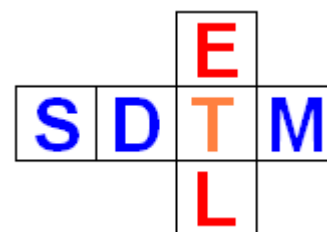


SDTM-ETL 4.4 User Manual and Tutorial

Author: Jozef Aerts, XML4Pharma

Last update: 2023-11-12



Tutorial: Additional filtering on "looping" variables

Table of Contents

Table of Contents	1
Introduction.....	1
Working Example	1
Limitations	20
Solution of the exercises	20
Conclusions.....	22

Introduction

SDTM-ETL already allows a lot of filtering using the wizards following drag-and-drop actions from the ODM tree to SDTM/SEND cells.

There are however situations where one would later want to add additional filters on which ODM data points make it into the SDTM. This may be the case when "fine-tuning" the SDTM datasets. For example:

- one only want to include "new" data points, i.e. that come in the ODM with "TransactionType='Insert'".
- the ODM comes with "artificial" datapoints with the value "M" (for "missed") although the test was optional, and no reason for the "missing" is provided. Classically, such data points do not go into the ODM at all.

SDTM-ETL v.4.4 now allows to add additional filters for those variables that essentially do the selection (as often described as "one record per subject per XXX"), which are called "looping variables" in SDTM-ETL, as they describe the "looping over ...".

Typically these "looping variables" in SDTM-ETL are:

- for "Interventions": the --TRT variable
- for "Events": the --TERM" variable
- for "Findings": the --TESTCD variable.

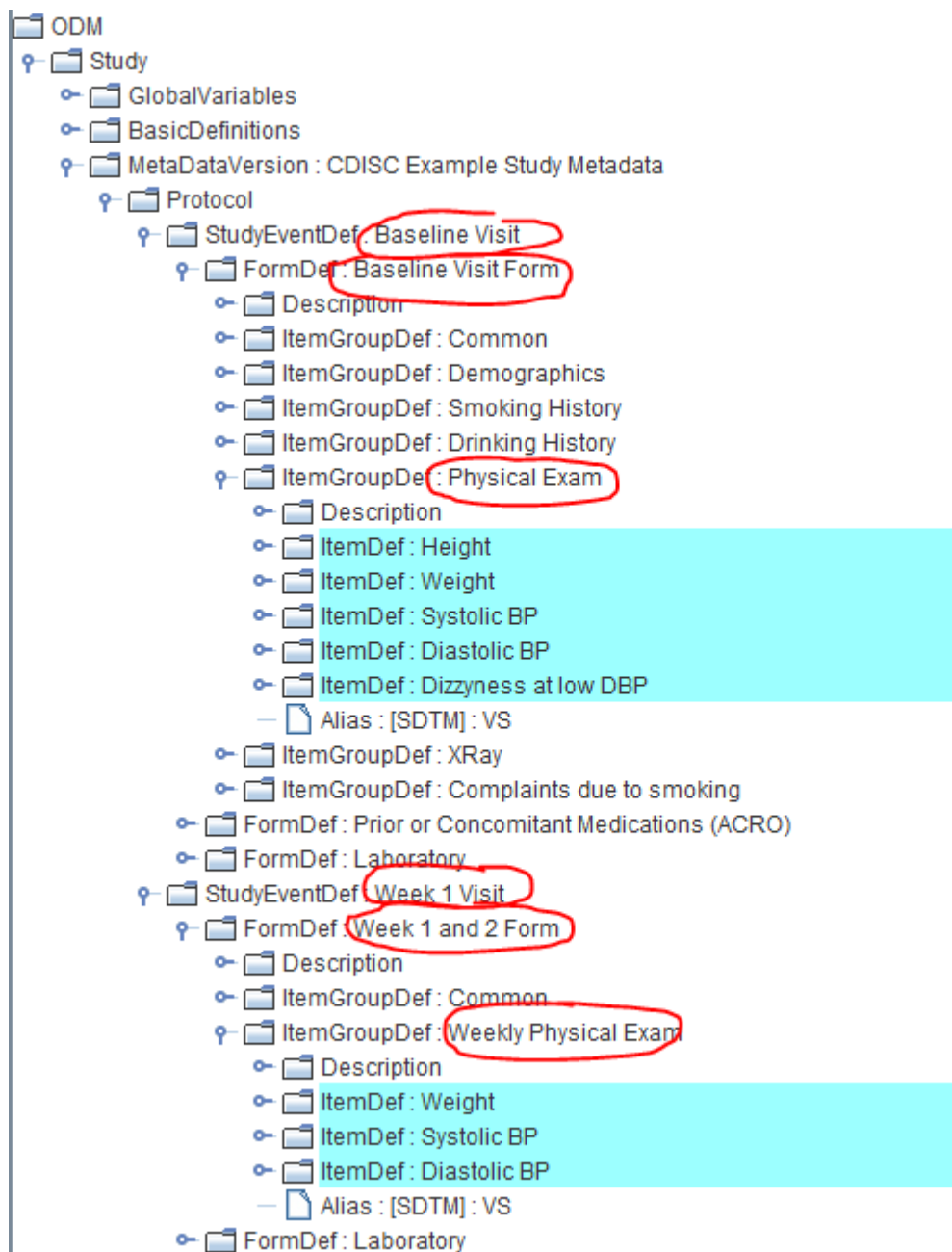
Additional "looping" variables may be assigned, but this is usually not really necessary. For example, for "Findings", typical additional (but usually optional) "looping" variables are VISITNUM, --TPTNUM (or --TPT).

Working Example

We will explain the new feature using the well known "CES" ("CDISC Example Study"), developed already many years ago by our colleague Dave Ibersen-Hurst. We will develop a few mappings for

the VS (Vital Signs) SDTM dataset.

Our ODM metadata are represented in SDTM-ETL as:

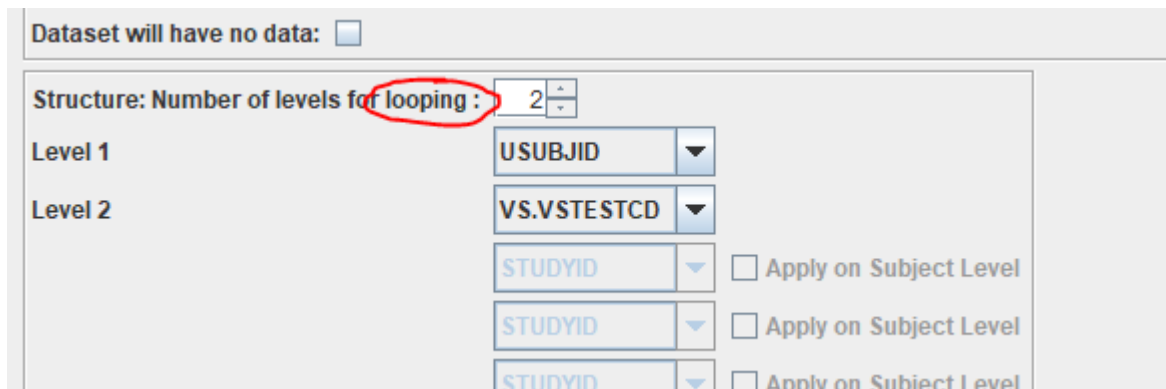


There are two (sets of) visits: a "baseline" visit, and a "Week 1" visit, which is repeating. Each of these has its own sets of forms, and within that form, a group "physical exam" (yes, I know, the name may be confusing), which is essentially a vital signs form.

