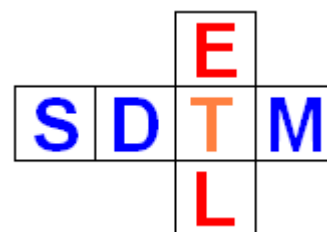


SDTM-ETL 4.1: Summary of New Features

Author: Jozef Aerts, XML4Pharma

Last update: 2022-10-11



Summary

This document contains a summary of the most important new features of SDTM-ETL 4.1. There are many minor improvements and new features that are not described in this document, but that can be found in other manuals / tutorials of SDTM-ETL 4.1.

Table of Contents

Summary	1
Table of Contents	1
Skipping ODM validation against the standard at ODM loading.....	1
SDTM/SEND datafiles export in CDISC Dataset-JSON format.....	2
SDTM/SEND datafiles export in CSV format.....	4
Generation of "merged" SAS Transport 5 files when having several instances of the same SDTM/SEND domain	5
Recalculation of ODM tree nodes.....	7
List of unmapped Items.....	7
Display of NCI codes for CDISC CodeLists	10
Trial Summary datasets - addition of CDISC-NCI codes in TS datasets	10
Generation of Define-XML CodeLists from Trial Design datasets	13
Addition of "Origin" to the Mapping Editor.....	23
Further improved "CodeList Mapping Wizard"	23
Encouraging the use of the "decode()" function	28
Parallel mapping script generation for as well --TESTCD as --TEST variables.....	33
Mapping Suggestions from SDTM annotations in the ODM file.....	39
Caching of LOINC code mappings.....	40
Adding new LOINC mappings to the local XML file with mappings in an automated way	45
Additional features for "Save cleaned define.xml"	48
Additional support for "Origin" for Define-XML 2.1	52
Extended support for SENDIG-DART 1.1	55
Extended support for the "Associated Persons" domain.....	56
Implementation of the "Metadata Submission Guide v.2.0"	57

Skipping ODM validation against the standard at ODM loading

Normally, when loading an ODM file, it is first checked (at least for the study design part) against the standard. This may be over-repetitive when working on a project, and loading the same ODM file each time.

This calculation step can now be skipped by editing the properties.dat file before starting the software, and setting the parameter "skipodmvalidation" to "false", i.e.:

```
*properties.dat - Editor
Datei Bearbeiten Format Ansicht Hilfe
language=en
languagefixed=true
# logfilepath=C:\temp
loglevel=DEBUG
sasviewerlocation=C:\Program Files\SAS Institute\SAS System \
adobereaderlocation="C:\Program Files\Adobe\Acrobat DC\Acroba
# CDISC Library API key
cdisclibraryapikey=
# other settings
advancedusage=true
skipodmvalidation=true
```

and replacing the word "true" by "false"

This is also interesting for OpenClinica users, as there is a discrepancy between the ODM that is exported and the XML-Schemas published by OpenClinica, leading to false positive validation errors.

SDTM/SEND datafiles export in CDISC Dataset-JSON format

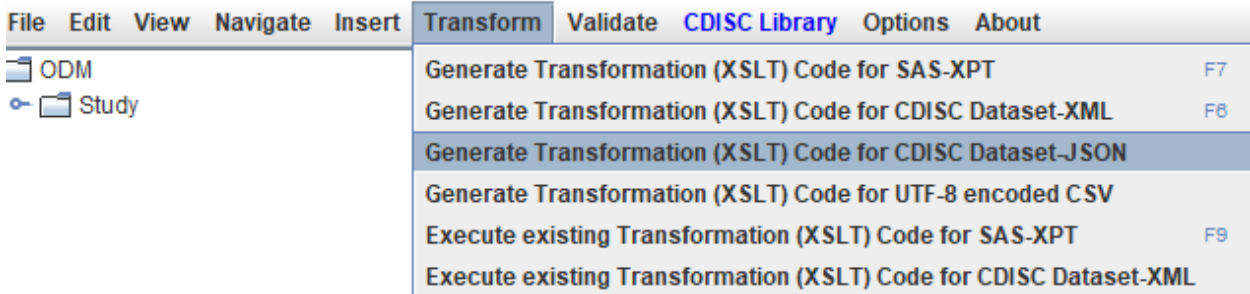
CDISC recently published a new format for SDTM/SEND/ADaM submissions to regulatory authorities: the [Dataset-JSON format](#). It is a perfect replacement for the completed outdated SAS Transport (5 or 8) format, due to:

- No more 8-, 40-, and 200-character limitations
- Full "out of the box" support for UTF-8 encoding, meaning that all written languages of the world are supported. This means that no "special tricks" are needed for incorporation of e.g. Chinese and Japanese characters.
- Considerable smaller file sizes than for SAS Transport 5 or 8.
- Ideal for the use with APIs and RESTful web services

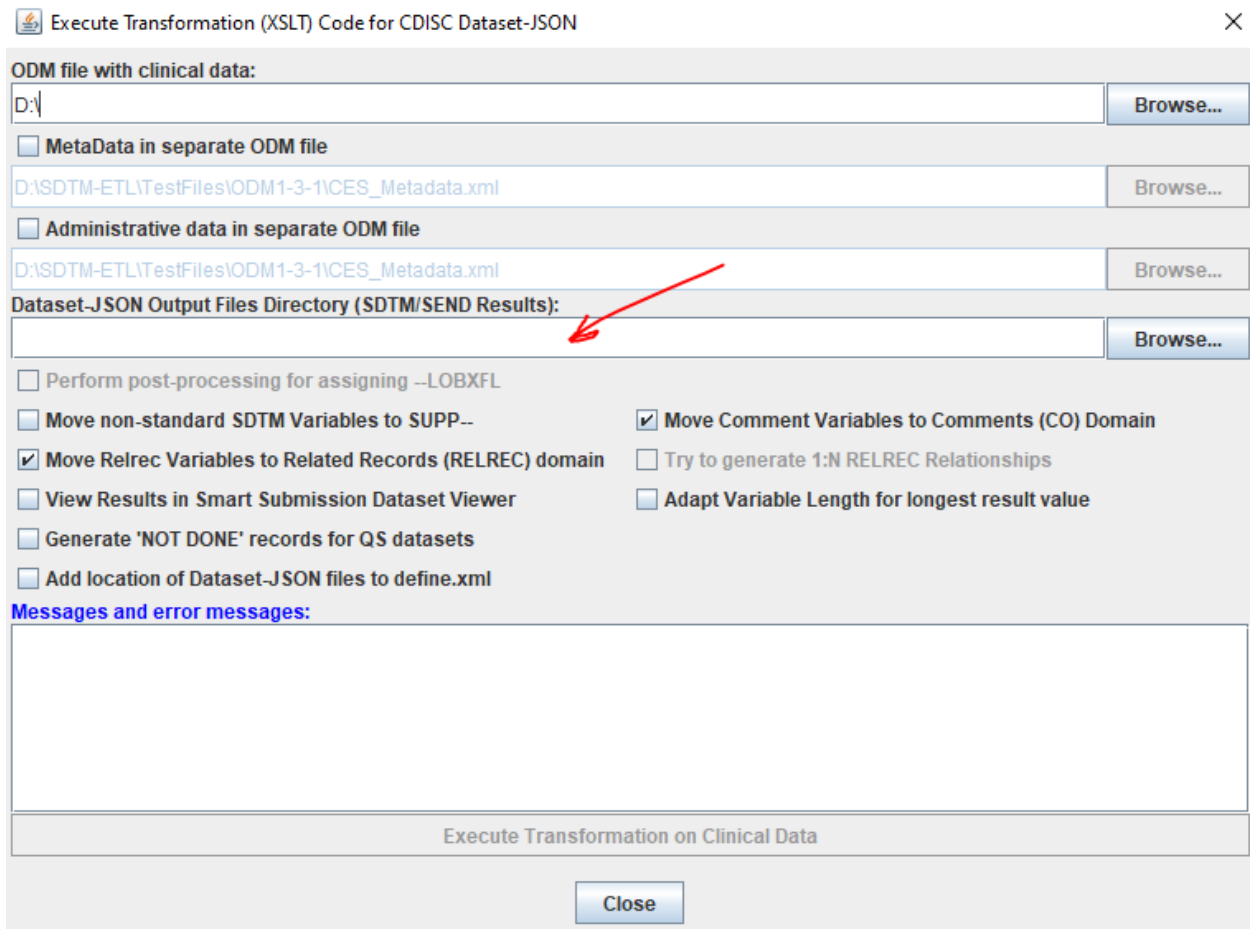
CDISC did already have a better (than SAS Transport) format for electronic submissions, the Dataset-XML format, but especially the FDA has been very reluctant to accept it, out of fear for large file sizes. This fear is unjustified, as the cost of storage nowadays is extremely small (less than 0.05 US\$ per Gigabyte).

With the new Dataset-JSON, there is however no excuse anymore to move to a modern transport format.

When having generated mappings, the submission Dataset-JSON files can be generated using the menu "Transform" followed by "Generate Transformation (XSLT) Code for CDISC Dataset-JSON":



The following screens are then identical as for any other transport format, and then leads to:



where then one only needs to provide the folder or directory where the Dataset-JSON files need to be written too. Also a "cleaned" define.xml file is then written to that folder, so that the set can immediately be used, and e.g. be inspected using the open source "[Smart Submission Dataset Viewer](#)". The latter can also be started automatically by checking the checkbox "View Results in Smart Submission Dataset Viewer".

Remark that there is essentially no need to check the checkbox "Adapt Variable Length for longest result value", as the JSON files generated are by nature, already optimized. The FDA requirement to "optimize for file size" is a typical issue when using SAS Transport (5 or 8) format only.

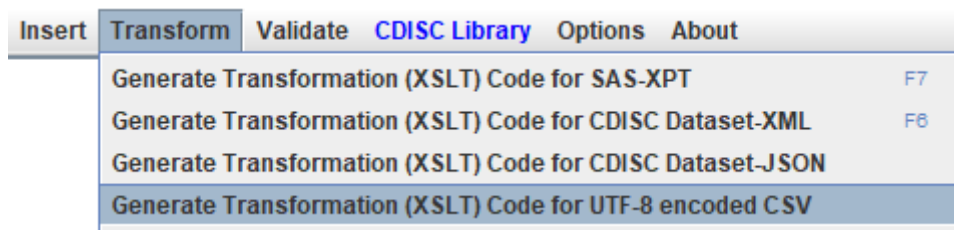
SDTM/SEND datafiles export in CSV format

In some cases, one will want to generate the SDTM datasets in simple CSV format. This is especially of importance when one e.g. want to populate a database, or e.g. when one want to import the generated datasets in Excel.

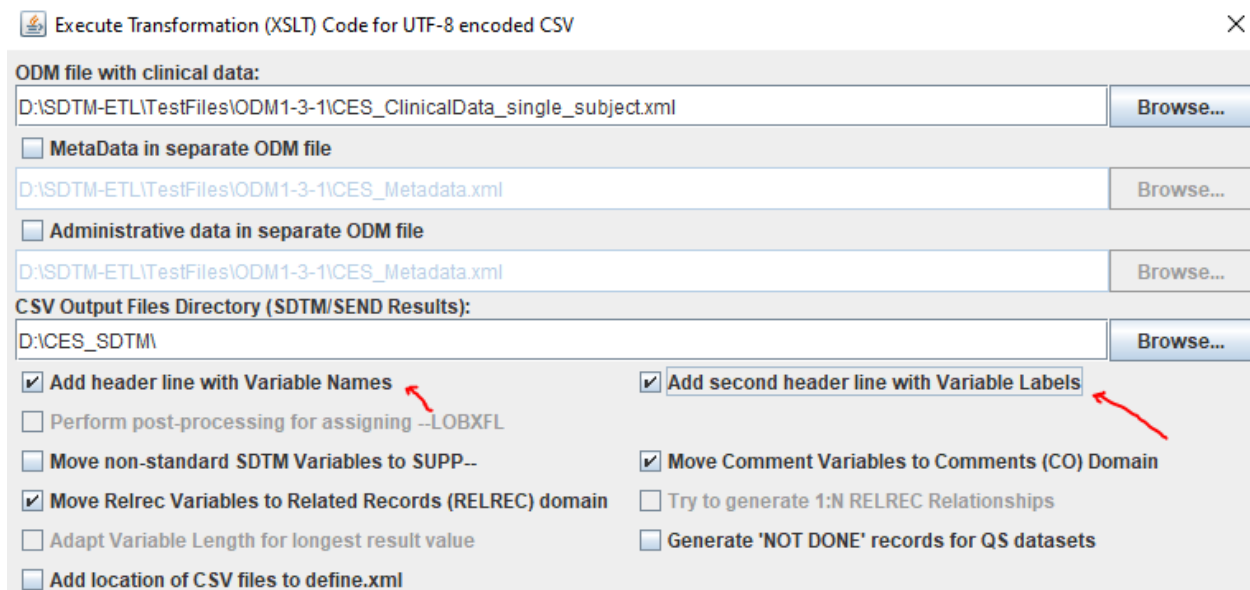
It is also of importance for research that is not submitted to regulatory authorities (who still require outdated SAS Transport format), such as in academic research.

Just like for the Dataset-XML and Dataset-JSON export, the CSV export has the advantage that it uses UTF-8 encoding (essentially meaning "Unicode"), meaning that also non-US-ASCII characters are fully supported. And of course that none of the character limitations (8-, 40-, 200-character limitations) is applicable.

In order to generate the datasets in UTF-8 encoded CSV format, use the menu "Transform - Generate Transformation (XSLT) Code for UTF-8 encoded CSV".



Besides the usual fields and choices, two new ones will appear, allowing you to have a first line in the CSV with the variable names and/or a second line with the variable labels:



The result will then be like:

```

^STUDYID,DOMAIN,USUBJID,VSEQ,VSTESTCD,VSTEST,VSORRES,VSORRESU,VSSSTRESC,VSSSTRESN,VSSSTRESU,VSBFL,VISITNUM,VSDTC
"Study Identifier","Domain Abbreviation","Unique Subject Identifier","Sequence Number","Vital Signs Test Short Name","Vital Signs Test Name","Res
MyStudy","VS","001","1","HEIGHT","Height","73","Inches","73","73","Inches","Y","1","2006-04-01"
MyStudy","VS","001","2","WEIGHT","Weight","204","Pound","204","204","Pound","Y","1","2006-04-01"
MyStudy","VS","002","1","HEIGHT","Height","164","Inches","164","164","Inches","Y","1","2006-04-02"
MyStudy","VS","002","2","WEIGHT","Weight","77","Kilogram","77","77","Kilogram","Y","1","2006-04-02"
MyStudy","VS","003","1","HEIGHT","Height","65","Inches","65","65","Inches","Y","1","2006-04-03"
MyStudy","VS","003","2","WEIGHT","Weight","122","Pound","122","122","Pound","Y","1","2006-04-03"
MyStudy","VS","004","1","HEIGHT","Height","69","Inches","69","69","Inches","Y","1","2006-04-04"
MyStudy","VS","004","2","WEIGHT","Weight","185","Pound","185","185","Pound","Y","1","2006-04-04"
MyStudy","VS","005","1","HEIGHT","Height","71","Inches","71","71","Inches","Y","1","2006-04-04"
MyStudy","VS","005","2","WEIGHT","Weight","244","Pound","244","244","Pound","Y","1","2006-04-04"
MyStudy","VS","006","1","HEIGHT","Height","71","Inches","71","71","Inches","Y","1","2006-04-06"
MyStudy","VS","006","2","WEIGHT","Weight","175","Pound","175","175","Pound","Y","1","2006-04-06"
MyStudy","VS","007","1","HEIGHT","Height","72","Inches","72","72","Inches","Y","1","2006-04-07"
MyStudy","VS","007","2","WEIGHT","Weight","168","Pound","168","168","Pound","Y","1","2006-04-07"
MyStudy","VS","008","1","HEIGHT","Height","62","Inches","62","62","Inches","Y","1","2006-04-08"
MyStudy","VS","008","2","WEIGHT","Weight","97","Pound","97","97","Pound","Y","1","2006-04-08"
MyStudy","VS","009","1","HEIGHT","Height","66","Inches","66","66","Inches","Y","1","2006-04-09"
MyStudy","VS","009","2","WEIGHT","Weight","171","Pound","171","171","Pound","Y","1","2006-04-09"
MyStudy","VS","010","1","HEIGHT","Height","69","Inches","69","69","Inches","Y","1","2006-04-10"
MyStudy","VS","010","2","WEIGHT","Weight","80","Kilogram","80","80","Kilogram","Y","1","2006-04-10"
MyStudy","VS","011","1","HEIGHT","Height","61","Inches","61","61","Inches","Y","1","2006-04-11"
MyStudy","VS","011","2","WEIGHT","Weight","114","Pound","114","114","Pound","Y","1","2006-04-11"
MyStudy","VS","012","1","HEIGHT","Height","66","Inches","66","66","Inches","Y","1","2006-04-12"
MyStudy","VS","012","2","WEIGHT","Weight","193","Pound","193","193","Pound","Y","1","2006-04-12"

```

Generation of "merged" SAS Transport 5 files when having several instances of the same SDTM/SEND domain

The FDA speaks about "split datasets", but this designation is completely wrong, as, when one is smart, one never "splits" datasets, but one generates different instances of the same domain leading to more than one dataset for that domain.

FDA also often requires that as well a single dataset as the set of "split" datasets is delivered. This is a bit strange, as they state that they cannot treat datasets that are very large in file size, and needed to be "split", so why do they then also ask for them? By the way, the reason for these very large file sizes is SAS Transport, which is very inefficient, but which is ... still mandated by the FDA ...

The better strategy is of course to generate separate datasets for a specific domain if there is fear that the (by XPT format caused!) file sizes will be too large. This is usually done on base of the xxCAT variable (and when necessary also on base of xxSCAT). For QS, it also makes sense to generate different datasets based on the type of questionnaire, not only for reasons of "review friendliness", but also as this makes mapping considerably easy.

Anyway, in case it is required, it should also be possible to merge separate datasets for a single domain into a single dataset. This feature has now been implemented in SDTM-ETL.

When generating the SAS Transport 5 files, the system will check whether there is more than one dataset to be generated for each single domain, and if so, an additional checkbox "Additionally generate a merged dataset for 'split domain' datasets" will be made available:

Execute Transformation (XSLT) Code for SAS-XPT

ODM file with clinical data:
 D:\SDTM-ETL\TestFiles\MyODM.xml Browse...

MetaData in separate ODM file
Browse...

Administrative data in separate ODM file
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 Comment Variables to Comments (CO) Domain

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

Generate 'NOT DONE' records for QS datasets

Save Result SDTM tables as SAS XPORT files

SAS XPORT files directory:
 D:\temp Browse...




Add location of SAS XPORT files to define.xml Store link as relative path

Additionally generate a merged dataset for 'split' domain datasets ←

Messages and error messages:

Execute Transformation on Clinical Data

When checked, an additional dataset is being generated with the suffix "_MERG.xpt". For example, when there are 2 instances of the VS domain, one with "normal" (classic) vital signs data, named VSNORM, and another one with oxygen saturation data, named VSOXY, then also a dataset VS_MERG will be generated, i.e. as a file "VS_MERG.xpt":

Name	Änderungsdatum	Typ
 VS_MERG.xpt	30.05.2022 20:00	SAS Xport Transpo...
 VSNORM.xpt	30.05.2022 20:00	SAS Xport Transpo...
 VSOXY.xpt	30.05.2022 20:00	SAS Xport Transpo...

