SDTM-ETL^{TM}

New features in version 3.2

Version 3.2 is available since spring 2017

Author: Jozef Aerts – XML4Pharma
August 2017

Table of Contents

Introduction

SDTM-ETL^{TM} 3.2 is another "big step forward" in the development of software for the mapping between operational clinical data and the CDISC SDTM and Define-XML standard.

Although SDTM itself is under pressure, because being outdated (still using 2-dimensional tables and the "medieval" SAS-XPT format), and because more and more "derived" variables are added (see here and here), making it unnecessarily complex and error-prone, we will still need to live some time with it, also as the tools of the regulatory authorities are not very modern.

To cope with this (unnecessary) complexity, we are currently adding a good number of new features to SDTM-ETL^{TM}, making it even more user-friendly than it was, and at the same time adding new support for the new standards that are currently been developed and will soon be released by CDISC: SDTM 1.5/SDTM-IG 3.3 and Define-XML 2.1.

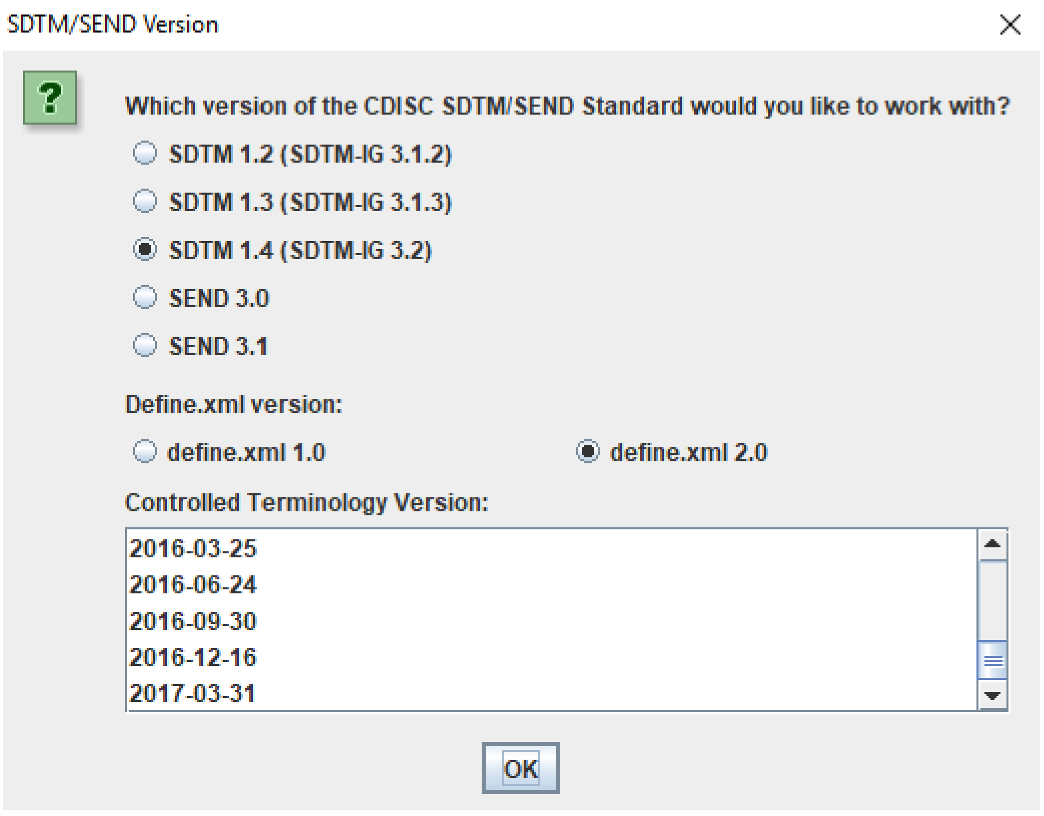
We are very proud to be able to present SDTM-ETL v.3.2.
We are convinced that this is the best tool on the market for mapping operational clinical data to submission data, with the lowest total cost of ownership (TCO).

Support for SEND-IG v.3.1

SDTM-ETL v.3.2 now fully supports SEND-IG v.3.1 which is based on the new SDTM v.1.5. No Implementation Guide for SDTM (SDTM-IG) has been published yet for this SDTM version. We have already make provisions for it and will implement it as soon as soon as CDISC publishes the final version.

For SEND-IG v.3.1 everything however has been implemented, including the define.xml template, and all the rules (as far as they can be implemented in software) from the SEND-IG v.3.1.

When the user uses the menu "File – Create define.xml", a new choice "SEND 3.1" is also displayed:



Which version of the CDISC SDTM/SEND Standard would you like to work with?

- SDTM 1.2 (SDTM-IG 3.1.2)
- SDTM 1.3 (SDTM-IG 3.1.3)
- [✓ SDTM 1.4 (SDTM-IG 3.2)]
- SEND 3.0
- SEND 3.1

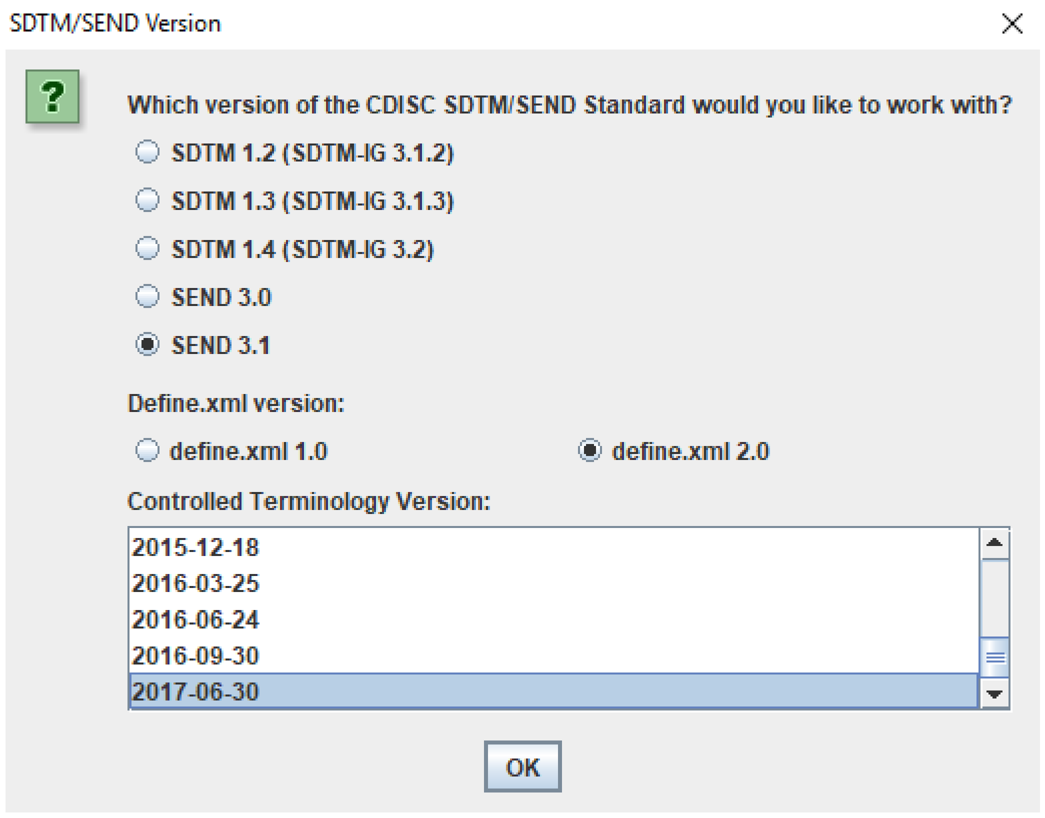
Define.xml version:

define.xml 1.0 [✓ define.xml 2.0]

Controlled Terminology Version:

2016-03-25
2016-06-24
2016-09-30
2016-12-16
2017-03-31

When the user chooses for "SEND 3.1", also the latest SEND controlled terminology (2017-06-30) can be selected:


SDTM/SEND Version

Which version of the CDISC SDTM/SEND Standard would you like to work with?

- ○ SDTM 1.2 (SDTM-IG 3.1.2)
- ○ SDTM 1.3 (SDTM-IG 3.1.3)
- ○ SDTM 1.4 (SDTM-IG 3.2)
- ○ SEND 3.0
- ● SEND 3.1

Define.xml version:

○ define.xml 1.0 ● define.xml 2.0

Controlled Terminology Version:

- 2015-12-18
- 2016-03-25
- 2016-06-24
- 2016-09-30
- [✓ 2017-06-30]

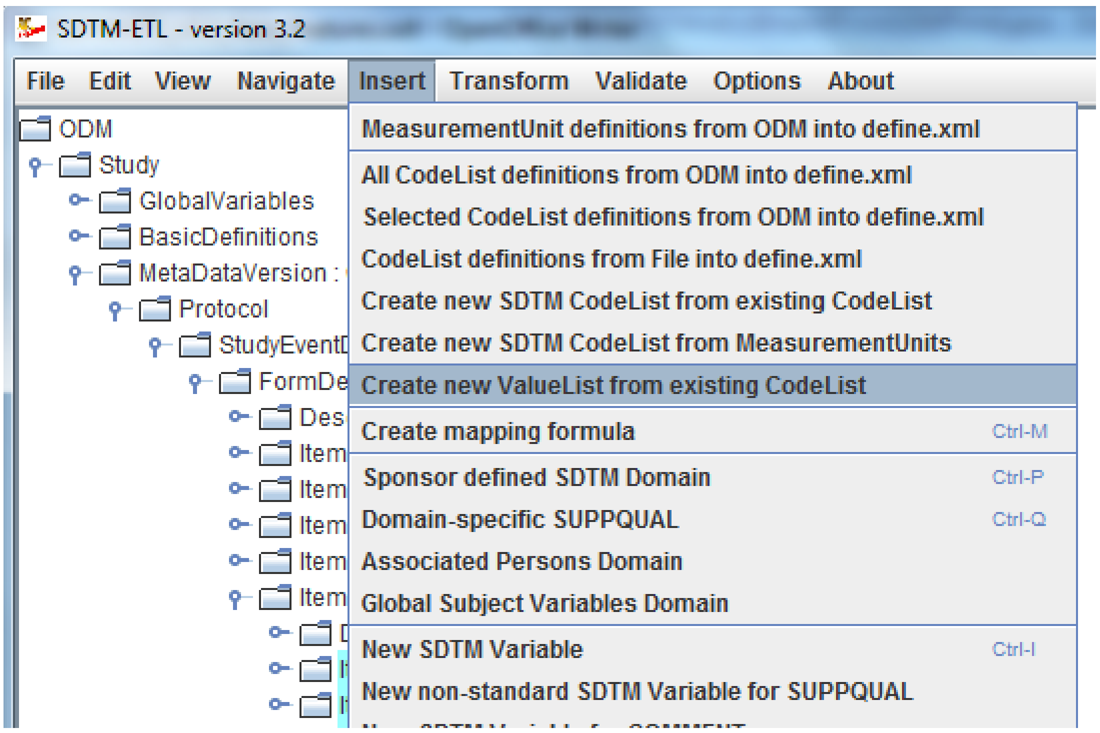
After clicking “OK”, the define.xml template for SEND 3.1, as well as the SEND controlled terminology is loaded.

Wizards for easier generation and editing of ValueLists

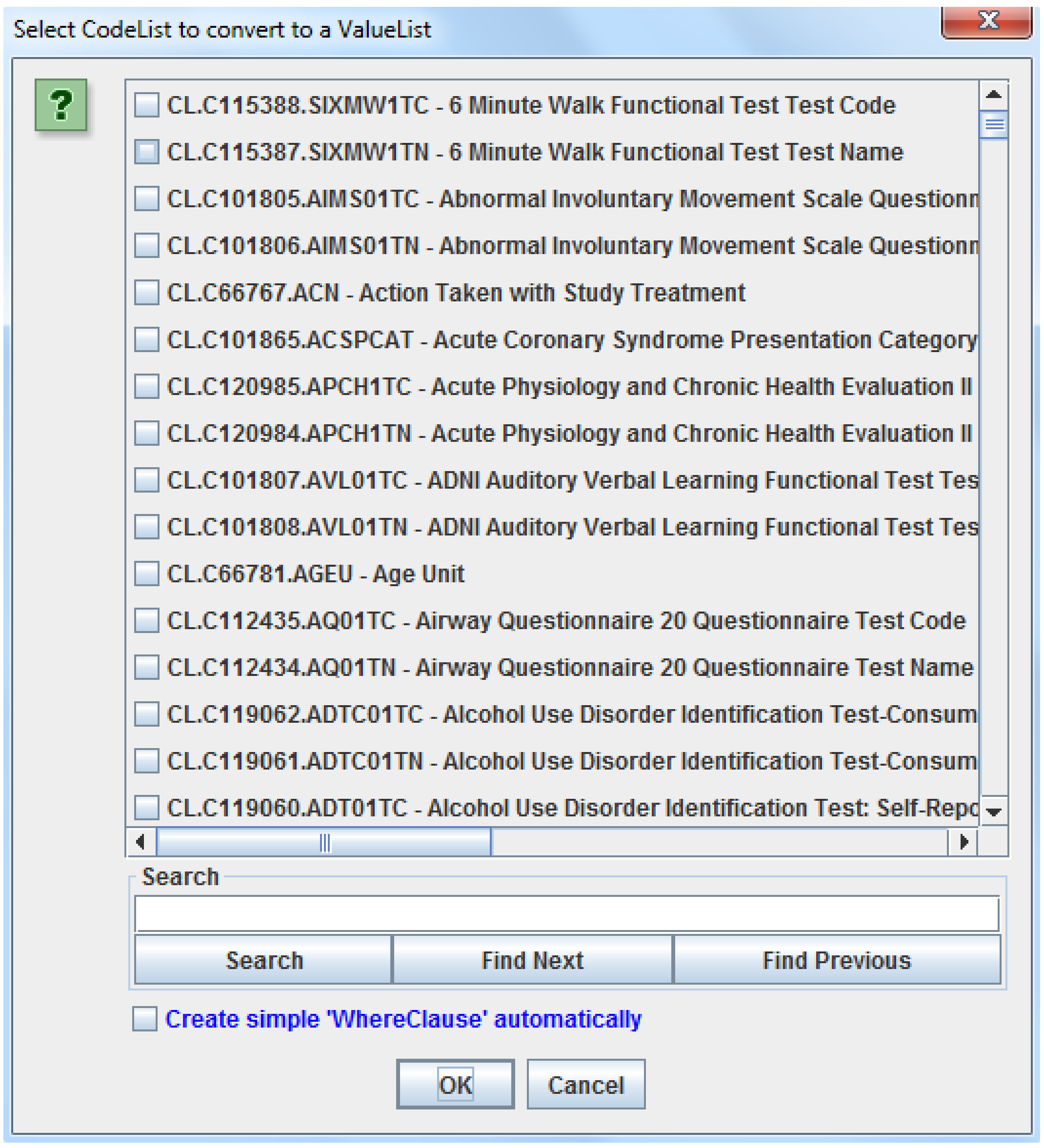
Generating and maintaining valuelists can be a lot of work. Essentially, one will need to generate a value-level variable for each test (or group of them) one has in each of the "Findings" domains. For example, if one has 50 different lab tests, each with their own metadata (datatype, max.length, number of figures after the decimal point, units used, or codelist for e.g. ordinal results), one may end with 50 different value-level variables, one for each test.

That looks like a lot of work, but we will show that this can be automated for a good part, without the user loosing control and oversight.

Usually, a ValueList is created started from a codelist. In SDTM-ETL^{TM} this is done using the menu "Insert - Create ValueList from CodeList":


SDTM-ETL - version 3.2
File Edit View Navigate Insert Transform Validate Options About
ODM
Study
GlobalVariables
BasicDefinitions
MetaDataVersion :
Protocol
StudyEvent
FormDe
MeasurementUnit definitions from ODM into define.xml
All CodeList definitions from ODM into define.xml
Selected CodeList definitions from ODM into define.xml
CodeList definitions from File into define.xml
Create new SDTM CodeList from existing CodeList
Create new SDTM CodeList from MeasurementUnits
Create new ValueList from existing CodeList
Create mapping formula Ctrl-M
Sponsor defined SDTM Domain Ctrl-P
Domain-specific SUPPQUAL Ctrl-Q
Associated Persons Domain
Global Subject Variables Domain
New SDTM Variable Ctrl-I
New non-standard SDTM Variable for SUPPQUAL

The user can then select one of the codelists that was loaded with the SDTM template. Usually (but not always) this will be CDISC CodeList:

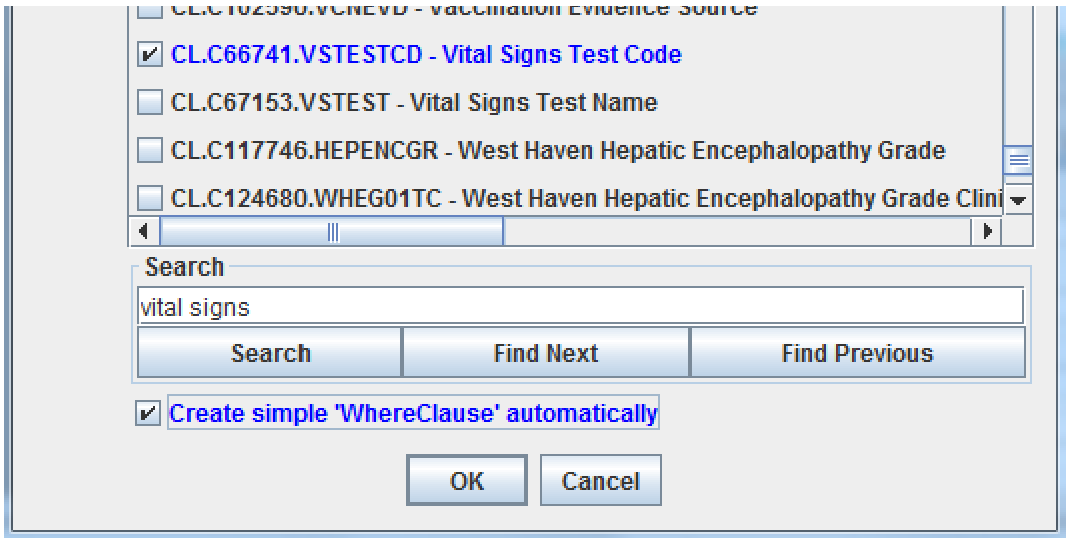

Select CodeList to convert to a ValueList

- [ ] 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
- [ ] CL.C112435.AQ01TC - Airway Questionnaire 20 Questionnaire Test Code
- [ ] CL.C112434.AQ01TN - Airway Questionnaire 20 Questionnaire Test Name
- [ ] CL.C119062.ADTC01TC - Alcohol Use Disorder Identification Test-Consum
- [ ] CL.C119061.ADTC01TN - Alcohol Use Disorder Identification Test-Consum
- [ ] CL.C119060.ADT01TC - Alcohol Use Disorder Identification Test: Self-Repc

Search

- [ ] Create simple 'WhereClause' automatically

The user can search for a specific codelist using the "Search" textfield and button. For example, for the "Vital Signs codes" codelist:


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

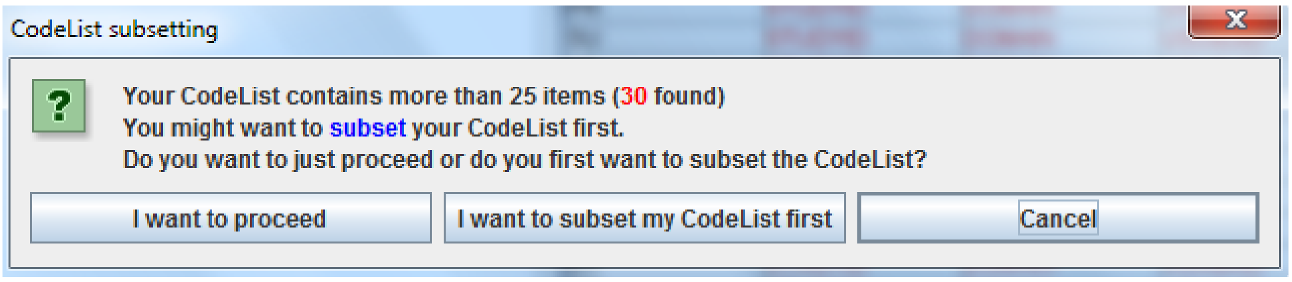
CL.C124680.WHEG01TC - West Haven Hepatic Encephalopathy Grade Clini

Search: vital signs

[✓] Create simple 'WhereClause' automatically

The checkbox "Create simple 'WhereClause' automatically" will allow to generate the "WhereClauses" for the valuelist items automatically. This is extremely useful, as in over 90% of the cases, the "WhereClause" will be very simple. For example for an item "PULSE", the "WhereClause" will typically be "WHERE VSTESTCD = DIABP". When this checkbox is checked, these simple "WhereClauses" will be generated automatically. This will often save an enormous amount of time.

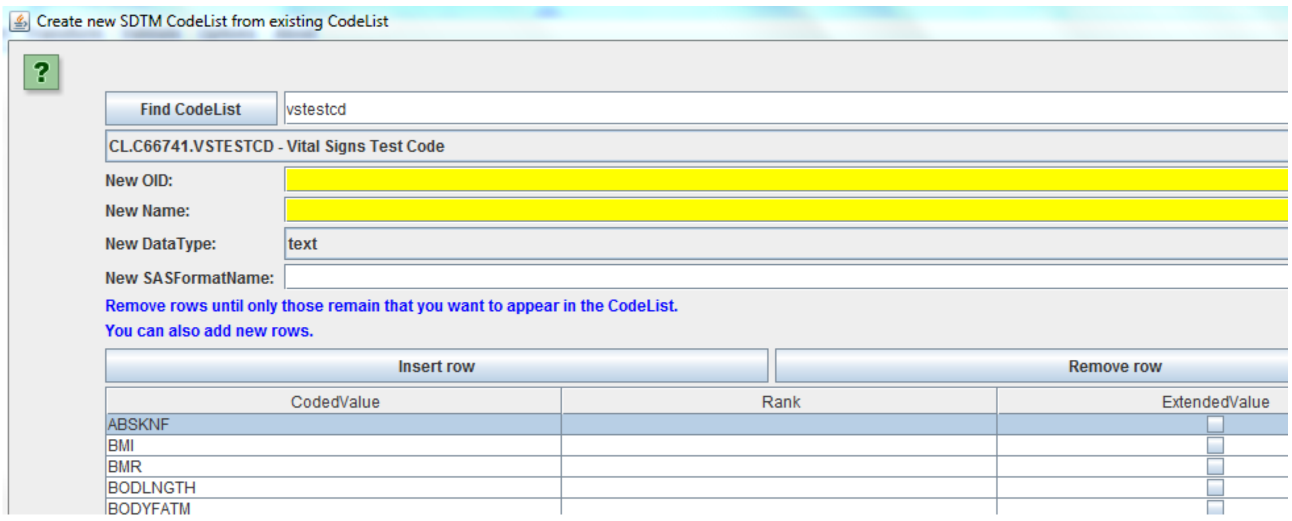
In case the codelist contains many items, it will take some time to generate the valuelist. However, you will very often want to subset a codelist first. For example, it is very unusual that you will have all 30 vital sign tests (from the CDISC CodeList) in your study. For the lab test codes (LBTESTCD) there are over 1400 test codes[1], so you will definitely want to subset that codelist first, only containing the codes for the test that were actually planned or used in the study[2].
When the system detects that you have more than 25 codes in your codelist, it will display the following dialog:


CodeList subsetting

Your CodeList contains more than 25 items (30 found)
You might want to subset your CodeList first.
Do you want to just proceed or do you first want to subset the CodeList?

