The Define.xml Designer 2016

User Manual for designing and developing Define.xml files

Last update: 2018-05-01

Introduction

This user manual describes the features for designing and developing define.xml v.1.0 and define.xml 2.0 files using one of the "Define.xml Designer" and the "ODM Study and Define.xml Designer 2016" software packages. For information about designing ODM Study design files, please see the separate user manual

Background

CDISC SDTM, SEND and ADaM electronic submissions to regulatory authorities such as the US FDA and the Japanese PMDA require a "define.xml" file to accompany the submission files. The define.xml file contains the metadata (data about data) of the information in the submission files. As such, a well designed define.xml file is of utmost importance for explaining the reviewers about what the data is about and how it was generated.
The define.xml file is an XML file that is as well human-readable as well as machine-readable. The human readability is ensured by a so called "stylesheet" (XSLT stylesheet), that transforms the XML into information that is interpretable by any modern browser (HTML+CSS).

It is however essentially the define.xml file itself (i.e. the XML) that carries the information - what is seen in the browser is just a human understandable presentation of that data[1].

Although XML is very easy to learn, many people desire to use a good software tool to design or develop a define.xml file describing the contents of their electronic submission, which does not require them to write XML. The "Define.xml Designer 2016" is such a tool.

Ideally, the define.xml file is already developed before study start or in the early stages of a clinical study. The reason for this is that the most important goal of a (commercial) clinical study is to get a market authorisation for the drug or therapy that is tested. So it is of utmost importance to already test whether the during the study captured data can be transformed to SDTM. If it is found that this is not possible at the end of the study, then the whole submission is endangered.
Many pharma companies already develop their mapping to SDTM together with the corresponding define.xml even before study start. The ideal tool to do so is XML4Pharma's SDTM-ETL software, which allows users to develop the mapping to SDTM starting from an ODM file with the study design.

There are however cases where such a synchronized (SDTM + define.xml) approach is not possible. This is especially the case for legacy studies where the SDTM files have already been generated and where the define.xml must be generated post-SDTM. Another case is when the user wants to start from a specific SDTM-IG and corresponding define.xml template file, and than add/remove variables and add all other information, adapting the define.xml to the needs of a specific study or submission. For example, many sponsors already do develop a define.xml as a specification to their third party service provider, containing the information how the sponsor wants to obtain the SDTM data files for a specific study at the end of the study. A third use case is that the user obtains a

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

define.xml that is incomplete, is not correct, or does not well explain the submission to the regulatory reviewer.

These use cases are covered by the "Define.xml Designer 2016" which is a part of the "ODM Study and Define.xml 2016 Designer", but is also available as a separate offering.

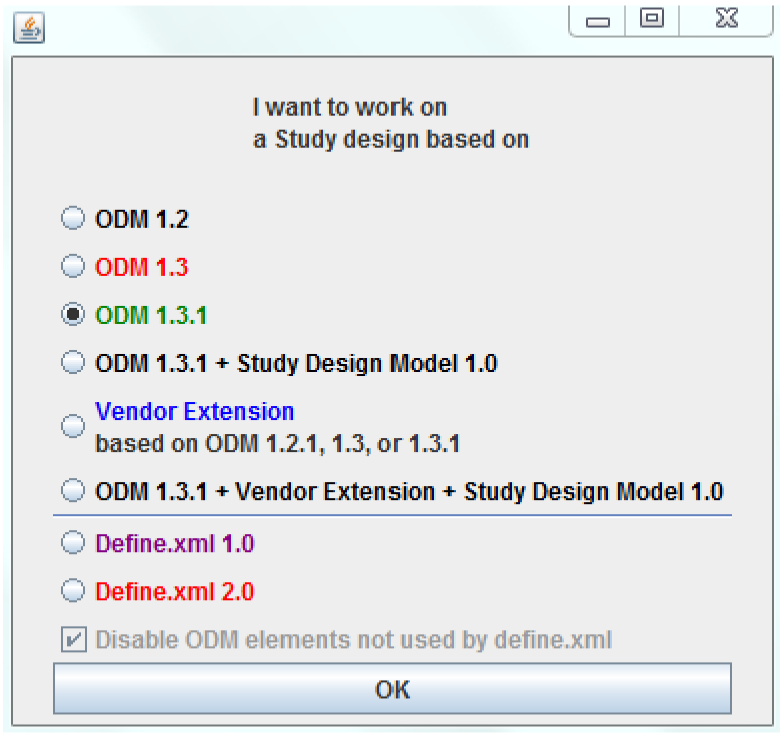
This manual describes how the "Define.xml Designer 2016" is used for these three and other use cases. No knowledge of XML is required. It is however recommended that the user has a basic understanding of the goals and structure of define.xml. This can be obtained by reading the define.xml specification (a copy is included with the distribution) and/or attending a one-day CDISC define.xml training. The latter will make you a define.xml expert, allowing you (when using this software) to develop jewel define.xml files that very well explain the regulatory reviewers what your data is about, how it was generated, and how it is organized.

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

Starting the software and main principles

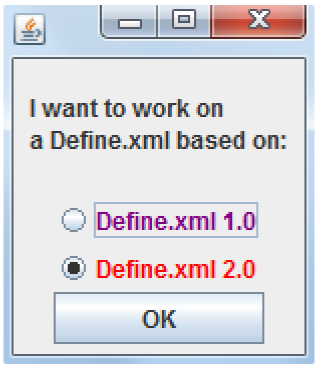
ODM Designer

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


I want to work on
a Study design based on

( ) ODM 1.2
( ) ODM 1.3
[✓ ODM 1.3.1]
( ) ODM 1.3.1 + Study Design Model 1.0
( ) Vendor Extension based on ODM 1.2.1, 1.3, or 1.3.1
( ) ODM 1.3.1 + Vendor Extension + Study Design Model 1.0
( ) Define.xml 1.0
( ) Define.xml 2.0
[✓] Disable ODM elements not used by define.xml

If you are using the "define.xml Designer" standalone software[2], the entry dialog will be:

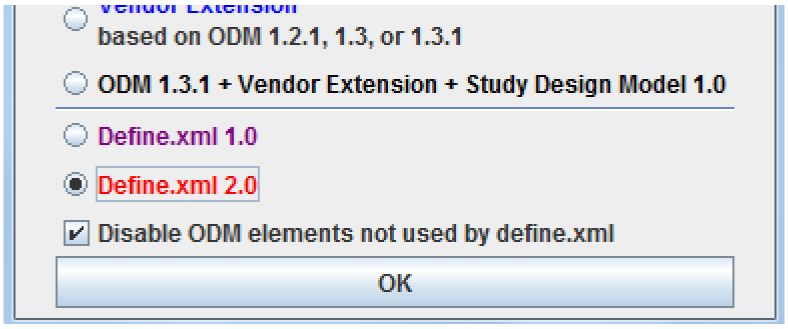

I want to work on
a Define.xml based on:

( ) Define.xml 1.0
[✓ Define.xml 2.0]

If you want to work with define.xml, select "Define.xml 2.0" in both cases. The software still supports define.xml 1.0, but it is **strongly recommended** to use define.xml 2.0 as this allows you to much better explain the contents and structure of your submission to the regulatory reviewers.

In case you purchased the "full" "ODM and define.xml Designer" software, and have selected the "Define.xml 2.0" radiobutton, there is a checkbox stating "Disable ODM elements not used by define.xml".

[2]The standalone "define.xml Designer" does not allow you to design studies in CDISC ODM, and to use a define.xml for a new study design.


based on ODM 1.2.1, 1.3, or 1.3.1

ODM 1.3.1 + Vendor Extension + Study Design Model 1.0

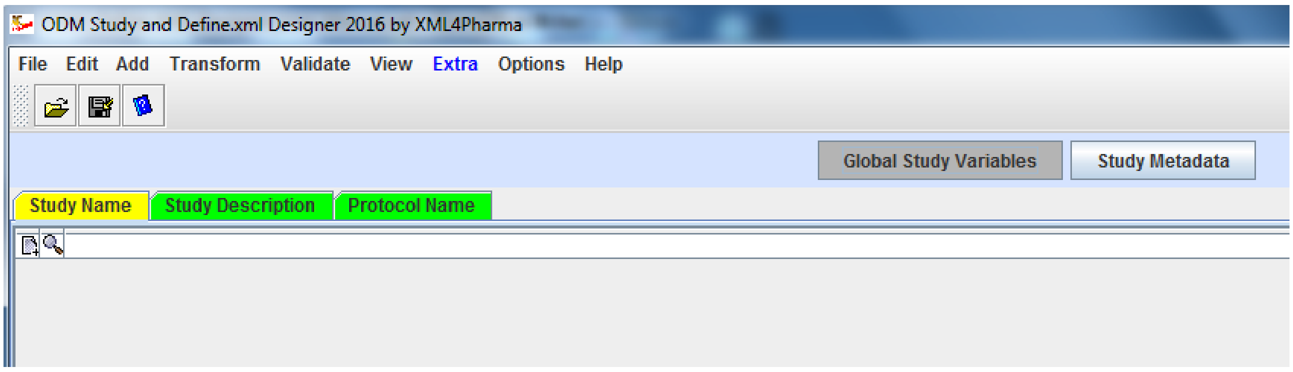
Define.xml 1.0

[✓ Define.xml 2.0]

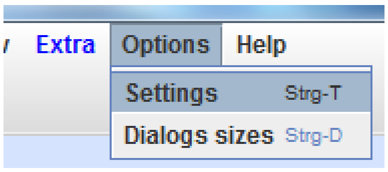
[✓] Disable ODM elements not used by define.xml

If you only want to develop a define.xml file, leave this button checked.
If you do however want to use an existing define.xml file to set up a new study design, with visits, forms, etc., uncheck the "Disable ODM elements not used by define.xml". Generating new study designs from an existing define.xml file is explained in a separate manual.

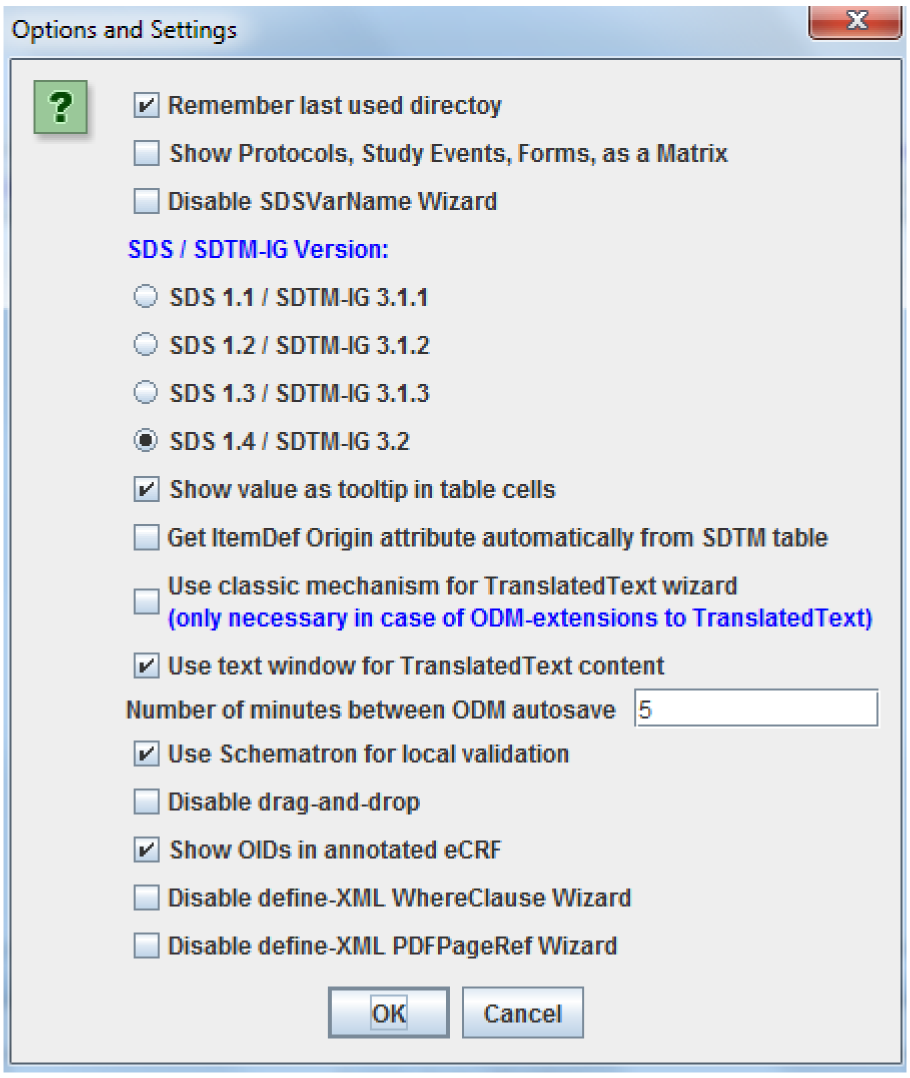
After having clicked the "OK" button, the software starts reading the define.xml XML-Schema (we use define.xml 2.0 as of this point), and generates the graphical user interface elements automatically from the XML-Schema. This results in:



The "Study Name", "Study Description" and "Protocol Name" already allow you to add some mandatory information. Whereas the "Study Name" and "Study Description" items are self-explaining, the "Protocol Name" is expected the contain the "*The sponsor's internal name assigned to the Study*" which usually is the title of the protocol.
Let us however first have a look at the "Options". To do so, use the menu "Options - Settings".



This opens a new dialog:



- [✓] Remember last used directoy
- [ ] Show Protocols, Study Events, Forms, as a Matrix
- [ ] Disable SDSVarName Wizard

SDS / SDTM-IG Version:

- ○ SDS 1.1 / SDTM-IG 3.1.1
- ○ SDS 1.2 / SDTM-IG 3.1.2
- ○ SDS 1.3 / SDTM-IG 3.1.3
- ● SDS 1.4 / SDTM-IG 3.2

- [✓] Show value as tooltip in table cells
- [ ] Get ItemDef Origin attribute automatically from SDTM table
- [ ] Use classic mechanism for TranslatedText wizard (only necessary in case of ODM-extensions to TranslatedText)
- [✓] Use text window for TranslatedText content

Number of minutes between ODM autosave: 5

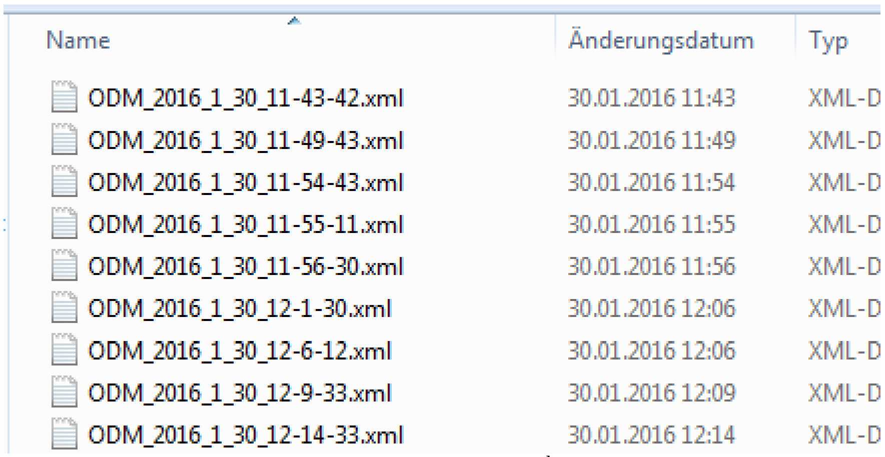
- [✓] Use Schematron for local validation
- [ ] Disable drag-and-drop
- [✓] Show OIDs in annotated eCRF
- [ ] Disable define-XML WhereClause Wizard
- [ ] Disable define-XML PDFPageRef Wizard

We will now discuss the most important options and settings.

- Usually you will want to have the software remember the last used import (file read) or export (file write) directory. You can disable this feature by unchecking the checkbox "Remember last used directory"
- The checkbox "Show protocols, Study Events, Forms, as a matrix is irrelevant for developing define.xml files. If you are however generating a new study design from an existing define.xml file, this checkbox allows you a "Schedule of Events" matrix of your study. This checkbox is disabled in the trimmed-down "define.xml only" version
- The checkbox "Disable SDSVarName Wizard" is used when generating a study design. It is not used for generating or developing define.xml file. See the other user manual for further details.
- The following set of radiobuttons allow you to select an SDS / SDTM-IG version for use with the wizards. Essentially all tables from the SDTM-IGs have been implemented in the software and are used by many of the wizards.
- The following checkbox "Get ItemDef Origin attribute automatically from SDTM table" is only used for designing ODM study designs in combination with "SDSVarName". It is irrelevant for pure define.xml designs.
- The wizard for "TranslatedText" (i.e. for support of multiple languages) can be disabled using the checkbox "Use classic mechanism for TranslatedText wizard".
- When adding text that goes within "TranslatedText", a separate small text window shows up for adding the text. If the user does not want this, but wants to add the text directly in the

table cell, then the checkbox "Use text window for TranslatedText content" should be unchecked.

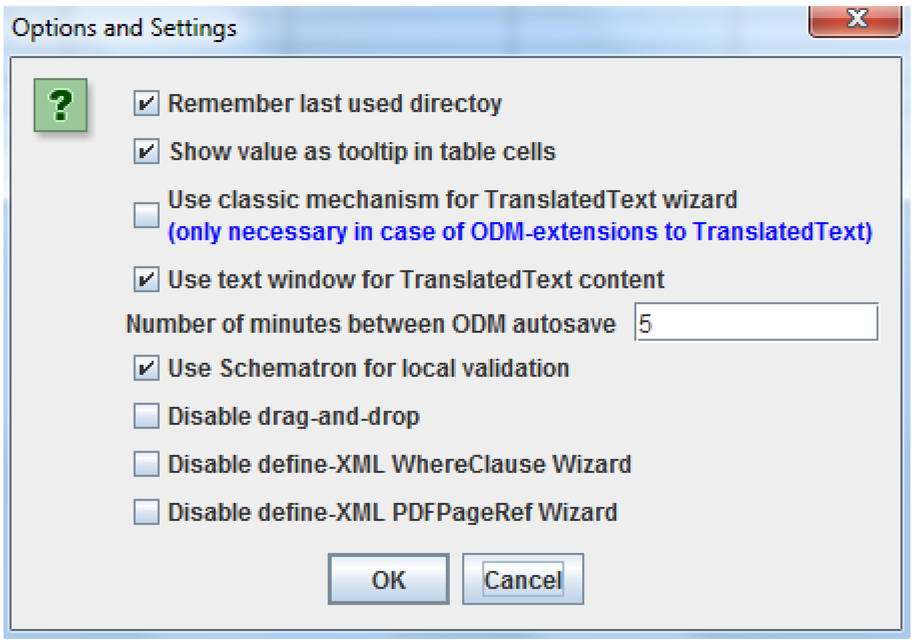
- The following option is a very important one: the software will save the full study or define.xml design to an define.xml or ODM file each 5 minutes in the directory "autosave". This allows you to pick up an earlier version of your design. The files in this directory use a naming convention for allowing the user to see when it has been created. For example:



Name	Änderungsdatum	Typ
ODM_2016_1_30_11-43-42.xml	30.01.2016 11:43	XML-D
ODM_2016_1_30_11-49-43.xml	30.01.2016 11:49	XML-D
ODM_2016_1_30_11-54-43.xml	30.01.2016 11:54	XML-D
ODM_2016_1_30_11-55-11.xml	30.01.2016 11:55	XML-D
ODM_2016_1_30_11-56-30.xml	30.01.2016 11:56	XML-D
ODM_2016_1_30_12-1-30.xml	30.01.2016 12:06	XML-D
ODM_2016_1_30_12-6-12.xml	30.01.2016 12:06	XML-D
ODM_2016_1_30_12-9-33.xml	30.01.2016 12:09	XML-D
ODM_2016_1_30_12-14-33.xml	30.01.2016 12:14	XML-D

  The first "autosave" file has been created on the 30th of January 2016 at 11 hours and 43 minutes and 42 seconds (24 hours system is used).
  The periods between an "autosave" can be changed by using the textfield after "Number of minutes between ODM autosave". Only positive numbers can be entered. Setting the value to "0" means that no autosaving is done.
  Please regularly have a look at the "autosave" directory and clean it up.
- The designer uses schematron technology during validation. This is the methodology of choice for validating XML files. The software comes with a set of schematron files for as well ODM as for define.xml. If you want to limit validation to XML-Schema validation and hardcoded rules when doing "local" validation (see further), uncheck the checkbox "Use schematron for local validation".
- Most people like to use "drag and drop". There are however some users that don't. If you are one of this users, check the checkbox "disable drag-and-drop". The latter mechanism is then replaced by a "select and click" mechanism.
- The checkbox "Show OIDs in annotated CRFs" allows you to add the OID in an annotated CRF. It is only supported for the use case of generating CRFs in an ODM study design.
- The last two checkboxes enable you to disable the wizards for the define.xml "WhereClause" and "PDFPageRef". If you disable these, you will need to enter some information by hand. This requires a very good knowledge of define.xml. Use with care!

If you have purchased the standalone "define.xml Designer" software, this dialog looks like:


Options and Settings

- [✓] Remember last used directoy
- [✓] Show value as tooltip in table cells
- [ ] Use classic mechanism for TranslatedText wizard (only necessary in case of ODM-extensions to TranslatedText)
- [✓] Use text window for TranslatedText content

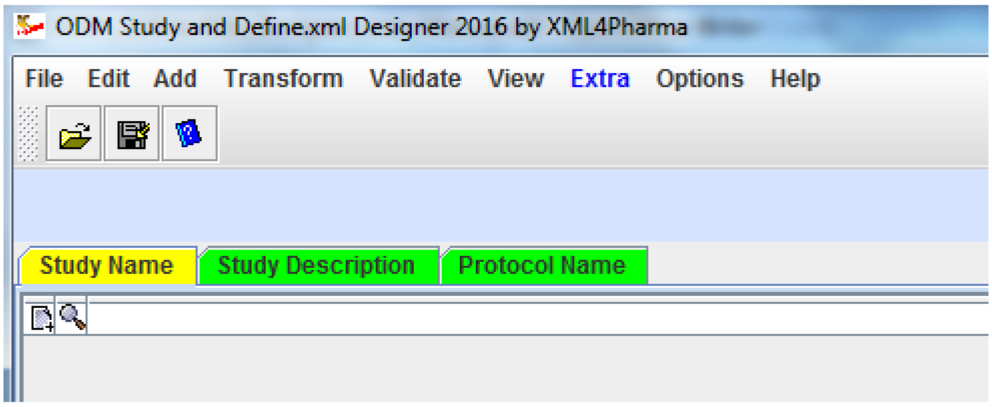
Number of minutes between ODM autosave: 5

- [✓] Use Schematron for local validation
- [ ] Disable drag-and-drop
- [ ] Disable define-XML WhereClause Wizard
- [ ] Disable define-XML PDFPageRef Wizard

with only the options that are relevant for "define.xml only".

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

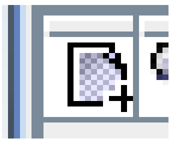
For example, for "Global Study Variables", you will find 3 tabs:



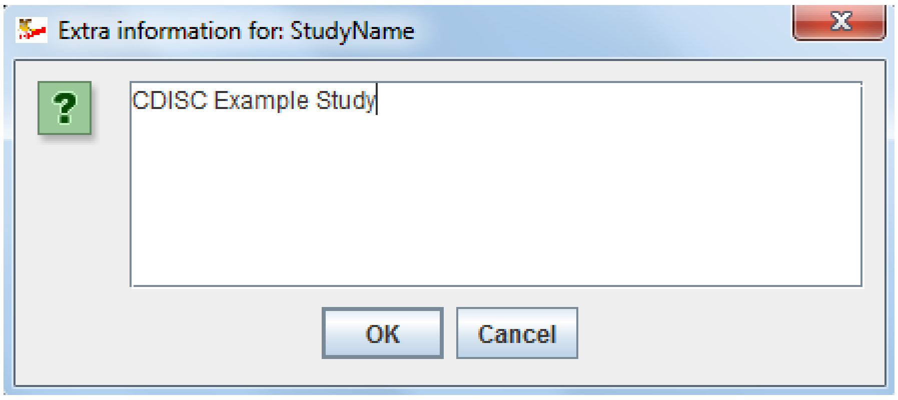
The currently selected tab is always colored yellow, a green tab means that it represents information that must be entered, i.e. is mandatory.

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

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



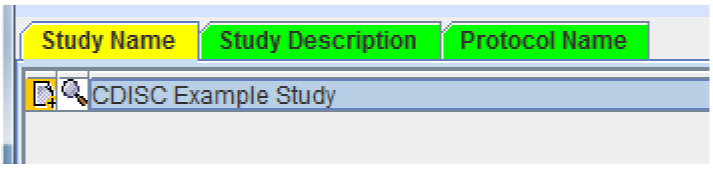
This will result in a new window being displayed, allowing you to enter the study name. For example:



Extra information for: StudyName

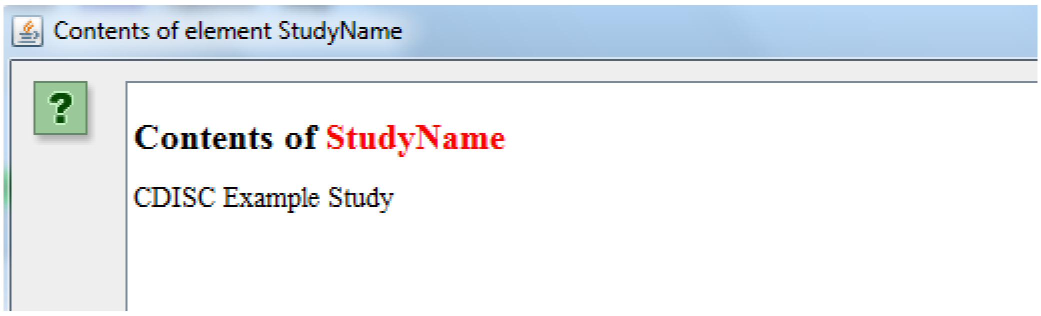
CDISC Example Study

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



CDISC Example Study

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



Contents of element StudyName

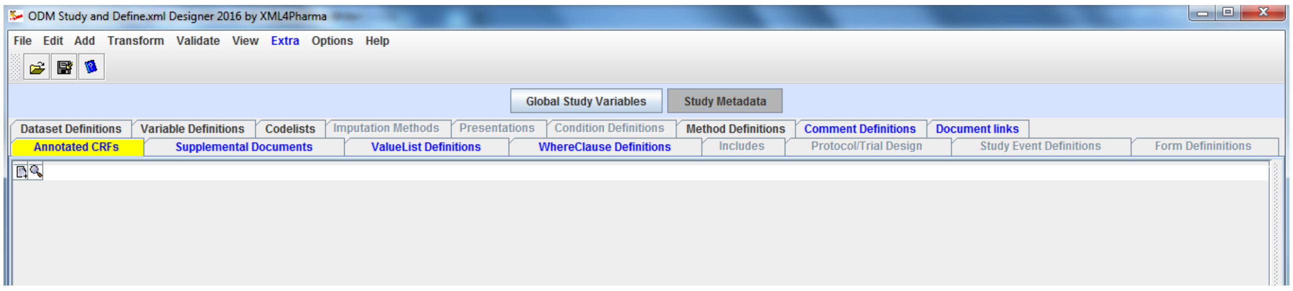
Contents of StudyName

CDISC Example Study

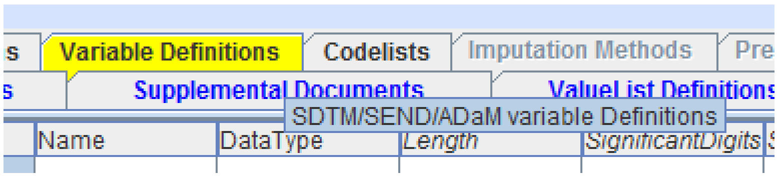
In more complicated cases, such as when the ODM or define.xml element has attributes and/or child elements, much more information will be displayed in this dialog window.

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

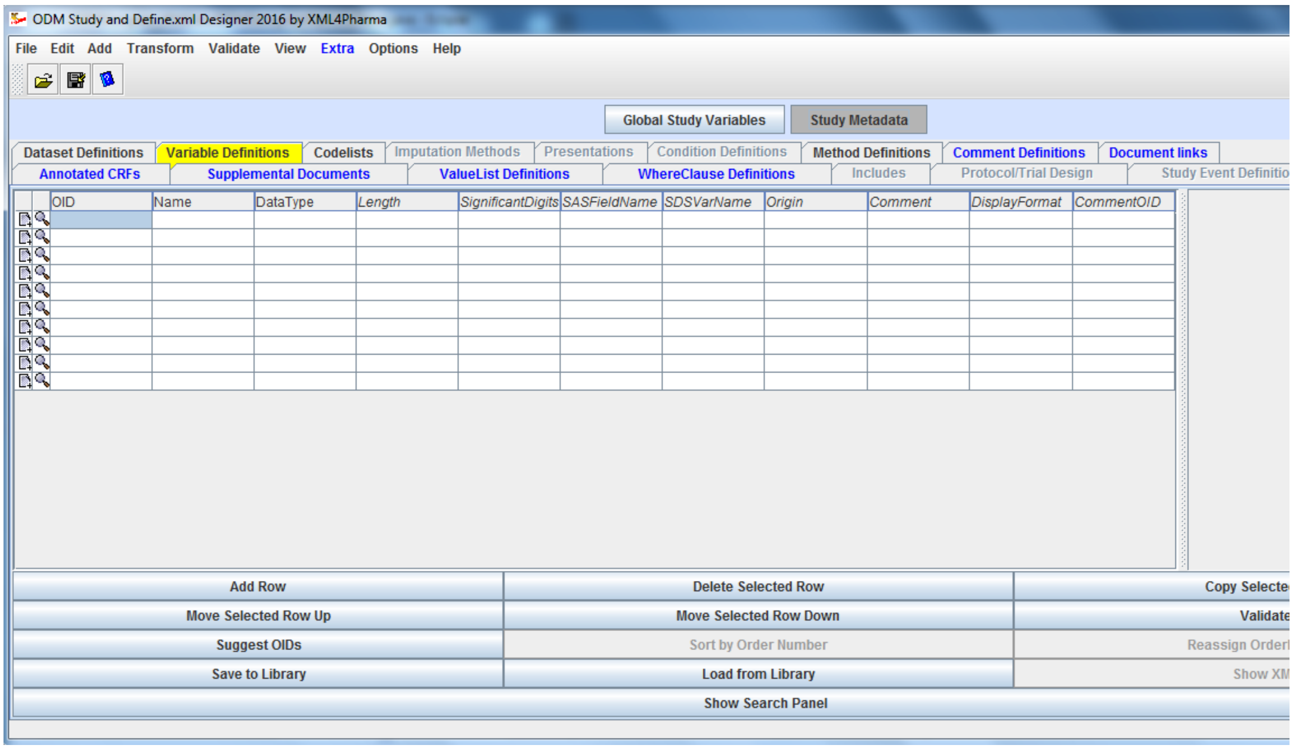
Let us now see how this works for an SDTM or SEND or ADaM variable. These, and all other metadata of the submission can be added/edited after clicking the "Study Metadata" button. This results in:



One sees that some of the tabs are disabled, as we choose for "disable ODM elements not used by define.xml" at startup. To add an SDTM/SEND/ADaM variable, click the tab "Variable Definitions":



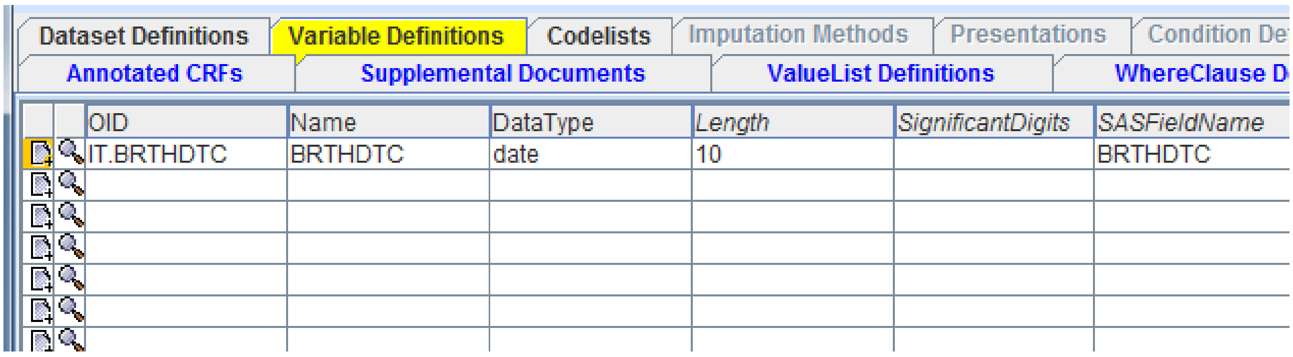
This then displays the following panel:



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

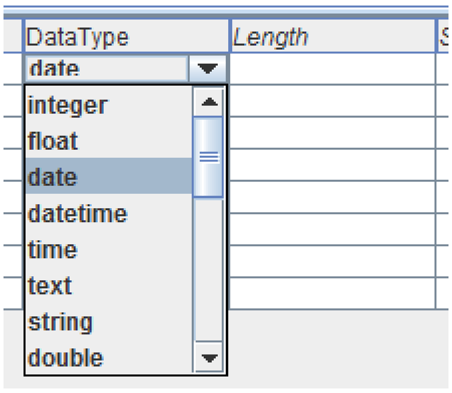
In the table, each row corresponds to a single SDTM/SEND/ADaM variable, and each column to an attribute for it. Adding child element information will be done by clicking the "+" icon on the left of each row, and inspecting all the contents can be done by clicking the "magnifying glass" icon (second from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

For a variable "BRTHDTC" (birthdate) in DM, a variable declaration will typically look like (for define.xml 2.0):

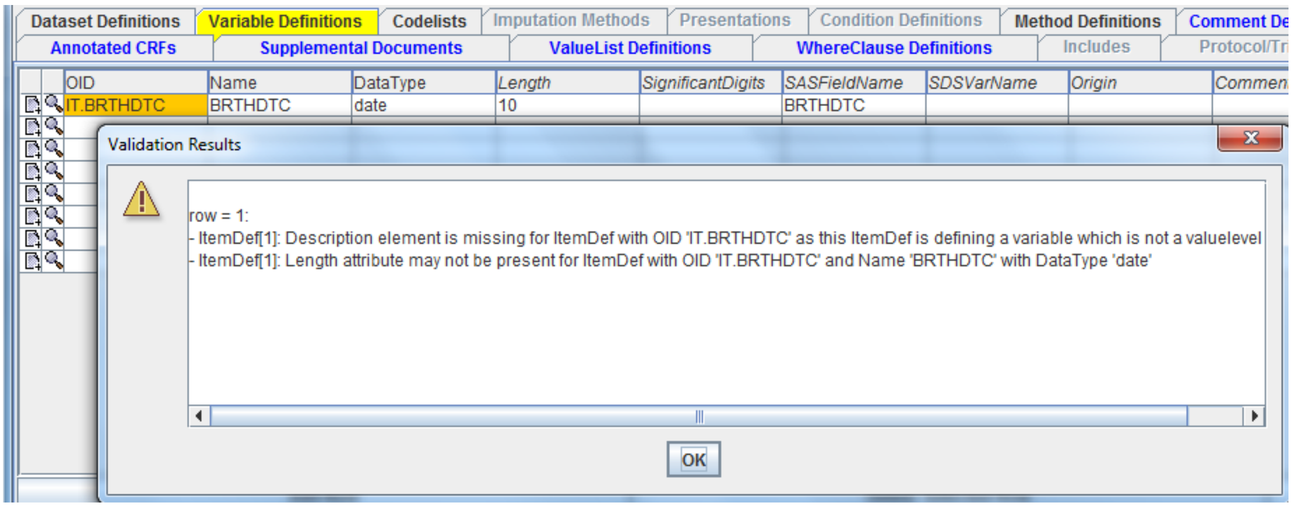


OID	Name	DataType	Length	SignificantDigits	SASFieldName
IT.BRTHDTC	BRTHDTC	date	10		BRTHDTC

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



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



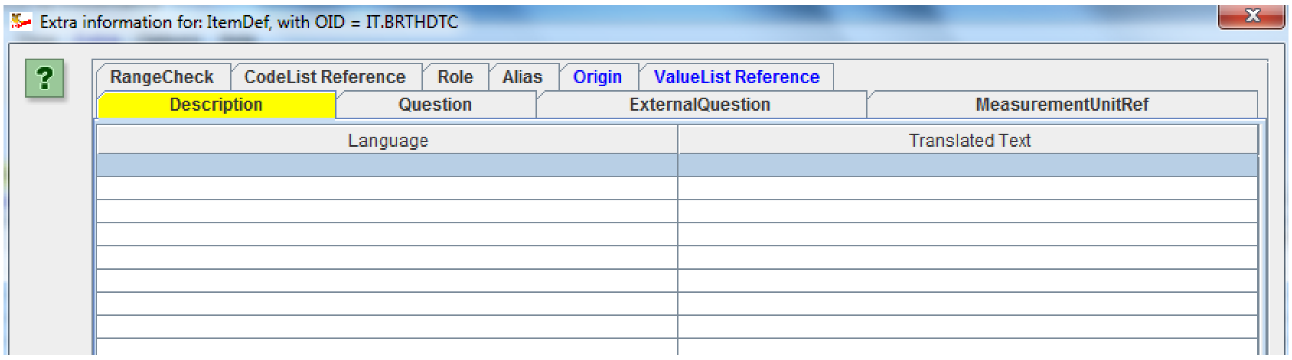
OID	Name	DataType	Length	SignificantDigits	SASFieldName	SDSVarName	Origin	Commen
IT.BRTHDTC	BRTHDTC	date	10		BRTHDTC			

```
row = 1:
- ItemDef[1]: Description element is missing for ItemDef with OID 'IT.BRTHDTC' as this ItemDef is defining a variable which is not a valuelevel
- ItemDef[1]: Length attribute may not be present for ItemDef with OID 'IT.BRTHDTC' and Name 'BRTHDTC' with DataType 'date'
```

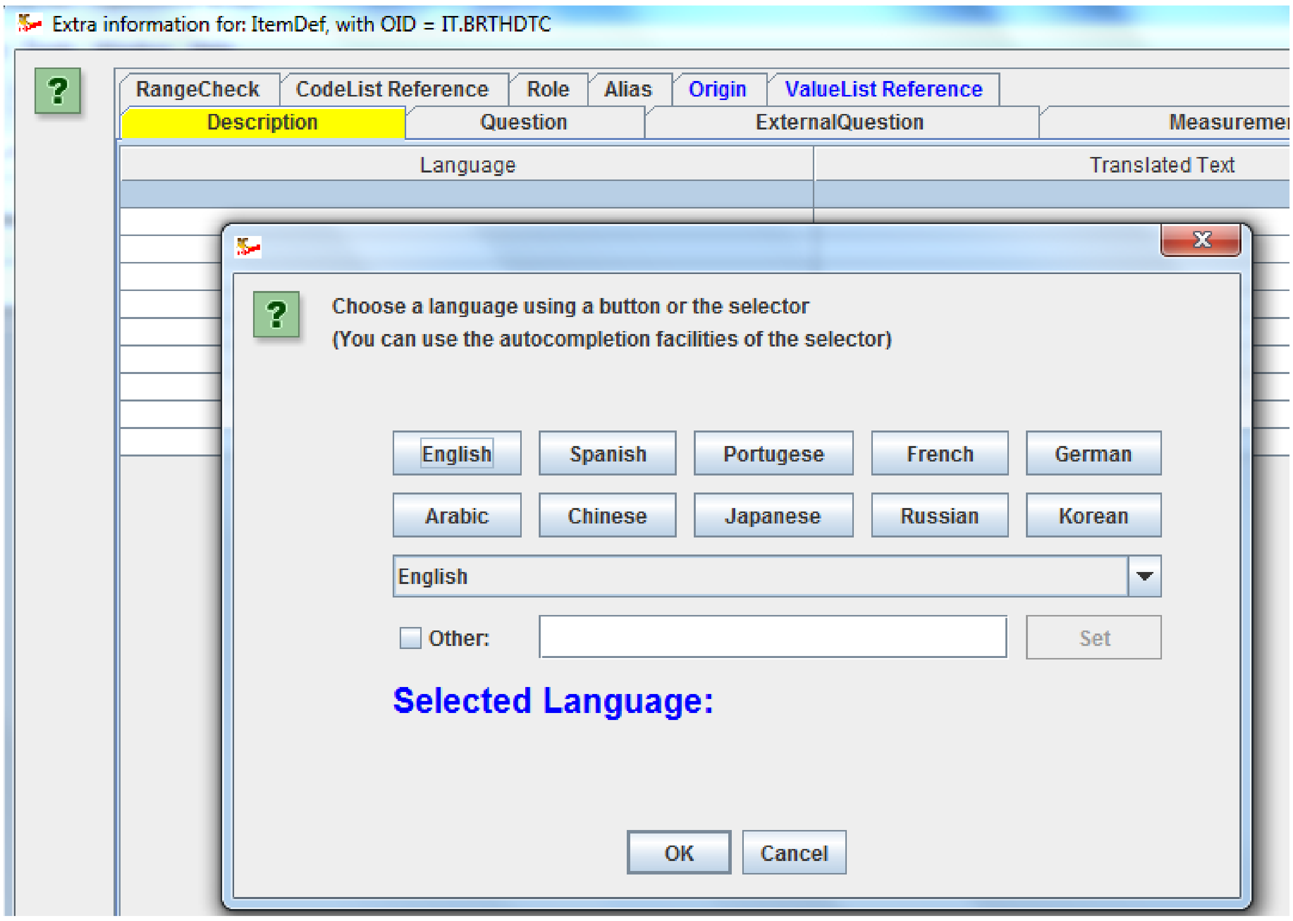
Two violations[3] of the define.xml standard (2.0) have been found:
a) No "Description" child element was found (this was known as "def:Label in define.xml 1.0)
b) The "Length" attribute is not allowed when the DataType has the value "datetime"

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

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

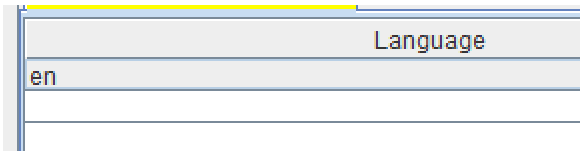

Extra information for: ItemDef, with OID = IT.BRTHDTC

As one sees, this table has several rows, as a description in several languages can be added. For a submission to the FDA, at least an english description must be provided, which must match[4] the "label" from the SDTM-IG or SEND-IG or AdaM-IG which is "Date/Time of Birth" in this case. We first want to add the language. Clicking in the first "Language" cell pops up a dialog:


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

Choose a language using a button or the selector
(You can use the autocompletion facilities of the selector)

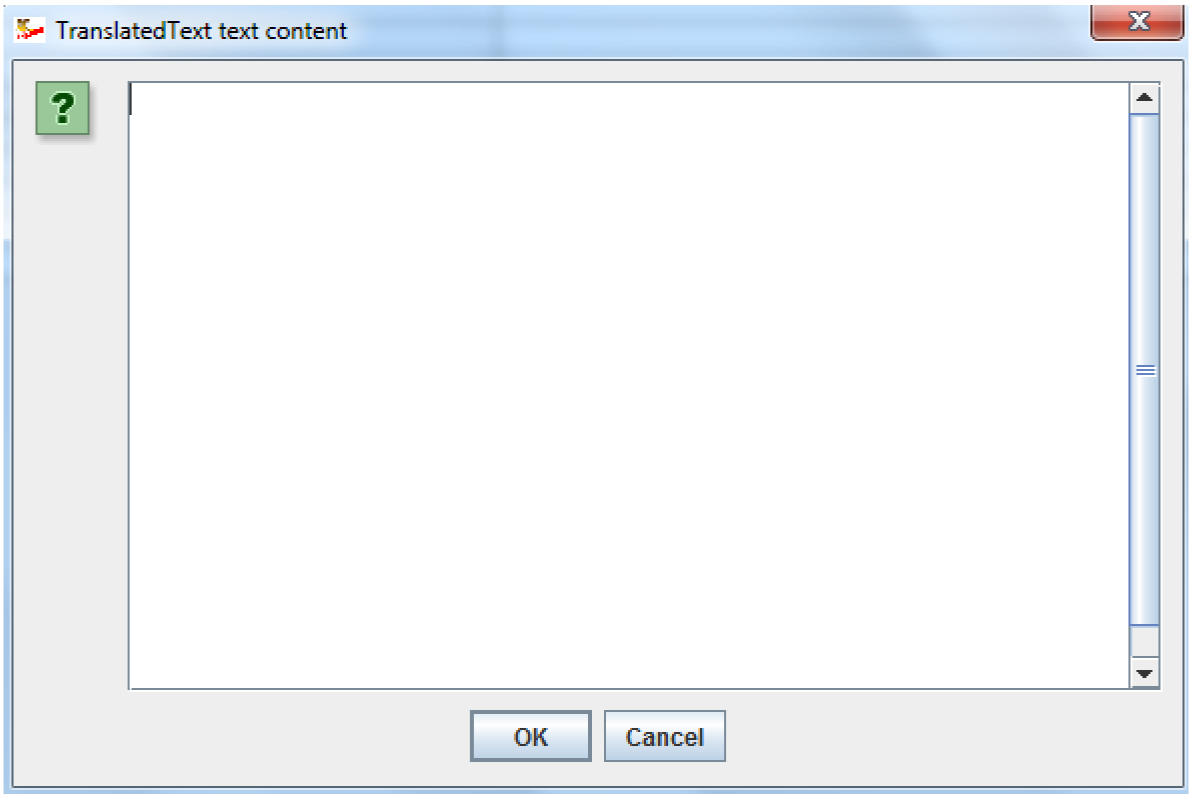
from we select "English" and then click OK[5]. Resulting in:



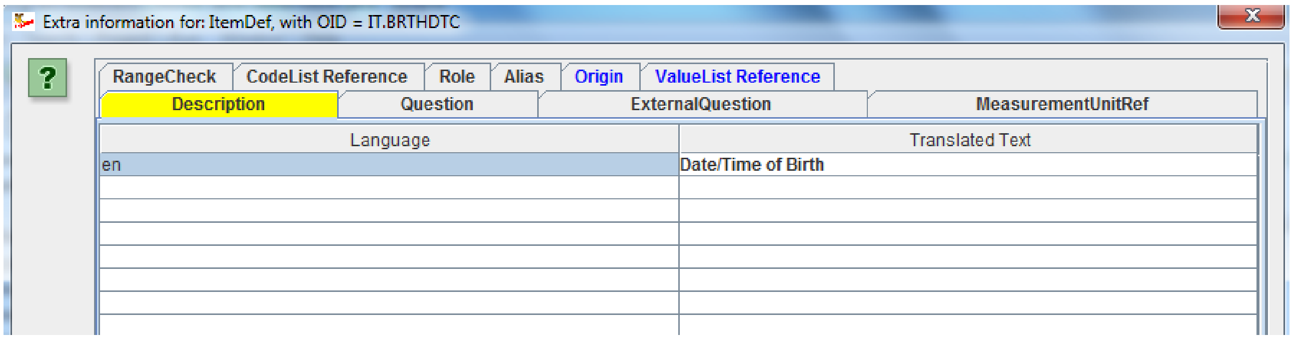
[4]In future, we intent an option that automatically looks up the variable label for a given variable name and SDS version, either locally (from a file that comes with the software), by a RESTful web service, or by a query to CDISC SHARE.

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

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

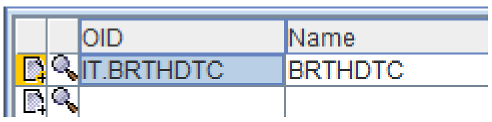


where we add the "SDTM label", in this case "Date/Time of Birth". After clicking "OK" leading to:



Language	Translated Text
en	Date/Time of Birth

We could now add the description in other languages[6], but this is usually not necessary in the case of submissions to the FDA.
After clicking the "OK" button:

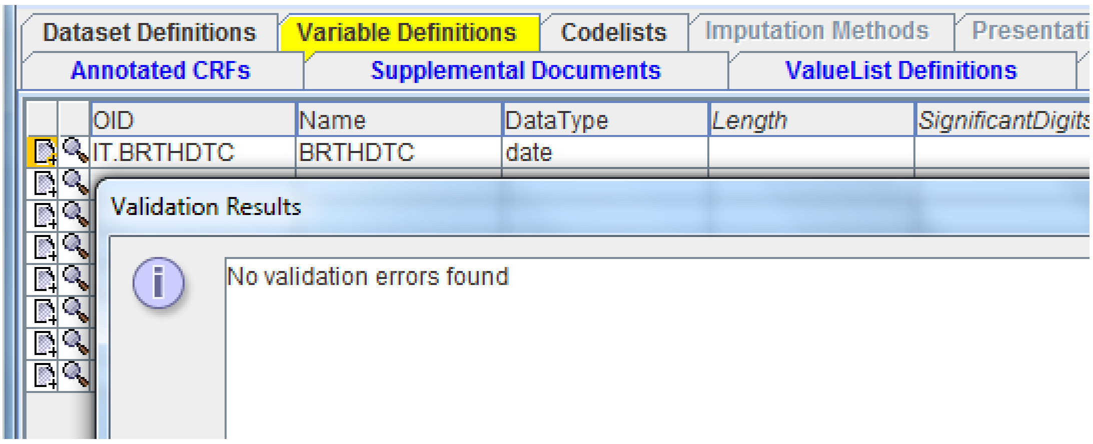


		OID	Name
		IT.BRTHDTC	BRTHDTC

we see that the "+" icon is now with a yellow background color, meaning that additional information has been added. The background color may also become red after validation, meaning that the additionally added information does not conform to the define.xml standard.

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

Clicking the "validate" button again leads to:



OID	Name	DataType	Length	SignificantDigits
IT.BRTHDTC	BRTHDTC	date		

Validation Results

No validation errors found

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

So, the essentials again:

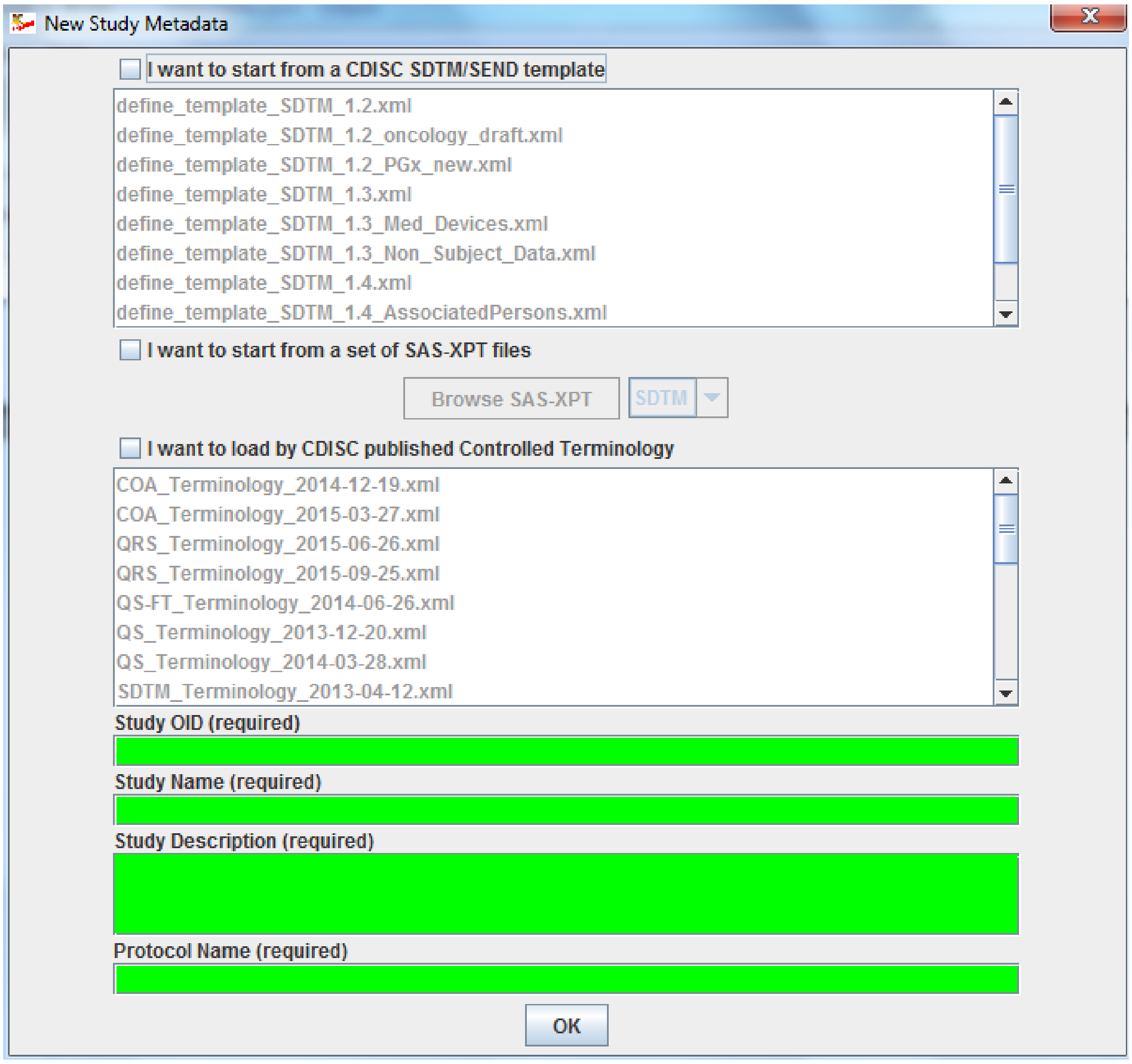
- Variable attributes are added by clicking the appropriate cell. In case the value is enumerated, a "selection list" will be displayed. In some cases, a wizard will pop up.
- Additional information that is stored in child elements are added by clicking the "+" icon, selecting a tab, and adding the information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

Creating a new define.xml

Everything has a beginning...

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

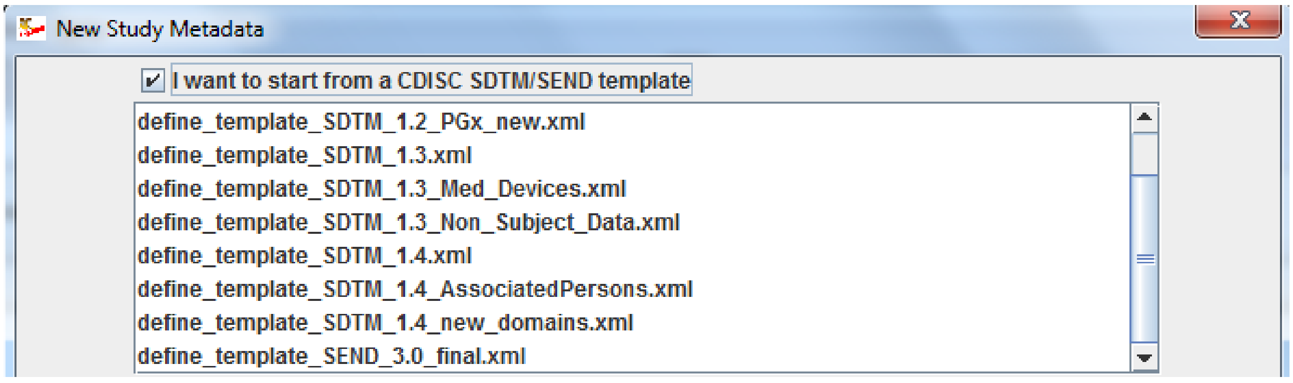
In order to start the development of a new define.xml either from an SDTM or SEND template, or from a set of existing SAS-XPT files with SDTM or SEND data, use the menu "File - New define.xml". The following dialog is presented:


New Study Metadata
I want to start from a CDISC SDTM/SEND template
define_template_SDTM_1.2.xml
define_template_SDTM_1.2_oncology_draft.xml
define_template_SDTM_1.2_PGx_new.xml
define_template_SDTM_1.3.xml
define_template_SDTM_1.3_Med_Devices.xml
define_template_SDTM_1.3_Non_Subject_Data.xml
define_template_SDTM_1.4.xml
define_template_SDTM_1.4_AssociatedPersons.xml
I want to start from a set of SAS-XPT files
Browse SAS-XPT
SDTM
I want to load by CDISC published Controlled Terminology
COA_Terminology_2014-12-19.xml
COA_Terminology_2015-03-27.xml
QRS_Terminology_2015-06-26.xml
QRS_Terminology_2015-09-25.xml
QS-FT_Terminology_2014-06-26.xml
QS_Terminology_2013-12-20.xml
QS_Terminology_2014-03-28.xml
SDTM_Terminology_2013-04-12.xml
Study OID (required)
Study Name (required)
Study Description (required)
Protocol Name (required)
OK

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

Creating a define.xml starting from an SDTM template

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

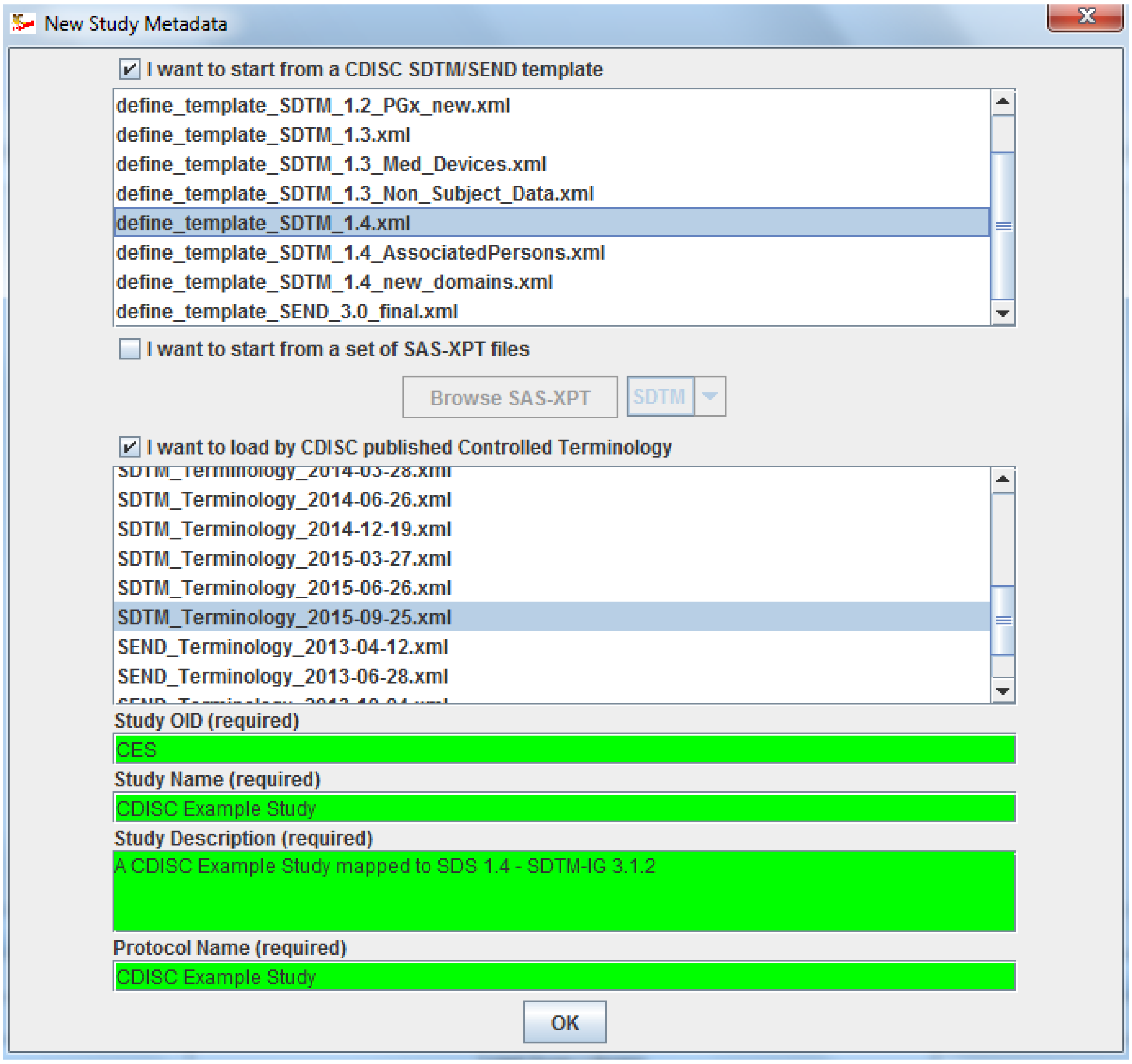


define_template_SDTM_1.2_PGx_new.xml
define_template_SDTM_1.3.xml
define_template_SDTM_1.3_Med_Devices.xml
define_template_SDTM_1.3_Non_Subject_Data.xml
define_template_SDTM_1.4.xml
define_template_SDTM_1.4_AssociatedPersons.xml
define_template_SDTM_1.4_new_domains.xml
define_template_SEND_3.0_final.xml

One can then select a template from the list. It contains all necessary information from the SDTM-IG or SEND-IG. One can later than still merge with another template. For example, if one has selected "define_template_SDTM_1.4.xml, then one can later still merge with domains from the "define_template_SDTM_1.4_AssociatedPersons" or even with templates for newly developed SDTM domains published by CDISC.
All these templates come as files with the software, and the content of the directory where they reside is inspected by the software. This also means that when SDTM 1.5 is published by CDISC, we will deliver the template file or files for them, and adding them to the directory will suffice to be able to use them - no software update will be necessary.

