SDTM-ETL 4.x User Manual and Tutorial

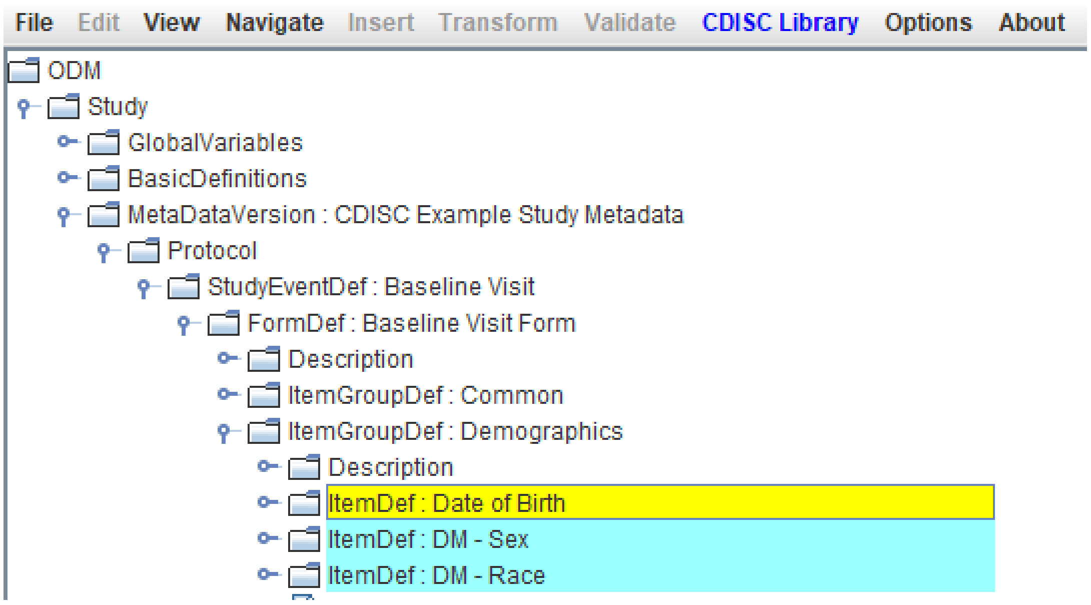
Author: Jozef Aerts, XML4Pharma

Last update: 2022-06-19

<u>Loading an SDTM template – mappings for DM</u>

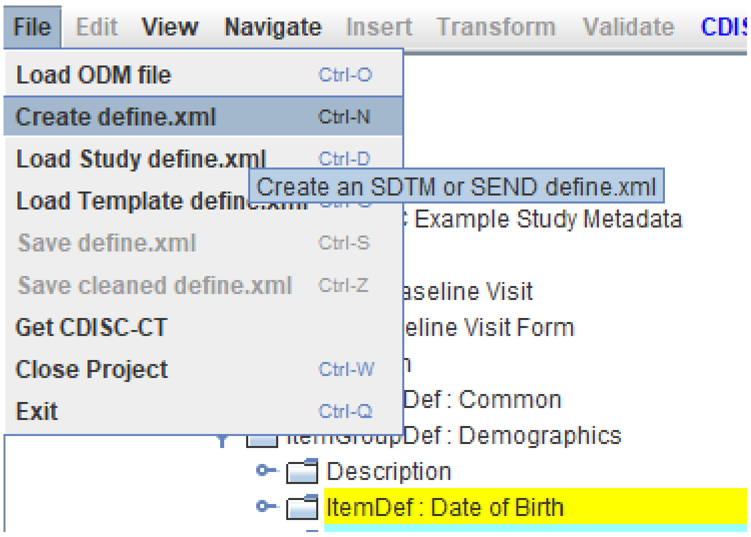
After having loaded and inspected a CDISC ODM file with the study design, we can start working on the mapping with SDTM or SEND.

At the left side of the screen, the tree view of the clinical study design is already shown, in this case of the CES study[1]:



the right side of the screen being still empty.

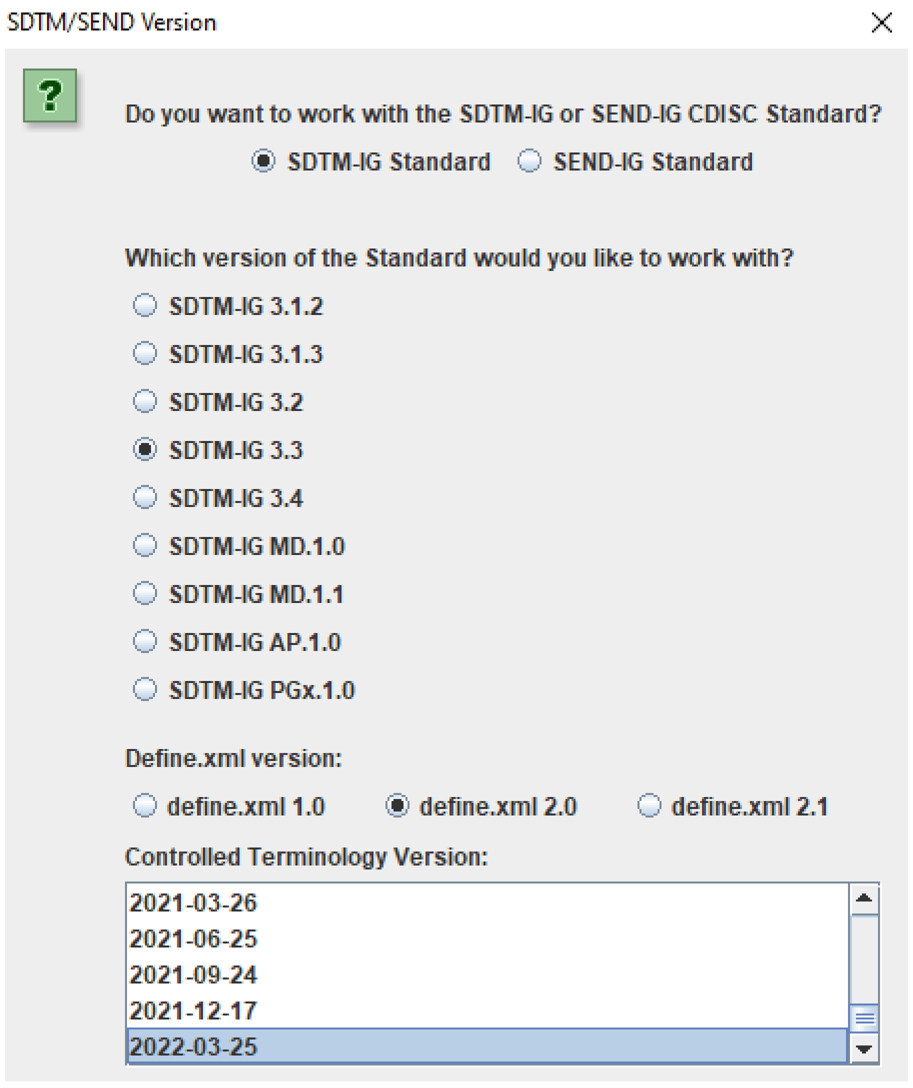
In order to start mapping to SDTM (or SEND) a template which is implementing the SDTM-IG or SEND-IG needs to be loaded. In order to do so, use the menu "File – Create define.xml":



[1]This is a study design originally developed by Dave Iberson-Hurst for demo purposes, and later extended by others.

The reason it speaks about a define.xml is that all our mappings, and any other metadata about our SDTM or SEND will be stored in a define.xml structure, which is kept in sync with everything that we do, so that at the end, we will be able to generate a define.xml file[2] for our study with just a few mouse clicks.

A dialog is then presented:



SDTM/SEND Version

Do you want to work with the SDTM-IG or SEND-IG CDISC Standard?

[✓ SDTM-IG Standard] SEND-IG Standard

Which version of the Standard would you like to work with?

- SDTM-IG 3.1.2
- SDTM-IG 3.1.3
- SDTM-IG 3.2
- [✓ SDTM-IG 3.3]
- SDTM-IG 3.4
- SDTM-IG MD.1.0
- SDTM-IG MD.1.1
- SDTM-IG AP.1.0
- SDTM-IG PGx.1.0

Define.xml version:

define.xml 1.0 [✓ define.xml 2.0] define.xml 2.1

Controlled Terminology Version:

- 2021-03-26
- 2021-06-25
- 2021-09-24
- 2021-12-17
- [✓ 2022-03-25]

The user can choose between differe nt SDTM versions and different SEND versions[3].
In case there are different versions of controlled terminology for the given standard, the versions are presented and the user can decide which version of the CDISC controlled terminology should be loaded[4]. Remark that each time new Controlled Terminologies are published by CDISC, these can be downloaded from our SDTM-ETL website and added to the folder "CDISC_CT", making them immediately available without the need of a software update.

Also, one can choose between using Define-XML 1.0, 2.0 and 2.1 for keeping the metadata. Remark however that Define-XML 1.0 is not accepted by the regulatory authorities, and that we

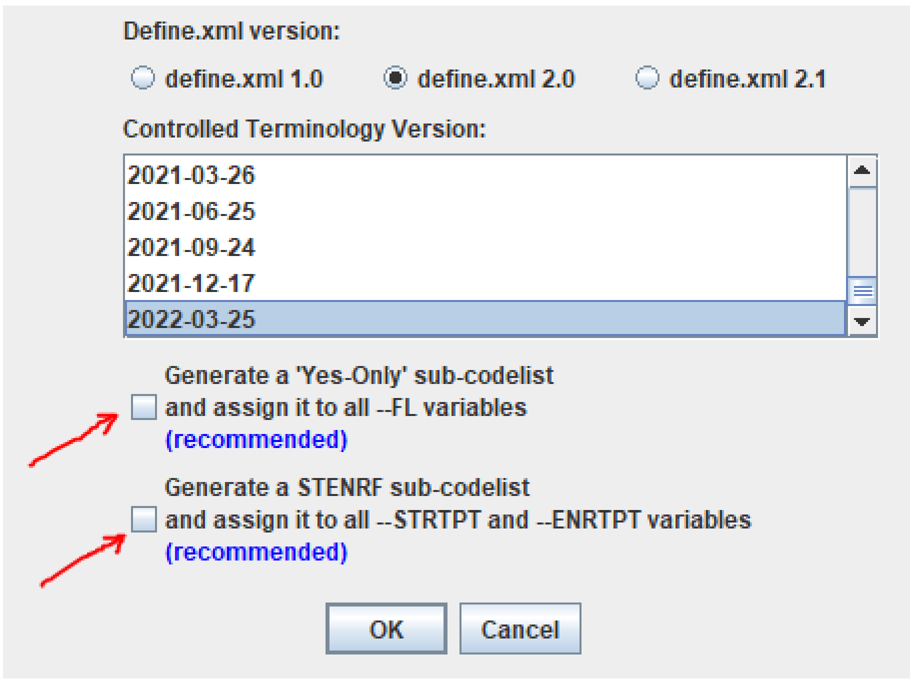
[2]For any SDTM or SEND submission, the FDA requires a define.xml file to be submitted together with the actual data sets, containing the metadata for the submission files.
3 Including SENDIG-DART 1.1
[4]Later we will learn how to load additional codelists when necessary.

don't maintain the templates for Define-XML 1.0 anymore.

Suppose we want to use Define-XML 2.0 with SDTM-IG 3.3, together with CDISC-CT version 2022-03-25[5].

One can also come to this dialog using the keyboard combination CTRL-N.

At the bottom of the dialog, one also finds two checkboxes:



Define.xml version:

( ) define.xml 1.0 [✓ define.xml 2.0] ( ) define.xml 2.1

Controlled Terminology Version:

2021-03-26
2021-06-25
2021-09-24
2021-12-17
[✓ 2022-03-25]

☐ Generate a 'Yes-Only' sub-codelist and assign it to all --FL variables (recommended)

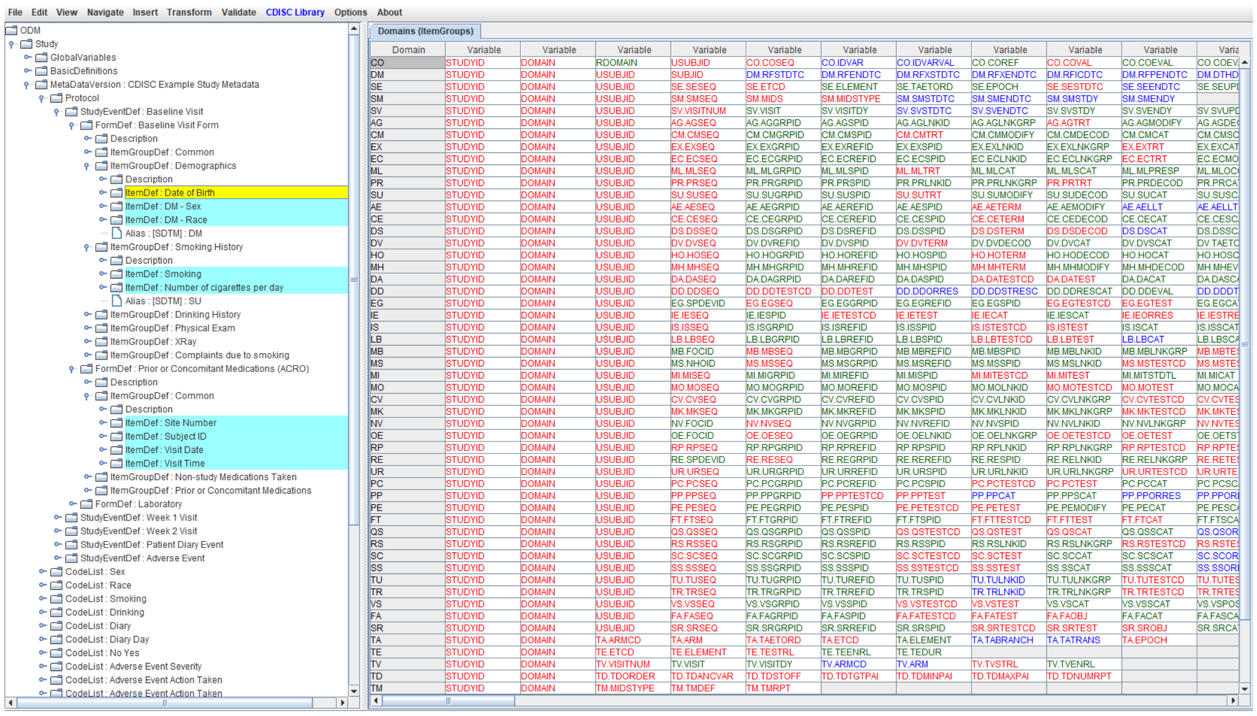
☐ Generate a STENRF sub-codelist and assign it to all --STRTPT and --ENRTPT variables (recommended)

When checked (highly recommended), the software will immediately and automatically generate subset codelists, one for "Yes only" and one for the -STRTPT and -ENRTPT variables and assign them to the variables where this useful and necessary.
That this is needed is unfortunately due to the stubborn refusal of the CDISC Controlled Terminology team to publish subset codelists for these.

After clicking "OK“, the system now starts loading the template, which can take a few minutes.

One can then see that the right side of the screen is now filled with an SDTM table, containing a row for each SDTM domain in the SDTM-IG, and a cell for each SDTM variable, with the first cell containing the SDTM domain name (DM, TE, ...):

[5] The files with CDISC-CT are kept in the directory "CDISC-CT“. In case new controlled terminology was published by CDISC, you can obtain the file for the SDTM-ETL website. If you then copy it in the "CDISC_CT“ directory, it will automatically be added to the list of available versions.



- ODM
  - Study
    - GlobalVariables
    - BasicDefinitions
    - MetaDataVersion : CDISC Example Study Metadata
      - Protocol
        - StudyEventDef : Baseline Visit
          - FormDef : Baseline Visit Form
            - Description
            - ItemGroupDef : Common
            - ItemGroupDef : Demographics
              - Description
              - ItemDef : Date of Birth
              - ItemDef : DM - Sex
              - ItemDef : DM - Race
              - Alias : [SDTM] : DM
            - ItemGroupDef : Smoking History
              - Description
              - ItemDef : Smoking
              - ItemDef : Number of cigarettes per day
              - Alias : [SDTM] : SU
            - ItemGroupDef : Drinking History
            - ItemGroupDef : Physical Exam
            - ItemGroupDef : XRay
            - ItemGroupDef : Complaints due to smoking
          - FormDef : Prior or Concomitant Medications (ACRO)
            - Description
            - ItemGroupDef : Common
              - Description
              - ItemDef : Site Number
              - ItemDef : Subject ID
              - ItemDef : Visit Date
              - ItemDef : Visit Time
            - ItemGroupDef : Non-study Medications Taken
            - ItemGroupDef : Prior or Concomitant Medications
          - FormDef : Laboratory
        - StudyEventDef : Week 1 Visit
        - StudyEventDef : Week 2 Visit
        - StudyEventDef : Patient Diary Event
        - StudyEventDef : Adverse Event
      - CodeList : Sex
      - CodeList : Race
      - CodeList : Smoking
      - CodeList : Drinking
      - CodeList : Diary
      - CodeList : Diary Day
      - CodeList : No Yes
      - CodeList : Adverse Event Severity
      - CodeList : Adverse Event Action Taken
      - CodeList : Adverse Event Action Taken

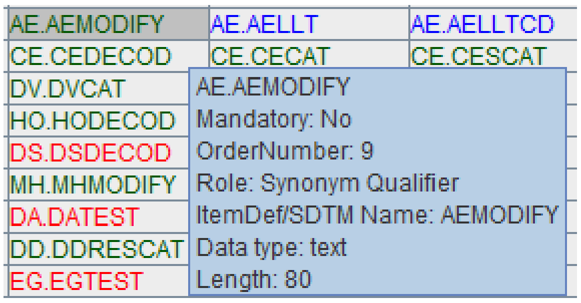
Domains (ItemGroups)