If we do subset our codelist first, using the menu "Insert - Create new SDTM CodeList from existing CodeList", we get the following dialog:

1 It is not understandable why CDISC has not switched yet to the LOINC system for lab tests. LOINC is used for lab coding in almost every hospital in the world.
2 Remind that you also need to include all tests that were planned, even when some of them were never executed, or for which no data was ever collected.


Create new SDTM CodeList from existing CodeList

Find CodeList: vstestcd

CL.C66741.VSTESTCD - Vital Signs Test Code

New OID:

New Name:

New DataType: text

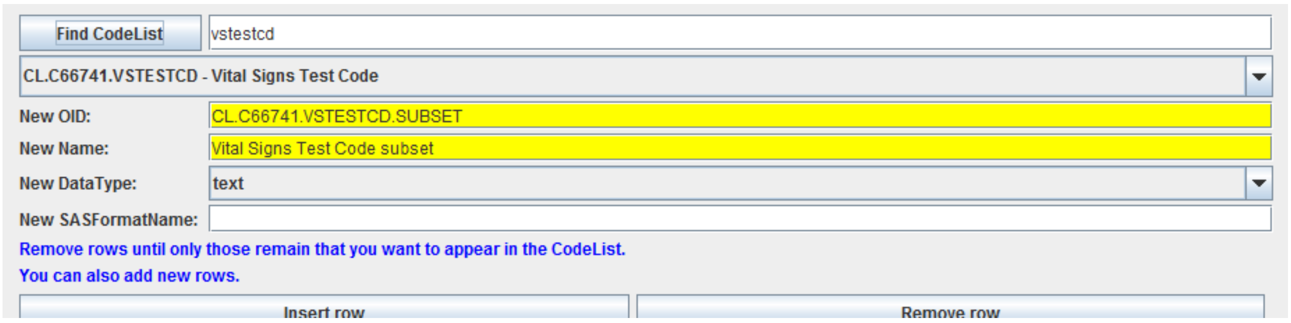
New SASFormatName:

Remove rows until only those remain that you want to appear in the CodeList.
You can also add new rows.

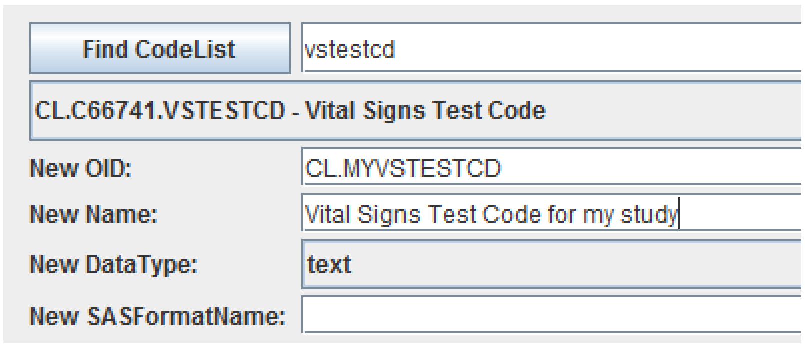
CodedValue	Rank	ExtendedValue
ABSKNF		☐
BMI		☐
BMR		☐
BODLNGTH		☐
BODYFATM		☐

also here, we can use the "Find" button to quickly find the desired one.

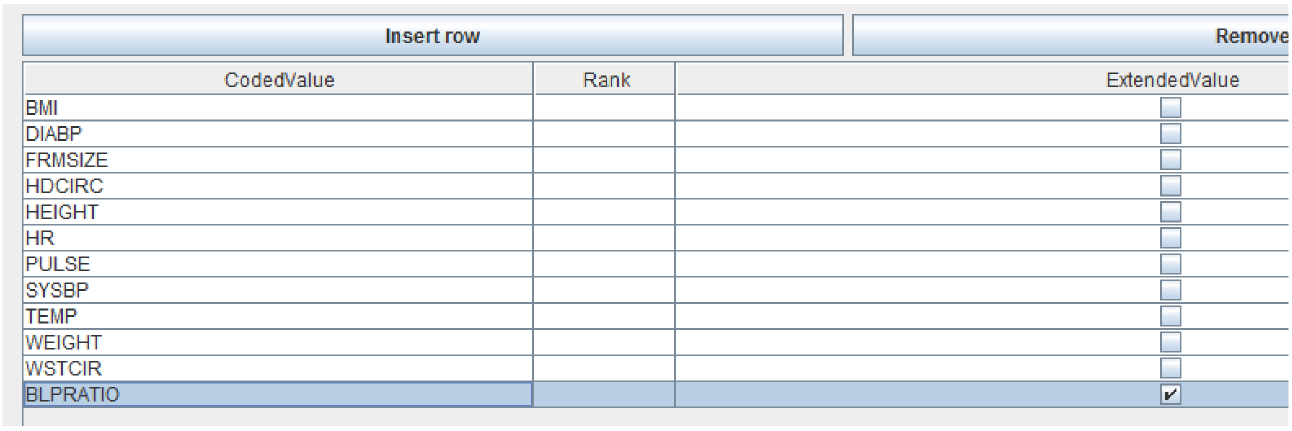
Once we made a selection for which codelist to subset, the system will make a proposal for a new OID and Name of the subset codelist, which you can of course always change:


Find CodeList
vstestcd
CL.C66741.VSTESTCD - Vital Signs Test Code
New OID:
CL.C66741.VSTESTCD.SUBSET
New Name:
Vital Signs Test Code subset
New DataType:
text
New SASFormatName:
Remove rows until only those remain that you want to appear in the CodeList.
You can also add new rows.
Insert row
Remove row

For example, you might want to change the OID in "CL.MYVSTESTCD" and the "Name" in "Vital Signs Test Code for my study":



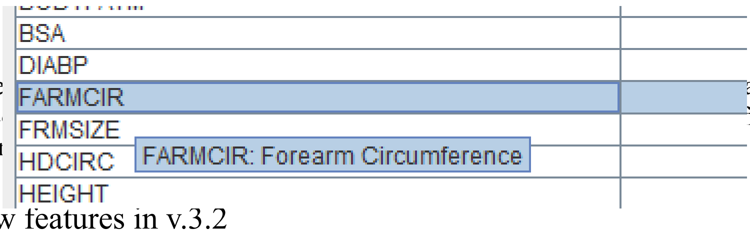
We can then start removing test codes (using the button "Remove row"), but also add new ones (using the button "Insert row"). For example, we remove those that were never planned in our study and we add a new test code "BLPRATIO"[3] ("blood pressure ratio"):



CodedValue	Rank	ExtendedValue
BMI		☐
DIABP		☐
FRMSIZE		☐
HDCIRC		☐
HEIGHT		☐
HR		☐
PULSE		☐
SYSBP		☐
TEMP		☐
WEIGHT		☐
WSTCIR		☐
BLPRATIO		☑

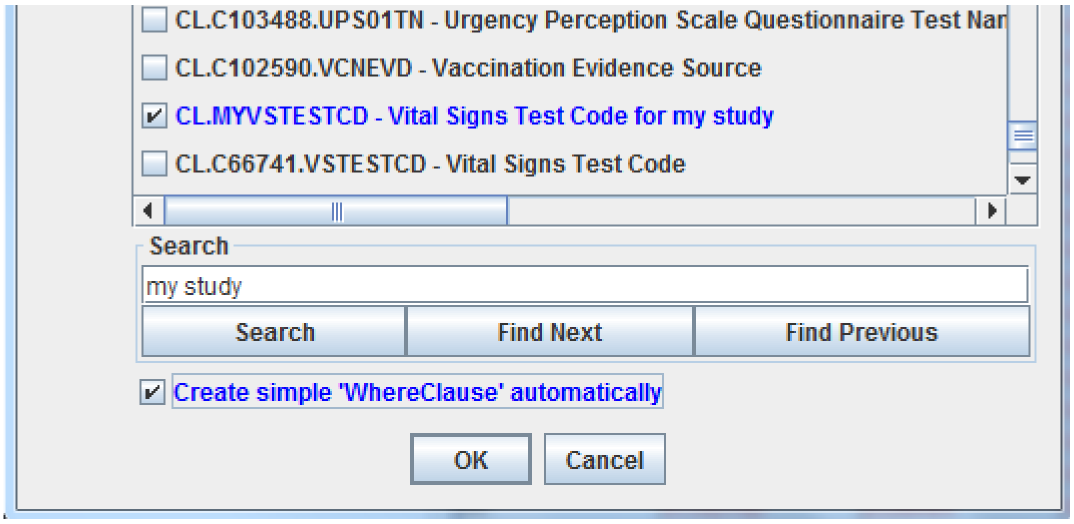
As one can see, when one adds a new term, the checkbox "ExtendedValue" is automatically selected. This information will then appear in the define.xml, meaning that the codelist has been extended with the value "BLPRATIO".

When removing items, and the codelist is a "test code" or "test name" codelist[4], you can always inspect the meaning (or code) by holding the mouse over an item, showing it as a tooltip. For example:

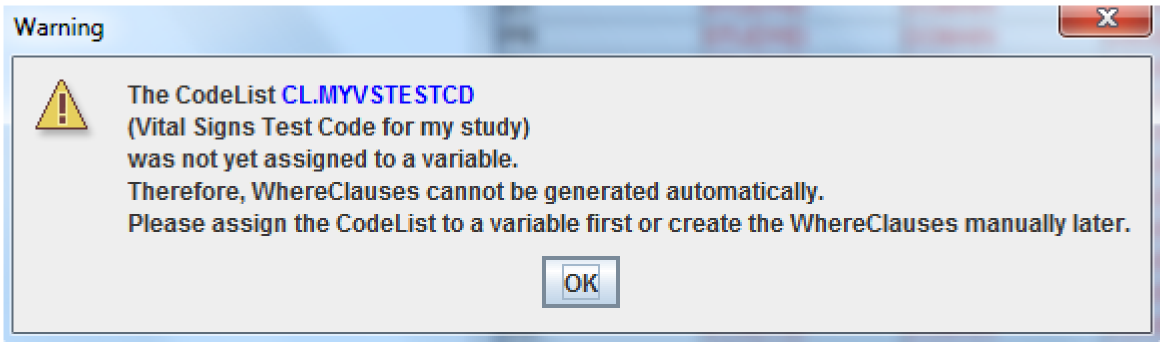


3 Unfortunately, there aracters.
4 Especially for test c ficult to immediately understand the mea

After you have subsetted the codelist, you can now try to make a valuelist of it again:

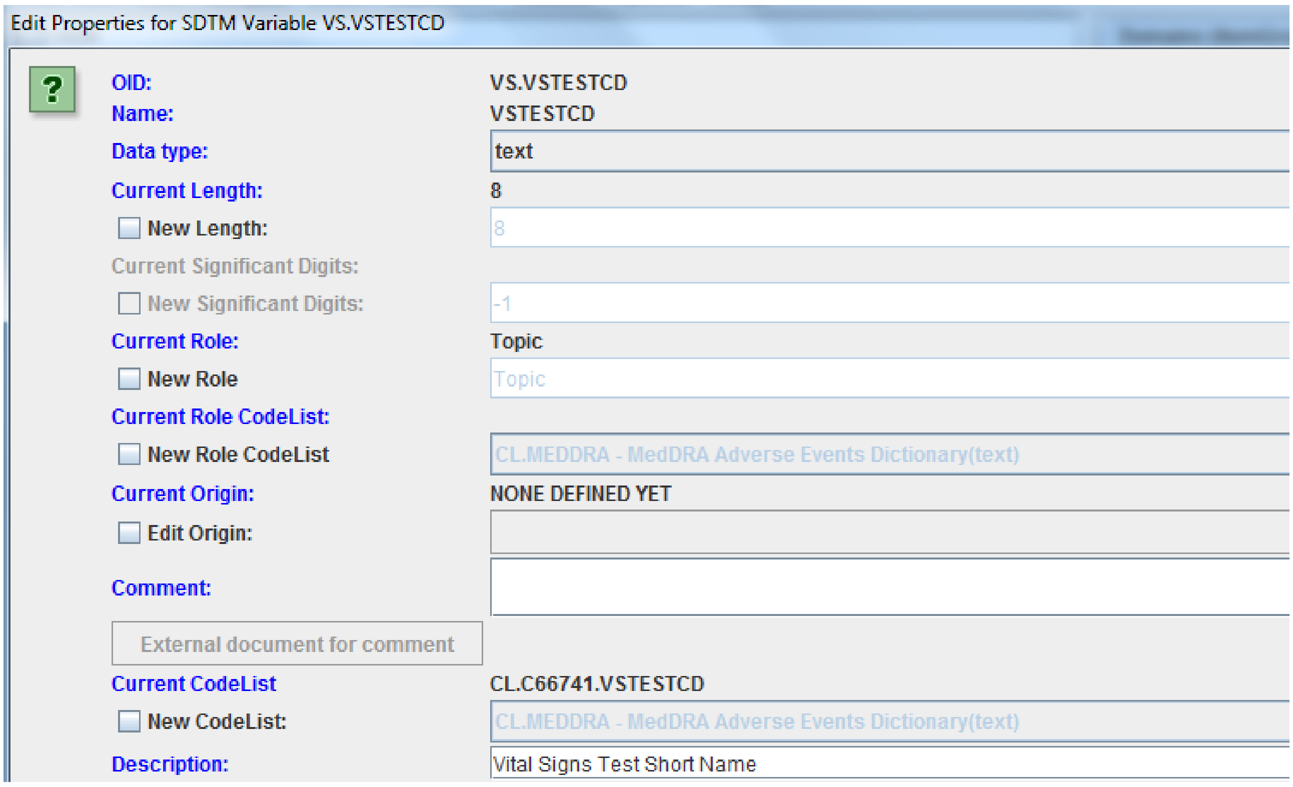

☐ CL.C103488.UPS01TN - Urgency Perception Scale Questionnaire Test Nar
☐ CL.C102590.VCNEVD - Vaccination Evidence Source
☑ CL.MYVSTESTCD - Vital Signs Test Code for my study
☐ CL.C66741.VSTESTCD - Vital Signs Test Code
Search
my study
Search
Find Next
Find Previous
☑ Create simple 'WhereClause' automatically
OK
Cancel

after clicking "OK", you will get the following message:


Warning

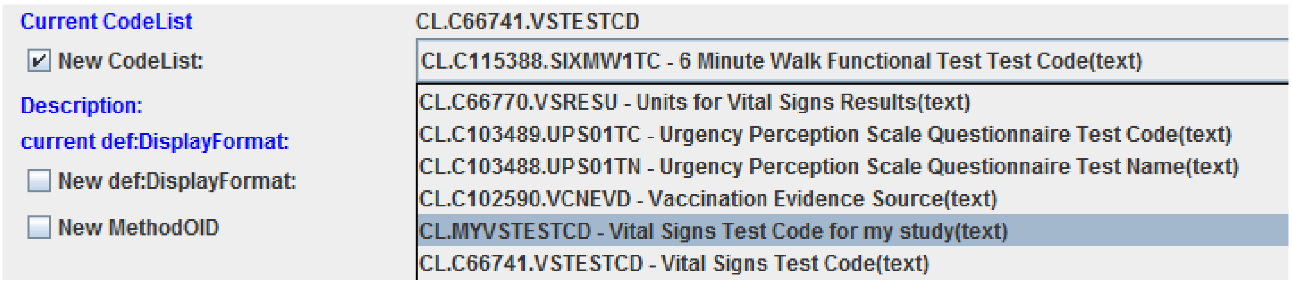
The CodeList CL.MYVSTESTCD
(Vital Signs Test Code for my study)
was not yet assigned to a variable.
Therefore, WhereClauses cannot be generated automatically.
Please assign the CodeList to a variable first or create the WhereClauses manually later.

The reason is that, when you automatically want to create the "WhereClause" the system needs to know to what SDTM variable the codelist has been assigned. As we still have the old codelist "CL.C66741.VSTESTCD" assigned to the SDTM variable VSTESTCD, it is now a good time to update that.
In order to do so, select VSTESTCD in the main window, and use the menu "Edit - SDTM Variable" (or use CTRL-E), leading to:

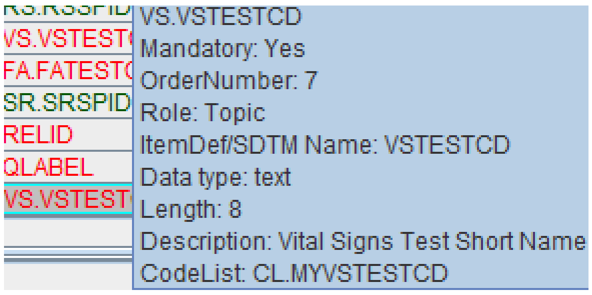

Edit Properties for SDTM Variable VS.VSTESTCD

OID:	VS.VSTESTCD
Name:	VSTESTCD
Data type:	text
Current Length:	8
☐ New Length:	8
Current Significant Digits:	
☐ New Significant Digits:	-1
Current Role:	Topic
☐ New Role	Topic
Current Role CodeList:	
☐ New Role CodeList	CL.MEDDRA - MedDRA Adverse Events Dictionary(text)
Current Origin:	NONE DEFINED YET
☐ Edit Origin:	
Comment:	
External document for comment	
Current CodeList	CL.C66741.VSTESTCD
☐ New CodeList:	CL.MEDDRA - MedDRA Adverse Events Dictionary(text)
Description:	Vital Signs Test Short Name

If the checkbox "New CodeList" is selected, you can reassign the codelist, using the dropdown:

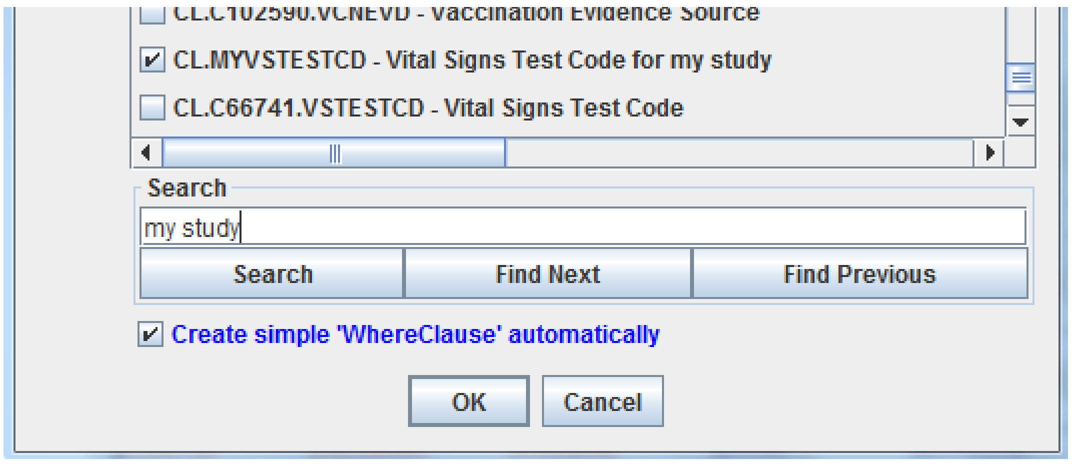

Current CodeList	CL.C66741.VSTESTCD
☑ New CodeList:	CL.C115388.SIXMW1TC - 6 Minute Walk Functional Test Test Code(text)
Description:	CL.C66770.VSRESU - Units for Vital Signs Results(text)
current def:DisplayFormat:	CL.C103489.UPS01TC - Urgency Perception Scale Questionnaire Test Code(text)
☐ New def:DisplayFormat:	CL.C103488.UPS01TN - Urgency Perception Scale Questionnaire Test Name(text)
	CL.C102590.VCNEVD - Vaccination Evidence Source(text)
☐ New MethodOID	CL.MYVSTESTCD - Vital Signs Test Code for my study(text)
	CL.C66741.VSTESTCD - Vital Signs Test Code(text)

after clicking "OK", the subset codelist has been assigned to variable VSTESTCD, as one can easily see from the tooltip on the variable:



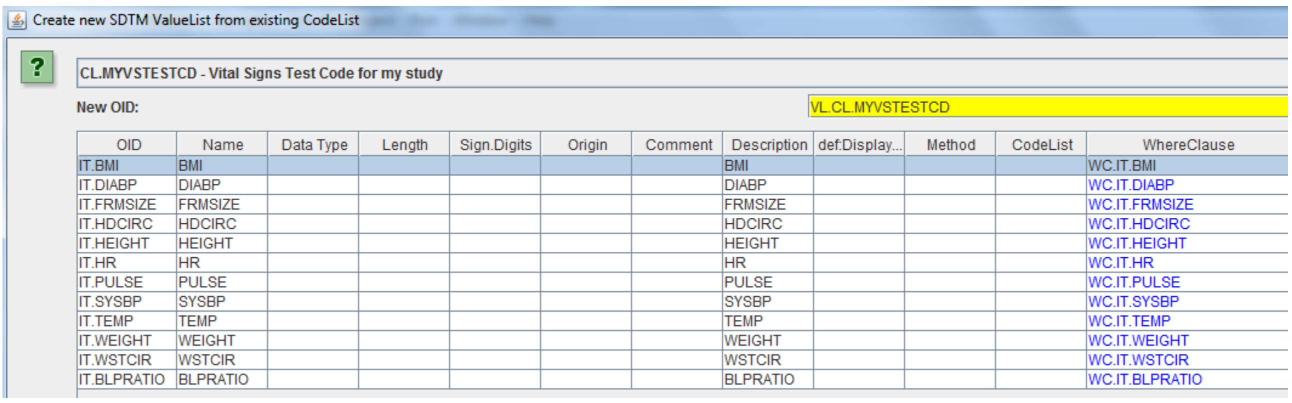
VS.VSTESTCD
Mandatory: Yes
OrderNumber: 7
Role: Topic
ItemDef/SDTM Name: VSTESTCD
Data type: text
Length: 8
Description: Vital Signs Test Short Name
CodeList: CL.MYVSTESTCD

We can then try to create the new ValueList again, this time for the codelist "CL.MYVSTESTCD":



CL.C102590.VCNEVD - Vaccination Evidence Source
☑ CL.MYVSTESTCD - Vital Signs Test Code for my study
CL.C66741.VSTESTCD - Vital Signs Test Code
Search
my study
Search
Find Next
Find Previous
☑ Create simple 'WhereClause' automatically
OK
Cancel

Clicking OK then leads to:



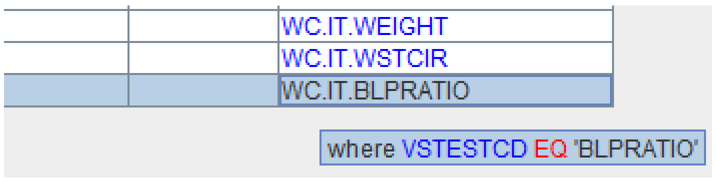
CL.MYVSTESTCD - Vital Signs Test Code for my study