Suppose we would like to use the latest SDTM version, so we select "define_template_SDTM_1.4.xml". It is then also extremely useful to already add the latest by CDISC published controlled terminology[7]. In order to do so, check the checkbox "I want to load by CDISC Controlled Terminology" and then select the latest version from the list. Also here, when new controlled terminology is published by CDISC, we will provide it as a file, and adding it to the directory where the controlled terminology files reside will be sufficient to appear it in the list in the software.
Additional CDISC controlled terminology can also be loaded and merged later.

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



New Study Metadata

[✓] I want to start from a CDISC SDTM/SEND template

define_template_SDTM_1.2_PGx_new.xml
define_template_SDTM_1.3.xml
define_template_SDTM_1.3_Med_Devices.xml
define_template_SDTM_1.3_Non_Subject_Data.xml
define_template_SDTM_1.4.xml
define_template_SDTM_1.4_AssociatedPersons.xml
define_template_SDTM_1.4_new_domains.xml
define_template_SEND_3.0_final.xml

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

Browse SAS-XPT | SDTM

[✓] I want to load by CDISC published Controlled Terminology

SDTM_Terminology_2014-03-28.xml
SDTM_Terminology_2014-06-26.xml
SDTM_Terminology_2014-12-19.xml
SDTM_Terminology_2015-03-27.xml
SDTM_Terminology_2015-06-26.xml
SDTM_Terminology_2015-09-25.xml
SEND_Terminology_2013-04-12.xml
SEND_Terminology_2013-06-28.xml

Study OID (required): CES

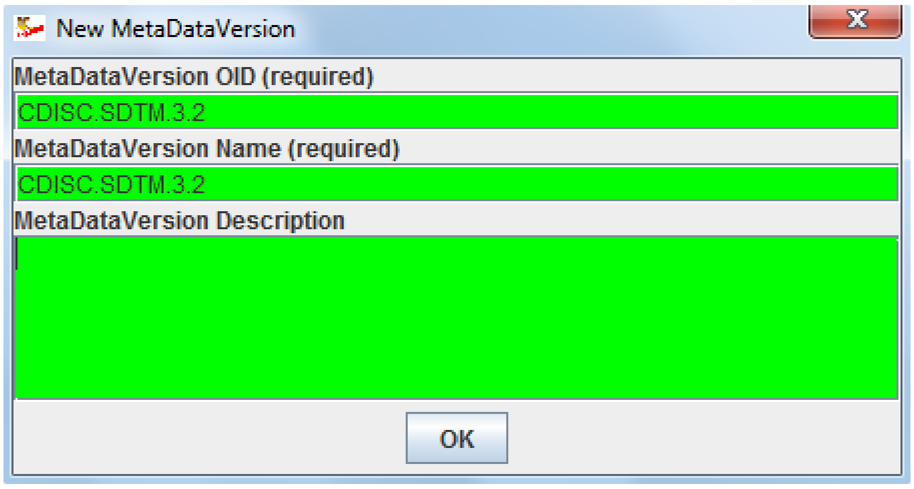
Study Name (required): CDISC Example Study

Study Description (required): A CDISC Example Study mapped to SDS 1.4 - SDTM-IG 3.1.2

Protocol Name (required): CDISC Example Study

OK

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



New MetaDataVersion

MetaDataVersion OID (required): CDISC.SDTM.3.2

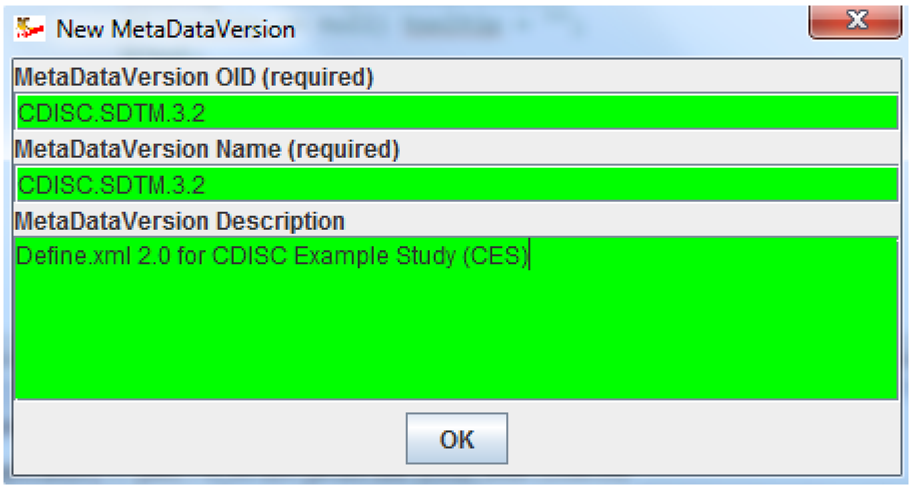
MetaDataVersion Name (required): CDISC.SDTM.3.2

MetaDataVersion Description:

OK

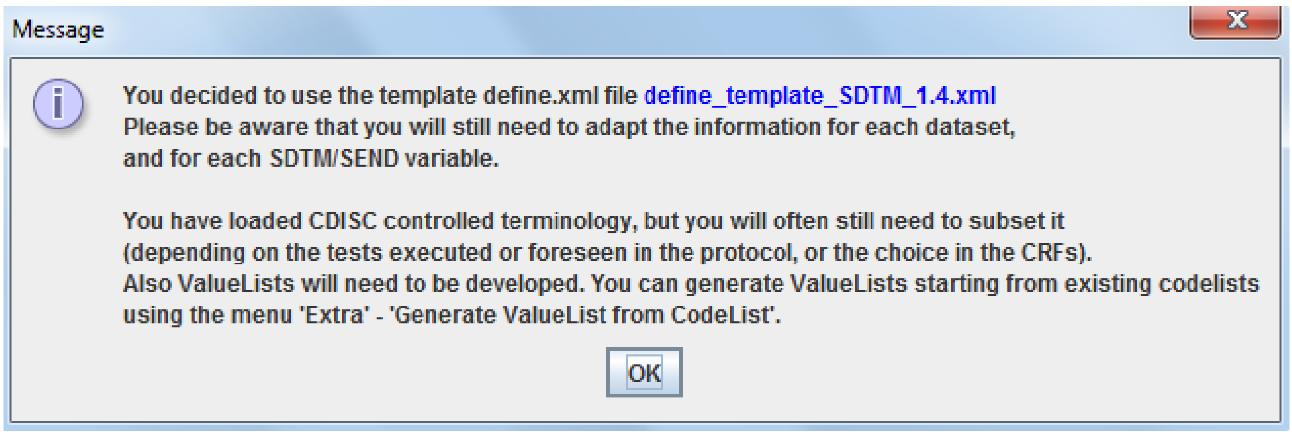
A proposal is made for the "MetaDataVersion OID" and the "MetaDataVersion Name". These are arbitrary - they are currently not really used as only one "MetaDataVersion" is currently allowed in a define.xml[8].

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



MetaDataVersion OID (required)
CDISC.SDTM.3.2
MetaDataVersion Name (required)
CDISC.SDTM.3.2
MetaDataVersion Description
Define.xml 2.0 for CDISC Example Study (CES)

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



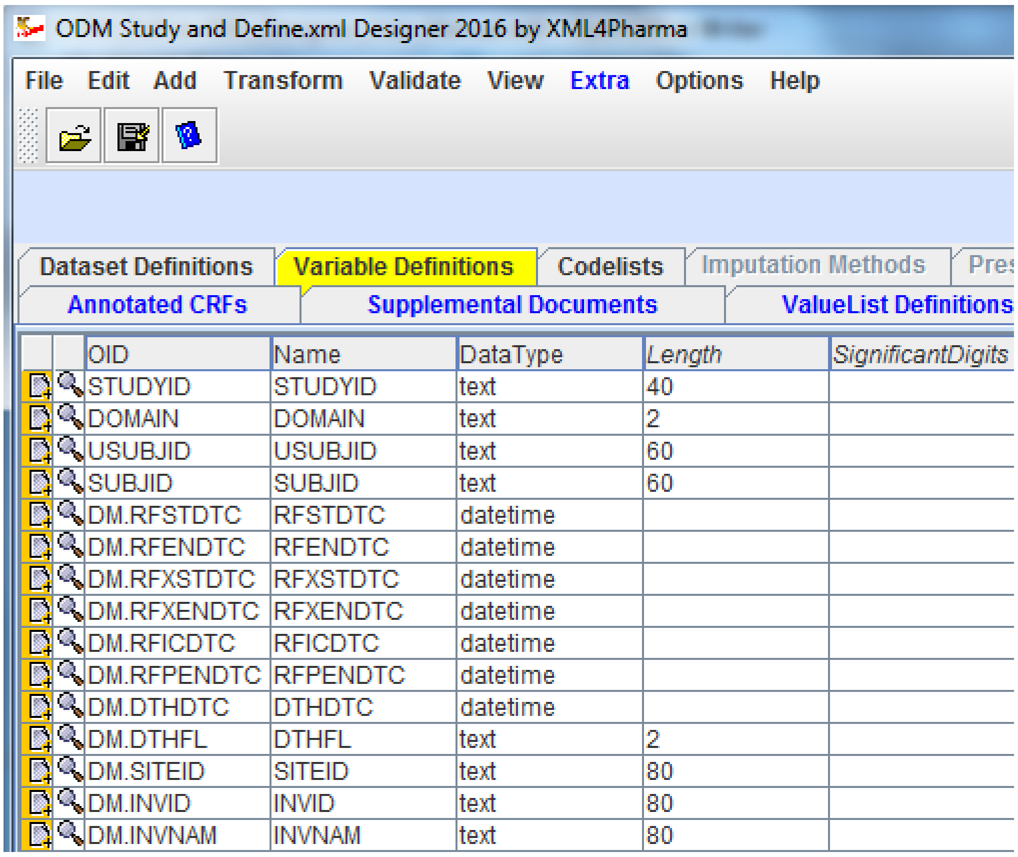
You decided to use the template define.xml file define_template_SDTM_1.4.xml
Please be aware that you will still need to adapt the information for each dataset,
and for each SDTM/SEND variable.

You have loaded CDISC controlled terminology, but you will often still need to subset it
(depending on the tests executed or foreseen in the protocol, or the choice in the CRFs).
Also ValueLists will need to be developed. You can generate ValueLists starting from existing codelists
using the menu 'Extra' - 'Generate ValueList from CodeList'.

Remark that the whole CDISC controlled terminology has been loaded, and that you will need to subset many of the codelists. For example, the "LBTESTCD" contains over different 1,300 codes for lab tests, but you will probably have considerably less in your study. We will later learn how to subset a codelist.

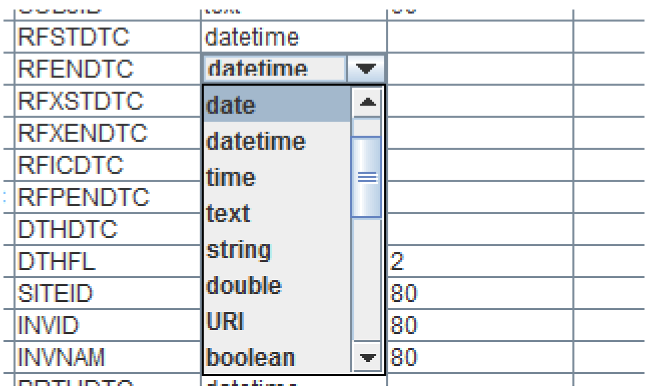
Also, you will have to develop the necessary "value-lists" (value-level metadata). The software has a wizard for this, allowing you to generate a "ValueList" from an existing codelist.
After clicking OK, and navigating to the "Variable Definitions", one gets:

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



OID	Name	DataType	Length	SignificantDigits
STUDYID	STUDYID	text	40	
DOMAIN	DOMAIN	text	2	
USUBJID	USUBJID	text	60	
SUBJID	SUBJID	text	60	
DM.RFSTDTC	RFSTDTC	datetime		
DM.RFENDTC	RFENDTC	datetime		
DM.RFXSTDTC	RFXSTDTC	datetime		
DM.RFXENDTC	RFXENDTC	datetime		
DM.RFICDTC	RFICDTC	datetime		
DM.RFPENDTC	RFPENDTC	datetime		
DM.DTHDTC	DTHDTC	datetime		
DM.DTHFL	DTHFL	text	2	
DM.SITEID	SITEID	text	80	
DM.INVID	INVID	text	80	
DM.INVNAM	INVNAM	text	80	

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



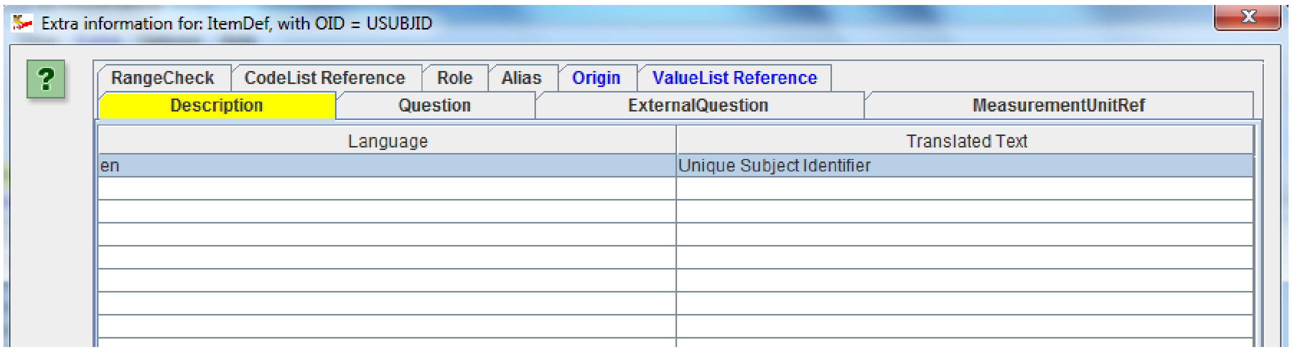
RFSTDTC	datetime	
RFENDTC	datetime	
RFXSTDTC		
RFXENDTC		
RFICDTC		
RFPENDTC		
DTHDTC		
DTHFL		2
SITEID		80
INVID		80
INVNAM		80

You will also need to adapt the value for "Length" to the maximum length the data for that variable can have in your SAS-XPT files. So if in your study "USUBJID" never has more than 20 characters, you should change the value of "60" into "20", <u>and</u> take care that in your SAS-XPT file the field length is exactly 20[9].

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

[9]This has to do with the fixed field and record length in SAS-XPT. Once the regulatory will start accepting Dataset-XML files instead of ancient SAS-XPT, this requirement will probably be dropped.

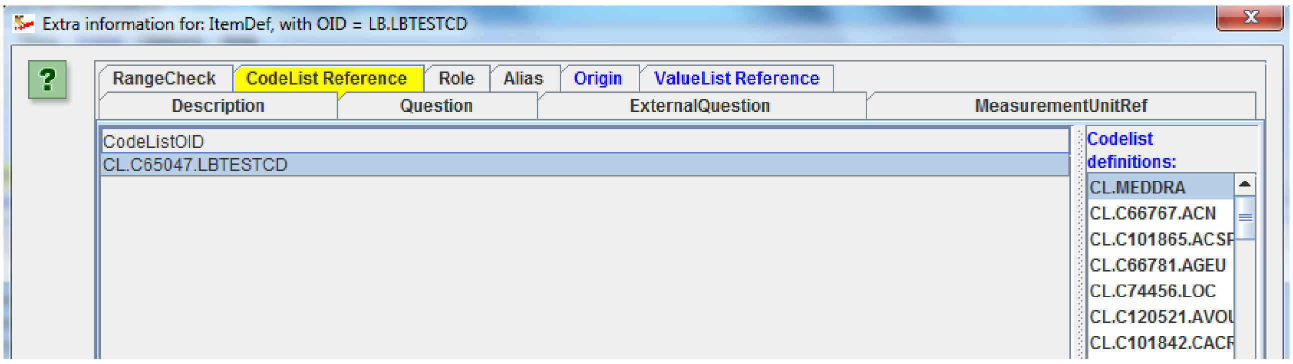
displayed:


Extra information for: ItemDef, with OID = USUBJID

Language	Translated Text
en	Unique Subject Identifier

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

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


Extra information for: ItemDef, with OID = LB.LBTESTCD

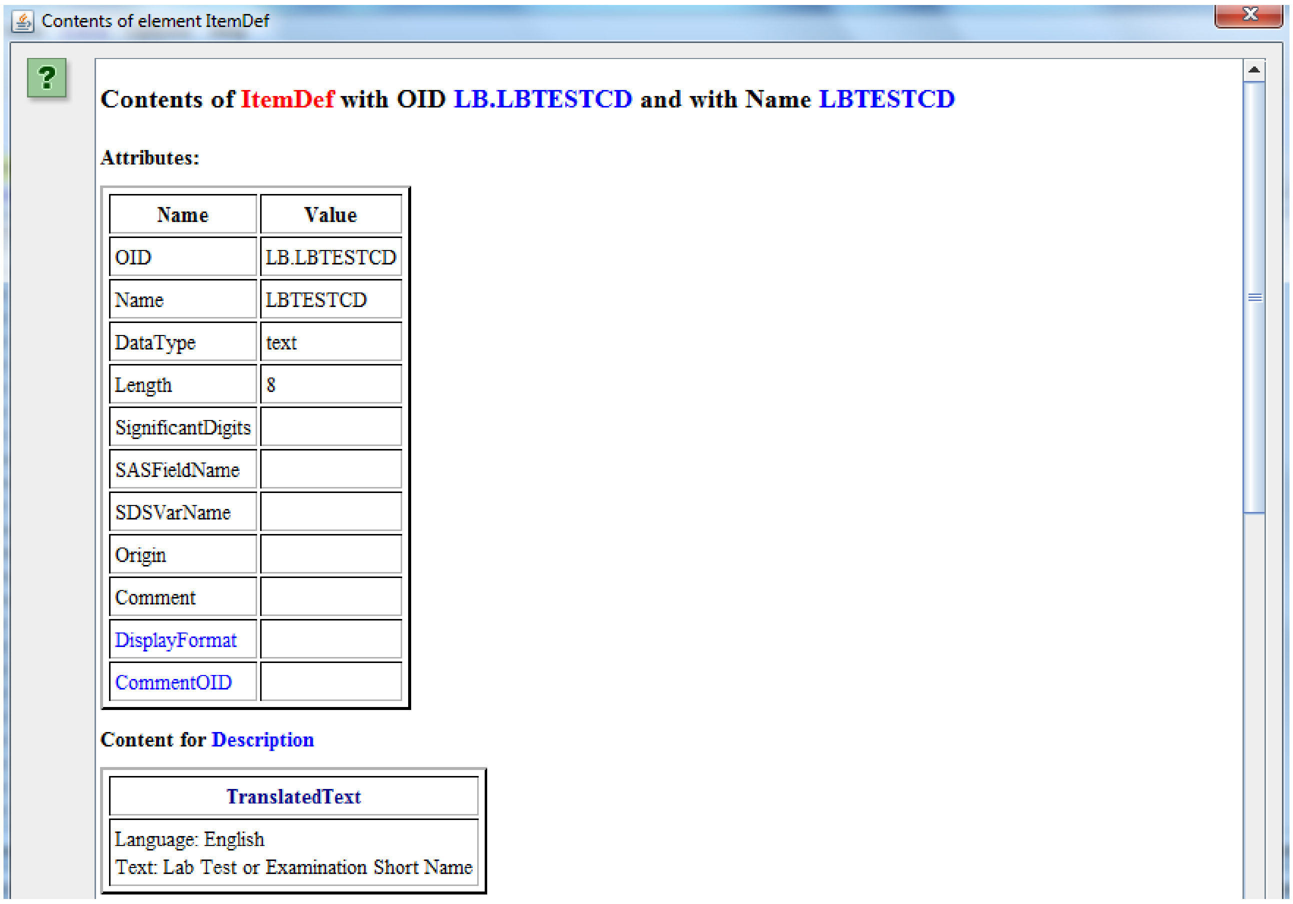
CodeListOID
CL.C65047.LBTESTCD

Codelist definitions:
CL.MEDDRA
CL.C66767.ACN
CL.C101865.ACSP
CL.C66781.AGEU
CL.C74456.LOC
CL.C120521.AVOU
CL.C101842.CACR

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

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

In the main window, clicking the "magnifying glass" icon on LBTESTCD leads to:


Contents of element ItemDef

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

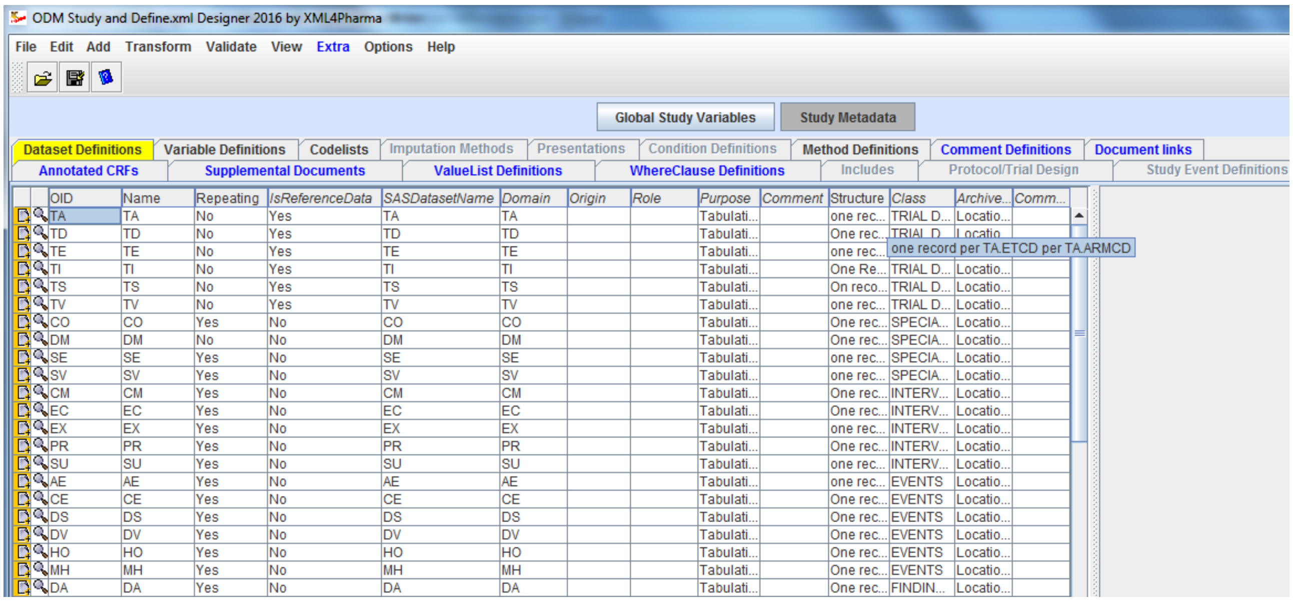
Attributes:

Name	Value
OID	LB.LBTESTCD
Name	LBTESTCD
DataType	text
Length	8
SignificantDigits	
SASFieldName	
SDSVarName	
Origin	
Comment	
DisplayFormat	
CommentOID	

Content for Description

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

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


ODM Study and Define.xml Designer 2016 by XML4Pharma

File Edit Add Transform Validate View Extra Options Help

Global Study Variables | Study Metadata

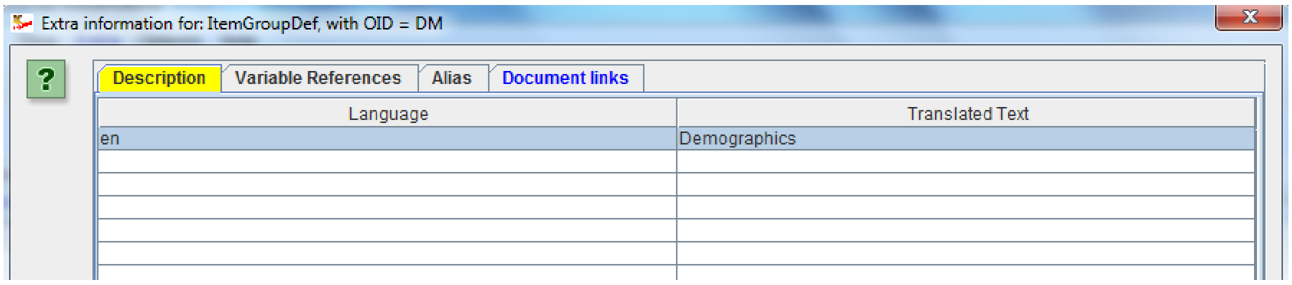
Dataset Definitions | Variable Definitions | Codelists | Imputation Methods | Presentations | Condition Definitions | Method Definitions | Comment Definitions | Document links

Annotated CRFs | Supplemental Documents | ValueList Definitions | WhereClause Definitions | Includes | Protocol/Trial Design | Study Event Definitions

OID	Name	Repeating	IsReferenceData	SASDatasetName	Domain	Origin	Role	Purpose	Comment	Structure	Class	Archive...	Comm...
TA	TA	No	Yes	TA	TA			Tabulati...		one rec...	TRIAL D...	Locatio...	
TD	TD	No	Yes	TD	TD			Tabulati...		One rec...	TRIAL D	Locatio	
TE	TE	No	Yes	TE	TE			Tabulati...		one rec...			
TI	TI	No	Yes	TI	TI			Tabulati...		One Re...	TRIAL D...	Locatio...	
TS	TS	No	Yes	TS	TS			Tabulati...		On reco...	TRIAL D...	Locatio...	
TV	TV	No	Yes	TV	TV			Tabulati...		one rec...	TRIAL D...	Locatio...	
CO	CO	Yes	No	CO	CO			Tabulati...		One rec...	SPECIA...	Locatio...	
DM	DM	No	No	DM	DM			Tabulati...		One rec...	SPECIA...	Locatio...	
SE	SE	Yes	No	SE	SE			Tabulati...		one rec...	SPECIA...	Locatio...	
SV	SV	Yes	No	SV	SV			Tabulati...		one rec...	SPECIA...	Locatio...	
CM	CM	Yes	No	CM	CM			Tabulati...		One rec...	INTERV...	Locatio...	
EC	EC	Yes	No	EC	EC			Tabulati...		One rec...	INTERV...	Locatio...	
EX	EX	Yes	No	EX	EX			Tabulati...		one rec...	INTERV...	Locatio...	
PR	PR	Yes	No	PR	PR			Tabulati...		One rec...	INTERV...	Locatio...	
SU	SU	Yes	No	SU	SU			Tabulati...		one rec...	INTERV...	Locatio...	
AE	AE	Yes	No	AE	AE			Tabulati...		one rec...	EVENTS	Locatio...	
CE	CE	Yes	No	CE	CE			Tabulati...		One rec...	EVENTS	Locatio...	
DS	DS	Yes	No	DS	DS			Tabulati...		One rec...	EVENTS	Locatio...	
DV	DV	Yes	No	DV	DV			Tabulati...		One rec...	EVENTS	Locatio...	
HO	HO	Yes	No	HO	HO			Tabulati...		One rec...	EVENTS	Locatio...	
MH	MH	Yes	No	MH	MH			Tabulati...		One rec...	EVENTS	Locatio...	
DA	DA	Yes	No	DA	DA			Tabulati...		One rec...	FINDIN...	Locatio...	

one record per TA.ETCD per TA.ARMCD

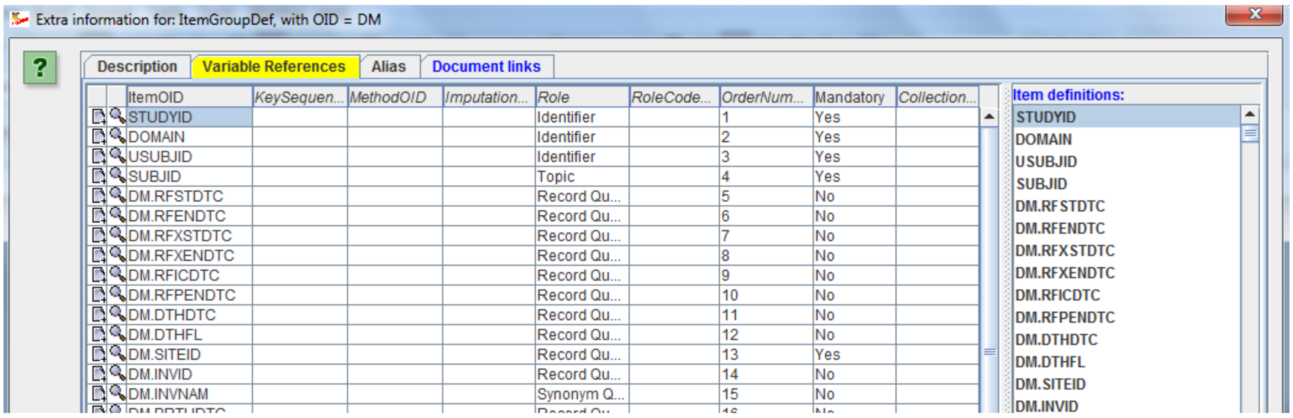
with one row per domain.
Clicking on the "+" icon e.g. For "DM" opens a new window with:


Extra information for: ItemGroupDef, with OID = DM

Description | Variable References | Alias | Document links

Language	Translated Text
en	Demographics

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


Extra information for: ItemGroupDef, with OID = DM

Description | Variable References | Alias | Document links

ItemOID	KeySequen...	MethodOID	Imputation...	Role	RoleCode...	OrderNum...	Mandatory	Collection...
STUDYID				Identifier		1	Yes	
DOMAIN				Identifier		2	Yes	
USUBJID				Identifier		3	Yes	
SUBJID				Topic		4	Yes	
DM.RFSTDTC				Record Qu...		5	No	
DM.RFENDTC				Record Qu...		6	No	
DM.RFXSTDTC				Record Qu...		7	No	
DM.RFXENDTC				Record Qu...		8	No	
DM.RFICDTC				Record Qu...		9	No	
DM.RFPENDTC				Record Qu...		10	No	
DM.DTHDTC				Record Qu...		11	No	
DM.DTHFL				Record Qu...		12	No	
DM.SITEID				Record Qu...		13	Yes	
DM.INVID				Record Qu...		14	No	
DM.INVNAM				Synonym Q...		15	No	

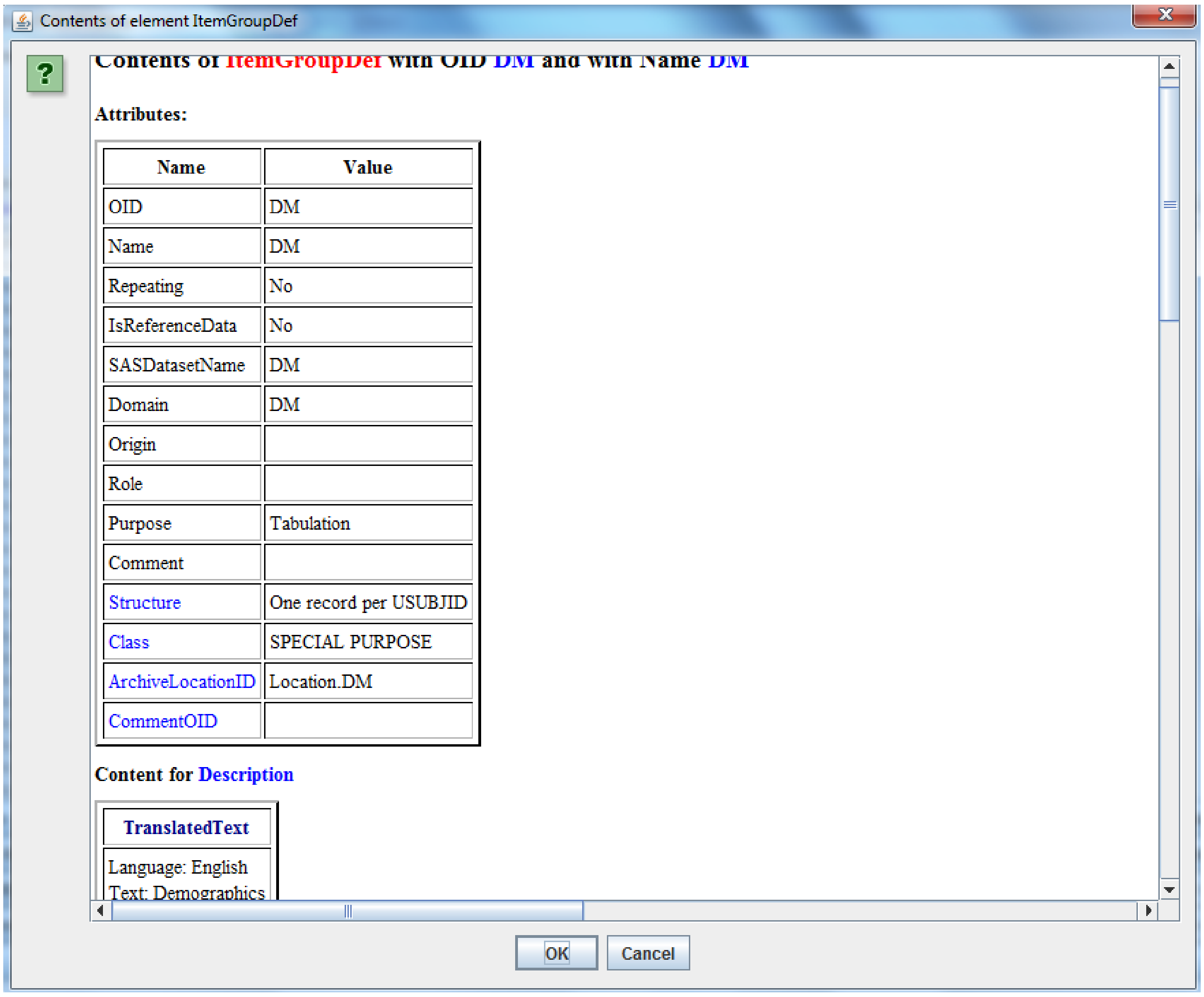
Item definitions:
STUDYID
DOMAIN
USUBJID
SUBJID
DM.RFSTDTC
DM.RFENDTC
DM.RFXSTDTC
DM.RFXENDTC
DM.RFICDTC
DM.RFPENDTC
DM.DTHDTC
DM.DTHFL
DM.SITEID
DM.INVID

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

One also sees that the "Role" attribute is added automatically. This is not absolutely necessary anymore in define.xml 2.0 when it is a standard SDTM or SEND variable. Also the "Mandatory" field has been filled. Its value is "Yes" for the case that the SDTM variable is "required", and to "No" when the SDTM variable is "expected" or "permissible"[10].

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

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



Contents of element ItemGroupDef

Contents of ItemGroupDef with OID DM and with Name DM

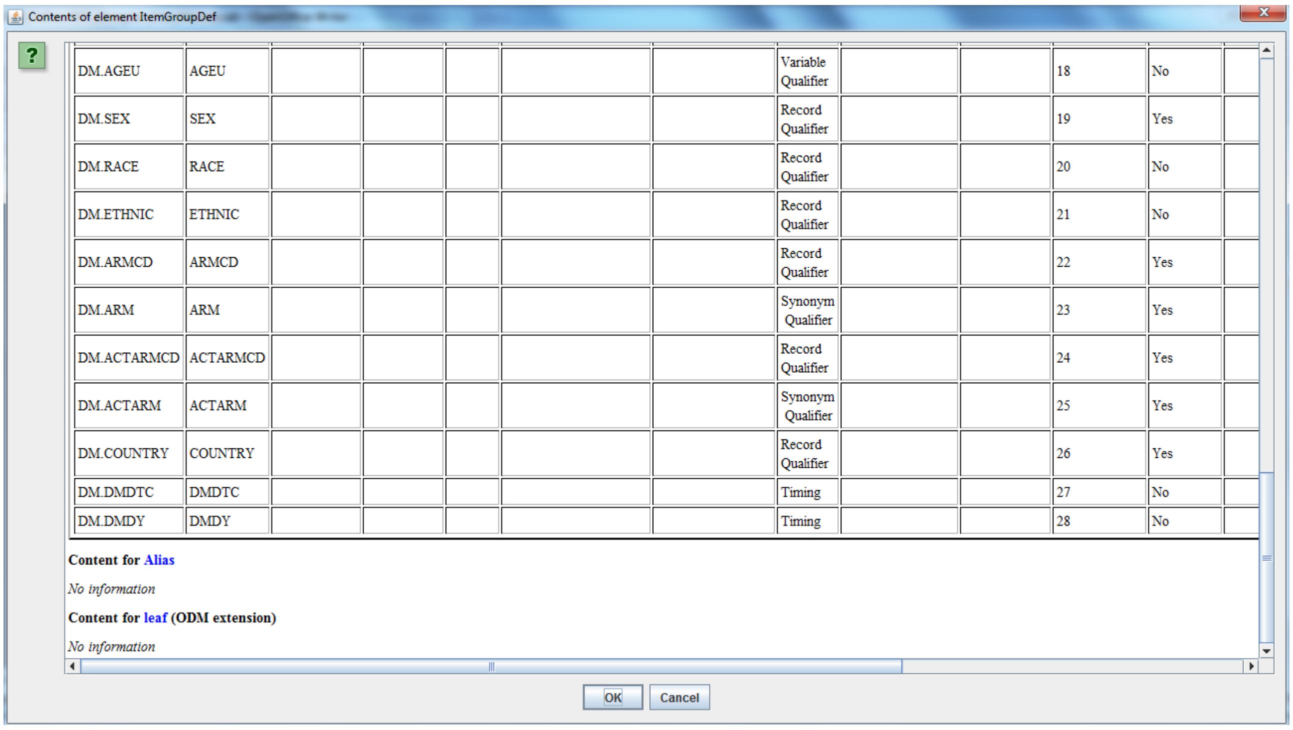
Attributes:

Name	Value
OID	DM
Name	DM
Repeating	No
IsReferenceData	No
SASDatasetName	DM
Domain	DM
Origin	
Role	
Purpose	Tabulation
Comment	
Structure	One record per USUBJID
Class	SPECIAL PURPOSE
ArchiveLocationID	Location.DM
CommentOID	

Content for Description

TranslatedText
Language: English Text: Demographics

and when scrolling to the bottom:



Contents of element ItemGroupDef

DM.AGEU	AGEU						Variable Qualifier			18	No	
DM.SEX	SEX						Record Qualifier			19	Yes	
DM.RACE	RACE						Record Qualifier			20	No	
DM.ETHNIC	ETHNIC						Record Qualifier			21	No	
DM.ARMCD	ARMCD						Record Qualifier			22	Yes	
DM.ARM	ARM						Synonym Qualifier			23	Yes	
DM.ACTARMCD	ACTARMCD						Record Qualifier			24	Yes	
DM.ACTARM	ACTARM						Synonym Qualifier			25	Yes	
DM.COUNTRY	COUNTRY						Record Qualifier			26	Yes	
DM.DMDTC	DMDTC						Timing			27	No	
DM.DMDY	DMDY						Timing			28	No	

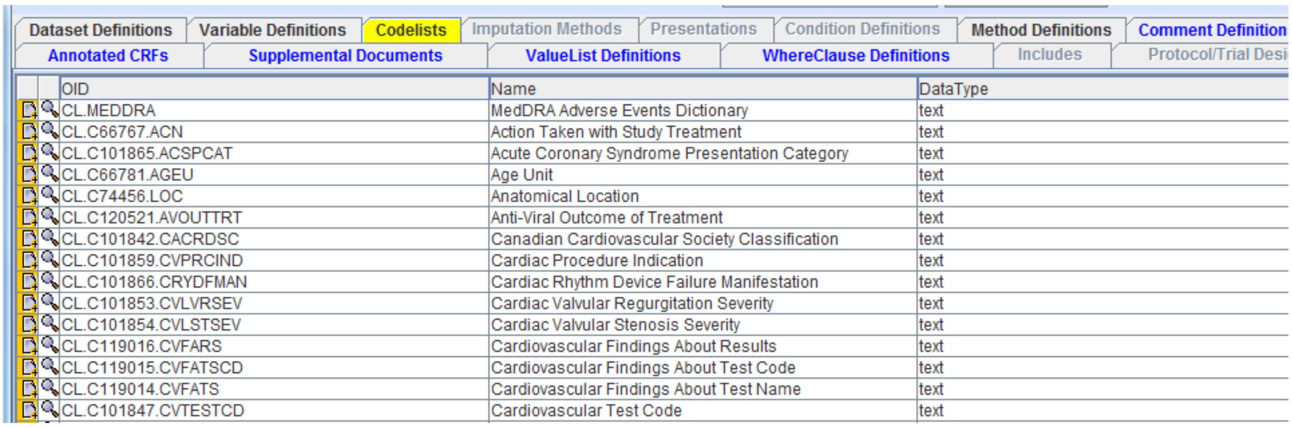
Content for Alias

No information

Content for leaf (ODM extension)

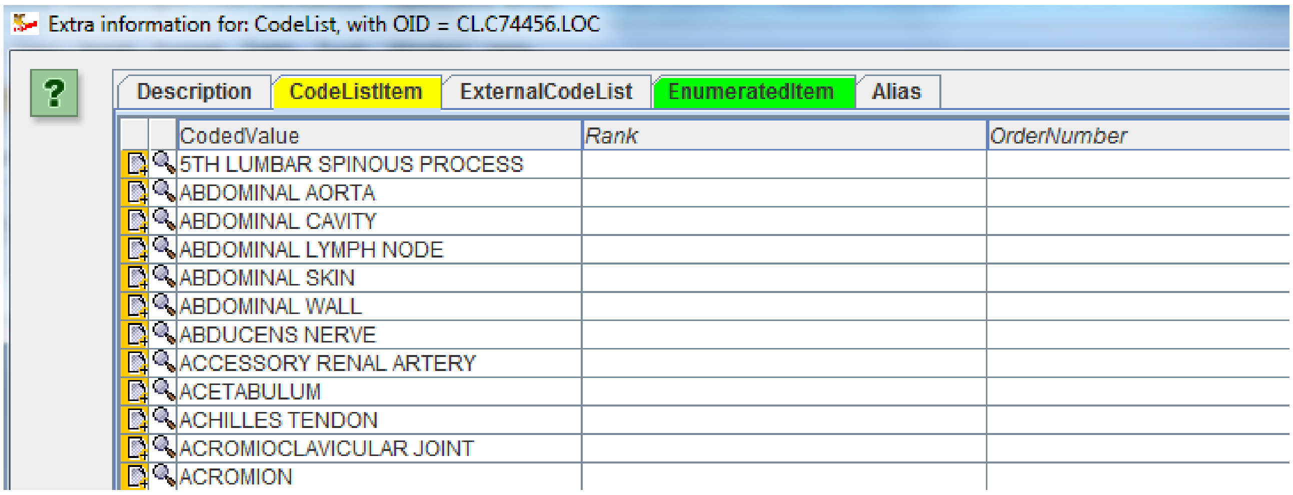
No information

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



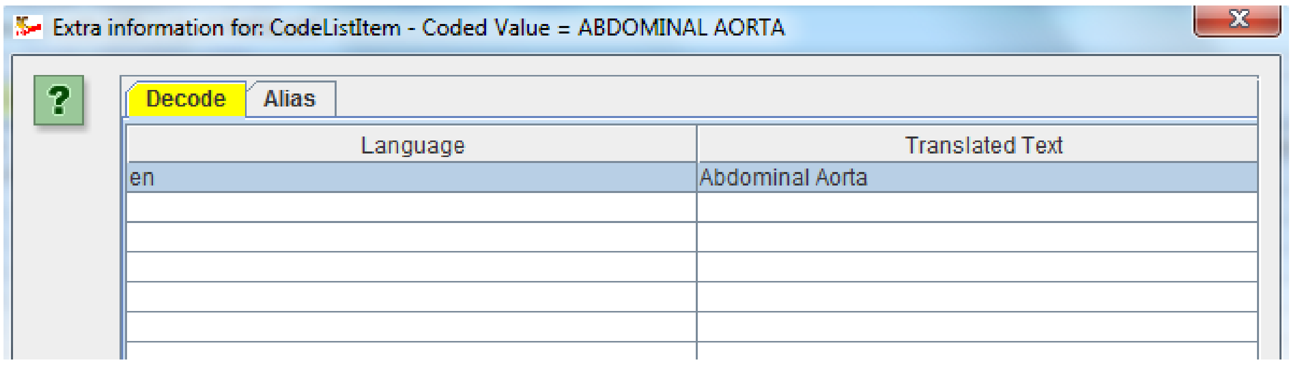
OID	Name	DataType
CL.MEDDRA	MedDRA Adverse Events Dictionary	text
CL.C66767.ACN	Action Taken with Study Treatment	text
CL.C101865.ACSPCAT	Acute Coronary Syndrome Presentation Category	text
CL.C66781.AGEU	Age Unit	text
CL.C74456.LOC	Anatomical Location	text
CL.C120521.AVOUTTRT	Anti-Viral Outcome of Treatment	text
CL.C101842.CACRDSC	Canadian Cardiovascular Society Classification	text
CL.C101859.CVPRCIND	Cardiac Procedure Indication	text
CL.C101866.CRYDFMAN	Cardiac Rhythm Device Failure Manifestation	text
CL.C101853.CVLVRSEV	Cardiac Valvular Regurgitation Severity	text
CL.C101854.CVLSTSEV	Cardiac Valvular Stenosis Severity	text
CL.C119016.CVFARS	Cardiovascular Findings About Results	text
CL.C119015.CVFATSCD	Cardiovascular Findings About Test Code	text
CL.C119014.CVFATS	Cardiovascular Findings About Test Name	text
CL.C101847.CVTESTCD	Cardiovascular Test Code	text

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



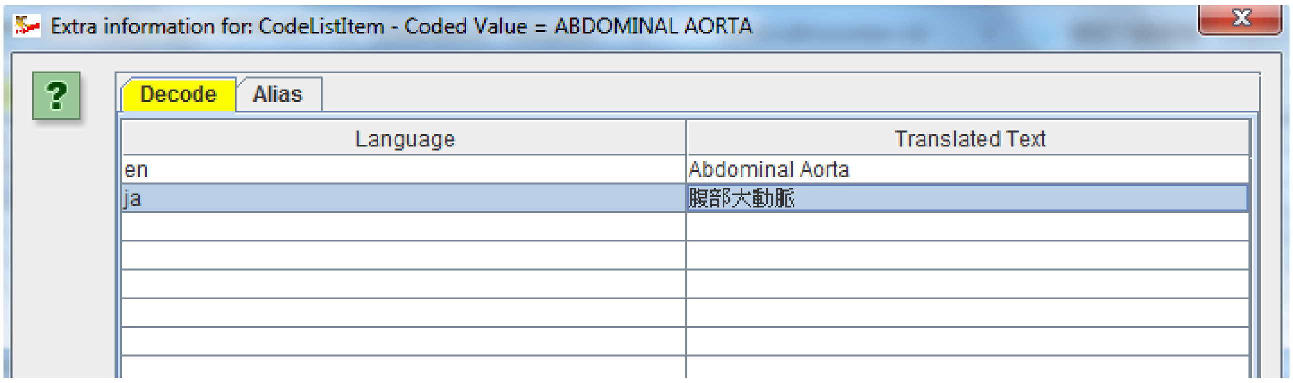
CodedValue	Rank	OrderNumber
5TH LUMBAR SPINOUS PROCESS		
ABDOMINAL AORTA		
ABDOMINAL CAVITY		
ABDOMINAL LYMPH NODE		
ABDOMINAL SKIN		
ABDOMINAL WALL		
ABDUCENS NERVE		
ACCESSORY RENAL ARTERY		
ACETABULUM		
ACHILLES TENDON		
ACROMIOCLAVICULAR JOINT		
ACROMION		

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


Extra information for: CodeListItem - Coded Value = ABDOMINAL AORTA
?
Decode
Alias

Language	Translated Text
en	Abdominal Aorta

If it would be necessary to also add the Japanese text, this is the place to do so: in the second row, click the cell under "Language" and select "Japanese". For "translated text" then add the japanese translation, e.g.:


Extra information for: CodeListItem - Coded Value = ABDOMINAL AORTA

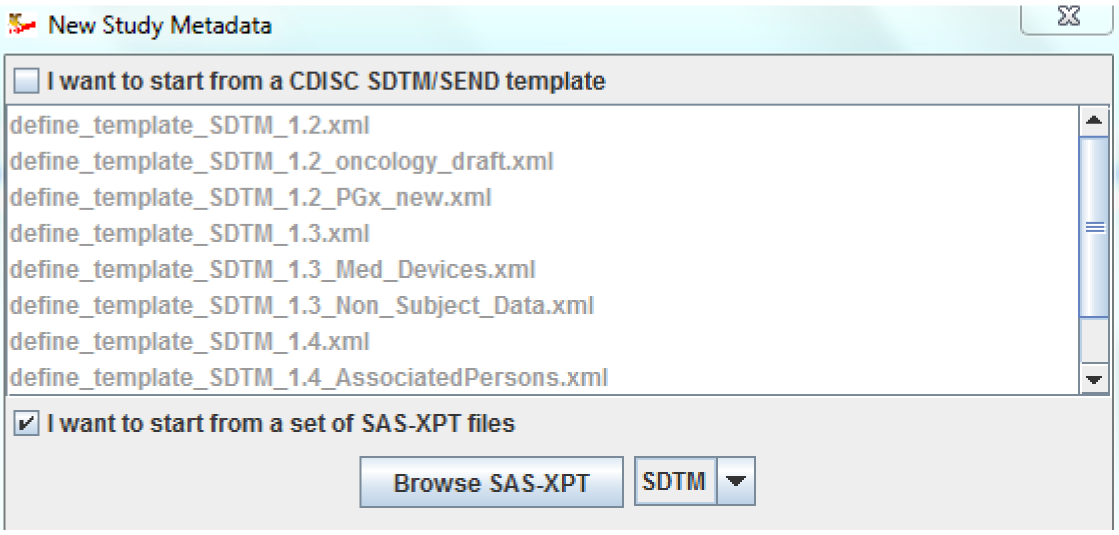
Language	Translated Text
en	Abdominal Aorta
ja	腹部大動脈

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

<u>Creating a define.xml starting from an existing set of SAS-XPT files</u>

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

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

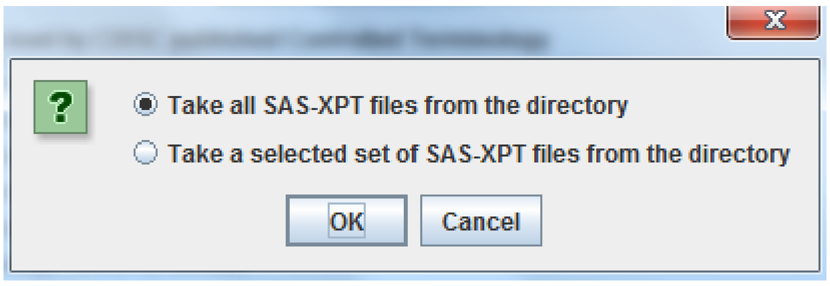

New Study Metadata

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

define_template_SDTM_1.2.xml
define_template_SDTM_1.2_oncology_draft.xml
define_template_SDTM_1.2_PGx_new.xml
define_template_SDTM_1.3.xml
define_template_SDTM_1.3_Med_Devices.xml
define_template_SDTM_1.3_Non_Subject_Data.xml
define_template_SDTM_1.4.xml
define_template_SDTM_1.4_AssociatedPersons.xml

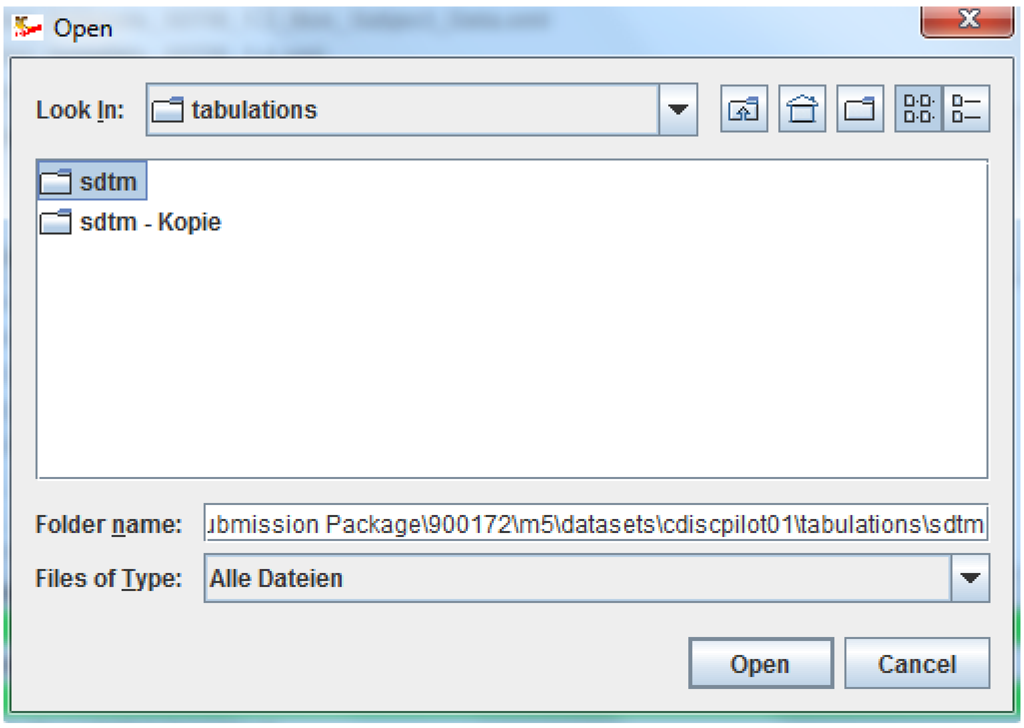
☑ I want to start from a set of SAS-XPT files

Then select the model, which can either be "SDTM" or "SEND" or "ADaM" or "Other". We will here demonstrate the functionality for the case of an SDTM define.xml.
Then click the "Browse SAS-XPT" button. A dialog is displayed:



(● selected) Take all SAS-XPT files from the directory
Take a selected set of SAS-XPT files from the directory

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

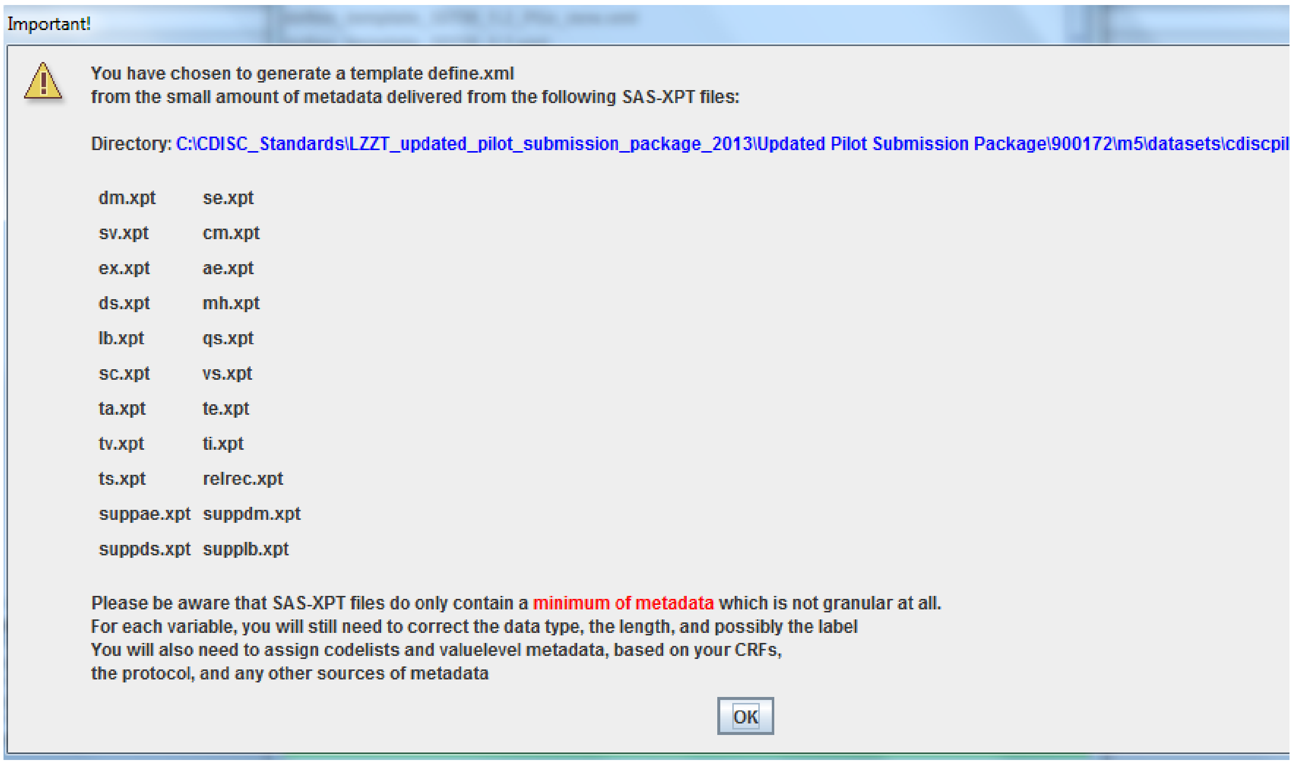


Look In: tabulations

sdtm
sdtm - Kopie

Folder name: ıbmission Package\900172\m5\datasets\cdiscpilot01\tabulations\sdtm
Files of Type: Alle Dateien

And after clicking "Open":


Important!

