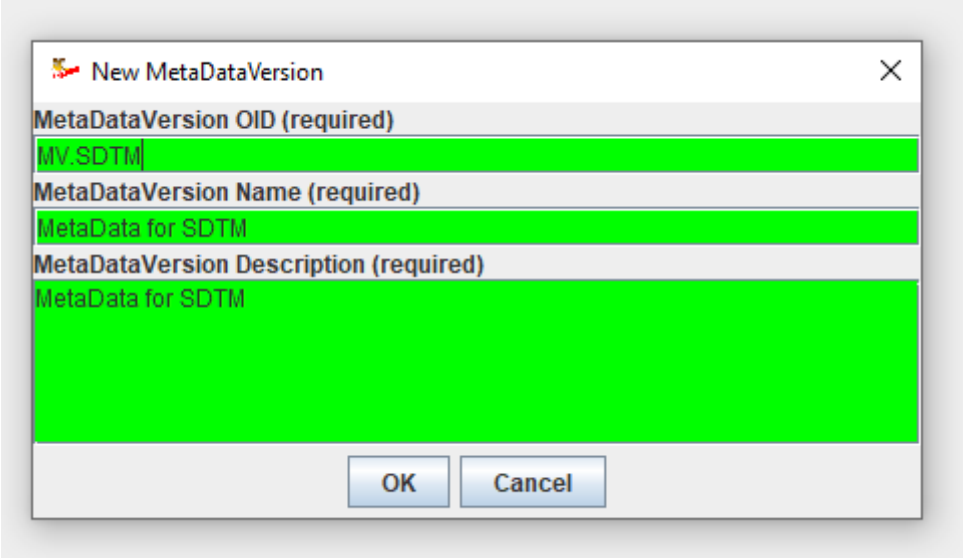


It surely is not a bad idea to check everything now before proceeding.
We have:



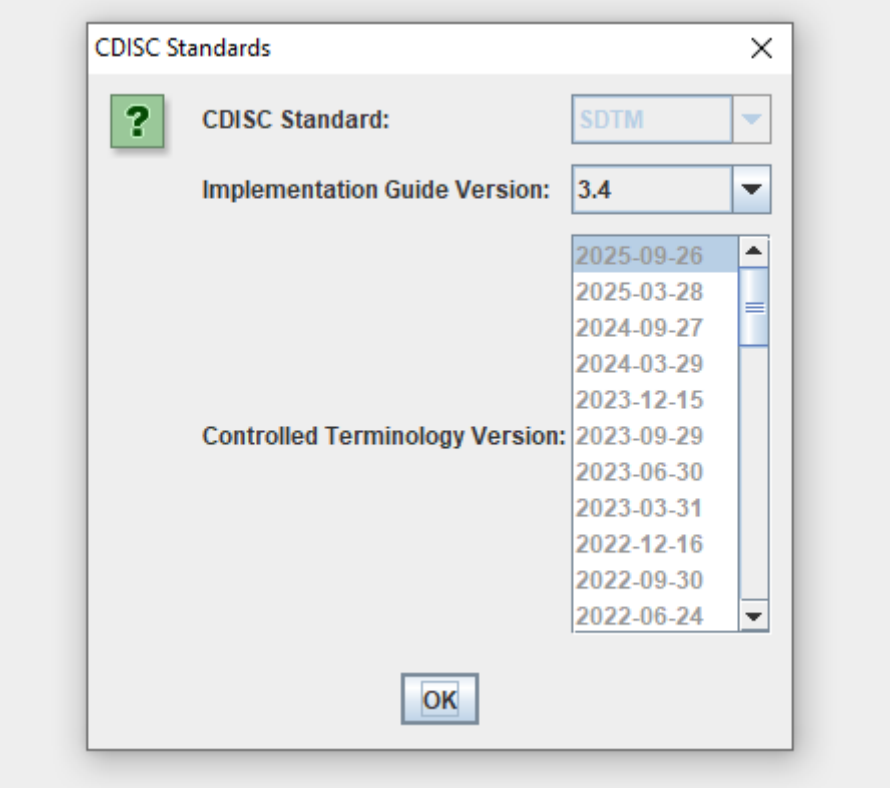
Remark that most of this information will later appear in the header of the HTML (browser) define.html, which is the visualization of the define.xml.

When then clicking "OK", the system starts generating a "prototype" define.xml, which we will then further refine. It first proposes some OIDs (identifiers) to be used:

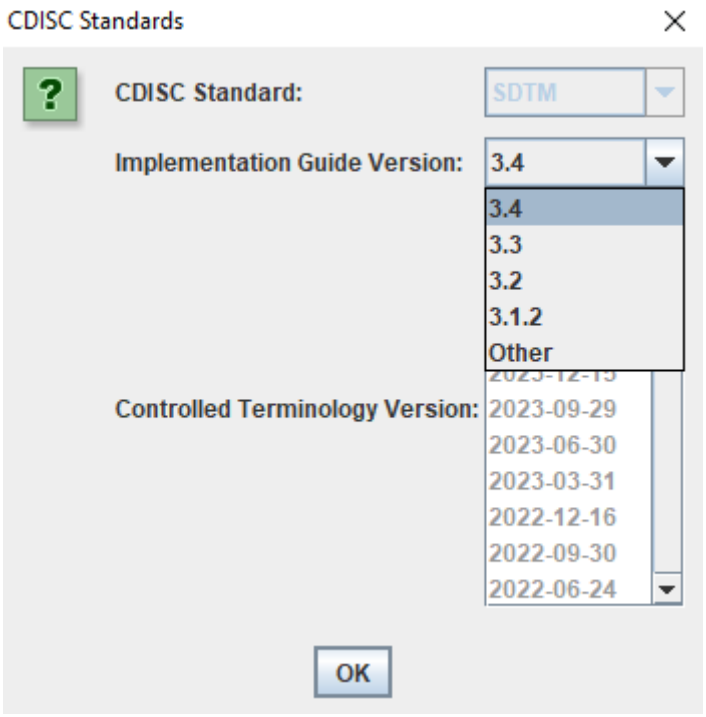


These will however later not appear in the "View" on the define.xml - they are for internal usage only.

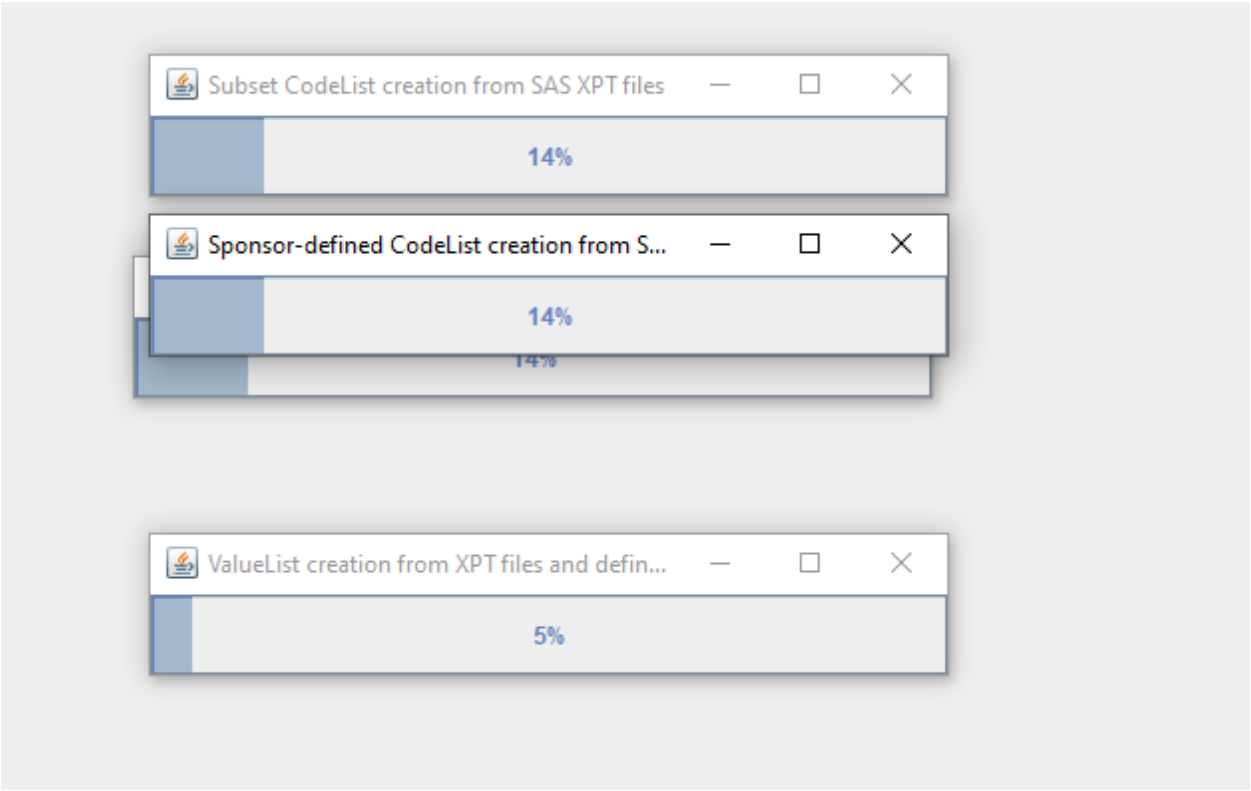
Clicking "OK" then leads to another dialog, allowing to select which standard version will be declared:



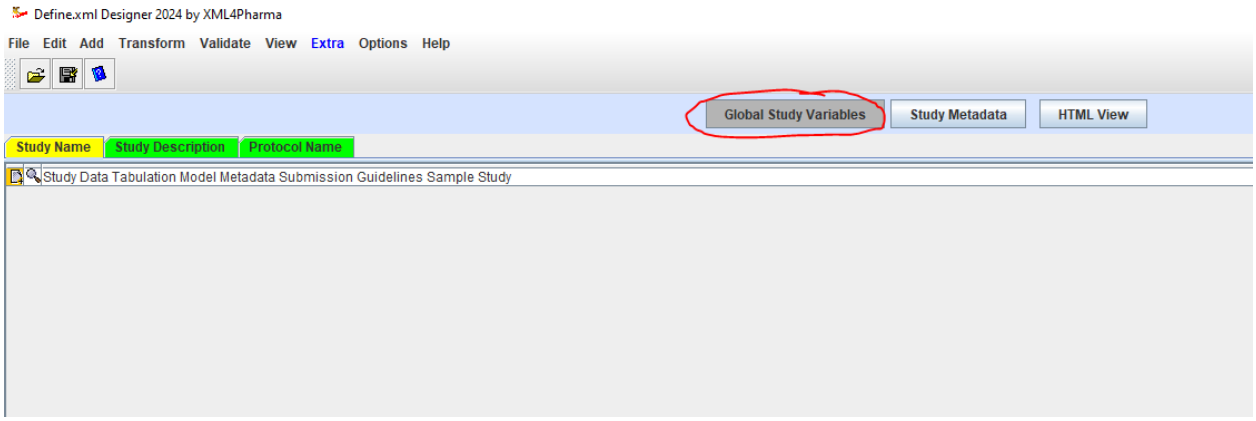
As we already stated that this is an SDTM set of data, and we selected version "2025-09-26" for the CDISC-CT version, we still only need to provide the version of the SDTMIG from the dropbox:



After clicking "OK", a number of "progress bars" appear showing us the progress of the generation of the different parts:

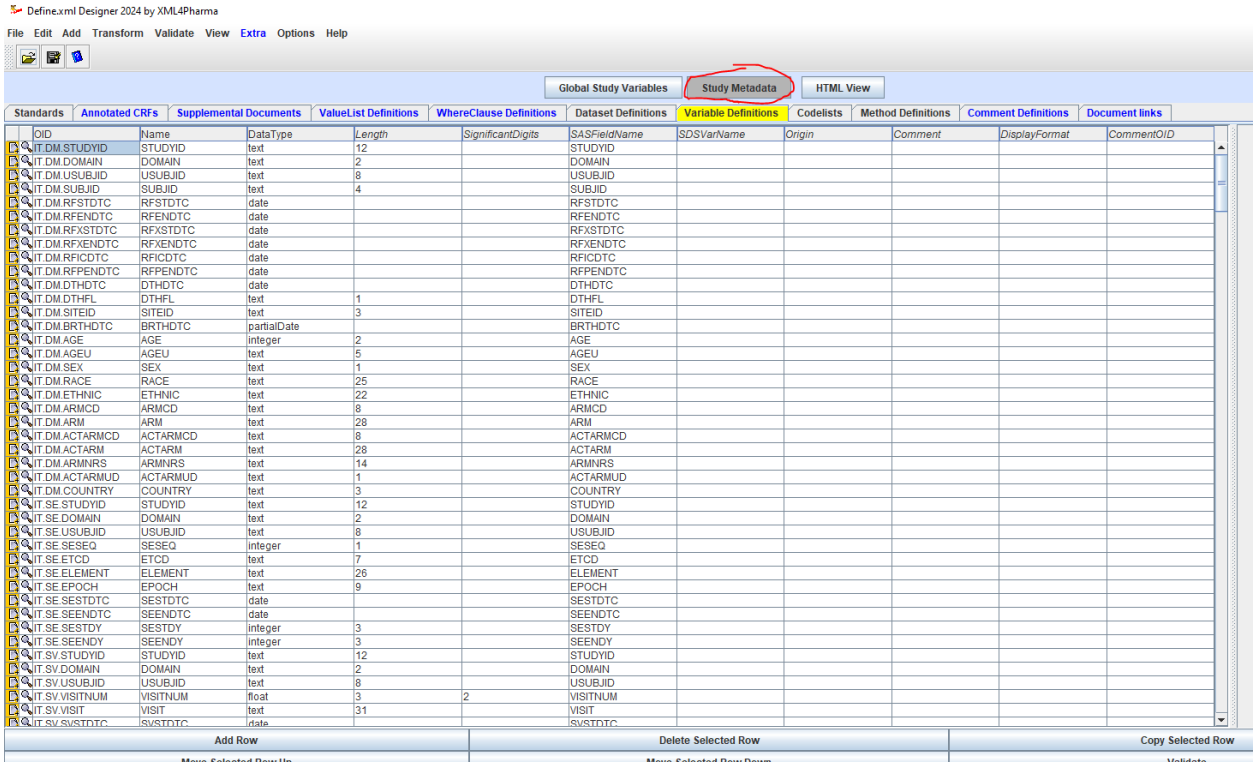


and when finished, the first set of generated data is displayed:

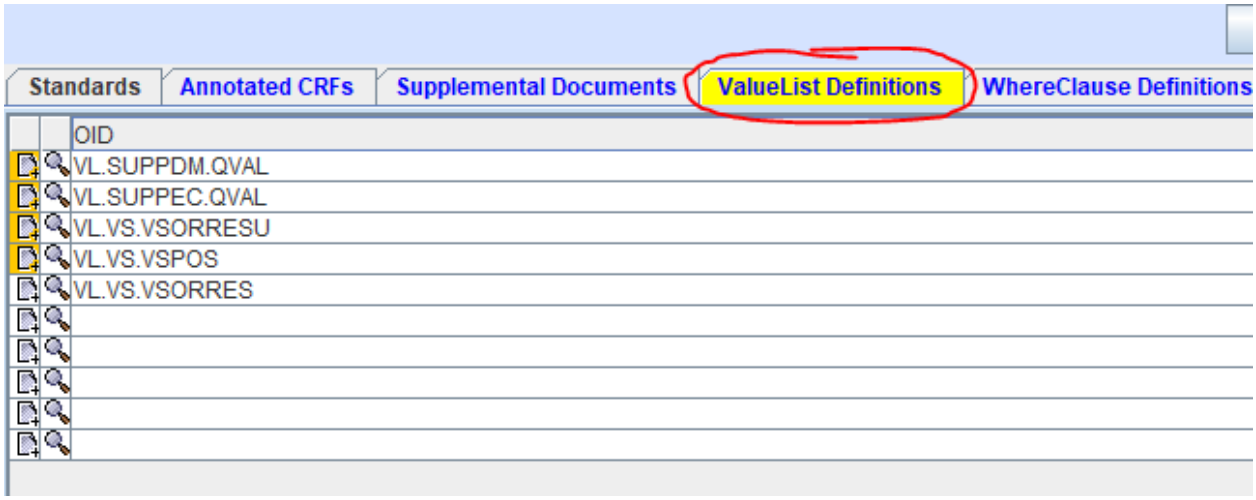


We can then still change the information - see the section "Editing define.xml information".

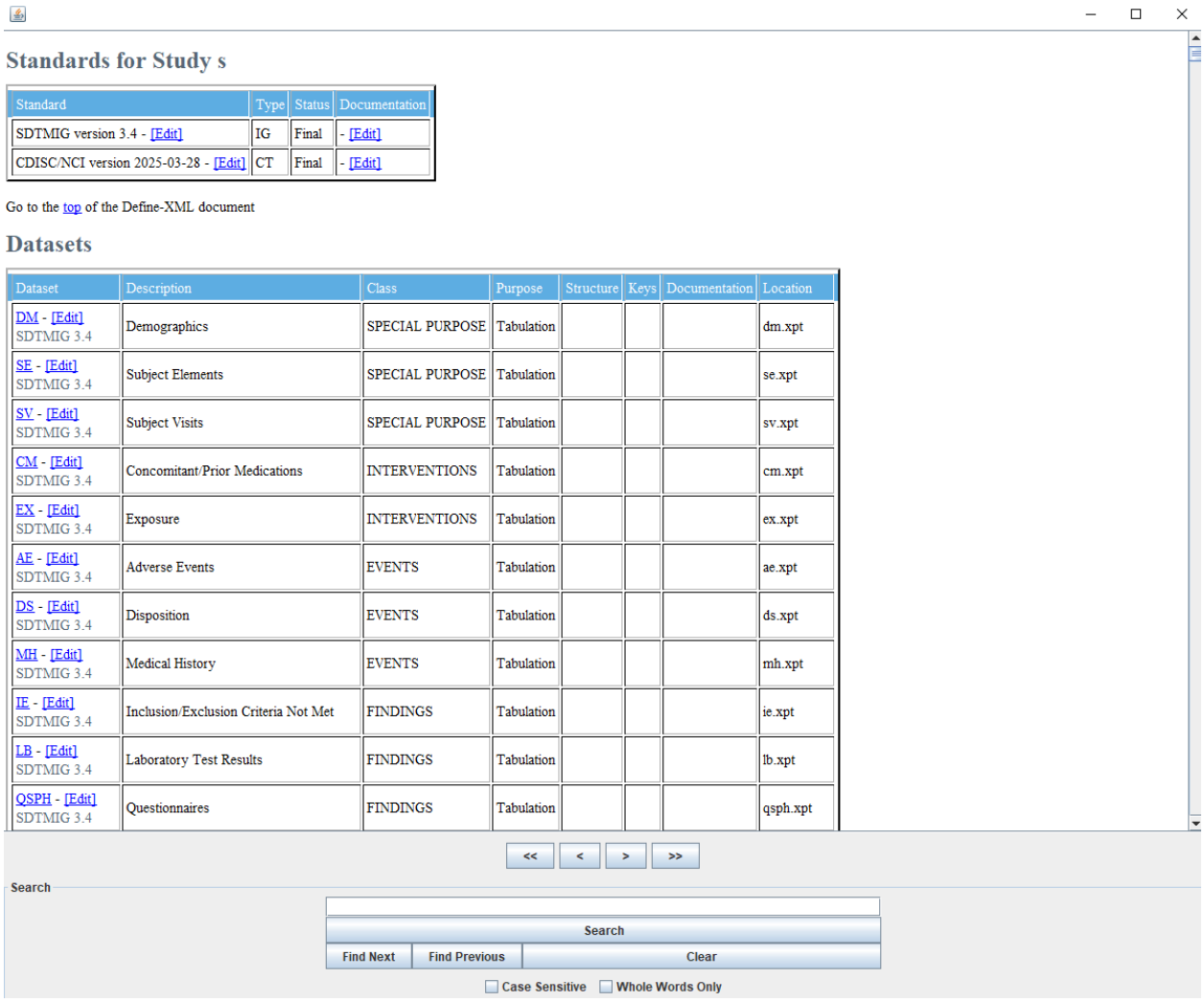
When selecting "Study Metadata" and then selecting the tab "Variable Definitions", we find:



and see that a good amount of information has automatically been generated from the content of the XPT files.
For example, for the "ValueLists":



Using the "HTML View" button (at the top, on the right), generates the "View" of our prototype define.xml:



Standards for Study s

Standard	Type	Status	Documentation
SDTMIG version 3.4 - [Edit]	IG	Final	- [Edit]
CDISC/NCI version 2025-03-28 - [Edit]	CT	Final	- [Edit]

Go to the top of the Define-XML document

Datasets

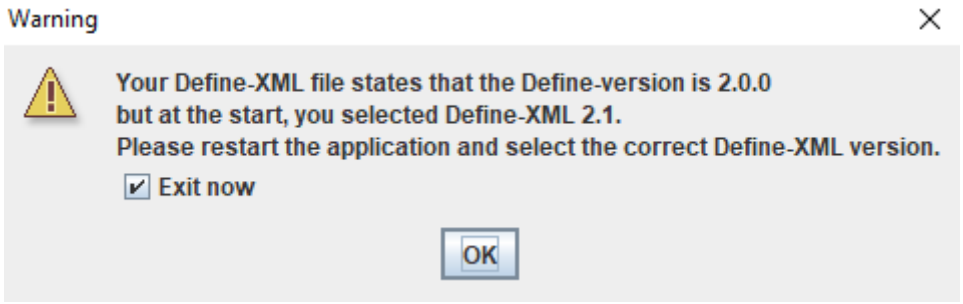
Dataset	Description	Class	Purpose	Structure	Keys	Documentation	Location
DM - [Edit] SDTMIG 3.4	Demographics	SPECIAL PURPOSE	Tabulation				dm.xpt
SE - [Edit] SDTMIG 3.4	Subject Elements	SPECIAL PURPOSE	Tabulation				se.xpt
SV - [Edit] SDTMIG 3.4	Subject Visits	SPECIAL PURPOSE	Tabulation				sv.xpt
CM - [Edit] SDTMIG 3.4	Concomitant/Prior Medications	INTERVENTIONS	Tabulation				cm.xpt
EX - [Edit] SDTMIG 3.4	Exposure	INTERVENTIONS	Tabulation				ex.xpt
AE - [Edit] SDTMIG 3.4	Adverse Events	EVENTS	Tabulation				ae.xpt
DS - [Edit] SDTMIG 3.4	Disposition	EVENTS	Tabulation				ds.xpt
MH - [Edit] SDTMIG 3.4	Medical History	EVENTS	Tabulation				mh.xpt
IE - [Edit] SDTMIG 3.4	Inclusion/Exclusion Criteria Not Met	FINDINGS	Tabulation				ie.xpt
LB - [Edit] SDTMIG 3.4	Laboratory Test Results	FINDINGS	Tabulation				lb.xpt
QSPH - [Edit] SDTMIG 3.4	Questionnaires	FINDINGS	Tabulation				qsph.xpt

Remark that you cannot close this window, it will automatically be updated each time you click the "HTML View" button. You can however of course "minimize" it.
Remark the "[Edit]" hyperlinks in the tables. We will later explain their usage.

Starting from an existing define.xml file

You can of course also start from an existing define.xml file, either version 2.0 or 2.1 (the software does not support Define-XML version 1.0 anymore). This can be a define.xml file that was created using this software, or from any other system or software.
In order to do so, use the menu "File - Open define.xml", select the file using the file chooser, and it will then be loaded.

If you choose the wrong Define-XML version at the start, a warning message will be displayed, e.g.:



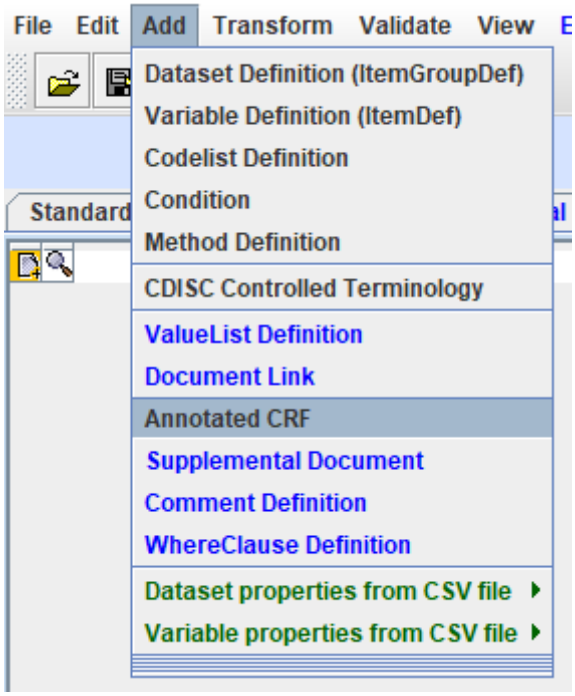
In most cases you will want to have the application stopped immediately. If you don't want this, uncheck the "Exit now" checkbox. However, this can then further lead to a lot of unexpected behavior.

Editing define.xml information - Basics

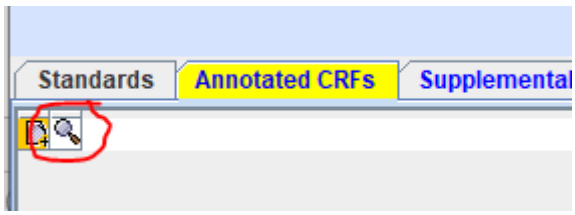
There are different ways in which the content of the define.xml can be altered using the Define-XML Designer: using the tables and the table editor, using wizards, and from the HTML view of the define.xml.

Adding the Annotated CRF information

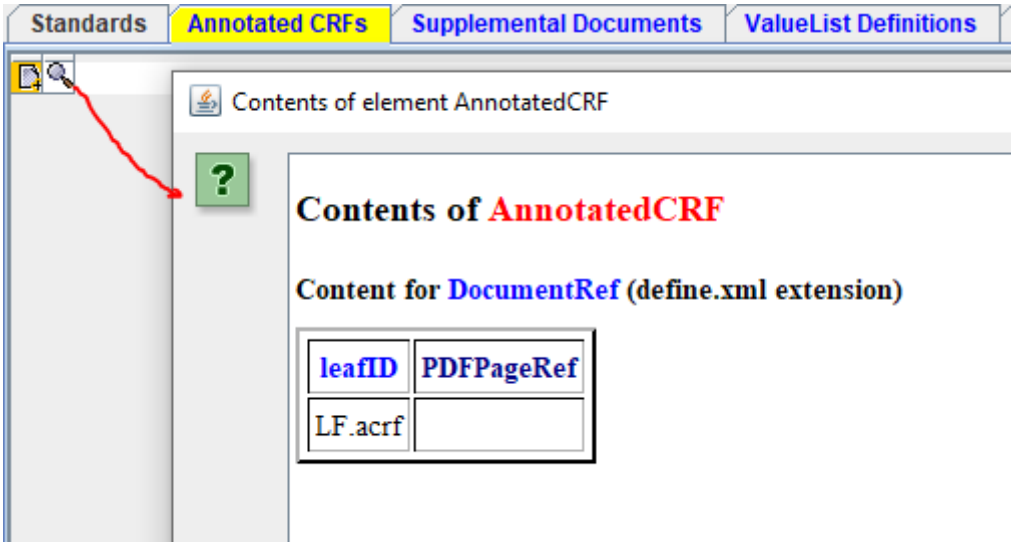
Especially in the case of SDTM, you will probably want to add the information that there is an annotated CRF.
If you know the Define-XML standard already well, you can do so using the tabs "Document links" (which allow you to generate define.xml "def:leaf" elements for pointing to external documents" and "Annotated CRFs".
However, there is an easier way that automates this step completely.
To do so, use the menu "Add Annotated CRF":



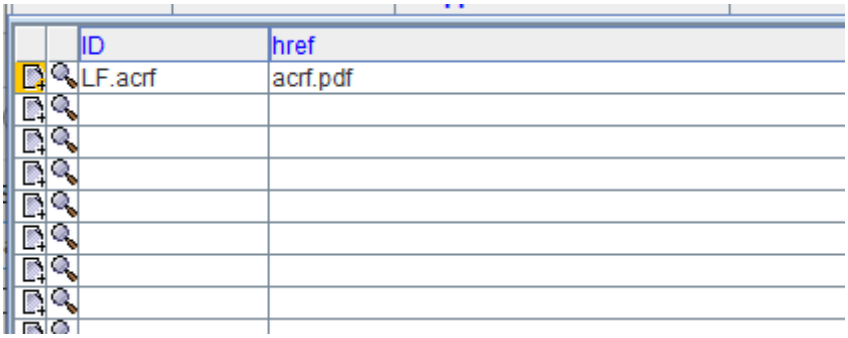
This then creates the "def:leaf" for an annotated CRF with the name "acrf.pdf" for you and references it in the "AnnotatedCRFs" element. The system then automatically jumps to the tab "Annotated CRFs" where one then finds:



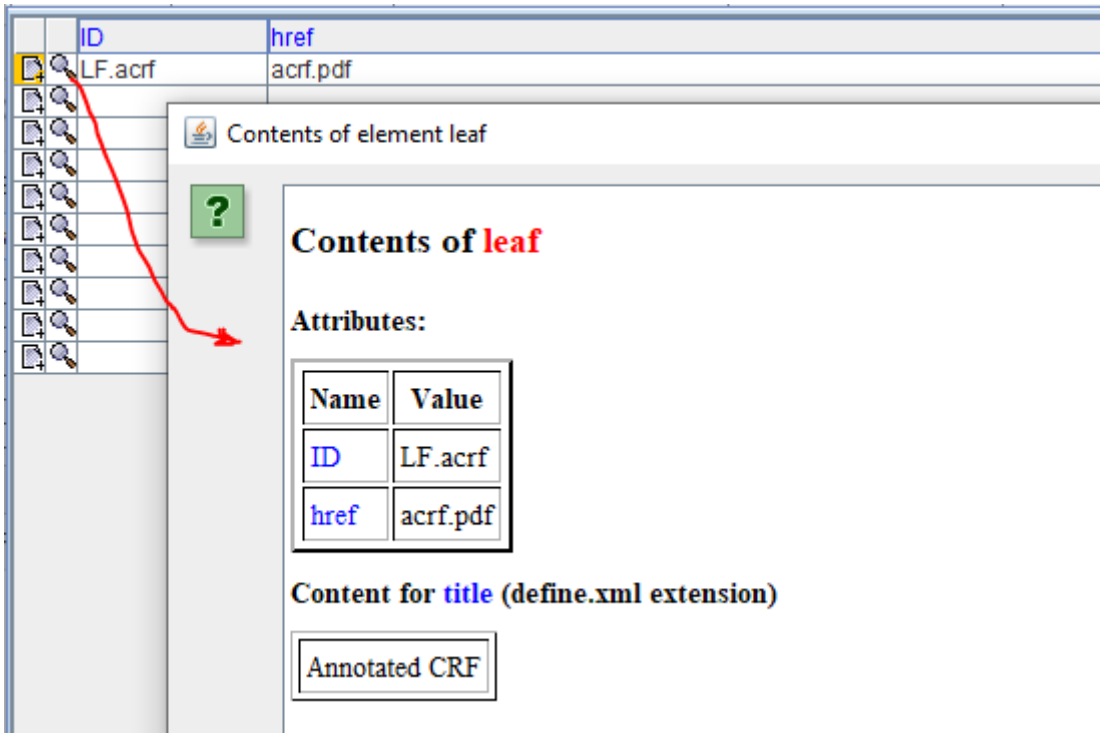
and if one then clicks on the "View" icon (the magnifying glass), the following information is displayed:



Similarly, when selecting the "Document links" tab, one finds:

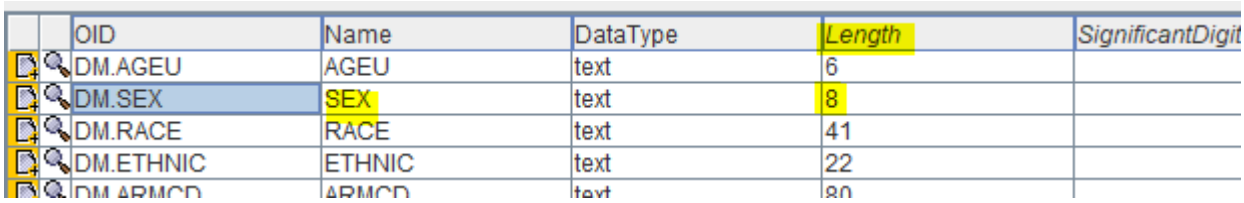


and one clicks on the "View" icon (the magnifying glass), one gets the detailed information:

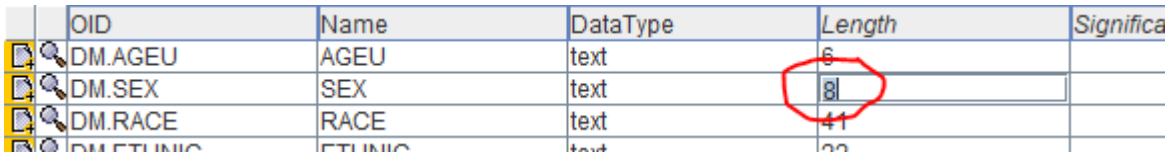


Simple Editing

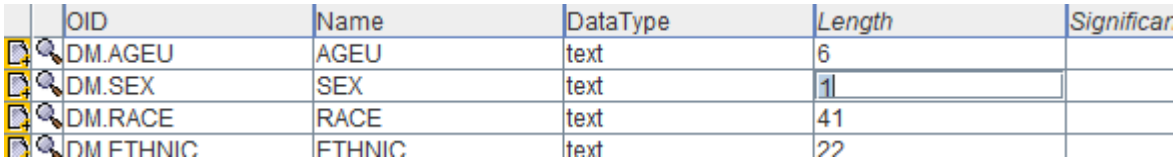
Let us start with the most easy one: editing an attribute of a variable. As an example, we can take the variable "SEX" in DM (Demographics). In case we started from a template, the "Length" attribute, representing the maximal length of the value for "SEX" has been set to "8" in case we opted to have the Length set from the longest value in the codelist. which is "INTERSEX".



But in the CRF, only "Male" and "Female" are used, which need to be translated to CDISC-CT values "M" and "F". So, we want to change the "Length" to "1".
In order to do so, just click in the cell for "Length" for "SEX" (which currently has the value "8"). The cell now becomes editable:

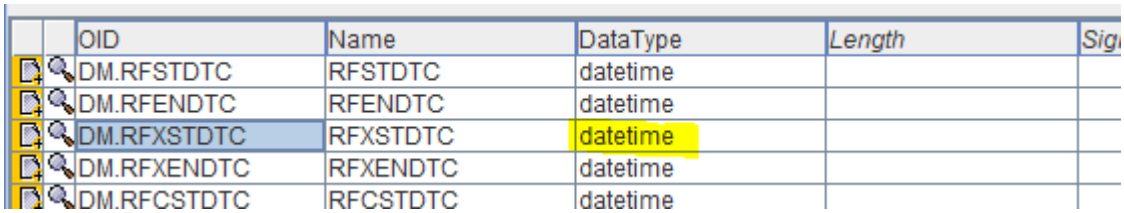


and one can just type in another value, e.g. "1":



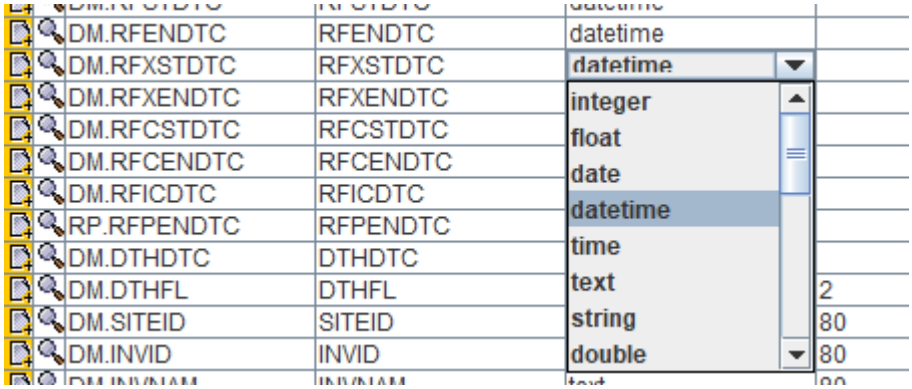
Remark that for such cells that expect an integer, any other character than 0 to 9 will be refused.
Later, we will also learn about how to automatically update the values for the variable length from one or more SAS-XPT files (menu "Extra - Adapt Variable Length from SAS-XPT file contents"). So, at the end of the generation of the datasets, you will not need to do this manually.

There are also columns where the value is enumerated, e.g. the "DataType" column. E.g. for RFXSTDTC (Date/Time of First Study Treatment), we currently have:

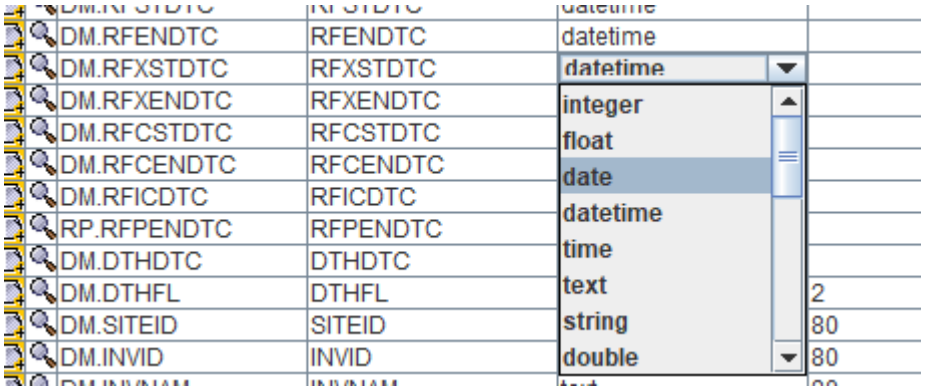


OID	Name	DataType	Length	Sig
DM.RFSTDTC	RFSTDTC	datetime		
DM.RFENDTC	RFENDTC	datetime		
DM.RFXSTDTC	RFXSTDTC	datetime		
DM.RFXENDTC	RFXENDTC	datetime		
DM.RFCSTDTC	RFCSTDTC	datetime		

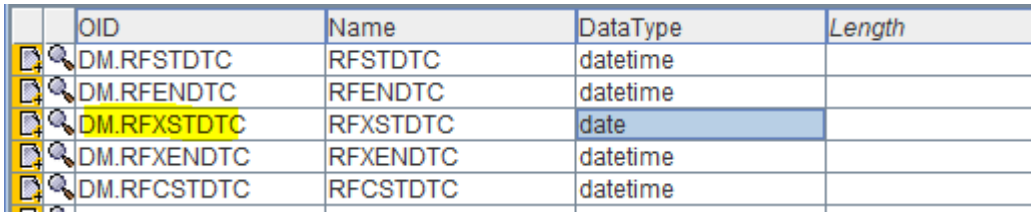
and when clicking in the cell, the following "dropdown" is shown:



allowing us to change the value for "DataType".
For example, if this is a very simple study where there can never be more than one study drug exposure per day, it may be that only a "date" (i.e. without time part) is collected, so we should set the "DataType" to "date":



with the result (after releasing the mouse button) being:



OID	Name	DataType	Length
DM.RFSTDTC	RFSTDTC	datetime	
DM.RFENDTC	RFENDTC	datetime	
DM.RFXSTDTC	RFXSTDTC	date	
DM.RFXENDTC	RFXENDTC	datetime	
DM.RFCSTDTC	RFCSTDTC	datetime	

We can then similarly do so for the other timing variables in DM.
Remark that for most studies, only collecting the date, without time part, if often a bad idea, as it can easily lead to problems when relative timings like "BEFORE", "AFTER", "DURING" need to be assigned, or when the EPOCH must be assigned based on timing variable values.

Some variables may need "floating point" assignment. This often is the case for --STRESN (Numeric Result/Finding in Standard Units), for which, when starting from a template, the value for "Length" is set to the default "8" and for "SignificantDigits" is set to "2".
Remark here that the designation "SignificantDigits" may be confusing: it defines the number of characters after the decimal point, and "Length" defines the total number of characters including the decimal point. Some examples:

Value	define.xml "Length"	define.xml "SignificantDigits"
3.14	4	2
997.23	6	2
0.1567	6	4
-0.1567	7	2

Another typical example where we may change want to change "DataType" and "Length" is "VISITNUM. For example, when starting from the template, we find for SV (Subject Visits):

OID	Name	DataType	*Length*	*SignificantDigits*
AG.AGDOSFRM	AGDOSFRM	text	64	
AG.AGDOSFRQ	AGDOSFRQ	text	17	
AG.AGROUTE	AGROUTE	text	28	
SV.VISITNUM	VISITNUM	float	8	1

which can be seen as a "safe choice" for the possibility that there are "unscheduled" visits.
However, if we have no unscheduled visits, and the visit numbers can e.g. only be "1" to "5", we can edit the cells for "DataType", "Length" and "SignificantDigits" to just:

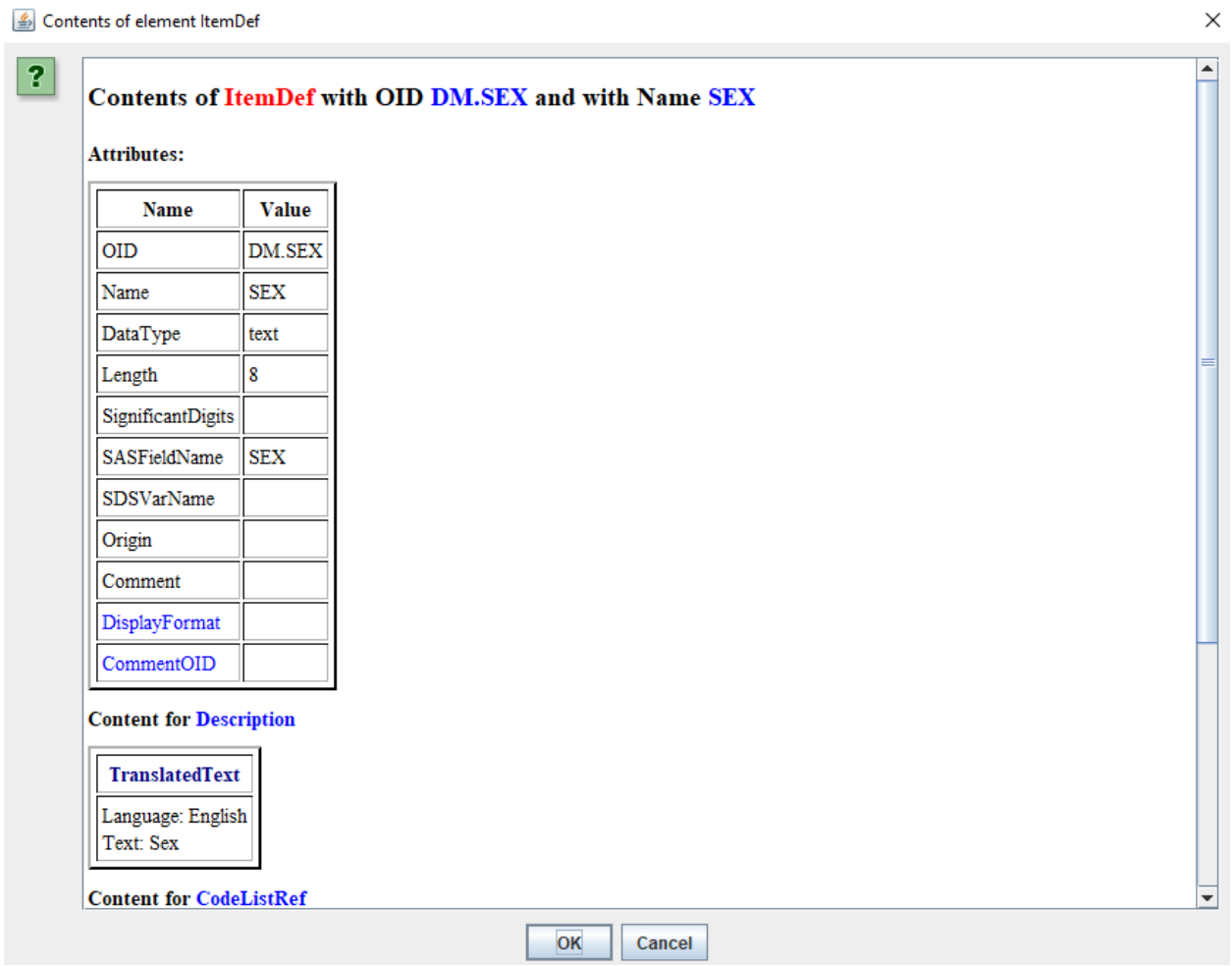
OID	Name	DataType	*Length*	*SignificantDigits*	*SA*
AG.AGDOSFRM	AGDOSFRM	text	64		AG
AG.AGDOSFRQ	AGDOSFRQ	text	17		AG
AG.AGROUTE	AGROUTE	text	28		AG
SV.VISITNUM	VISITNUM	integer	1		VIS

Simple Viewing

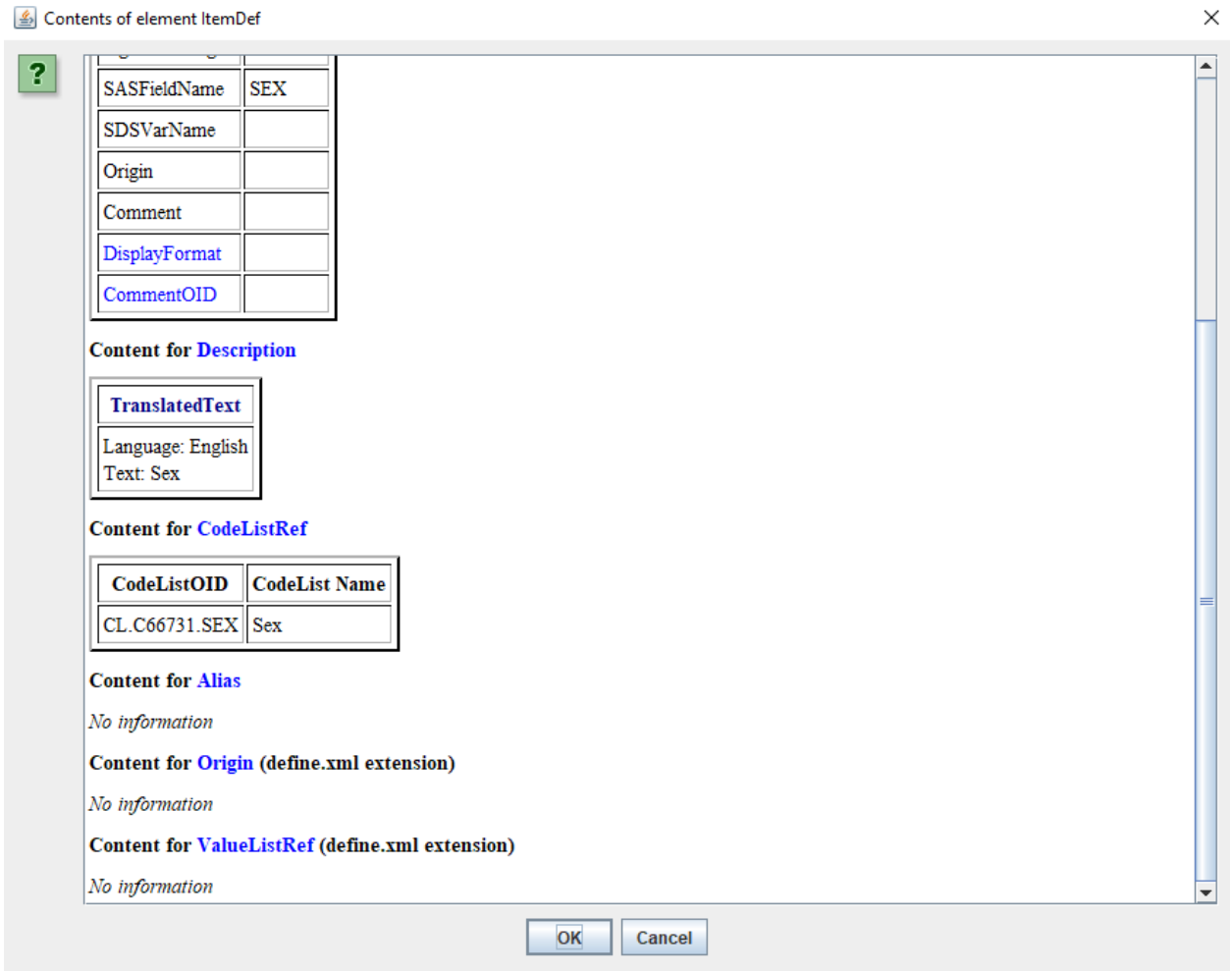
One will surely already have observed that in most of the tables, the first 2 columns contain "clickable" symbols, i.e.:

Symbol	Meaning
	Edit sub-information
	View sub-information

Let us start with the clickable symbol "View sub-information".
When we click it for the Variable Definition for "DM.SEX", a new read-only dialog is displayed:

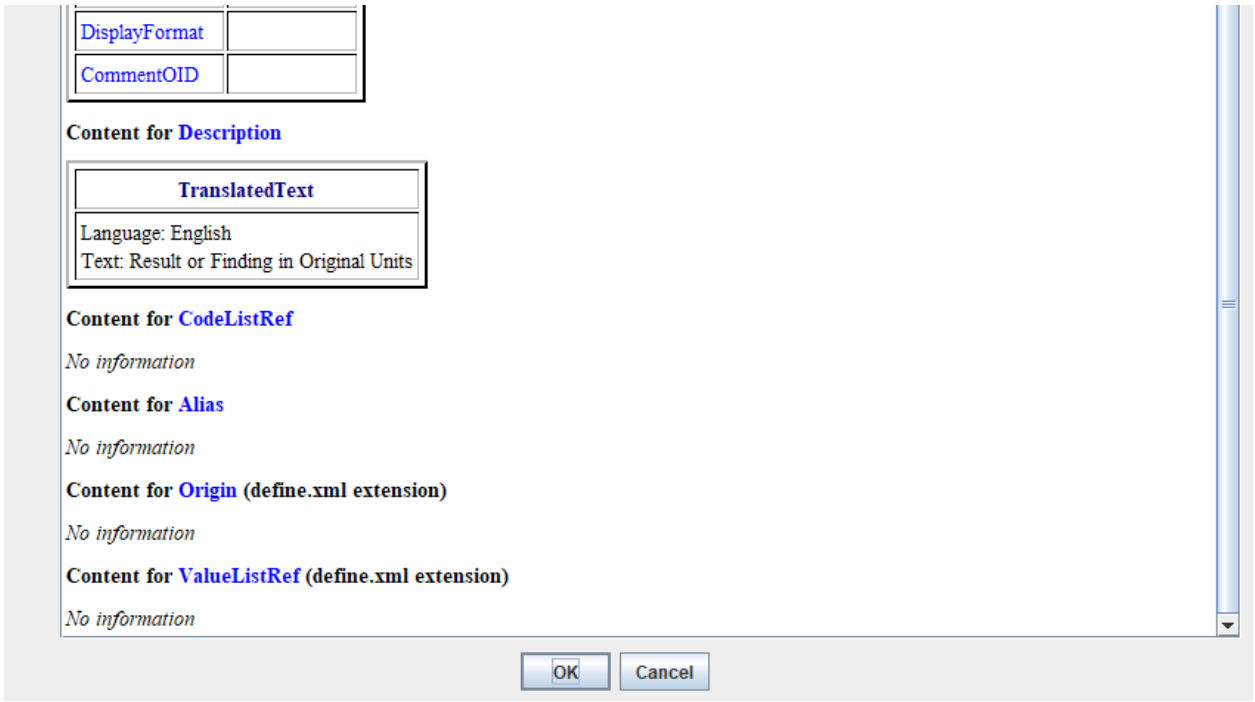


and when scrolling down:



showing us that there is a codelist with the name "Sex" and OID "CL.C66731" is associated with the variable "SEX", and that no "Origin" has been assigned, and that there is no "ValueList" assigned.

If we do the same e.g. for "LBORRES" (Results of Findings in Original Units), we also see that the "Origin" has not been assigned.

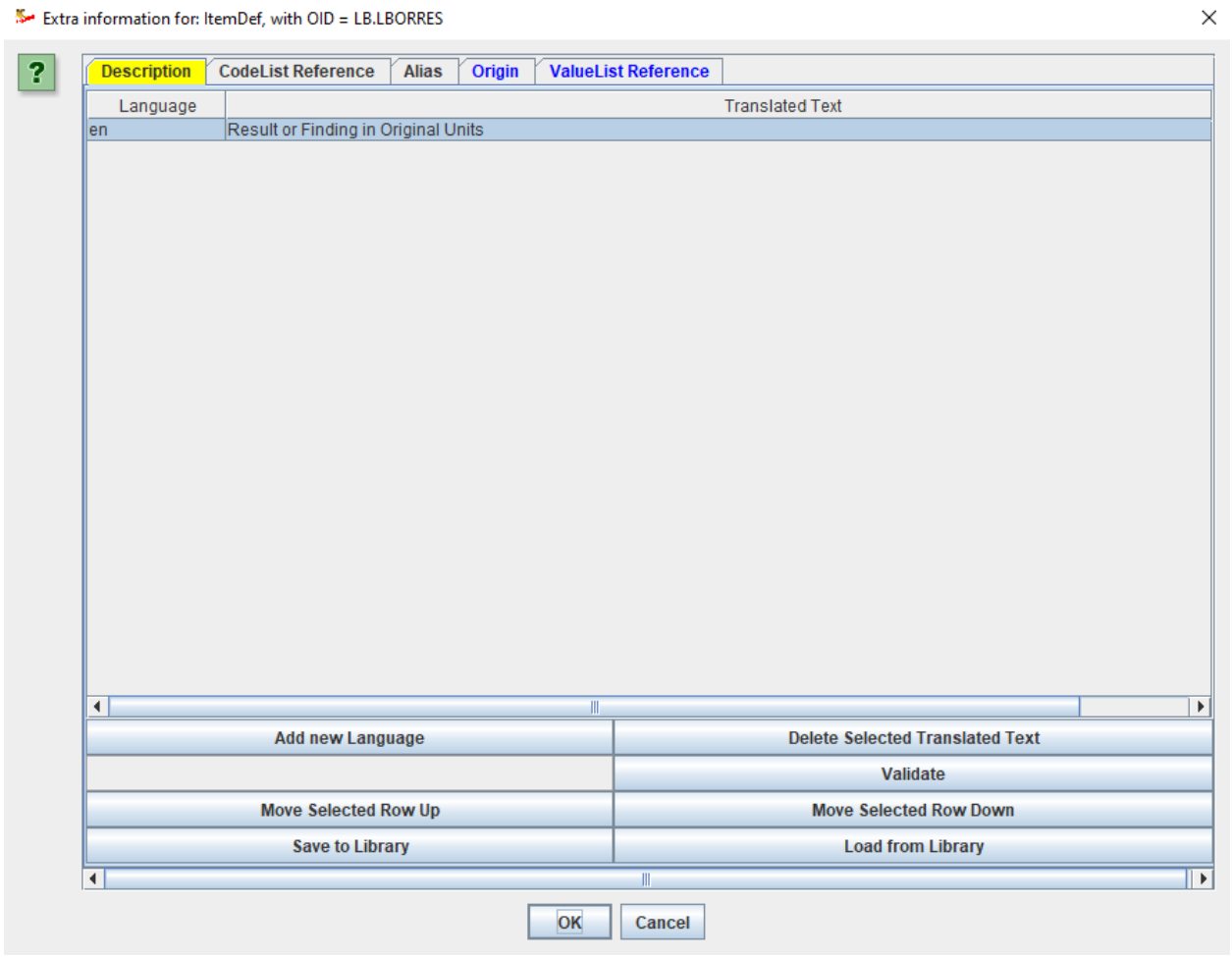


Although not formally mandatory, in many cases, we will often want to generate a "ValueList" for LBORRES, as the properties of LBORRES may depend on the value of LBTESTCD. For example, for some tests, such as for concentrations, we will have a numeric value, whereas for others, the value will just be text, or text that is enumerated, such as "POSITIVE" and "NEGATIVE". This will e.g. be often the case for urine tests.

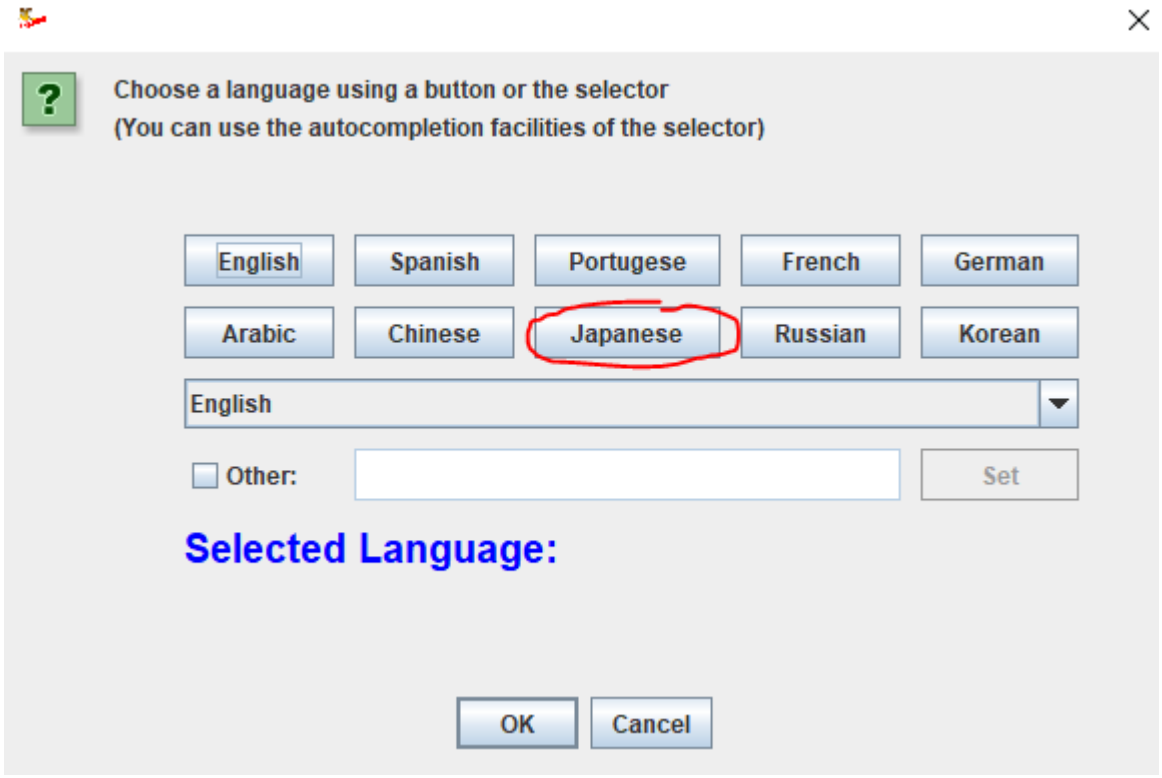
Editing sub-information

But how can we now edit this sub-information, e.g. assign the "Origin"?

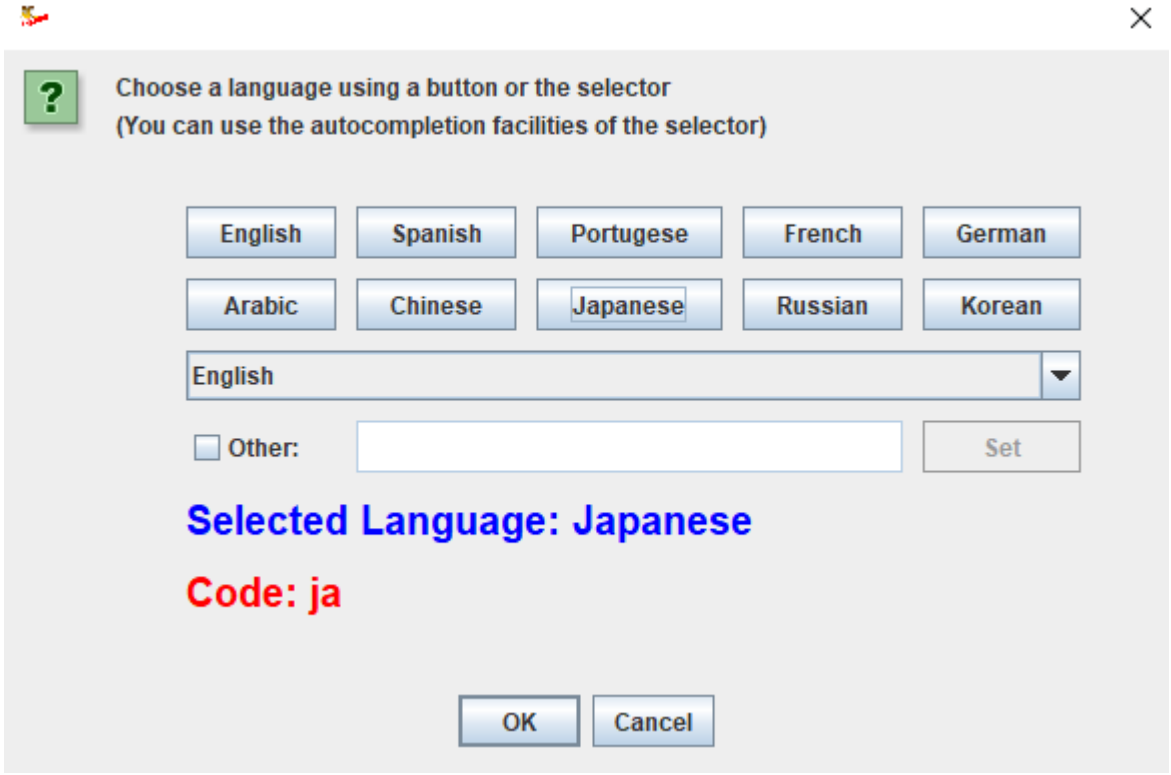
This becomes possible by clicking the "+" button  . For example, for LBORRES, this opens a new dialog with the following table:



It has a number of tabs: "Description", "CodeList Reference", "Alias", "Origin", and "ValueList Reference".
In the case of SDTM or SEND, one will usually not want to change the "Description", as this is essentially the "variable label". In the case of ADaM, this may be information that one needs to add, or wants to change.
Now, suppose that we also want to add a "Description" in another language than English, e.g. Japanese.
We then first click the "Add new Language" button, leading to:

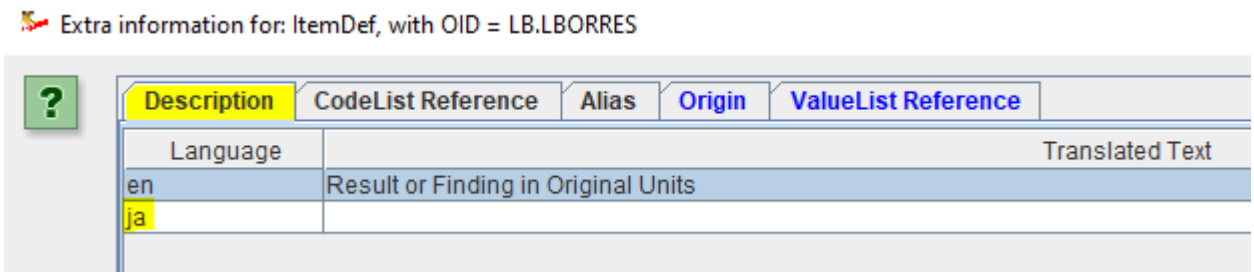


and then click "Japanese", leading to:

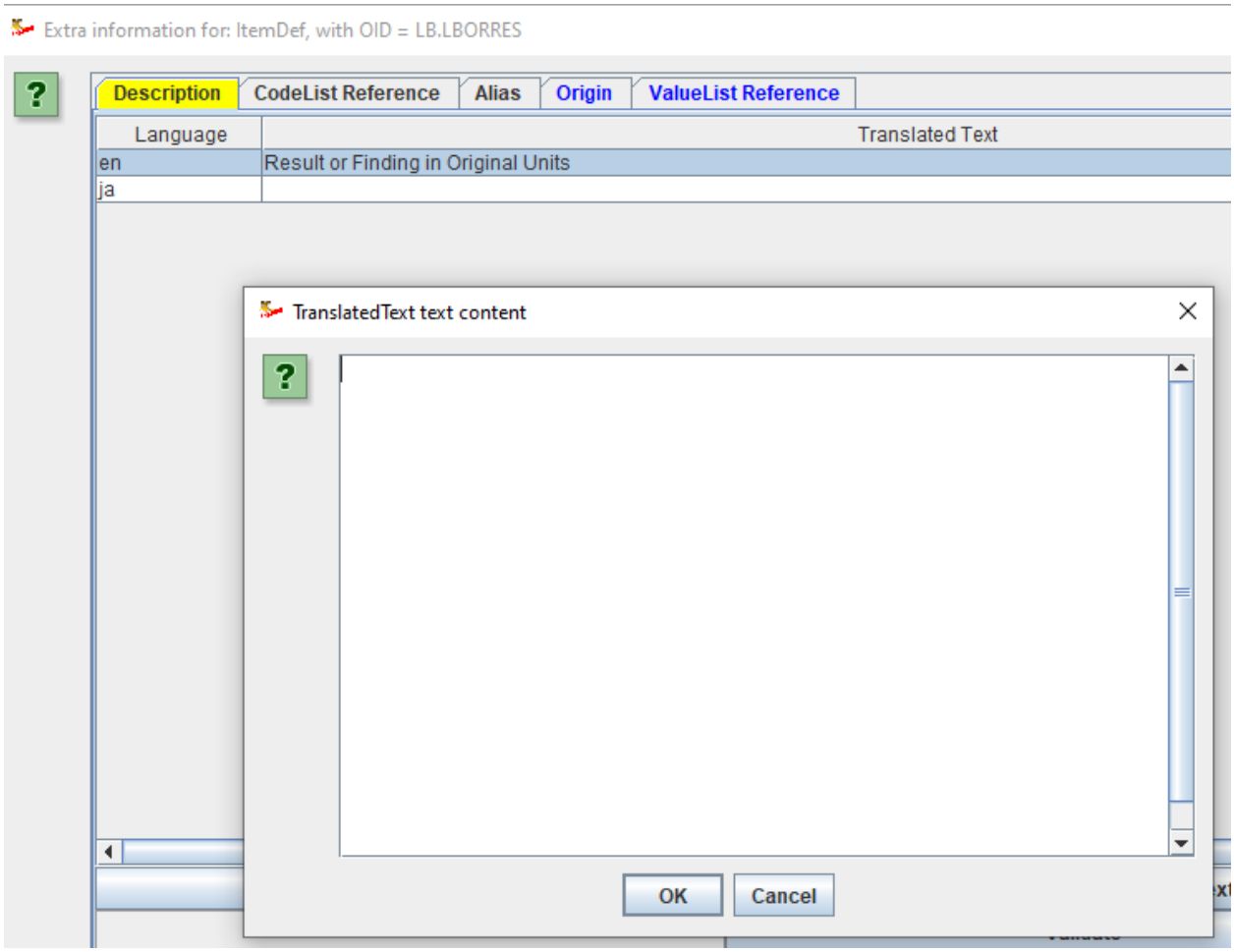


which shows us that the language code for "Japanese" is "ja".
Remark that if we want to have a language selected that is not one on one of the buttons or in the dropdown, we will need to use the checkbox "Other", and add the two-character code ourself.

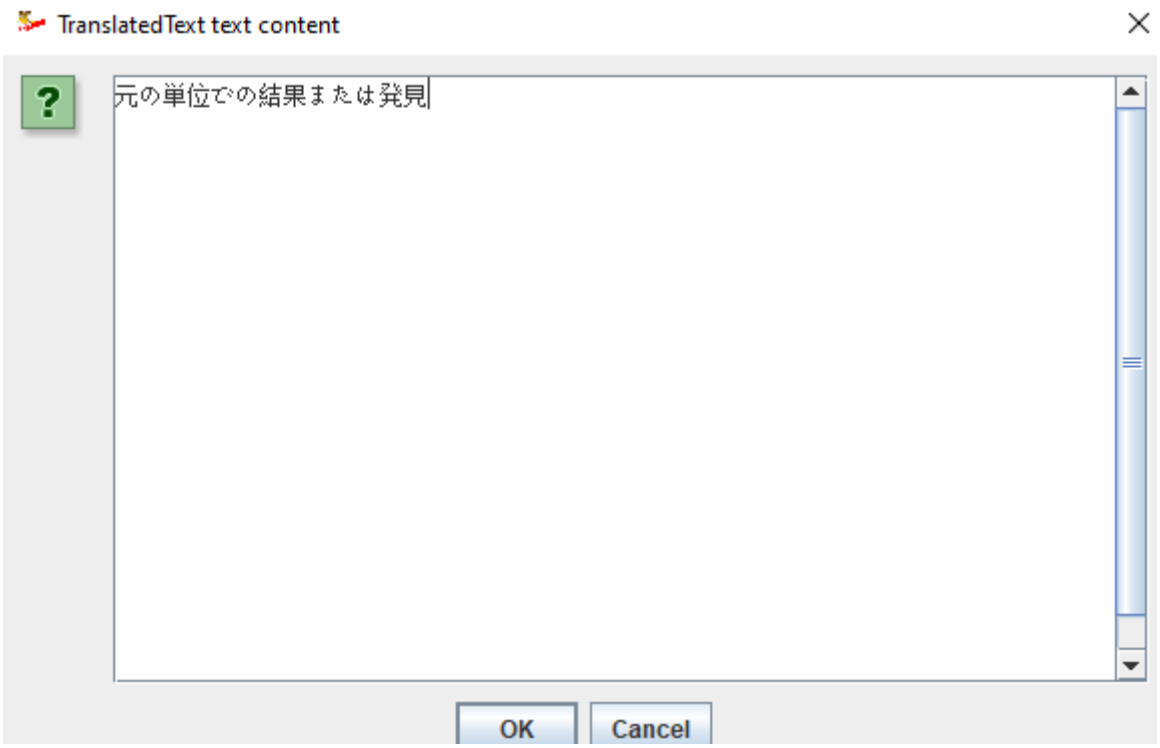
Clicking "OK" leads to:



We can now add the Japanese description by clicking in the empty cell on the right side for "ja", leading to a dialog:

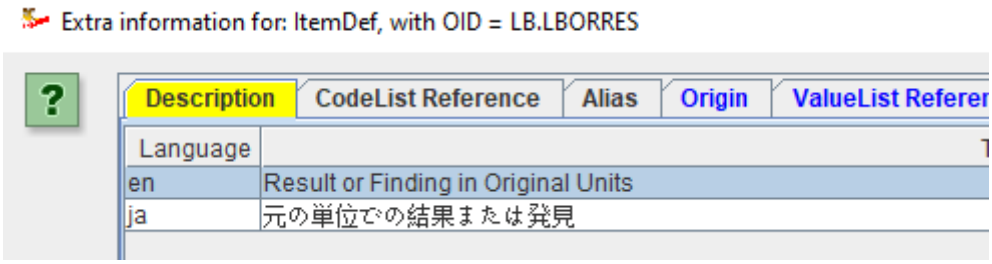


in which we can add our Japanese description, like:



Remark this is essentially a multiline editor. For a variable label, one will usually only have one line, but we will later also see cases where we want to add more than one line, e.g. for methods in the case of ADaM, where we e.g. want to add R- or SAS-code.

After clicking "OK", we get:



P.S. Currently, there is no obligation at all, also not for the PMDA (Japanese regulatory authorities) to have labels in the Japanese language. Reason is probably that we still need to submit datasets in outdated SAS-XPT format, which does not support non-ASCII (such as Japanese) characters, and the label in the define.xml must correspond to the label in the SAS-XPT. This may change once regulatory authorities start accepting submissions using the modern CDISC Dataset-JSON format. For information exchange between different departments in different countries of sponsors, having information in different languages may make sense.

When then clicking "OK" until we are in the main table, and then clicking the "View sub-information" icon then leads to:

SASFieldName	LBORRES
SDSVarName	
Origin	
Comment	
DisplayFormat	
CommentOID	

Content for Description

TranslatedText
Language: English Text: Result or Finding in Original Units Language: Japanese Text: 元の単位での結果または発見

Content for CodeListRef

No information

Content for Alias

No information

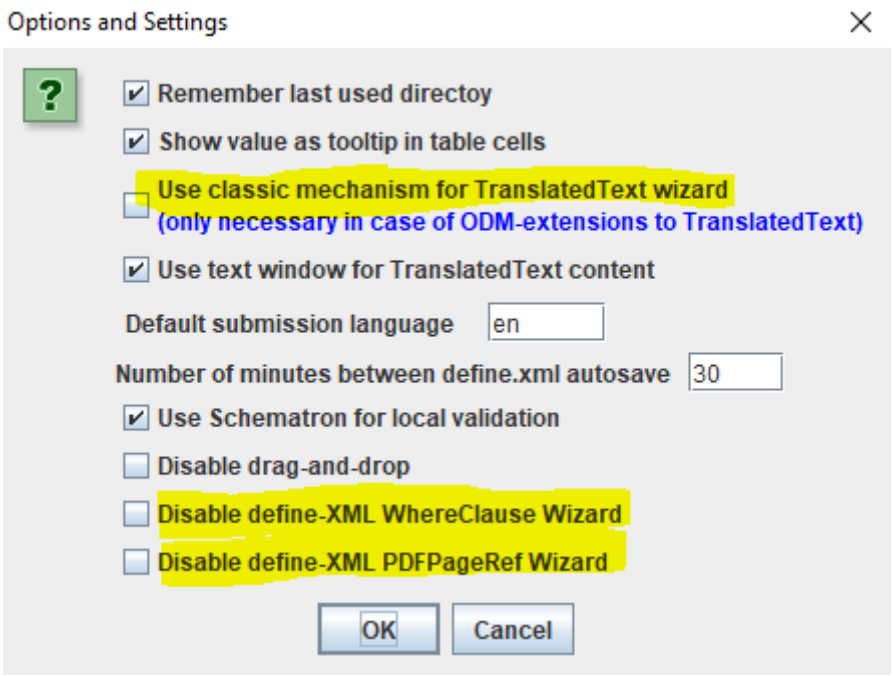
Content for Origin (define.xml extension)

No information

Content for ValueListRef (define.xml extension)

No information

We will often see that editing sub-information uses "wizards" or very user-friendly dialogs. If one does not want to use such wizards and "smart" dialogs, one can switch their use off by the menu "Options - Setting" by selecting one of the checkboxes:

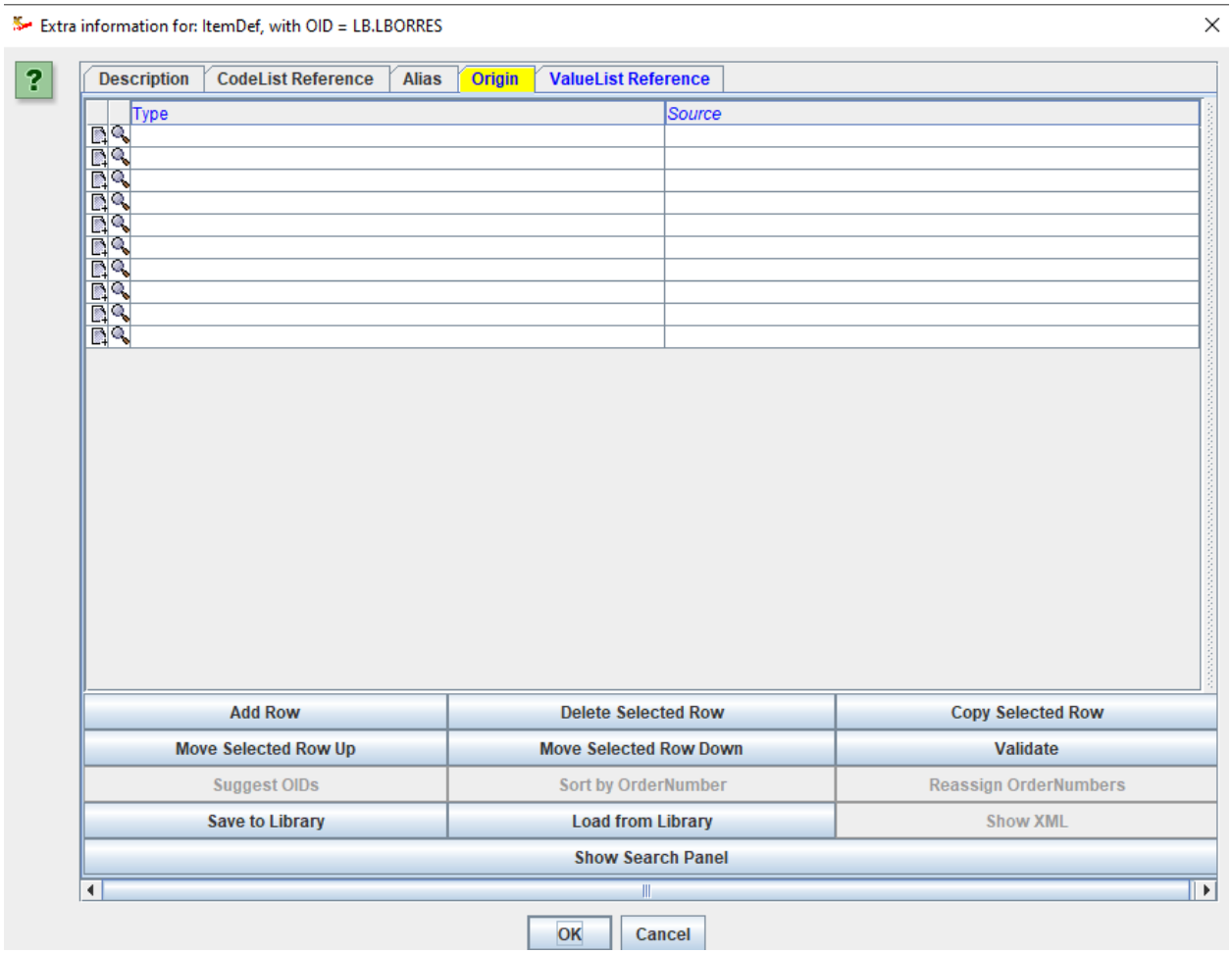


There are however only very few users who want to do this …

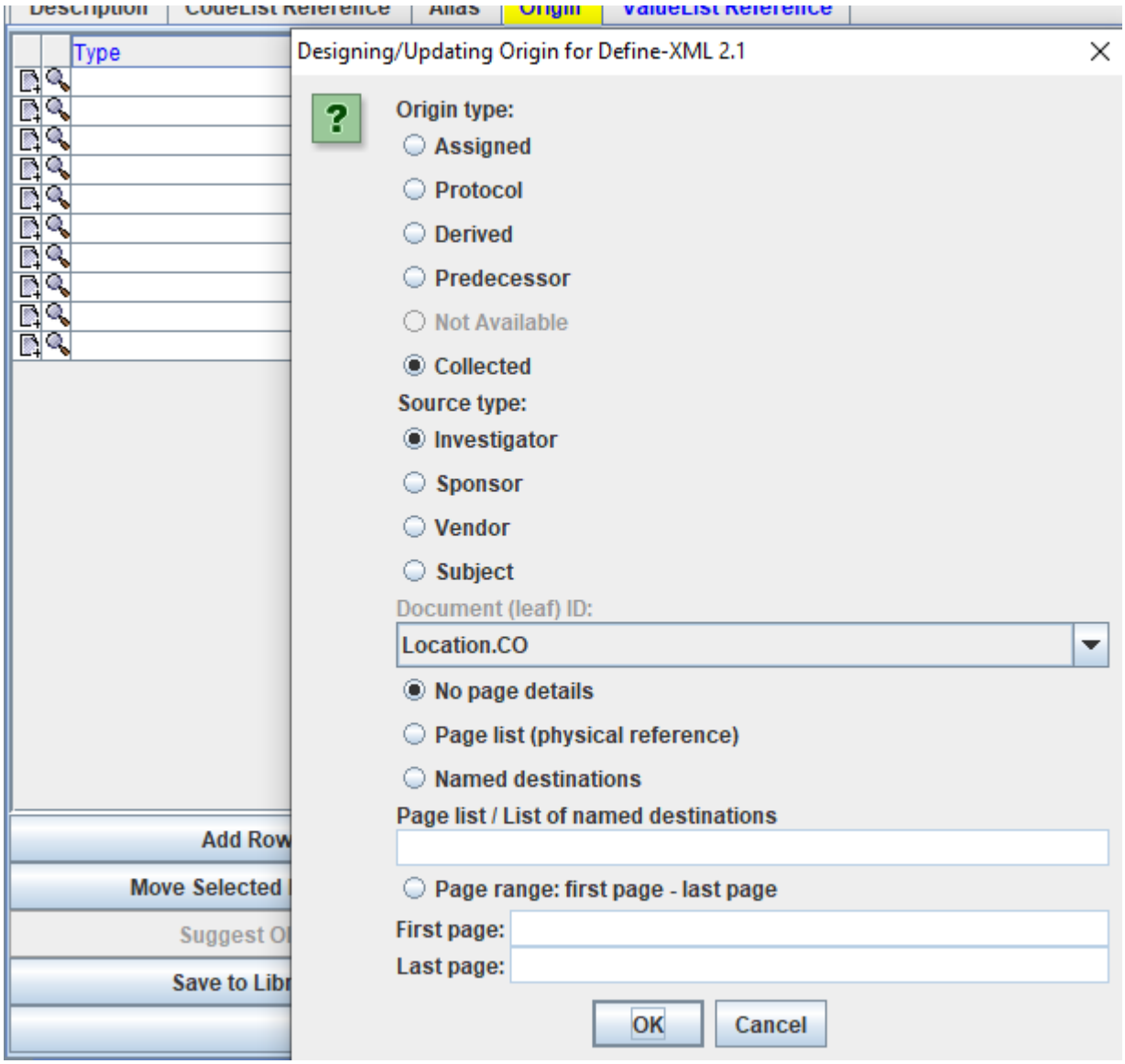
Adding / Editing Origin/Source information

One of the important pieces of information in the define.xml, whether it is for SDTM, SEND or ADaM, is the "Origin" information. In Define-XML 2.0, only "Origin" is used, whereas in Define-XML 2.1, it is extended with "Source.