Domain	Variable	Variable	Variable	Variable	Variable	Variable	Variable	Variable	Variable	Variable	Varia
CO	STUDYID	DOMAIN	RDOMAIN	USUBJID	CO.COSEQ	CO.IDVAR	CO.IDVARVAL	CO.COREF	CO.COVAL	CO.COEVAL	CO.COEV
DM	STUDYID	DOMAIN	USUBJID	SUBJID	DM.RFSTDTC	DM.RFENDTC	DM.RFXSTDTC	DM.RFXENDTC	DM.RFICDTC	DM.RFPENDTC	DM.DTHD
SE	STUDYID	DOMAIN	USUBJID	SE.SESEQ	SE.ETCD	SE.ELEMENT	SE.TAETORD	SE.EPOCH	SE.SESTDTC	SE.SEENDTC	SE.SEUPD
SM	STUDYID	DOMAIN	USUBJID	SM.SMSEQ	SM.MIDS	SM.MIDSTYPE	SM.SMSTDTC	SM.SMENDTC	SM.SMSTDY	SM.SMENDY	
SV	STUDYID	DOMAIN	USUBJID	SV.VISITNUM	SV.VISIT	SV.VISITDY	SV.SVSTDTC	SV.SVENDTC	SV.SVSTDY	SV.SVENDY	SV.SVUPD
AG	STUDYID	DOMAIN	USUBJID	AG.AGSEQ	AG.AGGRPID	AG.AGSPID	AG.AGLNKID	AG.AGLNKGRP	AG.AGTRT	AG.AGMODIFY	AG.AGDE
CM	STUDYID	DOMAIN	USUBJID	CM.CMSEQ	CM.CMGRPID	CM.CMSPID	CM.CMTRT	CM.CMMODIFY	CM.CMDECOD	CM.CMCAT	CM.CMSC
EX	STUDYID	DOMAIN	USUBJID	EX.EXSEQ	EX.EXGRPID	EX.EXREFID	EX.EXSPID	EX.EXLNKID	EX.EXLNKGRP	EX.EXTRT	EX.EXCAT
EC	STUDYID	DOMAIN	USUBJID	EC.ECSEQ	EC.ECGRPID	EC.ECREFID	EC.ECSPID	EC.ECLNKID	EC.ECLNKGRP	EC.ECTRT	EC.ECMO
ML	STUDYID	DOMAIN	USUBJID	ML.MLSEQ	ML.MLGRPID	ML.MLSPID	ML.MLTRT	ML.MLCAT	ML.MLSCAT	ML.MLPRESP	ML.MLOC
PR	STUDYID	DOMAIN	USUBJID	PR.PRSEQ	PR.PRGRPID	PR.PRSPID	PR.PRLNKID	PR.PRLNKGRP	PR.PRTRT	PR.PRDECOD	PR.PRCAT
SU	STUDYID	DOMAIN	USUBJID	SU.SUSEQ	SU.SUGRPID	SU.SUSPID	SU.SUTRT	SU.SUMODIFY	SU.SUDECOD	SU.SUCAT	SU.SUSC
AE	STUDYID	DOMAIN	USUBJID	AE.AESEQ	AE.AEGRPID	AE.AEREFID	AE.AESPID	AE.AETERM	AE.AEMODIFY	AE.AELLT	AE.AELLT
CE	STUDYID	DOMAIN	USUBJID	CE.CESEQ	CE.CEGRPID	CE.CEREFID	CE.CESPID	CE.CETERM	CE.CEDECOD	CE.CECAT	CE.CESC
DS	STUDYID	DOMAIN	USUBJID	DS.DSSEQ	DS.DSGRPID	DS.DSREFID	DS.DSSPID	DS.DSTERM	DS.DSDECOD	DS.DSCAT	DS.DSSC
DV	STUDYID	DOMAIN	USUBJID	DV.DVSEQ	DV.DVREFID	DV.DVSPID	DV.DVTERM	DV.DVDECOD	DV.DVCAT	DV.DVSCAT	DV.TAETO
HO	STUDYID	DOMAIN	USUBJID	HO.HOSEQ	HO.HOGRPID	HO.HOREFID	HO.HOSPID	HO.HOTERM	HO.HODECOD	HO.HOCAT	HO.HOSC
MH	STUDYID	DOMAIN	USUBJID	MH.MHSEQ	MH.MHGRPID	MH.MHREFID	MH.MHSPID	MH.MHTERM	MH.MHMODIFY	MH.MHDECOD	MH.MHEV
DA	STUDYID	DOMAIN	USUBJID	DA.DASEQ	DA.DAGRPID	DA.DAREFID	DA.DASPID	DA.DATESTCD	DA.DATEST	DA.DACAT	DA.DASC
DD	STUDYID	DOMAIN	USUBJID	DD.DDSEQ	DD.DDTESTCD	DD.DDTEST	DD.DDORRES	DD.DDSTRESC	DD.DDRESCAT	DD.DDEVAL	DD.DDDT
EG	STUDYID	DOMAIN	USUBJID	EG.SPDEVID	EG.EGSEQ	EG.EGGRPID	EG.EGREFID	EG.EGSPID	EG.EGTESTCD	EG.EGTEST	EG.EGCA
IE	STUDYID	DOMAIN	USUBJID	IE.IESEQ	IE.IESPID	IE.IETESTCD	IE.IETEST	IE.IECAT	IE.IESCAT	IE.IEORRES	IE.IESTRE
IS	STUDYID	DOMAIN	USUBJID	IS.ISSEQ	IS.ISGRPID	IS.ISREFID	IS.ISSPID	IS.ISTESTCD	IS.ISTEST	IS.ISCAT	IS.ISSCAT
LB	STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBGRPID	LB.LBREFID	LB.LBSPID	LB.LBTESTCD	LB.LBTEST	LB.LBCAT	LB.LBSCA
MB	STUDYID	DOMAIN	USUBJID	MB.FOCID	MB.MBSEQ	MB.MBGRPID	MB.MBREFID	MB.MBSPID	MB.MBLNKID	MB.MBLNKGRP	MB.MBTES
MS	STUDYID	DOMAIN	USUBJID	MS.NHOID	MS.MSSEQ	MS.MSGRPID	MS.MSREFID	MS.MSSPID	MS.MSLNKID	MS.MSTESTCD	MS.MSTES
MI	STUDYID	DOMAIN	USUBJID	MI.MISEQ	MI.MIGRPID	MI.MIREFID	MI.MISPID	MI.MITESTCD	MI.MITEST	MI.MITSTDTL	MI.MICAT
MO	STUDYID	DOMAIN	USUBJID	MO.MOSEQ	MO.MOGRPID	MO.MOREFID	MO.MOSPID	MO.MOLNKID	MO.MOTESTCD	MO.MOTEST	MO.MOCA
CV	STUDYID	DOMAIN	USUBJID	CV.CVSEQ	CV.CVGRPID	CV.CVREFID	CV.CVSPID	CV.CVLNKID	CV.CVLNKGRP	CV.CVTESTCD	CV.CVTES
MK	STUDYID	DOMAIN	USUBJID	MK.MKSEQ	MK.MKGRPID	MK.MKREFID	MK.MKSPID	MK.MKLNKID	MK.MKLNKGRP	MK.MKTESTCD	MK.MKTES
NV	STUDYID	DOMAIN	USUBJID	NV.FOCID	NV.NVSEQ	NV.NVGRPID	NV.NVREFID	NV.NVSPID	NV.NVLNKID	NV.NVLNKGRP	NV.NVTES
OE	STUDYID	DOMAIN	USUBJID	OE.FOCID	OE.OESEQ	OE.OEGRPID	OE.OELNKID	OE.OELNKGRP	OE.OETESTCD	OE.OETEST	OE.OETST
RP	STUDYID	DOMAIN	USUBJID	RP.RPSEQ	RP.RPGRPID	RP.RPREFID	RP.RPSPID	RP.RPLNKID	RP.RPLNKGRP	RP.RPTESTCD	RP.RPTES
RE	STUDYID	DOMAIN	USUBJID	RE.SPDEVID	RE.RESEQ	RE.REGRPID	RE.REREFID	RE.RESPID	RE.RELNKID	RE.RELNKGRP	RE.RETES
UR	STUDYID	DOMAIN	USUBJID	UR.URSEQ	UR.URGRPID	UR.URREFID	UR.URSPID	UR.URLNKID	UR.URLNKGRP	UR.URTESTCD	UR.URTE
PC	STUDYID	DOMAIN	USUBJID	PC.PCSEQ	PC.PCGRPID	PC.PCREFID	PC.PCSPID	PC.PCTESTCD	PC.PCTEST	PC.PCCAT	PC.PCSC
PP	STUDYID	DOMAIN	USUBJID	PP.PPSEQ	PP.PPGRPID	PP.PPTESTCD	PP.PPTEST	PP.PPCAT	PP.PPSCAT	PP.PPORRES	PP.PPORI
PE	STUDYID	DOMAIN	USUBJID	PE.PESEQ	PE.PEGRPID	PE.PESPID	PE.PETESTCD	PE.PETEST	PE.PEMODIFY	PE.PECAT	PE.PESCA
FT	STUDYID	DOMAIN	USUBJID	FT.FTSEQ	FT.FTGRPID	FT.FTREFID	FT.FTSPID	FT.FTTESTCD	FT.FTTEST	FT.FTCAT	FT.FTSCA
QS	STUDYID	DOMAIN	USUBJID	QS.QSSEQ	QS.QSGRPID	QS.QSSPID	QS.QSTESTCD	QS.QSTEST	QS.QSCAT	QS.QSSCAT	QS.QSOR
RS	STUDYID	DOMAIN	USUBJID	RS.RSSEQ	RS.RSGRPID	RS.RSREFID	RS.RSSPID	RS.RSLNKID	RS.RSLNKGRP	RS.RSTESTCD	RS.RSTES
SC	STUDYID	DOMAIN	USUBJID	SC.SCSEQ	SC.SCGRPID	SC.SCSPID	SC.SCTESTCD	SC.SCTEST	SC.SCCAT	SC.SCSCAT	SC.SCOR
SS	STUDYID	DOMAIN	USUBJID	SS.SSSEQ	SS.SSGRPID	SS.SSSPID	SS.SSTESTCD	SS.SSTEST	SS.SSCAT	SS.SSSCAT	SS.SSORI
TU	STUDYID	DOMAIN	USUBJID	TU.TUSEQ	TU.TUGRPID	TU.TUREFID	TU.TUSPID	TU.TULNKID	TU.TULNKGRP	TU.TUTESTCD	TU.TUTES
TR	STUDYID	DOMAIN	USUBJID	TR.TRSEQ	TR.TRGRPID	TR.TRREFID	TR.TRSPID	TR.TRLNKID	TR.TRLNKGRP	TR.TRTESTCD	TR.TRTES
VS	STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSGRPID	VS.VSSPID	VS.VSTESTCD	VS.VSTEST	VS.VSCAT	VS.VSSCAT	VS.VSPOS
FA	STUDYID	DOMAIN	USUBJID	FA.FASEQ	FA.FAGRPID	FA.FASPID	FA.FATESTCD	FA.FATEST	FA.FAOBJ	FA.FACAT	FA.FASCA
SR	STUDYID	DOMAIN	USUBJID	SR.SRSEQ	SR.SRGRPID	SR.SRREFID	SR.SRSPID	SR.SRTESTCD	SR.SRTEST	SR.SROBJ	SR.SRCA
TA	STUDYID	DOMAIN	TA.ARMCD	TA.ARM	TA.TAETORD	TA.ETCD	TA.ELEMENT	TA.TABRANCH	TA.TATRANS	TA.EPOCH	
TE	STUDYID	DOMAIN	TE.ETCD	TE.ELEMENT	TE.TESTRL	TE.TEENRL	TE.TEDUR				
TV	STUDYID	DOMAIN	TV.VISITNUM	TV.VISIT	TV.VISITDY	TV.ARMCD	TV.ARM	TV.TVSTRL	TV.TVENRL		
TD	STUDYID	DOMAIN	TD.TDORDER	TD.TDANCVAR	TD.TDSTOFF	TD.TDTGTPAI	TD.TDMINPAI	TD.TDMAXPAI	TD.TDNUMRPT		
TM	STUDYID	DOMAIN	TM.MIDSTYPE	TM.TMDEF	TM.TMRPT						

The division line between the two sides of the screen can be dragged, in order to see more or less of each side of the screen.

It probably has already been noticed that some of the SDTM variables are colored red, some blue and other ones green. The red ones are the ones that are designated as being "required“ in the SDTM-IG, the blue ones those that are designated as being "expected“, and the green ones those that are "permissible“.

In order to obtain more information about a specific variable, just hover the mouse over a cell, e.g.:

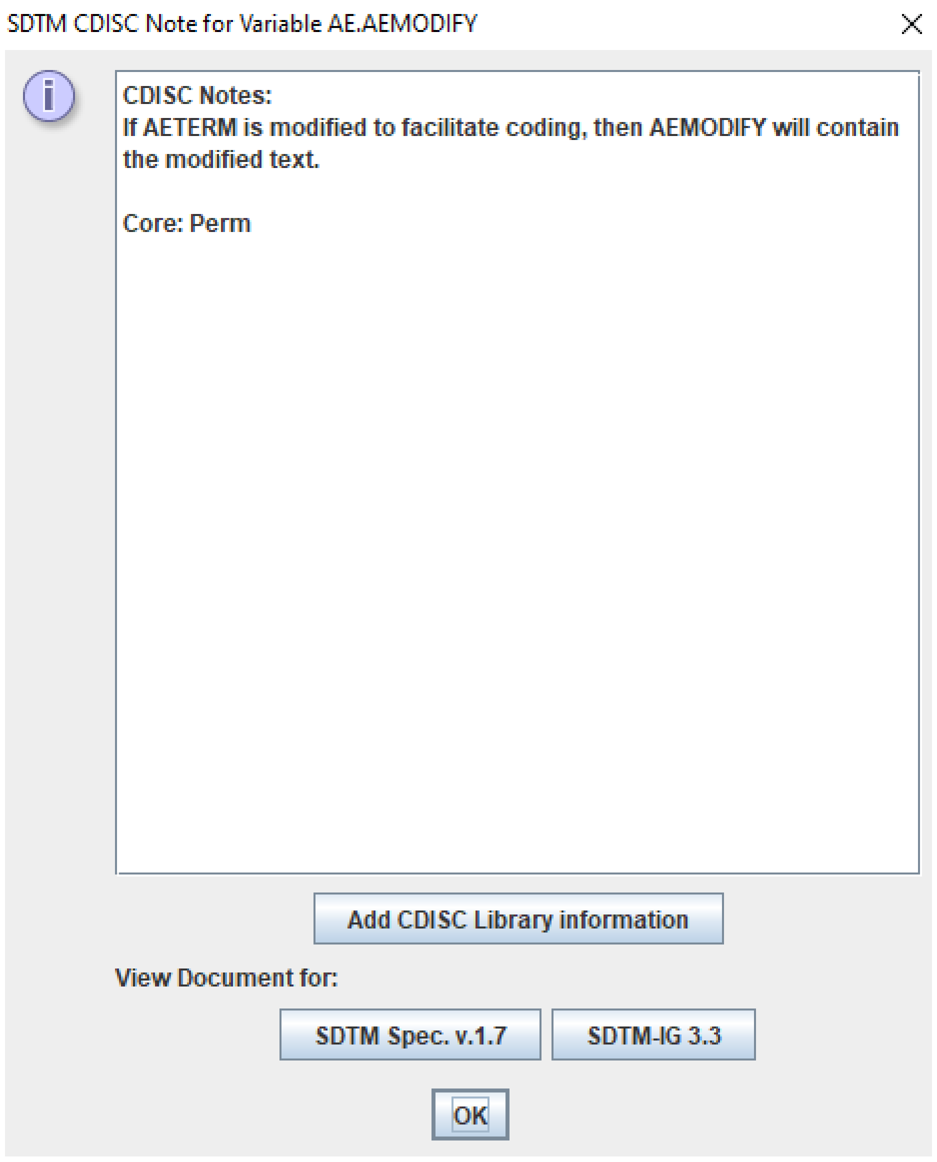


AE.AEMODIFY	AE.AELLT	AE.AELLTCD
CE.CEDECOD	CE.CECAT	CE.CESCAT
DV.DVCAT		
HO.HODECOD		
DS.DSDECOD		
MH.MHMODIFY		
DA.DATEST		
DD.DDRESCAT		
EG.EGTEST		

AE.AEMODIFY
Mandatory: No
OrderNumber: 9
Role: Synonym Qualifier
ItemDef/SDTM Name: AEMODIFY
Data type: text
Length: 80

One also sees that currently the "maximal length“ for this variable has been set to 80. Later it will be demonstrated how this value can be adapted to a more suitable value in agreement with what is in the collected data.

In order to get real in-depth information about a specific SDTM variable, select the cell, and then use "View – SDTM CDISC Notes“ or use CTRL-H. A new window is then displayed, e.g. for AEMODIFY:


SDTM CDISC Note for Variable AE.AEMODIFY

CDISC Notes:
If AETERM is modified to facilitate coding, then AEMODIFY will contain the modified text.

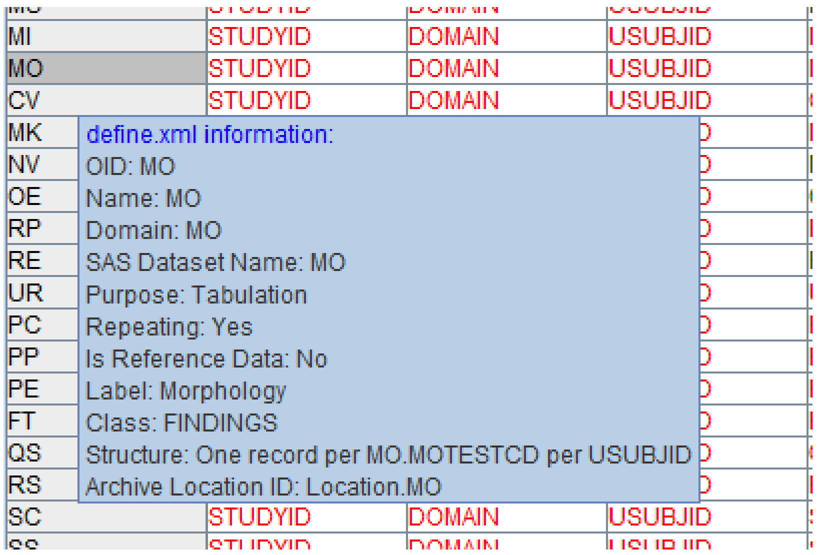
Core: Perm

One can then open the corresponding section of either the standard specification or implementation guide (SDTM-IG by either clicking the button "SDTM Spec. v.1.7" or "SDTM-IG v.3.3", as the latter documents come with the distribution[6].

Later we will also learn how to add additional standard variables, and how to add "non-standard" variables that later typically go into "SUPPQUAL".

Now have a look at the first cell in a row. Also here, hovering the mouse displays some more information, e.g.:

[6]One only need to set the path to the favorite PDF viewer in the "properties.dat" file, as explained in the SDTM-ETL installation guide.