You have chosen to generate a template define.xml
from the small amount of metadata delivered from the following SAS-XPT files:

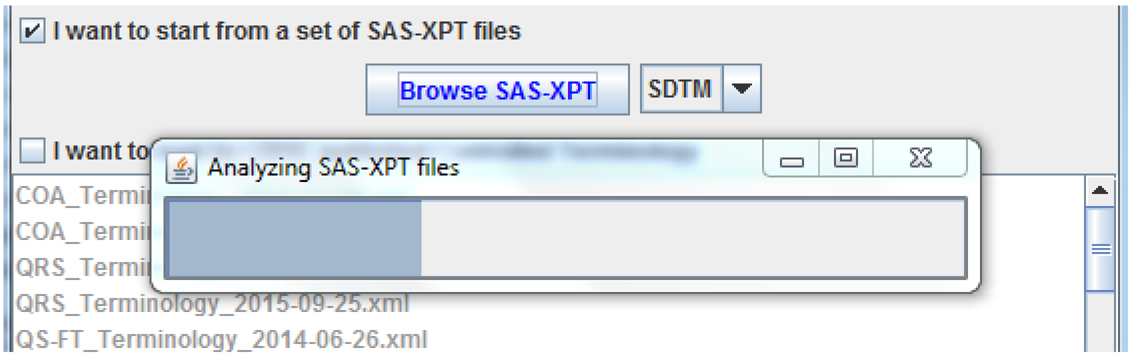
Directory: C:\CDISC_Standards\LZZT_updated_pilot_submission_package_2013\Updated Pilot Submission Package\900172\m5\datasets\cdiscpil

dm.xpt	se.xpt
sv.xpt	cm.xpt
ex.xpt	ae.xpt
ds.xpt	mh.xpt
lb.xpt	qs.xpt
sc.xpt	vs.xpt
ta.xpt	te.xpt
tv.xpt	ti.xpt
ts.xpt	relrec.xpt
suppae.xpt	suppdm.xpt
suppds.xpt	supplb.xpt

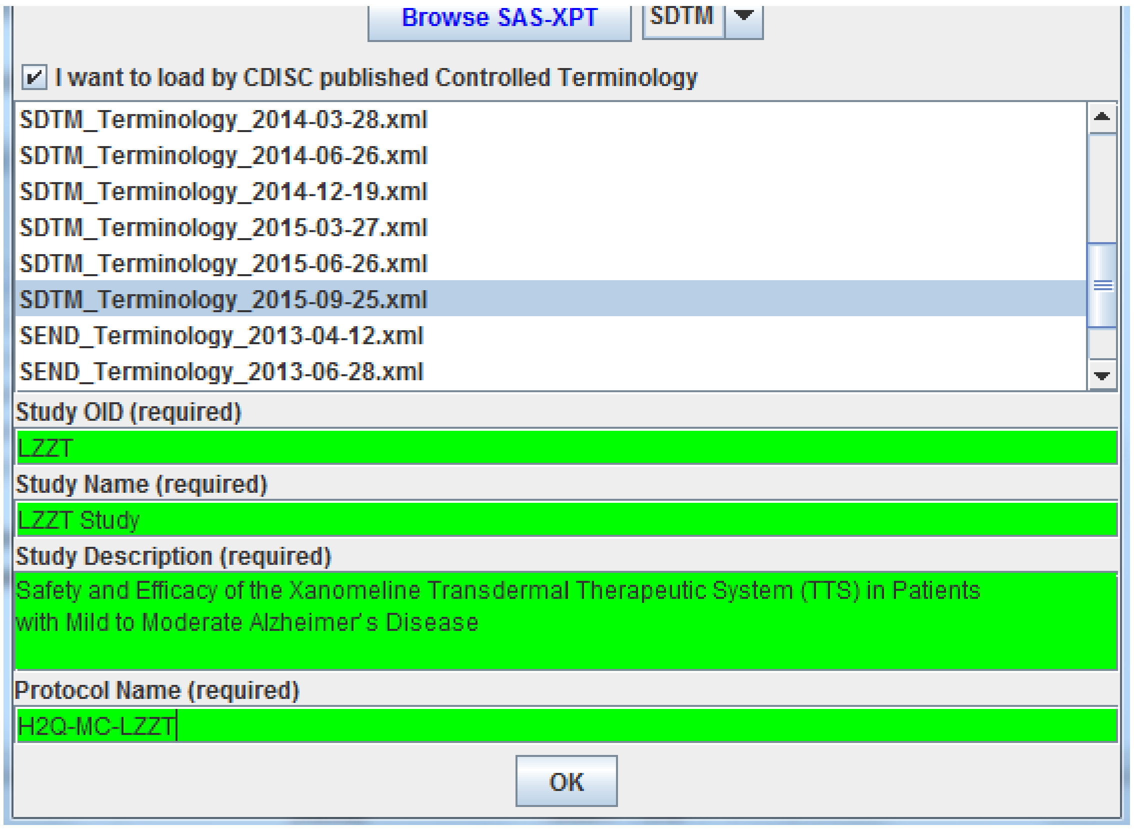
Please be aware that SAS-XPT files do only contain a minimum of metadata which is not granular at all.
For each variable, you will still need to correct the data type, the length, and possibly the label
You will also need to assign codelists and valuelevel metadata, based on your CRFs,
the protocol, and any other sources of metadata

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

After clicking OK:


✓ I want to start from a set of SAS-XPT files
Browse SAS-XPT
SDTM
I want to
Analyzing SAS-XPT files
COA_Termi
COA_Termi
QRS_Termi
QRS_Terminology_2015-09-25.xml
QS-FT_Terminology_2014-06-26.xml

The system starts analyzing the SAS-XPT files to generate a first primitive prototype of the define.xml. After that, it is not a bad idea to select a version of the CDISC controlled terminology, and to add the general information that goes into the define.xml:

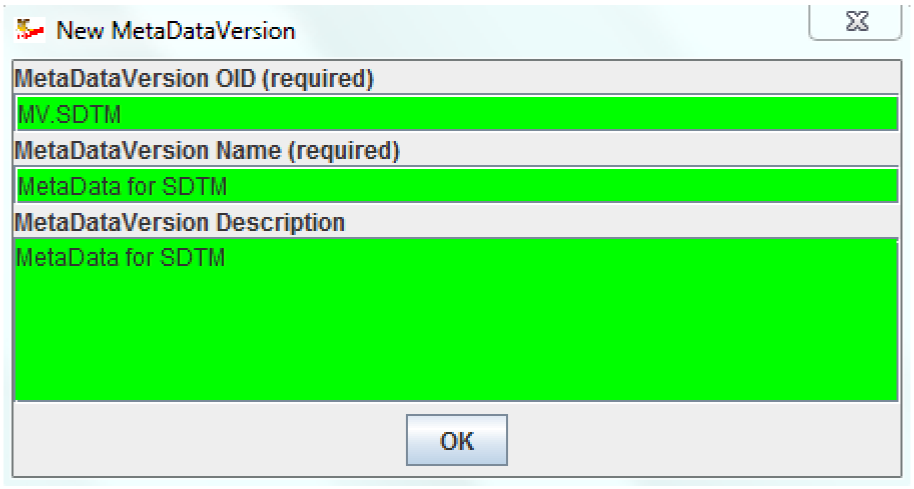


☑ I want to load by CDISC published Controlled Terminology

SDTM_Terminology_2014-03-28.xml
SDTM_Terminology_2014-06-26.xml
SDTM_Terminology_2014-12-19.xml
SDTM_Terminology_2015-03-27.xml
SDTM_Terminology_2015-06-26.xml
SDTM_Terminology_2015-09-25.xml
SEND_Terminology_2013-04-12.xml
SEND_Terminology_2013-06-28.xml

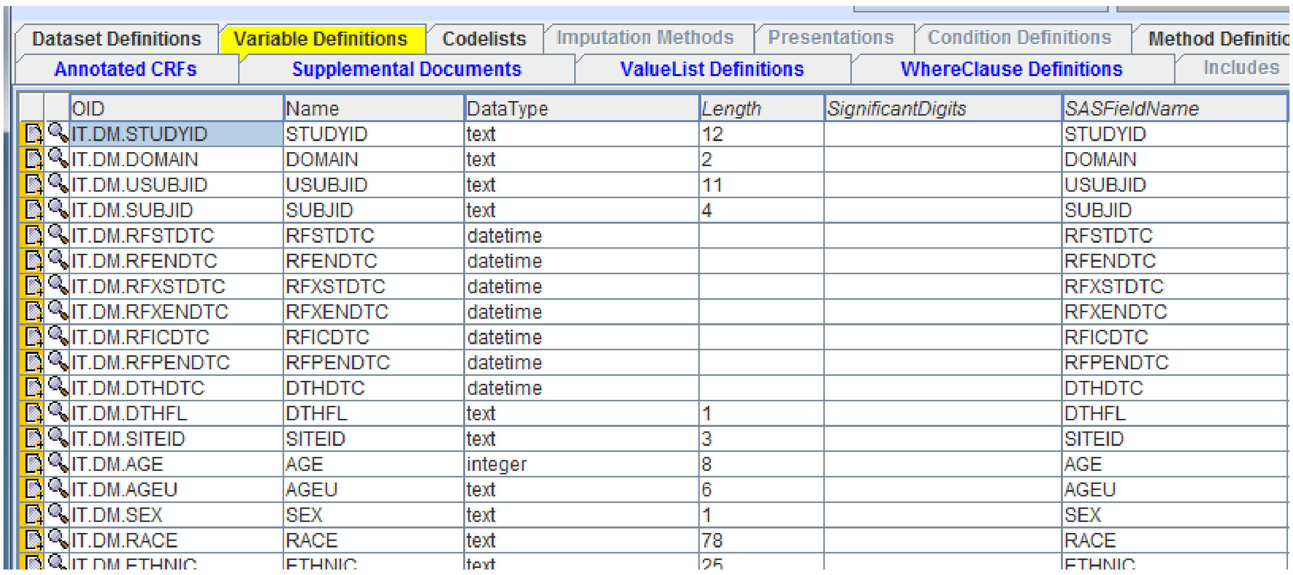
Study OID (required)
LZZT
Study Name (required)
LZZT Study
Study Description (required)
Safety and Efficacy of the Xanomeline Transdermal Therapeutic System (TTS) in Patients with Mild to Moderate Alzheimer's Disease
Protocol Name (required)
H2Q-MC-LZZT

After clicking OK, a proposal is made for the "MetaDataVersion OID", "Name" and Description":



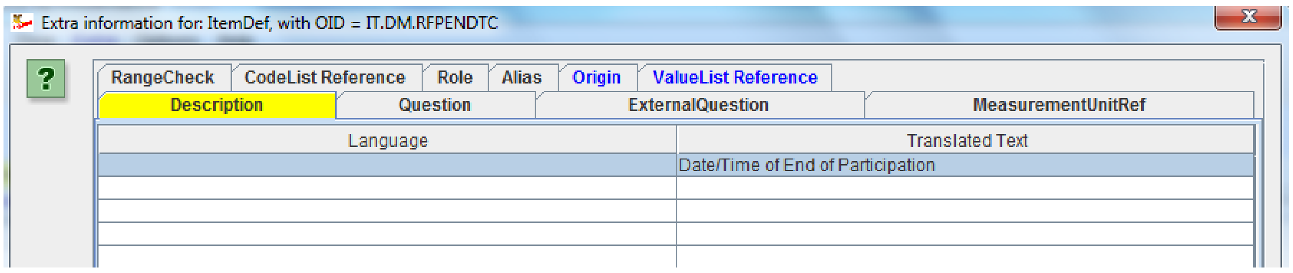
MetaDataVersion OID (required)
MV.SDTM
MetaDataVersion Name (required)
MetaData for SDTM
MetaDataVersion Description
MetaData for SDTM

Clicking OK again then leads to a first proposal for the variable definitions:



OID	Name	DataType	Length	SignificantDigits	SASFieldName
IT.DM.STUDYID	STUDYID	text	12		STUDYID
IT.DM.DOMAIN	DOMAIN	text	2		DOMAIN
IT.DM.USUBJID	USUBJID	text	11		USUBJID
IT.DM.SUBJID	SUBJID	text	4		SUBJID
IT.DM.RFSTDTC	RFSTDTC	datetime			RFSTDTC
IT.DM.RFENDTC	RFENDTC	datetime			RFENDTC
IT.DM.RFXSTDTC	RFXSTDTC	datetime			RFXSTDTC
IT.DM.RFXENDTC	RFXENDTC	datetime			RFXENDTC
IT.DM.RFICDTC	RFICDTC	datetime			RFICDTC
IT.DM.RFPENDTC	RFPENDTC	datetime			RFPENDTC
IT.DM.DTHDTC	DTHDTC	datetime			DTHDTC
IT.DM.DTHFL	DTHFL	text	1		DTHFL
IT.DM.SITEID	SITEID	text	3		SITEID
IT.DM.AGE	AGE	integer	8		AGE
IT.DM.AGEU	AGEU	text	6		AGEU
IT.DM.SEX	SEX	text	1		SEX
IT.DM.RACE	RACE	text	78		RACE
IT.DM.ETHNIC	ETHNIC	text	25		ETHNIC

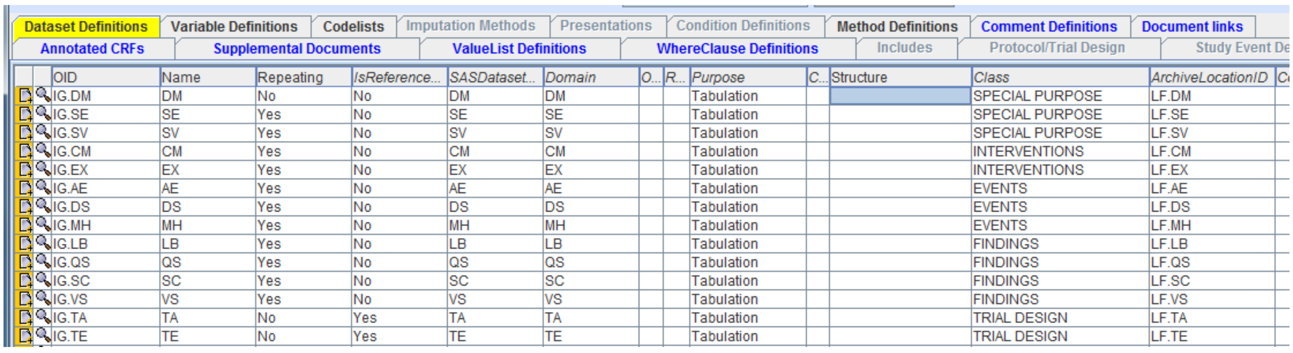
One sees that a first estimate of the datatype has been made, and that the maximum length has been taken from the field definitions within the SAS-XPT files. Also the "label" has been taken from the SAS-XPT file, for example for "RFPENDTC":



Language	Translated Text
	Date/Time of End of Participation

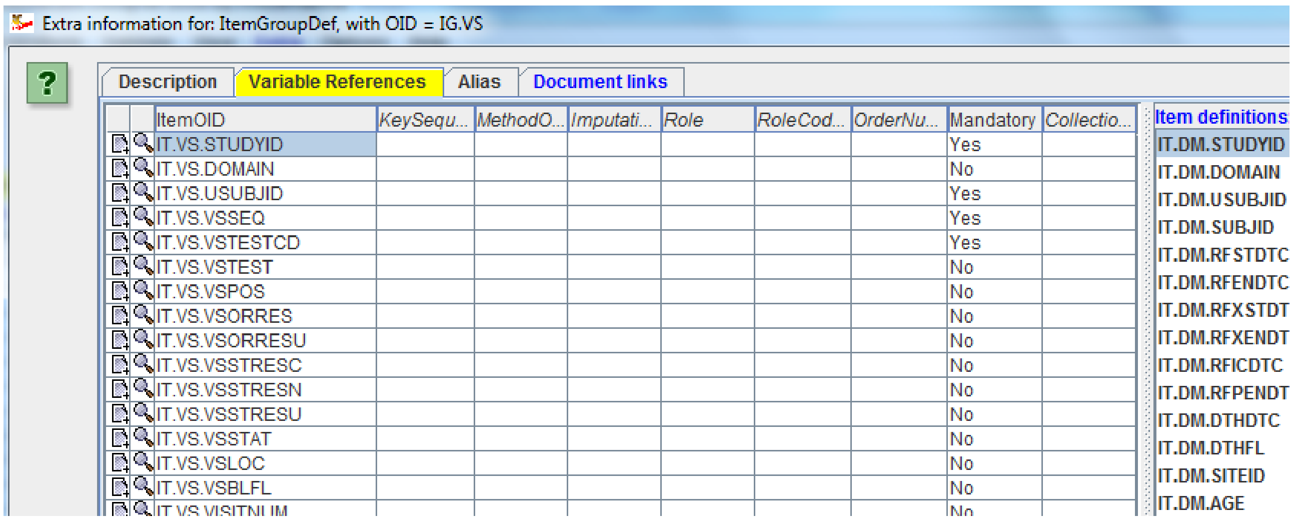
Remark that no "language" needs to be defined, as the english language is considered as the default language by the define.xml standard.

For the dataset definitions, we find:



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

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

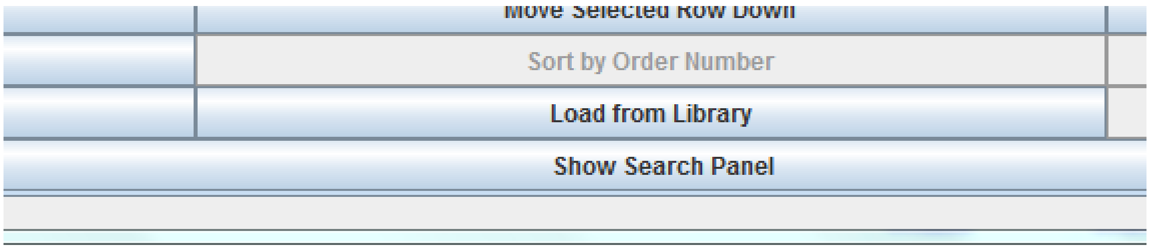


ItemOID	KeySequ...	MethodO...	Imputati...	Role	RoleCod...	OrderNu...	Mandatory	Collectio...
IT.VS.STUDYID							Yes	
IT.VS.DOMAIN							No	
IT.VS.USUBJID							Yes	
IT.VS.VSSEQ							Yes	
IT.VS.VSTESTCD							Yes	
IT.VS.VSTEST							No	
IT.VS.VSPOS							No	
IT.VS.VSORRES							No	
IT.VS.VSORRESU							No	
IT.VS.VSSTRESC							No	
IT.VS.VSSTRESN							No	
IT.VS.VSSTRESU							No	
IT.VS.VSSTAT							No	
IT.VS.VSLOC							No	
IT.VS.VSBLFL							No	
IT.VS.VISITNUM							No	

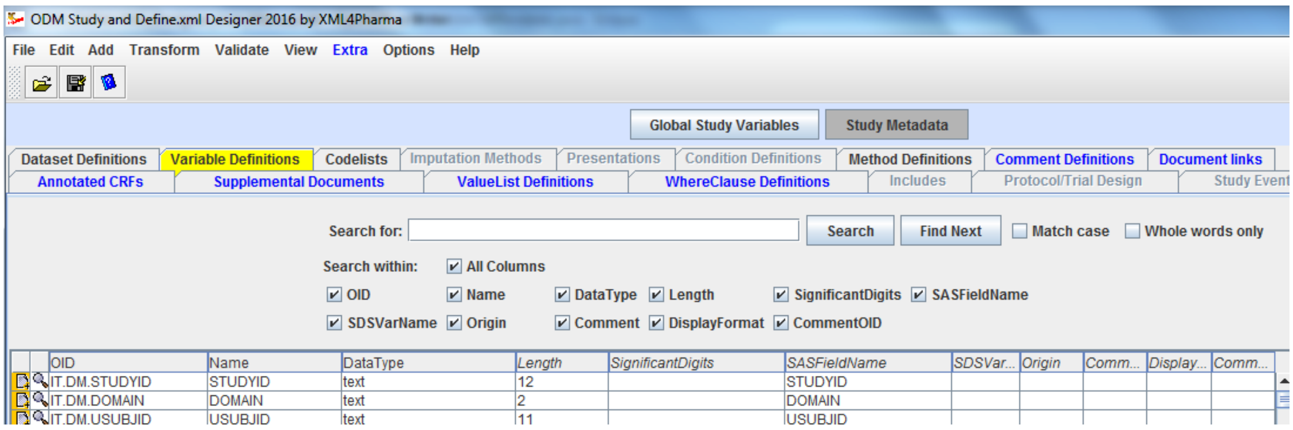
Item definitions
IT.DM.STUDYID
IT.DM.DOMAIN
IT.DM.USUBJID
IT.DM.SUBJID
IT.DM.RFSTDTC
IT.DM.RFENDTC
IT.DM.RFXSTDTC
IT.DM.RFXENDTC
IT.DM.RFICDTC
IT.DM.RFPENDTC
IT.DM.DTHDTC
IT.DM.DTHFL
IT.DM.SITEID
IT.DM.AGE

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

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



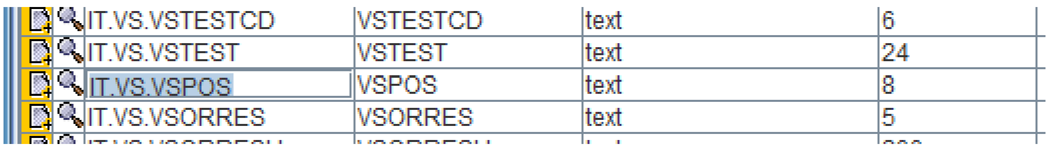
Clicking the "Show Search Panel" leads to:



OID	Name	DataType	Length	SignificantDigits	SASFieldName	SDSVar...	Origin	Comm...	Display...	Comm...
IT.DM.STUDYID	STUDYID	text	12		STUDYID					
IT.DM.DOMAIN	DOMAIN	text	2		DOMAIN					
IT.DM.USUBJID	USUBJID	text	11		USUBJID					

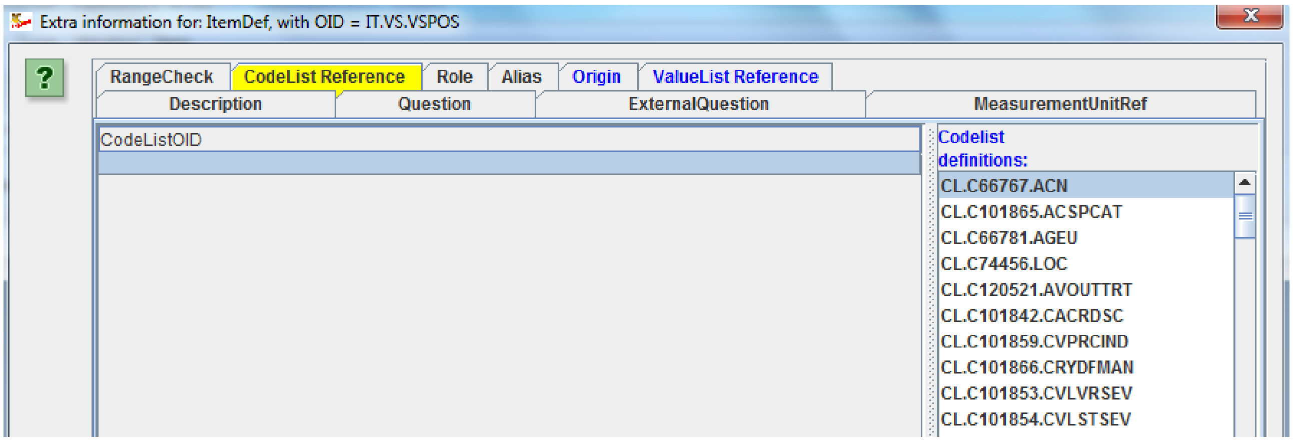
allowing to search in the table with variables. Using the checkboxes one can limit the search to certain fields/columns.

Let us know have a look at some variables that usually are coded, like VSPOS (Vital Sign Position):



IT.VS.VSTESTCD	VSTESTCD	text	6
IT.VS.VSTEST	VSTEST	text	24
IT.VS.VSPOS	VSPOS	text	8
IT.VS.VSORRES	VSORRES	text	5

It has been defined as being of the "text" datatype, with a maximum length of 8, which looks OK. Clicking the "+" icon to drill down into the details and selecting "CodeList Reference" leads to:

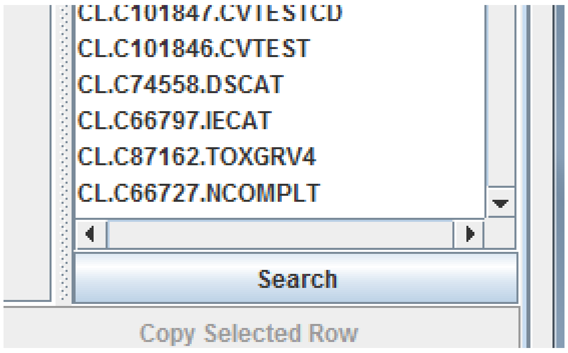

Extra information for: ItemDef, with OID = IT.VS.VSPOS

CL.C66767.ACN
CL.C101865.ACSPCAT
CL.C66781.AGEU
CL.C74456.LOC
CL.C120521.AVOUTTRT
CL.C101842.CACRDSC
CL.C101859.CVPRCIND
CL.C101866.CRYDFMAN
CL.C101853.CVLVRSEV
CL.C101854.CVLSTSEV

Noticing that the system hasn't assigned a codelist yet (and also did not try to generate one from the data in the SAS-XPT file).

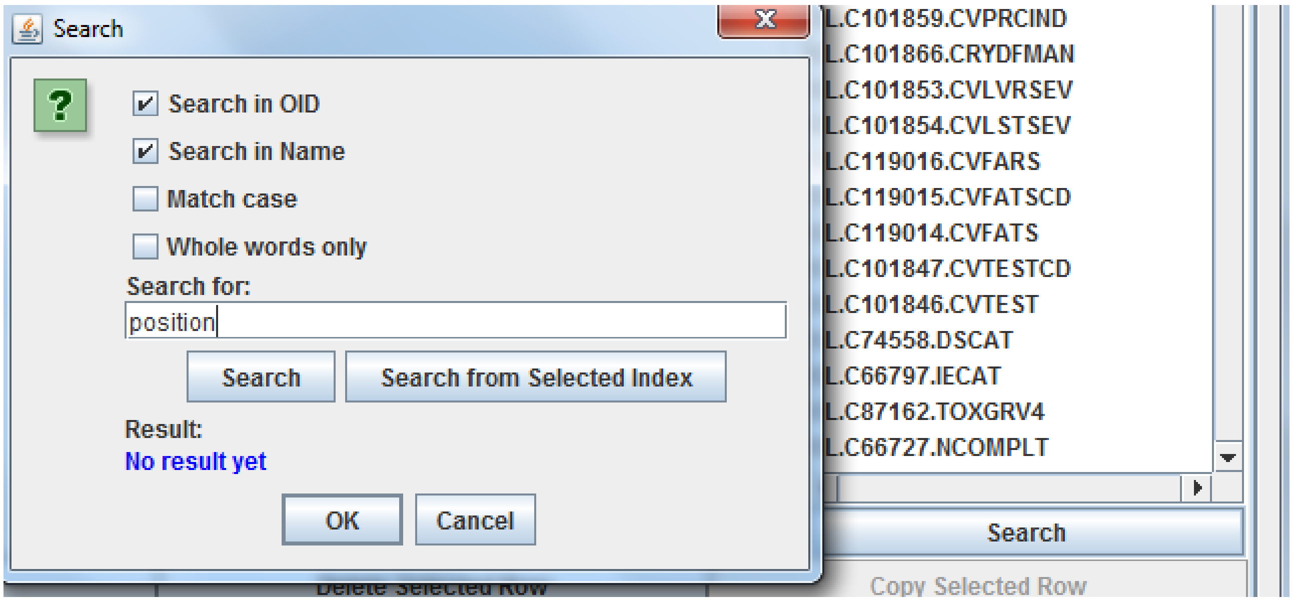
Why is this?

The reason is that the system cannot know whether the user used CDISC controlled terminology when designing the study. If it was the case, the by CDISC published codelist can now be assigned to the variable VSPOS. This can easily be achieved by using the "Search" button near the end of the list with codelists on the right side:


CL.C101847.CVTESTCD
CL.C101846.CVTEST
CL.C74558.DSCAT
CL.C66797.IECAT
CL.C87162.TOXGRV4
CL.C66727.NCOMPLT

and adding e.g. "position" to the search field:

leading to:


Search

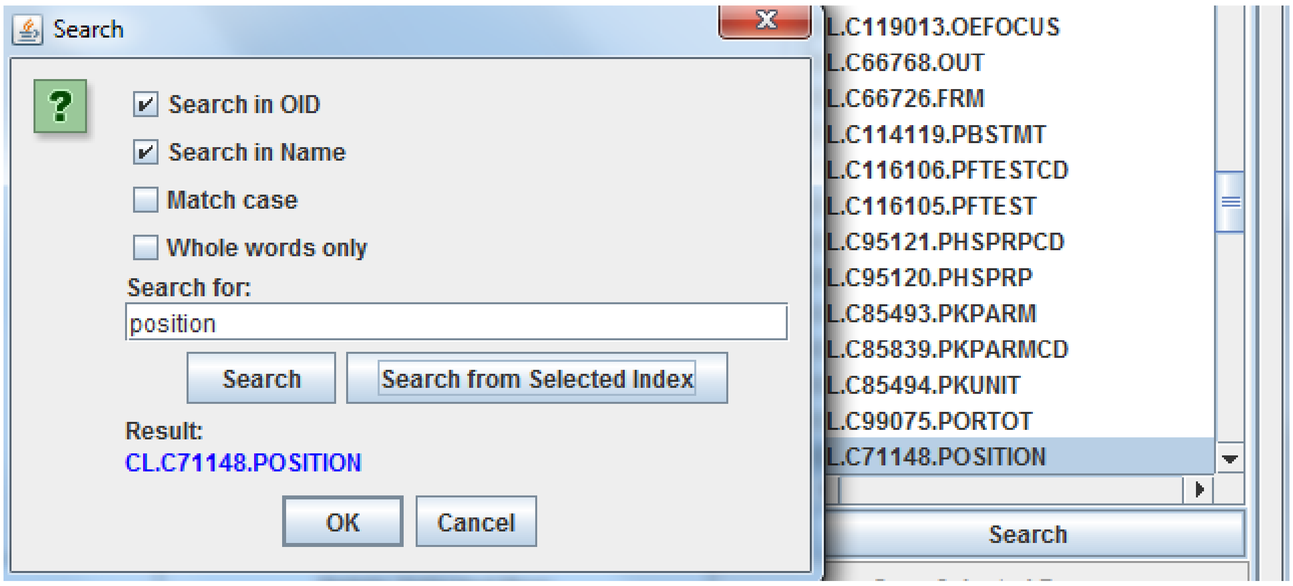
[✓] Search in OID
[✓] Search in Name
[ ] Match case
[ ] Whole words only

Search for: position

Result: No result yet

L.C101859.CVPRCIND
L.C101866.CRYDFMAN
L.C101853.CVLVRSEV
L.C101854.CVLSTSEV
L.C119016.CVFARS
L.C119015.CVFATSCD
L.C119014.CVFATS
L.C101847.CVTESTCD
L.C101846.CVTEST
L.C74558.DSCAT
L.C66797.IECAT
L.C87162.TOXGRV4
L.C66727.NCOMPLT

leading to:


Search

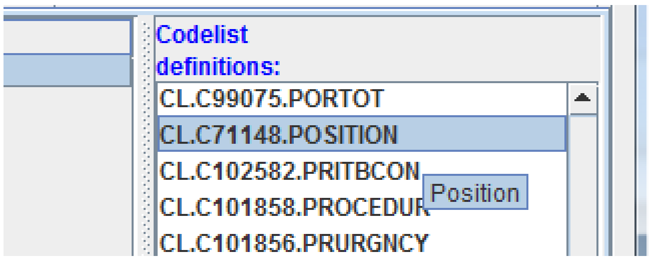
[✓] Search in OID
[✓] Search in Name
[ ] Match case
[ ] Whole words only

Search for: position

Result: CL.C71148.POSITION

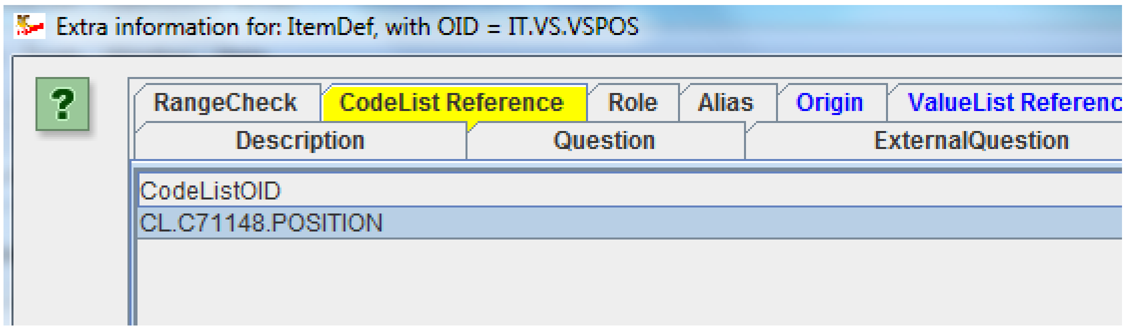
L.C119013.OEFOCUS
L.C66768.OUT
L.C66726.FRM
L.C114119.PBSTMT
L.C116106.PFTESTCD
L.C116105.PFTEST
L.C95121.PHSPRPCD
L.C95120.PHSPRP
L.C85493.PKPARM
L.C85839.PKPARMCD
L.C85494.PKUNIT
L.C99075.PORTOT
L.C71148.POSITION

After clicking OK:


Codelist definitions:

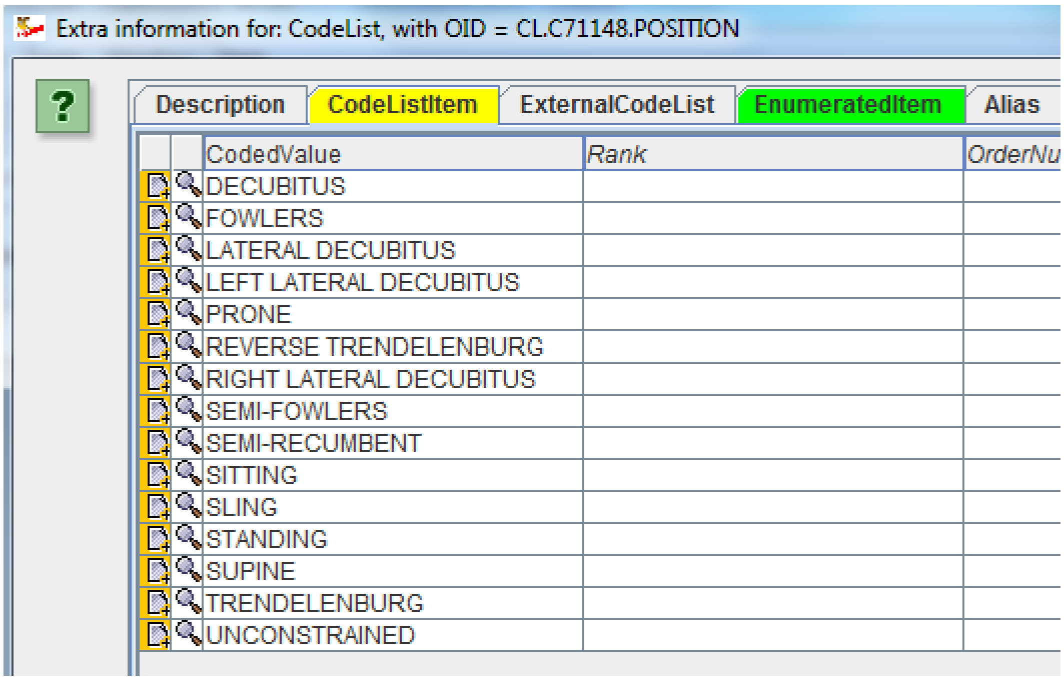
CL.C99075.PORTOT
CL.C71148.POSITION
CL.C102582.PRITBCON
CL.C101858.PROCEDUR
CL.C101856.PRURGNCY

One can then easily drag-and-drop "CL.C71148.POSITION" to the "CodeListOID" field, resulting in:



CodeListOID
CL.C71148.POSITION

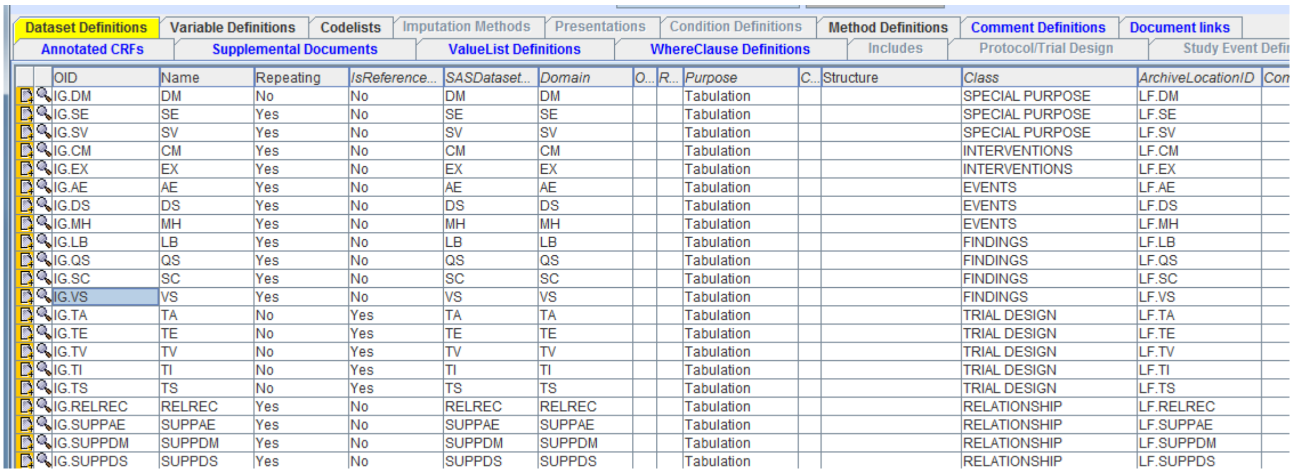
ATTENTION: this codelist contains 15 entries, as one can see when going back to the main panel, selecting the "CodeList" tab, and then selecting the codelist "CL.C71148.POSITION" and then clicking the "+" icon or the "magnifying glass" icon for going into the details:


Extra information for: CodeList, with OID = CL.C71148.POSITION
?
Description
CodeListItem
ExternalCodeList
EnumeratedItem
Alias

CodedValue	Rank	OrderNu
DECUBITUS		
FOWLERS		
LATERAL DECUBITUS		
LEFT LATERAL DECUBITUS		
PRONE		
REVERSE TRENDELENBURG		
RIGHT LATERAL DECUBITUS		
SEMI-FOWLERS		
SEMI-RECUMBENT		
SITTING		
SLING		
STANDING		
SUPINE		
TRENDELENBURG		
UNCONSTRAINED		

If you did not use all these vital signs positions, the better way is to subset the codelist, and to assign this subset codelist to VSPOS.
We will learn how to do this (and much more) in the next section.

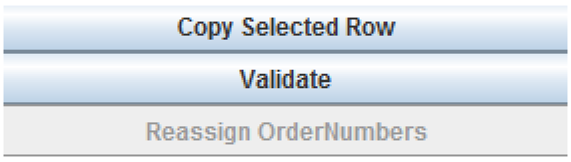
We can however already add some additiotional information at the "Dataset Definition" level:



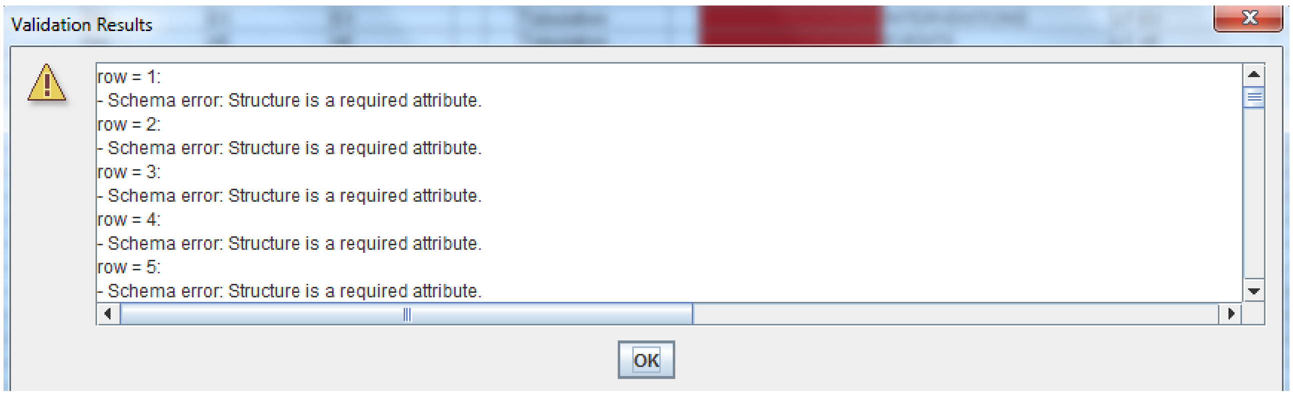
OID	Name	Repeating	IsReference...	SASDataset...	Domain	O...	R...	Purpose	C...	Structure	Class	ArchiveLocationID	Con
IG.DM	DM	No	No	DM	DM			Tabulation			SPECIAL PURPOSE	LF.DM	
IG.SE	SE	Yes	No	SE	SE			Tabulation			SPECIAL PURPOSE	LF.SE	
IG.SV	SV	Yes	No	SV	SV			Tabulation			SPECIAL PURPOSE	LF.SV	
IG.CM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

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

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

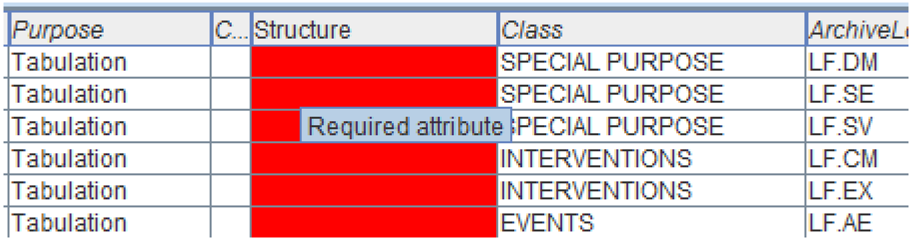


The software does a quick "local" validation and soon reports:



```
row = 1:
- Schema error: Structure is a required attribute.
row = 2:
- Schema error: Structure is a required attribute.
row = 3:
- Schema error: Structure is a required attribute.
row = 4:
- Schema error: Structure is a required attribute.
row = 5:
- Schema error: Structure is a required attribute.
```

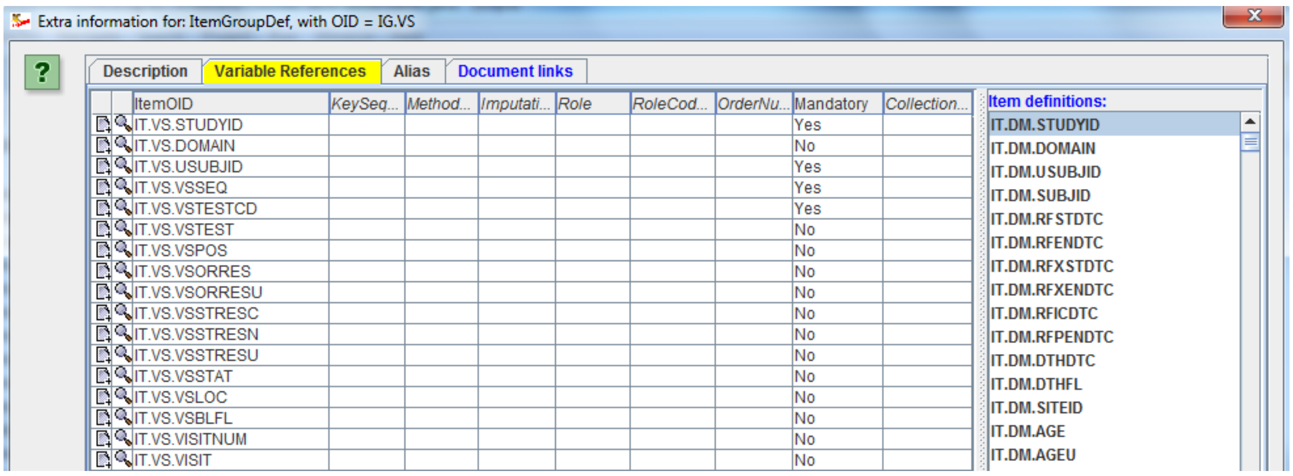
and in the table with the datasets itself, some cells are colored:



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

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

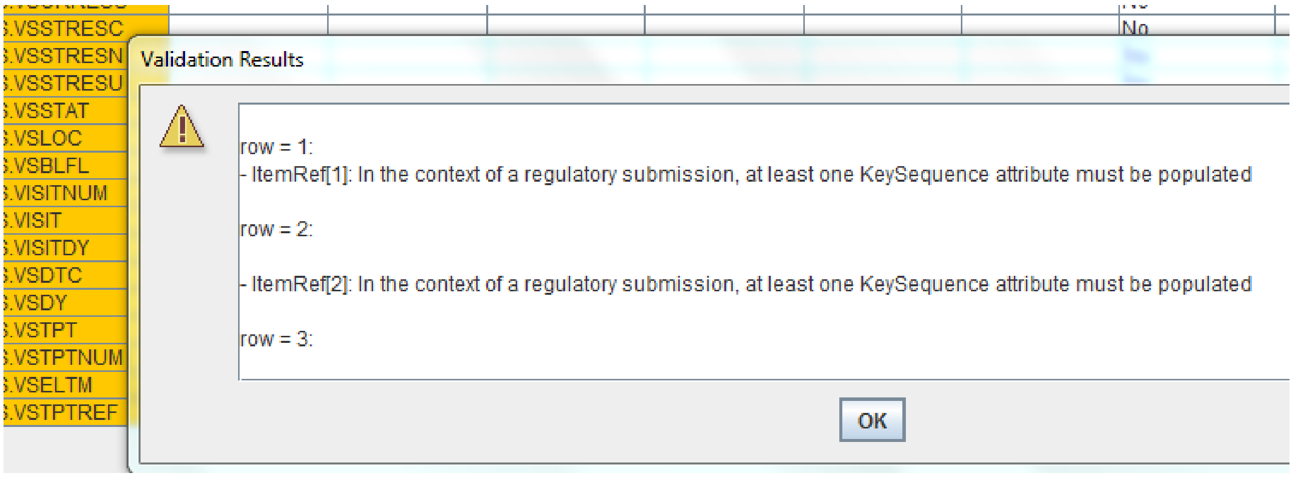
After having added the "structure" for each dataset, let us "drill down" into the details by clicking the "+" icon. For example for the "Variable References":


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

	ItemOID	KeySeq...	Method...	Imputati...	Role	RoleCod...	OrderNu...	Mandatory	Collection...
	IT.VS.STUDYID							Yes	
	IT.VS.DOMAIN							No	
	IT.VS.USUBJID							Yes	
	IT.VS.VSSEQ							Yes	
	IT.VS.VSTESTCD							Yes	
	IT.VS.VSTEST							No	
	IT.VS.VSPOS							No	
	IT.VS.VSORRES							No	
	IT.VS.VSORRESU							No	
	IT.VS.VSSTRESC							No	
	IT.VS.VSSTRESN							No	
	IT.VS.VSSTRESU							No	
	IT.VS.VSSTAT							No	
	IT.VS.VSLOC							No	
	IT.VS.VSBLFL							No	
	IT.VS.VISITNUM							No	
	IT.VS.VISIT							No	

Item definitions:
IT.DM.STUDYID
IT.DM.DOMAIN
IT.DM.USUBJID
IT.DM.SUBJID
IT.DM.RFSTDTC
IT.DM.RFENDTC
IT.DM.RFXSTDTC
IT.DM.RFXENDTC
IT.DM.RFICDTC
IT.DM.RFPENDTC
IT.DM.DTHDTC
IT.DM.DTHFL
IT.DM.SITEID
IT.DM.AGE
IT.DM.AGEU

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



Validation Results

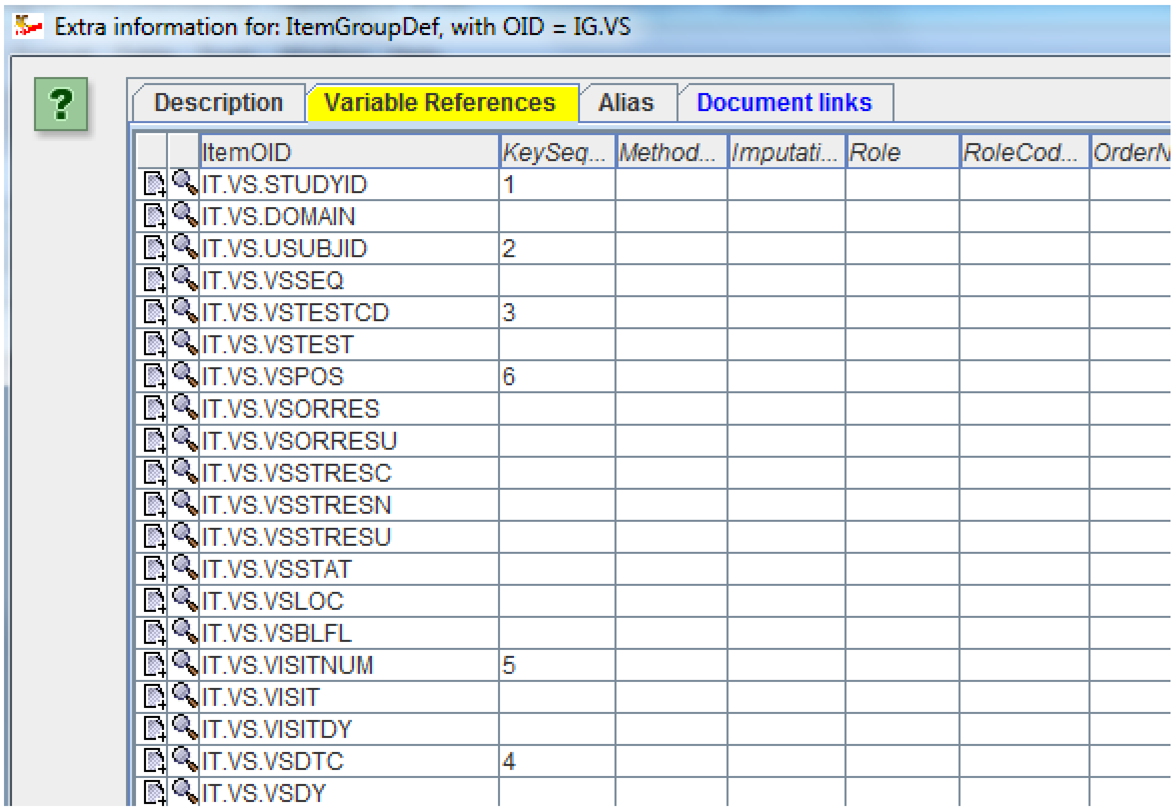
row = 1:
- ItemRef[1]: In the context of a regulatory submission, at least one KeySequence attribute must be populated

row = 2:

- ItemRef[2]: In the context of a regulatory submission, at least one KeySequence attribute must be populated

row = 3:

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

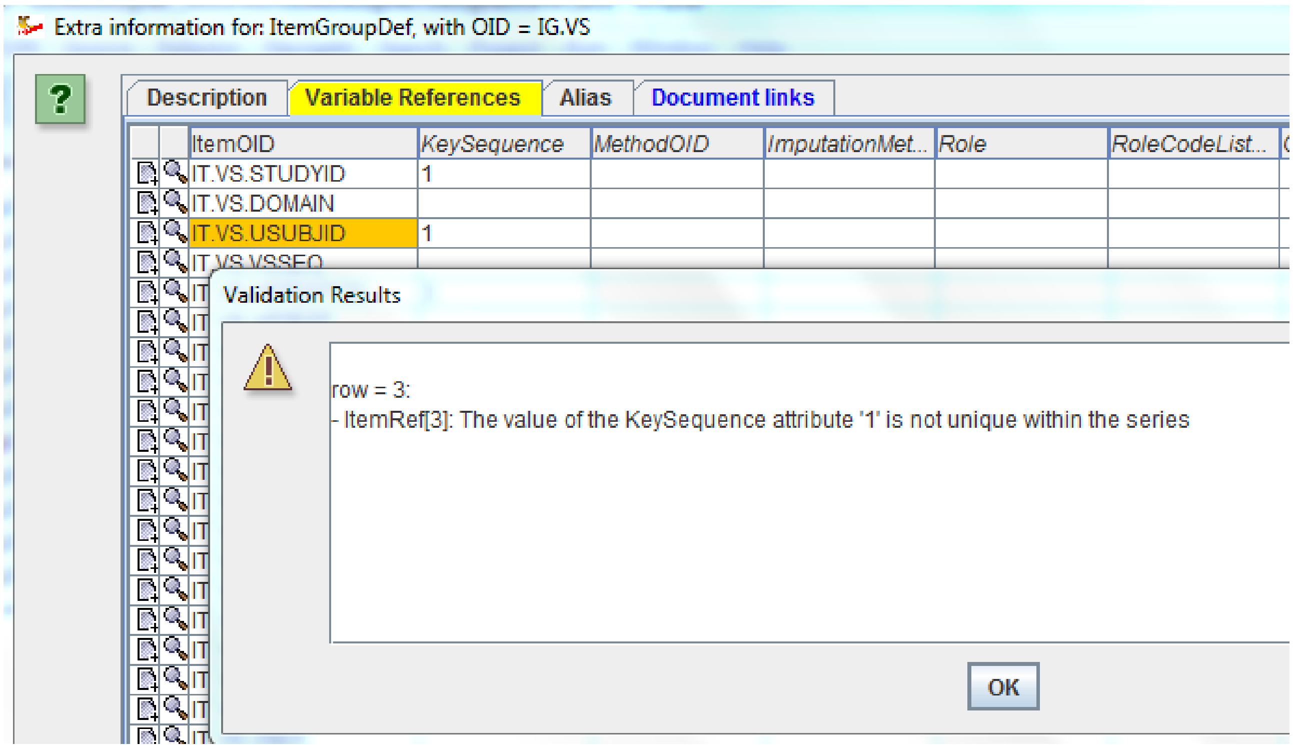

Extra information for: ItemGroupDef, with OID = IG.VS

Description | Variable References | Alias | Document links

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

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

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


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

ItemOID	KeySequence	MethodOID	ImputationMet...	Role	RoleCodeList...
IT.VS.STUDYID	1				
IT.VS.DOMAIN					
IT.VS.USUBJID	1				
IT.VS.VSSEQ					

Validation Results

row = 3:
- ItemRef[3]: The value of the KeySequence attribute '1' is not unique within the series

Adding and subsetting CDISC Controlled Terminology

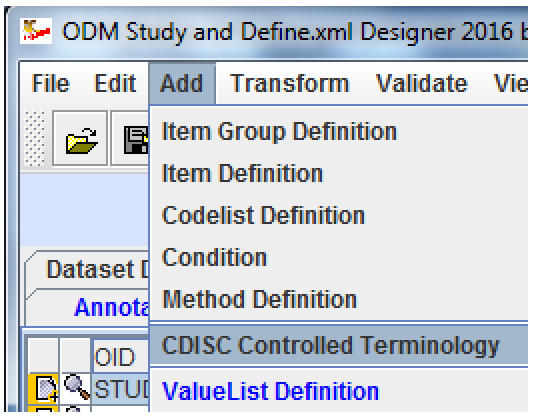
For most findings domains, you will want to use CDISC controlled terminology. When starting a new define.xml you will be able to load all the CDISC controlled terminology that is needed. When new CDISC controlled terminology is published, you can simply copy the new XML file to the directory, and it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

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

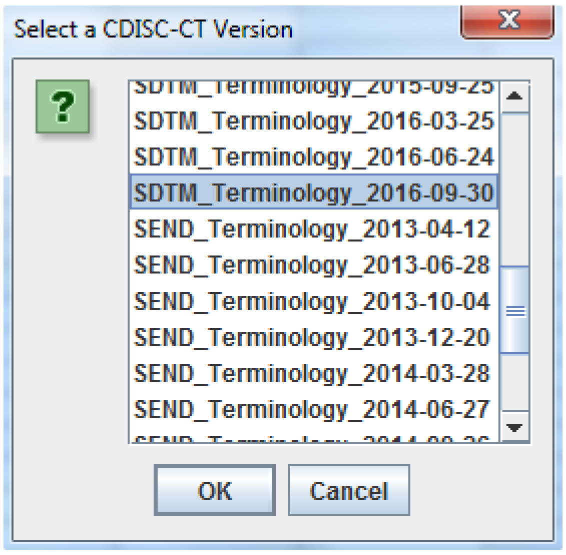
In case you create your define.xml before generating the SDTM/SEND/ADaM datasets, you will probably subset your codelists with information from the protocol, or from the CRFs.