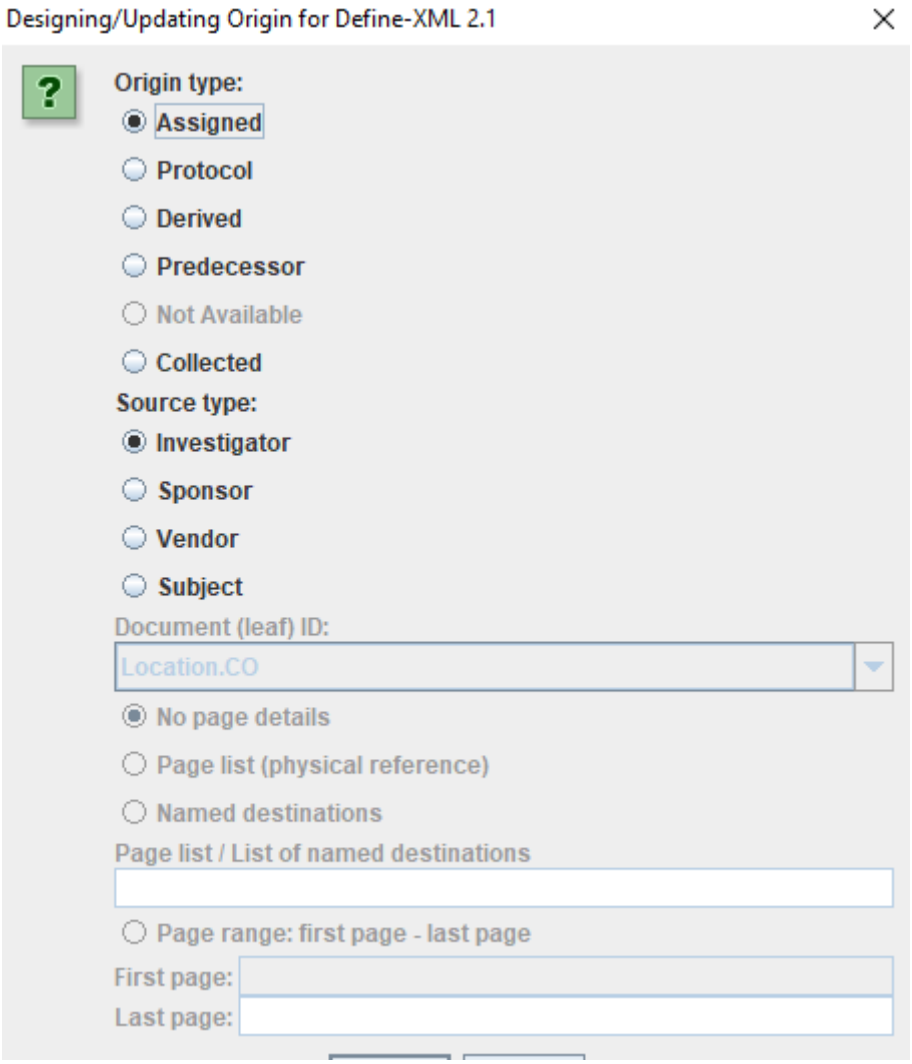
Let us take the "LBORRES" example which we demonstrate for Define-XML 2.1. After clicking the "Edit sub-information" icon from the main table of "Variable Definitions" for LBORRES, and then selecting the "Origin" tab, we get:



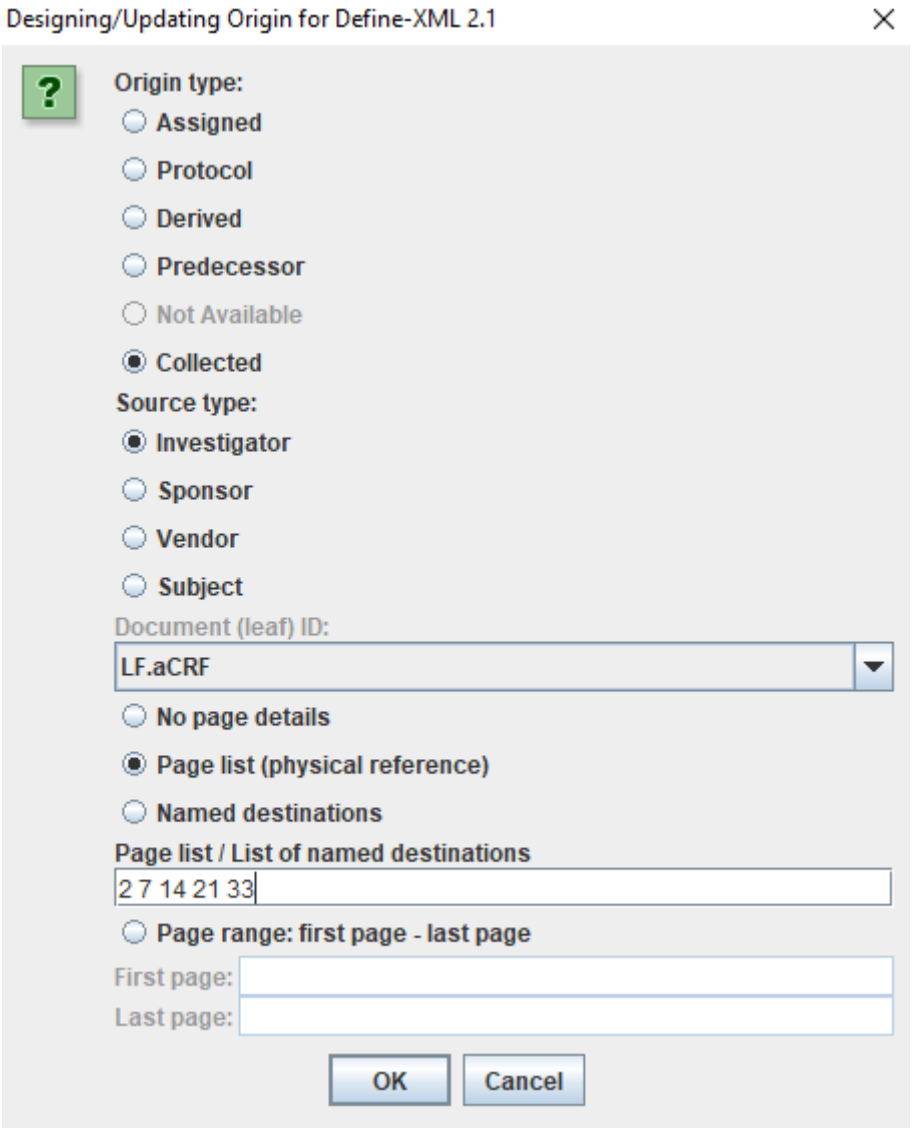
Remark that the Define-XML specification allows for multiple origins, but this will seldom be the case for SDTM and SEND. The better way of dealing with multiple origins is to use ValueLists anyway. For example, for LBORRES one can have that some lab data was collected on the CRF, and others were obtained from a lab (Source=Vendor).
Let us suppose, that we want to add the information that the lab data were obtained from the CRF.
When we first click "Type", the wizard shows up:



The wizard knows about the allowed combinations (see section 4.3.2 of the Define-XML 2.1 specification). For example, when selecting the radiobutton "Assigned", the other choices reduce to:



But let us add some information for the case the lab original results were collected by the investigator from the CRF. We can then add some information, such as:



P.S. We will later see how the "LF.aCRF", i.e. the document to the annotated CRF can be added.

If your annotated CRF is not final yet (which is typical the case when we develop the define.xml before the study starts, as a "specification" of what we later want to submit), one can select "No page details", and add the information later.

Clicking "OK" and using the "View sub-information" icon, we get:

Content for Description

TranslatedText
Language: English Text: Result or Finding in Original Units Language: Japanese Text: 元の単位での結果または発見

Content for CodeListRef

No information

Content for Alias

No information

Content for Origin (define.xml extension)

Type	Source	Description	DocumentRef
Collected	Investigator		Attr.Name: leafID; Attr.Value: LF.aCRF def:PDFPageRef

Content for ValueListRef (define.xml extension)

No information

OK Cancel

and when clicking the "HTML View" button (which takes more time), and looking for LBORRES, we see:

LBCAT - [Edit]		Category for Lab Test	text	Grouping Qualifier	80		[Add]		[Add]
LBSCAT - [Edit]		Subcategory for Lab Test	text	Grouping Qualifier	80		[Add]		[Add]
LBORRES - [Edit]		Result or Finding in Original Units	text	Result Qualifier	80		Collected / Investigator [Edit] Annotated CRF [2 7 14 21 33]		[Add]
LBORRESU - [Edit] [Create ValueList] [Add ValueList]		Original Units	text	Variable Qualifier	25	Unit - [Edit]	[Add]		[Add]
LBRESSCL - [Edit]		Result Scale	text	Record Qualifier	17	Result Scale Response - [Edit]	[Add]		[Add]
LBRESTYP - [Edit]		Result Type	text	Record Qualifier	32	Result Type Response - [Edit]	[Add]		[Add]
LBCOLSRT -		Collected Summary Result		Record		Collected Summarized Value Type			

<< < > >>

Search

LBORRES

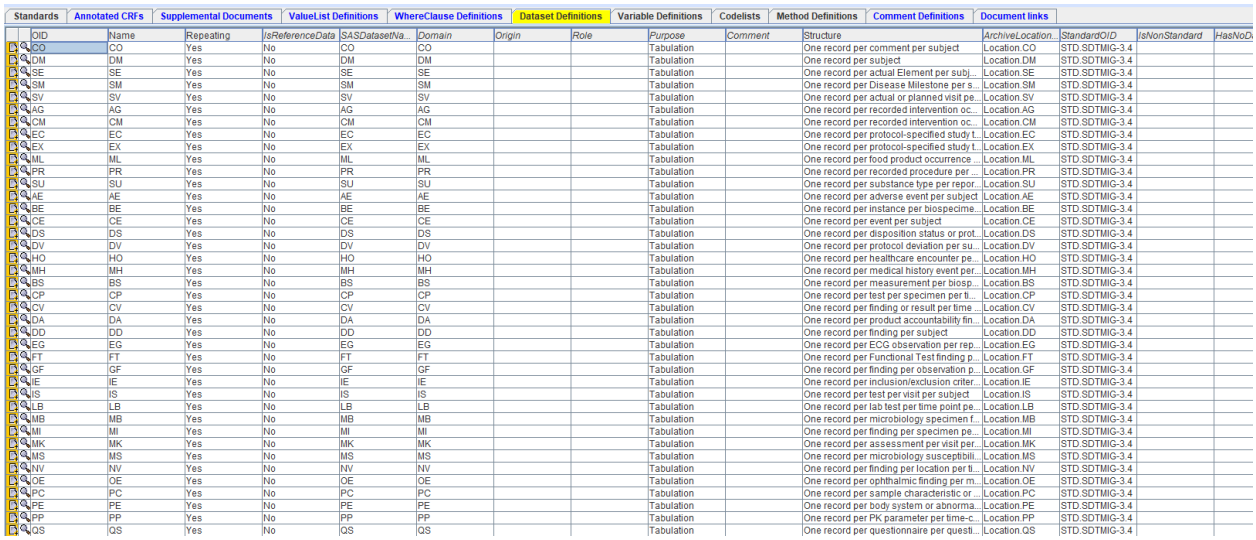
Search

Find Next | Find Previous | Clear

Case Sensitive | Whole Words Only

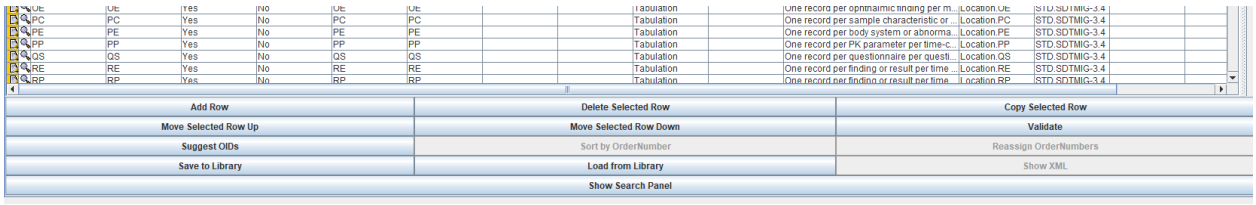
Adding / Editing "Dataset Structure" and other dataset properties

Similarly, we can of course also edit information regarding the datasets. In the define.xml, dataset definitions are represented by the "ItemGroupDef" XML element.
When we use the tab "Dataset Definitions", we e.g. find:



OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role	Purpose	Comment	Structure	ArchiveLocation...	StandardOID	IsNonStandard	HasNoD
CO	CO	Yes	No	CO	CO			Tabulation		One record per comment per subject	Location.CO	STD.SDTMIG-3.4		
DM	DM	Yes	No	DM	DM			Tabulation		One record per subject	Location.DM	STD.SDTMIG-3.4		
SE	SE	Yes	No	SE	SE			Tabulation		One record per actual Element per subj...	Location.SE	STD.SDTMIG-3.4		
SM	SM	Yes	No	SM	SM			Tabulation		One record per Disease Milestone per s...	Location.SM	STD.SDTMIG-3.4		
SV	SV	Yes	No	SV	SV			Tabulation		One record per actual or planned visit pe...	Location.SV	STD.SDTMIG-3.4		
AG	AG	Yes	No	AG	AG			Tabulation		One record per recorded intervention oc...	Location.AG	STD.SDTMIG-3.4		
CM	CM	Yes	No	CM	CM			Tabulation		One record per recorded intervention oc...	Location.CM	STD.SDTMIG-3.4		
EC	EC	Yes	No	EC	EC			Tabulation		One record per protocol-specified study t...	Location.EC	STD.SDTMIG-3.4		
EX	EX	Yes	No	EX	EX			Tabulation		One record per protocol-specified study t...	Location.EX	STD.SDTMIG-3.4		
ML	ML	Yes	No	ML	ML			Tabulation		One record per food product occurrence ...	Location.ML	STD.SDTMIG-3.4		
PR	PR	Yes	No	PR	PR			Tabulation		One record per recorded procedure per ...	Location.PR	STD.SDTMIG-3.4		
SU	SU	Yes	No	SU	SU			Tabulation		One record per substance type per repor...	Location.SU	STD.SDTMIG-3.4		
AE	AE	Yes	No	AE	AE			Tabulation		One record per adverse event per subject	Location.AE	STD.SDTMIG-3.4		
BE	BE	Yes	No	BE	BE			Tabulation		One record per instance per biospecime...	Location.BE	STD.SDTMIG-3.4		
CE	CE	Yes	No	CE	CE			Tabulation		One record per event per subject	Location.CE	STD.SDTMIG-3.4		
DS	DS	Yes	No	DS	DS			Tabulation		One record per disposition status or prot...	Location.DS	STD.SDTMIG-3.4		
DV	DV	Yes	No	DV	DV			Tabulation		One record per protocol deviation per su...	Location.DV	STD.SDTMIG-3.4		
HO	HO	Yes	No	HO	HO			Tabulation		One record per healthcare encounter pe...	Location.HO	STD.SDTMIG-3.4		
MH	MH	Yes	No	MH	MH			Tabulation		One record per medical history event per...	Location.MH	STD.SDTMIG-3.4		
BS	BS	Yes	No	BS	BS			Tabulation		One record per measurement per biosp...	Location.BS	STD.SDTMIG-3.4		
CP	CP	Yes	No	CP	CP			Tabulation		One record per test per specimen per ti...	Location.CP	STD.SDTMIG-3.4		
CV	CV	Yes	No	CV	CV			Tabulation		One record per finding or result per time ...	Location.CV	STD.SDTMIG-3.4		
DA	DA	Yes	No	DA	DA			Tabulation		One record per product accountability fin...	Location.DA	STD.SDTMIG-3.4		
DD	DD	Yes	No	DD	DD			Tabulation		One record per finding per subject	Location.DD	STD.SDTMIG-3.4		
EG	EG	Yes	No	EG	EG			Tabulation		One record per ECG observation per rep...	Location.EG	STD.SDTMIG-3.4		
FT	FT	Yes	No	FT	FT			Tabulation		One record per Functional Test finding p...	Location.FT	STD.SDTMIG-3.4		
GF	GF	Yes	No	GF	GF			Tabulation		One record per finding per observation p...	Location.GF	STD.SDTMIG-3.4		
IE	IE	Yes	No	IE	IE			Tabulation		One record per inclusion/exclusion criter...	Location.IE	STD.SDTMIG-3.4		
IS	IS	Yes	No	IS	IS			Tabulation		One record per test per visit per subject	Location.IS	STD.SDTMIG-3.4		
LB	LB	Yes	No	LB	LB			Tabulation		One record per lab test per time point pe...	Location.LB	STD.SDTMIG-3.4		
MB	MB	Yes	No	MB	MB			Tabulation		One record per microbiology specimen f...	Location.MB	STD.SDTMIG-3.4		
MI	MI	Yes	No	MI	MI			Tabulation		One record per finding per specimen pe...	Location.MI	STD.SDTMIG-3.4		
MK	MK	Yes	No	MK	MK			Tabulation		One record per assessment per visit per...	Location.MK	STD.SDTMIG-3.4		
MS	MS	Yes	No	MS	MS			Tabulation		One record per microbiology susceptibili...	Location.MS	STD.SDTMIG-3.4		
NV	NV	Yes	No	NV	NV			Tabulation		One record per finding per location per ti...	Location.NV	STD.SDTMIG-3.4		
OE	OE	Yes	No	OE	OE			Tabulation		One record per ophthalmic finding per m...	Location.OE	STD.SDTMIG-3.4		
PC	PC	Yes	No	PC	PC			Tabulation		One record per sample characteristic or ...	Location.PC	STD.SDTMIG-3.4		
PE	PE	Yes	No	PE	PE			Tabulation		One record per body system or abnorma...	Location.PE	STD.SDTMIG-3.4		
PP	PP	Yes	No	PP	PP			Tabulation		One record per PK parameter per time-c...	Location.PP	STD.SDTMIG-3.4		
QS	QS	Yes	No	QS	QS			Tabulation		One record per questionnaire per questi...	Location.QS	STD.SDTMIG-3.4		

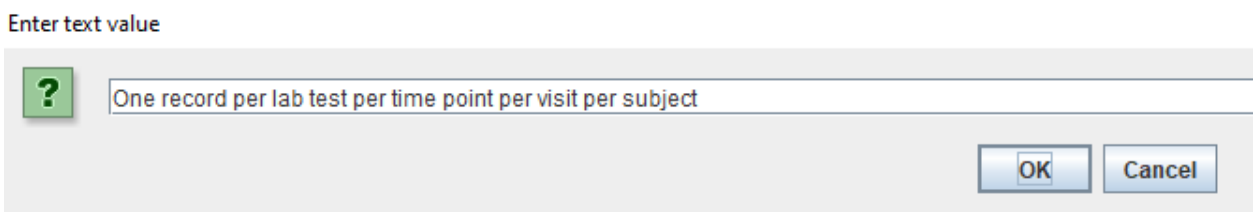
If we already know which domains will really be used, and which not, we can already delete some rows, by selecting a cell in that row, and then use the "Delete selected row" button, which we find in the lower part:



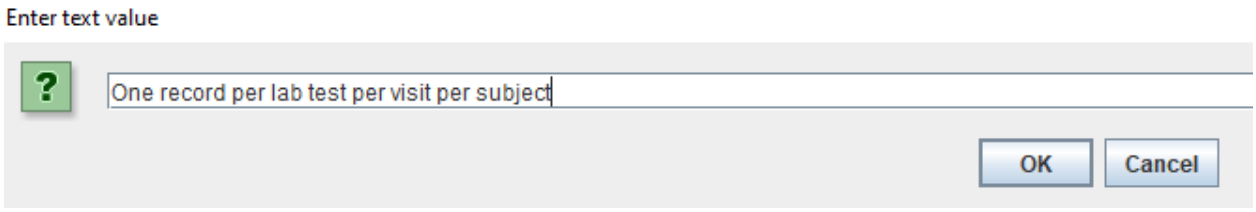
If we delete a row by accident, no panic, we can always revert to an earlier version of our define.xml (see the section "Autosaving", or add a new row, but will then need to add the information like the variables to be used. It is also possible to "merge" define.xml-s. (using the menu "File - Add/Merge define.xml"). Also this will be explained later.

One of the things one always needs to do is to check the column "Structure". Reason is that what is provided by the template is just a first proposal, something that is not always well understood by people who generate the datasets.

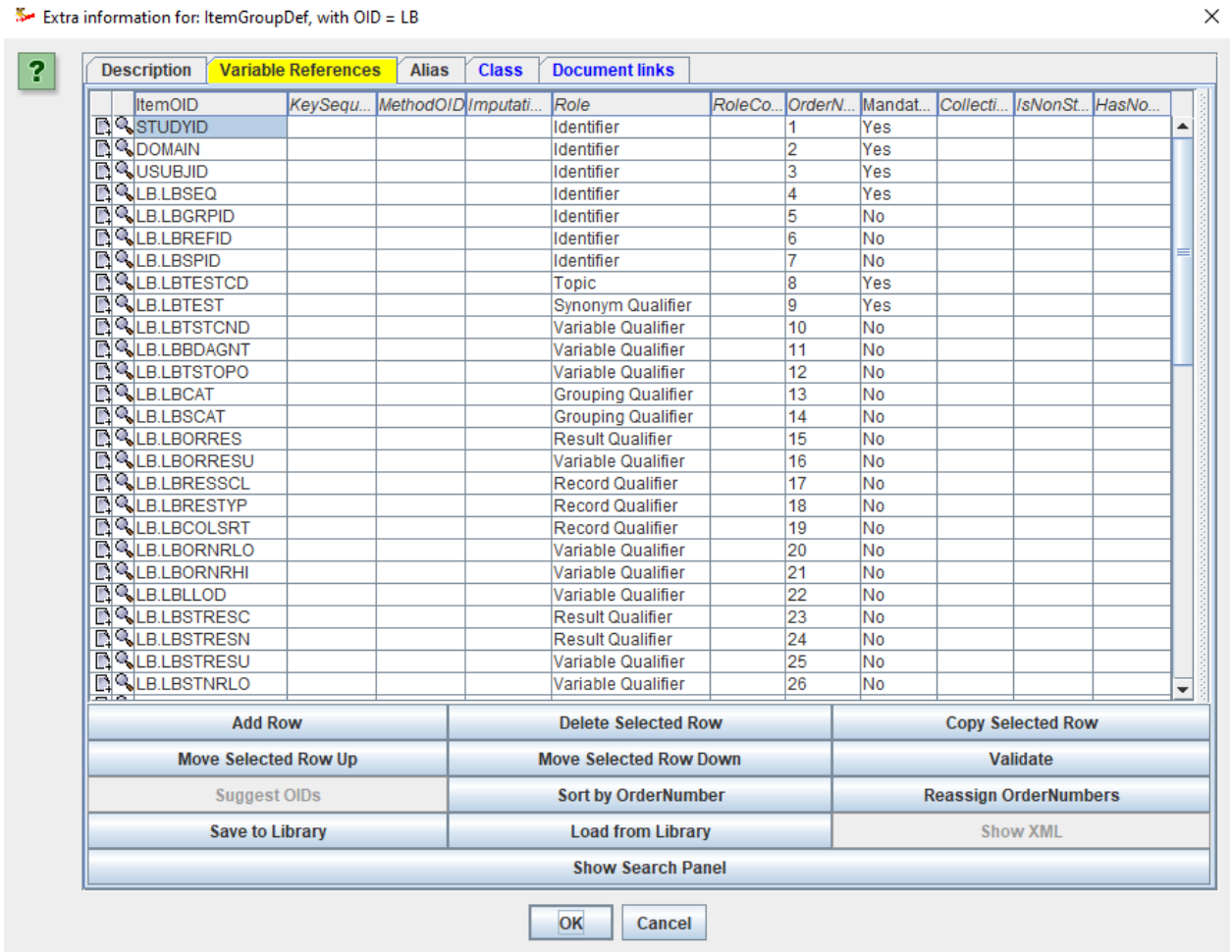
For example, for LB, the suggestion is "One record per lab test per time point per visit per subject", but if we do not have any time points within visit, i.e. that only one measurement per test within a visit is made. By clicking the cell, we can easily change this from:



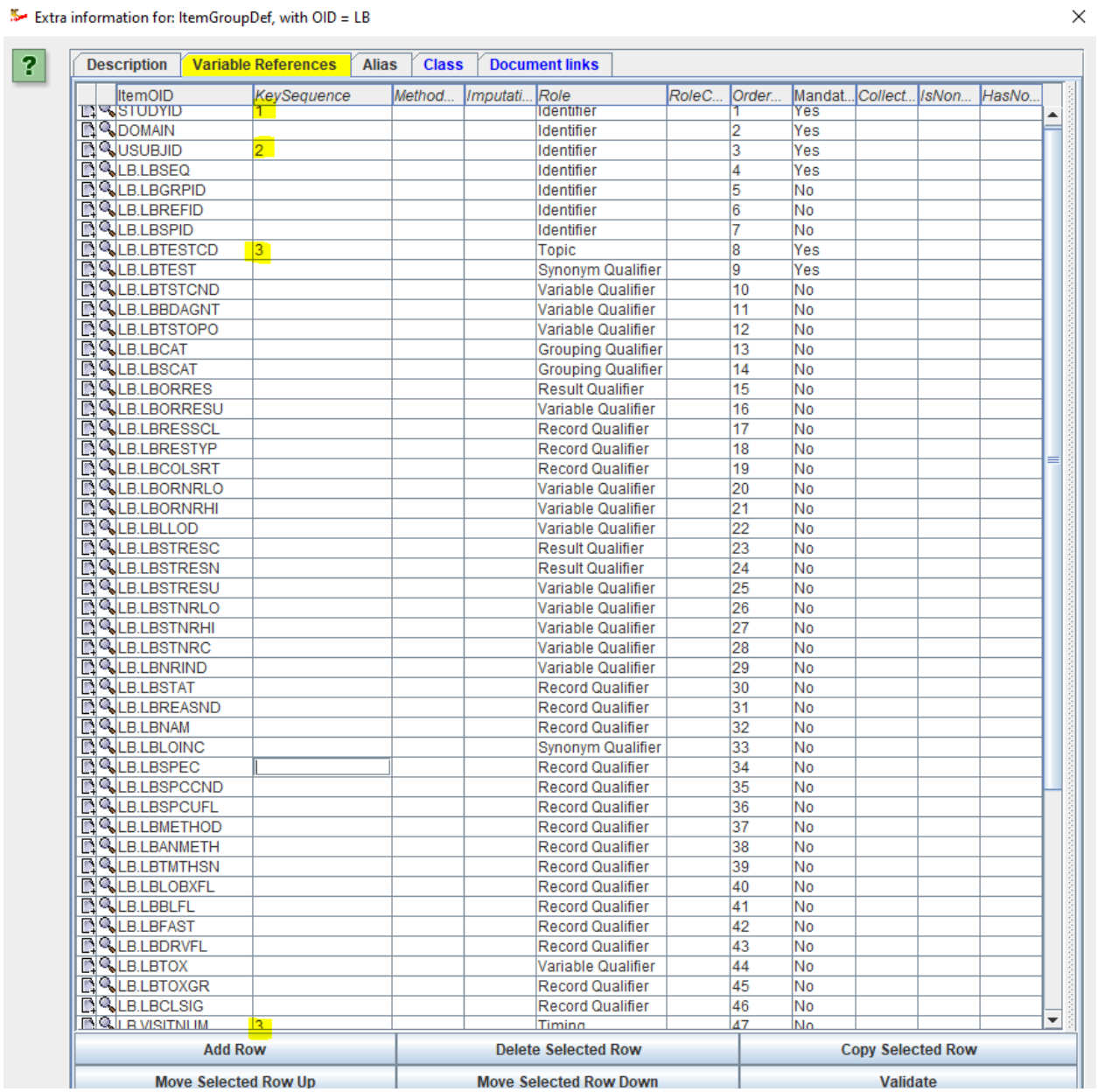
in e.g.:



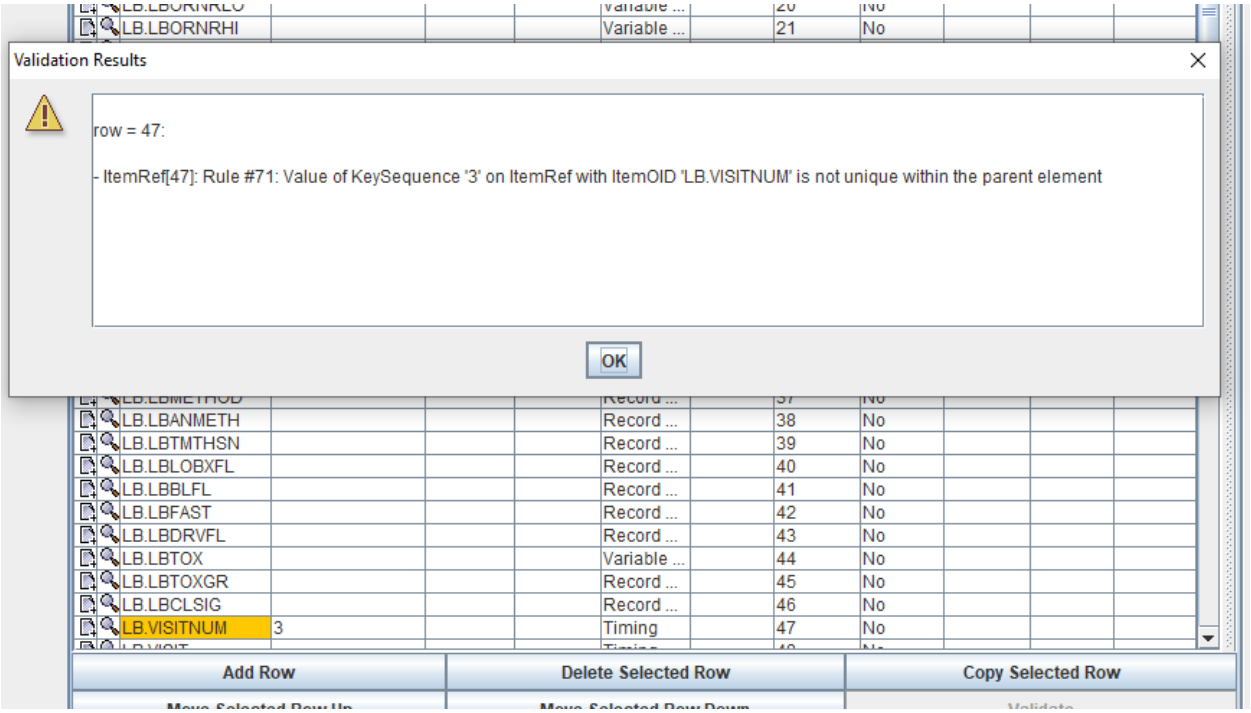
It is important that this is just text to give the reviewer a clue how the data within the dataset is organized, I.e. it doesn't impose anything. The "real structure", in a machine readable format is however provided by the "keys", which are provided at the "ItemRef" level, i.e. the XML elements that define which variables are used within each dataset.
To add this information, we thus need to go into the sub-information, by clicking the "Edit sub-information" icon, and then selecting the "Variable References" tab:



This is also the table where one will add or remove variables for each of the domains.
Let us however first assign the "keys" defining record uniqueness (as in a relational database, though SDTM surely isn't one). This is done in the "KeySequence" column.
As we don't have time points for lab tests in our study, the logical key (sequence) for record uniqueness is:
STUDYID, USUBJID, VISITNUM (or VISIT), LBTESTCD.
"Locally", "STUDYID" is essentially not necessary, as it is fixed, but it is essential anyway for the use case that the metadata of different studies is merged. So, just by clicking cells, and adding an integer number (other characters will be refused anyway), we can e.g. come to:

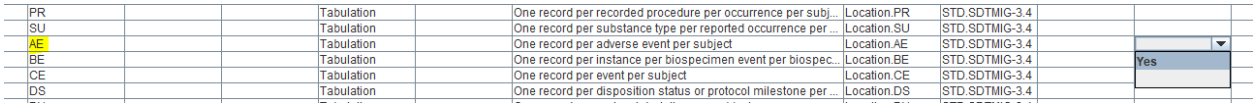


All fine? To check, we can use the "Validate" button (near the bottom, on the right). When it is clicked, we get:

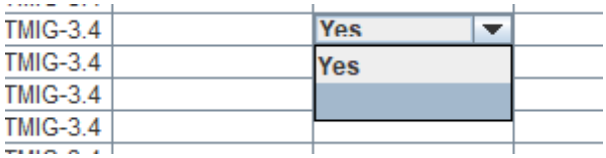


as KeySequence numbers must be unique within the list of ItemRef elements:

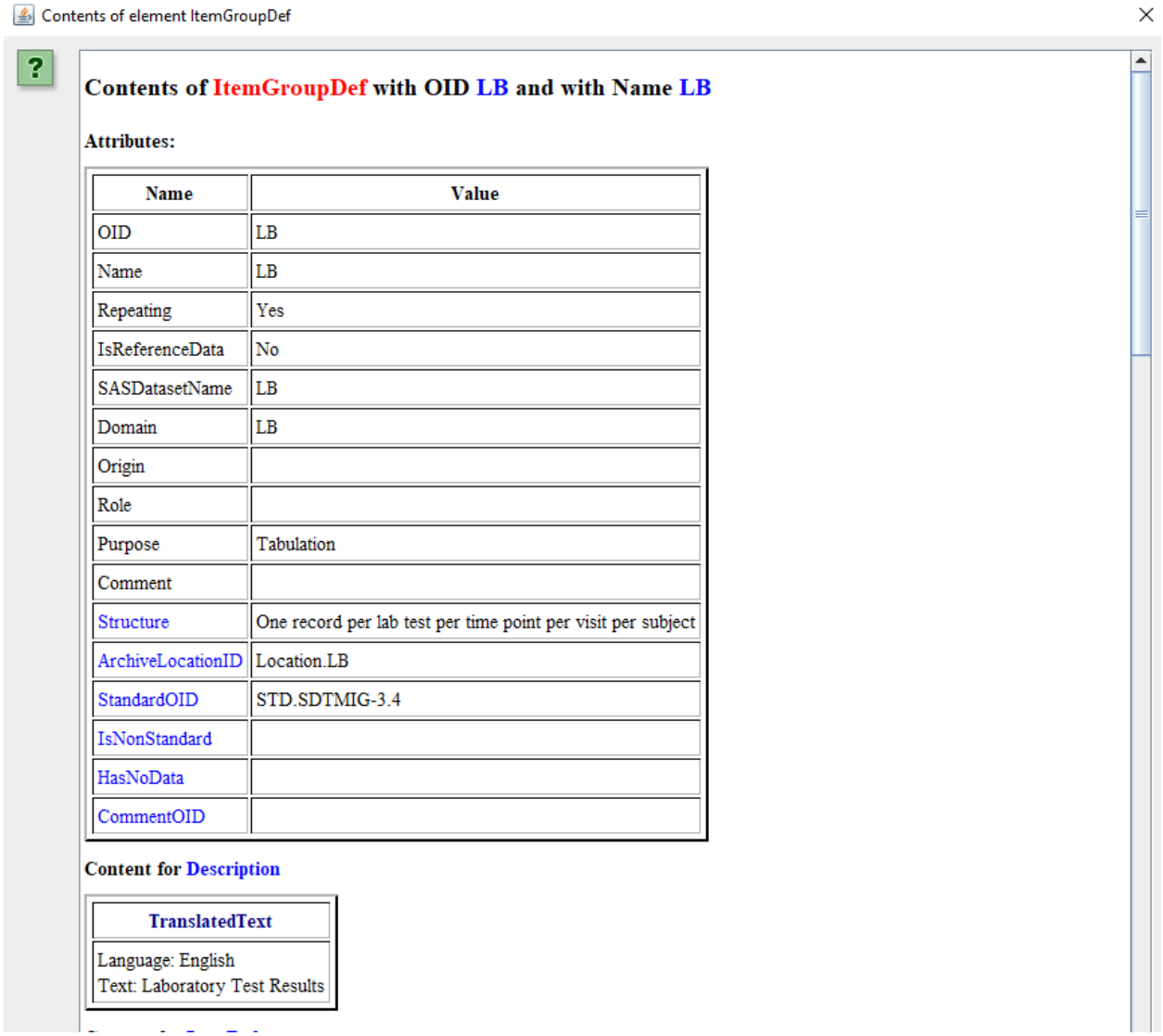
Some other attributes that we can change, but that is often done later in the course of the project, are "IsNonStandard" and "HasNoData". The latter is used to define that an empty dataset is being submitted. For example, after database closure it is found that no a single adverse event has been reported, one can set "HasNoData" to "Yes":



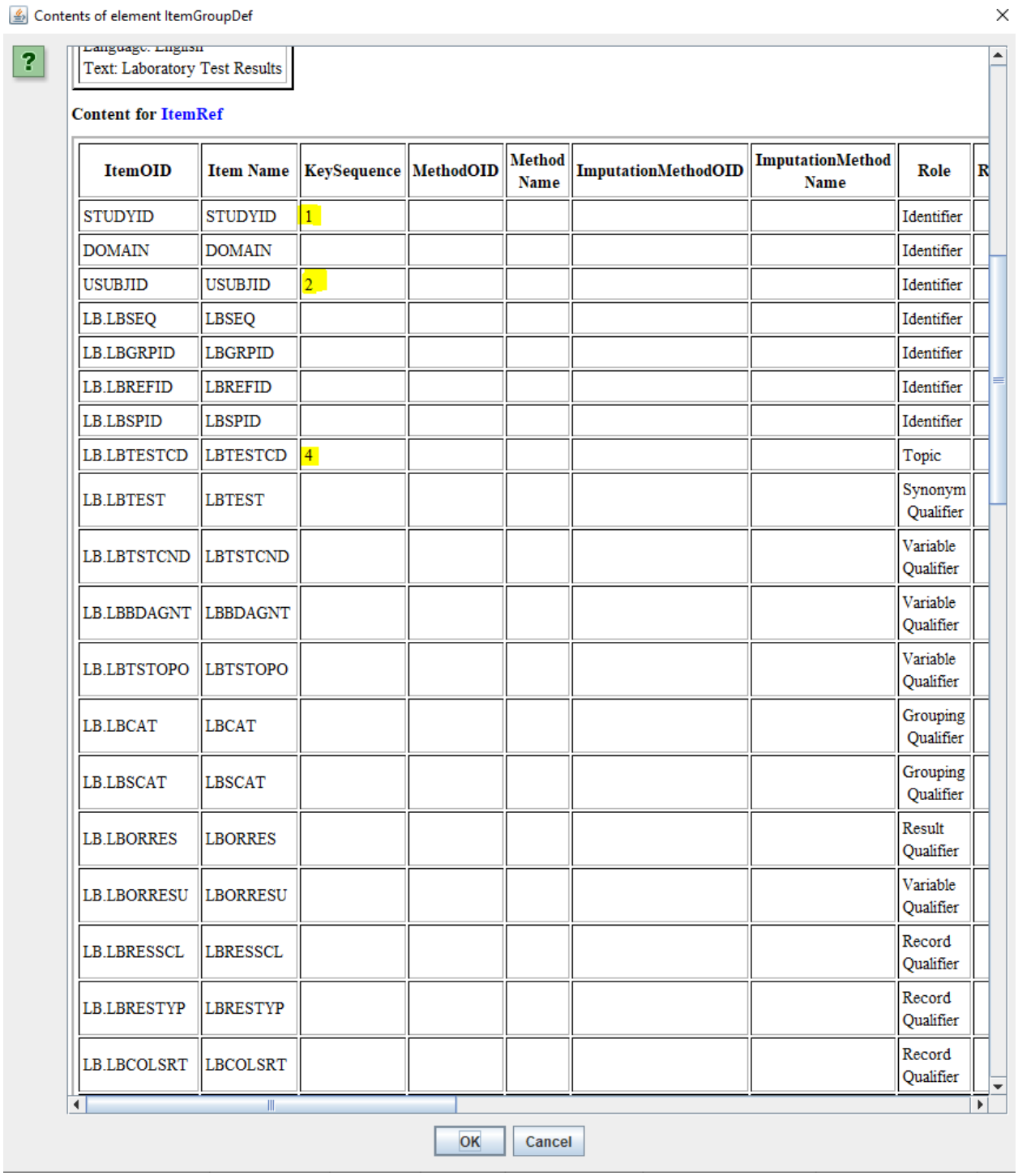
One can always reset it to "null" (i.e. it will be removed in the define.xml) by doing the selection as:



When we now click the "View sub-information" icon for the LB dataset definition, we e.g. get:



and, when scrolling down:



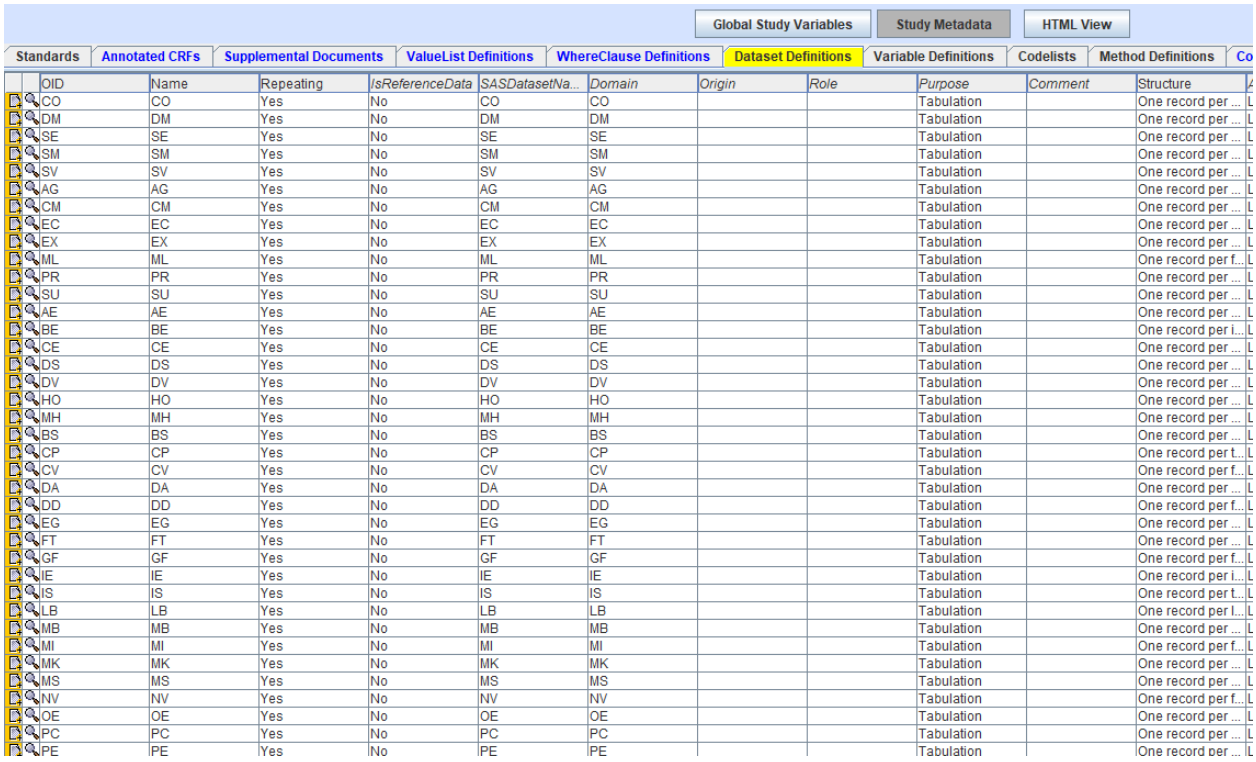
ItemOID	Item Name	KeySequence	MethodOID	Method Name	ImputationMethodOID	ImputationMethod Name	Role	R
STUDYID	STUDYID	1					Identifier	
DOMAIN	DOMAIN						Identifier	
USUBJID	USUBJID	2					Identifier	
LB.LBSEQ	LBSEQ						Identifier	
LB.LBGRPID	LBGRPID						Identifier	
LB.LBREFID	LBREFID						Identifier	
LB.LBSPID	LBSPID						Identifier	
LB.LBTESTCD	LBTESTCD	4					Topic	
LB.LBTEST	LBTEST						Synonym Qualifier	
LB.LBTSTCND	LBTSTCND						Variable Qualifier	
LB.LBBDAGNT	LBBDAGNT						Variable Qualifier	
LB.LBTSTOPO	LBTSTOPO						Variable Qualifier	
LB.LBCAT	LBCAT						Grouping Qualifier	
LB.LBSCAT	LBSCAT						Grouping Qualifier	
LB.LBORRES	LBORRES						Result Qualifier	
LB.LBORRESU	LBORRESU						Variable Qualifier	
LB.LBRESSCL	LBRESSCL						Record Qualifier	
LB.LBRESTYP	LBRESTYP						Record Qualifier	
LB.LBCOLSRT	LBCOLSRT						Record Qualifier	

Adding and removing dataset definitions

Adding and removing dataset definitions will usually not be done when starting from a set of SAS-XPT datasets.
It is however very important when using the define.xml as the definition of the deliverables of the submission, whether it is about SDTM, SEND or ADaM. For example, the sponsor can set up a define.xml for an external vendor (or one department for another department within the sponsor) as a specification of which datasets need to be generated and what variables each dataset contain. This can then even be a "partial" define.xml, as one will e.g. not know the maximal length of each variable in advance.
Whereas for SDTM and SEND, the names of the datasets and their content is strongly defined by the SDTM and SEND Implementation Guides (IGs), this is much less the case for ADaM, where this will mostly defined by what is in the Statistical Analysis Plan (SAP), instead there are a lot of "naming conventions" for the datasets and variables in ADaM.

We will here show how dataset definitions can be added and removed for the case of SDTM, starting from the template. The same principles however also apply to SEND and ADaM.

We select the tab "Dataset Definitions":



OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role	Purpose	Comment	Structure
CO	CO	Yes	No	CO	CO			Tabulation		One record per ...
DM	DM	Yes	No	DM	DM			Tabulation		One record per ...
SE	SE	Yes	No	SE	SE			Tabulation		One record per ...
SM	SM	Yes	No	SM	SM			Tabulation		One record per ...
SV	SV	Yes	No	SV	SV			Tabulation		One record per ...
AG	AG	Yes	No	AG	AG			Tabulation		One record per ...
CM	CM	Yes	No	CM	CM			Tabulation		One record per ...
EC	EC	Yes	No	EC	EC			Tabulation		One record per ...
EX	EX	Yes	No	EX	EX			Tabulation		One record per ...
ML	ML	Yes	No	ML	ML			Tabulation		One record per f...
PR	PR	Yes	No	PR	PR			Tabulation		One record per ...
SU	SU	Yes	No	SU	SU			Tabulation		One record per ...
AE	AE	Yes	No	AE	AE			Tabulation		One record per ...
BE	BE	Yes	No	BE	BE			Tabulation		One record per i...
CE	CE	Yes	No	CE	CE			Tabulation		One record per ...
DS	DS	Yes	No	DS	DS			Tabulation		One record per ...
DV	DV	Yes	No	DV	DV			Tabulation		One record per ...
HO	HO	Yes	No	HO	HO			Tabulation		One record per ...
MH	MH	Yes	No	MH	MH			Tabulation		One record per ...
BS	BS	Yes	No	BS	BS			Tabulation		One record per ...
CP	CP	Yes	No	CP	CP			Tabulation		One record per t...
CV	CV	Yes	No	CV	CV			Tabulation		One record per f...
DA	DA	Yes	No	DA	DA			Tabulation		One record per ...
DD	DD	Yes	No	DD	DD			Tabulation		One record per f...
EG	EG	Yes	No	EG	EG			Tabulation		One record per ...
FT	FT	Yes	No	FT	FT			Tabulation		One record per ...
GF	GF	Yes	No	GF	GF			Tabulation		One record per f...
IE	IE	Yes	No	IE	IE			Tabulation		One record per i...
IS	IS	Yes	No	IS	IS			Tabulation		One record per t...
LB	LB	Yes	No	LB	LB			Tabulation		One record per l...
MB	MB	Yes	No	MB	MB			Tabulation		One record per ...
MI	MI	Yes	No	MI	MI			Tabulation		One record per f...
MK	MK	Yes	No	MK	MK			Tabulation		One record per ...
MS	MS	Yes	No	MS	MS			Tabulation		One record per ...
NV	NV	Yes	No	NV	NV			Tabulation		One record per f...
OE	OE	Yes	No	OE	OE			Tabulation		One record per ...
PC	PC	Yes	No	PC	PC			Tabulation		One record per ...
PE	PE	Yes	No	PE	PE			Tabulation		One record per ...

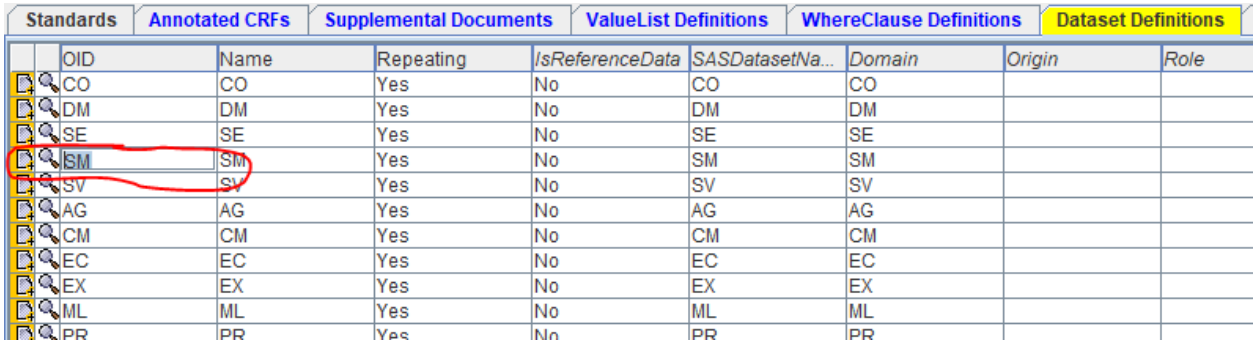
From our study design (and possible the (a)CRFs) we deduce that we will e.g. not need the following domains:

SM (Subject Disease Milestones), AG (Procedure Agents), ML (Meal Data), BE (Biospecimen Events), CP (Cell Phenotype Findings), CV (Cardiovascular System Findings), DA (Product Accountability), FT (Functional Tests), GF (Genomic Findings), IS (Immunogenicity Specimen Assessments), MB (Microbiology Specimen), MI (Microbiology Findings), MK (Musculoskeletal System Findings), MS (Microbiology Susceptibility), NV (Nervous System Findings), OE (Ophthalmic Examinations), RE (Respiratory System Findings), RP (Reproductive System Findings), SS (Subject Status), RS (Disease Response and Clin Classification), TR (Tumor/Lesion Results) and TU (Tumor/Lesion Identification) as this is not a cancer study. Furthermore, we do not need UR (Urinary System Findings), SR (Skin Response). We also keep all "Trial Design" dataset definitions for now except for TD (Trial Disease Assessments), TM (Trial Disease Milestones), OI (Non-host Organism Identifiers) which we will remove.

We keep DD (Death Details) as we cannot know in advance whether our study may have subjects dying during the study period. If none, we can always still remove it later. Similar applies to IE (Inclusion/Exclusion Criteria Not Met). For QS (Questionnaires) we will need to "split" as we have several questionnaires, and it is custom to have one QSxx dataset per type of questionnaire. For FA (Findings About), we just keep it for now, we may want to have several instances later, like "FAMH" (Findings About Medical History), "FAAE" (Findings About Adverse Events). Normally this should however be clear from the (annotated) CRF.

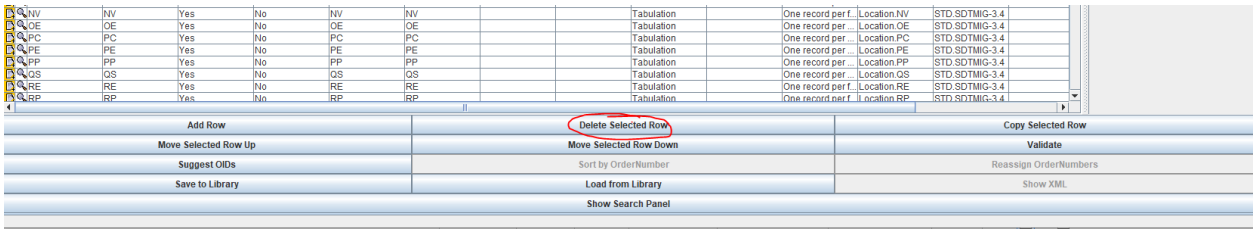
Deciding for which domains we will have dataset definitions in our define.xml is of course a crucial step, based on information from the aCRF (when already available) and/or the protocol. In many cases, the user's company will maintain libraries for this, which we can import into our define.xml (see section "Using own Libraries").
If one deletes a dataset definition by accident - no panic, we will later see how one can either return to a prior state of the development of the define.xml, or to merge with already existing define.xml-s.

To remove a dataset definition from the define.xml, select a cell of the dataset definition to be removed, e.g. SM (one from the list of dataset definitions we want to remove):

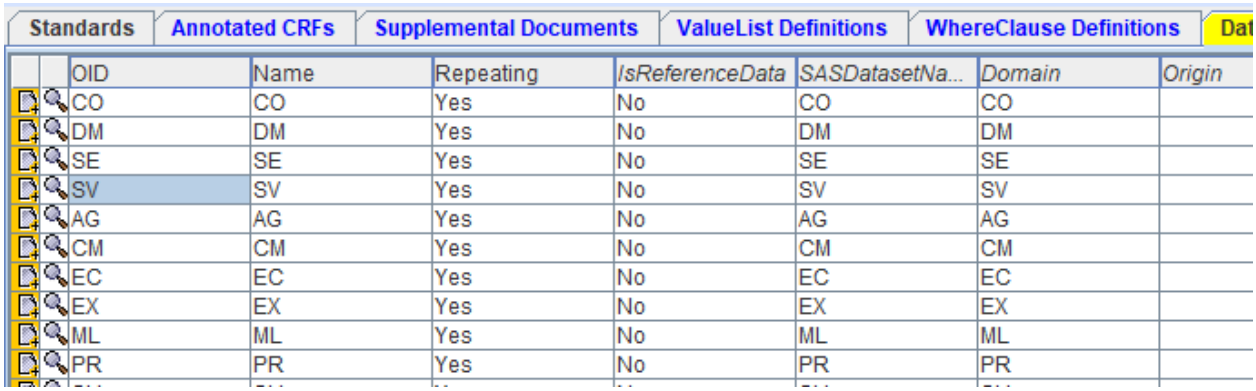


	OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role
	CO	CO	Yes	No	CO	CO		
	DM	DM	Yes	No	DM	DM		
	SE	SE	Yes	No	SE	SE		
	SM	SM	Yes	No	SM	SM		
	SV	SV	Yes	No	SV	SV		
	AG	AG	Yes	No	AG	AG		
	CM	CM	Yes	No	CM	CM		
	EC	EC	Yes	No	EC	EC		
	EX	EX	Yes	No	EX	EX		
	ML	ML	Yes	No	ML	ML		
	PR	PR	Yes	No	PR	PR		

and the click the "Delete Selected Row" button which is near the bottom of the window:

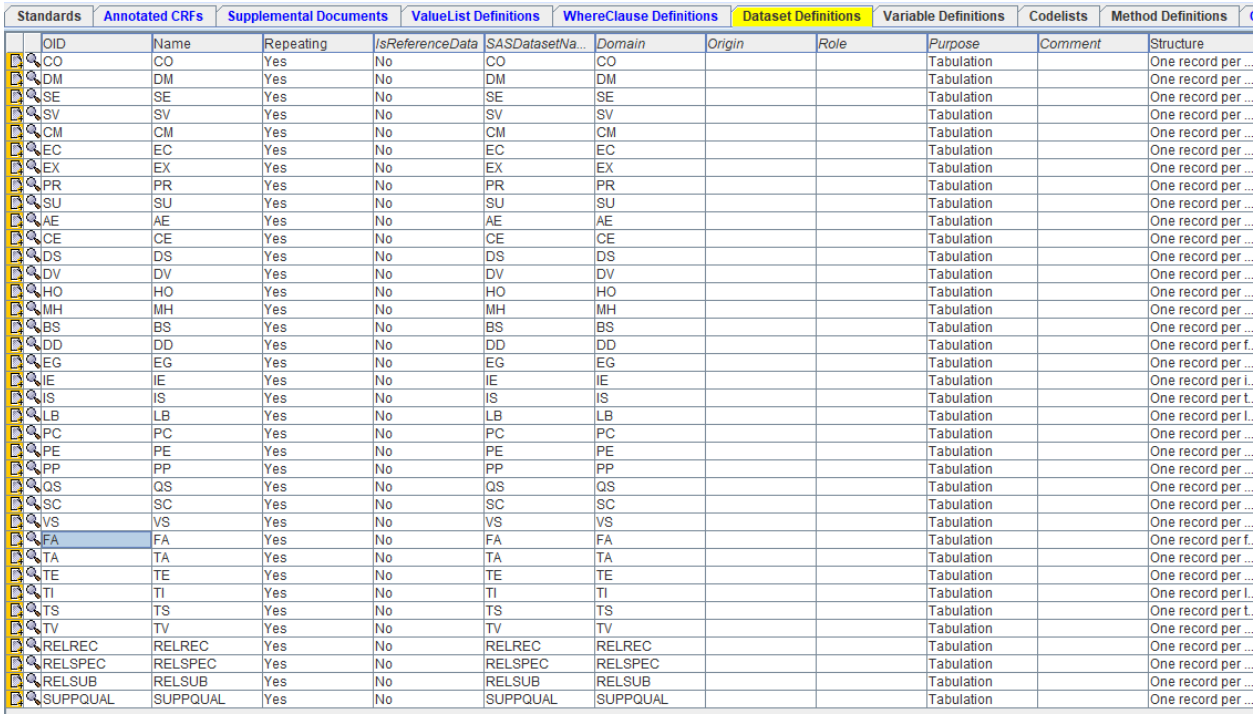


resulting in the "SM" row being removed:



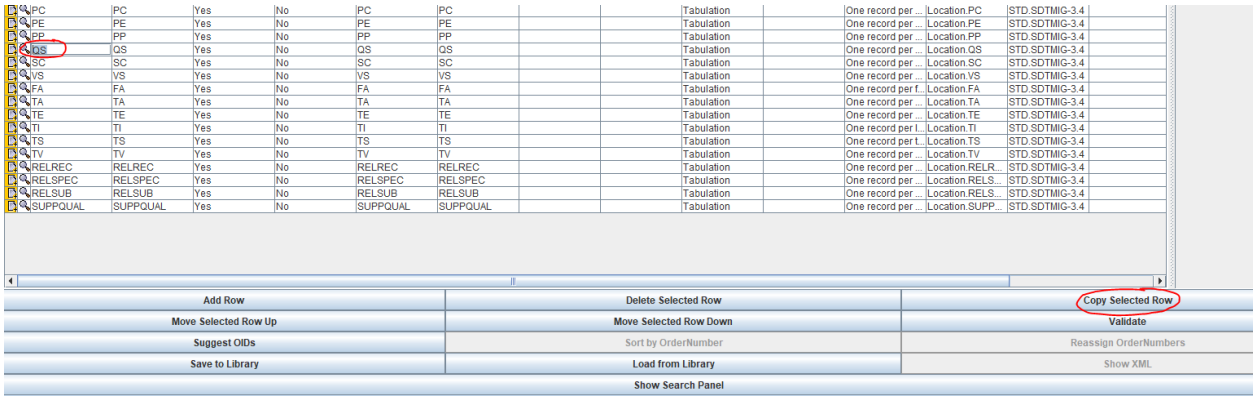
	OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin
	CO	CO	Yes	No	CO	CO	
	DM	DM	Yes	No	DM	DM	
	SE	SE	Yes	No	SE	SE	
	SV	SV	Yes	No	SV	SV	
	AG	AG	Yes	No	AG	AG	
	CM	CM	Yes	No	CM	CM	
	EC	EC	Yes	No	EC	EC	
	EX	EX	Yes	No	EX	EX	
	ML	ML	Yes	No	ML	ML	
	PR	PR	Yes	No	PR	PR	

One can now repeat the process for all other dataset definitions to be removed. In our case, this leads to:

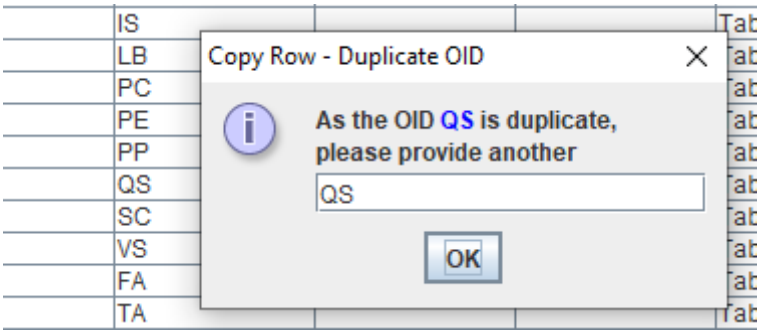


	OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role	Purpose	Comment	Structure
	CO	CO	Yes	No	CO	CO			Tabulation		One record per ...
	DM	DM	Yes	No	DM	DM			Tabulation		One record per ...
	SE	SE	Yes	No	SE	SE			Tabulation		One record per ...
	SV	SV	Yes	No	SV	SV			Tabulation		One record per ...
	CM	CM	Yes	No	CM	CM			Tabulation		One record per ...
	EC	EC	Yes	No	EC	EC			Tabulation		One record per ...
	EX	EX	Yes	No	EX	EX			Tabulation		One record per ...
	PR	PR	Yes	No	PR	PR			Tabulation		One record per ...
	SU	SU	Yes	No	SU	SU			Tabulation		One record per ...
	AE	AE	Yes	No	AE	AE			Tabulation		One record per ...
	CE	CE	Yes	No	CE	CE			Tabulation		One record per ...
	DS	DS	Yes	No	DS	DS			Tabulation		One record per ...
	DV	DV	Yes	No	DV	DV			Tabulation		One record per ...
	HO	HO	Yes	No	HO	HO			Tabulation		One record per ...
	MH	MH	Yes	No	MH	MH			Tabulation		One record per ...
	BS	BS	Yes	No	BS	BS			Tabulation		One record per ...
	DD	DD	Yes	No	DD	DD			Tabulation		One record per f...
	EG	EG	Yes	No	EG	EG			Tabulation		One record per ...
	IE	IE	Yes	No	IE	IE			Tabulation		One record per i...
	IS	IS	Yes	No	IS	IS			Tabulation		One record per t...
	LB	LB	Yes	No	LB	LB			Tabulation		One record per l...
	PC	PC	Yes	No	PC	PC			Tabulation		One record per ...
	PE	PE	Yes	No	PE	PE			Tabulation		One record per ...
	PP	PP	Yes	No	PP	PP			Tabulation		One record per ...
	QS	QS	Yes	No	QS	QS			Tabulation		One record per ...
	SC	SC	Yes	No	SC	SC			Tabulation		One record per ...
	VS	VS	Yes	No	VS	VS			Tabulation		One record per ...
	FA	FA	Yes	No	FA	FA			Tabulation		One record per f...
	TA	TA	Yes	No	TA	TA			Tabulation		One record per ...
	TE	TE	Yes	No	TE	TE			Tabulation		One record per ...
	TI	TI	Yes	No	TI	TI			Tabulation		One record per l...
	TS	TS	Yes	No	TS	TS			Tabulation		One record per t...
	TV	TV	Yes	No	TV	TV			Tabulation		One record per ...
	RELREC	RELREC	Yes	No	RELREC	RELREC			Tabulation		One record per ...
	RELSPEC	RELSPEC	Yes	No	RELSPEC	RELSPEC			Tabulation		One record per ...
	RELSUB	RELSUB	Yes	No	RELSUB	RELSUB			Tabulation		One record per ...
	SUPPQUAL	SUPPQUAL	Yes	No	SUPPQUAL	SUPPQUAL			Tabulation		One record per ...

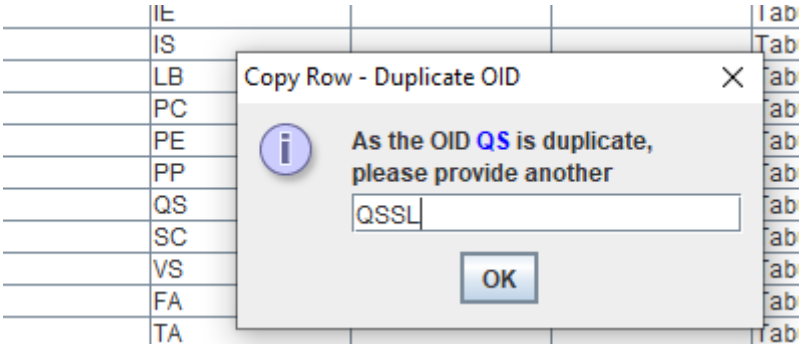
For QS, we will "split" into 2 dataset definitions, as we have two questionnaires, e.g. QSPH (PATIENT HEALTH QUESTIONNAIRE-9 - PHQ-9) and QSSL (SATISFACTION WITH LIFE SURVEY - SWLS).
In order to do so, select a cell in the QS row, and click the "Copy Selected Row" button:



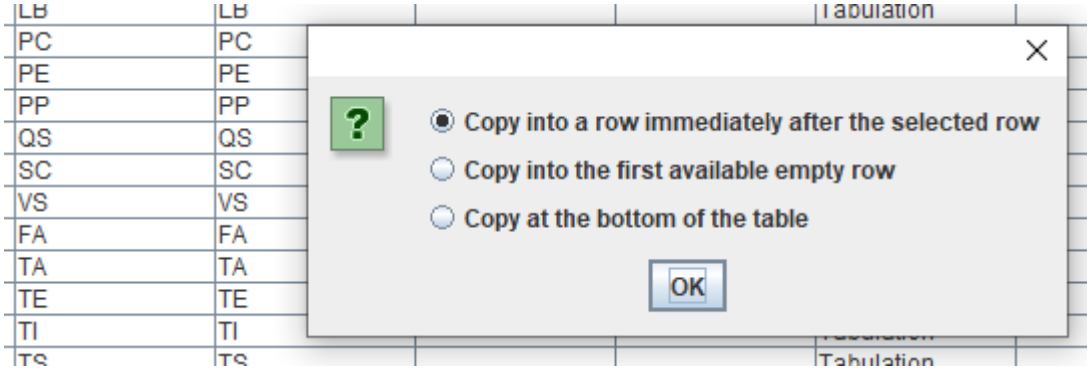
A question dialog is displayed:



which we fill with:



which is followed by another dialog:



In most cases one will select the first option …
Clicking "OK" leads to:

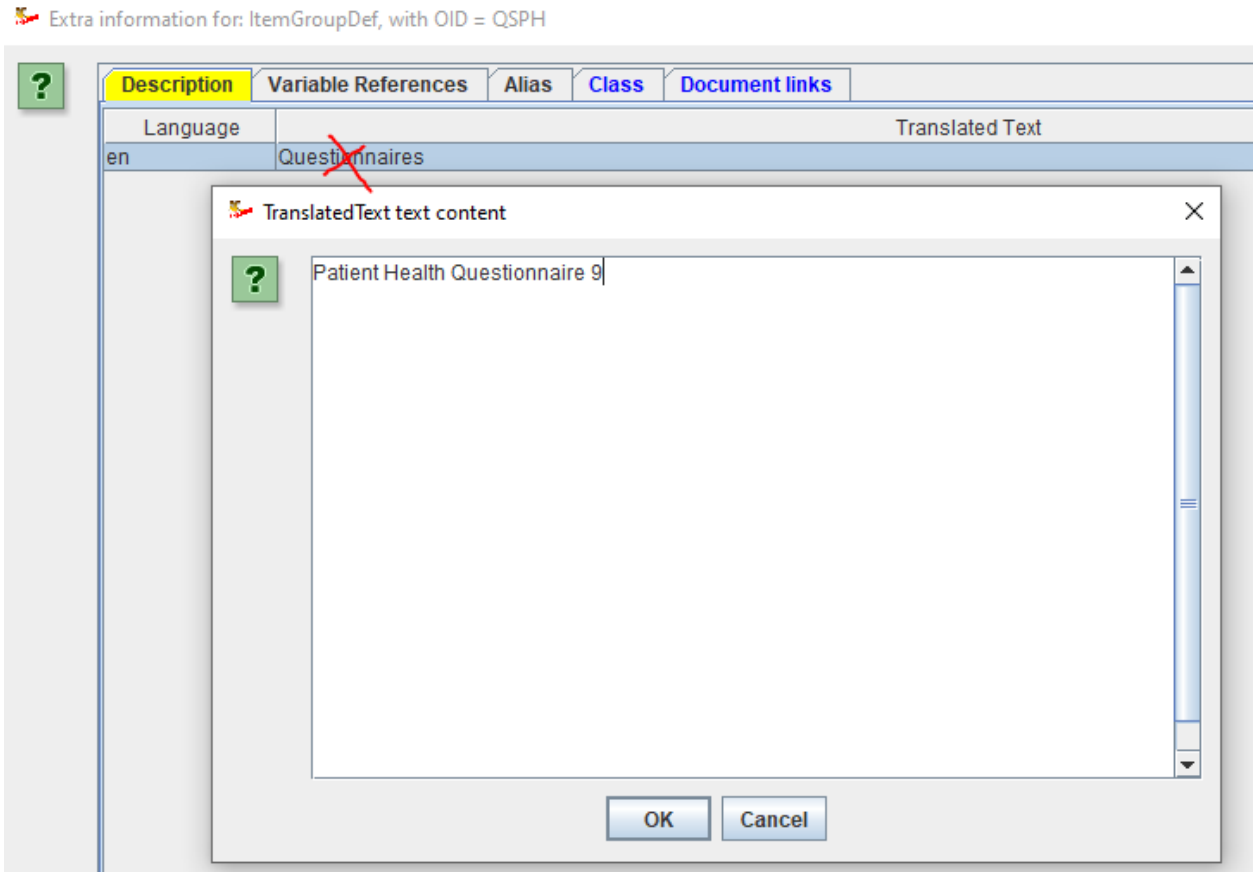
LB	LB	Yes	No	LB
PC	PC	Yes	No	PC
PE	PE	Yes	No	PE
PP	PP	Yes	No	PP
QS	QS	Yes	No	QS
QSSL	QS	Yes	No	QS
SC	SC	Yes	No	SC
VS	VS	Yes	No	VS
FA	FA	Yes	No	FA

and which we can now start editing: we want as well different OIDs (first column) as well as separate dataset names (second column) as also for the "SASDatasetName". Just clicking in the cell and editing e.g. leads then to:

Standards | Annotated CRFs | Supplemental Documents | ValueList Definitions | WhereClause Definitions | Dataset Definitions | Variable Definitions

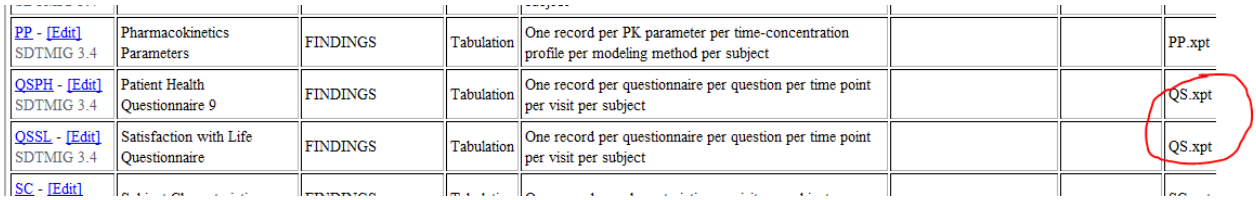
OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role	Purpose
CO	CO	Yes	No	CO	CO			Tabulation
DM	DM	Yes	No	DM	DM			Tabulation
SE	SE	Yes	No	SE	SE			Tabulation
SV	SV	Yes	No	SV	SV			Tabulation
CM	CM	Yes	No	CM	CM			Tabulation
EC	EC	Yes	No	EC	EC			Tabulation
EX	EX	Yes	No	EX	EX			Tabulation
PR	PR	Yes	No	PR	PR			Tabulation
SU	SU	Yes	No	SU	SU			Tabulation
AE	AE	Yes	No	AE	AE			Tabulation
CE	CE	Yes	No	CE	CE			Tabulation
DS	DS	Yes	No	DS	DS			Tabulation
DV	DV	Yes	No	DV	DV			Tabulation
HO	HO	Yes	No	HO	HO			Tabulation
MH	MH	Yes	No	MH	MH			Tabulation
BS	BS	Yes	No	BS	BS			Tabulation
DD	DD	Yes	No	DD	DD			Tabulation
EG	EG	Yes	No	EG	EG			Tabulation
IE	IE	Yes	No	IE	IE			Tabulation
IS	IS	Yes	No	IS	IS			Tabulation
LB	LB	Yes	No	LB	LB			Tabulation
PC	PC	Yes	No	PC	PC			Tabulation
PE	PE	Yes	No	PE	PE			Tabulation
PP	PP	Yes	No	PP	PP			Tabulation
QSPH	QSPH	Yes	No	QSPH	QS			Tabulation
QSSL	QSSL	Yes	No	QSSL	QS			Tabulation
SC	SC	Yes	No	SC	SC			Tabulation
VS	VS	Yes	No	VS	VS			Tabulation
FA	FA	Yes	No	FA	FA			Tabulation

Important remark: We should not change the value for "Domain", as both dataset definitions still share the same Domain name. We still must adapt the "Label", which in the define.xml is covered by the "Description" element. For QSPH, by clicking on the "Edit sub-information", we change the existing value of "Questionnaires" into e.g.:



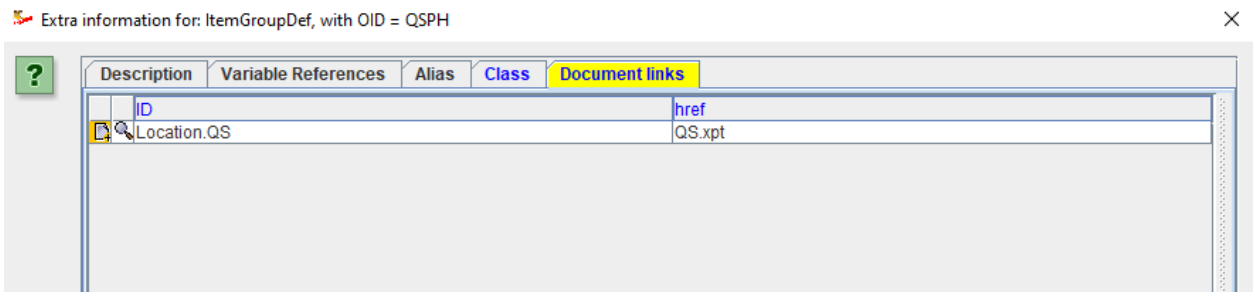
and similar for QSSL into "Satisfaction with Life Questionnaire".
Remark that due to the current restrictions of SAS-XPT, the "label" may not be more than 40 characters. This rule will probably be relaxed in future when datasets in modern CDISC Dataset-JSON format becomes become accepted by the regulatory authorities (the sooner the better …).

When going back to the main window and clicking the "HTML View" button, and navigating to the questionnaires dataset definitions, we find:

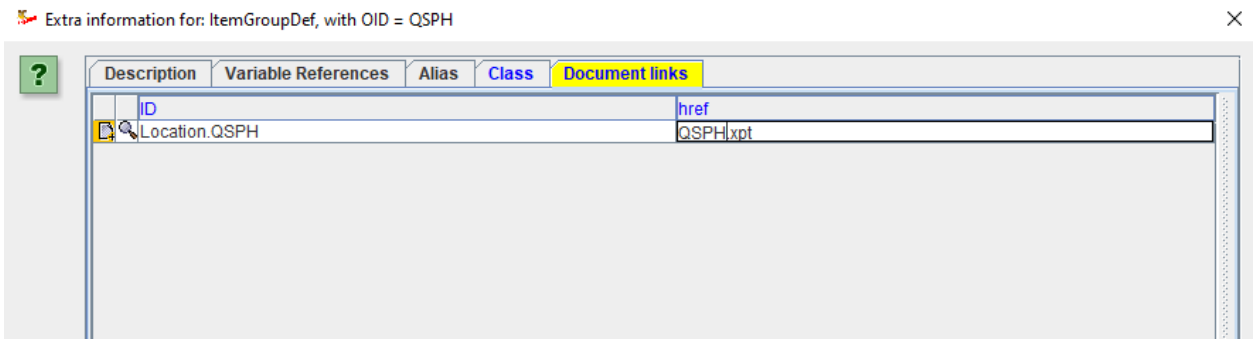


PP - [Edit] SDTMIG 3.4	Pharmacokinetics Parameters	FINDINGS	Tabulation	One record per PK parameter per time-concentration profile per modeling method per subject			PP.xpt
QSPH - [Edit] SDTMIG 3.4	Patient Health Questionnaire 9	FINDINGS	Tabulation	One record per questionnaire per question per time point per visit per subject			QS.xpt
QSSL - [Edit] SDTMIG 3.4	Satisfaction with Life Questionnaire	FINDINGS	Tabulation	One record per questionnaire per question per time point per visit per subject			QS.xpt
SC - [Edit]							

where we see that this will need further refinement for the "Structure" (which may be different between both) and especially the "Location" showing the XPT file name. The latter can easily be changed by clicking the "Edit sub-information" icon, navigating to the "Document links" tab:



and editing the information into:

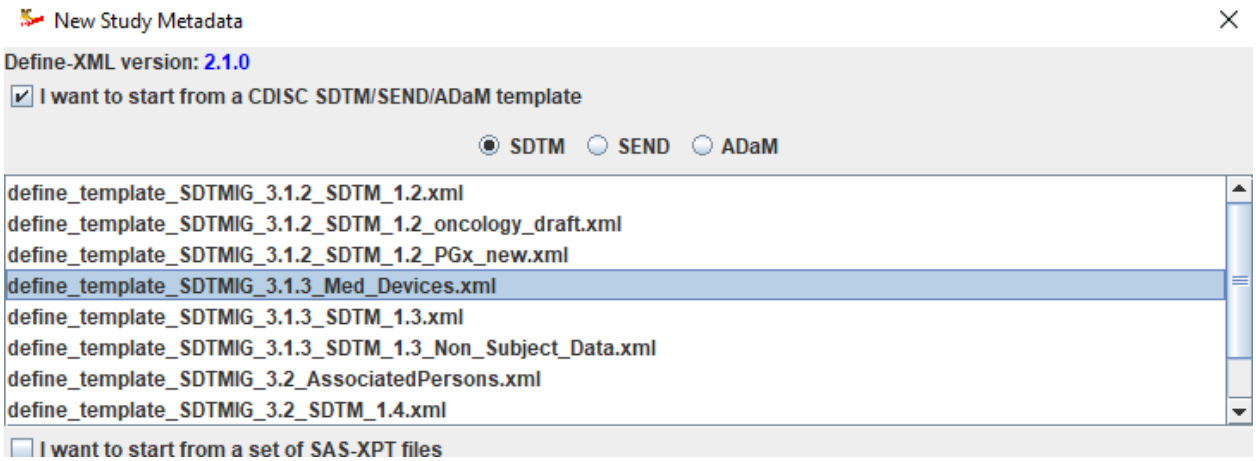


and similar for QSSL …
Updating the "HTML View" then e.g. leads to:

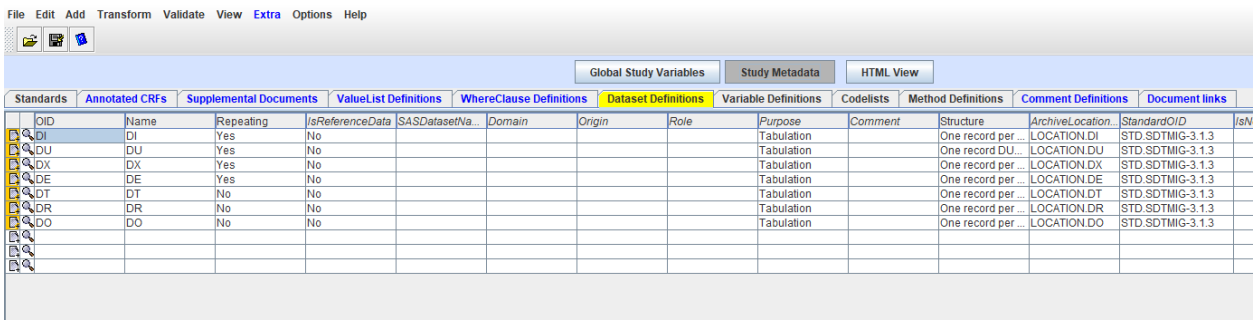
SDTMIG 3.4	Parameters	FINDINGS	Tabulation	profile per modeling method per subject			PP.xpt
QSPH - [Edit] SDTMIG 3.4	Patient Health Questionnaire 9	FINDINGS	Tabulation	One record per questionnaire per question per time point per visit per subject			QSPH.xpt
QSSL - [Edit] SDTMIG 3.4	Satisfaction with Life Questionnaire	FINDINGS	Tabulation	One record per questionnaire per question per time point per visit per subject			QSSL.xpt
SC - [Edit]	Subject Characteristics	FINDINGS	Tabulation	One record per characteristic per visit per subject.			SC.xpt

Adding dataset definitions (domains) from another template

In some cases, one wants to also include dataset definitions (i.e. "domains") from other versions of the standard, such as the "Medical Devices" (MD) standard for which there is a separate template available. Let us suppose the user wants to first set up the definitions for the MD domains, and after that, add definitions from SDTMIG-3.4 (remark that the other way around is of course also possible). So, when starting, the user selects:

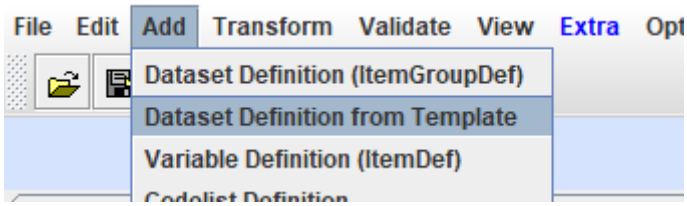


Not a bad idea to also already load a version of the CDISC Controlled Terminology.
This leads to:

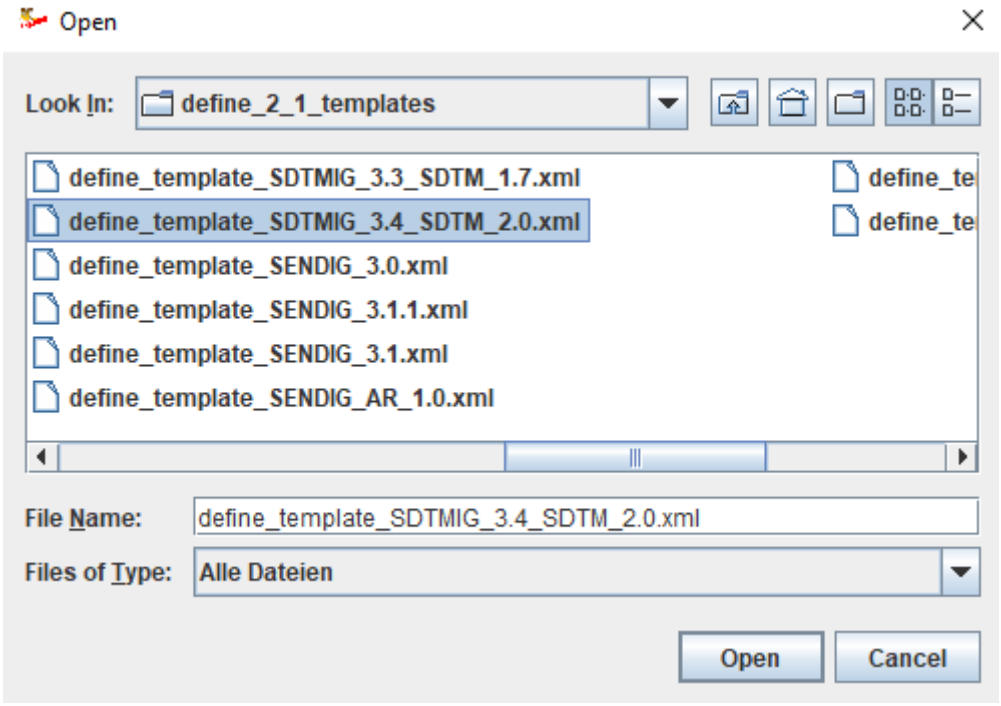


i.e. 7 definitions for DI (Device Identifiers), DU (Device In-Use), DX (Device Exposure), DE (Device Events), DT (Device Tracking and Disposition), DR (Device-Subject Relationships) and DO (Device Properties), including the variable definitions under the "Variable Definitions" tab.

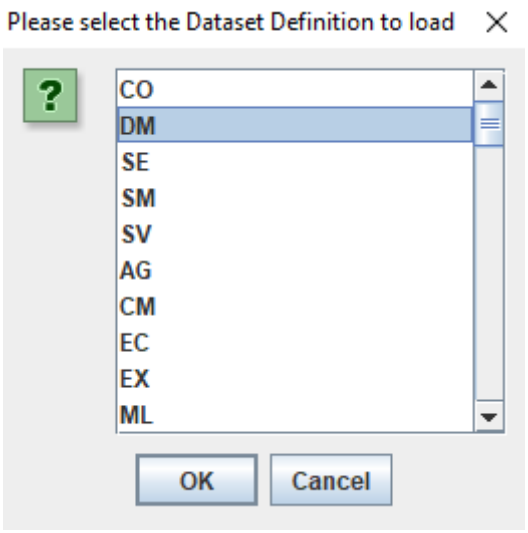
After having worked on these definitions, the user then wants e.g. to add the DM (Demographics) domain from version 3.4 of the SDTM-IG. To do so, use the menu "Add - Dataset Definition from Template":



which then displays a file chooser from which the template for the chosen standard version can be selected[6], e.g.:



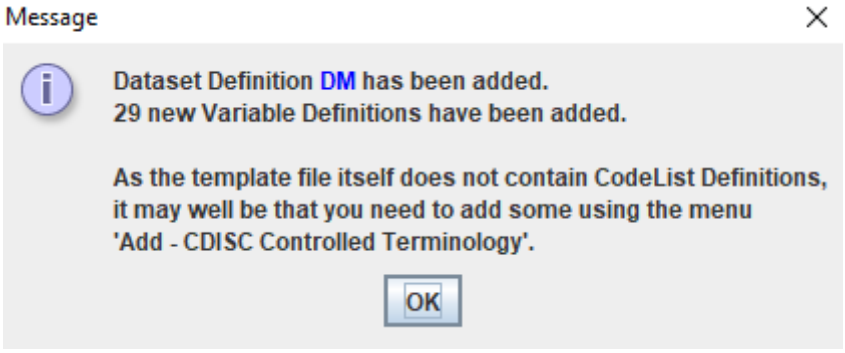
When then clicking "Open", the user is invited to select the domain from the selected template he/she wants to have the dataset definition added. For example:



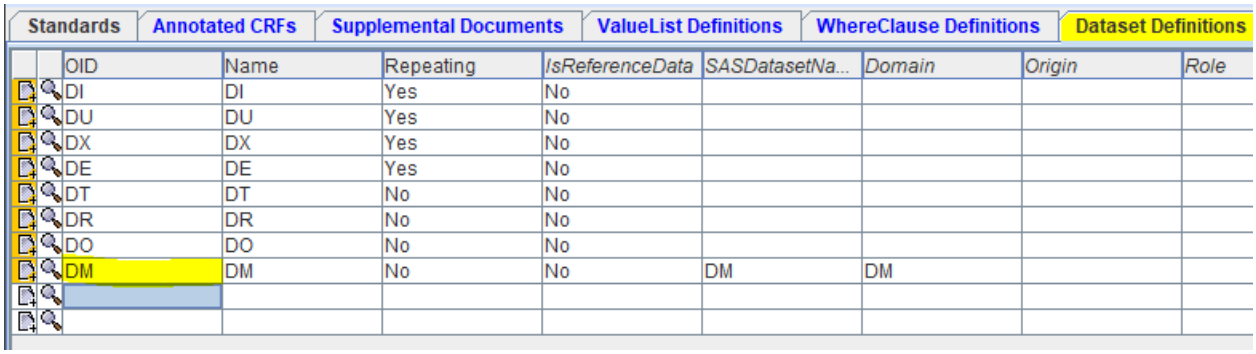
Remark that at this moment, only one dataset definition can be loaded at the time.
After selecting the wanted one, and clicking "OK", this leads to a message:

---

[6] The file chooser automatically opens in the folder where all templates are stored.