New OID: VL.CL.MYVSTESTCD

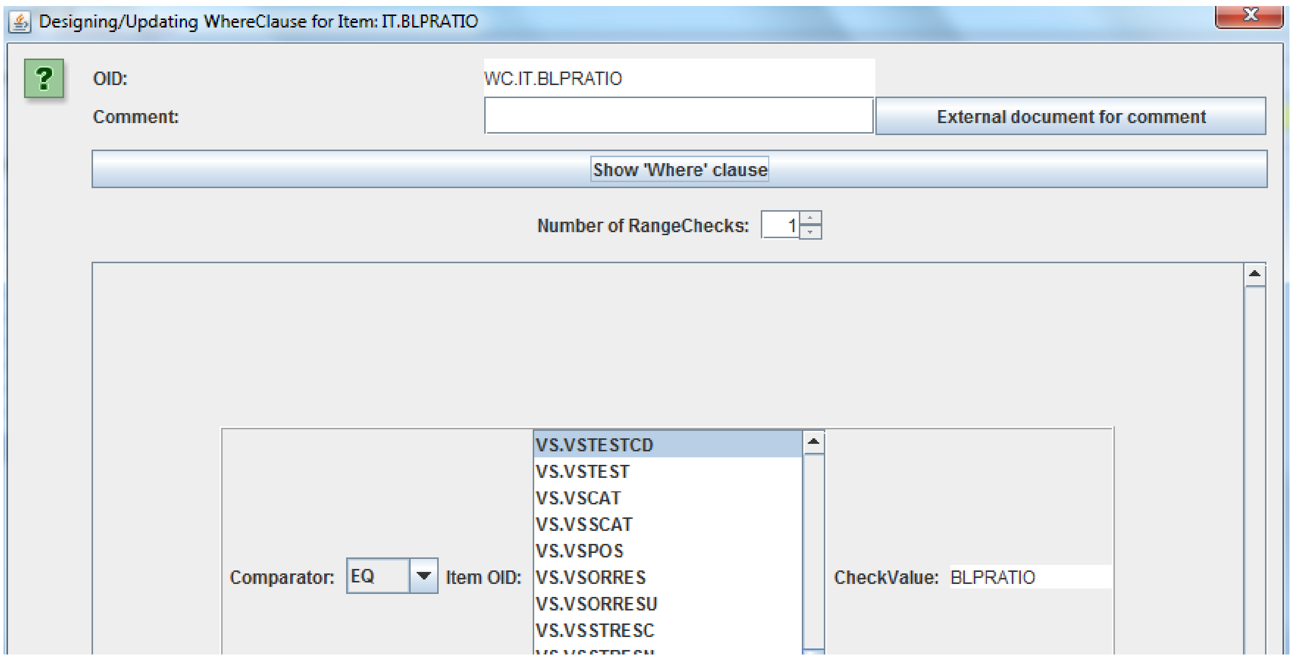
OID	Name	Data Type	Length	Sign.Digits	Origin	Comment	Description	def:Display...	Method	CodeList	WhereClause
IT.BMI	BMI						BMI				WC.IT.BMI
IT.DIABP	DIABP						DIABP				WC.IT.DIABP
IT.FRMSIZE	FRMSIZE						FRMSIZE				WC.IT.FRMSIZE
IT.HDCIRC	HDCIRC						HDCIRC				WC.IT.HDCIRC
IT.HEIGHT	HEIGHT						HEIGHT				WC.IT.HEIGHT
IT.HR	HR						HR				WC.IT.HR
IT.PULSE	PULSE						PULSE				WC.IT.PULSE
IT.SYSBP	SYSBP						SYSBP				WC.IT.SYSBP
IT.TEMP	TEMP						TEMP				WC.IT.TEMP
IT.WEIGHT	WEIGHT						WEIGHT				WC.IT.WEIGHT
IT.WSTCIR	WSTCIR						WSTCIR				WC.IT.WSTCIR
IT.BLPRATIO	BLPRATIO						BLPRATIO				WC.IT.BLPRATIO

For each test code, a value-level variable has been created, and one needs to provide the datatype, in most cases the maximal length and sometimes also the number of digits after the decimal point, and sometimes an associated codelist, like for "FRMSIZE" ("Frame Size": small, medium, large).

This looks like a lot of work! But fortunately, all the "WhereClauses" have already been generated automatically, as one easily sees when holding the mouse over one of them:



Or when clicking in the cell "WC.IT.BLPRATIO":


Designing/Updating WhereClause for Item: IT.BLPRATIO

OID: WC.IT.BLPRATIO

Comment:

Number of RangeChecks: 1

Comparator: EQ

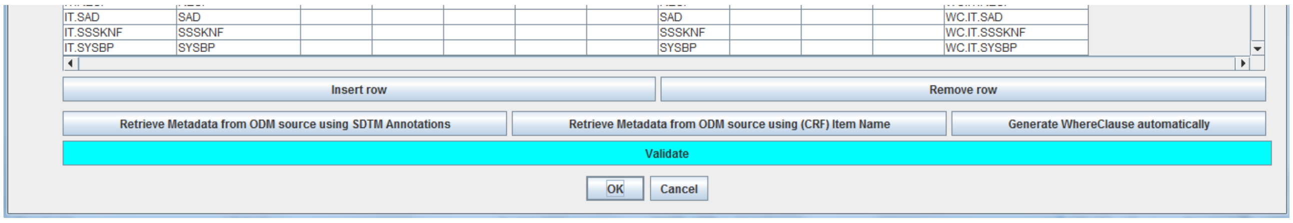
Item OID: VS.VSTESTCD

CheckValue: BLPRATIO

showing us the editor for the "WhereClause". We can still change anything here, for example to generate a single "WhereClause" for both diastolic blood pressure (DIABP) and systolic blood pressure (SYSBP). The "WhereClause" will then typically be like "WHERE VSTESTCD IN [DIABP,SYSBP].

In the ValueList table, we can now still remove rows in the table for which there was no test anyway in our study, or add new rows (tests). We can do this manually, but we can also do a lookup in the source ODM, to find out whether there was such a test.

In the lower part of the screen, we find 3 new buttons:



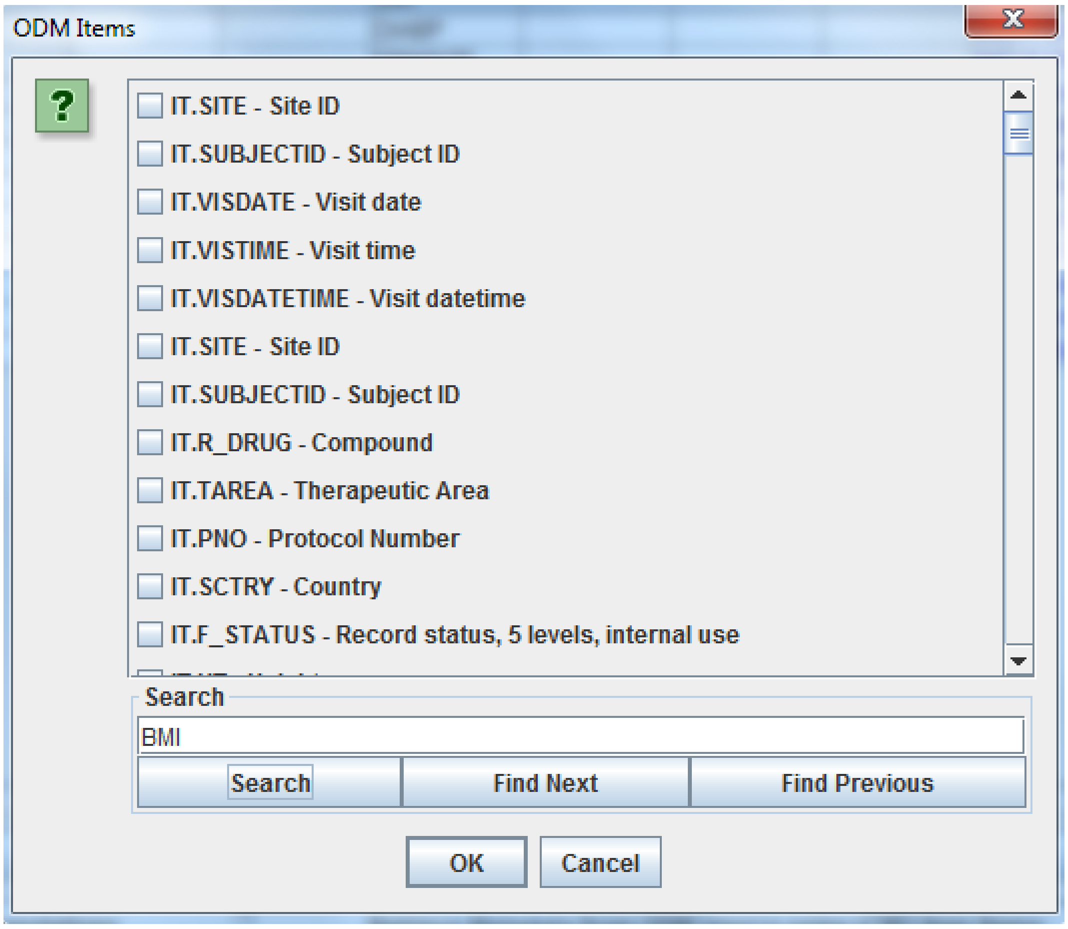
IT.SAD	SAD						SAD				WC.IT.SAD
IT.SSSKNF	SSSKNF						SSSKNF				WC.IT.SSSKNF
IT.SYSBP	SYSBP						SYSBP				WC.IT.SYSBP

Insert row
Remove row
Retrieve Metadata from ODM source using SDTM Annotations
Retrieve Metadata from ODM source using (CRF) Item Name
Generate WhereClause automatically
Validate
OK
Cancel

stating:

- Retrieve metadata from ODM source using SDTM annotations
- Retrieve metadata from ODM using (CRF) Item name
- Generate WhereClause automatically

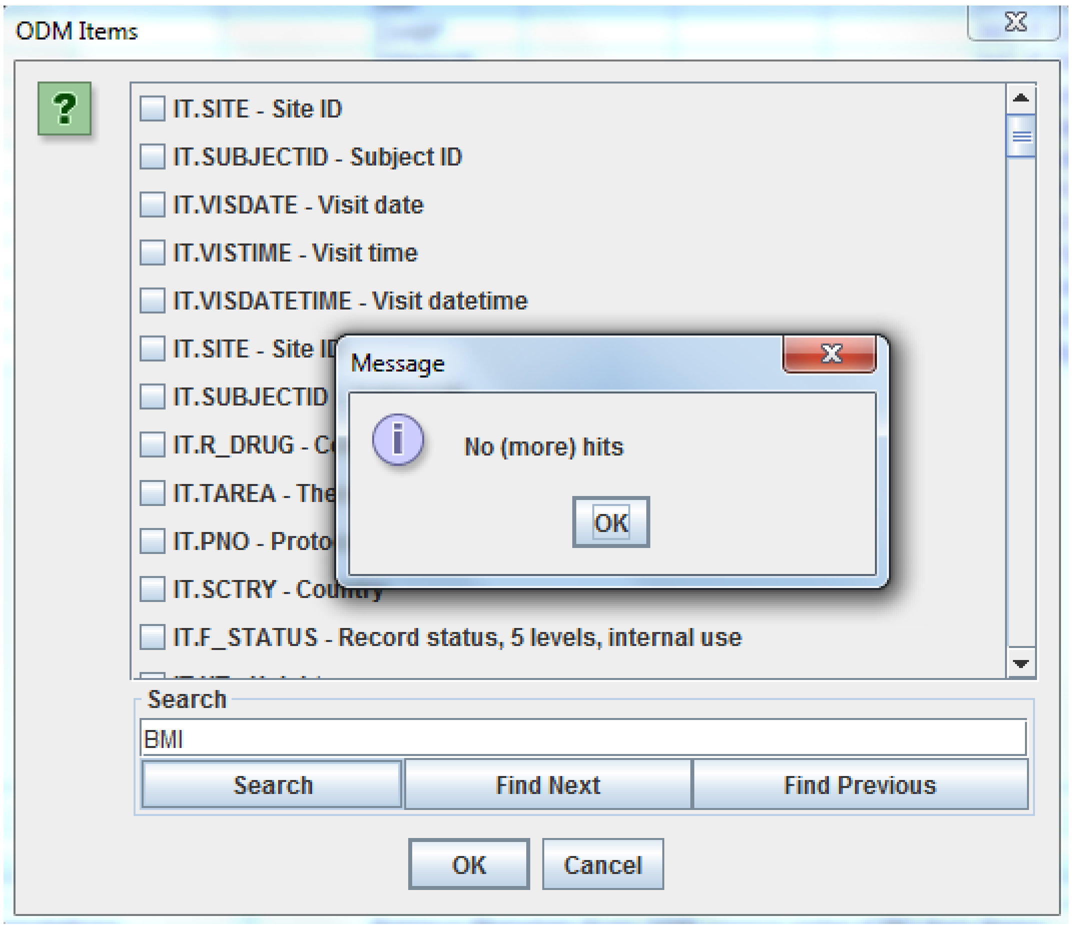
First of all, let us look which vital signs were really present in the source ODM (study design). We can easily find out using the button "Retrieve metadata from ODM using (CRF) Item name". For example, when we select the first row containing "BMI", and then click the button, we are presented with the list of items (questions/datapoints) from the ODM:



IT.SITE - Site ID
IT.SUBJECTID - Subject ID
IT.VISDATE - Visit date
IT.VISTIME - Visit time
IT.VISDATETIME - Visit datetime
IT.SITE - Site ID
IT.SUBJECTID - Subject ID
IT.R_DRUG - Compound
IT.TAREA - Therapeutic Area
IT.PNO - Protocol Number
IT.SCTRY - Country
IT.F_STATUS - Record status, 5 levels, internal use

Search: BMI

and using the "Search" button:


ODM Items

- IT.SITE - Site ID
- IT.SUBJECTID - Subject ID
- IT.VISDATE - Visit date
- IT.VISTIME - Visit time
- IT.VISDATETIME - Visit datetime
- IT.SITE - Site ID
- IT.SUBJECTID
- IT.R_DRUG - C
- IT.TAREA - The
- IT.PNO - Proto
- IT.SCTRY - Country
- IT.F_STATUS - Record status, 5 levels, internal use

Message

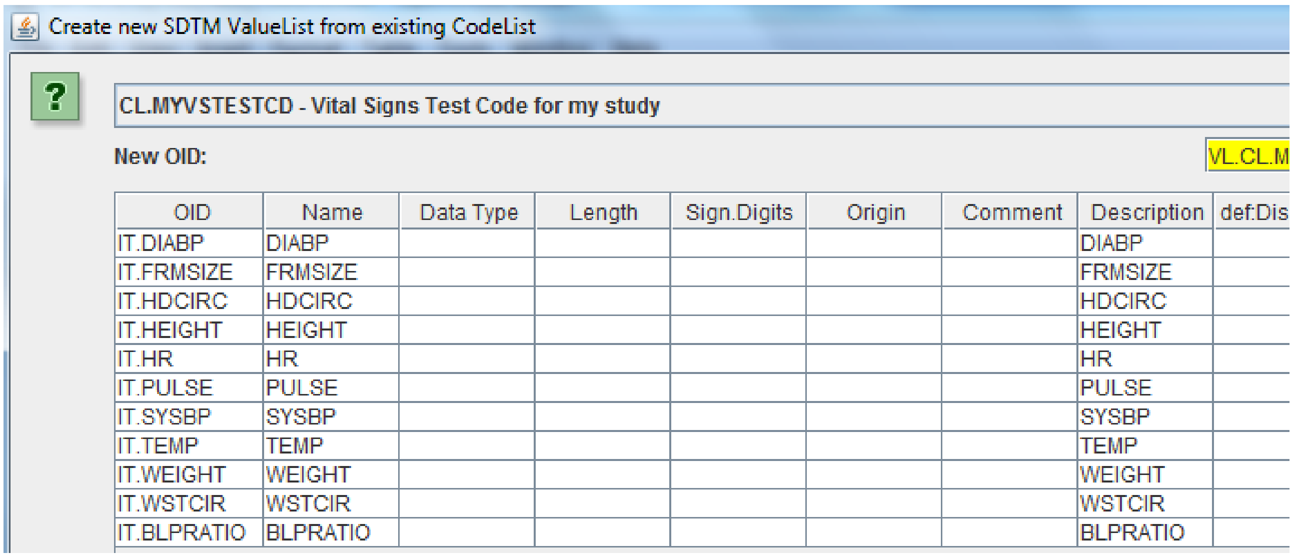
No (more) hits

OK

Search: BMI

So, the system did not find any item in which the wording "BMI" appears. We can of course also do a search on "body mass index" or similar, but will not find something like that either. So we can assume that this test was not in the study design (it might of course be that it was named completely differently, but then we have a semantic problem) and so can be removed from the valuelist.

So, after removing the first row, the valuelist reduces a little bit as:


Create new SDTM ValueList from existing CodeList

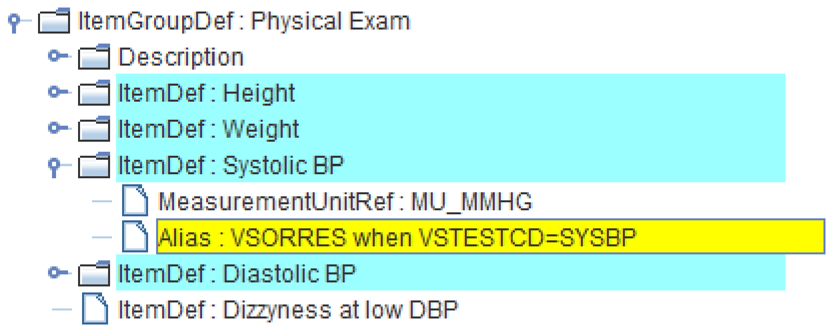
CL.MYVSTESTCD - Vital Signs Test Code for my study

New OID: VL.CL.M

OID	Name	Data Type	Length	Sign.Digits	Origin	Comment	Description	def:Dis
IT.DIABP	DIABP						DIABP	
IT.FRMSIZE	FRMSIZE						FRMSIZE	
IT.HDCIRC	HDCIRC						HDCIRC	
IT.HEIGHT	HEIGHT						HEIGHT	
IT.HR	HR						HR	
IT.PULSE	PULSE						PULSE	
IT.SYSBP	SYSBP						SYSBP	
IT.TEMP	TEMP						TEMP	
IT.WEIGHT	WEIGHT						WEIGHT	
IT.WSTCIR	WSTCIR						WSTCIR	
IT.BLPRATIO	BLPRATIO						BLPRATIO	

Finding out which tests were in the study design goes pretty fast in this way. We can then remove rows until we only have the vital signs that were really present in our study design.

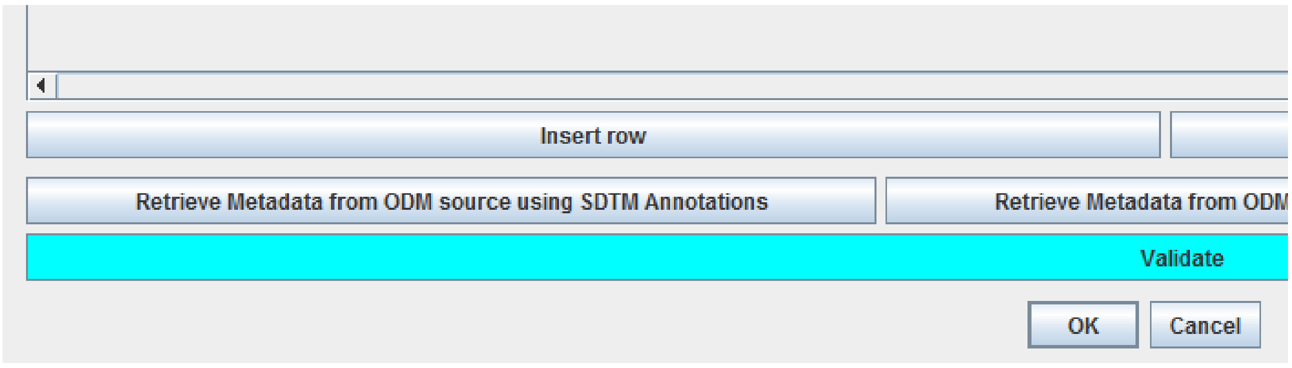
In our valuelist table, there is a second way to find out whether a test was in the study design or not. This method is based on the annotations that were added to the ODM study design, usually already at study design time[5]. These annotations use the "Alias" element in the ODM, usually with the "Context" being "SDTM". For example:



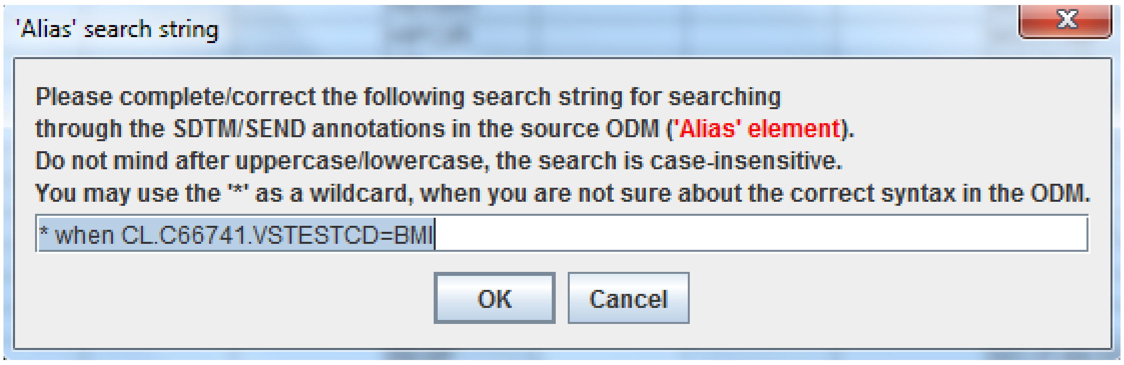
ItemGroupDef : Physical Exam
- Description
- ItemDef : Height
- ItemDef : Weight
- ItemDef : Systolic BP
  - MeasurementUnitRef : MU_MMHG
  - Alias : VSORRES when VSTESTCD=SYSBP
- ItemDef : Diastolic BP
- ItemDef : Dizzyness at low DBP

As the annotations (or better, their way of writing) is not standardized, some variations of it can be found, like "VSORRES where VSTESTCD = 'SYSBP'".