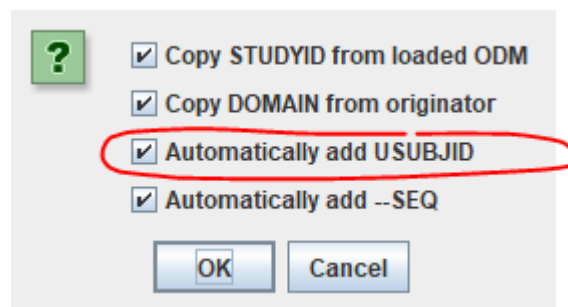
After we have instantiated the VS domain (drag-and-drop to the bottom), a row "CES:VS" is created:

TI	STUDYID	DOMAIN	TI.IETESTCD	TI.IETEST	TI.IECAT	TI.IESCAT	TI.TIRL	TI
TS	STUDYID	DOMAIN	TS.TSSEQ	TS.TSGRPID	TS.TSPARMCD	TS.TSPARM	TS.TSVAL	T:
RELREC	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	RELTYPE	RELID	
SUPPQUAL	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	SUPPQUAL.QN...	SUPPQUAL.QL...	S
RELSUB	STUDYID	USUBJID	RELSUB.POOLID	RELSUB.RSUB...	RELSUB.SREL			
OI	STUDYID	DOMAIN	OI.NHOID	OI.OISEQ	OI.OIPARMCD	OI.OIPARM	OI.OIVAL	
CES:VS	STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSGRPID	VS.VSSPID	VS.VSTESTCD	V:

One also sees that "VSTESTCD" has a cyan edge, meaning that it is a "looping variable", i.e. the selection of the data from the ODM will be done using the mapping code in VSTESTCD. A double-click on "CES:VS" also shows this:



We always start the developing of the mappings with the looping variables. For USUBJID, this was probably already done when doing the "drag-and-drop", by leaving the checkbox "" on:

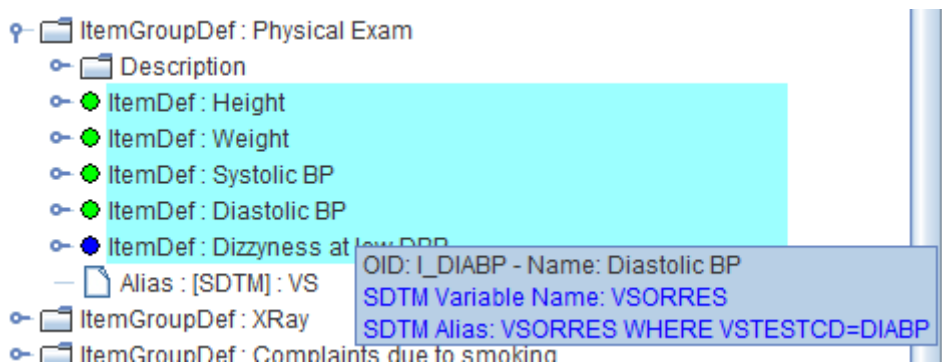


meaning that for the USUBJID, we start from the "SubjectKey" in the ODM data. The mapping script for USUBJID then is:

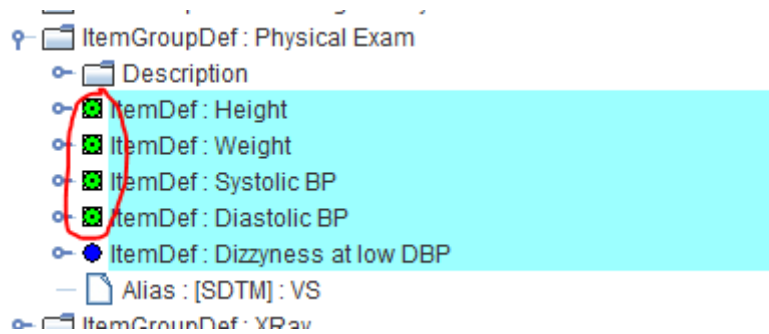
```
The Transformation Script
1 $USUBJID = usubjid();
```

If desired, it can be adapted/extended e.g. as:

In our case, most items in the ODM metadata have already been annotated for use in SDTM. One can see this from the cyan background color:



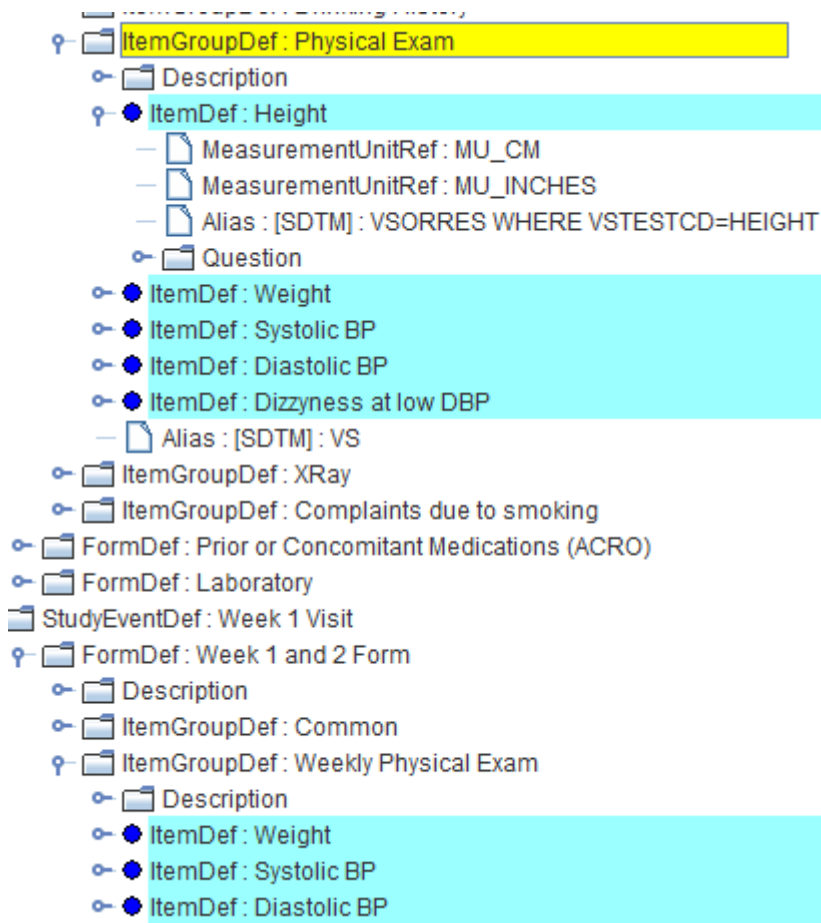
This also means that they are "hot candidates" for VSORRES (and thus for VSTESTCD), which can be seen when selecting "VSORRES" as the SDTM cell. The "traffic lights" then change into:



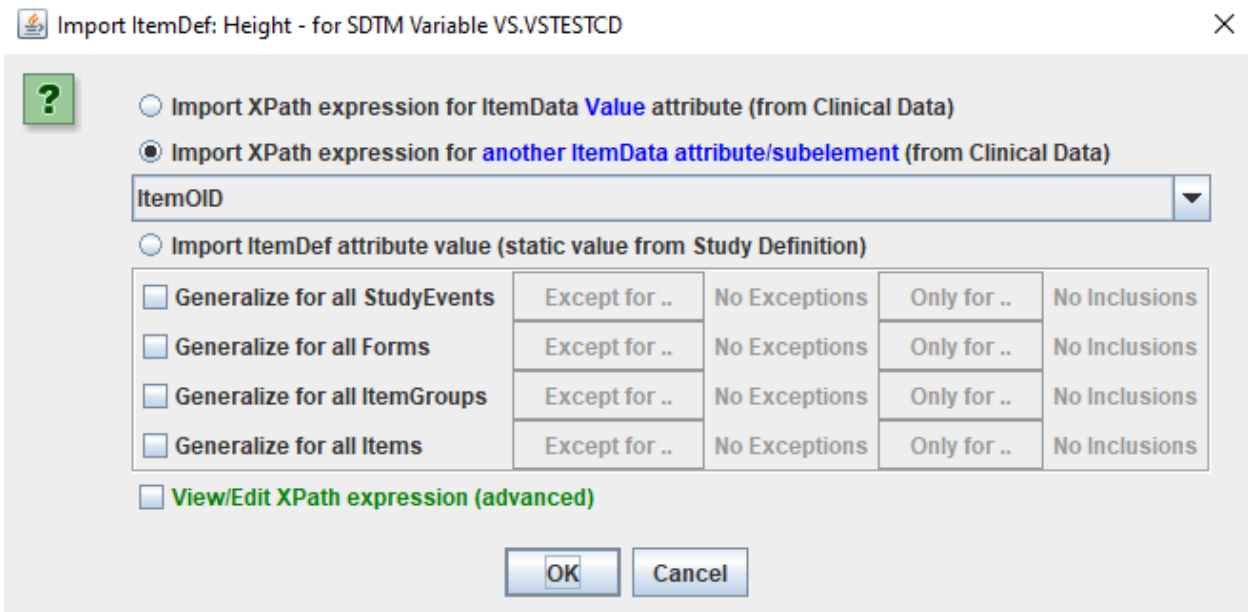
i.e. one sees a square around them.