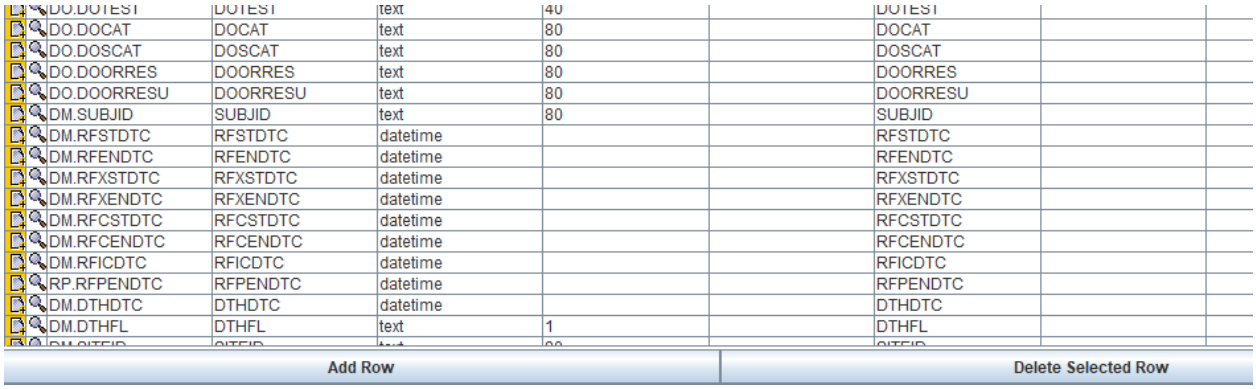


and when then looking into the "Dataset Definitions" tab, we find:



OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role
DI	DI	Yes	No				
DU	DU	Yes	No				
DX	DX	Yes	No				
DE	DE	Yes	No				
DT	DT	No	No				
DR	DR	No	No				
DO	DO	No	No				
DM	DM	No	No	DM	DM		

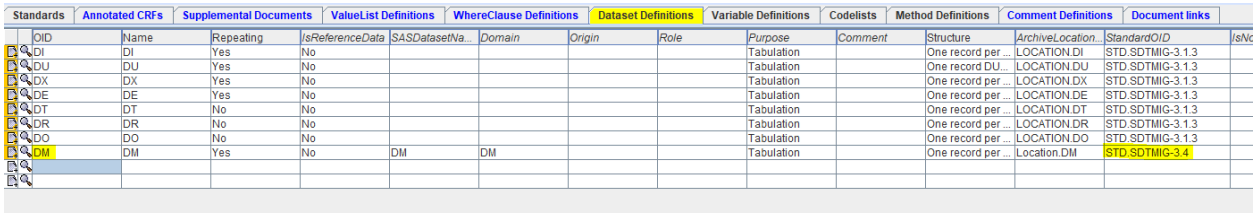
and in the "Variable Definitions" tab:



DO.DOTEST	DOTEST	text	40		DOTEST		
DO.DOCAT	DOCAT	text	80		DOCAT		
DO.DOSCAT	DOSCAT	text	80		DOSCAT		
DO.DOORRES	DOORRES	text	80		DOORRES		
DO.DOORRESU	DOORRESU	text	80		DOORRESU		
DM.SUBJID	SUBJID	text	80		SUBJID		
DM.RFSTDTC	RFSTDTC	datetime			RFSTDTC		
DM.RFENDTC	RFENDTC	datetime			RFENDTC		
DM.RFXSTDTC	RFXSTDTC	datetime			RFXSTDTC		
DM.RFXENDTC	RFXENDTC	datetime			RFXENDTC		
DM.RFCSTDTC	RFCSTDTC	datetime			RFCSTDTC		
DM.RFCENDTC	RFCENDTC	datetime			RFCENDTC		
DM.RFICDTC	RFICDTC	datetime			RFICDTC		
RP.RFPENDTC	RFPENDTC	datetime			RFPENDTC		
DM.DTHDTC	DTHDTC	datetime			DTHDTC		
DM.DTHFL	DTHFL	text	1		DTHFL		

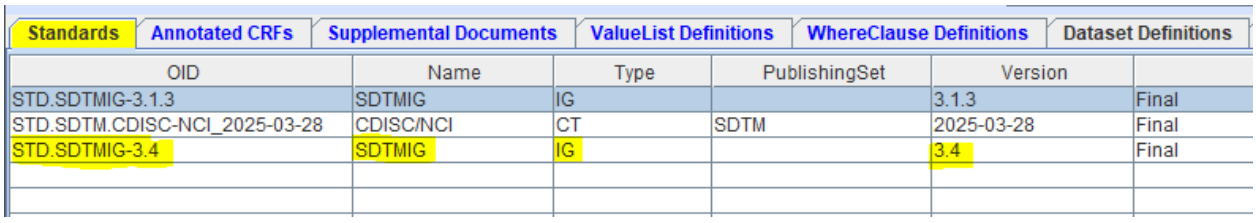
Add Row | Delete Selected Row

Also remark that when using Define-XML 2.1, the system takes care that the correct standard and version is assigned to each of the dataset definitions:



OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role	Purpose	Comment	Structure	ArchiveLocation...	StandardOID	IsNo
DI	DI	Yes	No					Tabulation		One record per ...	LOCATION.DI	STD.SDTMIG-3.1.3	
DU	DU	Yes	No					Tabulation		One record DU...	LOCATION.DU	STD.SDTMIG-3.1.3	
DX	DX	Yes	No					Tabulation		One record per ...	LOCATION.DX	STD.SDTMIG-3.1.3	
DE	DE	Yes	No					Tabulation		One record per ...	LOCATION.DE	STD.SDTMIG-3.1.3	
DT	DT	No	No					Tabulation		One record per ...	LOCATION.DT	STD.SDTMIG-3.1.3	
DR	DR	No	No					Tabulation		One record per ...	LOCATION.DR	STD.SDTMIG-3.1.3	
DO	DO	No	No					Tabulation		One record per ...	LOCATION.DO	STD.SDTMIG-3.1.3	
DM	DM	Yes	No	DM	DM			Tabulation		One record per ...	Location.DM	STD.SDTMIG-3.4	

and in the "Standards" tab, we find:

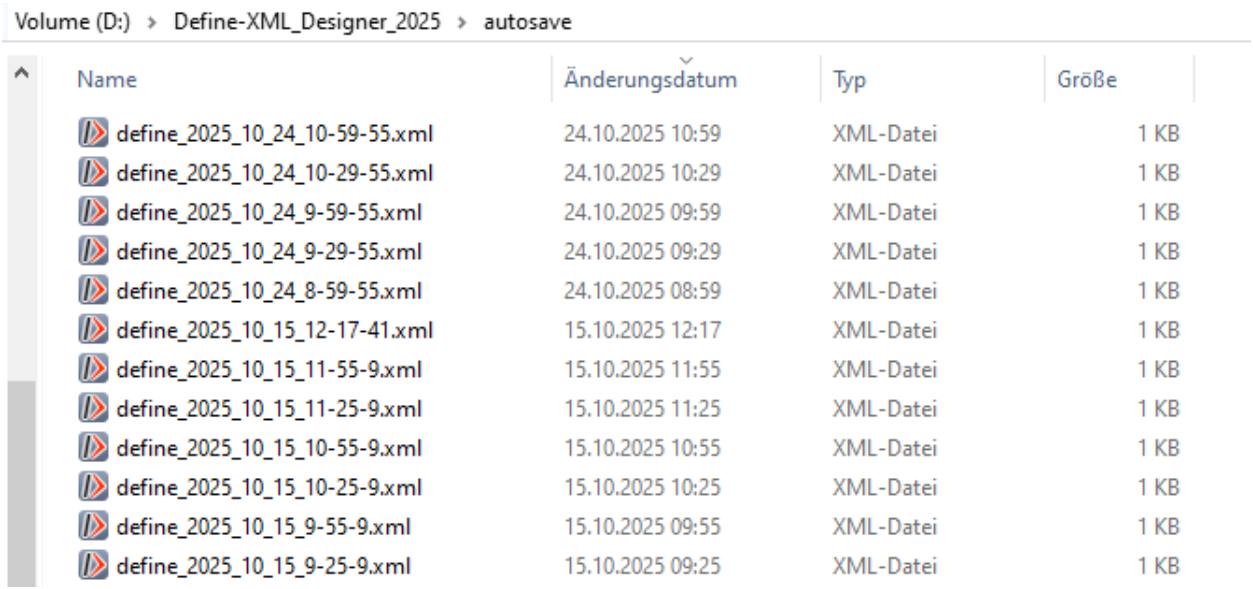


OID	Name	Type	PublishingSet	Version	
STD.SDTMIG-3.1.3	SDTMIG	IG		3.1.3	Final
STD.SDTM.CDISC-NCI_2025-03-28	CDISC/NCI	CT	SDTM	2025-03-28	Final
STD.SDTMIG-3.4	SDTMIG	IG		3.4	Final

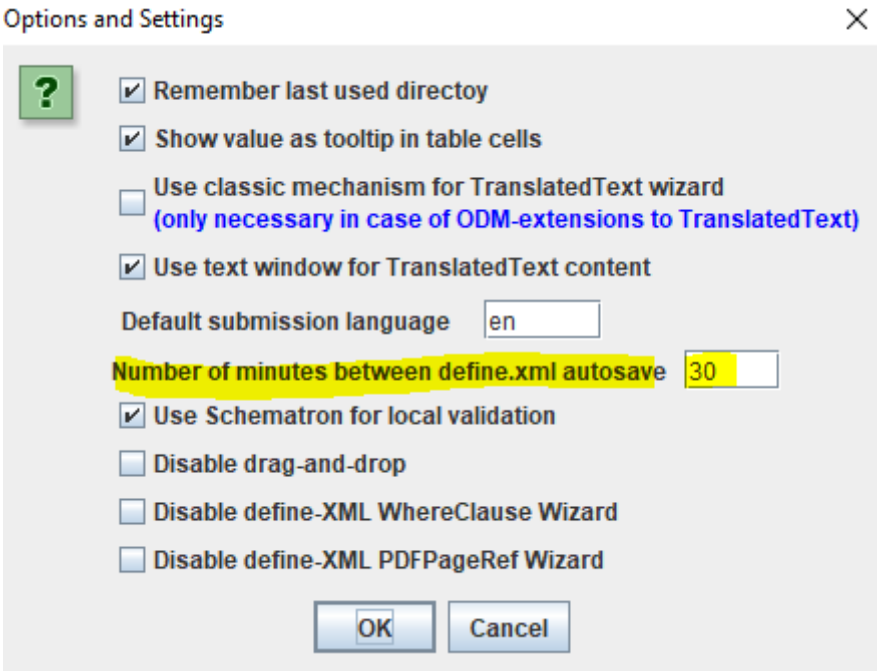
Remark that some adaptions may still be necessary, so it is not a bad idea to check the results of the additions.
The same procedure can then be followed for adding other domains from the same of other templates.

Using Autosave

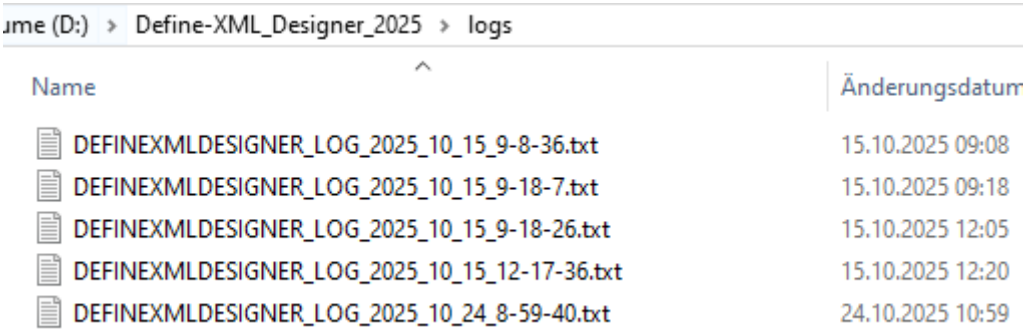
As already mentioned, when something goes wrong, one can always revert to an earlier version of the define.xml that was automatically saved. These "backup" define.xml are stored in the folder "autosave". For example:



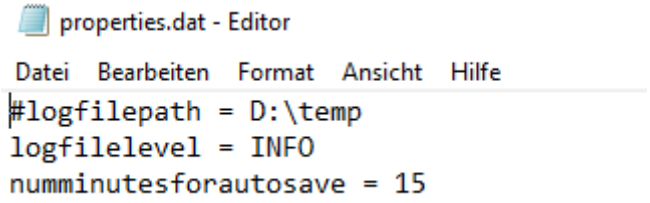
Such a "backup" define.xml is generated each 30 minutes, and can be loaded using the menu "File - Open define.xml". The interval between such "autosaves" can be changed using the menu "Options - Settings" and changing the value in the field "Number of minutes between define.xml autosave":



During each session, also a log file is generated and stored in the directory "logs", e.g.



The amount of logging can be set in the file "properties.dat". The allowed log-levels are "INFO" and "DEBUG":

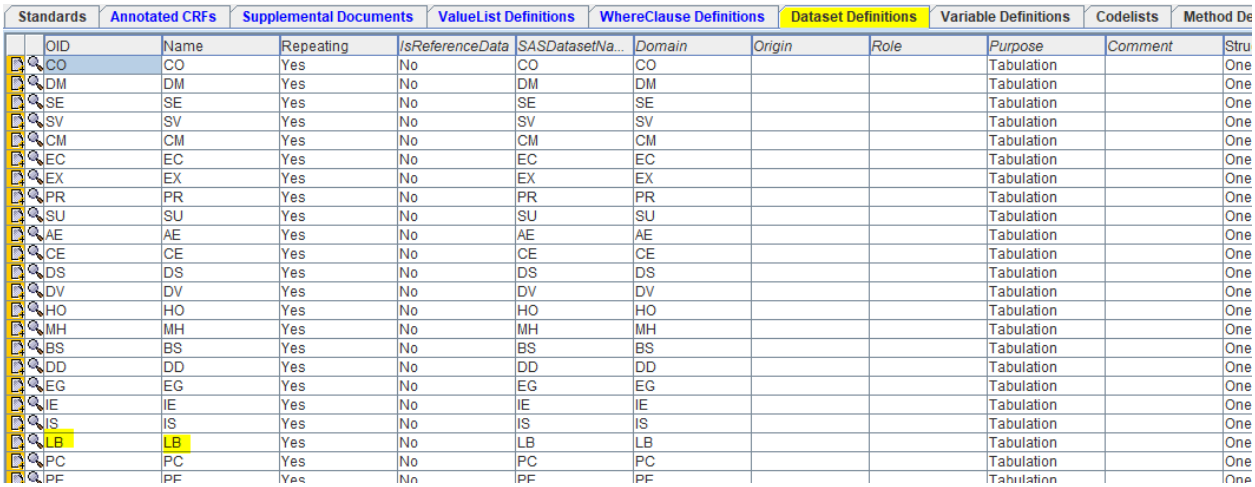


In this file, also the default value for the number of minutes between "autosaving" can be set, as well as the file path to which the log files are to be generated. Remark that lines starting with a "#" are "commented out" lines, and are ignored at startup

Adding to / Editing the list of variables for each dataset definition

Once we have decided which dataset definitions we want to have or retain, we should have a look at which variables we want to use in each of these definitions. Especially in SDTM and SEND, we need to take into account that some of the variables are "Required" (meaning the variable must be present and a value must always be present), "Expected" (meaning the variable must always be present, but there may be empty values" and "Permissible" (meaning that if no data is available for the variable, the column in the dataset may be omitted).
Furthermore, it is advisable to keep the order of the variables as provided in the corresponding Implementation Guide. It may sometimes also be necessary to add additional variables from the (SDTM) "Model", which are not mentioned in the IG for that domain. This is often the case for "Timing" variables.

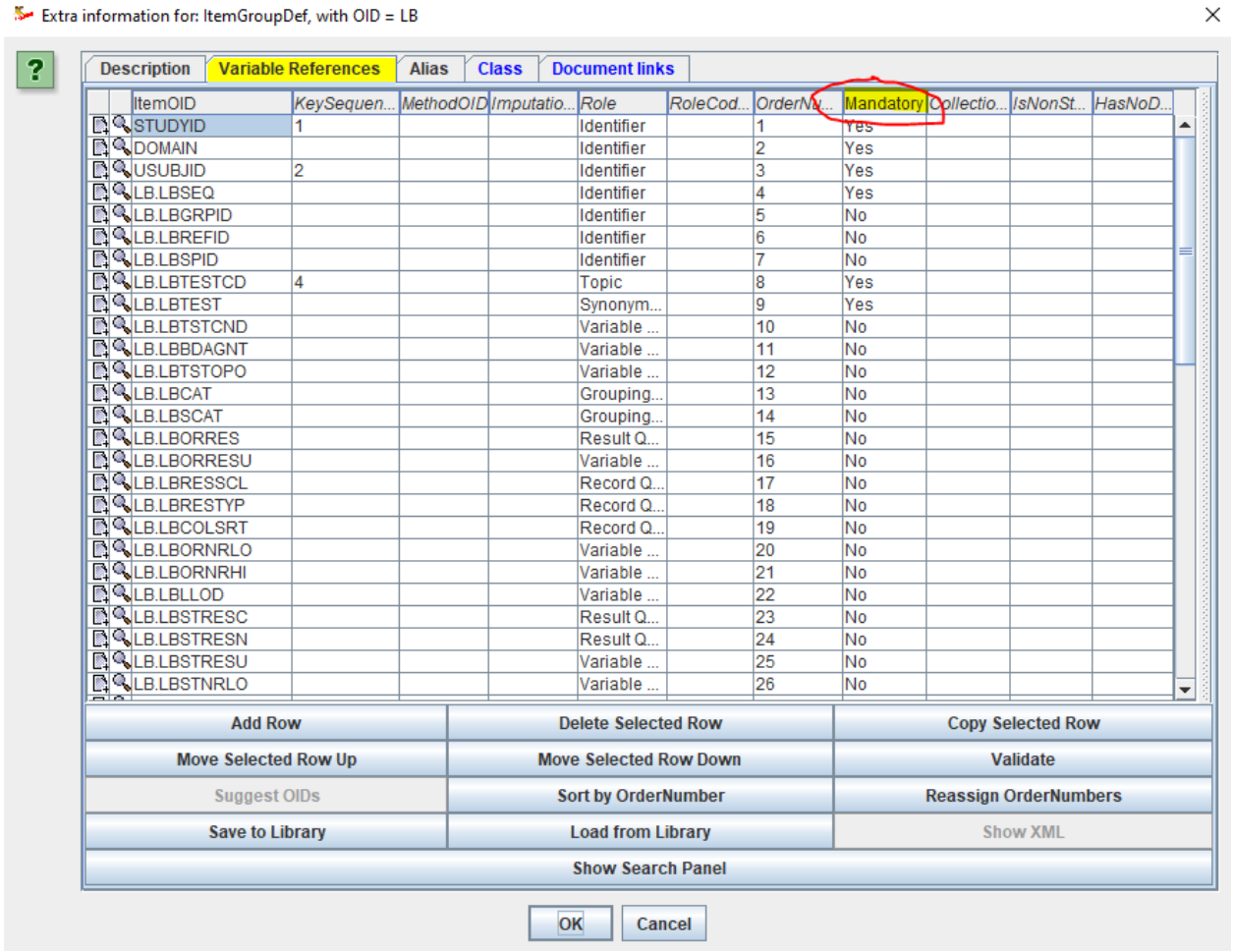
Let us take the LB (Laboratory) dataset definition as an example again. When using the tab "Dataset Definitions"



OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role	Purpose	Comment	Stru
CO	CO	Yes	No	CO	CO			Tabulation		One
DM	DM	Yes	No	DM	DM			Tabulation		One
SE	SE	Yes	No	SE	SE			Tabulation		One
SV	SV	Yes	No	SV	SV			Tabulation		One
CM	CM	Yes	No	CM	CM			Tabulation		One
EC	EC	Yes	No	EC	EC			Tabulation		One
EX	EX	Yes	No	EX	EX			Tabulation		One
PR	PR	Yes	No	PR	PR			Tabulation		One
SU	SU	Yes	No	SU	SU			Tabulation		One
AE	AE	Yes	No	AE	AE			Tabulation		One
CE	CE	Yes	No	CE	CE			Tabulation		One
DS	DS	Yes	No	DS	DS			Tabulation		One
DV	DV	Yes	No	DV	DV			Tabulation		One
HO	HO	Yes	No	HO	HO			Tabulation		One
MH	MH	Yes	No	MH	MH			Tabulation		One
BS	BS	Yes	No	BS	BS			Tabulation		One
DD	DD	Yes	No	DD	DD			Tabulation		One
EG	EG	Yes	No	EG	EG			Tabulation		One
IE	IE	Yes	No	IE	IE			Tabulation		One
IS	IS	Yes	No	IS	IS			Tabulation		One
LB	LB	Yes	No	LB	LB			Tabulation		One
PC	PC	Yes	No	PC	PC			Tabulation		One
PE	PE	Yes	No	PE	PE			Tabulation		One

and then clicking the "Edit sub-information"  , and selecting the "Variable References" tab, we get the list of the

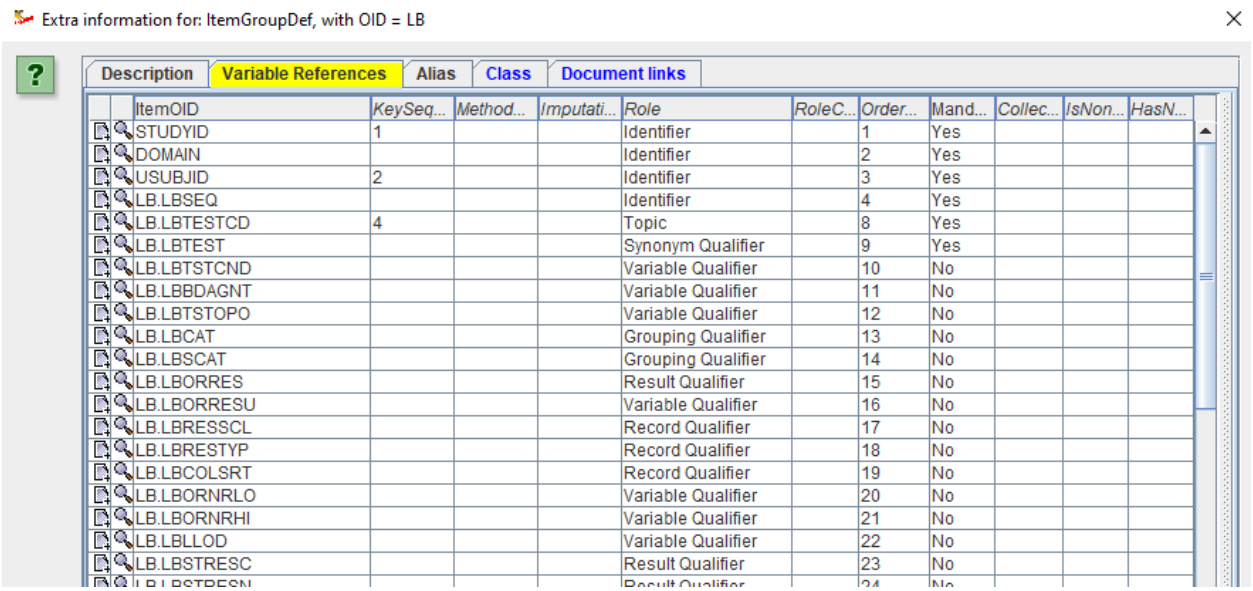
variables for this dataset definition, e.g. when coming from the template:

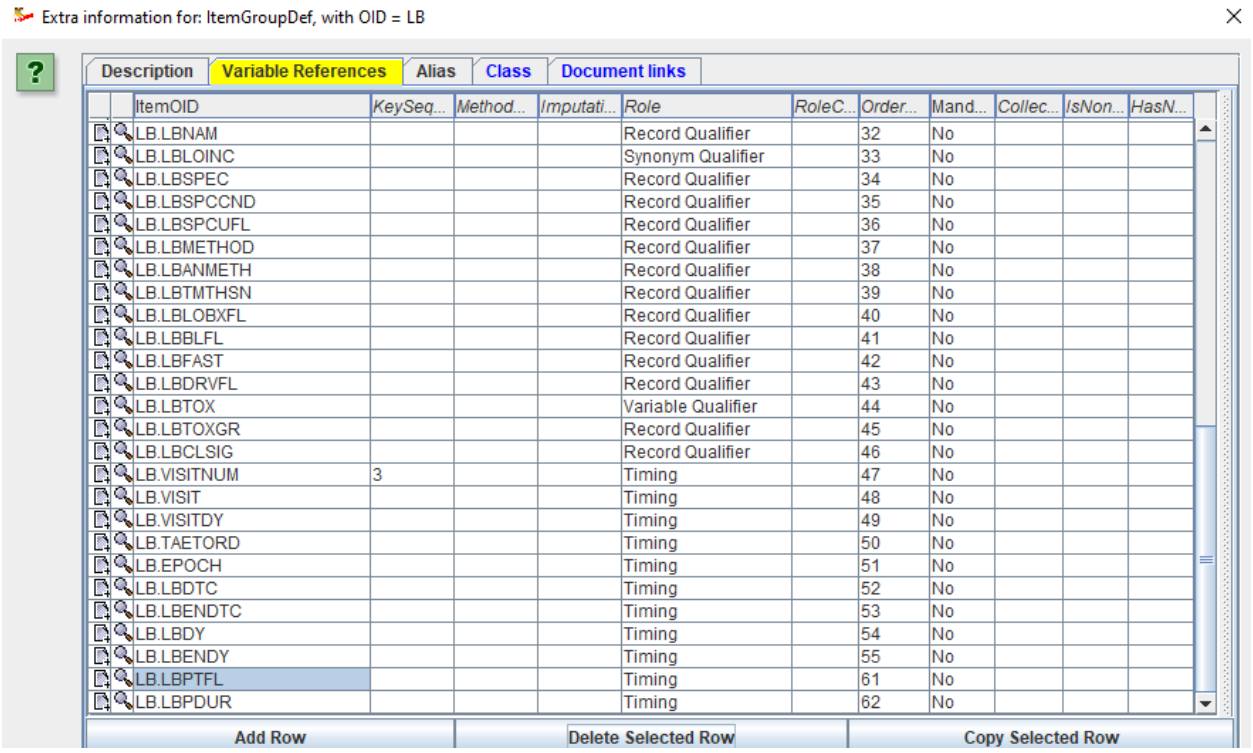


The column "Mandatory" is very important! For each variable, when the value is "Yes", this corresponds to either "Required" or "Expected" in the IG. So, when an standard-compliant dataset is envisaged, such variables should NOT be removed from the list! Variables with Mandatory=No can be removed when one is sure there is no data for it.

Suppose that for our submission, we do not need **LBGRPID** (Group ID), **LBREFID** (Specimen ID), and **LBSPID** (Sponsor-defined Identifier), but we do want to add LBSTDTC (Start Date/Time of Observation) and LBENDTC (End Date/Time of Observation) as we e.g. have tests that span over an amount of time, such as for Urine collected over a period of 24 hours. For such, we may also want to add LBDUR (Duration - defined as "Collected duration of an event, intervention, or finding").
As we do not have time points for the lab tests (i.e. we only can have one set of lab tests per visit), we will also remove **LBTPT** (Planned Time Point Name), **LBTPTNUM** (Planned Time Point Number), **LBELTM** (Planned Elapsed Time from Time Point Ref), **LBTPTREF** (Time Point Reference), **LBRFTDTC** (Date/Time of Reference Time Point). We however want to keep LBPTFL (Point in Time Flag) and LBPDUR (Planned Duration) to distinguish between tests for which is there is a time span of collection, and tests that are just "single point in time".

For the variables to be removed, we can just select any cell for each of them, and then use the button "Delete Selected Rows". This will lead to:

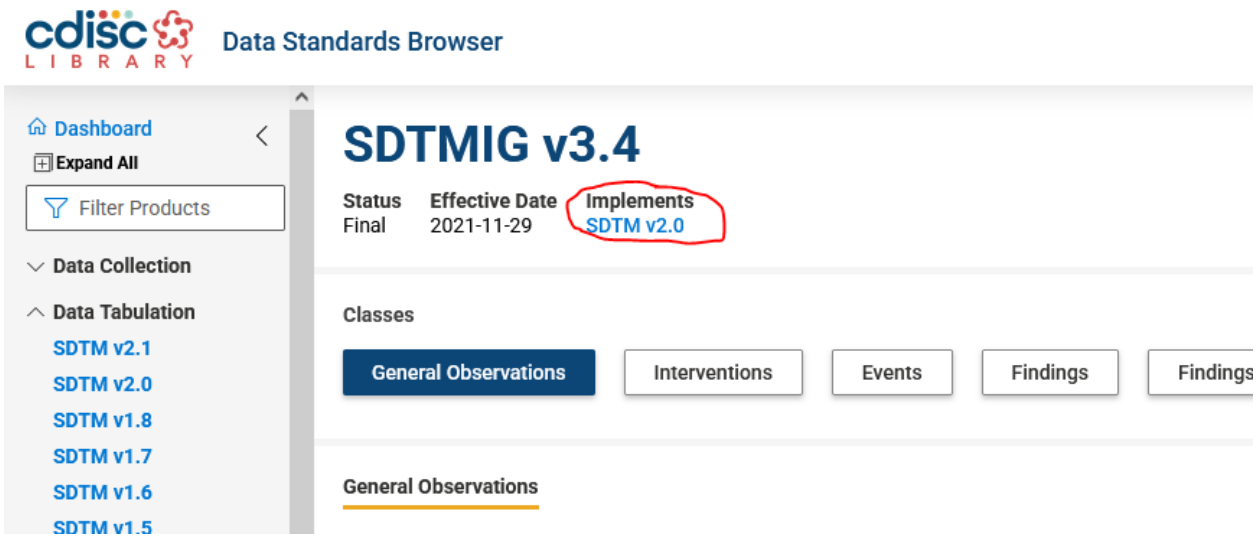




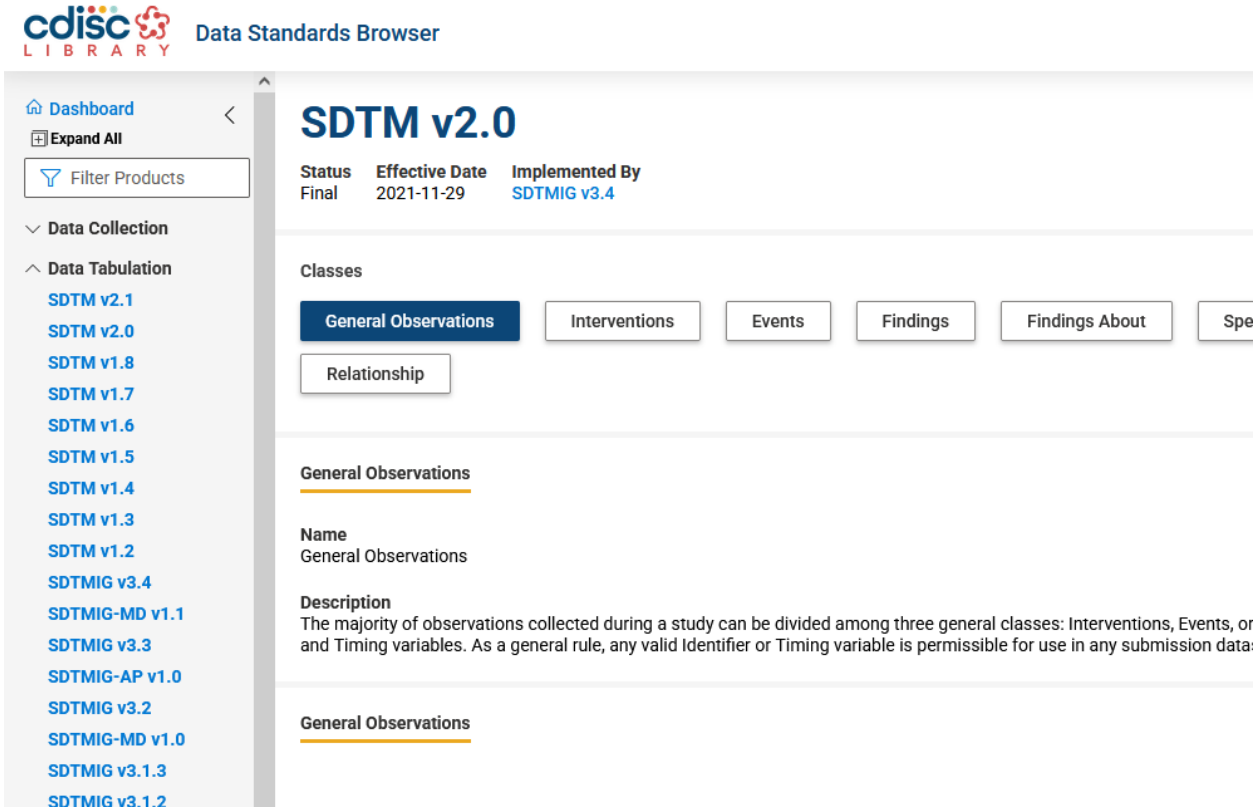
Remark that we have "gaps" in the "OrderNumber", but that is not a problem, as there is no Define-XML that states that the values should be subsequent numbers.

We do however also want to add LBSTDTC (Start Date/Time of Observation), LBENDTC (End Date/Time of Observation) and LBDUR (Duration). Question is of course where they should exactly come …

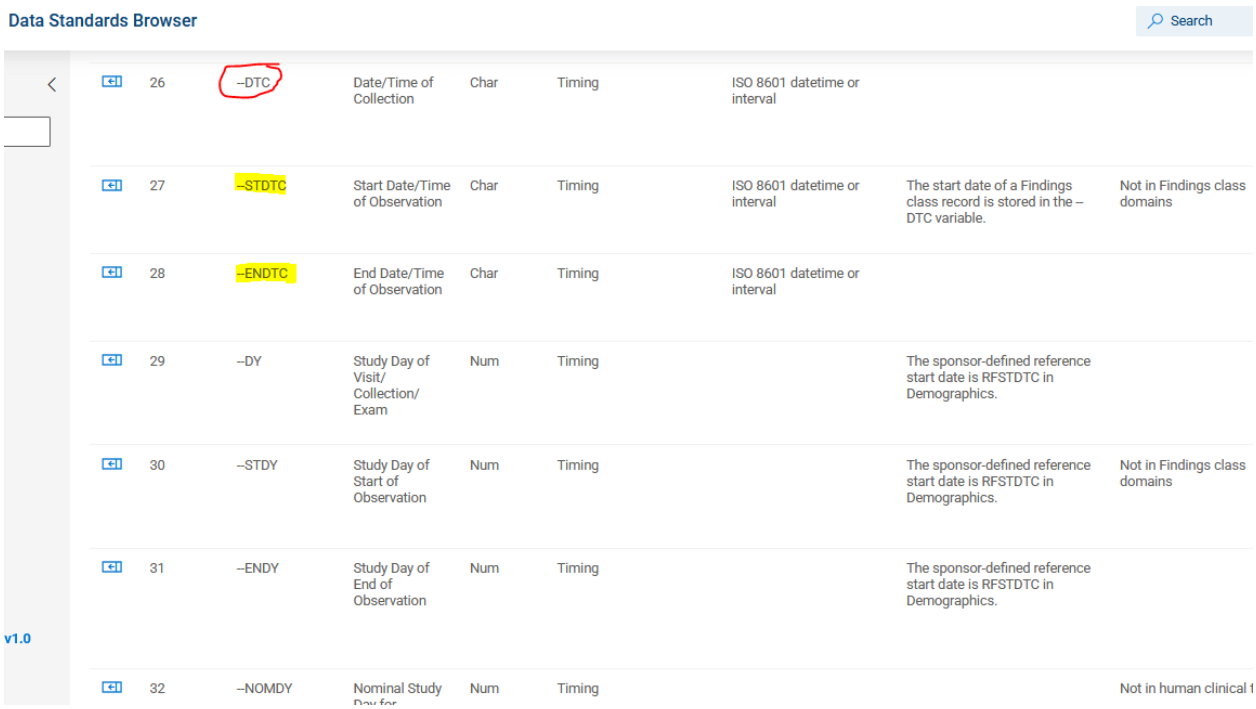
To know this, we need to look in the "SDTM Model", which is nicely available online through the "CDISC Library Browser". We there select SDTMIG-3.4 and see that it is based on the "SDTM Model" version 2.0:



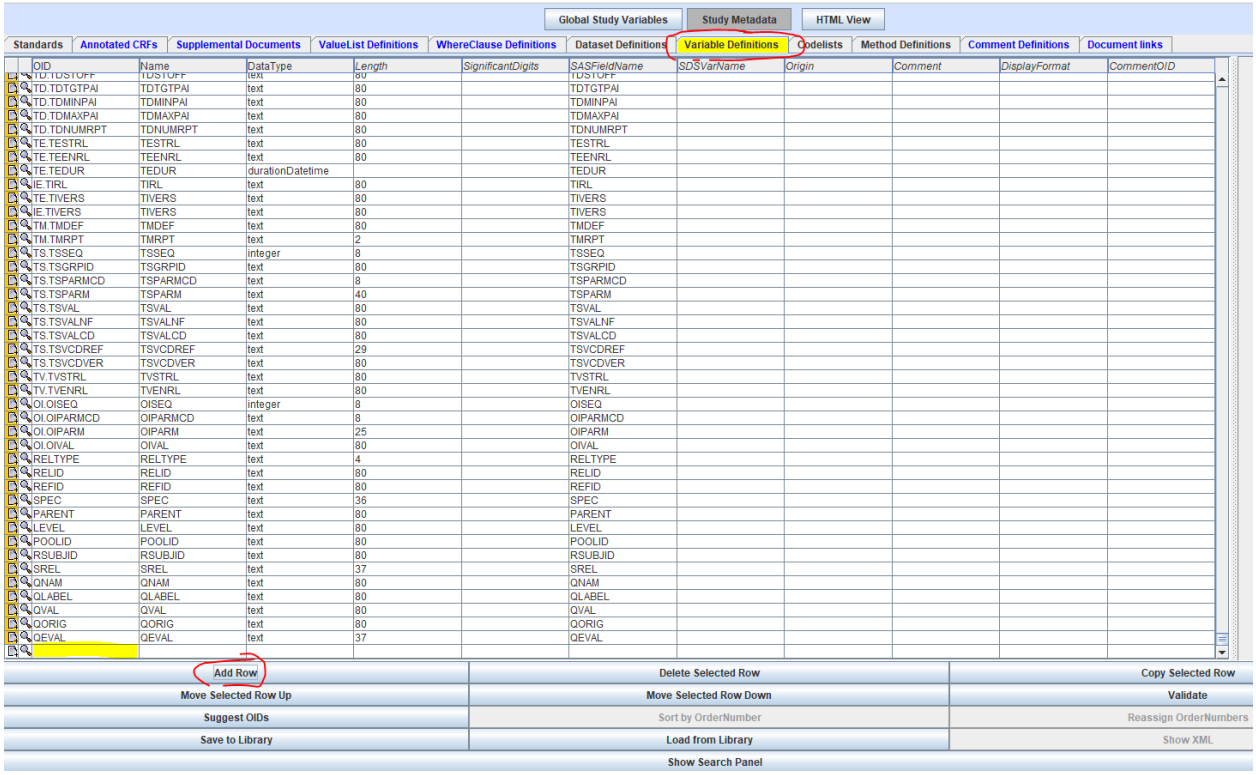
As we want to add variables for LB that are not in the IG, we need to look into the "Model" to get information about them, including the correct order, so we click on "SDTM v.2.0", leading to:



The timing variables we want to insert (LBSTDTC, LBENDTC, LBDUR) fall under "General Observations", and when scrolling down, we easily find them as "--STDTC", "--ENDTC" and "--DUR", as the model is meant for all "observation" domains:



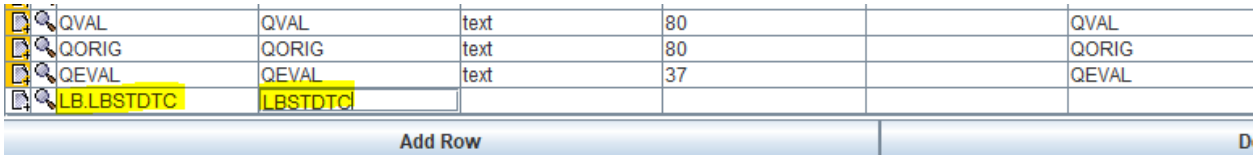
where we see that LBSTDTC and LBENDTC must come immediately after LBDTC.
However, we must first just just define them under "Variable Definitions" before we can reference them from the "Dataset definition" for LB[7].
So we first select the tab "Variable Definitions", and click "Add Row" to add a new, empty row at the bottom:



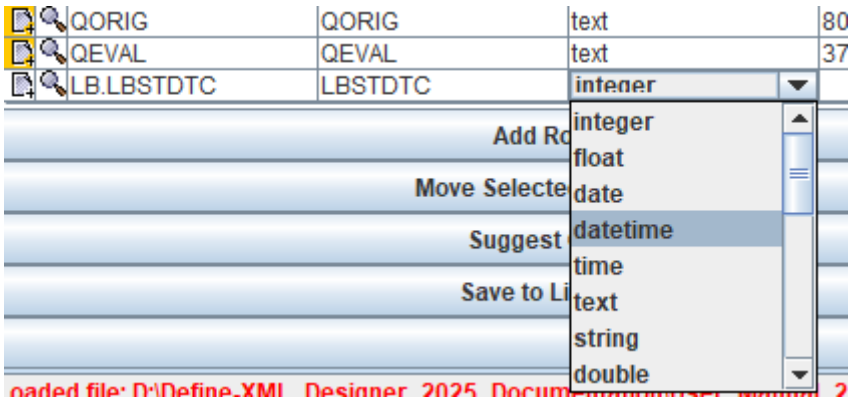
and add the necessary information in this new, empty row. We add an OID (identifier), like "LB.LBSTDTC" and the variable name "LBSTDTC":

---

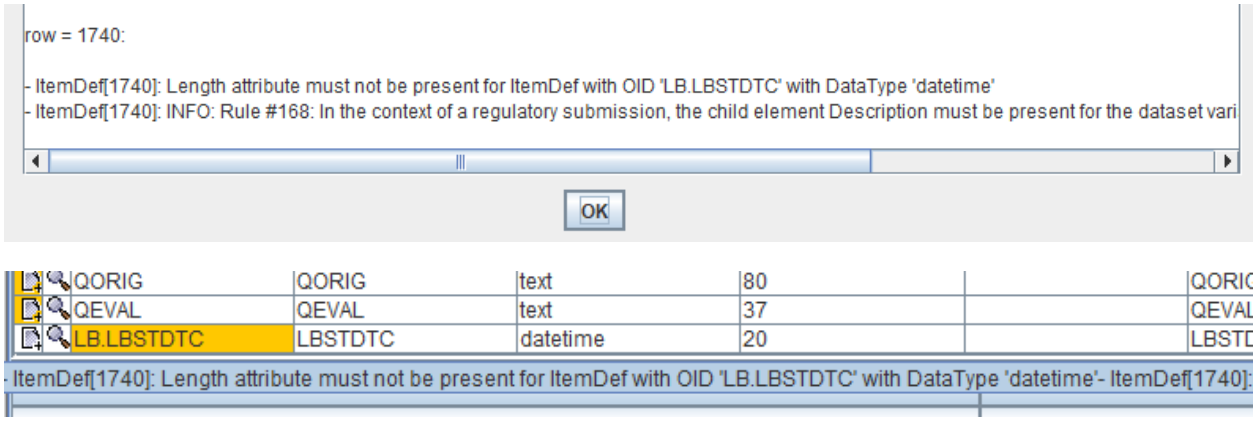
[7] In future we intend to automate this by getting the information from the CDISC Library API. This would however that the user has a Library API key.



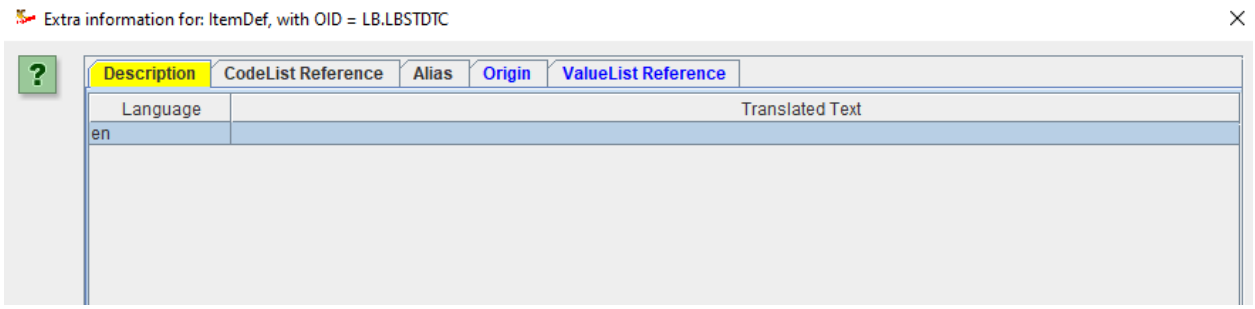
and set the "DataType" to "datetime":



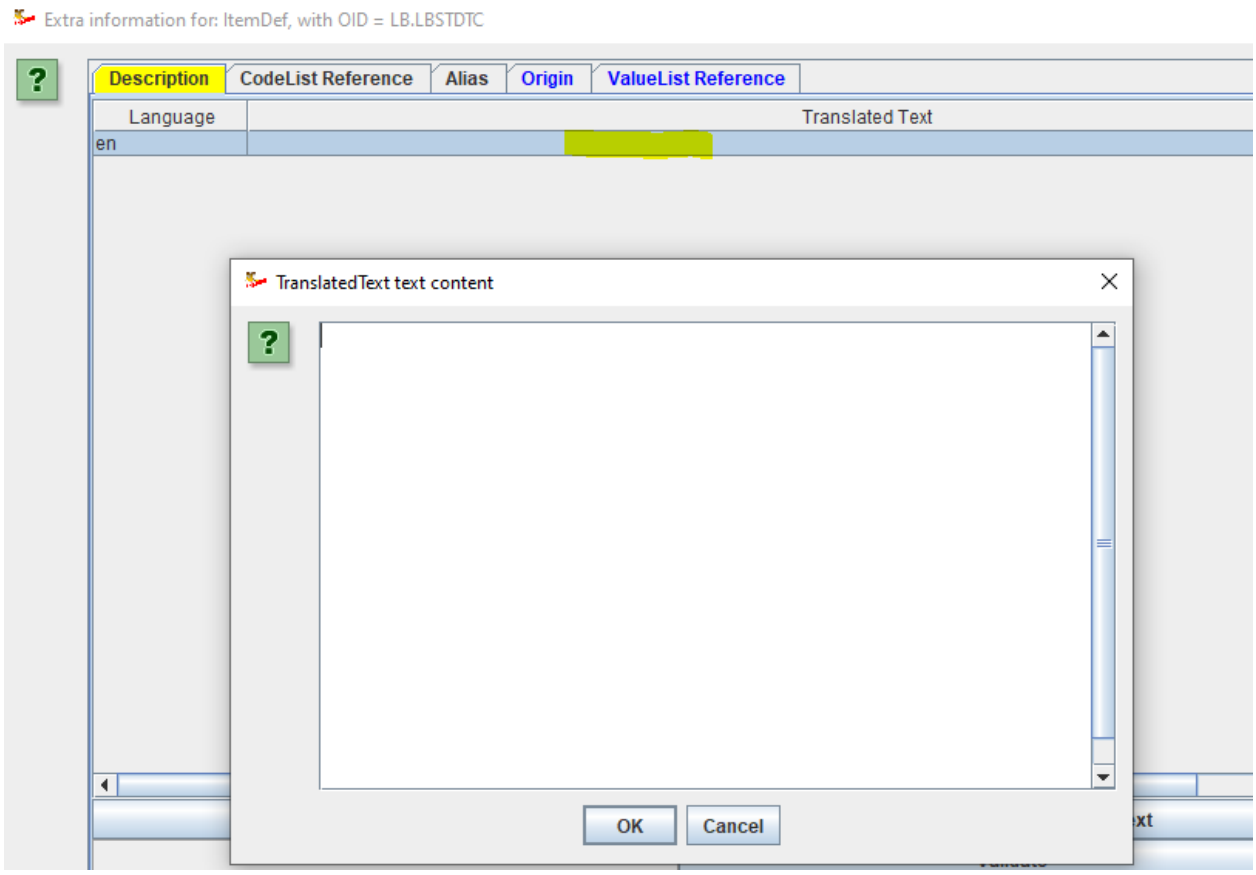
For "datetime", no "Length" should be assigned due to the Define-XML specification. But we set it to e.g. "20" just to see what happens... We also add "LBSTDTC" in the column "SASFieldName" as the Define-XML specification states "*Required in the context of a regulatory submission when the data is submitted as SAS XPT files.*".
Are we done? When clicking the "Validate" button, the first cell gets colored yellow, and a message is shown:



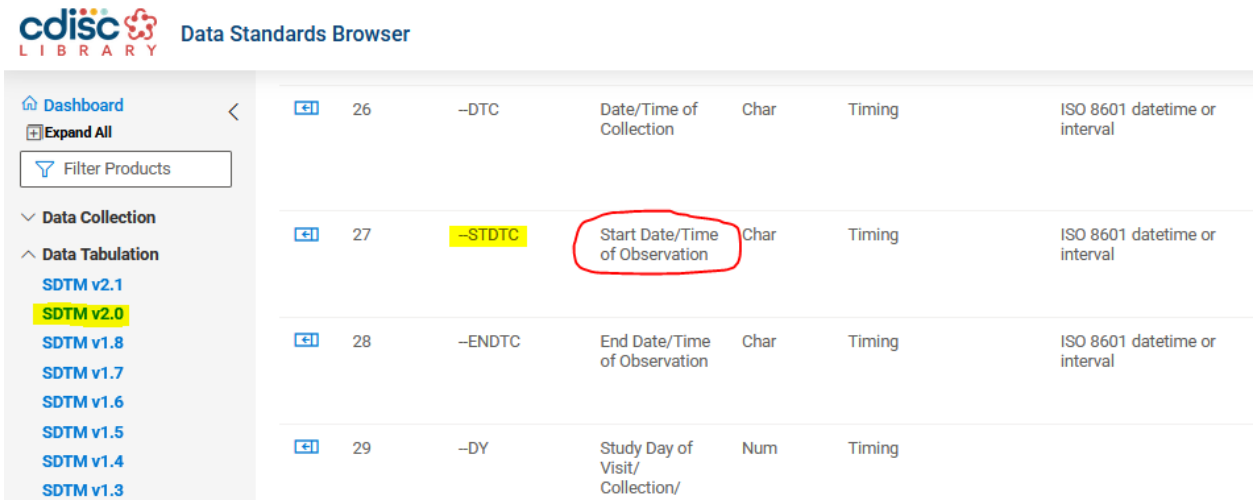
stating that for DataType=datetime, Length should not be populated, and a child "Description" element must be populated (this is the "Variable Label"). So we remove "20" from the "DataType" cell, and then click the "Edit sub-information" icon to add the variable label / description:



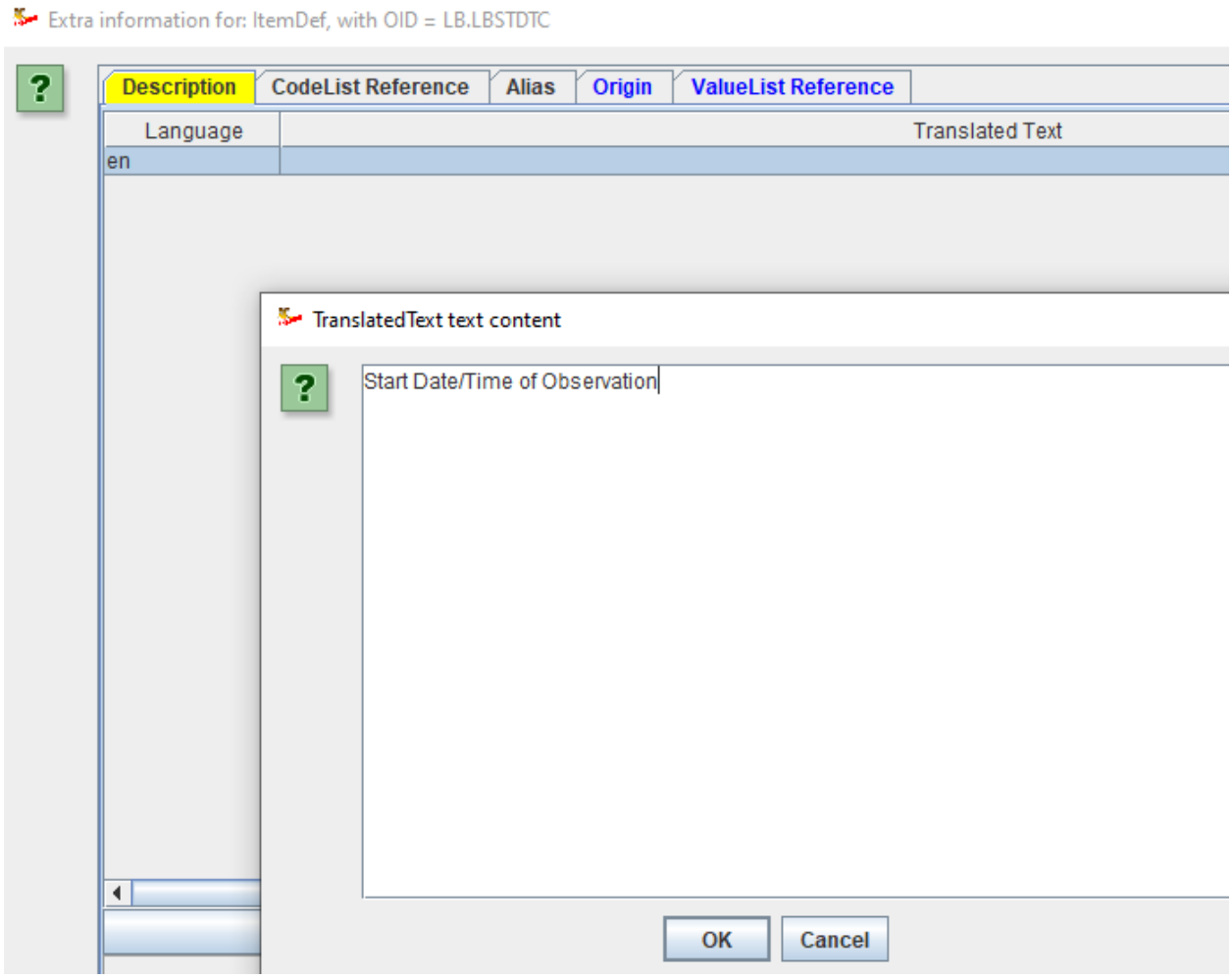
The English description can then be added by clicking in the cell "TranslatedText":



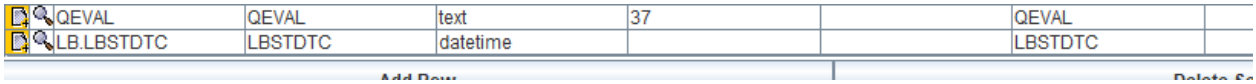
But what do we need as the "variable label", as LBSTDTC is not mentioned at all in the SDTMIG?
In such a case, we need to take the "Model" (here SDTM v.2.0), which is (as found in the "CDISC Library Browser"):



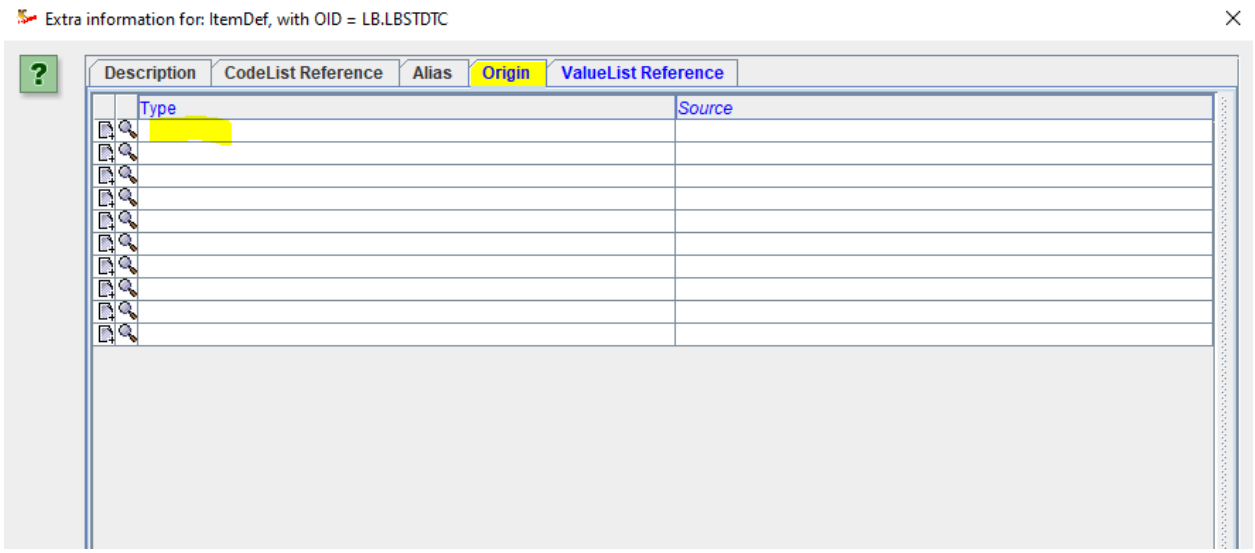
So we can just copy-paste "Start Date/Time of Observation":



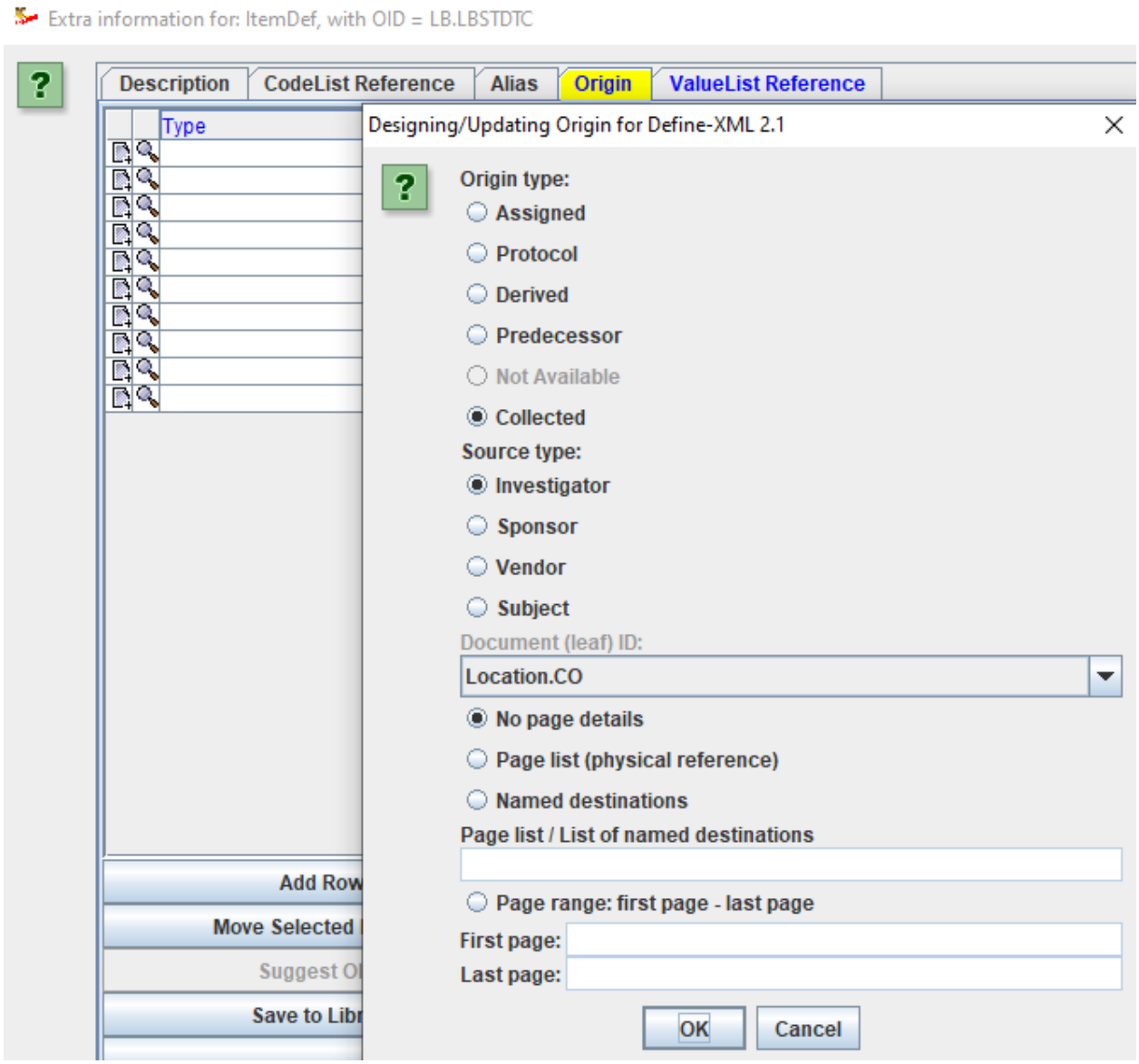
clicking "OK" twice, and then again clicking "Validate", the error messages (at least for LBSTDTC) disappear:



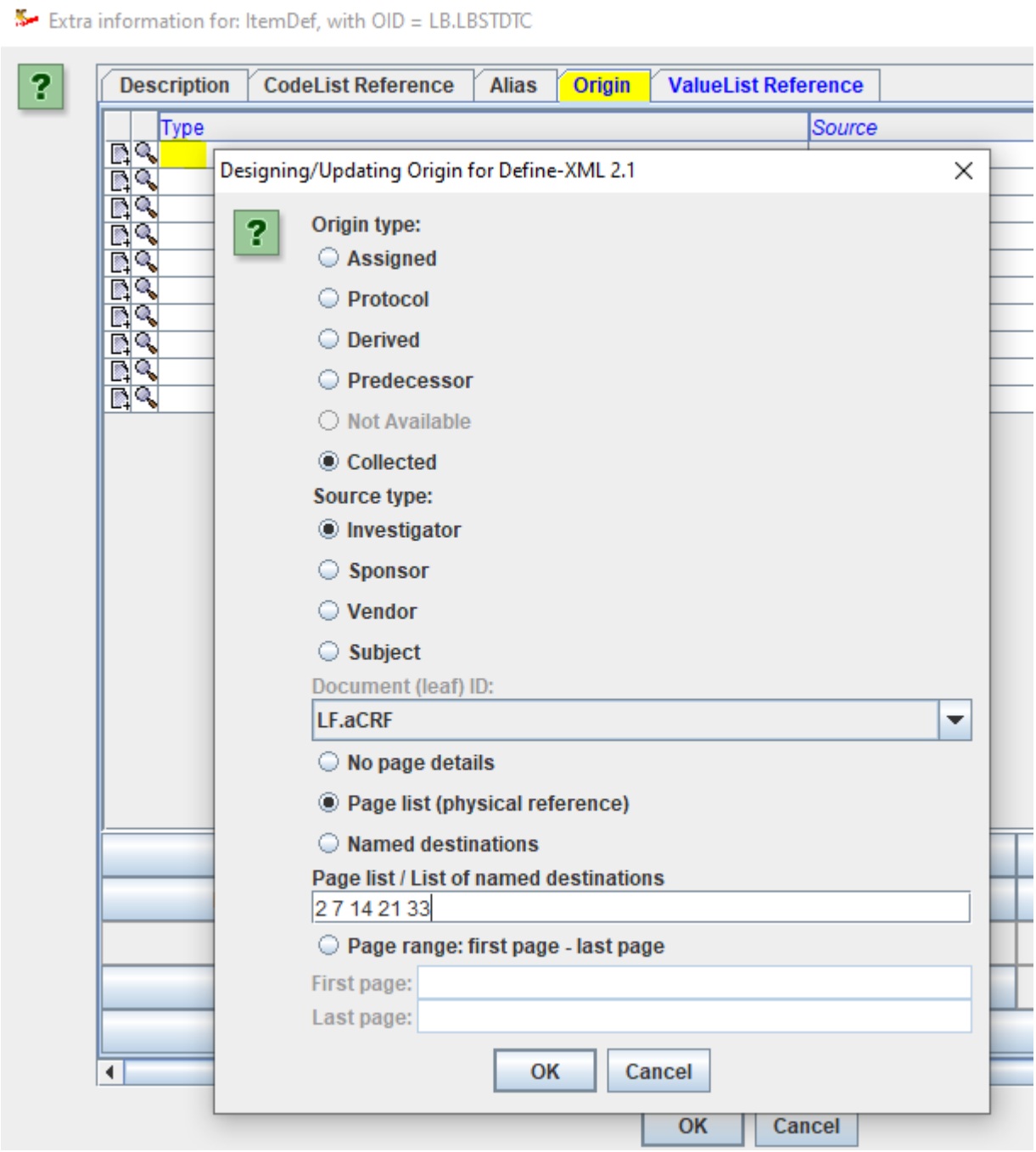
If we already know what the "source" of the LBSTDTC data points will be, we can add this information by clicking the "Edit sub-information" icon again, selecting the "Origin" tab, leading to:



When clicking in the first "Type" cell, the "Source/Origin" wizard will pop up:



and if the start-datetime was collected in the CRF by the investigator, we can point to it, and provide the page numbers when available, e.g.:

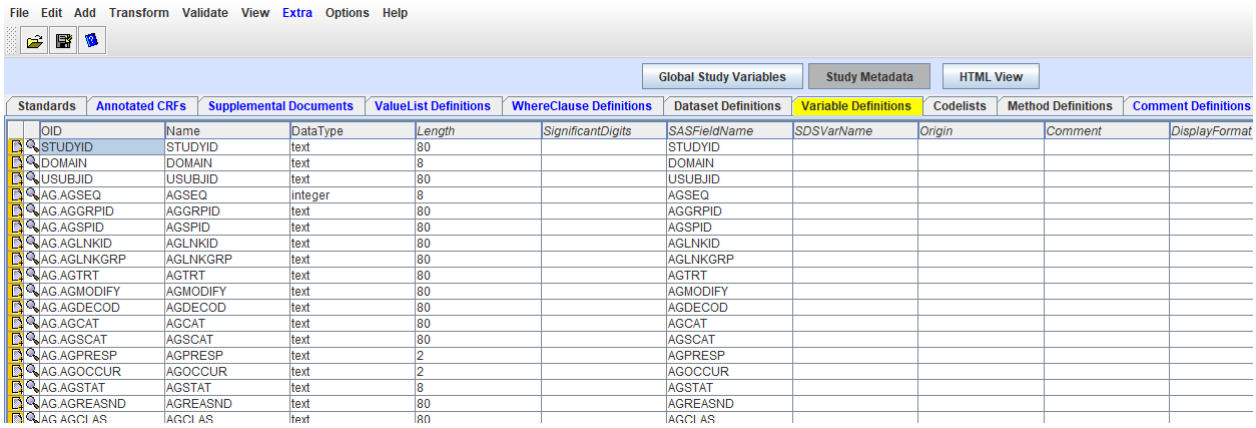


Clicking "OK several times lead to the main table.

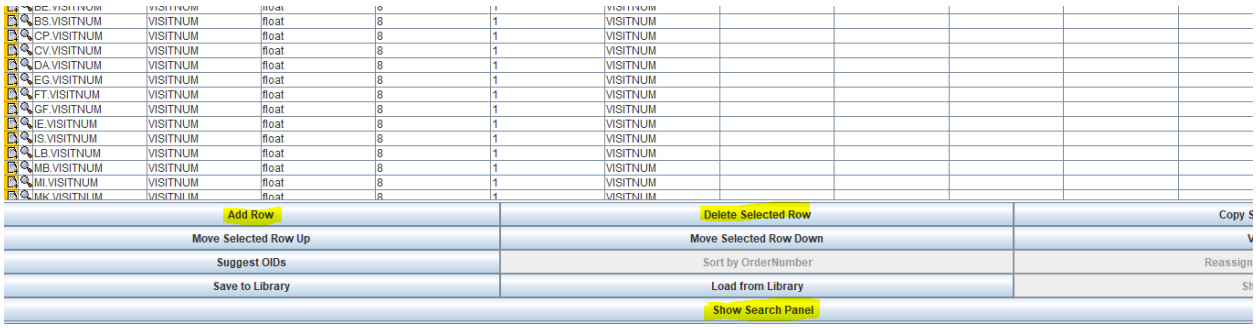
We should then do the same for LBENDTC and LBDUR.

Editing variable properties

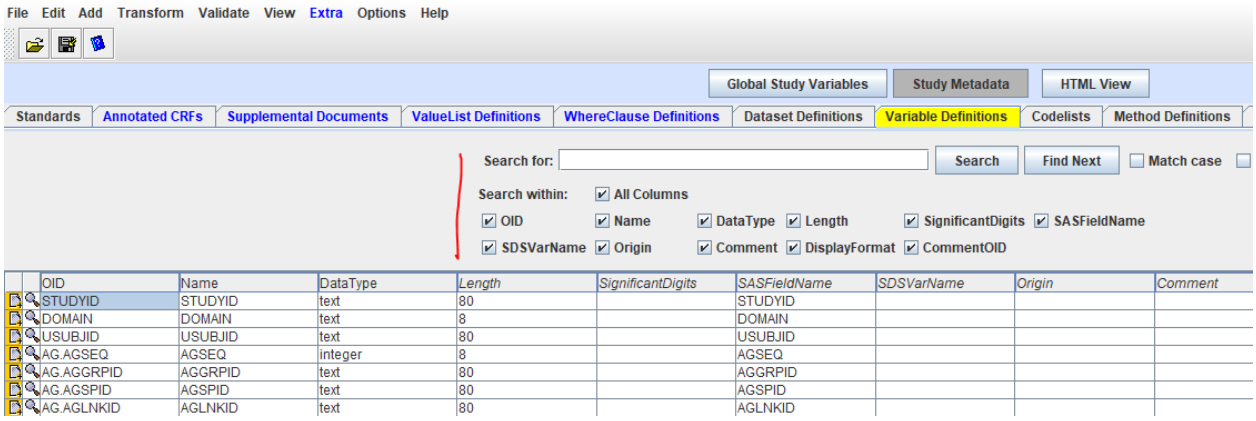
We have already seen how one can insert a new variable (e.g. from the SDTM "Model") and then "add" it (i.e. "reference it") to the dataset definition (ItemGroupDef).
We can of course also edit the properties of already defined variables.
To do so, select the tab "Variable Definitions":



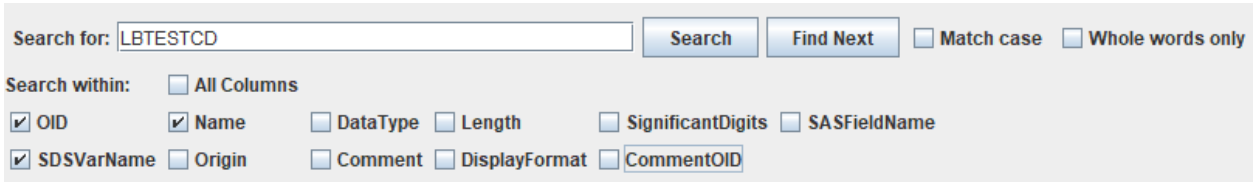
providing a list of all variable definitions currently available.
Also here, we can add or remove variable definitions using the "Add Row" and "Delete Selected Row" buttons neat the bottom:



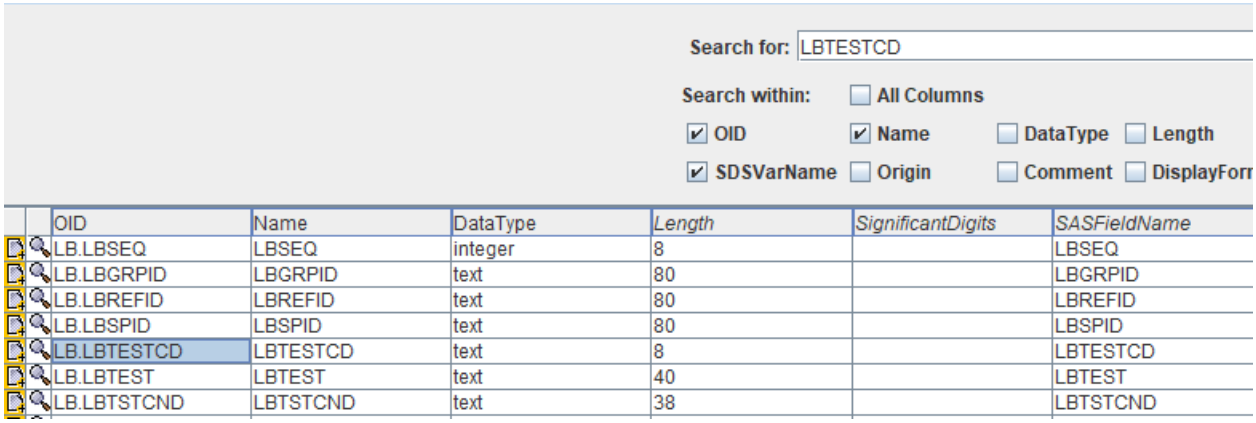
However, with "deleting" one must be careful, especially when it is about a "required" or "expected" variable.
Also, as we have a lot of variables, using the "Show Search Panel" may be very helpful to find a specific one:



E.g. when looking for "LBTESTCD":

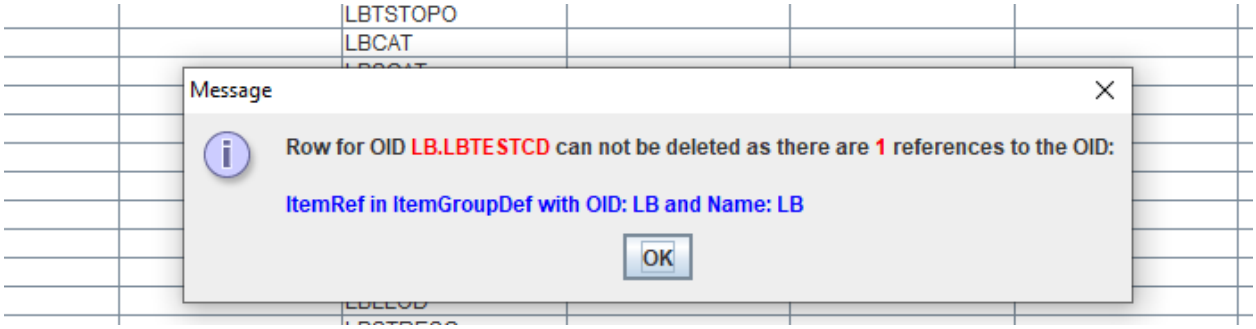


and then clicking the "Search" button leads to the "LBTESTCD" row being selected:

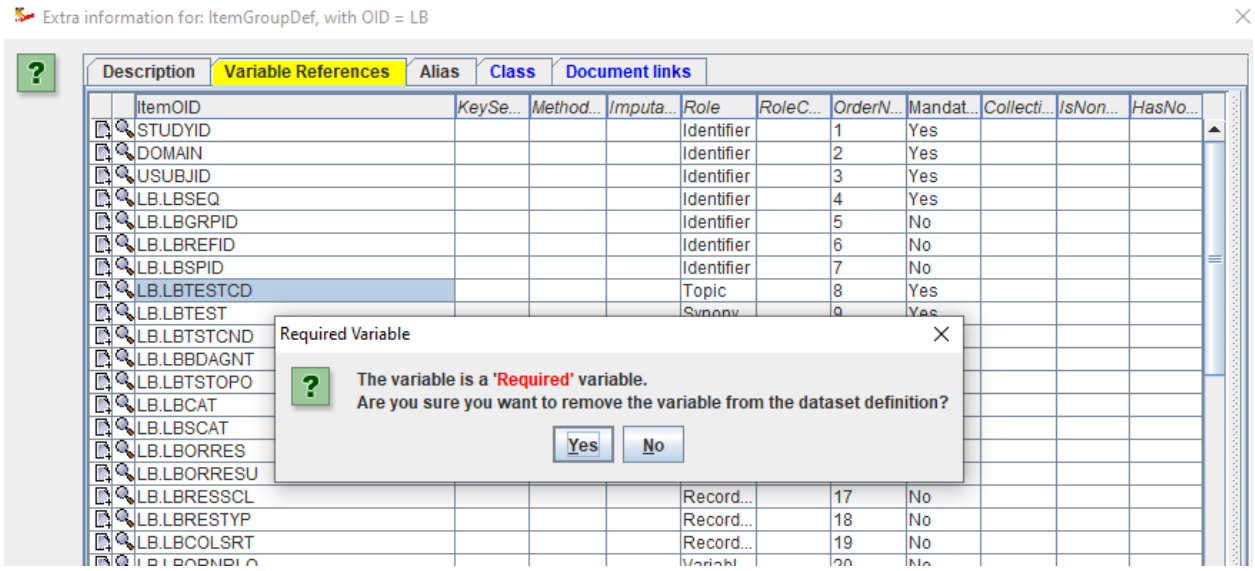


OID	Name	DataType	Length	SignificantDigits	SASFieldName
LB.LBSEQ	LBSEQ	integer	8		LBSEQ
LB.LBGRPID	LBGRPID	text	80		LBGRPID
LB.LBREFID	LBREFID	text	80		LBREFID
LB.LBSPID	LBSPID	text	80		LBSPID
LB.LBTESTCD	LBTESTCD	text	8		LBTESTCD
LB.LBTEST	LBTEST	text	40		LBTEST
LB.LBTSTCND	LBTSTCND	text	38		LBTSTCND

However, LBTESTCD is a "required" variable, so when we then click the "Delete Selected Row", the system reacts with:

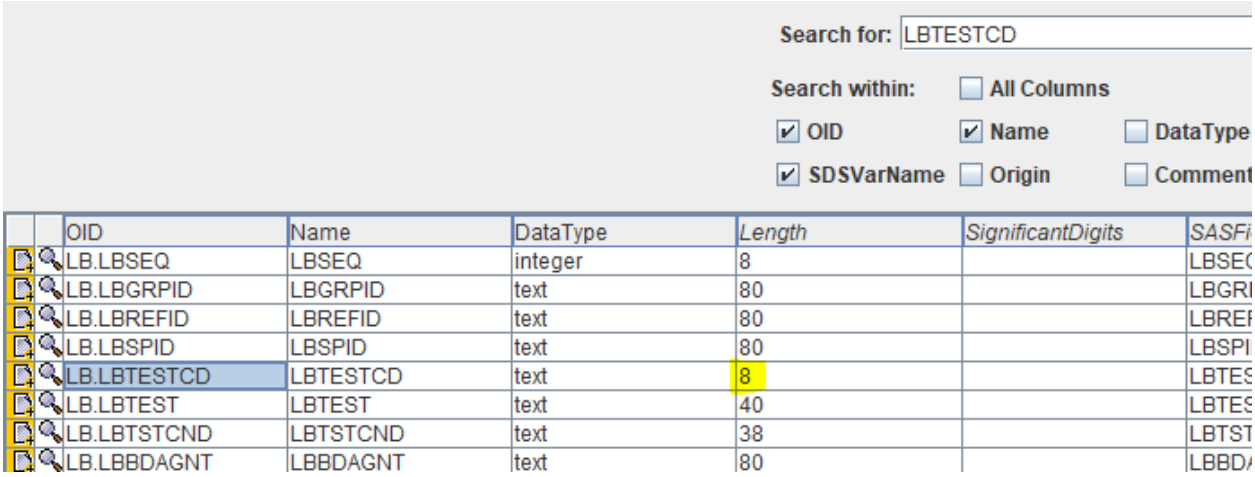


suggesting us to first remove it from the dataset definition contents. But if we try that (using tab "Dataset Definitions" and then clicking the "Edit" icon and select the "Variable References" tab), the system is asking whether we really want to remove LBTESTCD:



ItemOID	KeySe...	Method...	Imputa...	Role	RoleC...	OrderN...	Mandat...	Collecti...	IsNon...	HasNo...
STUDYID				Identifier		1	Yes			
DOMAIN				Identifier		2	Yes			
USUBJID				Identifier		3	Yes			
LB.LBSEQ				Identifier		4	Yes			
LB.LBGRPID				Identifier		5	No			
LB.LBREFID				Identifier		6	No			
LB.LBSPID				Identifier		7	No			
LB.LBTESTCD				Topic		8	Yes			
LB.LBTEST				Synony		9	Yes			
LB.LBTSTCND										
LB.LBBDAGNT										
LB.LBTSTOPO										
LB.LBCAT										
LB.LBSCAT										
LB.LBORRES										
LB.LBORRESU										
LB.LBRESSCL				Record...		17	No			
LB.LBRESTYP				Record...		18	No			
LB.LBCOLSRT				Record...		19	No			

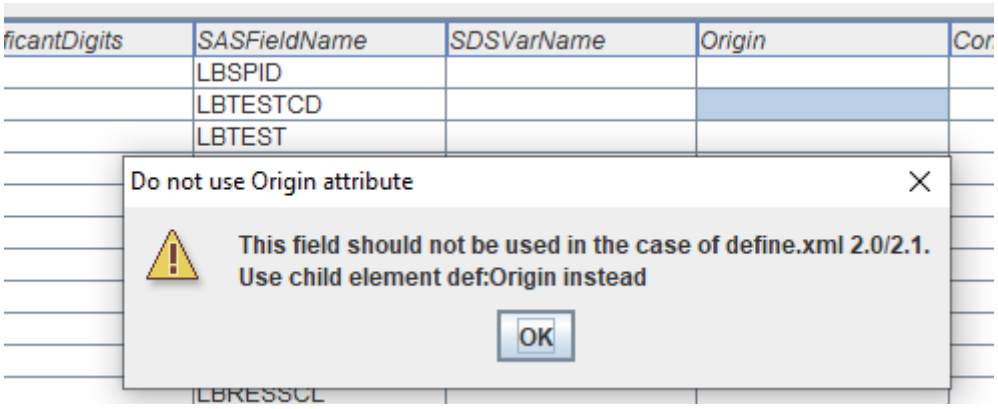
Back now to LBTESTCD in the "Variable Definitions" tab:



We e.g. see that the "Length" has been set to "8", but if we already know that the LBTESTCD value will never be longer than 6 characters, we can already change that.
In order to do that, just select the "Length" cell for LBTESTCD, and add the new length, e.g.:

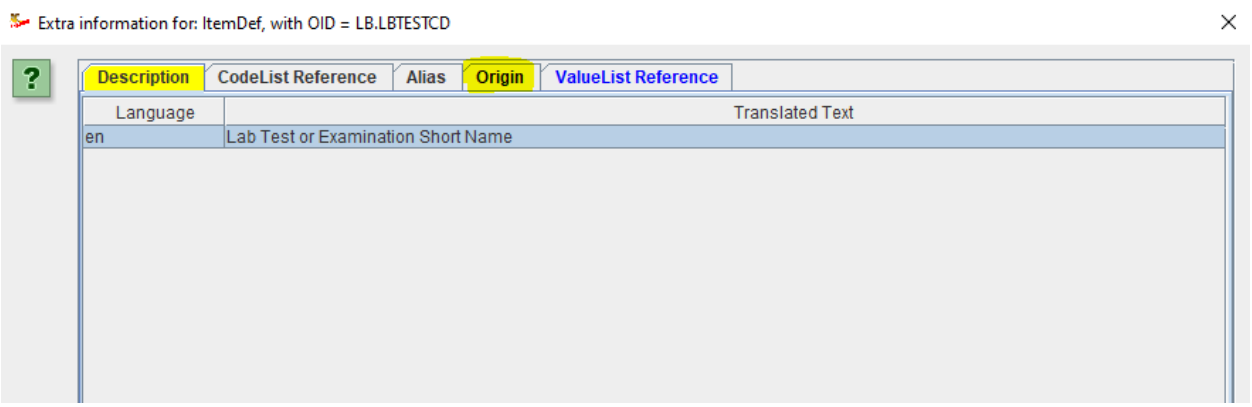
As you can easily find out, you can only type in non-integer values.