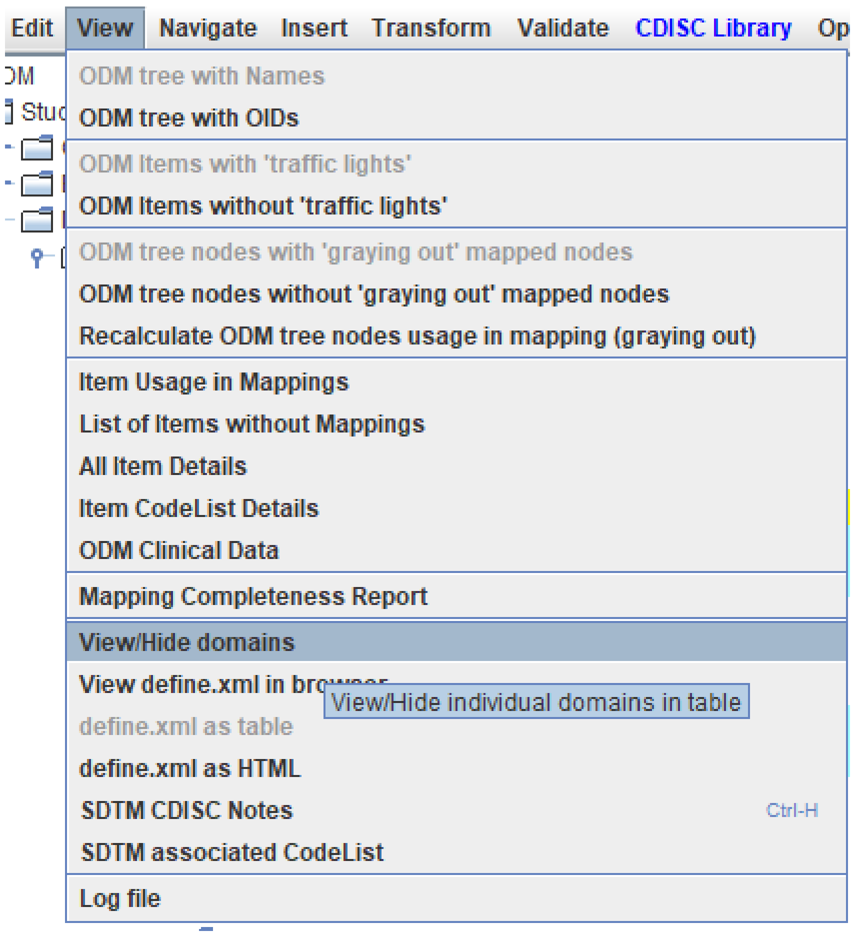
MI	STUDYID	DOMAIN	USUBJID
MO	STUDYID	DOMAIN	USUBJID
CV	STUDYID	DOMAIN	USUBJID
MK			
NV			
OE			
RP			
RE			
UR			
PC			
PP			
PE			
FT			
QS			
RS			
SC	STUDYID	DOMAIN	USUBJID

define.xml information:
OID: MO
Name: MO
Domain: MO
SAS Dataset Name: MO
Purpose: Tabulation
Repeating: Yes
Is Reference Data: No
Label: Morphology
Class: FINDINGS
Structure: One record per MO.MOTESTCD per USUBJID
Archive Location ID: Location.MO

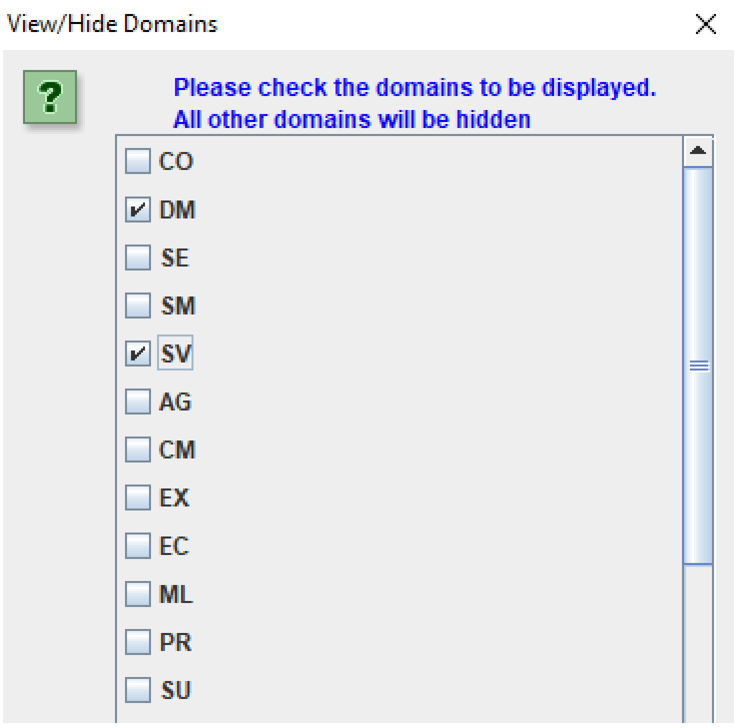
The label for this domain is "Morphology", and it belongs to the "Findings" class. The other information will be explained later when it is explained how the domain properties can be edited.

<u>Viewing and hiding domains</u>

Each new SDTM-IG version has a lot of new domains, and it is easy to loose overview. Therefore individual domains in the table on the right can be hidden or be displayed, so that one can concentrate on the ones that currently are of importance. To do so, use the menu "View – View/Hide domains":


Edit
View
Navigate
Insert
Transform
Validate
CDISC Library
ODM tree with Names
ODM tree with OIDs
ODM Items with 'traffic lights'
ODM Items without 'traffic lights'
ODM tree nodes with 'graying out' mapped nodes
ODM tree nodes without 'graying out' mapped nodes
Recalculate ODM tree nodes usage in mapping (graying out)
Item Usage in Mappings
List of Items without Mappings
All Item Details
Item CodeList Details
ODM Clinical Data
Mapping Completeness Report
View/Hide domains
View define.xml in browser
View/Hide individual domains in table
define.xml as table
define.xml as HTML
SDTM CDISC Notes Ctrl-H
SDTM associated CodeList
Log file

A list of domains then is displayed, and we can check the ones that we want to keep displayed in the table (all others are hidden). For the moment, we just keep DM (Demography) and "SV“ (Subject visits) as these can usually best be mapped first:

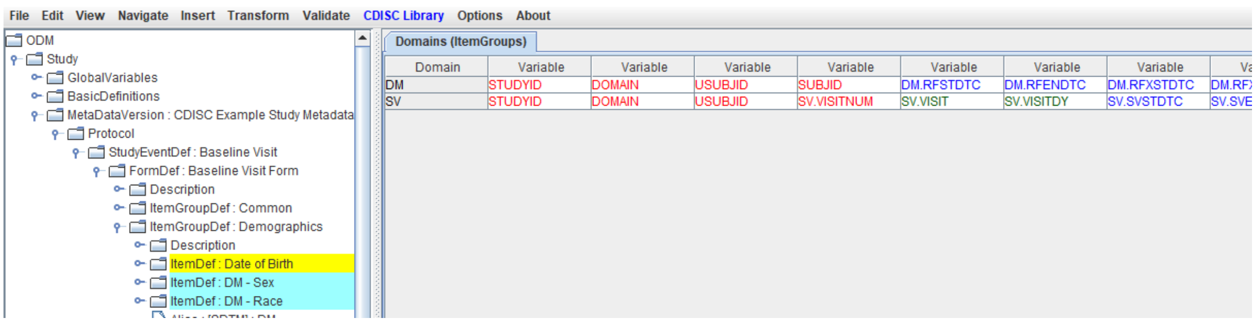


View/Hide Domains

Please check the domains to be displayed.
All other domains will be hidden

- [ ] CO
- [x] DM
- [ ] SE
- [ ] SM
- [x] SV
- [ ] AG
- [ ] CM
- [ ] EX
- [ ] EC
- [ ] ML
- [ ] PR
- [ ] SU

After clicking "OK“, the table on the right reduces to:


File Edit View Navigate Insert Transform Validate CDISC Library Options About

```
ODM
  Study
    GlobalVariables
    BasicDefinitions
    MetaDataVersion : CDISC Example Study Metadata
      Protocol
        StudyEventDef : Baseline Visit
          FormDef : Baseline Visit Form
            Description
            ItemGroupDef : Common
            ItemGroupDef : Demographics
              Description
              ItemDef : Date of Birth
              ItemDef : DM - Sex
              ItemDef : DM - Race
```

Domains (ItemGroups)

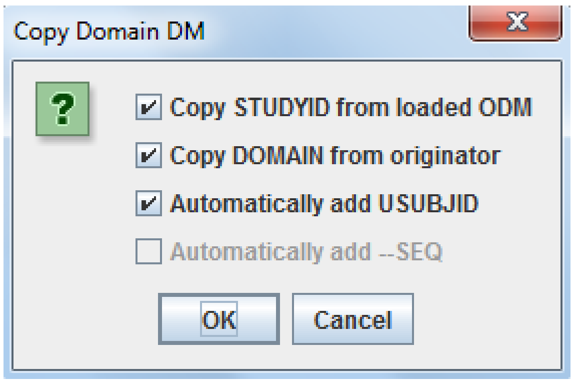
Domain	Variable	Variable	Variable	Variable	Variable	Variable	Variable	Va
DM	STUDYID	DOMAIN	USUBJID	SUBJID	DM.RFSTDTC	DM.RFENDTC	DM.RFXSTDTC	DM.RF
SV	STUDYID	DOMAIN	USUBJID	SV.VISITNUM	SV.VISIT	SV.VISITDY	SV.SVSTDTC	SV.SVE

The mapping can begin...
As we do not want to edit the template domains themselves (well, it is not possible within the tool anyway), we need to create a **study-specific instance**. We will start with the DM domain.
There are two ways to do so:

1) drag-an-drop the "DM“ row to the last row (which in our case is the "SV“) using the mouse with the left mouse button down (release the left mouse button to "drop“)

2) select one of the cells of the "DM“ row and use the menu "Edit – Copy Domain/Dataset“ (or use CTRL-B). Then select the last row of the table, and use the menu "Edit – Past Domain/Dataset“ (or use CTRL-U)

In both cases, the following dialog is displayed:


Copy Domain DM

[✓] Copy STUDYID from loaded ODM

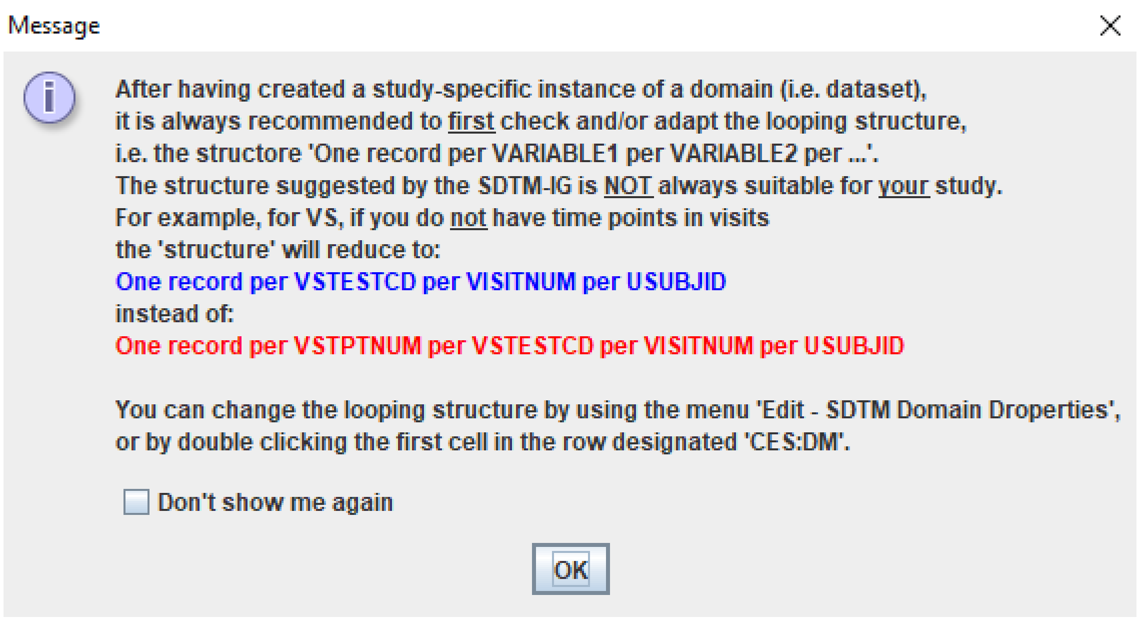
[✓] Copy DOMAIN from originator

[✓] Automatically add USUBJID

[ ] Automatically add --SEQ

The three first checkboxes are already checked in advance. The first means that the value for "STUDYID" in the SDTM will automatically be set to the value of the Study OID in the ODM (which is usually a wise decision).
The second will fix the value of the SDTM variable "DOMAIN" to the one from the template. This is almost always the case – later we will see in which cases one might want to make an exception.
The third tells the system that for the SDTM variable USUBJID, it can take the value from the ODM, i.e. from the "SubjectKey" attribute of the "SubjectData" element in the ODM file with clinical data.
The fourth checkbox allows to have the --SEQ variable be calculated automatically by the system. In the "DM" domain however, there is not DMSEQ variable, so this checkbox is disabled here.

After clicking "OK", an information message is displayed:


Message

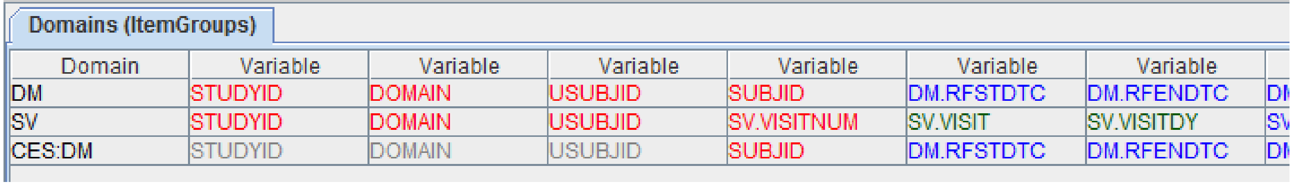
After having created a study-specific instance of a domain (i.e. dataset),
it is always recommended to first check and/or adapt the looping structure,
i.e. the structore 'One record per VARIABLE1 per VARIABLE2 per ...'.
The structure suggested by the SDTM-IG is NOT always suitable for your study.
For example, for VS, if you do not have time points in visits
the 'structure' will reduce to:
One record per VSTESTCD per VISITNUM per USUBJID
instead of:
One record per VSTPTNUM per VSTESTCD per VISITNUM per USUBJID

You can change the looping structure by using the menu 'Edit - SDTM Domain Droperties',
or by double clicking the first cell in the row designated 'CES:DM'.

[ ] Don't show me again

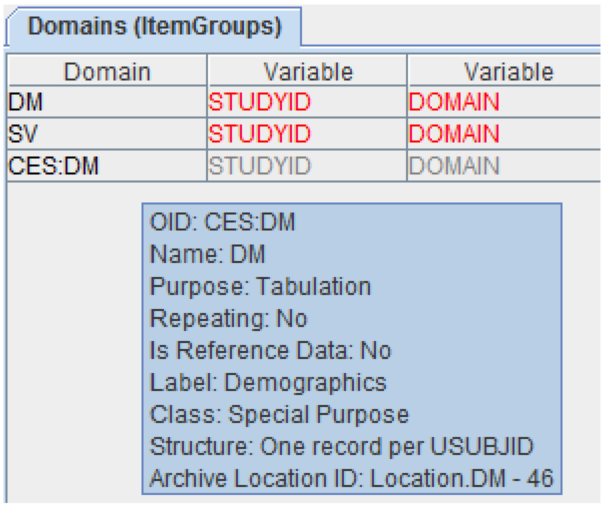
Stating that it is always a good idea to first look at, and often adapt the "structure" of the dataset. For this, don't automatically rely on what is provided in the SDTM- or SEND-IG: how the dataset is structures is **your choice**. What is provided in the SDTM-IG is just one of the possibilities.
If one does not want to see this information message each time a new study-specific instance is created, check the checkbox "Don't show me again".

Clicking the OK button leads to our first mappings.


Domains (ItemGroups)

Domain	Variable	Variable	Variable	Variable	Variable	Variable	
DM	STUDYID	DOMAIN	USUBJID	SUBJID	DM.RFSTDTC	DM.RFENDTC	DI
SV	STUDYID	DOMAIN	USUBJID	SV.VISITNUM	SV.VISIT	SV.VISITDY	SV
CES:DM	STUDYID	DOMAIN	USUBJID	SUBJID	DM.RFSTDTC	DM.RFENDTC	DI

One sees that a new row has been created, with the name (OID in the define.xml) "CES:DM" for our study-specific DM domain. The color of three cells (STUDYID, DOMAIN and USUBJID) is changed to grey, meaning that a mapping script for these variables now exists.
Hovering the mouse over the first cell (CES:DM) shows:

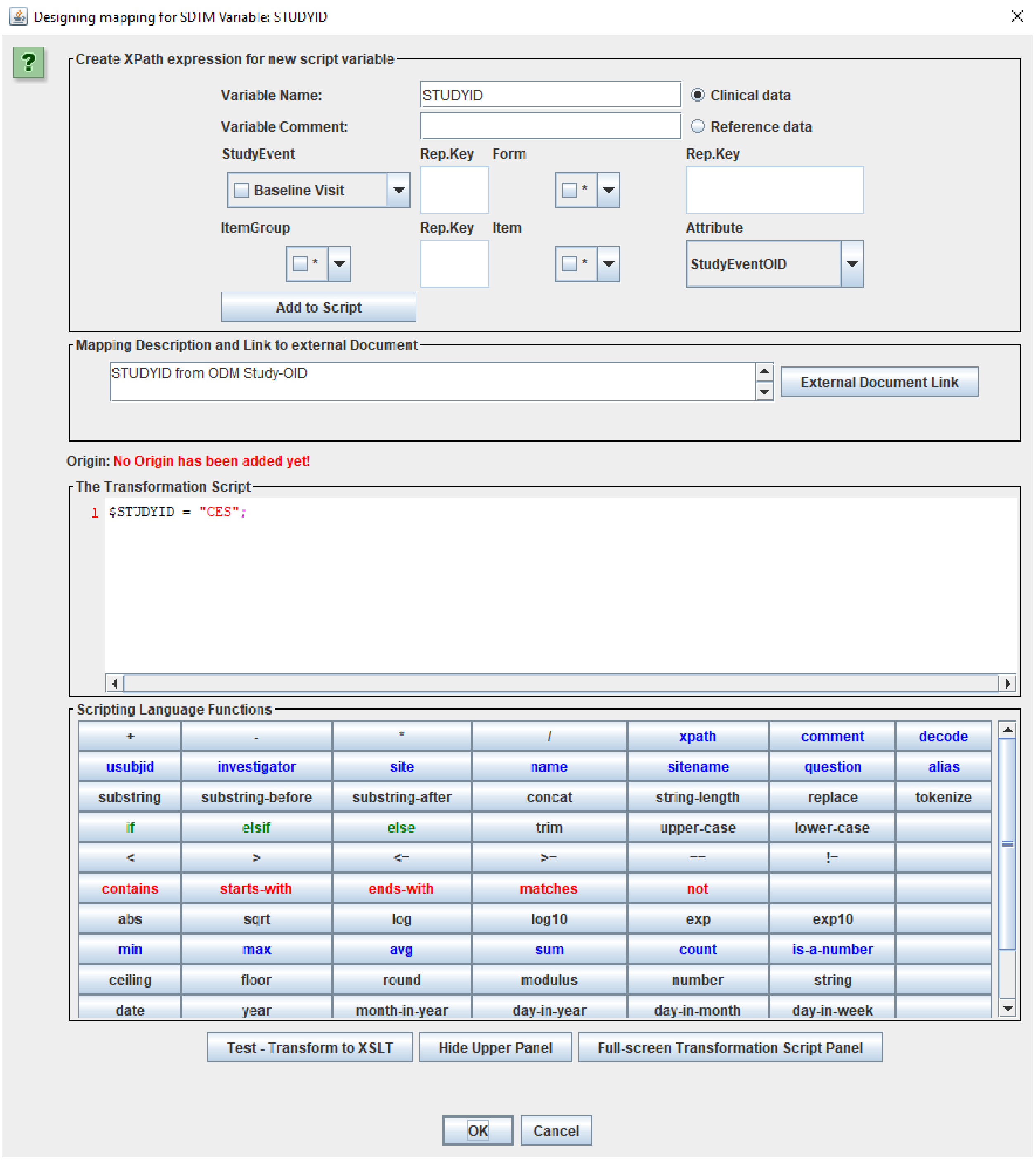

Domains (ItemGroups)

Domain	Variable	Variable
DM	STUDYID	DOMAIN
SV	STUDYID	DOMAIN
CES:DM	STUDYID	DOMAIN

OID: CES:DM
Name: DM
Purpose: Tabulation
Repeating: No
Is Reference Data: No
Label: Demographics
Class: Special Purpose
Structure: One record per USUBJID
Archive Location ID: Location.DM - 46

Later we will learn how to edit the properties of the domain instance. In the case of the DM domain there is currently no necessity to do so.

