SDTM-ETL 4.0 Preview of New Features

Author: Jozef Aerts, XML4Pharma

Last update: **2018-12-01**

Automated installation of new or additional SDTM/SEND templates

When new versions of the SDTM-IG or SEND-IG are published, an update of the software is not necessary anymore. The initial screen allowing to select which standard (SDTM or SEND) is used and which IG version is now steered by an external file "SDTM_SEND_standards.xml". The contents look like:

```
<SDTM-ETL_SDTM_SEND_standards>
    <Standard Name="SDTM-IG" Version="3.1.2" SDSVersion="1.2" DefineXMLVersion="2.0">
        <TemplateFile xlink:href="define_2_0/define_template_SDTMIG_3.1.2.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SDS_v1.2.xml"/>
    </Standard>
    <Standard Name="SDTM-IG" Version="3.1.3" SDSVersion="1.3" DefineXMLVersion="2.0">
        <TemplateFile xlink:href="define_2_0/define_template_SDTMIG_3.1.3.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SDS_v1.3.xml"/>
    </Standard>
    <Standard Name="SDTM-IG" Version="3.2" SDSVersion="1.4" DefineXMLVersion="2.0" IsDefault="Yes">
        <TemplateFile xlink:href="define_2_0/define_template_SDTMIG_3.2.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SDS_v1.4.xml"/>
    </Standard>
    <Standard Name="SEND-IG" Version="3.0" SDSVersion="1.4" DefineXMLVersion="2.0">
        <TemplateFile xlink:href="define_2_0/define_template_SENDIG_3.0_final.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SEND_v3.0.xml"/>
    </Standard>
    <Standard Name="SEND-IG" Version="3.1" SDSVersion="1.5" DefineXMLVersion="2.0">
        <TemplateFile xlink:href="define_2_0/define_template_SENDIG_3.1_final.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SEND_v3.1.xml"/>
    </Standard>
    <Standard Name="SEND DART" Version="1.1" SDSVersion="1.6" DefineXMLVersion="2.0">
        <TemplateFile xlink:href="define_2_0/define_template_SENDIG_DART.xml"/>
        <!-- TODO: is this separate? -->
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SEND_v3.1.xml"/>
    </Standard>
    <!-- Additions for define.xml 2.1 2019-09-02 -->
    <Standard Name="SDTM-IG" Version="3.1.2" SDSVersion="1.2" DefineXMLVersion="2.1">
        <TemplateFile xlink:href="define_2_1/define_template_SDTMIG_3.1.2.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SDS_v1.2.xml"/>
    </Standard>
    <Standard Name="SDTM-IG" Version="3.1.3" SDSVersion="1.3" DefineXMLVersion="2.1">
        <TemplateFile xlink:href="define_2_1/define_template_SDTMIG_3.1.3.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SDS_v1.3.xml"/>
    </Standard>
```

Each entry "Standard" in this file points to a template for that standard for a specific version of the define.xml (2.0 or 2.1 – see further), and (when available) a file with the "CDISC notes". This means that when a new version of e.g. the SDTM-IG is published, no software update is needed, but only the new template files need to be installed, and a link to it added to the file "SDTM_SEND_standards.xml" file. These files will then be provided by XML4Pharma at a low price.

For example, we added "SEND DART 1.1" templates in the current release in this way. Of course, even when choosing "plain" SEND (3.0 or 3.1), one can always add the "SEND DART" afterwards and merge it with the originally loaded template.

Implementation of SDTM-IG v.3.3

Immediately after it was published, we implemented the SDTM-IG v.3.3 and its corresponding SDTM model v.1.7. When starting up, SDTM-IG v.3.2 however remains the default, as the regulatory authorities such as the FDA do not accept SDTM-IG v.3.3 yet (status November 2018).

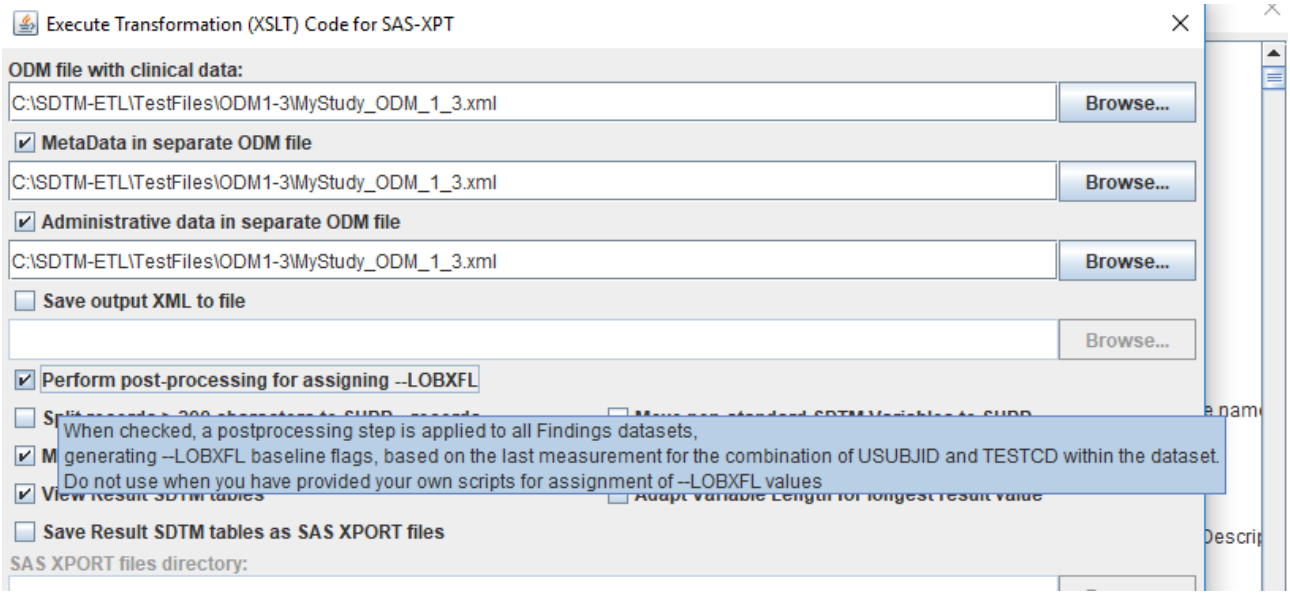
Automated generation of –LOBXFL variable values for SDTM-IG v.3.3

SDTM v.1.7 / SDTM-IG v.3.3 introduces a new set of variables, all ending with "-LOBXFL". These "flag" variables represent the "Last Observation Before Exposure" and are used in almost all Findings domains.
We regret that these variables have been added, as these are essentially "analysis" variables (like other baseline flags), and "analysis variables" essentially should not be present in SDTM (they should go into ADaM). These variables seem to have been added on request of FDA reviewers, who are themselves not able to perform the necessary derivations to get "last observation before treatment" data points. These derivations are however very easy to perform.

But things are as they are …

As the derivation of values for –LOBXFL is based on that the first exposure is known, the derivation may need an additional run over all generated data. Such an "extra run" has now been implemented in the software.

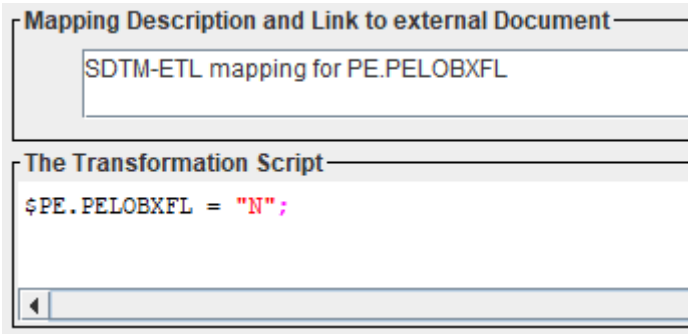
In the stage where the mappings are executed (both for SAS-XPT as for Dataset-XML format), an extra checkbox is presented:



The checkbox "Perform post-processing for assigning –LOBXFL" will indeed initiate a post-processing step, in which the data is analyzed and the very last observation before the first treatment, based on RFXSTDTC ("Date/Time of First Study Treatment") in DM ("Demographics") is assigned for each unique test. When RFXSTDTC is a date only (without part), and the observation has a "datetime of collection" that is on the same date of the first exposure, then it is assumed that the observation is before the treatment. See further how to change this behavior.

Please do remark, that even when this checkbox is checked, you will need to provide a "default" mapping for each –LOBXFL variable, as otherwise the system assumes that the variable is not to be

populated and will even not appear in the output files.
So, you will probably want to set the default value to "N" (meaning "no")



Stating that the initial value (before the post-processing step) is the "no" value.
Remark that in case of output to Dataset-XML, it is not allowed to set the "default" value to the empty value (e.g. $VSLOBXFL = '') as this will not create LOBXFL data points in the intermediate file, as in Dataset-XML, empty values are just not "printed out". In case you only want to generate XPT files, the use of setting the default value to the empty value is unproblematic.

Remark also that the SDTM-IG explicitly allows to have an "N" value for -LOBXFL variables.