There is also a column "Origin". When we select it, the following message appears:

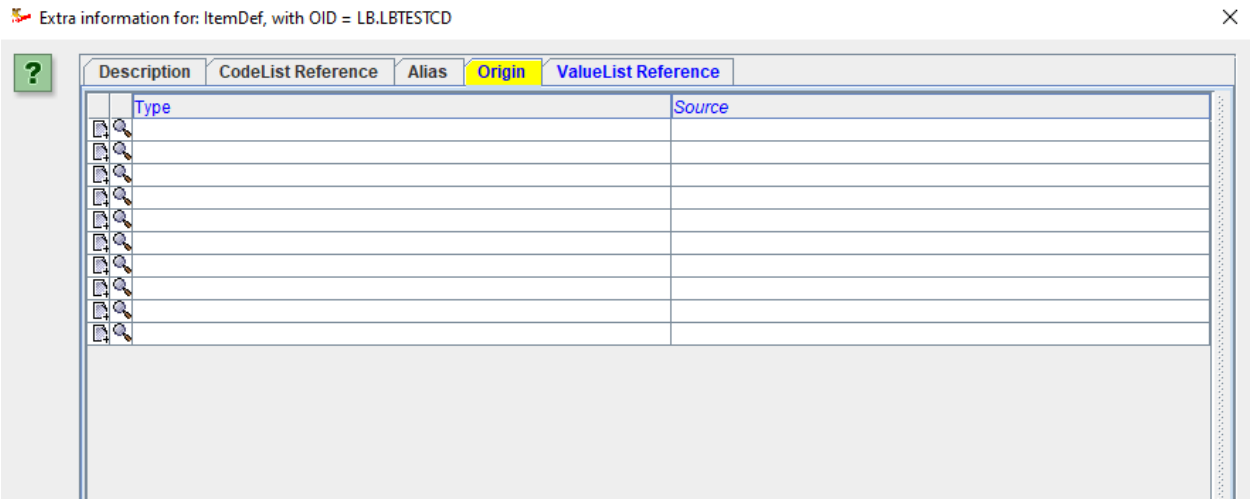


Reason is that "Origin" as an attribute of "ItemDef" is deprecated in Define-XML 2.1, and we need to use the child "def:Origin" child element instead.

In order to do so, click the "Edit" icon left to "LB.LBTESTCD", leading to:

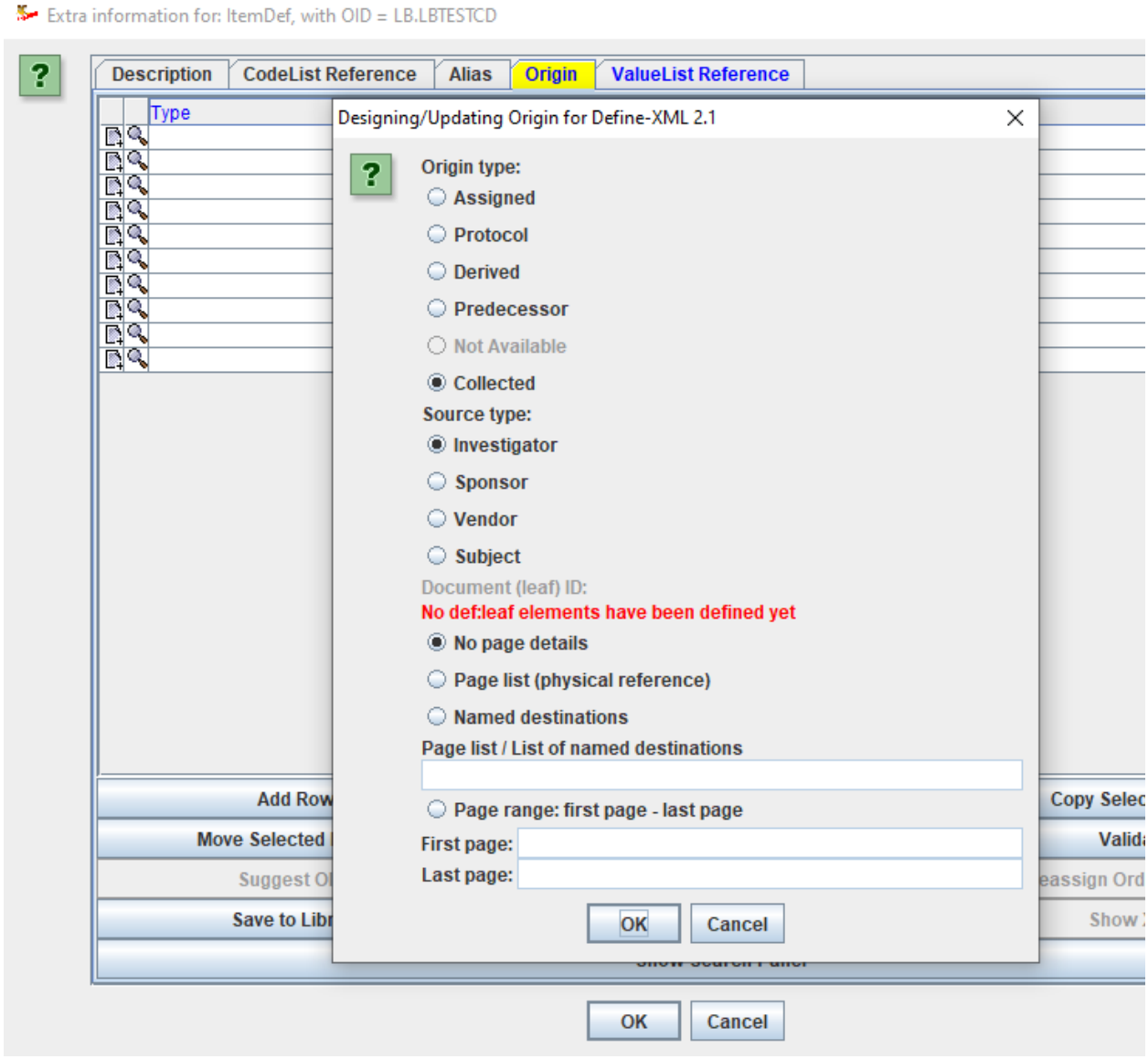


where we also find a tab "Origin". When it is selected:

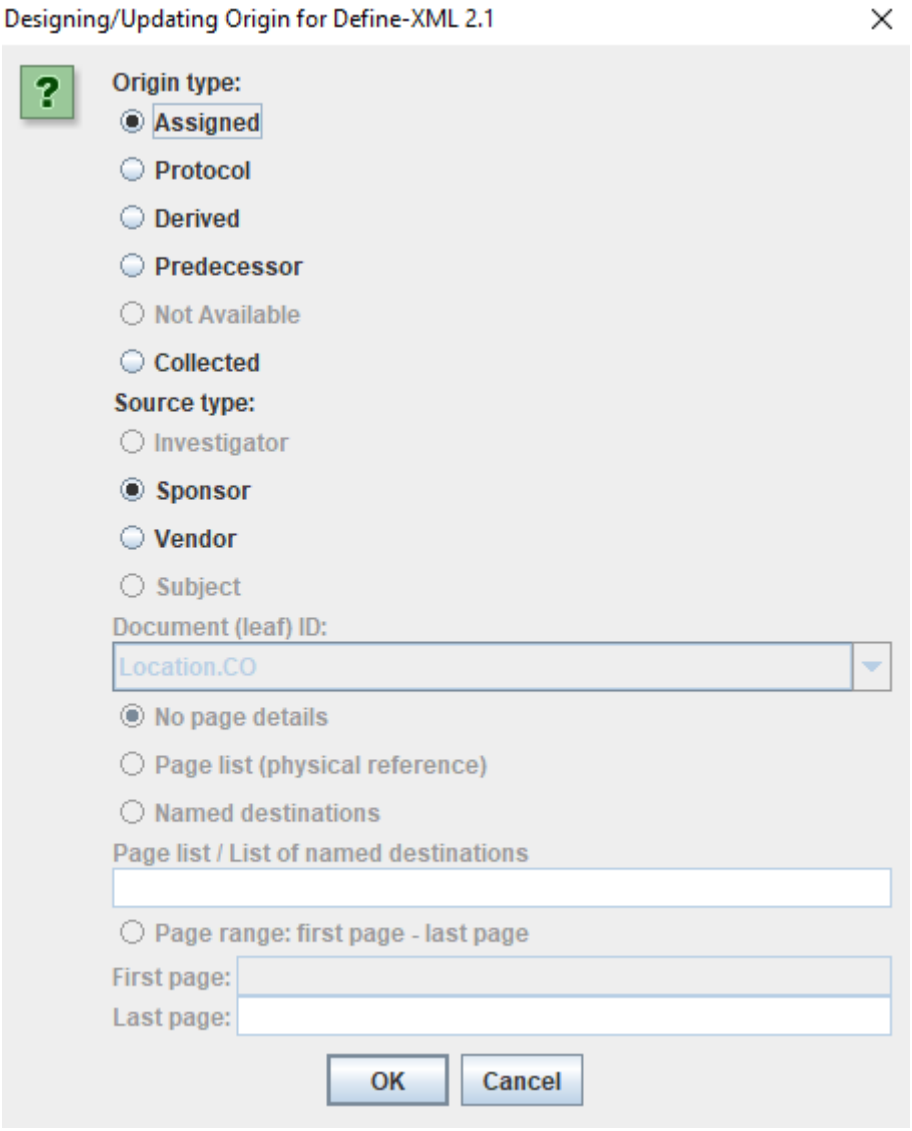


By default (we will later see how this can be switched off) when we then click in the "Type" cell of the first row, a "wizard" is started.
In many cases, the system will show a "Wizard", guiding the user when the Define-XML standard for that piece of information is a bit more complicated. For example, for "def:Origin" in Define-XML 2.1, there are dependencies between "Origin Type" and "Source". See section 4.3.2 "Origin/Source/Traceability Considerations" of the Define-XML 2.1 specification.

So when the first "Type" cell is clicked, the "wizard" is started:



For SDTM, the wizard sets the default combination to Origin-Type="Collected" and Source-type="Investigator". In most cases however (but now always), the "test code" will be assigned by a mapper. So when we select "Assigned", the available choices change into:



and the choice "Sponsor" is suggested.
The other possible choice is "Vendor", to be used when an external vendor did the assignment.
Also notice that the whole section about page numbers is automatically disabled.

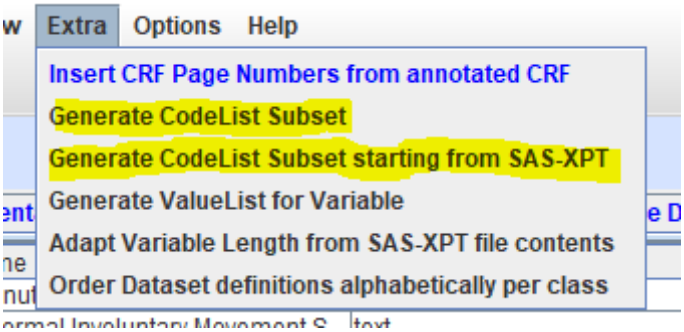
Subsetting CodeLists

The CDISC Controlled Terminology has some very long codelists, especially for --TESTCD and --TEST variables, but also e.g. for specimens (SPECTYPE codelist - C78734) and for units (UNIT codelist - C71620).
In case one has generated the prototype define.xml starting from a set of SAS-XPT files, and the checkbox "Try to create subset CodeLists from XPT content ..." was checked, such subset codelists should already have been created, at least for the entries in the "subsetcodelistvariables.dat" file, and one will probably only want to extend them, or designate which of the items in such codelists are "extended".
For example, the LBTESTCD codelist contains almost 2500 items[8], and one surely do not wants to submit a define.xml with all these 2500 terms. Normally, if one has generated the mappings with "SDTM-savvy" user-friendly software such as the SDTM-ETL software, the "cleaned" define.xml will already have taken care, but we see that still often (statistical) software is used to generate SDTM datasets, without generating a synchronized define.xml at the same time. In such a case, one should have look into the annotated CRF and look which values of LBTESTCD have been used in the annotations. If one already has an, even temporary, LB XPT dataset, one can also generate subset codelist from that XPT file.

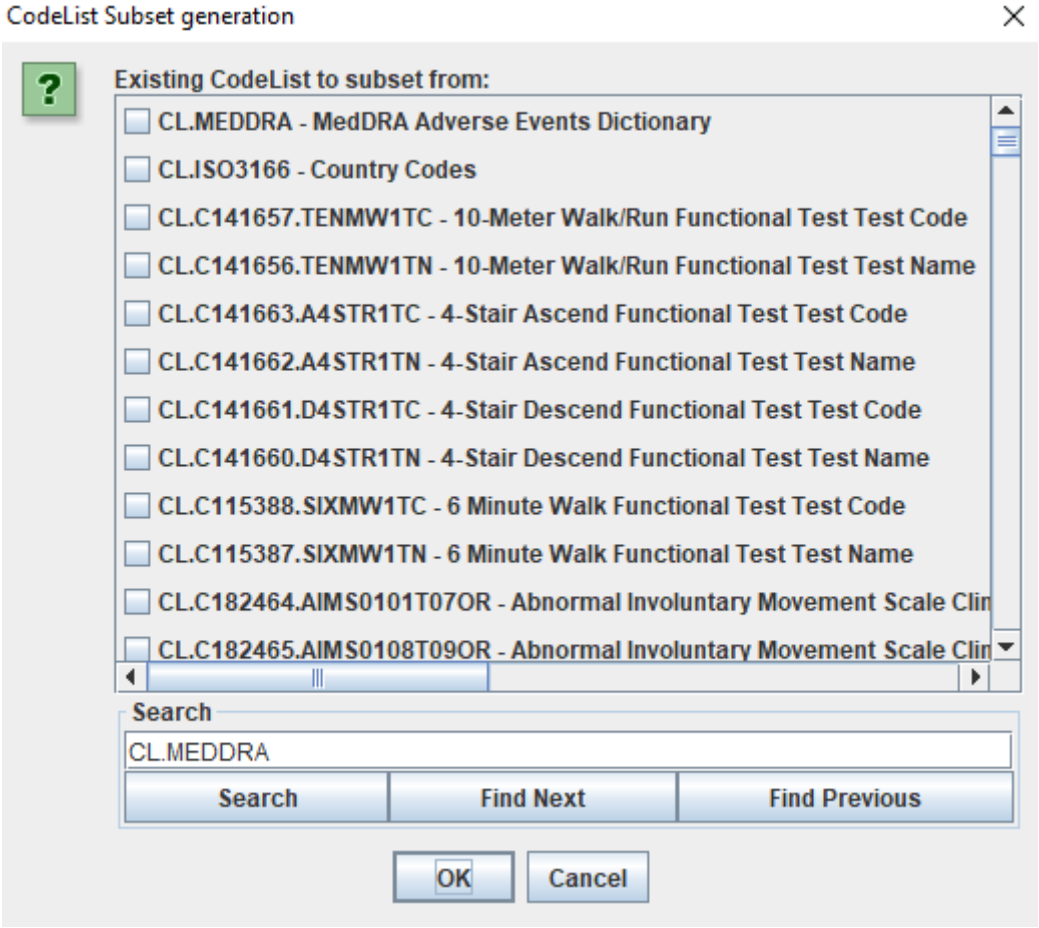
---

[8] The exact number for codelist version 2025-09-26 is 2474.

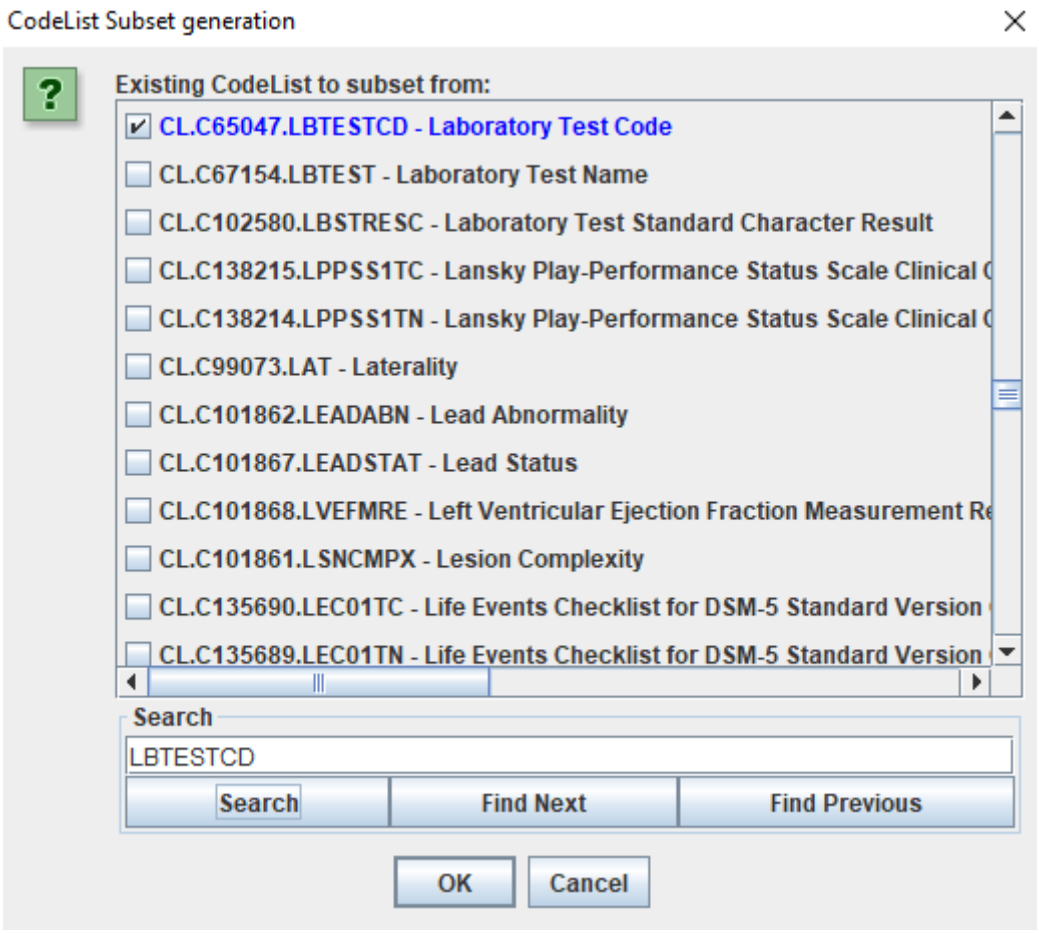
In order to subset a codelist like the one for LBTESTCD, use the menu "Extra":



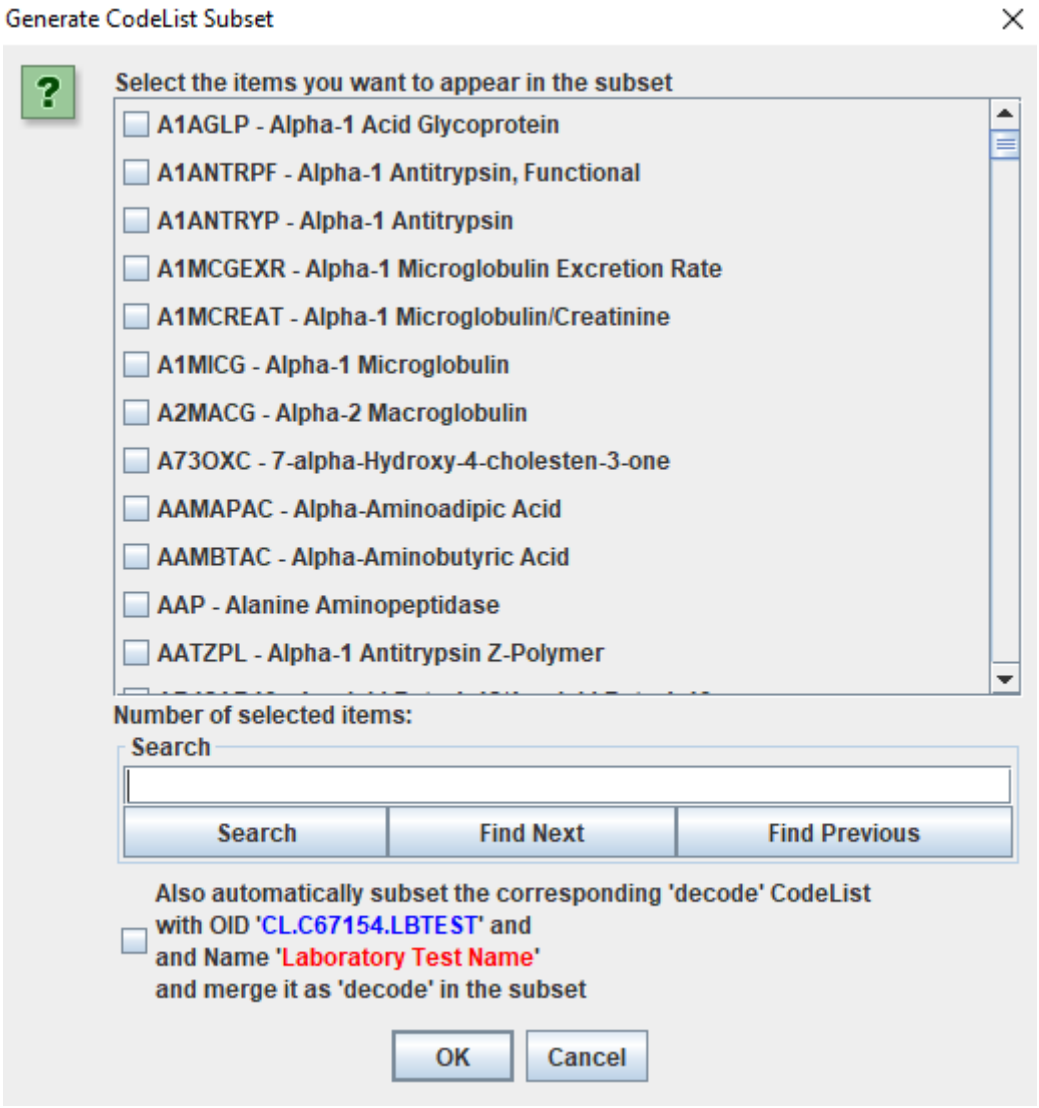
We have 2 features here to generate a codelist subset.
Let us start with the "Generate CodeList Subset". When we select it, this leads to a dialog where we can select for which codelist we want to generate a subset:



Using the "Search", we can quickly find the one for LBTESTCD:

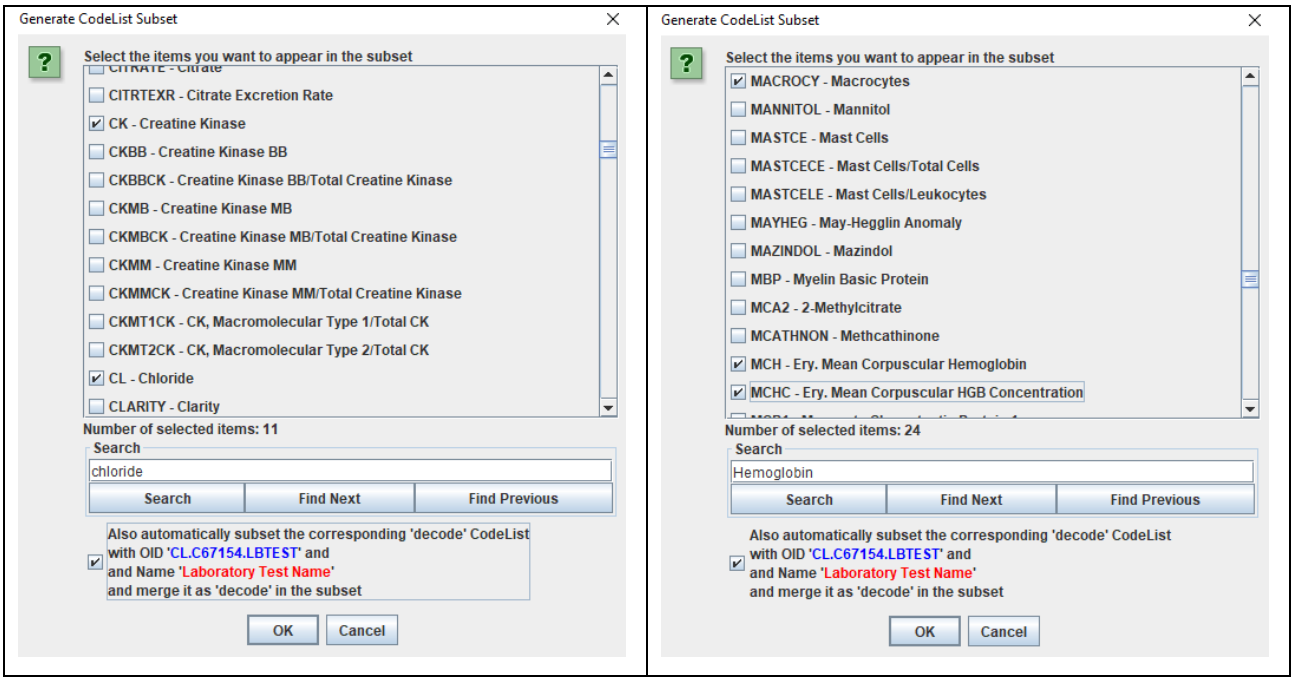


and clicking "OK" leads to:



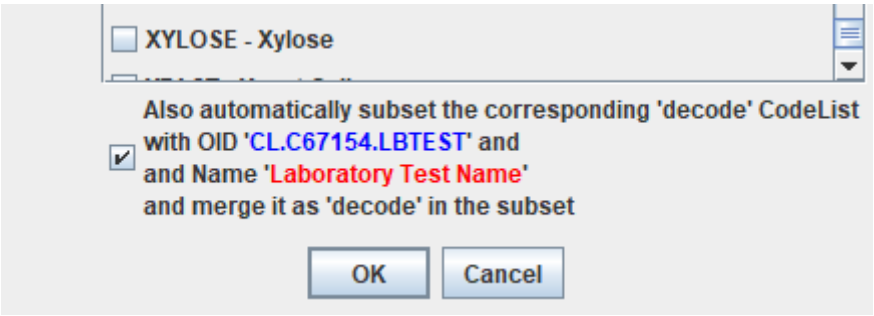
We can then compare this list with the annotations on the CRF[9] and check the checkboxes for the codes that we want to have in the subset. For example:

[9] We are currently working on a feature to automatically retrieve the LBTESTCD annotations from the Annotated CRF in PDF form.



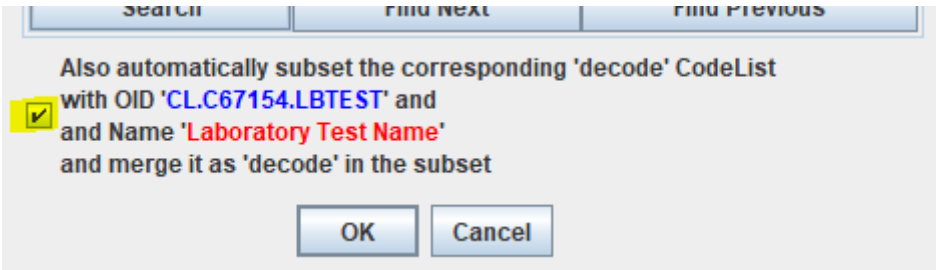
There is no problem if one forgets one or makes an error, as the list can always be corrected later.

As this moment, it is also wise to check the checkbox "Also automatically subset the corresponding 'decode' CodeList

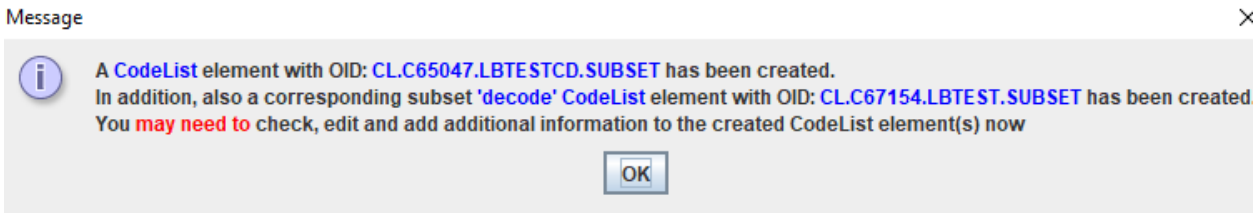


as this will take care that the corresponding codelist for LBTEST (test name) is generated.

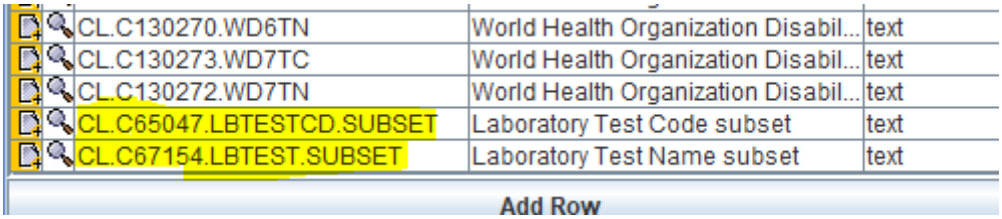
When all found and checked, clicking OK then e.g. leads to:



allowing the OID of the subset codelist and its name to be changed, if desired.
Clicking "OK" then leads to:



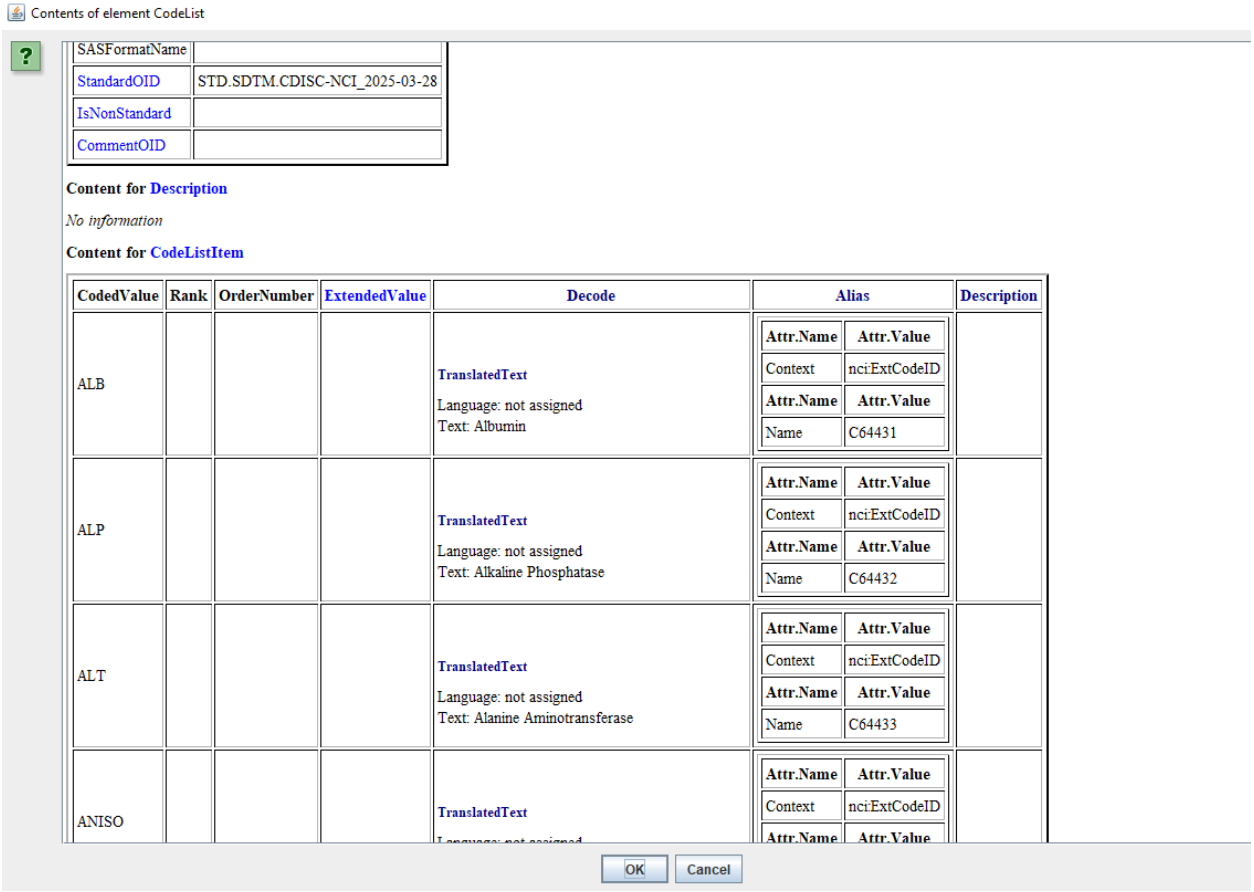
If after that, one scrolls to the bottom of the list with codelists, one finds that 2 "subset" codelists have been added:



CL.C130270.WD6TN	World Health Organization Disabil...	text
CL.C130273.WD7TC	World Health Organization Disabil...	text
CL.C130272.WD7TN	World Health Organization Disabil...	text
CL.C65047.LBTESTCD.SUBSET	Laboratory Test Code subset	text
CL.C67154.LBTEST.SUBSET	Laboratory Test Name subset	text

Add Row

and if we click on the "View" (magnifying glass) icon for the first one, we e.g. get:



Contents of element CodeList

SASFormatName	
StandardOID	STD.SDTM.CDISC-NCI_2025-03-28
IsNonStandard	
CommentOID	

Content for Description

No information

Content for CodeListItem

CodedValue	Rank	OrderNumber	ExtendedValue	Decode	Alias	Description
ALB				TranslatedText Language: not assigned Text: Albumin	Attr.Name: Context / Attr.Value: nci:ExtCodeID Attr.Name: Name / Attr.Value: C64431	
ALP				TranslatedText Language: not assigned Text: Alkaline Phosphatase	Attr.Name: Context / Attr.Value: nci:ExtCodeID Attr.Name: Name / Attr.Value: C64432	
ALT				TranslatedText Language: not assigned Text: Alanine Aminotransferase	Attr.Name: Context / Attr.Value: nci:ExtCodeID Attr.Name: Name / Attr.Value: C64433	
ANISO				TranslatedText	Attr.Name: Context / Attr.Value: nci:ExtCodeID	

OK Cancel

and for the LBTEST-subset:

Contents of element CodeList

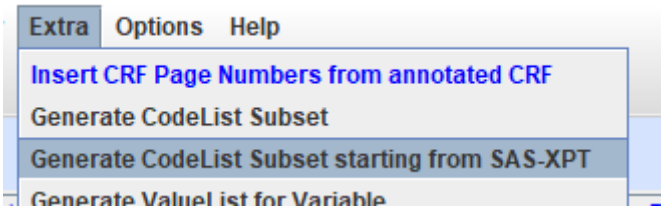
No information

Content for EnumeratedItem

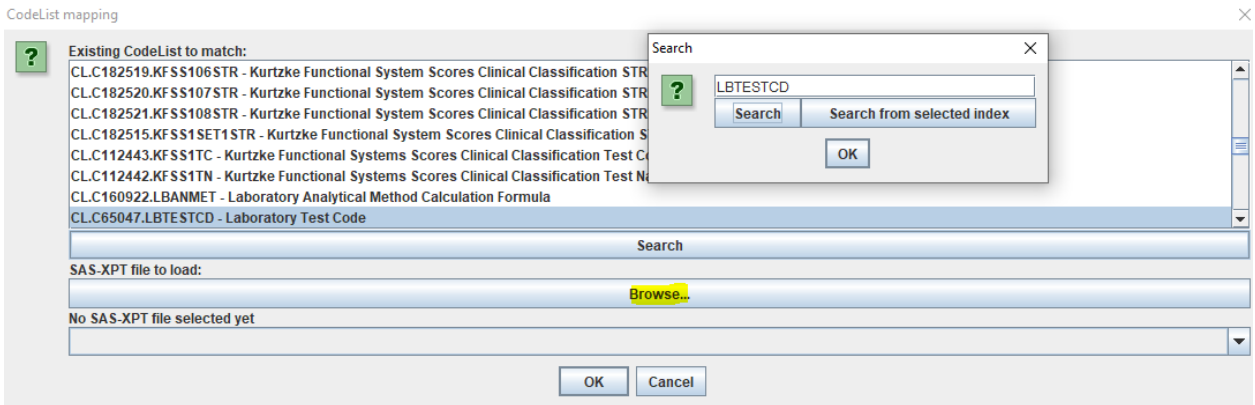
CodedValue	Rank	OrderNumber	ExtendedValue	Alias	Description
Albumin				Attr.Name: Attr.Value Context: nci:ExtCodeID Attr.Name: Attr.Value Name: C64431	
Alkaline Phosphatase				Attr.Name: Attr.Value Context: nci:ExtCodeID Attr.Name: Attr.Value Name: C64432	
Alanine Aminotransferase				Attr.Name: Attr.Value Context: nci:ExtCodeID Attr.Name: Attr.Value Name: C64433	
Anisocytes				Attr.Name: Attr.Value Context: nci:ExtCodeID Attr.Name: Attr.Value Name: C74797	
Aspartate Aminotransferase				Attr.Name: Attr.Value Context: nci:ExtCodeID Attr.Name: Attr.Value Name: C64467	

If we did something wrong or forgot something, we can still edit the list by clicking the "Edit" icon, and then make corrections.

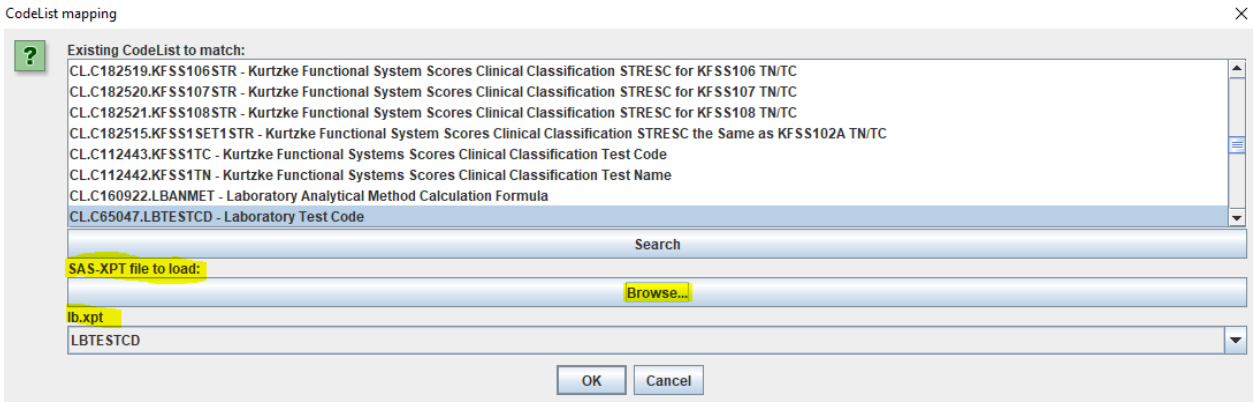
If we already have an XPT dataset for LB, we can also retrieve the subset values from that. In order to do so, we need to use the menu "Extra - Generate CodeList subset starting from SAS-XPT":



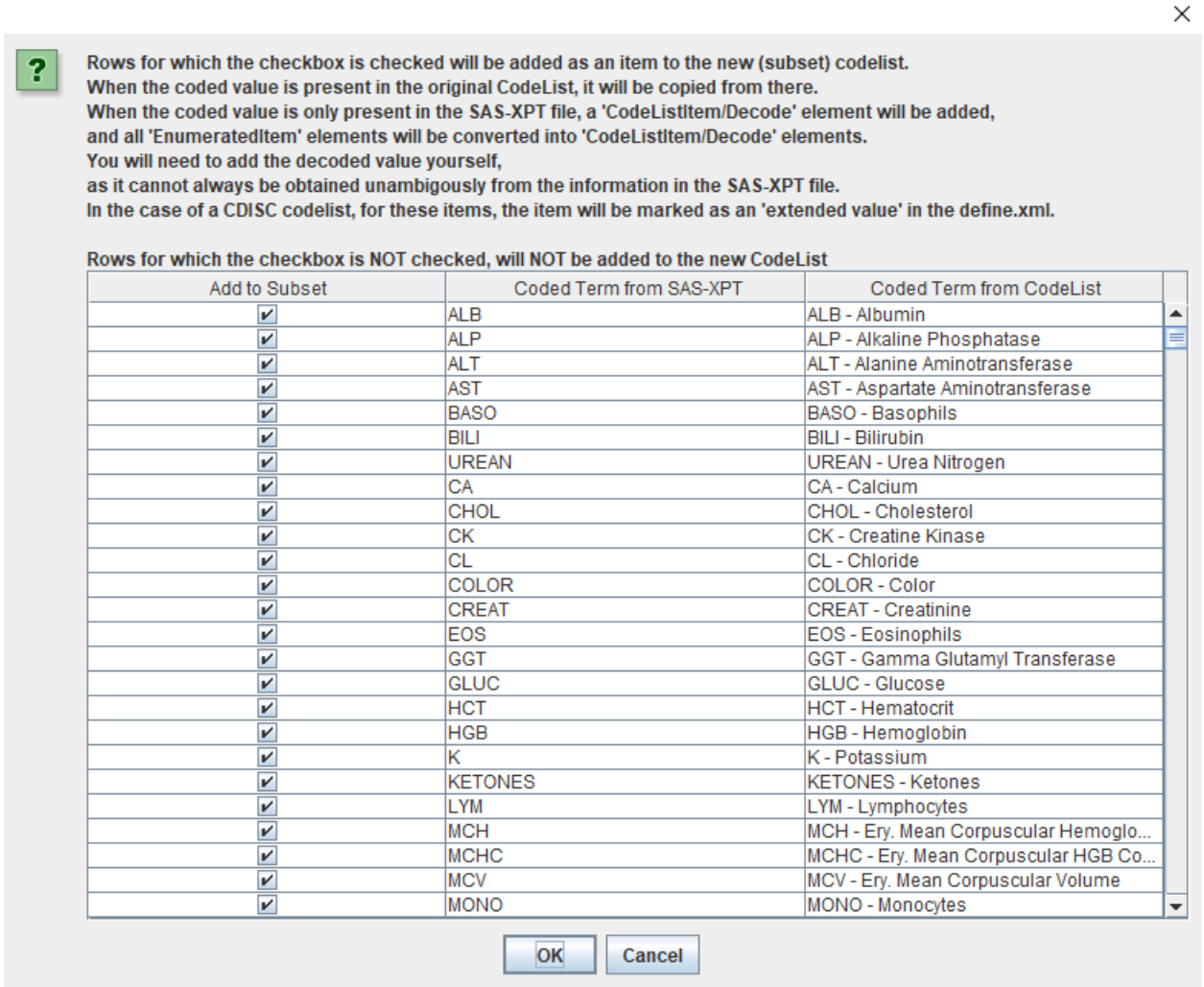
It is then asked for which codelist we want to generate a subset:



and provide a SAS-XPT file by using the button "Browse…", e.g. leading to:



and after "OK", we get the proposed list:

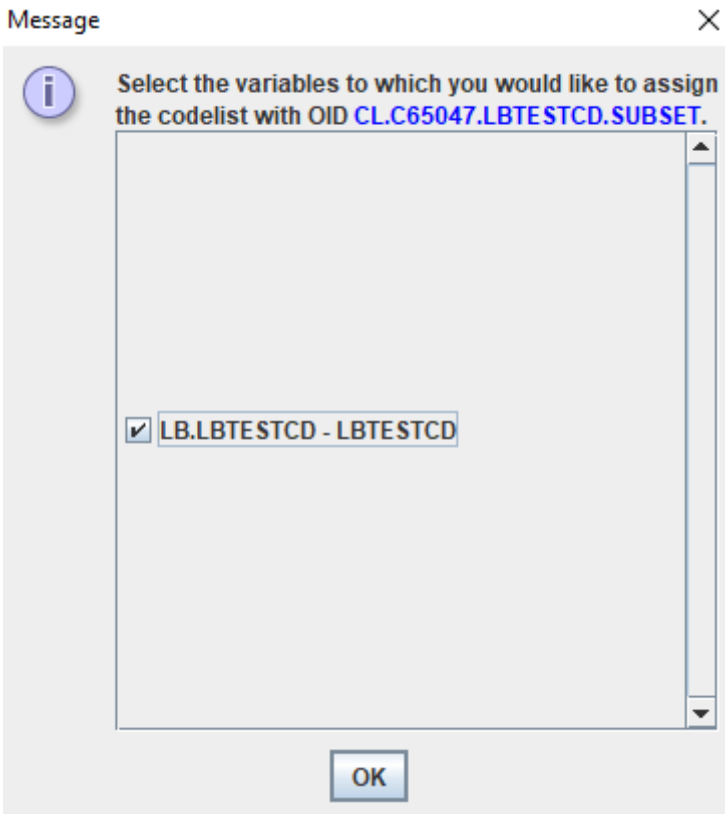


If necessary, we can then still make corrections, or decide to not have some items included in the subset codelist. If something is missing, we can even add it now. For example, if "Blood Group" was planned to be collected, but there is no data for it in the SAS-XPT file, we can still add it by checking the checkbox for it:

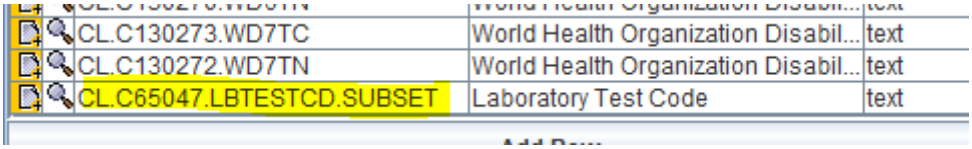
Rows for which the checkbox is NOT checked, will NOT be added to the new CodeList

Add to Subset	Coded Term from SAS-XPT	Coded Term from CodeList
☑	UROBIL	UROBIL - Urobilinogen
☑	VITB12	VITB12 - Vitamin B12
☑	WBC	WBC - Leukocytes
☑	ANISO	ANISO - Anisocytes
☑	POIKILO	POIKILO - Poikilocytes
☑	MACROCY	MACROCY - Macrocytes
☐		A1AGLP - Alpha-1 Acid Glycoprotein
☐		A1ANTRPF - Alpha-1 Antitrypsin, Functio...
☐		A1ANTRYP - Alpha-1 Antitrypsin
☐		A1MCGEXR - Alpha-1 Microglobulin Excr...
☐		A1MCREAT - Alpha-1 Microglobulin/Crea...
☐		A1MICG - Alpha-1 Microglobulin
☐		A2MACG - Alpha-2 Macroglobulin
☐		A73OXC - 7-alpha-Hydroxy-4-cholesten-...
☐		AAMAPAC - Alpha-Aminoadipic Acid
☐		AAMBTAC - Alpha-Aminobutyric Acid
☐		AAP - Alanine Aminopeptidase
☐		AATZPL - Alpha-1 Antitrypsin Z-Polymer
☐		AB42AB40 - Amyloid Beta 1-42/Amyloid ...
☐		ABFBCA - AB-FUBINACA
☐		ABNCE - Abnormal Cells
☐		ABNCECE - Abnormal Cells/Total Cells
☐		ABNCELE - Abnormal Cells/Leukocytes
☑		ABO - ABO Blood Group
☐		ABOA1 - ABO A1 Subtype

Clicking "OK" then leads to the sub-codelist to be generated, and it is asked whether we want to have this generated subset codelist to be assigned already to the variable LBTESTCD:



Again, an information message is being shown, and the codelist is added to the list:



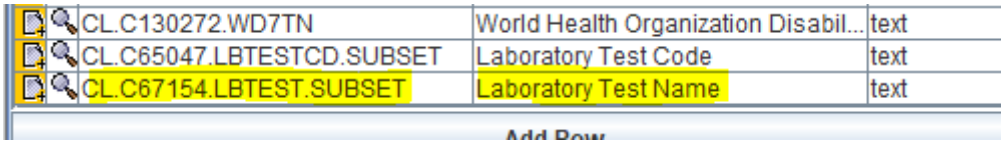
We can then do the same for LBTEST, again using the menu "Extra - Generate CodeList subset starting from SAS-XPT", e.g. leading to:

Rows for which the checkbox is checked will be added as an item to the new (subset) codelist.
When the coded value is present in the original CodeList, it will be copied from there.
When the coded value is only present in the SAS-XPT file, a 'CodeListItem/Decode' element will be added,
and all 'EnumeratedItem' elements will be converted into 'CodeListItem/Decode' elements.
You will need to add the decoded value yourself,
as it cannot always be obtained unambigously from the information in the SAS-XPT file.
In the case of a CDISC codelist, for these items, the item will be marked as an 'extended value' in the define.xml.

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
☑	Albumin	Albumin
☑	Alkaline Phosphatase	Alkaline Phosphatase
☑	Alanine Aminotransferase	Alanine Aminotransferase
☑	Aspartate Aminotransferase	Aspartate Aminotransferase
☑	Basophils	Basophils
☑	Bilirubin	Bilirubin
☑	Urea Nitrogen	Urea Nitrogen
☑	Calcium	Calcium
☑	Cholesterol	Cholesterol
☑	Creatine Kinase	Creatine Kinase
☑	Chloride	Chloride
☑	Color	Color
☑	Creatinine	Creatinine
☑	Eosinophils	Eosinophils
☑	Gamma Glutamyl Transferase	Gamma Glutamyl Transferase
☑	Glucose	Glucose

and ultimately to having the LBTEST subset-codelist being generated and added to the list, and being assigned to LBTEST:



CL.C130272.WD7TN	World Health Organization Disabil...	text
CL.C65047.LBTESTCD.SUBSET	Laboratory Test Code	text
CL.C67154.LBTEST.SUBSET	Laboratory Test Name	text

Add Row

We can visualize the result by clicking the button "HTML View", and scroll down LBTESTCD:

LBGRPID - [Edit]		Group ID	text	Identifier	80		[Add]
LBREFID - [Edit]		Specimen ID	text	Identifier	80		[Add]
LBSPID - [Edit]		Sponsor-Defined Identifier	text	Identifier	80		[Add]
LBTESTCD - [Edit]		Lab Test or Examination Short Name	text	Topic	8	Laboratory Test Code - [Edit]	[Add]
LBTEST - [Edit]		Lab Test or Examination Name	text	Synonym Qualifier	40	Laboratory Test Name - [Edit]	[Add]
LBTSTCND - [Edit]		Test Condition	text	Variable Qualifier	38	Test Condition Response - [Edit]	[Add]

and then e.g. clicking the link "Laboratory Test Code", it displays the codelist we have just generated:

Laboratory Test Code [C65047] [CDISC/NCI SDTM 2025-03-28]

[Edit]

Permitted Value (Code)	Display Value (Decode)
ALB	Albumin
ALP	Alkaline Phosphatase
ALT	Alanine Aminotransferase
AST	Aspartate Aminotransferase
BASO	Basophils
BILI	Bilirubin
UREAN	Urea Nitrogen
CA	Calcium
CHOL	Cholesterol
CK	Creatine Kinase
CL	Chloride
COLOR	Color
CREAT	Creatinine
EOS	Eosinophils
GGT	Gamma Glutamyl Transferase
GLUC	Glucose

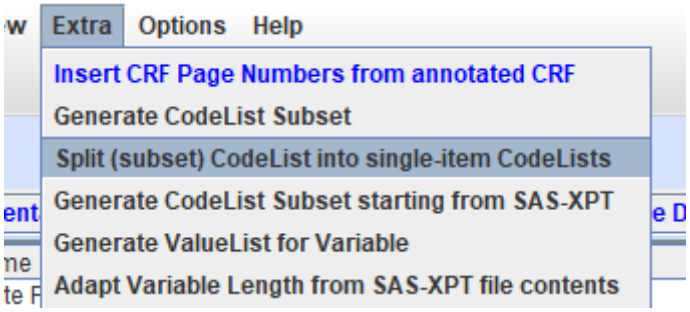
Please do not worry about that the original codelist with the almost 2500 items is still in our define.xml. We will later automatically have it removed when generating a "cleaned" define.xml.

Generating Single-Item Subset CodeLists

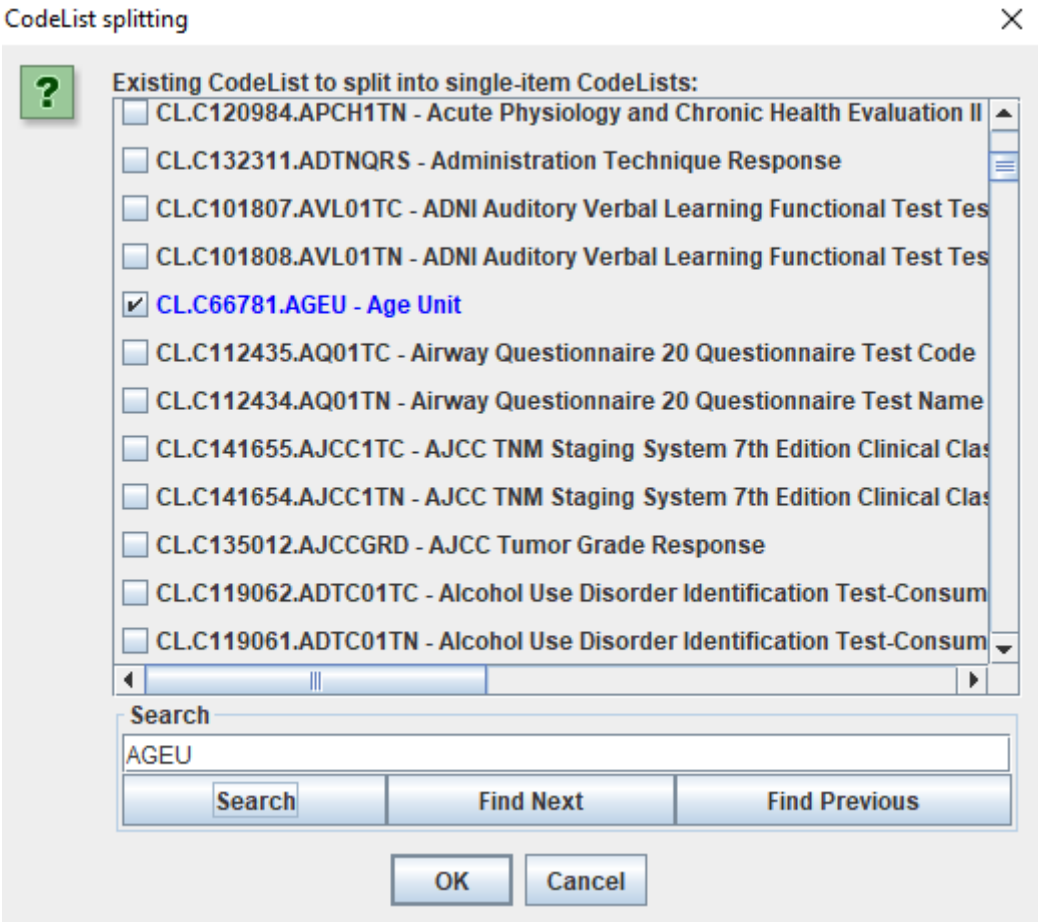
In some cases, one has created a (subset) codelist and wants to further create several subset codelists containing one item each. This can e.g. be the case when one wants to make a ValueList to state which unit was used for which test (xxSTRESU or xxORRESU for SDTM/SEND as function of xxTESTCD, or AVALU as function of PARAMCD). In SDTM/SEND one will however mostly use it on already existing subset codelists.

In order to generate a set of subset codelists from another (usually subset) codelist, use the menu "Extra - Split".

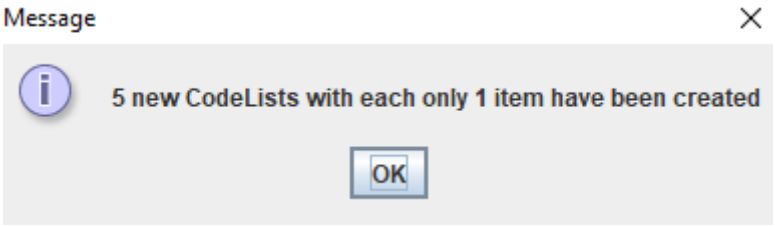
For example, if one would create subset codelists for each individual item of the "AGEU" (Age Units) codelist one starts with the menu "Extra - Split (subset) CodeList into single-item CodeLists":



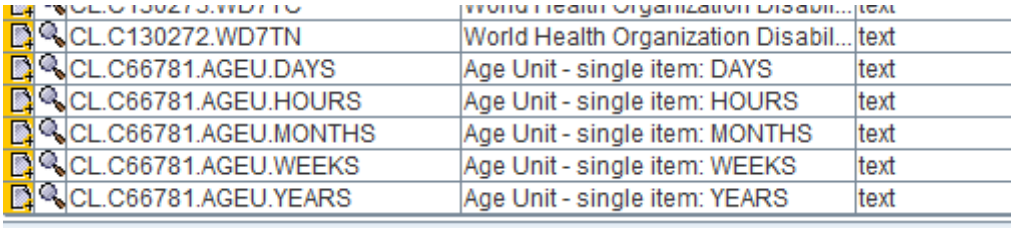
The system then asks us which codelist should be used as the base for splitting:



Clicking OK starts the process, with the final message:

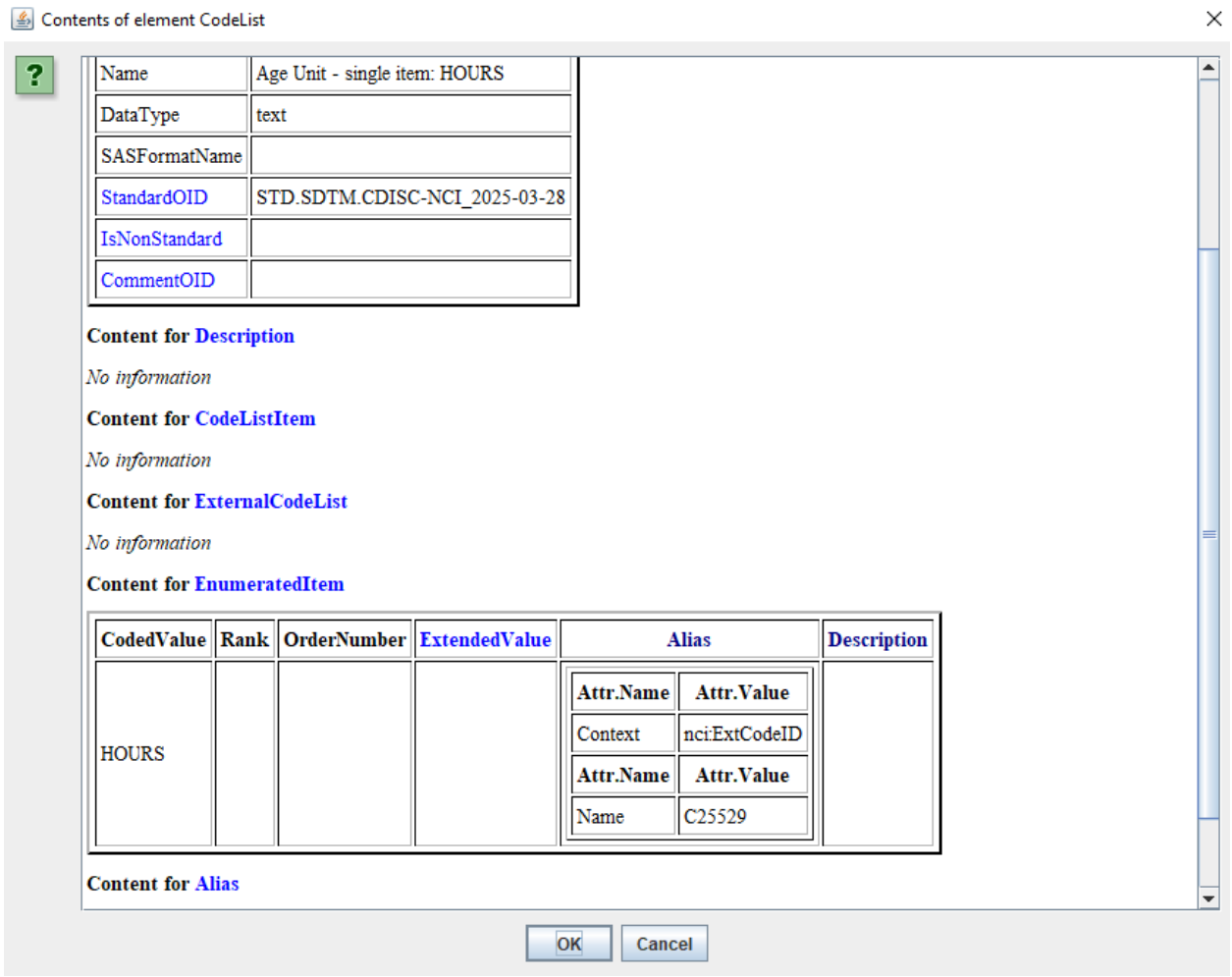


and if we then look at the list with codelists, we find:



CL.C130272.WD7TN	World Health Organization Disabil...	text
CL.C66781.AGEU.DAYS	Age Unit - single item: DAYS	text
CL.C66781.AGEU.HOURS	Age Unit - single item: HOURS	text
CL.C66781.AGEU.MONTHS	Age Unit - single item: MONTHS	text
CL.C66781.AGEU.WEEKS	Age Unit - single item: WEEKS	text
CL.C66781.AGEU.YEARS	Age Unit - single item: YEARS	text

and then click on the "View" (magnifying glass) icon e.g. for "CL.C66781.AGEU.HOURS":



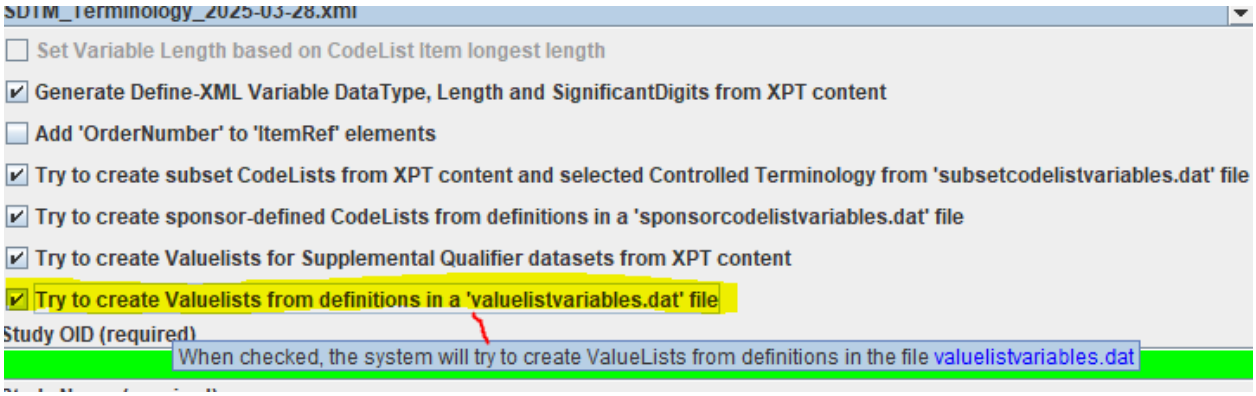
Please notice that the existing "AGEU" codelist is still present.
We can then use these single-item codelists for a ValueList stating which age unit for which case (e.g. based on birthdate) was used.

Generating ValueLists

ValueLists can be set up in several different ways:

- by first defining define.xml "WhereClauses" using the "WhereClause Definitions" tab, and then defining "ValueLists" using the "ValueList Definitions" tab, and for each added "ValueList", adding a "WhereClause".

- When starting from a set of SAS-XPT files, reading "WhereClauses" from a file "valuelistvariables.dat" and having them executed during generation of the define.xml from the XPT files.
In order to use this, one should have the checkbox "Try to create ValueLists from definitions in a 'valuelistvariables.dat file" checked:



The "valuelistvariables.dat" file contains "WhereClause" definitions in a "human-friendly" format. For example:

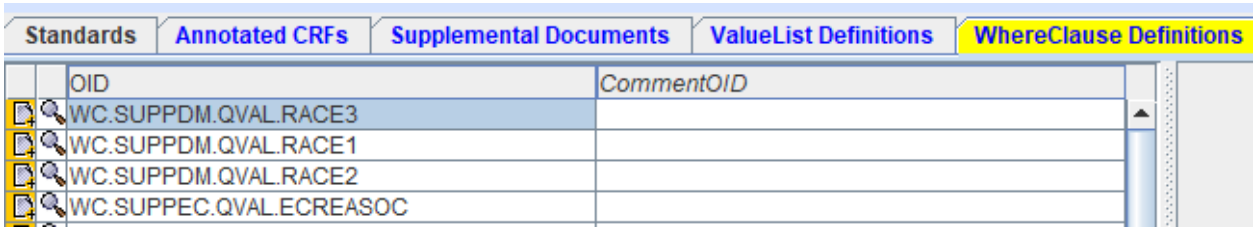
```
FTORRES WHERE FTTESTCD EQ TUG0101

LBORRES WHERE LBTESTCD EQ ALP
LBORRES WHERE LBTESTCD EQ CA
LBORRES WHERE LBTESTCD EQ CREAT
LBORRES WHERE LBTESTCD EQ HCGQUAL
LBORRES WHERE LBTESTCD EQ HCGQUAN
LBORRES WHERE LBTESTCD EQ PHOS
LBORRES WHERE LBTESTCD EQ PTHI
LBORRES WHERE LBTESTCD EQ RUCA
LBORRES WHERE LBTESTCD EQ VITD2T
LBORRES WHERE LBTESTCD EQ VITD3T
LBORRES WHERE LBTESTCD EQ VITDAT
LBORRES WHERE LBTESTCD EQ RUCREAT
LBORRES WHERE LBTESTCD EQ RUPCRTR
LBORRES WHERE LBTESTCD EQ RUPROT
LBORRES WHERE LBTESTCD EQ HCG
LBORRES WHERE LBTESTCD EQ ORT8814
LBORRES WHERE LBTESTCD EQ ORT8815
LBORRES WHERE LBTESTCD EQ ORT8828
LBORRES WHERE LBTESTCD EQ ORT8829
LBORRES WHERE LBTESTCD EQ CACREAT
LBORRES WHERE LBTESTCD EQ CAEXR
LBORRES WHERE LBTESTCD EQ CREATEXR
```

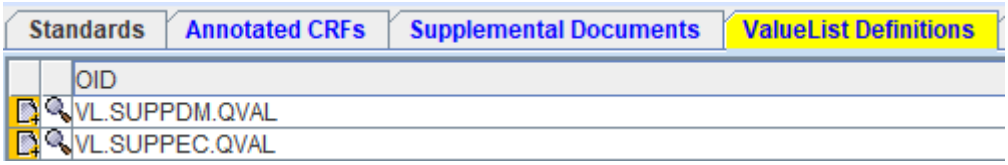
Lines in this file starting with a "#" are "commented out" and will be ignored.
Also have a look at the checkbox "Try to create ValueLists for Supplemental Qualifier datasets from XPT content". Having such may be a requirement of some regulatory authorities such as the FDA. An example is provided in Section 4.5.2.3 of the Define-XML specification. The specification states in Section 5.3.9:
"*Business Rule: For SDTM SUPPQUAL datasets, a def:ValueListDef element must be provided to describe the QVAL variable*".

ValueLists for Supplemental Qualifiers (SDTM)

When the checkbox "Try to create ValueLists for Supplemental Qualifier datasets from XPT content" is checked, we will find automatically generated entries such as:



and for the ValueLists:



When using the "HTML View" button, we e.g. find:

SUPPDM (Supplemental Qualifiers for DM) - [SDTMIG 3.4]

Variable	ValueList Where Condition	Label / Description	Type	Role	Length or Display Format	Controlled Terms or ISO Format	Origin/Sou
STUDYID - [Edit]		Study Identifier	text		12		[Add]
RDOMAIN - [Edit]		Related Domain Abbreviation	text		2	SDTM Domain Abbreviation - [Edit]	[Add]
USUBJID - [Edit]		Unique Subject Identifier	text		8		[Add]
IDVAR - [Edit]		Identifying Variable	text		1		[Add]
IDVARVAL - [Edit]		Identifying Variable Value	text		1		[Add]
QNAM - [Edit]		Qualifier Variable Name	text		5		[Add]
QLABEL - [Edit]		Qualifier Variable Label	text		6		[Add]
QVAL - [Edit] [Remove/Replace ValueList] [Edit ValueList]		Data Value	text		25		[Add]
ValueList variable for QVAL - [Edit]	[Edit] QNAM EQ RACE3		text		5	CodeList for QVAL for QNAM = RACE3 in dataset SUPPDM - [Edit]	- [Edit]
ValueList variable for QVAL - [Edit]	[Edit] QNAM EQ RACE1	Race 1	text		5	CodeList for QVAL for QNAM = RACE1 in dataset SUPPDM - [Edit]	- [Edit]
ValueList variable for QVAL - [Edit]	[Edit] QNAM EQ RACE2		text		25	CodeList for QVAL for QNAM = RACE2 in dataset SUPPDM - [Edit]	- [Edit]

stating that there is a valuelist for QNAM=RACE3. When clicking the hyperlink "CodeList for QVAL for QNAM = RACE3", we get:

CodeList for QVAL for QNAM = RACE3 in dataset SUPPDM [Non Standard]

[Edit]

Permitted Value (Code)
WHITE

CodeList for QVAL for QNAM = RACE1 in dataset SUPPDM [Non Standard]

[Edit]

Permitted Value (Code)
ASIAN

CodeList for QVAL for QNAM = RACE2 in dataset SUPPDM [Non Standard]

[Edit]

Permitted Value (Code)
BLACK OR AFRICAN AMERICAN

of course, this doesn't make sense, so we will better delete such valuelists with their where-clauses.

There is another however, also a ValueList for ECREASOC

QLABEL - [Edit]		Qualifier Variable Label	text		22		[A
QVAL - [Edit] [Remove/Replace ValueList] [Edit ValueList]		Data Value	text		21		[A
ValueList variable for QVAL - [Edit]	[Edit] QNAM EQ ECREASOC	Reason for Occur Value	text		21	CodeList for QVAL for QNAM = ECREASOC in dataset SUPPEC - [Edit]	-
QORIG - [Edit]		Origin	text		9		[A

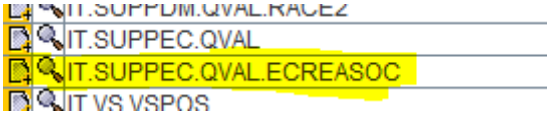
stating that a codelist was automatically generated, so when we click the hyperlink:

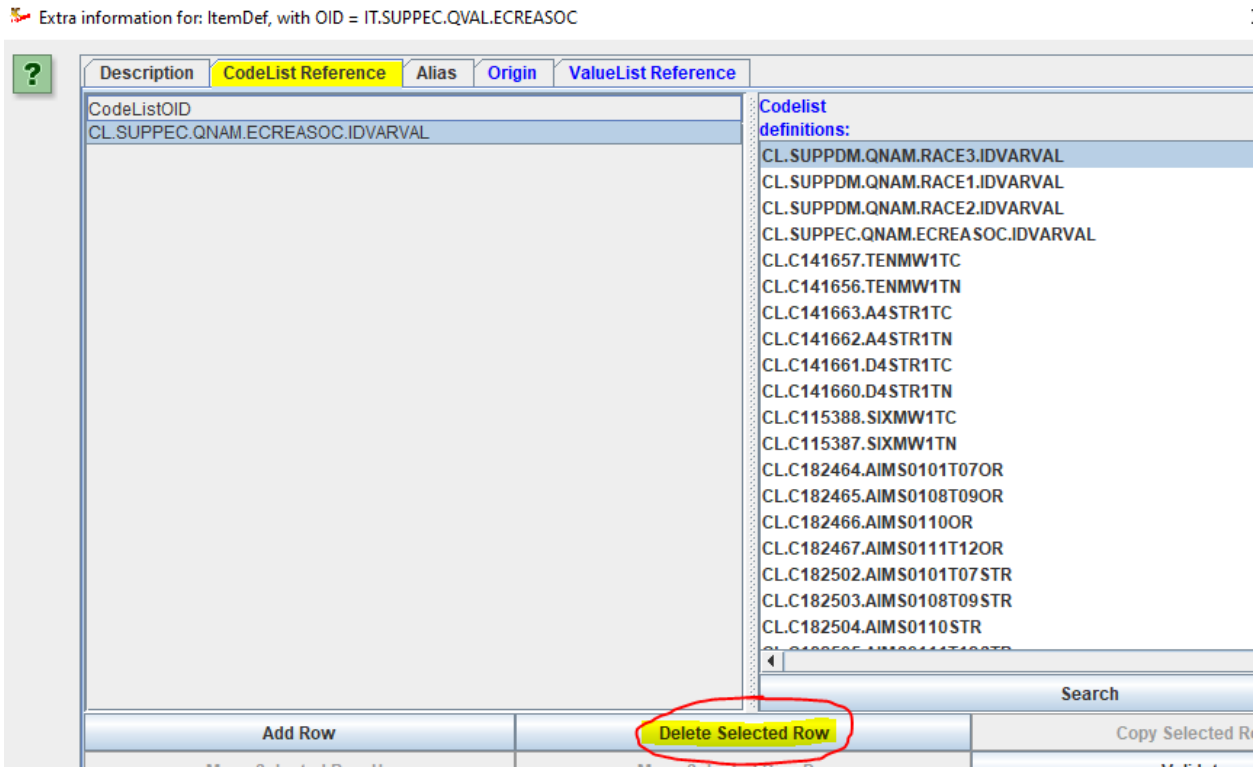
CodeList for QVAL for QNAM = ECREASOC in dataset SUPPEC [Non Standard]

[Edit]

Permitted Value (Code)
INVESTIGATOR DECISION

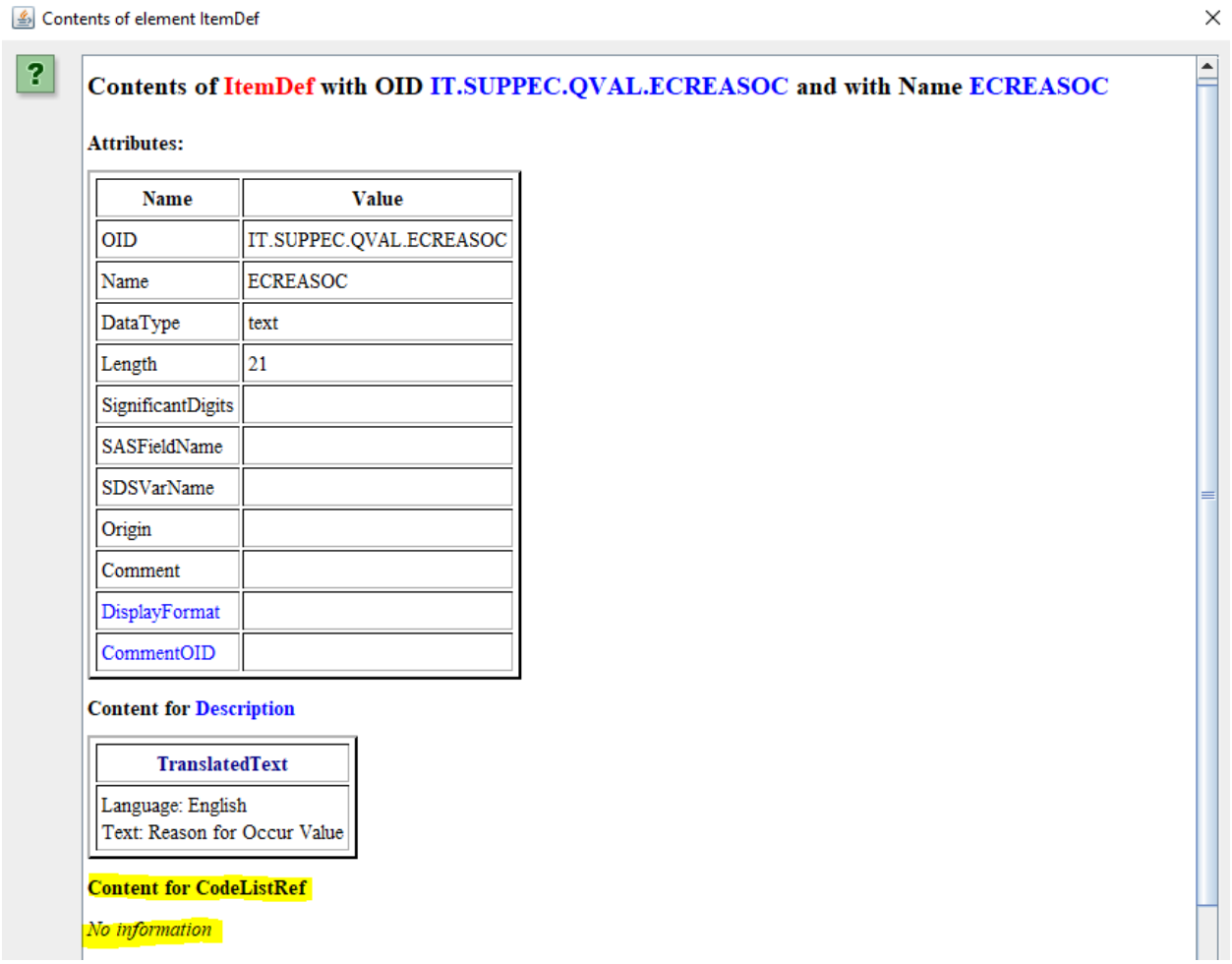
showing there is only "investigator" decision.
It then is wise to have a look at the CRF: when the field for "Reason Occurrence" is free text, we surely should no have a codelist nor valuelist for it, and we should remove the codelist reference and the codelist definition itself from the define.xml. This can easily be done by going to the item definition in the "Variable Definitions" tab, select the one, click the "Edit" icon, and then remove the codelist reference, i.e.:



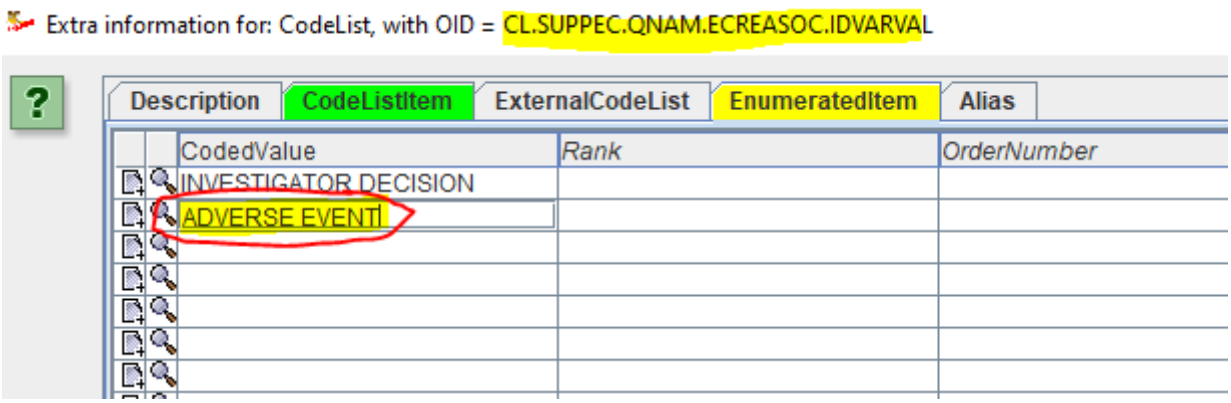


Later we will see that there is an easier way by starting from the HTML view itself.

The ValueList for QVAL for QNAM=ECREASOC then will still be present, as required, but the codelist refererence and the codelist itself have been removed:



IF, however, the field on the CRF is not free text, but has more choices than "Investigator Decision", like "Adverse Event", but the latter was never used and thus didn't make it into the XPT dataset, then we need to add these extra choices to the codelist. For example, if the codelist needs to be extended with:



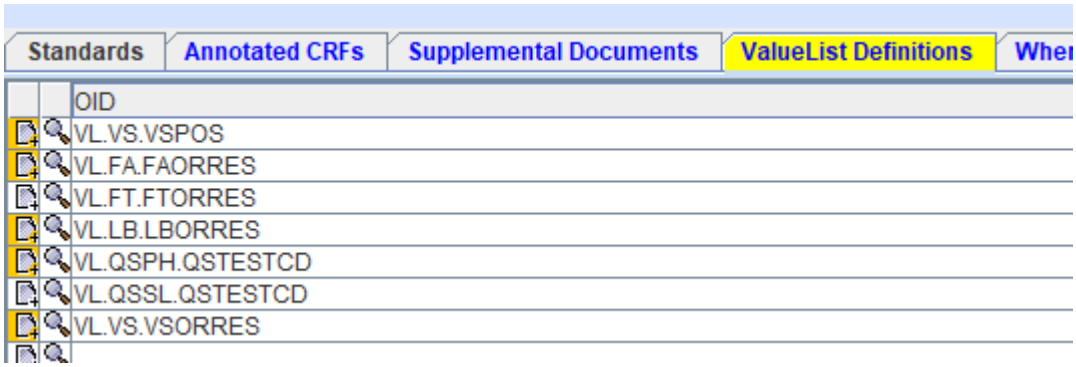
This demonstrates once again how dangerous it can be to generate a define.xml starting from the SDTM/SEND/ADaM XPT datasets! Define-XML is about what was planned, and not only about what was collected!

ValueLists for other variables from SAS-XPT files

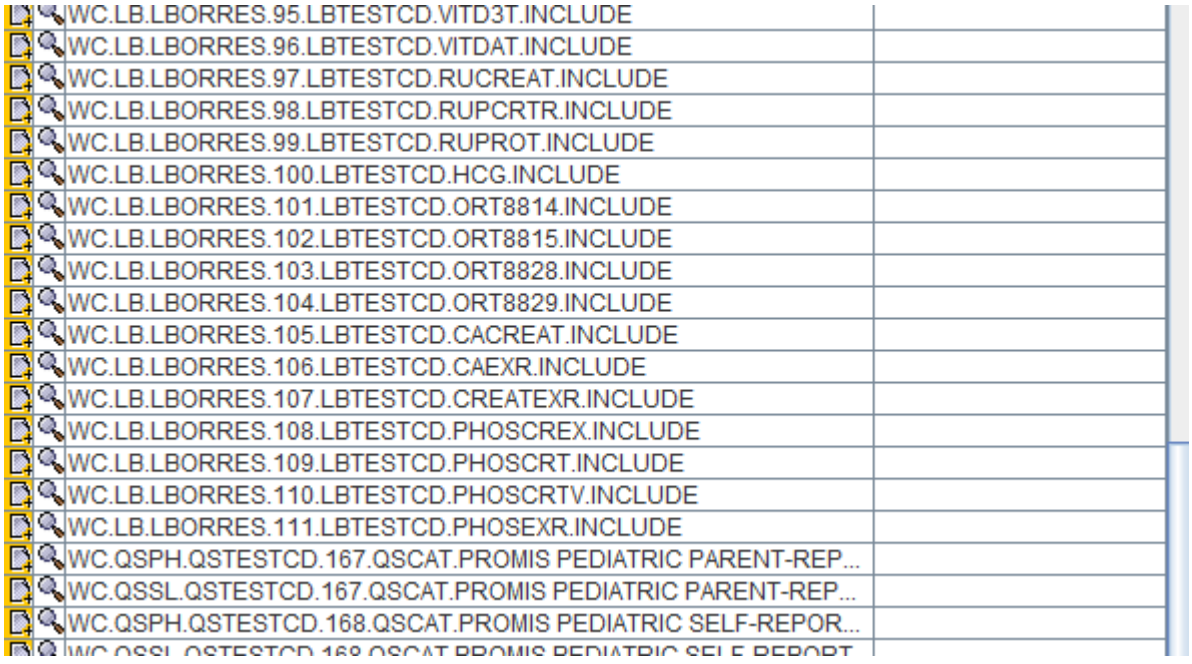
SDTM Example

When the checkbox "Try to create ValueLists from definitions in a 'valuelistvariables.dat file" was checked and a list of "whereclause statements" is provided in the file "valuelistvariables.dat", then also other ValueLists will be created.

For example:



and for the "where-clauses":



where we see that for LBORRES dependent on LBTESTCD, we have 111 (!) where-clauses. So very probably, we have some "overkill" there …

Let us first have a look at the valuelist generated for VSPOS in the HTML View:

VSTESTCD - [Edit]		Name	text		6	for VSTESTCD - [Edit]	[Add]
VSTEST - [Edit]		Vital Signs Test Name	text		24	Vital Signs Test Name subset for VSTEST - [Edit]	[Add]
VSPOS - [Edit] [Remove/Replace ValueList] [Edit ValueList]		Vital Signs Position of Subject	text		8	Position - [Edit]	[Add]
ValueList variable for VSPOS - [Edit]	[Edit] VSTESTCD IN [SYSBP (Systolic Blood Pressure), DIABP (Diastolic Blood	VSPOS	text		8	Position subset for ValueList - [Edit]	- [Edi

and when clicking the hyperlink for "Position":

Position [C71148] [CDISC/NCI SDTM 2025-03-28]

[Edit]

Permitted Value (Code)
DECUBITUS
FOWLERS
LATERAL DECUBITUS
LEFT LATERAL DECUBITUS
PRONE
REVERSE TRENDELENBURG
RIGHT LATERAL DECUBITUS
SEMI-FOWLERS
SEMI-RECUMBENT
SITTING
SITTING, LEGS DEPENDENT
SLING

showing us the complete codelist for "Position".
When however also the checkbox "Try to create subset CodeLists from XPT content and selected Controlled Terminology from 'subsetcodelistvariables.dat' file" has been checked, the subset codelist is referenced:

ValueList variable for VSPOS - [Edit]	[Edit] VSTESTCD IN [SYSBP (Systolic Blood Pressure), DIABP (Diastolic Blood Pressure)]	VSPOS	text		8	Position subset for ValueList - [Edit]	- [

with the subset CodeList being:

Position subset for ValueList [C71148] [CDISC/NCI SDTM 2025-03-28]

[Edit]

Permitted Value (Code)
STANDING
SUPINE

as only "STANDING" and "SUPINE" appeared in the dataset. It may however also be that e.g. "LATERAL DECUBITUS" was on the CRF, but never selected, and so did not appear in the XPT file, then it must still be added to the subset codelist, as the define.xml is about "planned"! This shows again how dangerous generating the define.xml solely from the set of XPT files can be.
Also notice the ValueList text in the second column, essentially stating that VSPOS is only populated when VSTESTCD is either "SYSBP" or "DIABP".