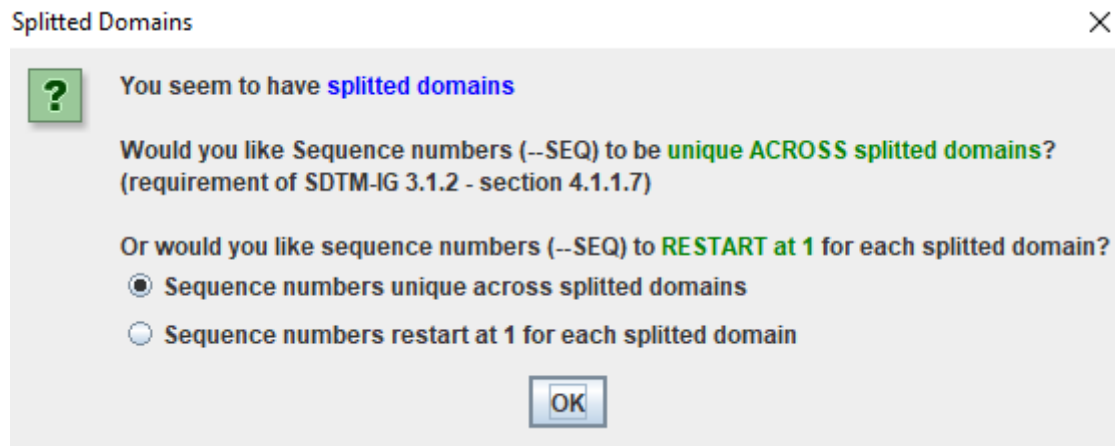
P.S.: this still need to be reflected in a "cleaned" define.xml file when using the menu "File - Save cleaned define.xml"

Remark that it is important that both source datasets have the same properties for all the variables used. This is however usually already guaranteed by SDTM-ETL itself, as the properties of the variables reside in define.xml "ItemDef" elements, and in SDTM-ETL, each "ItemDef" for a domain variable is shared between instances of the same domain. However, it also means that when a variable is used (i.e. a mapping is present) in one instance of the domain, it must also be present in the other instances of the domain. For example, if we populated VSSCAT (subcategory) in VSOXY, we must also populate it in VSNORM. If there is no value for it, one can simply set the mapping to be empty.

For our VSNORM instance this would mean that the mapping for VSSCAT simply is:

```
$VS.VSSCAT = "";
```

Also take care to choose for "Sequence numbers unique across split domains" is checked when running the process. This is however already the default.



Recalculation of ODM tree nodes

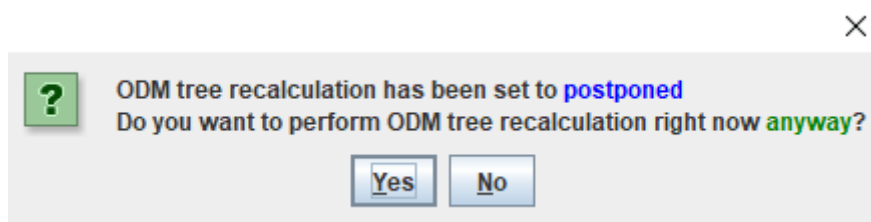
Before v.4.1, there was already an option "Allow postponing ODM tree node usage recalculation", taking care that coloring and "greying out" of ODM tree nodes is not performed each time a new mapping has been defined or the properties of an SDTM/SEND variable is changed.

This option can be set/unset using the menu "Options - Settings".

When "postponing" was selected, it did however not apply to when a new define.xml is loaded, which makes sense.

As some users mentioned that such a "recalculation" can take considerable time when a new define.xml is loaded, they asked to also apply the "postponing" to the case that a new define.xml is loaded.

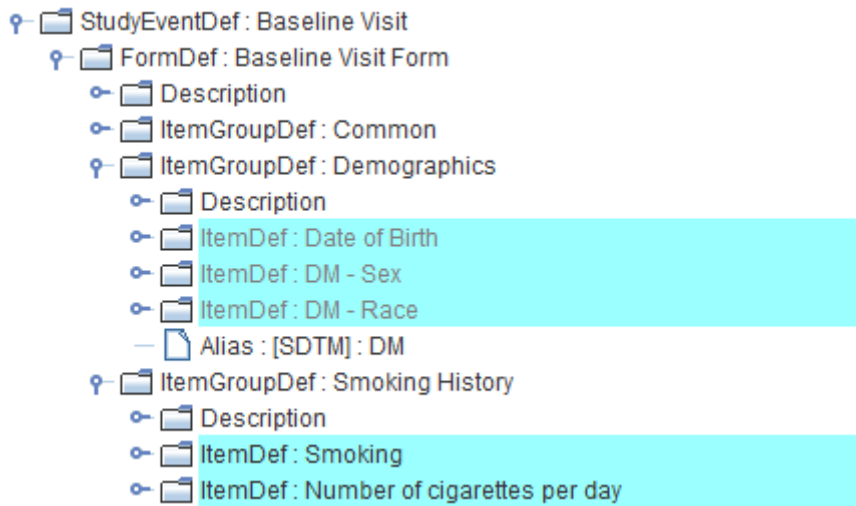
This has now been implemented, thus speeding up the loading of the define.xml. When the define.xml is then loaded, a message will then appear asking whether recalculation of the ODM tree nodes should then be performed anyway:



If the user then selects "Yes", ODM tree recalculation will be performed, and will be skipped otherwise.

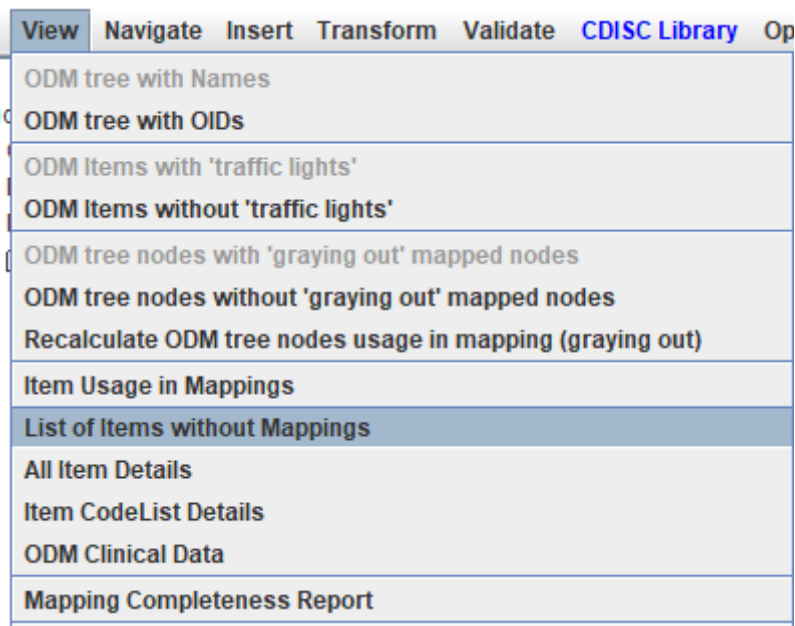
List of unmapped Items

SDTM-ETL already provides an indication of which of the ODM items is directly used in mappings by "greying" these in the ODM tree. For example:



One sees that the items "Date of Birth", "Sex", and "Race" are "greyed", i.e. indicating that they have been used in mappings, whereas the items "Smoking" and "Number of cigarettes per day" is still in the normal color (black), meaning that these have not been used in mappings (yet).

New in SDTM-ETL 4.1 is that one can also obtain a list of items that have not been used in direct mappings, using the menu "View - List of Items without Mappings":



When used, the list is then displayed in a separate dialog. For example:



Below is a list of items for which there is no direct mapping to SDTM. The occurrence of a collected item in this list does **not necessarily** mean that the mapping is incorrect or incomplete. The list can however be used to check which items still require mapping to SDTM.

- I_SUBJECTID - Subject ID
- I_VISITTIME - Visit Time
- I_SMOKING - Smoking
- I_NR_CIGARETTES - Number of cigarettes per day
- I_DRINKING - Drinking
- I_DIZZY - Dizziness at low DBP
- I_XRAY - Link to XRay
- I_BREATHING - Breath complaints
- I_COUGHING - Coughing problems
- I_HEART_COMPLAINTS - Hart complaints
- I_ILNESS_DAYS_LAST_YEAR - No. of illness days last year
- I_BRONCHITS - No. of bronchitis illnesses
- I_PNEUMONIA - No. of pneumonia illnesses
- I_SITE - Site Number
- I_VISIT - Visit Date
- I_CM_TAKEN - Non-study Medications taken
- I_CM_NUMBER - Medication Number
- I_CM_NAME - Medication Name
- I_CM_DOSE - Medication Dose
- I_CM_DOSEUNIT - Medication Dose Units



Occurrence of an item in this list does not necessarily mean that the mapping is incorrect or incomplete. For example, the item "Subject ID" appears in the list, as the "SubjectKey" in the ODM was used instead of the CRF field "Subject ID". As all the data in the ODM export is however organized per subject, identified by the "SubjectKey", both will however usually be identical. Also, one may have items like "did any adverse event occur", which are usually not mapped to SDTM at all.

Also remark that there is also an even more extended way to see which ODM items were used in mappings, and which not, by using the menu "View - Mapping Completeness Report":

Mapping Report

Mapping information is supplied for each ODM Item for which the value of the **Value** attribute has been used in a mapping

StudyEvent OID (Name)	Form OID (Name)	ItemGroup OID (Name)	Item OID (Name)	(SDTM) Variable OID	Mapping script
BASELINE (Baseline Visit)	F_BASELINE (Baseline Visit Form)	IG_COMMON (Common)	I_SITE (Site Number)	DM.SITEID	# Mapping using ODM element ItemData with ItemOID I_SITE SDM.SITEID = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupData[@ItemGroupOID='IG_COMMON']/ItemData
BASELINE (Baseline Visit)	F_BASELINE (Baseline Visit Form)	IG_COMMON (Common)	I_SUBJECTID (Subject ID)		
BASELINE (Baseline Visit)	F_BASELINE (Baseline Visit Form)	IG_COMMON (Common)	I_VISIT (Visit Date)	DM.RFSTDTC	# Mapping using ODM element ItemData with ItemOID I_VISIT SDM.RFSTDTC = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupData[@ItemGroupOID='IG_COMMON']/ItemData
BASELINE (Baseline Visit)	F_BASELINE (Baseline Visit Form)	IG_COMMON (Common)	I_VISITTIME (Visit Time)		
BASELINE (Baseline Visit)	F_BASELINE (Baseline Visit Form)	IG_DM (Demographics)	I_BRTHDT (Date of Birth)	DM.BRTHDTC	# Mapping using ODM element ItemData with ItemOID I_BRTHDT SDM.BRTHDTC = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupData[@ItemGroupOID='IG_DM']/ItemData[@I
					# Mapping using ODM element ItemData with ItemOID I_SEX

As this report can be saved to file, it is very useful for mapping discussions with other people.

Display of NCI codes for CDISC CodeLists

More and more, the CDISC-NCI codes ("C-codes") are used as the real identifier for items in by CDISC codelists. When using the menu "View - SDTM associated CodeList" (or "SEND associated CodeList" of course), the CDISC-NCI codes are now also displayed in square brackets. For example:

Details for CodeList: Laboratory Test Code (OID: CL.C65047.LBTTESTCD) ×

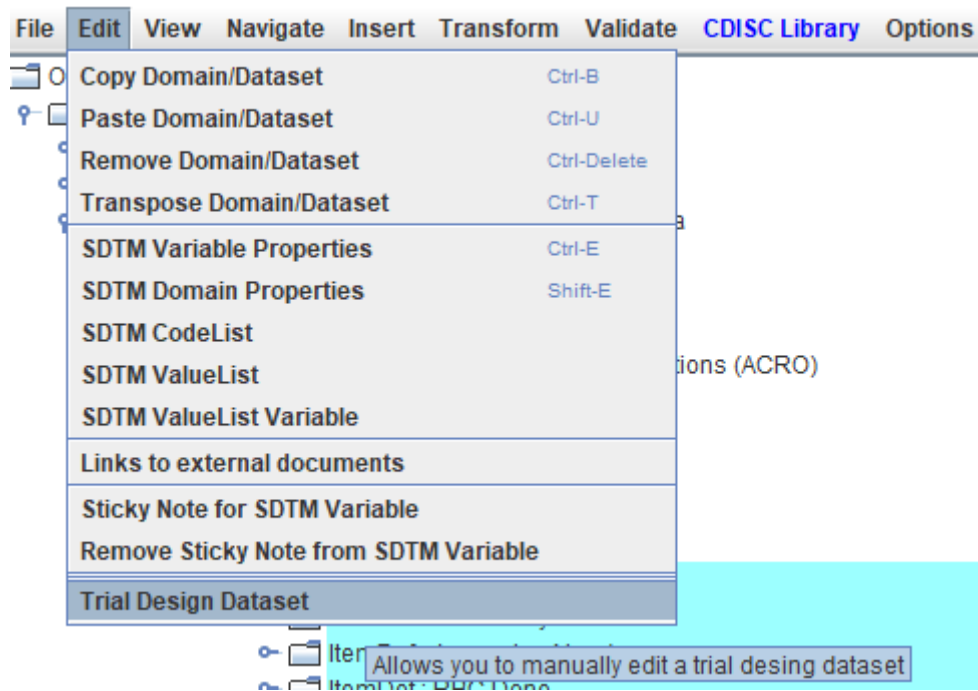
Coded Value	Language	Decoded Text
A1AGLP [C65047]		Alpha-1 Acid Glycoprotein Meas
A1ANTRYP [C65047]		Alpha-1 Antitrypsin Measureme
A1MCREAT [C65047]		Alpha-1 Microglobulin to Creatin
A1MICG [C65047]		Alpha-1 Microglobulin Measure
A2MACG [C65047]		Alpha-2 Macroglobulin Measure
AAP [C65047]		Alanine Aminopeptidase Measu
ABNCE [C65047]		Abnormal Cell Count
ABO [C65047]		ABO Blood Group Determination
ABO A1 [C65047]		ABO A1 Subtype Determination
ACANT [C65047]		Acanthocyte Count
ACANTRBC [C65047]		Acanthocyte to Erythrocyte Rat
ACE [C65047]		Angiotensin Converting Enzyme
ACETAMIN [C65047]		Acetaminophen Measurement

Max. Length for CodedValue: 8

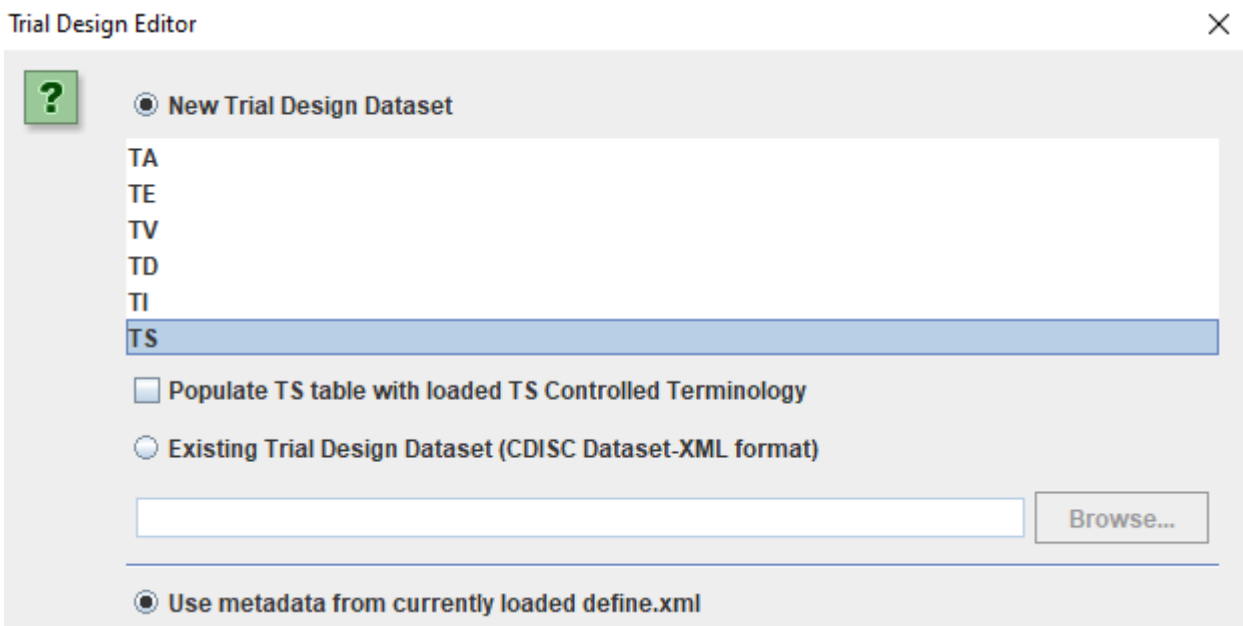
Trial Summary datasets - addition of CDISC-NCI codes in TS datasets

The TS dataset has some newer variables allowing to add codings for the terms in the trials summary parameters. These can now semi-automatically be retrieved.

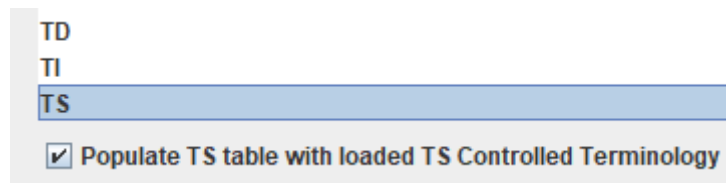
An initial instance of the TS dataset definition can be created using the menu "Edit - Trial Design Dataset:



followed by selecting "TS" and requesting to create a new one:



In most cases, you will want to start from controlled terminology that has been published for TS. If so, also check the checkbox "Populate TS table with loaded TS Controlled Terminology":



After clicking "OK", first an information message and then the following table appears in a new window:

STUDYID	DOMAIN	TSSEQ	TSGRPID	TSPARMCD	TSPARM	TSVAL	TSVALNF	TSVALCD	TSVCDREF	TSVCDVER
CES	TS	1		ACTSUB	Actual Number of Subjects					
CES	TS	2		ADAPT	Adaptive Design					
CES	TS	3		ADDON	Added on to Existing Treatments					
CES	TS	4		AGEMAX	Planned Maximum Age of Subjects					
CES	TS	5		AGEMIN	Planned Minimum Age of Subjects					
CES	TS	6		BIOSPRET	Biospecimen Retention Contains DNA					
CES	TS	7		CITNSTDY	Citation Used in Study					
CES	TS	8		CMSPSTAT	Commercial Sponsor Status					
CES	TS	9		COMPTRT	Comparative Treatment Name					
CES	TS	10		CONEMAIL	Contact E-Mail Address					
CES	TS	11		CONMAIL	Contact Mailing Address					
CES	TS	12		CONNAME	Contact Name					
CES	TS	13		CONPHONE	Contact Phone Number					
CES	TS	14		CONROLE	Contact Role					
CES	TS	15		CRMDUR	Confirmed Response Minimum Duration					
CES	TS	16		CSRARDTC	Clinical Study Report Archive Date					
CES	TS	17		CTAUG	CDISC Therapeutic Area User Guide					
CES	TS	18		CURTRT	Current Therapy or Treatment					
CES	TS	19		DCUTDESC	Data Cutoff Description					
CES	TS	20		DCUTDTC	Data Cutoff Date					
CES	TS	21		DGFCRIT	Delayed Graft Function Dx Criteria					
CES	TS	22		DMCIND	Data Monitoring Committee Indicator					
CES	TS	23		DOSE	Dose per Administration					
CES	TS	24		DOSFRM	Dose Form					
CES	TS	25		DOSFRQ	Dosing Frequency					

Update variables for maximal length in define.xml when writing to file

One can now start removing the rows (i.e. trial parameters) that one does not need, and also add new rows e.g. for parameters for which no controlled terminology has been published. We then also start adding values. For example:

STUDYID	DOMAIN	TSSEQ	TSGRPID	TSPARMCD	TSPARM	TSVAL
CES	TS	1		ACTSUB	Actual Number of Subjects	100
CES	TS	2		ADAPT	Adaptive Design	
CES	TS	3		ADDON	Added on to Existing Treatments	
CES	TS	4		AGEMAX	Planned Maximum Age of Subjects	

For the next row "Adaptive Design", there is CDISC controlled terminology for TSVAL, so when we right-click the TSVAL field for "ADAPT", we get:

TSPARM	TSVAL	TSVALNF
Actual Number of Subjects	100	
Adaptive Design		
Added on to Existing Treatments		
Planned Maximum Age of Subjects		
Planned Minimum Age of Subjects		
Biospecimen Retention Contains DNA		
Citation Used in Study		
Commercial Sponsor Status		
Comparative Treatment Name		
Contact E-Mail Address		
Contact Mailing Address		
Contact Name		
Contact Phone Number		
Contact Role		
Confirmed Response Minimum Duration		
Clinical Study Report Archive Date		
CDISC Therapeutic Area User Guide		
Current Therapy or Treatment		
Data Cutoff Description		
Data Cutoff Date		
Delayed Graft Function Dx Criteria		

Select a coded value X

?