Or one uses the menu "Navigate - Find hot SDTM candidate", which will open the ODM tree at that point when it was collapsed, and highlight the first of these "hot candidates".

We also see that vital signs were measured in both the visits, but with different form and subform (ItemGroup) identifiers:

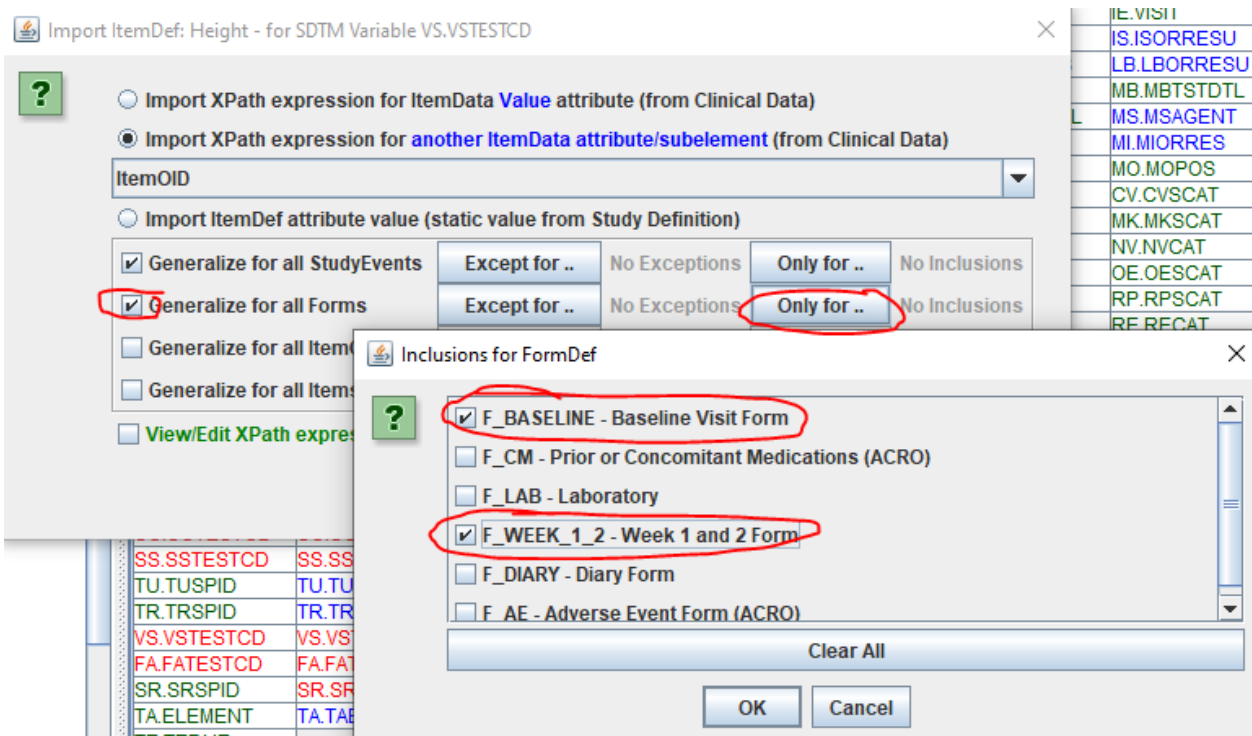


We can now drag-and-drop any of the hot candidates to the SDTM "VSTESTCD" cell, leading to:



as we want to obtain the vital signs data for any visit, we click the checkbox "Generalize for all StudyEvents".

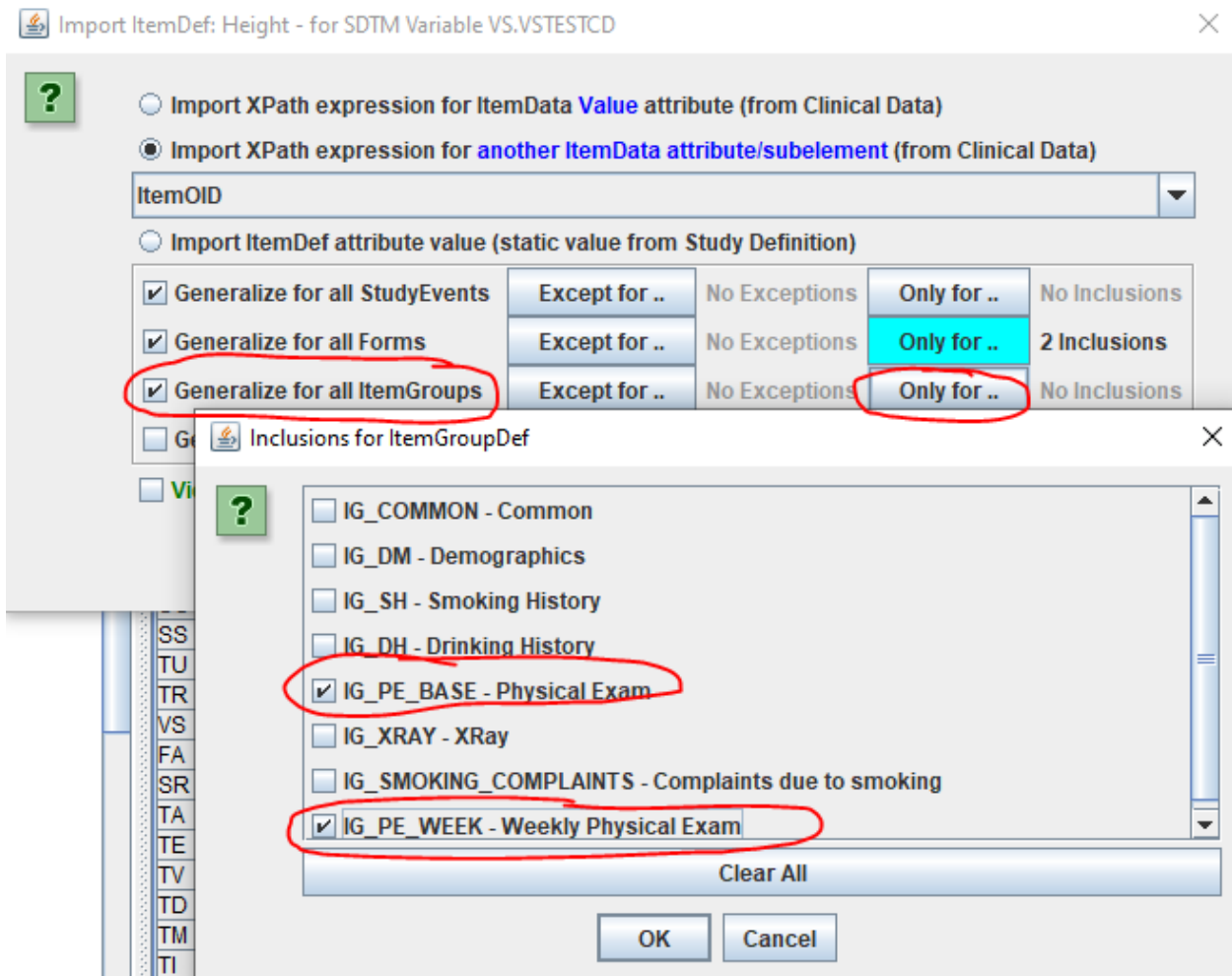
As the vital signs data come from different forms, we want to do a selection, so we check "Generalize for all Forms" and then use "Only for ...":