We also found that 4 ValueLists were generated for LBORRES with the dependency on LBTESTCD:

LBORRES - [Edit] [Remove/Replace ValueList] [Edit ValueList]		Result or Finding in Original Units	text		6	
ValueList variable for LBORRES - [Edit]	[Edit] LBTESTCD EQ ALP (Alkaline Phosphatase)	LBORRES	integer		3	
ValueList variable for LBORRES - [Edit]	[Edit] LBTESTCD EQ CA (Calcium)	LBORRES	float		3	
ValueList variable for LBORRES - [Edit]	[Edit] LBTESTCD EQ CREAT (Creatinine)	LBORRES	float		3	
ValueList variable for LBORRES - [Edit]	[Edit] LBTESTCD EQ PHOS (Phosphate)	LBORRES	float		3	

in which we see that the properties (as dependent on LBTESTCD) are the same for "CA" (calcium), "CREAT" (Creatinine) and "PHOS" (Phosphate) are identical. So it may be a good idea to "group" these. We can of course do this in the editor (see next section), but when we can know this in advance, we can also have the following entry in the "valuelistvariables.dat":

```
FTORRES WHERE FTTESTCD EQ TUG0101

LBORRES WHERE LBTESTCD IN CA,CREAT,PHOS
```

and the result then (in the "HTML View") then is:

LBORRES - [Edit] [Remove/Replace ValueList] [Edit ValueList]		Result or Finding in Original Units	text		6
ValueList variable for LBORRES - [Edit]	[Edit] LBTESTCD IN [CA (Calcium), CREAT (Creatinine), PHOS (Phosphate)]	LBORRES	float		3
ValueList variable for LBORRES - [Edit]	[Edit] LBTESTCD EQ ALP (Alkaline Phosphatase)	LBORRES	integer		3
LBORRESU - [Edit]					

Using automated generation of ValueLists from XPT files cannot only easily lead to "overkill", but will often also lead to incomplete information, as demonstrated before for the cases of VSPOS, ECREASOC, when choices on the CRF were never used and thus are not present in the data files.
Careful design of the entries in the "valuelistvariables.dat" file can take care that ValueLists are only generated for the cases where it make sense, but it cannot ensure completeness or correctness of the codelists generated for the ValueLists.

Therefore, it is always much better to already start designing the define.xml once the CRFs are final, even before the study start, and even when the define.xml will not always contain all the necessary information, such as the maximal lengths of variables. However, such define.xml files can already be complete for over 90%.
This also means that once the submission is prepared, the amount of work for getting the "perfect" define.xml will be minimal, whereas when starting from XPT files, getting to the "perfect" may lead to days or even weeks of corrections, improvements etc.. It is clear that this will be costly both in terms of time and money.

ADaM example

Let us have a look at an ADaM example: we have an XPT file ADPP.xpt (Pharmacokinetic Parameters Analysis Dataset). Typically for ADaM, this has PARAMCD, AVAL and AVALU columns. The AVAL column only contains numeric values, so it doesn't make much sense to generate a ValueList for it. When we however sort the XPT dataset by AVALU values, we find:

ADT	ADY	PARAMCD	PARAM	PARAMN	AVAL	AVALU	TRTP	T
025-02-19	1	HALF	Terminal Half-Life	5	21,84	h	Treatment 2	
025-02-19	1	TMAX	Time of Maximum...	2	2,73	h	Treatment 2	
025-11-03	29	HALF	Terminal Half-Life	5	19,63	h	Treatment 2	
025-11-03	29	TMAX	Time of Maximum...	2	3,51	h	Treatment 2	
025-12-07	57	HALF	Terminal Half-Life	5	22,28	h	Treatment 3	
025-12-07	57	TMAX	Time of Maximum...	2	1,53	h	Treatment 3	
025-10-12	1	TMAX	Time of Maximum...	2	3,84	h	Treatment 3	
025-10-12	1	HALF	Terminal Half-Life	5	12,41	h	Treatment 3	
025-10-22	57	VSS	Volume of Distrib...	7	91,66	L	Treatment 3	
025-09-24	29	VSS	Volume of Distrib...	7	49,84	L	Treatment 3	
025-08-27	1	VSS	Volume of Distrib...	7	111,01	L	Treatment 3	
025-06-17	57	VSS	Volume of Distrib...	7	48,44	L	Treatment 2	
025-05-20	29	VSS	Volume of Distrib...	7	107,31	L	Treatment 2	

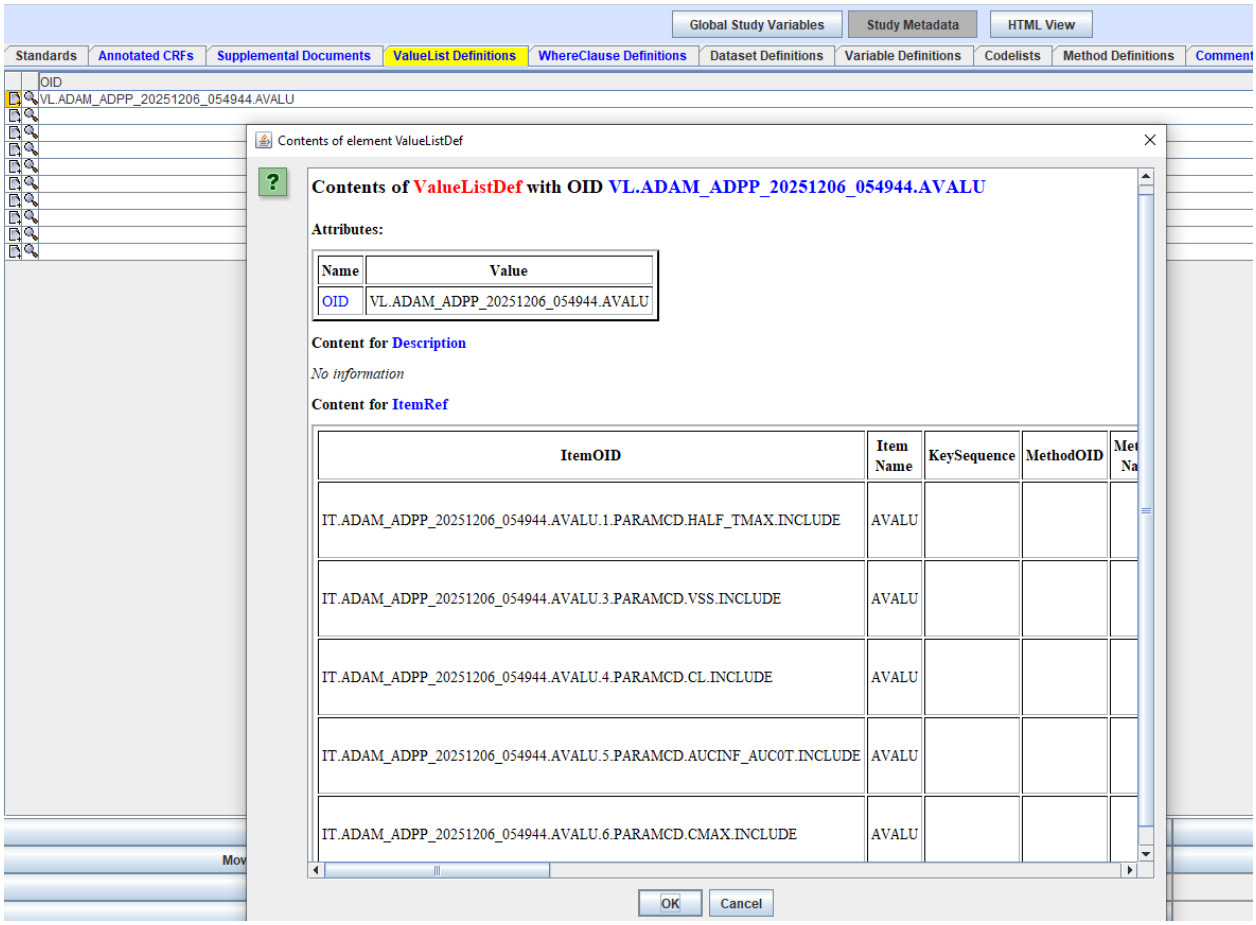
We see e.g. that for the unit "h", PARAMCD can have the values "HALF" and "TMAX", and that for the unit "L", PARAMCD can only have the value "VSS".

We can easily put this information in our file "valuelistvariables.dat":

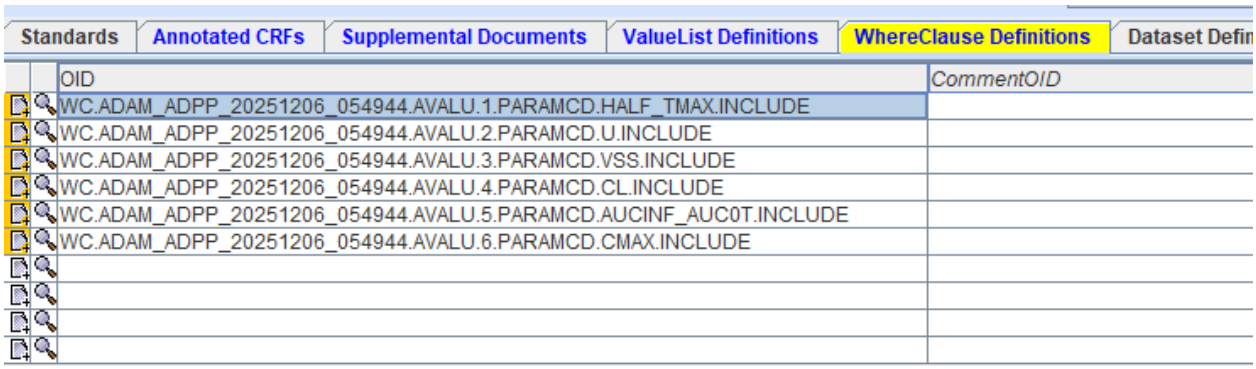
```
AVALU WHERE PARAMCD IN HALF,TMAX
AVALU WHERE PARAMCD EQ VSS
AVALU WHERE PARAMCD EQ CL
AVALU WHERE PARAMCD IN AUCINF,AUC0T
AVALU WHERE PARAMCD EQ CMAX
```

where we also define some "groupings" on which we want to generate ValueLists.

When the XPT file is then loaded, and one then selects the "ValueList Definitions" tab, one observes that one ValueList has been generated with 5 items:



and when one selects the "WhereClause Definitions", one finds:



Using the "HTML View" allows us to see the results in a "human-friendly" view:

PARAMN - [Edit]			integer		1	
AVAL - [Edit] [Create ValueList] [Add ValueList]			float		6	
AVALU - [Edit] [Remove/Replace ValueList] [Edit ValueList]			text		7	
ValueList variable for AVALU - [Edit]	[Edit] PARAMCD IN [HALF, TMAX]	AVALU	text		1	CodeList for ValueList Item IT.ADAM_ADPP_20251206_054944.AVALU.1.PARAMCD.HALF_TMAX.INCLUDE - [Edit]
ValueList variable for AVALU - [Edit]	[Edit] PARAMCD EQ VSS	AVALU	text		1	CodeList for ValueList Item IT.ADAM_ADPP_20251206_054944.AVALU.3.PARAMCD.VSS.INCLUDE - [Edit]
ValueList variable for AVALU - [Edit]	[Edit] PARAMCD EQ CL	AVALU	text		6	CodeList for ValueList Item IT.ADAM_ADPP_20251206_054944.AVALU.4.PARAMCD.CL.INCLUDE - [Edit]
ValueList variable for AVALU - [Edit]	[Edit] PARAMCD IN [AUCINF, AUC0T]	AVALU	text		7	CodeList for ValueList Item IT.ADAM_ADPP_20251206_054944.AVALU.5.PARAMCD.AUCINF_AUC0T.INCLUDE - [Edit]
ValueList variable for AVALU - [Edit]	[Edit] PARAMCD EQ CMAX	AVALU	text		5	CodeList for ValueList Item IT.ADAM_ADPP_20251206_054944.AVALU.6.PARAMCD.CMAX.INCLUDE - [Edit]

Notice the items where the "where" has the grouping as we defined it in the "valuelistvariables.dat" file.

And when then clicking e.g. the first codelist link:

CodeList for ValueList Item IT.ADAM_ADPP_20251206_054944.AVALU.1.PARAMCD.HALF_TMAX.INCLUDE

[Edit]

Permitted Value (Code)
h

so the ValueList statement essentially is "When PARAMCD is either "HALF" or "TMAX", then the unit in AVALU is "h".

Generating ValueLists starting from (subset) CodeLists and the CRF

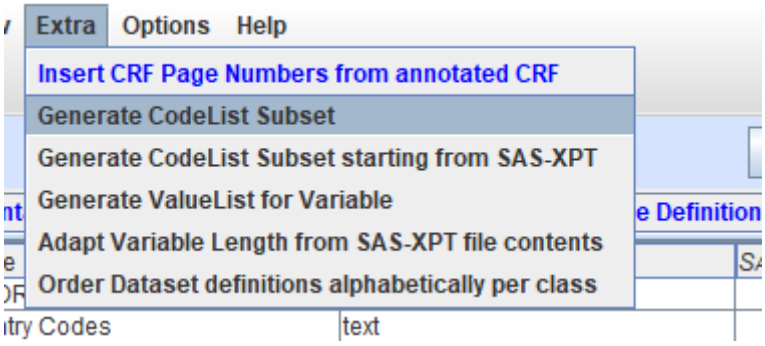
The best way to generate ValueLists when starting from a study design and/or CRF (annotated when possible) is to first set up a set of codelists. For SDTM/SEND "Findings" these will typically based on --TESTCD (test code), whereas for ADaM, these will typically be based on PARAMCD (parameter code).

ValueLists for SDTM and SEND - a simple example

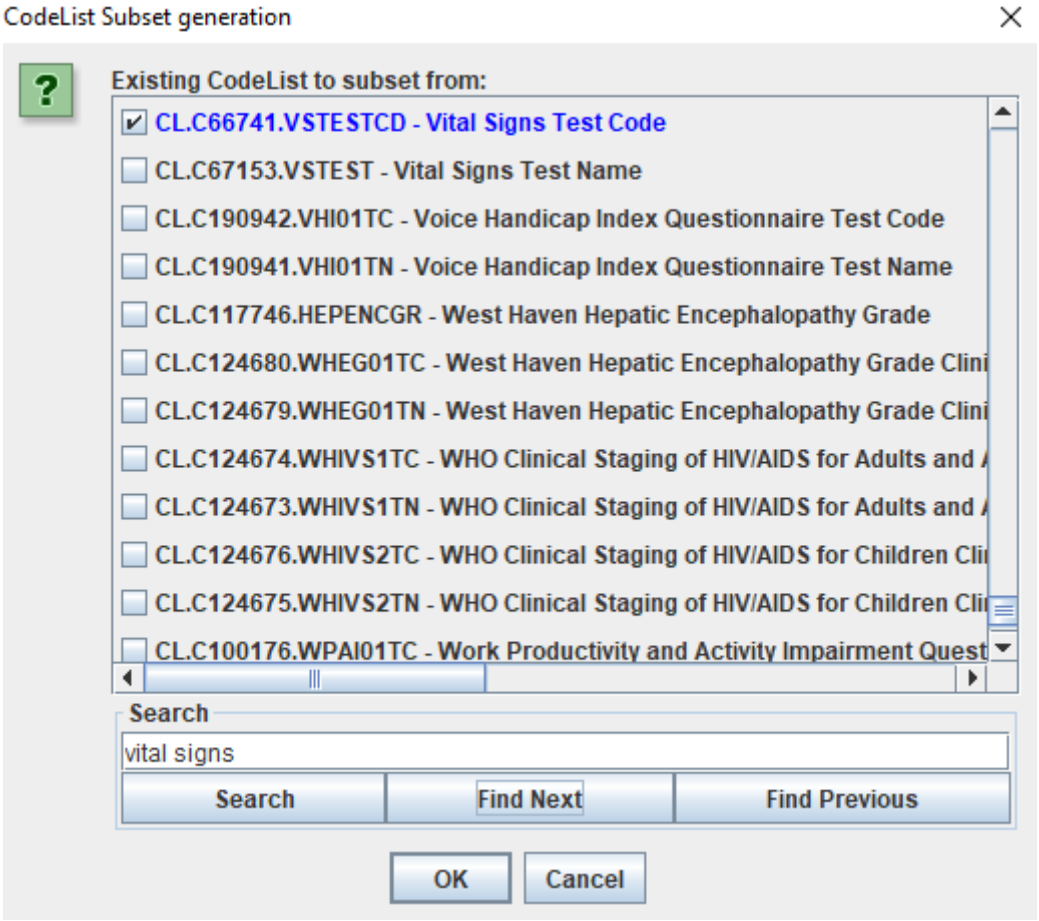
As a simple example, we take the case that vital signs measurements are collected. Assume the following measurements:
- systolic and diastolic blood pressure, with as unit mmHg (millimeter mercury column)
- height, either measured in cm (centimeters) or inches
- weight, either measured in kg (kilograms) or pounds
- frame size: with the possible values: small, medium and large
- heart rate, measured in beats per minute

We will start by subsetting the codelist for VSTESTCD and VSTEST to contain exactly these tests.

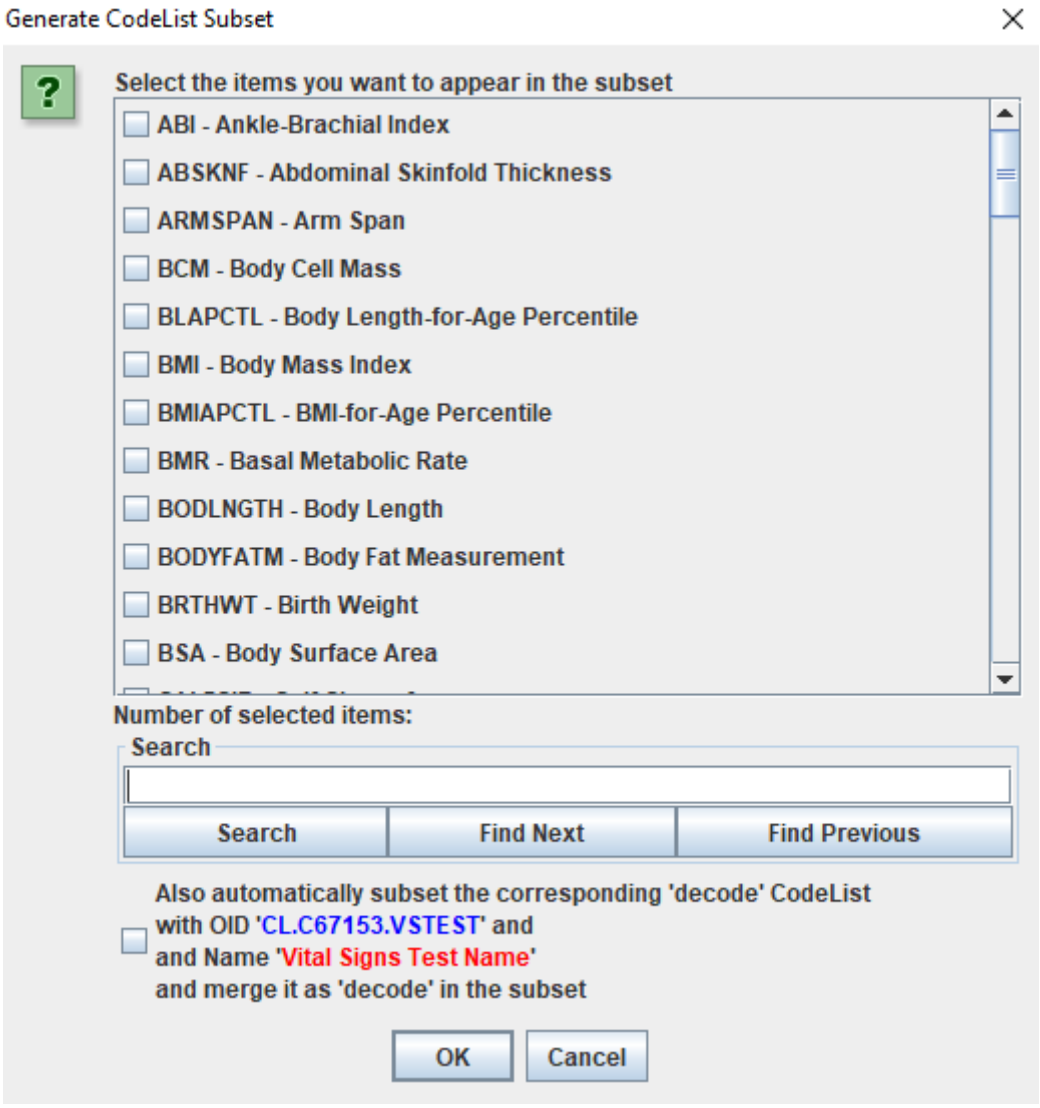
For this, we use the menu "Extra - Generate CodeList Subset":



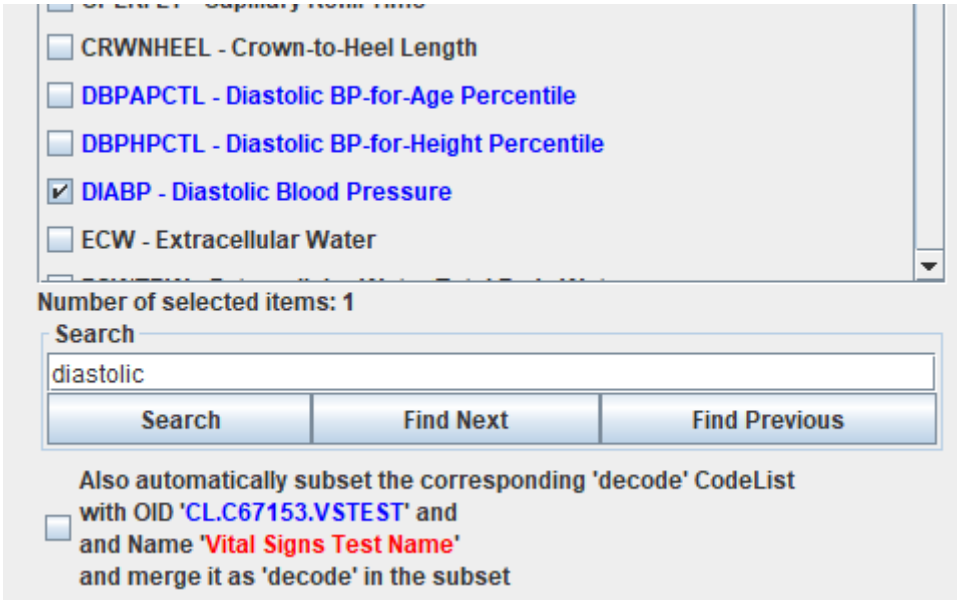
and then looking and selecting the codelist for "Vital Signs Test Code":



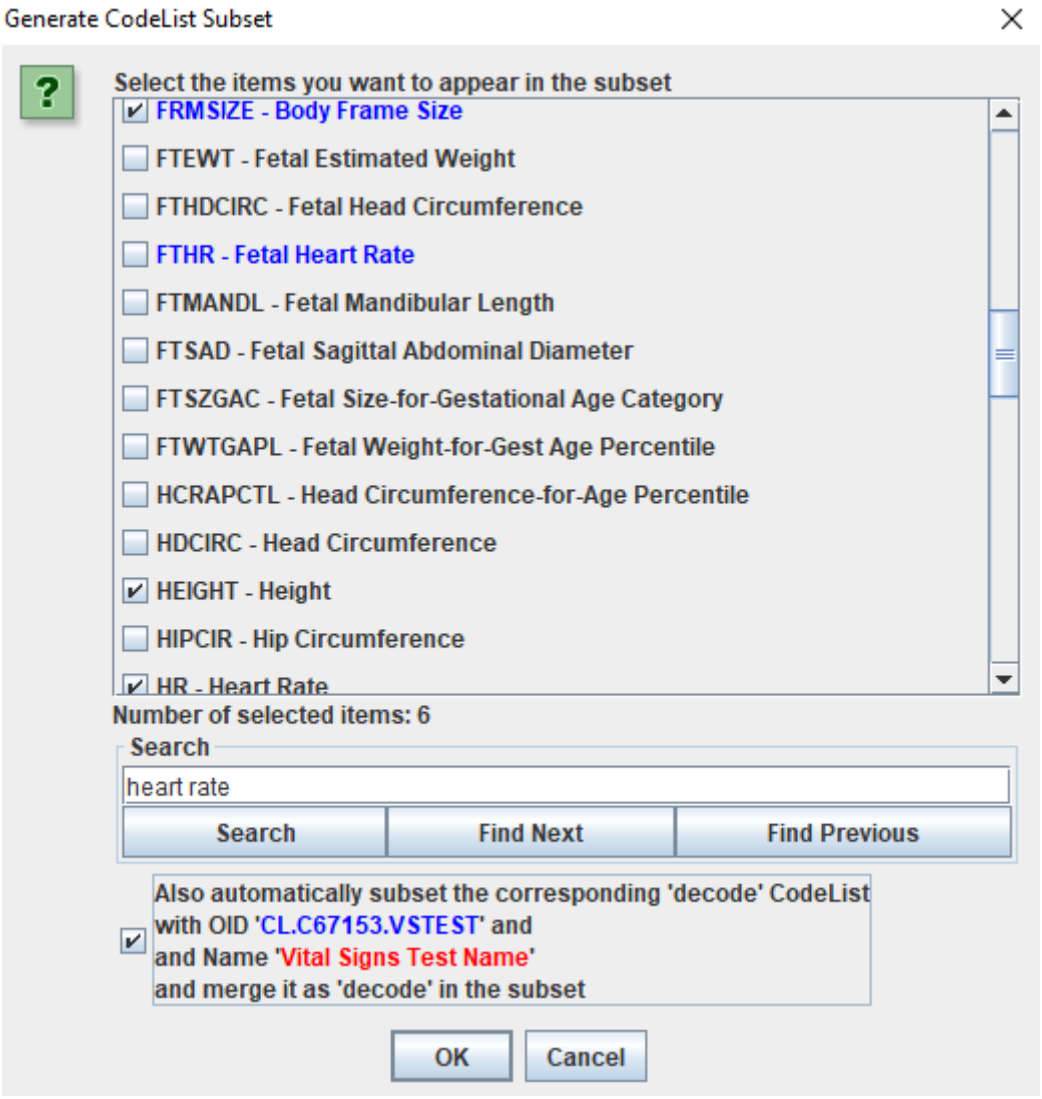
After clicking OK, leading to:



and then adding the different items we need, helped by the "Search" function:

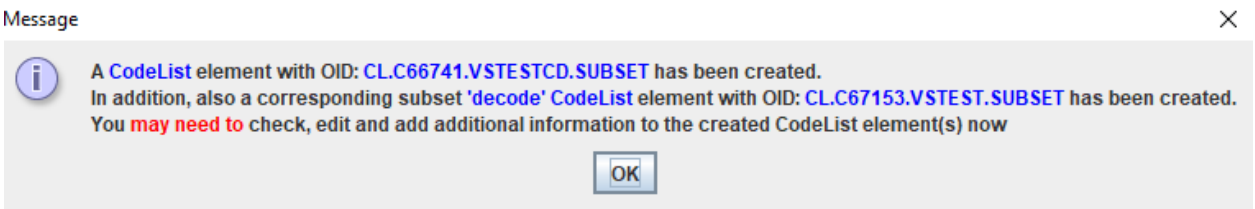


so that we quickly come to our 6 tests

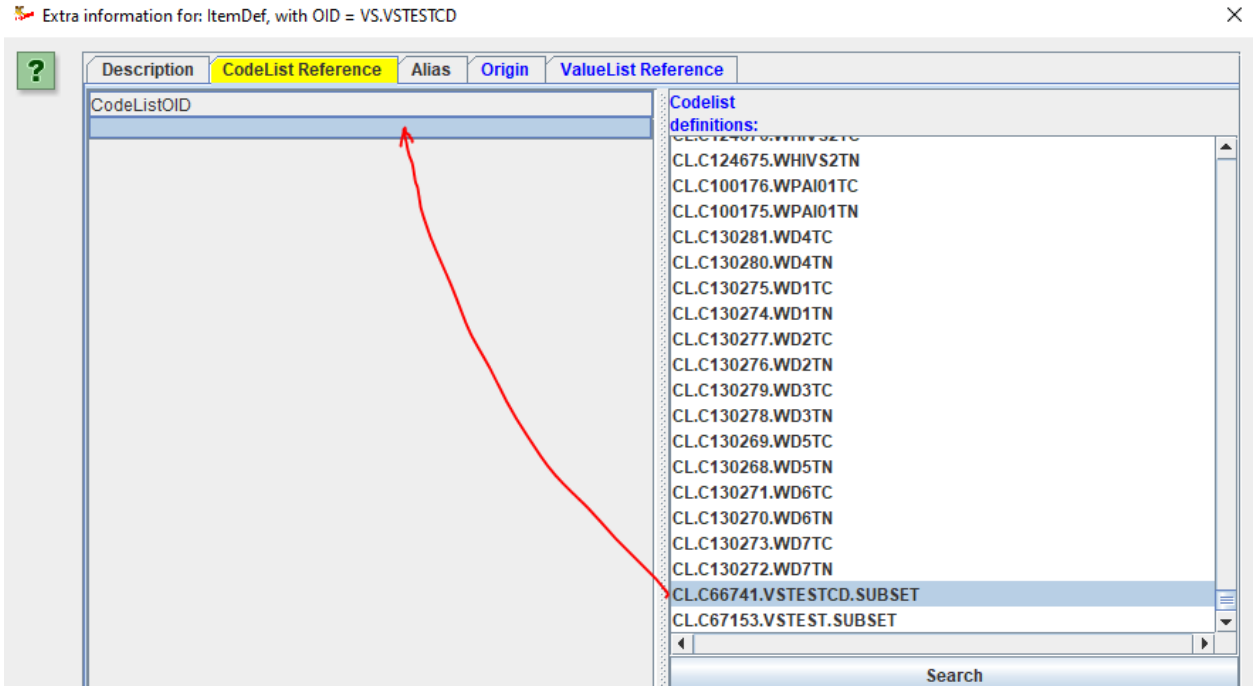


At this moment, it is also wise to check the checkbox "Also automatically subset the corresponding 'decode' CodeList", which in this case is the CodeList for VSTEST. This not only saves time, but also ensure that both codelists are synchronized. After clicking "OK", we can still change the OID (identifier) and the name of the codelist, but this is usually not necessary:

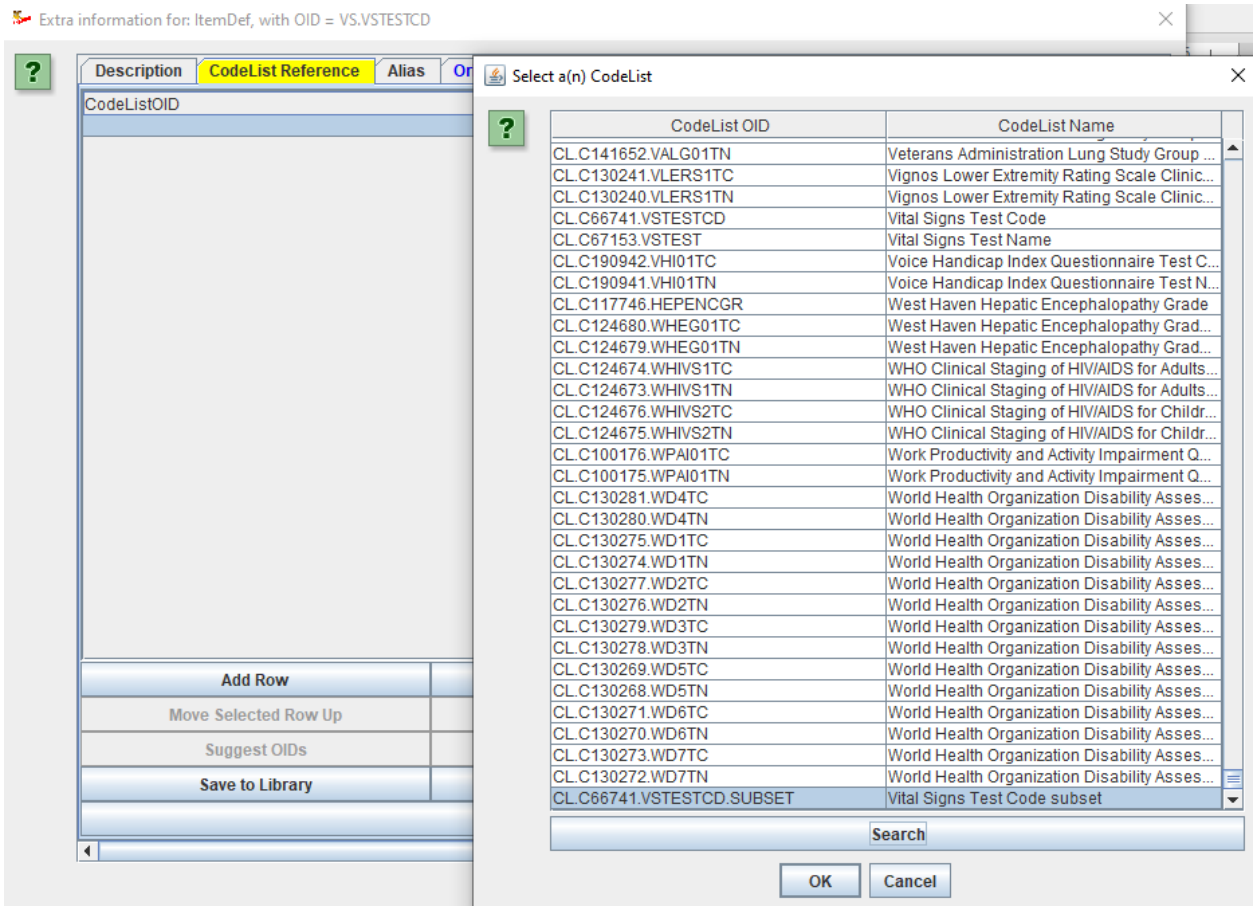
leading to the message:



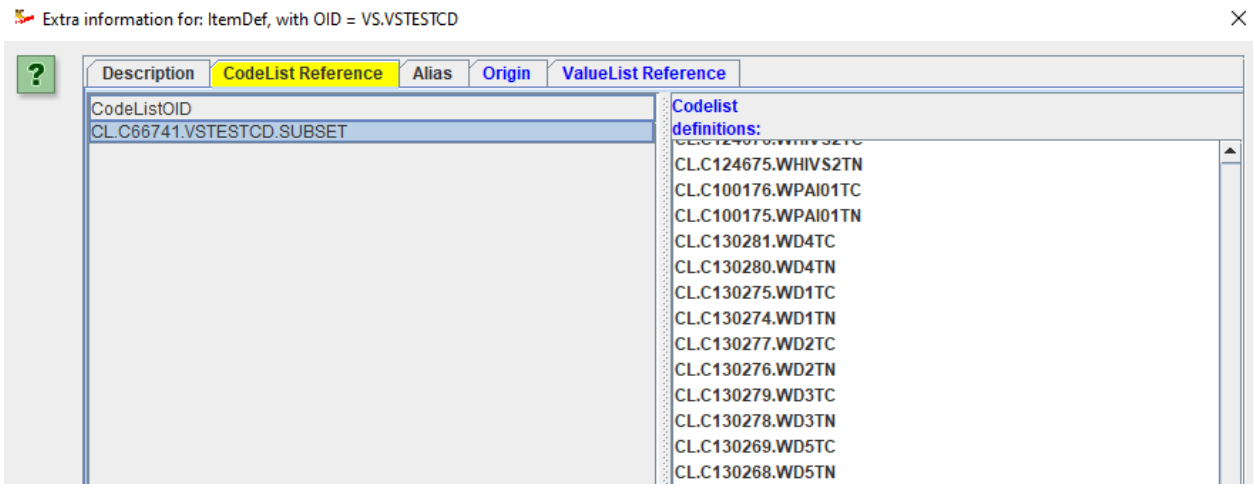
We can now also assign these codelists to VSTESTCD and VSTEST respectively, by selecting them one by one in the tab "Variable Definitions", click the "Edit" icon, select the "CodeListRef" tab, and then drag-and-drop them to the field. For example for VSTESTCD:



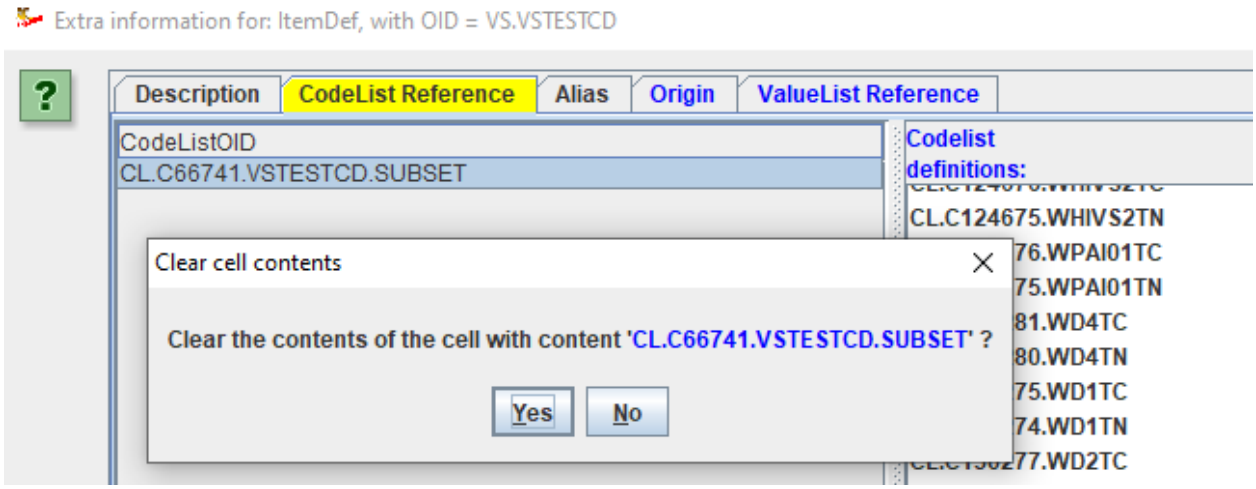
Another way is to click in the field, and the list with all available codelists is displayed, from which one can select just by a click:



The result is just the same:



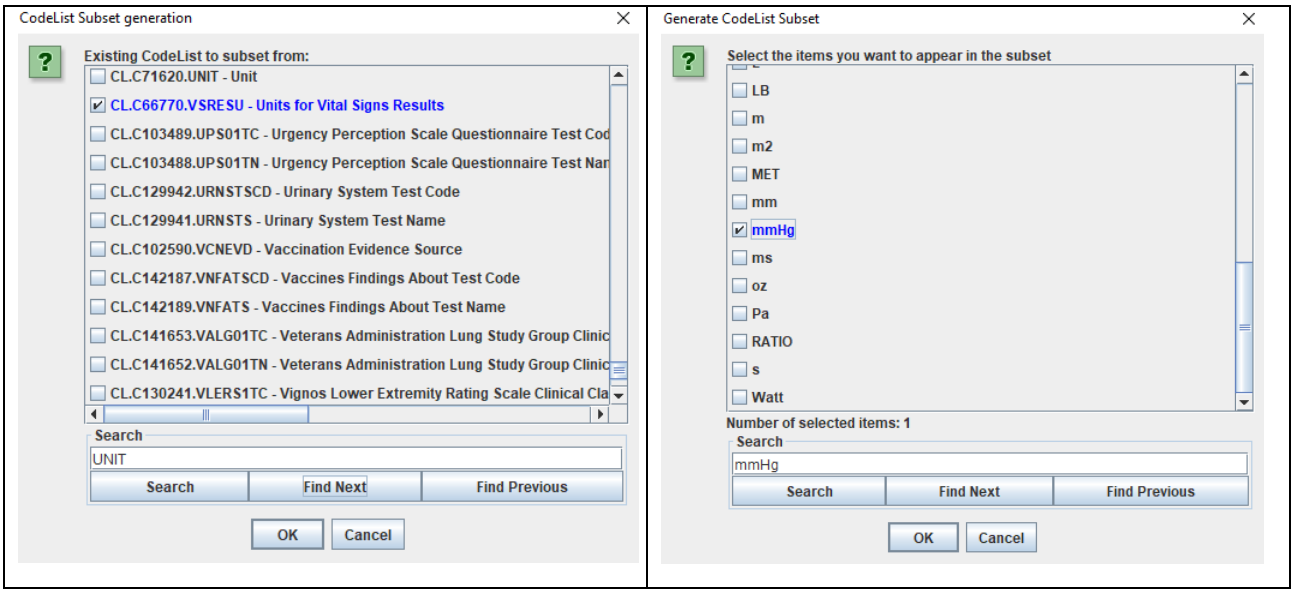
If one makes an error, one can delete the value in the cell by either using the "Delete Selected Row", or right-click, and then confirm that one wants to delete the item:



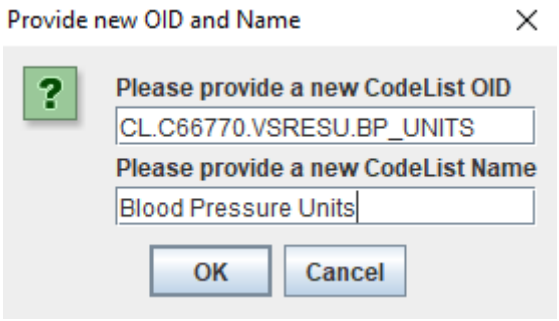
We also want to generate a subset codelist for VSORRESU, only containing the units really intended to be used, i.e. "mmHg", "cm", "in" (inches), "kg", "LB" (pounds) and "beats/min". This is easily done in the same way as for the VSTESTCD subset codelist. We then assign that subset codelist to VSORRESU.

Once we have created the subset codelists for VSTESTCD and VSTEST and have assigned them to VSTESTCD and VSTEST respectively, we are going to develop or subset some other codelists that we may need to the ValueList items.

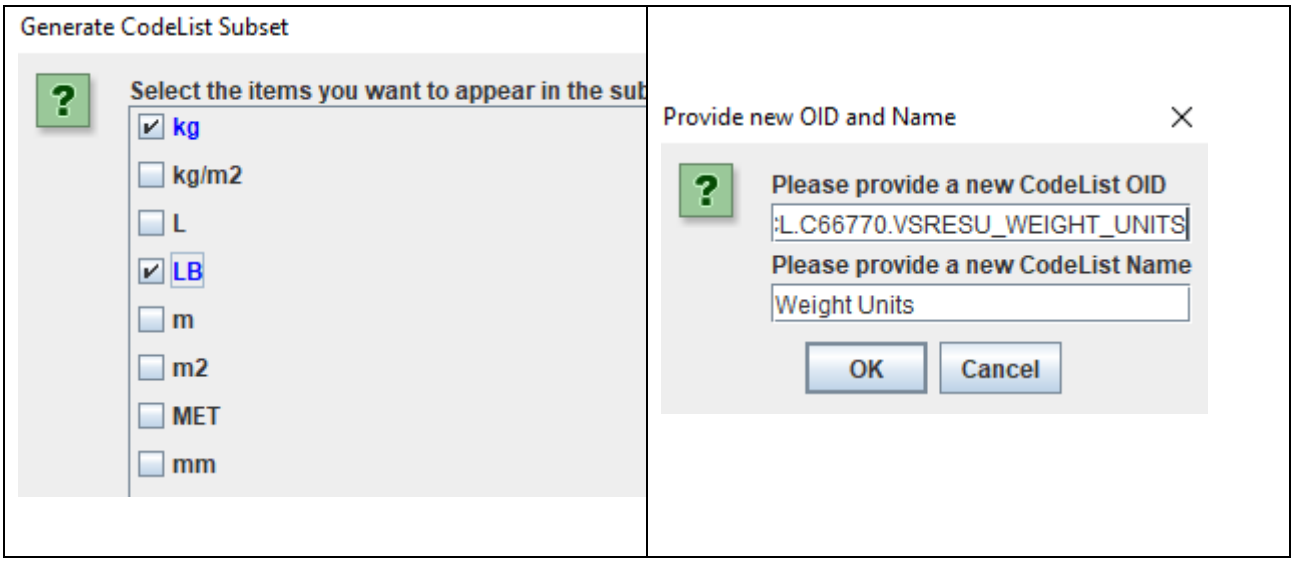
For "diastolic blood pressure" and "systolic blood pressure", we only allow the unit "mmHg", so we make a subset codelist for it from the VSRESU codelist only containing "mmHg":



but then give it a better OID and Name:



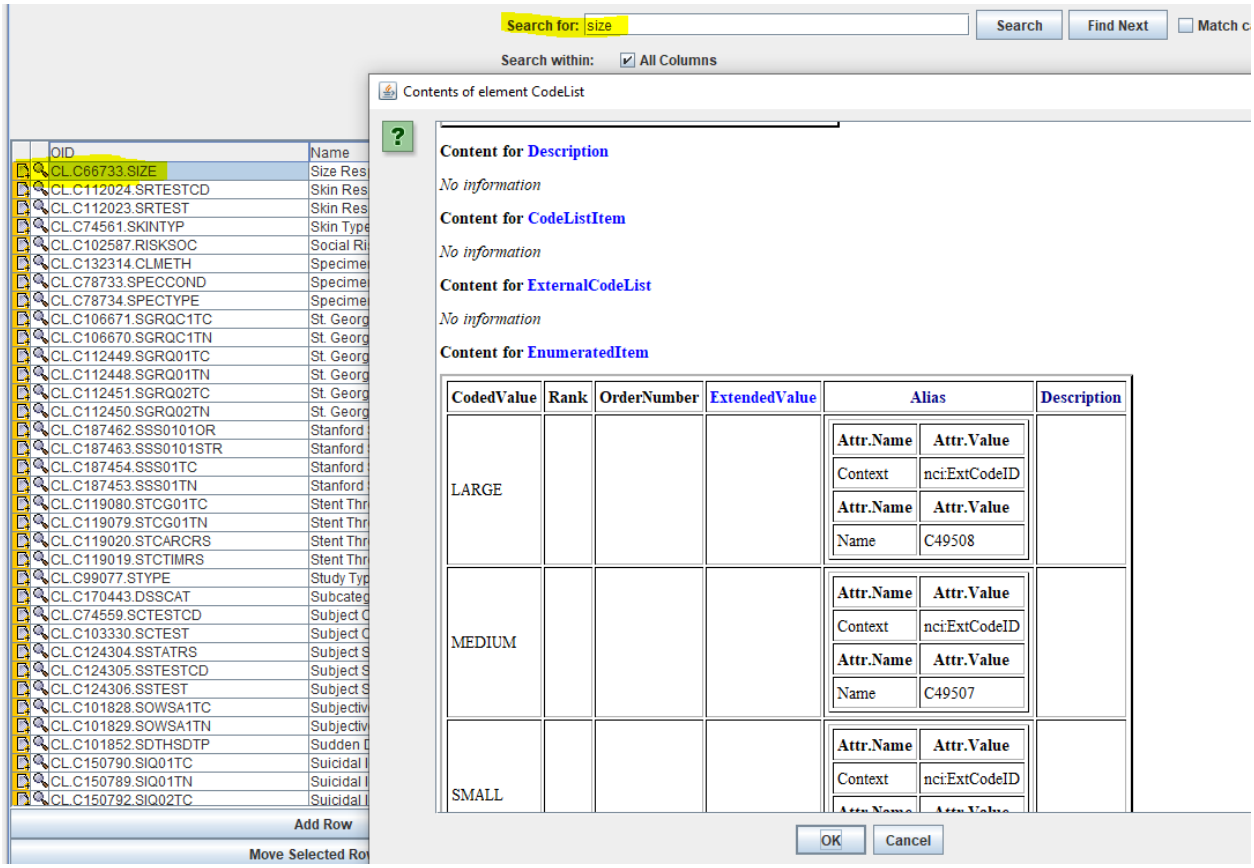
For "Weight", we generate a subset codelist only containing "kg" and "pounds" (for which the CDISC symbol is "LB"), and give the codelist a better OID and Name, e.g.:



For "Height" we generate a subset codelist only containing "cm" and "inches" (CDISC symbol "in"), and for "Heart rate" a subset codelist with only "beats/min". For each, we provide a unique OID and a suitable name.
If we then look into the list with codelists (tab CodeList Definitions), we find:

CL.C130273.WD7TC	World Health Organization Disab...	text
CL.C130272.WD7TN	World Health Organization Disab...	text
CL.C66741.VSTESTCD.SUBSET	Vital Signs Test Code subset	text
CL.C67153.VSTEST.SUBSET	Vital Signs Test Name subset	text
CL.C66770.VSRESU.BP_UNITS	Blood Pressure Units	text
CL.C66770.VSRESU_WEIGHT_UNITS	Weight Units	text
CL.C66770.VSRESU.HEIGHT_UNITS	Height Units	text
CL.C66770.VSRESU.HR_UNITS	Heart Rate Units	text

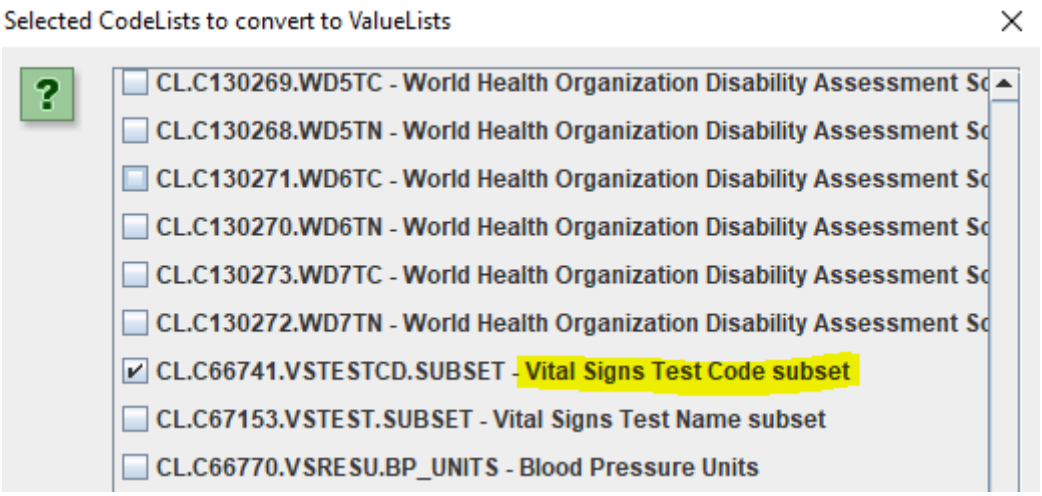
What about "Frame Size"?
A quick search in the list of codelists reveals that there is already a codelist that exactly is what we need: the "SIZE" codelist:



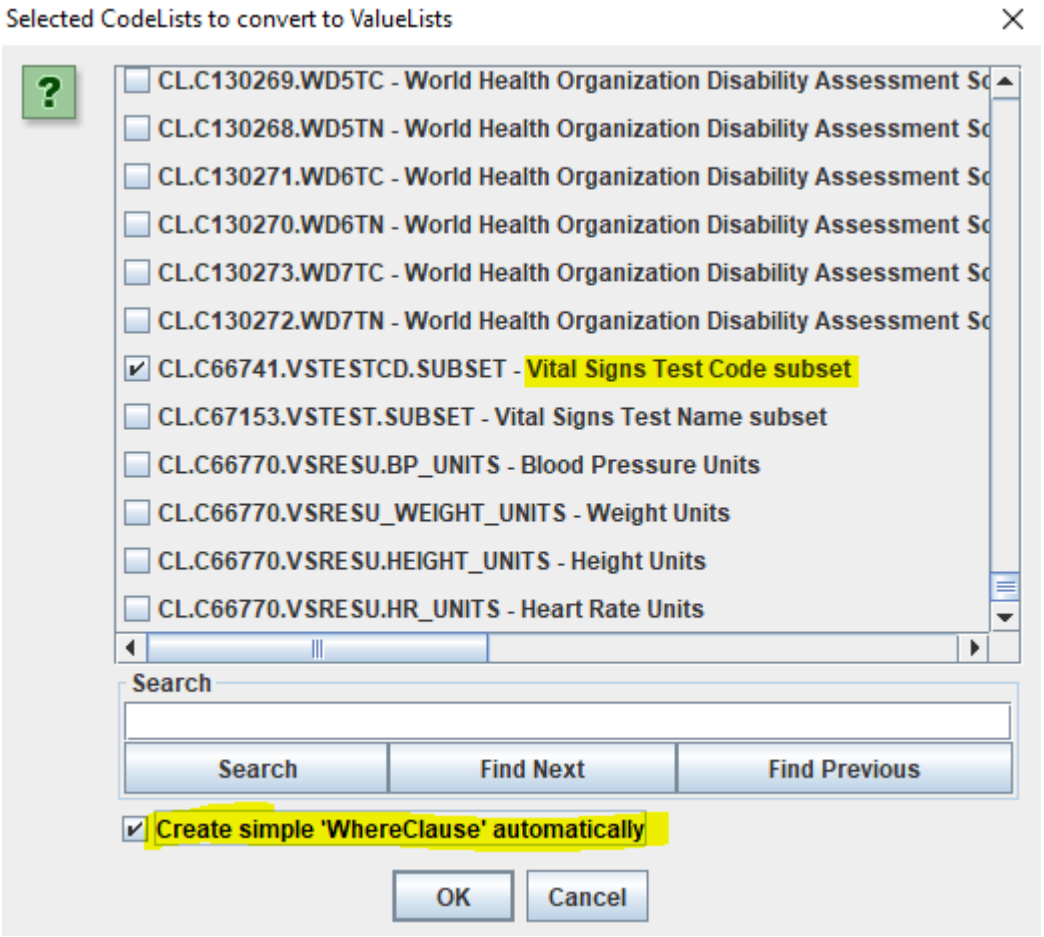
OK, we now have the codelists that we need to start generating the ValueList.

We start the process by using the menu "Transform - CodeList to ValueList":

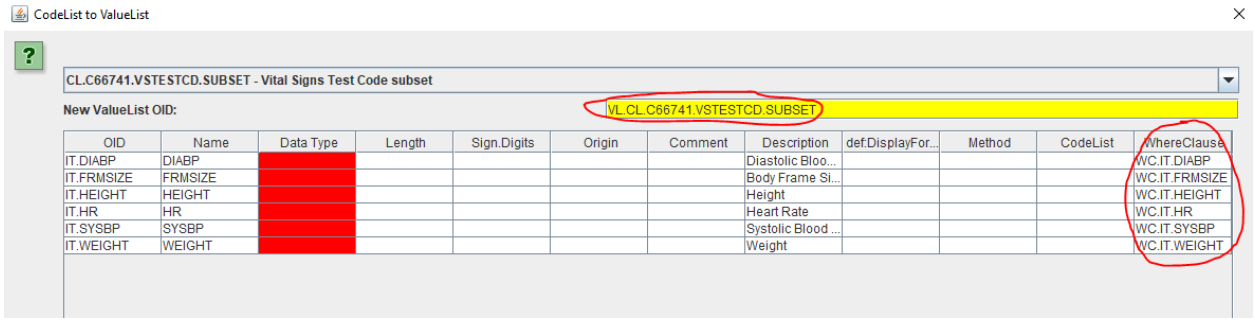
and select the "Vital Signs Test Code Subset", as that is the one we want to start from, I.e. we want to generate a ValueList that is dependent on the value of VSTESTCD for which we have developed a subset codelist:



This leads to another dialog:

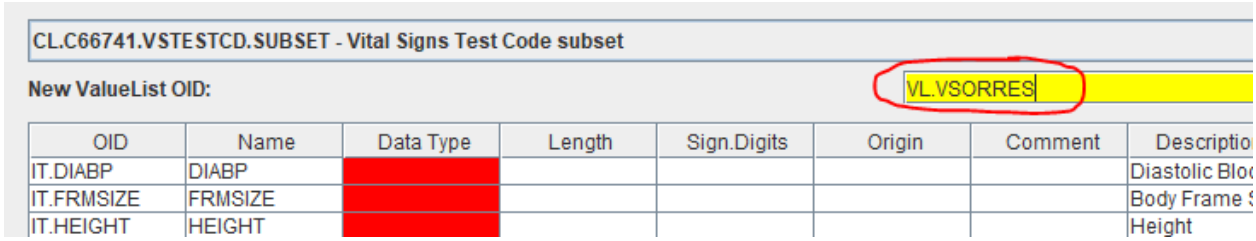


Very often, we will want to have the "WhereClause" being generated automatically (we can change everything in that later), so we do check that checkbox. After "OK", a table is generated:



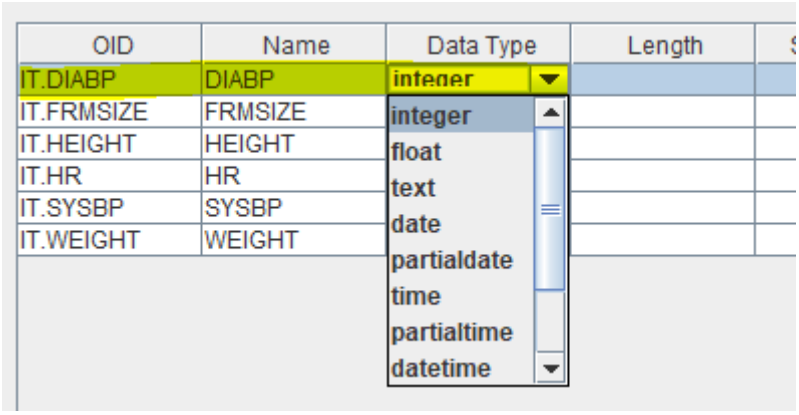
What do we want to use this ValueList for? In first instance, we want to have one explaining the properties for VSORRES depending on the value for VSTESTCD.
We will probably also want one describing the properties of VSORRESU depending on the value of VSTESTCD.

Let us start with the one describing the properties of VSORRES depending on VSTESTCD. So we assign a new valie for the OID, e.g.:

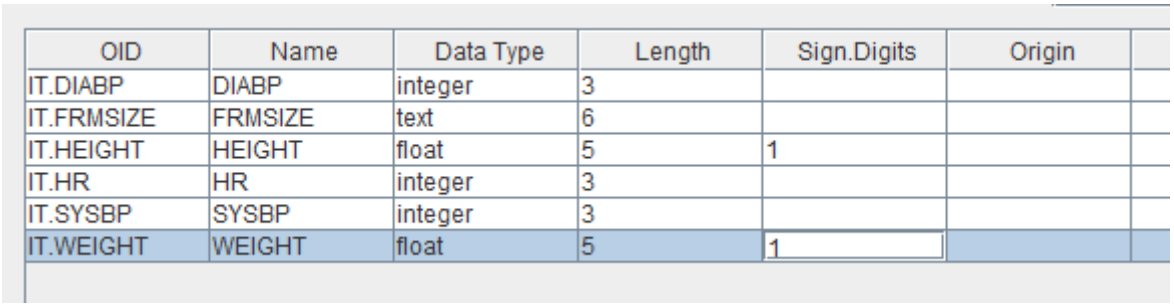


Remark that in Define-XML, ValueLists do not have a "Name", only an "OID". We can however add a "Description" later.

We can now start setting the properties. For example, for the blood pressures, we expect that the value is an integer with a maximum character length of 3. For "Height" and "Weight" we may expect a floating point number with one character after the decimal point, for hearth rate, we again expect an integer with a maximum character length of 3. For "Frame Size" we will use the codelist, having "SMALL", "MEDIUM", "LARGE", so this is text with a maximum of 6 characters. So we start filling:



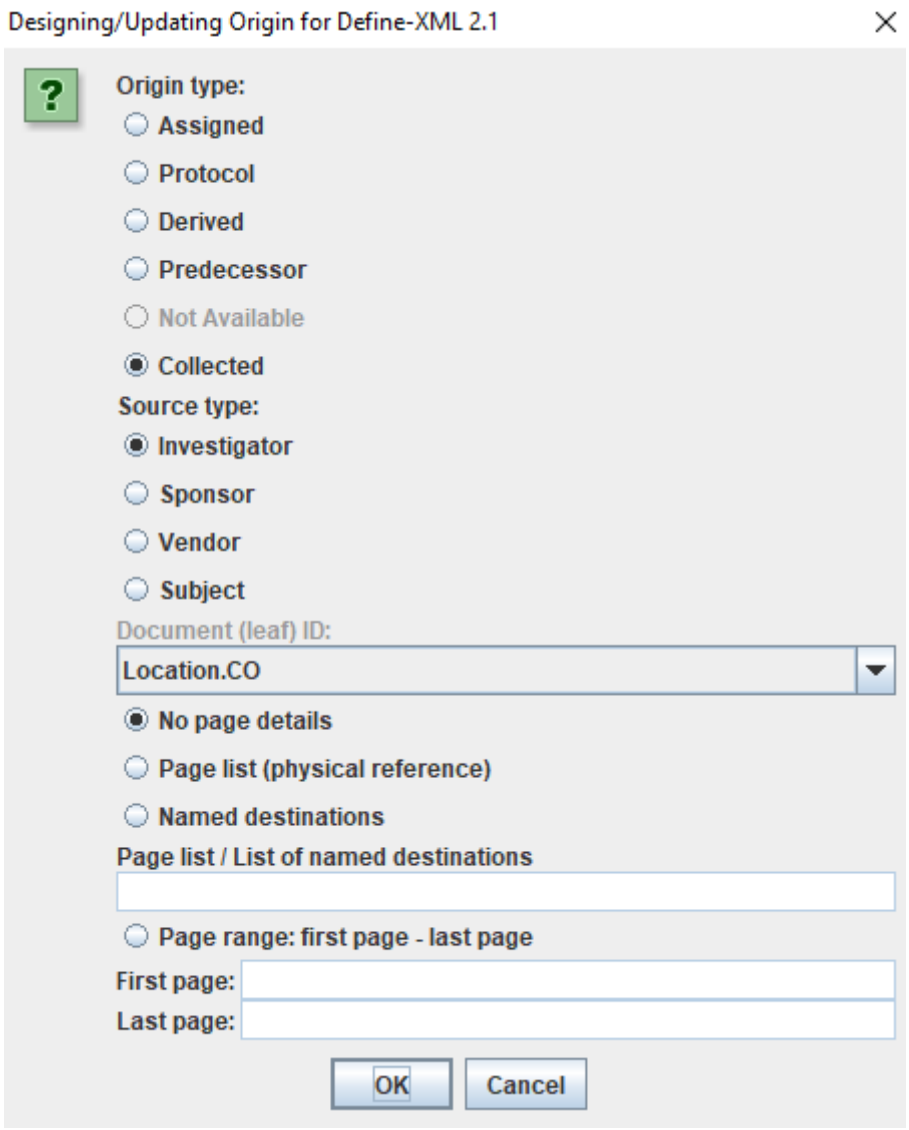
leading to e.g.:



We can always click the "Validate" button to check whether what we are doing is correct. For example, if we do not add a value for "Significant Digits" when the "Data Type" is "float", clicking "Validate" leads to the "Sign.Di," cell to be colored red:

IMAGE TO DO: something is still wrong here

We can also provide the "Origin" by clicking in the "Origin" cell, and a dialog shows up:

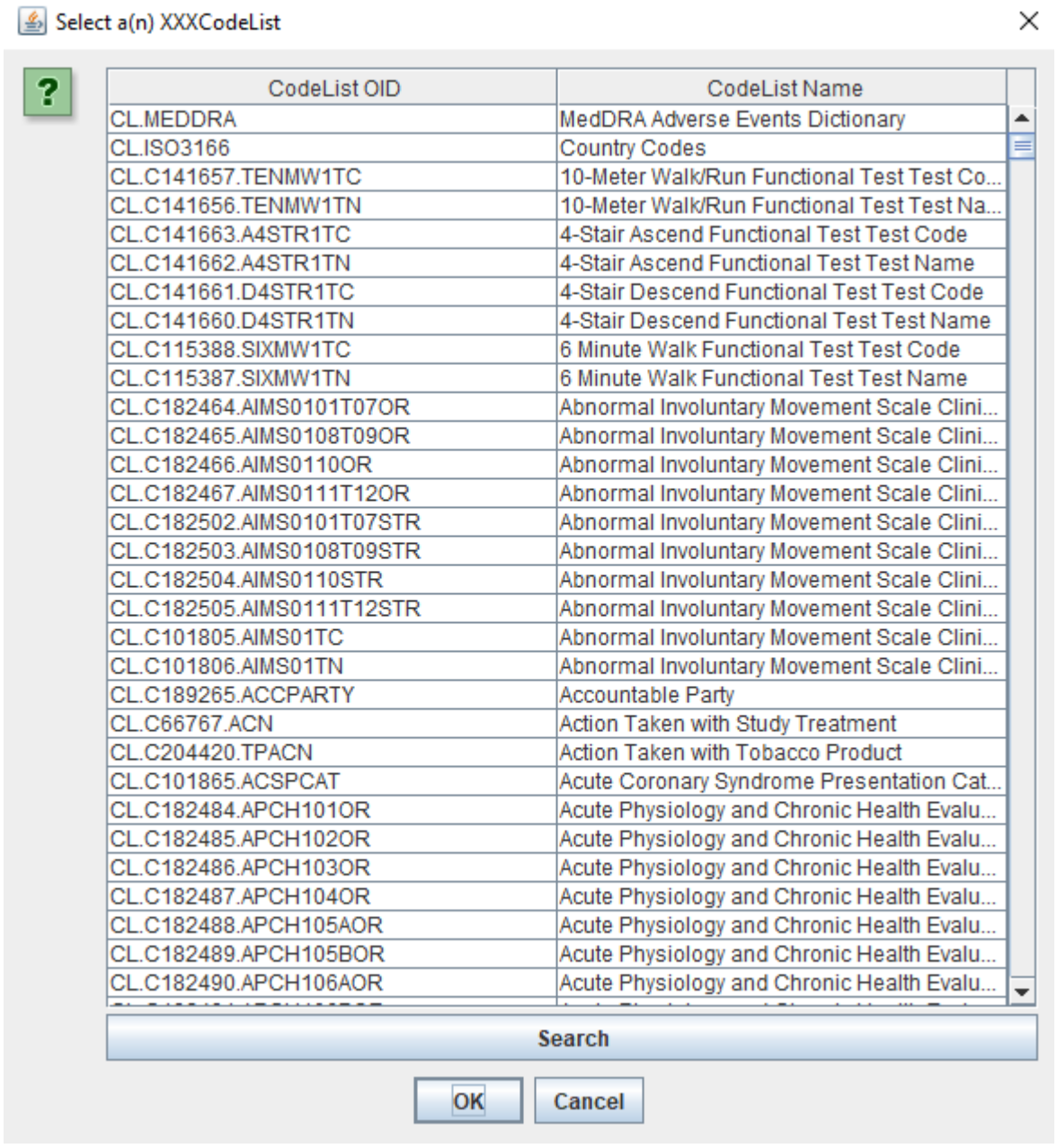


However, we will usually assign the Origin/Source on the variable level, not at the ValueList level.
A case where we want to assign it at the ValueList level is e.g. when some of the lab date comes from an external lab by electronic transfer (Origin Type = Collected, Source Type = Vendor) and some of the lab data comes from the CRF (Origin Type = Collected, Source Type = Investigator). We will go into more details in the section "Assigning Origin Information".

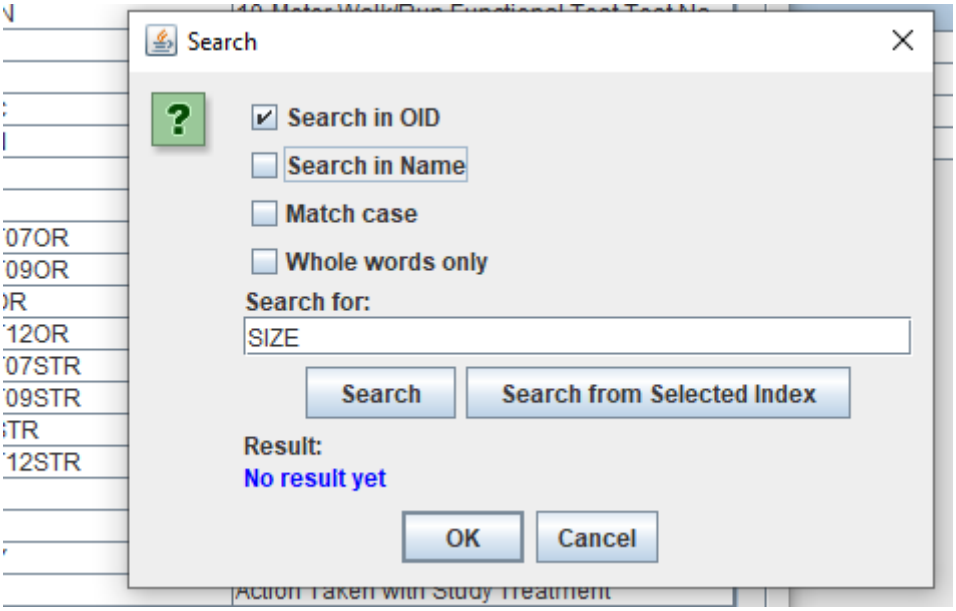
For "Frame Size", we need to state that there is a codelist, so we click the cell "CodeList":

OID	Name	Data Type	Length	Sign.Digits	Origin	Comment	Description	def:DisplayFor...	Method	CodeList	Whe
IT.DIABP	DIABP	integer	3				Diastolic Bloo...				WC.IT.
IT.FRMSIZE	FRMSIZE	text	6				Body Frame Si...				WC.IT.
IT.HEIGHT	HEIGHT	float	5	1			Height				WC.IT.
IT.HR	HR	integer	3				Heart Rate				WC.IT.
IT.SYSBP	SYSBP	integer	3				Systolic Blood ...				WC.IT.
IT.WEIGHT	WEIGHT	float	5				Weight				WC.IT.

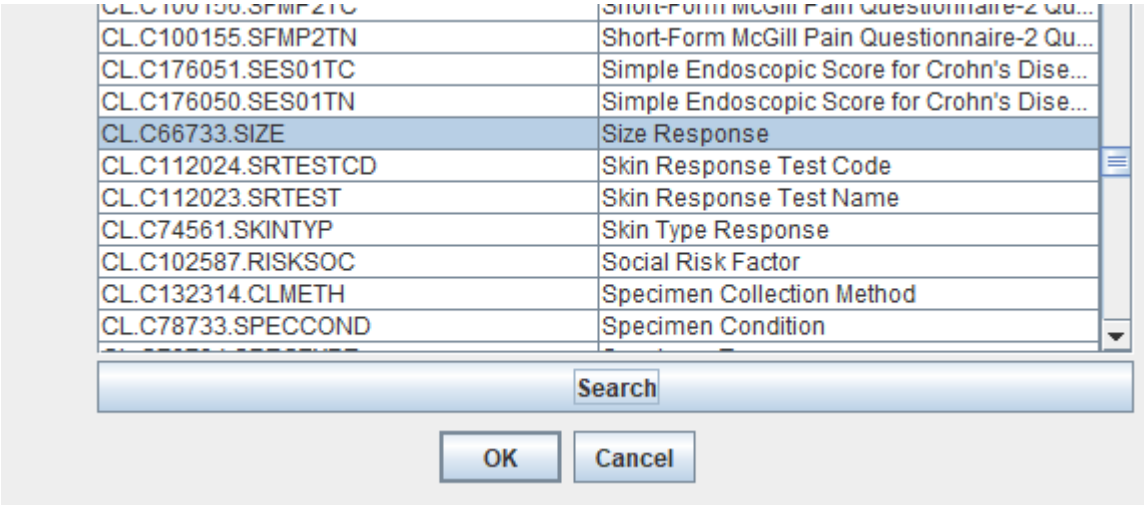
leading to a list with codelists:



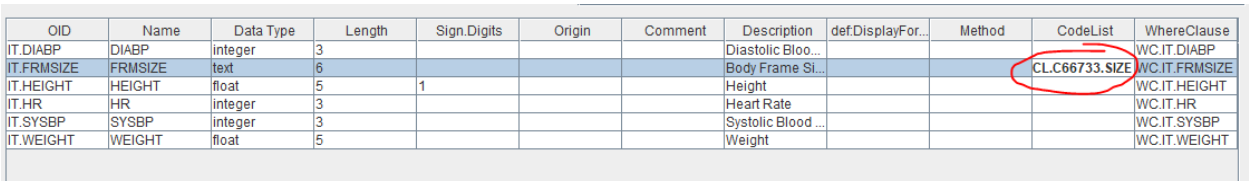
We can use the "Search" button to find our "SIZE" codelist:



leading to the selection:

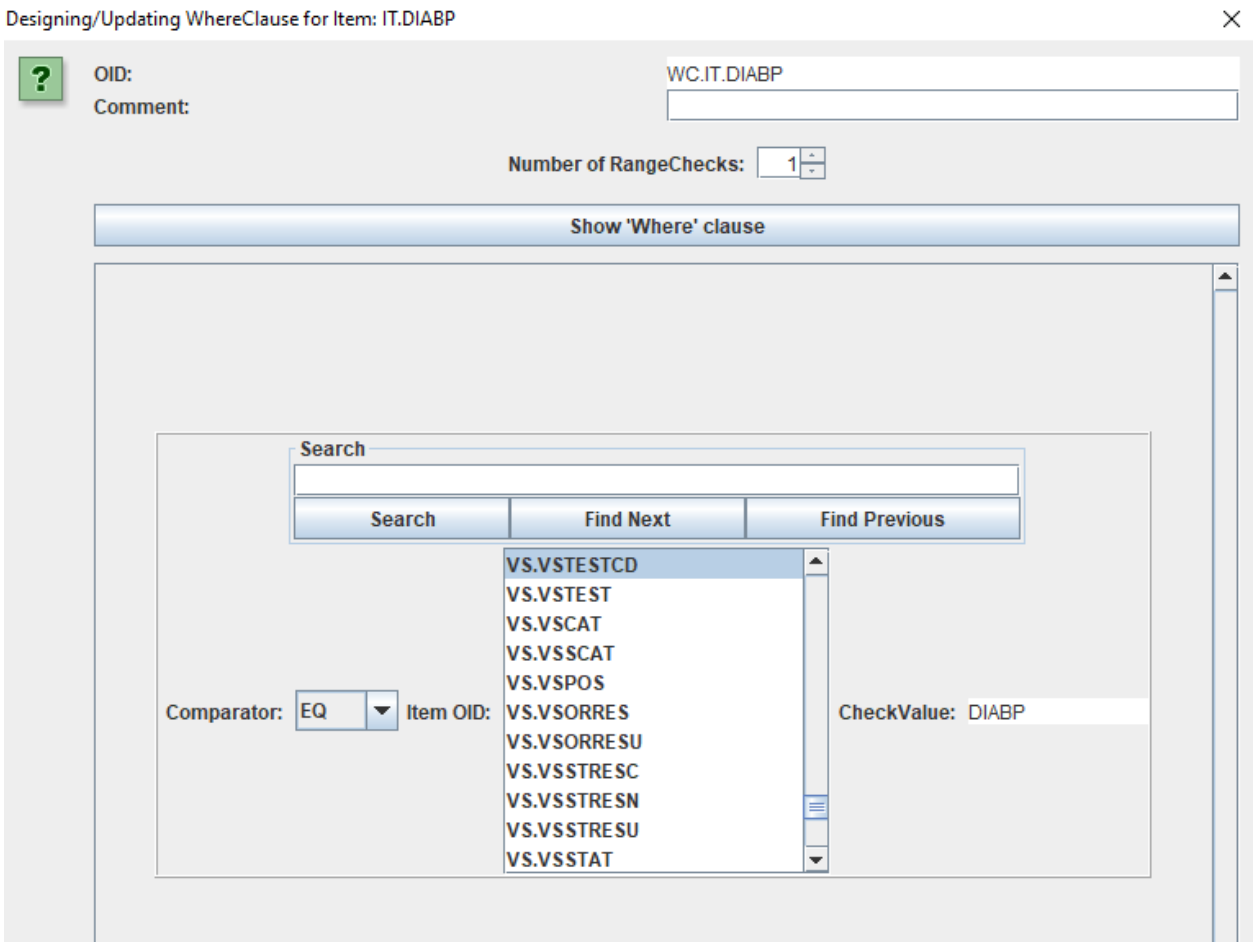


and after "OK" we find:

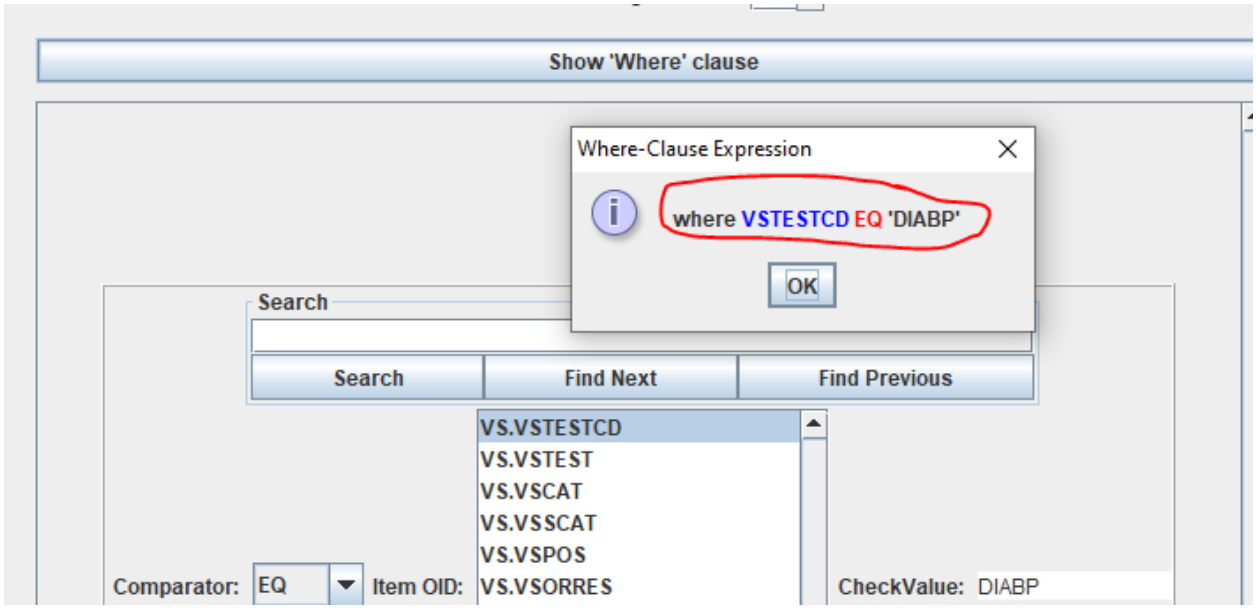


OID	Name	Data Type	Length	Sign.Digits	Origin	Comment	Description	def:DisplayFor...	Method	CodeList	WhereClause
IT.DIABP	DIABP	integer	3				Diastolic Bloo...				WC.IT.DIABP
IT.FRMSIZE	FRMSIZE	text	6				Body Frame Si...			CL.C66733.SIZE	WC.IT.FRMSIZE
IT.HEIGHT	HEIGHT	float	5	1			Height				WC.IT.HEIGHT
IT.HR	HR	integer	3				Heart Rate				WC.IT.HR
IT.SYSBP	SYSBP	integer	3				Systolic Blood ...				WC.IT.SYSBP
IT.WEIGHT	WEIGHT	float	5				Weight				WC.IT.WEIGHT

As we checked the checkbox, the "WhereClause"s (last column) were automatically created, let's have a look anyway. and click on the one for DIABP (WC.IT.DIABP). This then shows the wizard:

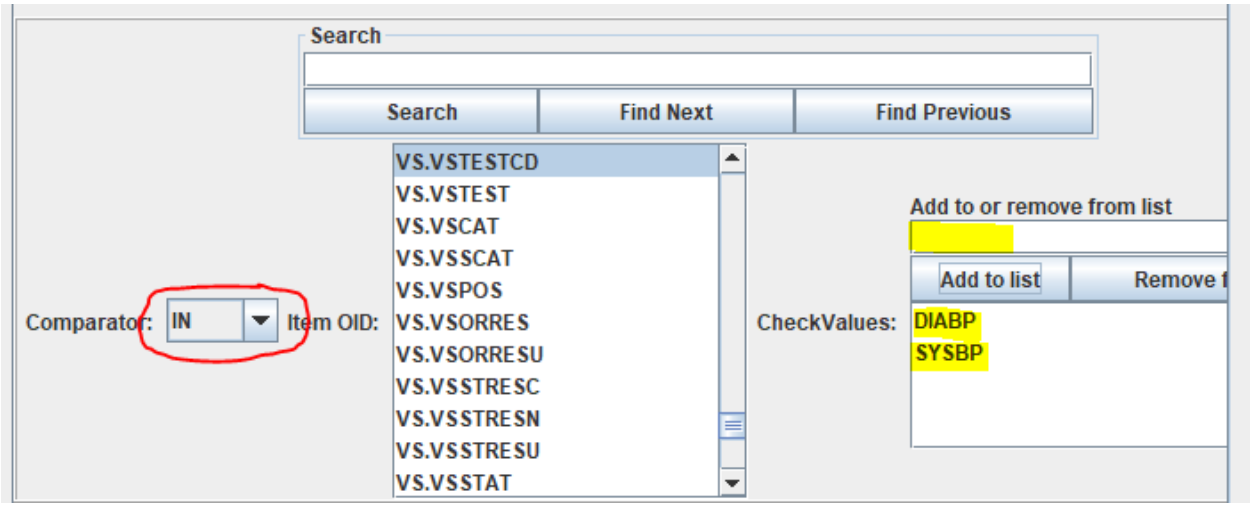


We can add a comment (this will lead to a def:Comment in the define.xml). Clicking the "Show 'Where' clause" shows the "human-readable" expression of it:

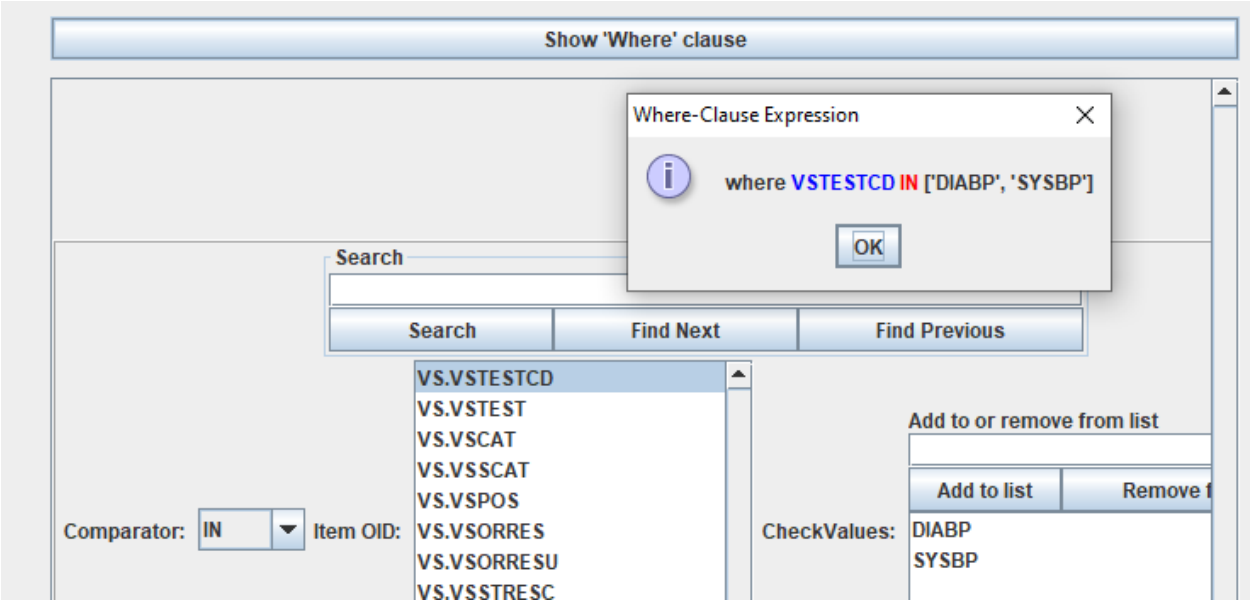


We could add additional "RangeChecks", but this will only be necessary when we need a combination of "checks". For example, we could add one for "pounds" stating "*where VSTESTCD EQ 'WEIGHT' and DM.COUNTRY EQ 'USA'*". This will however often be "overkill".