N
U
Y

and can choose a value, e.g. "N" (No).

This all was already implemented in SDTM-ETL 4.0.

What is new in v.4.1 is that after a (coded) value has been selected, the following dialog is displayed:

TSPARM	TSVAL	TSVALNF
Actual Number of Subjects	100	
Adaptive Design	N	
Added on to Existing Treatments		
Planned Maximum Age of Subjects		
Planned Minimum Age of Subjects		
Biospecimen Retention Contains DNA		
Citation Used in Study		
Commercial Sponsor Status		
Comparative Treatment Name		
Contact E-Mail Address		
Contact Mailing Address		
Contact Name		
Contact Phone Number		

X

? Do you want me to populate the columns TSVLCD (Parameter Value Code), TSVCDREF (Name of the Reference Terminology) and TSVCDVER (Version of the Reference Terminology) with the values 'C49487', 'CDISC' and '2022-03-25'?

asking whether also TSVLCD, TSVCDREF and TSVCDVER need to be automatically populated. If one clicks "Yes", the result in this case is:

TSPARMCD	TSPARM	TSVAL	TSVALNF	TSVLCD	TSVCDREF	TSVCDVER
ACTSUB	Actual Number of Subjects	100				
ADAPT	Adaptive Design	N		C49487	CDISC	2022-03-25
ADDON	Added on to Existing Treatments					
AGEMAX	Planned Maximum Age of Subjects					
AGFMIN	Planned Minimum Age of Subjects					

adding the CDISC-NCI code C49487 (for "No") in the coding system "CDISC" of version "2022-03-25", the codelist version that was selected when the SDTMIG template was loaded.

Remark that this only works for CDISC code, i.e. it doesn't work for e.g. SNOMED-CT codes. For the latter, one will still need to fill in the values for TSVLCD, TSVCDREF and TSVCDVER manually.

Generation of Define-XML CodeLists from Trial Design datasets

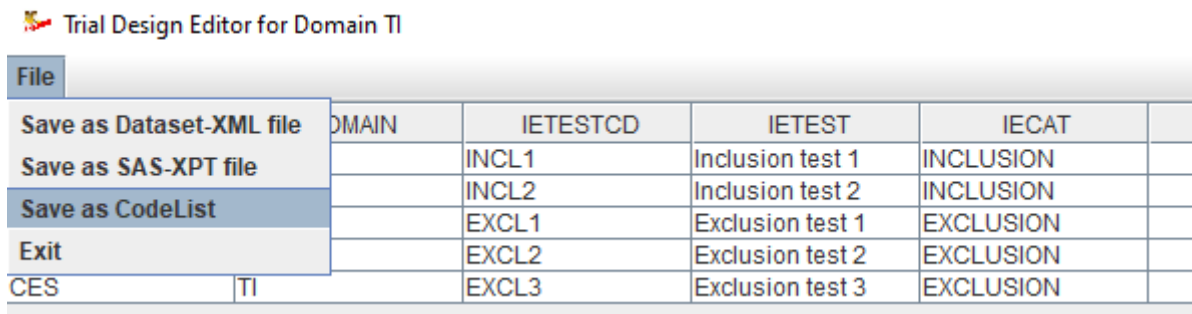
Before version 4.1, it was already possible to set up Trial Design datasets using the "Trial Design Editor" by using the menu "Edit - Trial Design Dataset" and to then export to either a SAS Transport 5 (XPT) file or to a more modern CDISC Dataset-XML file. The latter also allows to read in the dataset again and make changes or additions when necessary.

New in version 4.1 is that one can also generate an ODM/Define-XML codelist that can then be used for the corresponding variable (often a -TESTCD/-TEST type variable) in a non-trial-design dataset. This applies to the following Trial Design datasets:

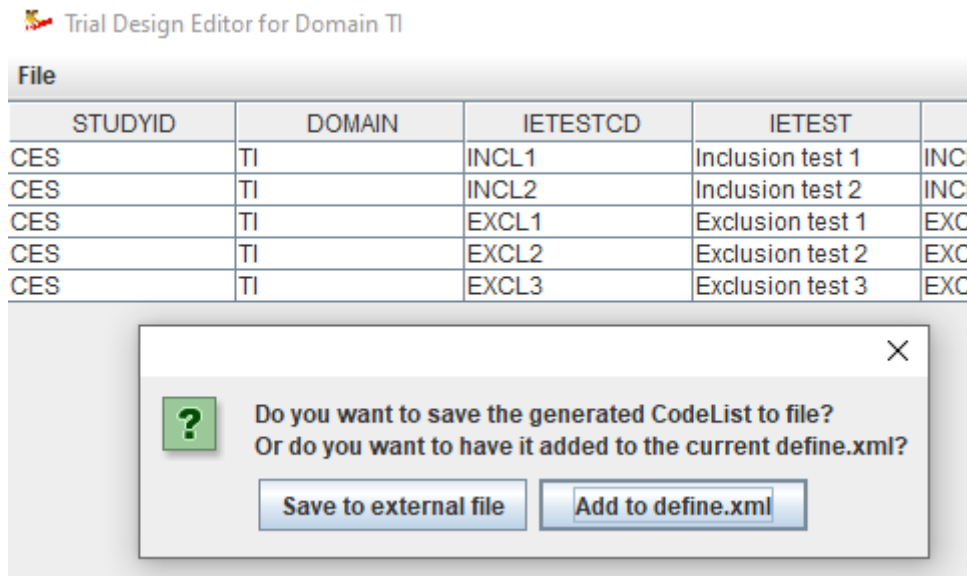
Trial Design Domain	Code	Decode	Used in Domain	Domain variables
TA	ARMCD	ARM	DM	ARMCD, ARM, ACTARMCD, ACTARM
TE	ETCD	ELEMENT	SE	ETCD, ELEMENT
TI	IETESTCD	IETEST	IE	IETESTCD, IETEST

TV	VISITNUM	VISIT	several domains	VISITNUM, VISIT
TS	TSPARMCD	TSPARM		
TX (SEND)	SETCD	SET	several domains	
TP (SEND-DART)	RPATHCD	RPATH	DM	RPATHCD
TT (SEND-DART)	RSTCGD	RSTAGE	TP, SJ	RSTCGD, RSTAGE

For example, for TI (Trial Inclusion/Exclusion Criteria), such a codelist (to be used in domain IE - Inclusion/Exclusion Criteria Not Met), such a can be generated by, within the "Trial Design Editor", using the menu "File - Save as CodeList"

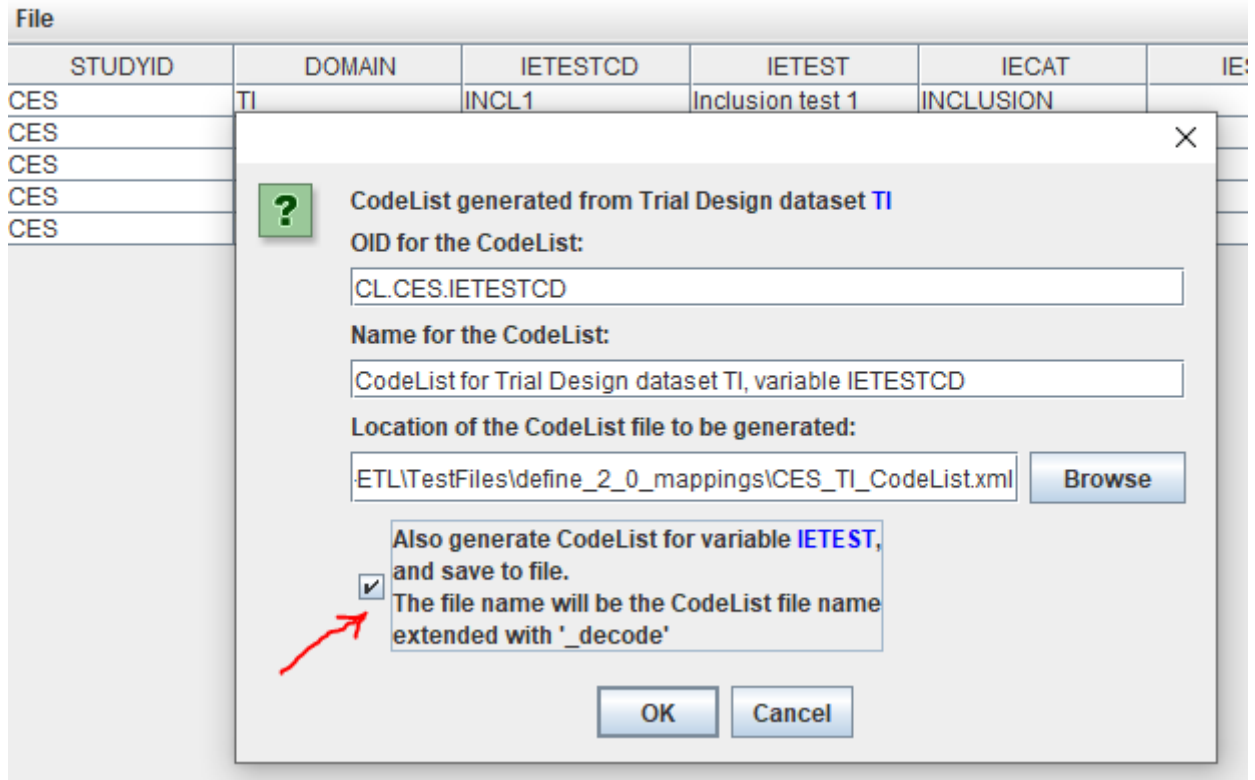


The user is then asked whether the generated codelist should be stored as an ODM-XML file, or directly added to the define.xml (when loaded):



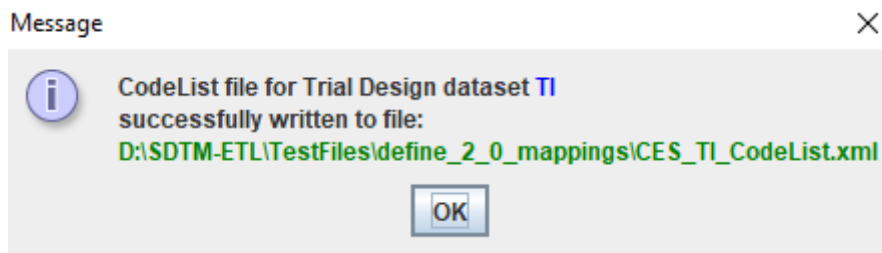
In case the user chooses for "Save to external file":

A new wizard dialog is then displayed, already providing a proposal for assigning the OID and Name of the codelist. The user than only needs to provide a location where the generated codelist must be stored. For example:



Checking the checkbox "Also generate CodeList for variable xxYYYY" (where "xxYYYY" can e.g. be IETEST, VISIT (for TV), ELEMENT (for TE)) allow to also generate a codelist with the "decoded" values only, in addition with the codelist with as well coded as decoded values¹.

When everything works fine, a message is displayed:



and when also the checkbox "Also generate Codelist for ..." was checked, and additional message is shown, e.g.:



When the generated codelist cannot be stored to the provided location (for example when the given

¹ This is due to SDTM/SEND wanting to have a separate codelist for the "decode" of each "code", like in the IETESTCD / IETEST pair.

drive does not exist, or the folder is write-protected), an error message is shown.

The generated codelist XML then looks like:

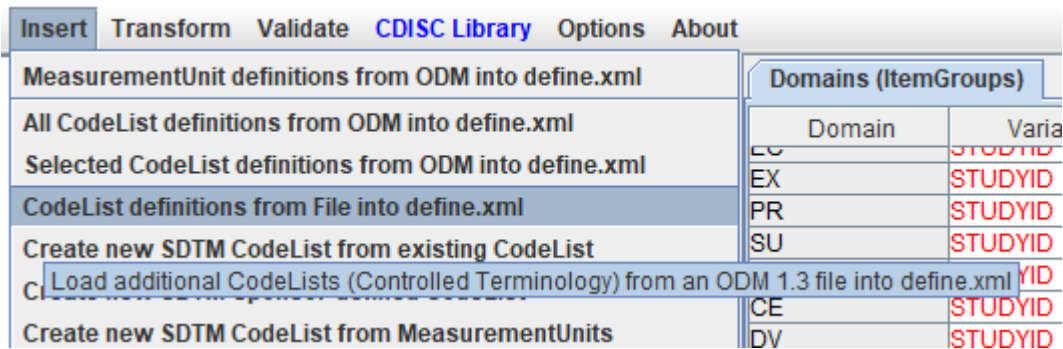
```
1 <CodeList xmlns="http://www.cdisc.org/ns/odm/v1.3" OID="CL.CES.IETESTCD"  
2   Name="CodeList for Trial Design dataset TI, variable IETESTCD" DataType="text">  
3   <CodeListItem CodedValue="INCL1">  
4     <Decode>  
5       <TranslatedText>Inclusion test 1</TranslatedText>  
6     </Decode>  
7   </CodeListItem>  
8   <CodeListItem CodedValue="INCL2">  
9     <Decode>  
10      <TranslatedText>Inclusion test 2</TranslatedText>  
11    </Decode>  
12  </CodeListItem>  
13  <CodeListItem CodedValue="EXCL1">  
14    <Decode>  
15      <TranslatedText>Exclusion test 1</TranslatedText>  
16    </Decode>  
17  </CodeListItem>  
18  <CodeListItem CodedValue="EXCL2">  
19    <Decode>  
20      <TranslatedText>Exclusion test 2</TranslatedText>  
21    </Decode>  
22  </CodeListItem>  
23  <CodeListItem CodedValue="EXCL3">  
24    <Decode>  
25      <TranslatedText>Exclusion test 3</TranslatedText>  
26    </Decode>  
27  </CodeListItem>  
28 </CodeList>
```

and the generated "decode" codelist XML like:

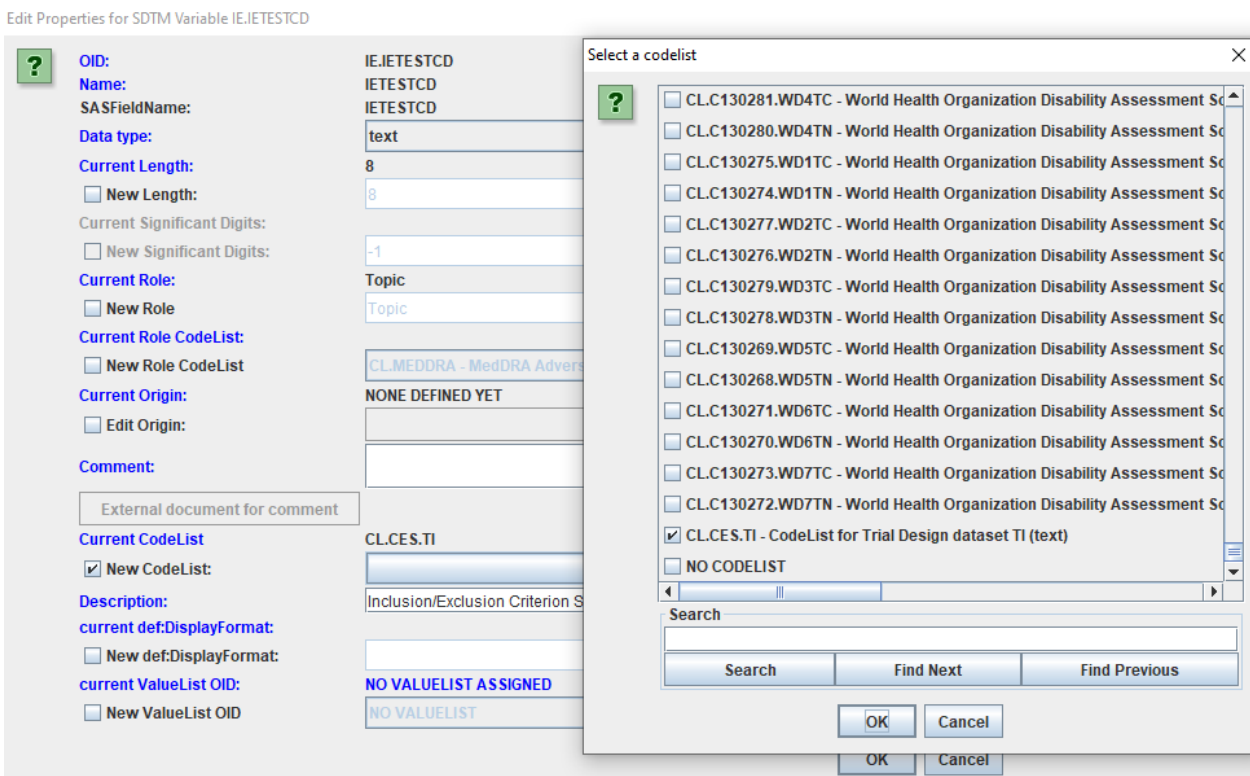
```
1 <CodeList xmlns="http://www.cdisc.org/ns/odm/v1.3" OID="CL.CES.IETESTCD.DECODE"  
2   Name="CodeList for Trial Design dataset TI, variable IETESTCD - decoded values" DataType="text">  
3   <EnumeratedItem CodedValue="Inclusion test 1"/>  
4   <EnumeratedItem CodedValue="Inclusion test 2"/>  
5   <EnumeratedItem CodedValue="Exclusion test 1"/>  
6   <EnumeratedItem CodedValue="Exclusion test 2"/>  
7   <EnumeratedItem CodedValue="Exclusion test 3"/>  
8 </CodeList>  
-
```

Having it stored locally has the advantage that it can still be edited, e.g. adding decodes for additional languages (e.g. for non-FDA regulatory authorities).

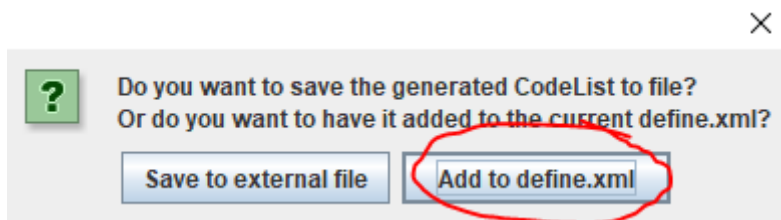
The codelist can then be imported and assigned to a variable (in this case to IETESTCD in the IE domain) by using the menu "Insert - CodeList definitions from File into define.xml":



and then assigned to e.g. IETESTCD by using the menu "Edit - SDTM Variable Properties" and assigning the codelist using the "New Codelist" checkbox and "Select Codelist" button:



A more direct way is to directly incorporate the generated codelist into the define.xml, and assign it to the corresponding variable:



When "Add to define.xml" is selected, the following dialog is displayed:

STUDYID	DOMAIN	IETESTCD	IETEST	IECAT
CES	TI	INCL1	Inclusion test 1	INCLUSION
CES	TI	INCL2	Inclusion test 2	INCLUSION
CES	TI	EXCL1	Exclusion test 1	EXCLUSION
CES	TI	EXCL2	Exclusion test 2	EXCLUSION
CES	TI	EXCL3	Exclusion test 3	EXCLUSION

×

?

CodeList generated from Trial Design dataset TI

OID for the CodeList:

Name for the CodeList:

Automatically assign the CodeList to variable **IETESTCD**

Also generate a CodeList to variable **IETEST** and automatically assign it to variable **IETEST**

By checking both the checkboxes, the system takes care that the codelist with codes and decodes is added to the define.xml and assigned to TITESTCD, and that the codelist with only the decoded is added and assigned to TITEST.