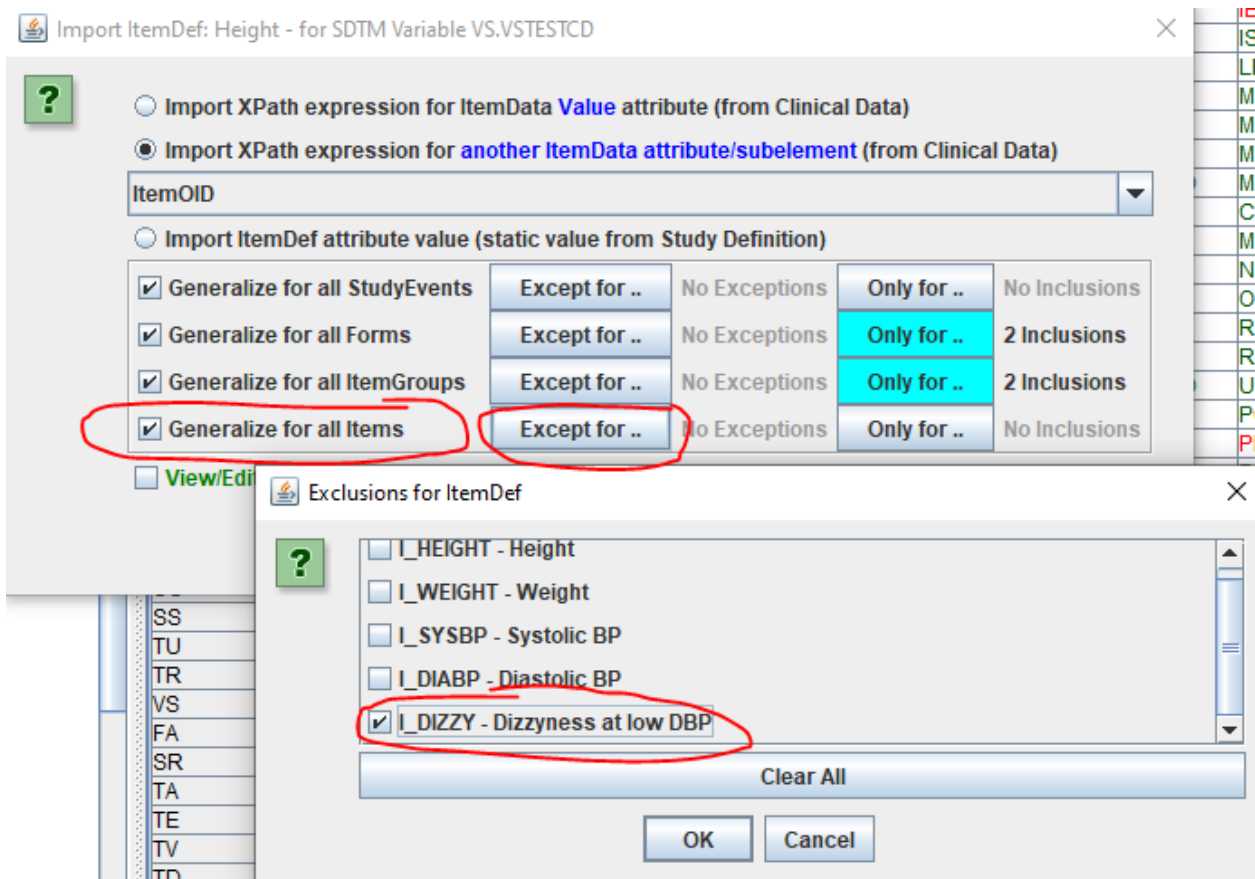
and then select the two forms that contain the items on vital signs.

For the subforms (ItemGroups), from the ODM tree, we see that those for the vital signs have a different ID depending on whether it is the baseline form, or the week_1_2 form¹, so also here, we must make a selection. So we check "Generalize for all Items" and use "Only for", and select the appropriate subforms (ItemGroups):

¹ The reason is that in the repeating weekly (repeating) visit, the height is not measured anymore.

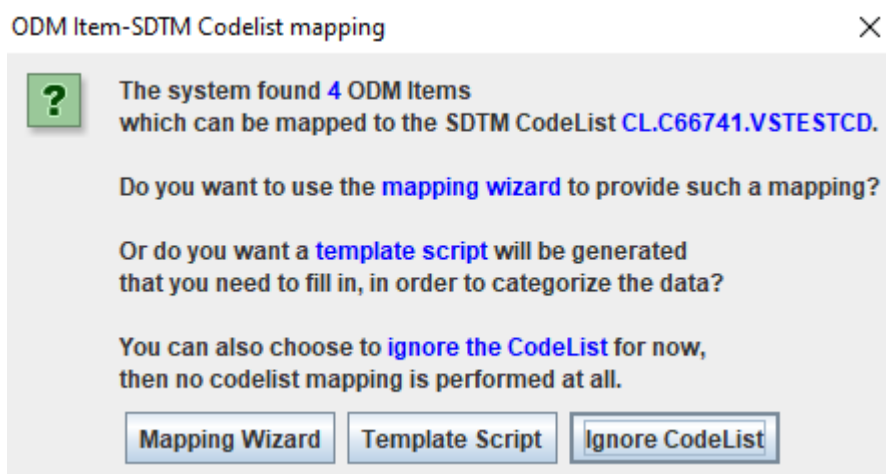


and as we do not want to use "Dizziness" as a VS test (we can later add it as a "non-standard variable going into SUPPVS), we also select on the items, using "Generalize for all Items". In the current case, it is then easier to exclude "dizziness", so we can better use "Except for ...":



When all done, we get:

After clicking "OK", as new wizard is displayed:



In most cases, we will want to do the "Mapping Wizard" do the work to map to CDISC Controlled Terminology (CT), so we click "Mapping Wizard", leading to:

?

ODM Item	SDTM CodeList Item
<input type="checkbox"/> Show ODM decoded values	
I_HEIGHT	ABI <input type="button" value="Search"/>
I_WEIGHT	ABI <input type="button" value="Search"/>
I_SYSBP	ABI <input type="button" value="Search"/>
I_DIABP	ABI <input type="button" value="Search"/>
MISSING VALUE	ABI <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

Add comment line to each mapping

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

and we can give the automated mapping (based on word similarity with the ODM "names" - these can be shown using the "Show ODM decoded values") a chance by clicking "Attempt 1:1 mapping", leading to:

?

ODM Item	SDTM CodeList Item
<input type="checkbox"/> Show ODM decoded values	
I_HEIGHT	HEIGHT <input type="text"/> <input type="button" value="Search"/>
I_WEIGHT	WEIGHT <input type="text"/> <input type="button" value="Search"/>
I_SYSBP	SYSBP <input type="text"/> <input type="button" value="Search"/>
I_DIABP	DIABP <input type="text"/> <input type="button" value="Search"/>
MISSING VALUE	<input type="text"/> <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

Add comment line to each mapping

Except for items already mapped

Also use CDISC Synonym List

Also use Company Synonym List

which ("oh surprise") to exactly what we want.

After clicking "OK", the mapping script is automatically created, and assigned to the "VSTESTCD" cell:

Mapping Script Editor for SDTM Variable VS.VSTESTCD


```

1 # Mapping using ODM element ItemData with ItemOID I_HEIGHT - value from attribute ItemOID
2 # Generalized for all StudyEvents
3 # Generalized for all Forms within the StudyEvent
4 # Generalized for all ItemGroups within the Form
5 # Generalized for all Items within the ItemGroup
6 # Except for: I_DIZZY
7 # Mapping for ODM Items (I_HEIGHT, I_WEIGHT, I_SYSBP, I_DIABP) to SDTM CodeList VS.VSTESTCD
8 # with CodeList OID 'CL06741.VSTESTCD'
9 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_BASELINE' or @FormOID='F_WEEK_1_2']/ItemGroupData[@ItemGroupOID='IG_FE_BASE' or @ItemGroupOID='IG_FE_WEEK']/ItemData[not(@ItemOID='I_DIZZY')]/@ItemOID);
10 if ($CODEDVALUE == 'I_HEIGHT') {
11   $NEWCODEDVALUE = 'HEIGHT';
12 } elseif ($CODEDVALUE == 'I_WEIGHT') {
13   $NEWCODEDVALUE = 'WEIGHT';
14 } elseif ($CODEDVALUE == 'I_SYSBP') {
15   $NEWCODEDVALUE = 'SYSBP';
16 } elseif ($CODEDVALUE == 'I_DIABP') {
17   $NEWCODEDVALUE = 'DIABP';
18 } elseif ($CODEDVALUE == '') {
19   $NEWCODEDVALUE = '';
20 } else {
21   $NEWCODEDVALUE = 'NULL';
22 }
23 $VS.VSTESTCD = $NEWCODEDVALUE;

```

Sometimes it is a bit difficult to see the whole XPath expression, and one needs to scroll a bit to the right.