What we could do, is to combine the properties for DIABP and SYSBP. We could then use:

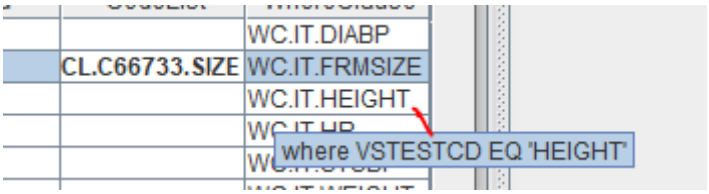


by typing "DIABP" in the field "Add to or remove from list" click "Add to list" and then do the same for "SYSBP". Clicking the "Show 'Where' clause" button then leads to:

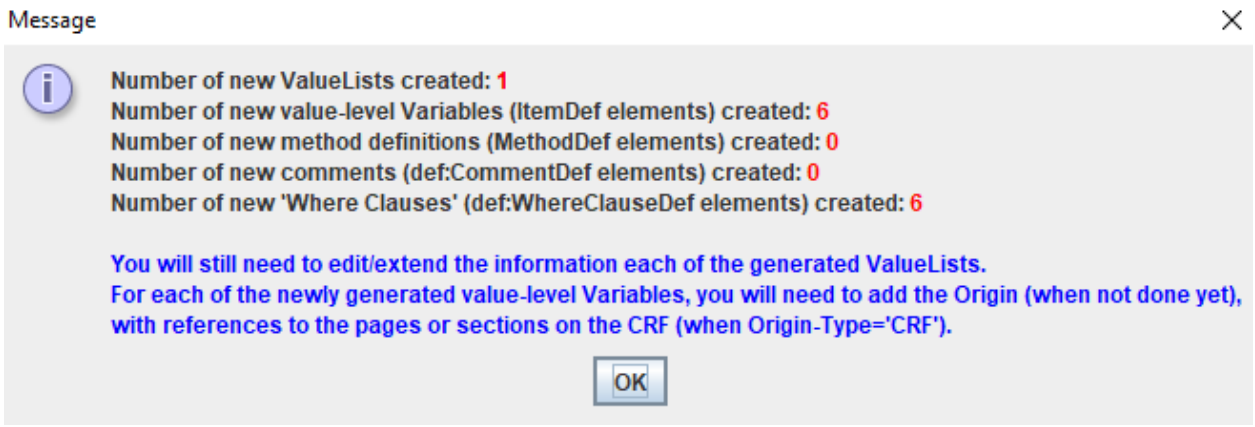


Often, this will however also be "overkill" …

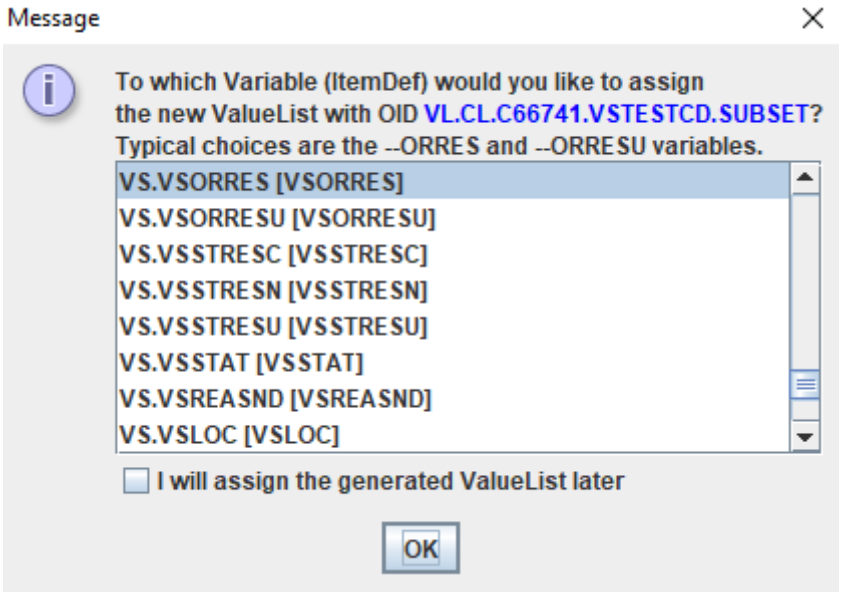
To show the "human-readable expressions, we do not necessarily go into clicking the "WhereClause" cell, we can also simply hoover the mouse over it, e.g.:



When all done, a message is displayed containing a summary:

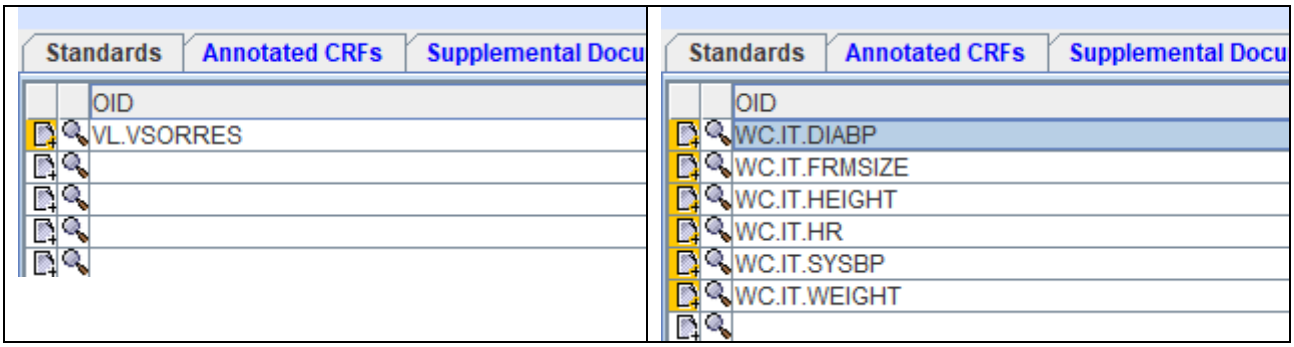


followed by a proposal to which variable the ValueList must be added:



VSORRES is the one we indeed need, so we accept that.

One then finds the newly created valuelist in the "ValueLists" tab and the newly created "WhereClause"s in the "WhereClause Definitions" tab:

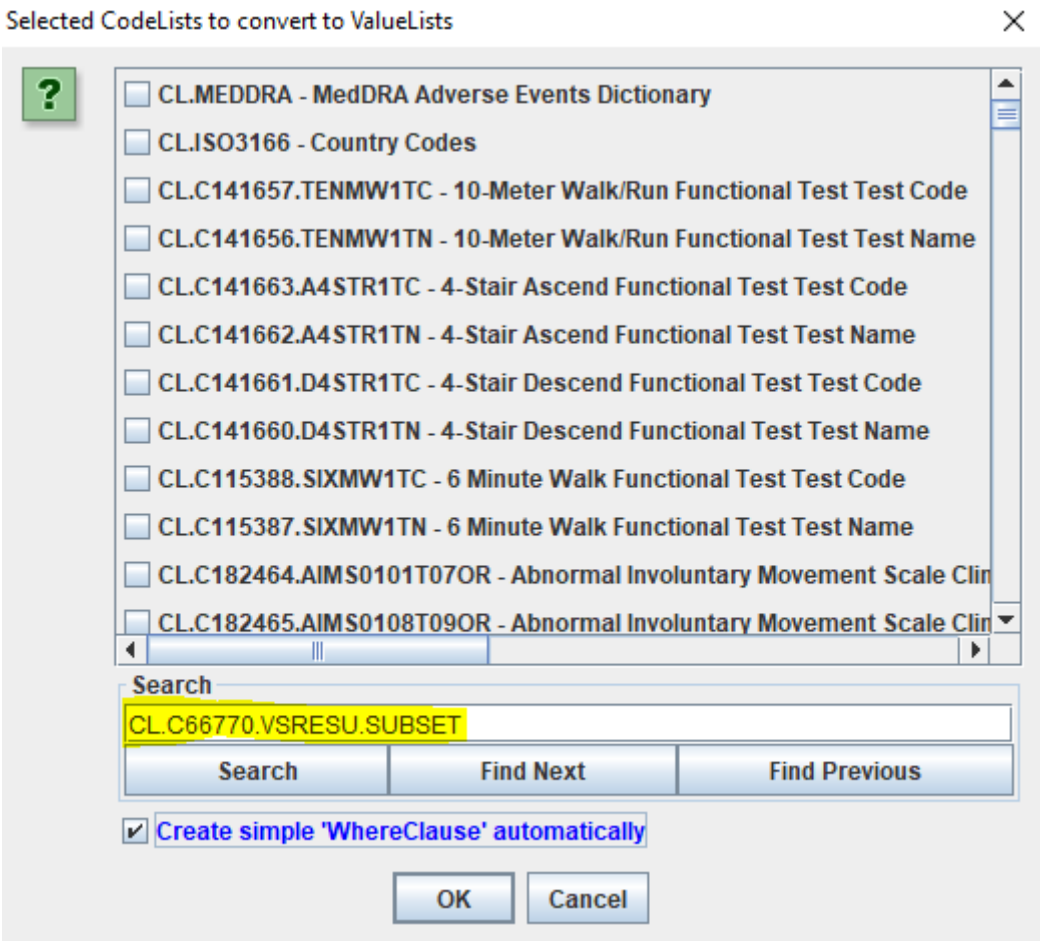


We can now also inspect the result in the "HTML View":

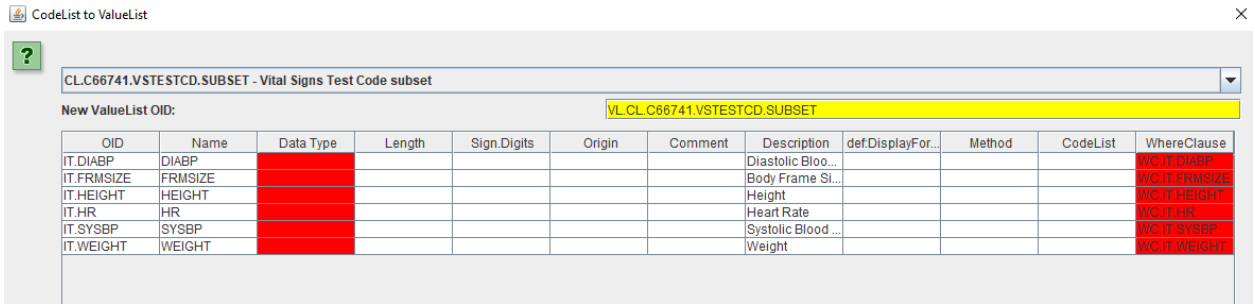
VSPOS - [Edit]		Vital Signs Position of Subject	text	Record Qualifier	23	Position - [Edit]	
VSORRES - [Edit] [Remove/Replace ValueList] [Edit ValueList]		Result or Finding in Original Units	text	Result Qualifier	80		
ValueList variable for VSORRES - [Edit]	[Edit] VSTESTCD EQ DIABP (Diastolic Blood Pressure)	Diastolic Blood Pressure	integer		3		
ValueList variable for VSORRES - [Edit]	[Edit] VSTESTCD EQ FRMSIZE (Body Frame Size)	Body Frame Size	text		6	Size Response - [Edit]	
ValueList variable for VSORRES - [Edit]	[Edit] VSTESTCD EQ HEIGHT (Height)	Height	float		5		
ValueList variable for VSORRES - [Edit]	[Edit] VSTESTCD EQ HR (Heart Rate)	Heart Rate	integer		3		
ValueList variable for VSORRES - [Edit]	[Edit] VSTESTCD EQ SYSBP (Systolic Blood Pressure)	Systolic Blood Pressure	integer		3		
ValueList variable for VSORRES - [Edit]	[Edit] VSTESTCD EQ WEIGHT (Weight)	Weight	float		5		
VSORRESU - [Edit]							

Another ValueList we may want to set up is for the unit used for the measurement which goes into VSORRESU.

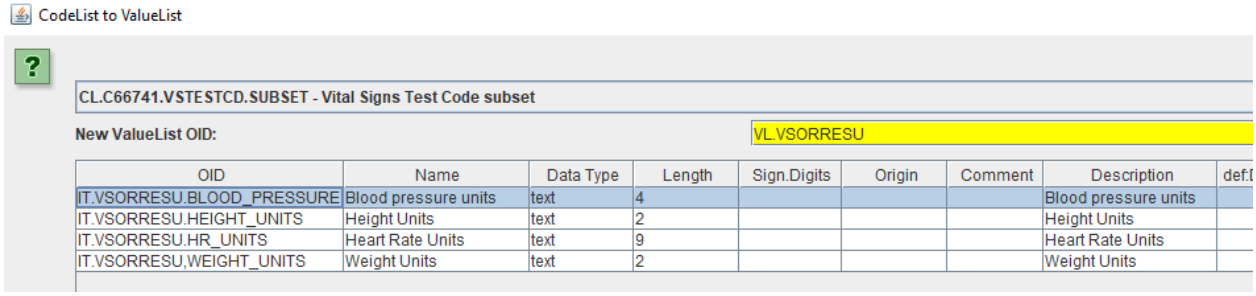
For this, we start from our VSRESU-subset codelist:



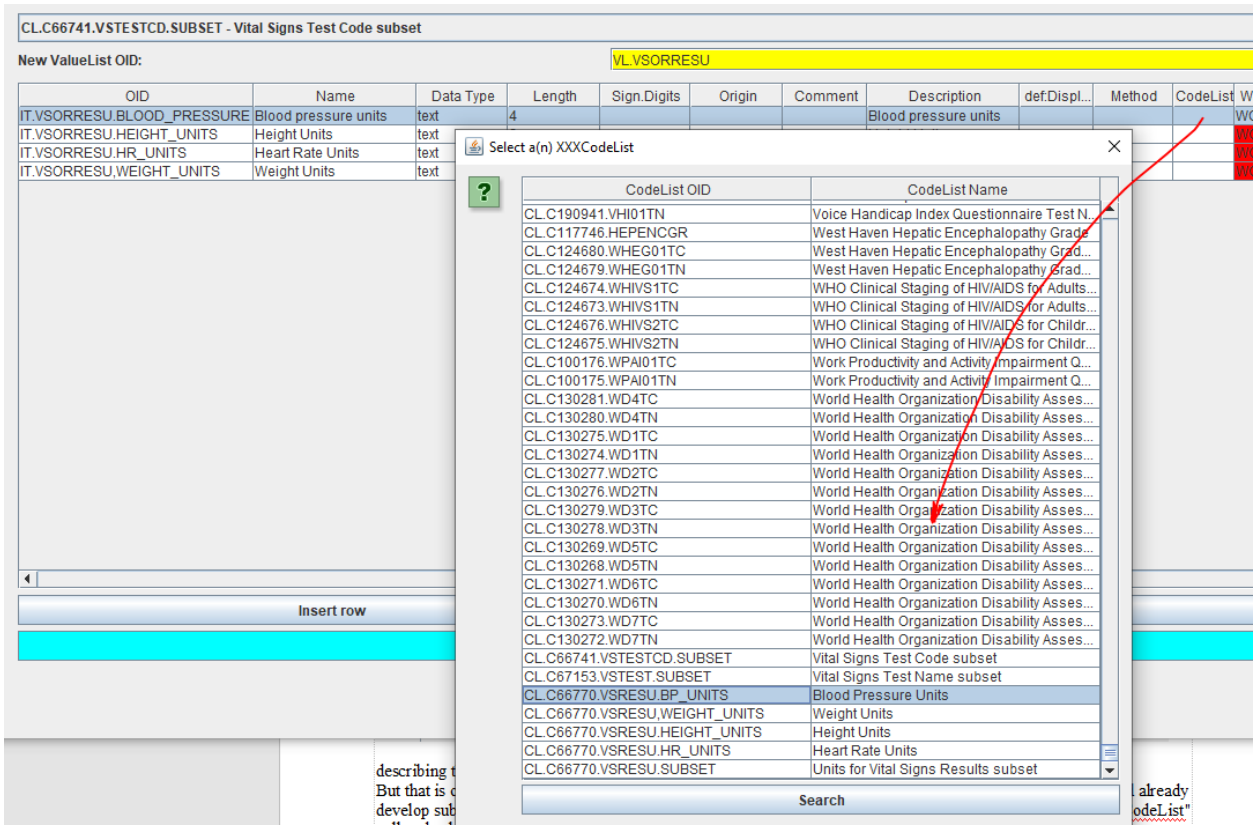
Leading to:



but essentially, this is not what we want: we want to state which of the values is used for which of the test codes, i.e. "LB" and "kg" for "WEIGHT", "cm" and "in" for "HEIGHT" etc.. So it is a question whether it really is a good idea to start from a codelist here.
We can however still start from here, and simplify e.g. to:



describing that e.g. "blood pressure units" are of type text with a maximum length of 4.
But that is of course not sufficient. We also need to e.g. add which units may be used for "weight units". We did already develop subset codelists for these, so we can add these now. So, for e.g. "Blood pressure units", we click the "CodeList" cell and select the codelist for "blood pressure units", which only contains "mmHg":

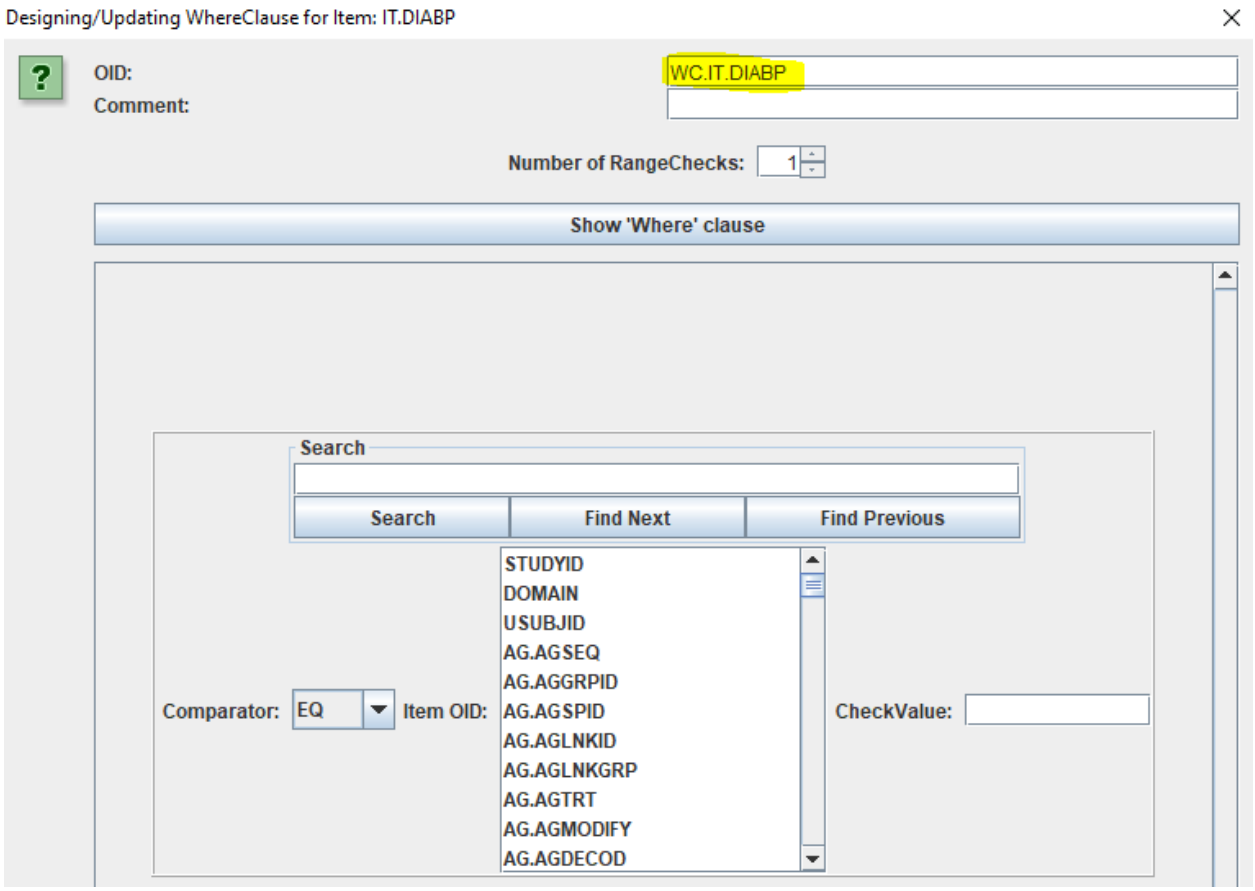


and when doing the same for "Height Units", "Weight Units" and "Heart Rate Units" leading to:

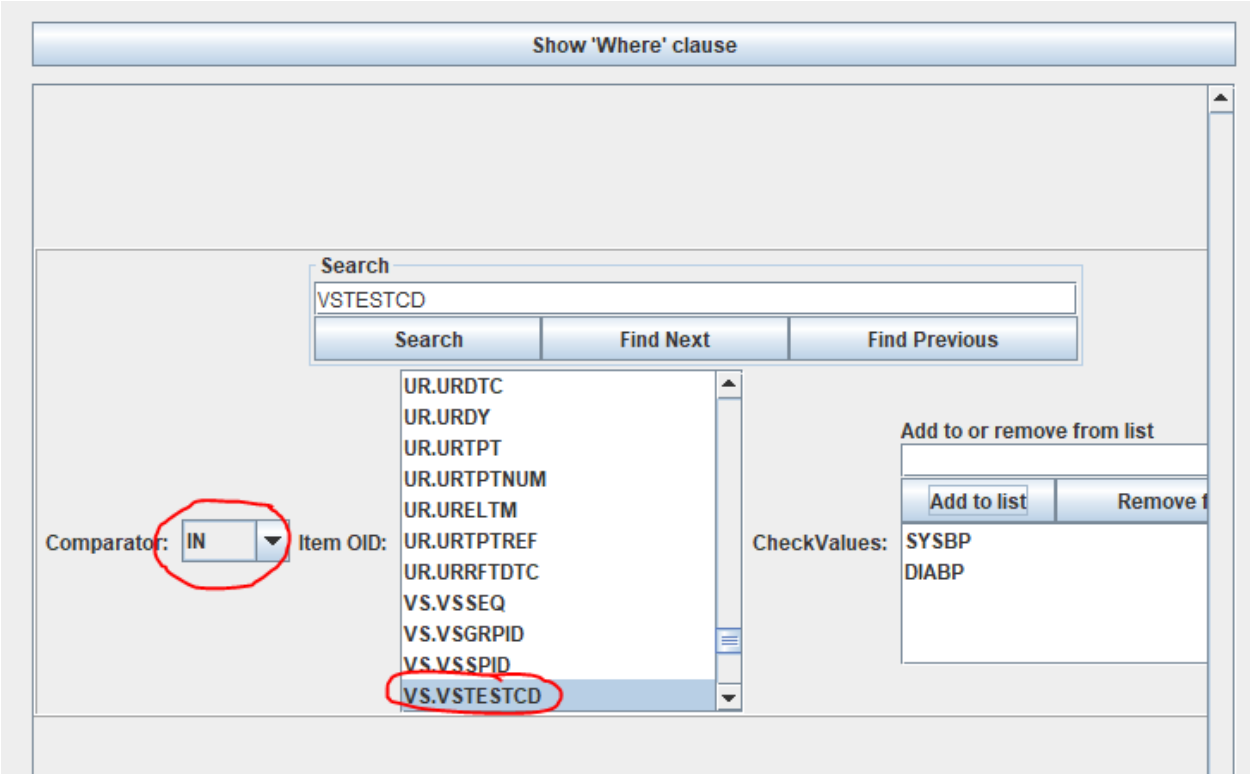
New ValueList OID: VL.VSORRESU

OID	Name	Data Type	Length	Si...	Origin	Comment	Description	def:DisplayFo...	Method	CodeList	WhereClause
IT.VSORRESU.BLOOD_PRESSU...	Blood pressure units	text	4				Blood pressure units			CL.C66770.VSRESU.BP_UNITS	WC.IT.DIABP
IT.VSORRESU.HEIGHT_UNITS	Height Units	text	2				Height Units			CL.C66770.VSRESU.HEIGHT_UNITS	WC.IT.HEIGHT
IT.VSORRESU.HR_UNITS	Heart Rate Units	text	9				Heart Rate Units			CL.C66770.VSRESU.HR_UNITS	WC.IT.HR
IT.VSORRESU.WEIGHT_UNITS	Weight Units	text	2				Weight Units			CL.C66770.VSRESU.WEIGHT_UNITS	WC.IT.WEIGHT

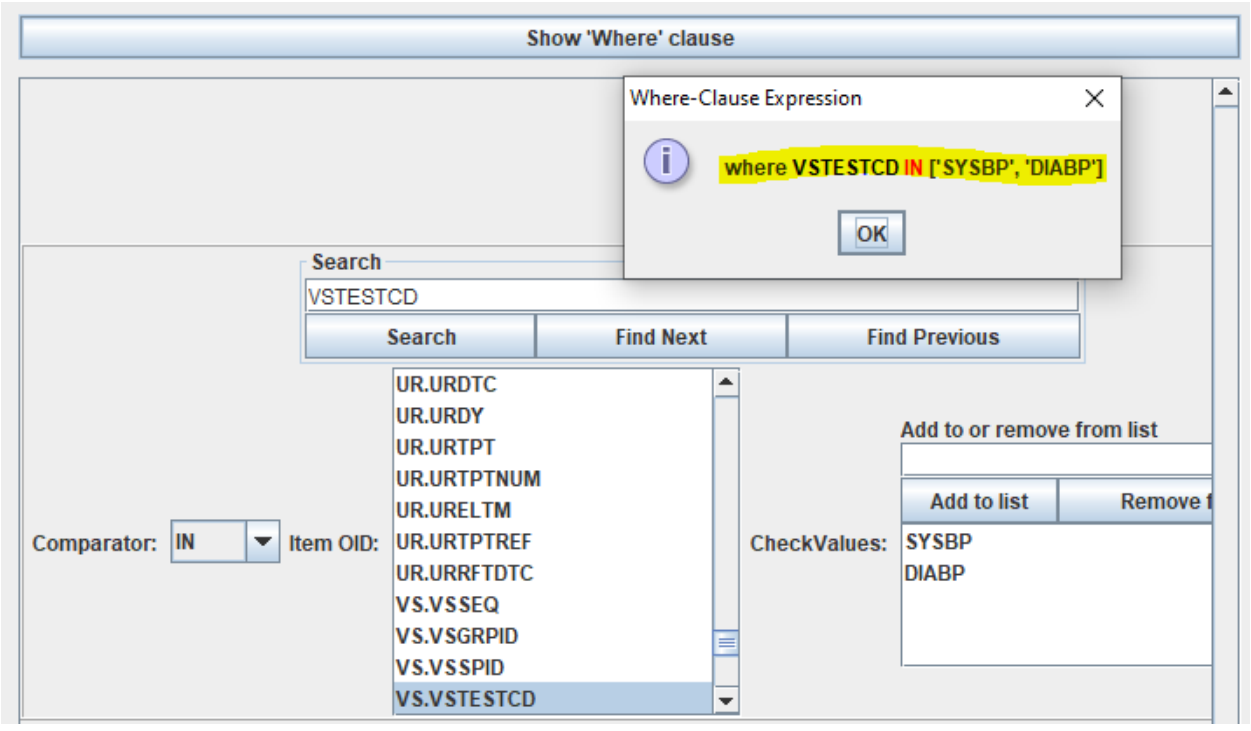
When validating however, we still see that we still need to develop the "Where Clauses". Clicking the one "WC.IT.DIABP", leads to:



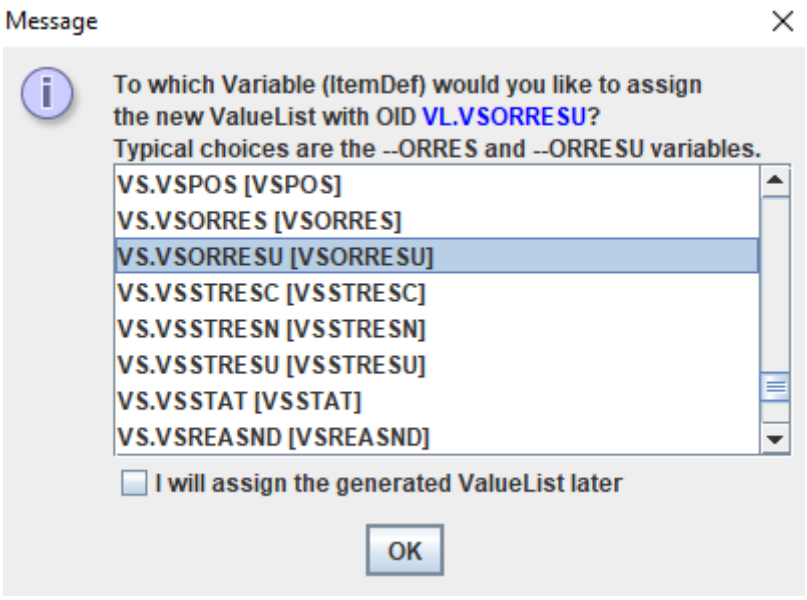
I.e. containing no information at all …
As "WC.IT.DIABP" is not well describing what this is about, we change it into WC.IT.BLOOD_PRESSURE (unfortunately WhereClause elements do not have a "Name" attribute, nor have a "Description" child element), and than add the cases under which one of the "blood pressure units", i.e. only "mmHg" will be used, which is VSTESTCD=DIABP or VSTESTCD=SYSBP:



and when using the "Show 'Where' clause" button, we get:



We then follow the same procedure for "Weight Units", "Height Units", and "Heart Rate Units".
When all is OK (good idea to each time click the "Validate" button), and everything is done, clicking "OK" first shows us a message with an overview, and then asks us to which variable we want assign this newly created ValueList, where we of course select VSORRESU:



When then doing an "HTML View", we find:

VSORRES - [Edit]		Result or Finding in Original Units	text	Result Qualifier	80	
VSORRESU - [Edit] [Remove/Replace ValueList] [Edit ValueList]		Original Units	text	Variable Qualifier	11	Units for Vital Signs Results - [Edit]
ValueList variable for VSORRESU - [Edit]	[Edit] VSTESTCD IN [SYSBP (Systolic Blood Pressure), DIABP (Diastolic Blood Pressure)]	Blood pressure units	text		4	Blood Pressure Units - [Edit]
ValueList variable for VSORRESU - [Edit]	[Edit] VSTESTCD EQ HEIGHT (Height)	Height Units	text		2	Height Units - [Edit]
ValueList variable for VSORRESU - [Edit]	[Edit] VSTESTCD EQ HR (Heart Rate)	Heart Rate Units	text		9	Heart Rate Units - [Edit]
ValueList variable for VSORRESU - [Edit]	[Edit] VSTESTCD EQ WEIGHT (Weight)	Weight Units	text		2	Weight Units - [Edit]
VSSTRESC - [Edit]		Character Result/Finding		Result		

A few remarks:
- We did not use "frame size" here, as frame size has no units.
- Generating ValueLists for xxORRESU (Original Units) is a matter of taste and choice. There is no formal obligation to do so. As we have seen, it is a bit more complicated than generating ValueLists for xxORRES variables. The latter are mostly very helpful for the reviewers. When developing a "high quality" define.xml, one should always ask oneself: "*do I do the reviewer a pleasure adding this ValueList or not?*" and "*does it help the reviewer better understanding the datasets?*".

ValueLists for ADaM - a simple example

It can also be useful to develop generate ValueLists for ADaM even when no data is available yet, but one already knows what parameters (PARAMCD) one will in a specific analysis dataset.

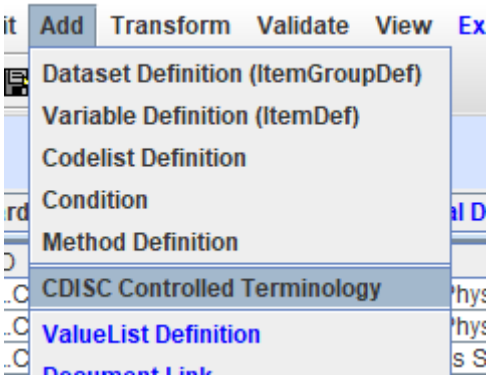
For example, when one is planning an ANALYSIS dataset "Pharmacokinetic Parameters Analysis Dataset" (ADPP), with the parameters AUCOT (Area Under the Curve from 0 to Last Observation), AUCINF (Area Under the Curve from 0 to Infinity), CL (Clearance), CMAX (Maximum Observed Concentration), TMAX (Time of Maximum Observed Concentration), HALF (Terminal Half-Life), and VSS (Volume of Distribution at Steady State), and want to indicate which units were used for which parameter, one can already develop the ValueList from the planning information (e.g. Statistical Analysis Plan).

For the units, we do know which one will be used:

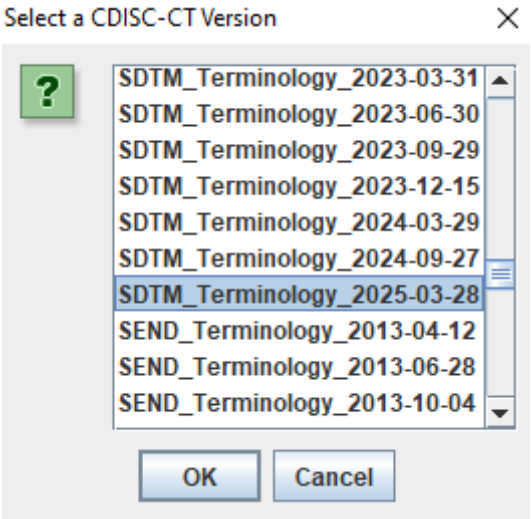
PARMCD (Parameter Code)	Description	AVALU unit
AUCOT	Area Under the Curve from 0 to Last Observation	ng*h/mL
AUCINF	Area Under the Curve from 0 to Infinity	ng*h/mL
CL	Clearance	mL/min
CMAX	Maximum Observed Concentration	ng/mL
HALF	Terminal Half-Life	h
TMAX	Time of Maximum Observed Concentration	h
VSS	Volume of Distribution at Steady State	L

For PARMCD for Pharmacokinetic Parameters , there is no specific controlled terminology from CDISC, so we want to make a "sponsor-defined codelist". For "Unit" there is no CDISC-CT in ADaM either, but there is a PKUNIT CodeList (C85494) in SDTM. It has all our needed terms except for "ng*h/mL" and (surprisingly) "mL/min". So for the AVALU variable, we will start from that SDTM codelist.

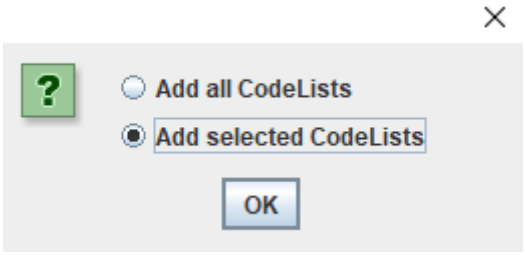
As we currently only have loaded ADaM-CT, we need to "import" this SDTM codelist first. For this, we use the menu "Add - CDISC Controlled Terminology":



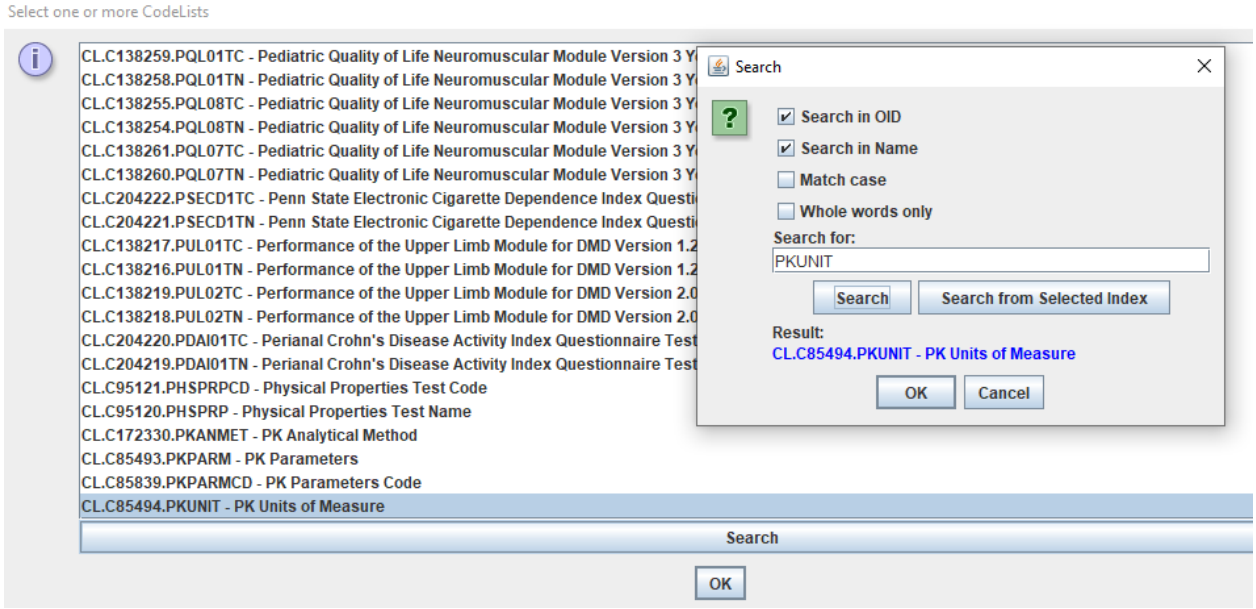
The system then asks us what version of CT we want to import. Scrolling down to the latest SDTM-CT:



It then asks whether we want to add all codelists, or just a single one. We select "Add Selected CodeLists" as we only want to import the PKUNIT codelist:

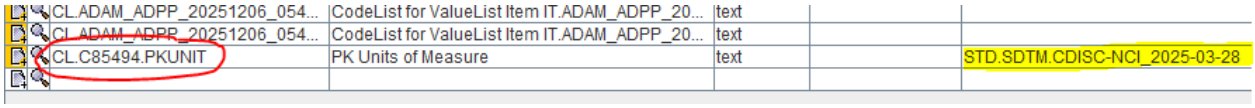


It then presents a list, which we search and find PKUNIT:

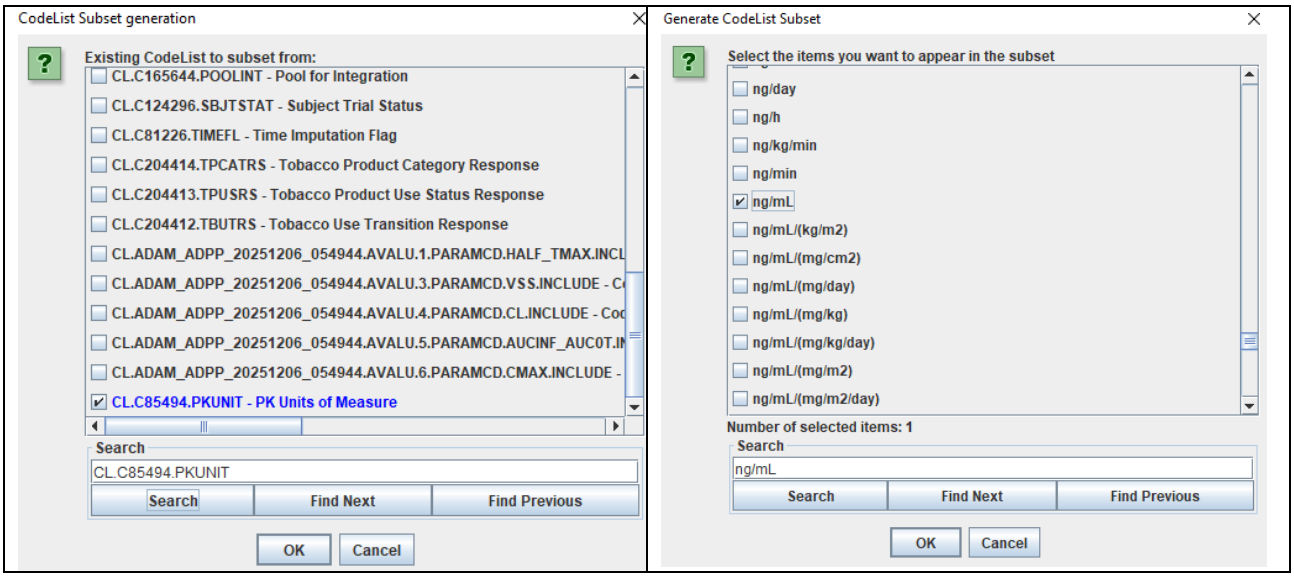


(Remark that the list allows to select several codelists).

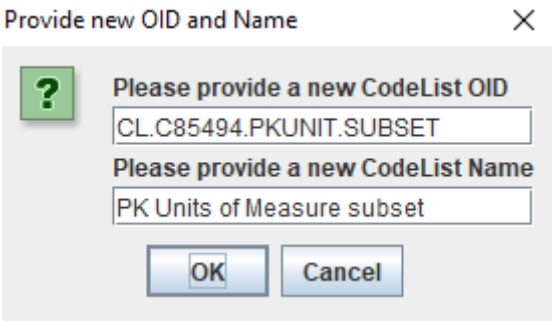
After "OK", in the "CodeList Definitions" tab, this leads to:



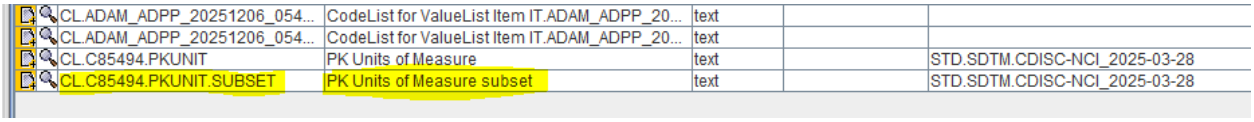
which is an SDTM codelist, but that's just fine.
We can now either edit this codelist or make a subset of it containing only the units that we need. Making a subset is faster, so we do that, using the menu "Edit - Generate Subset CodeList", and then checking the units we want to keep (there are only 3, as 2 of the 5 are not in the list):



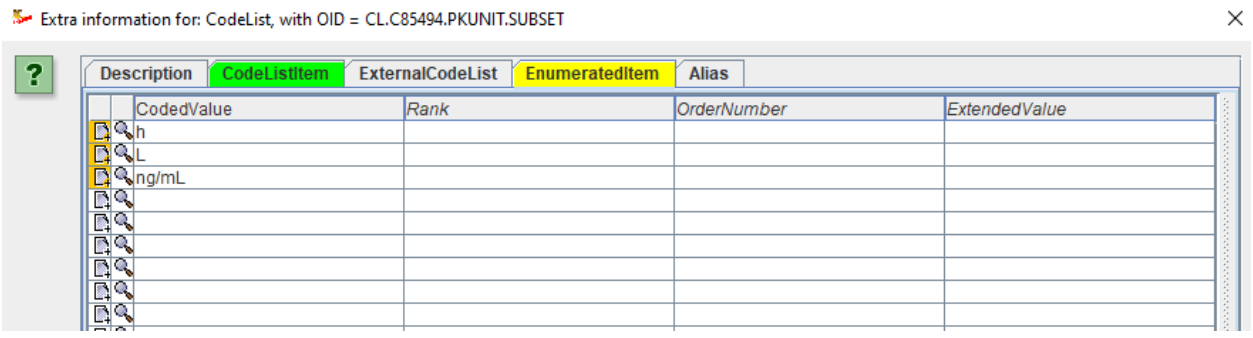
The system then proposes an OID (identifier) and a Name, which is just fine:



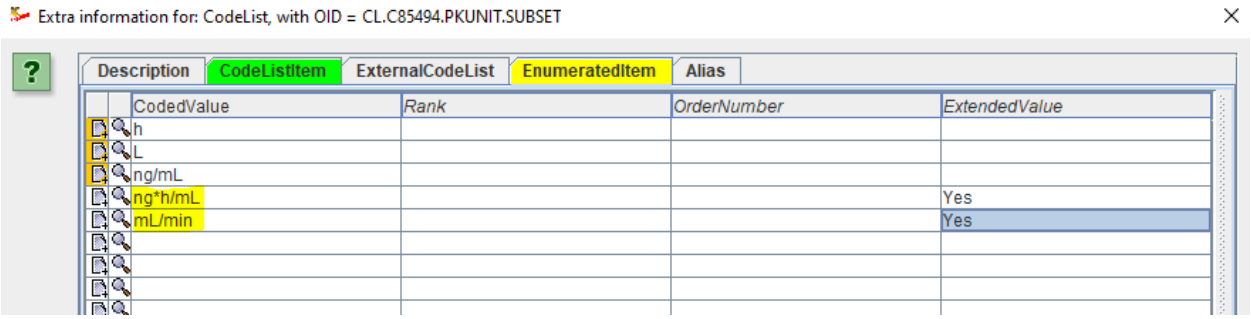
After "OK", the CodeList is generated, and we see it in our list in the "CodeList Definitions" tab:



We then still need to add "ng*h/mL" and "mL/min" to it, which will then be considered "Extended" values. So we click the "Edit" icon (first on the left), and choose the tab "EnumeratedItem" (as there are no "translations" for the units) leading to:

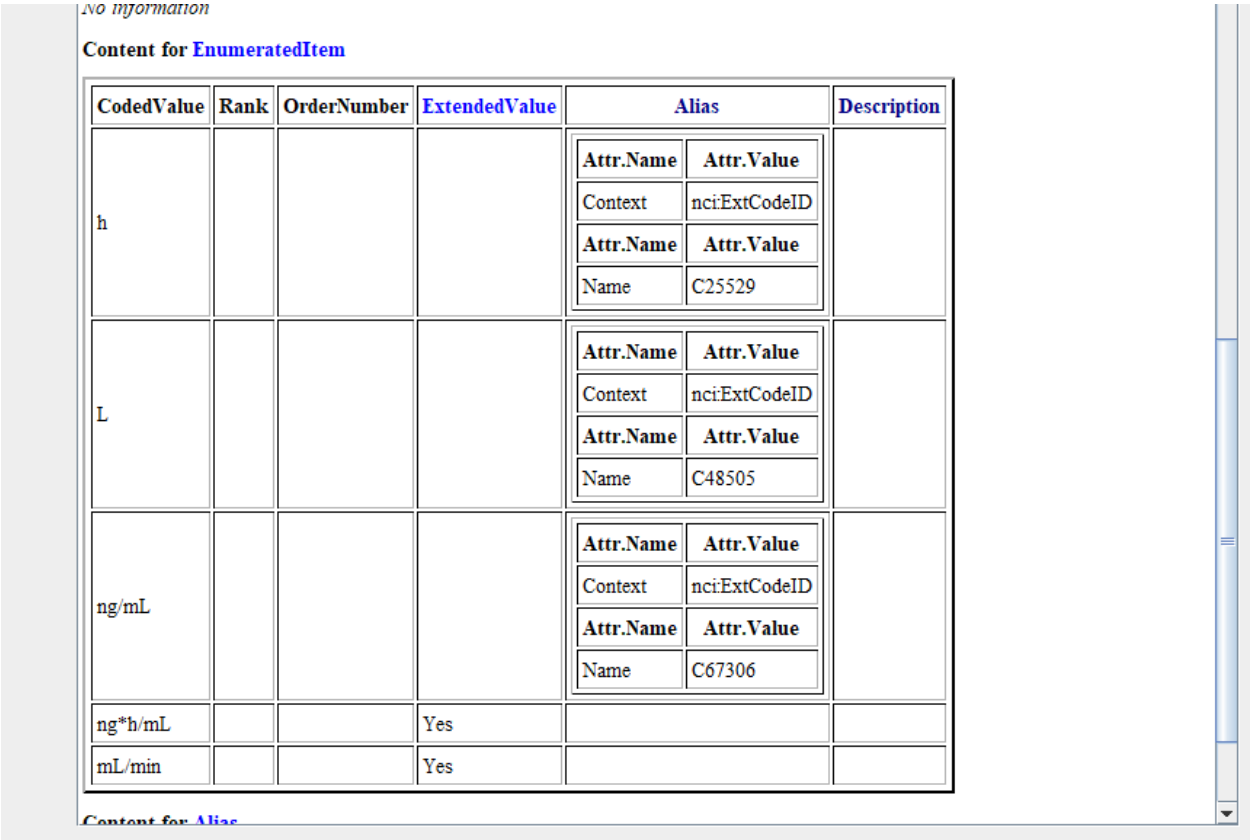


In the first empty cells (4th and 5th line) we can now add our 2 other units:

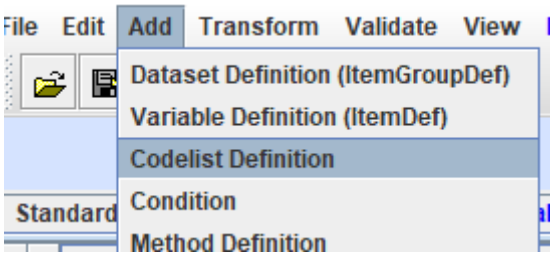


Do not forget to mark them as "ExtendedValue". If we forgot and use the button "Validate", the system will complain:

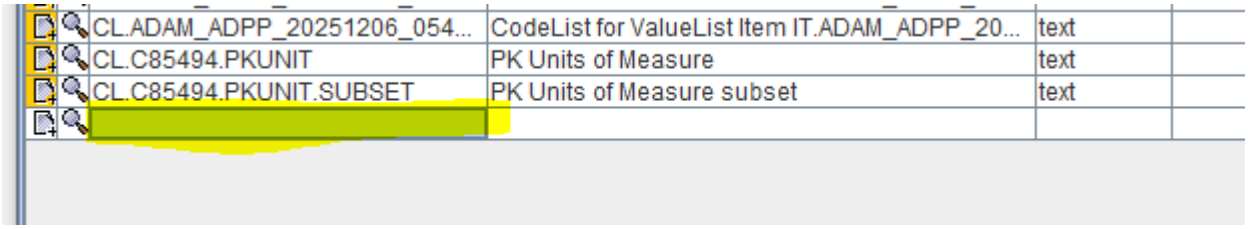
Clicking "OK", the updates the CodeList. We can then still check using the "View" icon (the one with the magnifying glass):



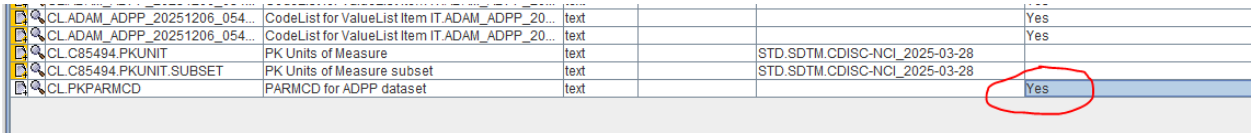
Let us set up a new CodeList with the PARMCD values. In order to do so, use the menu "Add - CodeList Definition":



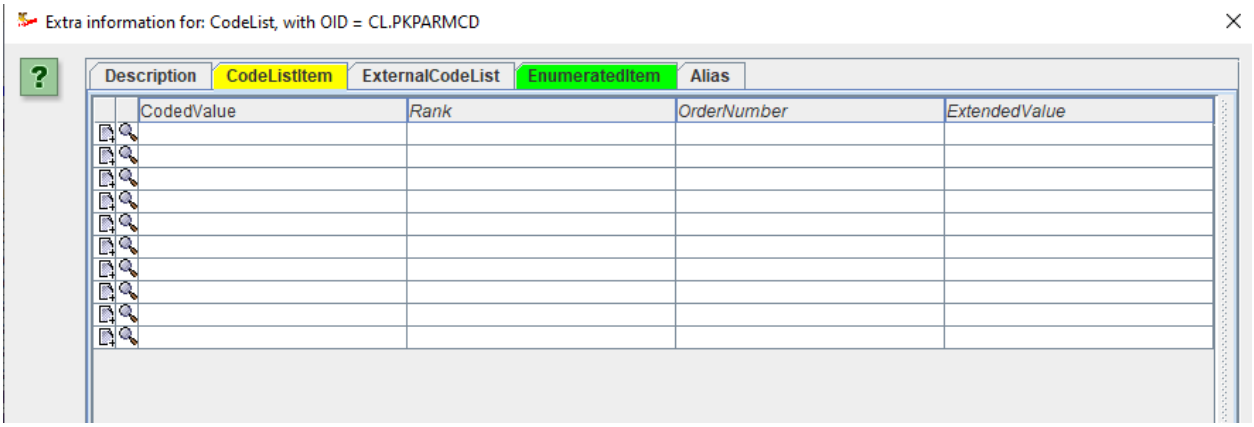
This creates an empty row in the table of our "CodeList Definitions" tab table:



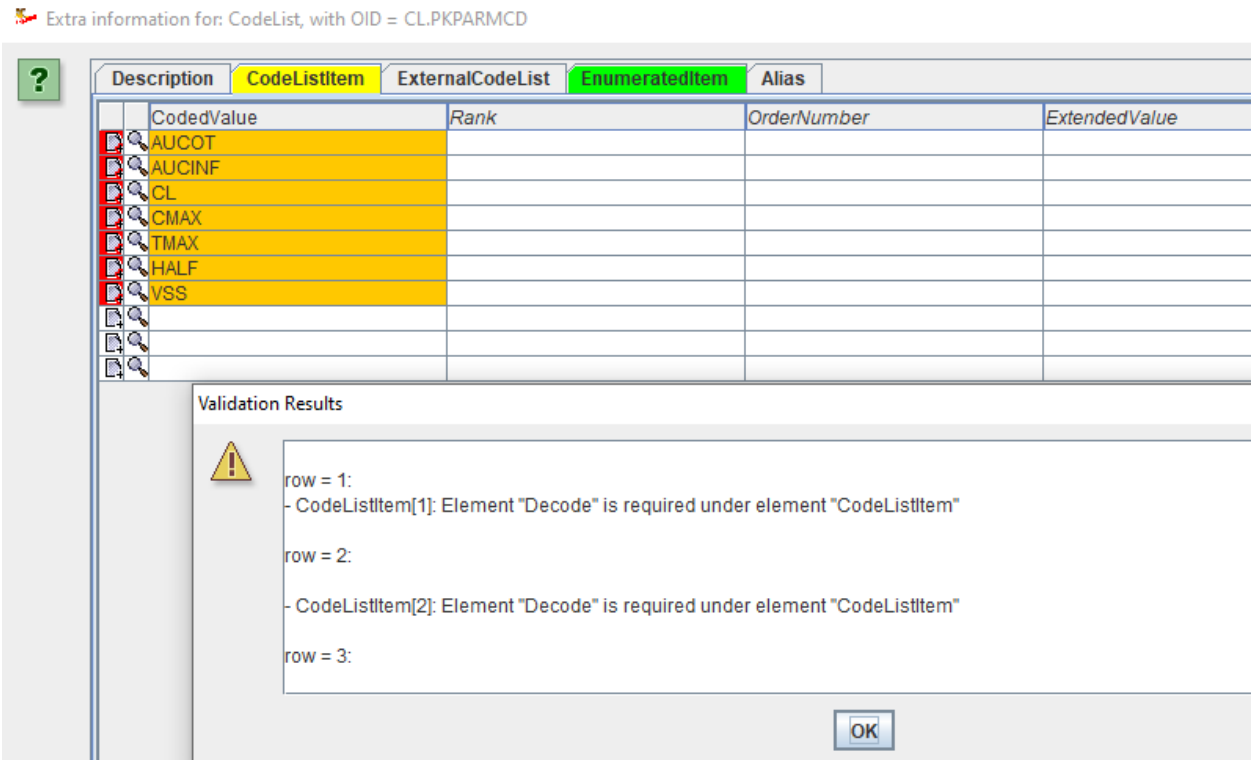
We assign it an OID (identifier), and a Name, and set the datatype to "text", and set "IsNonStandard" to "Yes". For example:



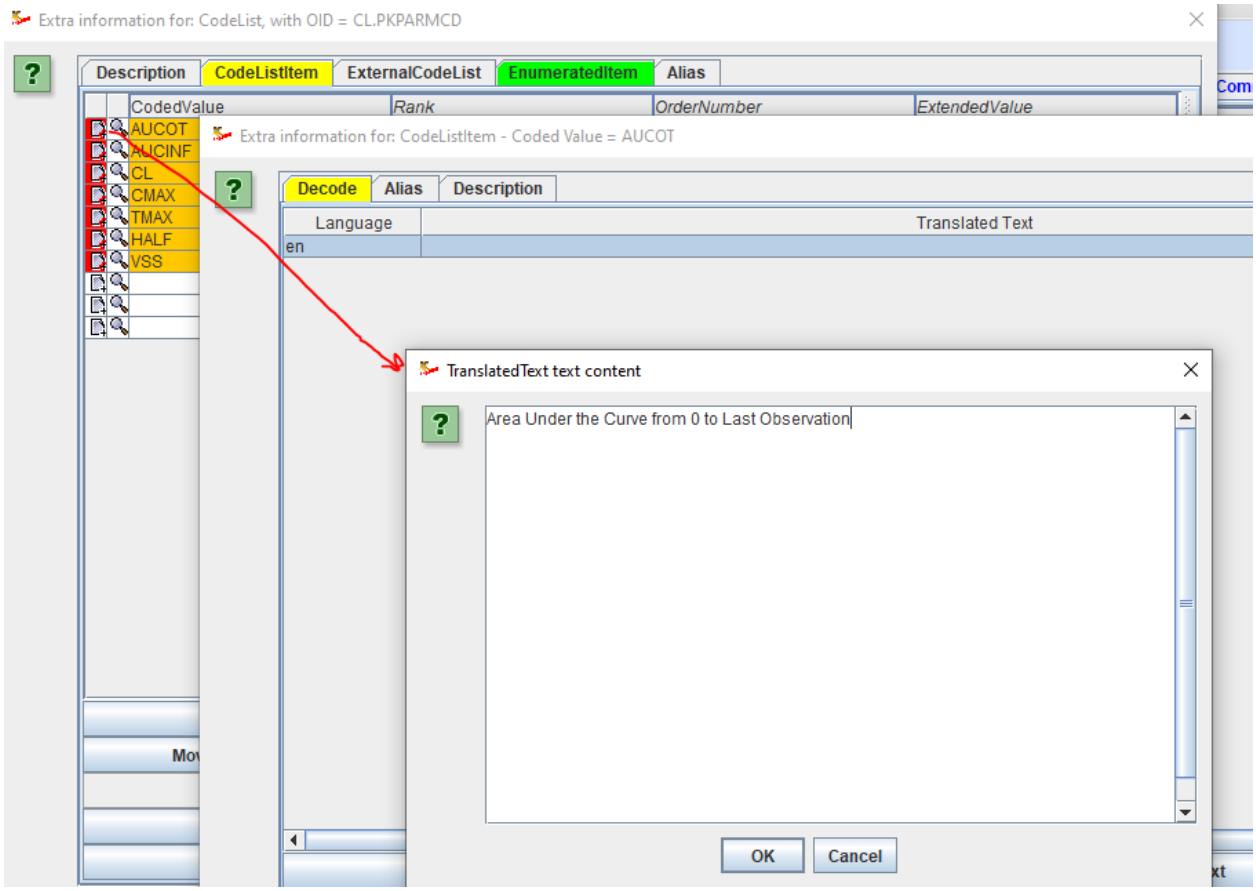
We can then start adding the parameter codes and names by clicking the "Edit" icon on the left, leading to an empty table for which we select the "CodeListItem" tab, as we also want to add the "decoded" values, i.e. the description of each of the PARAMCD values:



which we then fill with the values for PARAMCD and then click the "Validate" button:

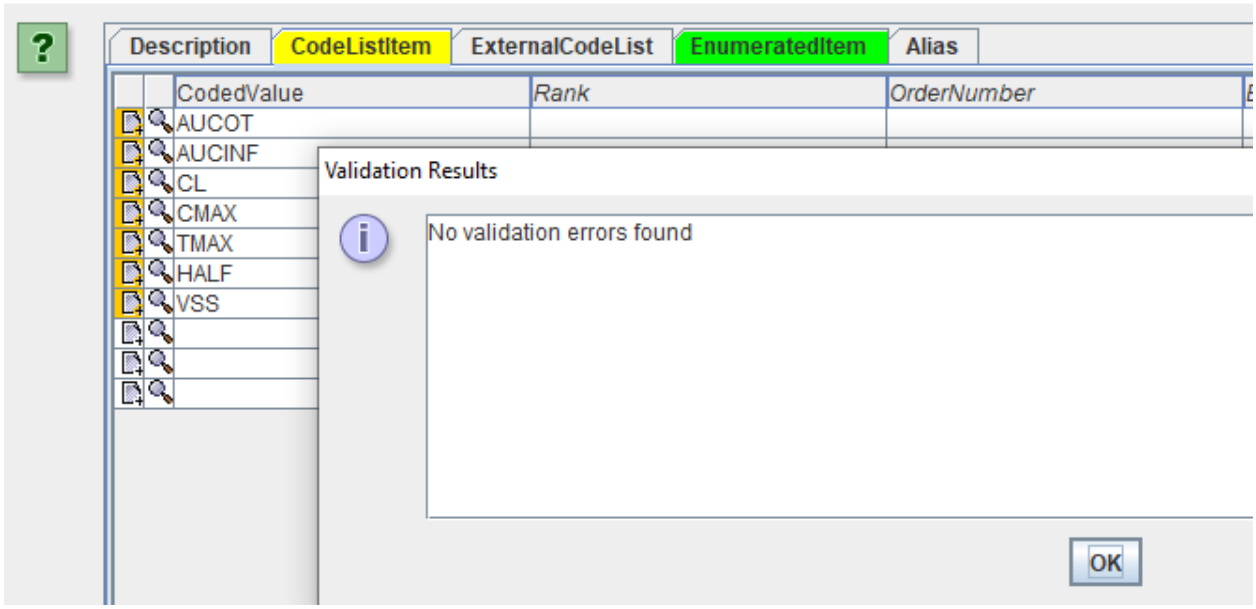


Notice that the "Edit" icon changes color to red, meaning that something is missing.
stating that when we use "CodeListItem", we must also add "Decode" values. So, for each of them we click the "Edit" icon and then also add the "decode" information. E.g.:

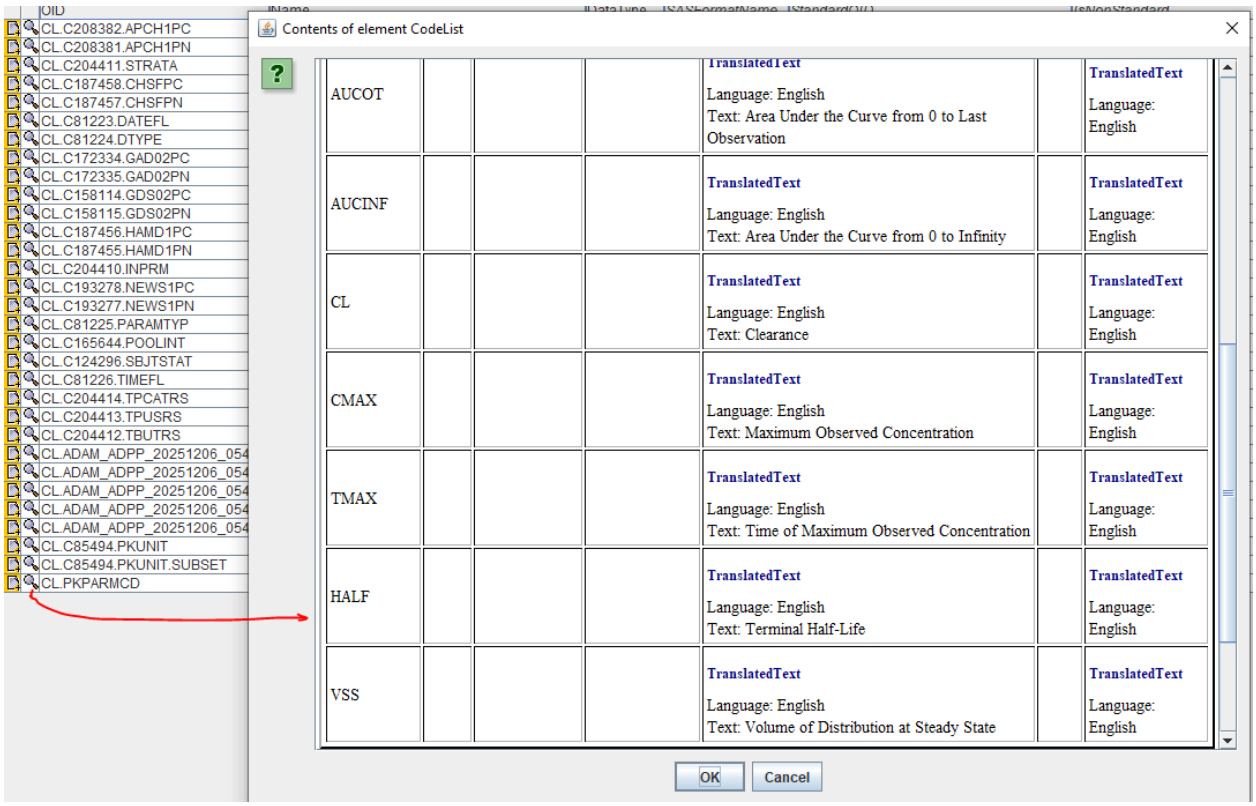


and of course also for the other ones.

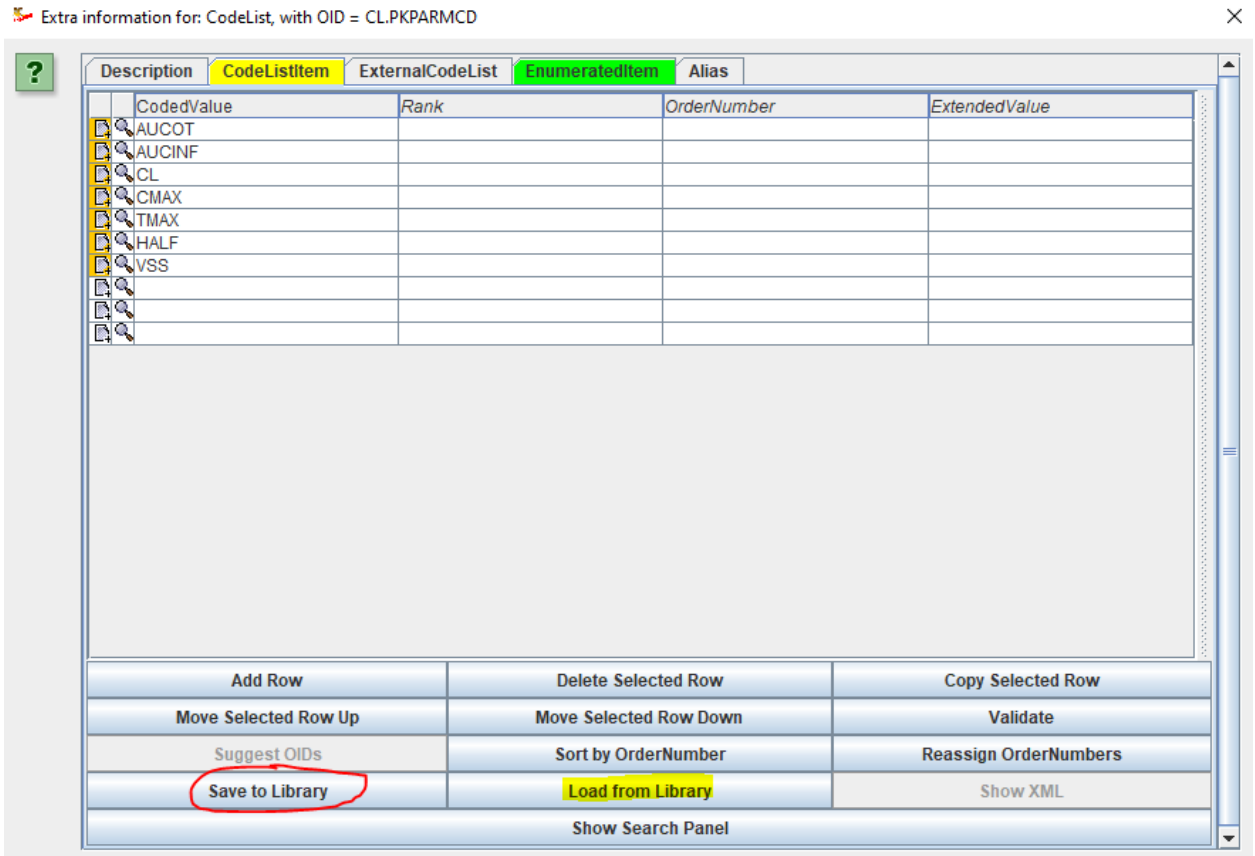
When all done, and clicking the "Validate" button again, we get (if we have done everything right):



After we go back to the main table by clicking "OK" until we get there, we can still inspect our work by clicking the "View" icon (magnifying glass):



As one can guess, generating such "sponsor-defined" codelists is something we want to do over and over again for each study. Using the button "Save to Library":



we can then save the codelist to file (in Define-XML format) and when needed in another study, just use "Load from Library". This means that the software fully supports reuse of Define-XML objects.

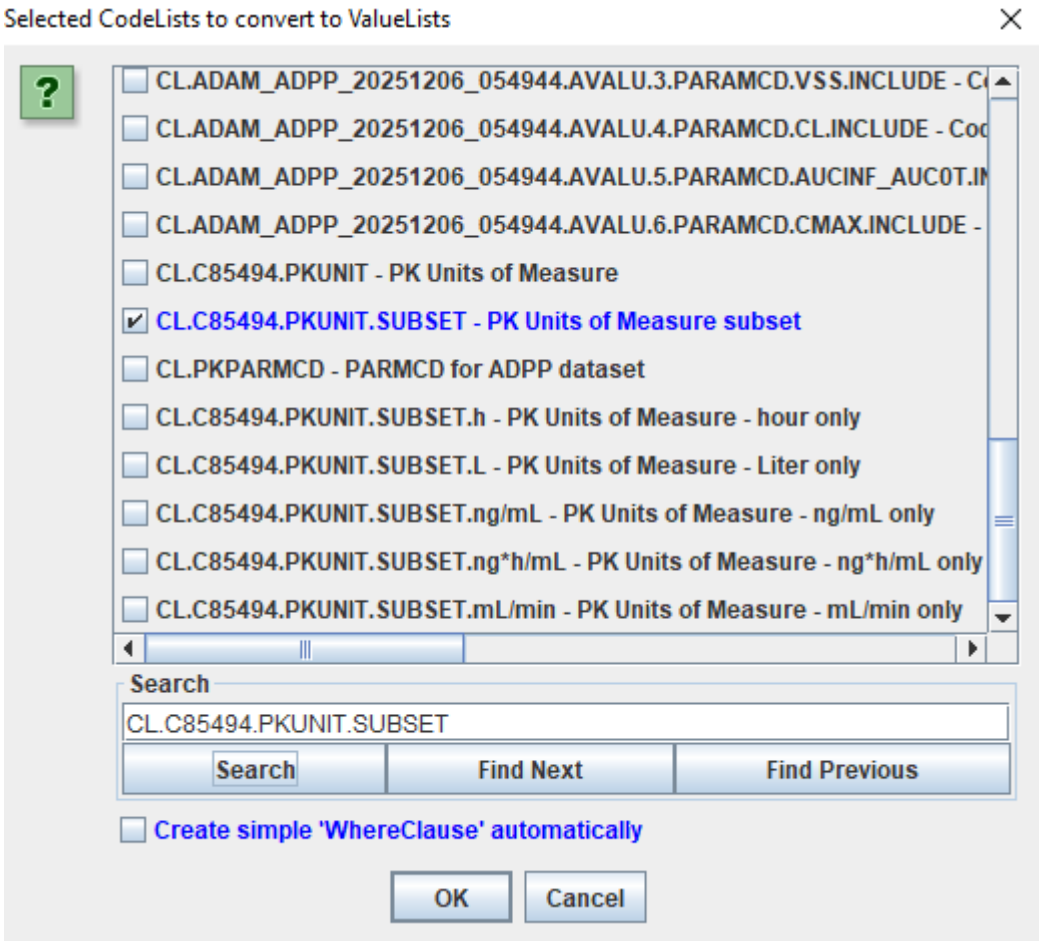
Now that we have our PARAMCD CodeList and our "PKUNIT subset" CodeList, we just still need to generate "single-

item" codelists for units, as in our ValueList, we want to state that when e.g. PARAMCD = AUCOT (Area Under the Curve from 0 to Last Observation), then the only unit (in AVALU) is "ng*h/mL".
This means that we need our "PKUNIT subset" to be "splitted" in different codelists each containing one single value.
To do so, we select the codelist, and then use the menu "Extra - Generate single-item CodeList from (subset) CodeList".
This then leads to the following new codelists:

CL.ADAM_ADPP_20251206_054944.AVAL...	CodeList for ValueList Item IT.ADAM_ADPP_...	text
CL.C85494.PKUNIT	PK Units of Measure	text
CL.C85494.PKUNIT.SUBSET	PK Units of Measure subset	text
CL.PKPARMCD	PARMCD for ADPP dataset	text
CL.C85494.PKUNIT.SUBSET.h	PK Units of Measure - hour only	text
CL.C85494.PKUNIT.SUBSET.L	PK Units of Measure - Liter only	text
CL.C85494.PKUNIT.SUBSET.ng/mL	PK Units of Measure - ng/mL only	text
CL.C85494.PKUNIT.SUBSET.ng*h/mL	PK Units of Measure - ng*h/mL only	text
CL.C85494.PKUNIT.SUBSET.mL/min	PK Units of Measure - mL/min only	text

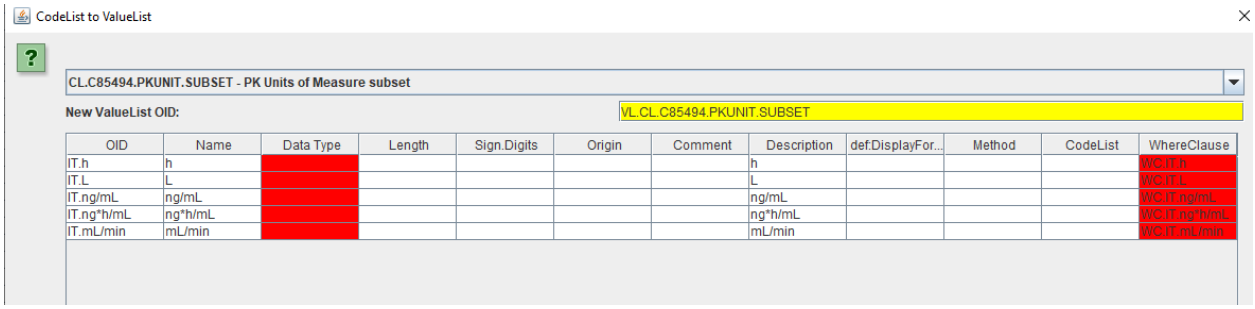
OK! Let's start generating the ValueList!

We start from the codelist "CL.C85494.PKUNIT.SUBSET" that has our 5 units. We use the menu "Transform - CodeList to ValueList", and select the codelist:

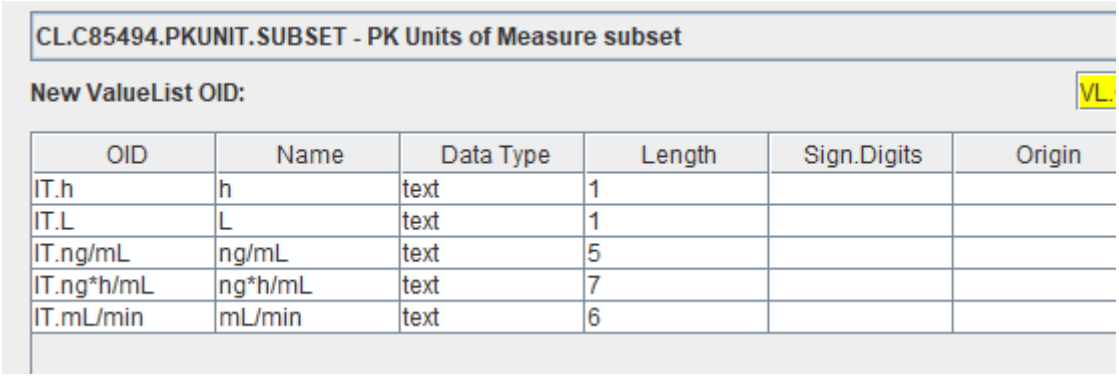


This time, we do not check "Create simple 'WhereClause' automatically", as we suspect that the system will not be able to make an educated guess what our units will be dependent on anyway.

In the next step, a prototype ValueList is generated:



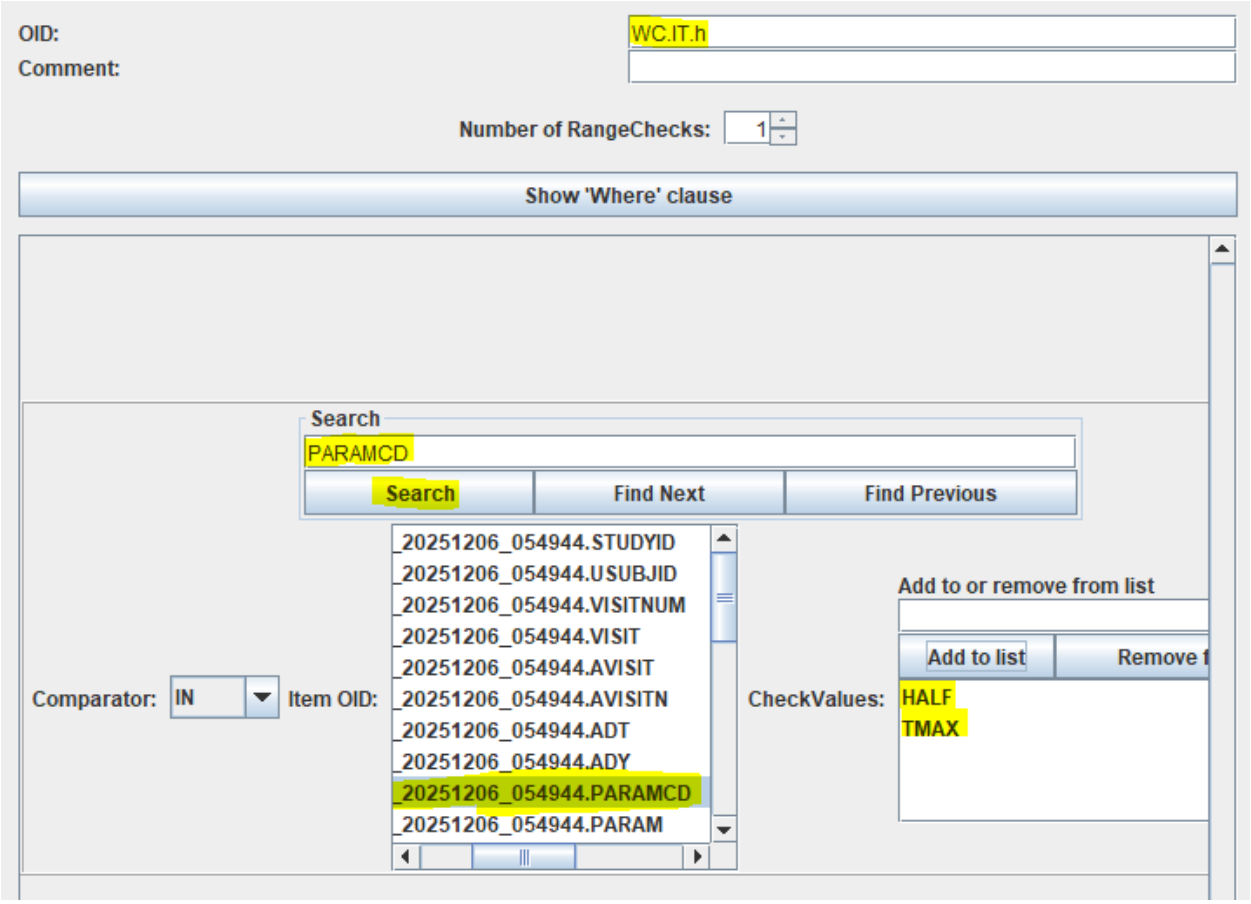
For each of the 5 units, the data type is of course "text", and the length just the number of characters. This leads to:



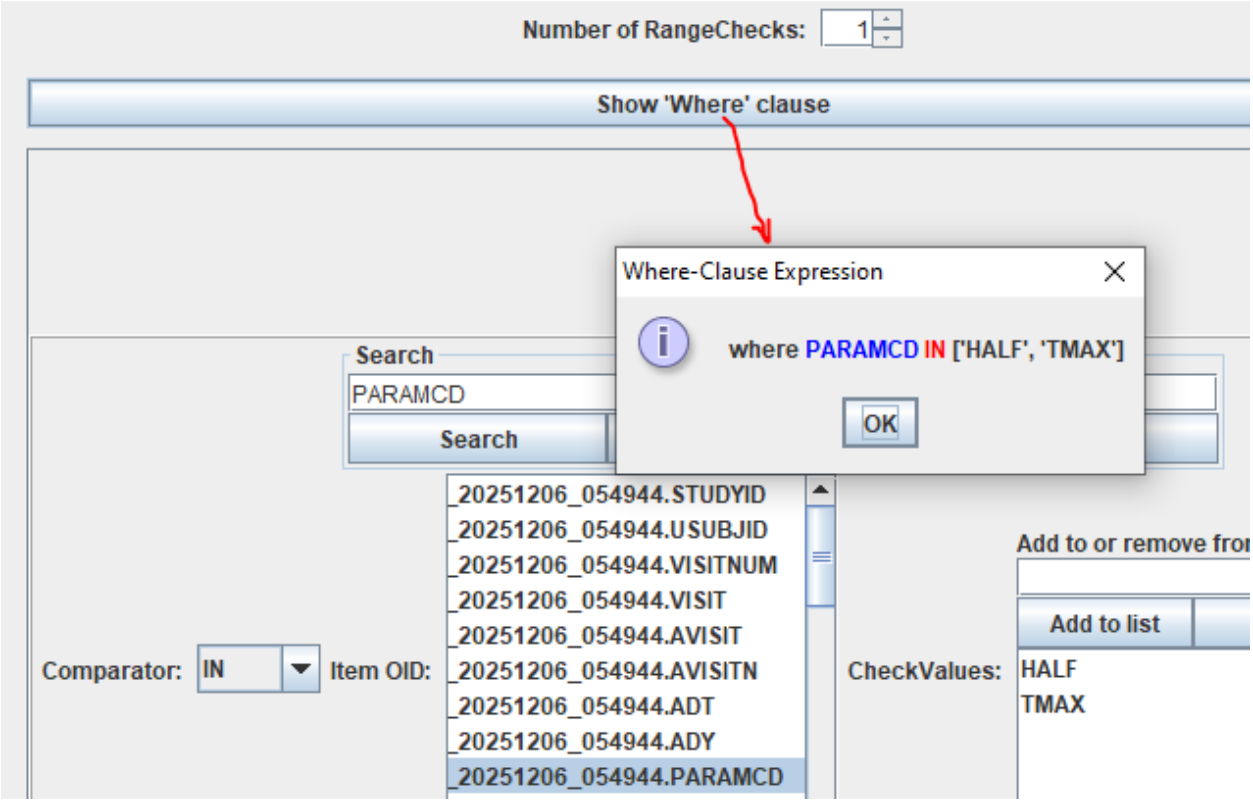
Now, for each of the "cases", we need to add the "where-clause". The table we start from is again:

PARMCD (Parameter Code)	**Description**	**AVALU unit**
AUCOT	Area Under the Curve from 0 to Last Observation	ng*h/mL
AUCINF	Area Under the Curve from 0 to Infinity	ng*h/mL
CL	Clearance	mL/min
CMAX	Maximum Observed Concentration	ng/mL
HALF	Terminal Half-Life	h
TMAX	Time of Maximum Observed Concentration	h
VSS	Volume of Distribution at Steady State	L

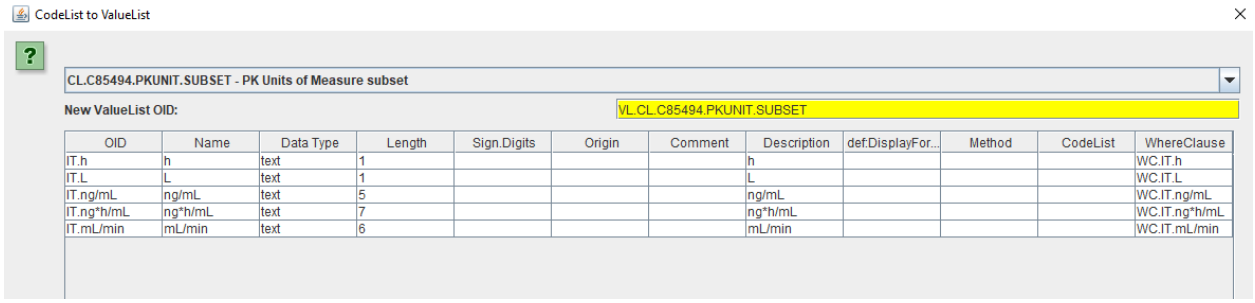
So, for the unit "ng*h/mL" we need to make the statement "WHERE PARAMCD EQ AUCOT", and for the unit "h", we need to make the statement "WHERE PARAMCD IN ["HALF","TMAX"], etc..
Clicking the cell "WC.IT.h", we add exactly that information as:



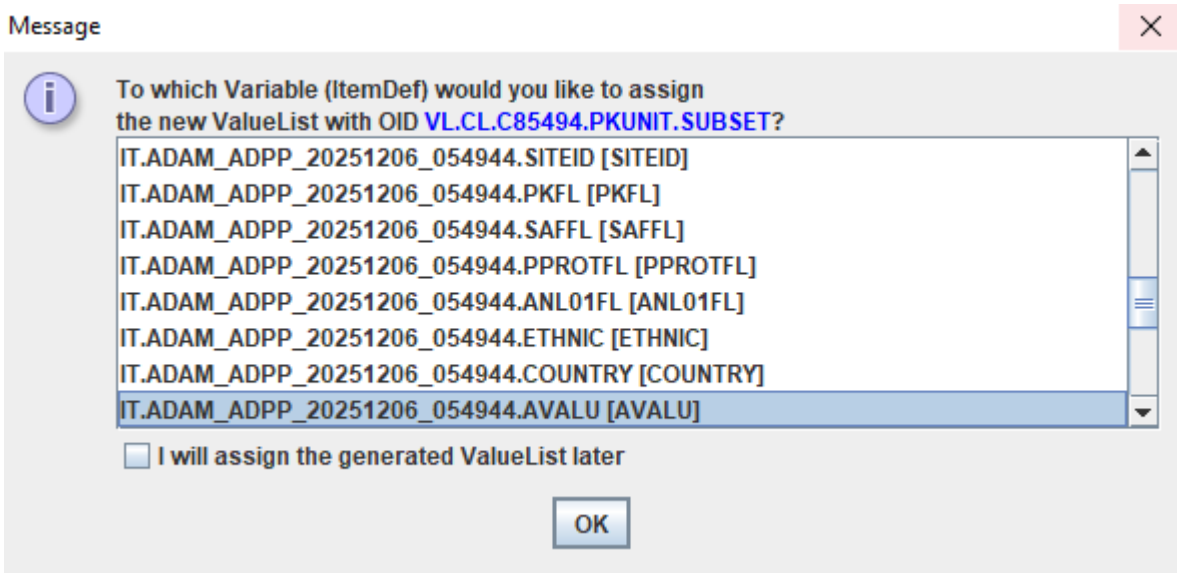
We can check by using the "Show 'Where' clause:



We than also do similar for the other 4 units, based on the above table. This finally leads to:



When done, clicking "OK" first gives us a summary, and then asks us to what ADaM variable we want to assign the ValueList, for which we select "AVALU":



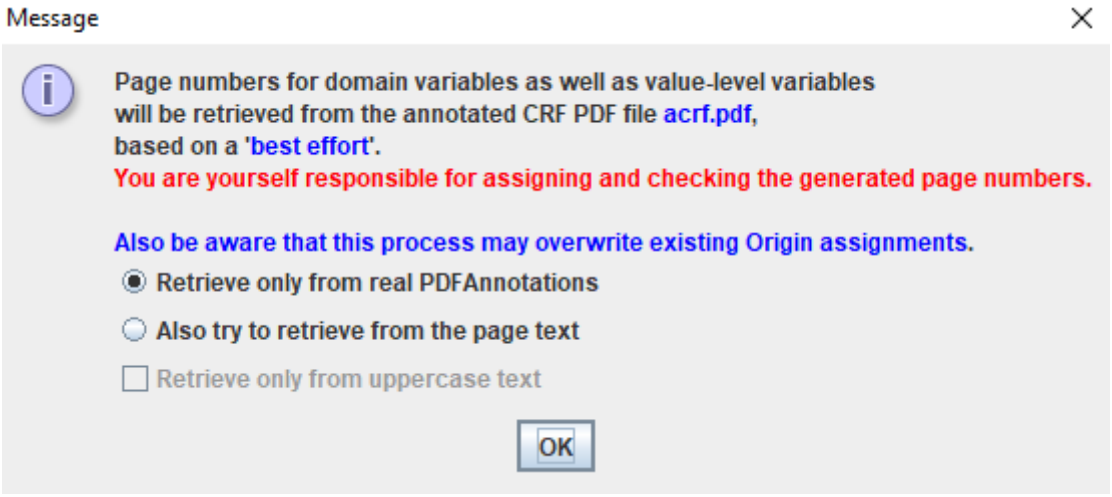
When we then use "HTML View", we find:

AVAL - [Edit] [Create ValueList] [Add ValueList]			float		6	
AVALU - [Edit] [Remove/Replace ValueList] [Edit ValueList]			text		7	
ValueList variable for AVALU - [Edit]	[Edit] PARAMCD IN [HALF, TMAX]	h	text		1	
ValueList variable for AVALU - [Edit]	[Edit] PARAMCD EQ VSS	L	text		1	
ValueList variable for AVALU - [Edit]	[Edit] PARAMCD EQ CMAX	ng/mL	text		5	
ValueList variable for AVALU - [Edit]	[Edit] PARAMCD IN [AUCOT, AUCINF]	ng*h/mL	text		7	
ValueList variable for AVALU - [Edit]	[Edit] PARAMCD EQ CL	mL/min	text		6	
TRTP - [Edit]			text		11	

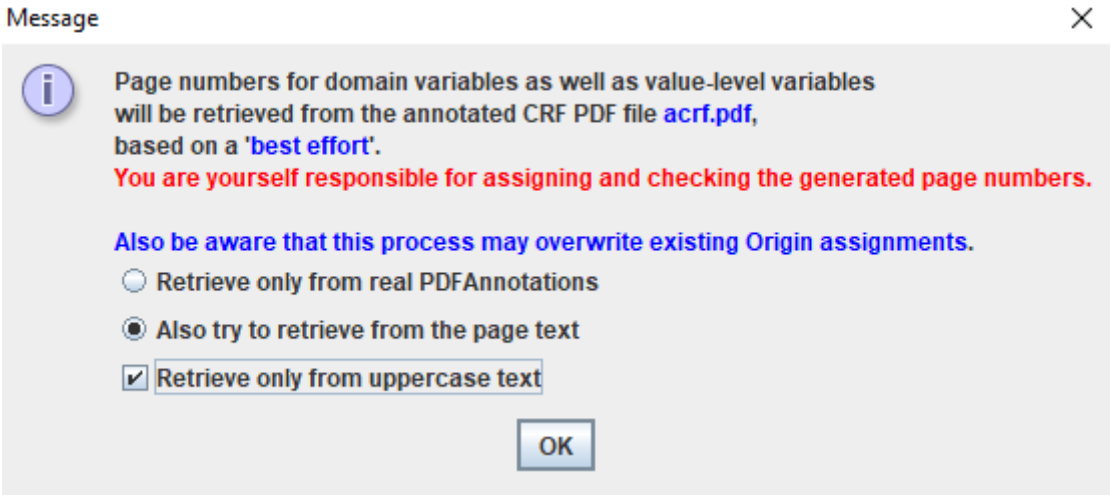
which is exactly what we intended to obtain ...