After clicking OK, the following messages are shown:

×

i

CodeList with OID CL.CES.IETESTCD
and Name **CodeList for Trial Design dataset TI, variable IETESTCD**
added to the define.xml

and:

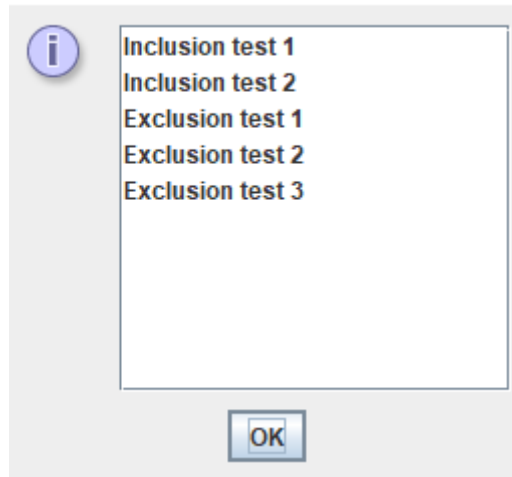
×

i

CodeList with OID CL.CES.IETESTCD.DECODE
and Name **CodeList for Trial Design dataset TI, variable IETESTCD - decoded values**
added to the define.xml

After closing all the "Trial Design Editor" windows, one can then check whether the assignment has been done correctly. For example, when selecting the "TITESTCD" cell for "TI" in the SDTM table, and then using "View - SDTM Associated CodeList", one gets:

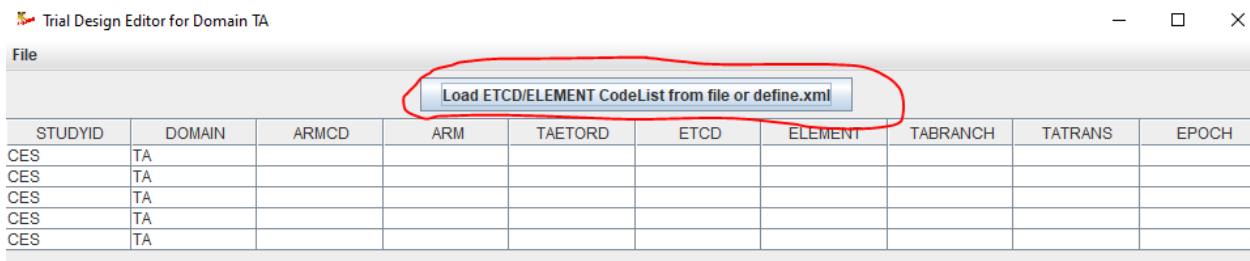
CodeList: CL.CES.IETESTCD.DECODE



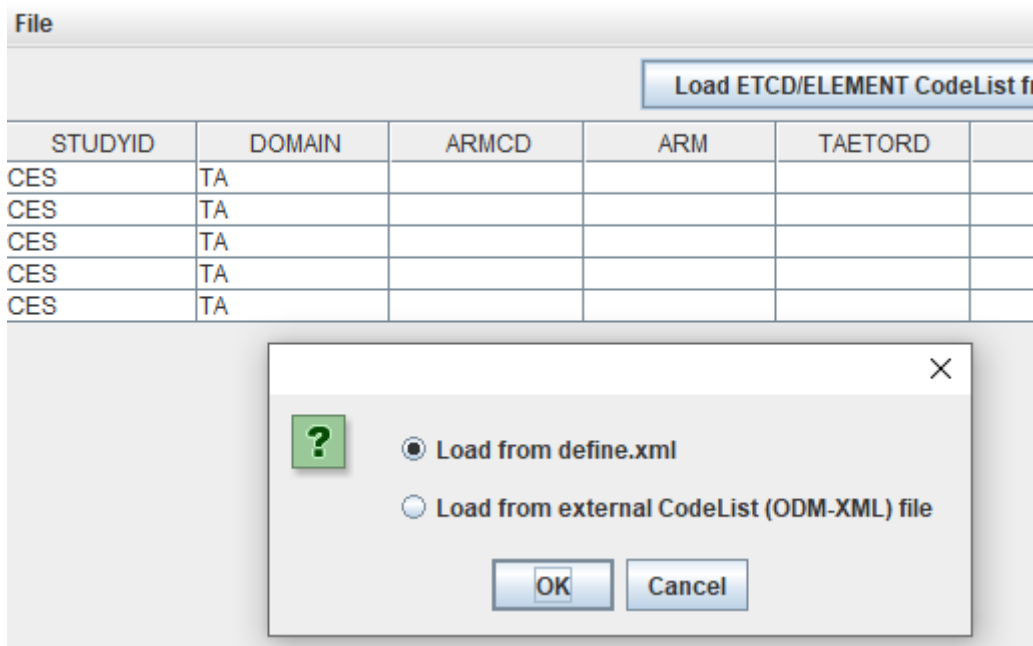
For TA (Trial Arms), there is however somewhat more.

Essentially, the name "Trial Arms" for this domain/dataset is completely wrong, as it describes the presence of the trial elements within each of the arms. This also means that there usually will be more than one row per arm definition. In good database design, there would be 3 tables: one with the trial elements, one with only the trial arms, and a "connection" table, showing which trial elements are used in each trial arm. However, [SDTM is far from good database design](#) ...

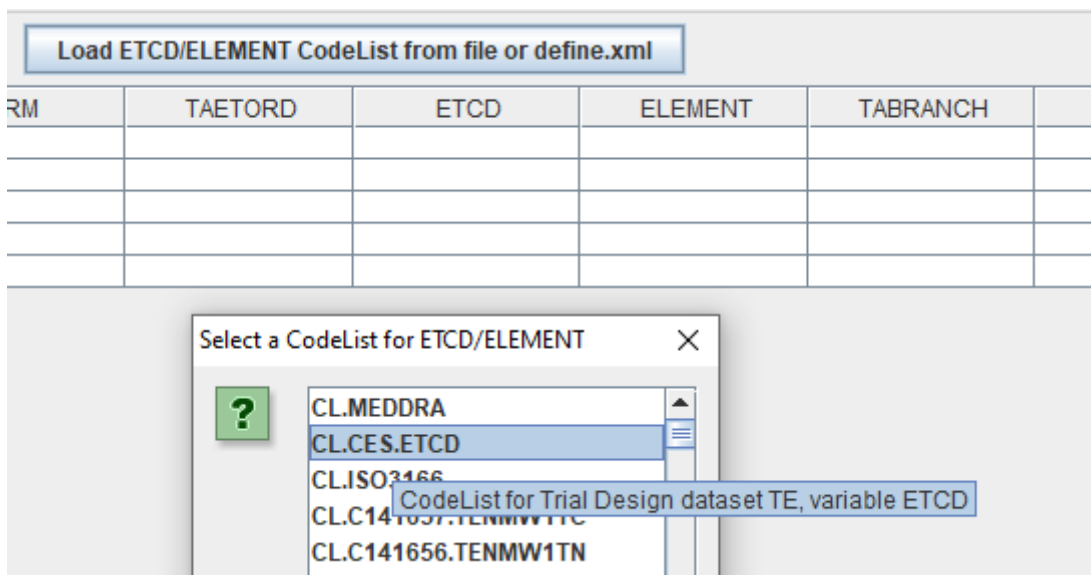
In order to facilitate the generation of the TA dataset, we added a feature, allowing to add a codelist with elements created before. So, for TA, when starting the generation, an additional button is shown:



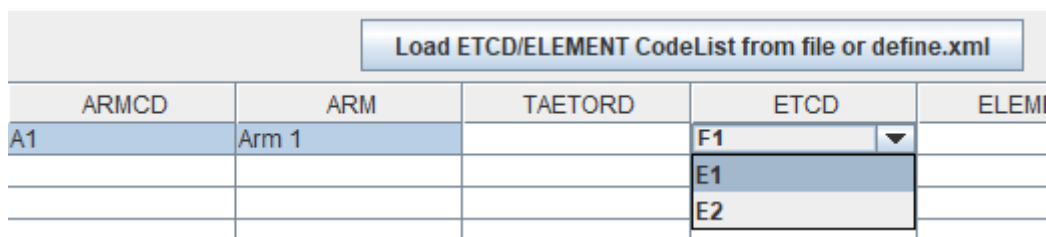
When clicked (best before adding any table data), one is asked:



When "Load from define.xml" is selected, and OK is clicked, all the codelists from the define.xml are listed, usually with the last ones generated on top (except for MedDRA):



and one can select the codelist with codes and decodes for ETCD (Element Code) and ELEMENT (Element Name). When the right one is selected, the values then appear as a dropdown for the ETCD and ELEMENT fields:



Also, if one e.g. selects "E1", the "ELEMENT" variable value will automatically be filled with the corresponding "decode" value, e.g. leading to:

Load ETCD/ELEMENT CodeList from file or define.xml				
RM	TAETORD	ETCD	ELEMENT	TABRANC
		E1	Element 1	

Also special is that the field EPOCH is mandatory, and there is a codelist for it, but it is extensible. So, if one tries to add a value for EPOCH, a dropdown is displayed:

ELEMENT	TABRANCH	TATRANS	EPOCH
Element 1			RASFI INF
			BASELINE
			BLINDED TREAT
			CONTINUATION T
			FOLLOW-UP
			INDUCTION TREA
			LONG-TERM FOL
			OBSERVATION
			OPEN LABEL TRI

but we can add a new (extended) value to it by scrolling down to the bottom and selecting "Other":

ELEMENT	TABRANCH	TATRANS	EPOCH
Element 1			RASFI INF
			LONG-TERM FOL
			OBSERVATION
			OPEN LABEL TRI
			RUN-IN
			SCREENING
			TREATMENT
			WASHOUT
			Other

and then adding a new (extended) value for our epoch in the textfield that is showing up:

ELEMENT	TABRANCH	TATRANS	EPOCH
Element 1			Other

Please specify other value for EPOCH ✕

?

After clicking "OK", the newly added value is added to the list and automatically selected:

ELEMENT	TABRANCH	TATRANS	EPOCH
Element 1			MYOWNEPOCH

The full TA table may then e.g. look like:

Trial Design Editor for Domain TA

File

Load ETCD/ELEMENT CodeList from file or define.xml

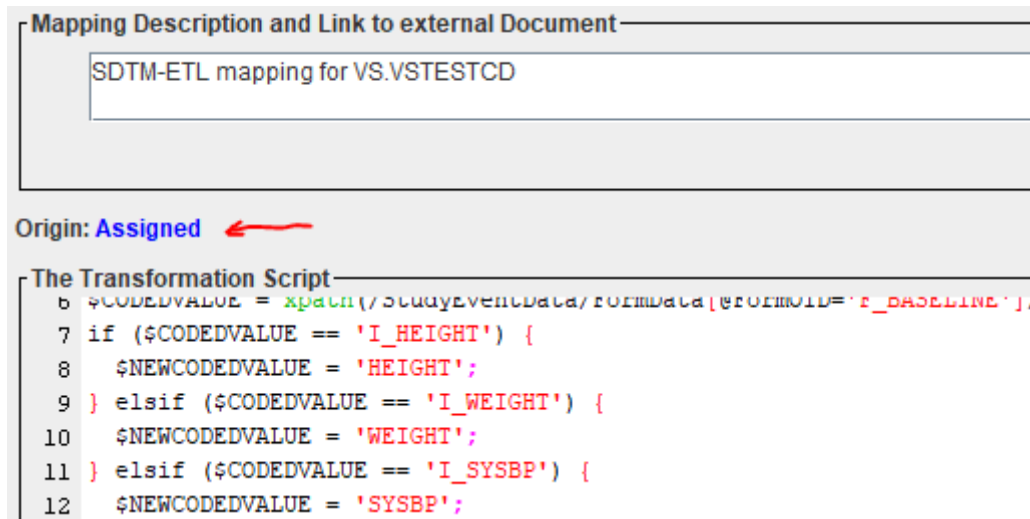
STUDYID	DOMAIN	ARMCD	ARM	TAETORD	ETCD	ELEMENT	TABRANCH	TATRANS	EPOCH
CES	TA	A1	Arm 1	1	E1	Element 1			MYOWNEPOCH
CES	TA	A1	Arm 1	2	E2	Element 2			MYOWNEPOCH
CES	TA	A1	Arm 1	3	E1	Element 1			BASELINE
CES	TA	A1	Arm 1	4	E2	Element 2			BLINDED TREATMENT
CES	TA	A2	Arm 2	1	E1	Element 1			MYOWNEPOCH
CES	TA	A2	Arm 2	2	E2	Element 2			MYOWNEPOCH
CES	TA	A2	Arm 2	3	E1	Element 1			BASELINE
CES	TA	A2	Arm 2	4	E2	Element 2			BLINDED TREATMENT

Everything further works as with the other trial design datasets.

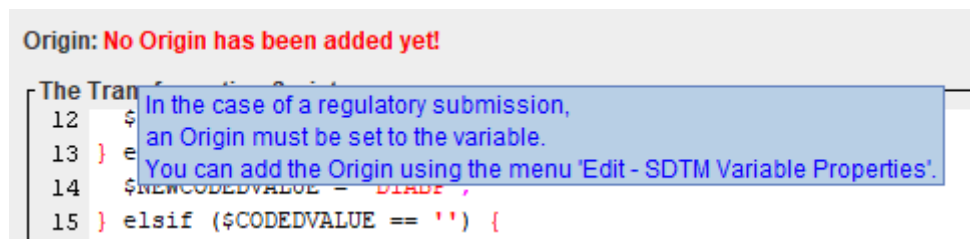
When saving as a codelist, this will use ARMCD and ARM for the coded and decoded values, but only the unique ARMCD and ARM values, as, as already stated, the TA table is not a real "trial arms definition" table.

Addition of "Origin" to the Mapping Editor

The value of the "Origin" has been added as a label to the mapping editor. For example:



If no "Origin" has been assigned (yet), this is also displayed, with a tooltip giving information how a value for "Origin" can be added.

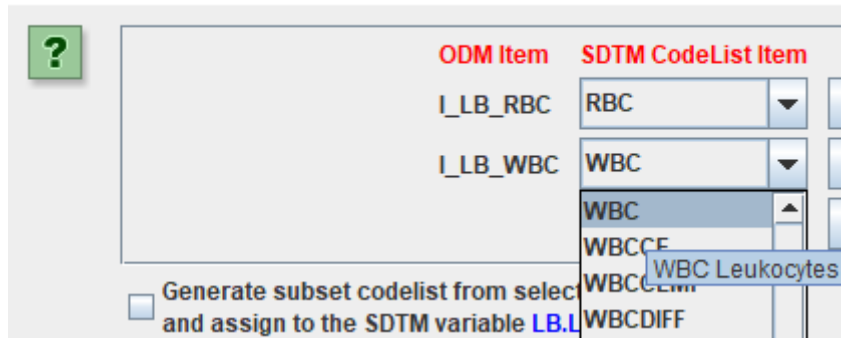


We have added this feature, as we noticed that many users forget to assign an "Origin" to each SDTM/SEND variable, which is required in the context of a regulatory submission.

Further improved "CodeList Mapping Wizard"

The CodeList Mapper Wizard is already one of the most powerful features of the SDTM-ETL software, even allowing semi-automated mapping between ODM items (either from a list, or from a codelist) and the SDTM or SEND codelist using the button "Attempt 1:1 mapping". When using this button, the system sorts the SDTM/SEND codelist items according to word similarity with the ODM item name and OID. For example, if we want to map the ODM item "WBC" to the SDTM codelist for LBTESTCD, the system suggests:

CodeList mapping between a set of ODM Items and SDTM CodeList "Laboratory"

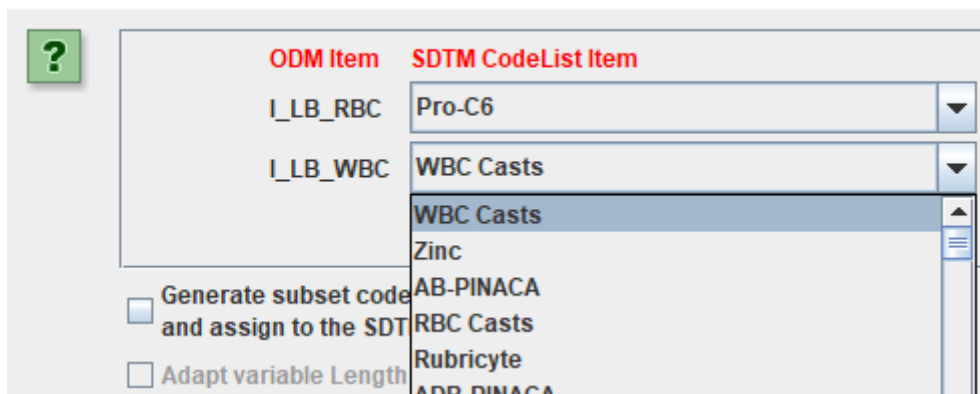


and the "1:1 mapping attempt" works perfectly.

However, when we do the same for LBTEST (test name), we have the problem that the SDTM associated codelist (CL.67154.LBTEST) knows nothing about "WBC" or "RBC", as CDISC codelists are just lists with little or no relations, and for example for Leukocytes, the word "WBC" does even not appear in the published synonyms.

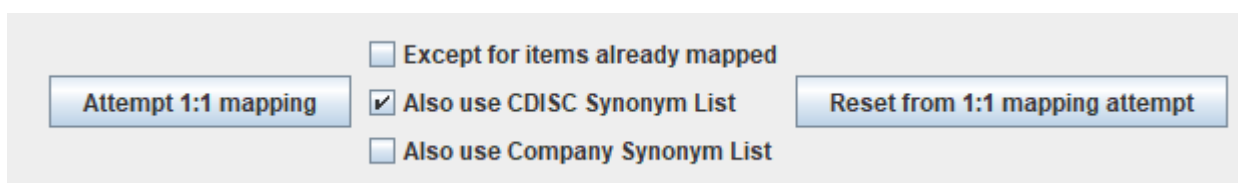
So, if we do a "1:1 mapping attempt" in the case of LBTEST, we get pretty strange proposals:

CodeList mapping between a set of ODM Items and SDTM CodeList "Laboratory Test Name"



"WBC casts" is not what we want², we want "Leukocytes" ...

What we can then do is to "reset" from the 1:1 mapping using the button "Reset from 1:1 mapping attempt"



and then look for "Leukocytes" and "Erythrocytes" in the dropbox, by scrolling. For example:

² This shows how illogical CDISC-CT sometimes is: "WBC" is not used for "Leukocytes", but for "Leukocytes casts", the name is "WBC casts" ...