If we had also checked the checkbox "View/Edit the XPath expression", the more user-friendly dialog splitting up the XPath over the different levels would now appear:

	Condition for StudyEventData:	<input type="checkbox"/> Edit	
	Condition for FormData:	<input type="checkbox"/> Edit	[@FormOID='F_BASELINE' or @FormOID='F_WEEK_1_2']
	Condition for ItemGroupData:	<input type="checkbox"/> Edit	[@ItemGroupOID='IG_PE_BASE' or @ItemGroupOID='IG_PE_WEEK']
	Condition for ItemData:	<input type="checkbox"/> Edit	[not(@ItemOID='1_DIZZY')]
	Value selection:	<input type="checkbox"/> Edit	@ItemOID

After clicking "OK" (one may still want to edit the mapping script, but it is better to first test it using collected (or mock) ODM data), we see that the VSTESTCD cell is "grayed out", meaning that there is mapping data available for that variable.

It now usually is custom to also generate the mapping for VSTEST. We can go through the same procedure (but another codelist will be presented to map to). However, as there is a 1:1 correspondence between VSTESTCD and VSTEST, and the software knows this, the easier is to just double click on VSTEST, and a dialog will show up, asking:

Use decode() function? ✕

 The easy way to get the values for the variable **VS.VSTEST** is to use the **decode** function on the codelist **CL.C66741.VSTESTCD** of the variable **VS.VSTESTCD**.

The mapping script then reduces to:
\$VS.VSTEST = decode(\$VS.VSTESTCD, 'CL.C66741.VSTESTCD', '');

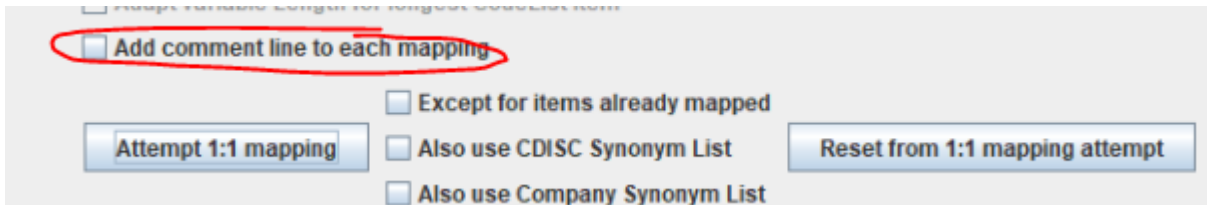
Do you want me to implement this mapping script?

which would take the "decode" value of the codelist for VSTESTCD, which usually is present when the CDISC-CT that comes with the software is used. When clicking "Yes, please", the mapping script for VSTEST simply becomes:

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

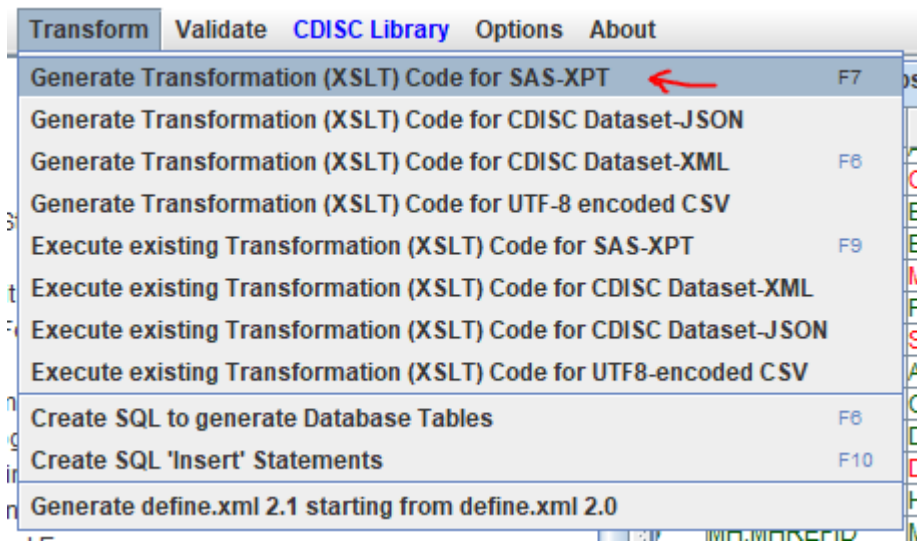
establishing the 1:1 relation.

REMARK: when using the "CodeList mapping wizard", it may be a good idea to also check the checkbox "add comment line to each mapping", especially as you also want other people to understand how your mapping was done.



Time to test!

As we have already mappings for different variables: STUDYID (taken from the Study-OID in the ODM), DOMAIN, USUBJID, VSTESTCD and VSTEST, the best is to use the menu "Transform - Generate Transformation Code for SAS-XPT²"



and, after going to some intermediate dialogs (e.g. enabling to save the generated XSLT code to file) then select the ODM file with the clinical data, e.g.:

² We will now still stick to XPT format, although we expect that FDA and other regulatory authorities will switch to modern [CDISC Dataset-JSON](#) in the next 2 years.

Execute Transformation (XSLT) Code for SAS-XPT

ODM file with clinical data:
 D:\SDTM-ETL\TestFiles\ODM1-3-1\CES_ClinicalData_single_subject.xml Browse...

MetaData in separate ODM file
 D:\SDTM-ETL\Development_TestFiles_v_4-4\CES_Metadata_better_decodes.xml Browse...

Administrative data in separate ODM file
 D:\SDTM-ETL\Development_TestFiles_v_4-4\CES_Metadata_better_decodes.xml Browse...

Save output XML to file
Browse...

Perform post-processing for assigning --LOBXFL

Split records > 200 characters to SUPP-- records

Move non-standard SDTM Variables to SUPP-- Move 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 Re-sort records using define.xml keys

Save Result SDTM tables as SAS XPORT files Perform CDISC CORE validation on generated SAS XPORT files

SAS XPORT files directory:
Browse...

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

For testing, it usually is not necessary to generate XPT files already, as SDTM-ETL has a build-in viewer for inspecting generated SDTM, independent of the submission format. So we leave "Save SDTM tables as SAS XPORT files" unchecked. When then clicking "Execute Transformation on Clinical Data"

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 Re-sort records using define.xml keys

Save Result SDTM tables as SAS XPORT files Perform CDISC CORE validation on generated SAS XPORT files

SAS XPORT files directory:
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

Close

the transformation is executed (some messages may appear in the "Messages and error messages box", and the generated table or tables are displayed:

CES:VS						
STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSTESTCD	VS.VSTEST	
CES	VS	001	1	HEIGHT	Height	
CES	VS	001	2	WEIGHT	Weight	
CES	VS	001	3	SYSBP	Systolic Blood Pressure	
CES	VS	001	4	DIABP	Diastolic Blood Pressure	
CES	VS	001	5	WEIGHT	Weight	
CES	VS	001	6	SYSBP	Systolic Blood Pressure	
CES	VS	001	7	DIABP	Diastolic Blood Pressure	
CES	VS	001	8	WEIGHT	Weight	
CES	VS	001	9	SYSBP	Systolic Blood Pressure	
CES	VS	001	10	DIABP	Diastolic Blood Pressure	

For VSORRES, we can now do the same drag-and-drop, use the same "generalization" and "only for ..." and "except for" - all these checkbox values are remembered.