Extracting page numbers from an annotated Case Report Form (aCRF)

The capability of extracting annotations and their page numbers from annotated CRFs in PDF format[10], and analyzing them for incorporation into the define.xml have been extended.
Before starting this feature, ensure that the aCRF-PDF file location is defined as a document (tab "Document Links") and has been declared as the annotated CRF (tab "Annotated CRFs").
Then use the menu "Extra - Insert CRF Page Numbers from Annotated CRF". This leads to the dialog:



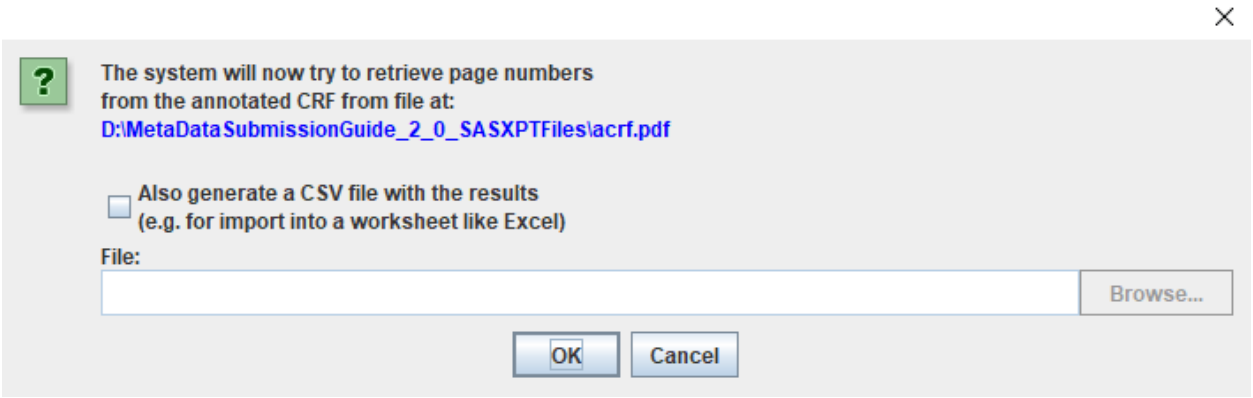
Reason for these choices is that there are many different ways to add "annotations" to CRFs. The first is to use real PDF annotations., i.e. "PDF Annotations". This method is becoming more seldom, as it does not lead to nicely colored boxes with the SDTM annotations. The second method is to add "text boxes" with a colored background, which has become more popular. If you still have another method for adding annotations, please feel free to send an example aCRF, and we will add an additional choice for it.
If you see such nicely colored text boxes, you will probably want to use the second method. If you are sure that all relevant "annotations" are in uppercase, also check the checkbox "Retrieve only from uppercase text". That will surely speed up the process, as otherwise also all other "non-annotation" texts will be retrieved and analyzed against what is in the define.xml. So, in such a case, use:



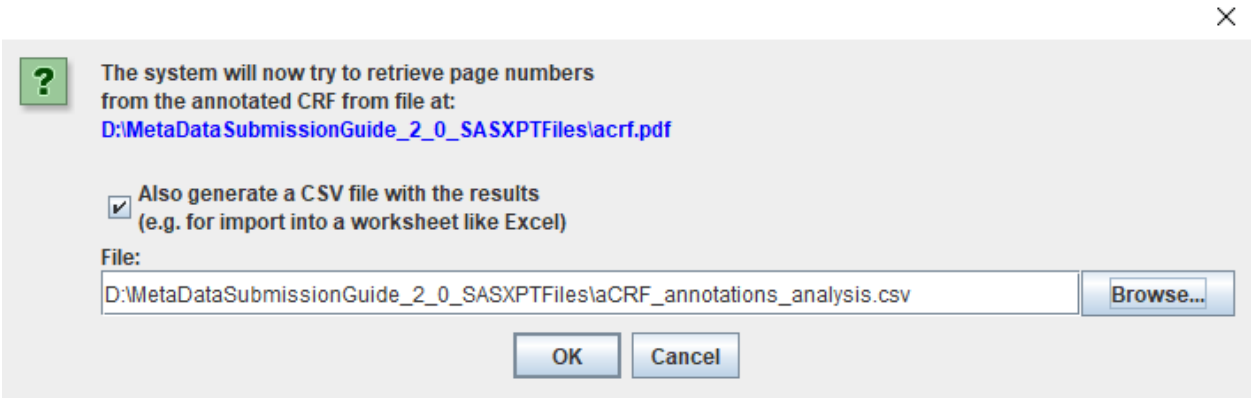
When clicking "OK" another dialog is displayed:

---

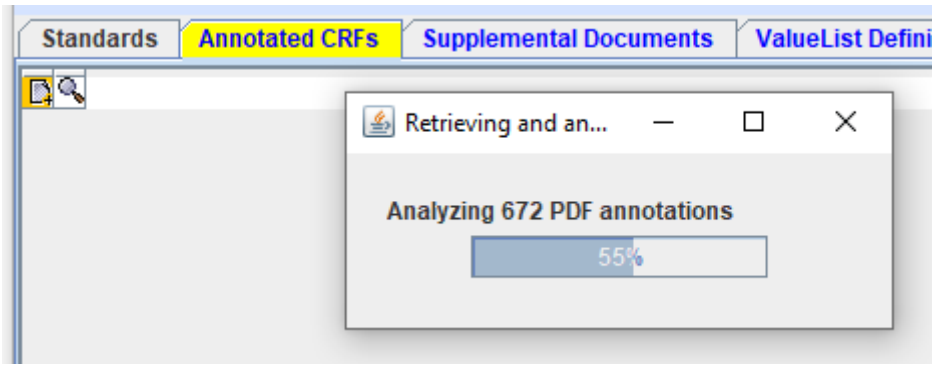
[10] Unfortunately, FDA and other regulatory authorities still require this ancient PDF technology for annotated CRFs to be used, this although already 15-20 years ago it was demonstrated that this can be done much better by using CDISC ODM with a stylesheet. This would also enable to do everything electronically in an automated way.



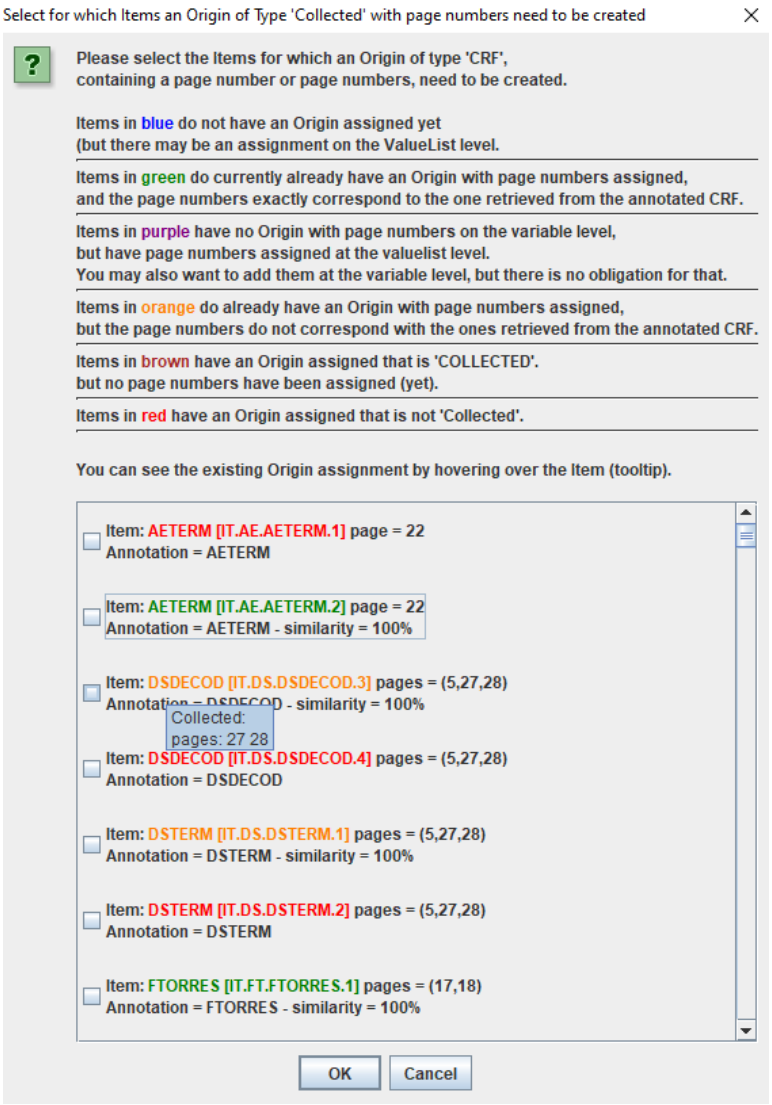
confirming on what PDF file the extraction and analysis will be performed (you can still cancel now), and also asking whether you want to also have a CSV file generated that than can be imported into a worksheet like Excel.
This can be very interesting for large aCRFs and when decisions will have to be made - the presence of an annotation in the PDF doesn't necessarily mean that the Origin needs to be set to "Collected" ("CRF" in the case of Define-XML v.2.0), and one wants to make such decisions as a team, or wants to make these decisions later. For example:



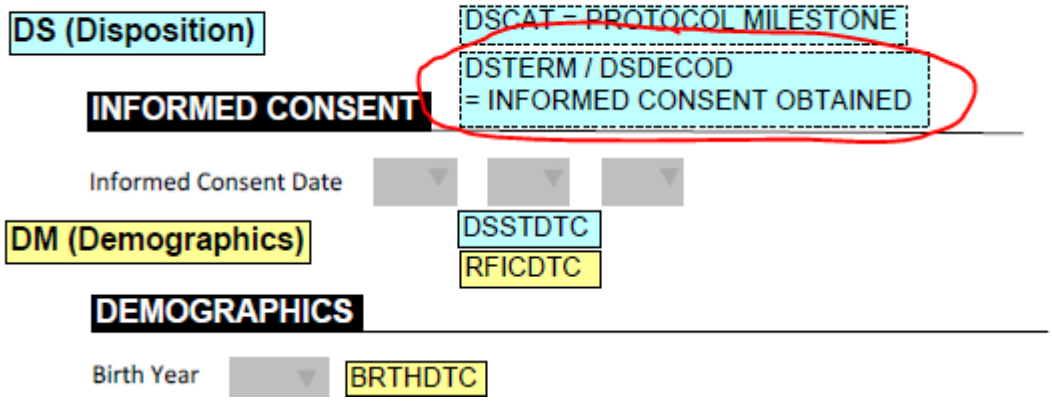
When clicking "OK" the system will start retrieving the annotations and analyze them by comparing with what is in the define.xml, especially on the content of the variable names (including ValueList-level variables). Depending on the size of the aCRF and what is in the define.xml, this can take considerable time, so a "progress bar" is displayed:



When finished, the following dialog with results is displayed:



For example, for the ValueList variable with OID=IT.DS.DSDECOD.3, the system found annotations on pages 5, 27 and 28, whereas the define.xml only has page numbers assigned for this variable. If we then check page 5, we find:

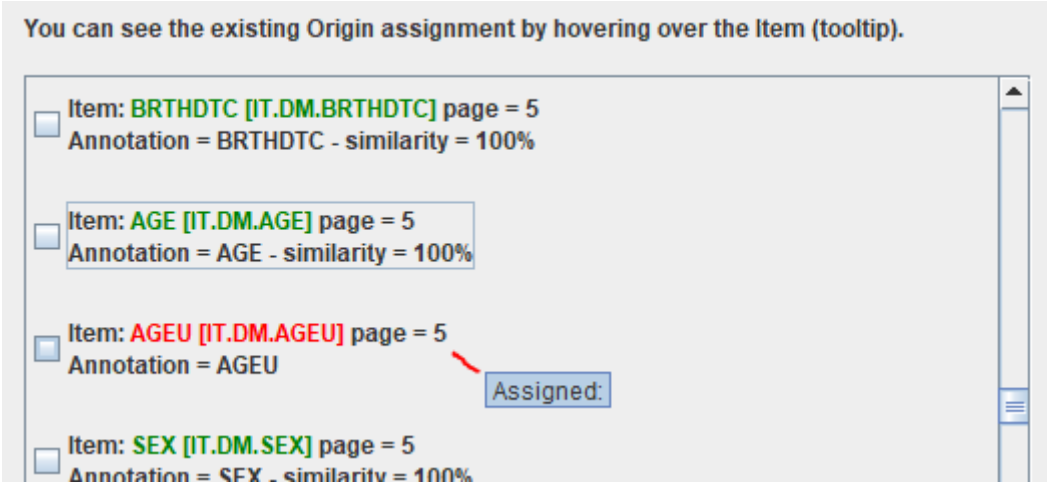


and in the HTML "View", we find:

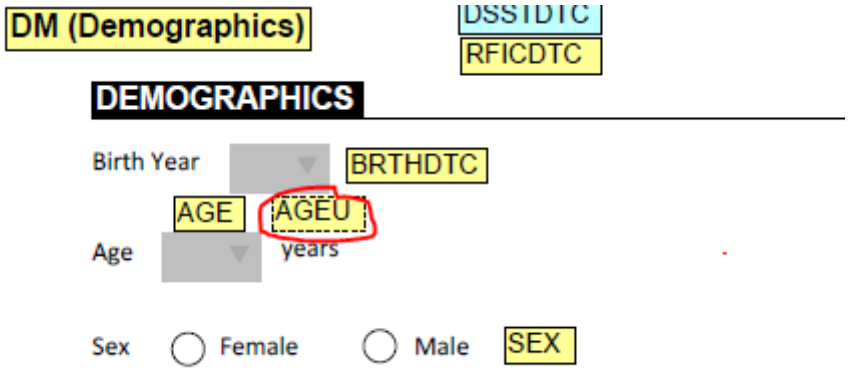
							Annotated CRF [5]
DSDECOD VLM		Standardized Disposition Term	text	Synonym Qualifier	29		
	DSSCAT ≠ ""	Standardized Disposition Term	text		29	Completion/Reason for Non-Completion [13 Terms]	Collected (Source: Investigator) Annotated CRF [27 28]
	DSSCAT = ""	Standardized Disposition Term	text		29	Protocol Milestone • "INFORMED CONSENT OBTAINED" = "Informed Consent"	Assigned (Source: Sponsor) Annotated CRF [5]

i.e. that is not applicable for the case that DSSCAT is empty[11]. This means that in this case, one will probably not want to change the existing PDF-page numbers assigned, and thus leave the checkbox unchecked.

Another example is AGEU (Age Unit):



where we indeed find an annotation on page 5:



which is preprinted. So the assignment of "Assigned" in the define.xml is correct.
In the future, we want to develop AI-based systems to even better interpret aCRFs to automatically set the "Origin" in the define.xml.

When looking into the CSV file that is generated, we find:

[11] We are further working on also interpreting the "WhereClause-s" to further refine the analys.

aCRF_annotations_analysis.csv - Editor

Datei Bearbeiten Format Ansicht Hilfe

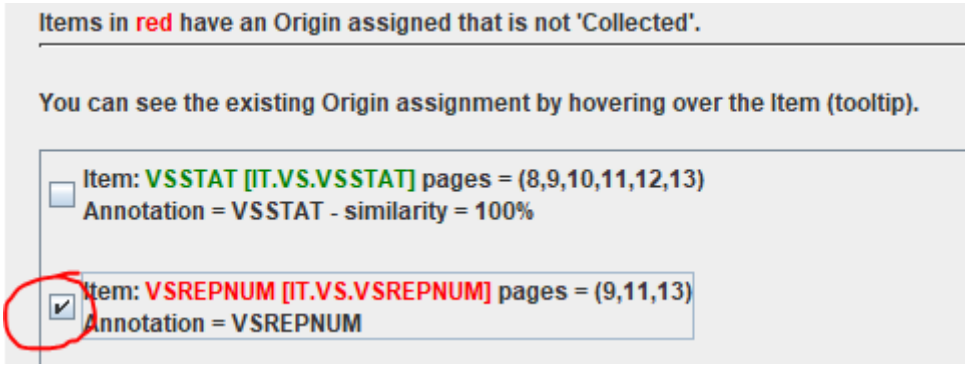
```
define.xml Item OID,define.xml Item Name,PDF Annotation,define.xml Origin and Source,Result,define.xml page numbers,PDF page numbers
IT.AE.AETERM.1,AETERM,AETERM,Assigned Sponsor,ORIGIN TYPE IS DIFFERENT,,22
IT.AE.AETERM.2,AETERM,AETERM,Collected Investigator,COMPLETE AGREEMENT,22,22
IT.DS.DSDECOD.3,DSDECOD,DSDECOD,Collected Investigator,PARTIAL AGREEMENT,27 28,5 27 28
IT.DS.DSDECOD.4,DSDECOD,DSDECOD,Assigned Sponsor,ORIGIN TYPE IS DIFFERENT,,5 27 28
IT.DS.DSTERM.1,DSTERM,DSTERM,Collected Investigator,PARTIAL AGREEMENT,27 28,5 27 28
IT.DS.DSTERM.2,DSTERM,DSTERM,Assigned Sponsor,ORIGIN TYPE IS DIFFERENT,,5 27 28
IT.FT.FTORRES.1,FTORRES,FTORRES,Collected Investigator,COMPLETE AGREEMENT,17 18,17 18
IT.FT.FTORRES.2,FTORRES,FTORRES,Collected Investigator,COMPLETE AGREEMENT,17 18,17 18
IT.FT.FTORRES.3,FTORRES,FTORRES,Collected Investigator,PARTIAL AGREEMENT,17,17 18
IT.FT.FTORRES.4,FTORRES,FTORRES,Collected Investigator,PARTIAL AGREEMENT,17,17 18
IT.QSPH.QSORRES.1,QSORRES,QSORRES,,PARTIAL AGREEMENT,,15 16
IT.QSPH.QSORRES.2,QSORRES,QSORRES,,PARTIAL AGREEMENT,,15 16
IT.QSPH.QSORRES.3,QSORRES,QSORRES,,PARTIAL AGREEMENT,,15 16
IT.QSPH.QSSTRESC.1,QSSTRESC,QSSTRESC,Derived Sponsor,ORIGIN TYPE IS DIFFERENT,,15 16
IT.QSPH.QSSTRESC.2,QSSTRESC,QSSTRESC,Derived Sponsor,ORIGIN TYPE IS DIFFERENT,,15 16
IT.QSPH.QSSTRESC.3,QSSTRESC,QSSTRESC,Predecessor,ORIGIN TYPE IS DIFFERENT,,15 16
IT.DM.RACE.1,RACE,RACE,Collected Investigator,COMPLETE AGREEMENT,5,5
IT.DM.RACE.2,RACE,RACE,Assigned Sponsor,ORIGIN TYPE IS DIFFERENT,,5
IT.RS.RSORRES.1,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.2,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.3,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.4,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.5,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.6,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.7,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.8,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.9,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.10,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.11,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.12,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.13,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.14,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.15,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
IT.RS.RSORRES.16,RSORRES,RSORRES,Collected Investigator,COMPLETE AGREEMENT,19,19
```

and when then imported into a worksheet like Excel:

aCRF_annotations_analy...

	A	B	C	D	E	F	G	H
1	define.xml Item OID	define.xml Item Name	PDF Annotation	define.xml Origin and Source	Result	define.xml page numbe	PDF page numbers	
2	IT.AE.AETERM.1	AETERM	AETERM	Assigned Sponsor	ORIGIN TYPE IS DIFFERENT		22	
3	IT.AE.AETERM.2	AETERM	AETERM	Collected Investigator	COMPLETE AGREEMENT	22	22	
4	IT.DS.DSDECOD.3	DSDECOD	DSDECOD	Collected Investigator	PARTIAL AGREEMENT	27 28	5 27 28	
5	IT.DS.DSDECOD.4	DSDECOD	DSDECOD	Assigned Sponsor	ORIGIN TYPE IS DIFFERENT		5 27 28	
6	IT.DS.DSTERM.1	DSTERM	DSTERM	Collected Investigator	PARTIAL AGREEMENT	27 28	5 27 28	
7	IT.DS.DSTERM.2	DSTERM	DSTERM	Assigned Sponsor	ORIGIN TYPE IS DIFFERENT		5 27 28	
8	IT.FT.FTORRES.1	FTORRES	FTORRES	Collected Investigator	COMPLETE AGREEMENT	17 18	17 18	
9	IT.FT.FTORRES.2	FTORRES	FTORRES	Collected Investigator	COMPLETE AGREEMENT	17 18	17 18	
10	IT.FT.FTORRES.3	FTORRES	FTORRES	Collected Investigator	PARTIAL AGREEMENT	17	17 18	
11	IT.FT.FTORRES.4	FTORRES	FTORRES	Collected Investigator	PARTIAL AGREEMENT	17	17 18	
12	IT.QSPH.QSORRES.1	QSORRES	QSORRES		PARTIAL AGREEMENT		15 16	
13	IT.QSPH.QSORRES.2	QSORRES	QSORRES		PARTIAL AGREEMENT		15 16	
14	IT.QSPH.QSORRES.3	QSORRES	QSORRES		PARTIAL AGREEMENT		15 16	
15	IT.QSPH.QSSTRESC.1	QSSTRESC	QSSTRESC	Derived Sponsor	ORIGIN TYPE IS DIFFERENT		15 16	
16	IT.QSPH.QSSTRESC.2	QSSTRESC	QSSTRESC	Derived Sponsor	ORIGIN TYPE IS DIFFERENT		15 16	
17	IT.QSPH.QSSTRESC.3	QSSTRESC	QSSTRESC	Predecessor	ORIGIN TYPE IS DIFFERENT		15 16	
18	IT.DM.RACE.1	RACE	RACE	Collected Investigator	COMPLETE AGREEMENT	5	5	
19	IT.DM.RACE.2	RACE	RACE	Assigned Sponsor	ORIGIN TYPE IS DIFFERENT		5	
20	IT.RS.RSORRES.1	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
21	IT.RS.RSORRES.2	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
22	IT.RS.RSORRES.3	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
23	IT.RS.RSORRES.4	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
24	IT.RS.RSORRES.5	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
25	IT.RS.RSORRES.6	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
26	IT.RS.RSORRES.7	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
27	IT.RS.RSORRES.8	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
28	IT.RS.RSORRES.9	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
29	IT.RS.RSORRES.10	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
30	IT.RS.RSORRES.11	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
31	IT.RS.RSORRES.12	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
32	IT.RS.RSORRES.13	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
33	IT.RS.RSORRES.14	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
34	IT.RS.RSORRES.15	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
35	IT.RS.RSORRES.16	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
36	IT.RS.RSORRES.17	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
37	IT.RS.RSORRES.18	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
38	IT.RS.RSORRES.19	RSORRES	RSORRES	Collected Investigator	COMPLETE AGREEMENT	19	19	
39	IT.SUPPEC.QVAL.1	ECREASOC	ECREASOC	Collected Investigator	COMPLETE AGREEMENT	14	14	
40	IT.SUPPOE.QVAL.1	OECLSIG	OECLSIG	Collected Investigator	COMPLETE AGREEMENT	21	21	
41	IT.VS.VSORRES.1	VSORRES	VSORRES	Collected Investigator	PARTIAL AGREEMENT	8 9 10 11 12 13 14	8 9 10 11 12 13	

which can then be used for making decisions whether (and for which) variables, the define.xml needs to be updated. The latter is done by checking the checkbox in the above dialog.
For example, for VSREPNUM, if the CRF had a field for the "repetition number" (in our case it is preprintet), one would check the checkbox for it:



and the result in the HTML "View" would become:

VSLOBXFL - [Edit] [Remove/Replace CodeList]		Last Observation Before Exposure Flag	text	Record Qualifier	1	for variables with only "Y" or null values - [Edit] ● Y	Derived / Sponsor [Edit]	result on or before the first dose date (RFXSTDTC). Null otherwise. [Edit] Show details	[Add]
VSREPNUM - [Edit]		Repetition Number	integer	Record Qualifier	8		Collected / Investigator [Edit] Annotated CRF [9 11 13]		[Add]
VISITNUM - [Edit]		Visit Number	float	Timing	8		Assigned / Sponsor [Edit]		[Add]
							Collected /		

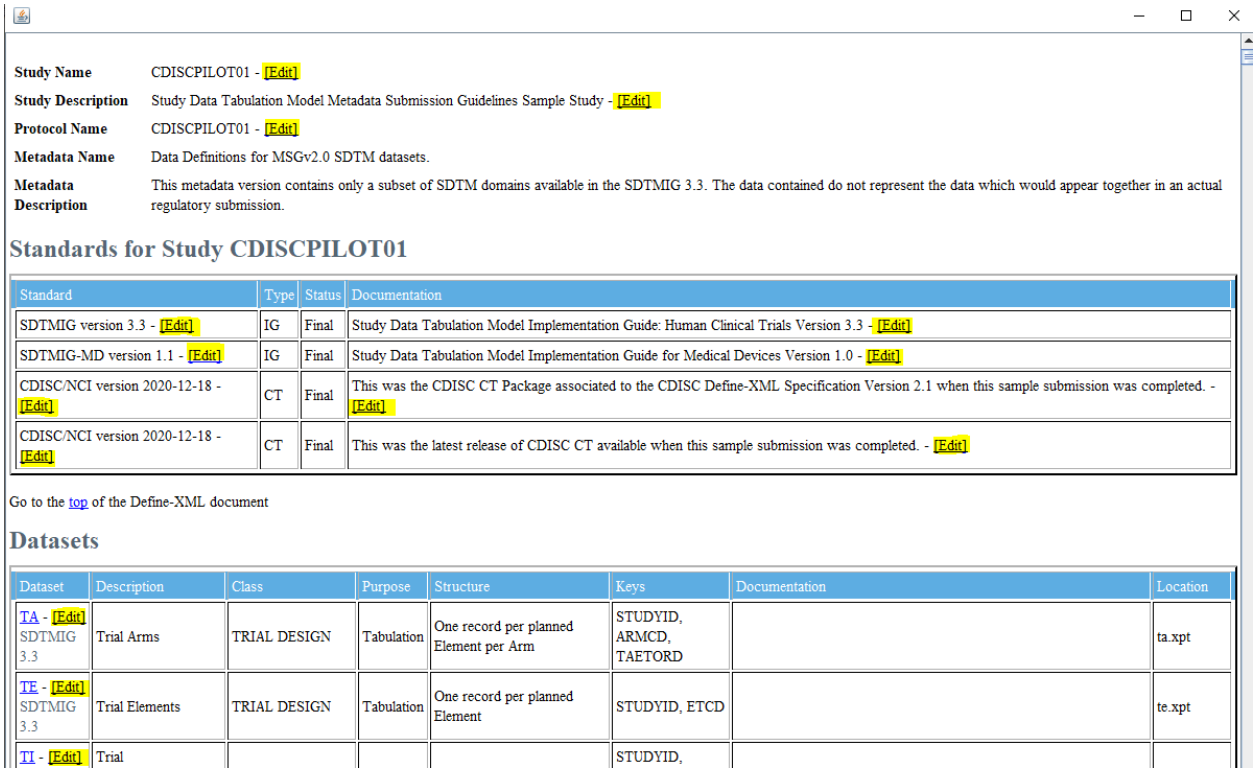
One can of course then still edit the page numbers, e.g. by clicking the "Edit" "hyperlink", which is explained in the next section.

IMPORTANT: generating the page numbers in the define.xml from extracting the annotations from the aCRF is a nice help to speed up the work[12], but it still your own responsibility to make the decisions to do the assigments correctly.

Starting Editing from within the "HTML View"

From the screenshots in the manual, one will have already noticed something special in the "views" in the "HTML View", I.e. the "Edit" links, like:

[12] User of other software told us that such assignments often costs them several days, and must be repeated when something changed in the aCRF.



Study Name CDISCPILOT01 - [Edit]

Study Description Study Data Tabulation Model Metadata Submission Guidelines Sample Study - [Edit]

Protocol Name CDISCPILOT01 - [Edit]

Metadata Name Data Definitions for MSGv2.0 SDTM datasets.

Metadata Description This metadata version contains only a subset of SDTM domains available in the SDTMIG 3.3. The data contained do not represent the data which would appear together in an actual regulatory submission.

Standards for Study CDISCPILOT01

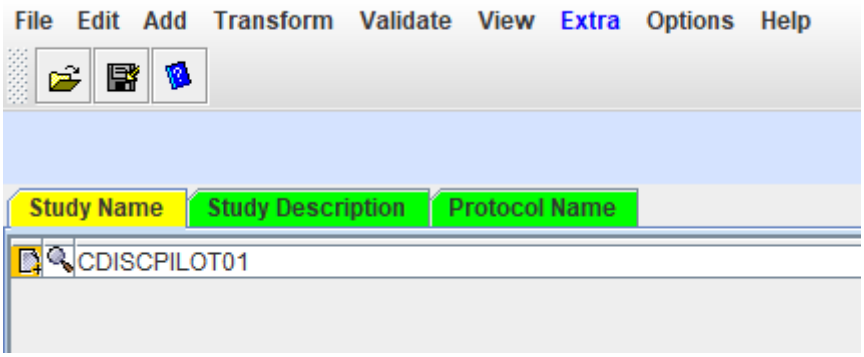
Standard	Type	Status	Documentation
SDTMIG version 3.3 - [Edit]	IG	Final	Study Data Tabulation Model Implementation Guide: Human Clinical Trials Version 3.3 - [Edit]
SDTMIG-MD version 1.1 - [Edit]	IG	Final	Study Data Tabulation Model Implementation Guide for Medical Devices Version 1.0 - [Edit]
CDISC/NCI version 2020-12-18 - [Edit]	CT	Final	This was the CDISC CT Package associated to the CDISC Define-XML Specification Version 2.1 when this sample submission was completed. - [Edit]
CDISC/NCI version 2020-12-18 - [Edit]	CT	Final	This was the latest release of CDISC CT available when this sample submission was completed. - [Edit]

Go to the top of the Define-XML document

Datasets

Dataset	Description	Class	Purpose	Structure	Keys	Documentation	Location
TA - [Edit] SDTMIG 3.3	Trial Arms	TRIAL DESIGN	Tabulation	One record per planned Element per Arm	STUDYID, ARMCD, TAETORD		ta.xpt
TE - [Edit] SDTMIG 3.3	Trial Elements	TRIAL DESIGN	Tabulation	One record per planned Element	STUDYID, ETCD		te.xpt
TI - [Edit]	Trial				STUDYID,		

When one e.g. the "Edit" on the line "Study Name" clicks, then the "HTML View" is pushed to the background and the field where one can change the value for "Study Name" becomes available:



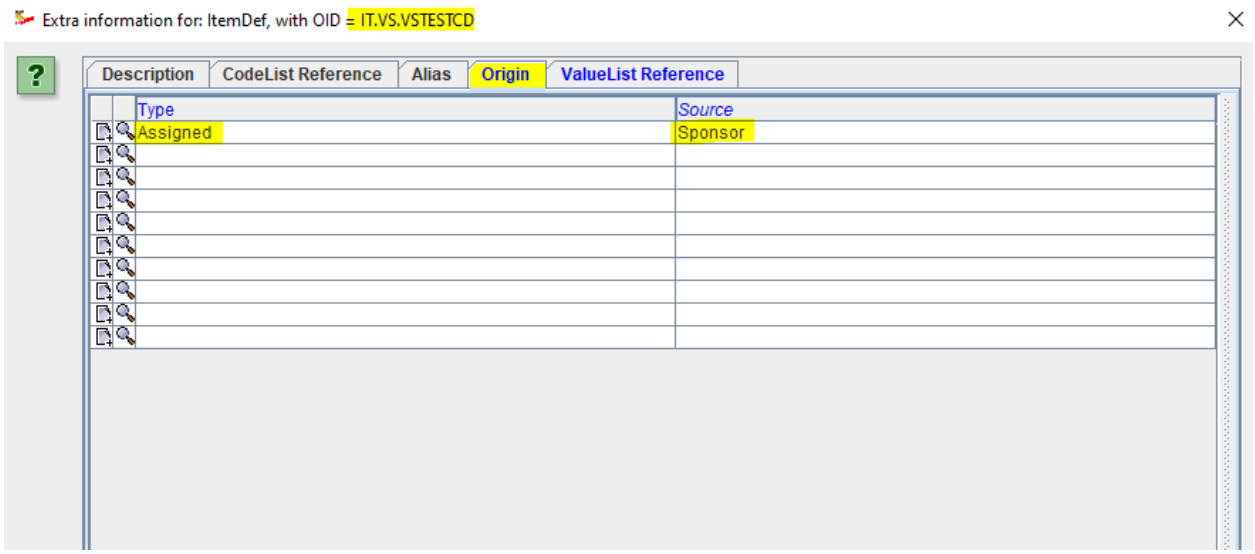
and one can then easily change the value in the field.

This is a very interesting feature, allowing to easily switch between "View" and "Edit" for very many of the pieces of information. For example, when one clicks on "Edit" for "Source / Origin" for "VSTESTCD":

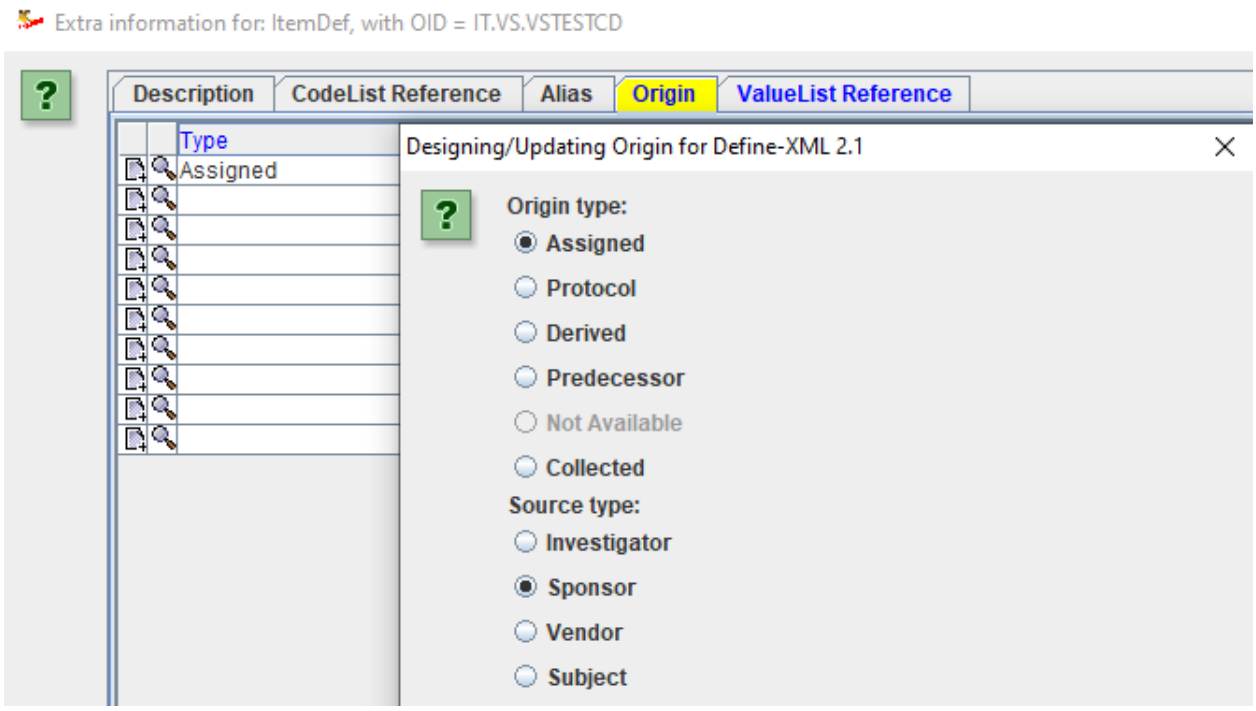
VS (Vital Signs) - [SDTMIG 3.3]

Variable	ValueList Where Condition	Label / Description	Type	Role	Length or Display Format	Controlled Terms or ISO Format	Origin/Source	Method	Comments
STUDYID - [Edit]		Study Identifier	text	Identifier	12		Protocol / Sponsor [Edit]		[Add]
DOMAIN - [Edit]		Domain Abbreviation	text	Identifier	2	SDTM Domain Abbreviation, subset used for Vital Signs - [Edit] • VS	Assigned / Sponsor [Edit]		[Add]
USUBJID - [Edit]		Unique Subject Identifier	text	Identifier	8		Assigned / Sponsor [Edit]		[Add]
VSSEQ - [Edit]		Sequence Number	integer	Identifier	3		Derived / Sponsor [Edit]	Unique sequence number within a subject, restarting at 1 for every subject, applied to sorted data. [Edit] Show details	[Add]
VSTESTCD - [Edit]		Vital Signs Test Short Name	text	Topic	6	Vital Signs Test Code - [Edit]	Assigned / Sponsor [Edit]		[Add]
VSTEST - [Edit]		Vital Signs Test Name	text	Synonym Qualifier	24	Vital Signs Test Name - [Edit]	Assigned / Sponsor [Edit]		[Add]
		Vital Signs				Position, subset to be used for VSPOS - [Edit]	Assigned / Sponsor		

then the system jumps back to the editor, presenting:



allowing to change the information "Assigned" and "Sponsor". Suppose e.g. that we want to change "Assigned" to "Derived" (which in future may be well possible with the raise of e.g. AI methods", we just click on "Assigned", and the wizard shows up:

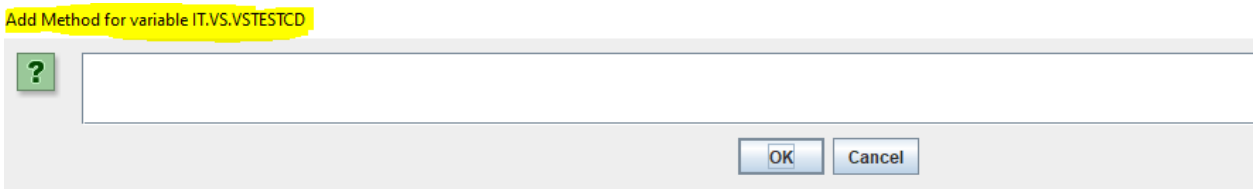


and one can change the value by clicking the "Derived" radiobutton.

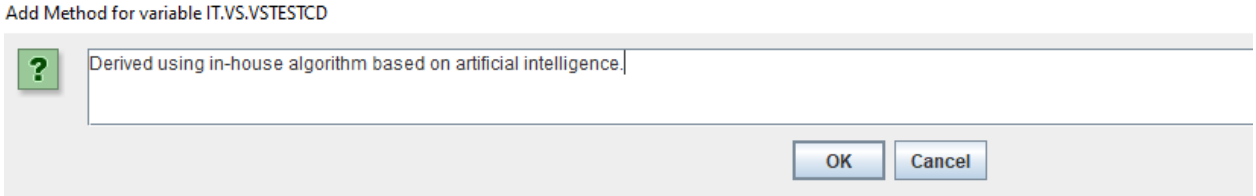
When then returning to the "HTML View"

								[Edit] Show details	
VSTESTCD - [Edit]		Vital Signs Test Short Name	text	Topic	6	Vital Signs Test Code - [Edit]	Derived / Sponsor [Edit]	[ADD]	[Add]
VSTEST - [Edit]		Vital Signs Test Name	text	Synonym Qualifier	24	Vital Signs Test Name - [Edit]	Assigned / Sponsor [Edit]		[Add]

we see that for "Method", a new link "ADD" has appeared, as when the "Assigned" is provided, the Define-XML rules state that then also the derivation method must be provided. Then clicking the "ADD" link, the editor opens an entry screen:



which could e.g. be filled with:



After clicking "OK" and navigating to the "Method Definitions" panel, one finds the method with an automatically assigned OID and Name:

Standards | Annotated CRFs | Supplemental Documents | ValueList Definitions | WhereClause Definitions | Dataset Definitions | Variable Definitions | Codelists | Method Definitions

OID	Name	Typ
MT.AEENTPT	Algorithm to derive AEENTPT	Cor
MT.CMENTPT	Algorithm to derive CMENTPT	Cor
MT.DAYCALC	Algorithm to derive DAYCALC	Cor
MT.DISEQ	Algorithm to derive DISEQ	Cor
MT.DTHFL	Algorithm to derive DTHFL	Cor
MT.EPOCH	Algorithm to derive EPOCH	Cor
MT.EXDOSE	Algorithm to derive EXDOSE	Cor
MT.FTSTRESN	Algorithm to derive FTSTRESN	Cor
MT.IEORRES	Algorithm to derive IEORRES	Cor
MT.LBSTRESC	Algorithm to derive LBSTRESC	Cor
MT.LOBXFL	Algorithm to derive LOBXFL	Cor
MT.QSSTRESC_PH	Algorithm to derive QSSTRESC_PH	Cor
MT.QSSTRESC_PH_10_11	Algorithm to derive QSSTRESC_PH_10_11	Cor
MT.QSSTRESC_SL	Algorithm to derive QSSTRESC_SL	Cor
MT.QSSTRESN	Algorithm to derive QSSTRESN	Cor
MT.RFENDTC	Algorithm to derive RFENDTC	Cor
MT.RFPENDTC	Algorithm to derive RFPENDTC	Cor
MT.RFSTDTC	Algorithm to derive RFSTDTC	Cor
MT.RFXENDTC	Algorithm to derive RFXENDTC	Cor
MT.RFXSTDTC	Algorithm to derive RFXSTDTC	Cor
MT.RSSTRESC	Algorithm to derive RSSTRESC	Cor
MT.SEENDTC	Algorithm to derive SEENDTC	Cor
MT.SEQ	Algorithm to derive SEQ	Cor
MT.SESTDTC	Algorithm to derive SESTDTC	Cor
MT.STRESN	Algorithm to derive STRESN	Cor
MT.SVENDTC	Algorithm to derive SVENDTC	Cor
MT.SVSTDTC	Algorithm to derive SVSTDTC	Cor
MT.TSSEQ	Algorithm to derive TSSEQ	Cor
MT.VSSTRESC	Algorithm to derive VSSTRESC	Cor
MT.IT.VS.VSTESTCD	Method definition for variable with OID IT.VS.VSTESTCD	Cor

and when then returning to the "HTML View" by clicking the button, one finds:

								[Edit] Show details	
VSTESTCD - [Edit]		Vital Signs Test Short Name	text	Topic	6	Vital Signs Test Code - [Edit]	Derived / Sponsor [Edit]	Derived using in-house algorithm based on artificial intelligence. [Edit] Show details	[Add]
VSTEST - [Edit]		Vital Signs Test Name	text	Synonym Qualifier	24	Vital Signs Test Name - [Edit]	Assigned / Sponsor [Edit]		[Add]

Of course, this is just a hypothetical example …
If one then clicks "Show Details", we get:

Algorithm to derive VSSTRESC - [Edit]	Computation	**Language:**en Data collected in conventional units (i.e. F, lbs, inches) is converted using
Method definition for variable with OID IT.VS.VSTESTCD - [Edit]	Computation	**Language:**en Derived using in-house algorithm based on artificial intelligence.

I.e. the HTML View jumps to the "Methods" section, and shows further details, if any.
Remark that Define-XML is essentially multi-language, so one could also add an additional Japanese or Chinese text.

Adding definitions from CSV files

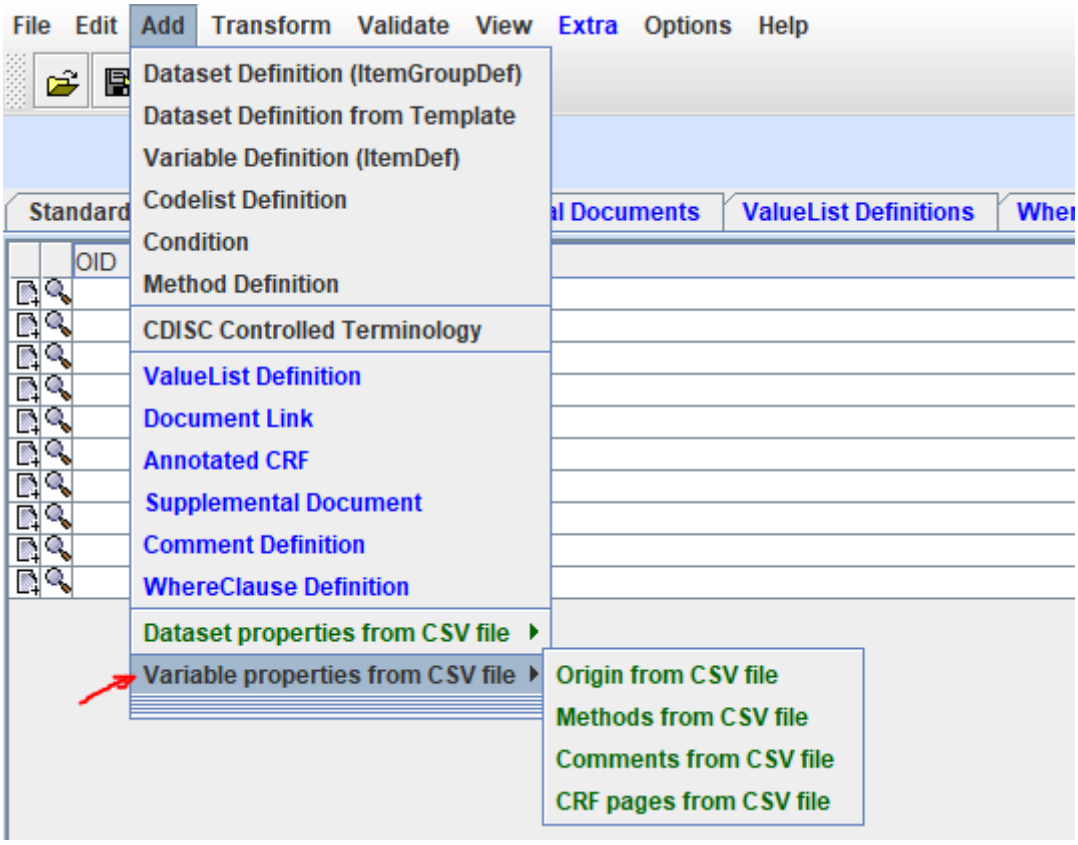
Unfortunately, there are still many companies who use worksheets like Excel to set up definitions for generating a define.xml. In combination with (usually low-budget or free) "black box" software for which no official manual is available, this will very often lead to disaster, and to a "trial and error" methodology with many "try" cycles.
Some of our customers asked us to be able to use the information from these worksheets anyway, as using spreadsheets is the usual way of working within their company.

The Define-XML Designer enables to use CSV files exported from worksheets to be used as input into the define.xml, either at the variable level or at the dataset level. We will start with the variable level.

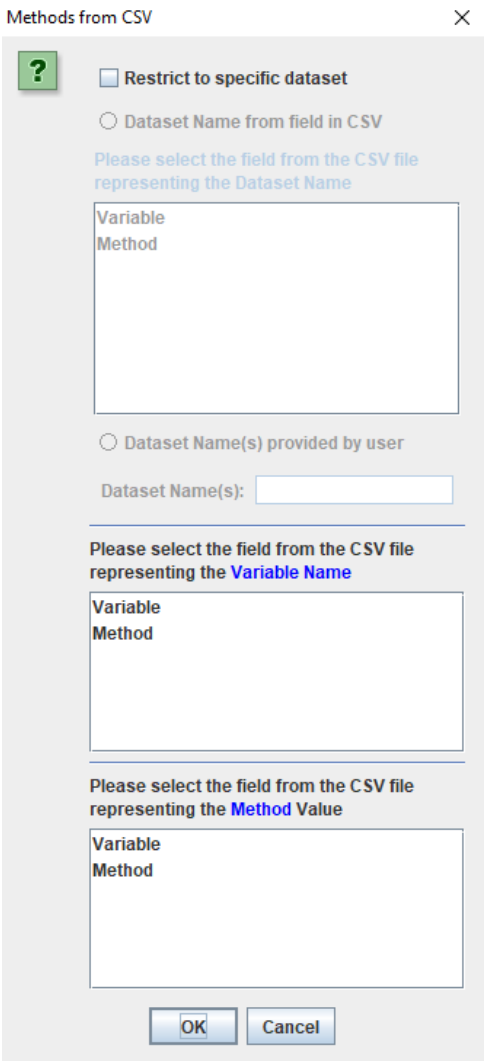
Adding variable properties from CSV files

Condition for using these features is that some dataset definitions and their variables are already available, e.g. from a selected template or generated from SAS-XPT files.

When using the menu "Add - Variable Properties from CSV File":

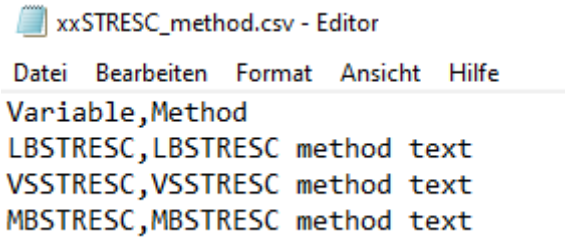


one has the choice of adding values for "Origin", "Methods", "Comments" and "CRF pages". We will demonstrate here for the case of the user wanting to add some Method definitions (define.xml "MethodDef") from a CSV file. So, when having selected "Comments from CSV file", this will first lead to a "file chooser", allowing the user to select a CSV file, and then to another dialog:

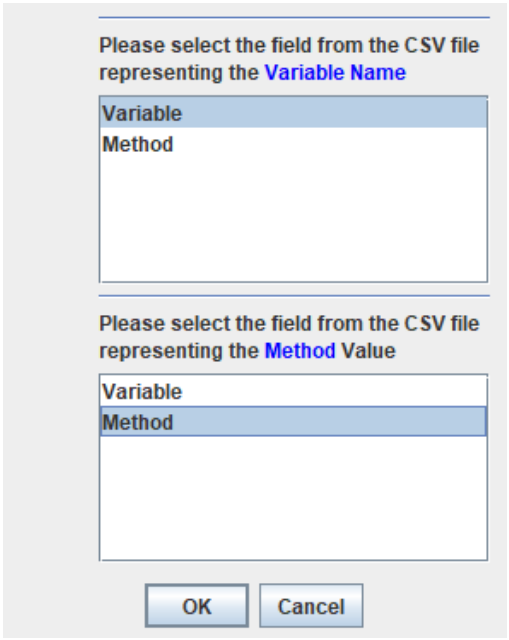


In most cases, one will only need the lower part of the dialog, the use of the upper part will be explained later.

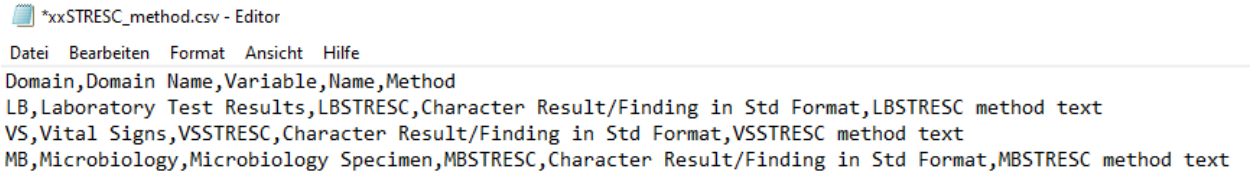
Important in the CSV file is that the first line is a "header line" containing the "field names", for example:



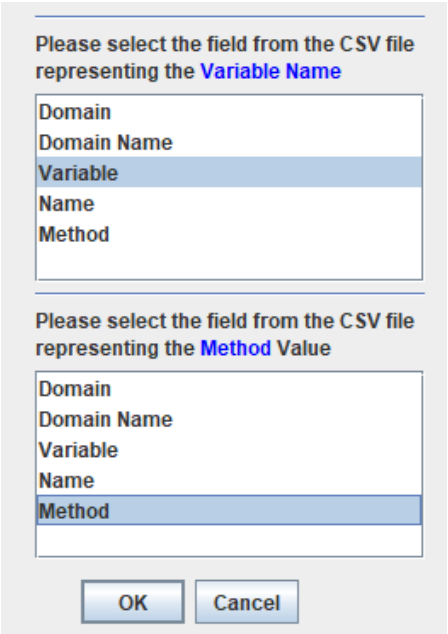
defining that for LBSTRESC, VSSTRESC and MBSTRESC (the list will usually be longer) we will add specific method descriptions to "MethodDef" in the define.xml.



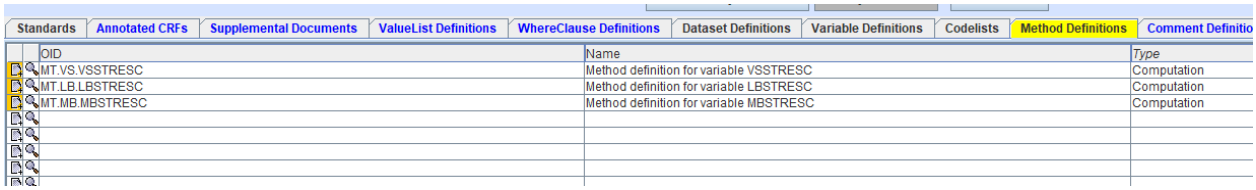
It can of course be that (as the CSV is exported from a worksheet), that there are more fields, for example:



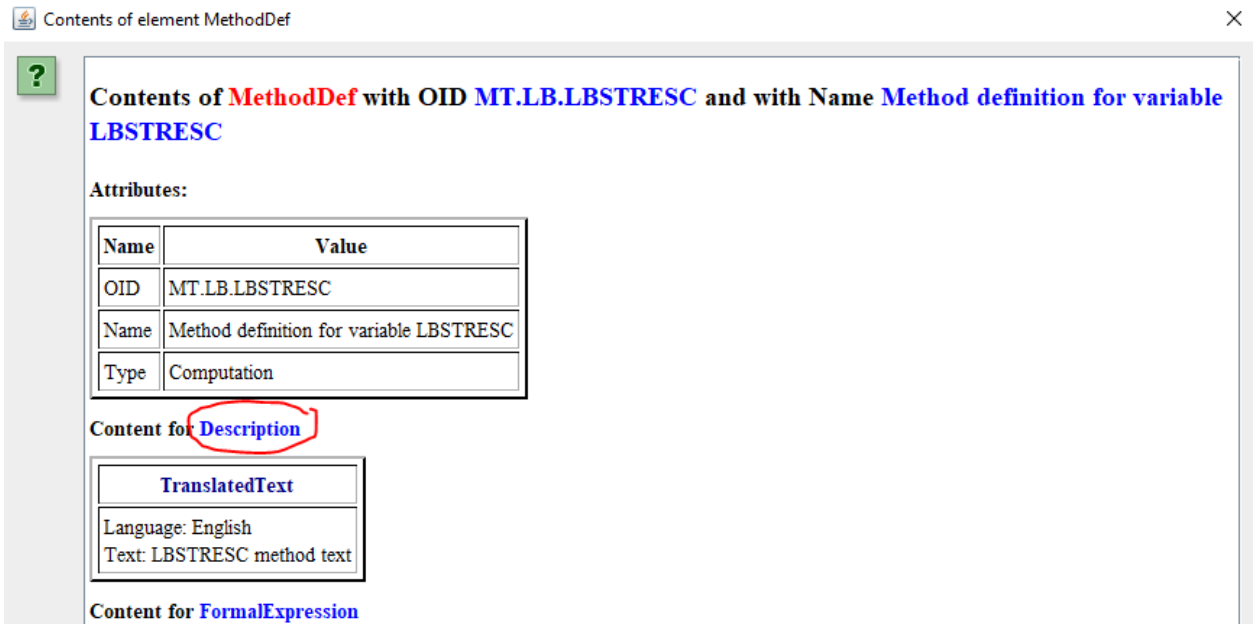
The choices then are:



After clicking "OK", new method definitions will be added to the "MethodDef" elements, and in the "Method Definitions" tab, we find 3 new entries:



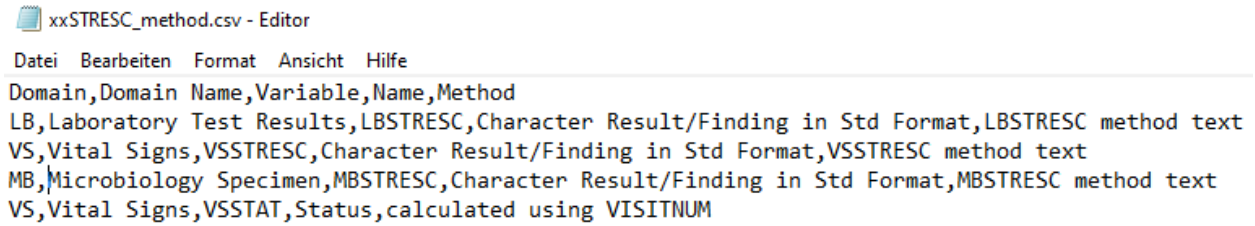
Remark that the description text of the method is not in the "Name" attribute, but in the underlying "Description" element[13]. We can see this by clicking on the "magnifying glass" icon on the left:



Also, each of the newly defined methods is then immediately assigned to the variables VSSTRESC, LBSTRESC and MBSTRESC respectively. If we use "View - define.xml in the browser, we e.g. find:

MBORRES	Result or Finding in Original Units	text	Result Qualifier	80		
MBORRESU	Original Units	text	Variable Qualifier	25	Unit [929 Terms]	
MBSTRESC	Result or Finding in Standard Format	text	Result Qualifier	80		MBSTRESC method text
MBSTRESN	Numeric Result/Finding in Standard Units	float	Result Qualifier	8		
MBSTRESU	Standard Units	text	Variable Qualifier	25	Unit [929 Terms]	
MBRESCAT	Result Category	text	Variable Qualifier	80		

Of course it is not needed to have all the lines in the CSV file to point to the same "type" of variable (in our case xxSTRESC variables). If the CSV file also contains a definition for e.g. VSSTAT, like:



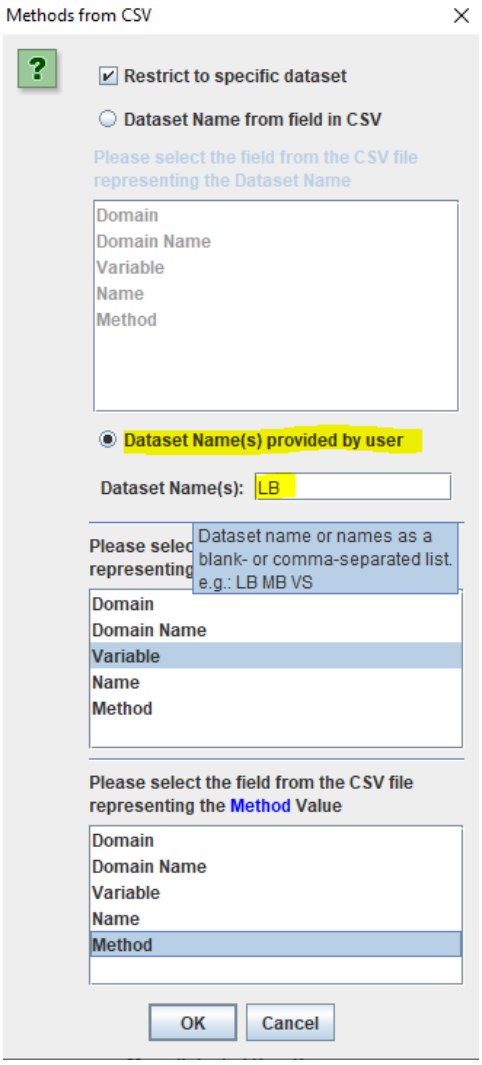
then also a method definition for VSSTAT will be added and assigned to VSSTAT.

Important to notice here is that when there was already a method definition assigned to e.g. MBSTRESC, the newly generated one (with a separate OID) will be assigned to MBSTRESC, but the old MethodDef will not be deleted - it just is then "orphaned" and may not be referenced from any variable. It can later be removed using the "Cleaning" procedure.

It can also be that one has such a CSV file, but only want to use part of it, e.g. restrict the assignment to specific domains or datasets. In such a case, the upper part of the dialog comes into play.
Suppose that we want to restrict the assignment of the method to LBSTRESC, the following is then used:

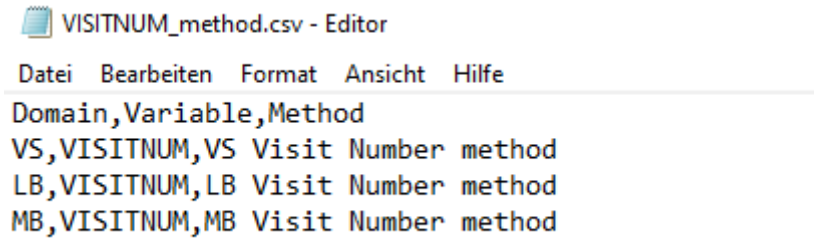
---

[13] The reason is that the description text may be language-dependent, i.e. one may have different description texts for different languages.

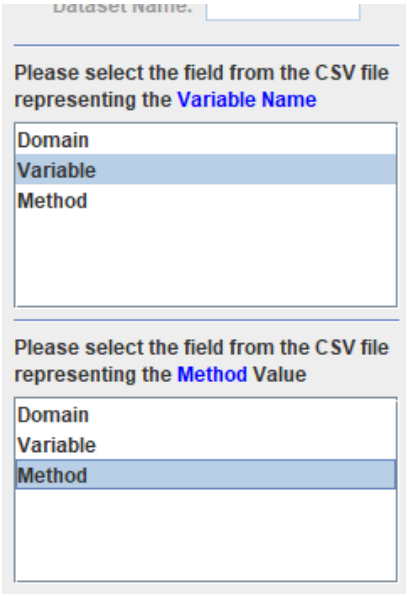


Where one can provide a single dataset name, or a blank-separated (or comma-separated) list of dataset names.

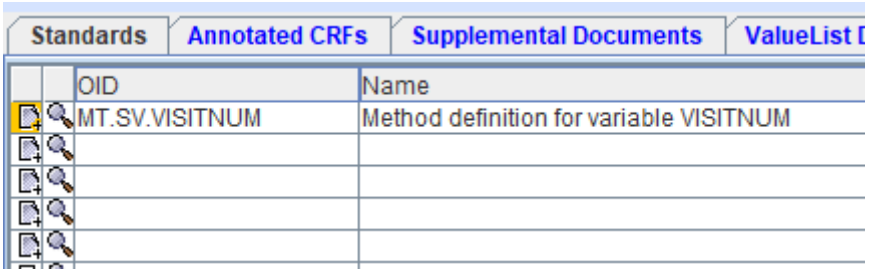
Some variables such as VISITNUM are usually used in a good number of datasets. If we want to add the text for the method from the CSV file for specific datasets (so, not for all of them), the latter may e.g. look like:



For the system, this may be confusing, as it provides 3 different methods for a single "generic" variable (VISITNUM), so if we just use the lower part of the dialog:



the system does not understand which of the 3 to use, reading them one after each other from the CSV file, and, for safety reasons, just generates one, and solely assigns it to the first dataset definition it finds:

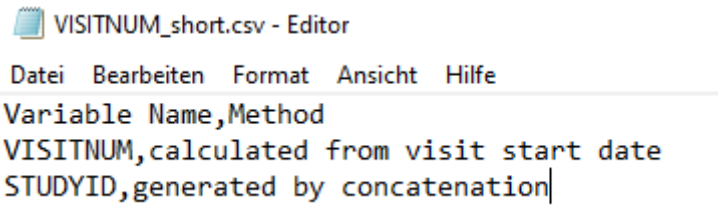


and in the HTML View:

SV (Subject Visits) - SPECIAL PURPOSE [SDTMIG 3.4]

Variable	Label / Description	Type	Role	Length or Display Format	Controlled Terms or ISO Format	Origin / Source / Meth
STUDYID	Study Identifier	text	Identifier	80		
DOMAIN	Domain Abbreviation	text	Identifier	8		
USUBJID	Unique Subject Identifier	text	Identifier	80		
VISITNUM	Visit Number	float	Topic	8		MB Visit Number method
VISIT	Visit Name	text	Synonym Qualifier	80		
SVPRESP	Pre-specified	text	Variable Qualifier	80	No Yes Response • "N" • "NA" • "U" • "Y"	
SVOCCUR	Occurrence	text	Record Qualifier	80	No Yes Response • "N"	

If we however only use a single method for VISITNUM, as in:

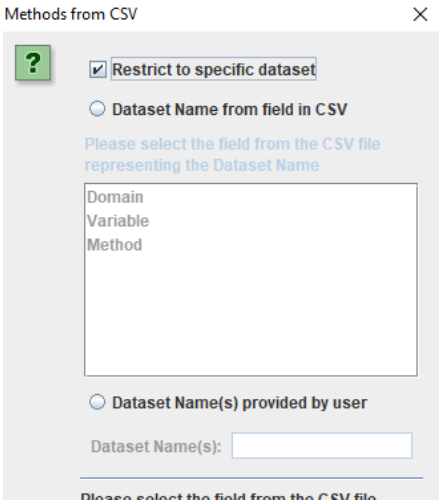


and only use the lower part of the dialog, the system still doubts whether the method for VISITNUM should be assigned

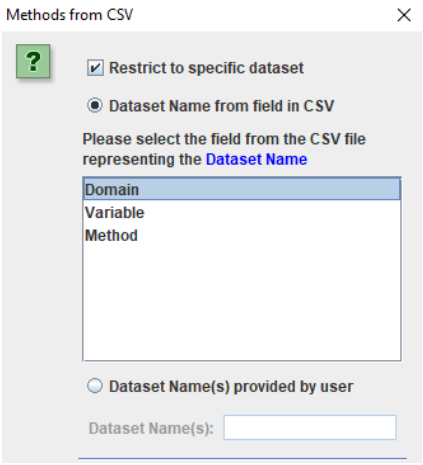
to all instances, i.e. to each dataset-VISITNUM, and for security reasons, only assigns it to the first it encounter. The user can then still assign it to all others using the editor. The reason is that some sponsors use a single method definition for variables such as VISITNUM, and others generate a different method definition for VISITNUM per dataset definition. We have however also seen a lot that "Assigned" is used, which one could regard as "the lazy method". For "STUDYID", the case is however clear. Essentially, the value, and thus also the properties must always be the same, for each row in each dataset described in the define.xml.

If we want to assign a single method for VISITNUM to all the datasets, there are different ways, that all involve the upper part of the dialog.

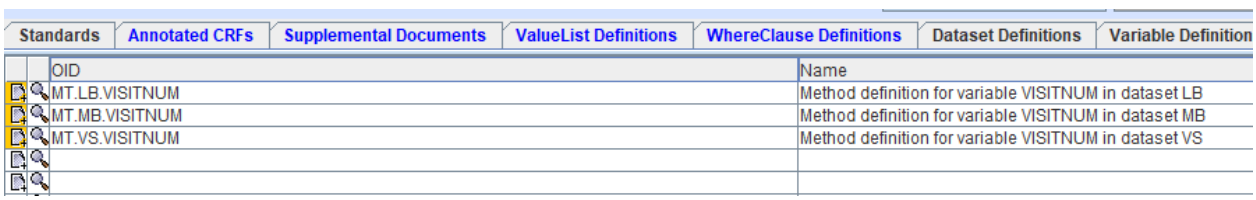
In such a case, one must also use the upper part of the dialog, i.e.:



For the dataset choice, one must then choose between taking the dataset name from a CSV field (this will be the usual case) or set the dataset to be applied to using the radiobutton "Dataset Name provided by the user". The usual case is:



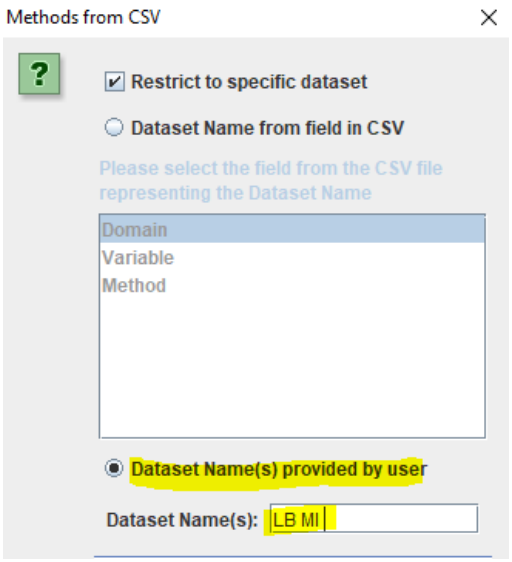
Then the specific descriptions for the method for VISITNUM will be applied to VS, LB and MB, but not to other ones, e.g. leading to:



and in the "View", e.g. for MB:

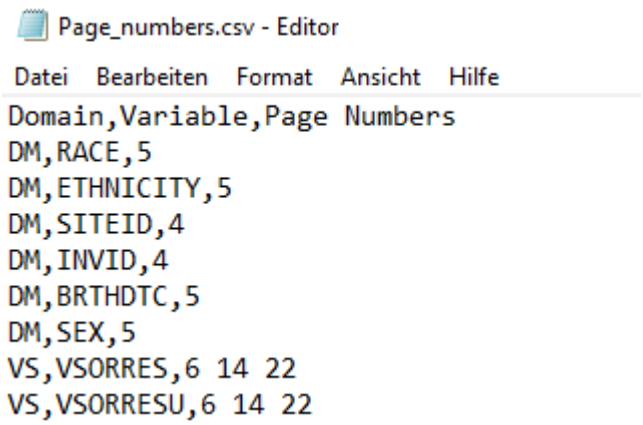
MBFAST	Fasting Status	text	Record Qualifier	2	No Yes Response • "N" • "NA" • "U" • "Y"	
MBDRVFL	Derived Flag	text	Record Qualifier	2	No Yes Response • "N" • "NA" • "U" • "Y"	
VISITNUM	Visit Number	float	Timing	8		MB Visit Number method
VISIT	Visit Name	text	Timing	80		
VISITDY	Planned Study Day of Visit	integer	Timing	8		

Alternatively, the user can assign one or more datasets/domains, like e.g.:

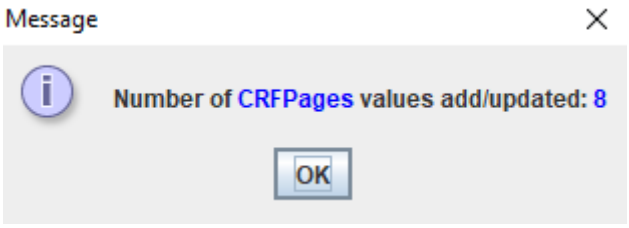


in which case the VISITNUM method will only be assigned to the datasets MB and MI.

One can also, in a very similar way import the page numbers on the aCRF for the variables, e.g. as:



This then only requires the lower part of the dialog to be used.
Remark that the page numbers must be delivered as a **blank-separated** list. Do not use commas!
When finished, the system then shows a message:



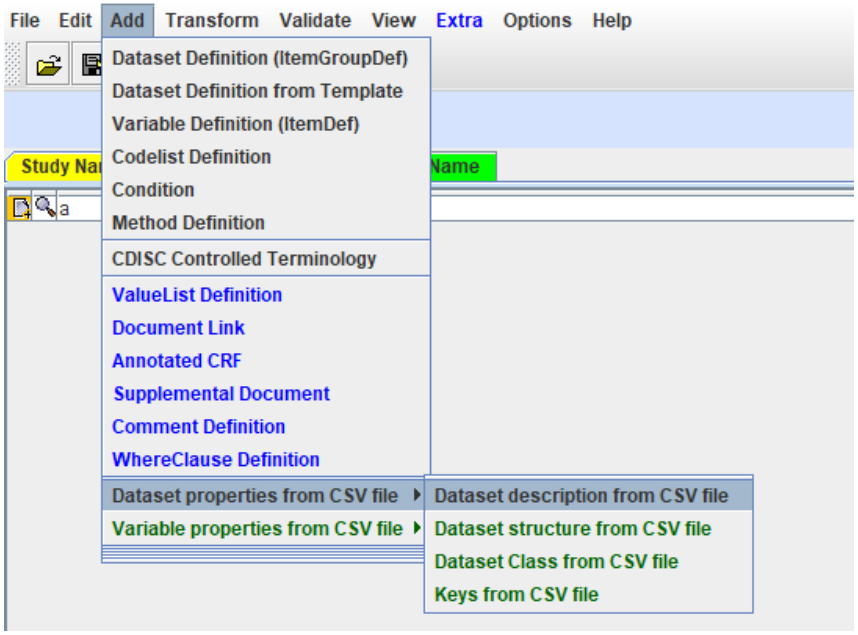
and we e.g. find in the HTML View:

VSCAT	Category for Vital Signs	text	Grouping Qualifier	80		
VSSCAT	Subcategory for Vital Signs	text	Grouping Qualifier	80		
VSPOS	Vital Signs Position of Subject	text	Record Qualifier	23	Position [17 Terms]	
VSORRES	Result or Finding in Original Units	text	Result Qualifier	80		Collected [unresolved: LF.blankCRF] [6 14 22]
VSORRESU	Original Units	text	Variable Qualifier	11	Units for Vital Signs Results [29 Terms]	Collected [unresolved: LF.blankCRF] [6 14 22]

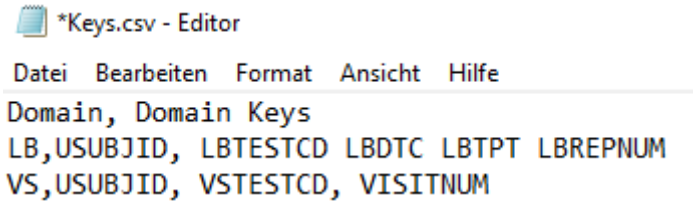
where we see that for VSORRES and VSORRESU the page numbers have been imported from the CSV file, and "Origin" be set to "Collected". However "Source" has not been assigned, as the system cannot know who collected the data. This can e.g. also have been the subject itself.

Adding dataset properties from CSV files

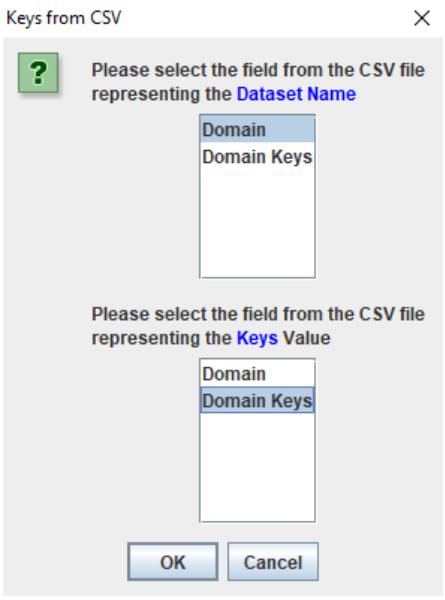
Similar can be done for some properties of the datasets:



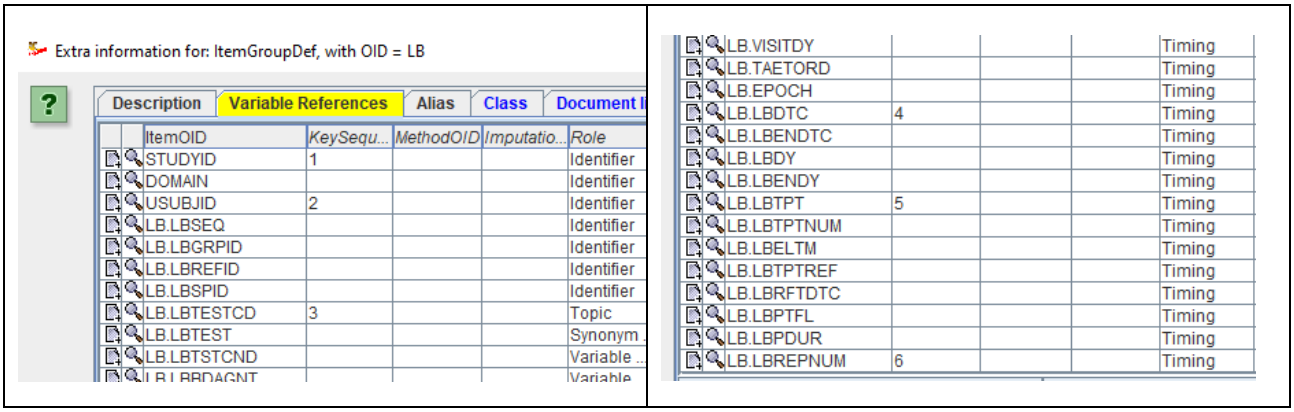
As well dataset descriptions (like "Adverse Events" - but also for custom domains), dataset structures (e.g. "One record per time point per visit per test per subject"), the "dataset class" (especially for custom domains - e.g. "Findings") or the dataset key variables) can be added from a CSV file.
We will elaborate this for the latter. For example, we have a file "Keys.csv" containing:



Remark that the list of the keys is "blank-separated".
When we then use the menu "Add - Dataset properties from CSV file - Keys from CSV file", after the CSV file selection, the following dialog is shown, asking us which "fields" in the CSV must be taken for selecting the information:



It can of course be that the file contains more fields … or that there is one single file for all four types of information: dataset description, structure, class and keys.
One may also notice that this dialog is very similar to that one for adding properties for variables, but with the upper part of the dialog missing, as there is nothing to do additional filtering on: the "dataset" is already the highest level.
After clicking "OK", the information is added, and we e.g. find for LB:



and in the HTML View:

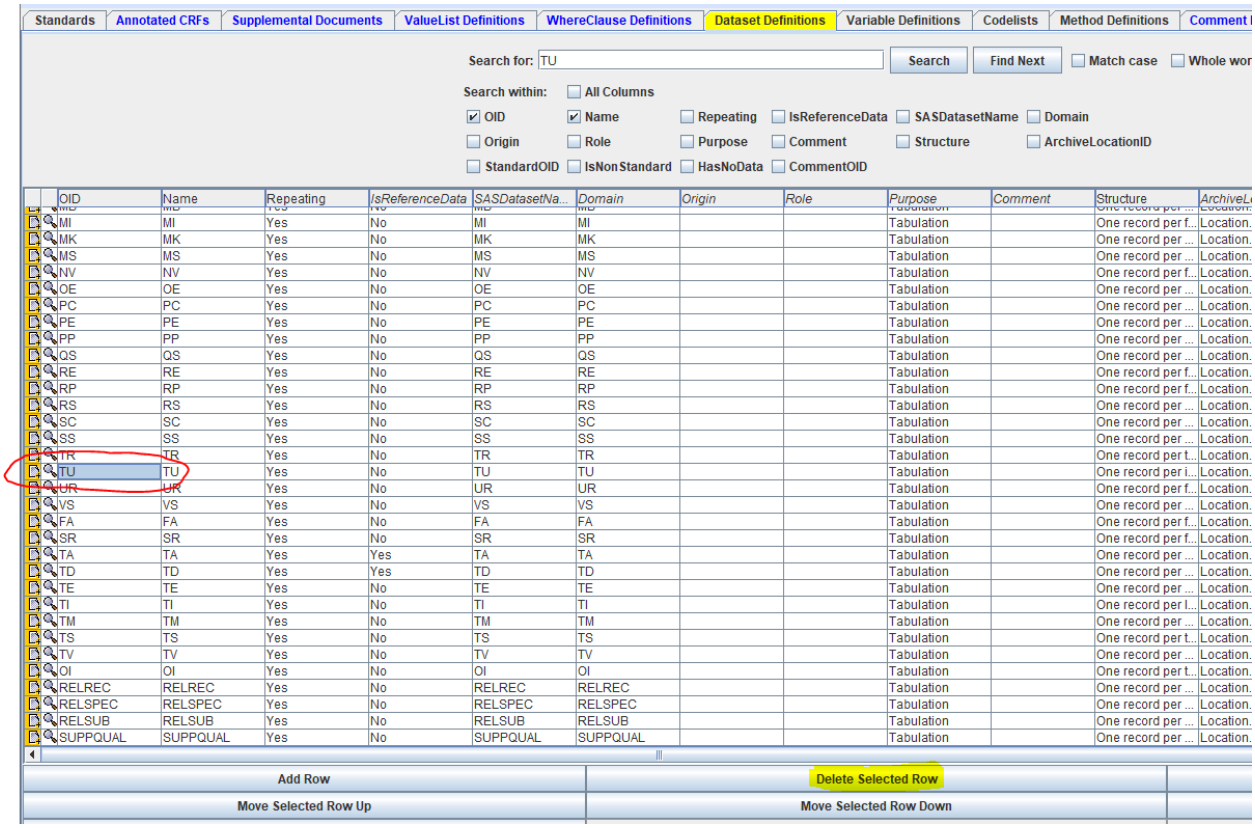
			per biospecimen per subject		
IE [SDTMIG 3.4]	Inclusion/Exclusion Criteria Not Met	FINDINGSFINDINGS	One record per inclusion/exclusion criterion not met per subject	Tabulation	
IS [SDTMIG 3.4]	Immunogenicity Specimen Assessments	FINDINGSFINDINGS	One record per test per visit per subject	Tabulation	
LB [SDTMIG 3.4]	Laboratory Test Results	FINDINGSFINDINGS	One record per lab test per time point per visit per subject	Tabulation	STUDYID, USUBJID, LBTESTCD, LBDTC, LBTPT, LBREPNUM
MB [SDTMIG 3.4]	Microbiology Specimen	FINDINGSFINDINGS	One record per microbiology specimen finding per time point per visit per subject	Tabulation	
MI [SDTMIG 3.4]	Microscopic Findings	FINDINGSFINDINGS	One record per finding per specimen per subject	Tabulation	
MK [SDTMIG 3.4]	Musculoskeletal	FINDINGSFINDINGS	One record per assessment per visit	Tabulation	

Once again, using the menu "Add - Dataset Properties from CSV file" and "Add - Variable Properties from CSV file"

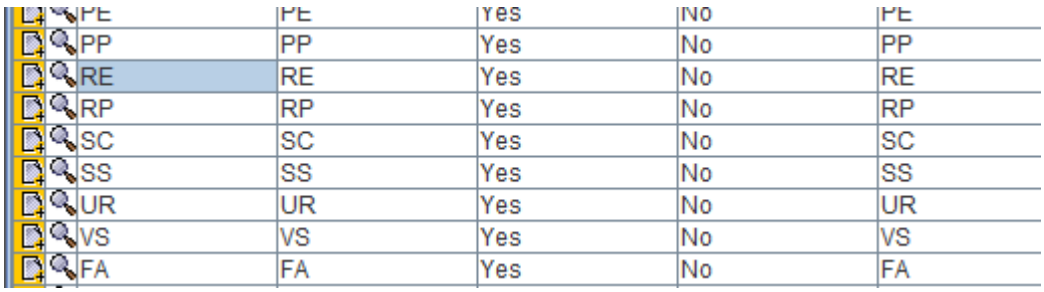
should not be the "normal" way to add information to the system. It is just a "workaround" for those companies who still keep their SDTM, SEND or ADaM specifications in worksheets like Excel, which we consider "bad practice".