In our ValueList table editor, we can make advantage of this, by using the button "Retrieve Metadata from ODM Source using Annotations":


Insert row
Retrieve Metadata from ODM source using SDTM Annotations
Retrieve Metadata from ODM
Validate
OK
Cancel

For example, if we first select the row "WSTCIR" and then click the button, we get:



'Alias' search string

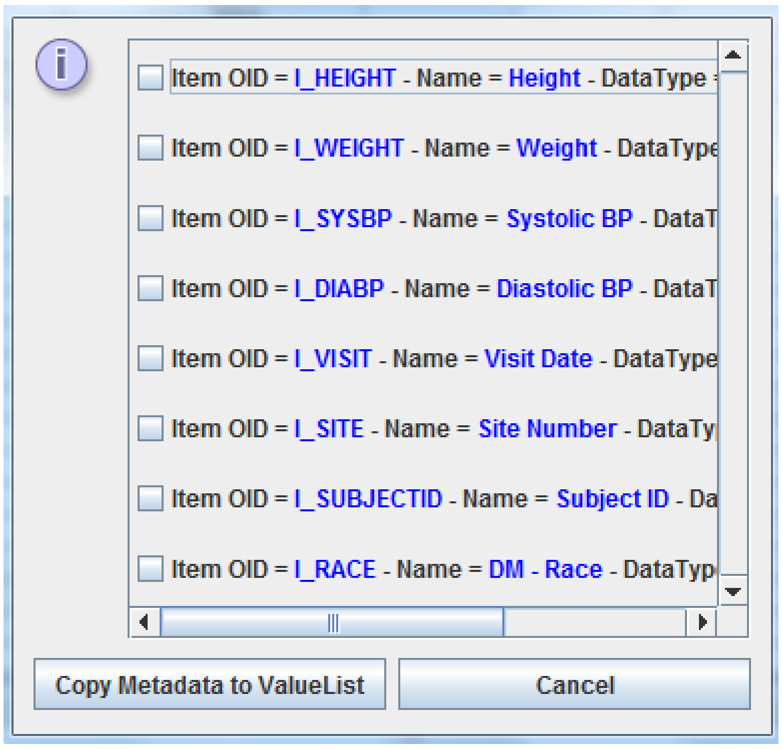
Please complete/correct the following search string for searching through the SDTM/SEND annotations in the source ODM ('Alias' element). Do not mind after uppercase/lowercase, the search is case-insensitive. You may use the '*' as a wildcard, when you are not sure about the correct syntax in the ODM.

* when CL.C66741.VSTESTCD=BMI

TODO: replace image by the one for WSTCIR

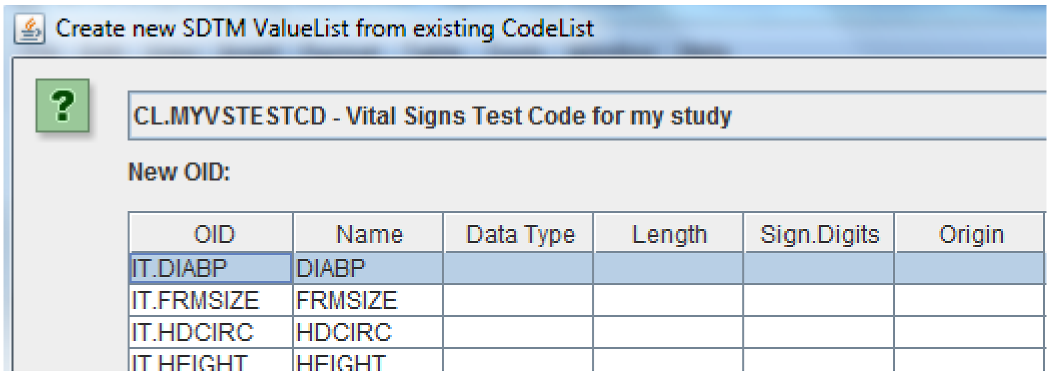
5 This is an extremely good practice, explained in our "ODM Wiki"

We can still edit the search expression, and also use a "wildcard" ("*") if we do not know part of the annotation. However, the system is very smart, it will look for sentence similarity in all the ODM-SDTM annotations, and rank them according to the similarity. For the above search term (without any editing) we get:


Item OID = I_HEIGHT - Name = Height - DataType
Item OID = I_WEIGHT - Name = Weight - DataType
Item OID = I_SYSBP - Name = Systolic BP - DataT
Item OID = I_DIABP - Name = Diastolic BP - DataT
Item OID = I_VISIT - Name = Visit Date - DataType
Item OID = I_SITE - Name = Site Number - DataTy
Item OID = I_SUBJECTID - Name = Subject ID - Da
Item OID = I_RACE - Name = DM - Race - DataTyp

indicating that none of the data points in the study design was annotated as being to be used later with VSTESTCD=WSTCIR. So we can remove that row too.

If we do the same however for the row with entry "DIABP":

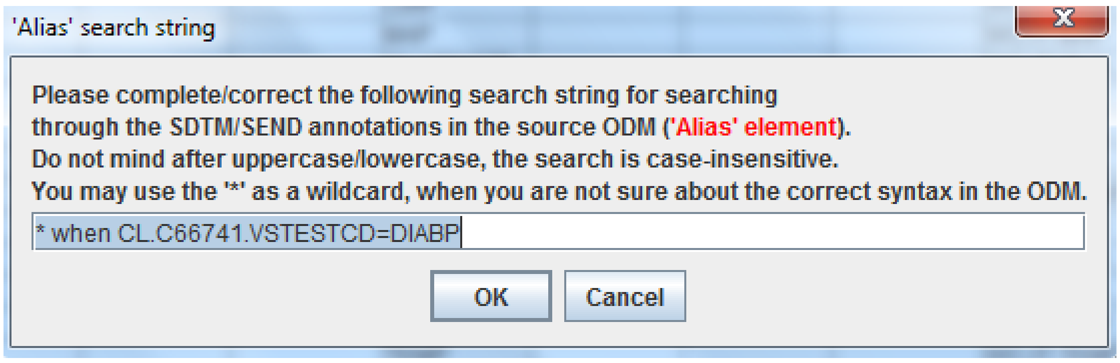

Create new SDTM ValueList from existing CodeList

CL.MYVSTESTCD - Vital Signs Test Code for my study

New OID:

OID	Name	Data Type	Length	Sign.Digits	Origin
IT.DIABP	DIABP				
IT.FRMSIZE	FRMSIZE				
IT.HDCIRC	HDCIRC				
IT.HEIGHT	HEIGHT				

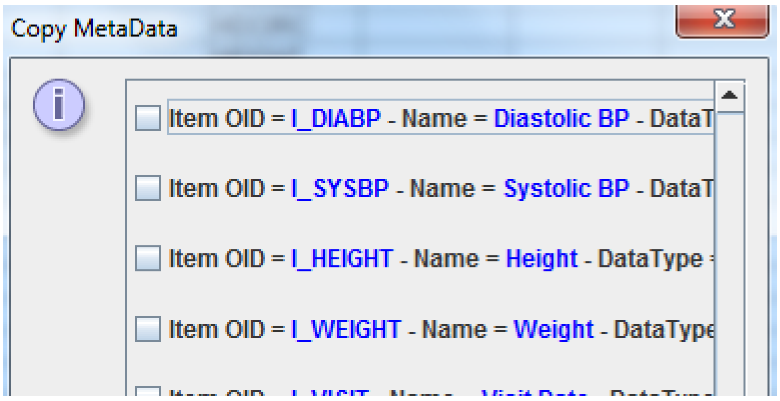
leading to:


'Alias' search string

Please complete/correct the following search string for searching through the SDTM/SEND annotations in the source ODM ('Alias' element). Do not mind after uppercase/lowercase, the search is case-insensitive. You may use the '*' as a wildcard, when you are not sure about the correct syntax in the ODM.

* when CL.C66741.VSTESTCD=DIABP

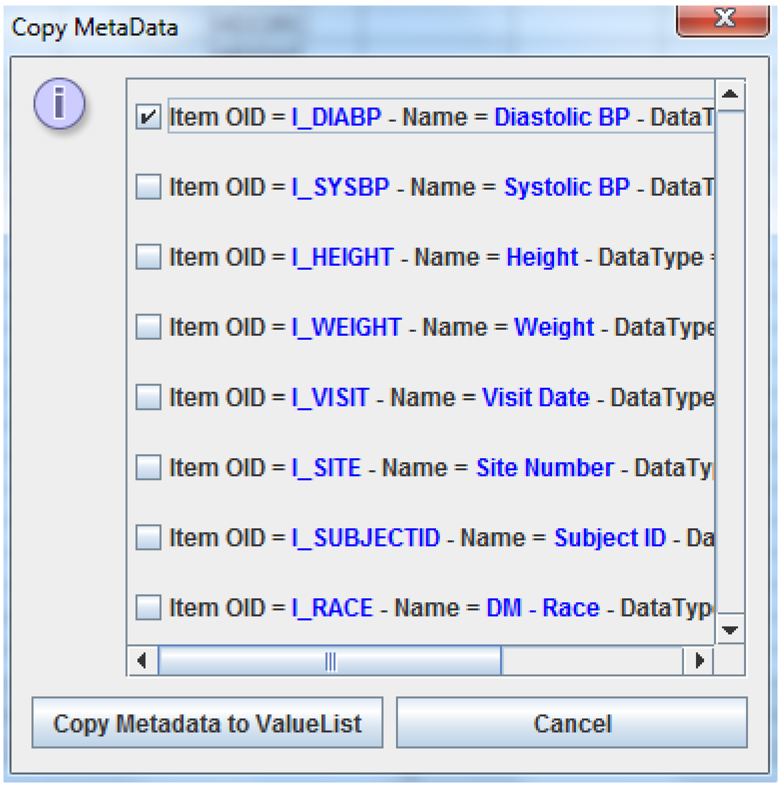
and then executing the search, leading to:


Copy MetaData

- [ ] Item OID = I_DIABP - Name = Diastolic BP - DataT
- [ ] Item OID = I_SYSBP - Name = Systolic BP - DataT
- [ ] Item OID = I_HEIGHT - Name = Height - DataType
- [ ] Item OID = I_WEIGHT - Name = Weight - DataType

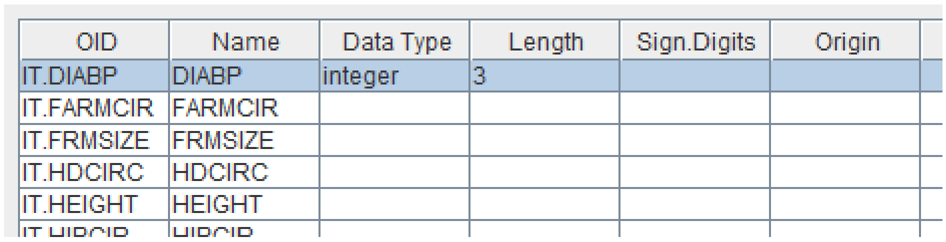
the items in the list being ordered by similarity with the search string, and showing that there is indeed an item in the study design that was annotated and that is a "diastolic blood pressure".

We can than copy its metadata from the ODM to the SDTM, by selecting the item and clicking the button "Copy Metadata to ValueList":


Copy MetaData

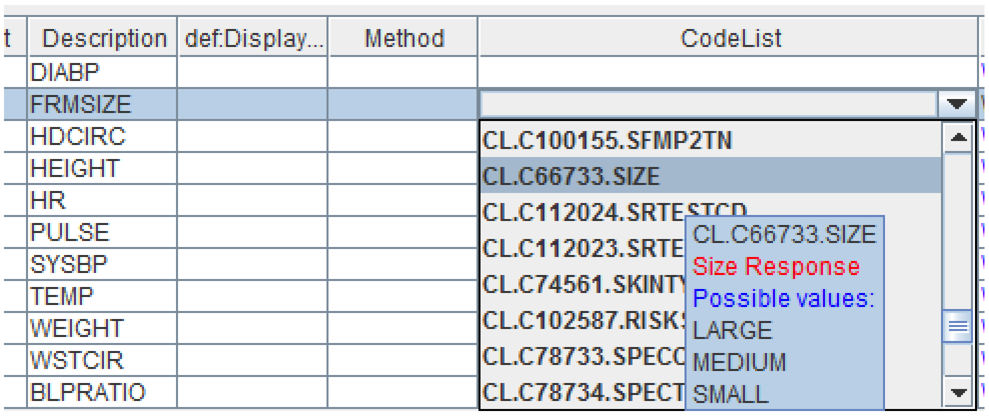
- [x] Item OID = I_DIABP - Name = Diastolic BP - DataT
- [ ] Item OID = I_SYSBP - Name = Systolic BP - DataT
- [ ] Item OID = I_HEIGHT - Name = Height - DataType
- [ ] Item OID = I_WEIGHT - Name = Weight - DataType
- [ ] Item OID = I_VISIT - Name = Visit Date - DataType
- [ ] Item OID = I_SITE - Name = Site Number - DataTy
- [ ] Item OID = I_SUBJECTID - Name = Subject ID - Da
- [ ] Item OID = I_RACE - Name = DM - Race - DataTyp

and immediately, we see in our valuelist table:



OID	Name	Data Type	Length	Sign.Digits	Origin
IT.DIABP	DIABP	integer	3		
IT.FARMCIR	FARMCIR				
IT.FRMSIZE	FRMSIZE				
IT.HDCIRC	HDCIRC				
IT.HEIGHT	HEIGHT				
IT.HIPCIR	HIPCIR				

that the datatype (in this case "integer") maximal length (in this case "3") have been copied.
If a codelist would be involved, also the codelist reference would be copied in the cell "CodeList".
If that codelist is not present in the loaded controlled terminology (in the underlying define.xml), the user is invited to copy the codelist from the ODM into the SDTM, or to choose another one that is already in the SDTM. The latter can easily be done by using the dropdown, e.g. for "FRMSIZE":



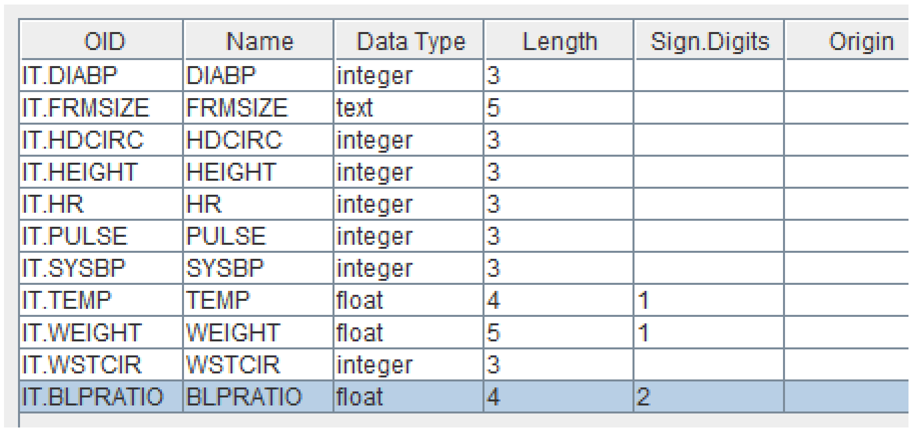
Description	def:Display...	Method	CodeList
DIABP			
FRMSIZE			
HDCIRC			
HEIGHT			
HR			
PULSE			
SYSBP			
TEMP			
WEIGHT			
WSTCIR			
BLPRATIO			

CL.C100155.SFMP2TN
CL.C66733.SIZE
CL.C112024.SRTESTCD
CL.C112023.SRTE
CL.C74561.SKINTY
CL.C102587.RISKS
CL.C78733.SPECC
CL.C78734.SPECT

CL.C66733.SIZE
Size Response
Possible values:
LARGE
MEDIUM
SMALL

With these two new features, one can easily and relatively quickly find out which tests were in the study design, for which value-level metadata need to provided, and copy them from the study design automatically[6]. The ones in the list that were not in the study design can then easily removed from the valuelist table.

Doing so in our case, quickly leads to :



OID	Name	Data Type	Length	Sign.Digits	Origin
IT.DIABP	DIABP	integer	3		
IT.FRMSIZE	FRMSIZE	text	5		
IT.HDCIRC	HDCIRC	integer	3		
IT.HEIGHT	HEIGHT	integer	3		
IT.HR	HR	integer	3		
IT.PULSE	PULSE	integer	3		
IT.SYSBP	SYSBP	integer	3		
IT.TEMP	TEMP	float	4	1	
IT.WEIGHT	WEIGHT	float	5	1	
IT.WSTCIR	WSTCIR	integer	3		
IT.BLPRATIO	BLPRATIO	float	4	2	

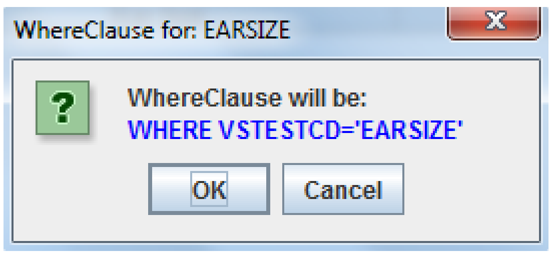
6 We decided not to make this a "bulk" or "blind" automatism, as that can easily lead to errors, like removing tests that were known under a (more or less slightly) different name or code in the study design. We consider it to be important that the user understands what he/she is doing, unlike in "black box" tools.

with datatype and maximal length being copied from the source ODM study design.

The third button under the table states "Generate WhereClause Automatically".

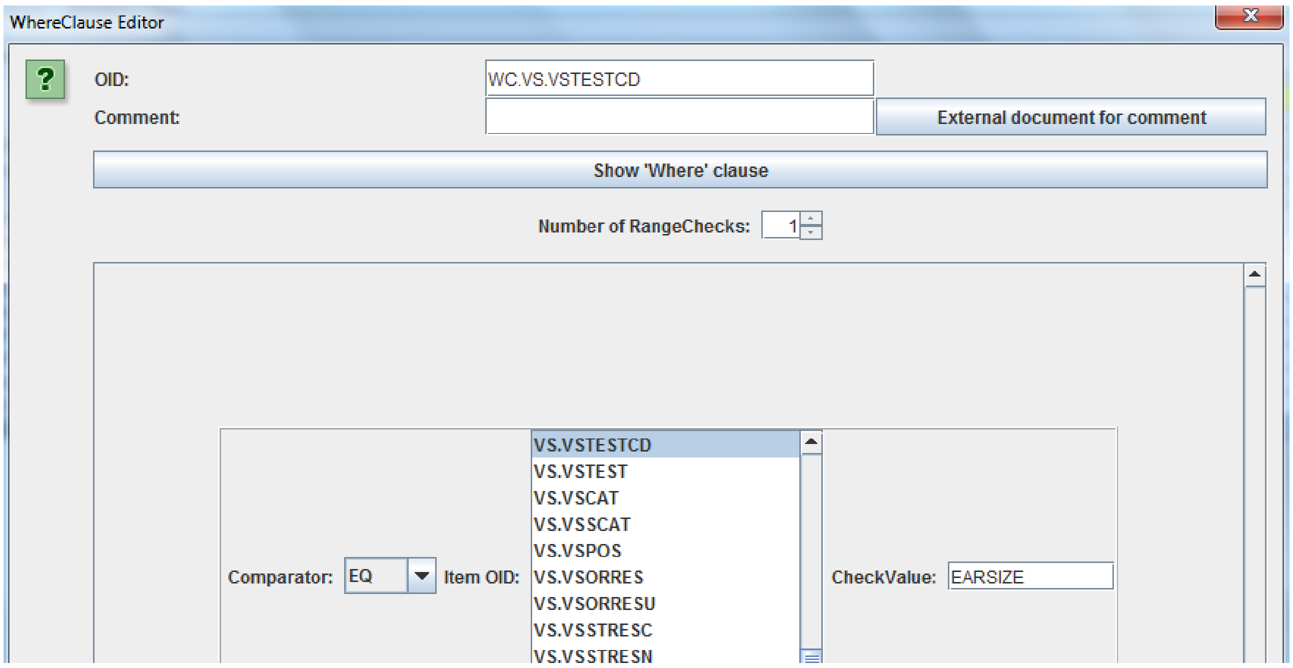
Most of our "WhereClauses" will be similar to "where VSTESTCD=DIABP" for the value-level variable "SYSBP". So, when one has added new items to the valuelist after it has been created, one can still automatically generate the "simple" "WhereClause".

For example, when we add a new entry "EARSIZE", select it, and then click the button "Generate WhereClause Automatically" is clicked, a dialog is displayed with:


WhereClause for: EARSIZE

WhereClause will be:
WHERE VSTESTCD='EARSIZE'

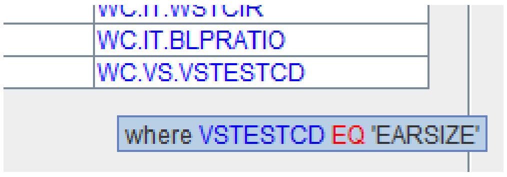
You can still cancel this now. If "OK" is clicked, the "WhereClause Wizard" shows up, allowing to still make changes to our "WhereClause":


WhereClause Editor

OID: WC.VS.VSTESTCD
Comment:
Number of RangeChecks: 1
Comparator: EQ
Item OID: VS.VSTESTCD
CheckValue: EARSIZE

Also here, we can validate whether the underlying machine-readable expression is correct, using the "Show 'Where' Clause" button. We can however also add a comment and/or add a link to a comment to an external document (like a reviewers guide).

If you feel fine, just click the "OK" button, returning to the valuelist table. We then see that the cell on the right is highlighted (blue color).When holding the mouse over the cell, we also see the "WhereClause" as a human-readable expression:

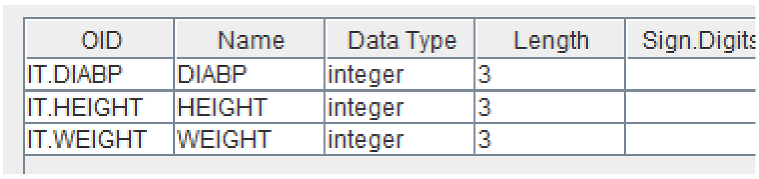


	WC.IT.WSTCIR
	WC.IT.BLPRATIO
	WC.VS.VSTESTCD

where VSTESTCD EQ 'EARSIZE'

Just for the exercise, let us know make a single "WhereClause" for as well the systolic blood pressure. This makes sense, as both are very similar properties, both are measured as an integer with a maximal length of 3.