CodeList mapping between a set of ODM Items and SDTM CodeList "Laboratory Test Name"

ODM Item	SDTM CodeList Item
I_LB_RBC	1,25-Dihydroxyvitamin D2
I_LB_WBC	Ery. Mean Corpuscular Hemoglobin

Generate subset code and assign to the SDT
 Adapt variable Length

In SDTM-ETL v.4.1 there is however another, new, possibility to find the suitable term in the SDTM codelist, using the new "Search" buttons:

CodeList mapping between a set of ODM Items and SDTM CodeList "Laboratory Test Name"

ODM Item	SDTM CodeList Item
I_LB_RBC	1,25-Dihydroxyvitamin D2
I_LB_WBC	1,25-Dihydroxyvitamin D2
WBC	1,25-Dihydroxyvitamin D2

These become interesting especially when one is not sure about the exact word or the way it is written. When the button is clicked for "WBC", a new dialog appears:

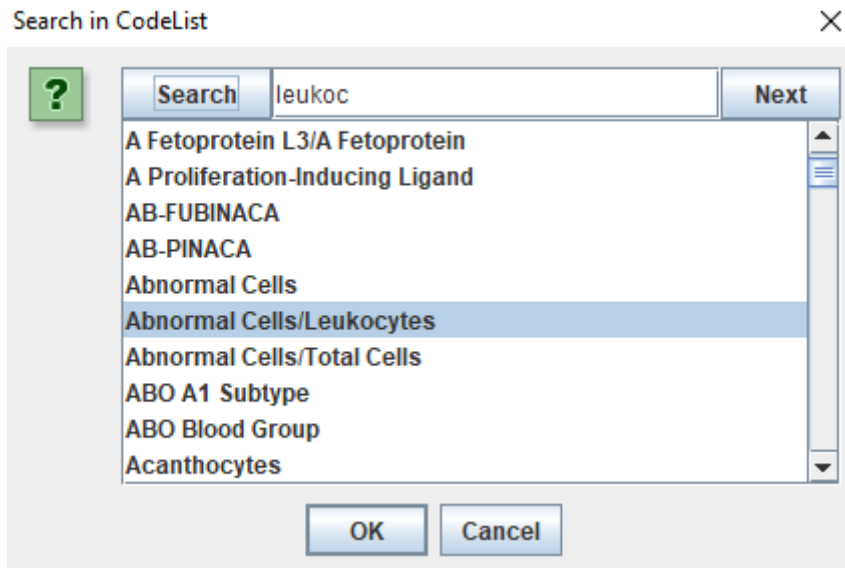
Search in CodeList

Search Next

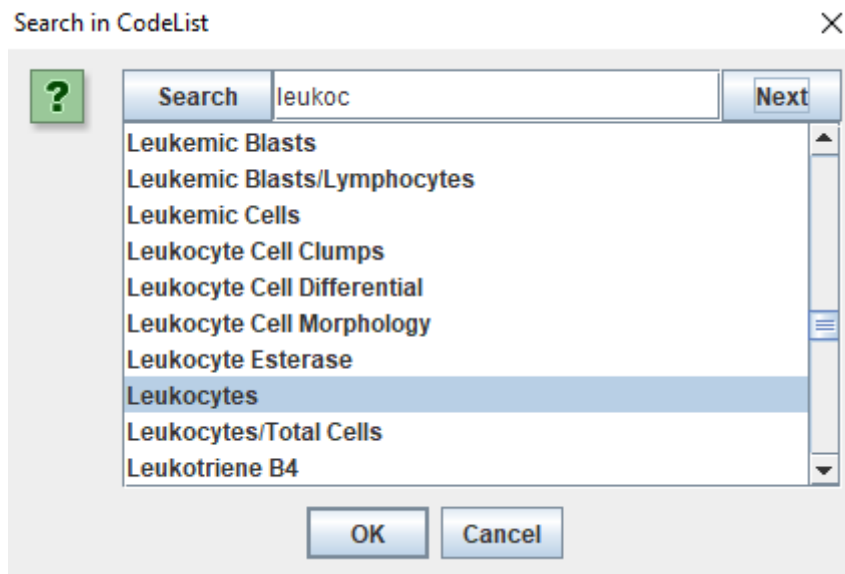
- 1,25-Dihydroxyvitamin D2
- 1,25-Dihydroxyvitamin D3
- 1,25-DihydroxyvitD2+1,25-DihydroxyvitD3
- 1,3-Beta-D-Glucan
- 1,5-Anhydroglucitol
- 1-Hydroxymidazolam
- 11-Dehydro-Thromboxane B2 Excretion Rate
- 11-Dehydro-Thromboxane B2
- 11-Nor-Delta9-THC-9-Carboxylic Acid
- 17-Hydroxyprogesterone

OK Cancel

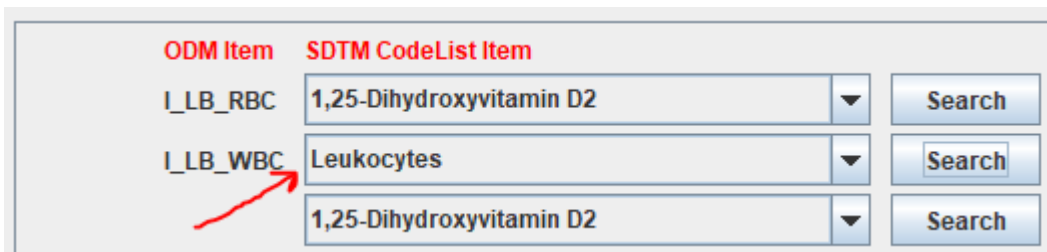
which can then be used to perform a search. For example, when we add "leukoc" and click "Search":



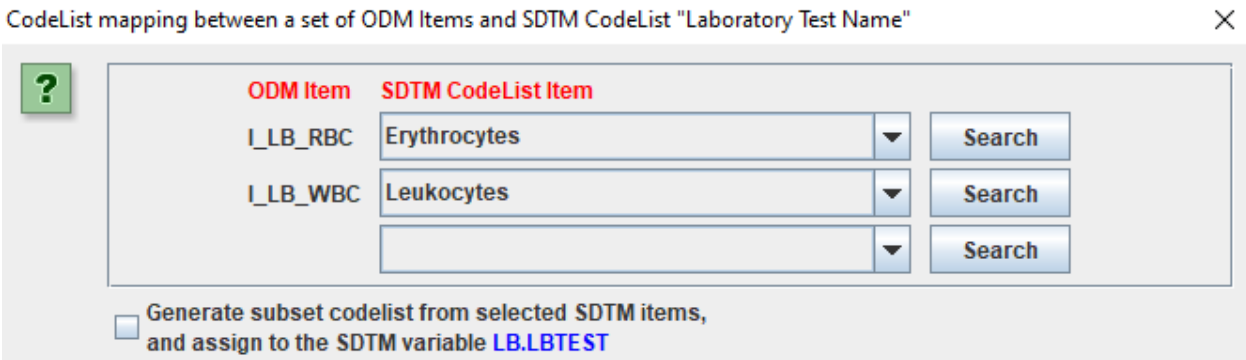
and then keep clicking "Next", until we find what we want:



and when then clicking "OK", the selected term value is added to the mapping:

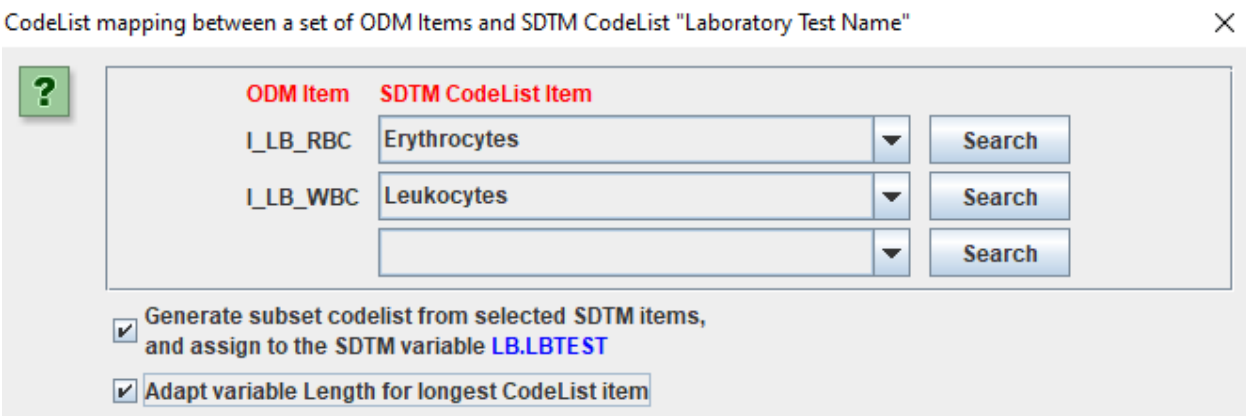


We can then do the same for "Erythrocytes" (as also "RBC" is not in the list, even not as a synonym), and setting the mapping for the missing value (blank) to blank:

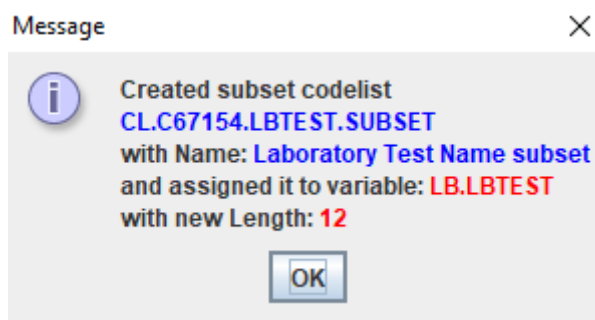


we have the mapping that we want.

At that point, it is highly recommended to check the checkbox "Generate subset codelist from selected SDTM items, and assign to the SDTM variable ...", and also, especially in the case of SAS Transport 5 files to be generated, to check the checkbox "Adapt variable Length for longest codelist Item"³:



After clicking "OK", the mapping script is generated, and a "subset codelist" is generated and assigned to the variable "LBTEST":



and the mapping script itself is shown:

³ As otherwise (but only for the outdated SAS Transport format, as it is a "fixed field length" format, just like punch cards), the field will be further filled with blanks when "Length" is set to a too high value, and values may be truncated when "Length" is set to a too low value.

Designing mapping for SDTM Variable: LB.LBTEST

Mapping Description and Link to external Document

SDTM-ETL mapping for LB.LBTEST

External Document Link

The Transformation Script

```

1 # Mapping using ODM element ItemData with ItemOID I_LB_RBC - value from attribute ItemOID
2 # Generalized for all StudyEvents
3 # Generalized for all Items within the ItemGroup
4 # Mapping for ODM Items [I_LB_RBC, I_LB_WBC] to SDTM CodeList LB.LBTEST
5 # with CodeList OID 'CL.C67154.LBTEST'
6 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/
7 if ($CODEDVALUE == 'I_LB_RBC') {
8   $NEWCODEDVALUE = 'Erythrocytes';
9 } elseif ($CODEDVALUE == 'I_LB_WBC') {
10  $NEWCODEDVALUE = 'Leukocytes';
11 } elseif ($CODEDVALUE == '') {
12  $NEWCODEDVALUE = '';
13 } else {
14  $NEWCODEDVALUE = '';
15 }
16 $LB.LBTEST = $NEWCODEDVALUE;

```

Essentially, the "Search" button makes it even easier to find the appropriate SDTM/SEND codelist value for the given item of the source item or code, especially when the SDTM/SEND codelist is long or very long. For example, the codelists for LBTESTCD and LBTEST already contains over 2,400 terms, growing with each new release of CDISC-CT⁴.

Of course, one will usually have much more lab tests in a real study, but for the explanation of the new feature, we kept it very simple here.

Encouraging the use of the "decode()" function

CDISC publishes its controlled terminology as "lists", leaving it up to the user to develop the relationships between terms in the different lists. For example, for "WBC" in the codelist for "LBTESTCD" (laboratory test code), in the XML publication, we find:

```

<EnumeratedItem CodedValue="WBC" nciidm:ExtCodeID="C51948">
  <nciodm:CDISCSynonym>Leukocytes</nciodm:CDISCSynonym>
  <nciodm:CDISCSynonym>White Blood Cells</nciodm:CDISCSynonym>
  <nciodm:CDISCDefinition>A measurement of the leukocytes in a bi
  <nciodm:PreferredTerm>Leukocyte Count</nciodm:PreferredTerm>
</EnumeratedItem>

```

Although "Leukocytes" is mentioned as a synonym (but also "White blood cells" is mentioned), there is no indication that "Leukocytes" is the submission value needed in the parallel codelist "LBTEST" (laboratory test name). The only way this relation can be resolved is by looking at the CDISC NCI code, in this case "C51948").

One also sees that the way this is published, is not compatible with the define.xml standard, as for the latter, the CDISC-NCI code is expected to come in an "Alias" element.

When we then look up the CDISC-NCI code "C51948" in the codelist for "LBTEST" (laboratory test name), we find:

⁴ This is due to the refusal of CDISC to make the LOINC code as the unique identifier for a test, as used everywhere in healthcare, except for ... clinical research.

```

<EnumeratedItem CodedValue="Leukocytes" ncioidm:ExtCodeID="C51948">
  <ncioidm:CDISCSynonym>Leukocytes</ncioidm:CDISCSynonym>
  <ncioidm:CDISCSynonym>White Blood Cells</ncioidm:CDISCSynonym>
  <ncioidm:CDISCDefinition>A measurement of the leukocytes in a biological specimen.</ncioidm:
  <ncioidm:PreferredTerm>Leukocyte Count</ncioidm:PreferredTerm>

```

with the "CodedValue" being "Leukocytes", which is the value required for "LBTEST".

As the codelists, as published by CDISC do not conform to the define.xml standard, each time CDISC publishes new controlled terminology (typically every quarter of the year), we immediately transform these codelists to codelists that can immediately be used in define.xml (i.e. conform to the ODM and Define-XML standards). These can then be found and downloaded from our SDTM-ETL website at: www.xml4pharma.com/SDTM-ETL/CDISC-CT/ followed by the codelist name/version. So e.g. to download the version 2022-03-25 for SDTM, use the link: http://www.xml4pharma.com/SDTM-ETL/CDISC-CT/SDTM_Terminology_2022-03-25.xml

If we look into the "define-xml friendly" codelist, we find for "WBC" in the codelist for variable "LBTESTCD":

```

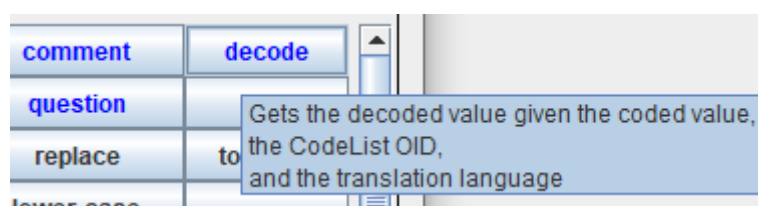
<CodeListItem CodedValue="WBC">
  <Decode>
    <TranslatedText>Leukocytes</TranslatedText>
  </Decode>
  <Alias Context="nci:ExtCodeID" Name="C51948"/>
</CodeListItem>

```

Where the value "Leukocytes" is the "decoded" value to be used in the "decoded" (of LBTESTCD) variable "LBTEST".

This also means that, at least for xxTESTCD/xxTEST combinations, a lookup in the codelist for xxTEST, and thus also developing a separate mapping for xxTEST is unnecessary, once the mapping for xxTESTCD has been developed.

SDTM-ETL has a function "decode" for this, which is also shown in the mapping editor as a button:



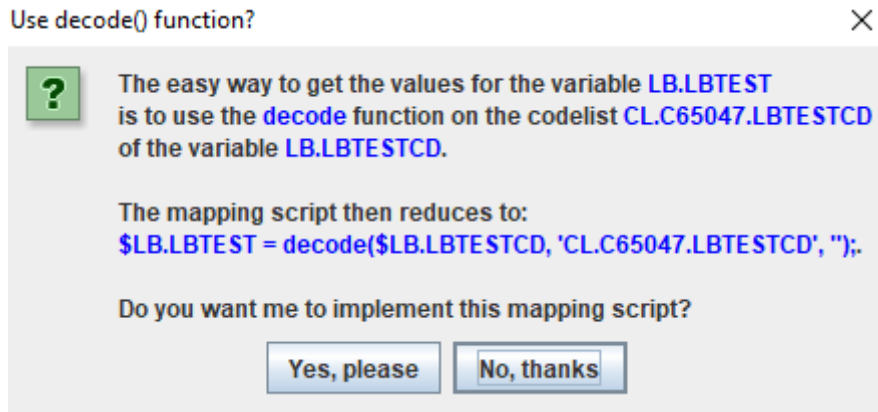
This also means that, for example for the variable "LBTEST", and when the (default) codelist "CL.C65047.LBTESTCD" has been assigned to the variable "LBTESTCD", the mapping script for variable "LBTEST" can be as simple as:

```
$LB.LBTEST = decode($LB.LBTESTCD, 'CL.C65047.LBTESTCD', );
```

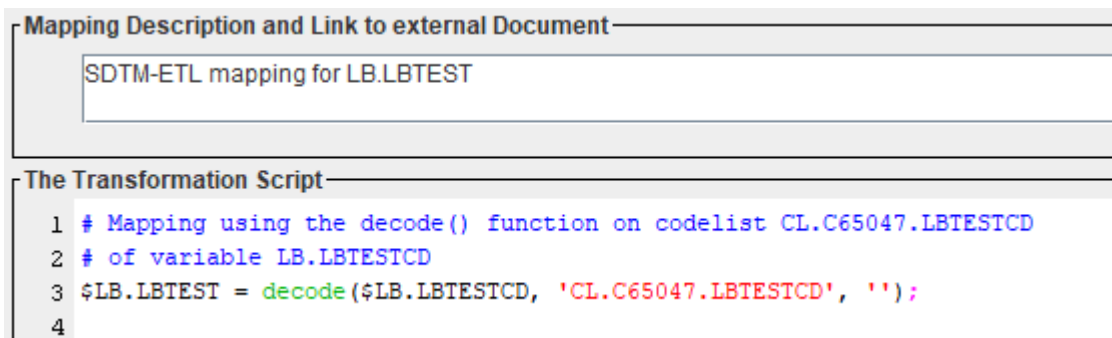
with the third parameter (") left empty, meaning "the default" language).

Many people however do not immediately know how to use the "decode()" function, also a they do not immediately know the OID of the codelist for the xxTESTCD codelist. Instead, they develop a whole new mapping script for the "xxTEST" variable, which is fine, but of course more work.

New in SDTM-ETL 4.1 is that for "xxTEST" variables for which the related "xxTESTCD" has a codelist with "decode" values, the system will suggest to use the "decode()" function, making the mapping script very simple, which also helps to avoid mapping errors. For example, when one double-clicks the "LB.LBTEST" cell in the table, the following dialog is displayed:



When the user then clicks "Yes, please", the following mapping script is generated:

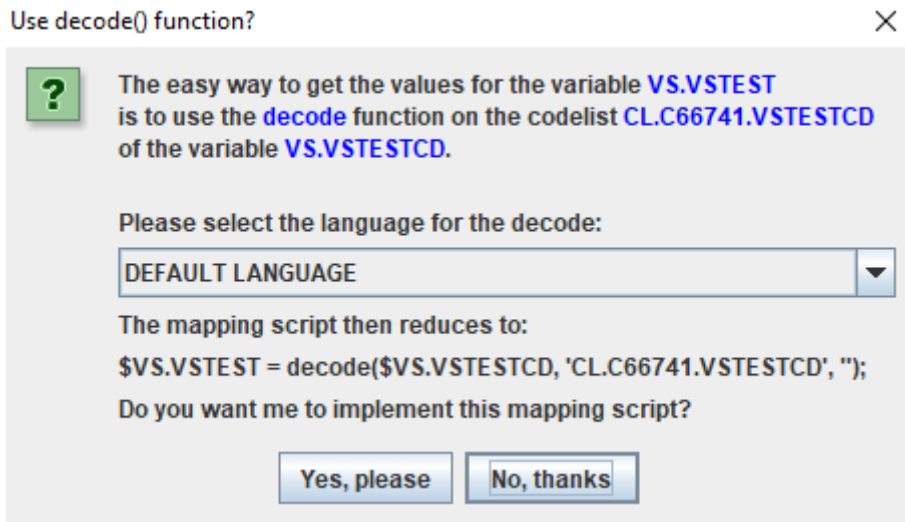


In case a "subset codelist" was assigned to the "LBTESTCD" variable, that one will of course be taken.

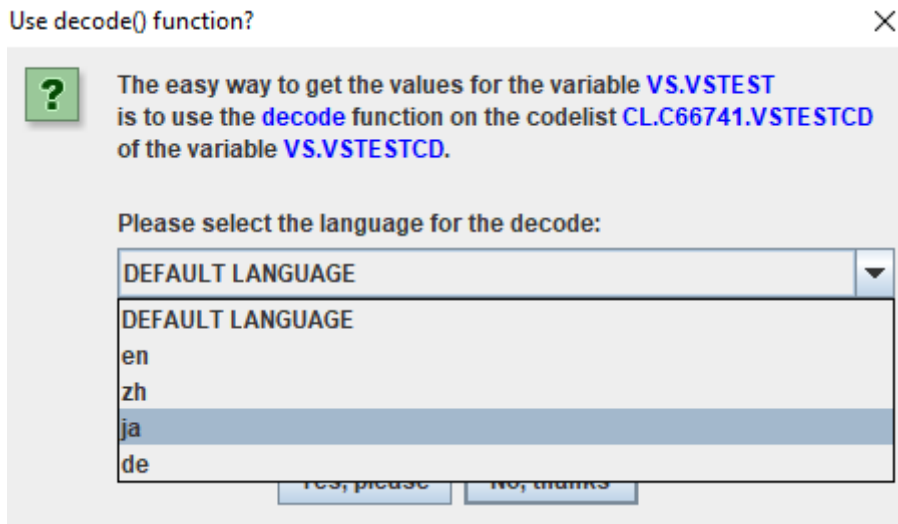
In case there is more than one language represented in the codelist, which may be (or become) the case for submissions in especially Japan, and China, e.g.

```
<CodeListItem CodedValue="HEIGHT">
  <Decode>
    <TranslatedText xml:lang="en">Height</TranslatedText>
    <TranslatedText xml:lang="zh">高度</TranslatedText>
    <TranslatedText xml:lang="ja">身長</TranslatedText>
    <TranslatedText xml:lang="de">Größe</TranslatedText>
  </Decode>
  <Alias Context="nci:ExtCodeID" Name="C25347"/>
</CodeListItem>
```

then the user is asked to also select the "decode" language. For example, for VSTEST:



and when the user chooses for "Japanese":




and then clicks "Yes, please", the mapping script reduces to:

```
The Transformation Script
1 # Mapping using the decode() function on codelist CL.C66741.VSTESTCD
2 # of variable VS.VSTESTCD
3 $VS.VSTEST = decode($VS.VSTESTCD, 'CL.C66741.VSTESTCD', 'ja');
4
5 |
```

which, upon execution, will then pick up the Japanese term for VSTEST.