An example outcome of using the "post-processing", here for the PE dataset, is:

PETEST	PEORRES	PESTRESC	PESTRESN	PESTRESU	PESTAT	PEREASND	PELOBXFL	VISIT	VISITNUM	PEDTC
Head, Nec	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Eyes, Ear	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Chest	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Lungs	Abnormal	MILD WHEE	.				Y	VISIT0	1	2006-04-01
Heart	Abnormal	TACHYCARD	.				Y	VISIT0	1	2006-04-01
Lymph Nod	Abnormal	SLIGHTLY	.				Y	VISIT0	1	2006-04-01
Abdomen	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Anorectal			.		NOT DONE	The reaso		VISIT0	1	2006-04-01
Genitalia	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Skin	Abnormal	PET	.				Y	VISIT0	1	2006-04-01
Musculosk	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Neurologi	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Other	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Head, Nec	Normal	NORMAL	.					VISIT1	2	2006-05-12
Eyes, Ear	Normal	NORMAL	.					VISIT1	2	2006-05-12
Chest	Normal	NORMAL	.					VISIT1	2	2006-05-12
Lungs	Abnormal	MILD WHEE	.					VISIT1	2	2006-05-12
Heart	Abnormal	TACHYCARD	.					VISIT1	2	2006-05-12
Lymph Nod	Abnormal	SLIGHTLY	.					VISIT1	2	2006-05-12

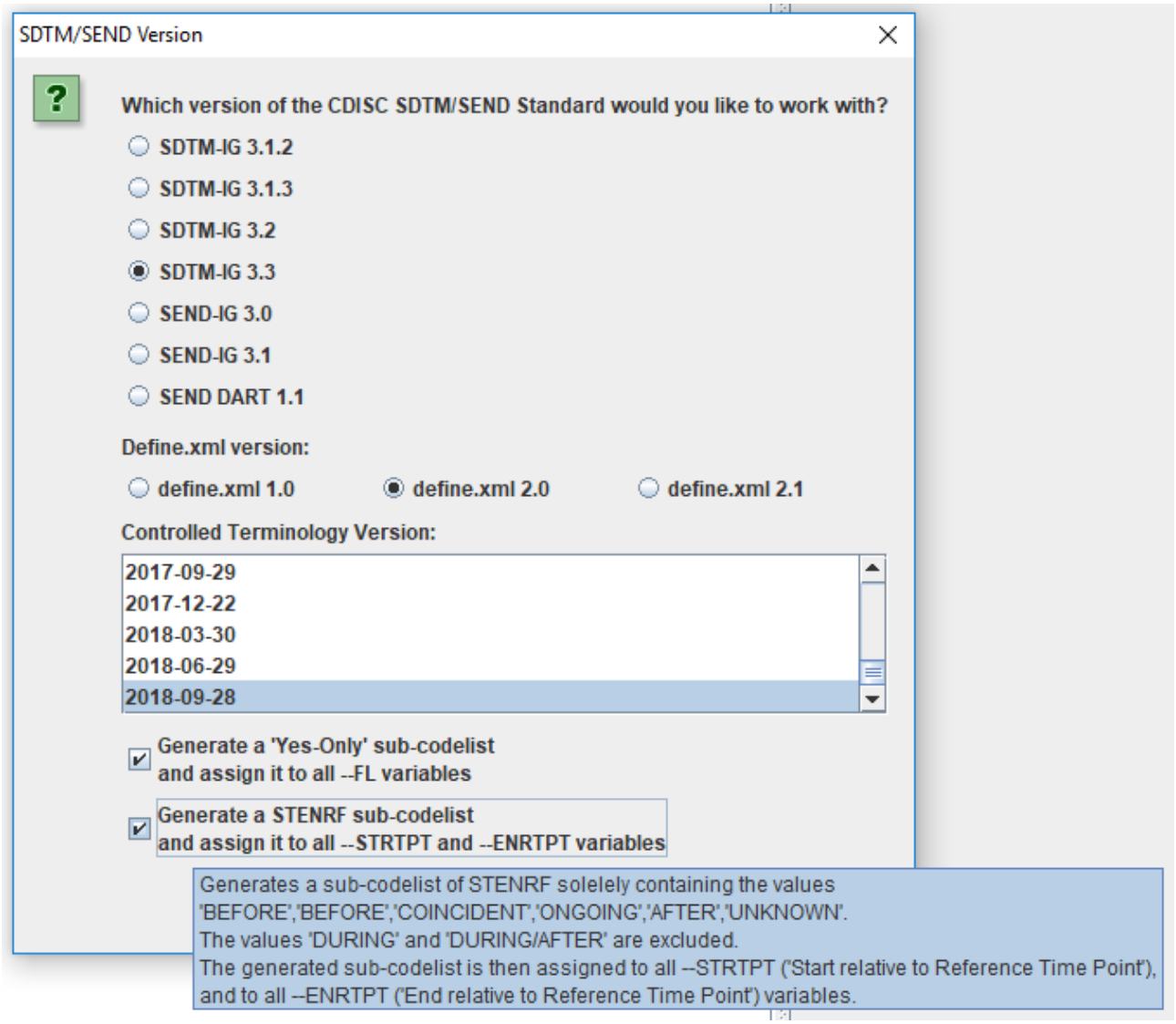
If you do not want this use this post-processing step that uses the above-mentioned assumptions, leave the checkbox "Perform post-processing for assigning –LOBXFL" unchecked, and provide the mapping for each "--LOBXFL" variable yourself.
As you might need the value of "RFXSTDTC", you may declare this variable as a "global variable" (see the manual "Creating and working with Subject Global Variables") and reference it in the mapping. This may e.g. allow you to assign the "last observation" flag to the last value that is at least one day before the first exposure date.

Automated creation of codelist subsets at startup time

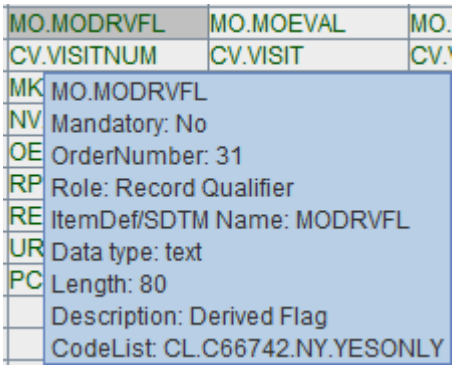
One of the problems with CDISC controlled terminology is that the SDTM-IG often constraints them further, just as text, so non-machine-readable. Even though many implementers have asked CDISC to also publish subsets of such codelists, the CT team keeps refusing this.
A famous example is the NY (Yes-No codelist) which contains "N", "Y", "NA", and "U" as allowed values. For "flag variables", variables ending with "FL" however, only the "Y" value is allowed (by constraint). The CDISC-CT team however still refuses to also publish a "Yes only" codelist.
Another example in the "STENRF" (Relation to Reference Period) codelist, which has 7 allowed values. This codelist is also assigned by the SDTM-IG to the –STRTPT (Start Relative to Reference Time Point) and -ENRTPT (End Relative to Reference Time Point) variables, but in that case, the values "DURING" and "DURING/AFTER" are forbidden. Also here, the CT team refused to publish a subset.

There are two ways this can be managed. The first is to add such subset codelists to each of the set of codelists quarterly by CDISC. This is the strategy followed by Pinnacle21 (but it often takes them several months to do so, at least for the "Community" version). It is however questionable whether editing by CDISC published codelists is morally acceptable. We don't think so.
The second possibility is to generate such a "subset-codelist" "on the fly" when loading the CDISC-CT as published by CDISC, which does not rely on editing existing codelists.
This is the strategy we have chosen.

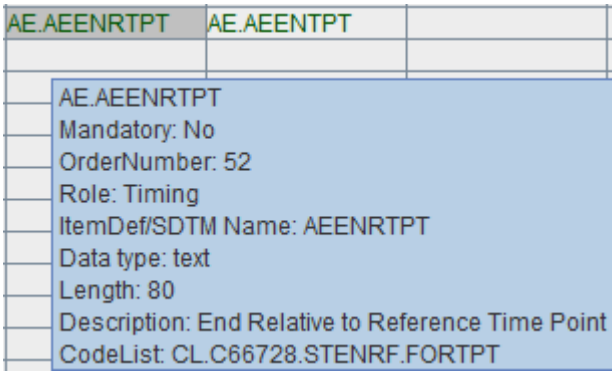
When the choice of the SDTM/SEND version is due, the dialog now shows an extra panel with two checkboxes, the first asking whether a "Yes only" codelist should be generated, and automatically assign this codelist to all "flag" variables, all ending with "FL", like all the "baseline flags".
The second checkbox is about generating a subset of the "STENRF" codelist to the 5 values(without "DURING" and "DURING/AFTER", and automatically assign this subset-codelist to all "-STRTPT" and "-ENRTPT" variables:



The result when both these checkboxes are checked, is for example, for an -FL variable and for an -ENRTPT variable:



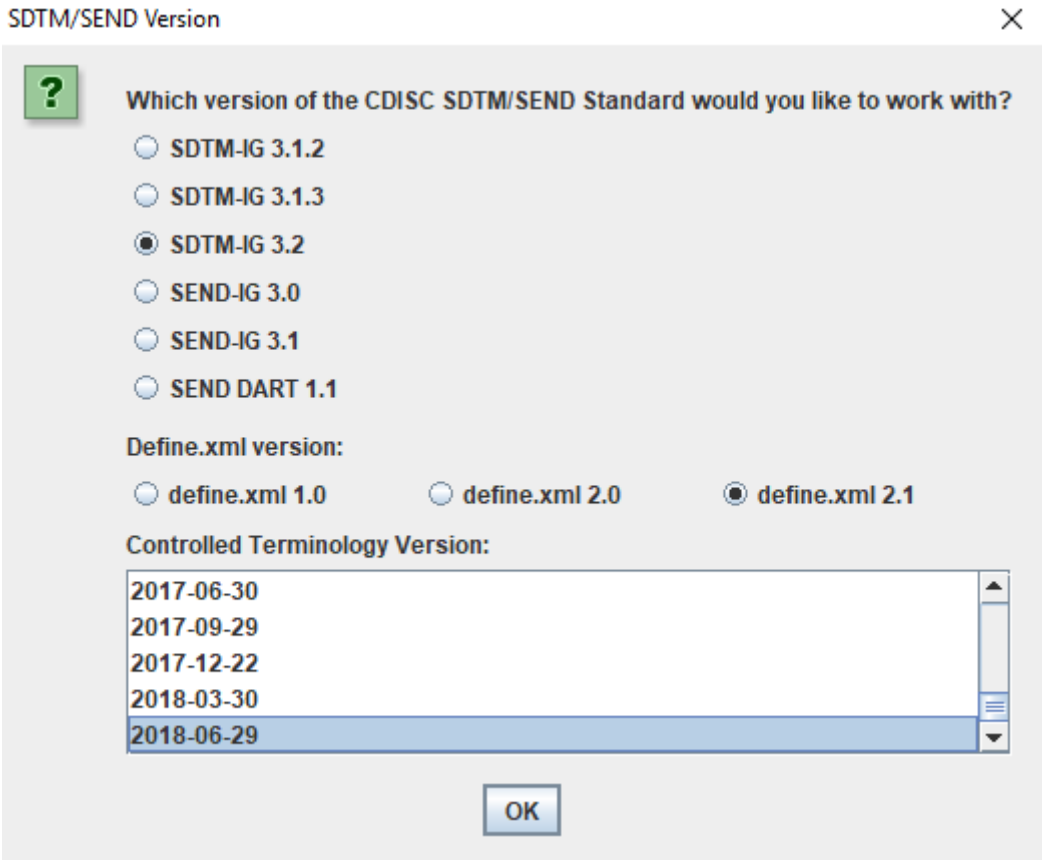
"Yes Only" codelist assigned to MODRVFL



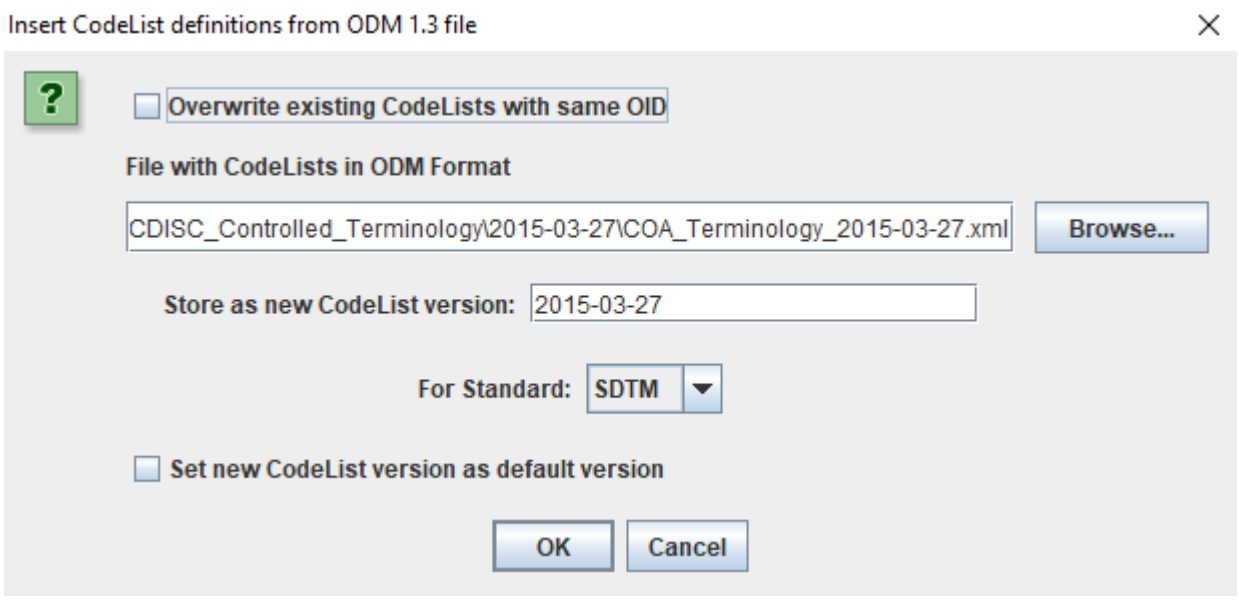
Subset-STENRF ("for TPT") codelist assigned to AEENRTPT

Define.xml 2.1 Implementation

The major feature is that define.xml 2.1 is fully implemented.
As the FDA however does not accept define.xml 2.1 yet, version 2.0 remains the default:



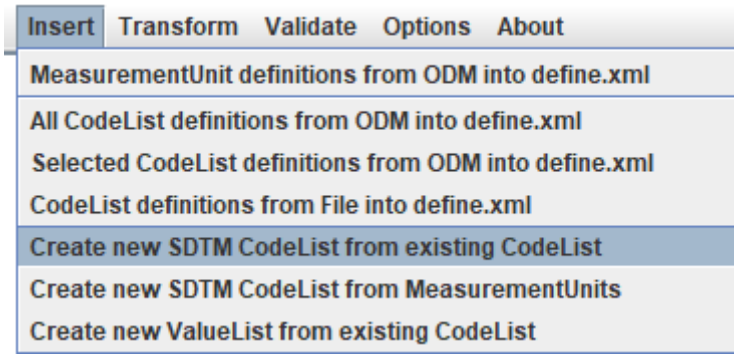
One of the new issues in define.xml is that for each domain, it can be defined at the level of the domain/dataset which version of the SDTM-IG (or extension of it, like "associated persons") is used. Also, and that is very useful, for each codelist loaded, it can be defined what the codelist version is.
For example, if additional codelists are loaded from file, using the menu "Insert – CodeList definitions from File", and a codelist (as ODM-XML) is selected, the following dialog appears (example):



The newly loaded set of codelists can then be set as the "default version", or the already loaded codelists can remain the "default".

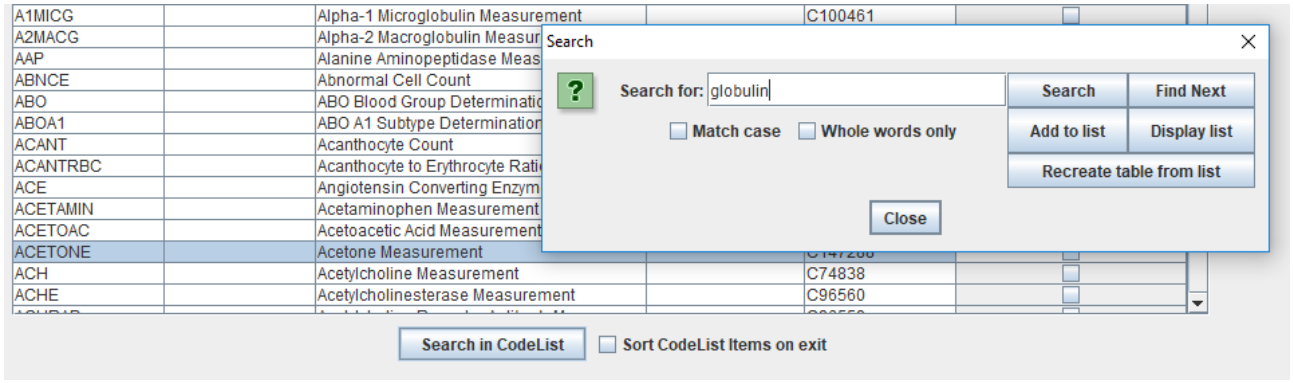
Improved "New codelist from existing CodeList" dialog

The dialog that appears when using the menu "Insert – New CodeList from existing CodeList" has obtained a lot more functionality that makes it even more easy to generate sub-codelists:

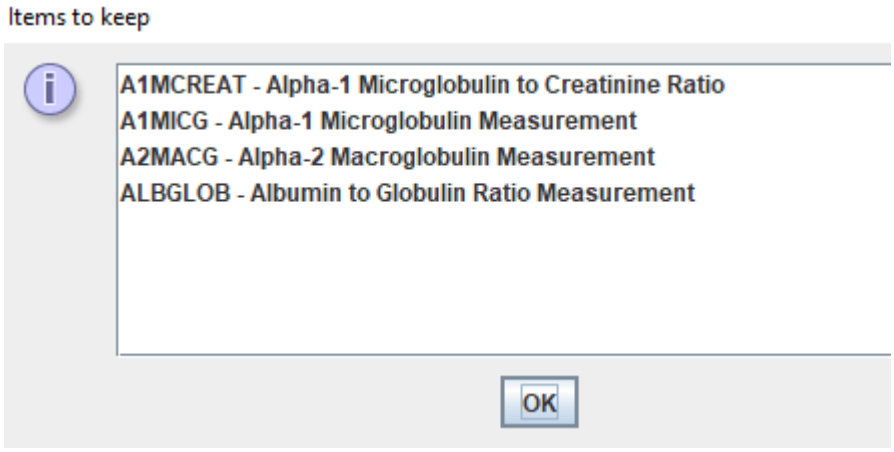


It was already possible to use the "Search" button to find a specific codelist, and then "click through" (i.e. a "Next" functionality) until the desired codelist to be sub-setted is found, but one can now also use the new "Search in CodeList" button, to find entries in the existing codelist.