The mapping for a specific variable (e.g. "STUDYID“) can be edited by double-clicking the cell. This leads to a new window that opens and shows:



Designing mapping for SDTM Variable: STUDYID

Create XPath expression for new script variable

Variable Name: STUDYID
Variable Comment:
(● Clinical data) (○ Reference data)
StudyEvent: Baseline Visit
Attribute: StudyEventOID

Mapping Description and Link to external Document: STUDYID from ODM Study-OID

Origin: No Origin has been added yet!

The Transformation Script

```
$STUDYID = "CES";
```

Scripting Language Functions

+	-	*	/	xpath	comment	decode
usubjid	investigator	site	name	sitename	question	alias
substring	substring-before	substring-after	concat	string-length	replace	tokenize
if	elsif	else	trim	upper-case	lower-case	
<	>	<=	>=	==	!=	
contains	starts-with	ends-with	matches	not		
abs	sqrt	log	log10	exp	exp10	
min	max	avg	sum	count	is-a-number	
ceiling	floor	round	modulus	number	string	
date	year	month-in-year	day-in-year	day-in-month	day-in-week	

This window is named the "**mapping editor**", which we will use a lot. Let us first look at the basic features of this mapping editor.

The upper panel is for advanced usage when complicated selections for items must be made. It can be hidden by using the button "Hide Upper Panel“.

The smaller panel "Mapping Description“ has already been prefilled. It contains a short description of the mapping. Please feel free to edit its text. Remark that for "derived" variables, the text will later flow into the "Method description" in the define.xml.

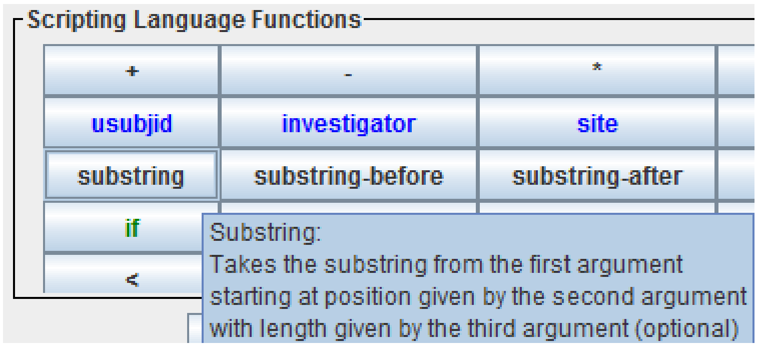
As of SDTM-ETL v.4.1, it is also indicated whether an "Origin" has been defined for the variable. Remind that in the case of a regulatory submission, the origin must be provided for each variable. It will later be explained in more detail how an origin can be assigned or edited.

The most important panel is the panel "The Transformation Script“. This is where the script is generated and/or edited. The scripting is in a special, easy-to-learn language. Although most of the scripts are generated automatically, it will be necessary to learn about this scripting language, which is described in a special document "SDTM-ETL Scripting Language“.
In the current case the mapping script is very simply:

$STUDYID = "CES";

stating the the variable STUDYID is a string (remark the quotes) with a fixed value of "CES“. Also notice the semicolon at the end marking the end of the statement.

The lower panel "Scripting Language Functions" contain a series of buttons for generating snippets of coding involving build-in functions. To get more explanation about a specific function, just hover the mouse over a button, e.g.:



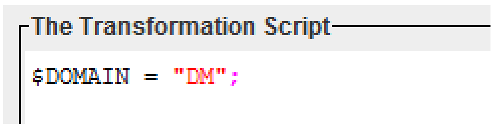
Substring:
Takes the substring from the first argument starting at position given by the second argument with length given by the third argument (optional)

We will later treat the use of functions in detail.

For very complicated mappings (which I hope is the minority – but that depends on your study design), one can "blow up“ the central panel using the button "Full Screen Transformation Script Panel" which generates a full screen script editor panel.

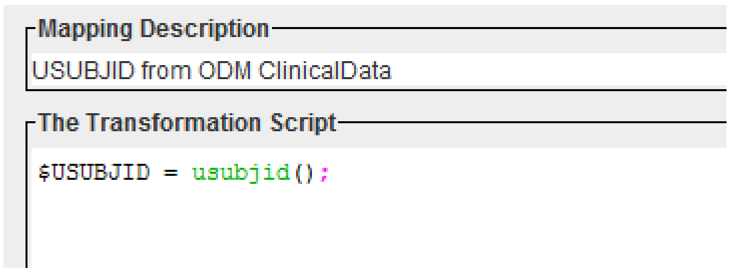
When done editing the mapping script, click the "OK“ button, or use the "Cancel" button to cancel all editing.

For the DM variables "DOMAIN“ a similar mapping has already been generated automatically:



```
$DOMAIN = "DM";
```

Double-clicking the cell "USUBJID" provides the mapping for the variable "USUBJID":


Mapping Description

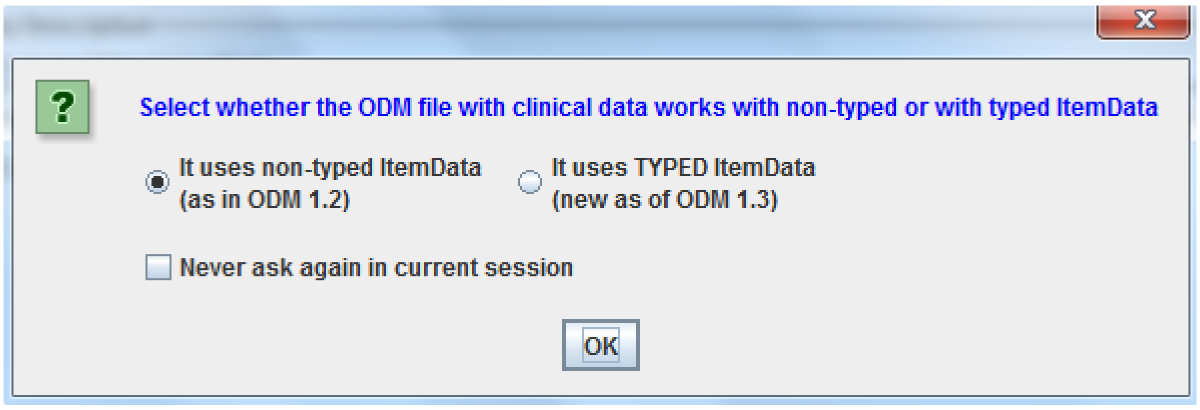
USUBJID from ODM ClinicalData

The Transformation Script

```
$USUBJID = usubjid();
```

The field "Mapping Description" has been pre-filled (but you can edit it) stating that the value will be taken from the ODM ClinicalData.
The transformation script itself uses a function usubjid(), which simply takes the value of the "SubjectKey" attribute of the SubjectData element in the ODM file with clinical data.
Later in this tutorial, we will see how one can edit this, e.g. to set the value of the USUBJID as a concatenation of the study ID with the subject ID.

Let us now test this mapping on a real set of clinical data. For this, click the button "Test – Transform to XSLT". This will generate a mapping script in XSLT language[7] (which you do not need to learn) to transform XML files or to extract information from XML files such as CDISC ODM files with clinical data.
The result of clicking the button "Test – Transform to XSLT" is a new window:

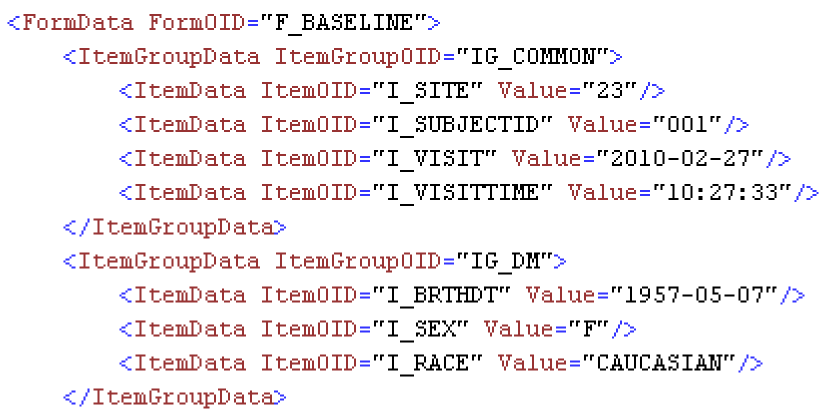

Select whether the ODM file with clinical data works with non-typed or with typed ItemData

[✓ It uses non-typed ItemData (as in ODM 1.2)] It uses TYPED ItemData (new as of ODM 1.3)

☐ Never ask again in current session

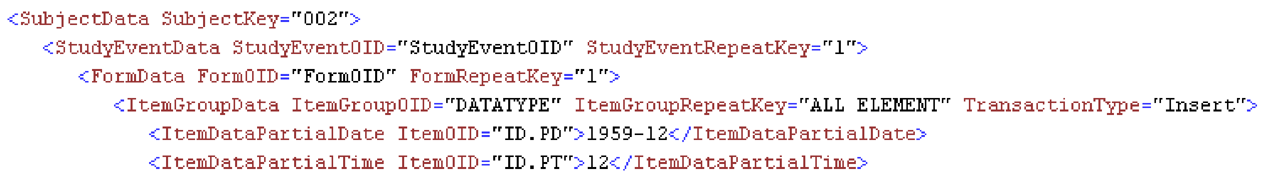
It asks you whether your ODM clinical data is "non-typed" or "typed". If you don't know, ask your EDC vendor or the source of your clinicalm data, or just try one of both possibilities (you will immediately find out which one applies). You can also have a quick look at a file with clinical data. In case you find a lot of "ItemData“ elements with a "Value" attribute, this means that your data is "untyped". For example:

[7]XSLT is an international standard from the W3C for transforming XML documents



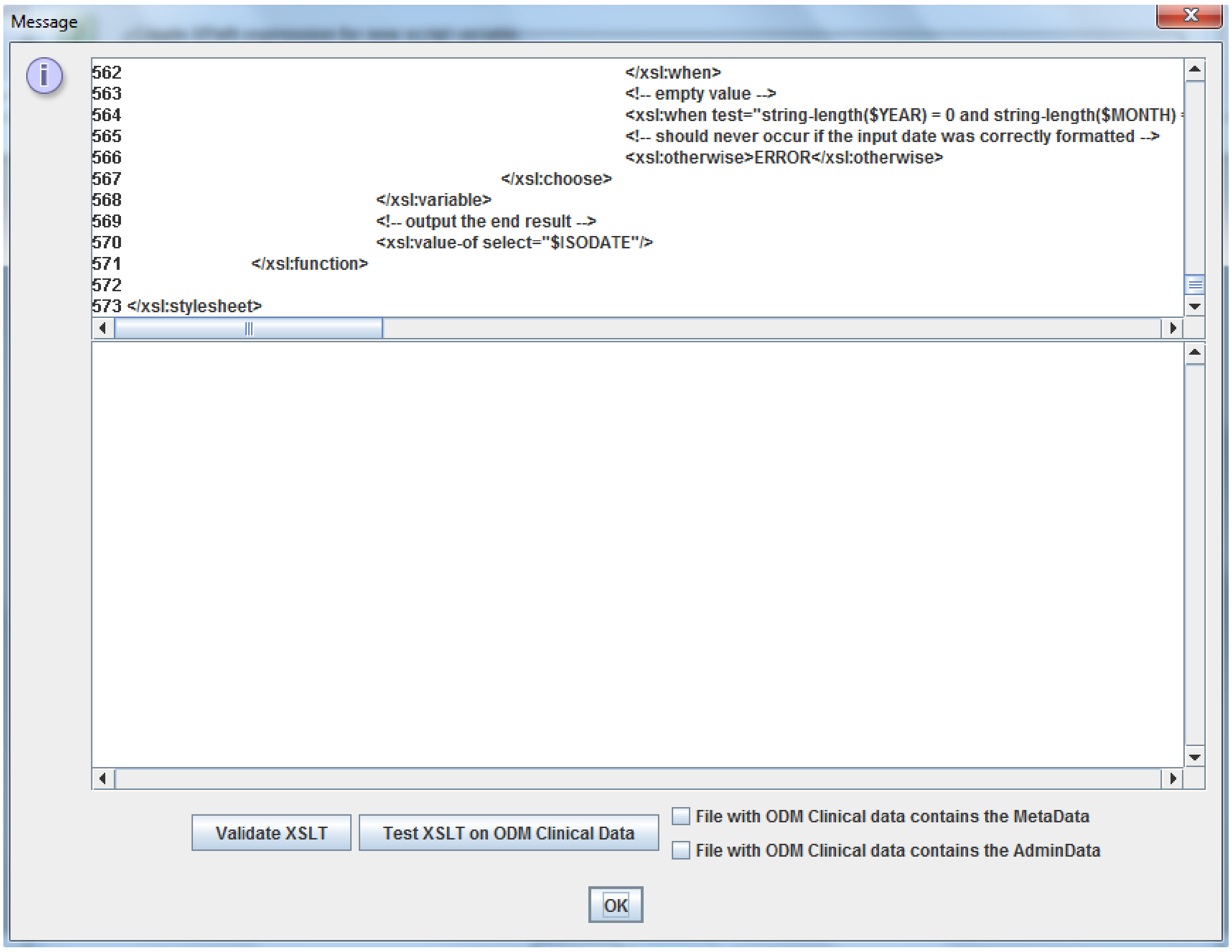
```
<FormData FormOID="F_BASELINE">
    <ItemGroupData ItemGroupOID="IG_COMMON">
        <ItemData ItemOID="I_SITE" Value="23"/>
        <ItemData ItemOID="I_SUBJECTID" Value="001"/>
        <ItemData ItemOID="I_VISIT" Value="2010-02-27"/>
        <ItemData ItemOID="I_VISITTIME" Value="10:27:33"/>
    </ItemGroupData>
    <ItemGroupData ItemGroupOID="IG_DM">
        <ItemData ItemOID="I_BRTHDT" Value="1957-05-07"/>
        <ItemData ItemOID="I_SEX" Value="F"/>
        <ItemData ItemOID="I_RACE" Value="CAUCASIAN"/>
    </ItemGroupData>
```

If your data however contain elements like "ItemDataString" or "ItemDataDate" and there is no "Value" attribute, this means that your data is "typed". For example:



```
<SubjectData SubjectKey="002">
   <StudyEventData StudyEventOID="StudyEventOID" StudyEventRepeatKey="1">
      <FormData FormOID="FormOID" FormRepeatKey="1">
         <ItemGroupData ItemGroupOID="DATATYPE" ItemGroupRepeatKey="ALL ELEMENT" TransactionType="Insert">
            <ItemDataPartialDate ItemOID="ID.PD">1959-12</ItemDataPartialDate>
            <ItemDataPartialTime ItemOID="ID.PT">12</ItemDataPartialTime>
```

In our case, we work with "untyped" data, so we leave the radiobutton "It uses non-typed ItemData" selected. If it is sure that your clinical data will always come as "untyped", one can check the checkbox "Never ask again in current session", and then this dialog will not show up again. Clicking "OK" leads to a dialog:

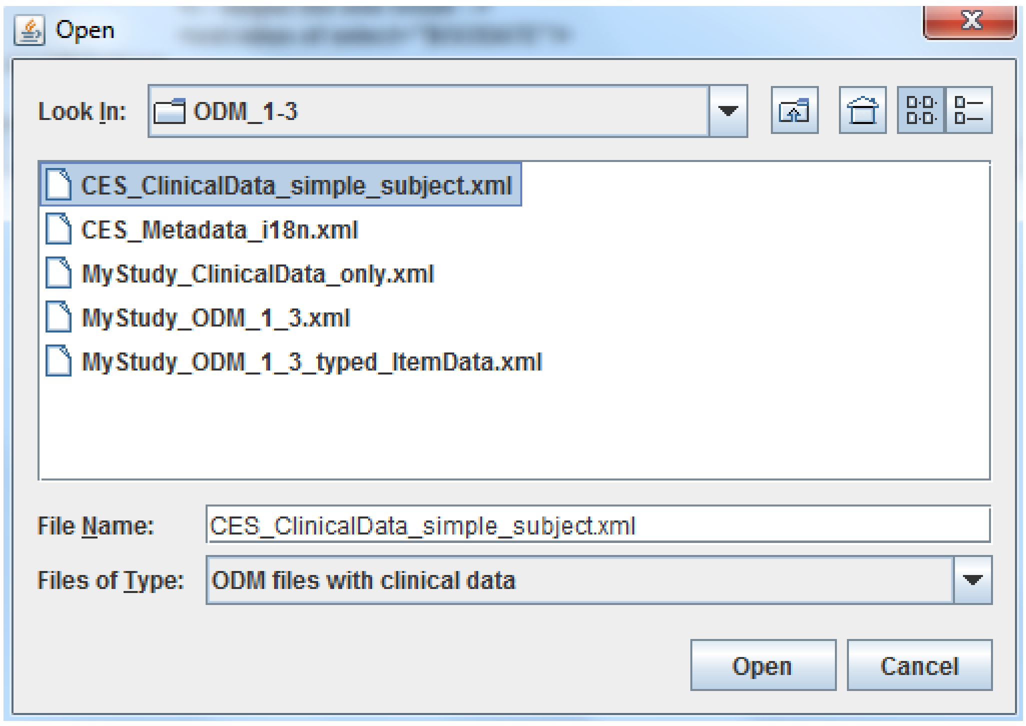


```
                                                    </xsl:when>
                                                    <!-- empty value -->
                                                    <xsl:when test="string-length($YEAR) = 0 and string-length($MONTH)
                                                    <!-- should never occur if the input date was correctly formatted -->
                                                    <xsl:otherwise>ERROR</xsl:otherwise>
                                       </xsl:choose>
                           </xsl:variable>
                           <!-- output the end result -->
                           <xsl:value-of select="$ISODATE"/>
               </xsl:function>

</xsl:stylesheet>
```

Validate XSLT
Test XSLT on ODM Clinical Data
File with ODM Clinical data contains the MetaData
File with ODM Clinical data contains the AdminData
OK

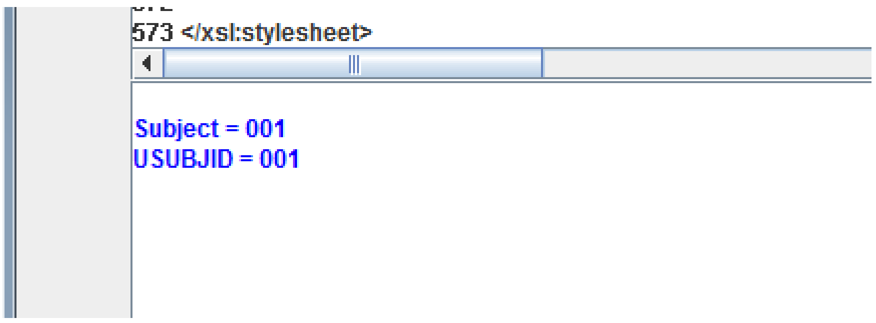
One can then validate the correctness of the generated XSLT, or just inspect it (specialists with very complicated scripts like to do so for debugging). In 99% of the cases, you will however just want to continue by clicking the "Test XSLT on ODM Clinical Data". This leads to a filechooser allowing to pick the ODM file with clinical data. For example:



```
CES_ClinicalData_simple_subject.xml
CES_Metadata_i18n.xml
MyStudy_ClinicalData_only.xml
MyStudy_ODM_1_3.xml
MyStudy_ODM_1_3_typed_ItemData.xml
```

File Name: CES_ClinicalData_simple_subject.xml
Files of Type: ODM files with clinical data

Clicking "Open" then immediately executes the script. As our file only contains the data for a single subject, the output is:

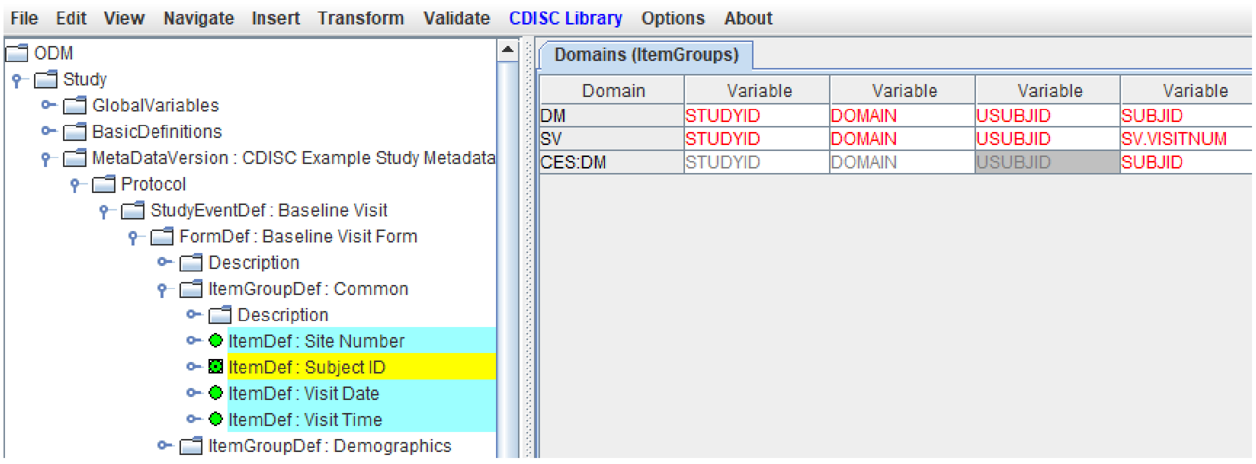


```
573 </xsl:stylesheet>
```

```
Subject = 001
USUBJID = 001
```

Notice that this testing mechanism only works for a single variable in a single domain, and e.g. not when the the script references other variables from the same or other domains. Later we will learn how to do more sophisticated testing.