So doing drag-and-drop for the first "Height" item to this time VSORRES, and accepting all earlier selected choices, leads to the automatically generated mapping script for VSORRES:

```

1 # Mapping using ODM element ItemData with ItemOID I_HEIGHT
2 # Generalized for all StudyEvents
3 # Generalized for all Forms within the StudyEvent
4 # Generalized for all Items within the Form
5 # Generalized for all Items within the ItemGroup
6 # Except for: I_DIZZY
7 $VS.VSORRES = xpath(/StudyEventData/FormData[@FormOID='F_BASELINE' or @FormOID='F_WEEK_1_2']/ItemGroupData[@ItemGroupOID='IG_FE_BASE' or @ItemGroupOID='IG_FE_WEEK']/ItemData[not(@ItemOID='I_DIZZY')])/$Value;
8

```

and executing the mapping on the (test) clinical data again, leading to:

CES:VS							VS.VSORRES
STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSTESTCD	VS.VSTEST		
CES	VS	001	1	HEIGHT	Height		193
CES	VS	001	2	WEIGHT	Weight		90
CES	VS	001	3	SYSBP	Systolic Blood Pressure		120
CES	VS	001	4	DIABP	Diastolic Blood Pressure		80
CES	VS	001	5	WEIGHT	Weight		90.1
CES	VS	001	6	SYSBP	Systolic Blood Pressure		123
CES	VS	001	7	DIABP	Diastolic Blood Pressure		90
CES	VS	001	8	WEIGHT	Weight		89.9
CES	VS	001	9	SYSBP	Systolic Blood Pressure		127
CES	VS	001	10	DIABP	Diastolic Blood Pressure		84

We can then continue providing the mapping for other variables, as explained in other tutorials. We will here show only how this can be very easily be done for VISITNUM and VISIT, and then start explaining the new feature of adding additional filters on the looping variables (here: VSTESTCD).

For the visit number, simply drag-and-drop from the ODM item "StudyEventDef: Baseline Visit":

☞ MetaDataVersion : CDISC Example Study Metadata

☞ Protocol

☞ StudyEventDef : Baseline Visit

☞ FormDef : Baseline Visit Form

☞ Description

☞ ItemGroupDef : Common

☞ ItemGroupDef : Demographics

☞ ItemGroupDef : Smoking History

to the SDTM cell "VISITNUM".

The system remembers that we want to have VS data retrieved for all visits, so will present us:

Import StudyEventDef: BASELINE - for SDTM Variable VS.VISITNUM

Import XPath expression for
 Import attribute value (static value) for

OID

Generalize for all StudyEvents Except for .. No Exceptions Only for .. No Inclusions

View/Edit XPath expression (advanced)

OK Cancel

stating "do so for all visits (StudyEvents). After clicking OK, the mapping script is automatically created:

```
The Transformation Script
1 # Mapping using ODM element StudyEventData using value from attribute StudyEventOID
2 # Generalized for all StudyEvents
3 $VS.VISITNUM = xpath(/StudyEventData/@StudyEventOID/);
4
```

and we then test on the clinical data, the result is:

SDTM Tables

CES:VS

STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSTESTCD	VS.VSTEST	VS.VSORRES	VS.VISITNUM
CES	VS	001	1	HEIGHT	Height	193	BASELINE
CES	VS	001	2	WEIGHT	Weight	90	BASELINE
CES	VS	001	3	SYSBP	Systolic Blood Press...	120	BASELINE
CES	VS	001	4	DIABP	Diastolic Blood Pres...	80	BASELINE
CES	VS	001	5	WEIGHT	Weight	90.1	WEEK_1
CES	VS	001	6	SYSBP	Systolic Blood Press...	123	WEEK_1
CES	VS	001	7	DIABP	Diastolic Blood Pres...	90	WEEK_1
CES	VS	001	8	WEIGHT	Weight	89.9	WEEK_2
CES	VS	001	9	SYSBP	Systolic Blood Press...	127	WEEK_2
CES	VS	001	10	DIABP	Diastolic Blood Pres...	84	WEEK_2

which is ... not what we want.

So we need to adapt and edit the mapping script manually.

We can generate an "if-elsif-construct" e.g. :

The Transformation Script

```

1 # Mapping using ODM element StudyEventData using value from attribute StudyEventOID
2 # Generalized for all StudyEvents
3 $VISIT = xpath(/StudyEventData/@StudyEventOID/);
4 if($VISIT = 'BASELINE') {
5     $VS.VISITNUM = 0;
6 } elseif($VISIT = 'WEEK_1') {
7     $VS.VISITNUM = 1;
8 } elseif($VISIT = 'WEEK_2') {
9     $VS.VISITNUM = 2;
10 } else {
11     $VS.VISITNUM = '';
12 }

```

Scripting Language Functions

+	-	*	/	xpath	com
usubjid	investigator	site	name	sitename	que
substring	substring-before	substring-after	concat	string-length	rep
if	elseif	else	trim	upper-case	lower

leading to the result:

USUBJID	VS.VSSEQ	VS.VSTESTCD	VS.VSTEST	VS.VSORRES	VS.VISITNUM
001	1	HEIGHT	Height	193	0
001	2	WEIGHT	Weight	90	0
001	3	SYSBP	Systolic Blood Press...	120	0
001	4	DIABP	Diastolic Blood Pres...	80	0
001	5	WEIGHT	Weight	90.1	1
001	6	SYSBP	Systolic Blood Press...	123	1
001	7	DIABP	Diastolic Blood Pres...	90	1
001	8	WEIGHT	Weight	89.9	2
001	9	SYSBP	Systolic Blood Press...	127	2
001	10	DIABP	Diastolic Blood Pres...	84	2

For "VISIT" (Visit Name) we can do the same drag-and-drop, use an if-elseif-else structure as for VISITNUM, or try to do it in a similar, but (for the fun) somewhat different way. For example:

STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSTESTCD	VS.VSTEST	VS.VSORRES	VS.VISITNUM	VS.VISIT
CES	VS	001	1	HEIGHT	Height	193	0	BASELINE
CES	VS	001	2	WEIGHT	Weight	90	0	BASELINE
CES	VS	001	3	SYSBP	Systolic Blood Press...	120	0	BASELINE
CES	VS	001	4	DIABP	Diastolic Blood Pres...	80	0	BASELINE
CES	VS	001	5	WEIGHT	Weight	90.1	1	WEEK 1
CES	VS	001	6	SYSBP	Systolic Blood Press...	123	1	WEEK 1
CES	VS	001	7	DIABP	Diastolic Blood Pres...	90	1	WEEK 1
CES	VS	001	8	WEIGHT	Weight	89.9	2	WEEK 2
CES	VS	001	9	SYSBP	Systolic Blood Press...	127	2	WEEK 2
CES	VS	001	10	DIABP	Diastolic Blood Pres...	84	2	WEEK 2

Performing additional filtering on "looping" variables

As of SDTM-ETL 4.4, it is possible to add additional filters on the "looping" variables such as "--TESTCD" (in the case of a Findings domain), "--TRT" (in case of an Interventions domain), "--TERM" (in the case of an Events domain), without needing to edit the XPath expression.

Editing the XPath expression for a "looping" variable can vary from very easy to very difficult, depending on the complexity of the XPath expression (we would categorize the above generated XPath expression as "medium complex") and the level of expertise on working with XPath expressions. XPath is pretty easy to learn, but not all users of the software are willing to invest the time.