Similarly, when the user uses "drag-and-drop" to an "xxTEST" variable, an extra button appears with "":

 The system found **2** ODM Items which can be mapped to the SDTM CodeList **CL.C67154.LBTEST**.

Do you want to use the **mapping wizard** to provide such a mapping?

Or do you want a **template script** will be generated that you need to fill in, in order to categorize the data?

You can also use the **decode function** on the value of **LB.LBTESTCD**

You can also choose to **ignore the CodeList** for now, then no codelist mapping is performed at all.

Suggesting the user that the "decode()" function can be used, and there is essentially no need to go through the mapping wizard, or start filling a template script.

When the user then clicks "Decode function", the mapping script becomes:

```
The Transformation Script
1 # Mapping using ODM element ItemData with ItemOID I_LB_RBC - value from attribute ItemOID
2 # Generalized for all StudyEvents
3 # Generalized for all Items within the ItemGroup
4 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID
5 # Mapping using the decode() function on codelist CL.C65047.LBTESTCD
6 # of variable LB.LBTESTCD
7 $LB.LBTEST = decode($LB.LBTESTCD, 'CL.C65047.LBTESTCD', '');
8
```

The line starting with "\$CODEDVALUE =" remains in the mapping script though, although it is not used anymore, for the case that the user changes his/her mind, and still wants to develop the mapping script him/her-self. This might e.g. be the case that some values cannot be mapped to CDISC-CT, although in that case, the usual procedure would be to first develop a subset codelist for the variable "LBTESTCD". For example (but not encouraged to do so):

```
The Transformation Script
1 # Mapping using ODM element ItemData with ItemOID I_LB_RBC - value from attribute ItemOID
2 # Generalized for all StudyEvents
3 # Generalized for all Items within the ItemGroup
4 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG
5 # Mapping using the decode() function on codelist CL.C65047.LBTESTCD
6 # of variable LB.LBTESTCD,
7 # with the exception for test "MYTEST"
8 if($CODEDVALUE = 'MYTEST') {
9     $LB.LBTEST = 'My special test';
10 } else {
11     $LB.LBTEST = decode($LB.LBTESTCD, 'CL.C65047.LBTESTCD', '');
12 }
13
```

Remark that this new feature is limited to "xxTEST" variables for which there is a parallel codelist for the "xxTESTCD" variable with "decoded" values. No such suggestion will be made for other variables such as "AEDECOD".

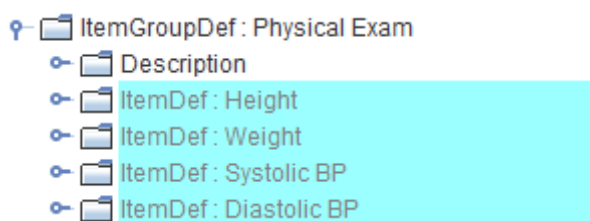
The "decode" function can however also be used for other variables, but the suggestion to use it will not automatically be made.

Also remark the new "Ignore CodeList" button. This button can be used when the ODM already implements the SDTM codelists, as is often the case when CDASH forms in electronic form have been implemented. In that case, clicking "Ignore CodeList" leads to just taking the value from the ODM "as is", i.e. it is supposed that it already conforms to the SDTM codelist.

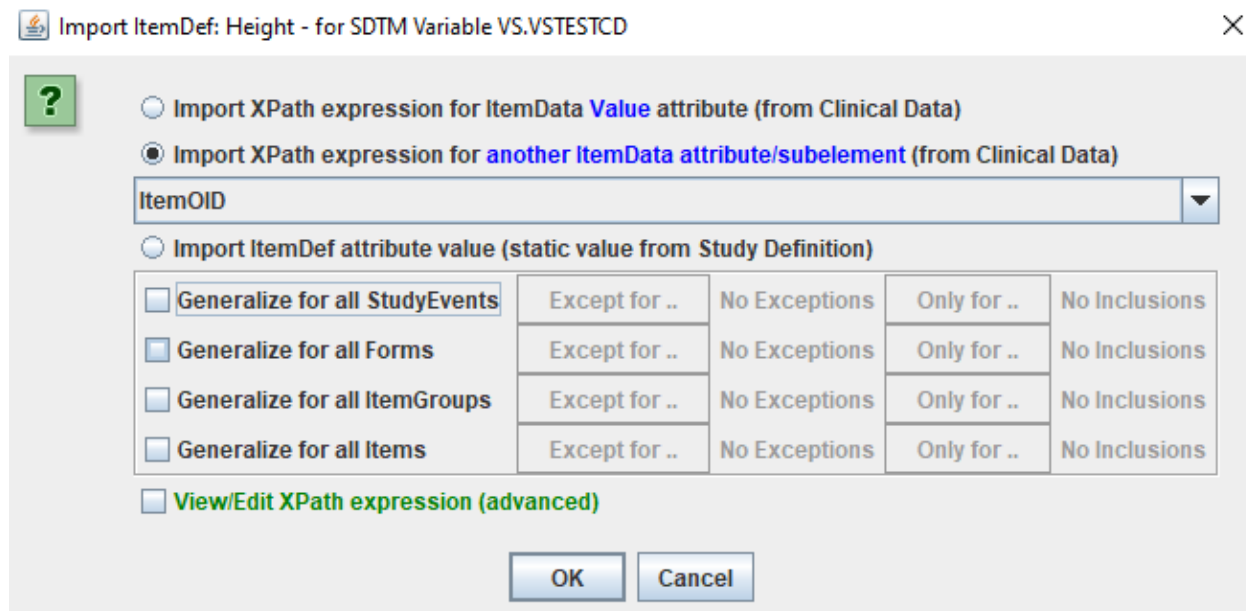
Parallel mapping script generation for as well --TESTCD as --TEST variables

In SDTM-ETL 4.1, we have even gone one step further: It is now possible to generate the mapping scripts for -TESTCD and -TEST variables simultaneously using the "mapping wizard". This however only works well when all items of a series (or in the ODM codelist for an item) can be mapped to CDISC controlled terminology for -TESTCD and -TEST. The reason for this is that there is a 1:1 relationship between the values of -TESTCD and -TEST, but that (until now) both still need to be submitted⁵.

Suppose that we want to map the items "Height", "Weight", "Systolic Blood Pressure" and "Diastolic Blood Pressure" to both VSTESTCD and VSTEST:



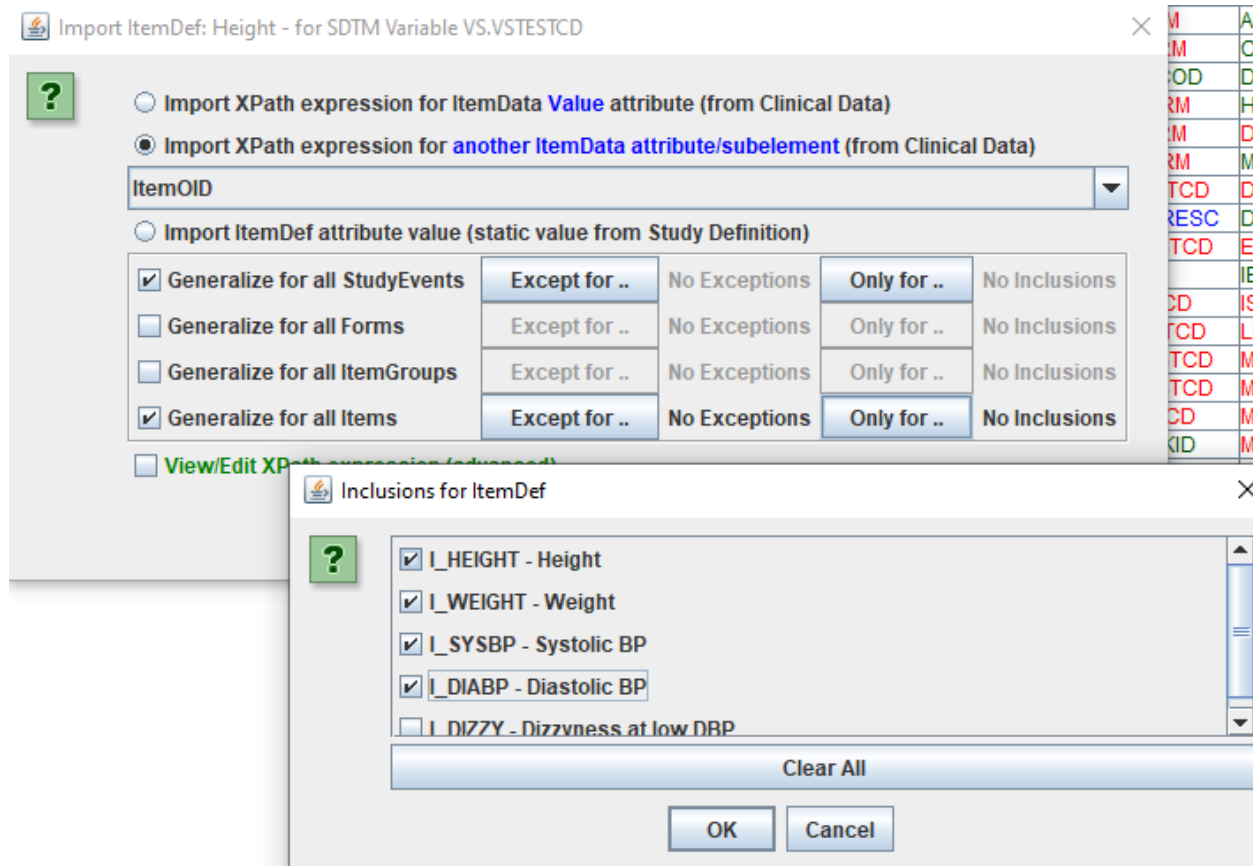
We select one of them and drag it to the cell "VS.VSTESTCD". A dialog is shown:



We indeed want to pick up the OID (identifier) of the item and map that to both VSTESTCD and VSTEST. As we want to have this for all visits, we check the checkbox "Generalize for all

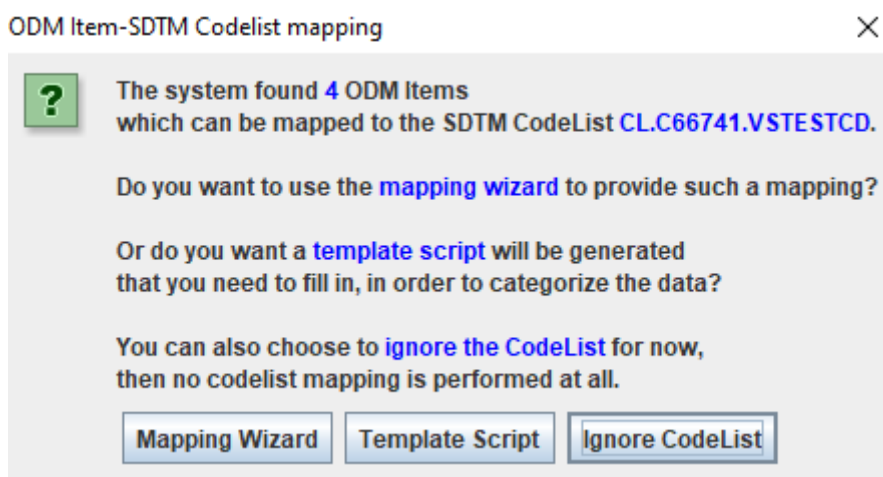
⁵ This once again is a relict of the mandated use of SAS Transport 5 and the rather primitive review tools at the regulatory authorities.

StudyEvents". As we also want to do this not only for "Height", but also for "Weight", "Systolic Blood Pressure" and "Diastolic Blood Pressure", we also check the checkbox "Generalize for all Items" (in the group) and then use the button "Only for ..." leading to:




where we check the checkboxes for "Height", "Weight", "Systolic BP", and "Diastolic BP".

After clicking "OK", the following dialog is displayed:



We select "Mapping Wizard". Remark that "Ignore CodeList" can be used when the OIDs used are identical (or contain) the SDTM VSTESTCD value. This can e.g. be the case when using CDASH CRFs implemented in ODM.

After clicking "Mapping Wizard", the "mapping wizard" is displayed:



ODM Item	SDTM CodeList Item	
I_HEIGHT	ABSKNF	<input type="button" value="Search"/>
I_WEIGHT	ABSKNF	<input type="button" value="Search"/>
I_SYSBP	ABSKNF	<input type="button" value="Search"/>
I_DIABP	ABSKNF	<input type="button" value="Search"/>
	ABSKNF	<input type="button" value="Search"/>

Generate subset codelist from selected SDTM items, and assign to the SDTM variable **VS.VSTESTCD**

Also create a subset codelist for the corresponding VS.VSTEST (test name) variable, and generate the corresponding mapping script for the corresponding VS.VSTEST variable

Adapt variable Length for longest CodeList item

Except for items already mapped

Also use CDISC Synonym List

Also use Company Synonym List

Use SDTM *decoded* value

Ask to store mappings as synonyms to Company Synonym List

In case the OIDs have a meaning and are "look alike" to the SDTM test codes, one can use the button "Attempt 1:1 mapping", in our case leading to:

ODM Item	SDTM CodeList Item	
I_HEIGHT	HEIGHT	<input type="button" value="Search"/>
I_WEIGHT	WEIGHT	<input type="button" value="Search"/>
I_SYSBP	SYSBP	<input type="button" value="Search"/>
I_DIABP	DIABP	<input type="button" value="Search"/>
		<input type="button" value="Search"/>

Generate subset codelist from selected SDTM items, and assign to the SDTM variable **VS.VSTESTCD**

Also create a subset codelist for the corresponding **VS.VSTEST (test name) variable**, and generate the corresponding mapping script for the corresponding **VS.VSTEST variable**

Adapt variable Length for longest CodeList item

Except for items already mapped

Also use CDISC Synonym List

Also use Company Synonym List

Use SDTM **decoded** value

Ask to store mappings as synonyms to Company Synonym List

Remember that such mappings can also be stored and reused by having them in a "Company Synonym List" (which can be found in the folder "Company_CT"). For example, if the company internal code for "Systolic Blood Pressure" is "N43712", one can store the mapping to "Systolic Blood Pressure" in the file "Company_CT.txt" in the folder "Company_CT".

As we don't want to keep the full CDISC codelist for vital signs in our define.xml (it contains 60 items of which we only use 4. For LBTESTCD/LBTEST, there currently are even almost 2,500 terms, still growing), we check the checkbox "Generate subset codelist ...". Two other checkboxes then become available:

ODM Item	SDTM CodeList Item	
I_HEIGHT	HEIGHT	<input type="button" value="Search"/>
I_WEIGHT	WEIGHT	<input type="button" value="Search"/>
I_SYSBP	SYSBP	<input type="button" value="Search"/>
I_DIABP	DIABP	<input type="button" value="Search"/>
		<input type="button" value="Search"/>

Generate subset codelist from selected SDTM items, and assign to the SDTM variable **VS.VSTESTCD**

Also create a subset codelist for the corresponding **VS.VSTEST (test name) variable**, and generate the corresponding mapping script for the corresponding **VS.VSTEST variable**

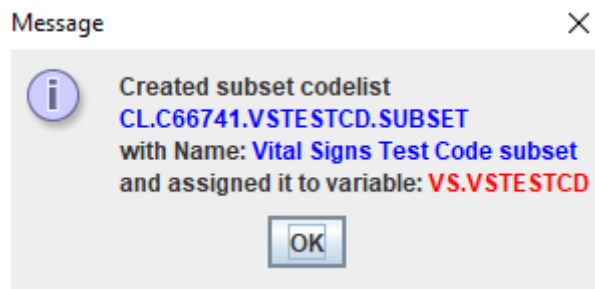
Adapt variable Length for longest CodeList item

The checkbox "Also create a subset codelist for the corresponding VS.VSTEST (test name) variable, and generate the corresponding mapping script for the corresponding VS.VSTEST variable is new in SDTM-ETL 4.1.

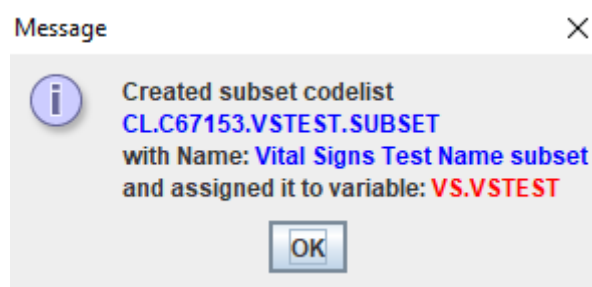
When it is checked, an additional subset codelist for VSTEST will be generated, and also a mapping script will be automatically be generated for VSTEST, in addition to the one for VSTEST.

When then "OK" is is clicked, subset codelists for as well VSTESTCD as VSTEST are generated (and assigned to the variable), and the mapping scripts for as well VSTESTCD as VSTEST are automatically generated.

First, a message appears:



followed by:



The mapping editor for VSTESTCD is then opened:

```
The Transformation Script
1 # Mapping using ODM element ItemData with ItemOID I_HEIGHT - value from attribute ItemOID
2 # Generalized for all StudyEvents
3 # Generalized for all Items within the ItemGroup
4 # Mapping for ODM Items [I_HEIGHT, I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.VSTESTCD
5 # with CodeList OID 'CL.C66741.VSTESTCD'
6 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_BASELINE']/ItemGroupData[@ItemGroupOID='IG_PE_BASE']/I
7 if ($CODEDVALUE == 'I_HEIGHT') {
8   $NEWCODEDVALUE = 'HEIGHT';
9 } elseif ($CODEDVALUE == 'I_WEIGHT') {
10  $NEWCODEDVALUE = 'WEIGHT';
11 } elseif ($CODEDVALUE == 'I_SYSBP') {
12  $NEWCODEDVALUE = 'SYSBP';
13 } elseif ($CODEDVALUE == 'I_DIABP') {
14  $NEWCODEDVALUE = 'DIABP';
15 } elseif ($CODEDVALUE == '') {
16  $NEWCODEDVALUE = '';
17 } else {
18  $NEWCODEDVALUE = 'NULL';
19 }
20 $VS.VSTESTCD = $NEWCODEDVALUE;
21
22
```

Which then can still be edited, when necessary.

However, also a mapping script for the corresponding VSTEST variable has been generated. It can be inspected and edited by a double click on the VSTEST cell in the SDTM/SEND table on the main screen.