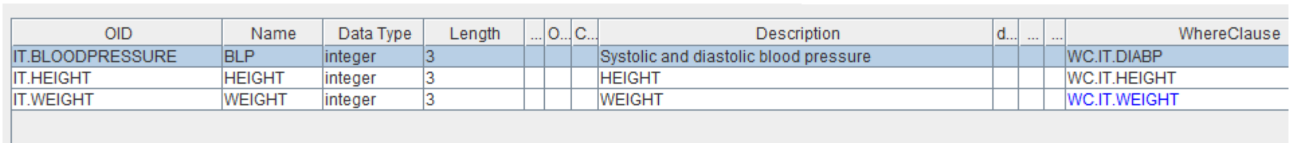
First, we remove one of both entries. Let's says we just keep the first which is "DIABP", and remove "SYSBP":



OID	Name	Data Type	Length	Sign.Digits
IT.DIABP	DIABP	integer	3	
IT.HEIGHT	HEIGHT	integer	3	
IT.WEIGHT	WEIGHT	integer	3	

Then we edit the OID and the Name to reflect that this is about blood pressure. For example:

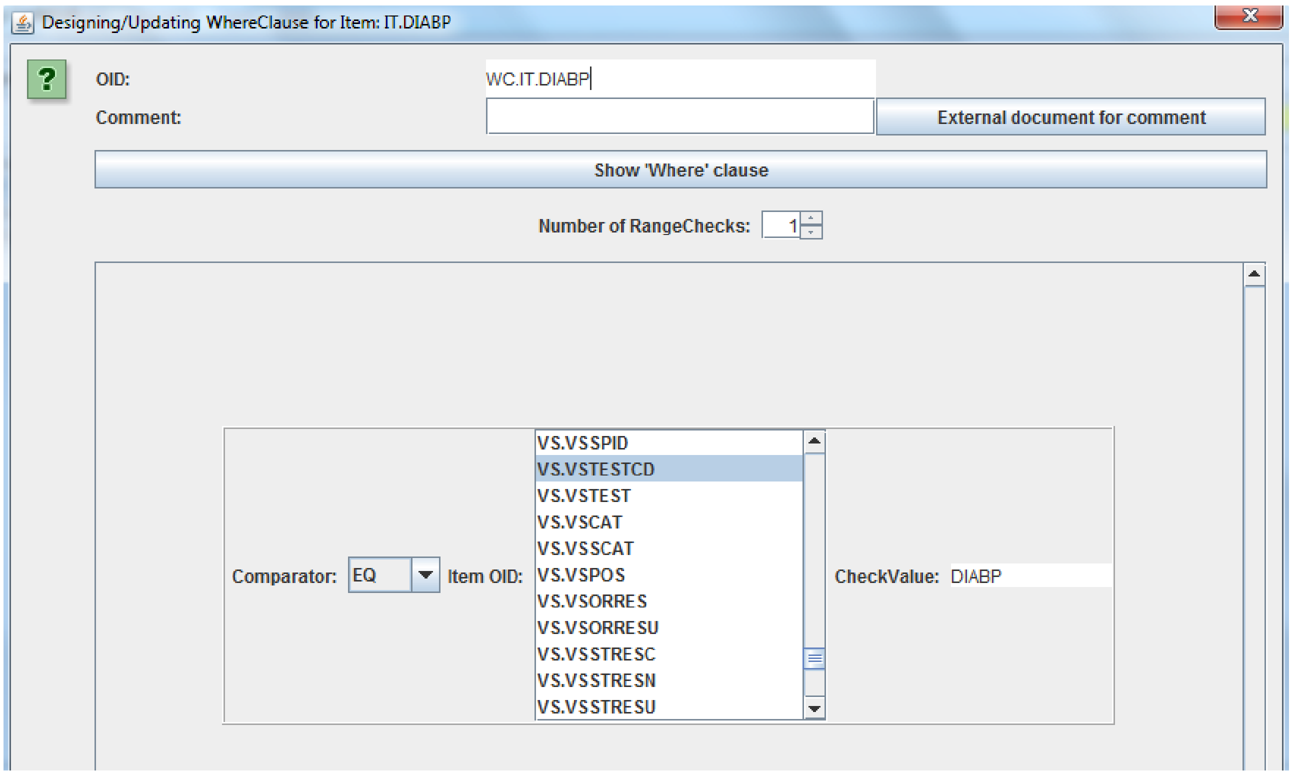
Remember that (as long as we are forced to use SAS-XPT), the value for "Name" may not be longer than 8 characters. For the "Description", we write "Systolic and diastolic blood pressure". Due again to the SAS-XPT limitation, the "Description" may not be longer than 40 characters[7]:



OID	Name	Data Type	Length	...	O...	C...	Description	d...	...	...	WhereClause
IT.BLOODPRESSURE	BLP	integer	3				Systolic and diastolic blood pressure				WC.IT.DIABP
IT.HEIGHT	HEIGHT	integer	3				HEIGHT				WC.IT.HEIGHT
IT.WEIGHT	WEIGHT	integer	3				WEIGHT				WC.IT.WEIGHT

The one thing we still need to adapt is the "WhereClause". So we click the corresponding cell on the right, and the "WhereClause Wizard" is displayed:

7 Well, isn't it time the FDA moves away from SAS-XPT?


Designing/Updating WhereClause for Item: IT.DIABP

OID: WC.IT.DIABP

Comment:

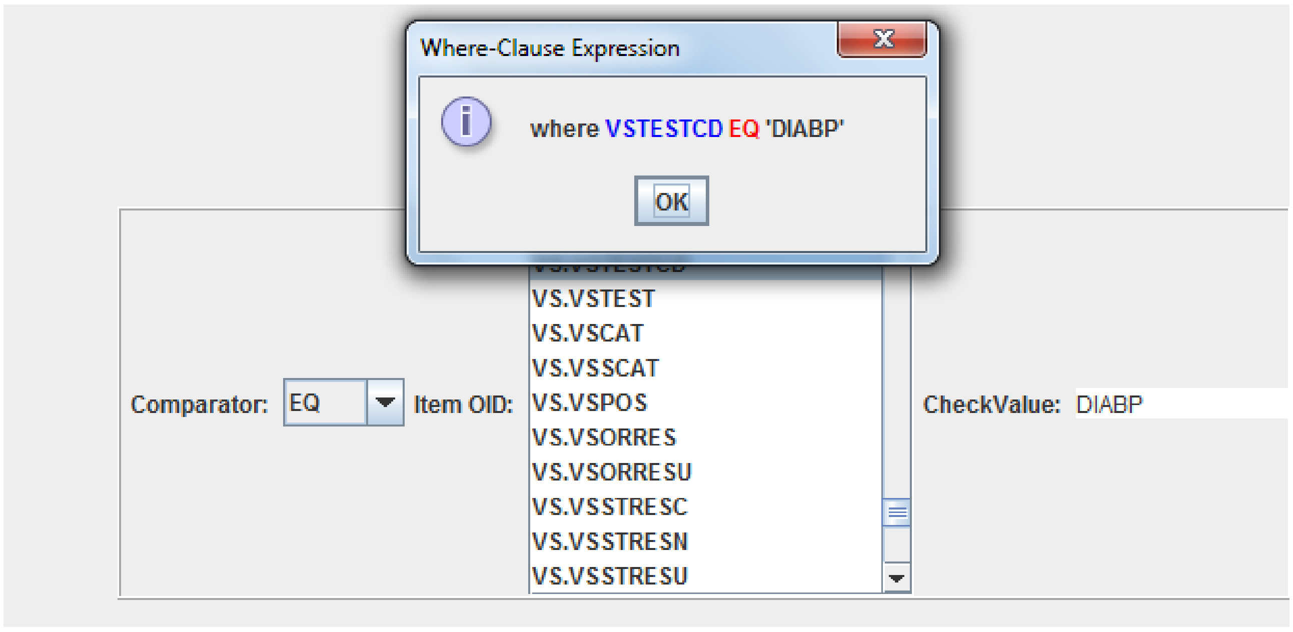
Number of RangeChecks: 1

Comparator: EQ

Item OID:
VS.VSSPID
VS.VSTESTCD
VS.VSTEST
VS.VSCAT
VS.VSSCAT
VS.VSPOS
VS.VSORRES
VS.VSORRESU
VS.VSSTRESC
VS.VSSTRESN
VS.VSSTRESU

CheckValue: DIABP

using the "Show 'Where' Clause" button, we see that currently, only DIABP is covered:


Where-Clause Expression

where VSTESTCD EQ 'DIABP'

OK

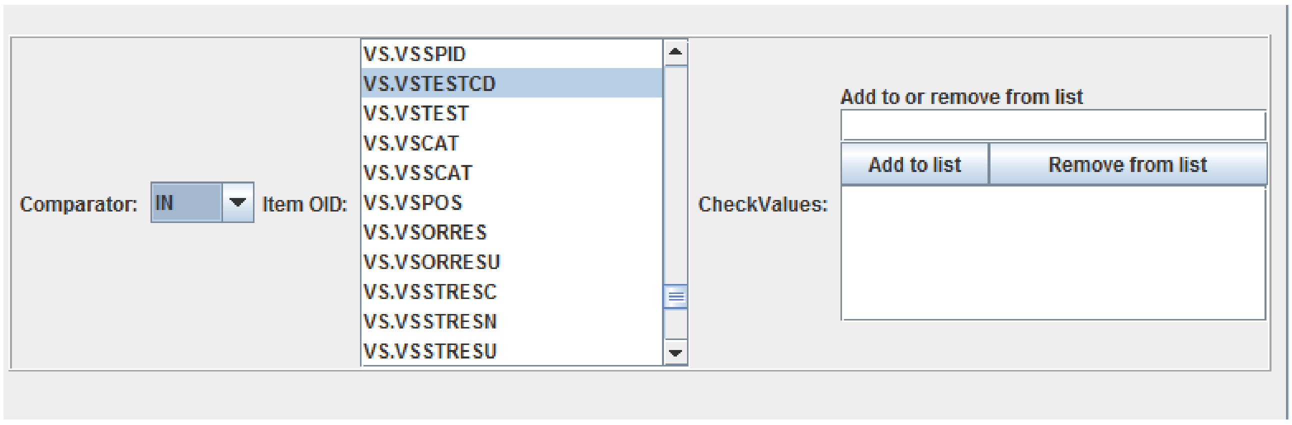
Comparator: EQ

Item OID:

VS.VSTEST
VS.VSCAT
VS.VSSCAT
VS.VSPOS
VS.VSORRES
VS.VSORRESU
VS.VSSTRESC
VS.VSSTRESN
VS.VSSTRESU

CheckValue: DIABP

but we of course also want it to cover SYSBP. Such an "OR" statement is essentially accomplished by providing a list, and then stating that the value must be in the list.
So we set the "Comparator" to "IN" (using the dropdown), leading to:


Comparator: IN

Item OID:

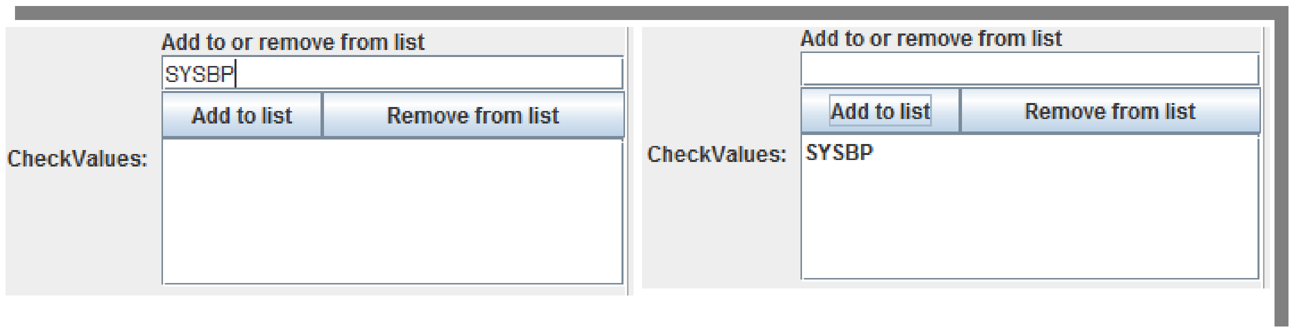
VS.VSSPID
VS.VSTESTCD
VS.VSTEST
VS.VSCAT
VS.VSSCAT
VS.VSPOS
VS.VSORRES
VS.VSORRESU
VS.VSSTRESC
VS.VSSTRESN
VS.VSSTRESU

CheckValues:

Add to or remove from list

Add to list | Remove from list

"VSTESTCD" is still OK, and we still need to add "SYSBP" and "DIABP" to our list. We can just do by typing into the "Add to or remove from list", and then click "Add to list", like:


Add to or remove from list

SYSBP

Add to list | Remove from list

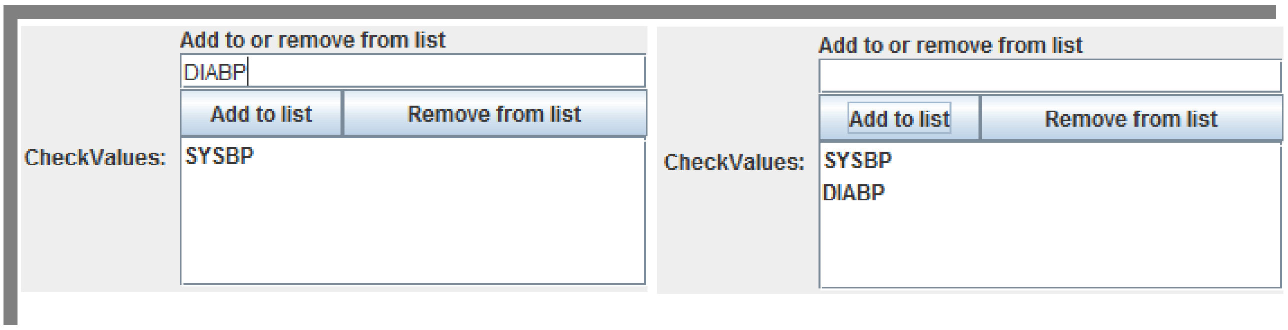
CheckValues:

Add to or remove from list

Add to list | Remove from list

CheckValues: SYSBP

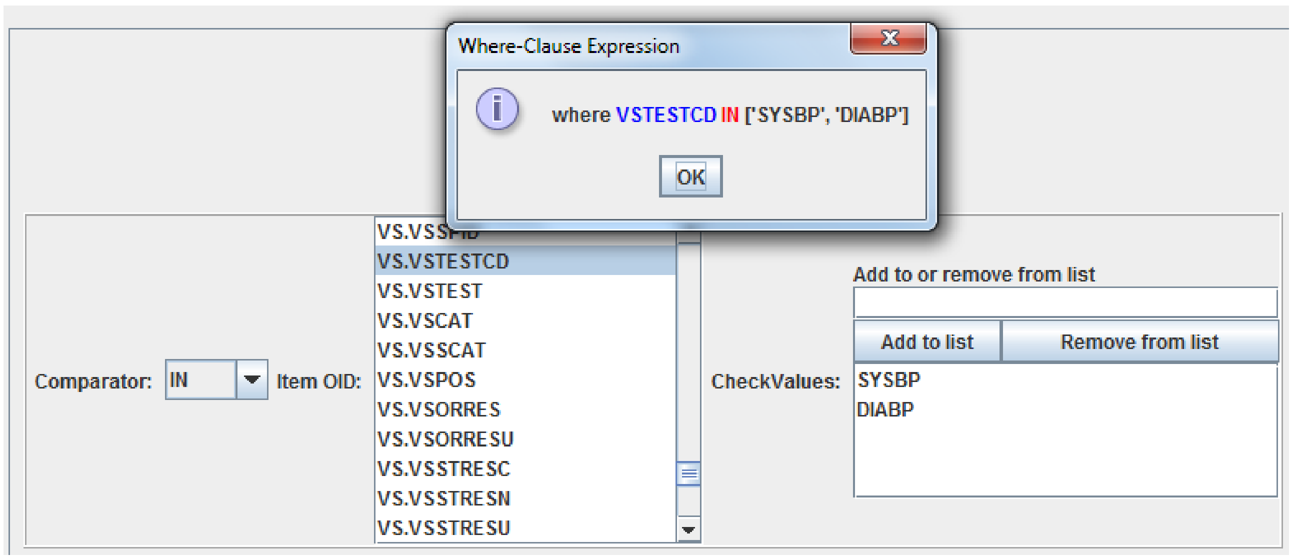
and again for "DIABP":



Add to or remove from list: DIABP
CheckValues: SYSBP

Add to or remove from list:
CheckValues: SYSBP, DIABP

When then clicking the "Show 'Where' Clause again, we get:



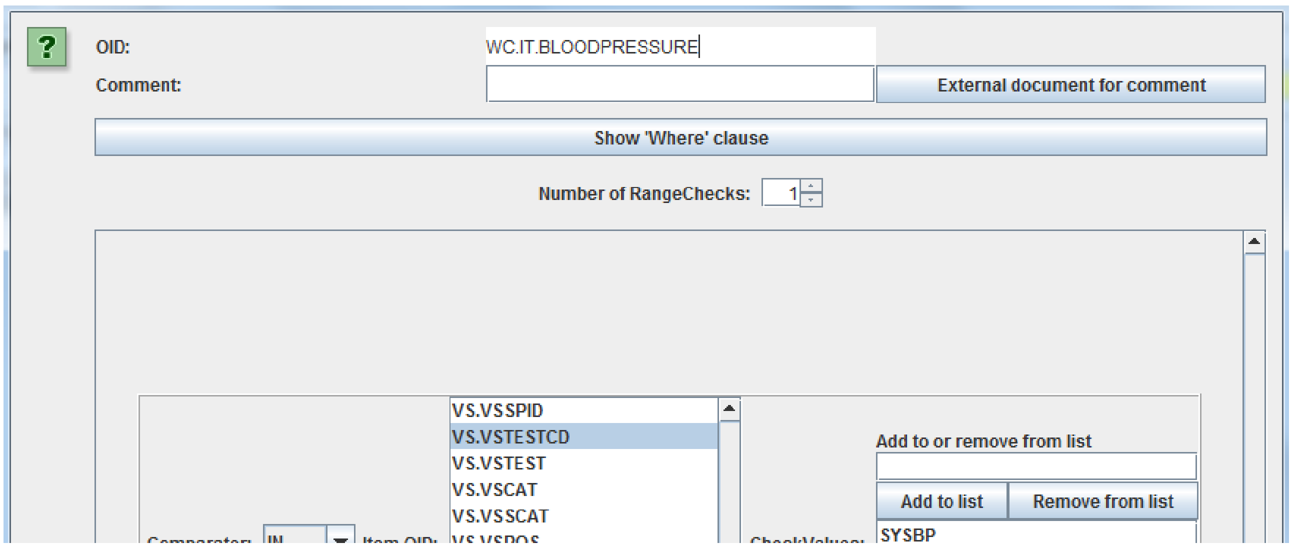
Where-Clause Expression

where VSTESTCD IN ['SYSBP', 'DIABP']

Comparator: IN
Item OID: VS.VSTESTCD
CheckValues: SYSBP, DIABP

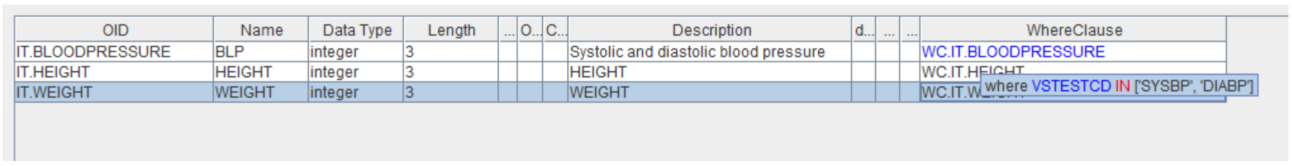
which is exactly what we want.

We also change the OID (although it is nothing else than an identifier, so it is not mandatory, but we like our OIDs to be "mnenomic", so we make it:



OID: WC.IT.BLOODPRESSURE
Comment:
Number of RangeChecks: 1
CheckValues: SYSBP

Clicking "OK" then brings us back to the valuelist table:



OID	Name	Data Type	Length	...	O...	C...	Description	d...	...	...	WhereClause
IT.BLOODPRESSURE	BLP	integer	3				Systolic and diastolic blood pressure				WC.IT.BLOODPRESSURE
IT.HEIGHT	HEIGHT	integer	3				HEIGHT				WC.IT.HEIGHT
IT.WEIGHT	WEIGHT	integer	3				WEIGHT				WC.IT.W

where VSTESTCD IN ['SYSBP', 'DIABP']

again showing the "human-readable" expression as a tooltip.

We should not forget to validate the table, using the "Validate" button which is near the bottom. Using it leads to … nothing, meaning that our "ValueList" table is technically valid.

We could still add the "Origin", but we could also do that at the higher level, when all vital signs where collected on the same CRF "page"[8]. We will describe how to semi-automatically add CRF page numbers in one of the next sections.

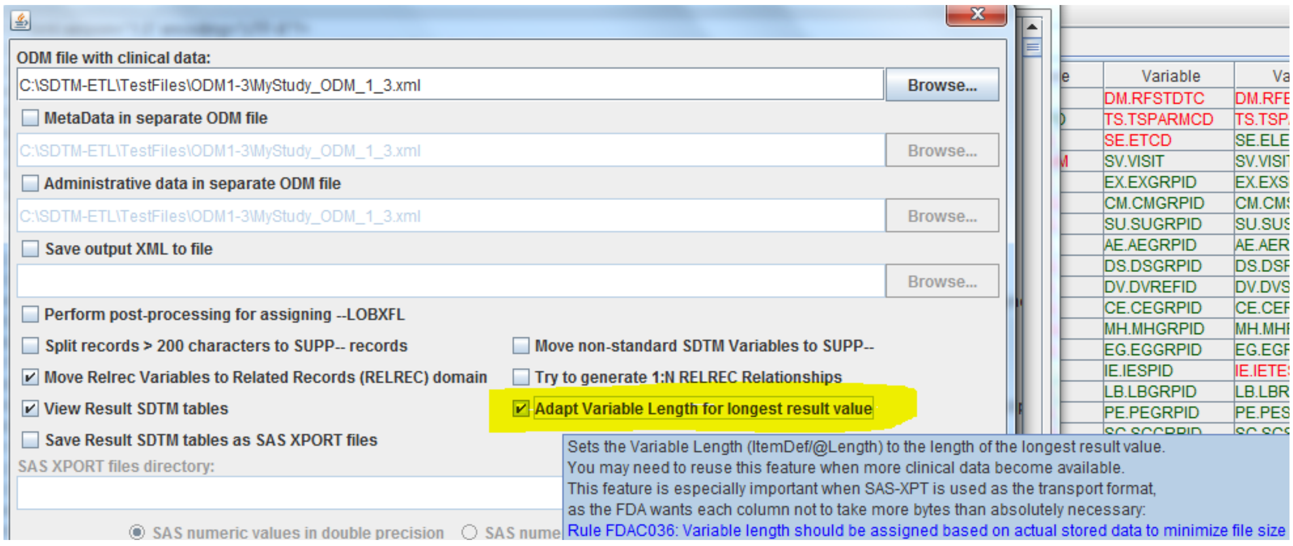
8 When using EDC, this essentially does not make sense. As we start from a study design in ODM, our mapping script IS the description of the origin. However, the FDA still want sponsors to generate a paper-like annotated CRF artificially, with page numbers, although the EDC screens do not have such at all.

Automated adaption of variable length from longest result value

The FDA requires that the variable lengths in the SAS-XPT files exactly match the length of the longest value in the variable column[9]. So they introduced the famous rule FDAC036 "Variable length should be assigned based on actual stored data to minimize file size. Datasets should be re-sized to the maximum length of actual data used...".

Essentially, this means that one would need a postprocessing step when generating the SAS-XPT files, as one cannot know the "maximal length" in advance. We did however find a smarter way.