Let us now generate an alternative mapping for USUBJID. For example, we would like to have the value of USUBJID to be a concatenation of the STUDYID and of the subject ID from the "Common" section of each form. For doing so, first select the cell "USUBJID" and then expand the tree with the study design so that you see an item "Subject ID“ in a group of items "Common". One can of course also do a search in the study design tree (see the document "Loading ODM"). For example:

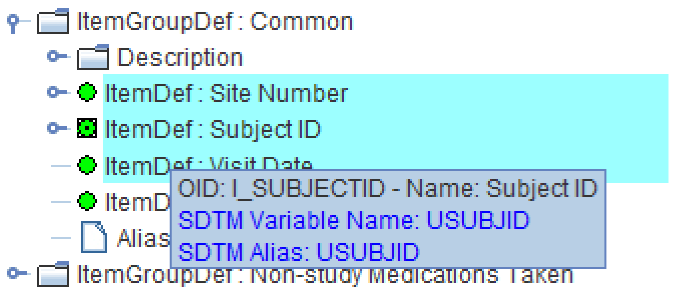


```
ODM
  Study
    GlobalVariables
    BasicDefinitions
    MetaDataVersion : CDISC Example Study Metadata
      Protocol
        StudyEventDef : Baseline Visit
          FormDef : Baseline Visit Form
            Description
            ItemGroupDef : Common
              Description
              ItemDef : Site Number
              ItemDef : Subject ID
              ItemDef : Visit Date
              ItemDef : Visit Time
            ItemGroupDef : Demographics
```

Domain	Variable	Variable	Variable	Variable
DM	STUDYID	DOMAIN	USUBJID	SUBJID
SV	STUDYID	DOMAIN	USUBJID	SV.VISITNUM
CES:DM	STUDYID	DOMAIN	USUBJID	SUBJID

If one looks carefully, two important observations can be made:
a) the items that are visible have a green "traffic light" in front of them
b) the item "Subject ID" has a traffic light that has a **square** around it

The green "traffic light" means that the item is of a suitable data type for mapping to the SDTM variable. For example, if one expects a datetime for an SDTM variable, the traffic light on the item "Subject ID" in the study design tree will be red[8]. The square around the green "traffic light" means that the item is a "hot candidate", i.e. has been annotated in the ODM as being ideally suited for mapping with the given SDTM variable.
This can also be seen by hovering the mouse over the item "Subject ID" in the study design tree:



```
ItemGroupDef : Common
  Description
  ItemDef : Site Number
  ItemDef : Subject ID
  ItemDef : Visit Date
  ItemD
  Alias
ItemGroupDef : Non-study Medications Taken
```

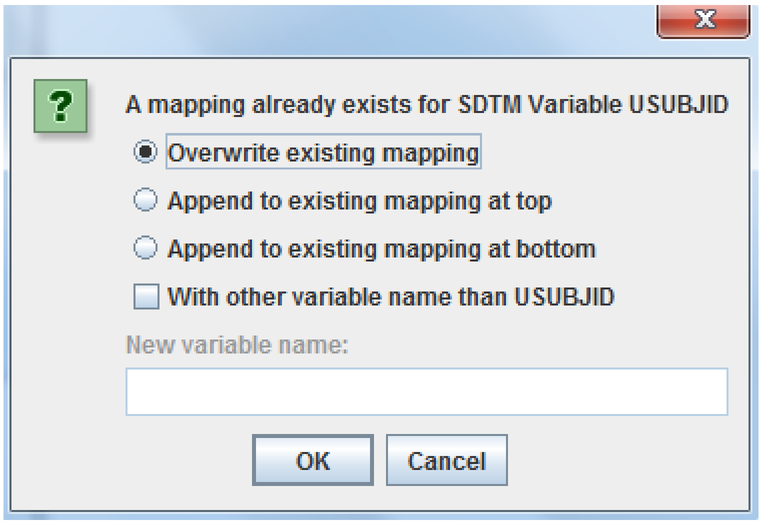
```
OID: I_SUBJECTID - Name: Subject ID
SDTM Variable Name: USUBJID
SDTM Alias: USUBJID
```

Technically, this was done by adding the attribute SDSVarName="USUBJID" in the ODM study design[9].

To use the item "Subject ID" in the mapping for the SDTM variable "USUBJID", select the item "Subject ID" in the tree with the mouse, then drag it (keep the left mouse button down) to the cell "USUBJID" in the table on the right, then drop it by releasing the left mouse button. During the dragging, you will see a yellow "copy" symbol replacing your mouse cursor, meaning that you are in the "copy" mode.

After having dropped in the "USUBJID" cell, the following dialog is displayed:

[8]Which does not mean that it cannot be used in that mapping – people drive through red traffic lights, but that is taking a big risk ...
[9] It is always a very good idea to develop the study design "with the end" (the submission) "in mind". More modern study designers such as the "ODM Designer" allow this.


A mapping already exists for SDTM Variable USUBJID

● Overwrite existing mapping

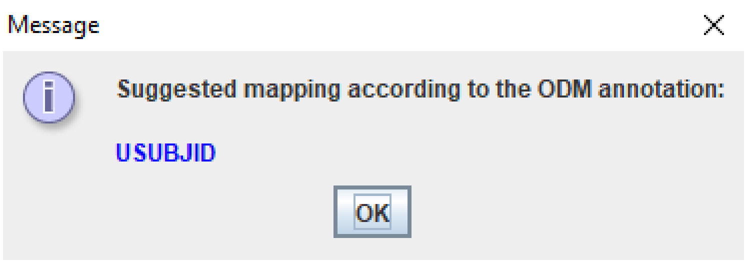
○ Append to existing mapping at top

○ Append to existing mapping at bottom

☐ With other variable name than USUBJID

New variable name:

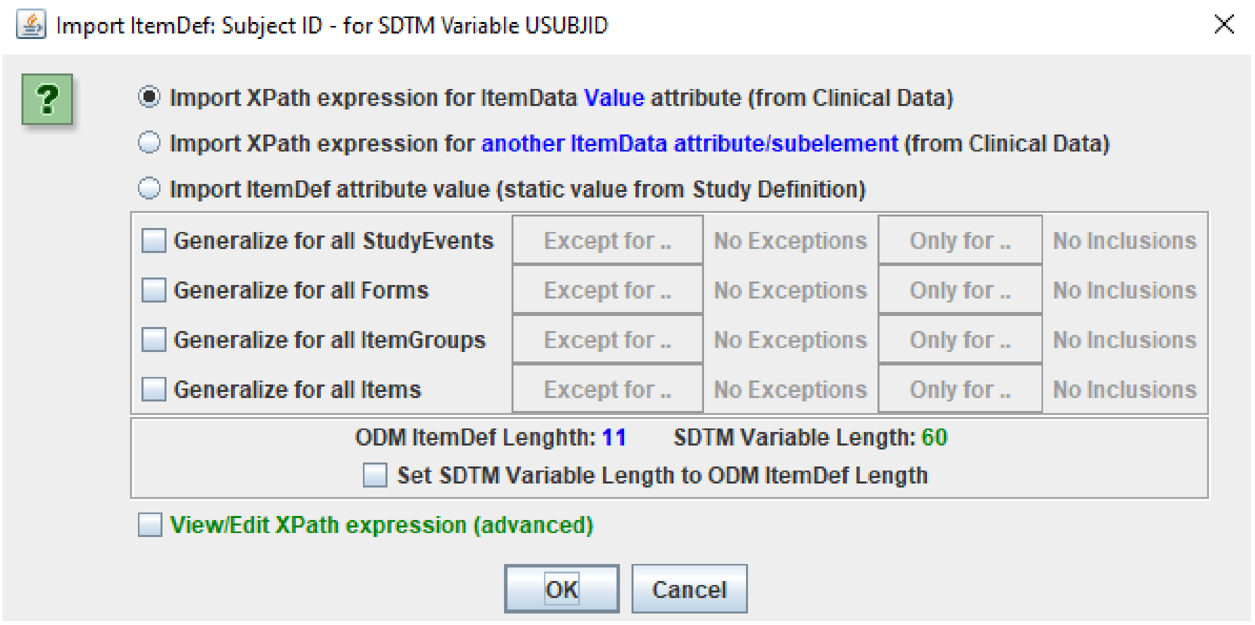
as a mapping already exists for USUBJID. Select "Overwrite existing mapping" and click "OK". As the ODM ItemDef is having an SDTM annotation that it maps to "USUBJID", an information message is displayed:


Message

Suggested mapping according to the ODM annotation:

USUBJID

This displays a new dialog:


Import ItemDef: Subject ID - for SDTM Variable USUBJID

● Import XPath expression for ItemData Value attribute (from Clinical Data)

○ Import XPath expression for another ItemData attribute/subelement (from Clinical Data)

○ Import ItemDef attribute value (static value from Study Definition)

☐ Generalize for all StudyEvents	Except for ..	No Exceptions	Only for ..	No Inclusions
☐ Generalize for all Forms	Except for ..	No Exceptions	Only for ..	No Inclusions
☐ Generalize for all ItemGroups	Except for ..	No Exceptions	Only for ..	No Inclusions
☐ Generalize for all Items	Except for ..	No Exceptions	Only for ..	No Inclusions

ODM ItemDef Lenghth: 11 SDTM Variable Length: 60

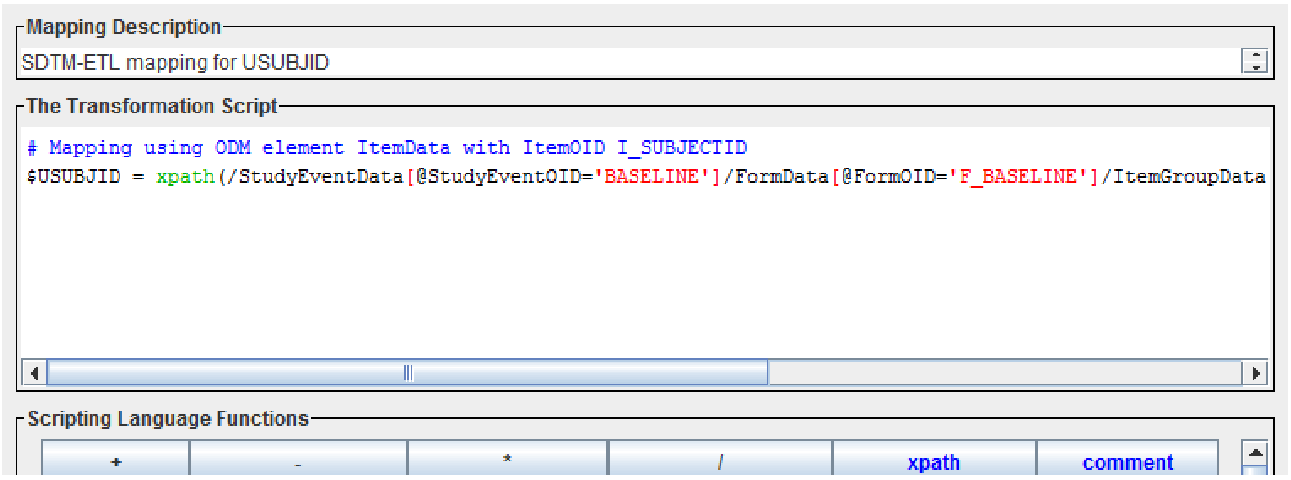
☐ Set SDTM Variable Length to ODM ItemDef Length

☐ View/Edit XPath expression (advanced)

The most important radiobutton is the button "Import Xpath expression for ItemData Value attribute (from Clinical Data) meaning that we want to import a collected value (this will be >90% of the cases). We will come to the function of the other radiobuttons later.
The lower part of the dialog states that we currently have set the maximal length for USUBJID to 60 (being the default) from the template, but that the maximal length in the study was defined to be 11. Checking the box "Set SDTM Variable Length to ODM ItemDef Length" allows to reduce the

SDTM variable length to the one given in the study design, wich is 11.
Don't check the checkbox for now, as we still want to concatenate with the Study ID.

After clicking the OK button, the mapping scripting shows up:

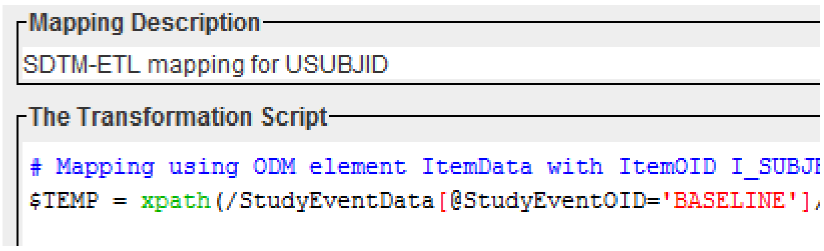


SDTM-ETL mapping for USUBJID

```
# Mapping using ODM element ItemData with ItemOID I_SUBJECTID
$USUBJID = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupData
```

Essentially what is does, is to define a path to the item in the clinical data, and store the result in the variable $USUBJID. As it is a path in XML, this is called an "XPath" expression.
One can now test this script again on clinical data (as before), giving the same result as before.

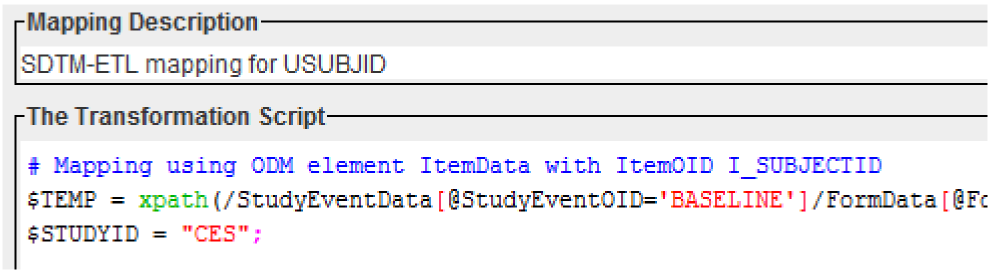
Now we want to concatenate the value of STUDYID with the above result. In order to do so, we need to adapt the script slightly. First, the variable $USUBJID is renamed into $TEMP. We then have:



SDTM-ETL mapping for USUBJID

```
# Mapping using ODM element ItemData with ItemOID I_SUBJE
$TEMP = xpath(/StudyEventData[@StudyEventOID='BASELINE'],
```

Do not change anything in the "XPath" expression[10].
Now, we do already have a mapping for the SDTM variable $STUDYID. We can just copy-paste from the previous mapping which results in:



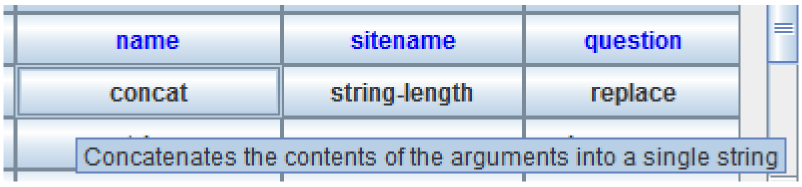
SDTM-ETL mapping for USUBJID

```
# Mapping using ODM element ItemData with ItemOID I_SUBJECTID
$TEMP = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@Fc
$STUDYID = "CES";
```

Now have a look at the functions in the lower panel, the "Scripting Language Functions“ panel. You

[10] It will be very seldom that one needs to change something in the XPath expression. We will give some examples later though

will find a "concat“ function with the following explanation:



name	sitename	question
concat	string-length	replace

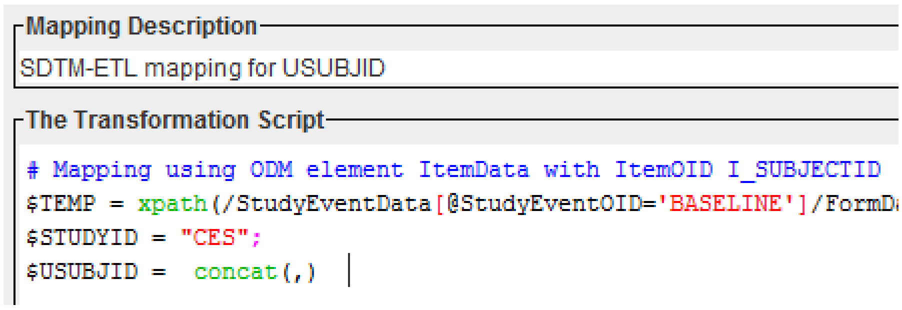
Concatenates the contents of the arguments into a single string

The "concat“ function has at least two arguments, but there can be more. It is used to concatenate a set of strings into a new string.