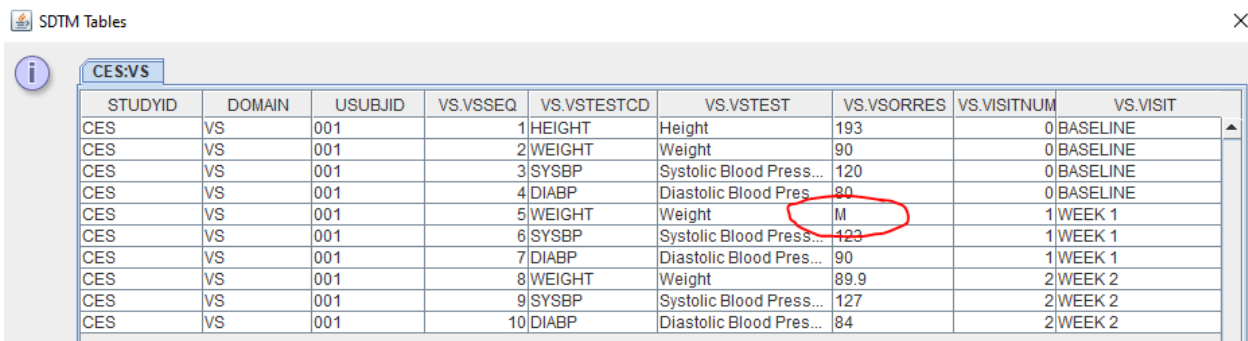
With the new feature however, only a basic understanding of XPath and of ODM are required.

When would you want to add such an additional filter? Some (fictional?) examples:

- A colleague asks you to generate a VS XPT dataset with only the systolic and diastolic blood pressures, but you do not want to start editing the XPath expression or redo the mapping
- Even worse, the week later, the colleagues wants to obtain a VS XPT dataset with only the systolic blood pressures with values higher than 120 mmHg (the colleague is new, and does not now how to do filtering in XPT datasets).
- In ODM, values that have not been collected should not be in the ODM dataset, unless also the reason why the measurement has not been collected is provided. However, we sometimes see that some EDC vendors provide a value of "M" (missing) for measurements that were not done, although they were optional. In such cases, we may do not want to appear these as rows in the SDTM dataset.

We will demonstrate the new feature using the case of "Weight" data points in our ODM, for which the value "M" has been provided, and that we want to be filtered out

The result than e.g. is:



STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSTESTCD	VS.VSTEST	VS.VSORRES	VS.VISITNUM	VS.VISIT
CES	VS	001	1	HEIGHT	Height	193	0	BASELINE
CES	VS	001	2	WEIGHT	Weight	90	0	BASELINE
CES	VS	001	3	SYSBP	Systolic Blood Press...	120	0	BASELINE
CES	VS	001	4	DIABP	Diastolic Blood Pres...	80	0	BASELINE
CES	VS	001	5	WEIGHT	Weight	M	1	WEEK 1
CES	VS	001	6	SYSBP	Systolic Blood Press...	123	1	WEEK 1
CES	VS	001	7	DIABP	Diastolic Blood Pres...	90	1	WEEK 1
CES	VS	001	8	WEIGHT	Weight	89.9	2	WEEK 2
CES	VS	001	9	SYSBP	Systolic Blood Press...	127	2	WEEK 2
CES	VS	001	10	DIABP	Diastolic Blood Pres...	84	2	WEEK 2

and we don't want this row with VSORRES='M' to appear at all.

Our mapping script for the looping variable VSTESTCD is:


```

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 Forms within the StudyEvent
4 # Generalized for all ItemGroups within the Form
5 # Generalized for all Items within the ItemGroup
6 # Except for: I_DIZZY
7 # Mapping for ODM Items [I_HEIGHT, I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.VSTESTCD
8 # with CodeList OID 'CL.C66741.VSTESTCD'
9 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_BASELINE' or @FormOID='F_WEEK_1_2']/ItemGroupData[@ItemGroupOID='IG_PE_BASE' or
10 # map the ODM OIDs to CDISC-CT
11 if ($CODEDVALUE == 'I_HEIGHT') {
12   $NEWCODEDVALUE = 'HEIGHT';
13 } elseif ($CODEDVALUE == 'I_WEIGHT') {
14   $NEWCODEDVALUE = 'WEIGHT';
15 } elseif ($CODEDVALUE == 'I_SYSBP') {
16   $NEWCODEDVALUE = 'SYSBP';

```

with the selection done in the statement \$CODEDVALUE = xpath(.....);

We are at the level of the data point itself in the ODM, i.e. on the "ItemData" level. The attributes of "ItemData" are, according to the [ODM specification](#):

3.1.4.1.1.1.1 ItemData

Body:

([AuditRecord?](#), [Signature?](#), [MeasurementUnitRef?](#), [Annotation*](#))

Attributes:

ItemOID	oidref		Reference to the ItemDef .
TransactionType	(Insert Update Remove Upsert Context)	(optional)	The TransactionType attribute need not be present in a Snapshot document.
Value	text	(optional)	The data collected for an item. This data is represented according to DataType attribute of the ItemDef.
IsNull	(Yes)	(optional)	IsNull is a flag to signify that an item's value is to be set to null. If the Value attribute is set, IsNull must not be set. If IsNull is set, the Value attribute must not be provided. In the interest of creating non-verbose XML instances, one should not use ItemData elements with IsNull set to "Yes" to indicate uncollected data. The better practice is to transmit only collected data.

so, we can further filter on "ItemOID", "TransactionType", "Value" and "IsNull" (the latter is seldom used). We could also further filter on values within the child elements "AuditRecord", "Signature", "MeasurementUnitRef", or "Annotation", when present. This will however be extremely seldom be necessary.

We can even further filter on ancestor element values - we will give an example later.

So, we do have a data point with value "M" in the source, for which we don't want to have a row generated. In the ODM it is represented by:

```

<ItemGroupData ItemGroupOID="IG_PE_WEEK">
  <ItemData ItemOID="I_WEIGHT" Value="M">
    <MeasurementUnitRef MeasurementUnitOID="MU_KG"/>
  </ItemData>
  <ItemData ItemOID="I_SYSBP" Value="123">
    <MeasurementUnitRef MeasurementUnitOID="MU_MMHG"/>
  </ItemData>
  <ItemData ItemOID="I_DIABP" Value="90">
    <MeasurementUnitRef MeasurementUnitOID="MU_MMHG"/>
  </ItemData>
</ItemGroupData>

```

In order to filter out, we use the new function "xpathfilter":

```

8 # with CodeList OID 'CL.C66741.VSTESTCD'
9 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_BASELINE' or @For:
10 $CODEDVALUE = xpathfilter($CODEDVALUE, "[not(@Value='M')]");
11 if ($CODEDVALUE == 'I_HEIGHT') {
12   $NEWCODEDVALUE = 'HEIGHT';
13 } elseif ($CODEDVALUE == 'I_WEIGHT') {
14   $NEWCODEDVALUE = 'WEIGHT';
15 } elseif ($CODEDVALUE == 'I_SYSBP') {
16   $NEWCODEDVALUE = 'SYSBP';
17 } elseif ($CODEDVALUE == 'I_DIABP') {

```

i.e. immediately after the primary selection (the XPath statement generated by the drag-and-drop and the subsequent wizards), we add a statement:

```
$CODEDVALUE = xpathfilter($CODEDVALUE, "[not(@Value='M')]");
```

Resulting in:

SDTM Tables

STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSTESTCD	VS.VSTEST	VS.VSORRES	VS.VISITNUM	VS.VISIT
CES	VS	001	1	HEIGHT	Height	193	0	BASELINE
CES	VS	001	2	WEIGHT	Weight	90	0	BASELINE
CES	VS	001	3	SYSBP	Systolic Blood Pressure	120	0	BASELINE
CES	VS	001	4	DIABP	Diastolic Blood Pressure	80	0	BASELINE
CES	VS	001	5	SYSBP	Systolic Blood Pressure	123	1	WEEK 1
CES	VS	001	6	DIABP	Diastolic Blood Pressure	90	1	WEEK 1
CES	VS	001	7	WEIGHT	Weight	89.9	2	WEEK 2
CES	VS	001	8	SYSBP	Systolic Blood Pressure	127	2	WEEK 2
CES	VS	001	9	DIABP	Diastolic Blood Pressure	84	2	WEEK 2

where one sees that the row with VSORRES=M has disappeared.