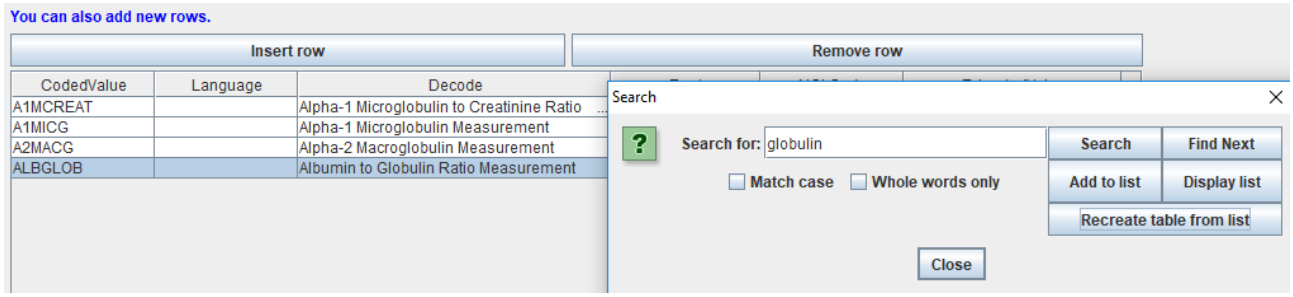
For example, when the "Search in CodeList" button is clicked and "globulin" is set as the search term in the dialog that pops up:



one can then search for all occurrences of "globulin" in that codelist using the "Search" and "Find Next" buttons. Upon each "hit", one can then use the button "Add to list", to add the found entry to the new codelist. With the "Display list" button, one can then display the list of those added, e.g.:



When the button "Recreate table from list" is then clicked, only the ones from the "additions" are added to the new codelist:



One can then still add or remove items to/from the newly created codelist.
Also remark that the NCI code appears in the table when the starting codelist is a CDISC codelist.

This newly added "wizard" makes it even more easy to quickly generate subsets of codelists.

When the new codelist is then added (to the underlying define.xml), the user is also asked whether he/she wants to store this newly generated codelist to file for future reuse (or even for use in study design!). Such a generated file can then be read in again in mappings for other studies using the menu "Insert – CodeList Definitions from File into define.xml".

Also new is that when a new term is added manually (using "Insert Row"), the checkbox "Extended Value" is checked automatically. It can of course be unchecked again when the manually added

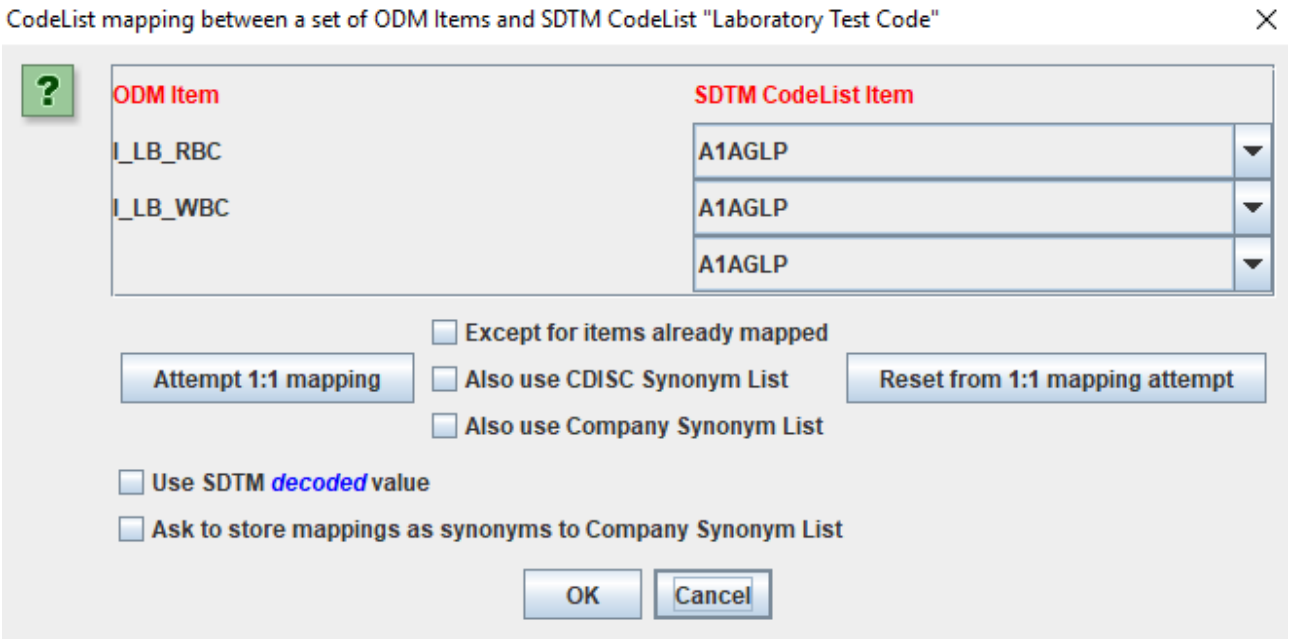
term is an official CDISC term for that codelist. In that case, also the NCI code must be entered.

Additionally, when a cell in the SDTM/SEND table is selected, and a codelist is already associated with that SDTM/SEND variable (e.g. "LBTESTCD"), then the codelist is pre-selected when using the menu "Insert – Create new CodeList from existing CodeList".

CodeList-CodeList mapping improved

Codelist to codelist mapping can sometimes be tedious because CDISC controlled terminology uses "jargon" that is not known to the mapper. For example, not anyone may know that "WBC" (the CDISC code) means "White Blood Count" which is a also knows as "Leukocytes".

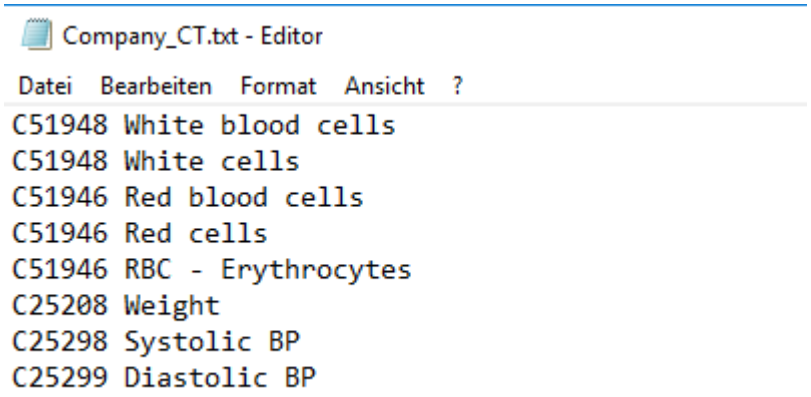
When using the codelist to codelist mapper wizard, a few new options are now available.



One sees two additional checkboxes:

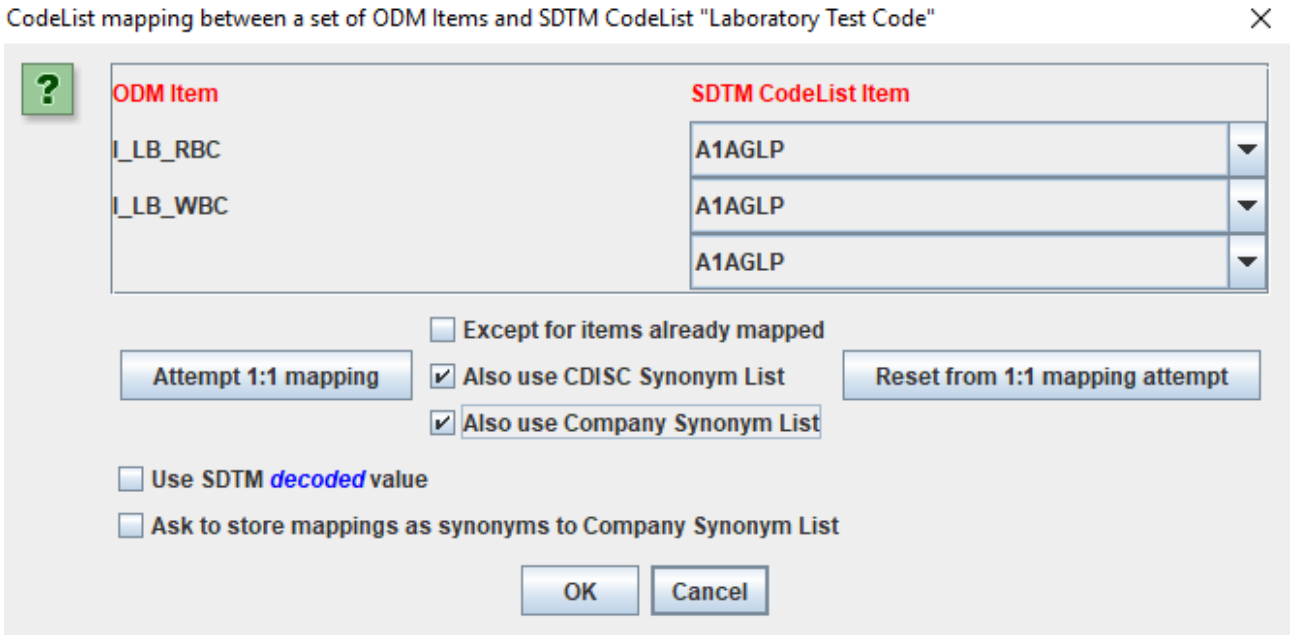
- Also use CDISC Synonym List
- Also use Company Synonym List

Using the "CDISC Synonym List", when allowing the system to attempt a 1:1 mapping, will also look into the published CDISC synonyms. One can also use a "Company Synonym List" containing the CDISC-NCI codes and locally used synonyms. This list is expected to be located in the folder "Company_CT" and the file name being "Company_CT.txt". Its contents look like:

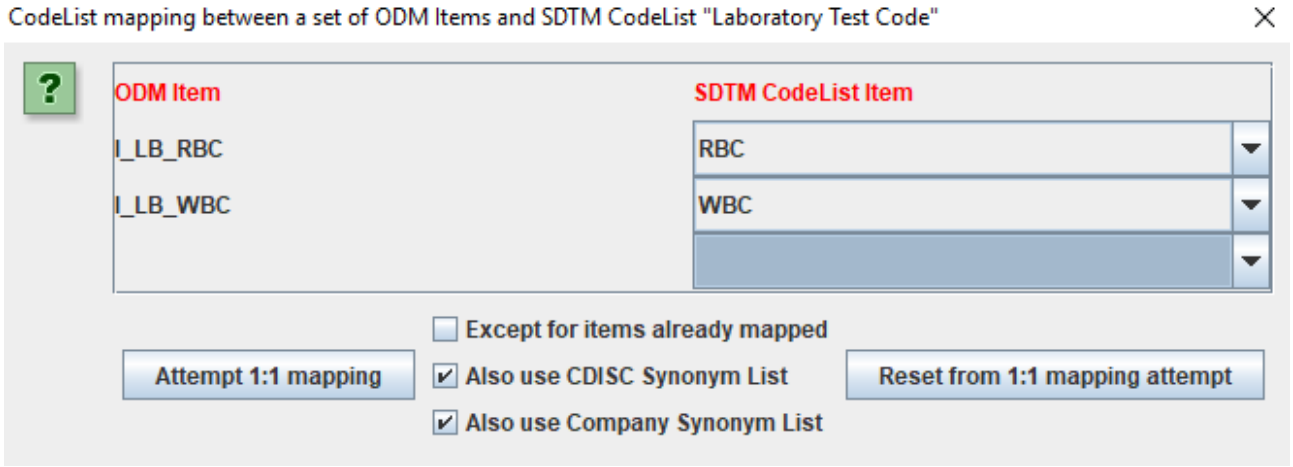


Using the checkbox "Ask to store mappings as synonyms to Company Synonym List" allows to extend this list with new mappings to "locally" used terms.

In our case, using both above mentioned options (checking both checkboxes):

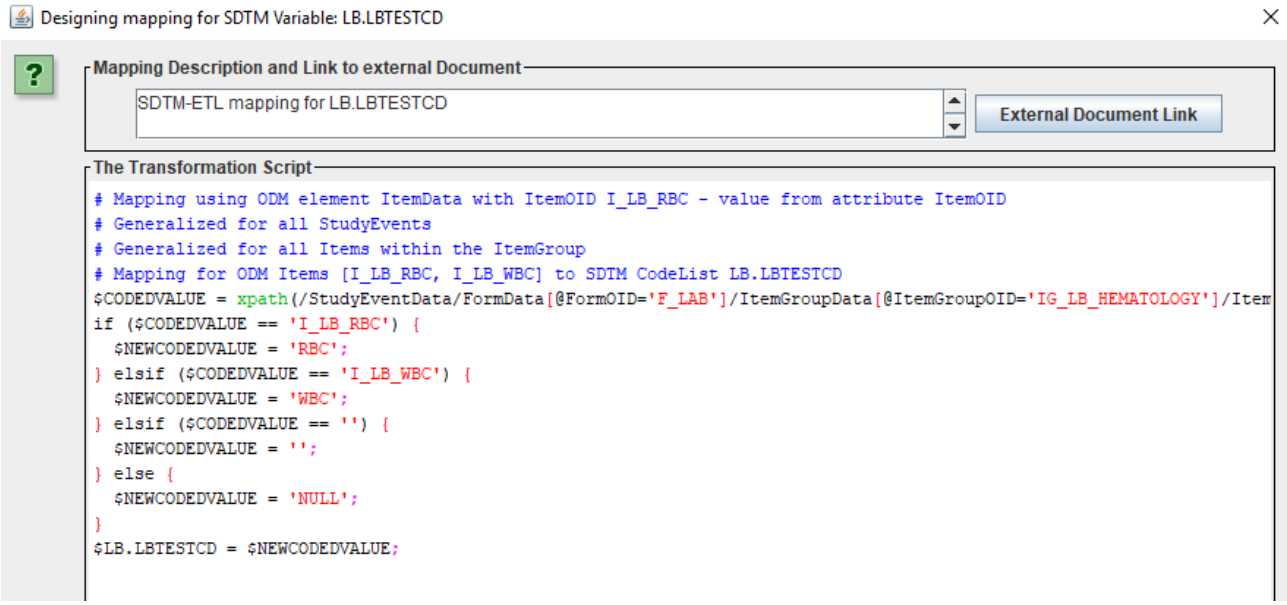


and clicking "Attempt 1:1 Mapping" leads to:



which in this case is 100% correct.
If one does not like what the system proposes, one can always revert and go to a "manual" mapping, by clicking the "Reset from 1:1 mapping attempt".

The automatically generated script then is:

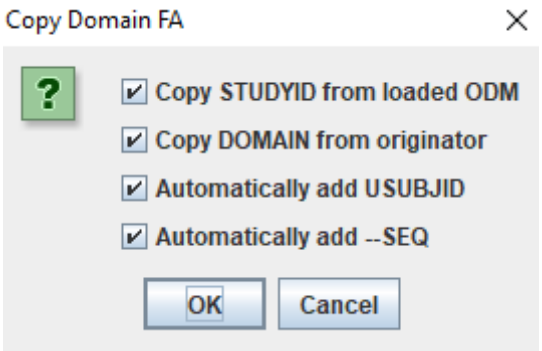


Working with the "Findings About" domain

When SDTM was first developed, there were about 20 domains, and everybody (except a few) was expecting that this number would not increase very much. In the latest SDTM-IG version 3.2 there are over 50 domains, and even that was not sufficient, as it did not cover well the use case of findings that are related to interventions and to events.

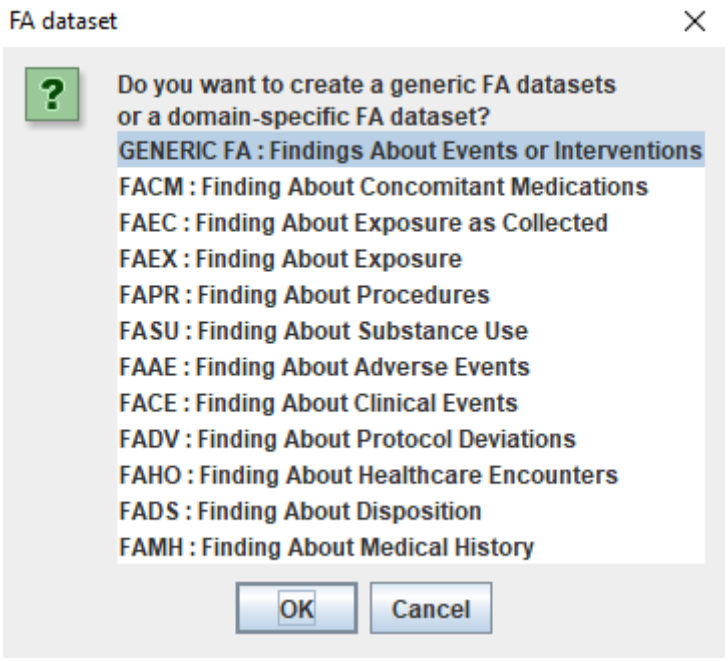
For this case, the FA domain "Findings about Events and Interventions" was created.
The SDTM-IG lets implementers the choice between a single FA dataset, and a series of "splitted" datasets for findings about different existing SDTM "Events" and "Interventions" domains. In the latter case one may e.g. have an "FAMH" dataset ("Findings About Medical History") and an "FAAE" ("Findings About Adverse Events") dataset.
The name "splitted" is confusing, as one will usually not split a single FA dataset in datasets such as FAMH and FAAE, but one will create a single "FA" instance for "Medical History" and one for "Adverse Events" right from the start.

As of SDTM-ETL v.4.0, when one "drags-and-drops" the FA row to the bottom (after the last template domain row), or when one selects the FA row from the template and then uses the menu "Edit – Copy Domain" followed by "Edit – Paste Domain", the following dialog is displayed:

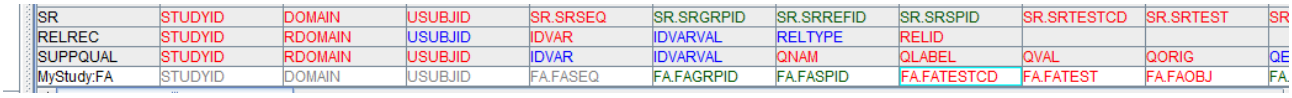


Which is the usual dialog asking whether mappings for STUDYID, DOMAIN, USUBJID, and FASEQ can be automatically generated (recommended when using standard ODM as the source).