Now in the mapping script editor, just type:

$USUBJID =

and then click the "concat“ button. The string is extended with the function with empty parameters:



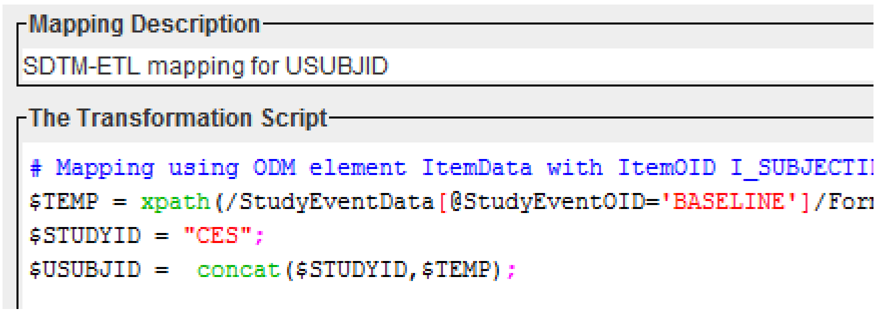
Mapping Description

SDTM-ETL mapping for USUBJID

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_SUBJECTID
$TEMP = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormD
$STUDYID = "CES";
$USUBJID =  concat(,)
```

which can now easily be extended as:



Mapping Description

SDTM-ETL mapping for USUBJID

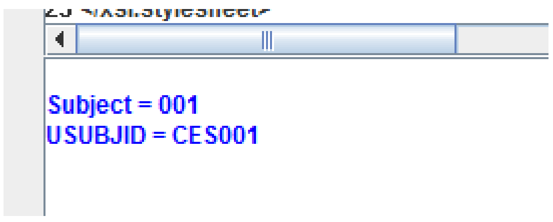
The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_SUBJECTI
$TEMP = xpath(/StudyEventData[@StudyEventOID='BASELINE']/For
$STUDYID = "CES";
$USUBJID =  concat($STUDYID,$TEMP);
```

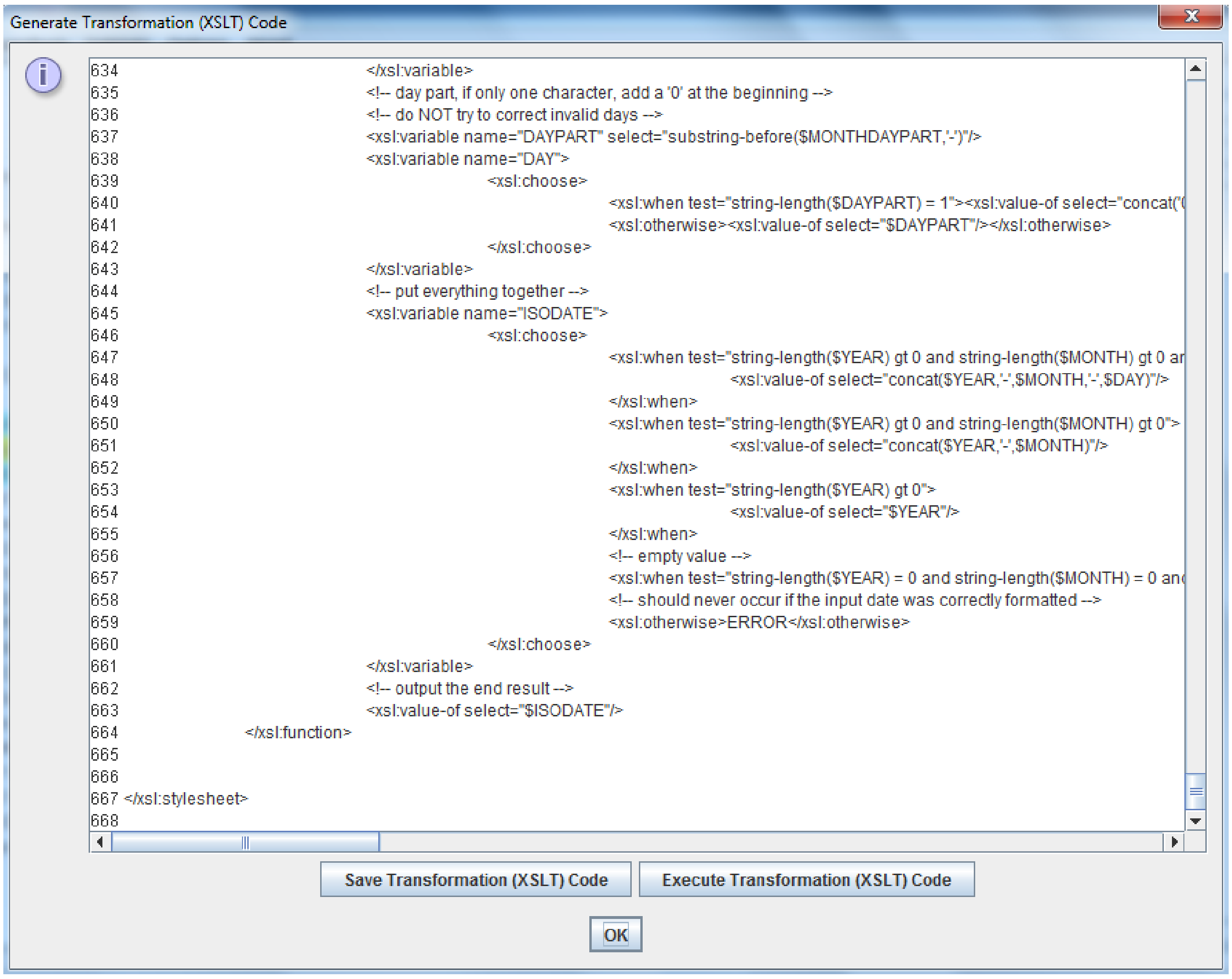
Do not forget the semicolon at the end[11].
You might already have noticed the coloring in the script: **comments** (starting with a "#“) are colored blue[12]. Functions are colored green, and strings (that are between quotes) are colored red.

Re-executing the mapping script on real clinical data results in:

[11]If the semicolon is forgotten, a warning message will be displayed when trying to execute the mapping.
[12]As in every programming effort, it is advised to add as many comments as possible, for a later good understanding what the intention of the statement or snippet was.


```
Subject = 001
USUBJID = CES001
```

One can also execute all the available mappings together. After clicking OK for the mapping script editor, we come into the main window again. Now, use the menu "Transform – Generate Transformation (XSLT) Code for SAS-XPT" or use "Transform - Generate Transformation (XSLT) Code for CDISC Dataset-XML"[13]. The former will generate data files in the newer CDISC Dataset-XML format, the latter in the classic SAS-XPT format. Let us first try the classic way[14]. The following dialog is presented:


Generate Transformation (XSLT) Code

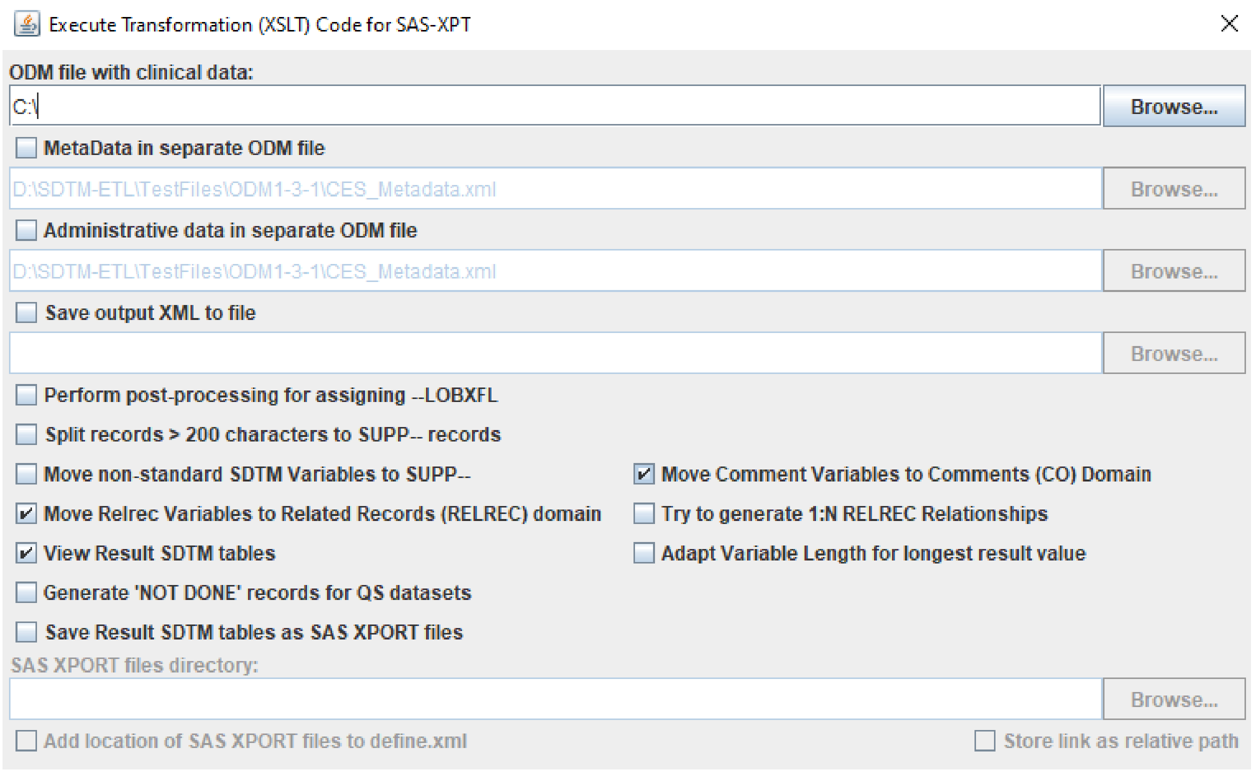
```
                </xsl:variable>
                <!-- day part, if only one character, add a '0' at the beginning -->
                <!-- do NOT try to correct invalid days -->
                <xsl:variable name="DAYPART" select="substring-before($MONTHDAYPART,'-')"/>
                <xsl:variable name="DAY">
                    <xsl:choose>
                        <xsl:when test="string-length($DAYPART) = 1"><xsl:value-of select="concat('(
                        <xsl:otherwise><xsl:value-of select="$DAYPART"/></xsl:otherwise>
                    </xsl:choose>
                </xsl:variable>
                <!-- put everything together -->
                <xsl:variable name="ISODATE">
                    <xsl:choose>
                        <xsl:when test="string-length($YEAR) gt 0 and string-length($MONTH) gt 0 ar
                            <xsl:value-of select="concat($YEAR,'-',$MONTH,'-',$DAY)"/>
                        </xsl:when>
                        <xsl:when test="string-length($YEAR) gt 0 and string-length($MONTH) gt 0">
                            <xsl:value-of select="concat($YEAR,'-',$MONTH)"/>
                        </xsl:when>
                        <xsl:when test="string-length($YEAR) gt 0">
                            <xsl:value-of select="$YEAR"/>
                        </xsl:when>
                        <!-- empty value -->
                        <xsl:when test="string-length($YEAR) = 0 and string-length($MONTH) = 0 and
                        <!-- should never occur if the input date was correctly formatted -->
                        <xsl:otherwise>ERROR</xsl:otherwise>
                    </xsl:choose>
                </xsl:variable>
                <!-- output the end result -->
                <xsl:value-of select="$ISODATE"/>
        </xsl:function>


</xsl:stylesheet>

```
Save Transformation (XSLT) Code
Execute Transformation (XSLT) Code
OK

One can now save the transformation XSLT code to file[15], but we will execute the code within the software itself, so click "Execute Transformation (XSLT) Code". This results in a new dialog:

13 As of SDTM-ETL version 4.1, one can also choose for the new CDISC Dataset-JSON format, which is envisaged to be the successor of SAS-XPT.
[14] Later, it will explain how to do the same generating results in the new SDS-XML format.
[15] This can be useful to execute the transformations off-line.



Execute Transformation (XSLT) Code for SAS-XPT

ODM file with clinical data: C:\

- [ ] MetaData in separate ODM file — D:\SDTM-ETL\TestFiles\ODM1-3-1\CES_Metadata.xml
- [ ] Administrative data in separate ODM file — D:\SDTM-ETL\TestFiles\ODM1-3-1\CES_Metadata.xml
- [ ] Save output XML to file
- [ ] 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

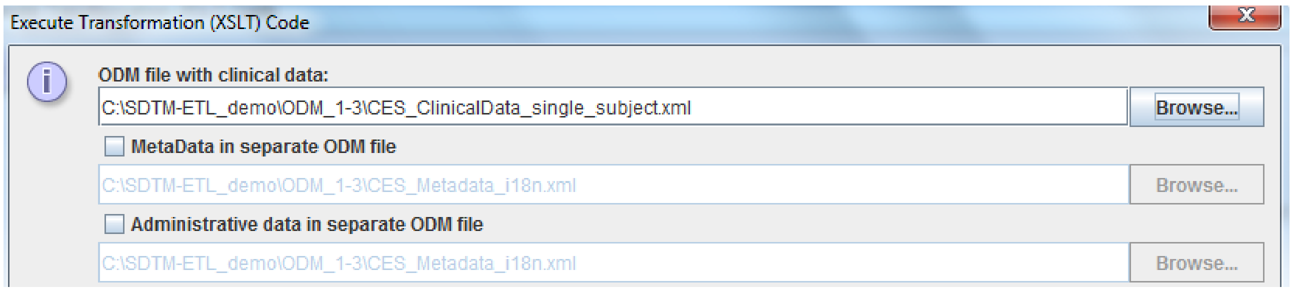
SAS XPORT files directory:

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

The upper field allows to add the location of the ODM file with clinical data. One can use the "Browse“ button to locate this file.

At the moment, we do not need to generate any SAS-XPT files, so we leave the checkbox "Save Result SDTM tables as SAS XPORT files“ unchecked. The checkbox "View Result SDTM files“ remains checked – this will open an own viewer for the results that we have sofar.

One of the advantages of the SDTM-ETL software is that one can start developing the mappings even before the first subject has enrolled. But in order to test the mappings, we need some clinical data, even if it is mock data.
Consider the case that we already have some (but not all) collected data of a first subject. We can already use these data to test our mapping.
For example:

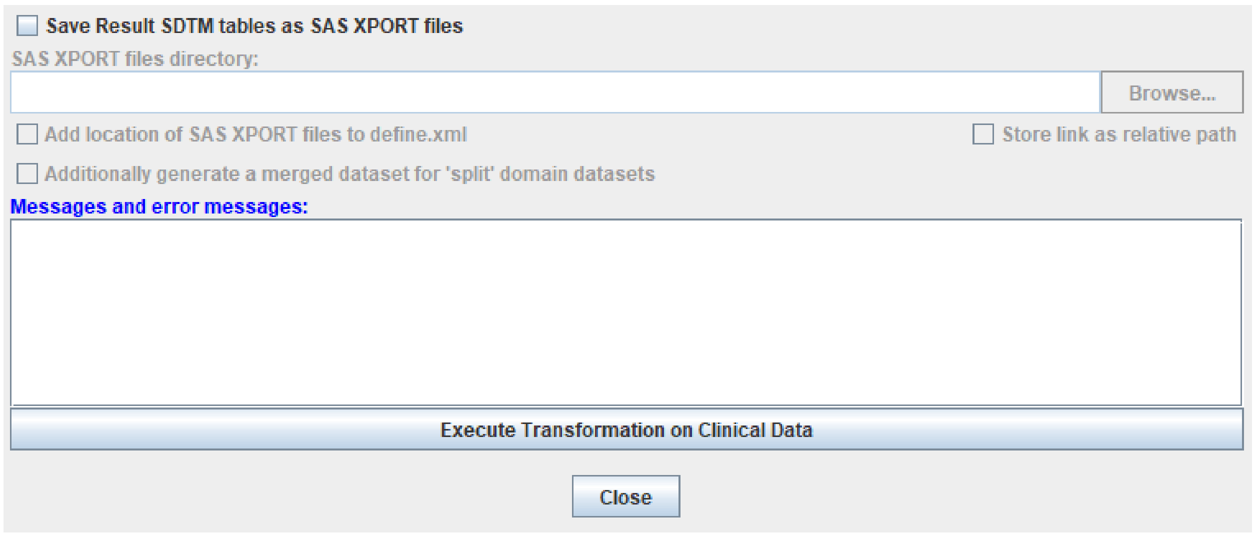


Execute Transformation (XSLT) Code

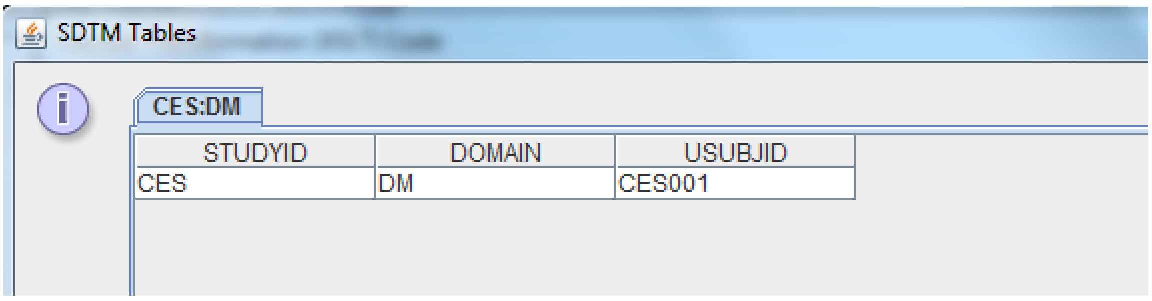
ODM file with clinical data: C:\SDTM-ETL_demo\ODM_1-3\CES_ClinicalData_single_subject.xml

- [ ] MetaData in separate ODM file — C:\SDTM-ETL_demo\ODM_1-3\CES_Metadata_i18n.xml
- [ ] Administrative data in separate ODM file — C:\SDTM-ETL_demo\ODM_1-3\CES_Metadata_i18n.xml

and we can now click the button "Execute Transformation on Clinical Data“


Save Result SDTM tables as 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

After a few seconds, the transformation has been executed, and a new window with the results (those that we have so far) is displayed:



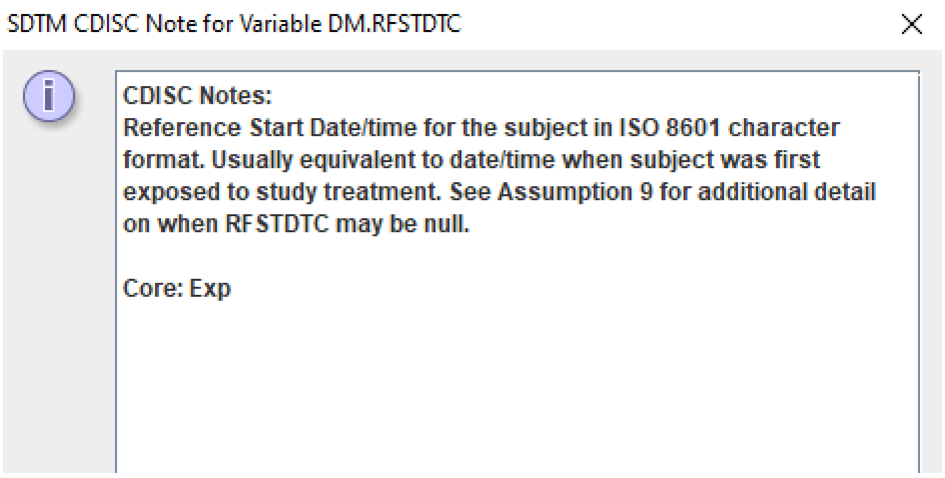
STUDYID	DOMAIN	USUBJID
CES	DM	CES001

Remarks:

If you would liked to have a dash between the study ID and the subject ID for USUBJID, you could have used: $USUBJID = concat($STUDYID, '-', $TEMP);
In the remaining of the tutorial, we will however use the default mapping which is:
$USUBJID = usubjid();
taking the value of the ODM "SubjectKey“ attribute of the "SubjectData“ element.

For the variable SUBJID, we can also use the same mapping $USUBJID=usubjid();
but you can also decide otherwise.