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

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

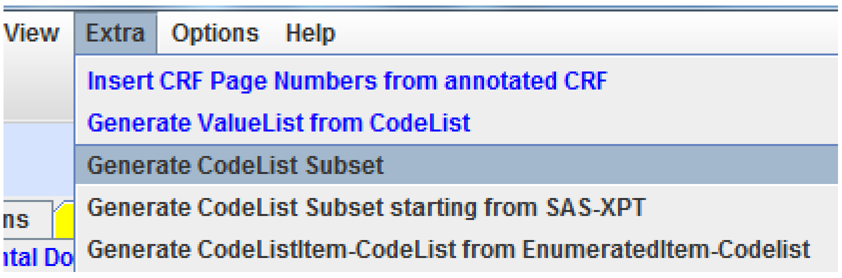


All CDISC controlled terminology is then presented as a list:

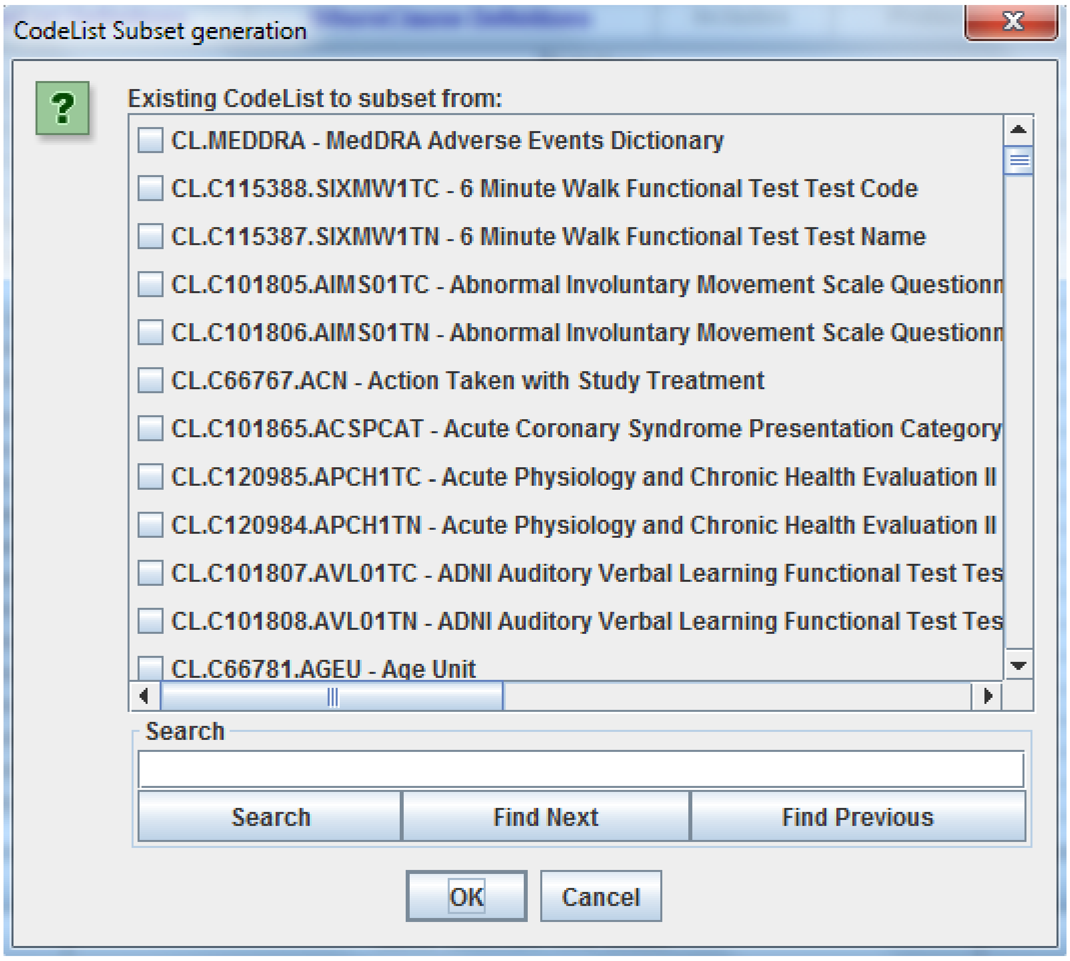

Select a CDISC-CT Version
SDTM_Terminology_2015-09-25
SDTM_Terminology_2016-03-25
SDTM_Terminology_2016-06-24
SDTM_Terminology_2016-09-30
SEND_Terminology_2013-04-12
SEND_Terminology_2013-06-28
SEND_Terminology_2013-10-04
SEND_Terminology_2013-12-20
SEND_Terminology_2014-03-28
SEND_Terminology_2014-06-27
OK
Cancel

We choose e.g. the SDTM controlled terminology version 2016-09-30.

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

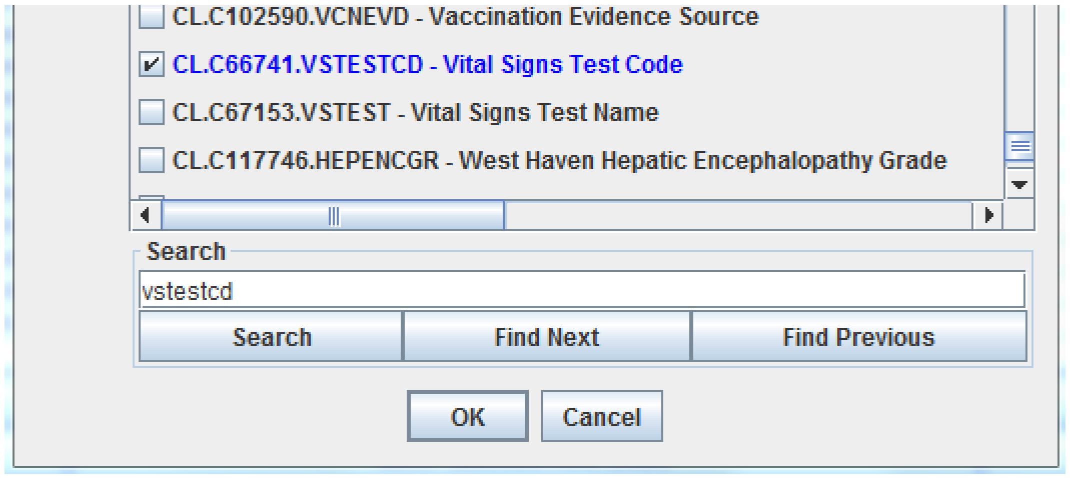

View
Extra
Options
Help
Insert CRF Page Numbers from annotated CRF
Generate ValueList from CodeList
Generate CodeList Subset
Generate CodeList Subset starting from SAS-XPT
Generate CodeListItem-CodeList from EnumeratedItem-Codelist

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



- [ ] CL.MEDDRA - MedDRA Adverse Events Dictionary
- [ ] CL.C115388.SIXMW1TC - 6 Minute Walk Functional Test Test Code
- [ ] CL.C115387.SIXMW1TN - 6 Minute Walk Functional Test Test Name
- [ ] CL.C101805.AIMS01TC - Abnormal Involuntary Movement Scale Questionn
- [ ] CL.C101806.AIMS01TN - Abnormal Involuntary Movement Scale Questionn
- [ ] CL.C66767.ACN - Action Taken with Study Treatment
- [ ] CL.C101865.ACSPCAT - Acute Coronary Syndrome Presentation Category
- [ ] CL.C120985.APCH1TC - Acute Physiology and Chronic Health Evaluation II
- [ ] CL.C120984.APCH1TN - Acute Physiology and Chronic Health Evaluation II
- [ ] CL.C101807.AVL01TC - ADNI Auditory Verbal Learning Functional Test Tes
- [ ] CL.C101808.AVL01TN - ADNI Auditory Verbal Learning Functional Test Tes
- [ ] CL.C66781.AGEU - Age Unit

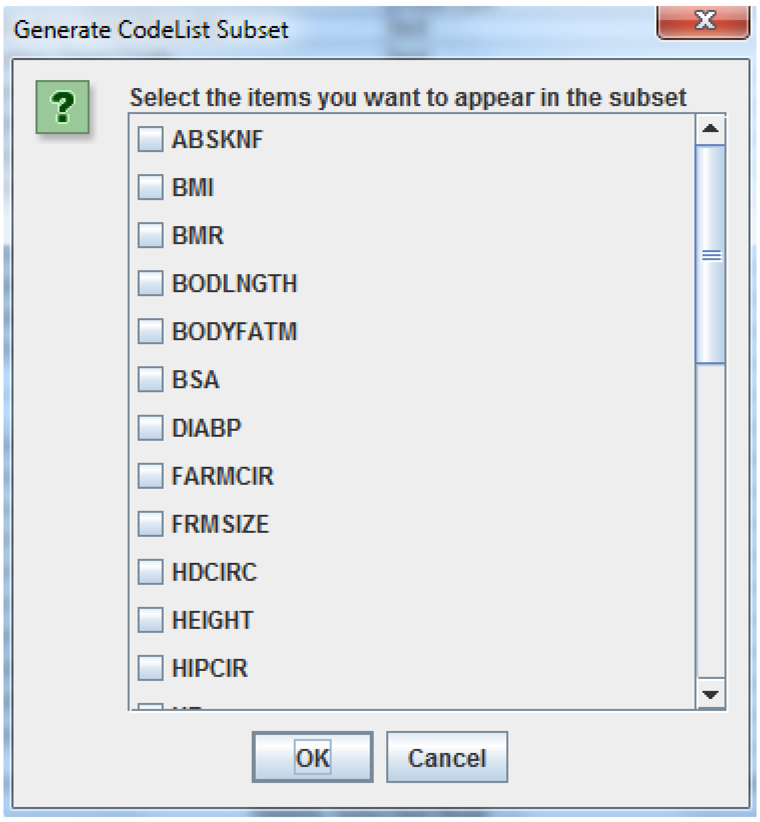
Searching for the right codelist is made ease by the "Search" field and buttons. One can search on the codelist OID (CDISC identifier) or the codelist name. So for "vital signs test code" one easily finds:



- [ ] CL.C102590.VCNEVD - Vaccination Evidence Source
- [x] CL.C66741.VSTESTCD - Vital Signs Test Code
- [ ] CL.C67153.VSTEST - Vital Signs Test Name
- [ ] CL.C117746.HEPENCGR - West Haven Hepatic Encephalopathy Grade

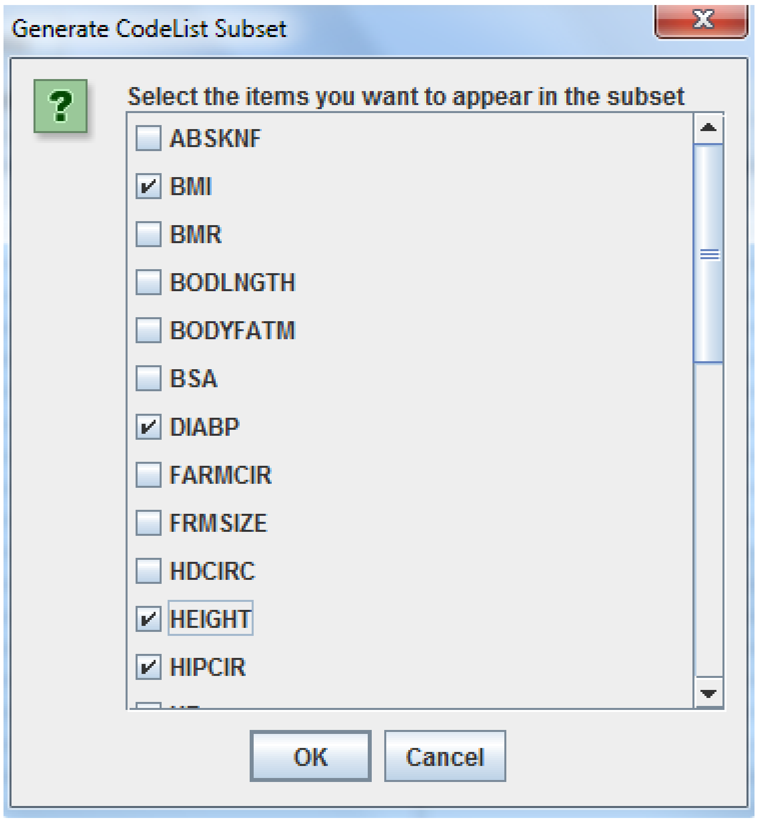
Search: vstestcd

Clicking "OK" leads to a new dialog:



from which the codes to appear in the subset can be selected.
Unfortunately, the CDISC Controlled Terminology Team still publishes codelist with test codes without any information about what the code means, i.e. without "decodes". The latter can only be found by taking the NCI code of the code, and then do a lookup in another codelist with the decodes).
Also, the SDTM and SEND standards requires us to both deliver the test code (--TESTCD) and the test name (--TEST) as two variables (two columns in the dataset), whereas this is completely unnecessary when using modern technologies, as there is a 1:1 relation between test code and test name (for more information, see here). So when creating subsets, we will both need to create a subset for --TESTCD as well as for --TEST. We are however working on an algorithm that enables to generate a --TESTCD subset, and that automatically generates the corresponding subset for --TEST.

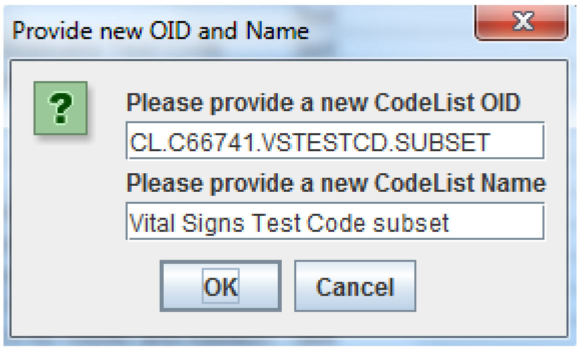
Suppose that we want to create a vital signs test code subset consisting of "DIABP" (diastolic blood pressure"), "SYSBP" (systolic blood pressure", "HEIGHT" (body height), "WEIGHT" (body weight), "BMI" (body mass index) and "HIPCIR" (hip circumference). We tick the boxes for those codes that we want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


Generate CodeList Subset

Select the items you want to appear in the subset

- [ ] ABSKNF
- [x] BMI
- [ ] BMR
- [ ] BODLNGTH
- [ ] BODYFATM
- [ ] BSA
- [x] DIABP
- [ ] FARMCIR
- [ ] FRMSIZE
- [ ] HDCIRC
- [x] HEIGHT
- [x] HIPCIR

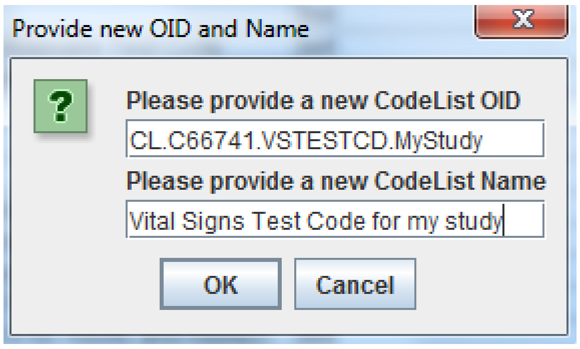
and then click the OK button. The software now proposes a new OID (identifier) and name for the subset codelist:


Provide new OID and Name

Please provide a new CodeList OID: CL.C66741.VSTESTCD.SUBSET

Please provide a new CodeList Name: Vital Signs Test Code subset

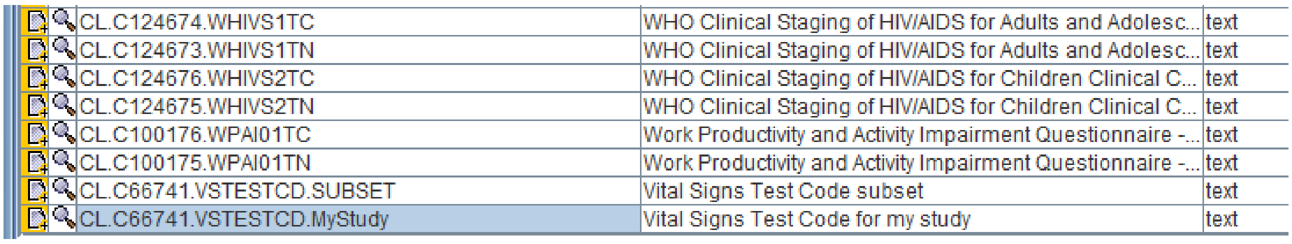
We can of course change these, for example:


Provide new OID and Name

Please provide a new CodeList OID: CL.C66741.VSTESTCD.MyStudy

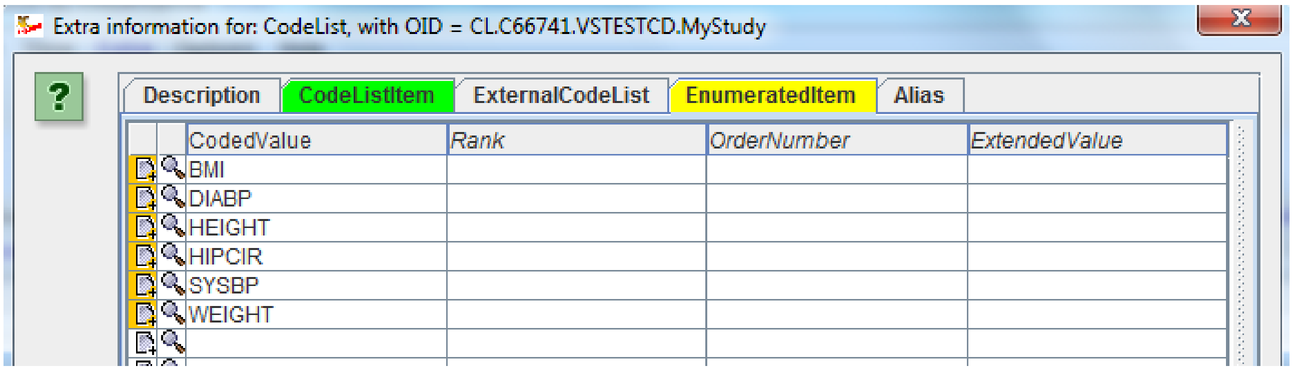
Please provide a new CodeList Name: Vital Signs Test Code for my study

After clicking "OK", the new codelist appears in the list of all codelists:



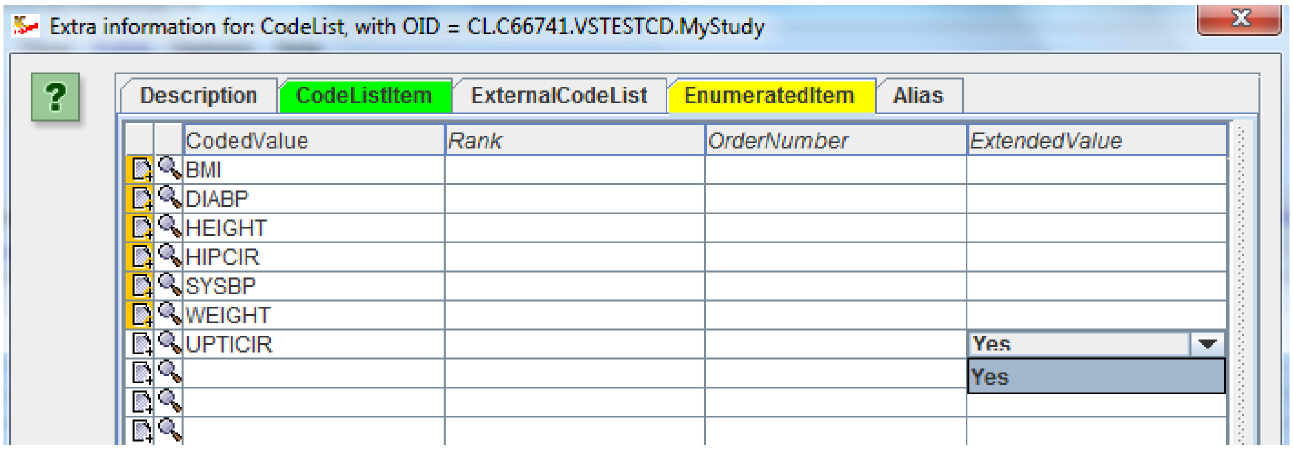
CL.C124674.WHIVS1TC	WHO Clinical Staging of HIV/AIDS for Adults and Adolesc...	text
CL.C124673.WHIVS1TN	WHO Clinical Staging of HIV/AIDS for Adults and Adolesc...	text
CL.C124676.WHIVS2TC	WHO Clinical Staging of HIV/AIDS for Children Clinical C...	text
CL.C124675.WHIVS2TN	WHO Clinical Staging of HIV/AIDS for Children Clinical C...	text
CL.C100176.WPAI01TC	Work Productivity and Activity Impairment Questionnaire -...	text
CL.C100175.WPAI01TN	Work Productivity and Activity Impairment Questionnaire -...	text
CL.C66741.VSTESTCD.SUBSET	Vital Signs Test Code subset	text
CL.C66741.VSTESTCD.MyStudy	Vital Signs Test Code for my study	text

One can then of course still change the subset codelist, add new terms. For example, if one wants to extend this subset vital signs codelist with the term "upper tight circumference", with a code "UPTICIR", just click the "add information" icon ("+"), leading to:


Extra information for: CodeList, with OID = CL.C66741.VSTESTCD.MyStudy
Description
CodeListItem
ExternalCodeList
EnumeratedItem
Alias

CodedValue	Rank	OrderNumber	ExtendedValue
BMI			
DIABP			
HEIGHT			
HIPCIR			
SYSBP			
WEIGHT			

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

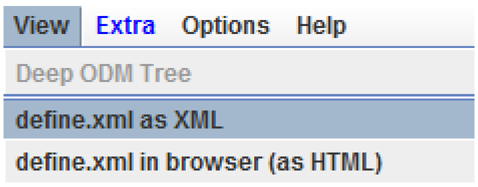

Extra information for: CodeList, with OID = CL.C66741.VSTESTCD.MyStudy
Description
CodeListItem
ExternalCodeList
EnumeratedItem
Alias

CodedValue	Rank	OrderNumber	ExtendedValue
BMI			
DIABP			
HEIGHT			
HIPCIR			
SYSBP			
WEIGHT			
UPTICIR			Yes

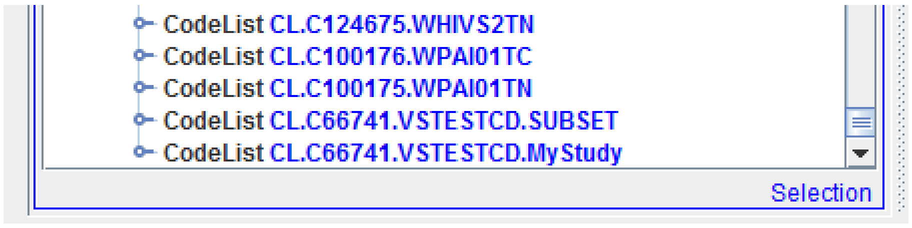
Do however NOT assign an NCI code to this new entry.

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

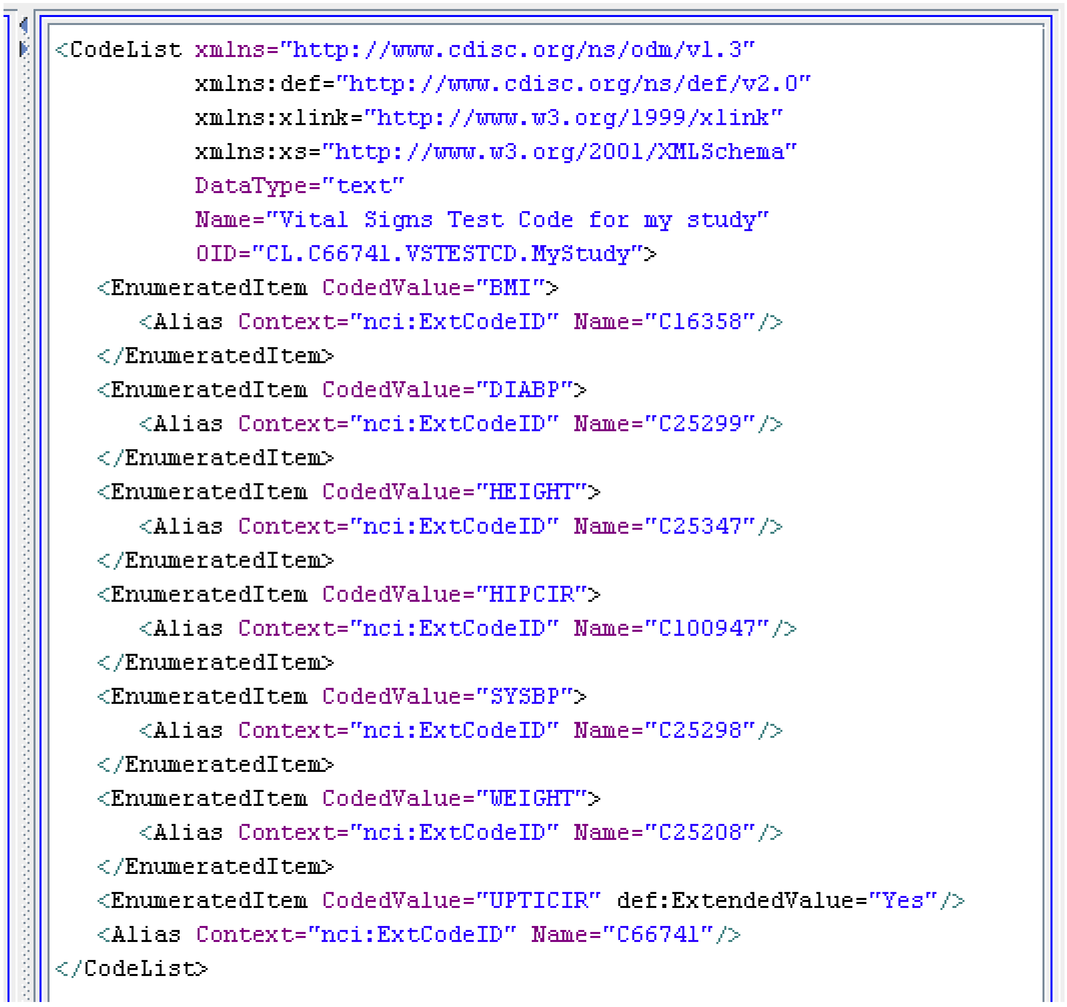
Let us now have a quick look to the generated XML in the define.xml. This is never a bad idea if one wants to understand what one has done.
In order to do so, use the menu "View - Define.xml as XML":



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


CodeList CL.C124675.WHIVS2TN
CodeList CL.C100176.WPAI01TC
CodeList CL.C100175.WPAI01TN
CodeList CL.C66741.VSTESTCD.SUBSET
CodeList CL.C66741.VSTESTCD.MyStudy
Selection

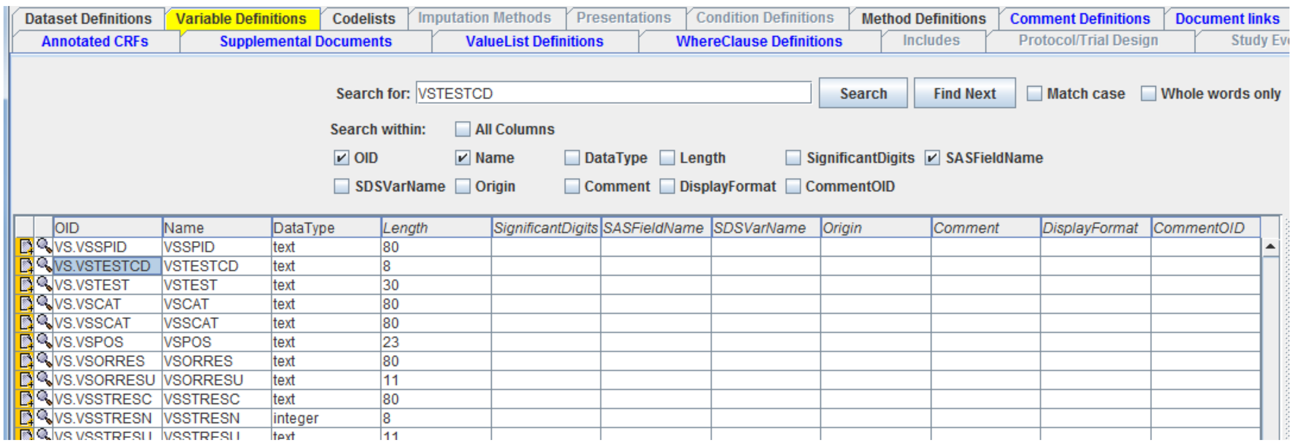
The XML is then shown on the right side:



```
<CodeList xmlns="http://www.cdisc.org/ns/odm/v1.3"
          xmlns:def="http://www.cdisc.org/ns/def/v2.0"
          xmlns:xlink="http://www.w3.org/1999/xlink"
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          DataType="text"
          Name="Vital Signs Test Code for my study"
          OID="CL.C66741.VSTESTCD.MyStudy">
   <EnumeratedItem CodedValue="BMI">
      <Alias Context="nci:ExtCodeID" Name="C16358"/>
   </EnumeratedItem>
   <EnumeratedItem CodedValue="DIABP">
      <Alias Context="nci:ExtCodeID" Name="C25299"/>
   </EnumeratedItem>
   <EnumeratedItem CodedValue="HEIGHT">
      <Alias Context="nci:ExtCodeID" Name="C25347"/>
   </EnumeratedItem>
   <EnumeratedItem CodedValue="HIPCIR">
      <Alias Context="nci:ExtCodeID" Name="C100947"/>
   </EnumeratedItem>
   <EnumeratedItem CodedValue="SYSBP">
      <Alias Context="nci:ExtCodeID" Name="C25298"/>
   </EnumeratedItem>
   <EnumeratedItem CodedValue="WEIGHT">
      <Alias Context="nci:ExtCodeID" Name="C25208"/>
   </EnumeratedItem>
   <EnumeratedItem CodedValue="UPTICIR" def:ExtendedValue="Yes"/>
   <Alias Context="nci:ExtCodeID" Name="C66741"/>
</CodeList>
```

Where we see that there is an NCI code (using the "Alias" element) for each "standard" vital signs test code, whereas the extension code "UPTICIR" does not have an NCI code, but has the def:ExtendedValue="Yes" attribute. We also see that the codelist as a whole has an NCI code (C66741) from the last "Alias" element.
This is also a requirement from the CDISC standards, i.e. that when one subsets a codelist from an existing CDISC codelist, one must keep the NCI code of the codelist itself.

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



Search for: VSTESTCD

☐ Match case ☐ Whole words only

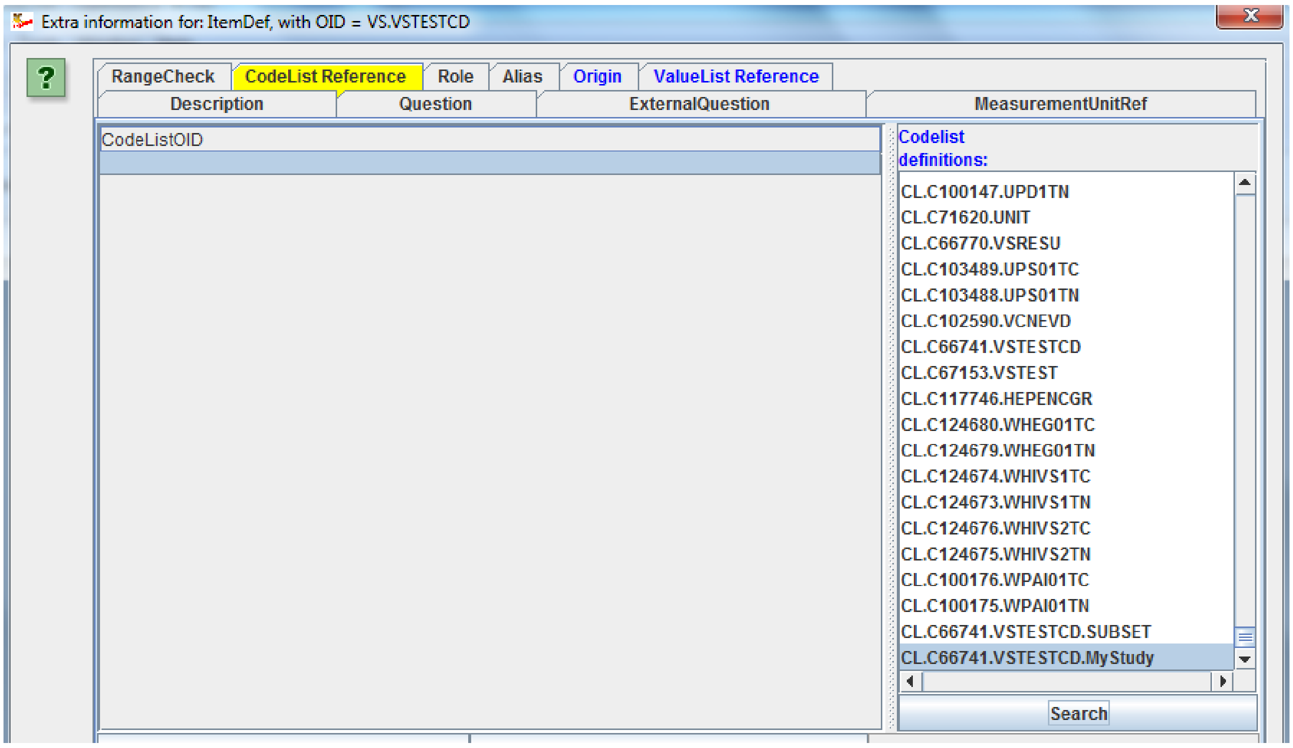
Search within: ☐ All Columns

☑ OID ☑ Name ☐ DataType ☐ Length ☐ SignificantDigits ☑ SASFieldName

☐ SDSVarName ☐ Origin ☐ Comment ☐ DisplayFormat ☐ CommentOID

OID	Name	DataType	Length	SignificantDigits	SASFieldName	SDSVarName	Origin	Comment	DisplayFormat	CommentOID
VS.VSSPID	VSSPID	text	80							
VS.VSTESTCD	VSTESTCD	text	8							
VS.VSTEST	VSTEST	text	30							
VS.VSCAT	VSCAT	text	80							
VS.VSSCAT	VSSCAT	text	80							
VS.VSPOS	VSPOS	text	23							
VS.VSORRES	VSORRES	text	80							
VS.VSORRESU	VSORRESU	text	11							
VS.VSSTRESC	VSSTRESC	text	80							
VS.VSSTRESN	VSSTRESN	integer	8							
VS.VSSTRESU	VSSTRESU	text	11							

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



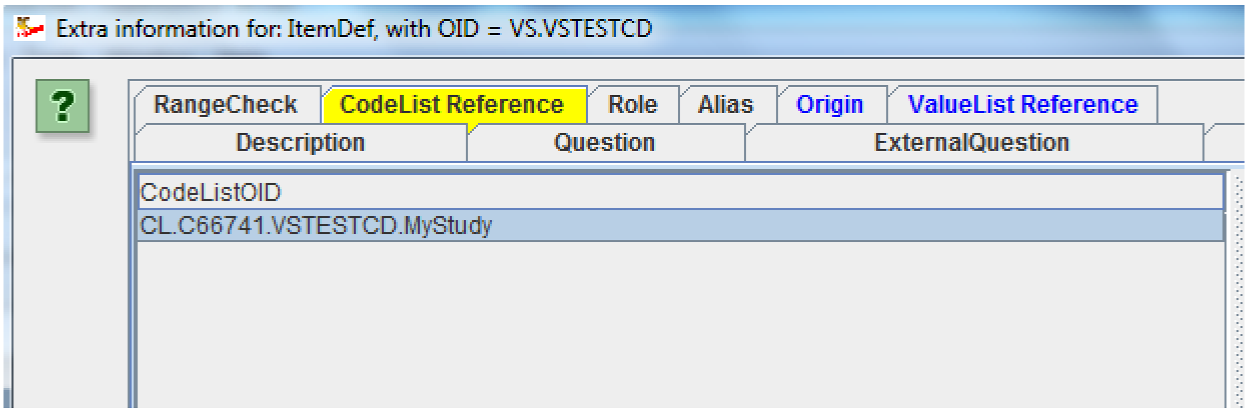
Extra information for: ItemDef, with OID = VS.VSTESTCD

CodeListOID

Codelist definitions:

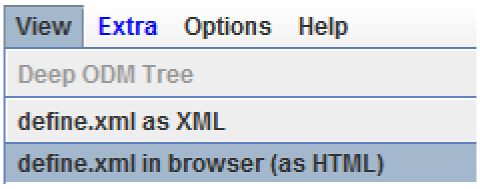
CL.C100147.UPD1TN
CL.C71620.UNIT
CL.C66770.VSRESU
CL.C103489.UPS01TC
CL.C103488.UPS01TN
CL.C102590.VCNEVD
CL.C66741.VSTESTCD
CL.C67153.VSTEST
CL.C117746.HEPENCGR
CL.C124680.WHEG01TC
CL.C124679.WHEG01TN
CL.C124674.WHIVS1TC
CL.C124673.WHIVS1TN
CL.C124676.WHIVS2TC
CL.C124675.WHIVS2TN
CL.C100176.WPAI01TC
CL.C100175.WPAI01TN
CL.C66741.VSTESTCD.SUBSET
CL.C66741.VSTESTCD.MyStudy

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

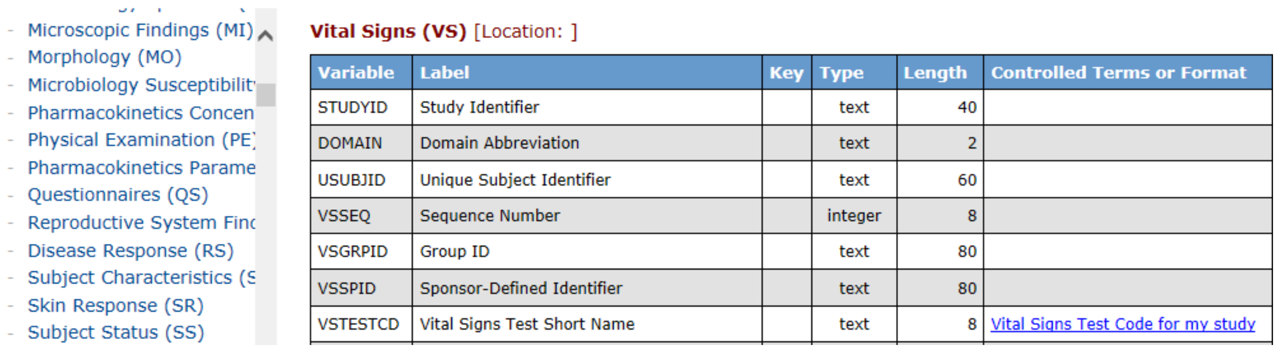

Extra information for: ItemDef, with OID = VS.VSTESTCD

CodeListOID
CL.C66741.VSTESTCD.MyStudy

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



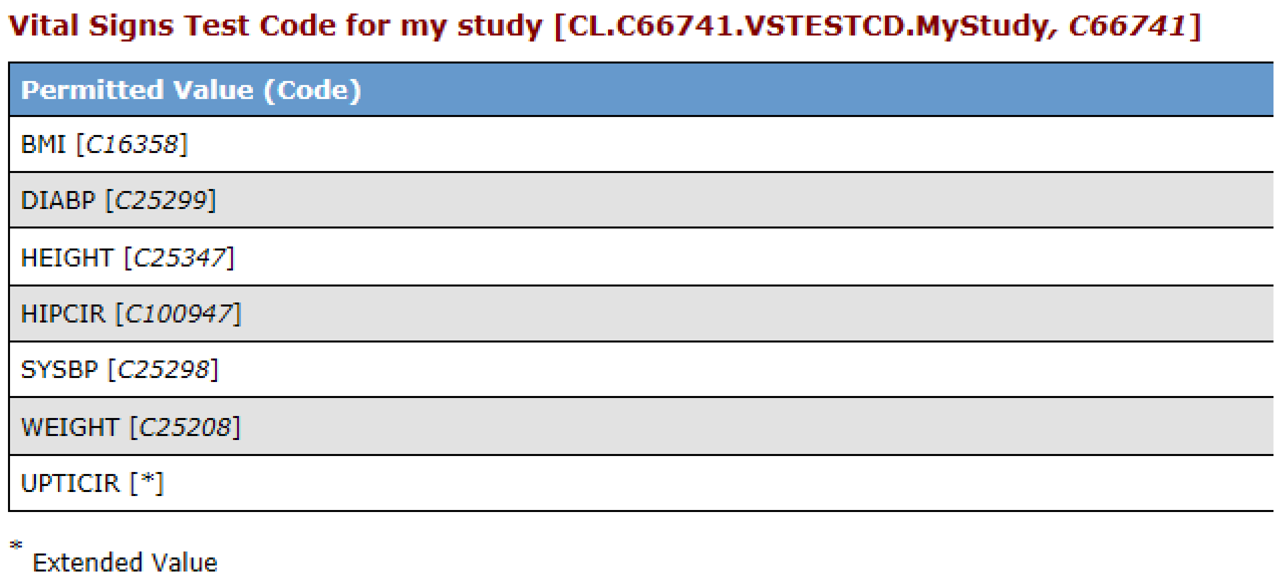
leading to:


- Microscopic Findings (MI)
- Morphology (MO)
- Microbiology Susceptibilit
- Pharmacokinetics Concen
- Physical Examination (PE
- Pharmacokinetics Parame
- Questionnaires (QS)
- Reproductive System Finc
- Disease Response (RS)
- Subject Characteristics (S
- Skin Response (SR)
- Subject Status (SS)

Vital Signs (VS) [Location:]

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

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


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

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

* Extended Value

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

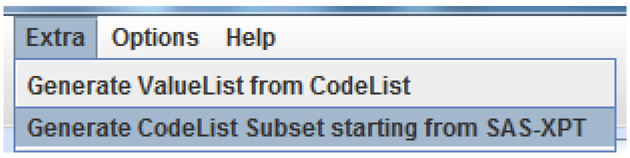
We now still need to subset the corresponding subset codelist for "VSTEST" (vital signs test name). As already stated, we will soon develop the algorithm and feature to subset a --TESTCD codelist and automatically also subset the corresponding --TEST codelist.

Adding and subsetting codelists from SAS-XPT files

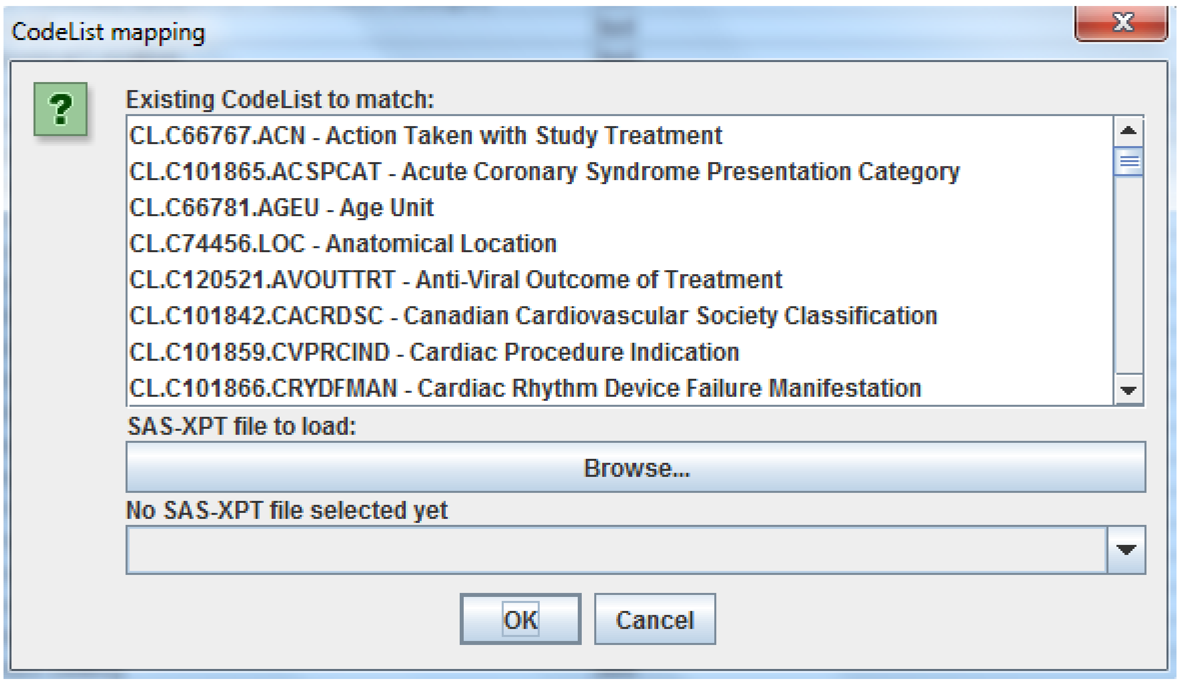
When generating a prototype define.xml, the system does NOT (unlike other Excel-based systems) try to automatically generate codelists from the data in the SAS-XPT files.The reason for this is that such an automated generation usually leads to disaster: often the codelists do not comply to the CDISC rules, and many of the codelists terms will be automatically assigned as a codelist extension (i.e. sponsor defined term). Most users then just accept that, instead of trying to bring the data themselves into compliance with CDISC controlled terminology.

Our software takes another approach. At startup, the software has already loaded CDISC controlled terminology, and one is then encouraged to subset the CDISC codelists so that they really represent the data in the data files. If a datapoint that is coded is found not to be in the by CDISC published codelist, then the user should either go back to the mappings for that datapoint (and similar ones) and correct the mapping, OR actively decide that the term is indeed a sponsor defined term, and assign it as a "codelist extended" term. As such, the decision is taken by the user.

In order to create such a subset codelist, starting from the contents of an existing SAS-XPT file, use the menu "Extra - Generate CodeList Subset starting from SAS-XPT":


Extra
Options
Help
Generate ValueList from CodeList
Generate CodeList Subset starting from SAS-XPT

The following dialog is presented:


CodeList mapping

Existing CodeList to match:

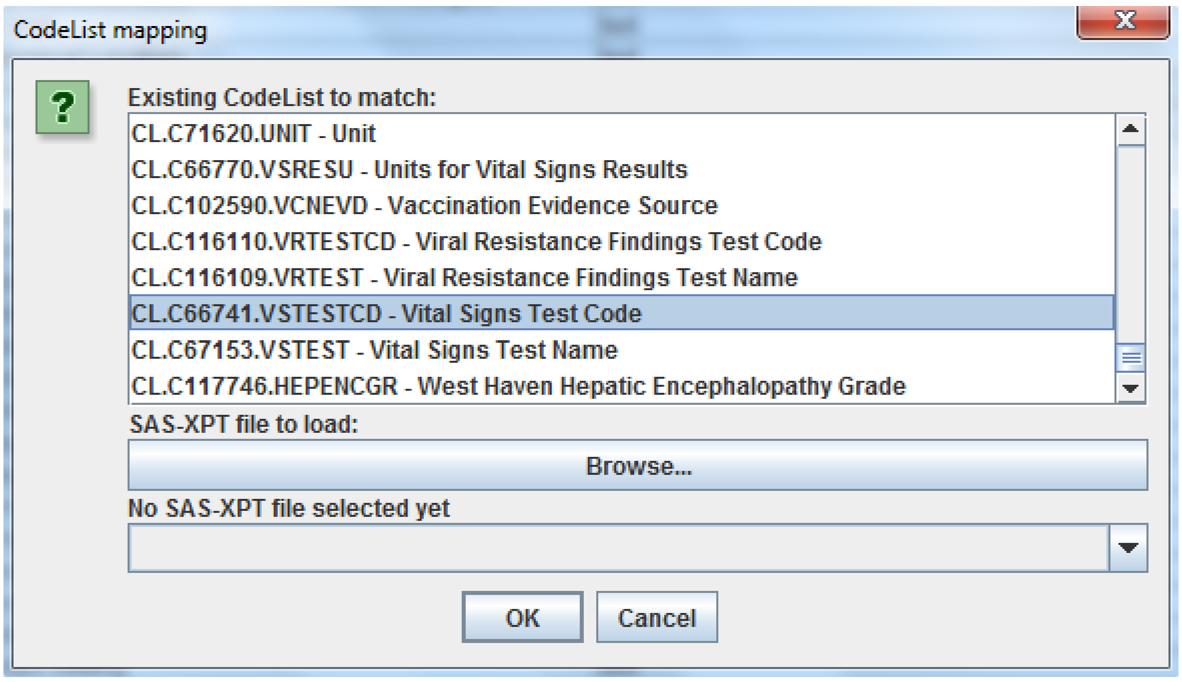
CL.C66767.ACN - Action Taken with Study Treatment
CL.C101865.ACSPCAT - Acute Coronary Syndrome Presentation Category
CL.C66781.AGEU - Age Unit
CL.C74456.LOC - Anatomical Location
CL.C120521.AVOUTTRT - Anti-Viral Outcome of Treatment
CL.C101842.CACRDSC - Canadian Cardiovascular Society Classification
CL.C101859.CVPRCIND - Cardiac Procedure Indication
CL.C101866.CRYDFMAN - Cardiac Rhythm Device Failure Manifestation

SAS-XPT file to load:

No SAS-XPT file selected yet

Now select a codelist you wish to subset.
In our example, we will do so for the VSTESTCD codelist:

CL.C66741.VSTESTCD - Vital Signs Test Code


CodeList mapping

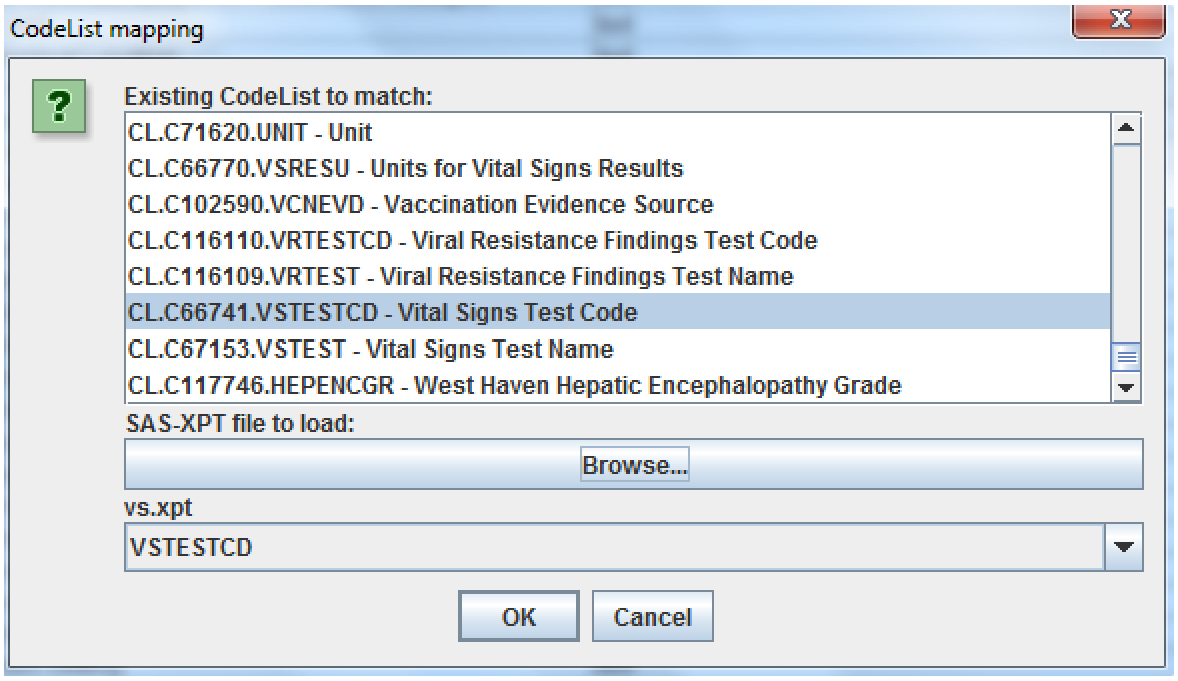
Existing CodeList to match:

CL.C71620.UNIT - Unit
CL.C66770.VSRESU - Units for Vital Signs Results
CL.C102590.VCNEVD - Vaccination Evidence Source
CL.C116110.VRTESTCD - Viral Resistance Findings Test Code
CL.C116109.VRTEST - Viral Resistance Findings Test Name
CL.C66741.VSTESTCD - Vital Signs Test Code
CL.C67153.VSTEST - Vital Signs Test Name
CL.C117746.HEPENCGR - West Haven Hepatic Encephalopathy Grade

SAS-XPT file to load:

No SAS-XPT file selected yet

Then select the SAS-XPT file you want to get the values from. Obviously this is the vs.xpt file in this case. Once the file selected, the system analyzes its metadata (this takes a few seconds), and makes a proposal for the variable for which the unique values from the SAS-XPT file will be retrieved:


CodeList mapping

Existing CodeList to match:

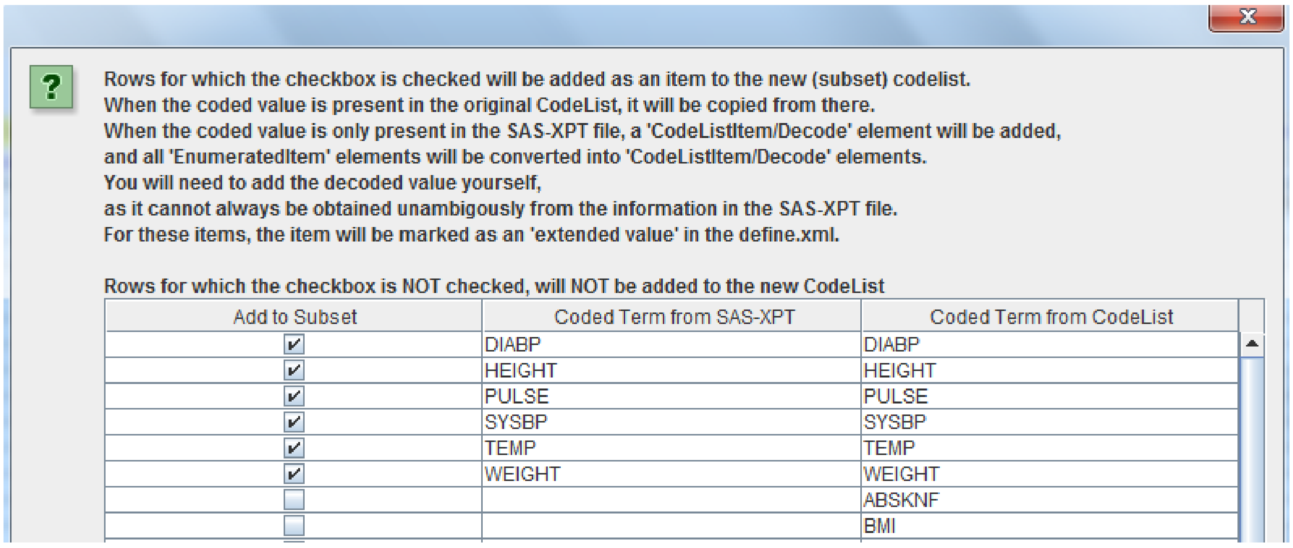
CL.C71620.UNIT - Unit
CL.C66770.VSRESU - Units for Vital Signs Results
CL.C102590.VCNEVD - Vaccination Evidence Source
CL.C116110.VRTESTCD - Viral Resistance Findings Test Code
CL.C116109.VRTEST - Viral Resistance Findings Test Name
CL.C66741.VSTESTCD - Vital Signs Test Code
CL.C67153.VSTEST - Vital Signs Test Name
CL.C117746.HEPENCGR - West Haven Hepatic Encephalopathy Grade

SAS-XPT file to load:

vs.xpt

VSTESTCD

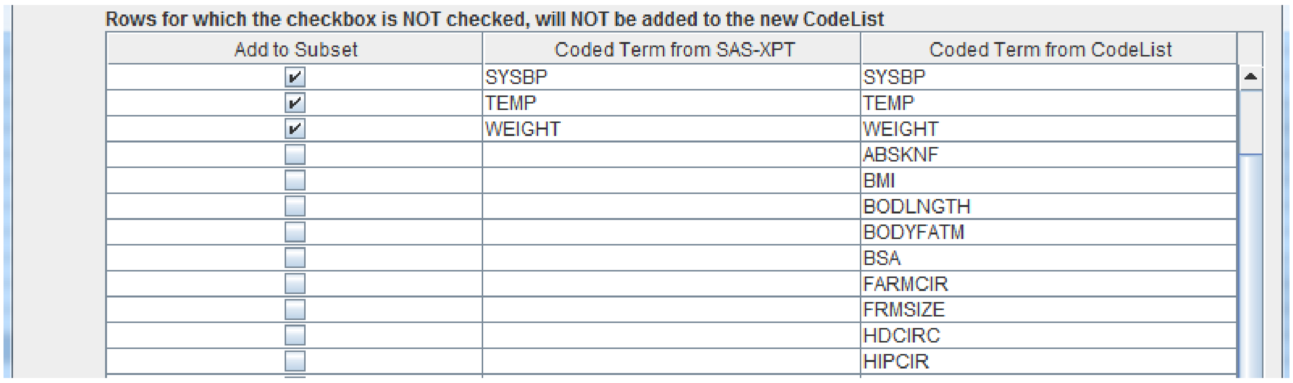
When clicking "OK", the system takes all the VSTESTCD values from the SAS-XPT "vs.xpt" file and takes all the unique ones. It then tries to match them to the values from the codelist CL.C66741.VSTESTCD "Laboratory Test Code". This leads to a new dialog:


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.
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
☑	DIABP	DIABP
☑	HEIGHT	HEIGHT
☑	PULSE	PULSE
☑	SYSBP	SYSBP
☑	TEMP	TEMP
☑	WEIGHT	WEIGHT
☐		ABSKNF
☐		BMI

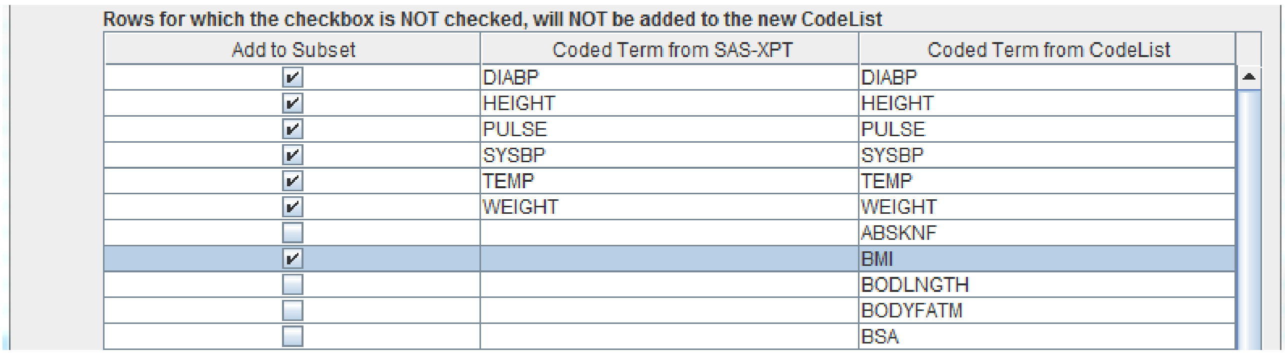
The first column contains a checkbox, allow the user to decide whether the coded term should be included in the subset.
The second column contains the term as found in the SAS-XPT file and the third column the coded term in the codelist. So if the term is present in as well the second as third column, there is a match between the SAS-XPT value and the already available codelist. In such a case, it is recommended to add the coded term to the subset codelist (so the checkbox is already checked).
When scrolling down:


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

Add to Subset	Coded Term from SAS-XPT	Coded Term from CodeList
☑	SYSBP	SYSBP
☑	TEMP	TEMP
☑	WEIGHT	WEIGHT
☐		ABSKNF
☐		BMI
☐		BODLNGTH
☐		BODYFATM
☐		BSA
☐		FARMCIR
☐		FRMSIZE
☐		HDCIRC
☐		HIPCIR

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

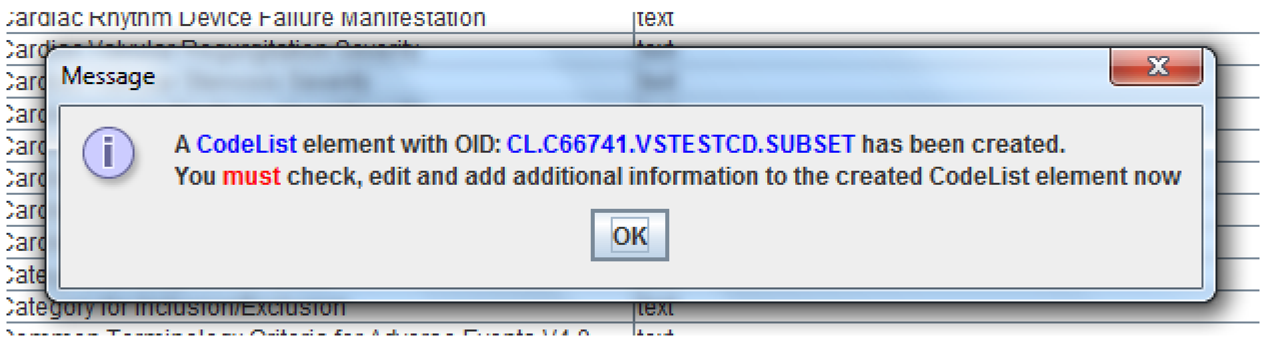
Suppose however that there was a field in the CRF for "BMI", but that there are no data for it (yet) in the SAS-XPT dataset. In such a case "BMI" must be added to the codelist, as it was on the CRF. So we do check the checkbox for "BMI":


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

Add to Subset	Coded Term from SAS-XPT	Coded Term from CodeList
✓	DIABP	DIABP
✓	HEIGHT	HEIGHT
✓	PULSE	PULSE
✓	SYSBP	SYSBP
✓	TEMP	TEMP
✓	WEIGHT	WEIGHT
☐		ABSKNF
✓		BMI
☐		BODLNGTH
☐		BODYFATM
☐		BSA

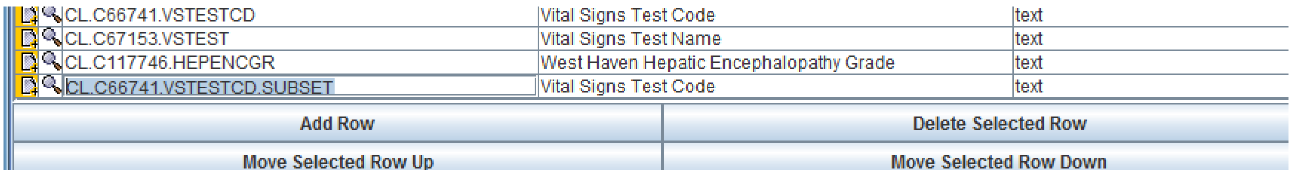
Clicking "OK" will then create a subset of the codelist. It automatically assigns the OID " CL.C66741.VSTESTCD.SUBSET". You will of course later be able to change the OID. The original codelist name is however retained.