After clicking OK, the following (new) wizard is displayed:



The software automatically looks up all "Interventions" and "Events" domains in the template, also sponsor-defined ones, and then creates a list of "FA domain specific" domains such as "FACM", FAEC, … The first entry however is "Generic FA", meaning that a single FA instance will be created, which should then contain all "Findings About" entries, independent of what the related domain is. If one selects "Generic FA", a single row is created at the bottom, from which one can start the mappings to a "generic" FA dataset:

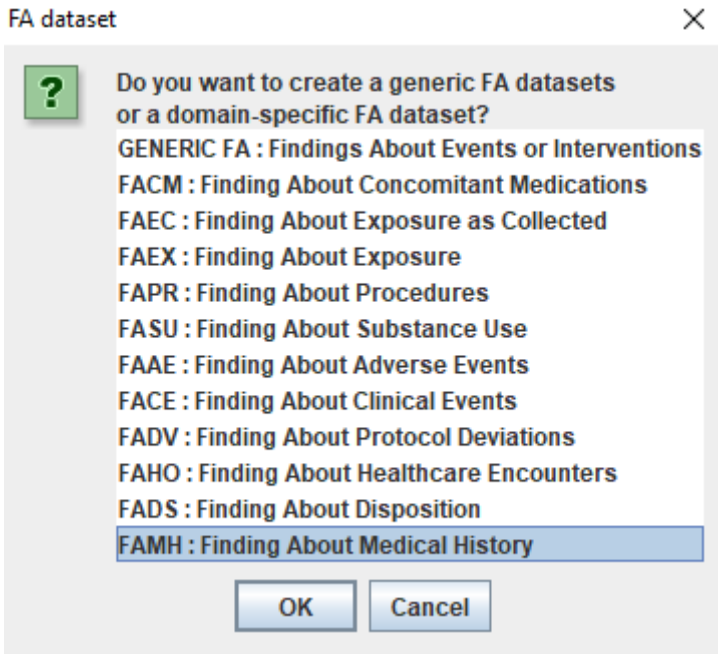


SR	STUDYID	DOMAIN	USUBJID	SR.SRSEQ	SR.SRGRPID	SR.SRREFID	SR.SRSPID	SR.SRTESTCD	SR.SRTEST	SR
RELREC	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	RELTYPE	RELID			
SUPPQUAL	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	QNAM	QLABEL	QVAL	QORIG	QE
MyStudy:FA	STUDYID	DOMAIN	USUBJID	FA.FASEQ	FA.FAGRPID	FA.FASPID	FA.FATESTCD	FA.FATEST	FA.FAOBJ	FA.

This is the simple case and does not deviate from the usual case for any other domain.

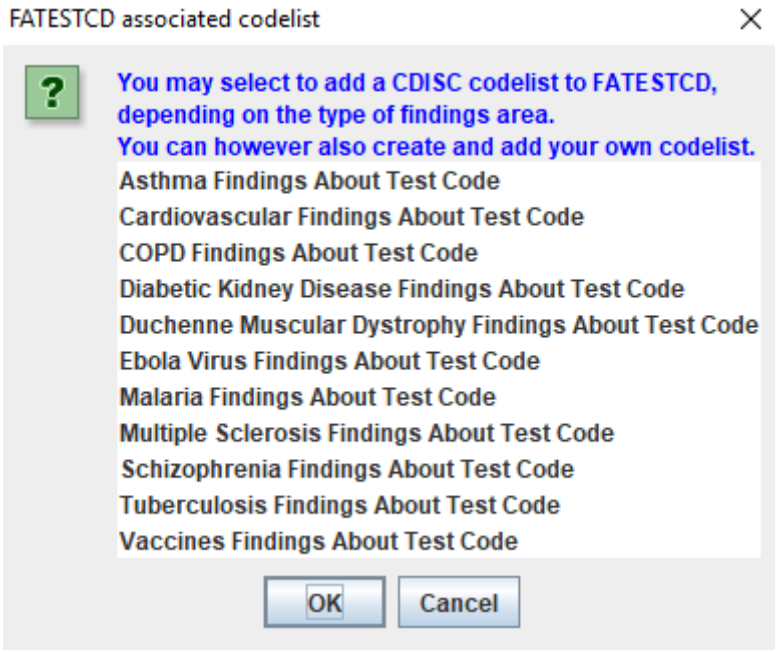
One can then start generating the mappings in the usual way.

In a number of cases however, one will want to generate "domain-specific" FA datasets, such as FAMH ("Findings About Medical History") or FACE ("Findings About Clinical Events"). For example, when one wants to create an FAMH dataset instance, one would choose:



meaning that an FAMH dataset will be created.

After clicking OK, a new wizard is displayed:



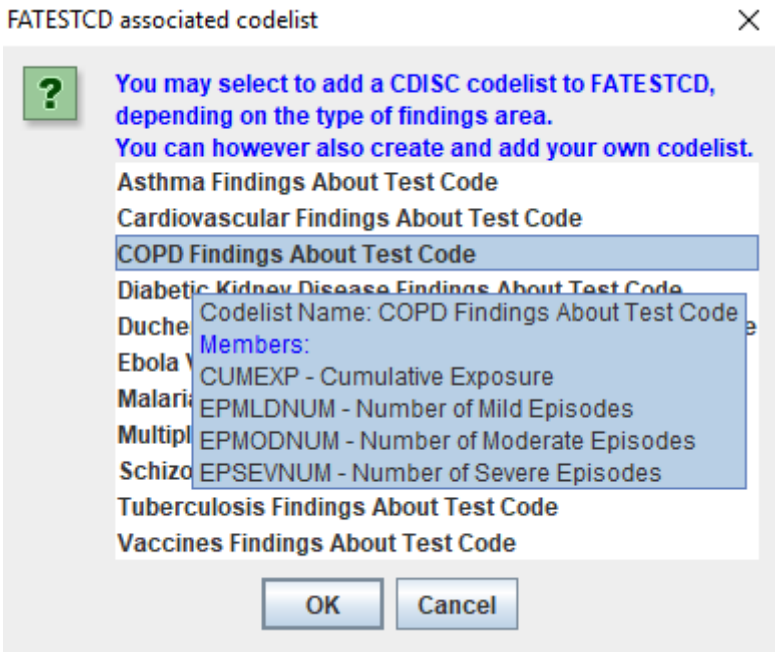
In the SDTM-IG 3.2, it is stated that the **codelist** "FATESTCD" needs to be used for the variable FATESTCD:

FASPID	Sponsor-Defined Identifier	Char		Identifier	Sponsor-defined reference explicit line identifier or d Line number on a CRF.
FATESTCD	Findings About Test Short Name	Char	(FATESTCD)	Topic	Short name of the measur can be used as a column n horizontal format. The val nor can it start with a num characters other than lette

However, the newer versions of CDISC Controlled Terminology (CDISC-CT) does NOT contain this codelist. Closer inspection of all the CDISC-CT of the last years show that it was deleted in September 2016.

Instead, a number of disease-specific "FATESTCD" codelists has been developed by the CDISC-CT team. All these obey to the pattern "–FATSCD", and the corresponding codelist for "FASTEST" obeys to the pattern "—FATS". For example, for "COPD Findings About Test Code", the identifier is "CPFATSCD" and the NCI code is "C122007".

In the wizard, when hovering the mouse over an entry, all the allowed values for that codelist is displayed. For example, for "CPFATSCD":

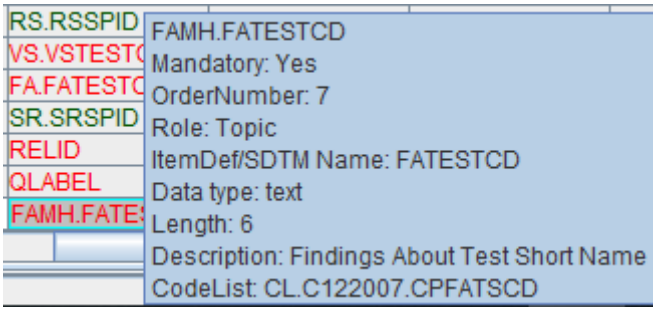


This allows to easier find a suitable codelist.
It is not mandatory to select an "—FATSCD" codelist, one can also not select anything (or click the Cancel button). One can then still later attach a codelist to both variables, and/or create a new one and attach it.
Also, when selecting an "—FATSCD" codelist, also the corresponding "—FATS" codelist will be loaded.

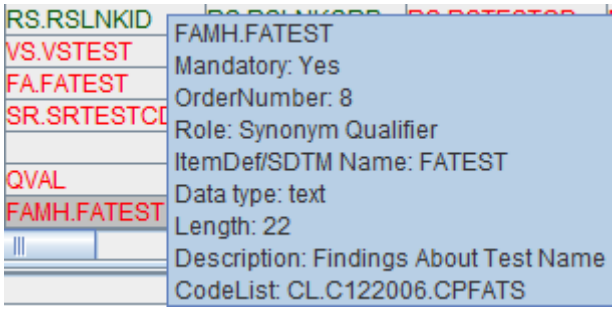
For example, when the "COPD Findings About Test Code" is selected, and OK is clicked, the FAMH instance is created:

RELREC	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL
SUPPQUAL	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL
MyStudy:FAMH	STUDYID	DOMAIN	USUBJID	FAMH.FASEQ	FAMH.FAGRPID

And the "COPD Findings About Test Code" is attached to FATESTCD:



And the corresponding codelist is attached to FATEST:

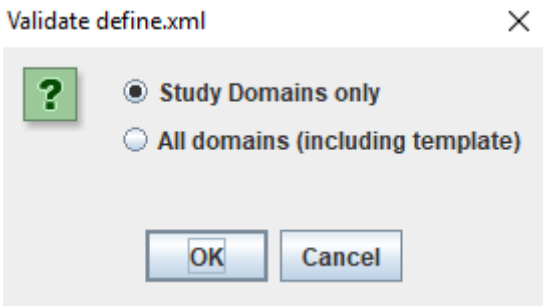


One can now start mapping as in the usual case, for example for all findings about medical history of COPD.