The "xpathfilter" line and functionality looks like a normal variable assignment, but there are some rules:

- Currently, the variable on the left, must be the same as the variable used in the primary selection (i.e. from the line " = xpath(.....);)
- The first argument currently must be the variable also set on the left of the equation (self-assignment).
- The second argument must be a string containing a relative XPath expression, which must be an XPath "predicate" (see further).

So e.g. the following are invalid statements and will not result in filtering

\$A = xpath(.....);
\$B = xpathfilter(\$A, ...);
\$A = xpath(.....);
\$B = concat(\$A, 'test');
\$A = xpathfilter(\$B);

We envisage to remove these limitations in future versions of the software.

For the second argument of the function, this must be an [XPath predicate](#).

XPath predicates are essentially "where statements", and where the statement is embedded in square brackets [...], i.e. the square brackets essentially define a "where".

In our example:

```
"[not(@Value='M')]"
```

we pass a string (within double quotes) with the string value being a predicate (a "where" statement in square brackets) stating "the filter selects the data points for which the value (value of the @Value attribute) is not equal to the string 'M'".

Other examples of predicates that can be used when the XPath result is at the level of "ItemData" (the datapoint level):

```
[@Value > 90] : selects the data points for which the value is higher than 90.  
[not(@IsNull)] : selects the data points that do not have an "IsNull" attribute  
[@ItemOID='I_SYSBP' and @Value > 120]: selects the diastolic blood pressure datapoints for which the value is higher than 120.
```

Remark that the latter will also filter out all heights, weights and systolic blood pressure rows in our example.

A nice XPath exercise is to find out the XPath predicate that does keep all height, weight and diastolic blood pressure, but filters out the systolic blood pressure higher than 120 (i.e. keeps diastolic BP less or equal than 120).

Solution at the end of this document!

The great advantage of this method is that one can always re-establish the original selection by simply commenting out the line with the "xpathfilter" statement. For example:

The Transformation Script

```
3 # Generalized for all Forms within the StudyEvent  
4 # Generalized for all ItemGroups within the Form  
5 # Generalized for all Items within the ItemGroup  
6 # Except for: I_DIZZY  
7 # Mapping for ODM Items [I_HEIGHT, I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.VSTESTCD  
8 # with CodeList OID 'CL.C66741.VSTESTCD'  
9 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_BASELINE' or @FormOID='F_WEEK_1_2']/ItemData)  
10 # $CODEDVALUE = xpathfilter($CODEDVALUE, "[not(@Value='M')]");  
11 if ($CODEDVALUE == 'I_HEIGHT') {  
12     $NEWCODEDVALUE = 'HEIGHT';
```

It is also possible to add an additional selection on elements / information higher up in the ODM tree. For example, we stated that we take the vital signs for all visits (StudyEvents in the ODM). So if we e.g. want to filter on the "baseline" visit, we need to go up in the ODM tree.

In XPath, going up in the tree, is done using "..". So, if we just want to keep the vitals from the baseline visit, as we are on the ItemData (data point) level, we must go up 3 levels (ItemGroupData, FormData, StudyEventData). So, if we use the xpathfilter as follows:

The Transformation Script

```
3 # Generalized for all Forms within the StudyEvent  
4 # Generalized for all ItemGroups within the Form  
5 # Generalized for all Items within the ItemGroup  
6 # Except for: I_DIZZY  
7 # Mapping for ODM Items [I_HEIGHT, I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.V  
8 # with CodeList OID 'CL.C66741.VSTESTCD'  
9 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_BASELINE' or @FormOID='F_WE  
10 $CODEDVALUE = xpathfilter($CODEDVALUE, "../../../../../@StudyEventOID='BASELINE'");  
11
```

resulting in an SDTM table with only the rows representing data from the baseline visit:

STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSTESTCD	VS.VSTEST	VS.VSORRES	VS.VISITNUM	VS.VISIT
CES	VS	001	1	HEIGHT	Height	193	0	BASELINE
CES	VS	001	2	WEIGHT	Weight	90	0	BASELINE
CES	VS	001	3	SYSBP	Systolic Blood Pressure	120	0	BASELINE
CES	VS	001	4	DIABP	Diastolic Blood Pressure	80	0	BASELINE

This is of course something we never want to do in a real submission!

It is also possible to have several additional filters.

For example, the following script:

```

The Transformation Script
3 # Generalized for all Forms within the StudyEvent
4 # Generalized for all ItemGroups within the Form
5 # Generalized for all Items within the ItemGroup
6 # Except for: I_DIZZY
7 # Mapping for ODM Items [I_HEIGHT, I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.
8 # with CodeList OID 'CL.C66741.VSTESTCD'
9 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_BASELINE' or @FormOID='F_W
10 $CODEDVALUE = xpathfilter($CODEDVALUE,"../../../../@StudyEventOID='BASELINE'");
11 $CODEDVALUE = xpathfilter($CODEDVALUE,"[not(@Value='M')]");

```

will only keep the vital signs information from the baseline visit for which the value is not "M".

Exercise: write the "xpathfilter" line for selecting subject with ODM SubjectKey=001.

Limitations

At the moment, one may only apply the "xpathfilter()" function on the "looping" variables, which usually are --TESTCD, --TERM, and --TRT depending on the SDTM/SEND class.

Adding such a filter in any non-looping variable mapping script will not have any effect at all.

Furthermore, currently, the "xpathfilter()" can only be used to filter a variable on itself, i.e. the result variable and the variable provided as the argument, must be the same.

This is a limitation that may be removed in future.

Solution of the exercises

1. Write the XPath predicate that does keep height, weight and diastolic blood pressure, but filters out the systolic blood pressure higher than 120 (i.e. keeps diastolic BP less or equal than 120).

Solution: `[@ItemOID='I_WEIGHT' or @ItemOID='I_HEIGHT' or @ItemOID='I_DIABP' or (@ItemOID='I_SYSBP' and @Value <= 120)]`

resulting in:

CES:VS								
STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSTESTCD	VS.VSTEST	VS.VSORRES	VS.VISITNUM	VS.VISIT
CES	VS	001	1	HEIGHT	Height	193	0	BASELINE
CES	VS	001	2	WEIGHT	Weight	90	0	BASELINE
CES	VS	001	3	SYSBP	Systolic Blood Pressure	120	0	BASELINE
CES	VS	001	4	DIABP	Diastolic Blood Pressu...	80	0	BASELINE
CES	VS	001	5	WEIGHT	Weight	90.1	1	WEEK 1
CES	VS	001	6	DIABP	Diastolic Blood Pressu...	90	1	WEEK 1
CES	VS	001	7	WEIGHT	Weight	89.9	2	WEEK 2
CES	VS	001	8	DIABP	Diastolic Blood Pressu...	84	2	WEEK 2

where one sees that the record with VSTESTCD=SYSBP and VSORRES=123 has disappeared.

One can also use (a bit more sophisticated):

[not(@ItemOID=I_SYSBP) or @Value <= 120]

2. Write the "xpathfilter" line for selecting subject with ODM SubjectKey=001

Solution: [../../../../@SubjectKey='001']

Our "looping" is on the "ItemData" (i.e. data point) level. To get to the "SubjectData" level, we need to go 4 levels up (ItemGroupData, FormData, StudyEventData, SubjectData), for which we use "../../../../" and then take the value of the "SubjectKey" attribute "@SubjectKey".

To exclude subjects, one just need to revert the expression using "not()", e.g.

[not../../../../@SubjectKey='007') and not../../../../@SubjectKey='013')]

would exclude all data for subjects 007 and 013.

Conclusions

The new (as of SDTM-ETL v.4.4) "xpathfilter()" function allows, for special and more complicated situations, to break up the filtering in several steps.

With this new function, one can start from a simple selection for the looping variable (usually -TESTCD, --TRT or --TERM), e.g. select all items of a specific form for all visits (StudyEvents), and then add additional filters, until the desired selection is obtained.

The use of the "xpathfilter()" requires a basic understanding of the [ODM standard](#) and the hierarchy of the "ClinicalData" part of ODM, and some basic understanding of XPath. XPath is however easy to learn, and many introduction tutorials can be found on the internet.

As for all mappings in SDTM-ETL, one should of course always test the results on correctness and completeness.