This results in:


Message

A CodeList element with OID: CL.C66741.VSTESTCD.SUBSET has been created.
You must check, edit and add additional information to the created CodeList element now

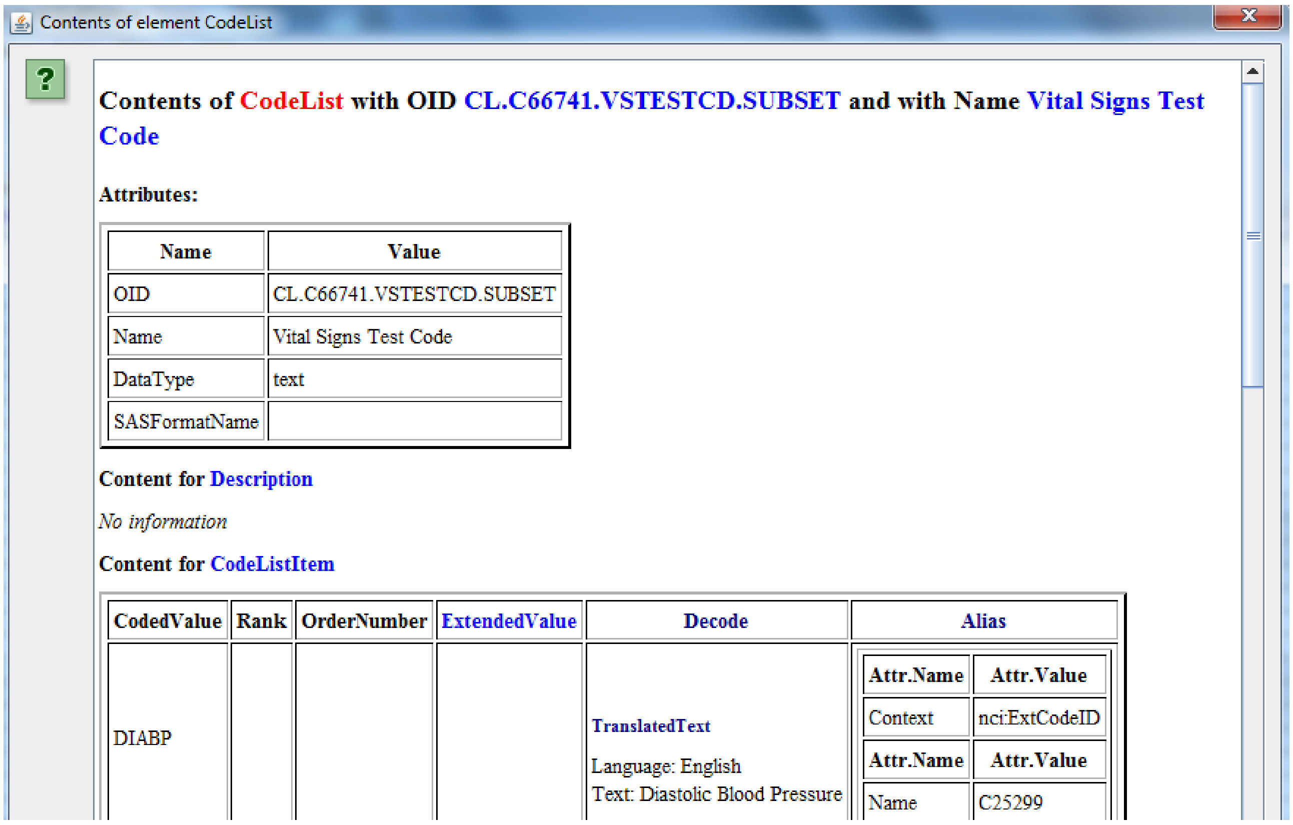
and when clicking "OK" again, the table with all the codelists will be displayed, containing a new row for the subset codelist:



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

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

Clicking the magnifying glass icon gives an overview:



Contents of element CodeList

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

Attributes:

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

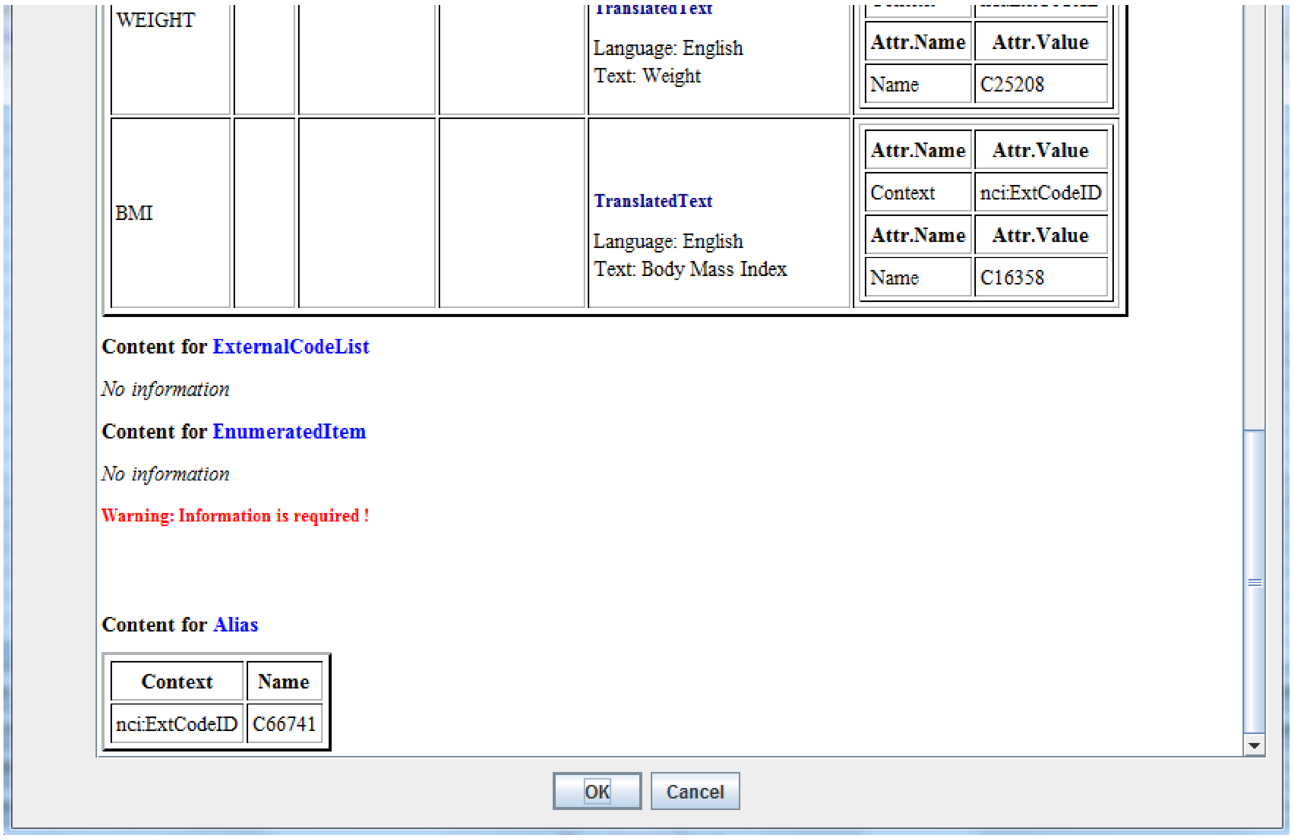
Content for Description

No information

Content for CodeListItem

CodedValue	Rank	OrderNumber	ExtendedValue	Decode	Alias
DIABP				TranslatedText Language: English Text: Diastolic Blood Pressure	Attr.Name / Attr.Value: Context / nci:ExtCodeID Attr.Name / Attr.Value: Name / C25299

also containing "BMI":



CodedValue	Rank	OrderNumber	ExtendedValue	Decode	Alias
WEIGHT				TranslatedText Language: English Text: Weight	Attr.Name / Attr.Value: Name / C25208
BMI				TranslatedText Language: English Text: Body Mass Index	Attr.Name / Attr.Value: Context / nci:ExtCodeID Attr.Name / Attr.Value: Name / C16358

Content for ExternalCodeList

No information

Content for EnumeratedItem

No information

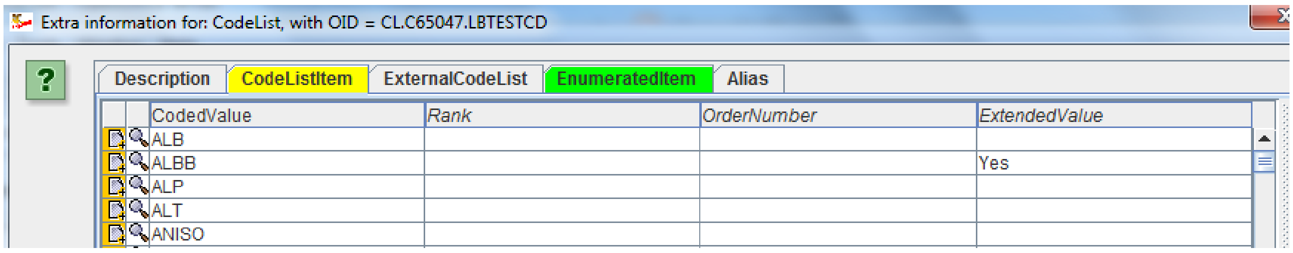
Warning: Information is required !

Content for Alias

Context	Name
nci:ExtCodeID	C66741

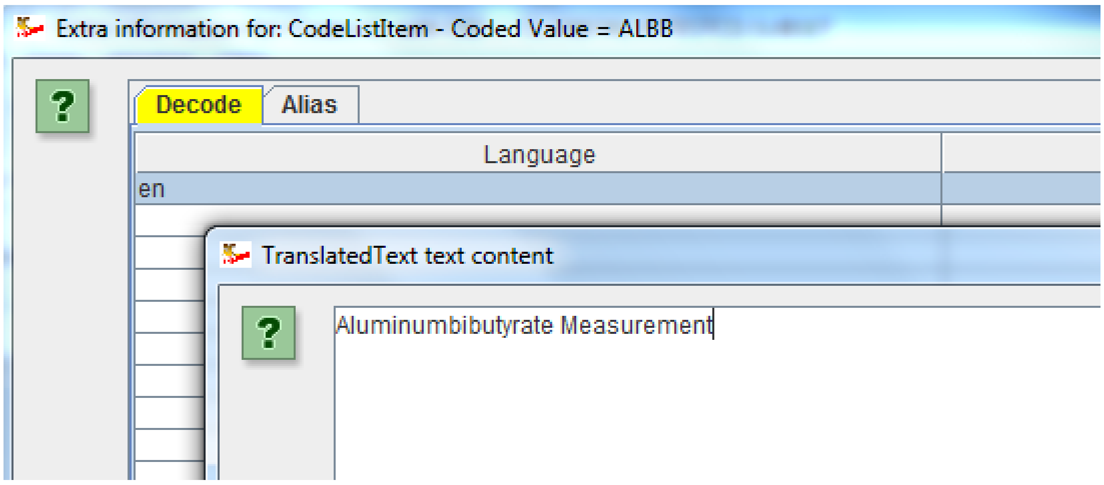
Remark that also the "Alias" child elements were copied from the original codelist.

Suppose now that we also subsetted the "LBTESTCD" codelist, and that the SAS-XPT file contains an LBTESTCD value "ALBB". It might be that this is a mapping order: in that case one would need to re-generate the lb.xpt file.
If however it was not an error, but "ALBB" corresponds to "Aluminumbibutyrate Measurement", one would find the following list of coded items in the codelist:


Extra information for: CodeList, with OID = CL.C65047.LBTESTCD
?
Description
CodeListItem
ExternalCodeList
EnumeratedItem
Alias

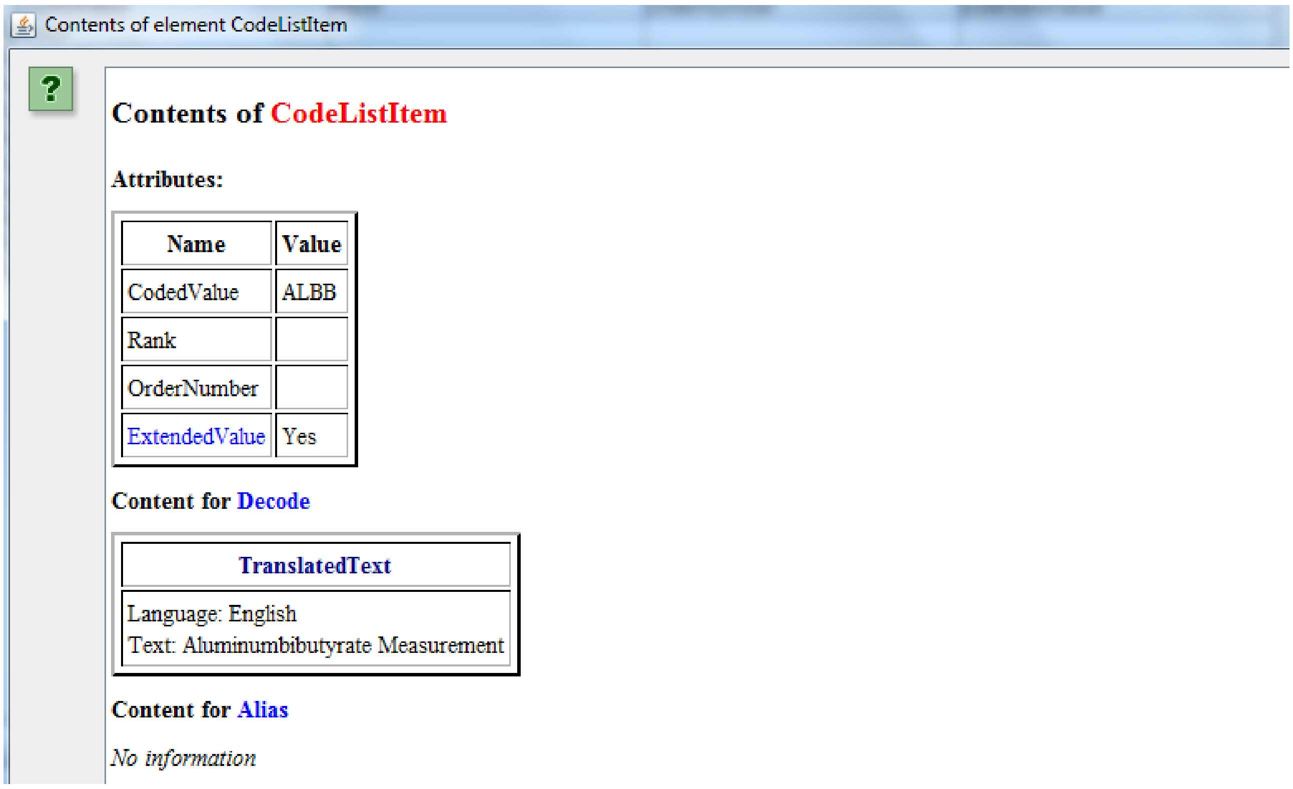
CodedValue	Rank	OrderNumber	ExtendedValue
ALB			
ALBB			Yes
ALP			
ALT			
ANISO			

Remark that it is automatically marked as "extended".
One would still need to add the "decoded value". This can be done using the "+" icon, and adding the text to the "Decode" element:


Extra information for: CodeListItem - Coded Value = ALBB
?
Decode
Alias
Language
en
TranslatedText text content
?

Aluminumbibutyrate Measurement

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


Contents of element CodeListItem

Contents of CodeListItem

Attributes:

Name	Value
CodedValue	ALBB
Rank	
OrderNumber	
ExtendedValue	Yes

Content for Decode

TranslatedText
Language: English Text: Aluminumbibutyrate Measurement

Content for Alias

No information

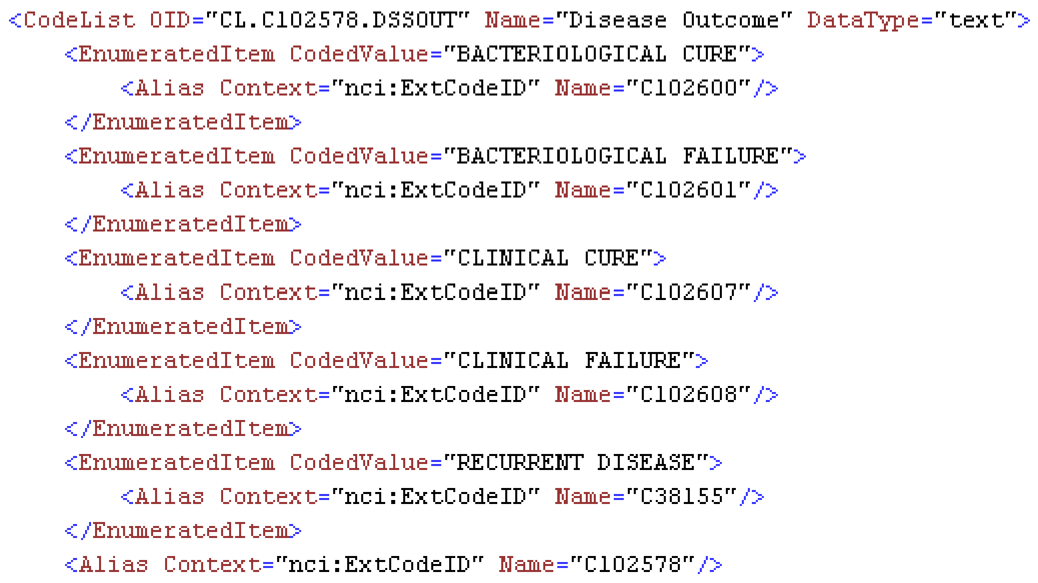
Remark the "ExtendedValue" with value "Yes" in the table of attributes, and that there is no "Alias" element with "nci:ExtCodeID", as this term is not in the CDISC controlled terminology.
You can of course always do a "new term request" to the CDISC "controlled terminology team", but this may take a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

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

Looks good isn't it?

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

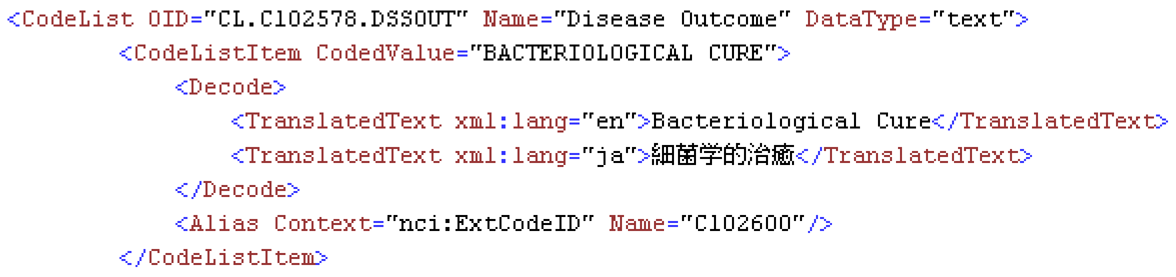
Many of the CDISC controlled terminology is published as an "Enumerated" CodeList, i.e. all values are published without description and without any translations into other languages. For example:



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

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

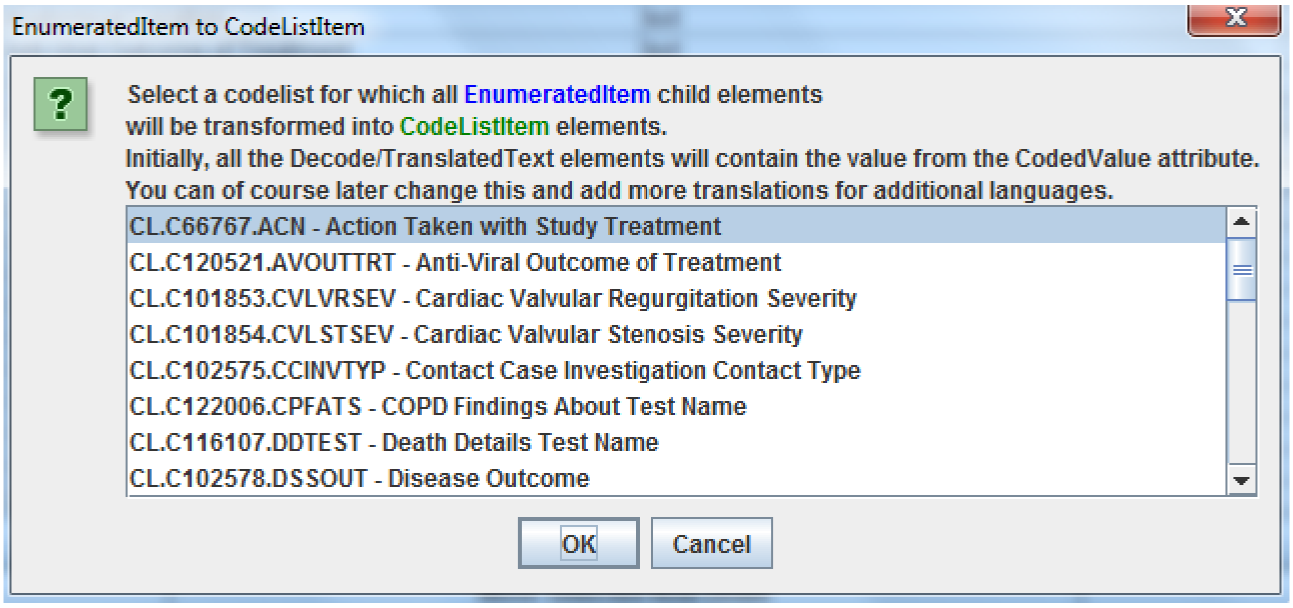
So we will want to transform such an "Enumerated" CodeList into a classic CodeList with a "Decode" and "TranslatedText" elements, like in:



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

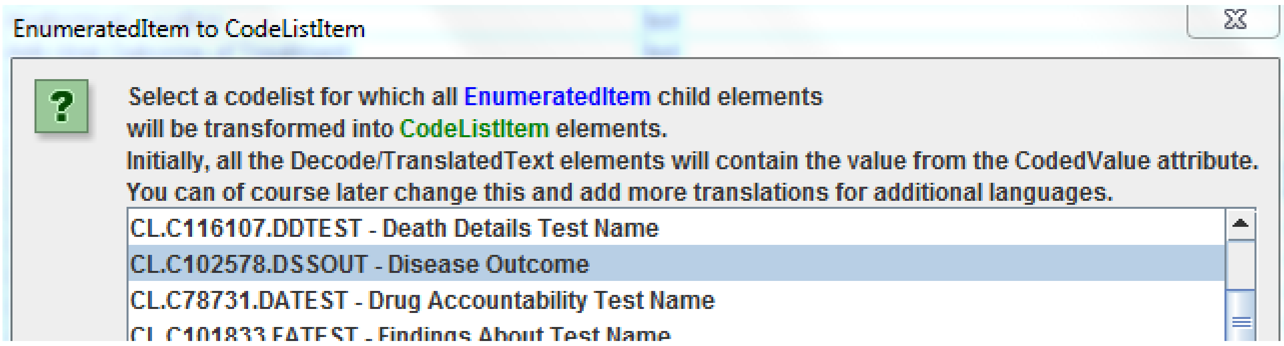
Our system has a feature to do such a transformation automatically. The result will initially always have the value for Decode/TranslatedText being the same value as for "CodedValue", but the user will then be able to change this and add translations for additional languages.

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


EnumeratedItem to CodeListItem

Select a codelist for which all EnumeratedItem child elements
will be transformed into CodeListItem elements.
Initially, all the Decode/TranslatedText elements will contain the value from the CodedValue attribute.
You can of course later change this and add more translations for additional languages.

CL.C66767.ACN - Action Taken with Study Treatment
CL.C120521.AVOUTTRT - Anti-Viral Outcome of Treatment
CL.C101853.CVLVRSEV - Cardiac Valvular Regurgitation Severity
CL.C101854.CVLSTSEV - Cardiac Valvular Stenosis Severity
CL.C102575.CCINVTYP - Contact Case Investigation Contact Type
CL.C122006.CPFATS - COPD Findings About Test Name
CL.C116107.DDTEST - Death Details Test Name
CL.C102578.DSSOUT - Disease Outcome


EnumeratedItem to CodeListItem

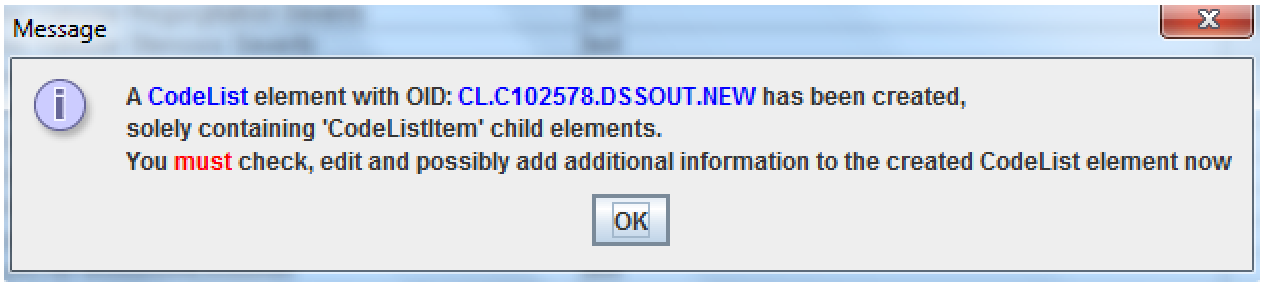
Select a codelist for which all EnumeratedItem child elements
will be transformed into CodeListItem elements.
Initially, all the Decode/TranslatedText elements will contain the value from the CodedValue attribute.
You can of course later change this and add more translations for additional languages.

CL.C116107.DDTEST - Death Details Test Name
CL.C102578.DSSOUT - Disease Outcome
CL.C78731.DATEST - Drug Accountability Test Name
CL.C101833.FATEST - Findings About Test Name

TODO: add button "Show CodeList Details"

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

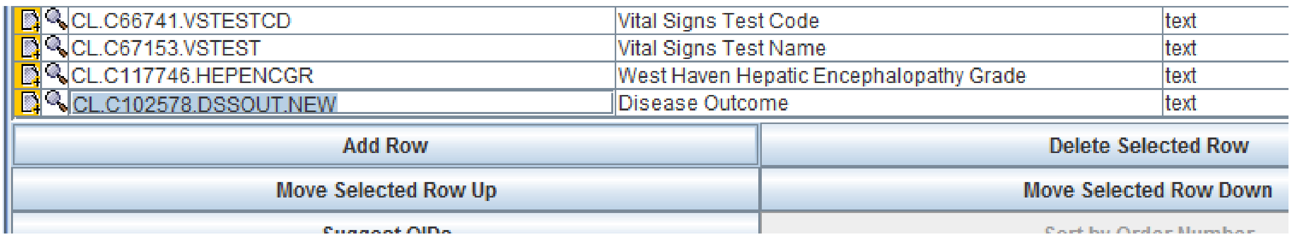
After clicking the "OK" button, the following message is displayed:


Message

A CodeList element with OID: CL.C102578.DSSOUT.NEW has been created,
solely containing 'CodeListItem' child elements.
You must check, edit and possibly add additional information to the created CodeList element now

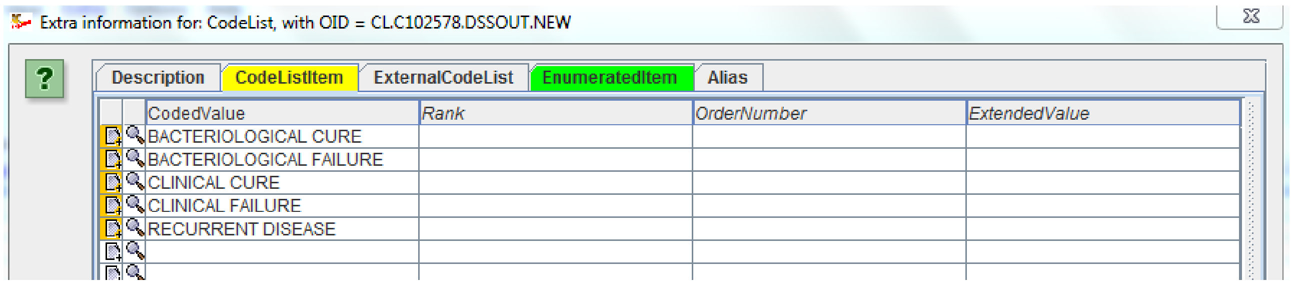
and the codelist is added at the bottom to the table with the codelists.
If you do not want the old codelist anymore, but only want to work with the new one, you can delete the old one and rename the new one.

The table is displayed automatically:



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

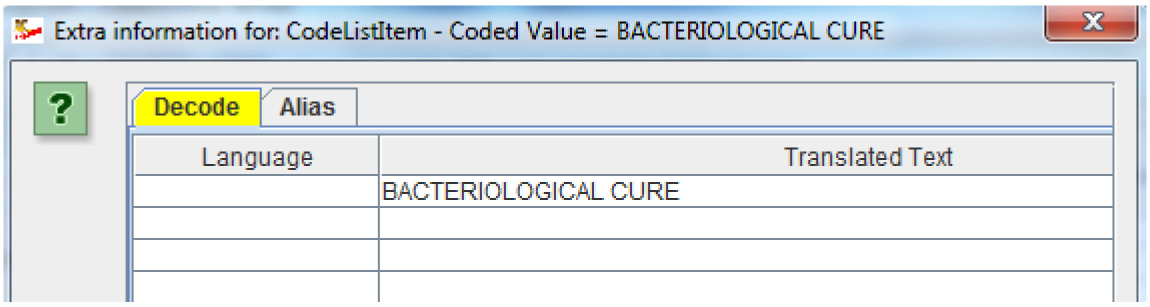
When clicking the "+" icon, one sees that the child elements are "CodeListItem" elements now instead of "EnumeratedItem" items:



Extra information for: CodeList, with OID = CL.C102578.DSSOUT.NEW

CodedValue	Rank	OrderNumber	ExtendedValue
BACTERIOLOGICAL CURE			
BACTERIOLOGICAL FAILURE			
CLINICAL CURE			
CLINICAL FAILURE			
RECURRENT DISEASE			

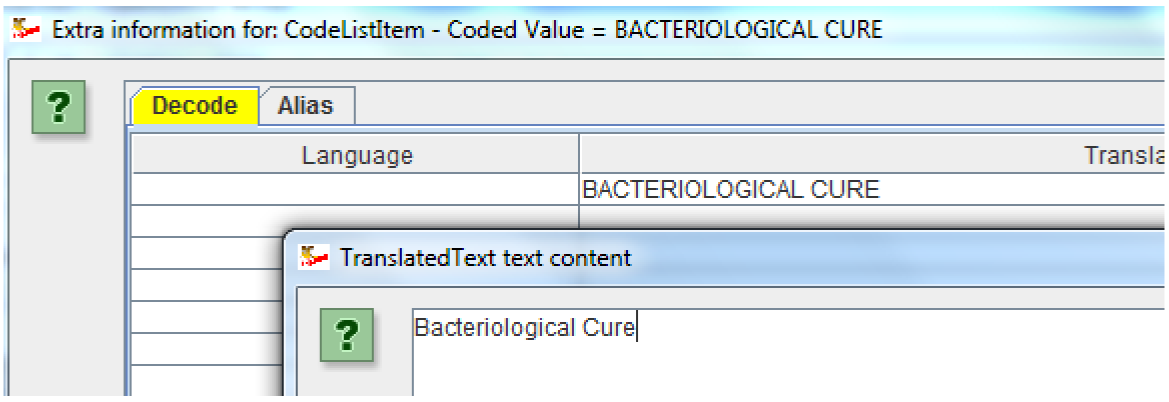
and one can start adding translations into other languages. For example, for "BACTERIOLOGICAL CURE":



Extra information for: CodeListItem - Coded Value = BACTERIOLOGICAL CURE

Language	Translated Text
	BACTERIOLOGICAL CURE

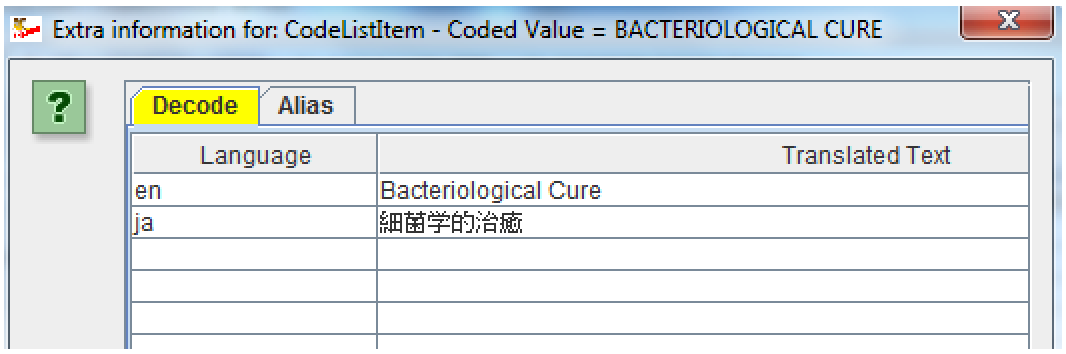
The default is that the term has just been copied from "CodedValue" into a "Decode/TranslatedText". One can now easily change this into:



Extra information for: CodeListItem - Coded Value = BACTERIOLOGICAL CURE

Bacteriological Cure

and e.g. add the Japanese translation, leading to:


Extra information for: CodeListItem - Coded Value = BACTERIOLOGICAL CURE

Language	Translated Text
en	Bacteriological Cure
ja	細菌学的治癒

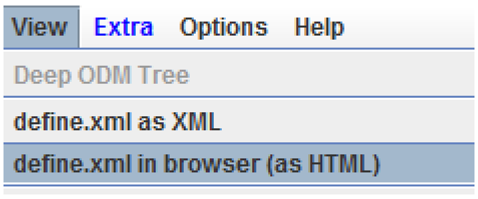
The "CodedValue" (as stored in a database) however remains "BACTERIOLOGICAL CURE".

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

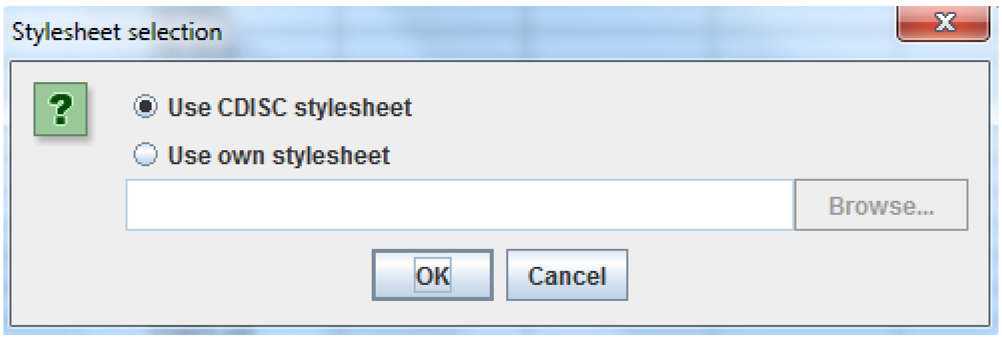
Many people (unfortunately) know define.xml only from what they see in the browser. What is seen in the browser is however only a particular view on the information in the define.xml provided by a stylesheet. When submitting the define.xml to the FDA or the PMDA, it is the sponsors duty[11] to also provide a stylesheet so that the reviewers can see the information in a user-friendly way.

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

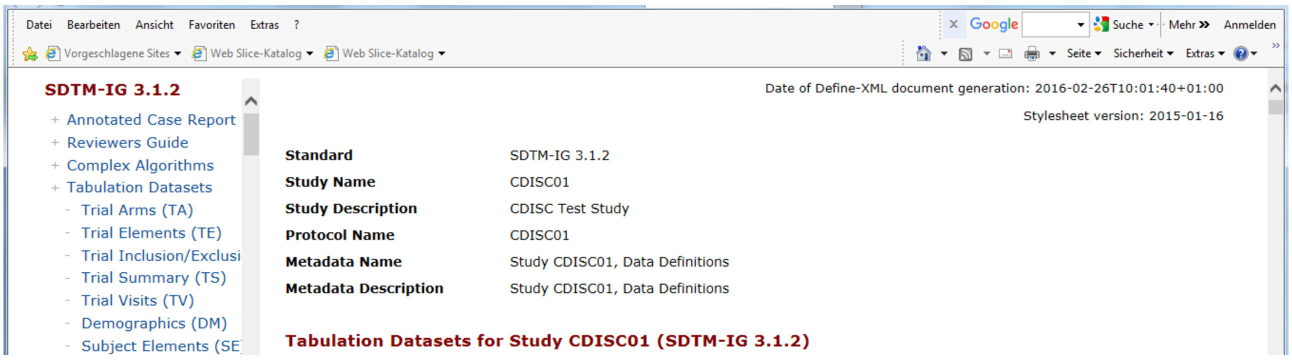
Also from within our software, one can already obtain a view of the developed define.xml in the browser by applying the stylesheet. To do so, use the menu "View - define.xml in browser":



You will then be asked whether you want to use the default CDISC stylesheet or your own one:



If you select "Use own stylesheet", you will need the "Browse" button to select a stylesheet file from your local file system.
Then click OK, and the define.xml will be displayed (as HTML of course) in your default[12] (favorite) browser. For example:



Date of Define-XML document generation: 2016-02-26T10:01:40+01:00

Stylesheet version: 2015-01-16

Standard	SDTM-IG 3.1.2
Study Name	CDISC01
Study Description	CDISC Test Study
Protocol Name	CDISC01
Metadata Name	Study CDISC01, Data Definitions
Metadata Description	Study CDISC01, Data Definitions

Tabulation Datasets for Study CDISC01 (SDTM-IG 3.1.2)

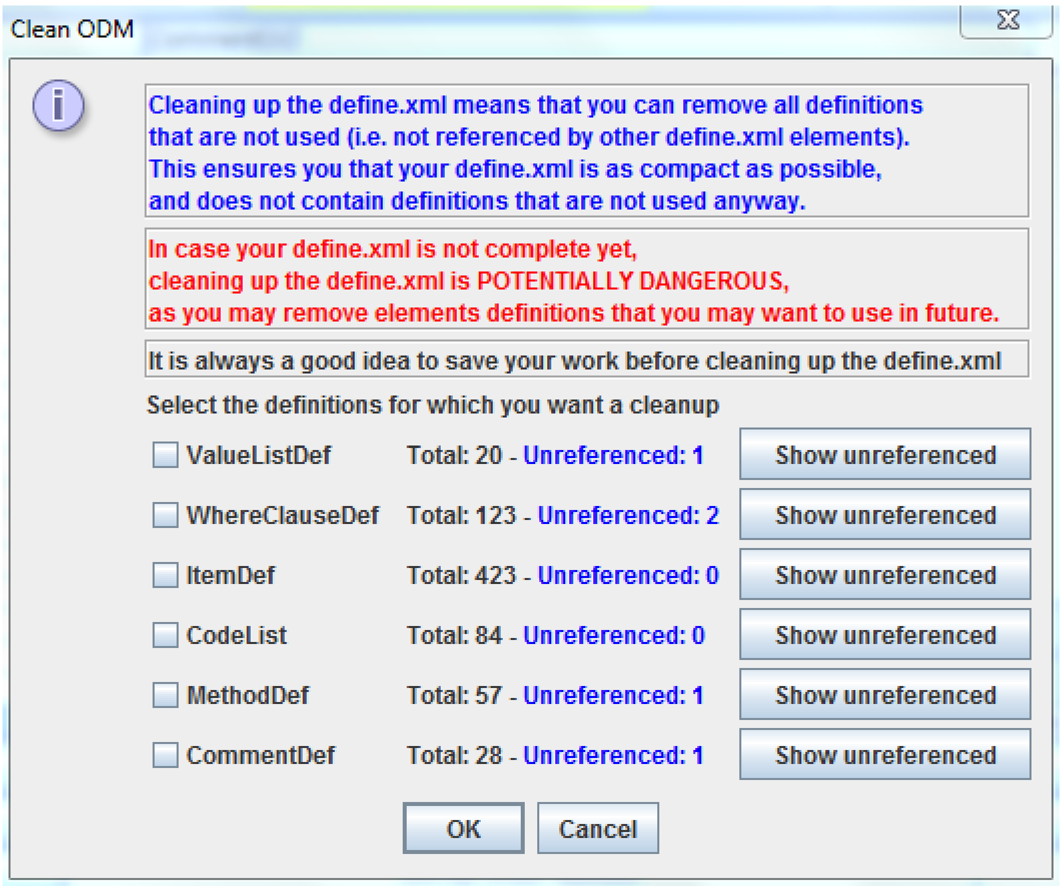
Cleaning up your define.xml

[11]The FDA would better forbid that the stylesheet is provided by the sponsor, and use their own one. This would ensure that all studies are displayed in the same, sponsor-independent way.
[12]The system recognizes what your default browser is.

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

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

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


Clean ODM

Cleaning up the define.xml means that you can remove all definitions that are not used (i.e. not referenced by other define.xml elements).
This ensures you that your define.xml is as compact as possible, and does not contain definitions that are not used anyway.

In case your define.xml is not complete yet, cleaning up the define.xml is POTENTIALLY DANGEROUS, as you may remove elements definitions that you may want to use in future.

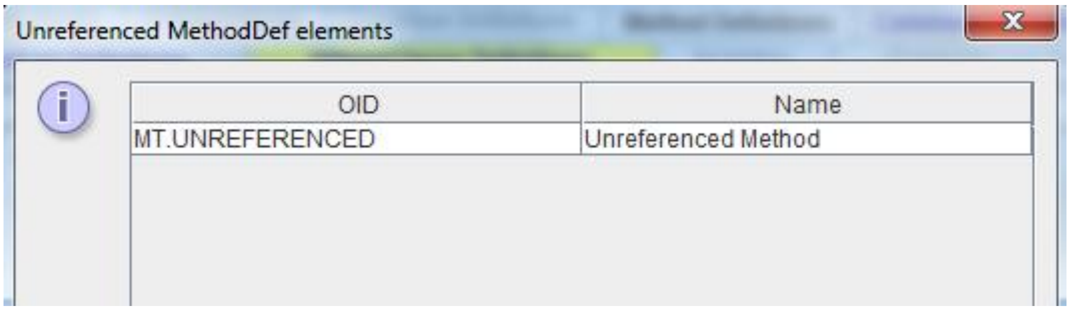
It is always a good idea to save your work before cleaning up the define.xml

Select the definitions for which you want a cleanup

☐ ValueListDef	Total: 20 - Unreferenced: 1	Show unreferenced
☐ WhereClauseDef	Total: 123 - Unreferenced: 2	Show unreferenced
☐ ItemDef	Total: 423 - Unreferenced: 0	Show unreferenced
☐ CodeList	Total: 84 - Unreferenced: 0	Show unreferenced
☐ MethodDef	Total: 57 - Unreferenced: 1	Show unreferenced
☐ CommentDef	Total: 28 - Unreferenced: 1	Show unreferenced

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

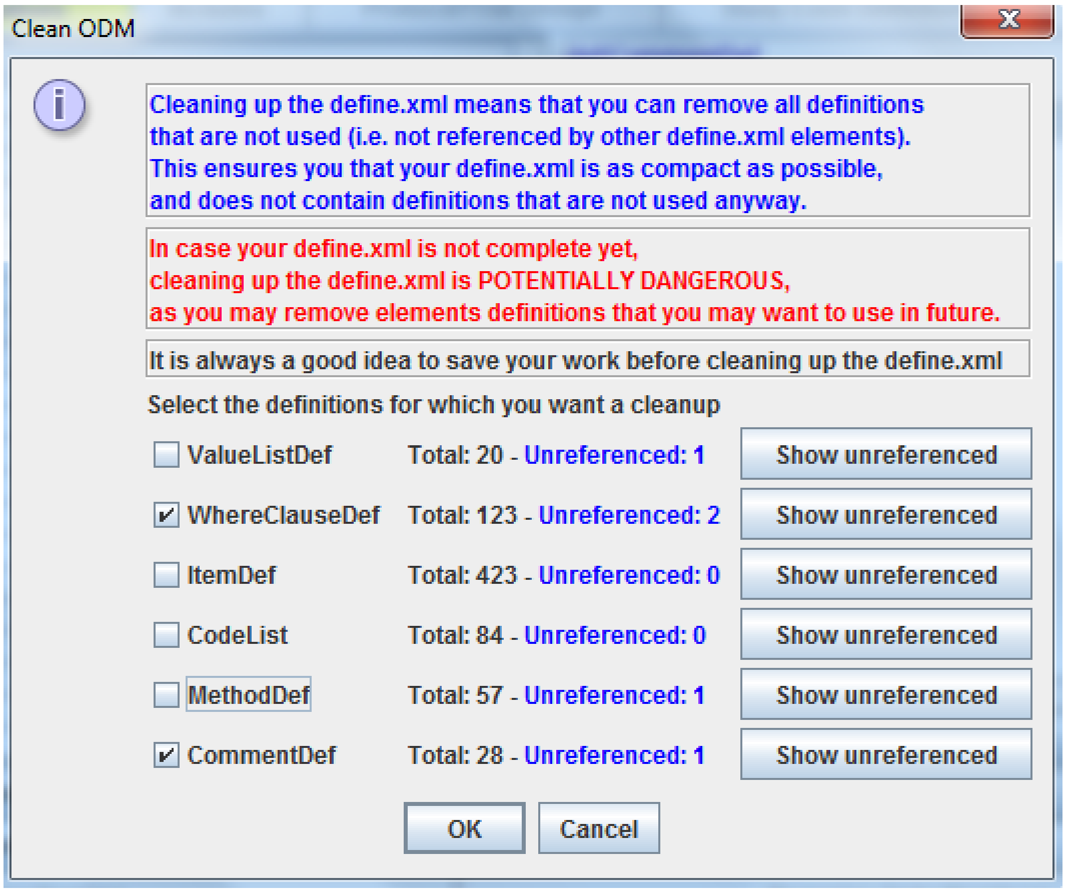
By clicking a "Show unreferenced" button, one can immediately see which definitions are unreferenced. For example for "MethodDef":



Unreferenced MethodDef elements

OID	Name
MT.UNREFERENCED	Unreferenced Method

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



Clean ODM

Cleaning up the define.xml means that you can remove all definitions
that are not used (i.e. not referenced by other define.xml elements).
This ensures you that your define.xml is as compact as possible,
and does not contain definitions that are not used anyway.

In case your define.xml is not complete yet,
cleaning up the define.xml is POTENTIALLY DANGEROUS,
as you may remove elements definitions that you may want to use in future.

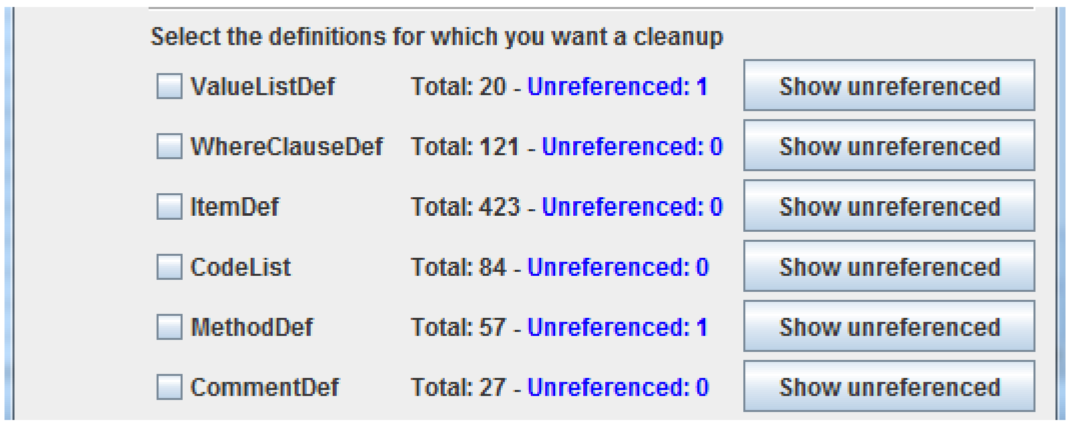
It is always a good idea to save your work before cleaning up the define.xml

Select the definitions for which you want a cleanup

☐ ValueListDef	Total: 20 - Unreferenced: 1	Show unreferenced
☑ WhereClauseDef	Total: 123 - Unreferenced: 2	Show unreferenced
☐ ItemDef	Total: 423 - Unreferenced: 0	Show unreferenced
☐ CodeList	Total: 84 - Unreferenced: 0	Show unreferenced
☐ MethodDef	Total: 57 - Unreferenced: 1	Show unreferenced
☑ CommentDef	Total: 28 - Unreferenced: 1	Show unreferenced

and just click "OK" to start the clean up action.

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



Select the definitions for which you want a cleanup

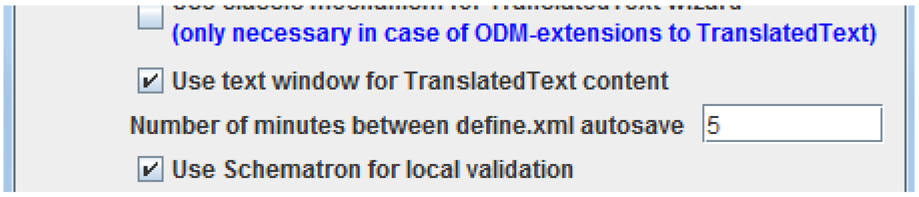
☐ ValueListDef	Total: 20 - Unreferenced: 1	Show unreferenced
☐ WhereClauseDef	Total: 121 - Unreferenced: 0	Show unreferenced
☐ ItemDef	Total: 423 - Unreferenced: 0	Show unreferenced
☐ CodeList	Total: 84 - Unreferenced: 0	Show unreferenced
☐ MethodDef	Total: 57 - Unreferenced: 1	Show unreferenced
☐ CommentDef	Total: 27 - Unreferenced: 0	Show unreferenced

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

Saving your work to file

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

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



☐ (only necessary in case of ODM-extensions to TranslatedText)

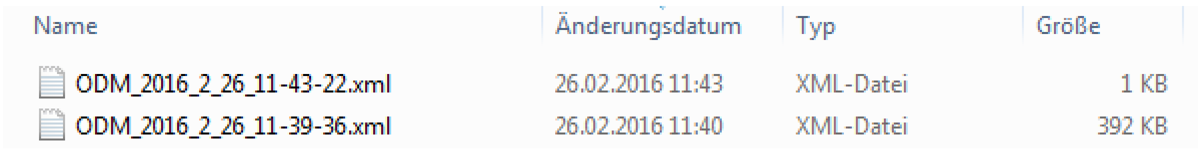
☑ Use text window for TranslatedText content

Number of minutes between define.xml autosave 5

☑ Use Schematron for local validation

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

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



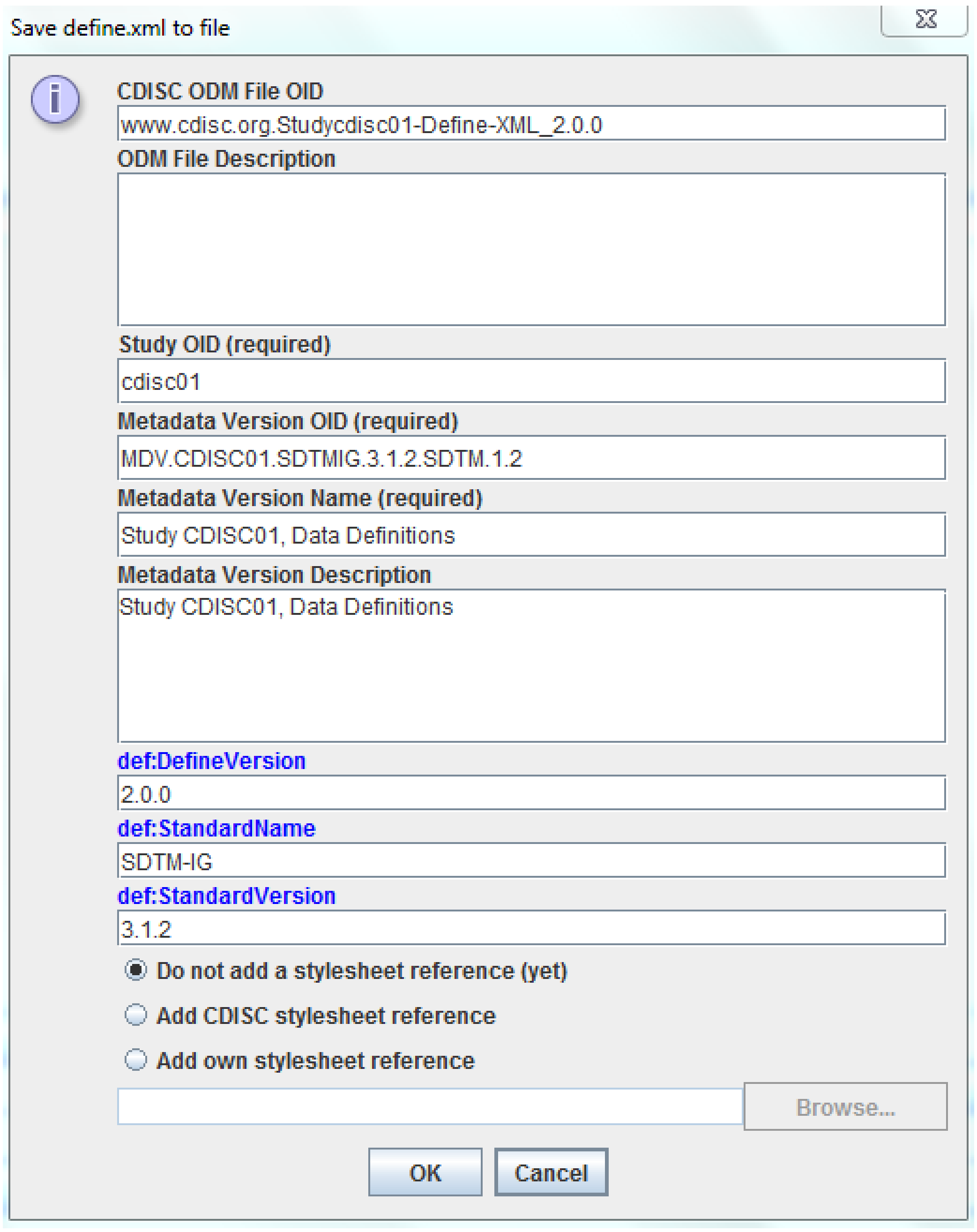
Name	Änderungsdatum	Typ	Größe
ODM_2016_2_26_11-43-22.xml	26.02.2016 11:43	XML-Datei	1 KB
ODM_2016_2_26_11-39-36.xml	26.02.2016 11:40	XML-Datei	392 KB

The file with name "ODM_2016_2_26_11-29-36.xml was created on February 2nd 2016 at 11:39:36 am (24 hour notation is used).