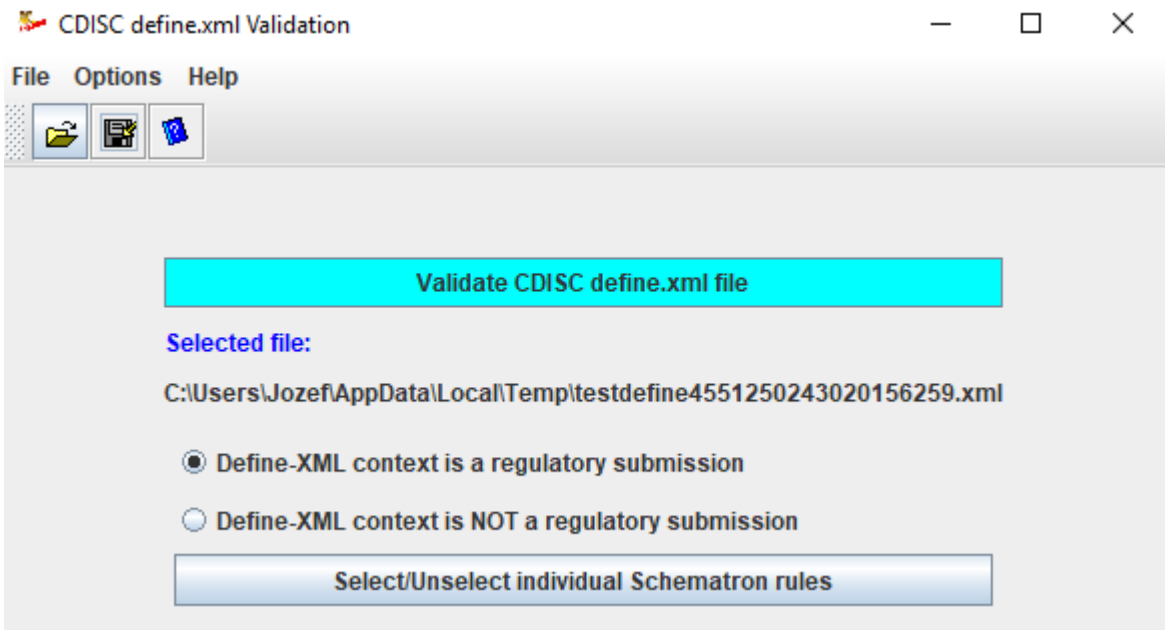
More details are found in the tutorial "Working with the "Findings About" domain

Validation of "Study-specific" domains only

With some new requirements on the define.xml, the template "define.xml" does not always contain the necessary information to go through validation without warnings. Therefore, we introduced the feature that when validation is requested, the user can choose between validating the "study-specific instances" of the domains/datasets (i.e. the non-template rows) and validating everything. In most cases, the user will want to validate the "study-specific domains" only:



It is then asked whether the define.xml is in the context of a regulatory submission. The reason for this is that the rules for define.xml are slightly different when it is used for a regulatory submission (e.g. SASName is required):



Most of the rules for define.xml are expression as so-called Schematron rules. Schematron is an open, international standard for validation of XML files developed by the World Wide Web Consortium (w3c). The Schematron rules for define.xml have been developed by members of the CDISC define.xml development team.
Using the button "Select/Unselect individual Schematron rules" allows to select / unselect individual rules for define.xml.

Non-Standard variables and domains in define.xml 2.1

In define.xml 2.0, there was no standardized mechanism to flag non-standard variables (NSVs). So, in SDTM-ETL 3.2 and earlier, we marked them with "Role=SUPPQUAL". This information was then used to color them differently in the SDTM/SEND table on the right side of the screen, and to "split them off" to a SUPPxx dataset at execution time (when the datasets are generated) when desired by the user, and to generate a "submission-ready" define.xml including the SUPPxx dataset descriptions.
Also, define.xml did not know a mechanism to mark sponsor-defined domains and datasets. By convention, the two-character name of such domains and datasets started with either "X", "Y" or "Z".

In define.xml 2.1, a new attribute was added to the "ItemRef" and "ItemGroupDef" elements: the "def:IsNonStandard" attribute. So, for example, the NSV "Completers Population Flag" (COMPLT) in the "Demographics" domain is defined by:

```
<ItemRef ItemOID="DM.DMDY" Mandatory="No" OrderNumber="28" Role="Timing"/>
<ItemRef ItemOID="DM.COMPLT"
         Mandatory="No"
         OrderNumber="29"
         Role="Record Qualifier"
         def:IsNonStandard="Yes"/>
<def:Class Name="SPECIAL PURPOSE"/>
```

This has the advantage that the user can now assign the value for the "Role" himself.

Similarly, for a sponsor defined domain/dataset:

```
<ItemGroupDef IsReferenceData="No"
              Name="XA"
              OID="CES:XA"
              Purpose="Tabulation"
              Repeating="Yes"
              def:ArchiveLocationID="Location.XA"
              def:IsNonStandard="Yes"
              def:Structure="One record per XA.XATESTCD per USUBJID">
   <Description>
      <TranslatedText xml:lang="en">Example Sponsor-defined domain</TranslatedText>
   </Description>
   <ItemRef ItemOID="STUDYID"
            Mandatory="Yes"
            MethodOID="IMP.CES:XA.47.STUDYID"
```

This is all done in the background in the software, there is nothing special or new the user has to do.

New functions in the mapping script language

Some new functions have been added to the scripting language, including for using RESTful web services (see next section).
Another useful function is the new "**alias()**" function, taking two arguments. The first argument is the OID of an Item, ItemGroup, Form or StudyEvent from the source data, the seconds the "Context" of the alias.

In ODM, the "Alias" element is used to define the synonym for an item in another context. Typically, this is used to assign the item a code, like a SNOMED-CT, RxNorm or LOINC code (the latter especially when the item represents a test).

For example, the protocol has stated that the "basic metabolic panel" lab tests must be executed. This consists of:

24320-4 Basic metabolic 1998 panel - Serum or Plasma

PANEL HIERARCHY (view this panel in the LForms viewer)

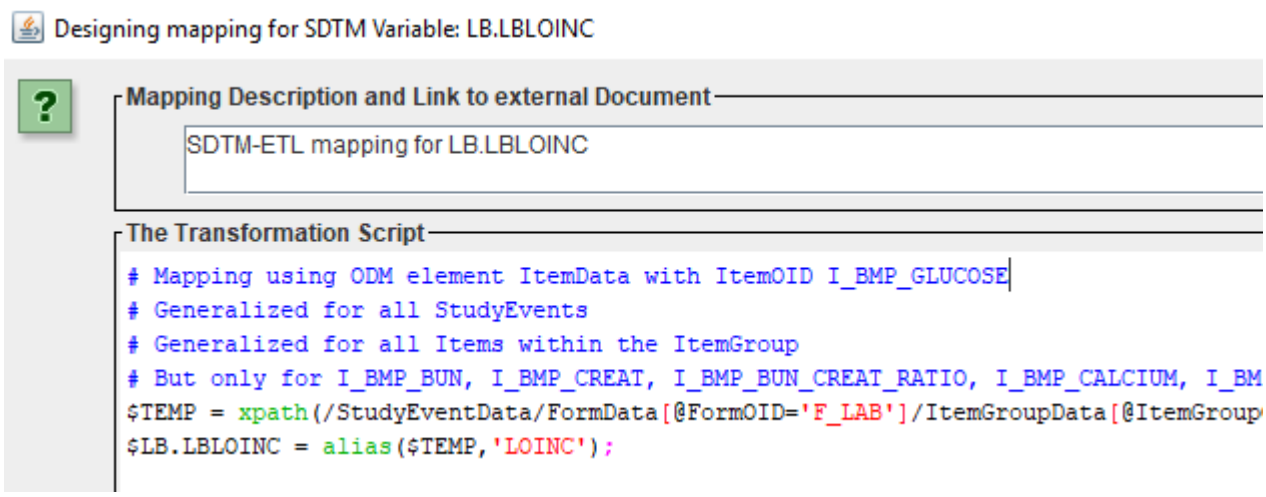
LOINC#	LOINC Name
24320-4	Basic metabolic 1998 panel - Serum or Plasma
2345-7	Glucose [Mass/volume] in Serum or Plasma
3094-0	Urea nitrogen [Mass/volume] in Serum or Plasma
2160-0	Creatinine [Mass/volume] in Serum or Plasma
3097-3	Urea nitrogen/Creatinine [Mass Ratio] in Serum or Plasma
24326-1	Electrolytes 1998 panel - Serum or Plasma
2951-2	Sodium [Moles/volume] in Serum or Plasma
2823-3	Potassium [Moles/volume] in Serum or Plasma
2075-0	Chloride [Moles/volume] in Serum or Plasma
1963-8	Bicarbonate [Moles/volume] in Serum or Plasma
2028-9	Carbon dioxide, total [Moles/volume] in Serum or Plasma

This is represented in the ODM study design e.g. as follows:

```
<!-- Item definitions - LOINC codes are provided in the Alias -->
<ItemDef OID="I_BMP_GLUCOSE" Name="Glucose" DataType="float" Length="6" SignificantDigits="2">
    <Description>
        <TranslatedText xml:lang="en">Glucose [Mass/volume] in Blood</TranslatedText>
    </Description>
    <Question>
        <TranslatedText xml:lang="en">Glucose [Mass/volume] in Blood</TranslatedText>
    </Question>
    <Alias Context="LOINC" Name="2339-0"/>
</ItemDef>
<ItemDef OID="I_BMP_GLUCOSE_UNITS" Name="Glucose Units" DataType="text" Length="10">
<ItemDef OID="I_BMP_BUN" Name="Blood Urea Nitrogen" DataType="float" Length="6" SignificantDigits="2">
    <Description>
        <TranslatedText xml:lang="en">Urea nitrogen [Mass/volume] in Blood</TranslatedText>
    </Description>
    <Question>
        <TranslatedText xml:lang="en">Urea nitrogen [Mass/volume] in Blood</TranslatedText>
    </Question>
    <Alias Context="LOINC" Name="6299-2"/>
</ItemDef>
```

i.e. each data point definition ("ItemDef") also contains an "Alias" element with "LOINC" as the "context", and the LOINC code as the name. This means that e.g. for "blood urea nitrogen", in the context of LOINC, the item is being defined as [the test with code 6299-2](#).