The next variable is RFSTDTC (Reference Start Date/time). In order to get more information on this item use CTRL-H or the menu "View – SDTM CDISC Notes“. This displays the window:

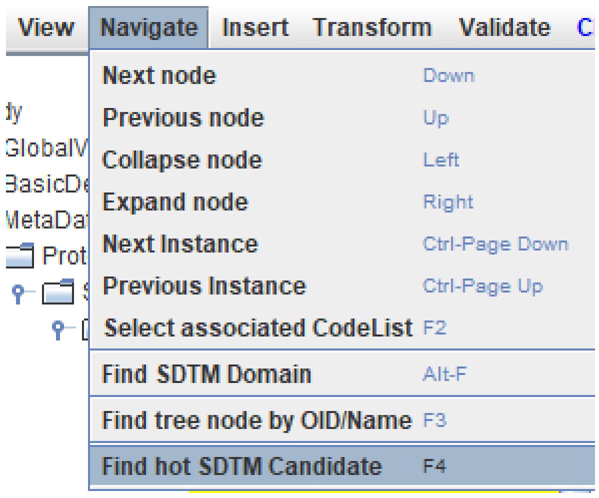

SDTM CDISC Note for Variable DM.RFSTDTC

CDISC Notes:
Reference Start Date/time for the subject in ISO 8601 character format. Usually equivalent to date/time when subject was first exposed to study treatment. See Assumption 9 for additional detail on when RFSTDTC may be null.

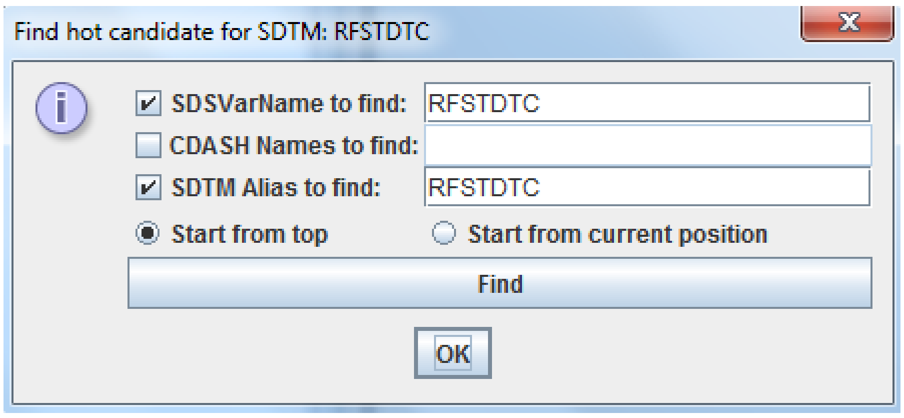
Core: Exp

We can easily map this to the date of the first visit[16].

Maybe there is a "hot candidate" in the ODM for RFSTDTC, i.e. the ODM has been annotated that the item is ideally suited to be used for RFSTDTC. For finding out, first select the RFSTDTC cell and then use the menu "Navigate – Find hot SDTM Candidate"::


View
Navigate
Insert
Transform
Validate
Next node Down
Previous node Up
Collapse node Left
Expand node Right
Next Instance Ctrl-Page Down
Previous Instance Ctrl-Page Up
Select associated CodeList F2
Find SDTM Domain Alt-F
Find tree node by OID/Name F3
Find hot SDTM Candidate F4

The following dialog is displayed:


Find hot candidate for SDTM: RFSTDTC

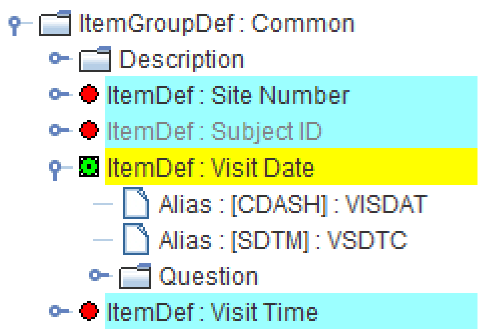
[✓] SDSVarName to find: RFSTDTC
[ ] CDASH Names to find:
[✓] SDTM Alias to find: RFSTDTC
[● Start from top] ○ Start from current position

One can select to search using the "SDSVarName" (meaning "SDTM Variable Name") in the ODM,

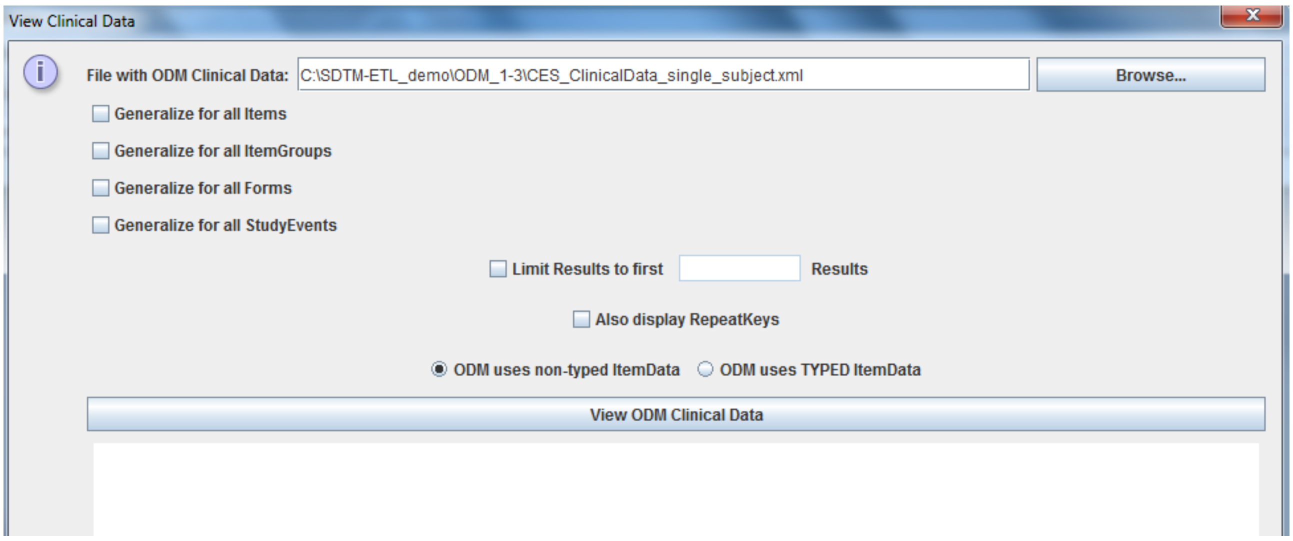
[16]Our very simple sample study does not have a data point for "date of first study treatment". If there is such a data point, the corresponding date can (or even is advised to) be used.

the CDASH name and/or the "SDTM Alias“.

After clicking "Find“, and if there is a "hot candidate“ in the ODM, the tree will automatically expand, and the "hot candidate" item is displayed and selected:



But, are there already any clinical data for this data point? One can test using the menu "View – ODM Clinical Data". This shows the window:


View Clinical Data

File with ODM Clinical Data: C:\SDTM-ETL_demo\ODM_1-3\CES_ClinicalData_single_subject.xml

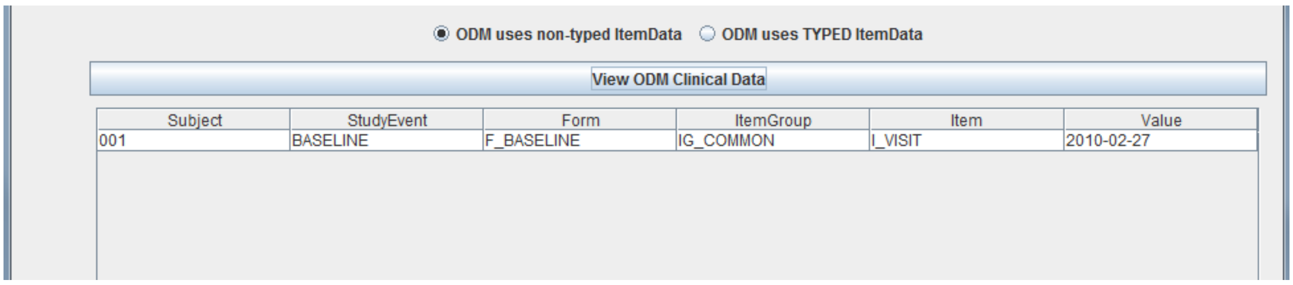
☐ Generalize for all Items
☐ Generalize for all ItemGroups
☐ Generalize for all Forms
☐ Generalize for all StudyEvents

☐ Limit Results to first Results

☐ Also display RepeatKeys

[✓ ODM uses non-typed ItemData] ODM uses TYPED ItemData

As a file with clinical data has already been used for testing, the field "File with ODM Clinical Data" is already pre-filled. So one only need to click the button "View ODM Clinical Data" which results in:

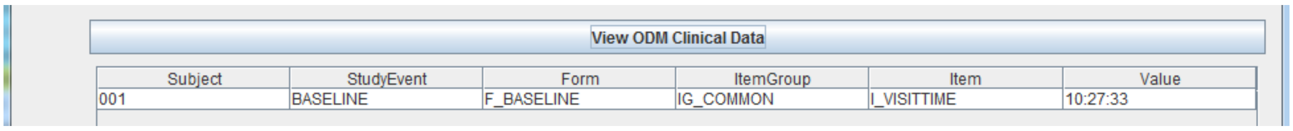

[✓ ODM uses non-typed ItemData] ODM uses TYPED ItemData

Subject	StudyEvent	Form	ItemGroup	Item	Value
001	BASELINE	F_BASELINE	IG_COMMON	I_VISIT	2010-02-27

The rightmost column showing the value, and the other columns the subject ID, the StudyEvent (visit), Form, and ItemGroup, as well as the current Item.

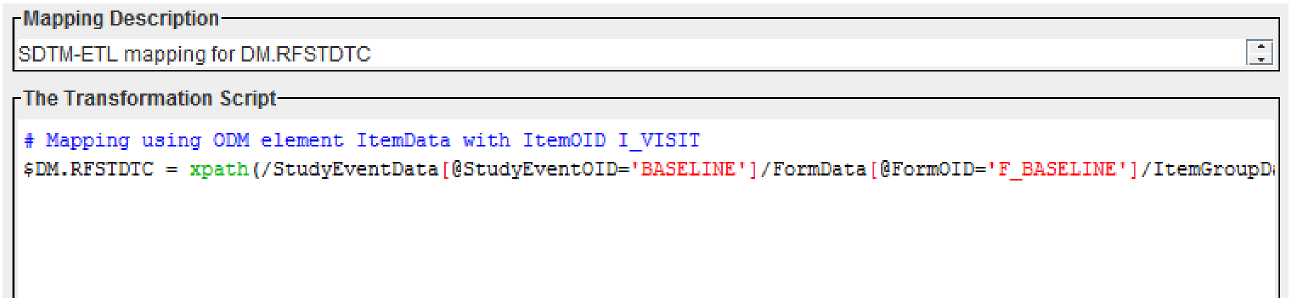
The menu "View – ODM Clinical Data" will often be extremely useful to find out whether the current item is really the one we need or want for the mapping.

The same can be applied to check whether also the time of the first visit was collected using the Item "Visit Time" (OID I_VISIT_TIME):



Subject	StudyEvent	Form	ItemGroup	Item	Value
001	BASELINE	F_BASELINE	IG_COMMON	I_VISITTIME	10:27:33

As as well a visit date as time is present, they can both be used to populate RFSTDTC. To do so, drag the item "Visit Date“ to the cell "RFSTDTC“, leading to:


Mapping Description

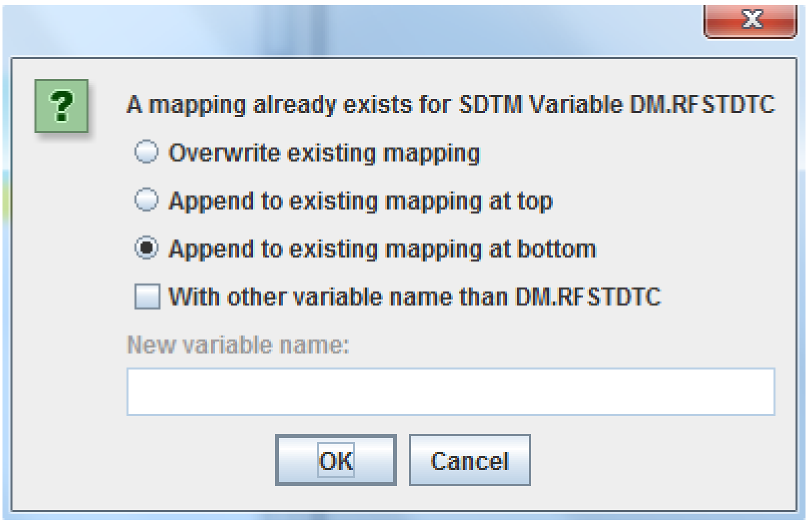
SDTM-ETL mapping for DM.RFSTDTC

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_VISIT
$DM.RFSTDTC = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupDa
```

and rename $DM.RFSTDTC into "$VISITDATE“.

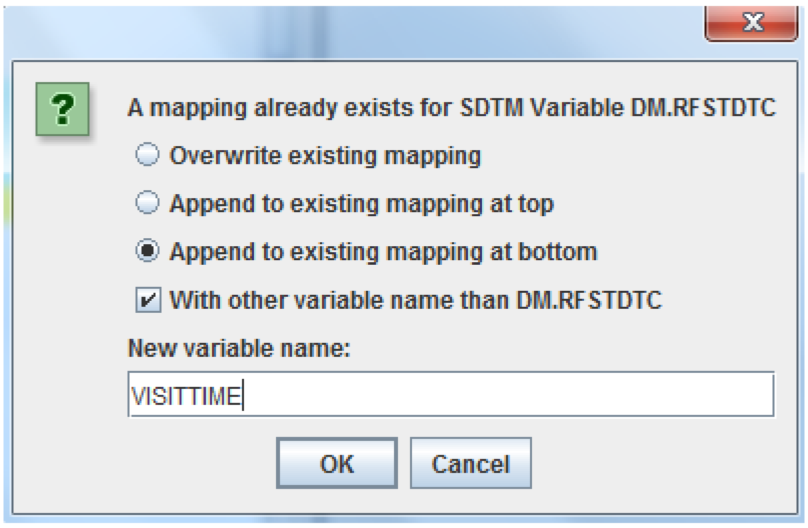
Then drag-and-drop the item "Visit Time“ to the same cell RFSTDTC. The following wizard is displayed:


A mapping already exists for SDTM Variable DM.RFSTDTC

- ○ Overwrite existing mapping
- ○ Append to existing mapping at top
- ● Append to existing mapping at bottom
- ☐ With other variable name than DM.RFSTDTC

New variable name:

We want to append to the existing mapping, but as we still need to combine both items, we choose to rename the current one, e.g. to $VISITTIME:

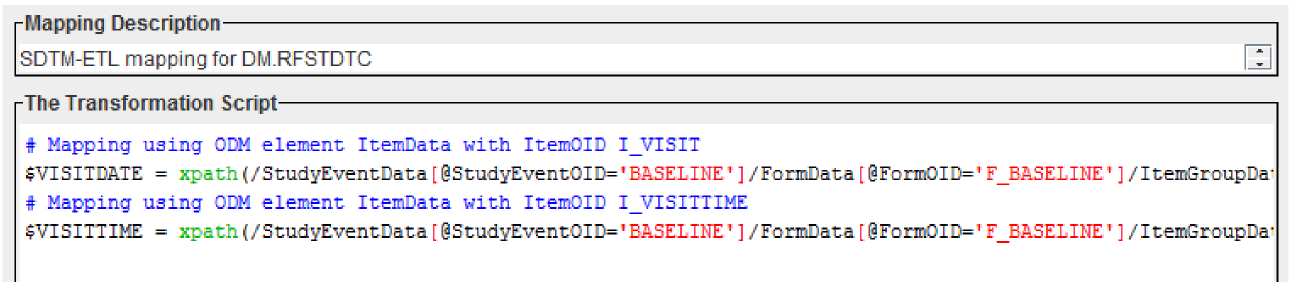

A mapping already exists for SDTM Variable DM.RFSTDTC

( ) Overwrite existing mapping
( ) Append to existing mapping at top
(●) Append to existing mapping at bottom
[✓] With other variable name than DM.RFSTDTC

New variable name:
VISITTIME

You do not need to add a "$“ in front of the new variable name, the system will take care of it.

This results in a mapping:


Mapping Description
SDTM-ETL mapping for DM.RFSTDTC

The Transformation Script
```
# Mapping using ODM element ItemData with ItemOID I_VISIT
$VISITDATE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupDa
# Mapping using ODM element ItemData with ItemOID I_VISITTIME
$VISITTIME = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupDa
```

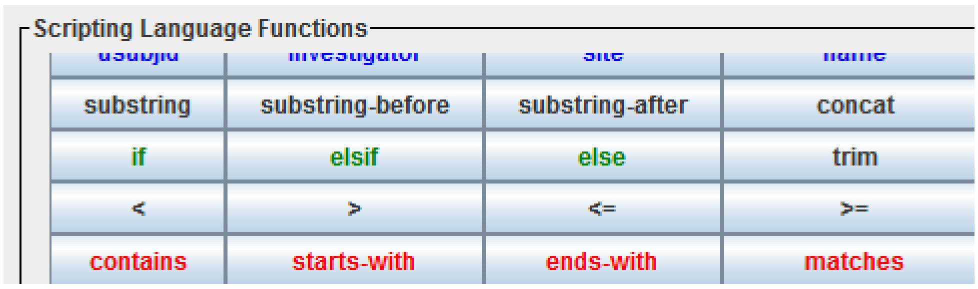
Remark that the two comment lines have been generated automatically.

The SDTM Implementation Guide explains the usage of ISO-8601 dates, times and datetimes. In case of a complete datetime, the format is: YYYY-MM-DDThh:mm:ss. The central "T“ separating the date part from the time part. So for our mapping, we can use:

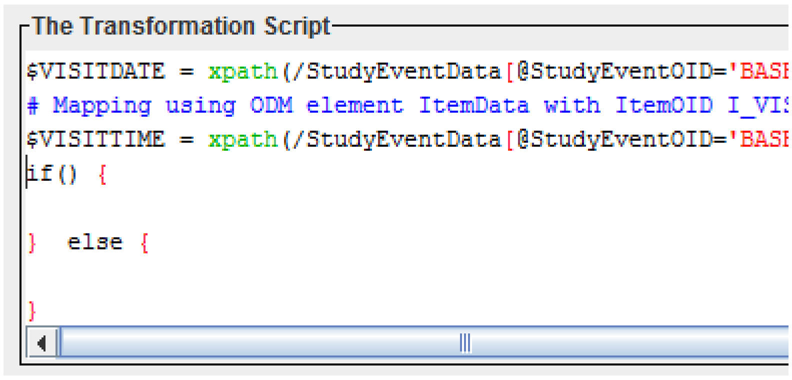
$DM.RFSTDTC = concat($VISITDATE, 'T', $VISITTIME);

Hey, wait a minute! What in the case that the visit time was not collected? Then the central "T“ should not be present! So … **time for our first if-else statement**!

Like for the "concat“ function, one can use the "if“, "elsif“ and "else“ buttons from the "Scripting Language Functions“ panel to insert snippets:



e.g. leading to:

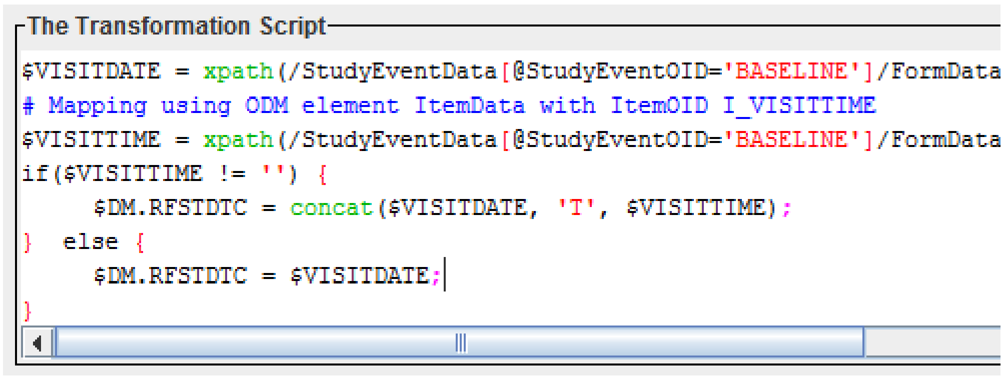

The Transformation Script