When generating the XSLT transformation script and then executing it, the following dialog is displayed:



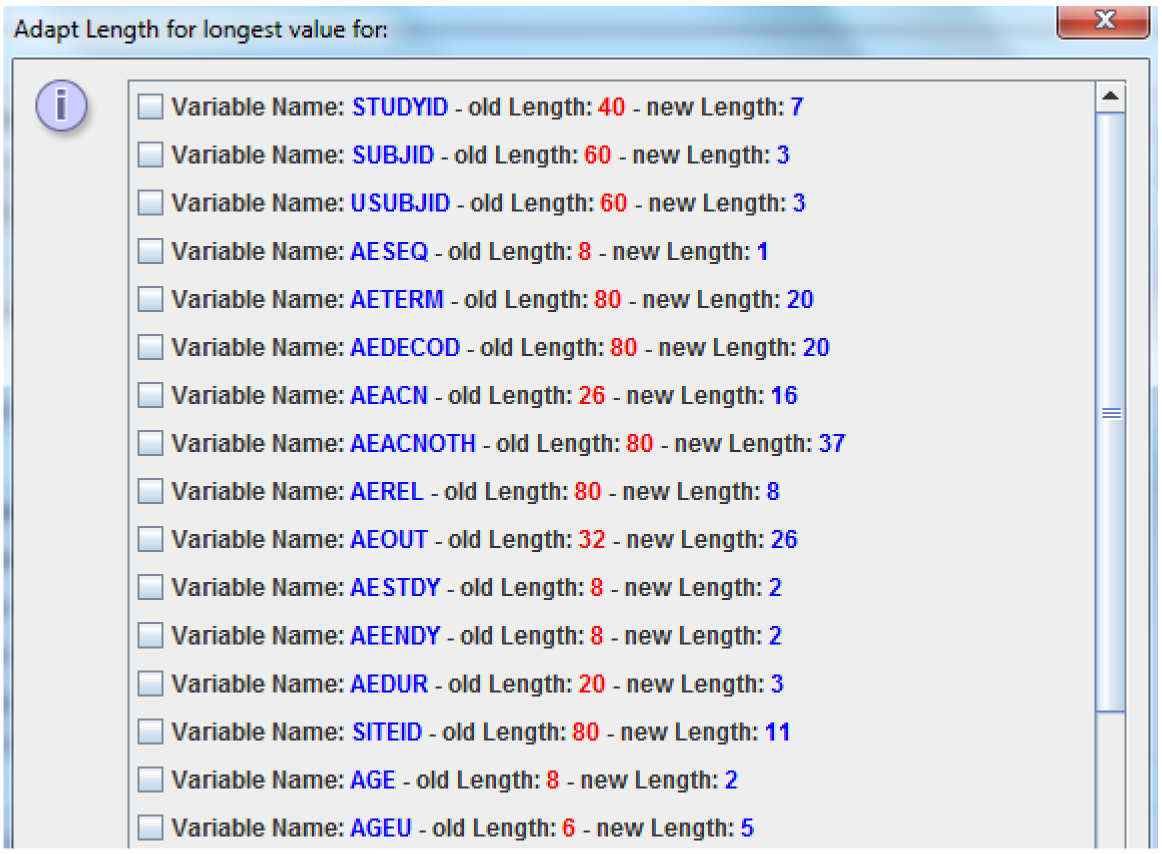
ODM file with clinical data:
C:\SDTM-ETL\TestFiles\ODM1-3\MyStudy_ODM_1_3.xml [Browse...]
☐ MetaData in separate ODM file
C:\SDTM-ETL\TestFiles\ODM1-3\MyStudy_ODM_1_3.xml [Browse...]
☐ Administrative data in separate ODM file
C:\SDTM-ETL\TestFiles\ODM1-3\MyStudy_ODM_1_3.xml [Browse...]
☐ Save output XML to file
[Browse...]
☐ Perform post-processing for assigning --LOBXFL
☐ Split records > 200 characters to SUPP-- records
☐ Move non-standard SDTM Variables to SUPP--
☑ Move Relrec Variables to Related Records (RELREC) domain
☐ Try to generate 1:N RELREC Relationships
☑ View Result SDTM tables
☑ Adapt Variable Length for longest result value
☐ Save Result SDTM tables as SAS XPORT files
SAS XPORT files directory:
● SAS numeric values in double precision ○ SAS nume

Sets the Variable Length (ItemDef/@Length) to the length of the longest result value.
You may need to reuse this feature when more clinical data become available.
This feature is especially important when SAS-XPT is used as the transport format,
as the FDA wants each column not to take more bytes than absolutely necessary:
Rule FDAC036: Variable length should be assigned based on actual stored data to minimize file size

e	Variable	Va
	DM.RFSTDTC	DM.RFE
D	TS.TSPARMCD	TS.TSP
	SE.ETCD	SE.ELE
M	SV.VISIT	SV.VISI
	EX.EXGRPID	EX.EXS
	CM.CMGRPID	CM.CMS
	SU.SUGRPID	SU.SUS
	AE.AEGRPID	AE.AER
	DS.DSGRPID	DS.DSF
	DV.DVREFID	DV.DVS
	CE.CEGRPID	CE.CEF
	MH.MHGRPID	MH.MHF
	EG.EGGRPID	EG.EGF
	IE.IESPID	IE.IETE
	LB.LBGRPID	LB.LBR
	PE.PEGRPID	PE.PES

Remark the new checkbox "Adapt Variable Length for longest result value".
When it is checked, the data will be generated from the mapping scripts, but not stored immediately into the SAS-XPT files, but instead they will be parsed again, and for each variable, the largest value length will be retained. This then leads to a new dialog:

9 This is a ridiculous requirement: the reason behind it is the unfounded fear of the FDA for large file sizes and the fact that SAS-XPT is a "constant record length" format, where blanks are inserted after the text value itself to have each value (per variable) the same length, and thus all records the same length. So SAS-XPT uses a format that is wasting an enormous amount of bytes. So SAS-XPT is a very inefficient format. Although this and the fact that SAS-XPT is a completely outdated format (stemming from the IBM mainframe era), the FDA requires us to use this outdated format for submissions. On the other hand, the FDA has over and over again added new derived variables in SDTM, introducing dangerous data redundancy and further enormously increasing the file sizes. In order to somewhat counteract these (by the FDA (unfounded) feared) large file sizes, they set a requirement in force that states that the length of a variable must exactly be the length of the longest value for that variable.
Modern information technology looks different...


Adapt Length for longest value for:

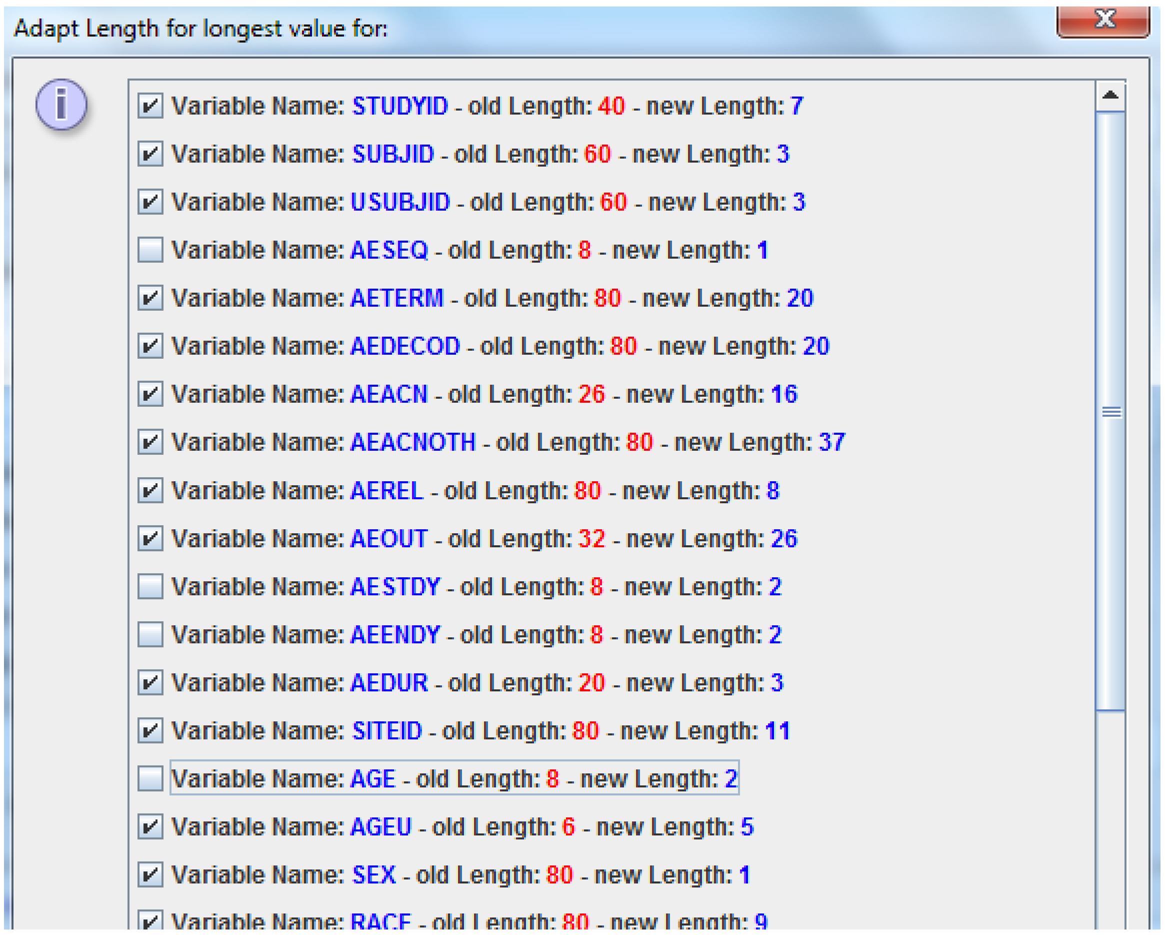
- Variable Name: STUDYID - old Length: 40 - new Length: 7
- Variable Name: SUBJID - old Length: 60 - new Length: 3
- Variable Name: USUBJID - old Length: 60 - new Length: 3
- Variable Name: AESEQ - old Length: 8 - new Length: 1
- Variable Name: AETERM - old Length: 80 - new Length: 20
- Variable Name: AEDECOD - old Length: 80 - new Length: 20
- Variable Name: AEACN - old Length: 26 - new Length: 16
- Variable Name: AEACNOTH - old Length: 80 - new Length: 37
- Variable Name: AEREL - old Length: 80 - new Length: 8
- Variable Name: AEOUT - old Length: 32 - new Length: 26
- Variable Name: AESTDY - old Length: 8 - new Length: 2
- Variable Name: AEENDY - old Length: 8 - new Length: 2
- Variable Name: AEDUR - old Length: 20 - new Length: 3
- Variable Name: SITEID - old Length: 80 - new Length: 11
- Variable Name: AGE - old Length: 8 - new Length: 2
- Variable Name: AGEU - old Length: 6 - new Length: 5

displaying all the variables where the maximal length found during generation of the data deviates from the information in our underlying define.xml. For example, our define.xml still contained a (from the template) maximal length of 40, whereas the STUDYID always has the value "MyStudy" which is only 7 characters. Similarly, our "SUBJID" values are "001","002", etc., so never longer than 3 characters.

The user can now decide for which variables the lengths in the define.xml will be adapted, and as the define.xml is stearing the generation of the SDTM files (as it should be), these will then also be the variable lengths in the SAS-XPT files. For text variables, this will usually make sense, for numeric variables, it will usually not be, as a numeric variable in SAS-XPT will always take 8 bytes[10] anyway. The "Length" in the define.xml for numeric values is the length in characters for the number anyway.

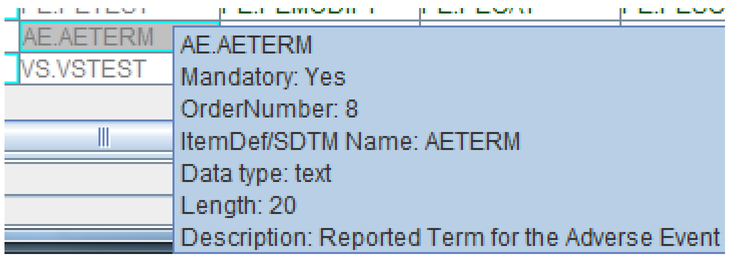
10 Another waste of bytes ...

So, let us select which variable lengths we want to adapt automatically, e.g.:


Adapt Length for longest value for:

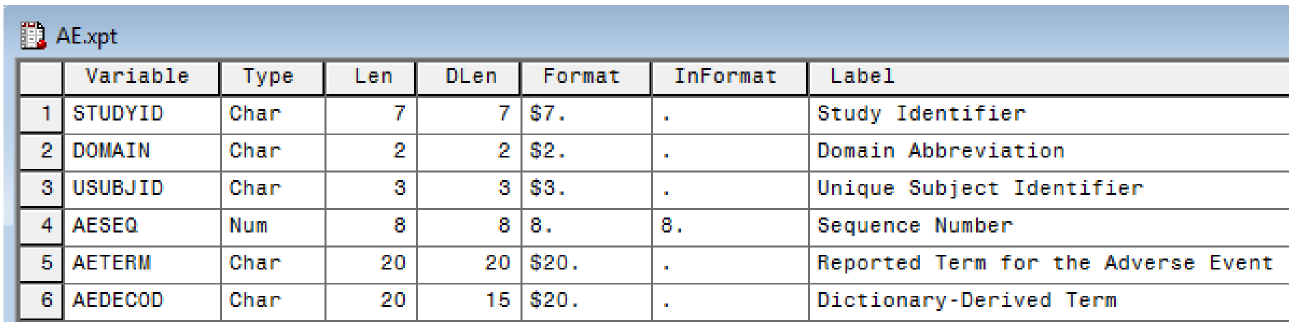
- ✓ Variable Name: STUDYID - old Length: 40 - new Length: 7
- ✓ Variable Name: SUBJID - old Length: 60 - new Length: 3
- ✓ Variable Name: USUBJID - old Length: 60 - new Length: 3
- ☐ Variable Name: AESEQ - old Length: 8 - new Length: 1
- ✓ Variable Name: AETERM - old Length: 80 - new Length: 20
- ✓ Variable Name: AEDECOD - old Length: 80 - new Length: 20
- ✓ Variable Name: AEACN - old Length: 26 - new Length: 16
- ✓ Variable Name: AEACNOTH - old Length: 80 - new Length: 37
- ✓ Variable Name: AEREL - old Length: 80 - new Length: 8
- ✓ Variable Name: AEOUT - old Length: 32 - new Length: 26
- ☐ Variable Name: AESTDY - old Length: 8 - new Length: 2
- ☐ Variable Name: AEENDY - old Length: 8 - new Length: 2
- ✓ Variable Name: AEDUR - old Length: 20 - new Length: 3
- ✓ Variable Name: SITEID - old Length: 80 - new Length: 11
- ☐ Variable Name: AGE - old Length: 8 - new Length: 2
- ✓ Variable Name: AGEU - old Length: 6 - new Length: 5
- ✓ Variable Name: SEX - old Length: 80 - new Length: 1
- ✓ Variable Name: RACE - old Length: 80 - new Length: 9

After executing the transformation, we can easily see that the length have indeed been adapted in the underlying define.xml. For example for the variable AETERM:


AE.AETERM
VS.VSTEST

AE.AETERM
Mandatory: Yes
OrderNumber: 8
ItemDef/SDTM Name: AETERM
Data type: text
Length: 20
Description: Reported Term for the Adverse Event

and in the SAS-XPT file (using the SAS-Viewer):


AE.xpt

	Variable	Type	Len	DLen	Format	InFormat	Label
1	STUDYID	Char	7	7	$7.	.	Study Identifier
2	DOMAIN	Char	2	2	$2.	.	Domain Abbreviation
3	USUBJID	Char	3	3	$3.	.	Unique Subject Identifier
4	AESEQ	Num	8	8	8.	8.	Sequence Number
5	AETERM	Char	20	20	$20.	.	Reported Term for the Adverse Event
6	AEDECOD	Char	20	15	$20.	.	Dictionary-Derived Term

When using Dataset-XML as the output format, the same checkbox is found again, but in this case the issue is very less critical, as XML is not a fixed-length record, and is much more efficient with storing data than SAS-XPT is[11]. In the case of Dataset-XML however, the "Length" instructions in the define.xml can e.g. be used to generate a relational database (i.e. create SQL "CREATE TABLE" statements), where the "Length" is used to make a "VARCHAR(length)" statement. As "VARCHAR" is a very efficient storage format, it is absolutely not necessary to set the length to the maximum of the encountered values, at the contrary, one can then better set it somewhat larger than the longest encountered variable value length.

Anyway, this new feature easily allows to obey the (although ridiculous) FDAC036 rule, without needing to do a postprocessing step, as is usually needed when using other software packages.

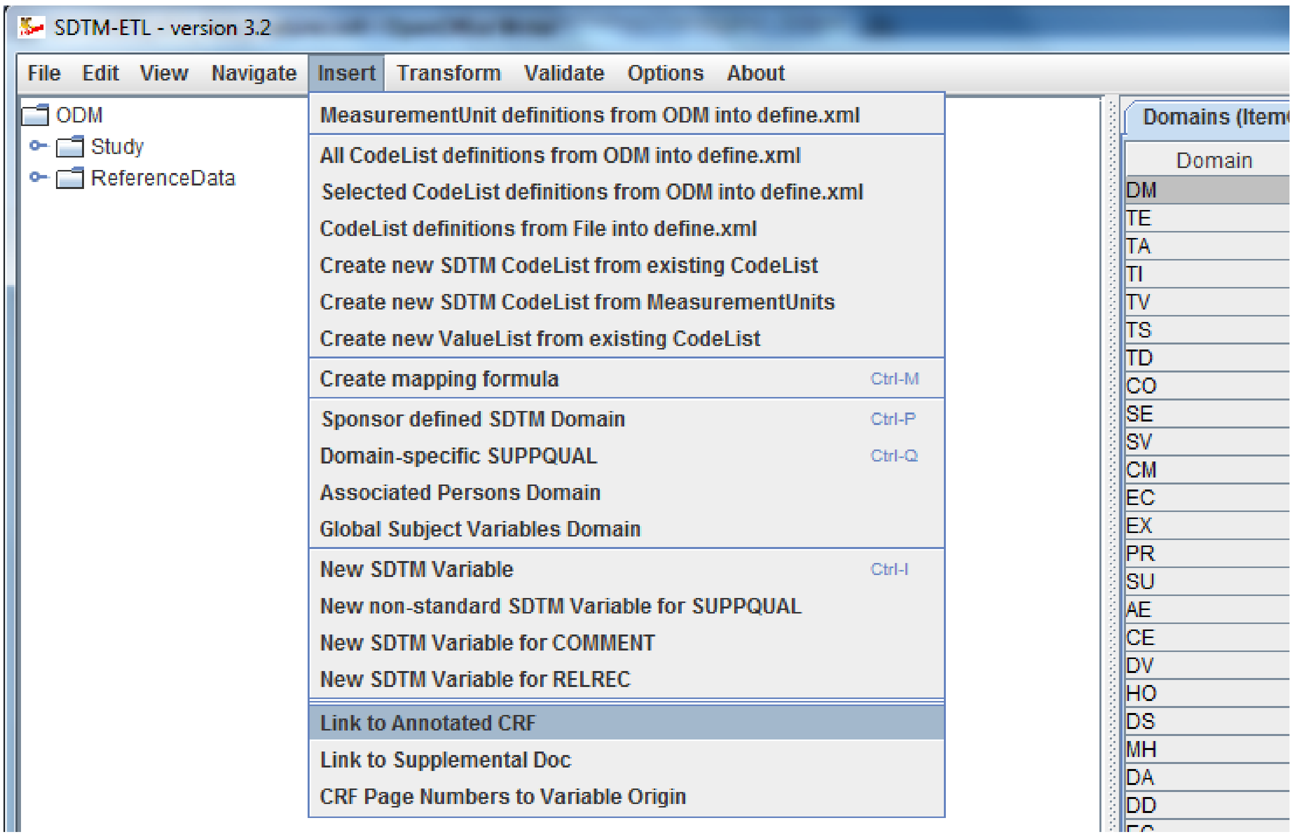
Important remark: when more data become available during the study, you might need to re-use this feature, as the new data can contain values with a larger length then when you first used this new feature.

11 This doesn't mean that XML files are always smaller in size. XML usually uses UTF-8 (allowing non-ASCII) characters, and XML also contains the tags. In the past, we found that, for the same data, the XML files for normal domains are usually larger when using XML with respect to SAS-XPT, whereas datasets for "special" domains such as CO and supplemental qualifier domains, the SAS-XPT are considerably larger than the SAS-XPT files. Filesize shouldn't be an issue anyway however, as the amount of storage in a database is independent from the transport format, and the cost of storage of even the largest submissions is typically below 1 US$.

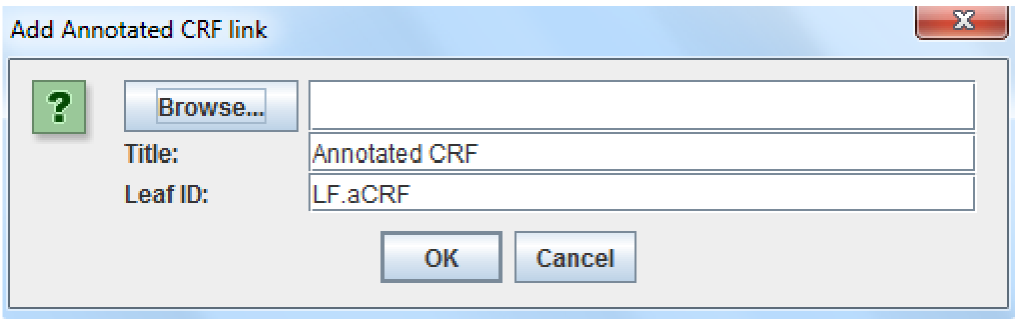
Retrieving page numbers from an annotated CRF in PDF format

Retrieving page number information from annotated Case Report Forms (aCRFs) has now become very easy using SDTM-ETL.

First, one of course needs to provide the location of the aCRF, which can be done using the menu "Insert - Link to annotated CRF":


SDTM-ETL - version 3.2
File Edit View Navigate Insert Transform Validate Options About
ODM
Study
ReferenceData
MeasurementUnit definitions from ODM into define.xml
All CodeList definitions from ODM into define.xml
Selected CodeList definitions from ODM into define.xml
CodeList definitions from File into define.xml
Create new SDTM CodeList from existing CodeList
Create new SDTM CodeList from MeasurementUnits
Create new ValueList from existing CodeList
Create mapping formula Ctrl-M
Sponsor defined SDTM Domain Ctrl-P
Domain-specific SUPPQUAL Ctrl-Q
Associated Persons Domain
Global Subject Variables Domain
New SDTM Variable Ctrl-I
New non-standard SDTM Variable for SUPPQUAL
New SDTM Variable for COMMENT
New SDTM Variable for RELREC
Link to Annotated CRF
Link to Supplemental Doc
CRF Page Numbers to Variable Origin
Domains (Item
Domain
DM
TE
TA
TI
TV
TS
TD
CO
SE
SV
CM
EC
EX
PR
SU
AE
CE
DV
HO
DS
MH
DA
DD

leading to:

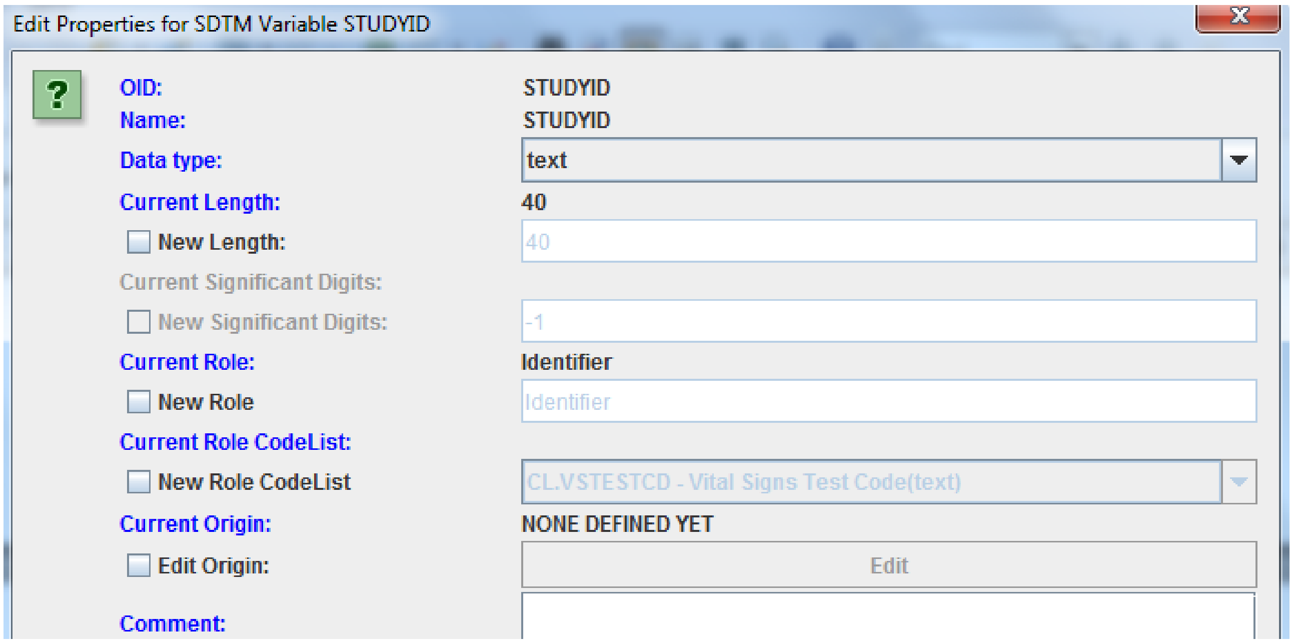


Title: Annotated CRF
Leaf ID: LF.aCRF

One gives the link a title (e.g. "Annotated CRF"), an ID (like "LF.aCRF"), and can then use the "Browse" button to add the location of the annotated CRF.