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

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

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


Save define.xml to file

CDISC ODM File OID

www.cdisc.org.Studycdisc01-Define-XML_2.0.0

ODM File Description

Study OID (required)

cdisc01

Metadata Version OID (required)

MDV.CDISC01.SDTMIG.3.1.2.SDTM.1.2

Metadata Version Name (required)

Study CDISC01, Data Definitions

Metadata Version Description

Study CDISC01, Data Definitions

def:DefineVersion

2.0.0

def:StandardName

SDTM-IG

def:StandardVersion

3.1.2

[✓ Do not add a stylesheet reference (yet)]

Add CDISC stylesheet reference

Add own stylesheet reference

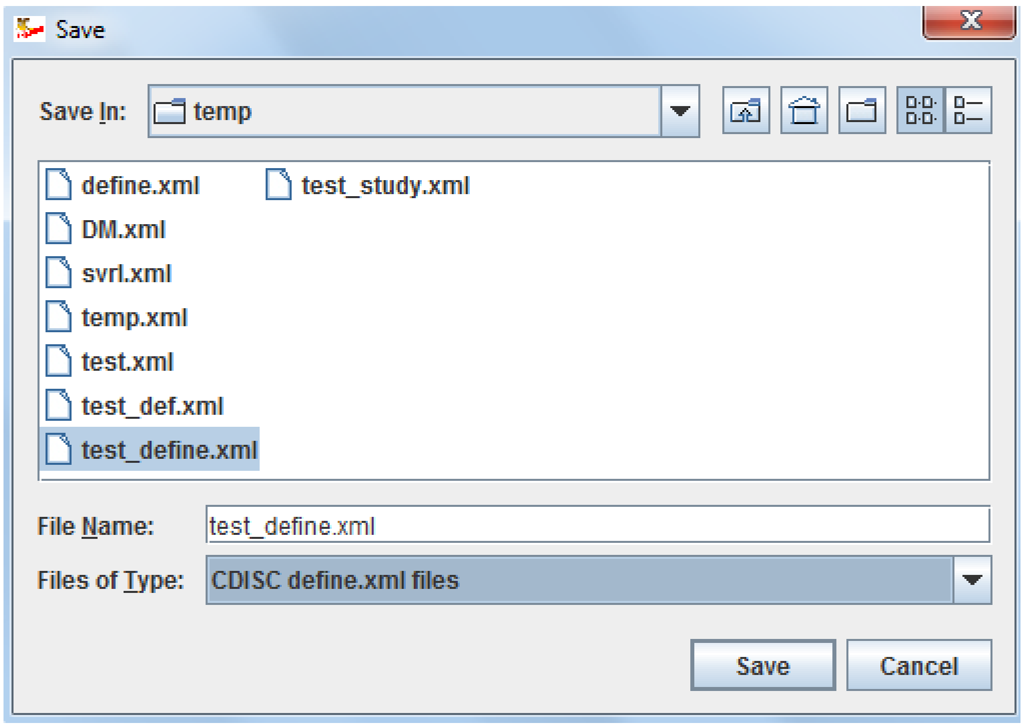
At this point, you can still change some of the metadata.

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

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

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

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


Save

Save In: temp

define.xml
DM.xml
svrl.xml
temp.xml
test.xml
test_def.xml
test_define.xml
test_study.xml

File Name: test_define.xml

Files of Type: CDISC define.xml files

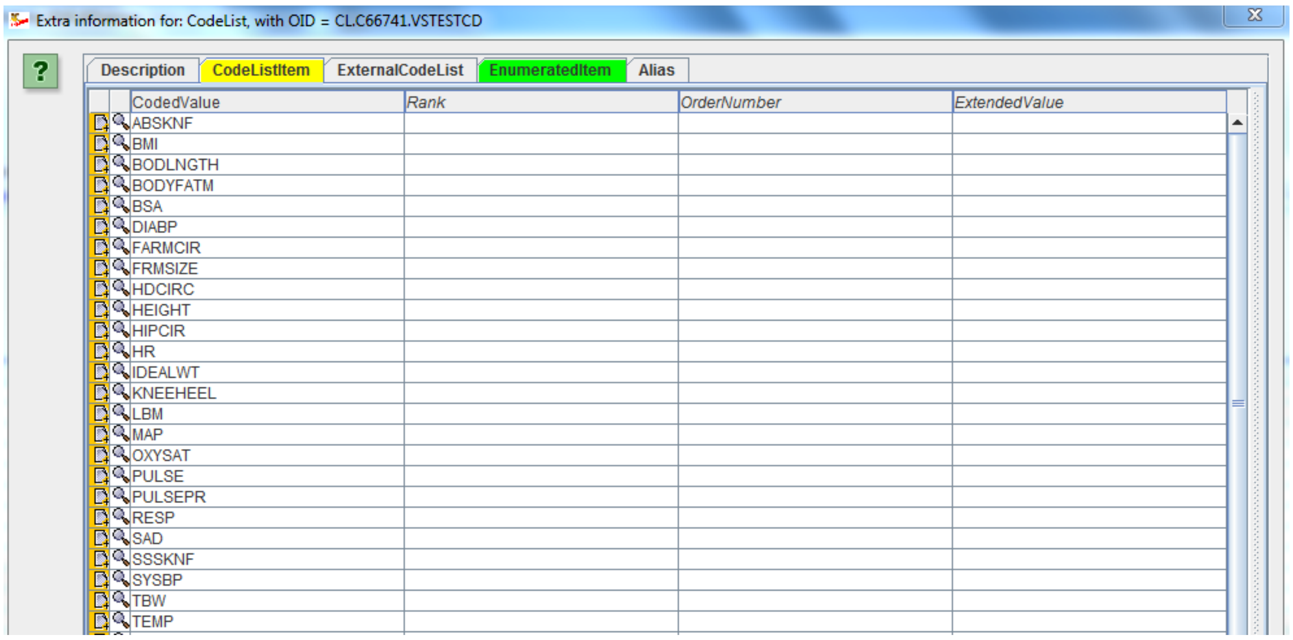
Please also remark that is not always necessary to first save your work to file in order to display your define.xml in your favorite browser. As explained before, you can always use the menu "View - define.xml in browser".

Creating ValueLists

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

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

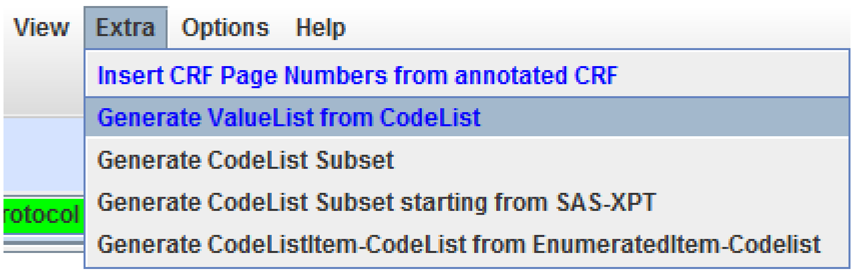
In this manual, we will show how to set up a valuelist for VS (vital signs)
The best way to start generating a valuelist is to start from a codelist. Let us suppose that you have already loaded the CDISC-CT codelist for VSTESTCD:


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

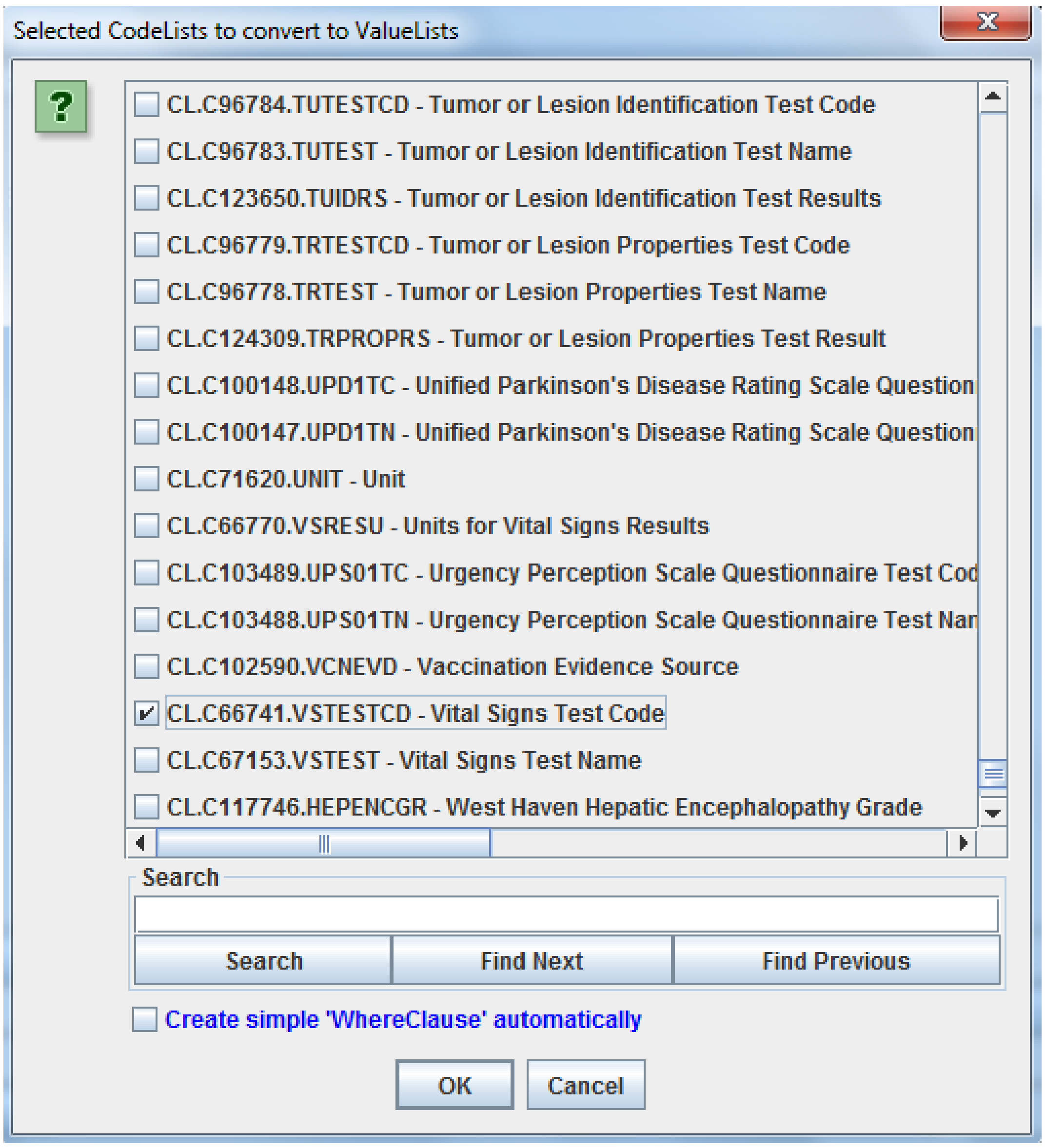
CodedValue	Rank	OrderNumber	ExtendedValue
ABSKNF			
BMI			
BODLNGTH			
BODYFATM			
BSA			
DIABP			
FARMCIR			
FRMSIZE			
HDCIRC			
HEIGHT			
HIPCIR			
HR			
IDEALWT			
KNEEHEEL			
LBM			
MAP			
OXYSAT			
PULSE			
PULSEPR			
RESP			
SAD			
SSSKNF			
SYSBP			
TBW			
TEMP			

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

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

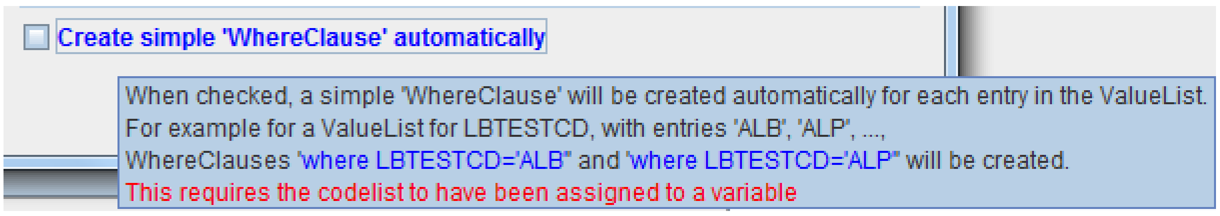

View
Extra
Options
Help
Insert CRF Page Numbers from annotated CRF
Generate ValueList from CodeList
Generate CodeList Subset
Generate CodeList Subset starting from SAS-XPT
Generate CodeListItem-CodeList from EnumeratedItem-Codelist
rotocol

This results in:


Selected CodeLists to convert to ValueLists
CL.C96784.TUTESTCD - Tumor or Lesion Identification Test Code
CL.C96783.TUTEST - Tumor or Lesion Identification Test Name
CL.C123650.TUIDRS - Tumor or Lesion Identification Test Results
CL.C96779.TRTESTCD - Tumor or Lesion Properties Test Code
CL.C96778.TRTEST - Tumor or Lesion Properties Test Name
CL.C124309.TRPROPRS - Tumor or Lesion Properties Test Result
CL.C100148.UPD1TC - Unified Parkinson's Disease Rating Scale Question
CL.C100147.UPD1TN - Unified Parkinson's Disease Rating Scale Question
CL.C71620.UNIT - Unit
CL.C66770.VSRESU - Units for Vital Signs Results
CL.C103489.UPS01TC - Urgency Perception Scale Questionnaire Test Cod
CL.C103488.UPS01TN - Urgency Perception Scale Questionnaire Test Nar
CL.C102590.VCNEVD - Vaccination Evidence Source
✓ CL.C66741.VSTESTCD - Vital Signs Test Code
CL.C67153.VSTEST - Vital Signs Test Name
CL.C117746.HEPENCGR - West Haven Hepatic Encephalopathy Grade
Search
Search
Find Next
Find Previous
Create simple 'WhereClause' automatically
OK
Cancel

and select the codelist "CL.C66741.VSTESTCD - Vital Signs Test Code" (you can search for a specific codelist using the "Search" field and buttons)
If you started from a set of SAS-XPT files, and have subsetted this codelist yet, you will of course select the subsetted codelist[13].

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

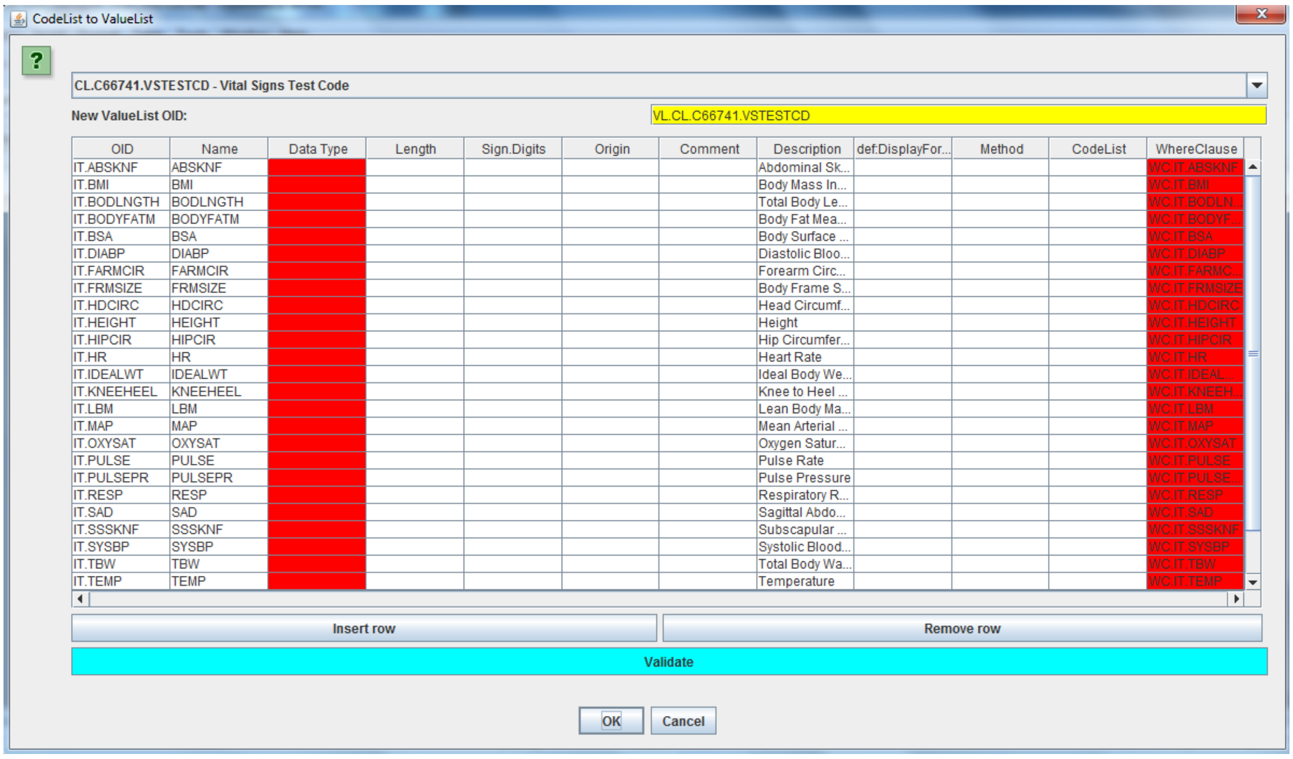


When checked, a simple 'WhereClause' will be created automatically for each entry in the ValueList.
For example for a ValueList for LBTESTCD, with entries 'ALB', 'ALP', ...,
WhereClauses 'where LBTESTCD='ALB'' and 'where LBTESTCD='ALP'' will be created.
This requires the codelist to have been assigned to a variable

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

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

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

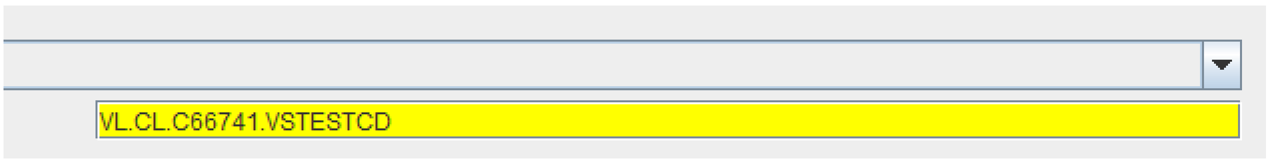


CL.C66741.VSTESTCD - Vital Signs Test Code

New ValueList OID: VL.CL.C66741.VSTESTCD

OID	Name	Data Type	Length	Sign.Digits	Origin	Comment	Description	def:DisplayFor...	Method	CodeList	WhereClause
IT.ABSKNF	ABSKNF	🟥					Abdominal Sk...				WC.IT.ABSKNF
IT.BMI	BMI	🟥					Body Mass In...				WC.IT.BMI
IT.BODLNGTH	BODLNGTH	🟥					Total Body Le...				WC.IT.BODLN...
IT.BODYFATM	BODYFATM	🟥					Body Fat Mea...				WC.IT.BODYF...
IT.BSA	BSA	🟥					Body Surface ...				WC.IT.BSA
IT.DIABP	DIABP	🟥					Diastolic Bloo...				WC.IT.DIABP
IT.FARMCIR	FARMCIR	🟥					Forearm Circ...				WC.IT.FARMC...
IT.FRMSIZE	FRMSIZE	🟥					Body Frame S...				WC.IT.FRMSIZE
IT.HDCIRC	HDCIRC	🟥					Head Circumf...				WC.IT.HDCIRC
IT.HEIGHT	HEIGHT	🟥					Height				WC.IT.HEIGHT
IT.HIPCIR	HIPCIR	🟥					Hip Circumfer...				WC.IT.HIPCIR
IT.HR	HR	🟥					Heart Rate				WC.IT.HR
IT.IDEALWT	IDEALWT	🟥					Ideal Body We...				WC.IT.IDEAL...
IT.KNEEHEEL	KNEEHEEL	🟥					Knee to Heel ...				WC.IT.KNEEH...
IT.LBM	LBM	🟥					Lean Body Ma...				WC.IT.LBM
IT.MAP	MAP	🟥					Mean Arterial ...				WC.IT.MAP
IT.OXYSAT	OXYSAT	🟥					Oxygen Satur...				WC.IT.OXYSAT
IT.PULSE	PULSE	🟥					Pulse Rate				WC.IT.PULSE
IT.PULSEPR	PULSEPR	🟥					Pulse Pressure				WC.IT.PULSE...
IT.RESP	RESP	🟥					Respiratory R...				WC.IT.RESP
IT.SAD	SAD	🟥					Sagittal Abdo...				WC.IT.SAD
IT.SSSKNF	SSSKNF	🟥					Subscapular ...				WC.IT.SSSKNF
IT.SYSBP	SYSBP	🟥					Systolic Blood...				WC.IT.SYSBP
IT.TBW	TBW	🟥					Total Body Wa...				WC.IT.TBW
IT.TEMP	TEMP	🟥					Temperature				WC.IT.TEMP

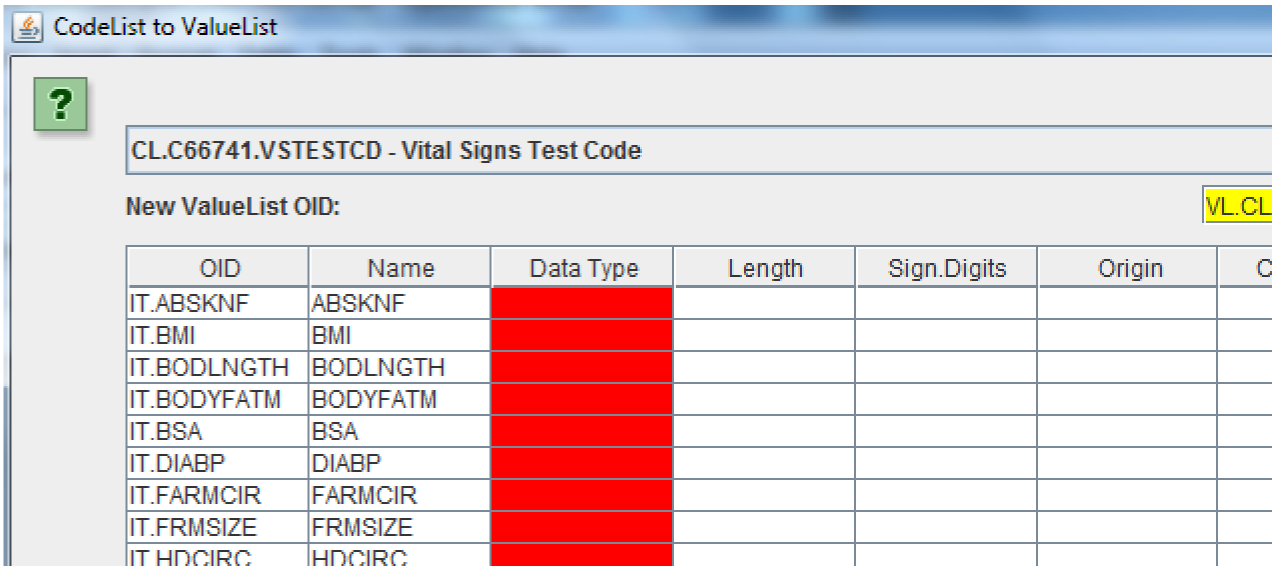
In the upper right part of the dialog, a new identifier (OID) is proposed for the ValueList, consisting of the prefix "VL." and the OID of the original codelist:



VL.CL.C66741.VSTESTCD

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

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


CodeList to ValueList

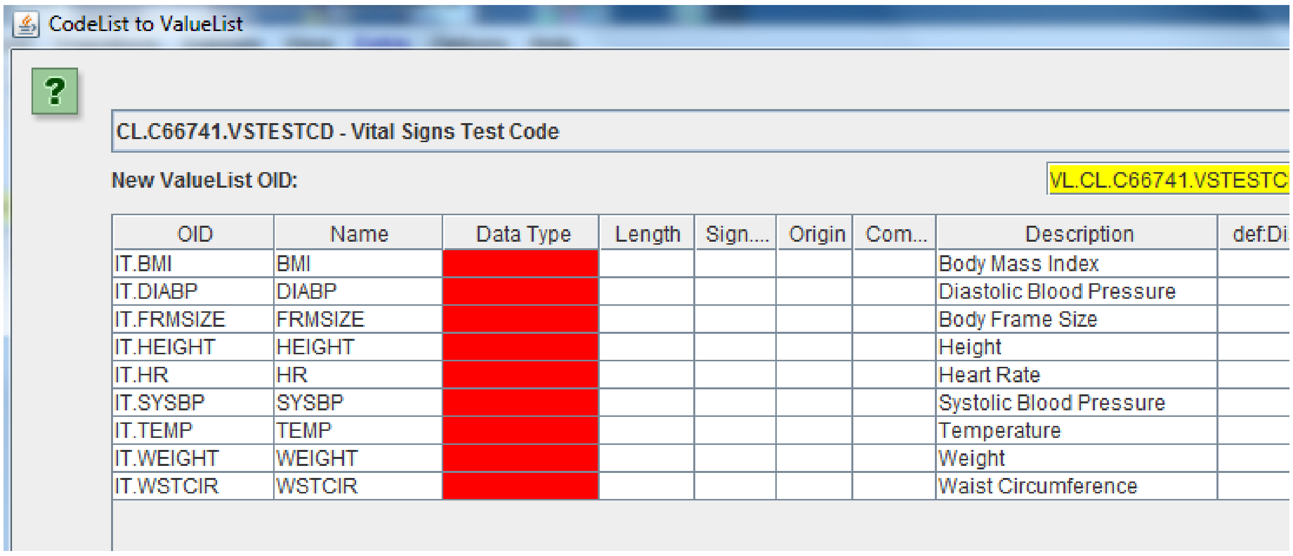
CL.C66741.VSTESTCD - Vital Signs Test Code

New ValueList OID: VL.CL

OID	Name	Data Type	Length	Sign.Digits	Origin	C
IT.ABSKNF	ABSKNF	🟥				
IT.BMI	BMI	🟥				
IT.BODLNGTH	BODLNGTH	🟥				
IT.BODYFATM	BODYFATM	🟥				
IT.BSA	BSA	🟥				
IT.DIABP	DIABP	🟥				
IT.FARMCIR	FARMCIR	🟥				
IT.FRMSIZE	FRMSIZE	🟥				
IT.HDCIRC	HDCIRC	🟥				

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

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


CodeList to ValueList

CL.C66741.VSTESTCD - Vital Signs Test Code

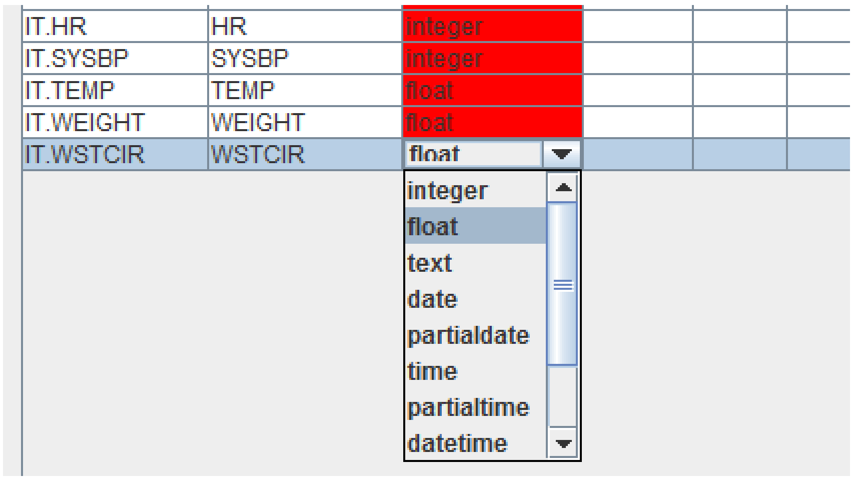
New ValueList OID: VL.CL.C66741.VSTESTC

OID	Name	Data Type	Length	Sign....	Origin	Com...	Description	def:Di
IT.BMI	BMI	🟥					Body Mass Index	
IT.DIABP	DIABP	🟥					Diastolic Blood Pressure	
IT.FRMSIZE	FRMSIZE	🟥					Body Frame Size	
IT.HEIGHT	HEIGHT	🟥					Height	
IT.HR	HR	🟥					Heart Rate	
IT.SYSBP	SYSBP	🟥					Systolic Blood Pressure	
IT.TEMP	TEMP	🟥					Temperature	
IT.WEIGHT	WEIGHT	🟥					Weight	
IT.WSTCIR	WSTCIR	🟥					Waist Circumference	

Remark that one can remove more than one row at the time by using the "Ctrl" key when selecting rows. Then use the "Remove Row" button to remove the selected rows.
Usually the field "Description" is very useful to understand the codes from the codelist. Use it!

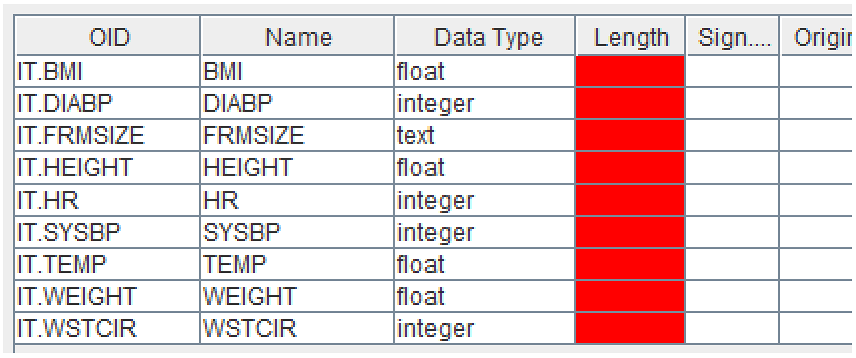
For each remaining test, we now need to assign the appropriate data type. Usually, this is immediately clear by a look on the CRF. For example for BMI we expect a float, whereas for "Hearth Rate" we probably inspect an integer (depending on the unit that was used, mostly "beats per minute"). For "Body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

For example:



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

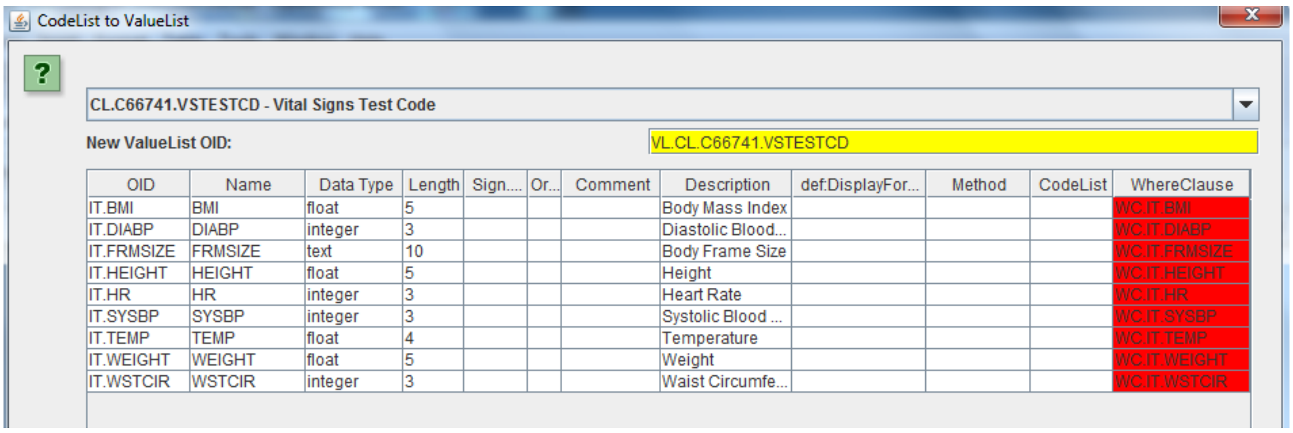
It is a good idea now to click the button "Validate" (the cyan colored button near the bottom). This will lead to:



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

I.e. we also need to provide the maximal length for each of the test results[14].

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



CL.C66741.VSTESTCD - Vital Signs Test Code

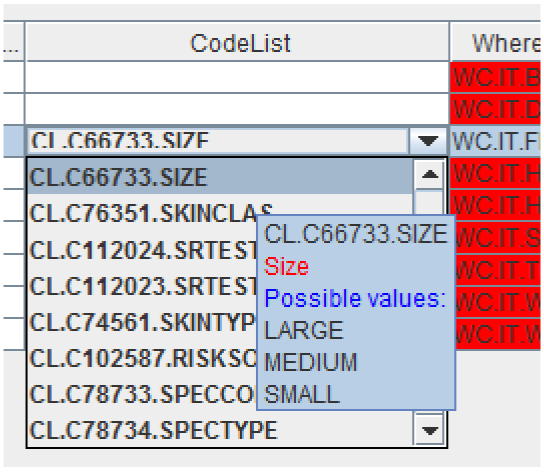
New ValueList OID: VL.CL.C66741.VSTESTCD

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

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

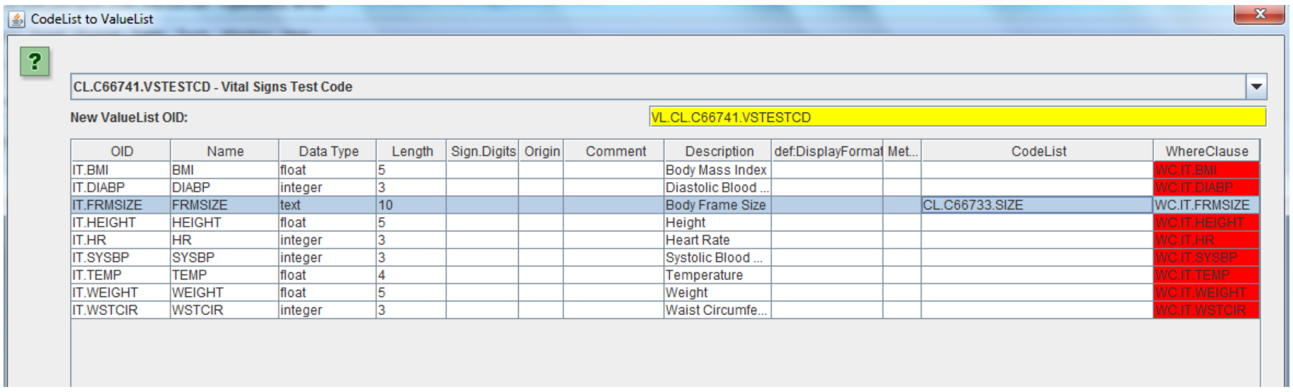
We also still need to assign a codelist to the item "Body Frame Size" (FRMSIZE) as the CRF shows checkboxes (choices) for this. It might be that we have already generated a separated codelist for this, but it might also be that we can use a CDISC codelist for "body frame size". For example, clicking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

[14]Remark that for datatype "Date", "Time" and similar data types, no maximal length needs to provided.



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

So we now have:



CL.C66741.VSTESTCD - Vital Signs Test Code

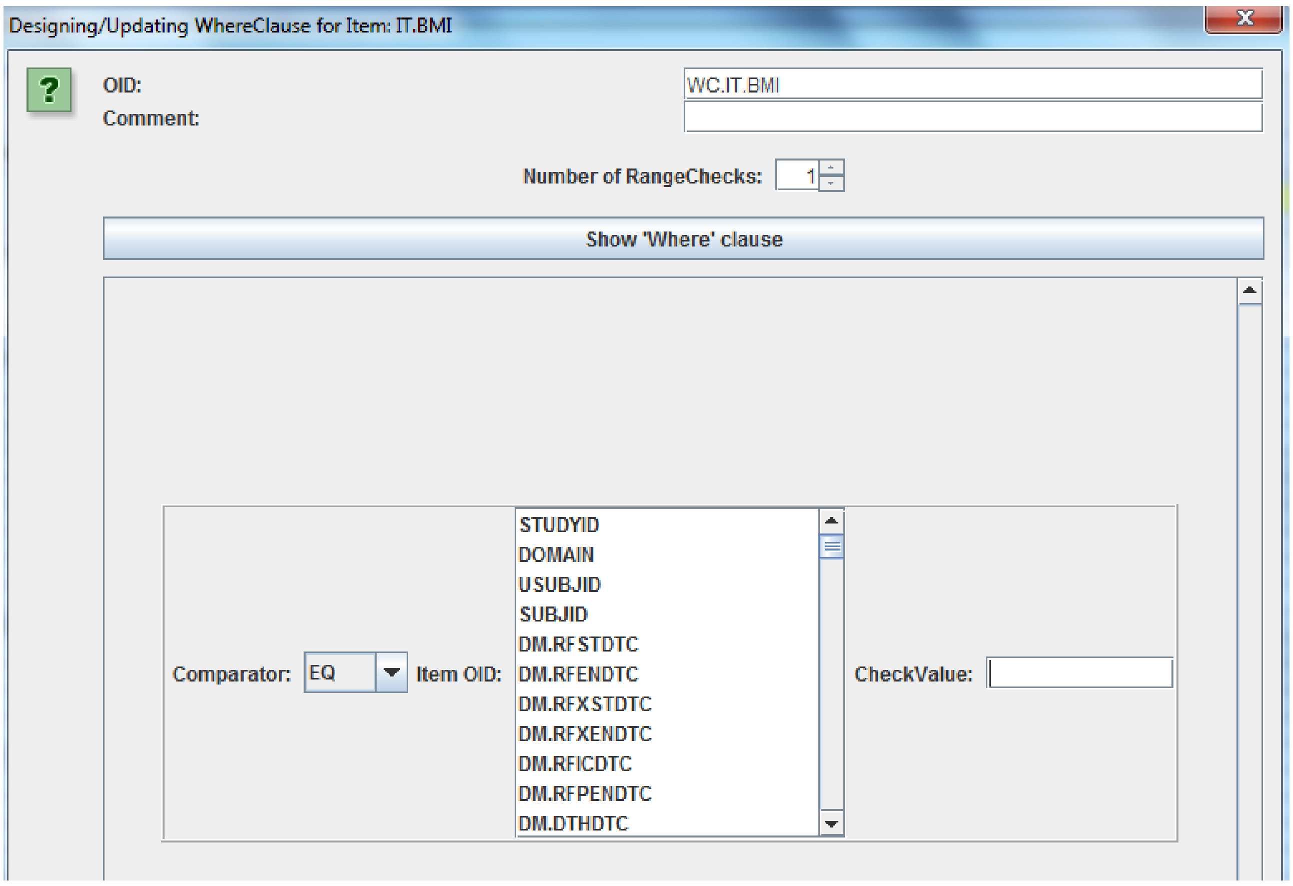
New ValueList OID: VL.CL.C66741.VSTESTCD

OID	Name	Data Type	Length	Sign.Digits	Origin	Comment	Description	def:DisplayFormat	Met...	CodeList	WhereClause
IT.BMI	BMI	float	5				Body Mass Index				WC.IT.BMI
IT.DIABP	DIABP	integer	3				Diastolic Blood ...				WC.IT.DIABP
IT.FRMSIZE	FRMSIZE	text	10				Body Frame Size			CL.C66733.SIZE	WC.IT.FRMSIZE
IT.HEIGHT	HEIGHT	float	5				Height				WC.IT.HEIGHT
IT.HR	HR	integer	3				Heart Rate				WC.IT.HR
IT.SYSBP	SYSBP	integer	3				Systolic Blood ...				WC.IT.SYSBP
IT.TEMP	TEMP	float	4				Temperature				WC.IT.TEMP
IT.WEIGHT	WEIGHT	float	5				Weight				WC.IT.WEIGHT
IT.WSTCIR	WSTCIR	integer	3				Waist Circumfe...				WC.IT.WSTCIR

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

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

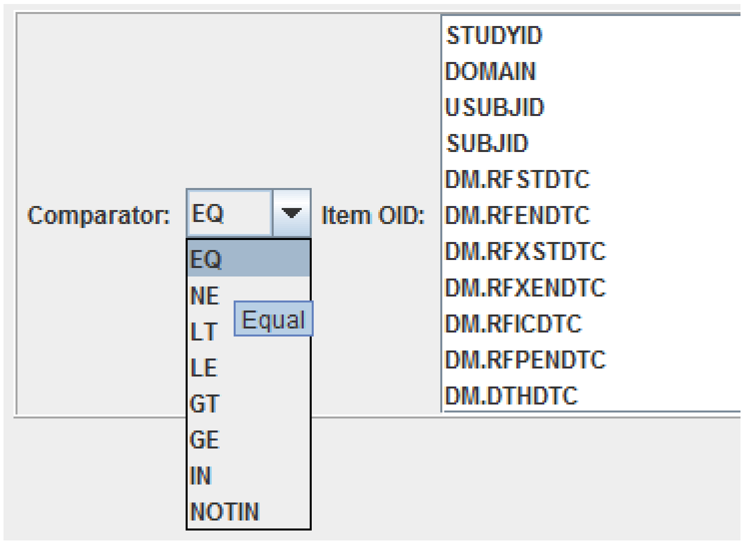
Now click the most-right cell "WC.IT.BMI". This opens a wizard for setting up the "where clause":



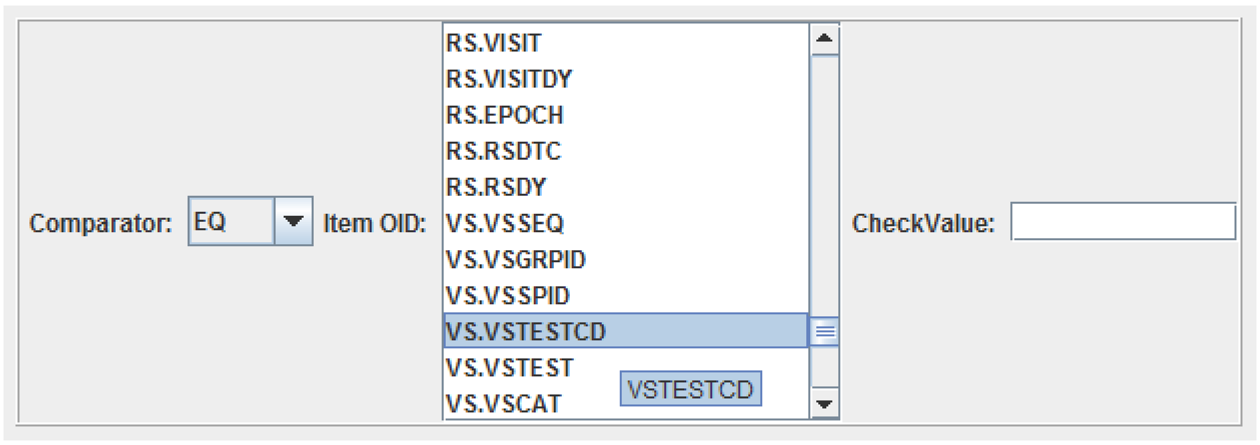
The identifier (the OID) has already been created, you can however change it (mostly you will want to keep the one that is suggested).

The "Comment" field only needs to be populated when your "where" contains variables from other domains. This is well explained in the define.xml 2.0 specification in example 4.2.2.2 about the metadata for VSORRESU (vital signs original result units) depending on the country (USA, Mexico or Canada).
If you fill something into the "Comment" field, a "def:CommentDef" element will be generated and a reference to it is added in the "ItemRef" element within the ValueList.

Then you can set the number of conditions "RangeChecks" that will be combined. For example, if you want to have a condition like "where the test code is BMI and the age is less than 70), then you will need two "RangeChecks" and need to set the "Number of RangeCheck" spinner to 2.
In our case however, we only need to state "where the vital signs test code is BMI", so we can leave the spinner to "1". Also the value for the "Comparator" can be left to "EQ" as it means "equals to" as is also shown by the tooltip:

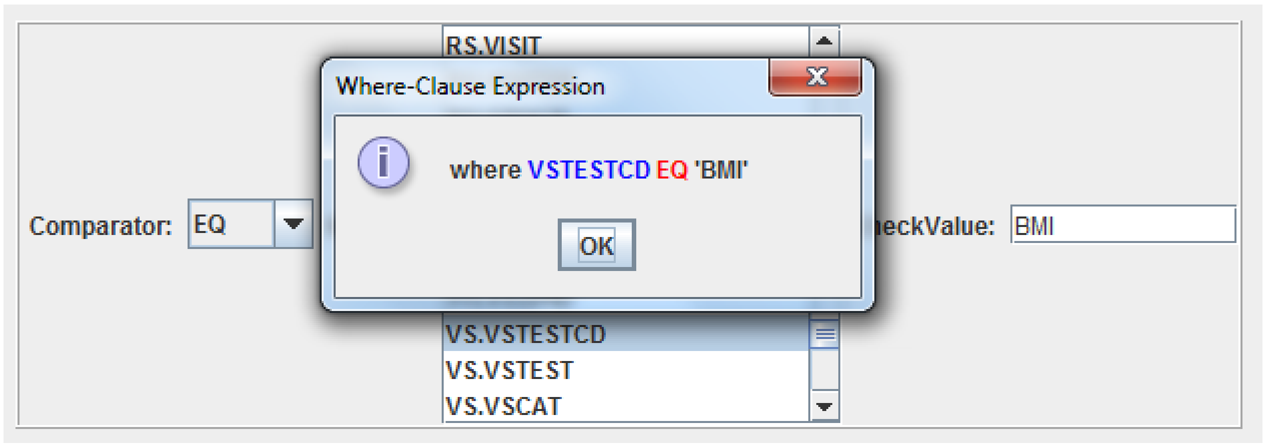


We then still need to assign the SDTM/SEND/ADaM variable in the condition definition which in our case is "VSTESTCD". It can quickl be found by typing the first characters "VS" and then selecting "VS.VSTESTCD":



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

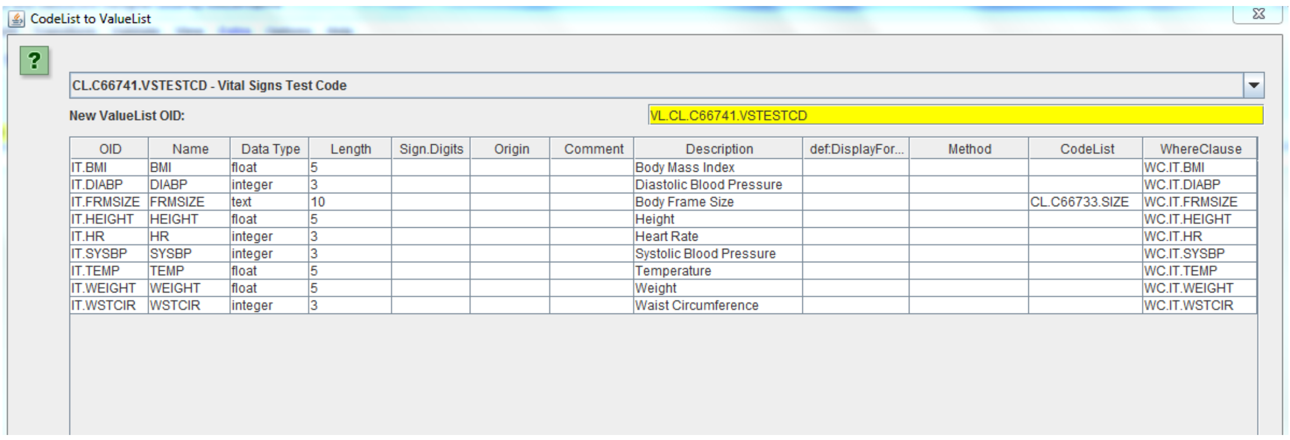
You can always check what you have developed as a condition by clicking the button "Show 'Where' Clause". The human readable condition is then displayed. In our case:



where VSTESTCD EQ 'BMI'

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

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

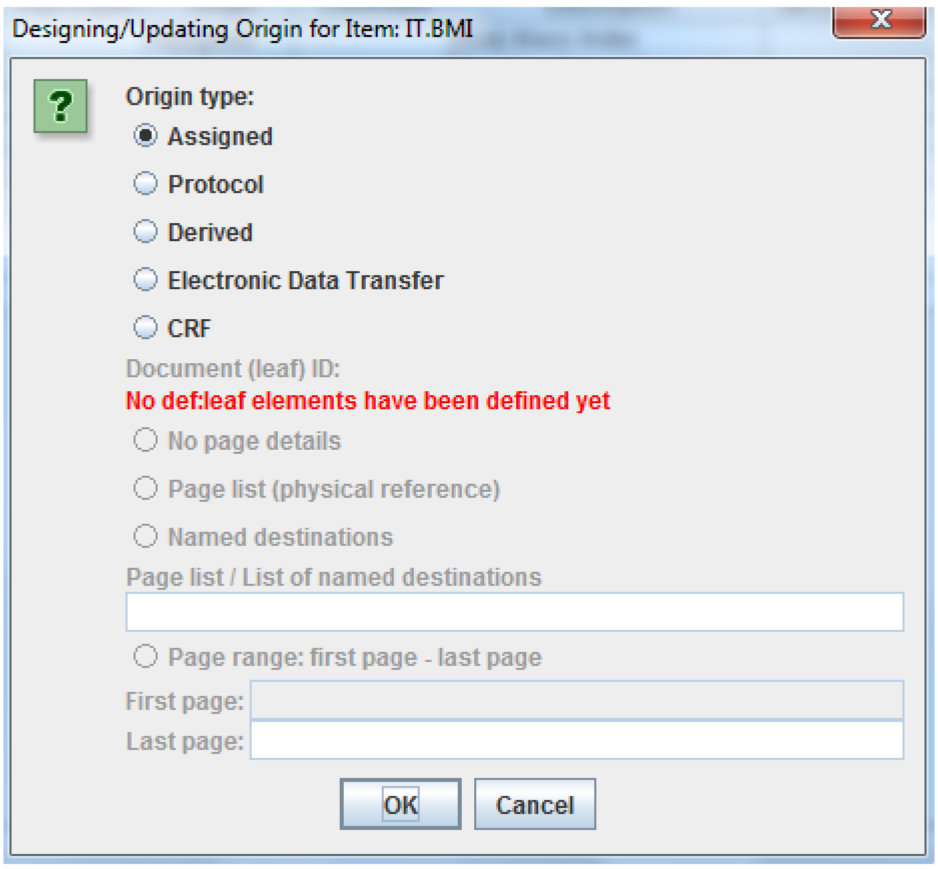

CodeList to ValueList

CL.C66741.VSTESTCD - Vital Signs Test Code

New ValueList OID: VL.CL.C66741.VSTESTCD

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

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


Designing/Updating Origin for Item: IT.BMI

Origin type:

[✓ Assigned]

Protocol

Derived

Electronic Data Transfer

CRF

Document (leaf) ID:

No def:leaf elements have been defined yet

No page details

Page list (physical reference)

Named destinations

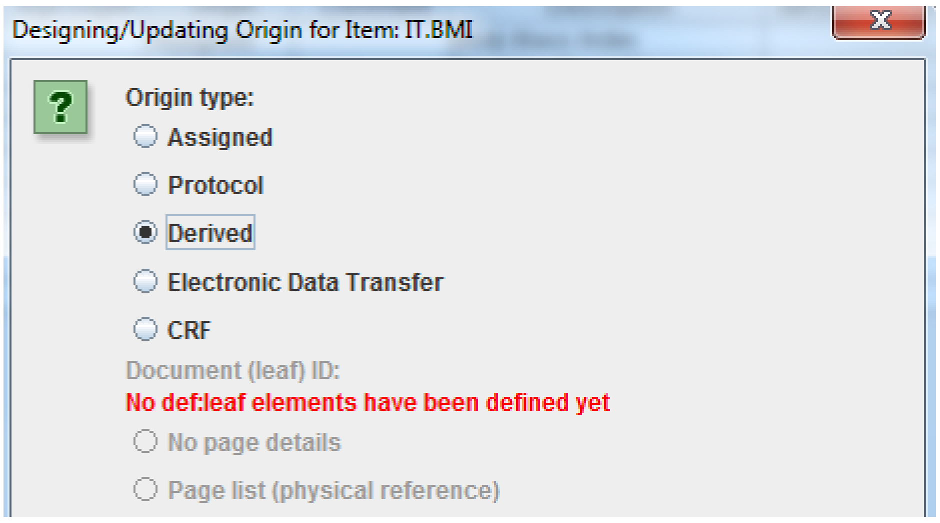
Page list / List of named destinations

Page range: first page - last page

First page:

Last page:

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



Designing/Updating Origin for Item: IT.BMI

Origin type:

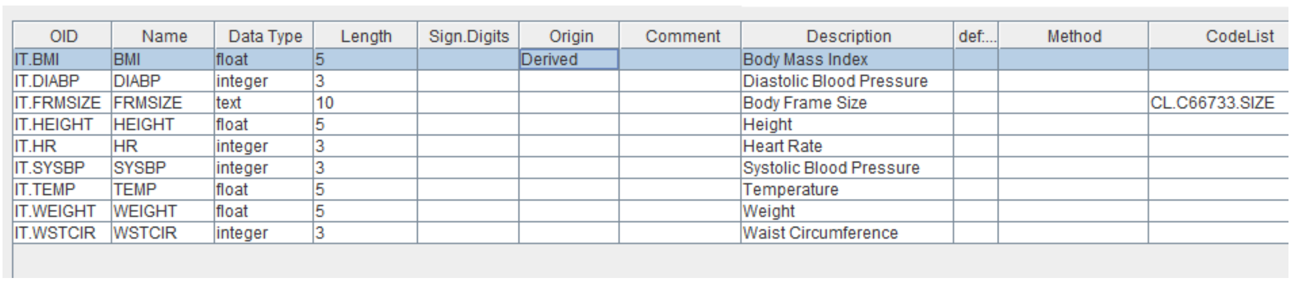
- Assigned
- Protocol
- [✓ Derived]
- Electronic Data Transfer
- CRF

Document (leaf) ID:

No def:leaf elements have been defined yet

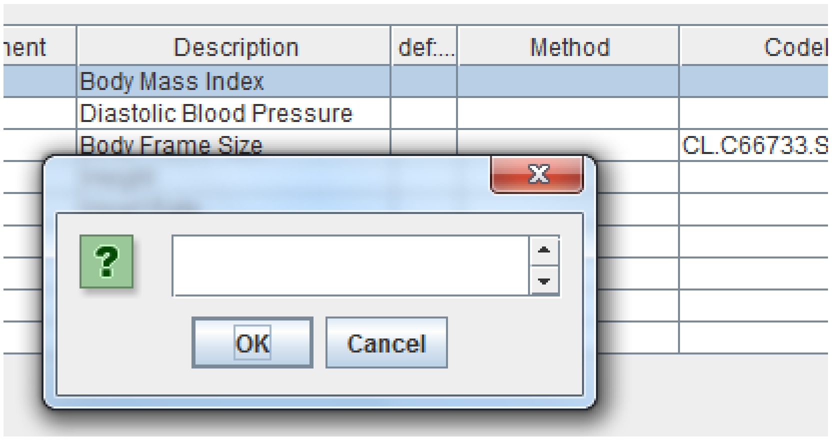
- No page details
- Page list (physical reference)

and after clicking "OK":



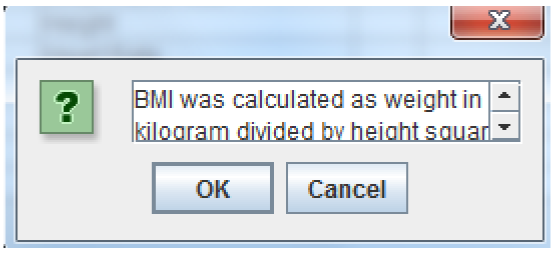
OID	Name	Data Type	Length	Sign.Digits	Origin	Comment	Description	def:...	Method	CodeList
IT.BMI	BMI	float	5		Derived		Body Mass Index			
IT.DIABP	DIABP	integer	3				Diastolic Blood Pressure			
IT.FRMSIZE	FRMSIZE	text	10				Body Frame Size			CL.C66733.SIZE
IT.HEIGHT	HEIGHT	float	5				Height			
IT.HR	HR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

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



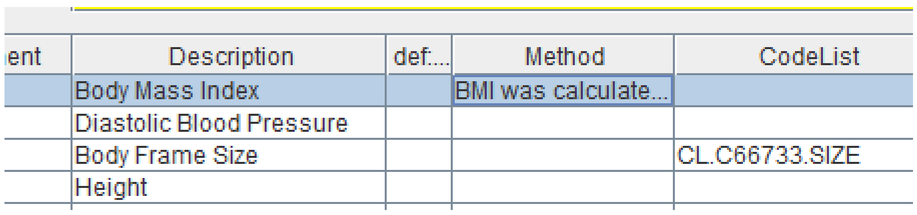
nent	Description	def:...	Method	Codel
	Body Mass Index			
	Diastolic Blood Pressure			
	Body Frame Size			CL.C66733.S

and one can e.g. add:



BMI was calculated as weight in kilogram divided by height squar

And after clicking "OK":

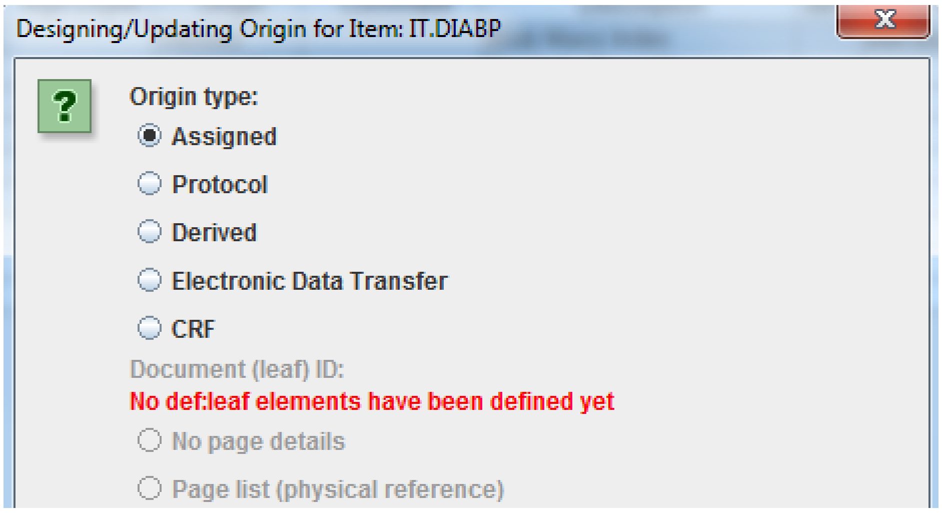


ent	Description	def:...	Method	CodeList
	Body Mass Index		BMI was calculate...	
	Diastolic Blood Pressure			
	Body Frame Size			CL.C66733.SIZE
	Height			

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

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

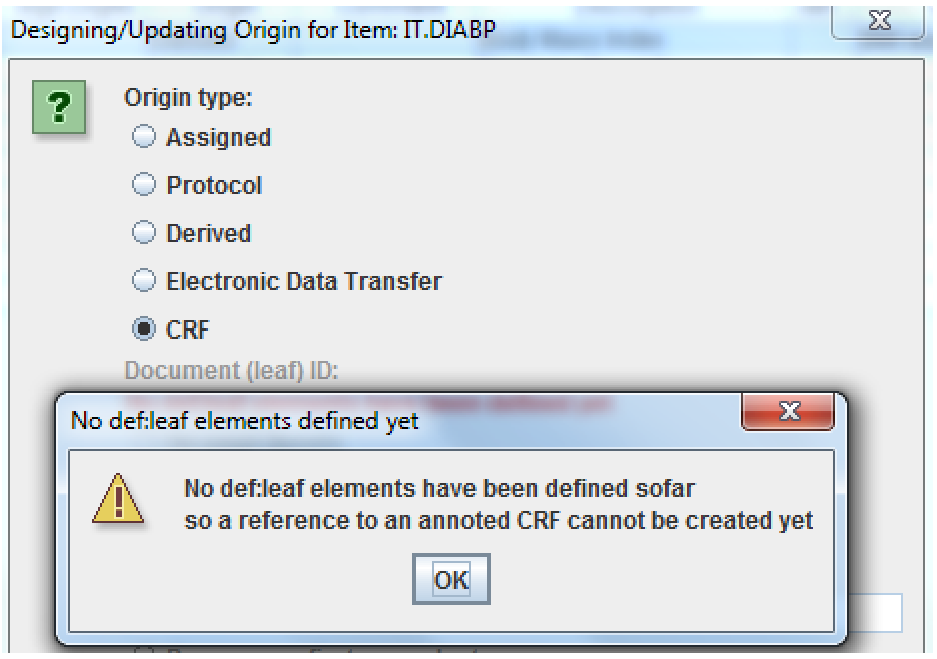
Essentially, we should add an "Origin" to each of the newly created value-level variables, for example for "DIABP":



Designing/Updating Origin for Item: IT.DIABP

Origin type:
[✓ Assigned]
Protocol
Derived
Electronic Data Transfer
CRF
Document (leaf) ID:
No def:leaf elements have been defined yet
No page details
Page list (physical reference)

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


Designing/Updating Origin for Item: IT.DIABP

Origin type:

- Assigned
- Protocol
- Derived
- Electronic Data Transfer
- [✓ CRF]

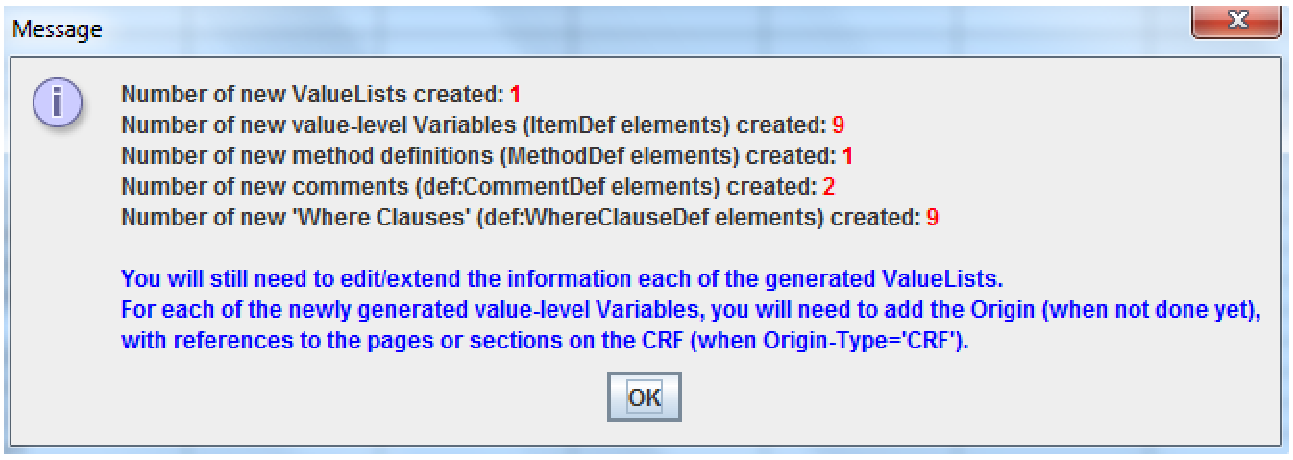
Document (leaf) ID:

No def:leaf elements defined yet

No def:leaf elements have been defined sofar
so a reference to an annoted CRF cannot be created yet

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

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

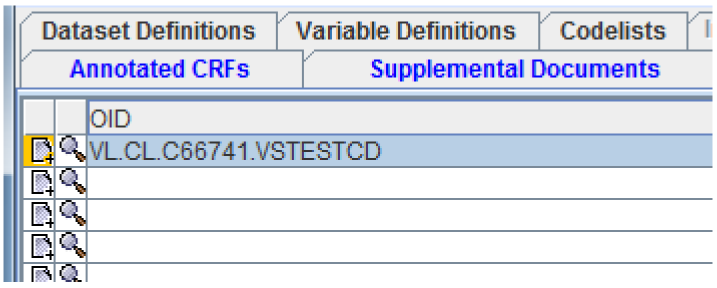

Message

Number of new ValueLists created: 1
Number of new value-level Variables (ItemDef elements) created: 9
Number of new method definitions (MethodDef elements) created: 1
Number of new comments (def:CommentDef elements) created: 2
Number of new 'Where Clauses' (def:WhereClauseDef elements) created: 9

You will still need to edit/extend the information each of the generated ValueLists.
For each of the newly generated value-level Variables, you will need to add the Origin (when not done yet),
with references to the pages or sections on the CRF (when Origin-Type='CRF').