Cleaning

Especially when starting from one of the templates, you will probably not want to keep a good number of domain or dataset definitions. For example, when your study is not a cancer study and also does not have questionnaires, in SDTM, you will probably want to drop QS (questionnaires), TU (Tumor/Lesion Identification), TR (Tumor/Lesion Results), RS (Disease Response and Clin Classification).
In order to do so, navigate to the tab "Dataset Definitions", and search for these domains (using the "Search" panel) one after the other, e.g.:

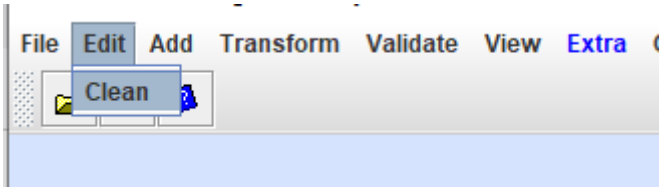


This will then immediately select the "TU" row.
Then remove it using the "Delete Selected Row" button. The system will ask for a confirmation.
After having removed QS, TU, TR and RS, the result is:

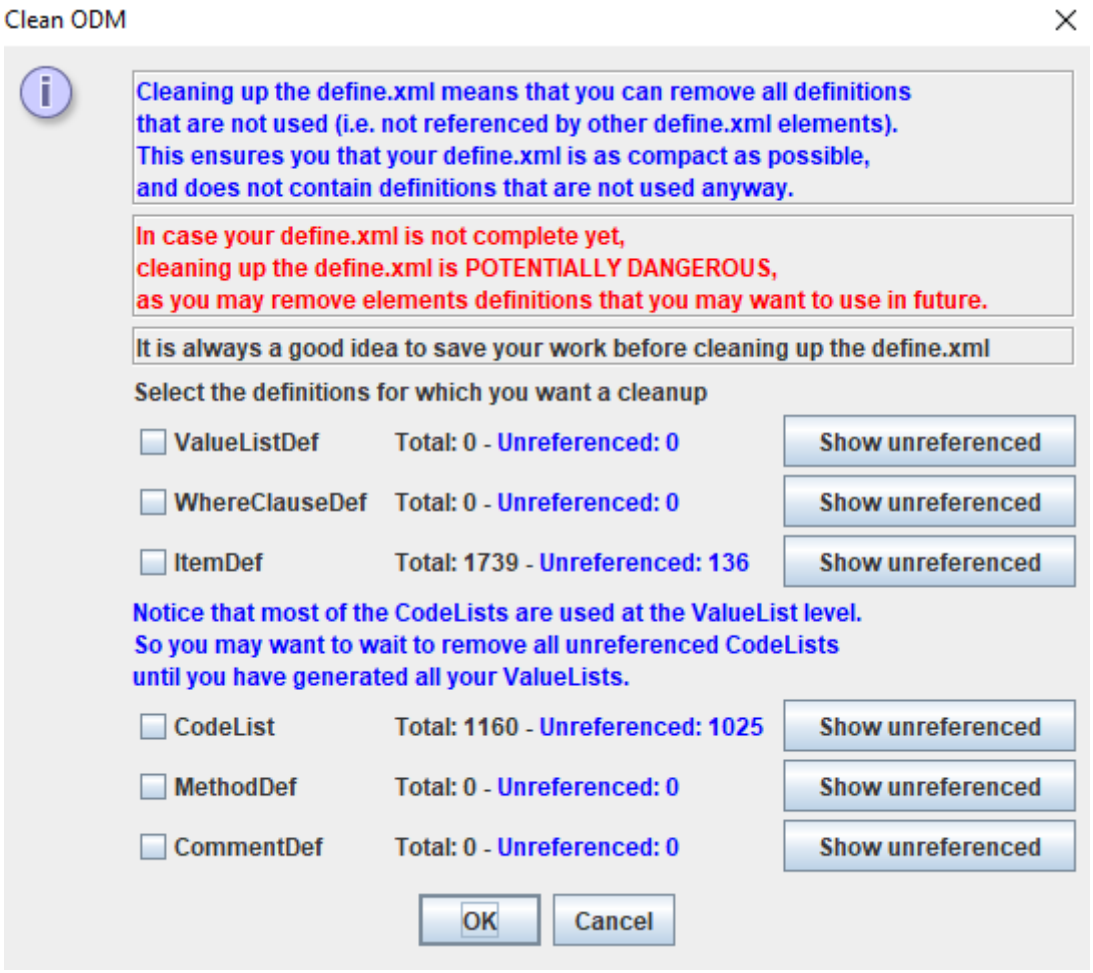


However, these (now removed) dataset definitions of course reference a lot of variable definitions, codelists, and maybe even valuelists, which have not been removed automatically. Keeping them to the end of the process, and then doing the "clean" doesn't harm, but some people prefer to remove them immediately after having removed the dataset definition itself.

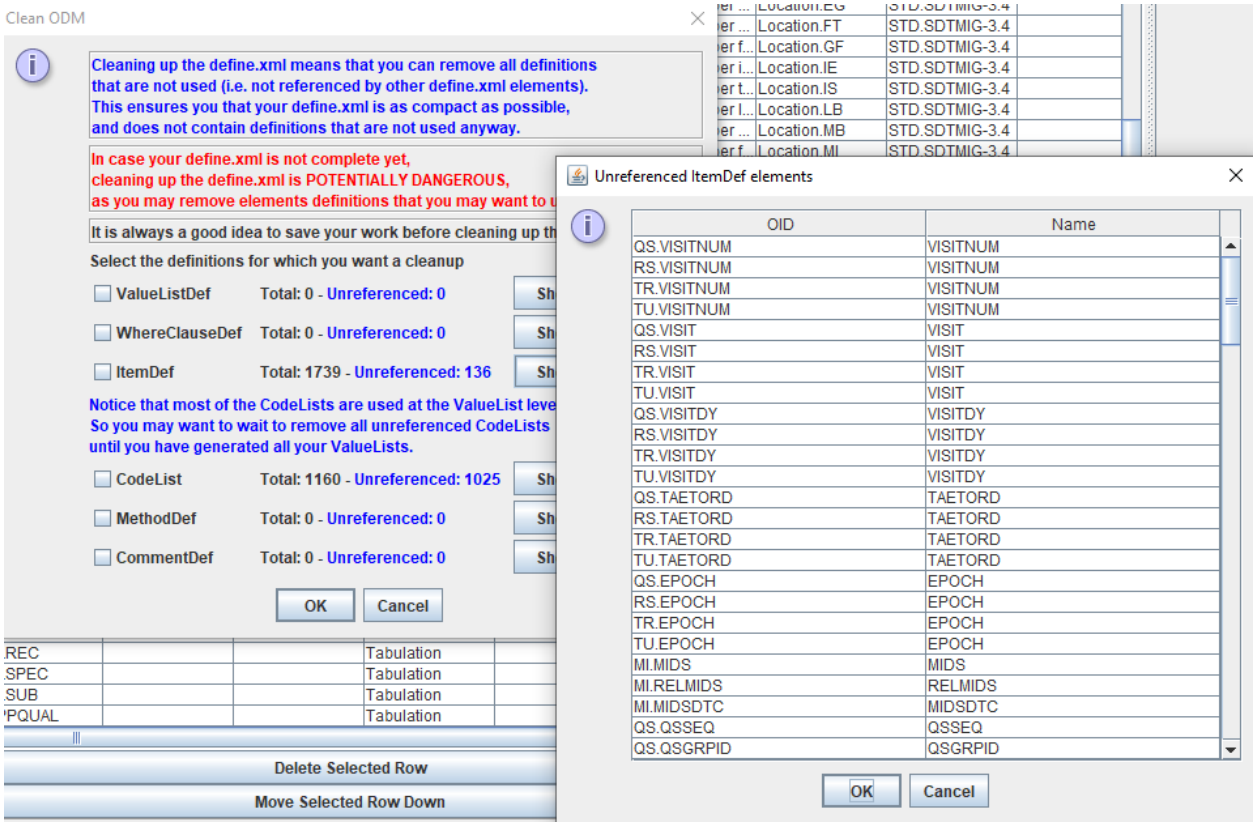
In order to do such a "Clean", use the menu "Edit - Clean":



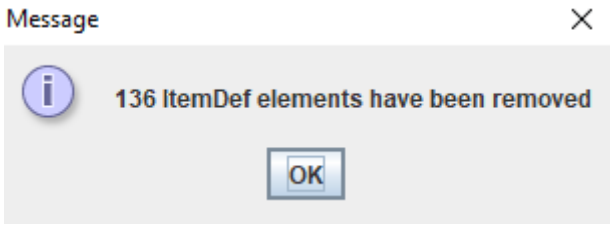
leading, after a few seconds, to a new dialog:



explaining that there are now 136 unused (unreferenced) variable definitions, and (surprise!) 1025 unreferenced CodeLists. As the dialog however states, most of these are to be used at the ValueList level, so removing them before all ValueLists have been developed, removing all unreferenced CodeLists may not be a good idea.
Even if one does, one can later always add one or more CodeLists from the CDISC Controlled Terminology using the menu "Add - CDISC Controlled Terminology".
When clicking "Show Unreferenced" right from "ItemDef", a list is displayed:



When then only checking the checkbox for "ItemDef" and clicking "OK", the listed variable definitions are removed from the system:



Loading additional domains / dataset definitions from a template

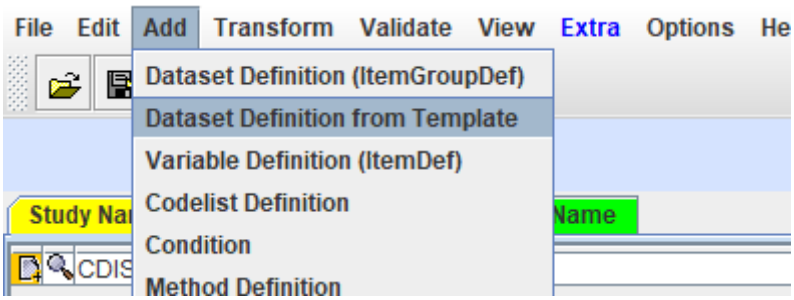
Suppose we have the following dataset definitions:

Standards | Annotated CRFs | Supplemental Documents | ValueList Definitions | WhereClause Definitions | Dataset Definitions | V

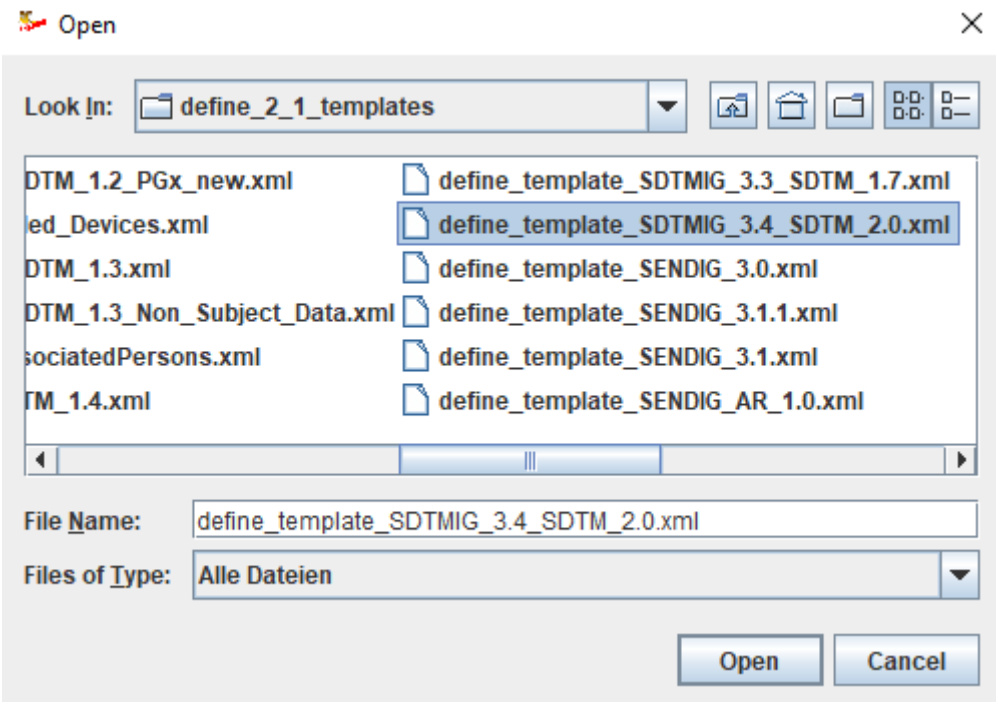
OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role
IG.TA	TA	No	Yes	TA	TA		
IG.TE	TE	No	Yes	TE	TE		
IG.TI	TI	No	Yes	TI	TI		
IG.TS	TS	No	Yes	TS	TS		
IG.TV	TV	No	Yes	TV	TV		
IG.DM	DM	No	No	DM	DM		
IG.SE	SE	Yes	No	SE	SE		
IG.SV	SV	Yes	No	SV	SV		
IG.CM	CM	Yes	No	CM	CM		
IG.EC	EC	Yes	No	EC	EC		
IG.EX	EX	Yes	No	EX	EX		
IG.AE	AE	Yes	No	AE	AE		
IG.DS	DS	Yes	No	DS	DS		
IG.MH	MH	Yes	No	MH	MH		
IG.DD	DD	Yes	No	DD	DD		
IG.FT	FT	Yes	No	FT	FT		
IG.IE	IE	Yes	No	IE	IE		
IG.LB	LB	Yes	No	LB	LB		
IG.NV	NV	Yes	No	NV	NV		
IG.OE	OE	Yes	No	OE	OE		
IG.QSPH	QSPH	Yes	No	QSPH	QS		
IG.QSSL	QSSL	Yes	No	QSSL	QS		
IG.RS	RS	Yes	No	RS	RS		
IG.VS	VS	Yes	No	VS	VS		
IG.FA	FA	Yes	No	FA	FA		
IG.RELREC	RELREC	Yes	No	RELREC	RELREC		
IG.SUPPDM	SUPPDM	Yes	No	SUPPDM	DM		
IG.SUPPEC	SUPPEC	Yes	No	SUPPEC	EC		
IG.SUPPNV	SUPPNV	Yes	No	SUPPNV	NV		
IG.SUPPOE	SUPPOE	Yes	No	SUPPOE	OE		
IG.DI	DI	No	Yes	DI	DI		

and want to add a dataset definition for the MB (Microbiology) domain, using one of the templates.
This can e.g. be the case when we start from a define.xml from another source, or from a prior, similar study, but we need to add additional dataset definitions.

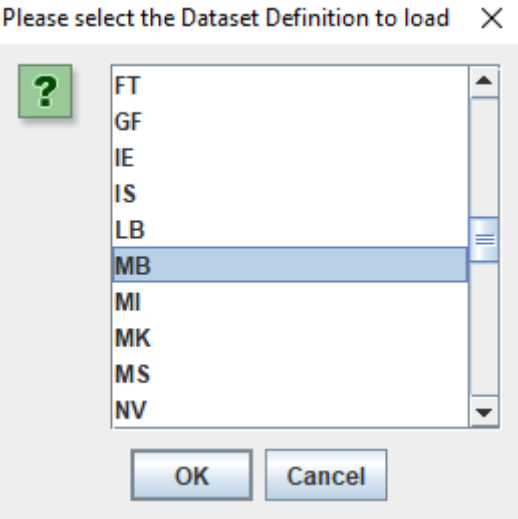
In order to add a dataset definition from a template, use the menu "Add - Dataset Definition from Template":



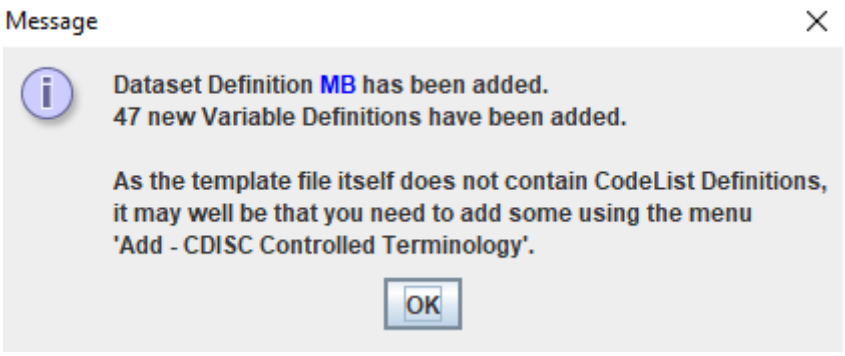
a file chooser is displayed, showing all the template files for the current version of the define.xml. For example for Define-XML v.2.1:



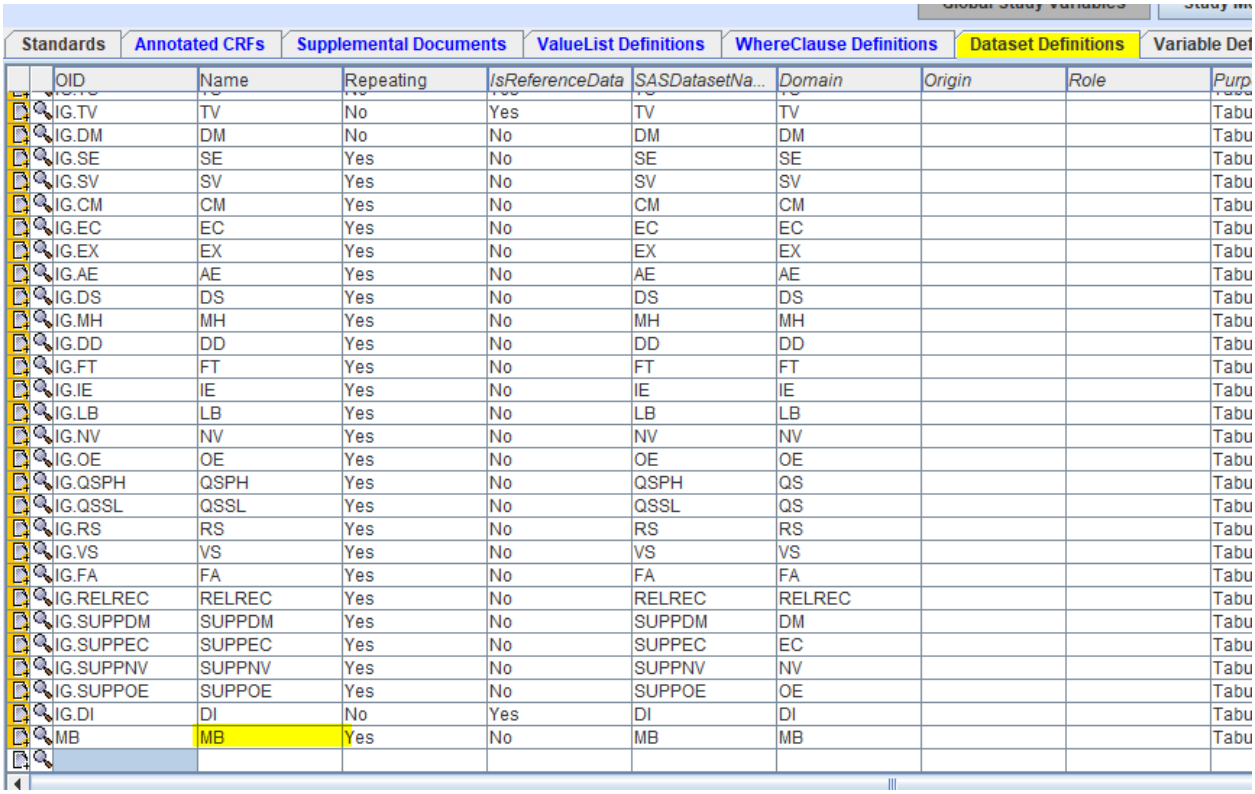
In our case, we select the template for SDTMIG.3.4. Remark that it is always a good idea to select a template for the standard version of the already loaded define.xml. This is not always possible, e.g. for SDTM when one want to load dataset definitions from the "Medical Device" standard or the "Associated Persons" standard.
After having selected the template file and clicking "Open", the system analyzes the file and shows us the dataset definitions that are present in that template file. For example:



where we select MB (Microbiology). After clicking "OK", the system loads the MB dataset definition from the template (ItemGroupDef) including all variable definitions (ItemDef) for that dataset definition, and that were not already present. It then comes with a summary of the results:



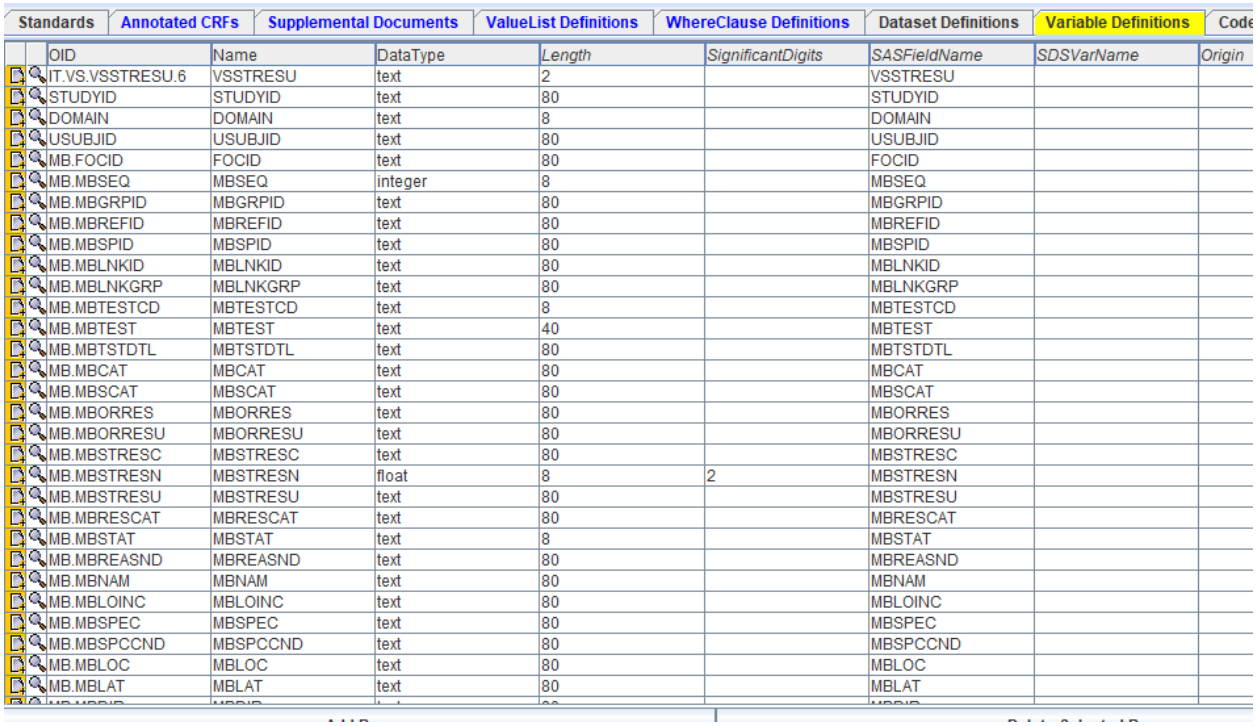
It also states that, as the template does not contain any CodeLists itself, one still may have to load additional controlled terminology and assign it to some of the variables.
If one then navigates to the "Dataset Definitions", one sees that a row has been added:



OID	Name	Repeating	IsReferenceData	SASDatasetNa...	Domain	Origin	Role	Purp
IG.TV	TV	No	Yes	TV	TV			Tabu
IG.DM	DM	No	No	DM	DM			Tabu
IG.SE	SE	Yes	No	SE	SE			Tabu
IG.SV	SV	Yes	No	SV	SV			Tabu
IG.CM	CM	Yes	No	CM	CM			Tabu
IG.EC	EC	Yes	No	EC	EC			Tabu
IG.EX	EX	Yes	No	EX	EX			Tabu
IG.AE	AE	Yes	No	AE	AE			Tabu
IG.DS	DS	Yes	No	DS	DS			Tabu
IG.MH	MH	Yes	No	MH	MH			Tabu
IG.DD	DD	Yes	No	DD	DD			Tabu
IG.FT	FT	Yes	No	FT	FT			Tabu
IG.IE	IE	Yes	No	IE	IE			Tabu
IG.LB	LB	Yes	No	LB	LB			Tabu
IG.NV	NV	Yes	No	NV	NV			Tabu
IG.OE	OE	Yes	No	OE	OE			Tabu
IG.QSPH	QSPH	Yes	No	QSPH	QS			Tabu
IG.QSSL	QSSL	Yes	No	QSSL	QS			Tabu
IG.RS	RS	Yes	No	RS	RS			Tabu
IG.VS	VS	Yes	No	VS	VS			Tabu
IG.FA	FA	Yes	No	FA	FA			Tabu
IG.RELREC	RELREC	Yes	No	RELREC	RELREC			Tabu
IG.SUPPDM	SUPPDM	Yes	No	SUPPDM	DM			Tabu
IG.SUPPEC	SUPPEC	Yes	No	SUPPEC	EC			Tabu
IG.SUPPNV	SUPPNV	Yes	No	SUPPNV	NV			Tabu
IG.SUPPOE	SUPPOE	Yes	No	SUPPOE	OE			Tabu
IG.DI	DI	No	Yes	DI	DI			Tabu
MB	MB	Yes	No	MB	MB			Tabu

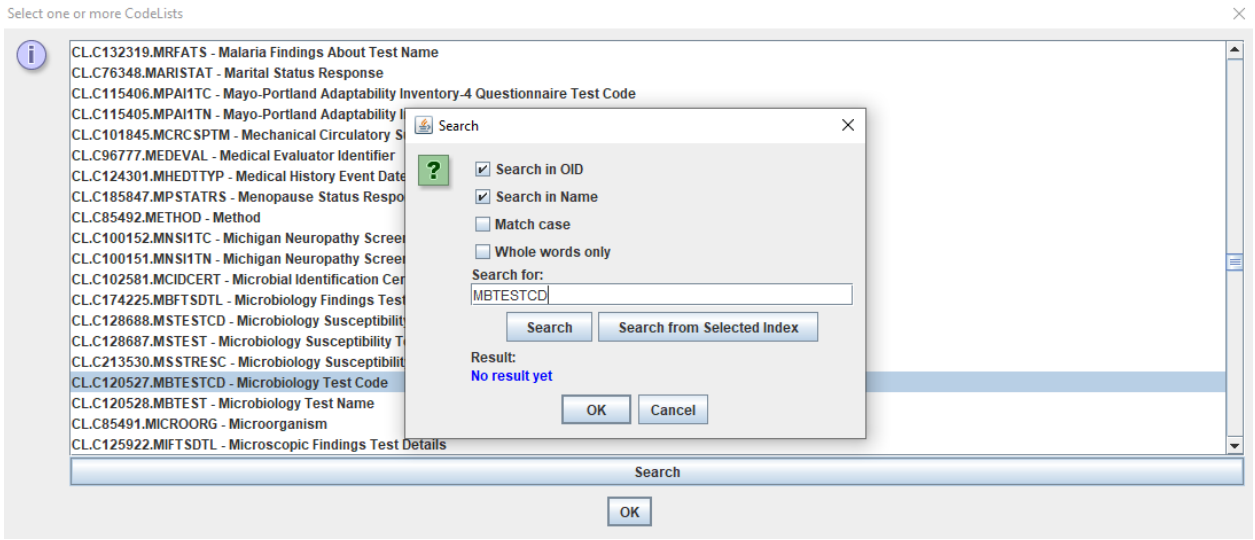
When the original dataset definition was not created by the "Define.xml Designer", it may well be that the OID (identifier) of the dataset definition has another form, but that is just fine, as OIDs are just arbitrary identifiers.

When one then navigates to the "Variable Definitions" tab, and scrolls to the bottom, one finds a number of variable definitions that were added for the MB dataset definition:

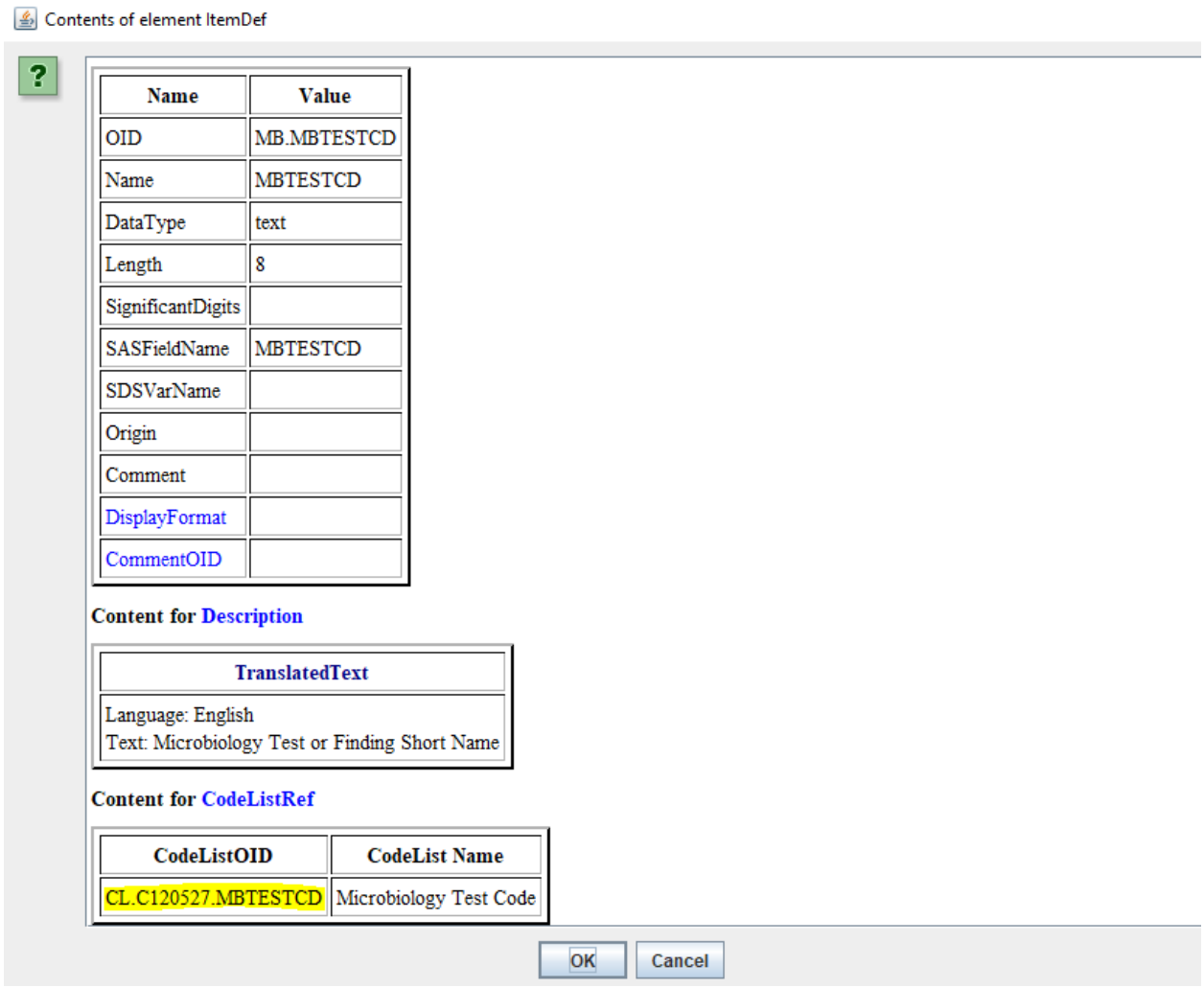


OID	Name	DataType	Length	SignificantDigits	SASFieldName	SDSVarName	Origin
IT.VS.VSSTRESU.6	VSSTRESU	text	2		VSSTRESU		
STUDYID	STUDYID	text	80		STUDYID		
DOMAIN	DOMAIN	text	8		DOMAIN		
USUBJID	USUBJID	text	80		USUBJID		
MB.FOCID	FOCID	text	80		FOCID		
MB.MBSEQ	MBSEQ	integer	8		MBSEQ		
MB.MBGRPID	MBGRPID	text	80		MBGRPID		
MB.MBREFID	MBREFID	text	80		MBREFID		
MB.MBSPID	MBSPID	text	80		MBSPID		
MB.MBLNKID	MBLNKID	text	80		MBLNKID		
MB.MBLNKGRP	MBLNKGRP	text	80		MBLNKGRP		
MB.MBTESTCD	MBTESTCD	text	8		MBTESTCD		
MB.MBTEST	MBTEST	text	40		MBTEST		
MB.MBTSTDTL	MBTSTDTL	text	80		MBTSTDTL		
MB.MBCAT	MBCAT	text	80		MBCAT		
MB.MBSCAT	MBSCAT	text	80		MBSCAT		
MB.MBORRES	MBORRES	text	80		MBORRES		
MB.MBORRESU	MBORRESU	text	80		MBORRESU		
MB.MBSTRESC	MBSTRESC	text	80		MBSTRESC		
MB.MBSTRESN	MBSTRESN	float	8	2	MBSTRESN		
MB.MBSTRESU	MBSTRESU	text	80		MBSTRESU		
MB.MBRESCAT	MBRESCAT	text	80		MBRESCAT		
MB.MBSTAT	MBSTAT	text	8		MBSTAT		
MB.MBREASND	MBREASND	text	80		MBREASND		
MB.MBNAM	MBNAM	text	80		MBNAM		
MB.MBLOINC	MBLOINC	text	80		MBLOINC		
MB.MBSPEC	MBSPEC	text	80		MBSPEC		
MB.MBSPCCND	MBSPCCND	text	80		MBSPCCND		
MB.MBLOC	MBLOC	text	80		MBLOC		
MB.MBLAT	MBLAT	text	80		MBLAT		

If no CodeList for e.g. MBTESTCD was already present, we will still need to add it using the menu "Add - CDISC Controlled Terminology", look for a codelist for MBTESTCD, and load it. I.e.:



and then check whether that it is indeed correctly referenced by the MBTESTCD variable definition:



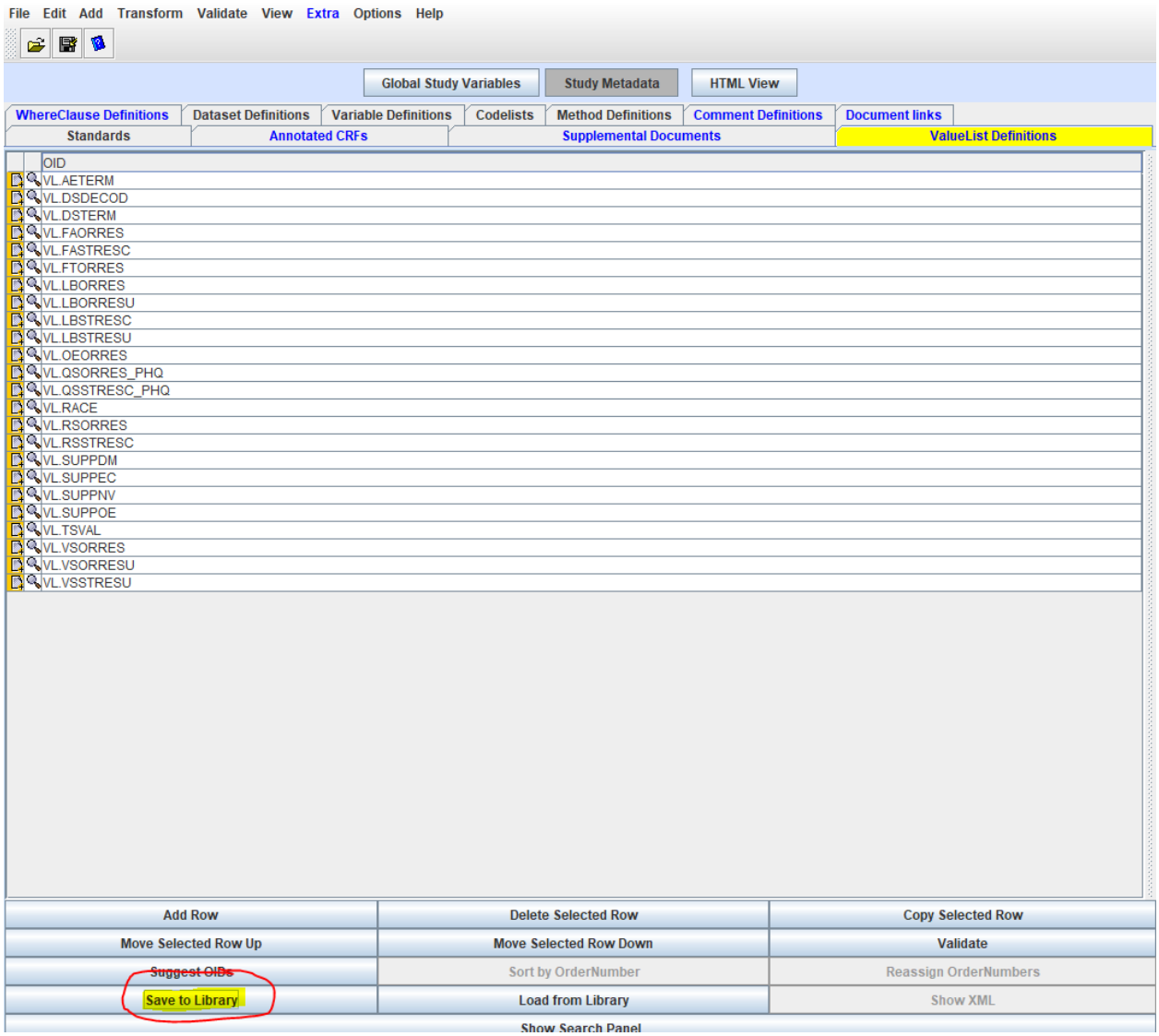
P.S. the other possible way to add a dataset definition is of course to use the menu "Add - Dataset Definition", which adds a row to the corresponding tab, and then adds an additional row at the bottom. This row can then be filled with information, and the necessary variable first be created and then added. This is of course a lot of work and can be error prone. This will more often the case when developing dataset definitions for ADaM, and seldom when generating dataset definitions for SDTM and SEND, with the exception of "sponsor-defined domains".

Saving to and loading from a local Library

When developing define.xml-s from "scratch", using e.g. based on a specification from the sponsor, i.e. the information which datasets need to be developed with which variables, which codelists, valuelists etc., it is always a good idea to develop "libraries" of items for later reuse. This can later save a lot of time. Suppose e.g. that a service provider received the specifications in the form of one or more Excel files, and for each new study, the list of datasets, variables, codelists etc. from that sponsor is similar, then the use of such "libraries" can be very efficient.

When we have e.g. developed a set of ValueLists and WhereClauses, then we can save these individually to the "library", and later reuse for the next study from the same sponsor and similar study.

To do so, select the tab of the type of items you would like to create a library file for:

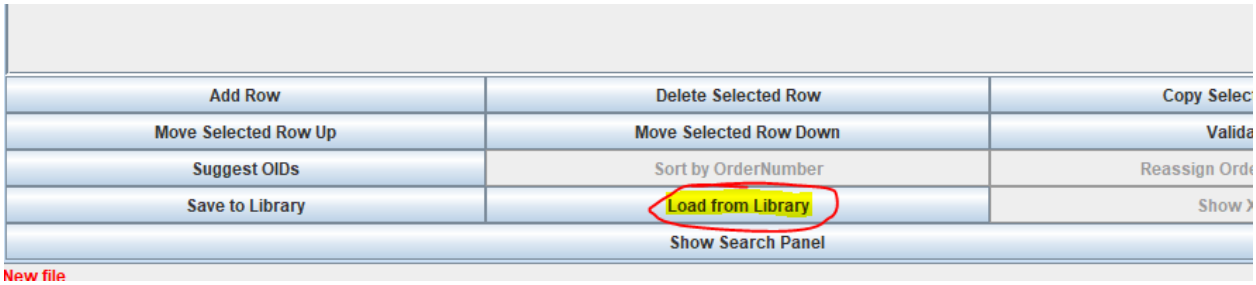


and click the button "Save to Library". The system will then first run a local validation against the Define-XML standard (using Schematron) and report possible issues. After that, a file chooser is displayed allowing to save the contents of the selected panel to an XML file. An example of the content of such a file is:

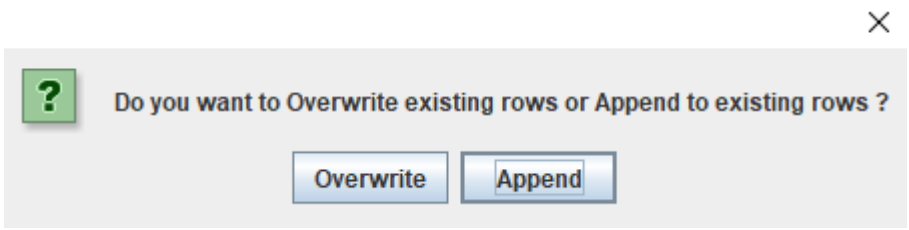
```xml
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.cdisc.org/ns/def/v2.1" ParentElementName="MetaDataVersion">
   <def:ValueListDef xmlns:def="http://www.cdisc.org/ns/def/v2.1" OID="VL.AETERM">
        <ItemRef xmlns="http://www.cdisc.org/ns/odm/v1.3" ItemOID="IT.AE.AETERM.1"
               Mandatory="Yes"
               OrderNumber="1">
          <def:WhereClauseRef WhereClauseOID="WC.AETERM1"/>
        </ItemRef>
        <ItemRef xmlns="http://www.cdisc.org/ns/odm/v1.3" ItemOID="IT.AE.AETERM.2"
               Mandatory="Yes"
               OrderNumber="2">
          <def:WhereClauseRef WhereClauseOID="WC.AETERM2"/>
        </ItemRef>
      </def:ValueListDef>
   <def:ValueListDef xmlns:def="http://www.cdisc.org/ns/def/v2.1" OID="VL.DSDECOD">
        <ItemRef xmlns="http://www.cdisc.org/ns/odm/v1.3" ItemOID="IT.DS.DSDECOD.3"
               Mandatory="No"
               OrderNumber="1">
          <def:WhereClauseRef WhereClauseOID="WC.DSDECOD1"/>
        </ItemRef>
        <ItemRef xmlns="http://www.cdisc.org/ns/odm/v1.3" ItemOID="IT.DS.DSDECOD.4"
               Mandatory="No"
               OrderNumber="2">
          <def:WhereClauseRef WhereClauseOID="WC.DSDECOD2"/>
        </ItemRef>
      </def:ValueListDef>
   <def:ValueListDef xmlns:def="http://www.cdisc.org/ns/def/v2.1" OID="VL.DSTERM">
        <ItemRef xmlns="http://www.cdisc.org/ns/odm/v1.3" ItemOID="IT.DS.DSTERM.1" Mandatory="No"
               OrderNumber="1">
          <def:WhereClauseRef WhereClauseOID="WC.DSTERM1"/>
        </ItemRef>
        <ItemRef xmlns="http://www.cdisc.org/ns/odm/v1.3" ItemOID="IT.DS.DSTERM.2" Mandatory="No"
               OrderNumber="2">
          <def:WhereClauseRef WhereClauseOID="WC.DSTERM2"/>
        </ItemRef>
      </def:ValueListDef>
   <def:ValueListDef xmlns:def="http://www.cdisc.org/ns/def/v2.1" OID="VL.FAORRES">
        <ItemRef xmlns="http://www.cdisc.org/ns/odm/v1.3" ItemOID="IT.FA.FAORRES.1"
```

which essentially is just a subset of a define.xml file, but only for the current type of element.

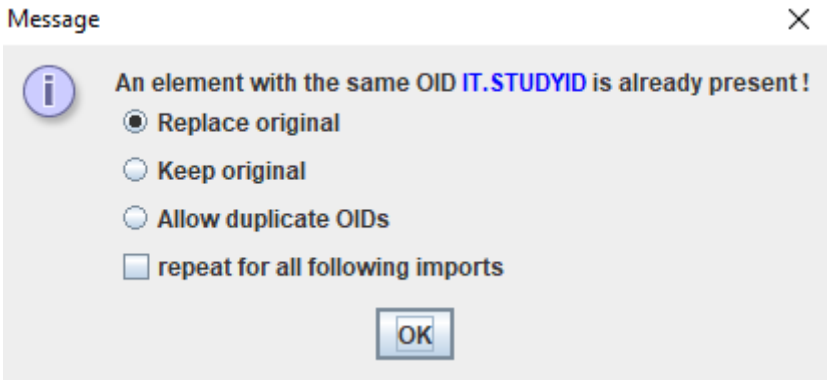
Like this, the user can develop sets of e.g. variables, dataset definitions, ValueLists and WhereClauses and reuse them. When then developing the define.xml for another study, the elements can then be loaded again using the button "Load from Library":



One can also repeat this when one has different such files for the same panel. For example, when one has such a library file for all variable of DM, and one for LB, one can load these after each other. The system will then first ask whether one want to append or replace the already present definitions:



When appending, and in case there are duplicates, such as for STUDYID, DOMAIN and USUBJID, the system will notice this, and ask the user what to do. For example, for STUDYID:



Normally, "Allow duplicate OIDs" is not a good idea, but can be helpful when one wants to change the OID and (possibly also) Name of the variable immediately after loading. This can e.g. be an option for VISITNUM, e.g. when one wants to have it as an integer in one domain/dataset (such as TV - Trial Visits) and as a float for another, e.g. when one expects "Unscheduled Visits" in SV (Subject Visits).

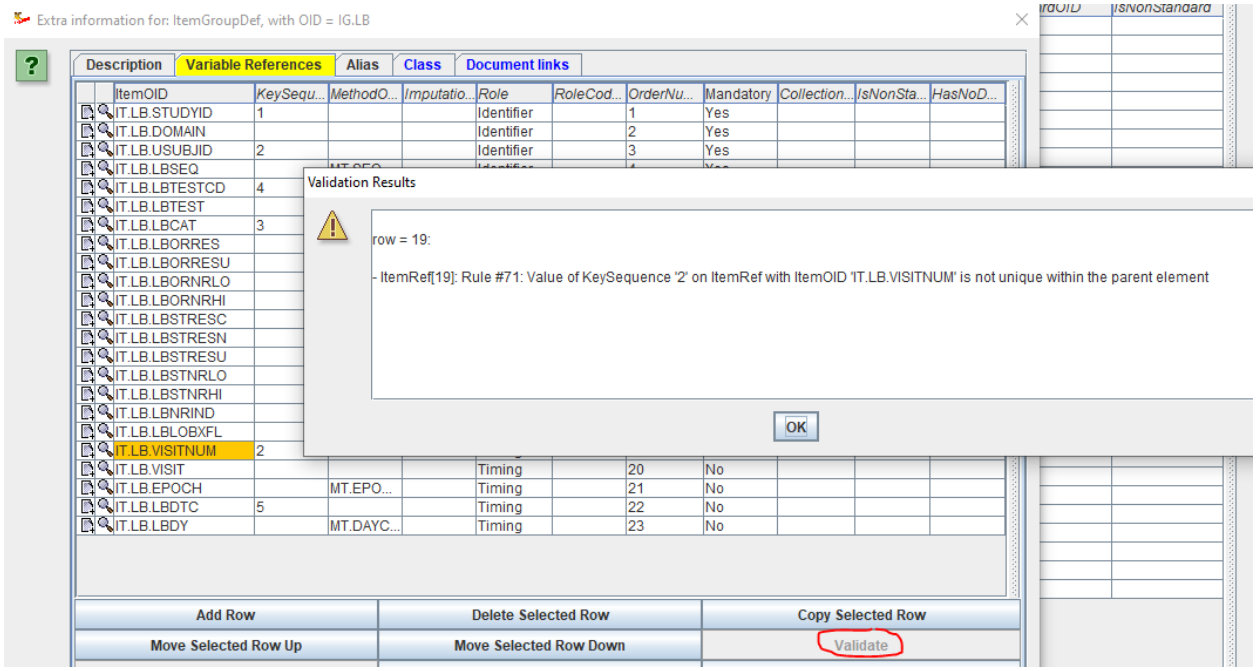
The result can then e.g. be:

Standards | Annotated CRFs | Supplemental Documents | ValueList Definitions | WhereClause Definitions | Dataset Definitions | Variable Definitions | Codelists

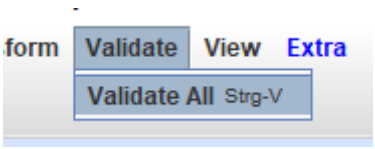
OID	Name	DataType	Length	SignificantDigits	SASFieldName	SDSVarName	Origin
IT.STUDYID	STUDYID	text	12		STUDYID		
IT.DOMAIN	DOMAIN	text	2		DOMAIN		
IT.USUBJID	USUBJID	text	8		USUBJID		
IT.DM.SUBJID	SUBJID	text	4		SUBJID		
IT.DM.RFSTDTC	RFSTDTC	date			RFSTDTC		
IT.DM.RFENDTC	RFENDTC	date			RFENDTC		
IT.DM.RFXSTDTC	RFXSTDTC	date			RFXSTDTC		
IT.DM.RFXENDTC	RFXENDTC	date			RFXENDTC		
IT.DM.RFICDTC	RFICDTC	date			RFICDTC		
IT.DM.RFPENDTC	RFPENDTC	date			RFPENDTC		
IT.DM.DTHDTC	DTHDTC	date			DTHDTC		
IT.DM.DTHFL	DTHFL	text	1		DTHFL		
IT.DM.SITEID	SITEID	text	3		SITEID		
IT.DM.BRTHDTC	BRTHDTC	date			BRTHDTC		
IT.DM.AGE	AGE	integer	8		AGE		
IT.DM.AGEU	AGEU	text	5		AGEU		
IT.DM.SEX	SEX	text	1		SEX		
IT.DM.RACE	RACE	text	41		RACE		
IT.DM.ETHNIC	ETHNIC	text	22		ETHNIC		
IT.DM.ARMCD	ARMCD	text	8		ARMCD		
IT.DM.ARM	ARM	text	28		ARM		
IT.DM.ACTARMCD	ACTARMCD	text	8		ACTARMCD		
IT.DM.ACTARM	ACTARM	text	28		ACTARM		
IT.DM.ARMNRS	ARMNRS	text	14		ARMNRS		
IT.DM.ACTARMUD	ACTARMUD	text	200		ACTARMUD		
IT.DM.COUNTRY	COUNTRY	text	3		COUNTRY		
IT.LB.LBSEQ	LBSEQ	integer	3		LBSEQ		
IT.LB.LBTESTCD	LBTESTCD	text	7		LBTESTCD		
IT.LB.LBTEST	LBTEST	text	39		LBTEST		
IT.LB.LBCAT	LBCAT	text	10		LBCAT		
IT.LB.LBORRES	LBORRES	text	6		LBORRES		
IT.LB.LBORRESU	LBORRESU	text	7		LBORRESU		
IT.LB.LBORNRLO	LBORNRLO	text	200		LBORNRLO		
IT.LB.LBORNRHI	LBORNRHI	text	200		LBORNRHI		
IT.LB.LBSTRESC	LBSTRESC	text	8		LBSTRESC		
IT.LB.LBSTRESN	LBSTRESN	float	8	5	LBSTRESN		
IT.LB.LBSTRESU	LBSTRESU	text	7		LBSTRESU		
IT.LB.LBSTNRLO	LBSTNRLO	float	5	3	LBSTNRLO		
IT.LB.LBSTNRHI	LBSTNRHI	float	5	2	LBSTNRHI		
IT.LB.LBNRIND	LBNRIND	text	8		LBNRIND		
IT.LB.LBLOBXFL	LBLOBXFL	text	1		LBLOBXFL		

Validating the define.xml

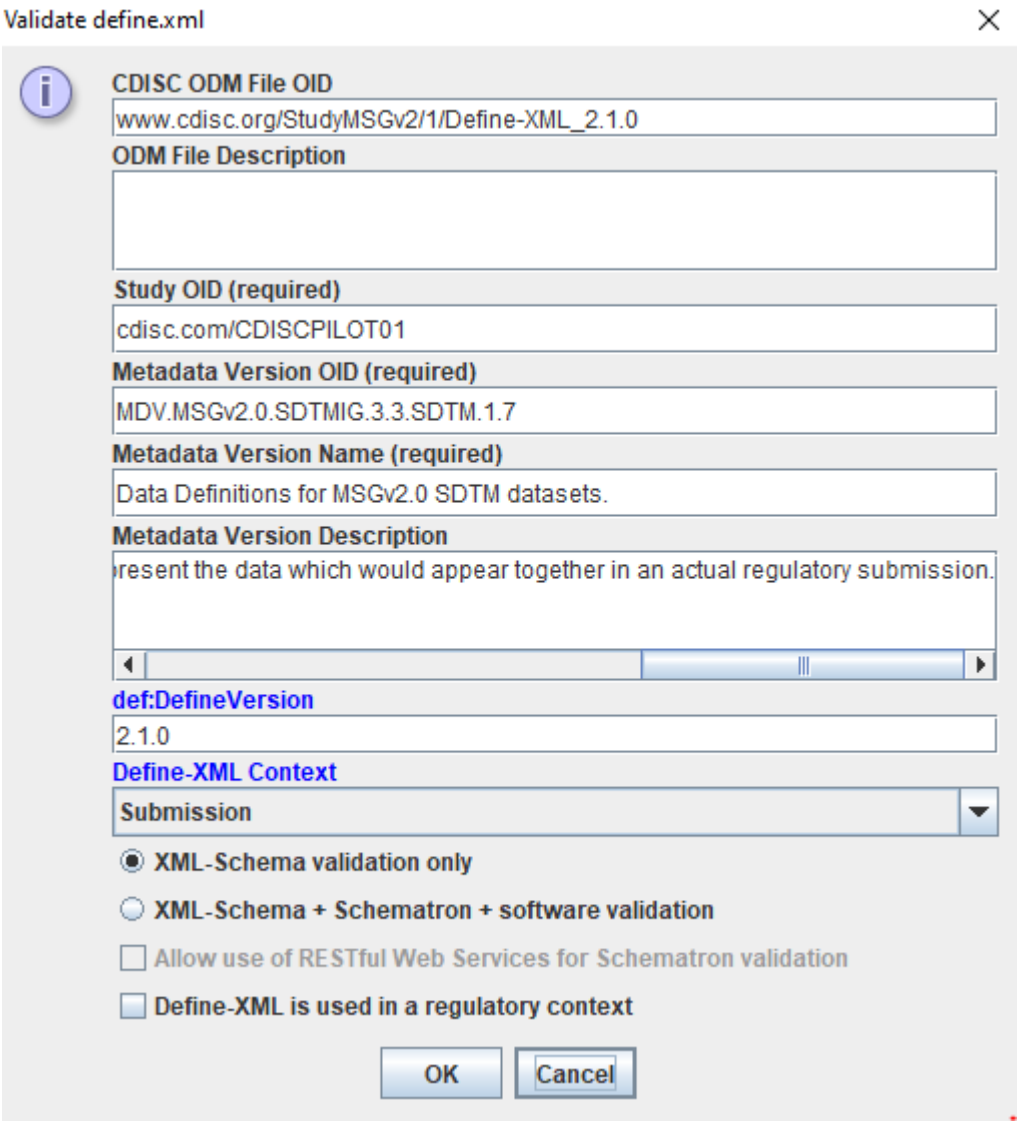
We have already seen that most of the panels have a "Validate" button to perform "local" validation. For example, when inspecting the "Variable References" (ItemRefs) for the LB dataset definition, and we made an error in assigning the "keys" (KeySequece attribute), and click the "Validate" button, we may find:



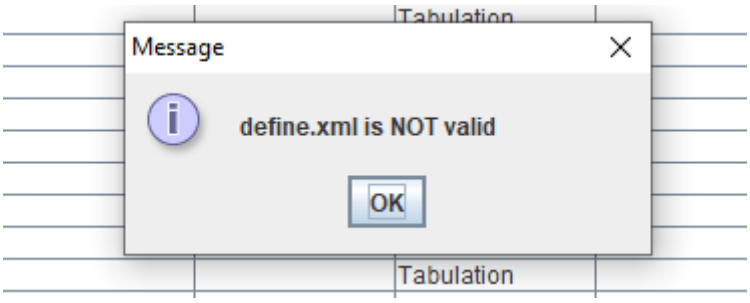
At regular moments in the process, we may however also want to do validation on the whole of the define.xml. In order to do so, use the menu "Validate - Validate All":



or use Ctrl-V on the keyboard. This leads to a dialog;

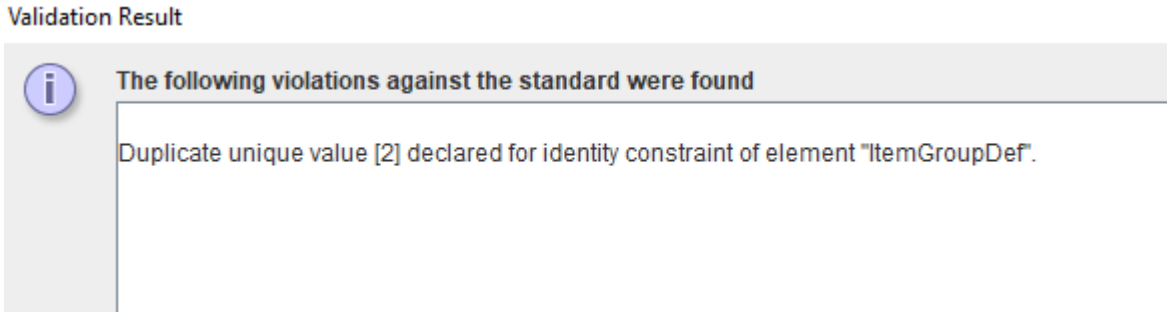


If some information is still missing (like a "Metadata Version Description") one will still want to add it into the fields of the upper part. In the lower part one can choose by either a (fast) validation against the XML-Schema, or a more deep validation using the Define-XML Schematron. The latter can be found in the folder "Schematron", so that one can inspect the validation rules oneself[14].
Let us first do a simple "XML-Schema validation only" and see whether it can detect the problem of the duplicate keys in the dataset definition for LB. So, we just click "OK". The result is:

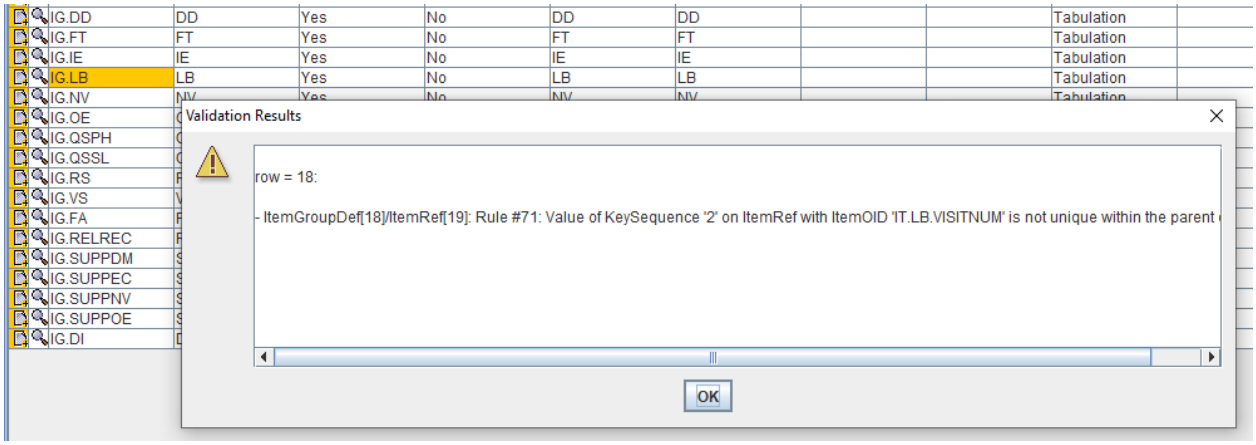


and more explanation is provided after clicking "OK":

[14] This is far superior to the Pinnacle21 validation for define.xml files. P21 is also completely "black box" - it is not possible at all to find out how the P21 "self-invented" rules have been implemented.

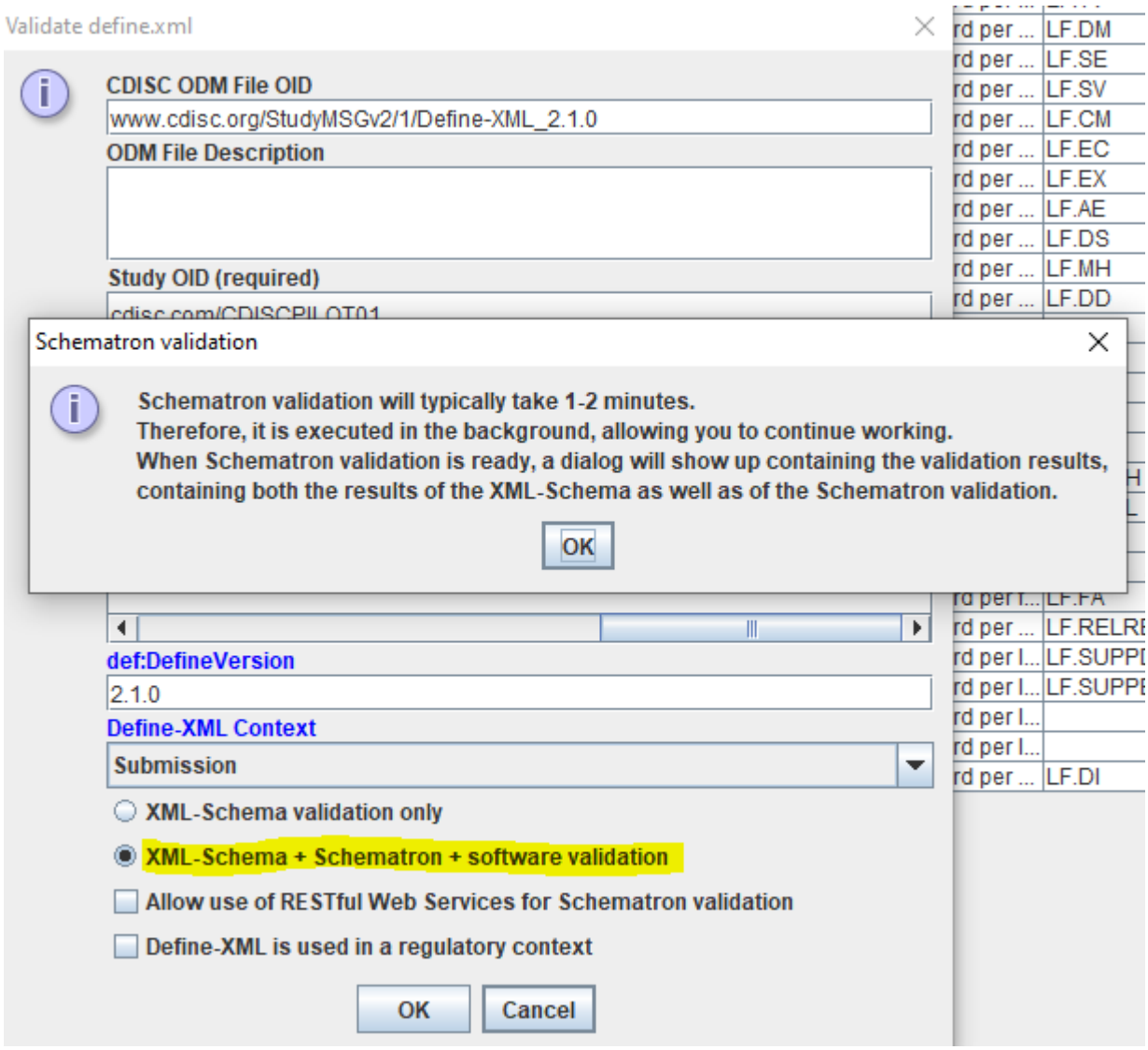


where we see the schema message "*Duplicate unique value [2] declared for identity constraint of element "ItemGroupDef"*."
For non-specialists, this may be not very well-explanatory, which is a well-known problem of XML-Schema validation messages. It however tells us that something is wrong with an "ItemGroupDef", which is representing dataset definitions, so we may want to go back to the "Dataset Definitions" panel, and do a "local" validation there. When we do so and use the "Validate" button (near the bottom, on the right), we get:

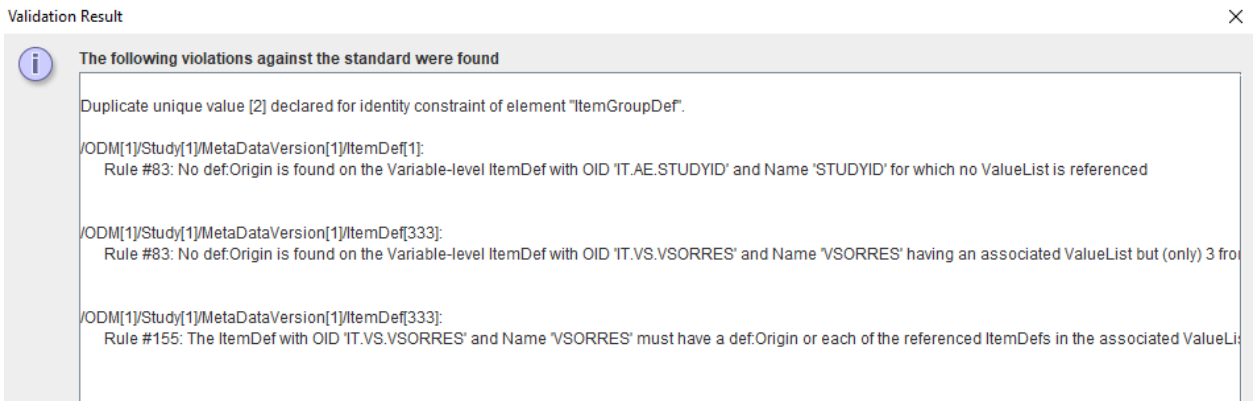


Providing a more clear message (as the local validation uses Schematron), and the cell for "IG.LB" being highlighted.

The second possibility is to do XML-Schema validation plus more advanced Schematron validation. When we select the radiobutton:



a message is displayed that the validation will be run in the background (so that the user can do other things in the mean time) as the process can take 1-2 minutes. After clicking "OK" in both the dialogs, the process starts and after 1-2 minutes, the message "define.xml is not valid" is shown again, and when then clicking "OK" more information is provided:



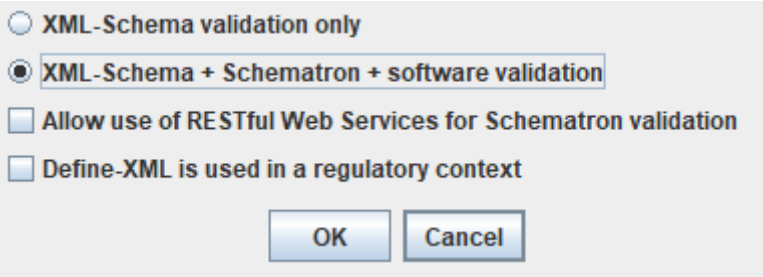
with the full text here:

Duplicate unique value [2] declared for identity constraint of element "ItemGroupDef".
/ODM[1]/Study[1]/MetaDataVersion[1]/ItemDef[1]: Rule #83: No def:Origin is found on the Variable-level ItemDef with OID 'IT.AE.STUDYID' and Name 'STUDYID' for which no ValueList is referenced

/ODM[1]/Study[1]/MetaDataVersion[1]/ItemDef[333]: Rule #83: No def:Origin is found on the Variable-level ItemDef with OID 'IT.VS.VSORRES' and Name 'VSORRES' having an associated ValueList but (only) 3 from 5 of the ValueList ItemDef-s have a def:Origin present
/ODM[1]/Study[1]/MetaDataVersion[1]/ItemDef[333]: Rule #155: The ItemDef with OID 'IT.VS.VSORRES' and Name 'VSORRES' must have a def:Origin or each of the referenced ItemDefs in the associated ValueList must have a def:Origin

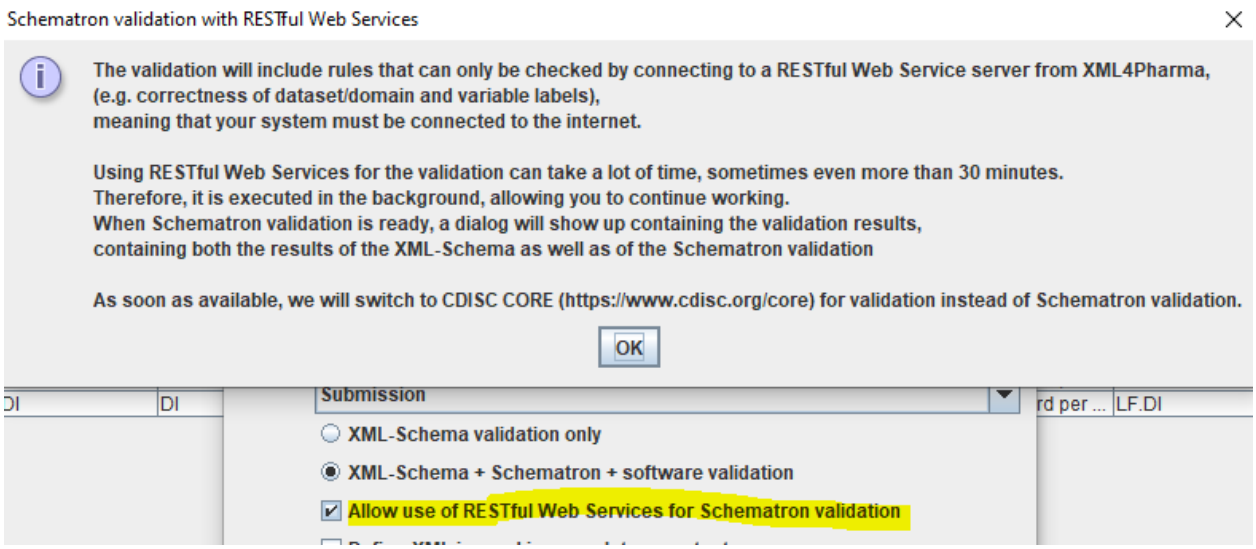
The first message comes from the XML-Schema, which we indeed already found before.
The second states that for the variable definition with OID "IT.AE.STUDYID" and Name "STUDYID", which does not have an associated ValueList, no def:Origin was found.
The third and fourth are variations of the same problem: there is no def:Origin on the variable definition for VSORRES, but not all ValueList associated ItemDefs do have an Origin assigned.

One surely has already noticed the two checkboxes near the bottom, the first only coming available when additional Schematron validation is selected:



The first one allows to also validate "special" Define-XML rules that require "lookups" using RESTful Web Services. These can be found in the file "define_2_1_rules_RWS.sch" for SDTM and "define_2_1_rules_SEND_RWS.sch" for SEND for Define-XML 2.1. There are also similar rules for Define-XML 2.0 for SDTM, but these are not actively maintained anymore.

When the checkbox "Allow use of RESTful Web Services …" is checked, a message is displayed:



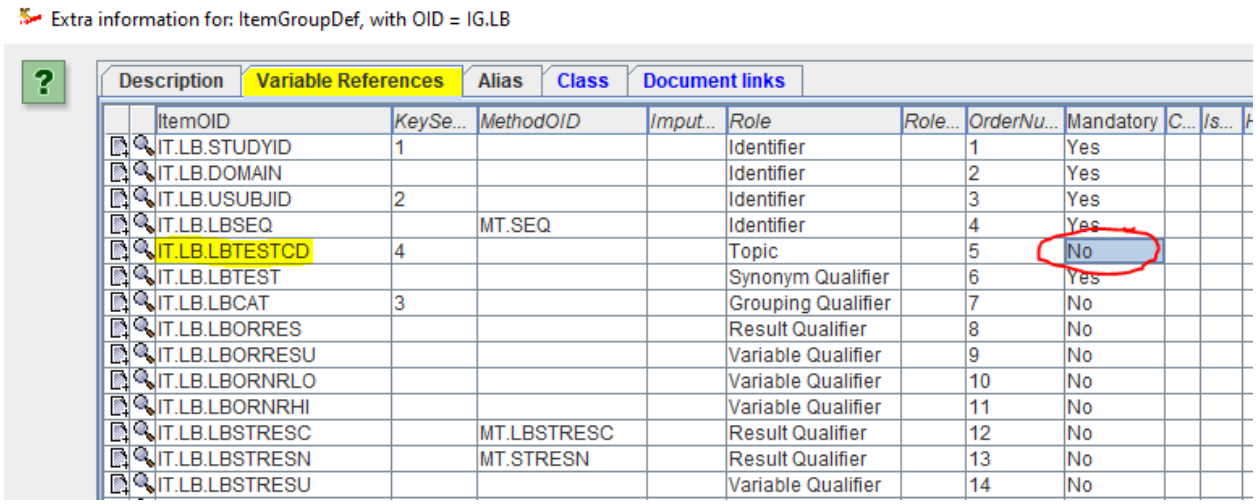
explaining that the system will submit queries to the XML4Pharma server (which of course requires an internet connection[15]). It also states that such a full analysis can take considerable time, so it is run in the background, so that the user can continue with other things. When the analysis is finalized, a message dialog will be displayed.

Within CDISC CORE, the team is currently (February 2026) busy starting bringing everything together to also implement Define-XML rules that need to make checks against the CDISC-Library using its API. When this is ready (which still may take some time), we will replace the current RestFul Web Service by the use of CORE.

---

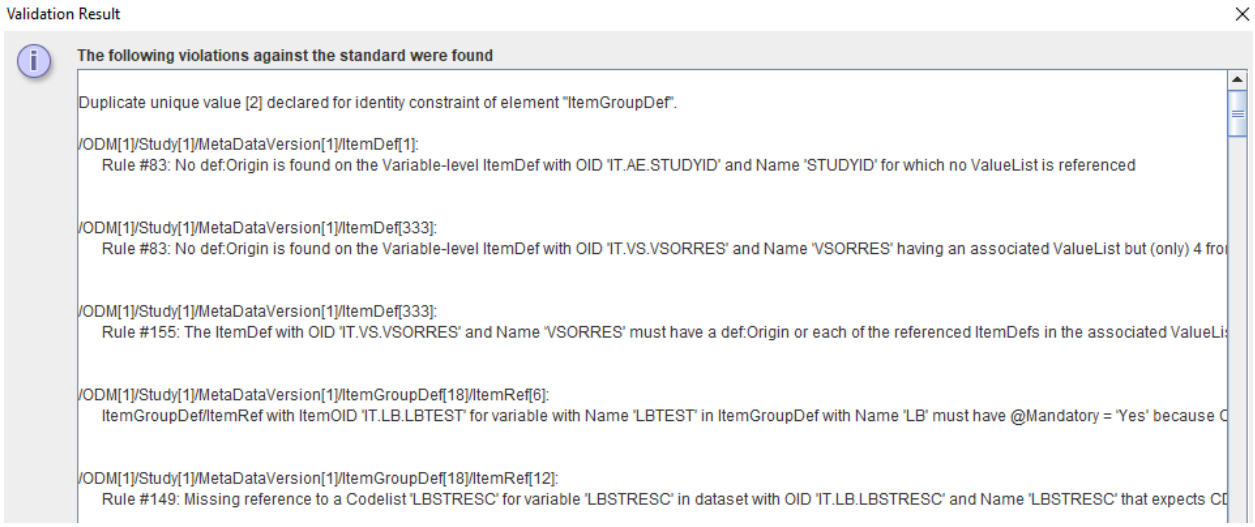
[15] It also requires that queries over port 8080 are allowed, as that is the port that the RESTful Web Services uses.
If no internet connection is available or fails (e.g. server down) the "local" validation part will still run, and a message about the failed use of the RESTful Web Services will be displayed.

One such a rule e.g. for SDTM is that when the variable is "Required", the variable reference to it (ItemGroupDef/ItemRef) must have the attribute and value Mandatory="Yes". So for testing, let us set Mandatory="No" on the "required" variable for LBTESTCD:



ItemOID	KeySe...	MethodOID	Imput...	Role	Role...	OrderNu...	Mandatory	C...	Is...
IT.LB.STUDYID	1			Identifier		1	Yes		
IT.LB.DOMAIN				Identifier		2	Yes		
IT.LB.USUBJID	2			Identifier		3	Yes		
IT.LB.LBSEQ		MT.SEQ		Identifier		4	Yes		
IT.LB.LBTESTCD	4			Topic		5	No		
IT.LB.LBTEST				Synonym Qualifier		6	Yes		
IT.LB.LBCAT	3			Grouping Qualifier		7	No		
IT.LB.LBORRES				Result Qualifier		8	No		
IT.LB.LBORRESU				Variable Qualifier		9	No		
IT.LB.LBORNRLO				Variable Qualifier		10	No		
IT.LB.LBORNRHI				Variable Qualifier		11	No		
IT.LB.LBSTRESC		MT.LBSTRESC		Result Qualifier		12	No		
IT.LB.LBSTRESN		MT.STRESN		Result Qualifier		13	No		
IT.LB.LBSTRESU				Variable Qualifier		14	No		

When we then run the validation again with the checkbox "Allow use of RESTful Web Services …" checked, at the end, we get:



Some of the messages in table form for better readability:

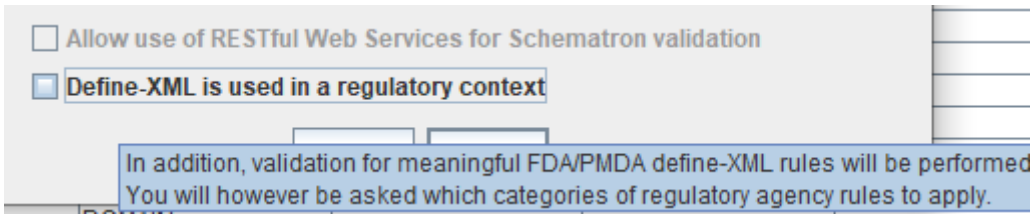
Message	**Reason / Explanation**
/ODM[1]/Study[1]/MetaDataVersion[1]/ItemDef[333]: Rule #83: No def:Origin is found on the Variable-level ItemDef with OID 'IT.VS.VSORRES' and Name 'VSORRES' having an associated ValueList but (only) 4 from 5 of the ValueList ItemDef-s have a def:Origin present	VSORRES itself does not have an Origin assigned. This has been delegated to the ValueList level. However, only 4 of the 5 ValueList ItemDefs do have an Origin assigned.
ODM[1]/Study[1]/MetaDataVersion[1]/ItemGroupDef[18]/ItemRef[6]: ItemGroupDef/ItemRef with ItemOID 'IT.LB.LBTEST' for variable with Name 'LBTEST' in ItemGroupDef with Name 'LB' must have @Mandatory = 'Yes' because Core = 'Req' in standard 'SDTMIG' version '3.3'	In the define.xml, LBTEST has not been assigned 'Mandatory="Yes"' although it it a "required" variable.
/ODM[1]/Study[1]/MetaDataVersion[1]/ItemGroupDef[18]/ItemRef[12]: Rule #149: Missing reference to a Codelist 'LBSTRESC' for variable 'LBSTRESC' in dataset with OID 'IT.LB.LBSTRESC' and Name 'LBSTRESC' that expects CDISC Controlled Terminology according to the 'SDTMIG' standard version '3.3' - Number of ValueList items = 17 - Number of non-numeric ValueLists with a CodeList = 16	The assignment of a codelist for LBSTRESC has been delegated to the ValueList level, but there is one ValueList ItemDef that did not get a codelist assigned although it is marked as "non-numeric".

For the third case, the reason is that for LBTESTCD=GLUC, there is a value "<2.2204" so the mappers decided to assign DataType="text" to LBSTRESC for that. Maybe the rule should be further refined for such a case.

16	GLUC	Glucose	CHEMISTRY	74	mg/dL	50	250	4.10774	4,10774	mmol/L
53	GLUC	Glucose	CHEMISTRY	85	mg/dL	50	250	4.71835	4,71835	mmol/L
87	GLUC	Glucose	CHEMISTRY	<40	mg/dL	50	250	<2.2204		mmol/L
114	GLUC	Glucose	CHEMISTRY	48	mg/dL	50	250	2.66448	2,66448	mmol/L
132	GLUC	Glucose	CHEMISTRY	91	mg/dL	50	250	5.05141	5,05141	mmol/L
163	GLUC	Glucose	CHEMISTRY	79	mg/dL	50	250	4.38529	4,38529	mmol/L
16	GLUC	Glucose	CHEMISTRY	91	mg/dL	50	250	5.05141	5,05141	mmol/L

IMPORTANT REMARK
We cannot guarantee 100% availability of this RESTful Web Service!
If you would have this RESTful Web Service available on one of your own servers, please let us know so that we can help you with make that realize.

The checkbox "Define-XML is used in a regulatory context" will usually only be used in the case of Define-XML v.2.0, as the latter does not have a method to provide this information. For Define-XML 2.1, there is the "def:Context" attribute on the ODM element. If it is present (essentially, it should), it's value, which can be "Submission" or "Other" will supersede the value of the checkbox. As the tooltip on it tells us:



it is meant to implement e.g. FDA- or PMDA-specific rules for the define.xml. This however has not been implemented yet.