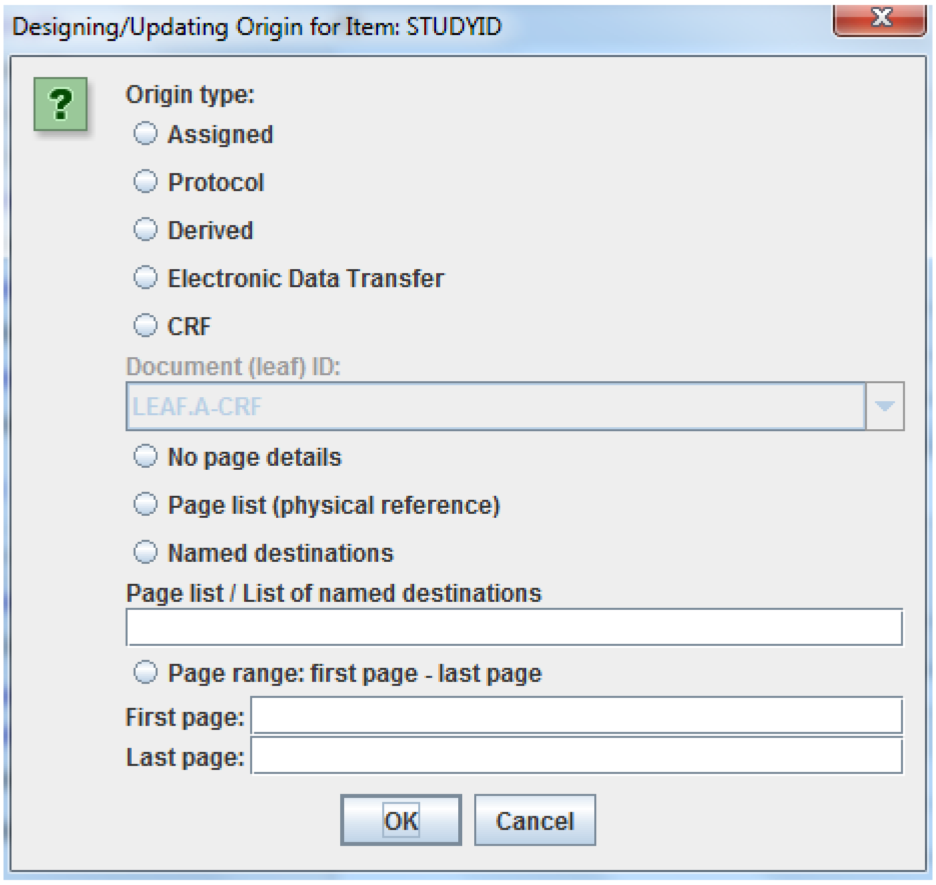
One can then later always change the information using the menu "Edit - Links to External Documents".

We have already learned how to add an "Origin" to an SDTM/SEND/ADaM variable, by selecting the cell in the table, and then using the menu "Edit - SDTM Variable Properties"[12]. For example for "STUDYID":


Edit Properties for SDTM Variable STUDYID

OID:	STUDYID
Name:	STUDYID
Data type:	text
Current Length:	40
☐ New Length:	40
Current Significant Digits:	
☐ New Significant Digits:	-1
Current Role:	Identifier
☐ New Role	Identifier
Current Role CodeList:	
☐ New Role CodeList	CL.VSTESTCD - Vital Signs Test Code(text)
Current Origin:	NONE DEFINED YET
☐ Edit Origin:	Edit
Comment:	

When then checking the "Edit Origin" checkbox, followed by clicking the "Edit" button, the "Origin" wizard will appear:


Designing/Updating Origin for Item: STUDYID

Origin type:
- ○ Assigned
- ○ Protocol
- ○ Derived
- ○ Electronic Data Transfer
- ○ CRF

Document (leaf) ID: LEAF.A-CRF

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

Page list / List of named destinations:

- ○ Page range: first page - last page

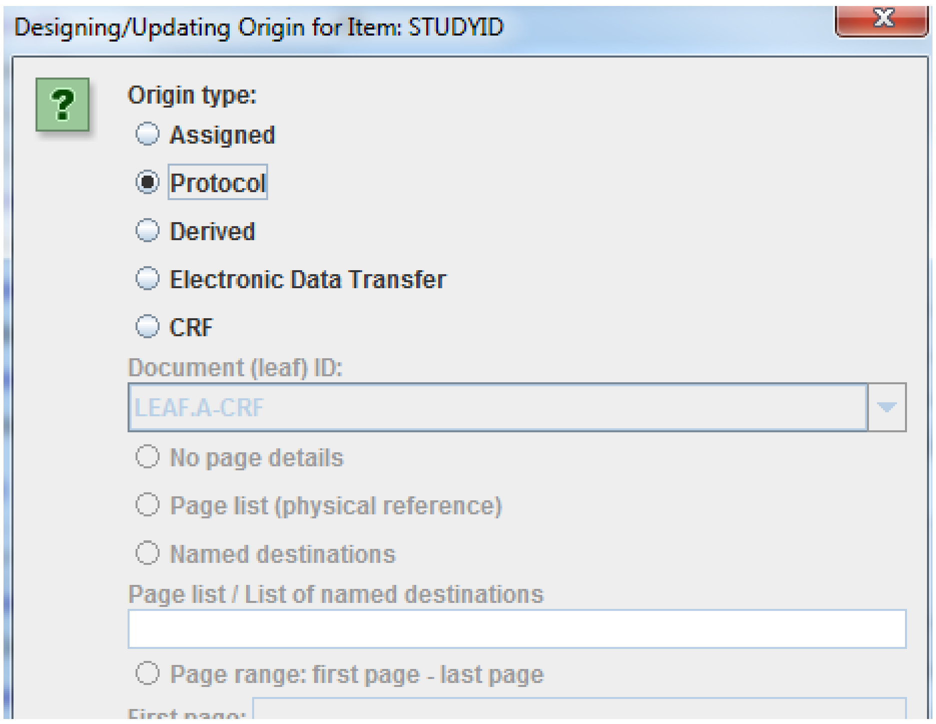
First page:

Last page:

OK | Cancel

12 In case of using SEND, the menu is automatically adapted to "SEND Variable Properties".

Depending on the origin type (Assigned, Protocol, Derived, Electronic Data Transfer, CRF), a number of fields will become disabled/enabled and to be filled or not. For example, when we state that the STUDYID is taken from the protocol, we just select "Protocol" and all other fields get disabled:


Designing/Updating Origin for Item: STUDYID

Origin type:
- ○ Assigned
- ● Protocol
- ○ Derived
- ○ Electronic Data Transfer
- ○ CRF

Document (leaf) ID: LEAF.A-CRF

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

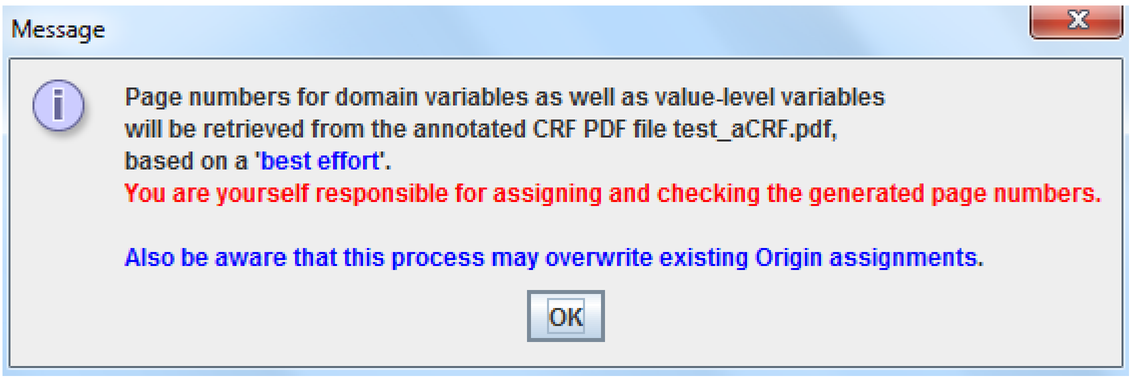
Page list / List of named destinations

- ○ Page range: first page - last page

This also allows to add page numbers in the case of the CRF being the origin, and this in a very user-friendly way. Doing so for each variable (we have hundreds of them) is still a cumbersome task, which we want to automate when possible.

If we do already have an annotated CRF, the page numbers can be semi-automatically retrieved and the "Origin" elements populated and assigned to the variables, this as well to the normal domain variables as well as to value-level variables.

To do so, use the menu "Insert - CRF Page Numbers to Variable Origin". The following information is displayed:


Message

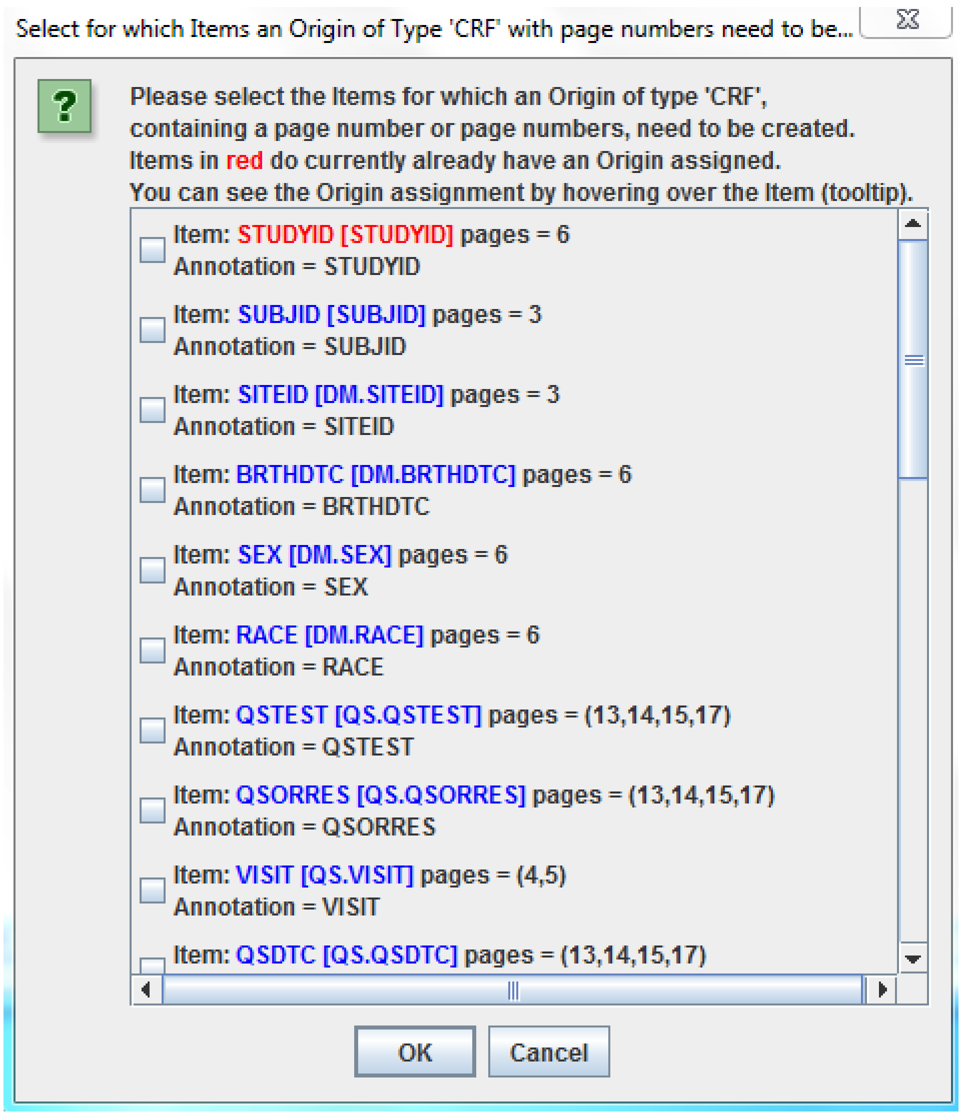
Page numbers for domain variables as well as value-level variables
will be retrieved from the annotated CRF PDF file test_aCRF.pdf,
based on a 'best effort'.
You are yourself responsible for assigning and checking the generated page numbers.

Also be aware that this process may overwrite existing Origin assignments.

Unlike other (black box) tools on the market, you will be guided through the process, allowing you to decide yourself which page numbers will be copied to the variables, and which not.
Normally, PDF page numbers will only be retrieved for variables for which you already provided a

mapping. If your mapping is not complete yet, you can later repeat the process if you want - the software will tell you for which variables you do have already the page numbers assigned and for which not.

After clicking OK, the system is retrieving all information (this may take a few seconds), and the following dialog is displayed:

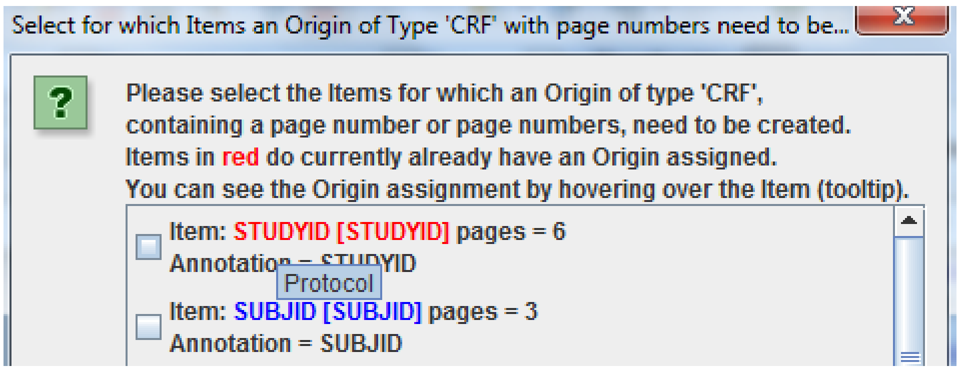


Select for which Items an Origin of Type 'CRF' with page numbers need to be...

Please select the Items for which an Origin of type 'CRF', containing a page number or page numbers, need to be created. Items in red do currently already have an Origin assigned. You can see the Origin assignment by hovering over the Item (tooltip).

- ☐ Item: STUDYID [STUDYID] pages = 6
  Annotation = STUDYID
- ☐ Item: SUBJID [SUBJID] pages = 3
  Annotation = SUBJID
- ☐ Item: SITEID [DM.SITEID] pages = 3
  Annotation = SITEID
- ☐ Item: BRTHDTC [DM.BRTHDTC] pages = 6
  Annotation = BRTHDTC
- ☐ Item: SEX [DM.SEX] pages = 6
  Annotation = SEX
- ☐ Item: RACE [DM.RACE] pages = 6
  Annotation = RACE
- ☐ Item: QSTEST [QS.QSTEST] pages = (13,14,15,17)
  Annotation = QSTEST
- ☐ Item: QSORRES [QS.QSORRES] pages = (13,14,15,17)
  Annotation = QSORRES
- ☐ Item: VISIT [QS.VISIT] pages = (4,5)
  Annotation = VISIT
- ☐ Item: QSDTC [QS.QSDTC] pages = (13,14,15,17)

In each row, one sees the suggested variable, the page number or numbers, and the PDF annotation itself. Rows with a red text indicate that there is an "Origin" assignment yet, and one can easily find out what assignment is already present, by hovering the mouse over the row. A tooltip is then displayed:


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

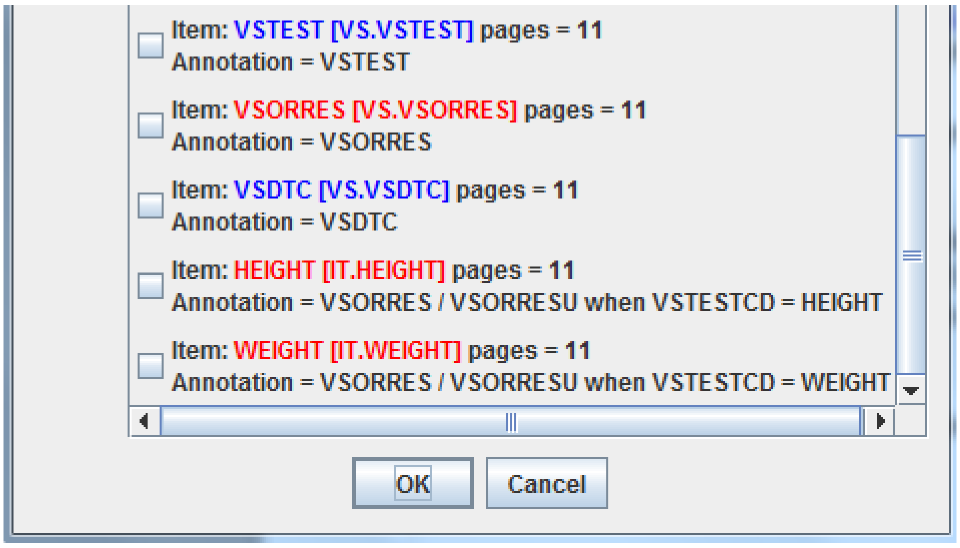
Please select the Items for which an Origin of type 'CRF', containing a page number or page numbers, need to be created.
Items in red do currently already have an Origin assigned.
You can see the Origin assignment by hovering over the Item (tooltip).

Item: STUDYID [STUDYID] pages = 6
Annotation = STUDYID
Protocol
Item: SUBJID [SUBJID] pages = 3
Annotation = SUBJID

If the assignment were a CRF page or pages, also the latter will be displayed in the tooltip (see later).

As a user, you can now decide yourself which page numbers must be copied to the variable "Origin". For example, an annotation of "STUDYID" was found on page 6, but you want to keep the old assignment, stating that it comes from the protocol, so you would in this case not check the checkbox.

Suggestions are also generated for value-level variables. For example, near the bottom of the list we find:


Item: VSTEST [VS.VSTEST] pages = 11
Annotation = VSTEST

Item: VSORRES [VS.VSORRES] pages = 11
Annotation = VSORRES

Item: VSDTC [VS.VSDTC] pages = 11
Annotation = VSDTC

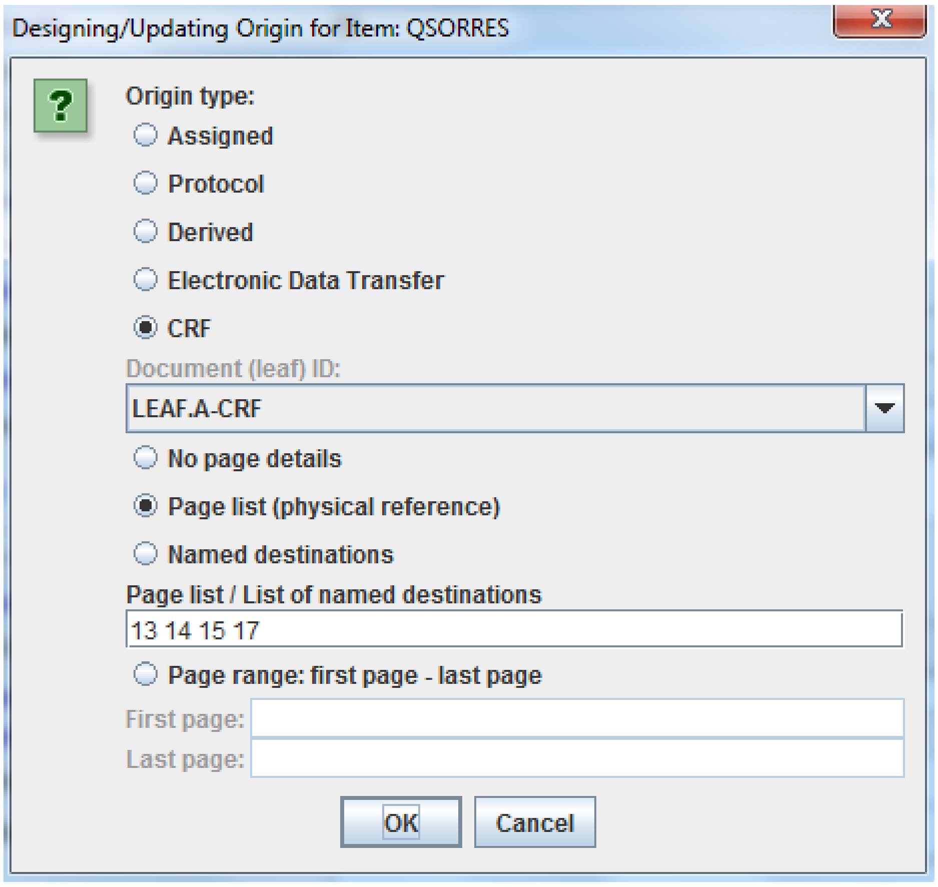
Item: HEIGHT [IT.HEIGHT] pages = 11
Annotation = VSORRES / VSORRESU when VSTESTCD = HEIGHT

Item: WEIGHT [IT.WEIGHT] pages = 11
Annotation = VSORRES / VSORRESU when VSTESTCD = WEIGHT

stating that annotations were found on page 11 about value-level variables "HEIGHT" and "WEIGHT". It is also indicated that an "Origin" assignment was already present, so if we check these checkboxes, the old assignments will be overwritten.

One can now decide for which variables (domain or value-level variables) the page numbers are copied. For example, we check all of them, except for "STUDYID". The assignments are then automatically performed, but only for the variables for which the checkbox was checked.