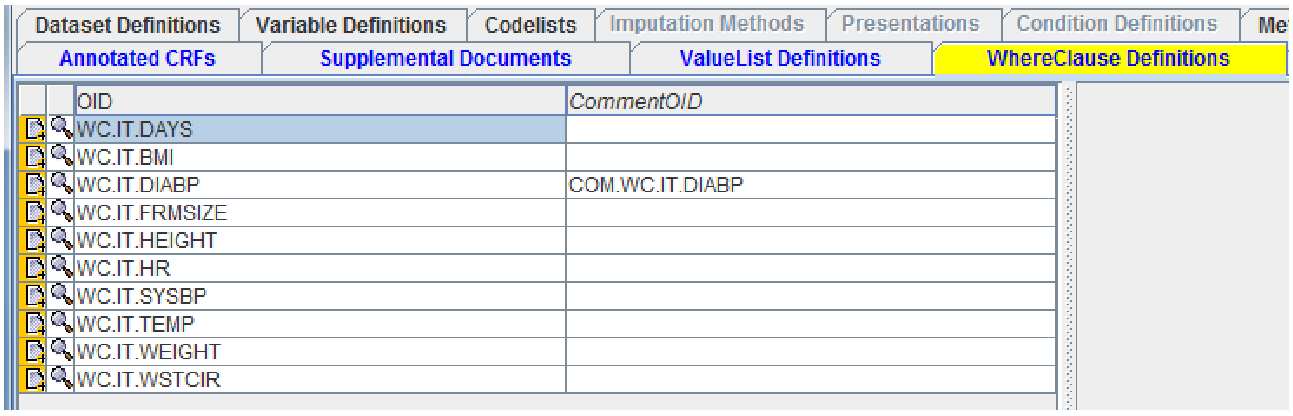
showing a nice summary of what has been created.

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



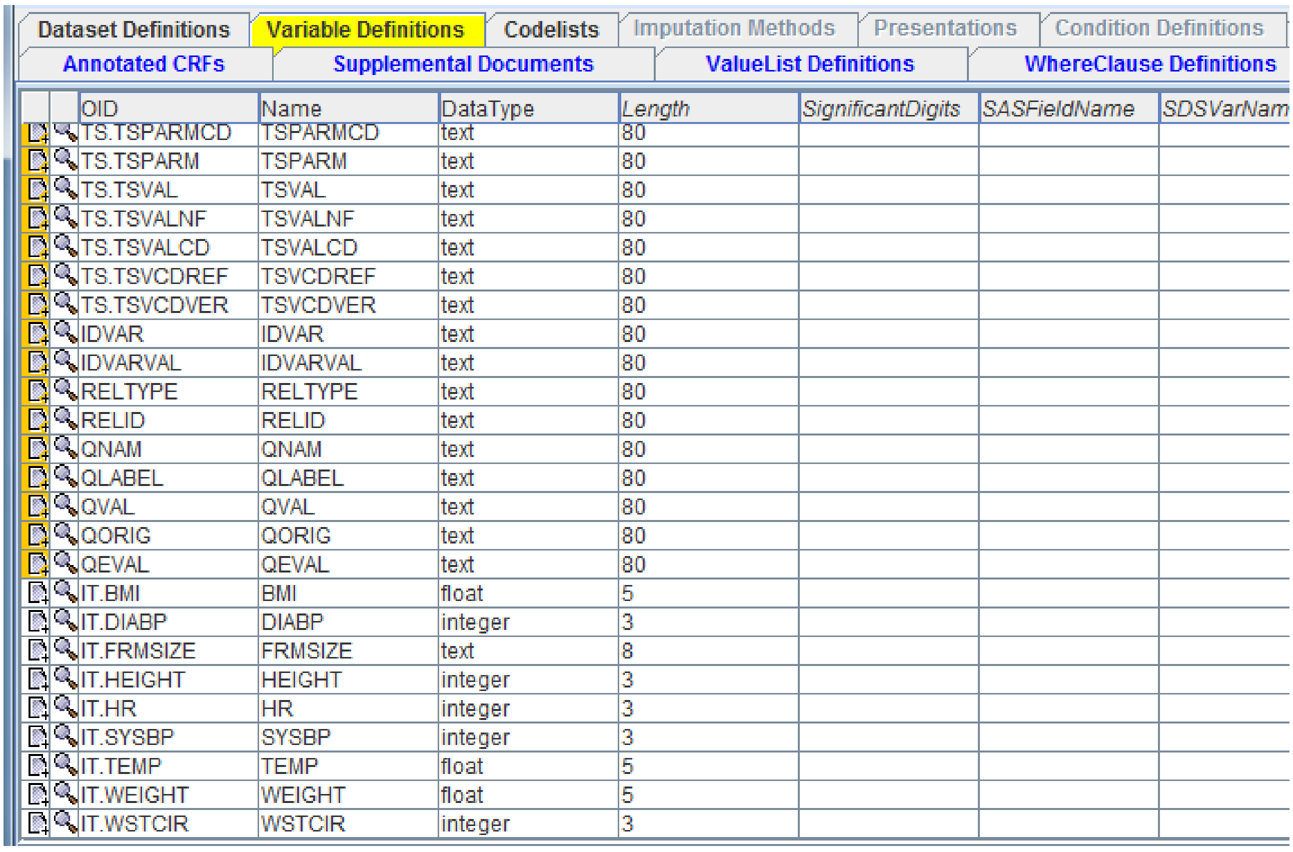
OID
VL.CL.C66741.VSTESTCD

or for "Where Clauses":



OID	CommentOID
WC.IT.DAYS	
WC.IT.BMI	
WC.IT.DIABP	COM.WC.IT.DIABP
WC.IT.FRMSIZE	
WC.IT.HEIGHT	
WC.IT.HR	
WC.IT.SYSBP	
WC.IT.TEMP	
WC.IT.WEIGHT	
WC.IT.WSTCIR	

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

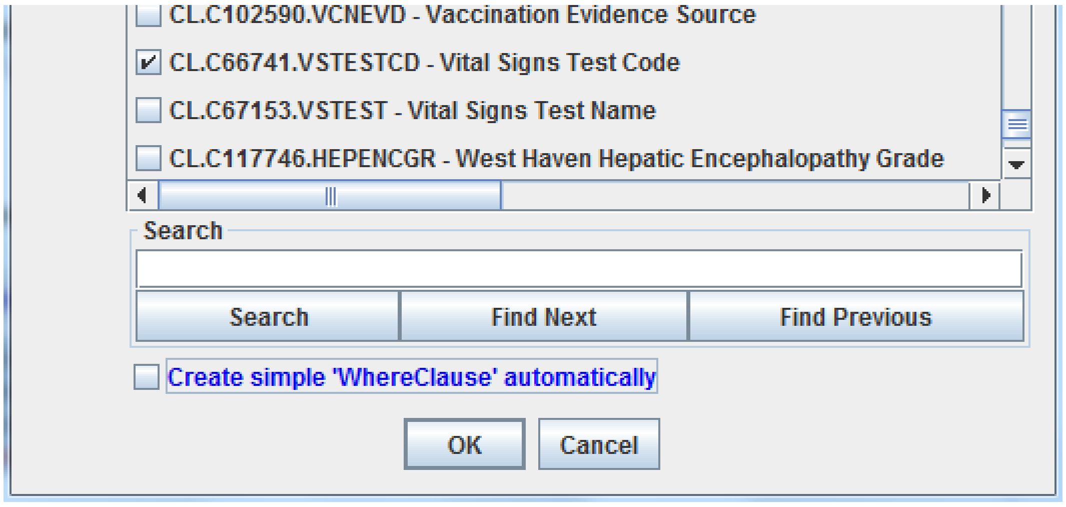


OID	Name	DataType	Length	SignificantDigits	SASFieldName	SDSVarNam
TS.TSPARMCD	TSPARMCD	text	80			
TS.TSPARM	TSPARM	text	80			
TS.TSVAL	TSVAL	text	80			
TS.TSVALNF	TSVALNF	text	80			
TS.TSVALCD	TSVALCD	text	80			
TS.TSVCDREF	TSVCDREF	text	80			
TS.TSVCDVER	TSVCDVER	text	80			
IDVAR	IDVAR	text	80			
IDVARVAL	IDVARVAL	text	80			
RELTYPE	RELTYPE	text	80			
RELID	RELID	text	80			
QNAM	QNAM	text	80			
QLABEL	QLABEL	text	80			
QVAL	QVAL	text	80			
QORIG	QORIG	text	80			
QEVAL	QEVAL	text	80			
IT.BMI	BMI	float	5			
IT.DIABP	DIABP	integer	3			
IT.FRMSIZE	FRMSIZE	text	8			
IT.HEIGHT	HEIGHT	integer	3			
IT.HR	HR	integer	3			
IT.SYSBP	SYSBP	integer	3			
IT.TEMP	TEMP	float	5			
IT.WEIGHT	WEIGHT	float	5			
IT.WSTCIR	WSTCIR	integer	3			

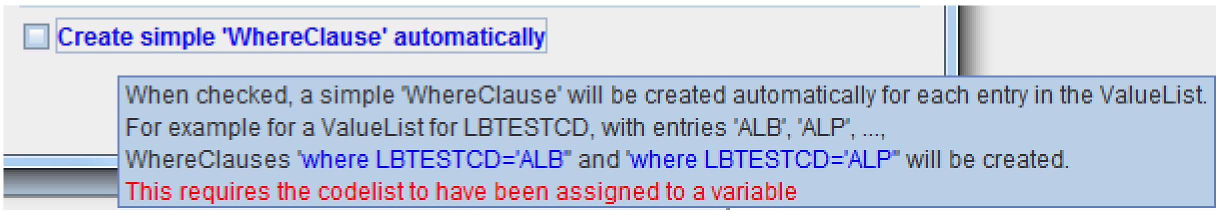
After you have defined where the annotated CRF resides, and how that file is named, you will be able to add the "Origin" to each of the variables, be it domain variables or valuelist variables.

Automatically assigning WhereClauses to ValueList entries

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

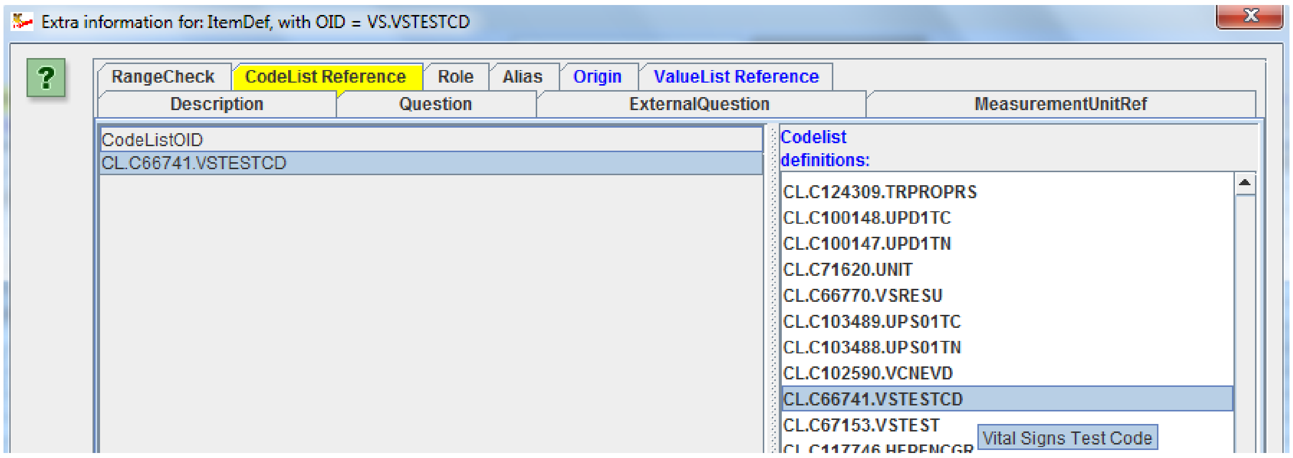

CL.C102590.VCNEVD - Vaccination Evidence Source
CL.C66741.VSTESTCD - Vital Signs Test Code
CL.C67153.VSTEST - Vital Signs Test Name
CL.C117746.HEPENCGR - West Haven Hepatic Encephalopathy Grade
Search
Search
Find Next
Find Previous
Create simple 'WhereClause' automatically
OK
Cancel

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



When checked, a simple 'WhereClause' will be created automatically for each entry in the ValueList.
For example for a ValueList for LBTESTCD, with entries 'ALB', 'ALP', ...,
WhereClauses 'where LBTESTCD='ALB'' and 'where LBTESTCD='ALP'' will be created.
This requires the codelist to have been assigned to a variable

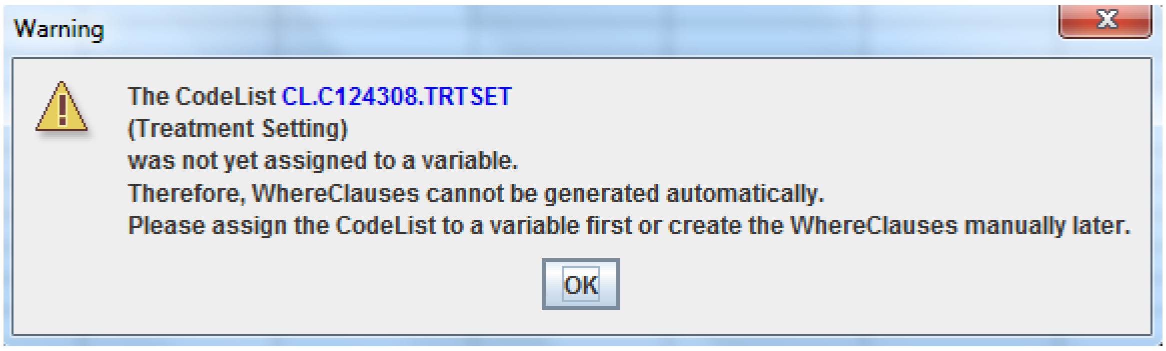
In order to generate such "WhereClauses" you will need to have assigned the codelist to a variable. In the case of "Vital Signs Code" (CL.C66741.VSTESTCD), you will almost surely have assigned this codelist to the variable "VSTESTCD":



CodeListOID
CL.C66741.VSTESTCD

CL.C124309.TRPROPRS
CL.C100148.UPD1TC
CL.C100147.UPD1TN
CL.C71620.UNIT
CL.C66770.VSRESU
CL.C103489.UPS01TC
CL.C103488.UPS01TN
CL.C102590.VCNEVD
CL.C66741.VSTESTCD
CL.C67153.VSTEST

If the codelist was not assigned to any SDTM/SEND/ADaM variable, and the checkbox "Create simple 'WhereClause' automatically" is checked, a warning message will be displayed:



Warning

The CodeList CL.C124308.TRTSET
(Treatment Setting)
was not yet assigned to a variable.
Therefore, WhereClauses cannot be generated automatically.
Please assign the CodeList to a variable first or create the WhereClauses manually later.

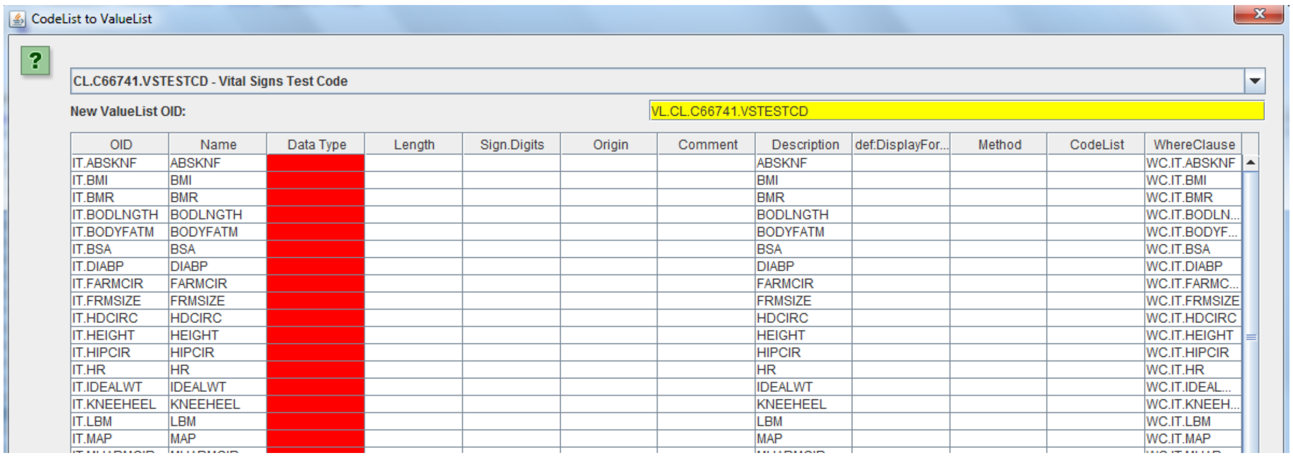
If however, the codelist was already assigned (for example the VSTESTCD codelist to the VSTESTCD variable), all the "WhereClauses" will be generated automatically. These are simple "WhereClauses" with the structure:

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

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

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

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



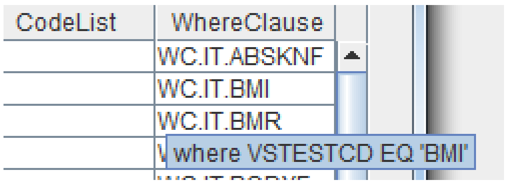
CodeList to ValueList

CL.C66741.VSTESTCD - Vital Signs Test Code

New ValueList OID: VL.CL.C66741.VSTESTCD

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

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

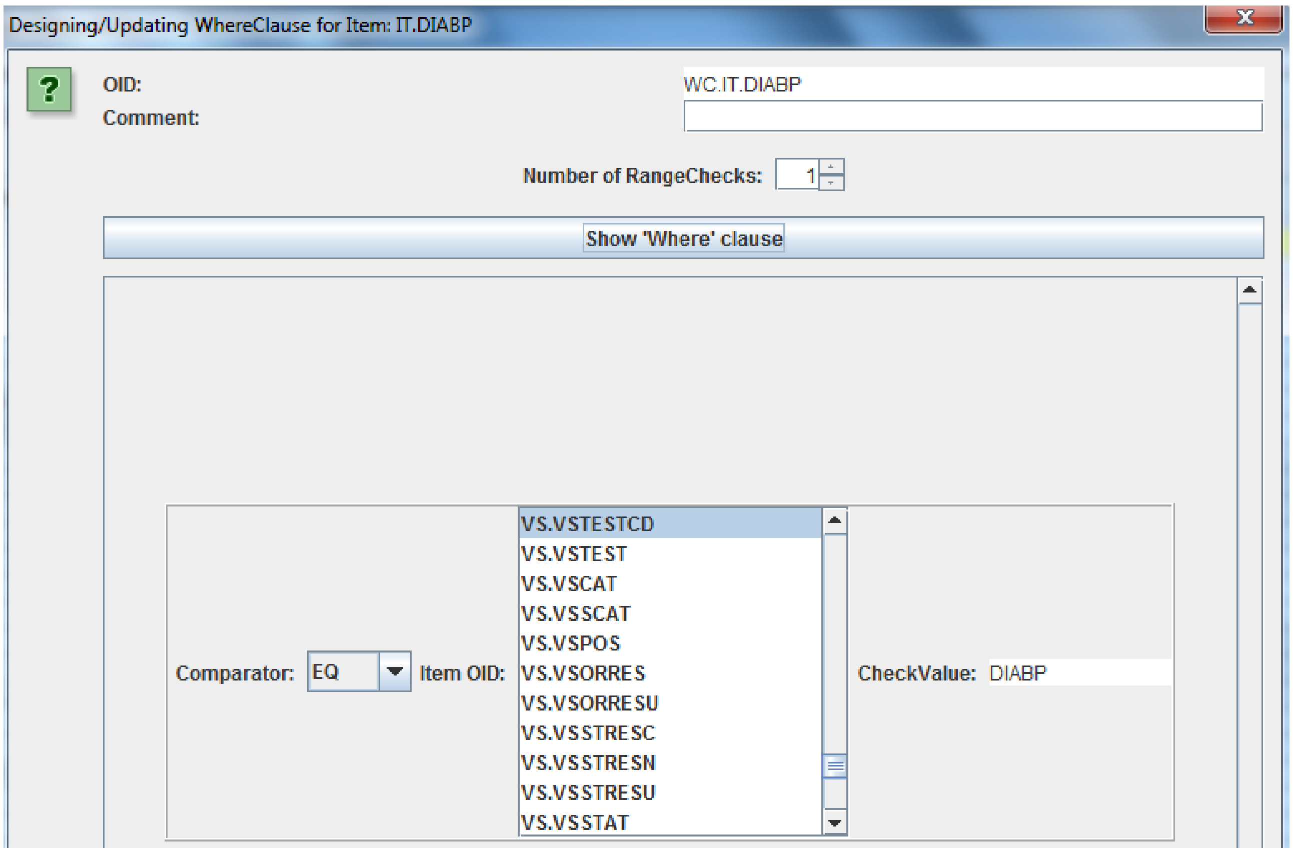


CodeList	WhereClause
	WC.IT.ABSKNF
	WC.IT.BMI
	WC.IT.BMR

where VSTESTCD EQ 'BMI'

for the cell "WC.IT.BMI"

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


Designing/Updating WhereClause for Item: IT.DIABP

OID: WC.IT.DIABP

Comment:

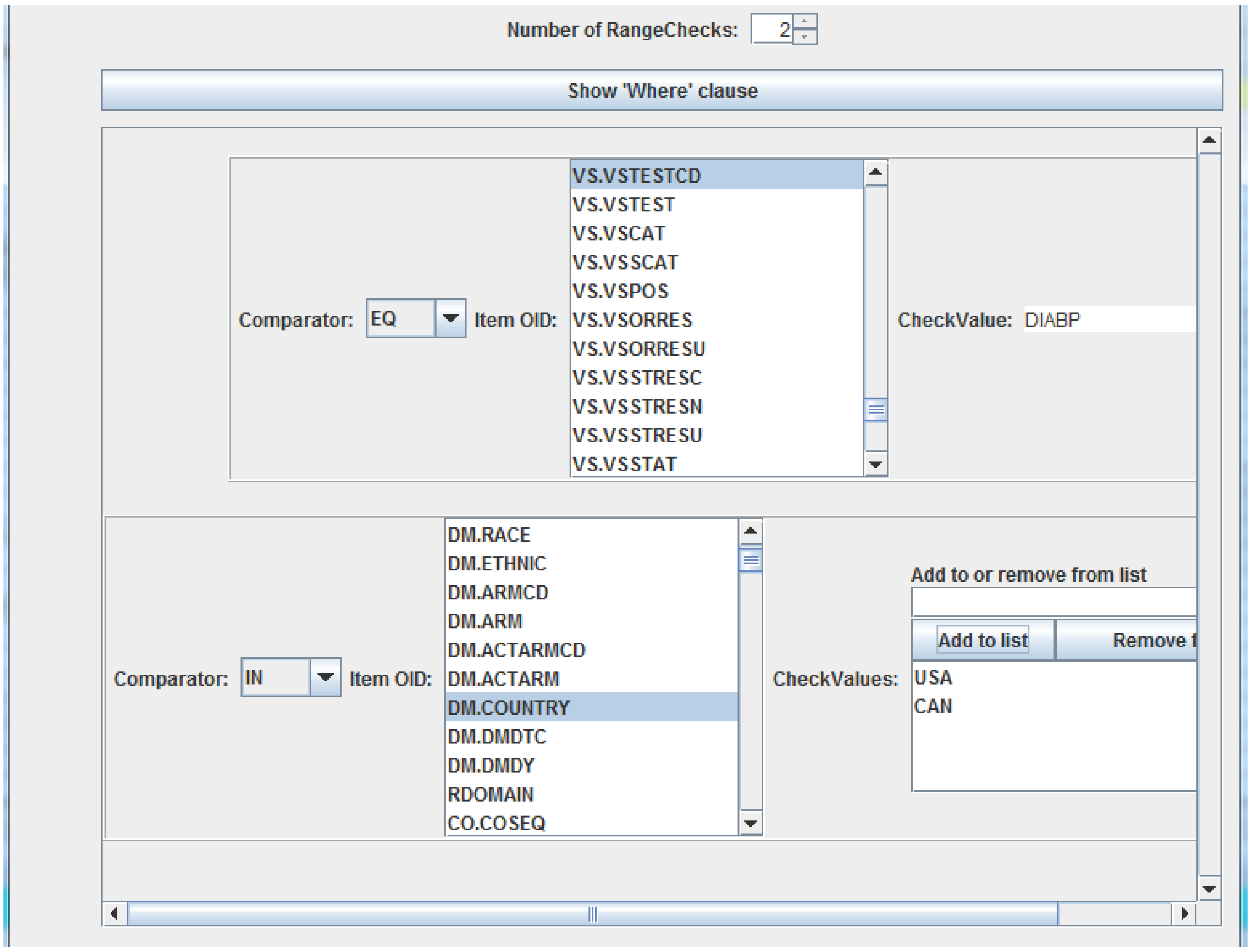
Number of RangeChecks: 1

Comparator: EQ

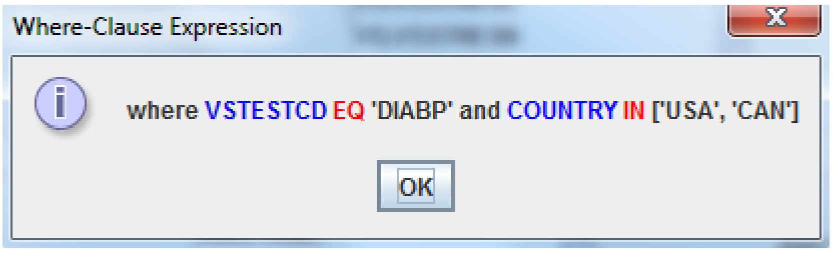
Item OID: VS.VSTESTCD

CheckValue: DIABP

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



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

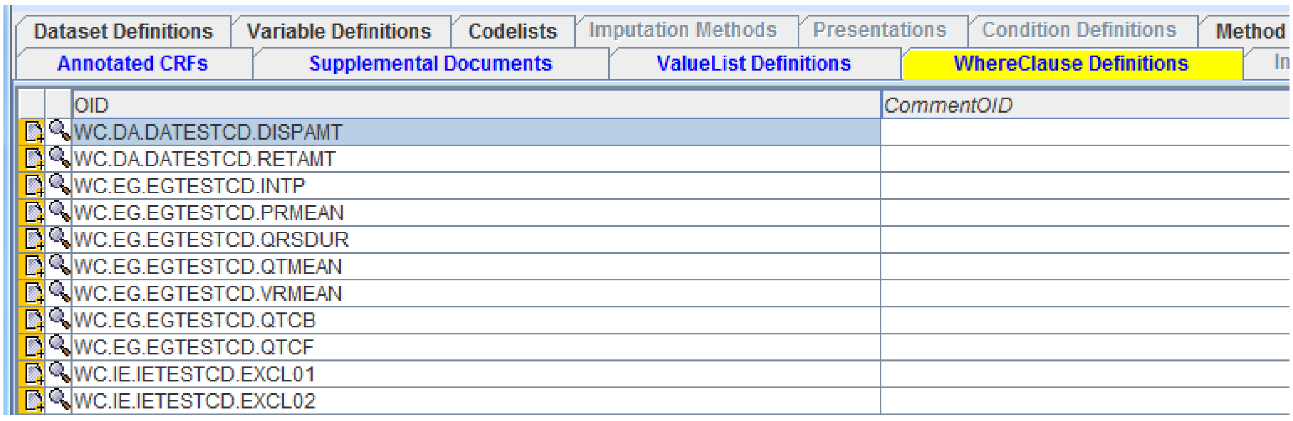


where VSTESTCD EQ 'DIABP' and COUNTRY IN ['USA', 'CAN']

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

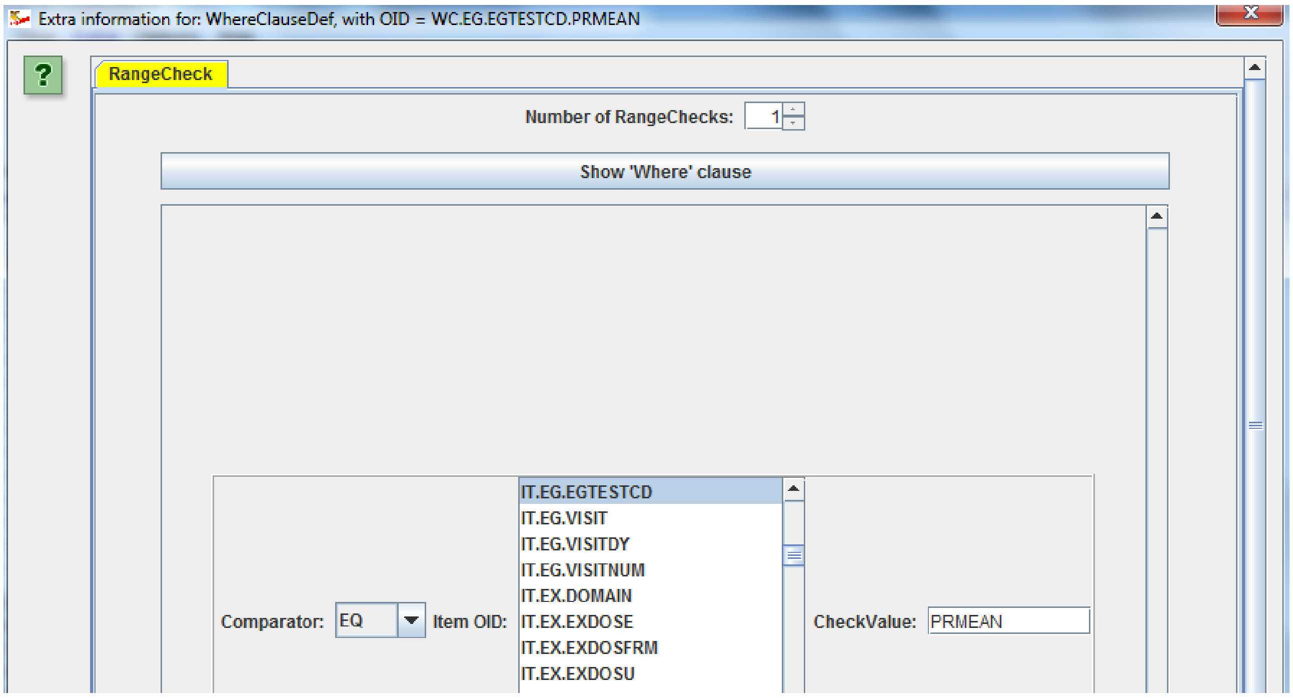
After clicking "OK" until one gets in the main window, and then selecting the tab "WhereClause Definition", one sees the newly generated "WhereClauses" in addition to any already generated:

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



OID	CommentOID
WC.DA.DATESTCD.DISPAMT	
WC.DA.DATESTCD.RETAMT	
WC.EG.EGTESTCD.INTP	
WC.EG.EGTESTCD.PRMEAN	
WC.EG.EGTESTCD.QRSDUR	
WC.EG.EGTESTCD.QTMEAN	
WC.EG.EGTESTCD.VRMEAN	
WC.EG.EGTESTCD.QTCB	
WC.EG.EGTESTCD.QTCF	
WC.IE.IETESTCD.EXCL01	
WC.IE.IETESTCD.EXCL02	

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


Extra information for: WhereClauseDef, with OID = WC.EG.EGTESTCD.PRMEAN

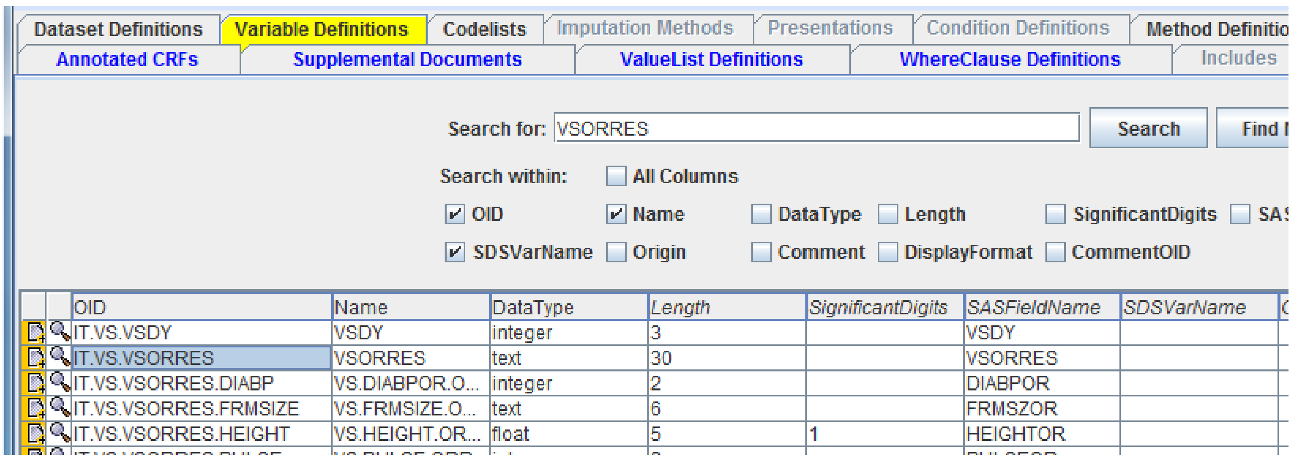
Number of RangeChecks: 1

Comparator: EQ
Item OID: IT.EG.EGTESTCD
IT.EG.VISIT
IT.EG.VISITDY
IT.EG.VISITNUM
IT.EX.DOMAIN
IT.EX.EXDOSE
IT.EX.EXDOSFRM
IT.EX.EXDOSU
CheckValue: PRMEAN

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

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

This is done when using the table with variables, e.g. for VSORRES:

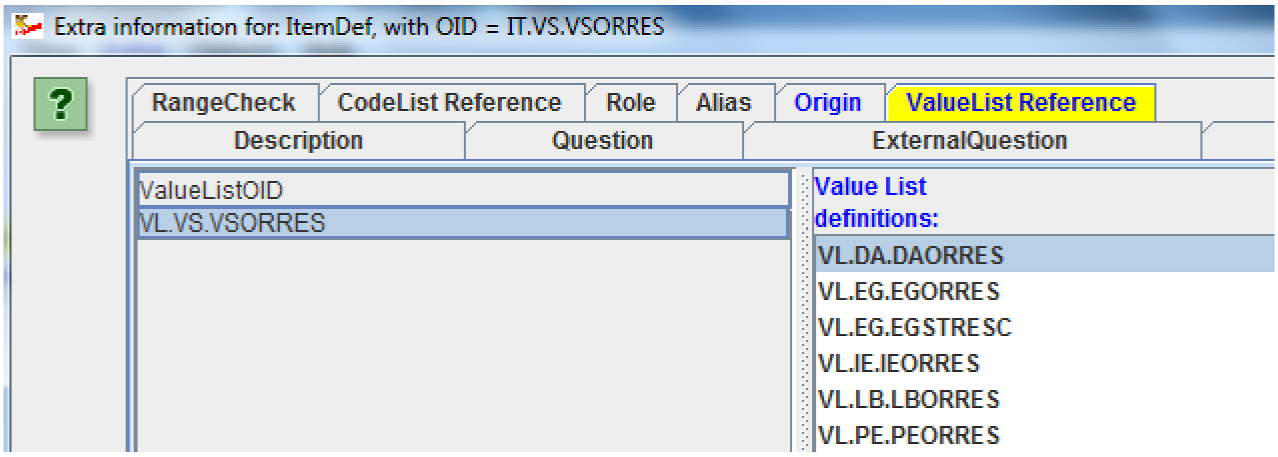

Dataset Definitions | Variable Definitions | Codelists | Imputation Methods | Presentations | Condition Definitions | Method Definitio
Annotated CRFs | Supplemental Documents | ValueList Definitions | WhereClause Definitions | Includes

Search for: VSORRES Search Find I

Search within: ☐ All Columns
☑ OID ☑ Name ☐ DataType ☐ Length ☐ SignificantDigits ☐ SA
☑ SDSVarName ☐ Origin ☐ Comment ☐ DisplayFormat ☐ CommentOID

OID	Name	DataType	Length	SignificantDigits	SASFieldName	SDSVarName
IT.VS.VSDY	VSDY	integer	3		VSDY	
IT.VS.VSORRES	VSORRES	text	30		VSORRES	
IT.VS.VSORRES.DIABP	VS.DIABPOR.O...	integer	2		DIABPOR	
IT.VS.VSORRES.FRMSIZE	VS.FRMSIZE.O...	text	6		FRMSZOR	
IT.VS.VSORRES.HEIGHT	VS.HEIGHT.OR...	float	5	1	HEIGHTOR	

and when clicking the "Add Information" ("+") icon on the left of "VSORRES", and choosing the tab "ValueList Reference":


Extra information for: ItemDef, with OID = IT.VS.VSORRES

RangeCheck | CodeList Reference | Role | Alias | Origin | ValueList Reference
Description | Question | ExternalQuestion

ValueListOID
VL.VS.VSORRES

Value List definitions:
VL.DA.DAORRES
VL.EG.EGORRES
VL.EG.EGSTRESC
VL.IE.IEORRES
VL.LB.LBORRES
VL.PE.PEORRES

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

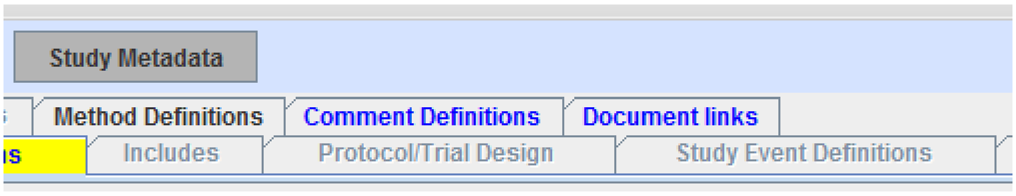
P.S. As ValueLists only have an OID (but no "Name" attribute), it may be useful to assign a meaningful OID to each entry in the "ValueList" table.

Annotated CRF and Supplemental Documents

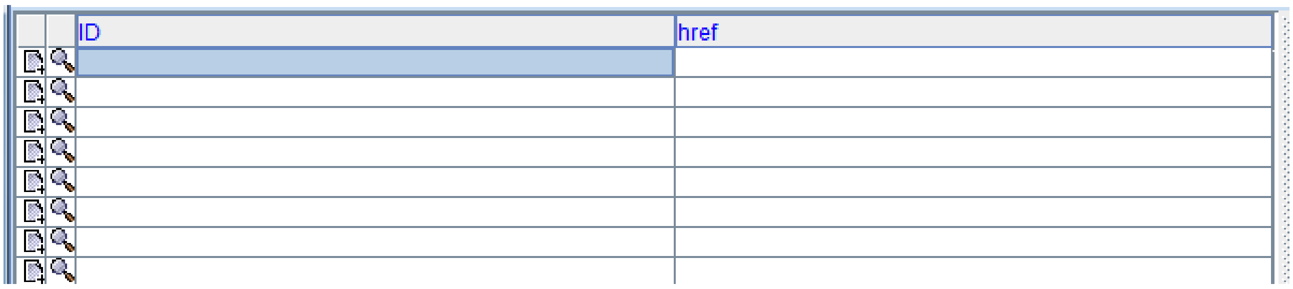
When doing a regulatory submission, you are still required to provide an "annotated CRF" in PDF format[15]. Also, you will often want to supply supplemental documents, like a "reviewers guide", also in PDF format.

In define.xml, you can provide links to such documents, so that when the reviewer inspects your define.xml in the browser, a single click suffices to open the annotated CRF or supplemental document, ideally on the page of interest.
This is taken care of in define.xml documents by so-called "def:leaf" elements, allowing you to specify a reference ("href") to the document of interest.

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

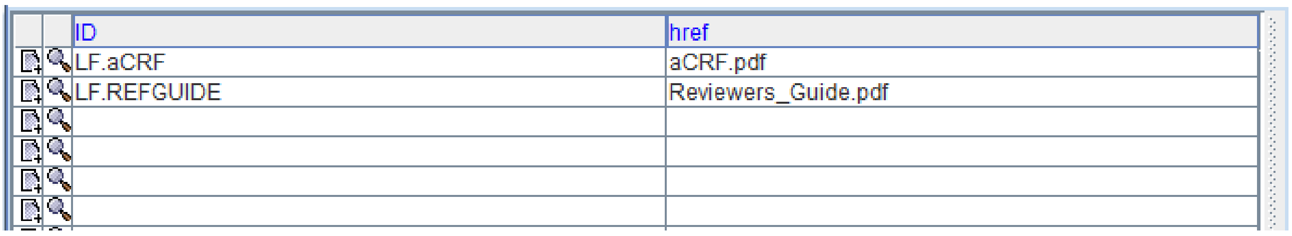


The following table is displayed:



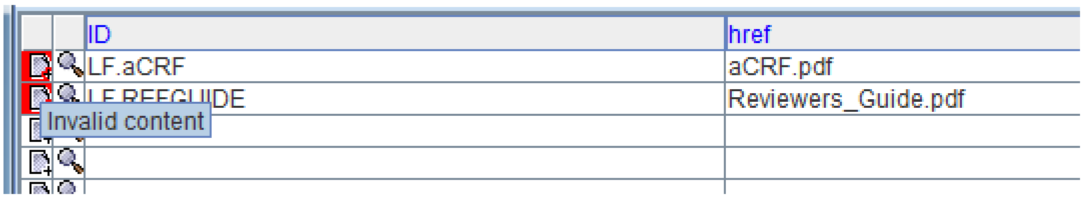
ID	href

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



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

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

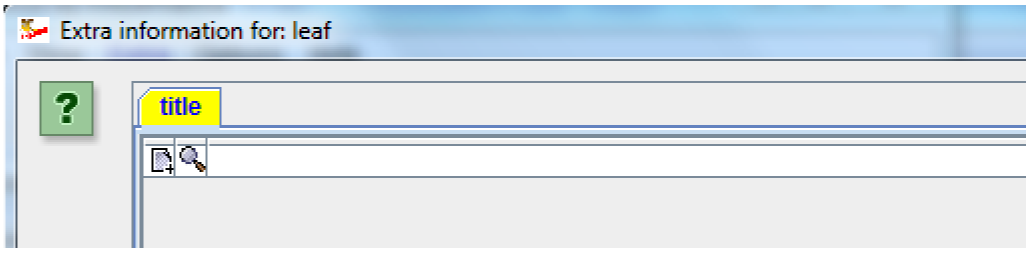


ID	href
LF.aCRF	aCRF.pdf
LF.REFGUIDE	Reviewers_Guide.pdf

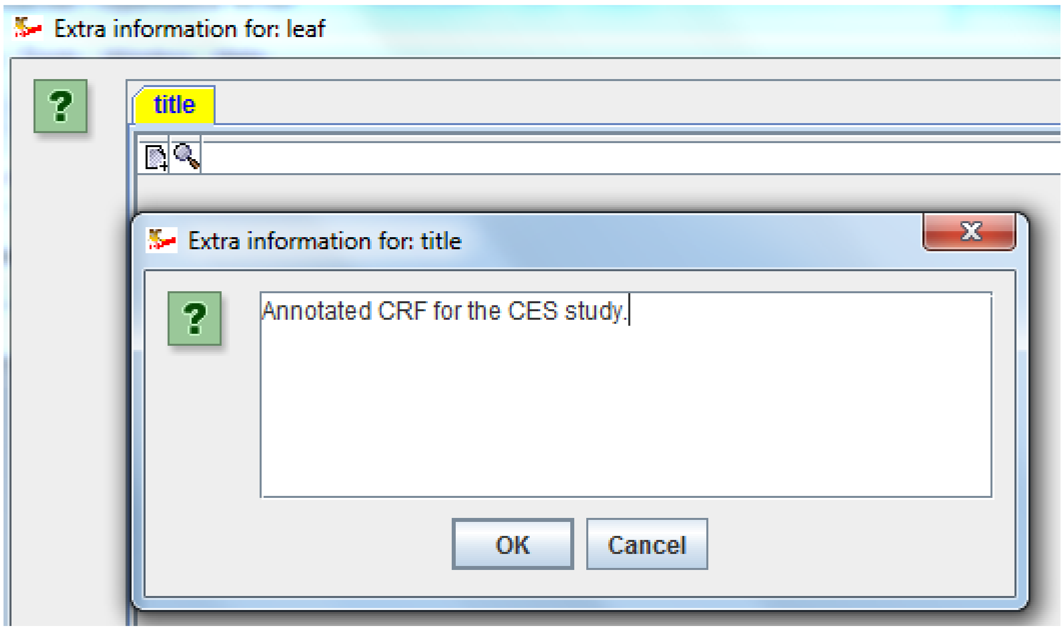
So you will need to provide additional content. This can be provided by clicking on the "+" icon,

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

leading to:

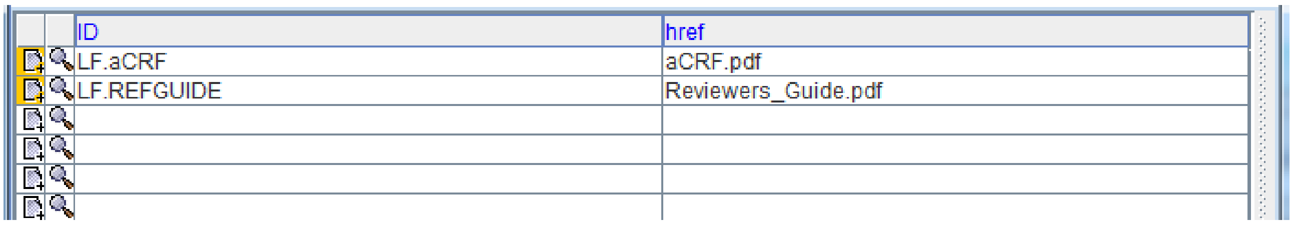


prompting you to provide a title by clicking the "+" icon again:



Annotated CRF for the CES study.

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

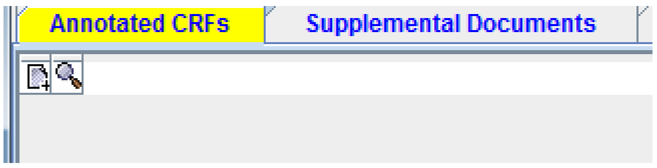


		ID	href
		LF.aCRF	aCRF.pdf
		LF.REFGUIDE	Reviewers_Guide.pdf

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

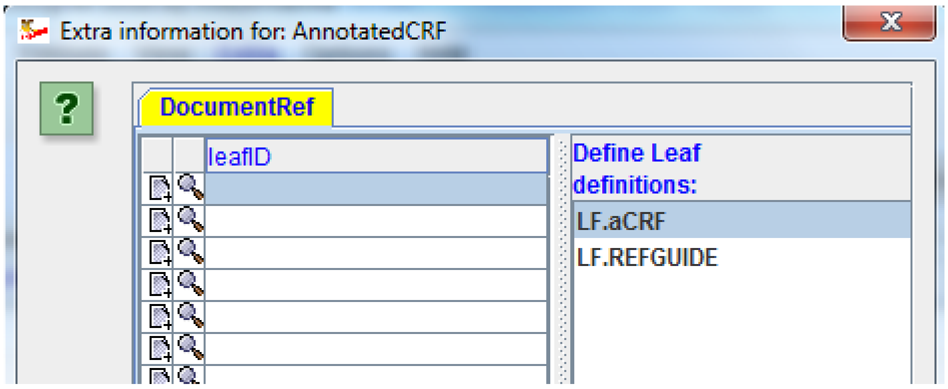
We now only defined the "leafs", i.e. the hyperlinks and a title for each of the documents, but the define.xml still requires us to state explicitely which of these is the "annotated CRF" (define.xml is machine-readable and machines should not rely on naming conventions only), and which documents belong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

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

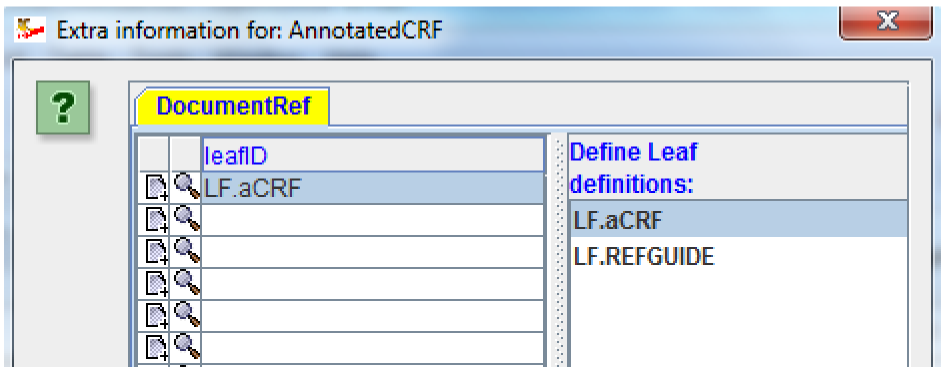


Currently, it is only allowed to provide a single PDF with all annotated CRFs in it (separate documents, e.g. one for each CRF, is not allowed).

Clicking the "+" icon leads to:

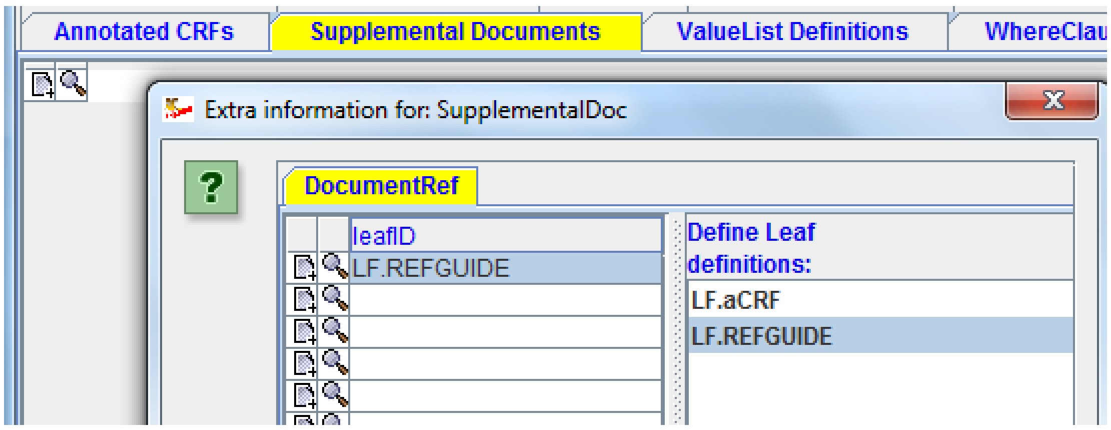


and one can now "drag-and-drop" a "leaf" from the right to the first cell on the left, with the result:



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

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



Remark that you can add more than one supplemental document.