In our case we get:

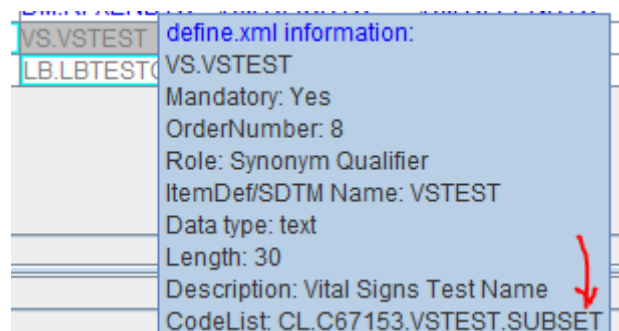
```

The Transformation Script
1 # Mapping using ODM element ItemData with ItemOID I_HEIGHT - value from attribute ItemOID
2 # Generalized for all StudyEvents
3 # Generalized for all Items within the ItemGroup
4 # Mapping for ODM Items [I_HEIGHT, I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.VSTESTCD
5 # with CodeList OID 'CL.C66741.VSTESTCD'
6 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_BASELINE']/ItemGroupData[@ItemGroupOID='IG_PE_BASE']/I
7 # Mapping code for variable VS.VSTEST
8 # automatically generated from the mapping script for the corresponding variable null
9 # using decoded values of the VS.VSTESTCD codelist
10 if ($CODEDVALUE == 'I_HEIGHT') {
11   $NEWCODEDVALUE = 'Height';
12 } elseif ($CODEDVALUE == 'I_WEIGHT') {
13   $NEWCODEDVALUE = 'Weight';
14 } elseif ($CODEDVALUE == 'I_SYSBP') {
15   $NEWCODEDVALUE = 'Systolic Blood Pressure';
16 } elseif ($CODEDVALUE == 'I_DIABP') {
17   $NEWCODEDVALUE = 'Diastolic Blood Pressure';
18 } elseif ($CODEDVALUE == '') {
19   $NEWCODEDVALUE = 'NULL';
20 } else {
21   $NEWCODEDVALUE = 'NULL';
22 }
23 $VS.VSTEST = $NEWCODEDVALUE;
24
25

```

It is important that one also inspects this generated mapping script for the -TEST variable, as the mapping script editor is not automatically opened (the editor of -TESTCD is opened). Especially if one still adapts the mapping script for -TESTCD, one should of course also still adapt the mapping script for the corresponding -TEST variable.

The generated subset codelists automatically get the OID of the parent codelist OID with a suffix of ".SUBSET". This can also be seen in the tooltip when hovering the mouse over the cell. For example:



If there was already a codelist with the same OID, the user will be asked to delete the old one, or to rename it. In most cases one will want to delete the old one, but in the case of keeping different subsets e.g. for ValueList usage, one will want to assign another OID.

This new feature is very powerful, but one should use it with care. So it is very important that in the final result, there is indeed a value for -TEST for each value of -TESTCD.

Mapping Suggestions from SDTM annotations in the ODM file

It has become good practice to annotate ODM study designs with SDTM annotations. This is done using the "Alias" element on the ODM "ItemDef" which represents a data point definition like a question on a form or CRF. For example:

```
<ItemDef DataType="integer" Length="3" Name="Systolic BP" OID="I_SYSBP" SASFieldName="SYSIL" SDSVarName="VSORRES">
  <Question>
    <TranslatedText xml:lang="en">Systolic blood pressure</TranslatedText>
    <TranslatedText xml:lang="fr">Tension artérielle systolique</TranslatedText>
    <TranslatedText xml:lang="de">Systolischer Blutdruck</TranslatedText>
    <TranslatedText xml:lang="ko">□□ □□</TranslatedText>
  </Question>
  <MeasurementUnitRef MeasurementUnitOID="MU_MMHG"/>
  <RangeCheck Comparator="LT" SoftHard="Hard"/>
  <Alias Context="SDTM" Name="VSORRES WHERE VSTESTCD=SYSBP"/>
</ItemDef>
```

indicating that the data point is to be used for mapping to VSORRES and VSTESTCD.

When selecting this item definition in the ODM tree, the cell for "VSORRES" will immediately be highlighted:

BSCAT	MB.MBORRES	MB.MBL
SSCAT	MS.MSORRES	MS.MSC
CAT	MI.MISCAT	MI.MIOF
OCAT	MO.MOSCAT	MO.MOF
CSCAT	PC.PCORRES	PC.PCC
PORRESU	PP.PPSTRESC	PP.PPS
ESCAT	PE.PEBODSYS	PE.PEO
SORRES	QS.QSORRESU	QS.QSE
PSCAT	RP.RPORRES	RP.RPC
CORRES	SC.SCORRESU	SC.SCS
SORRES	SS.SSSTRESC	SS.SSS
JORRES	TU.TUSTRESC	TU.TUN
RTEST	TR.TRORRES	TR.TRO
STEST	RS.RSCAT	RS.RSC
SPOS	VS.VSORRES	VS.VSO
ASCAT	FA.FAORRES	FA.FAOF
RCAT	SR.SRSCAT	SR.SRC
BSCAT	LB.LBORRES	LB.LBO
SPOS	VS.VSORRES	VS.VSO

"inviting" the user to do a drag-and-drop to "VSORRES".

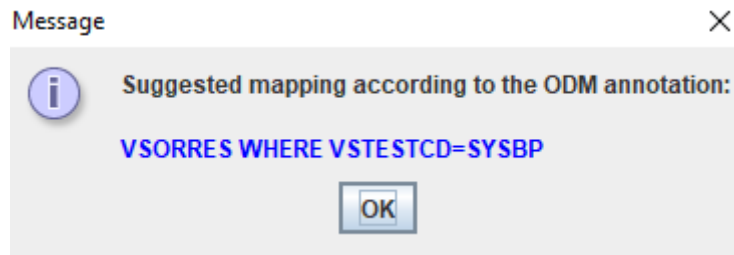
When one than drag-and-drops this from the ODM tree to e.g. VSCAT, a message will appear:

×

? The ODM annotation suggests that the target variable should be one of: **VSORRES**

Are you sure you want to map this into variable **VSCAT**?

When the user clicks "No", the whole action is cancelled. When the user then clicks "Yes", a reminder of what was meant originally is provided:



but the mapping is still continued as usual.

Caching of LOINC code mappings

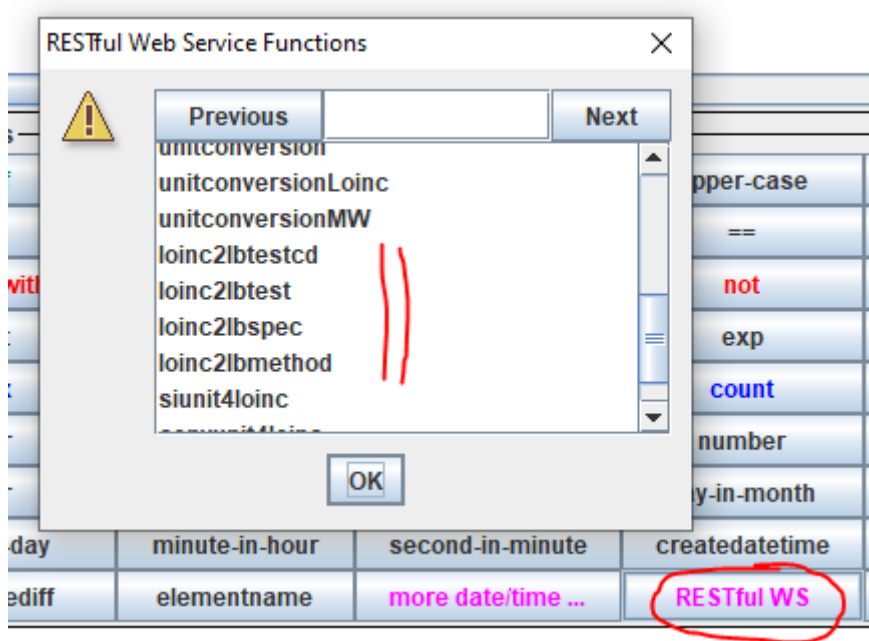
First after that the FDA made LOINC coding mandatory, people with the CDISC community started discovering the power of LOINC. Many still see LOINC coding as a "burden" whereas it essentially is a huge opportunity, especially as in many countries, laboratories are mandated by law or other regulations to provide the LOINC code when reporting lab results.

Essentially, when the LOINC code is provided for a test (whether it be a lab test, a microbiology test - e.g. COVID-19 test, a vital signs test), there should be obligation anymore to also provide values for xxTESTCD, xxTEST, xxSPEC, xxLOC, etc. as the LOINC coding system is superior to the CDISC-CT "lists", and the meaning of each LOINC code can easily be obtained and displayed by each modern review tool through the use of RESTful web services such as those from the NLM, LOINC itself, or MedLine-Plus. For more details, please see CDISC conference publications.

However, we are not that far yet, and CDISC still requires the population of xxTESTCD, xxTEST, xxSPEC, xxLOC and other variables, even when the LOINC code is provided.

In SDTM-ETL, there are some special functions to fully automatically populate these variables starting from the LOINC code for variables in the LB domain. These functions use RESTful web services.

These functions can be found when clicking the button "RESTful WS" in the mapping editor:



and when used, the mapping script e.g. looks like:

Designing mapping for SDTM Variable: LB.LBTESTCD

Mapping Description and Link to external Document
 SDTM-ETL mapping for LB.LBTESTCD External Document

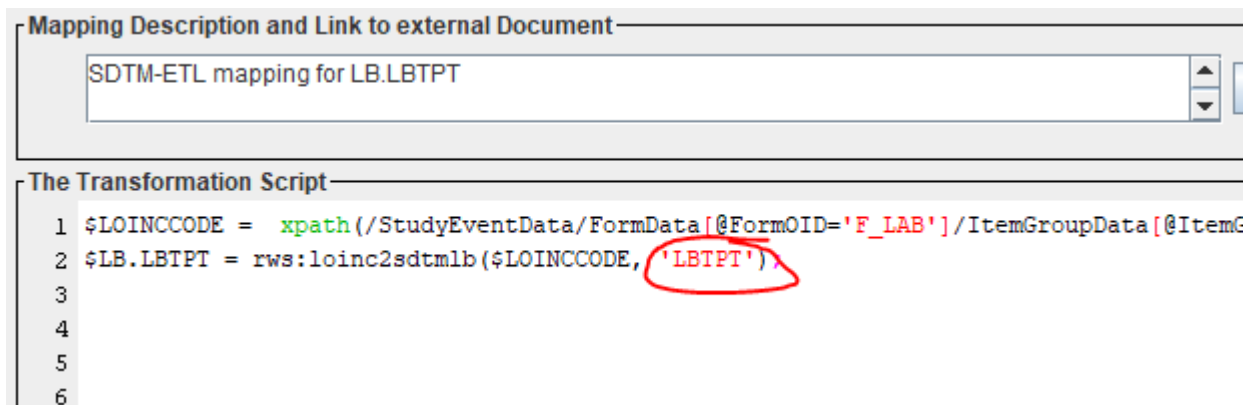
The Transformation Script

```

1 # Mapping using ODM element ItemData with ItemOID I_LB_RBC_LOINC - value from attribute ItemOID
2 # Generalized for all StudyEvents
3 # Generalized for all Items within the ItemGroup
4 # Using categorization as a CodeList is associated with the SDTM CodeList
5 # but no CodeList is associated with the ODM data
6 $LOINCCODE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_
7 # LOINC code 90236-1 is not covered by the LOINC to CDISC mapping
8 # LOINC code 90236-1 is for "Kearatan Sulfate substance concentration in blood dot"
9 if($LOINCCODE = '90236-1') {
10   $TEMP = 'KERATSLE';
11 } else {
12   TEMP = rws:loinc2lbtestcd($LOINCCODE);
13 }
14 $LB.LBTESTCD = $TEMP; ;
  
```

also providing a "classic" mapping for LOINC code 90236-1 which is not covered by the (extended) LOINC-CDISC mapping. Please remark the "rws:" prefix in "rws:loinc2lbtestcd(...)".

As there are also other variables than LBTESTCD, LBTEST, LBSPEC and LBMETHOD, there is an additional "generic" method allowing to also use the RESTful web service for these other SDTM variables. For example, for the LOINC code "11064-3", "Urea nitrogen [Mass/volume] in Serum or Plasma, post dialysis", also the variable "LBTPT" (planned time point name) can be automatically be populated. One would then use the "generic" SDTM-ETL function "rws:loinc2sdtmlb()":



with the second argument being the SDTM variable that is the target.

When used in execution of all the mapping scripts, and there is a LOINC code "11064-3", "Urea nitrogen [Mass/volume] in Serum or Plasma, post dialysis", the used RESTful web service will return:

```

<XML4PharmaServerWebServiceResponse ServerDateTime="2022-06-01T09:04:50">
  <WebServiceRequest>http://www.xml4pharmaserver.com:8080/CDISCCTService2/rest/LOINC2SDTMLB/11064-3</WebServiceRequest>
  <Response>
    <LOINC2SDTMMapping MappingSource="CDISC" TargetSDTMDomain="LB">
      <LBTESTCD NCIcode="C125949">UREAN</LBTESTCD>
      <LBTEST NCIcode="C125949">Urea Nitrogen</LBTEST>
      <LBORRESU_Example NCIcode="C67015">mg/dL</LBORRESU_Example>
      <LBPOS/>
      <LBLOINC>11064-3</LBLOINC>
      <LBSPEC NCIcode="C105706">SERUM OR PLASMA</LBSPEC>
      <LBLOC/>
      <LBMETHOD/>
      <LBANMETH/>
      <LBFAST/>
      <LBTPT>POST-DIALYSIS</LBTPT>
      <SUPPLB.LBPTFL>Y</SUPPLB.LBPTFL>
      <SUPPLB.LBPDUR/>
      <SUPPLB.LBRESTYP>MASS CONCENTRATION</SUPPLB.LBRESTYP>
      <SUPPLB.LBRSLSCL>QUANTITATIVE</SUPPLB.LBRSLSCL>
      <SUPPLB.LBTSTOPO/>
      <SUPPLB.LBLLOD/>
      <SUPPLB.LBTSTCND/>
      <SUPPLB.LBMTHSEN/>
      <Example_UCUM_Units>mg/dL</Example_UCUM_Units>
    </LOINC2SDTMMapping>
  </Response>
</XML4PharmaServerWebServiceResponse>

```

and the function will pick up the value in the element "LBTPT".

RESTful web services are very powerful, but there may be circumstances that one would not want to use them. For example:

- company IT politics do not allow them (stupid, but this happens)
- the RESTful server (xml4pharmaserver.com) is down, e.g. for maintenance⁶.
- the connection to the RESTful web server is slow (the server service itself is very fast)

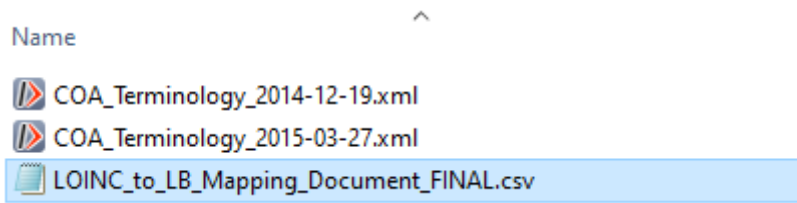
In such a case, one can revert to the classic mapping mechanisms for generating the LB mappings (costing a lot of time, sometime even days). SDTM-ETL however also provides to alternatives:

a) **use of a CSV file** containing the by the original CDISC published mappings

The SDTM-ETL distribution comes with a CSV file named

⁶ Although maintenance of the server is usually done in the weekend when there is a limited amount of traffic.

"LOINC_to_LB_Mapping_Document_FINAL.csv" in the folder "CDISC_CT":



which was extracted directly from the Excel file published by CDISC.
The contents are:



In SDTM-ETL, this file can be used using the function "loinc2sdtmlb_local()" which takes 2 arguments: the LOINC code, and the SDTM variable that needs to be automatically populated. For example, in a mapping for LBTESTCD, the mapping script then e.g. is:

```
The Transformation Script
1 # Mapping using ODM element ItemData with ItemOID I_LB_RBC_LOINC - value from attribute
2 # Generalized for all StudyEvents
3 # Generalized for all Items within the ItemGroup
4 # Using categorization as a CodeList is associated with the SDTM CodeList
5 # but no CodeList is associated with the ODM data
6 $LOINCCODE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupC
7 $LB.LBTESTCD = loinc2sdtmlb_local($LOINCCODE, 'LBTESTCD');
8
9
10
```

Important to remark here is that the scope of this function is very limited, as CDISC only published mappings for 1,400 LOINC codes. The "extended" mapping on the XML4PharmaServer however has mappings for almost 10,000 LOINC codes, still growing. The by CDISC published mapping is also not maintained. If you want to extend the mappings in the CSV file, you are of course free to do so, but ... at your own risk. Also, we do not take any responsibility at all for the correctness of the by CDISC published mappings.

b) using a local XML file with mappings

The folder "CDISC_CT" also contains a file "LOINC2SDTM_cached.xml":

Name
COA_Terminology_2014-12-19.xml
COA_Terminology_2015-03-27.xml
LOINC_to_LB_Mapping_Document_FINAL.csv
LOINC2SDTM_cached.xml
QRS_Terminology_2015-06-26.xml