The "alias()" function allows a lookup in the metadata for a data point, and to retrieve a code. This is especially interesting for e.g. populating "LBLOINC", as shown in the following mapping script:



The line with "$TEMP = " picks up the OID of the test (an iteration over all lab tests is performed), and the "alias()" function then retrieves the value of the LOINC code from the metadata in the source ODM file.
The result is:

LB.LBTESTCD	LB.LBTEST	LB.LBCAT	LB.LBORRES	LB.LBORRESU	LB.LBLOINC
GLUC	Glucose	Basic Metabolic Pa...	67.2	mg/dL	2339-0
UREAN	Urea Nitrogen	Basic Metabolic Pa...	7.0	mg/dL	6299-2
CREAT	Creatinine	Basic Metabolic Pa...	1.0	TO DO	38483-4
UREANCRT	Urea Nitrogen/Crea...	Basic Metabolic Pa...	9.6	g/g{creat}	44734-2
CA	Calcium	Basic Metabolic Pa...	8.75	TO DO	49765-1
SODIUM	Sodium	Basic Metabolic Pa...	140	TO DO	2947-0
K	Potassium	Basic Metabolic Pa...	4.2	TO DO	6298-4
CL	Chloride	Basic Metabolic Pa...	111	TO DO	2069-3
CO2	Carbon Dioxide	Basic Metabolic Pa...	26	TO DO	20565-8
GLUC	Glucose	Basic Metabolic Pa...	68.1	mg/dL	2339-0
UREAN	Urea Nitrogen	Basic Metabolic Pa...	7.2	mg/dL	6299-2
CREAT	Creatinine	Basic Metabolic Pa...	1.2	TO DO	38483-4
UREANCRT	Urea Nitrogen/Crea...	Basic Metabolic Pa...	9.3	g/g{creat}	44734-2
CA	Calcium	Basic Metabolic Pa...	8.9	TO DO	49765-1
SODIUM	Sodium	Basic Metabolic Pa...	137	TO DO	2947-0

Working with RESTful Web Services

More and more, the use of RESTful web services for automating tasks in software is becoming custom, also in clinical research. Also CDISC developed a number of RESTful web services for querying the SHARE metadata repository, and provides an API for SHARE.

In the field of clinical research, XML4Pharma and the National Library of Medicine have been pioneers: both provide a number of free RESTful web services that can be used in software applications in clinical research.

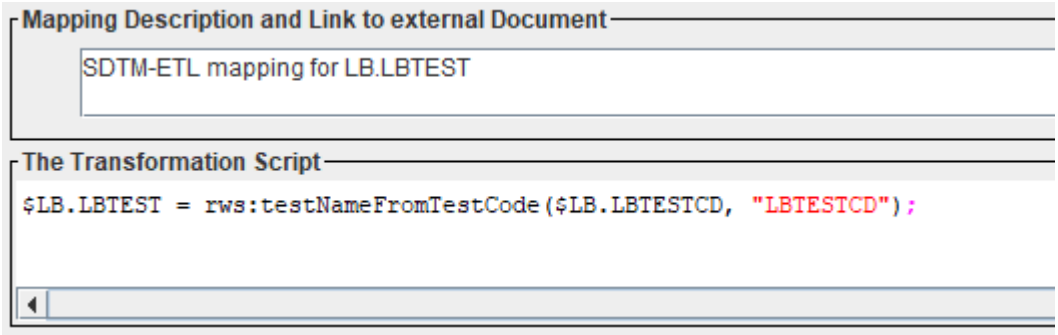
SDTM-ETL 4.0 comes with a number of pre-defined functions that use these RESTful web services, i.e. metadata information is requested from the XML4Pharma server about SDTM variables or LOINC codes.

For example, the function:

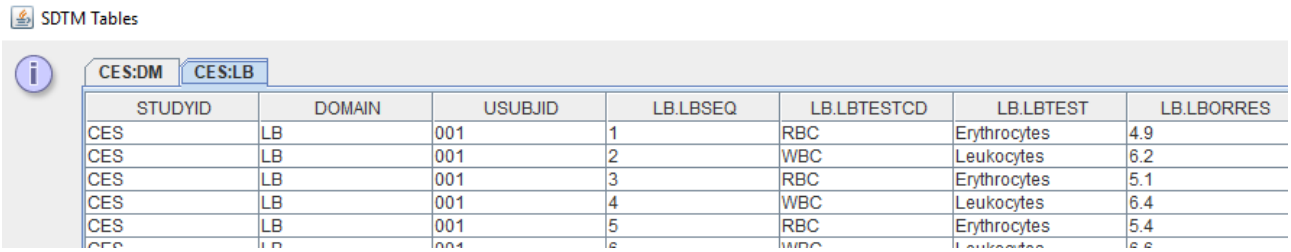
rws:testNameFromTestCode(String testCode, String variableName) can be used to obtain the "test name" (value for –TEST) for an SDTM variable that represents a test code (value for –TESTCD). For example, the function

rws:testNameFromTestCode("ALB", "LBTESTCD") will return "Albumin"

In a mapping script for LBTEST, this e.g. looks like:



And the result after execution for the LB dataset:



STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBTESTCD	LB.LBTEST	LB.LBORRES
CES	LB	001	1	RBC	Erythrocytes	4.9
CES	LB	001	2	WBC	Leukocytes	6.2
CES	LB	001	3	RBC	Erythrocytes	5.1
CES	LB	001	4	WBC	Leukocytes	6.4
CES	LB	001	5	RBC	Erythrocytes	5.4

With "Erythrocytes" and "Leukocytes" being retrieved from "RBC" and "WBC" using the RESTful web service.

Another such pre-defined function is:

rws:sdtmLabel(String sdsVersion, String sdtmVariable)

returning the "label" for the SDTM/SEND variable and SDS version (1.1, 1.2, 1.3)

For all other predefined functions, please see the separate "Using RESTful Web Services" document.

One can however also use any other RESTful web services that are based on "HTTP GET", e.g. company-internal RESTful web services, or RESTful web services made available by the National Library of Medicine, e.g. for working with RxNorm medication numbers and codes.

For example, for getting the name of the medication with the RxNorm number "131725", one could use the NLM RESTful web service "properties" described by the NLM as:

RxNorm RESTful API

Resource "/rxcui/{*rxcui*}/properties"

Get the RxNorm concept properties. The properties returned are:

- Concept name
- Concept identifier (RxCUI)
- Synonym
- RxNorm term type
- Language of the term
- UMLS CUI
- Suppress flag

You can test this in the browser for RxNorm number 131725 by using:
https://rxnav.nlm.nih.gov/REST/rxcui/131725/properties

Delivering the xml:

```
−<rxnormdata>
  −<properties>
      <rxcui>131725</rxcui>
      <name>Ambien</name>
      <synonym/>
      <tty>BN</tty>
      <language>ENG</language>
      <suppress>N</suppress>
      <umlscui>C0487782</umlscui>
    </properties>
  </rxnormdata>
```

However, you can also use this RESTful Web Service in your own mapping scripts, e.g. in "CMTRT" when the RxNorm medication number was e.g. retrieved from an electronic health record, or was collected as such on the CRF.

The mapping script would then be like:

$RXNORM = xpath(….);
$RWSQUERY = concat('https://rxnav.nlm.nih.gov/REST/rxcui/', $RXNORM, '/properties');
$CM.CMTRT = doc($RWSQUERY)/rxnormdata/properties/name;

In the mapping script editor:

The Transformation Script

```
$RXNORM = xpath(...);
# $RXNORM = '131725';
$RWSQUERY = concat('https://rxnav.nlm.nih.gov/REST/rxcui/', $RXNORM, '/properties');
$CM.CMTRT = doc($RWSQUERY)/rxnormdata/properties/name;
```

Where the "doc" function means that an XML document is obtained when the $RWSQUERY is executed, and the "/rxnormdata/properties/name" the path (XPath) in the result document is.

This e.g. leads to the CM record for the case of RxNorm=131725:

CES:DM | CES:LB | CES:CM

STUDYID	DOMAIN	USUBJID	CM.CMSEQ	CM.CMTRT
CES	CM	001	1	Ambien

At this moment, only RESTful web services that use HTTP or HTTPS and for which no authentication is necessary are supported.

Remark that it is always wise to store the "base" of the RESTful web service (in our case "'https://rxnav.nlm.nih.gov/REST/rxcui/") in a "GLOBAL" variable for easy reuse.

For further details and possibilities, see the separate "Using RESTful Web Services" document.

Bug fixes

- The element "BasicDefinitions" was, when present (e.g. using "Insert – MeasurementUnit definitions from ODM into define.xml") not automatically removed when "cleaning the define.xml" e.g. using "File – Save cleaned define.xml". This has been fixed.
- In case the location of the datasets is stored in the define.xml ("def:leaf") and the path to them is referencing a directory, and a "SASDatasetName" is stored, then the validation of the define.xml (using "Validate – define.xml) gave an error stating that the filename does not correspond to the SAS dataset name. This has been fixed in the schematron.

Limitations

- As of the moment of development (October 2018), the development of the rules for define.xml have not finalized. As such, validation when define.xml 2.1 is not supported yet.
- Pinnacle21 does not support define.xml 2.1. As such, using Pinnacle21 in combination with define.xml 2.1 does not make sense.