If we later than inspect the "Origin" for e.g. the variable "QSORRES" (using the menu "Edit - SDTM Variable Properties" again, we find:


Designing/Updating Origin for Item: QSORRES

Origin type:

- ○ Assigned
- ○ Protocol
- ○ Derived
- ○ Electronic Data Transfer
- ● CRF

Document (leaf) ID: LEAF.A-CRF

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

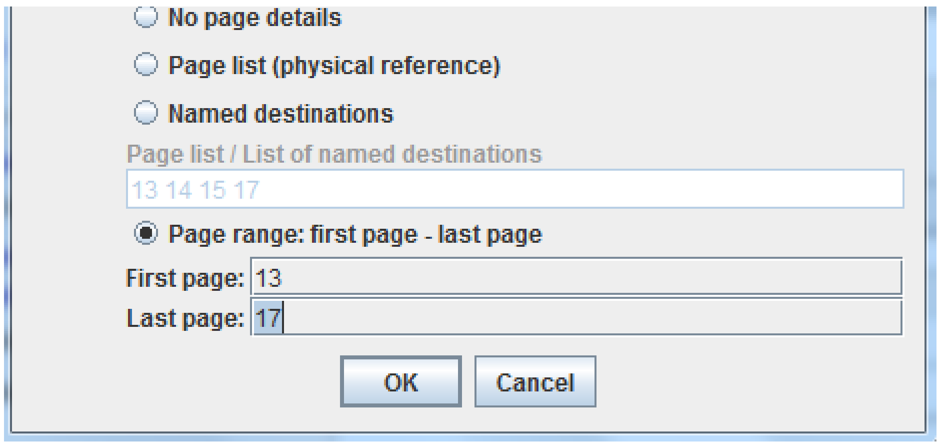
Page list / List of named destinations: 13 14 15 17

- ○ Page range: first page - last page

First page:

Last page:

We can then still change this, e.g. add page numbers, of changing the list into a "first page - last page" in this case (just as an example) set it to pages 13-17:


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

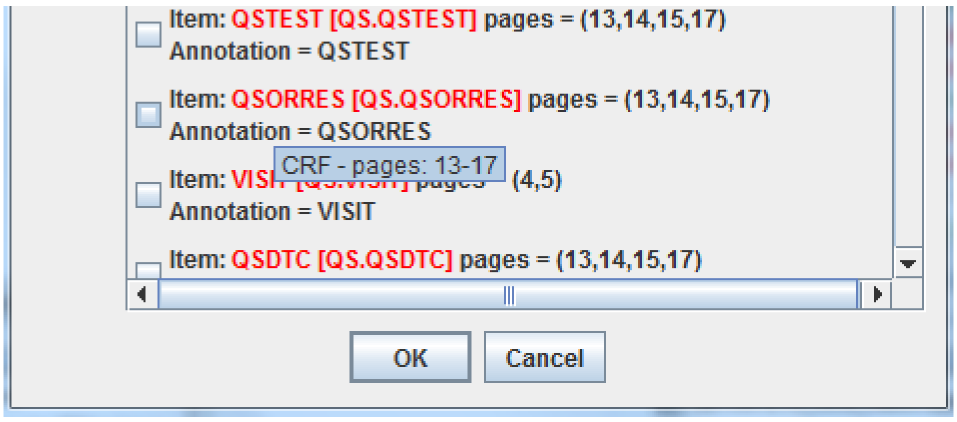
Page list / List of named destinations: 13 14 15 17

- ● Page range: first page - last page

First page: 13

Last page: 17

When one later then repeats the whole process, all the variables that already were assigned an origin are marked as such (red color), for example:


Item: QSTEST [QS.QSTEST] pages = (13,14,15,17)
Annotation = QSTEST

Item: QSORRES [QS.QSORRES] pages = (13,14,15,17)
Annotation = QSORRES

Item: VISI CRF - pages: 13-17 (4,5)
Annotation = VISIT

Item: QSDTC [QS.QSDTC] pages = (13,14,15,17)

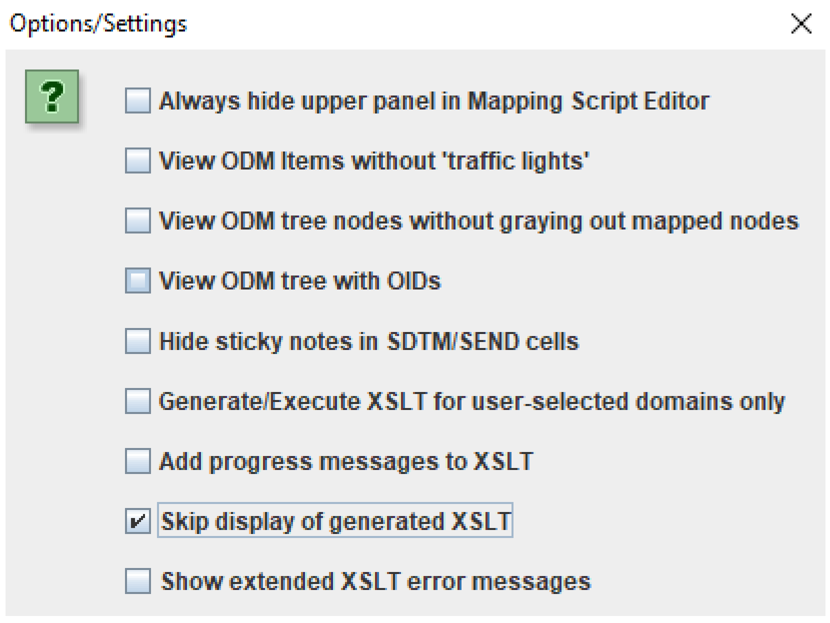
As one can see, this new feature can save enormous amounts of time when fine-tuning the define.xml.

Skipping display of the generated XSLT

When starting executing the transformation (menus "Transform - Generate Transformation (XSL) Code for SAS-XPT" and "Transform - Generate Transformation (XSL) Code for Dataset-XML" the system will generate XSLT (Extensible Stylesheet Language Transformations) code that is then applied to the ODM with clinical data. Until now, this transformation code was always displayed to the user, also allowing to save the XSLT transformation code to file, e.g. for batch processing. Some users however do not always want to see this code, and asked us to introduce a new feature to allow to skip the display of the XSLT transformation code and immediately proceed to the dialog "Execute Transformation Code".

This is now possible.

In order to skip the presentation of the generate XSLT, use the menu "Options - Setting" and look for the new item "Skip display of generated XSLT":



Options/Settings

- [ ] Always hide upper panel in Mapping Script Editor
- [ ] View ODM Items without 'traffic lights'
- [ ] View ODM tree nodes without graying out mapped nodes
- [ ] View ODM tree with OIDs
- [ ] Hide sticky notes in SDTM/SEND cells
- [ ] Generate/Execute XSLT for user-selected domains only
- [ ] Add progress messages to XSLT
- [✓] Skip display of generated XSLT
- [ ] Show extended XSLT error messages

When this checkbox is checked, and the transformation code is generated, the system will directly proceed with the dialog "Execute Transformation Code".

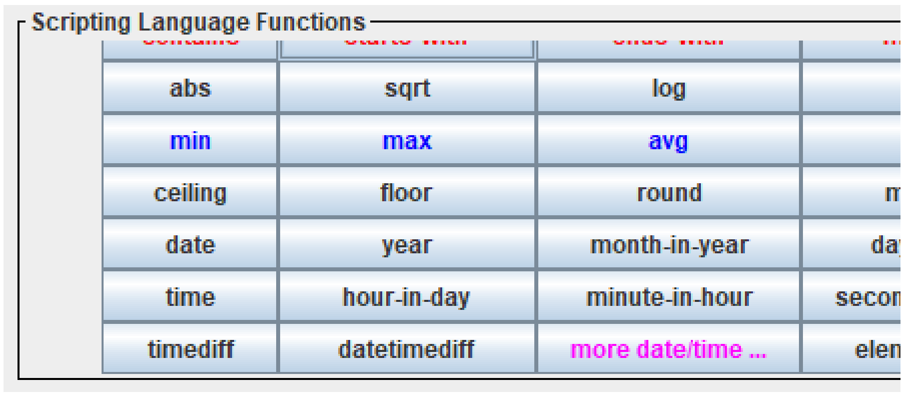
Additional date and time functions

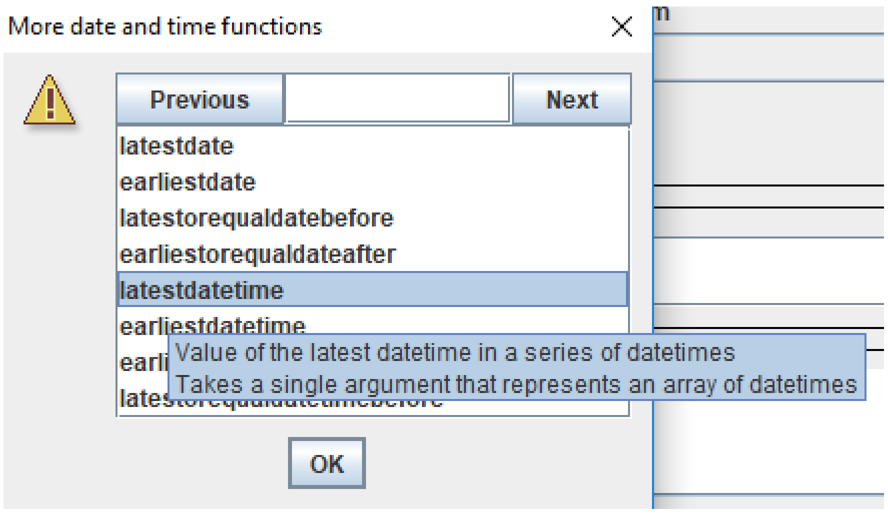
In SDTM, a lot of "timing variables" is present. About half of them are in principle unnecessary as they can easily be derived (also "on the fly"), and appear in the SDTM as redundant information (databases should not have any redundancy). It looks however as reviewers (or their tools) at the regulatory authorities are not capable of doing these "on the fly" derivations, which is the reason why they have been added.

To even make these derivations easier, a number of additional date and time functions have been added to SDTM-ETL v3.2. The complete list of all date and time functions (over 20 of them) is described in the document "Specification for the "SDTM -ETL" mini-scripting language". We limit ourselves here to the ones that are new of them in SDTM-ETL 3.2:

Function	Description	Example
latestdate()	Returns the latest date in an array or series (array variable) of dates	$d = xpath(….) ; # returns an array $lastdate = latestdate($d);
earliestdate()	Returns the earliest date in an array or series (array variable) of dates	$d = xpath(….) ; # returns an array $early = earliestdata($d);
latestorequaldatebefore()	Returns the latest date before or equal to a reference date. Very useful in combination with RFSTDTC, RFXSTDTC, RFENDTC, ...	$lastdatebeforereferencedata = latestorequaldatebefore($d, $DM.RFSTDTC);
earliestorequaldateafter()	Returns the earliest date after or equal to a reference date. Very useful in combination with RFSTDTC, RFXSTDTC, RFENDTC...	$firststudyaedate = earliestorequaldateafter($aedates, $DM.RFSTDTC);
latestdatetime()	Returns the latest datetime in an array or series (array variable) of datetimes	$dt = xpath(…); # array of datetimes $lastdatetime = latestdatetime($dt);
earliestdatetime()	Returns the earliest datetime in an array or series (array variable) of datetimes	# array of datetimes $dt = xpath(….) ; $early = earliestdatatime($dt);
earliestorequaldatetimeafter()	Returns the earliest datetime after or equal to a reference datetime. Very useful in combination with RFSTDTC, RFXSTDTC, RFENDTC...	$earliestdatetime = earliestorequaldatetimeafter($d, $DM.RFSTDTC);
latestorequaldatetimebefore()	Returns the latest datetime before or equal to a reference datetime. Very useful in combination with RFSTDTC, RFXSTDTC, RFENDTC...	$latestdatetime = latestorequaldatetimebefore($d, $DM.RFSTDTC)

These functions also appear when the user clicks the button “More date and time ...” in the lower part of the script editor:




More date and time functions

Previous		Next

latestdate
earliestdate
latestorequaldatebefore
earliestorequaldateafter
latestdatetime
earliestdatetime

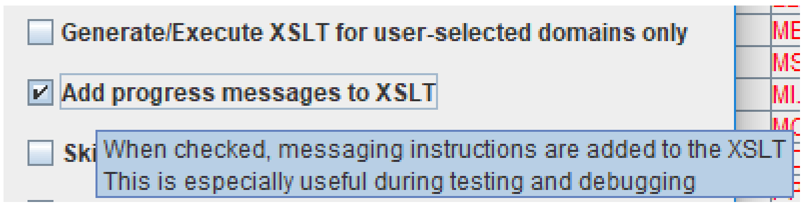
Value of the latest datetime in a series of datetimes
Takes a single argument that represents an array of datetimes

So one does not need to know the names of these functions (or have to look them up here), using the button and the subsequent choice lists allows the needed one to be copied to the script automatically.

Also remember than any XSLT2 date/time function (see e.g. http://www.xsltfunctions.com/xsl/c0002.html for a complete list) can always be used in our scripting language.

Progress messages

In the case of large input files, or complicated mappings, transformations can take some time. If the user wants to follow the progress of the transformation, he/she can choose to have intermediate messages shown during the transformation to see what the progress is. In order to do so, use "Options – Settings" and check the checkbox "Add progress messages to XSLT":

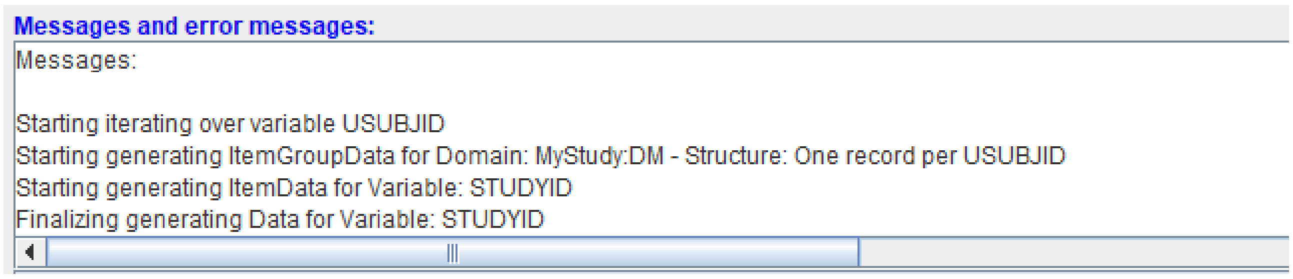


☐ Generate/Execute XSLT for user-selected domains only

☑ Add progress messages to XSLT

When checked, messaging instructions are added to the XSLT
This is especially useful during testing and debugging

The generated XSLT will then contain additional messaging statements that are executed during execution of the transformation. For example, one may find:



Messages and error messages:

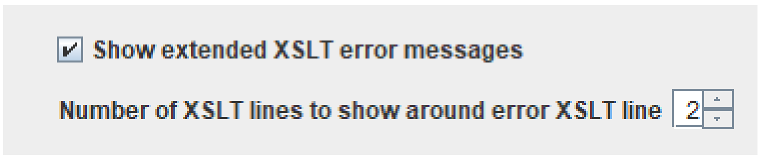
```
Messages:

Starting iterating over variable USUBJID
Starting generating ItemGroupData for Domain: MyStudy:DM - Structure: One record per USUBJID
Starting generating ItemData for Variable: STUDYID
Finalizing generating Data for Variable: STUDYID
```

This will e.g. also help when the transformation fails, as one can then see until what stage the transformation went well.

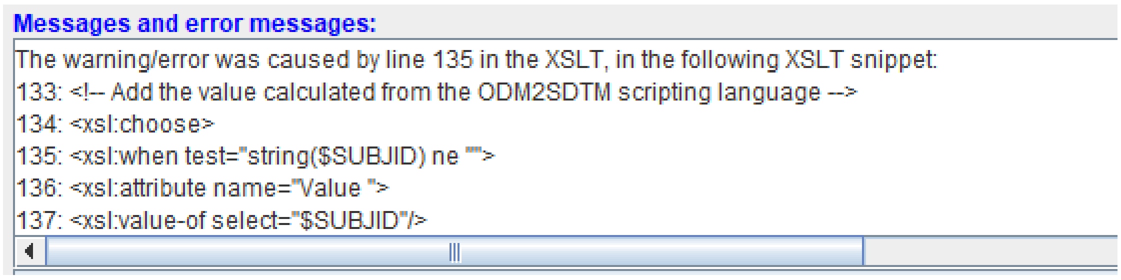
Extended messages for XSLT compilation and execution errors

Mapping scripts may fail during execution. It is always important to understand why they failed in order to then be able to correct the scripts (debugging).
In order to make debugging easier, an additional option has been added in SDTM-ETL v.3.2. You can switch it on using the menu "Options – Settings" and then check "Show extended XSLT error messages". A new field is then enabled allowing to define how many lines around the XSLT line that caused the error will be shown in the output:


☑ Show extended XSLT error messages

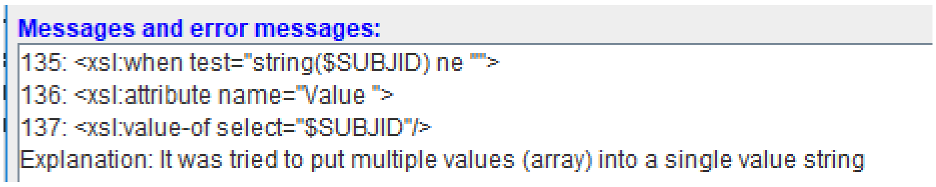
Number of XSLT lines to show around error XSLT line 2

If the value is set to "2", this means that the line causing the error will be displayed, together with 2 lines before and 2 lines after the line causing the error. For example, one may see the following in the output when execution failed:


Messages and error messages:

```
The warning/error was caused by line 135 in the XSLT, in the following XSLT snippet:
133: <!-- Add the value calculated from the ODM2SDTM scripting language -->
134: <xsl:choose>
135: <xsl:when test="string($SUBJID) ne ''">
136: <xsl:attribute name="Value ">
137: <xsl:value-of select="$SUBJID"/>
```

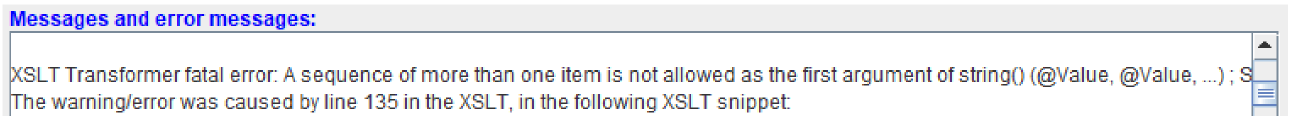
Line 135 caused the execution to fail, and lines 133-137 are displayed. In case the option "Show extended XSLT error messages" has been checked, the system will also not only display the error message, but also try to explain it. In our case, the following line is:


Messages and error messages:

```
135: <xsl:when test="string($SUBJID) ne ''">
136: <xsl:attribute name="Value ">
137: <xsl:value-of select="$SUBJID"/>
Explanation: It was tried to put multiple values (array) into a single value string
```

stating that $SUBJID is not a single value (as was expected), but an array of values. As this error occurred during the generation of the DM dataset, the probable reason is that the subject ID was taken from a form that was repeating during the visit from which the demographics data was retrieved.

The original XSLT message however does not well explain the error. It is a well known error for XSLT specialists, but not well suitable for normal users.


Messages and error messages:

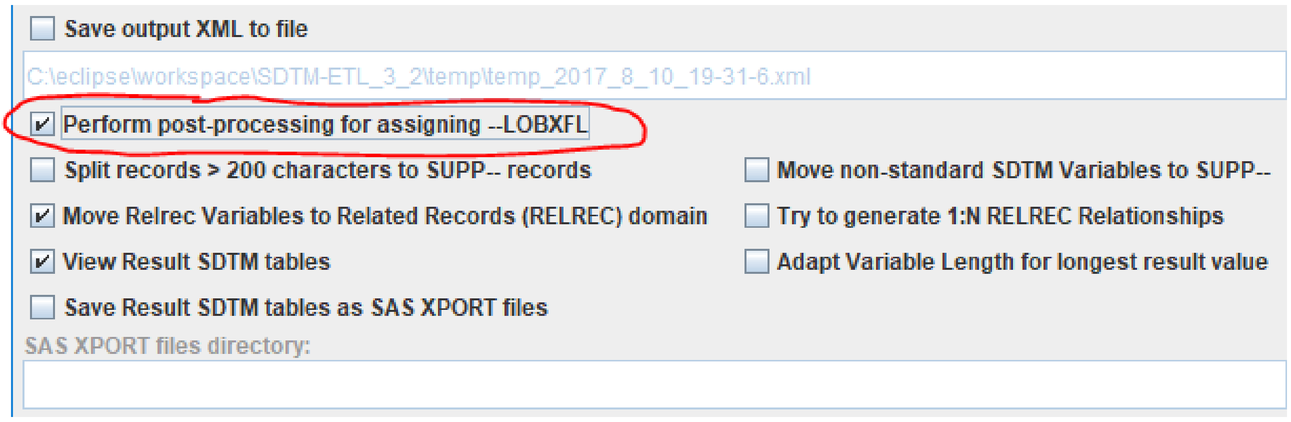
```
XSLT Transformer fatal error: A sequence of more than one item is not allowed as the first argument of string() (@Value, @Value, ...) ; S
The warning/error was caused by line 135 in the XSLT, in the following XSLT snippet:
```

Post-execution derivation of the new SDTM 1.5 –LOXBFL baseline flag

A new SDTM/SEND v.1.5 variable that has been added is the –LOXBFL “last observation before first exposure flag”. Again, this variable is essentially unnecessary, as any “smart” viewer tool such as the “Smart Dataset-XML Viewer” (also coming with the SDTM-ETL distribution) software can mark such baseline records “on the fly”, at least when observation dates and times and treatment exposure dates and times have been correctly (which unfortunately is not always the case).

When generating SDTM however, in some cases the derivation of –LOXBFL will require a postprocessing step, as all observation data need to be generated first before the system can decide which of them (by –TESTCD) is the last one before the first study treatment exposure.

When generating the SDTM datasets (using the menu “Transform – Generate transformation” followed by “Execute transformation”) one will also see that an extra checkbox has been added to the graphical user interface:



☐ Save output XML to file

C:\eclipse\workspace\SDTM-ETL_3_2\temp\temp_2017_8_10_19-31-6.xml

☑ Perform post-processing for assigning --LOBXFL

☐ Split records > 200 characters to SUPP-- records
☐ Move non-standard SDTM Variables to SUPP--
☑ Move Relrec Variables to Related Records (RELREC) domain
☐ Try to generate 1:N RELREC Relationships
☑ View Result SDTM tables
☐ Adapt Variable Length for longest result value
☐ Save Result SDTM tables as SAS XPORT files

SAS XPORT files directory:

In case the checkbox is checked, a post-processing step will be performed, sorting all observations and for each unique test code (based on –TESTCD), the last observation (based on --DTC) before the first exposure (based on DM-RFXSTDTC) will be marked, and the value for –LOBXFL set to “Y”.
Remark that this is a feature that is meant for the future, as –LOBXFL is a new variable for SDTM v.1.5, and there currently is no Implementation Guide for that version. In SEND-IG v.3.1, –LOBXFL is not used at all.

Also remark that although this feature is available, the user can still decide to write his own mapping scripts for –LOBXFL.