Essentially, this file must be considered as a "user maintained" file. When delivered, the file only contains mappings for 2 LOINC codes, which just serve as an example:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <LOINC2SDTM>
3    <LOINC2SDTMMapping MappingSource="CDISC" TargetSDTMDomain="LB">
4      <LBTESTCD NCICode="C64431">ALB</LBTESTCD>
5      <LBTEST NCICode="C64431">Albumin</LBTEST>
6      <LBORRESU_Example NCICode="C42576">g/L</LBORRESU_Example>
7      <LBPOS/>
8      <LBLOINC>1751-7</LBLOINC>
9      <LBSPEC NCICode="C105706">SERUM OR PLASMA</LBSPEC>
10     <LBLOC/>
11     <LBMETHOD/>
12     <LBANMETH/>
13     <LBFAST/>
14     <LBTPPT/>
15     <SUPPLB.LBPTFL>Y</SUPPLB.LBPTFL>
16     <SUPPLB.LBPDUR/>
17     <SUPPLB.LBRESTYP>MASS CONCENTRATION</SUPPLB.LBRESTYP>
18     <SUPPLB.LBRSLSCL>QUANTITATIVE</SUPPLB.LBRSLSCL>
19     <SUPPLB.LBTSTOPO/>
20     <SUPPLB.LBLLOD/>
21     <SUPPLB.LBTSTCND/>
22     <SUPPLB.LBMTHSEN/>
23     <Example_UCUM_Units>g/dL</Example_UCUM_Units>
24   </LOINC2SDTMMapping>
25   <LOINC2SDTMMapping MappingSource="CDISC" TargetSDTMDomain="LB">
26     <LBTESTCD NCICode="C51946">RBC</LBTESTCD>
27     <LBTEST NCICode="C51946">Erythrocytes</LBTEST>
28     <LBORRESU_Example NCICode="C67308">10^12/L</LBORRESU_Example>
29     <LBPOS/>
30     <LBLOINC>789-8</LBLOINC>
31     <LBSPEC NCICode="C12434">BLOOD</LBSPEC>
32     <LBLOC/>
33     <LBMETHOD NCICode="C154794">AUTOMATED COUNT</LBMETHOD>

```

This file is meant to be extended and maintained by the user as a "cache file". It is solely used by another function "loinc2sdtmlb" (this time **without** "rws:" prefix) which also takes 2 arguments: the LOINC code, and the SDTM variable to be populated.

The difference between the pure RESTful web service function "rws:loinc2sdtmlb" and "loinc2sdtmlb" is that the latter first tries to retrieve the information from the file "LOINC2SDTM_cached.xml", and if nothing found, then tries the RESTful web service. This also means that in the case that reading from the local file is faster than the RESTful web service (which is not always the case!), using the function "loinc2sdtmlb" instead of "rws:loinc2sdtmlb" can make sense.

Now, how can the user further populate this file with other mappings in XML form? One of the possibilities is to submit the LOINC code in the browser to the RESTful web service.

Extending the local LOINC-CDISC Mappings XML File

ODM File with Clinical Data:

Existing File with LOINC-CDISC Mappings in XML:

D:\SDTM-ETL_4_1\CDISC_CT\LOINC2SDTM_cached.xml

OID of the Item containing the LOINC Code:

LOINC2SDTM RESTful Web Service:

Edit

The location of the file with the existing mappings is already filled in automatically. We then must provide the location of a file with (new) ODM Clinical Data where the LOINC codes will be retrieved from. We also need to provide the identifier (OID of the item) that contains the LOINC codes in the file with clinical data.

For example:

Extending the local LOINC-CDISC Mappings XML File

ODM File with Clinical Data:

D:\SDTM-ETL\TestFiles\ODM1-3-1\CES_ClinicalData_simple_subject_Lab_LOINC_Coded.xml

Existing File with LOINC-CDISC Mappings in XML:

D:\SDTM-ETL_4_1\CDISC_CT\LOINC2SDTM_cached.xml

OID of the Item containing the LOINC Code:

I_LOINC

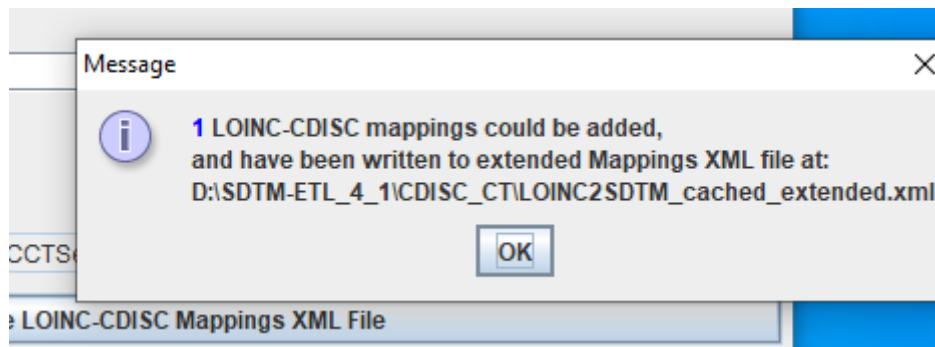
LOINC2SDTM RESTful Web Service:

Edit

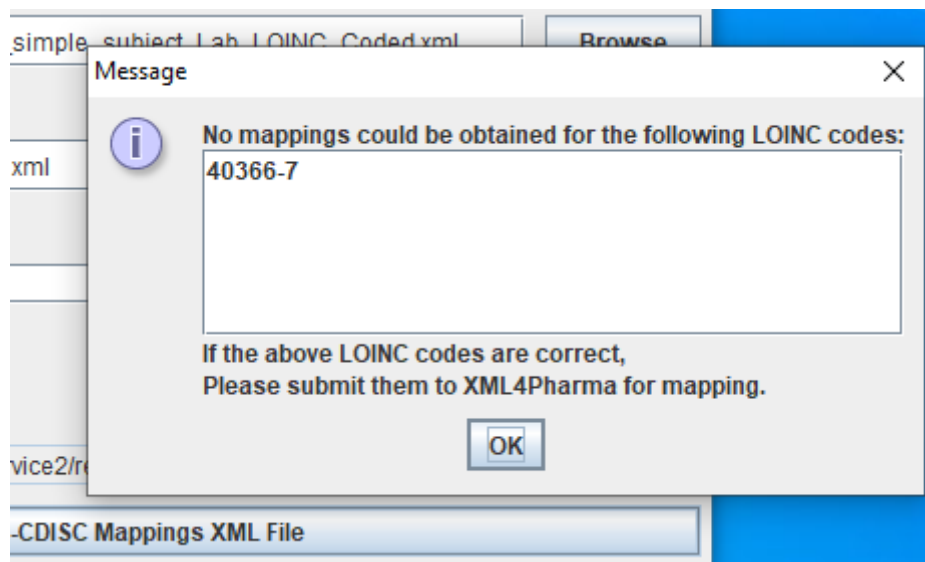
The window also displays the base of the RESTful web service. In normal cases, one would never have to change it, that is why we "protected" the field with an "Edit" checkbox, i.e. the base can only be changed after having checked the "Edit" checkbox. This has been done for customers who have a local installation of our RESTful web service on an own server.

When then clicking the "Extend the LOINC-CDISC Mappings XML file", the software will compare the contents of the mapping XML file with the LOINC codes retrieved from the ODM file.

If a code is found that is not in the mapping XML file, the RESTful web service is called, and the retrieved mapping XML is added. The combined results are then written to a new file "LOINC2SDTM_cached_extended.xml" in the same folder. The number of added mappings is then reported:



When there are codes found for which no mapping was obtained from the RESTful web service, this is also presented to the user. For example:



If you have such a case that there is no mapping yet, please submit the LOINC code or codes so that we can add it to the RESTful web service⁸.

We decided to not automatically overwrite the file "LOINC2SDTM_cached.xml" to allow the user to have inspection of the file "LOINC2SDTM_cached_extended.xml" first, before wanting to replace the original one.

⁸ The RESTful web service now already has almost 10,000 LOINC codes covered, but LOINC has over 90,000 codes ...

Additional features for "Save cleaned define.xml"

We further extended the features for saving a "cleaned" version of the underlying define.xml. This can be especially important, when, at the end of a project, one want to generate a "submission-ready" define.xml.

Two additional checkboxes have been added:

Save clean Define.xml

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.

**Template SDTM Domains will be removed,
Only study-specific domains are retained.
Sticky Notes will be removed.
Subject Global Domain will be removed.**

Order dataset definitions alphabetically within each SDTM class

Remove SDTM Variables that do not have a mapping provided

Remove Mapping Scripts from the define.xml

Remove Method References and Definitions from all Variables that are not marked as 'Derived' When checked, all mapping scripts (as 'FormalExpression'), will be removed from the define.xml, but the method description will be retained

Move non-standard SDTM Variables to SUPP--

Move Comment Variables to Comments (CO) Domain

Automatically generate multiple COVAL variables when COVAL length > 200 characters

Move Relrec Variables to Related Records (RELREC) domain

Store links to external files and documents as relative links

Unreferenced elements for which the checkbox is checked will be removed.

<input type="checkbox"/> ItemDef - SDTM Variables	Total: 1015 - Unreferenced: 1000	Show unreferenced
<input type="checkbox"/> CodeList - Controlled Terminology	Total: 712 - Unreferenced: 654	Show unreferenced
<input type="checkbox"/> MethodDef - Mappings	Total: 33 - Unreferenced: 15	Show unreferenced
<input type="checkbox"/> ValueListDef - Value lists	Total: 0 - Unreferenced: 0	Show unreferenced
<input type="checkbox"/> CommentDef - Comments	Total: 0 - Unreferenced: 0	Show unreferenced

OK Cancel

"Remove Mapping Scripts from the Define.xml" and:

"Remove Method References and Definitions from all Variables that are not marked as 'Derived'"

<input checked="" type="checkbox"/>	Remove Mapping Scripts from the define.xml
<input checked="" type="checkbox"/>	Remove Method References and Definitions from all Variables that are not marked as 'derived'

For the first "Remove Mapping Scripts" from the define.xml, all the "FormalExpression" elements from "MethodDef" elements will be removed, but the "Description" within "MethodDef" will be retained.

For example, when not checking the checkbox "Remove Mapping Scripts", a MethodDef in the define.xml may look like:

```

<MethodDef Name="Computation method for LBORNHRHI"
           OID="IMP.CES:LB.46.LB.LBORNHRHI"
           Type="Computation">
  <Description>
    <TranslatedText xml:lang="en">LBORNHRHI taken from CRF "RBC High" in the case of RBC, "WBC High" in the case of WBC</TranslatedText>
  </Description>
  <FormalExpression Context="SDTM-ETL"># Mapping using ODM element ItemData with ItemOID I_LB_RBC_HI
# Generalized for all StudyEvents
$RBCLBORNHRHI = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/ItemData[@ItemOID='I_LB_RBC_HI']/@Value);
# Mapping using ODM element ItemData with ItemOID I_LB_WBC_HI
# Generalized for all StudyEvents
$WBCLBORNHRHI = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/ItemData[@ItemOID='I_LB_WBC_HI']/@Value);
if($LB.LBTESTCD == 'RBC') {
  $LB.LBORNHRHI = $RBCLBORNHRHI;
} elseif($LB.LBTESTCD == 'WBC') {
  $LB.LBORNHRHI = $WBCLBORNHRHI;
} else {
  $LB.LBORNHRHI = 'UNKNOWN';
}
</FormalExpression>
</MethodDef>

```

but with the checkbox "Remove Mapping Scripts", the "FormalExpression" part is removed, and the MethodDef reduces to:

```

<MethodDef Name="Computation method for LBORNHRHI"
           OID="IMP.CES:LB.46.LB.LBORNHRHI"
           Type="Computation">
  <Description>
    <TranslatedText xml:lang="en">LBORNHRHI taken from CRF "RBC High" in the case of RBC, "WBC High" in the case of WBC</TranslatedText>
  </Description>
</MethodDef>

```

The second checkbox "Remove Method References and Definitions from all Variables that are not marked as 'Derived'" will remove all method references from all ItemDef elements (MethodOID attribute) and the corresponding MethodDef elements, unless the item has been marked as "derived" using the "def:Origin" element.

The reason for this is that most reviewers expect only to have a method definition in the define.xml for those variables that are "derived". Typical examples are the -DY variables in the "Findings" domains. So, for example, for VSORRES we may have that the value is taken from the CRF, in which case the ItemRef and ItemDef in the define.xml are:

```

<ItemRef ItemOID="VS.VSORRES"
         Mandatory="No"
         MethodOID="IMP.CES:VS.46.VS.VSORRES"
         OrderNumber="12"
         Role="Result Qualifier"/>

```

```

<ItemDef DataType="text" Length="80" Name="VSORRES" OID="VS.VSORRES">
  <Description>
    <TranslatedText xml:lang="en">Result or Finding in Original Units</TranslatedText>
  </Description>
  <def:Origin Type="CRF">
    <def:DocumentRef leafID="LF.aCRF">
      <def:PDFPageRef PageRefs="25 27" Type="PhysicalRef"/>
    </def:DocumentRef>
  </def:Origin>
</ItemDef>

```

With the referenced method definition (with OID "IMP.CES:46:VS.VSORRES"⁹) being:

```

<MethodDef Name="Computation method for VSORRES"
  OID="IMP.CES:VS.46.VS.VSORRES"
  Type="Computation">
  <Description>
    <TranslatedText xml:lang="en">SDTM-ETL mapping for VS.VSORRES</TranslatedText>
  </Description>
  <FormalExpression Context="SDTM-ETL"># Mapping using ODM element ItemData with ItemOID I_DIABP
# Generalized for all StudyEvents
# Generalized for all ItemGroups within the Form
# Generalized for all Items within the ItemGroup
$VS.VSORRES = xpath(/StudyEventData/FormData[@FormOID='F_BASELINE']/ItemGroupData/ItemData[@ItemOID='I_HEIGHT' or
  </MethodDef>

```

When however the checkbox "Remove Method References and Definitions from all Variables that are not marked as 'Derived'" is checked, the "MethodOID" on the ItemRef is removed, as the "Origin" (in element def:Origin within ItemDef) is of type "CRF", so, not "derived". The MethodDef element then disappears from the define.xml. The ItemDef will however remain unchanged.

For the ItemRef, we then get:

```

<ItemRef ItemOID="VS.VSORRES"
  Mandatory="No"
  OrderNumber="12"
  Role="Result Qualifier"/>

```

For VSDY however, which is "derived", the "MethodOID" is retained,

```

<ItemRef ItemOID="VS.VSDY"
  Mandatory="No"
  MethodOID="IMP.CES:VS.46.VS.VSDY"
  OrderNumber="27"
  Role="Timing"/>

```

with the ItemDef being:

⁹ Remark again the values of OIDs have no intrinsic meaning. For example, it could as well have been "6bc0e28d-e37d-4e16-b1f0-eedd9fa2afd3".

```

<ItemDef DataType="integer" Length="8" Name="VSDY" OID="VS.VSDY">
  <Description>
    <TranslatedText xml:lang="en">Study Day of Vital Signs</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>

```

and the Method definition remains:

```

<MethodDef Name="Computation method for VSDY"
  OID="IMP.CES:VS.46.VS.VSDY"
  Type="Computation">
  <Description>
    <TranslatedText xml:lang="en">Derived from VSDTC and DM.RFSTDTC</TranslatedText>
  </Description>
  <FormalExpression Context="SDTM-ETL"># Mapping using ODM element ItemData with ItemOID I_VISIT
# Generalized for all StudyEvents
$VISITDATE = xpath(/StudyEventData/FormData[@FormOID='F_WEEK_1_2']/ItemGroupData[@ItemGroupOID='IG_COMMON'])
# Mapping using ODM element ItemData with ItemOID I_VISIT of Baseline visit
DM.RFSTDTC = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupData
$DATEDIFF = datediff($VISITDATE,$DM.RFSTDTC);
if($DATEDIFF >= 0) {
  $VS.VSDY = $DATEDIFF + 1;
} else {
  $VS.VSDY = $DATEDIFF
}</FormalExpression>
</MethodDef>

```

When however, both checkboxes "Remove Mapping Scripts" and "Remove Method References and Definitions from all Variables that are not marked as 'Derived'", are checked, the method definition is reduced to:

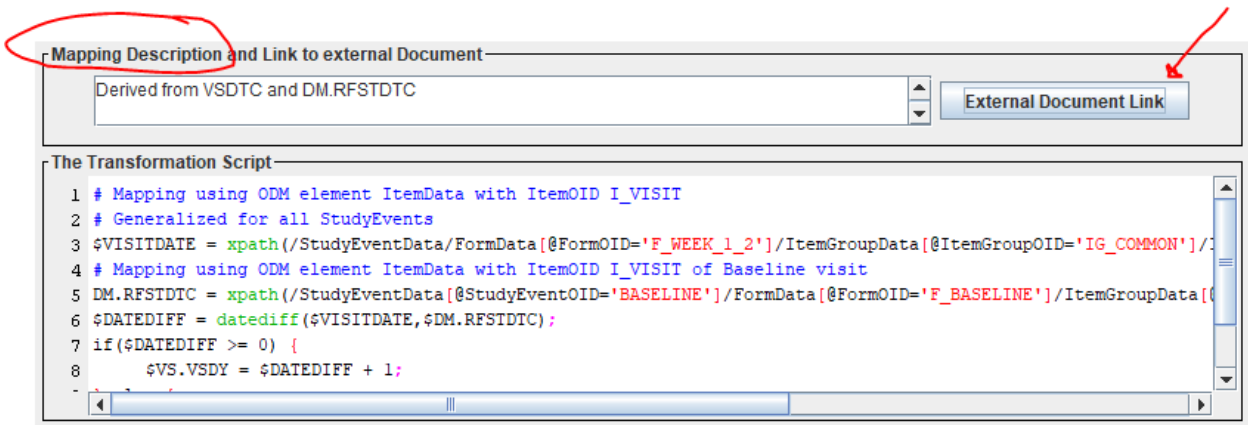
```

<MethodDef Name="Computation method for VSDY"
  OID="IMP.CES:VS.46.VS.VSDY"
  Type="Computation">
  <Description>
    <TranslatedText xml:lang="en">Derived from VSDTC and DM.RFSTDTC</TranslatedText>
  </Description>
</MethodDef>

```

and the define.xml will only contain "MethodDef" elements (and references to them) for variables of type "Derived".

Therefore, it is of utmost importance, than when one is preparing for a regulatory submission, the "Origin" is correctly set for each variable, and that for each variable mapping a suitable "mapping description" is supplied, as this is the only thing the reviewer will see when one has "cleaned" the define.xml using the above mechanism. This is done when generating or editing the mapping script by double-clicking the cell in the SDTM/SEND table:



Remark that one can also provide a link (e.g. with page number) to an external document (often the "Reviewers Guide"). This document must however already have been defined using the menu "Insert - Link to Supplemental Doc".

The "Origin" of each variable can (or better, should) be defined by selecting the variable in the table followed by using the menu "Edit - SDTM/SEND Variable Properties", and then using the button "Edit Origin":

Current Role:	Timing
<input type="checkbox"/> New Role	Timing
Current Role CodeList:	
<input type="checkbox"/> New Role CodeList	CL.C66742.Y - No Yes Response (Yes only) (text)
Current Origin:	NONE DEFINED YET
<input checked="" type="checkbox"/> Edit Origin:	Edit
Comment:	

Additional support for "Origin" for Define-XML 2.1

Define-XML 2.1 has a considerably different mechanism for "Origin" than in v.2.0.

When using the button feature "Edit Origin", for Define-XML v.2.0, the following wizard is displayed:

Designing/Updating Origin for Item: VSDTC ✕

? 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:

where one can choose between "Assigned", "Protocol", "Derived", "Electronic Data Transfer" and "CRF", where the lower part for providing page or named destination details will only be enabled when "CRF" is selected. For example, when "Protocol" is selected:

Designing/Updating Origin for Item: VSDTC ✕

? 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:

New in SDTM-ETL 4.1 is that one can also already fill the information even when the link to the

annotated CRF has not been provided yet.

Also remark that for SEND, the wizard will shown other choices, as e.g. there are no CRFs in non-clinical.

For Define-XML v.2.1, the wizard displayed is:

Designing/Updating Origin for Item: VSORRES

Origin type:

- Not Available
- Assigned
- Protocol
- Derived
- Predecessor
- Collected

Source type:

- Subject
- Investigator
- Vendor
- Sponsor

Origin description

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:

Title:

OK Cancel

One sees that there is an additional group "Source Type", which is however not displayed in the case of SEND (see section 4.3.2.3 in the Define-XML v.2.1 specification). The radio buttons of that group will be disabled/enabled depending on the choice for "Origin Type". For example, when "Protocol" is selected, only the radio button "Sponsor" is available and automatically selected:

Designing/Updating Origin for Item: VSORRES X

?

Origin type:

Not Available

Assigned

Protocol

Derived

Predecessor

Collected

Source type:

Subject

Investigator

Vendor

Sponsor

Also here, the lower part will only become available when "Collected" is selected. In the case of SEND, one will then usually also select "No Page details".

Also new in Define-XML v.2.1 is that when providing page details (a list of pages, a "named destination", or a range of pages), one can also add a title.

Important is that in the case of a list of pages, these must integers and be "blank separated", so a "comma-separated" list is invalid. This is currently checked by the software: if this rule is violated, a message will be displayed.

Extended support for SENDIG-DART 1.1

SDTM-ETL is also used a lot for the generation of [CDISC-SEND](#) datasets, this although non-clinical research does usually not use CRFs. So, SDTM-ETL already supported SEND-IG version 3.0 and 3.1 for a long time. Also SENDIG-DART 1.1 was already supported, but in view of the [recent FDA announcements for SENDIG-DART becoming required](#), we have further extended the support for SENDIG-DART.

Templates have been added for both Define-XML 2.0 as 2.1 for [SEND-DART v.1.1](#). This includes the SEND-DART specific domains SJ (Subject Repro Stages), IC (Implantation Classification), PY (Nonclinical Pregnancy Results), FX (Fetal Pathology Findings), FM (Fetal Measurements), and the Trial Design domains TP (Trial Repro Paths) and TT (Trial Repro Stages).

The "Trial Design Editor" allowing to enter design data into the datasets has also been extended with the SEND-DART trial design domains TP and TT.

It also includes a new "CDISC Notes" file that is used for assigning which variables are "required", which "expected" and which "permissible", and for use when the user wants to obtain more information about a specific variable by using the menu "View - SDTM/SEND CDISC Notes" (CTRL-H). For example, for the SENDIG-DART variable ICRPDY:

Implementation of the "Metadata Submission Guide v.2.0"

Before the publication of the "[Metadata Submission Guide v.2.0](#)" the relation between the SDTMIGs and the metadata in the define.xml was often not very clear. The SDTMIG and SENDIG development teams are not very "Define-XML savvy" and as a consequence, the information about how to document certain information in the define.xml, is often very short and even inaccurate. Sentences like "If the date/time of a transition element was not collected directly, the method used to infer the element start date/time should be explained in the Comments column of the Define-XML document" are even counter-productive, as **there are no columns in Define-XML**¹⁰.

Also, just as an example, the SDTMIGs do not explicitly state the "class" of each of the domains, one should interpret this from the header of the domain explanation table, such as:

LB – Specification

lb.xpt, Laboratory Test Results — Findings. One record per lab test per time point per visit per subject, Tabulation.

Variable Name	Variable Label	Type	Controlled Terms, Codelist or Format ¹	Role	CDISC Notes
STUDYID	Study Identifier	Char		Identifier	Unique identifier for a study.
DOMAIN	Domain	Char	LB	Identifier	Two-character abbreviation for the domain.

There is no sentence in the SDTMIG that states that the word or words after the long dash define the class the domain belongs to. Also, in the define.xml, the "class" designation must be in uppercase, so e.g. for LB as "FINDINGS" whereas it is "camel case" in the SDTMIG specification.

Therefore, it is of importance that the guiding of the "Metadata Submission Guide" is implemented in the software, which has been done.

¹⁰ The only place where one could find "columns" is in the VIEW of the define.xml that is created through the XSLT stylesheet.