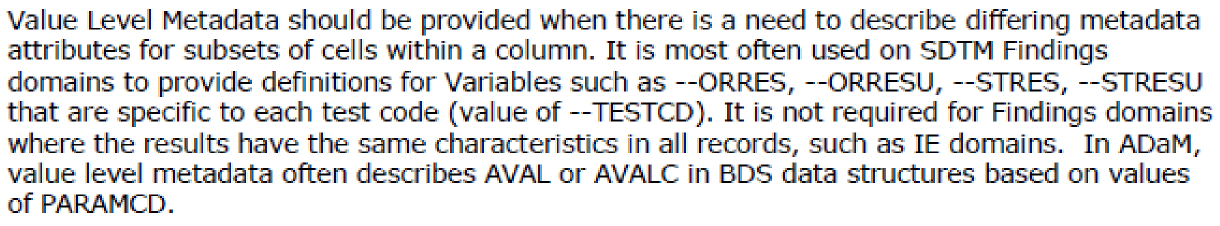
Assigning valuelists to SDTM/SEND/ADaM variables

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

Essentially, most findings domains have "hypervertical" structure, with a data model based on the "Entity - Attribute - Value" model (EAV model) This means that we have a single "entity" column (defining what the row "is about"). In the findings domains, this is usually the --TESTCD column[16]. There then are a number of attributes (like additional names and synonyms, but most importantly the --ORRESU (original result unit) and the --STRESU (standardized result unit). The "value" in EAV is delivered by the --ORRES column and by the --STRESC column and or the --STRESN column[17].

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

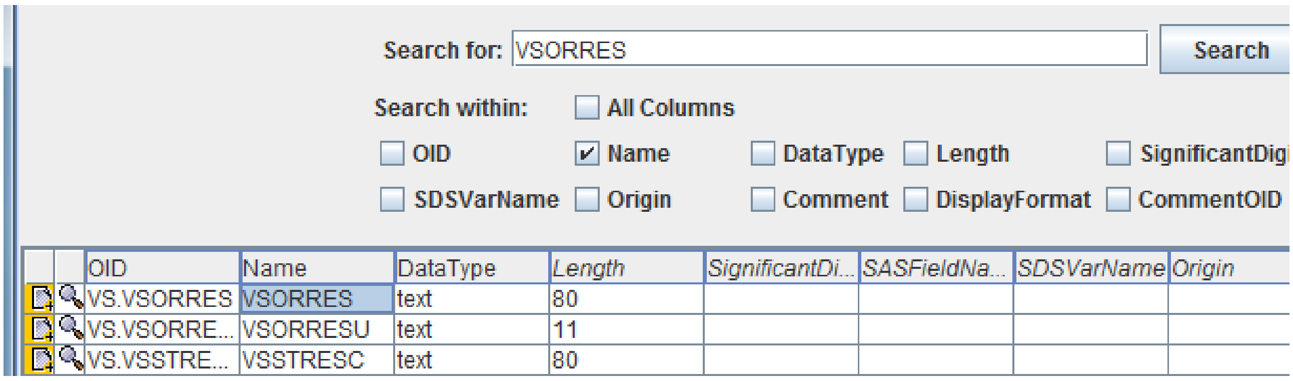
The define.xml 2.0 specification is very clear about this (section 4.4):


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

In our case, we provided information about what length of result can be expected, what datatype, and whether the result is coded, depending on the value of the VSTESTCD. So we must assign the valuelist "VL.CL.C66741.VSTESTCD" to VSORRES.
If the value of the standardized result (VSSTRESC/VSSTRESN) is just copied from VSORRES, we can additionally also assign this valuelist to VSSTRESC and/or VSSTRESN.

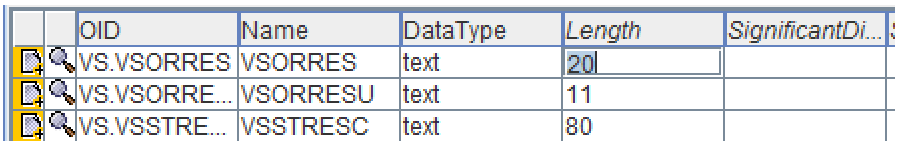
In order to assign the valuelist "VL.CL.C66741.VSTESTCD" to VSORRES, select the tab "Variable Definitions" and look for "VSORRES" (one can use the "Show Search Panel"). One easily finds the row:

[16]A better choice would have been the --LOINC column, at least for LB and VS.
[17]The reason that two columns are defined for the standardized result is due to the extremely old SAS-XPT format, in which columns can only either be "numeric" or "character".


Search for: VSORRES
Search within: All Columns
[ ] OID [✓ Name] [ ] DataType [ ] Length [ ] SignificantDig
[ ] SDSVarName [ ] Origin [ ] Comment [ ] DisplayFormat [ ] CommentOID

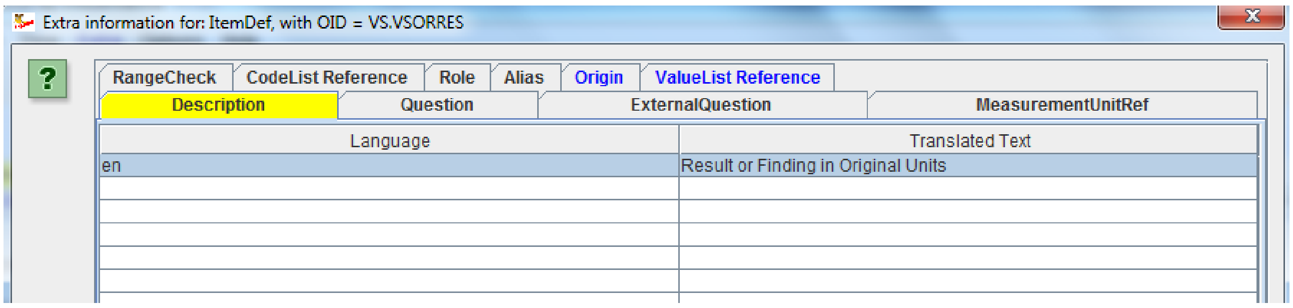
OID	Name	DataType	Length	SignificantDi...	SASFieldNa...	SDSVarName	Origin
VS.VSORRES	VSORRES	text	80				
VS.VSORRE...	VSORRESU	text	11				
VS.VSSTRE...	VSSTRESC	text	80				

Depending on how the variable definitions were created, you will still want to assign another "Length". The above picture shows a maximal length of "80" coming from the template SDTM. If you generated the variable definitions from a SAS-XPT file, the maximal length will probably already be correct, unless of course that you generated your SAS-XPT files with fields that are "too broad", something that the FDA doesn't like at all[18].
Suppose that the longest result for VSORRES was 20 characters. In that case we need to set the "Length" to 20, and also take care that this field is not longer than 20 bytes in the SAS-XPT file:


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

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

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


Extra information for: ItemDef, with OID = VS.VSORRES

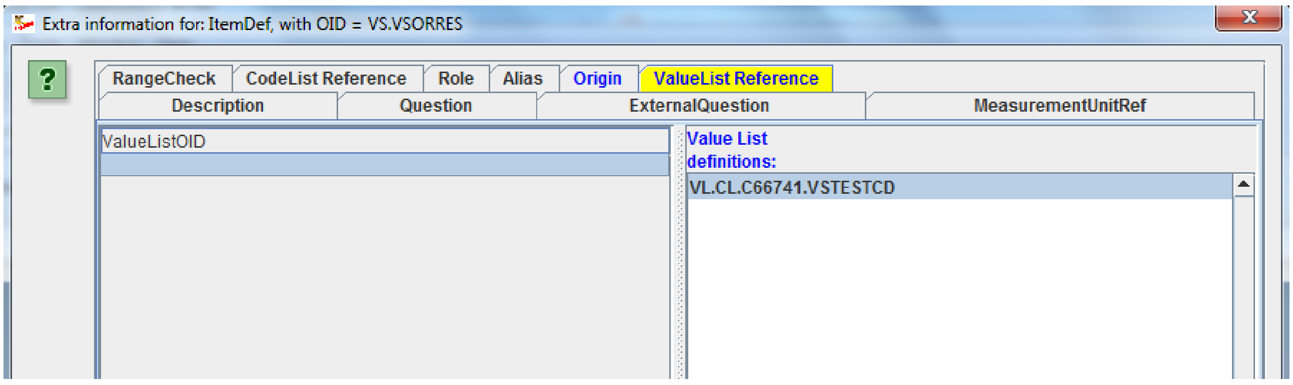
Language	Translated Text
en	Result or Finding in Original Units

In this case, the "Label" was already added (in define.xml 2.0, it is added to the "Description" element). Be sure that the label for any domain variable is exactly identical (case sensitive!) to what is defined in the SDTM-IG, SEND-IG, or ADaM-IG.
Unfortuanetely, there is no room for any deviation at all here, even if a slightly different label would better explain what the variable is about. See the article "http://cdiscguru.blogspot.com/2015/12/sdtm-labels-freedom-or-slavery.html" for further details.

Getting the label 100% correct can especially be challenging in case of additional variables that come from the model rather than from the implementation guide (like additional timing variables) and when coming from the SAS-XPT files themselves.

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

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

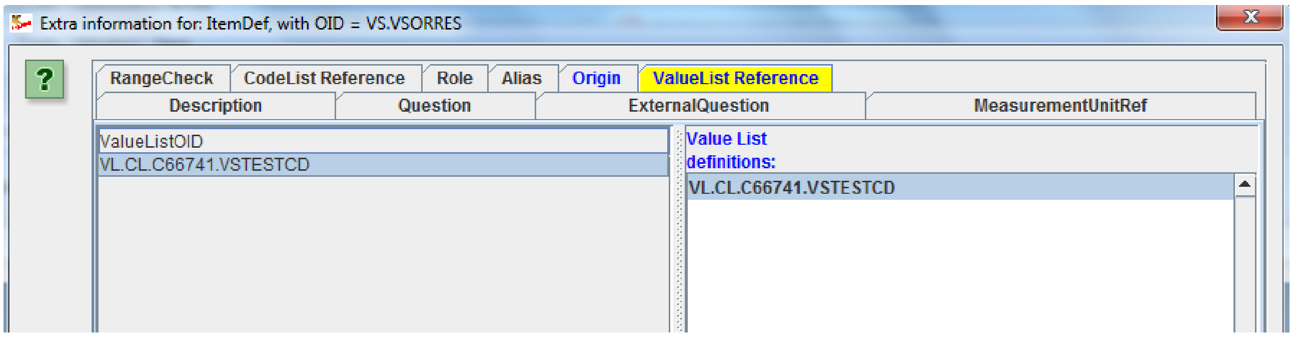


Extra information for: ItemDef, with OID = VS.VSORRES

ValueListOID

Value List definitions:
VL.CL.C66741.VSTESTCD

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



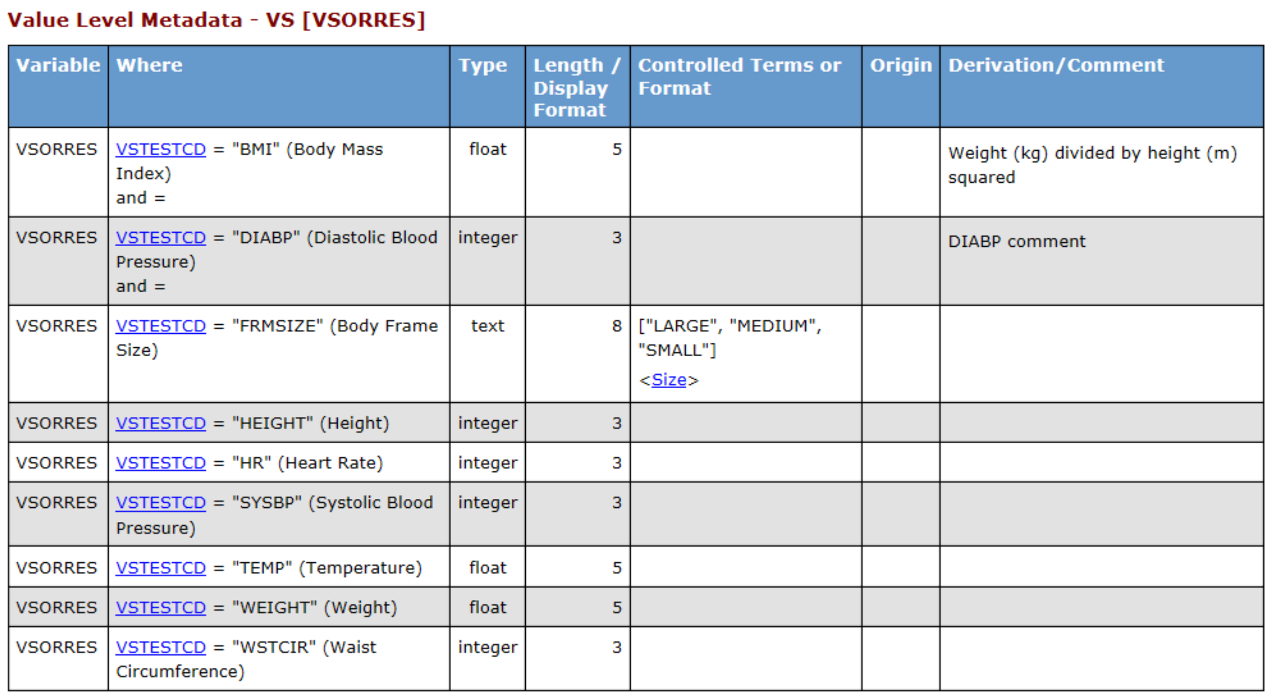
Extra information for: ItemDef, with OID = VS.VSORRES

ValueListOID
VL.CL.C66741.VSTESTCD

Value List definitions:
VL.CL.C66741.VSTESTCD

and you are done...

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



Value Level Metadata - VS [VSORRES]

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

Assigning an Origin to variables

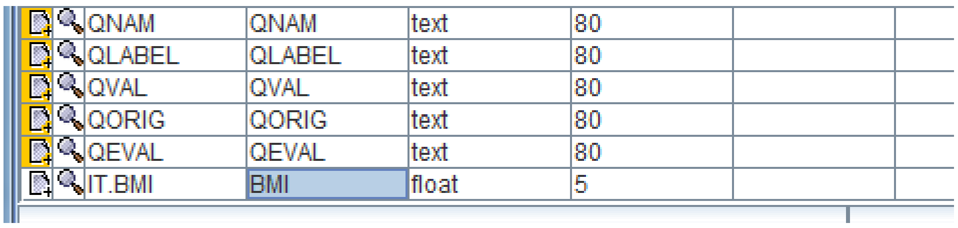
Every data point in a submission has an origin. It is of utmost importance for traceability that this information is provided to the reviewer. It can be that the data point comes from the CRF, it can come from the protocol, or it can be assigned or derived (e.g. all --DY variables). It can also come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

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

On the other hand, "VSORRES" will need to be assigned on the "valuelist level", as there usually is no field "vital sign original result" on the CRF, but there are fields "systolic blood pressure", "systolic blood pressure", "height", "weight" and so on.
So it does not make sense (it even would probably be an error) to assign an Origin to VSORRES. However, as we assigned a valuelist to "VSORRES", this is already an indication that origins will be assigned on the valuelist level, i.e. for "DIABP", "SYSBP", "HEIGHT" etc..

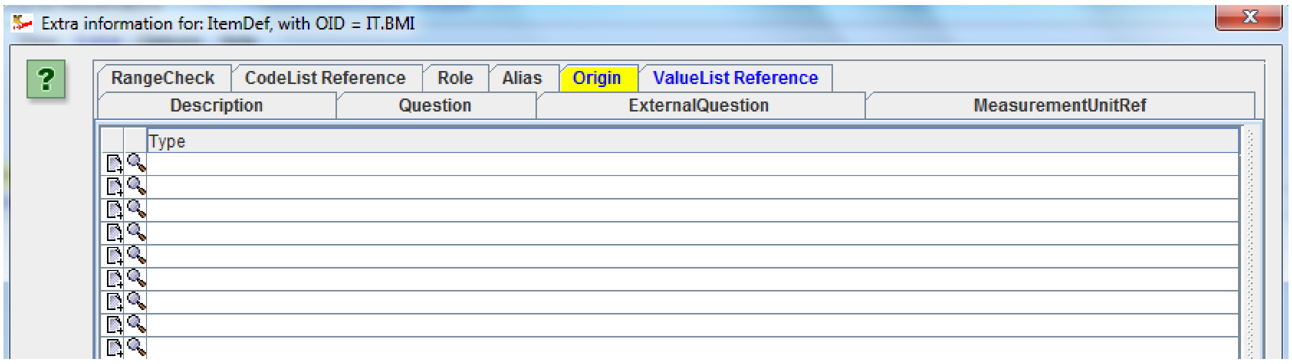
So in our variable list, we will leave the value for "Origin" empty for VSORRES, but will assign origins for the valuelist variables "DIABP", "SYSBP", "HEIGHT" etc..

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



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

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



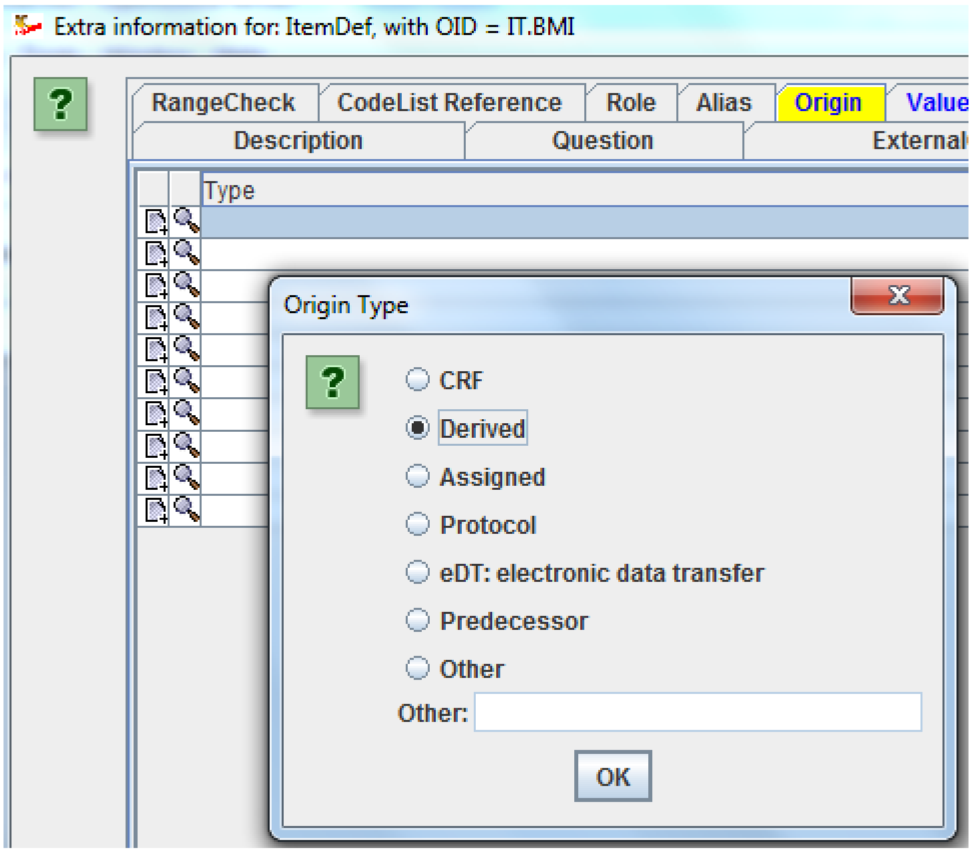
Extra information for: ItemDef, with OID = IT.BMI

Although the define.xml XML-Schema allows you to add more than one origin, this is essentially not allowed. If you need to assign more than one "origin", this usually means that you did not well enough develop your valuelist.
For example, if you have have different glucose values, and one set was delivered electronically by lab 1, and the other set was captured on the CRF, you may want to have two different variables "IT.GLUC.LAB1" and "IT.GLUC.LAB2" so that you can assign "electronic data transfer" to the former as being the origin, and "CRF" to the latter[19]. In the "where clause" you will then need to

[19]Unfortunately, the situation for lab values can be even much more complicated, as CDISC and the FDA still refuse to

differentiate e.g. by the "DM.SITE" or "LB.NAM" (laboratory name) variable.

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


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

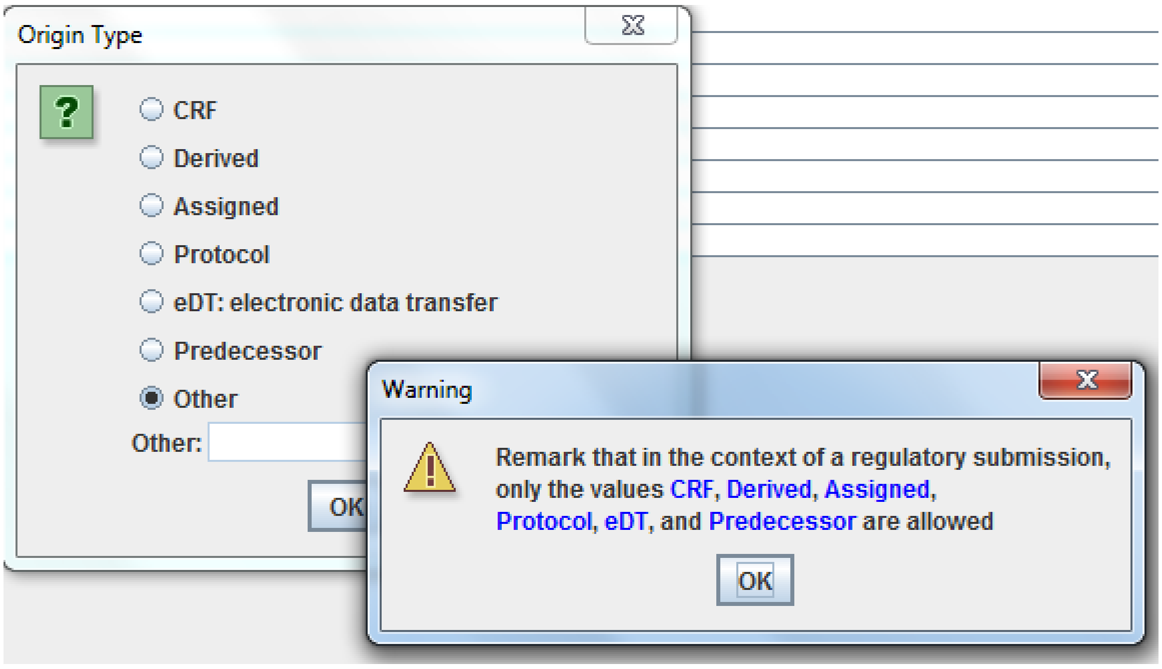
Origin Type

CRF [✓ Derived] Assigned Protocol eDT: electronic data transfer Predecessor Other

Other:

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

allow the LOINC code of the lab test as the unique identifier.


Origin Type

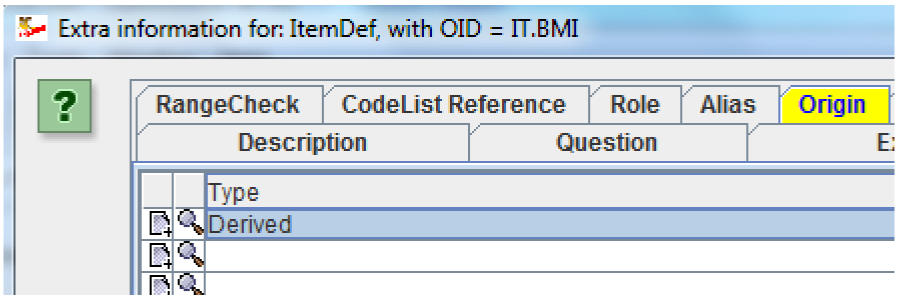
- CRF
- Derived
- Assigned
- Protocol
- eDT: electronic data transfer
- Predecessor
- [✓ Other]

Other:

Warning

Remark that in the context of a regulatory submission, only the values CRF, Derived, Assigned, Protocol, eDT, and Predecessor are allowed

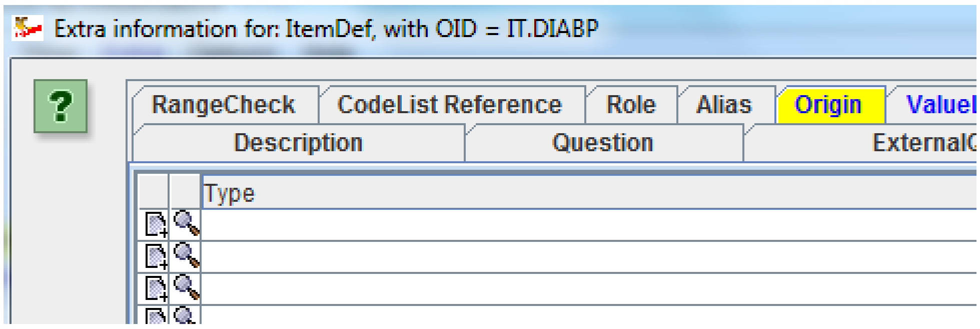
In our case however, we set the "Origin Type" to derived, leading to:


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

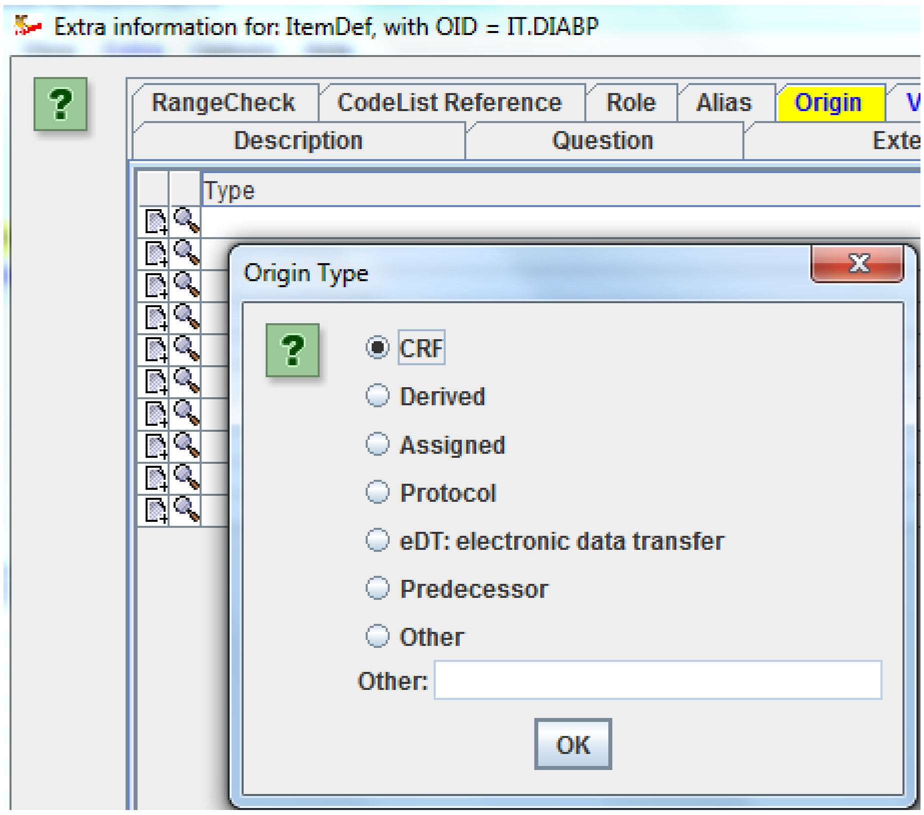
Type
Derived

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

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


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

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

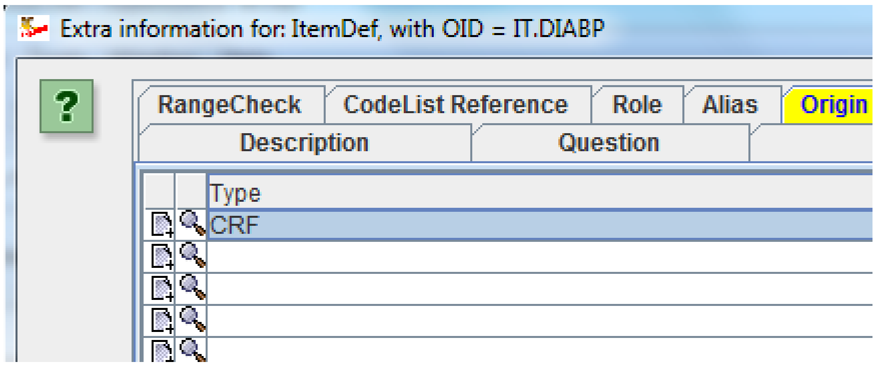

Extra information for: ItemDef, with OID = IT.DIABP

Origin Type

[✓ CRF]
Derived
Assigned
Protocol
eDT: electronic data transfer
Predecessor
Other

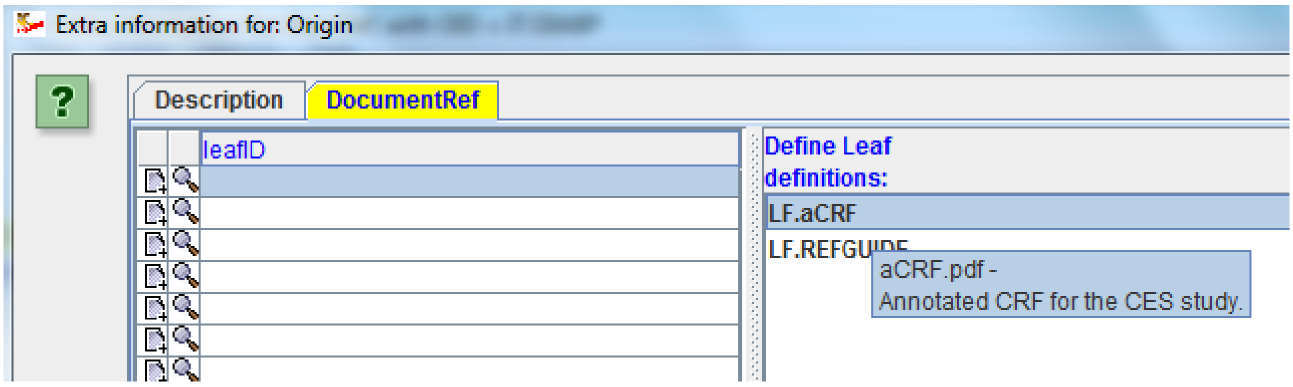
Other:

leading to:


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

Type
CRF

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

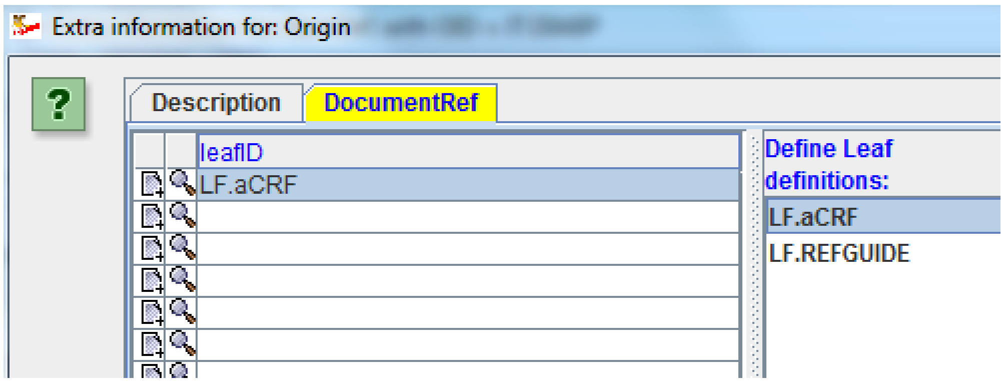

Extra information for: Origin

Define Leaf definitions:

LF.aCRF
LF.REFGUIDE

aCRF.pdf - Annotated CRF for the CES study.

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

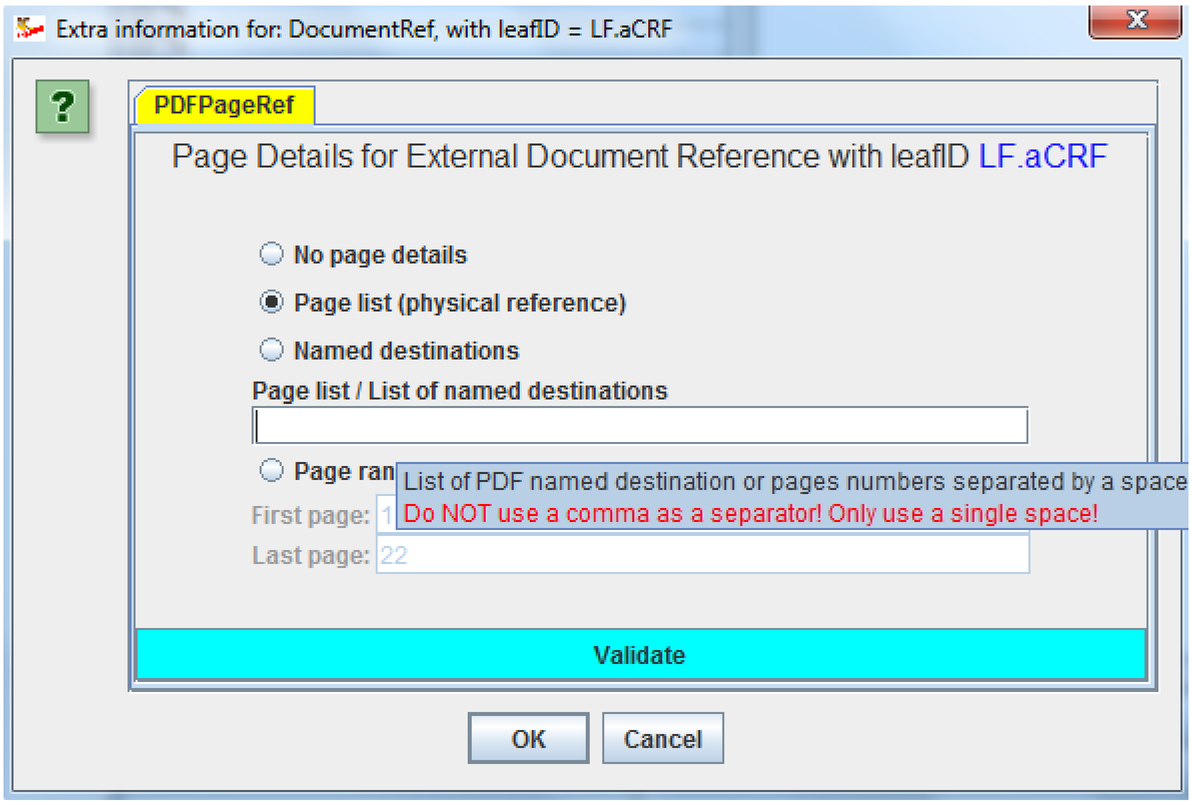

Extra information for: Origin

Description | DocumentRef

	leafID
	LF.aCRF

Define Leaf definitions:
LF.aCRF
LF.REFGUIDE

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


Extra information for: DocumentRef, with leafID = LF.aCRF

PDFPageRef

Page Details for External Document Reference with leafID LF.aCRF

No page details [✓ Page list (physical reference)] Named destinations

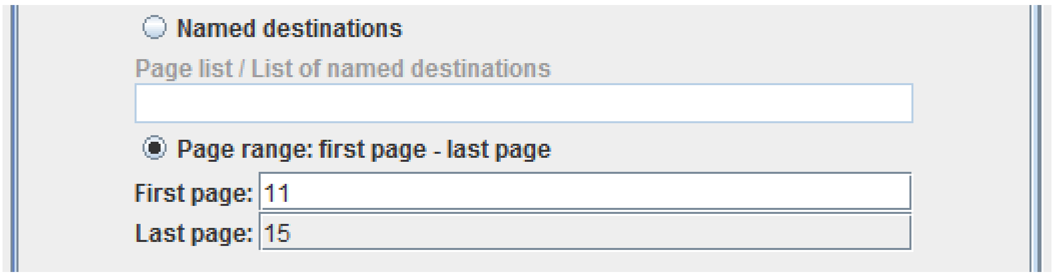
Page list / List of named destinations:

Page range

List of PDF named destination or pages numbers separated by a space
Do NOT use a comma as a separator! Only use a single space!

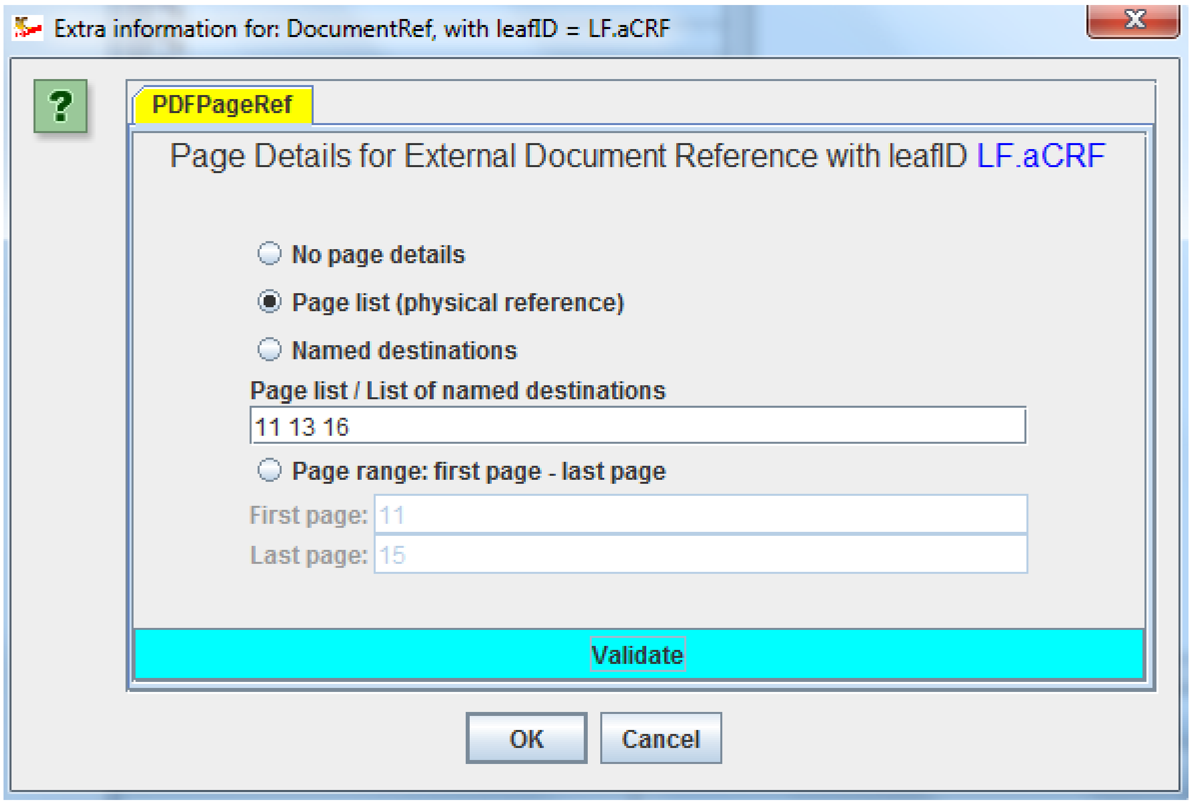
First page: 1
Last page: 22

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


Named destinations
Page list / List of named destinations
[✓ Page range: first page - last page]
First page: 11
Last page: 15

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

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


Extra information for: DocumentRef, with leafID = LF.aCRF

PDFPageRef

Page Details for External Document Reference with leafID LF.aCRF

No page details [✓ Page list (physical reference)] Named destinations

Page list / List of named destinations: 11 13 16

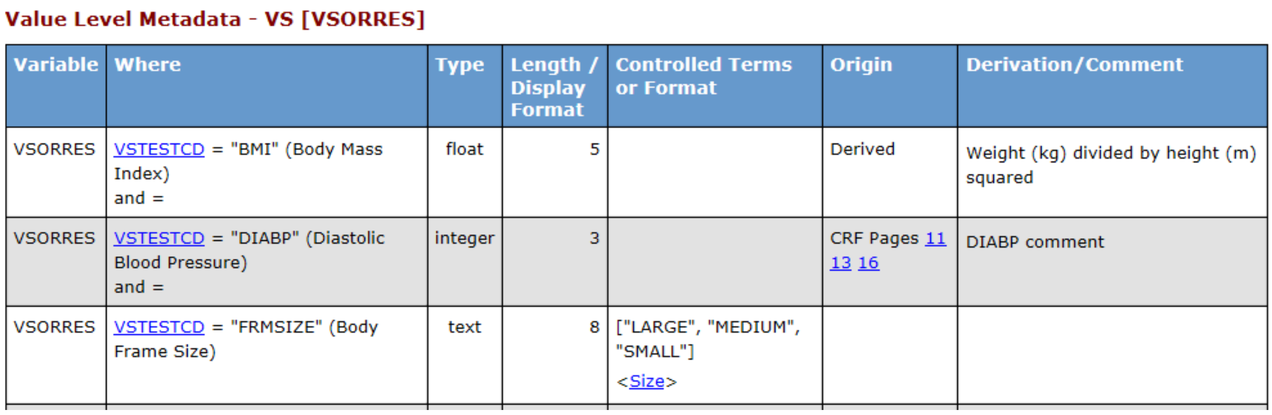
Page range: first page - last page

First page: 11

Last page: 15

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

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


Value Level Metadata - VS [VSORRES]

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

When the user than e.g. clicks on "11" in "CRF Pages", then the browser should automatically open the "annotated CRF PDF" on page 11[20].
Like this, one should essentially assign an "Origin" to every variable in the define.xml. If the variable is a domain variable to which a valuelist has been assigned, one will usually assign the "Origin" to the value-level variables rather than on the parent domain level variable (as was the case for VSORRES).

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

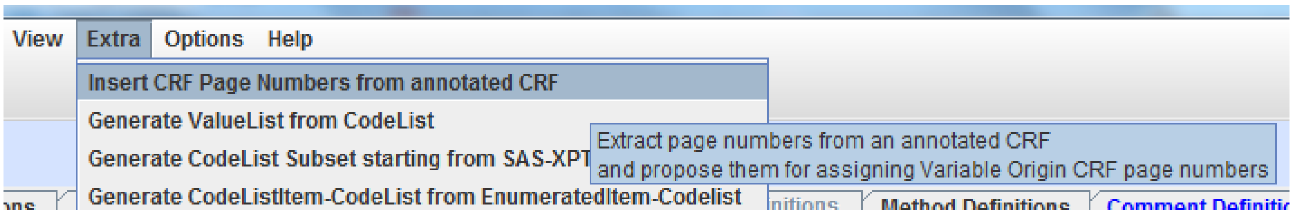
[20]Remark that this functionality is (or must be) provided by the stylesheet. It is not programmed in the define.xml itself.

Retrieving PDF Page Numbers from the annotated CRF

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

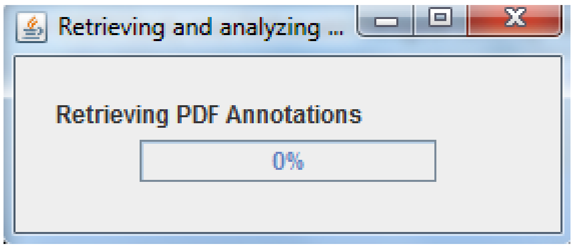
Of course, you will first need to add the location of the annotated CRF to the define.xml using the menus "Add - Document Link" and "Add - Annotated CRF".

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

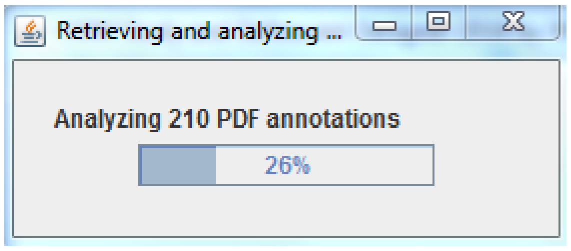

View
Extra
Options
Help
Insert CRF Page Numbers from annotated CRF
Generate ValueList from CodeList
Generate CodeList Subset starting from SAS-XPT
Generate CodeListItem-CodeList from EnumeratedItem-Codelist
Method Definitions
Comment Definitic

Extract page numbers from an annotated CRF
and propose them for assigning Variable Origin CRF page numbers

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

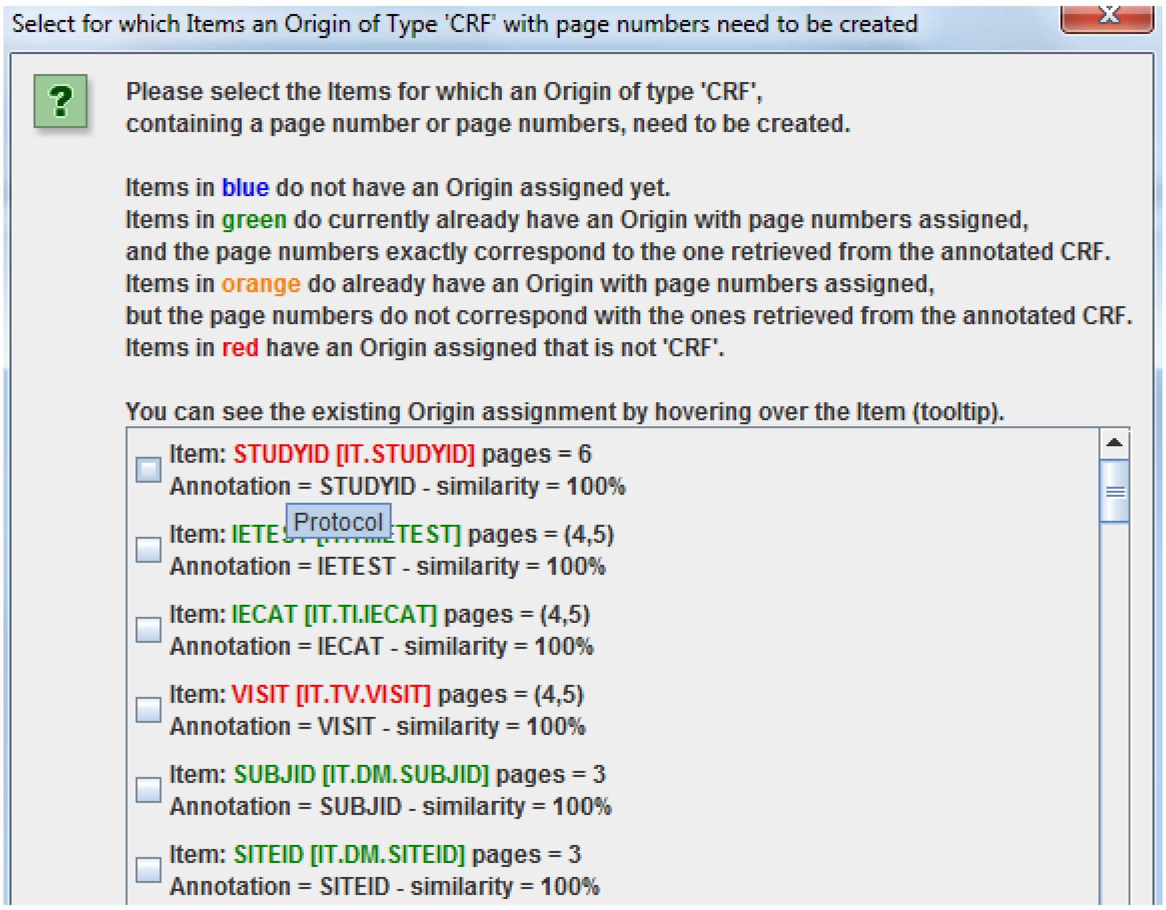


It will then analyze them, comparing them with all define.xml variable names that have been defined, but also with all "WhereClause" definitions.



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

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


Select for which Items an Origin of Type 'CRF' with page numbers need to be created

Please select the Items for which an Origin of type 'CRF',
containing a page number or page numbers, need to be created.

Items in blue do not have an Origin assigned yet.
Items in green do currently already have an Origin with page numbers assigned,
and the page numbers exactly correspond to the one retrieved from the annotated CRF.
Items in orange do already have an Origin with page numbers assigned,
but the page numbers do not correspond with the ones retrieved from the annotated CRF.
Items in red have an Origin assigned that is not 'CRF'.

You can see the existing Origin assignment by hovering over the Item (tooltip).

- [ ] Item: STUDYID [IT.STUDYID] pages = 6
  Annotation = STUDYID - similarity = 100%
- [ ] Item: IETES Protocol TEST] pages = (4,5)
  Annotation = IETEST - similarity = 100%
- [ ] Item: IECAT [IT.TI.IECAT] pages = (4,5)
  Annotation = IECAT - similarity = 100%
- [ ] Item: VISIT [IT.TV.VISIT] pages = (4,5)
  Annotation = VISIT - similarity = 100%
- [ ] Item: SUBJID [IT.DM.SUBJID] pages = 3
  Annotation = SUBJID - similarity = 100%
- [ ] Item: SITEID [IT.DM.SITEID] pages = 3
  Annotation = SITEID - similarity = 100%

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

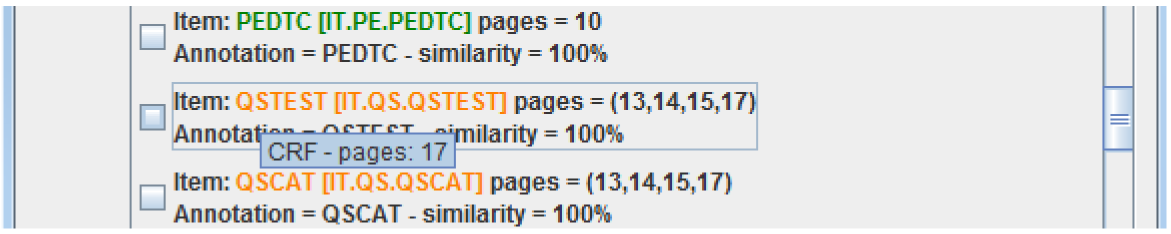
When the Item is colored green, this means that the Origin was already assigned in the define.xml as of type "CRF" and the page number(s) or the page range[21] exactly match with the page numbers retrieved from the annotated CRF. In this case you will probably not want to reassign the page number(s) in the define.xml.

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

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

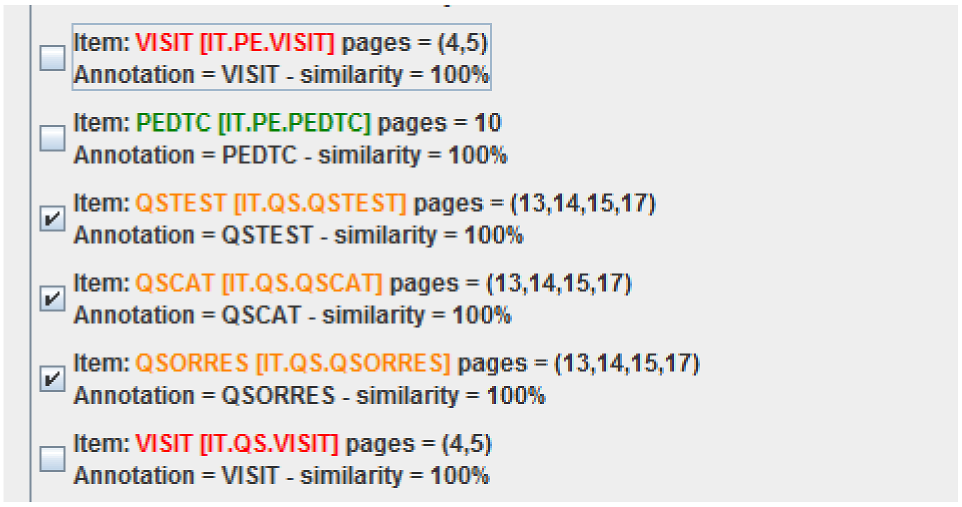
In case the Item is colored orange, this means that page numbers were retrieved from the annotated CRF, but these do not coincide with the ones that were already assigned. For example:

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


- [ ] Item: PEDTC [IT.PE.PEDTC] pages = 10
  Annotation = PEDTC - similarity = 100%
- [ ] Item: QSTEST [IT.QS.QSTEST] pages = (13,14,15,17)
  Annotation = QSTEST - similarity = 100%
  CRF - pages: 17
- [ ] Item: QSCAT [IT.QS.QSCAT] pages = (13,14,15,17)
  Annotation = QSCAT - similarity = 100%

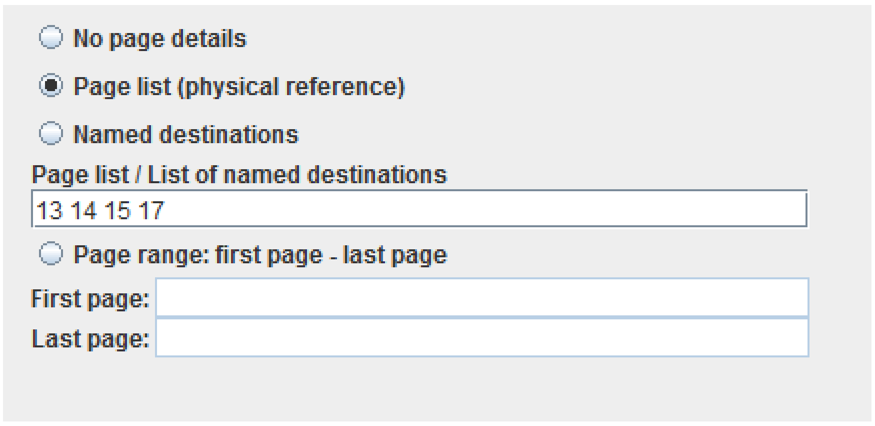
For QSTEST, an 100% matching annotation was found in the annotated CRF on pages 13, 14, 15 and 17, but in the define.xml, only page number 17 was assigned.

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


- [ ] Item: VISIT [IT.PE.VISIT] pages = (4,5)
  Annotation = VISIT - similarity = 100%
- [ ] Item: PEDTC [IT.PE.PEDTC] pages = 10
  Annotation = PEDTC - similarity = 100%
- [✓] Item: QSTEST [IT.QS.QSTEST] pages = (13,14,15,17)
  Annotation = QSTEST - similarity = 100%
- [✓] Item: QSCAT [IT.QS.QSCAT] pages = (13,14,15,17)
  Annotation = QSCAT - similarity = 100%
- [✓] Item: QSORRES [IT.QS.QSORRES] pages = (13,14,15,17)
  Annotation = QSORRES - similarity = 100%
- [ ] Item: VISIT [IT.QS.VISIT] pages = (4,5)
  Annotation = VISIT - similarity = 100%

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

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


○ No page details
● Page list (physical reference)
○ Named destinations

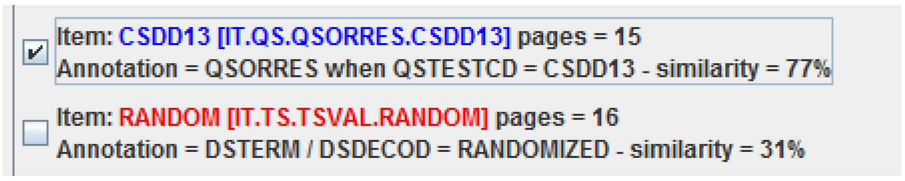
Page list / List of named destinations: 13 14 15 17

○ Page range: first page - last page

First page:
Last page:

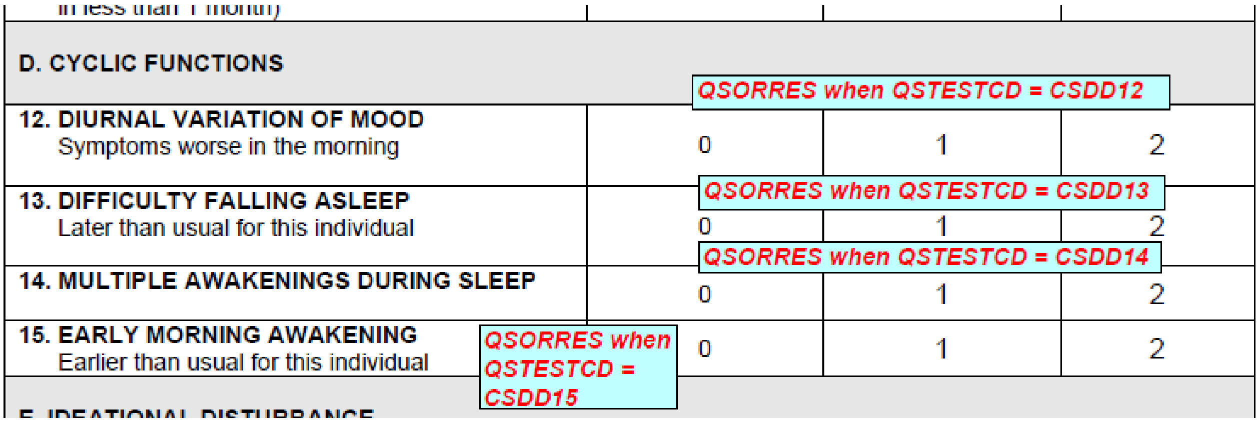
with these page numbers extracted from the annoted CRF.

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


[✓] Item: CSDD13 [IT.QS.QSORRES.CSDD13] pages = 15
Annotation = QSORRES when QSTESTCD = CSDD13 - similarity = 77%

[ ] Item: RANDOM [IT.TS.TSVAL.RANDOM] pages = 16
Annotation = DSTERM / DSDECOD = RANDOMIZED - similarity = 31%

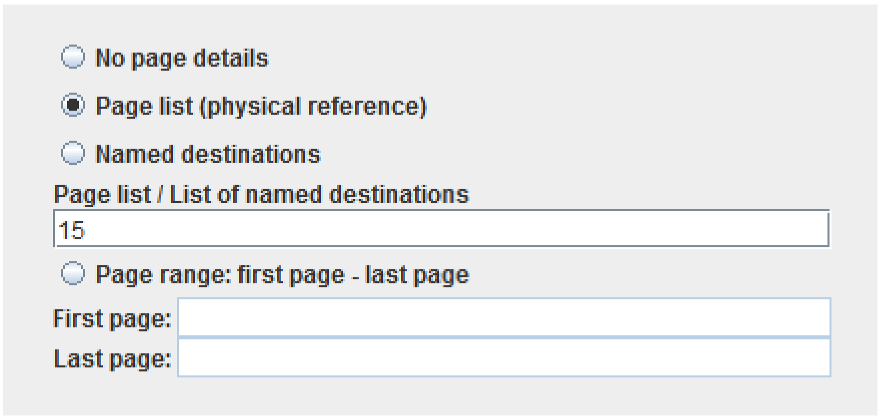
A "match" has been found for the CDSS13 "ValueList" variable with a high "similarity", for which the PDF annotation is "QSORRES when QSTESTCD = CSDD13".


in less than 1 month)

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

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

The result upon inspection is:


( ) No page details
(●) Page list (physical reference)
( ) Named destinations

Page list / List of named destinations: 15

( ) Page range: first page - last page

First page:

Last page:

<u>Conclusion:</u> this new feature in the ODM-Define-XML Designer allows you to extract page numbers from the annotated CRF semi-automatically, in such a way that the user still has full control of which page numbers are copied to the define.xml.
This feature will save you many hours of tedious and error prone work, for large studies maybe even days.