```
$VISITDATE = xpath(/StudyEventData[@StudyEventOID='BASE
# Mapping using ODM element ItemData with ItemOID I_VIS
$VISITTIME = xpath(/StudyEventData[@StudyEventOID='BASE
if() {

}  else {

}
```

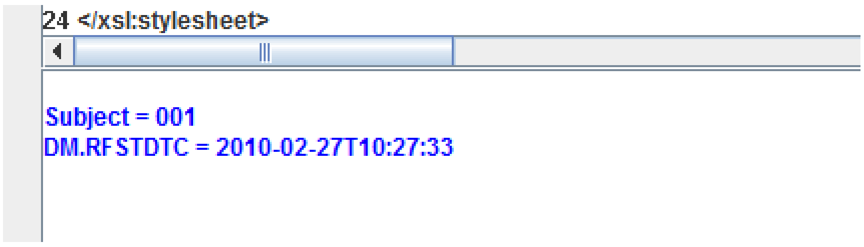
One can then fill in the individual parts:


The Transformation Script

```
$VISITDATE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData
# Mapping using ODM element ItemData with ItemOID I_VISITTIME
$VISITTIME = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData
if($VISITTIME != '') {
     $DM.RFSTDTC = concat($VISITDATE, 'T', $VISITTIME);
}  else {
     $DM.RFSTDTC = $VISITDATE;
}
```

The "if" statement saying that in case the VISITTIME variable is <u>not</u> empty ("!=" symbol), then the value of DM.RFSTDTC is the concatenation of the visit date with the character "T" and the visit time. In any other case ("else" statement), the value of DM.RFSTDTC only consist of the date.

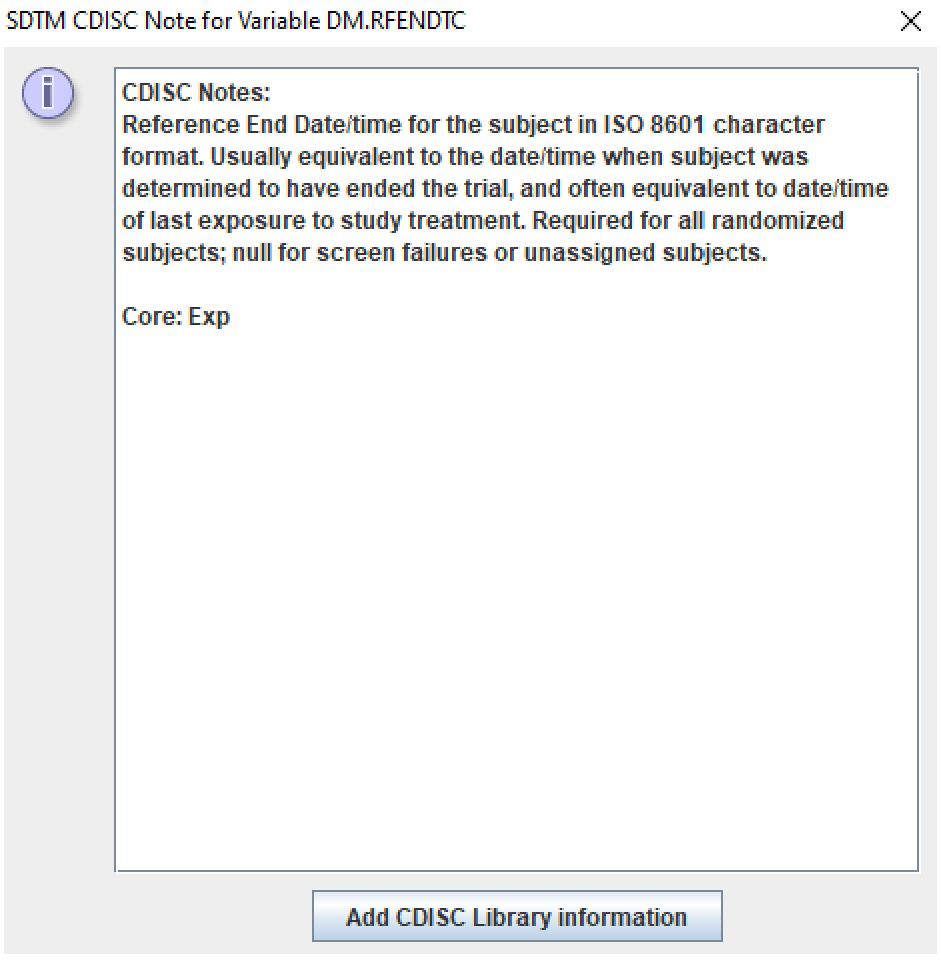
Testing on our single subject leads to:



```
24 </xsl:stylesheet>

Subject = 001
DM.RFSTDTC = 2010-02-27T10:27:33
```

The next SDTM variable that needs to be mapped is RFENDTC (Reference end date/time). Using CTRL-H tells us:


SDTM CDISC Note for Variable DM.RFENDTC

CDISC Notes:
Reference End Date/time for the subject in ISO 8601 character format. Usually equivalent to the date/time when subject was determined to have ended the trial, and often equivalent to date/time of last exposure to study treatment. Required for all randomized subjects; null for screen failures or unassigned subjects.

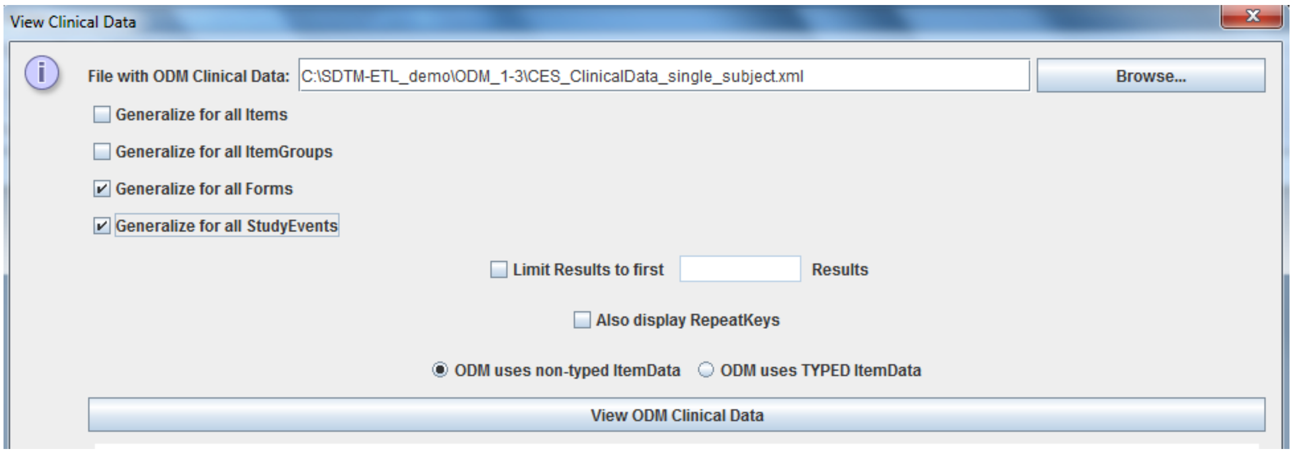
Core: Exp

The use of the "Add CDISC Library information" button will be explained in the tutorial "Working with the CDISC Library".

But now the question arises: what was the date the subject ended the trial? Was it the "Week 2 Visit", or was it the "Patient Diary Event", or maybe even the "Adverse Event" visit?
This time the menu "Navigate – Find hot SDTM Candidate“ does not give any results, so we need to find out ourselves...

We can easily find out what the last visit date is, as it was always collected (i.e. in each visit) using the same item ("Visit date“, with OID "I_VISIT“). One can easily see this by selecting the item, and then use "Navigate – Next Instance“ (or use CTRL-Page-down). One will then see that it was collected for each form for each visit.
But what was the last one?

Here again, the menu "View – ODM Clinical Data" is of great help. So select an item "Visit Date" and then use the menu "View – ODM Clinical Data":

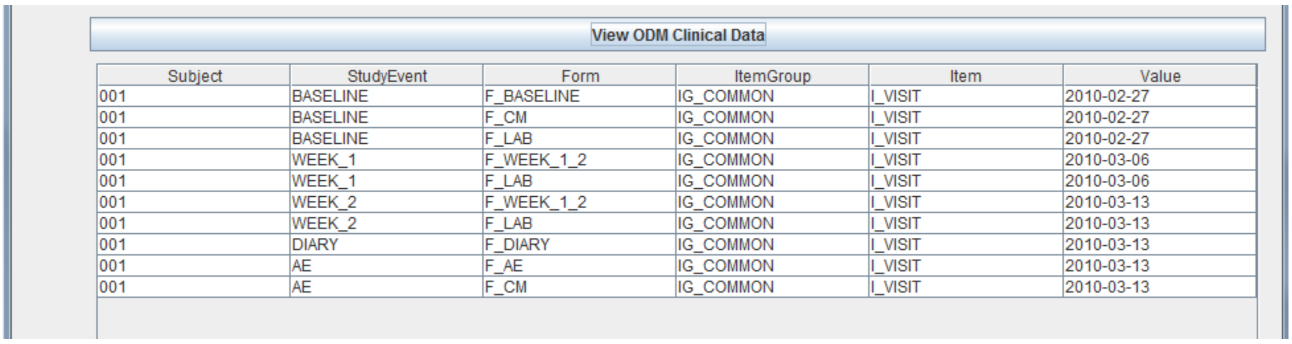

View Clinical Data

File with ODM Clinical Data: C:\SDTM-ETL_demo\ODM_1-3\CES_ClinicalData_single_subject.xml

- [ ] Generalize for all Items
- [ ] Generalize for all ItemGroups
- [x] Generalize for all Forms
- [x] Generalize for all StudyEvents
- [ ] Limit Results to first ___ Results
- [ ] Also display RepeatKeys

● ODM uses non-typed ItemData ○ ODM uses TYPED ItemData

This time, check the checkboxes "Generalize for all Forms“ and "Generalize for all StudyEvents“. This means that we want to see each data point "Visit Date“ independent from within which form and within which visit. Clicking the "View ODM Clinical Data“ button leads to:



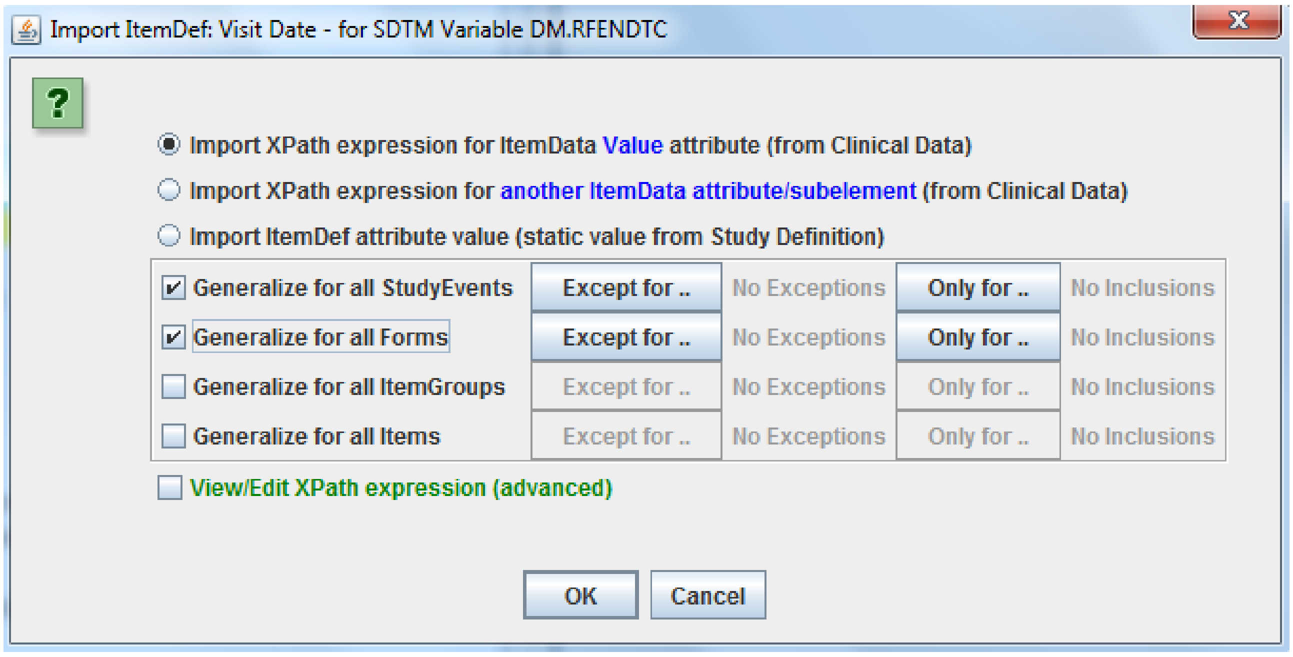
Subject	StudyEvent	Form	ItemGroup	Item	Value
001	BASELINE	F_BASELINE	IG_COMMON	I_VISIT	2010-02-27
001	BASELINE	F_CM	IG_COMMON	I_VISIT	2010-02-27
001	BASELINE	F_LAB	IG_COMMON	I_VISIT	2010-02-27
001	WEEK_1	F_WEEK_1_2	IG_COMMON	I_VISIT	2010-03-06
001	WEEK_1	F_LAB	IG_COMMON	I_VISIT	2010-03-06
001	WEEK_2	F_WEEK_1_2	IG_COMMON	I_VISIT	2010-03-13
001	WEEK_2	F_LAB	IG_COMMON	I_VISIT	2010-03-13
001	DIARY	F_DIARY	IG_COMMON	I_VISIT	2010-03-13
001	AE	F_AE	IG_COMMON	I_VISIT	2010-03-13
001	AE	F_CM	IG_COMMON	I_VISIT	2010-03-13

showing all the visit dates ever registered.

It looks as (at least for this subject) the last visit date was on March 13th 2010, and the visit was either "WEEK_2“ or "DIARY“ or "AE“, which all happened on the same day. However, we cannot know whether this will apply to all subjects.

The ODM standard states that clinical data for subjects MUST come in chronological order, with earliest data first, and latest data last in the file. So we can simply look for the last occurrence of "Visit Date" for each subject in the file with clinical data.

After having gone back to the main window, drag-and-drop one of the items "Visit date" from the tree with the study design (it doesn't matter which one), and drop it in the cell "RFENDTC“. The following dialog is displayed:


Import ItemDef: Visit Date - for SDTM Variable DM.RFENDTC

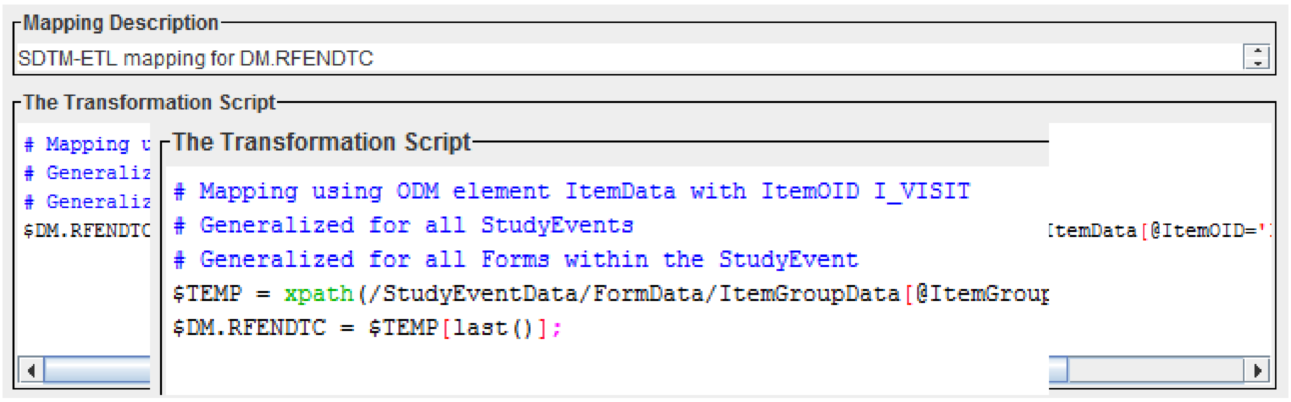
- [●] Import XPath expression for ItemData Value attribute (from Clinical Data)
- [ ] Import XPath expression for another ItemData attribute/subelement (from Clinical Data)
- [ ] Import ItemDef attribute value (static value from Study Definition)

[✓] Generalize for all StudyEvents	Except for ..	No Exceptions	Only for ..	No Inclusions
[✓] Generalize for all Forms	Except for ..	No Exceptions	Only for ..	No Inclusions
[ ] Generalize for all ItemGroups	Except for ..	No Exceptions	Only for ..	No Inclusions
[ ] Generalize for all Items	Except for ..	No Exceptions	Only for ..	No Inclusions

[ ] View/Edit XPath expression (advanced)

Check the checkboxes "Generalize for all StudyEvents“ and "Generalize for all Forms“, stating that we want to have the item independent of the form or visit[17].

This leads to the mapping:

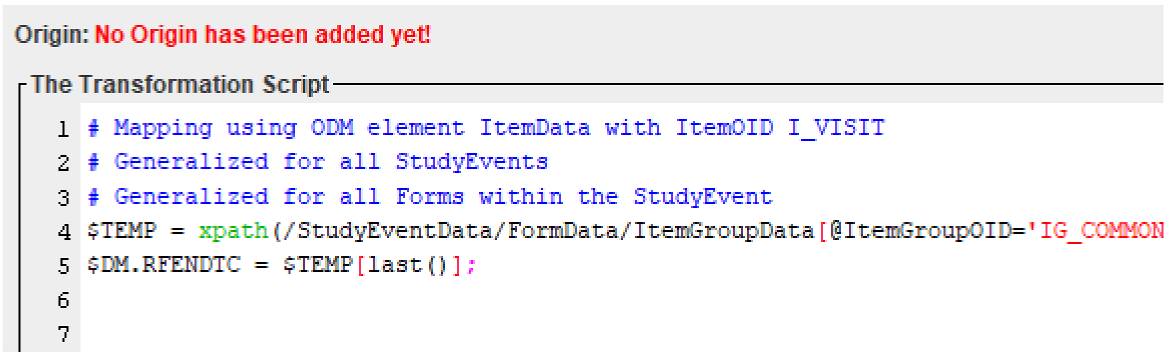

Mapping Description

SDTM-ETL mapping for DM.RFENDTC

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_VISIT
# Generalized for all StudyEvents
# Generalized for all Forms within the StudyEvent
$TEMP = xpath(/StudyEventData/FormData/ItemGroupData[@ItemGroup
$DM.RFENDTC = $TEMP[last()];
```

But we only want the last one, so we do a little rewrite into:


Origin: No Origin has been added yet!

The Transformation Script

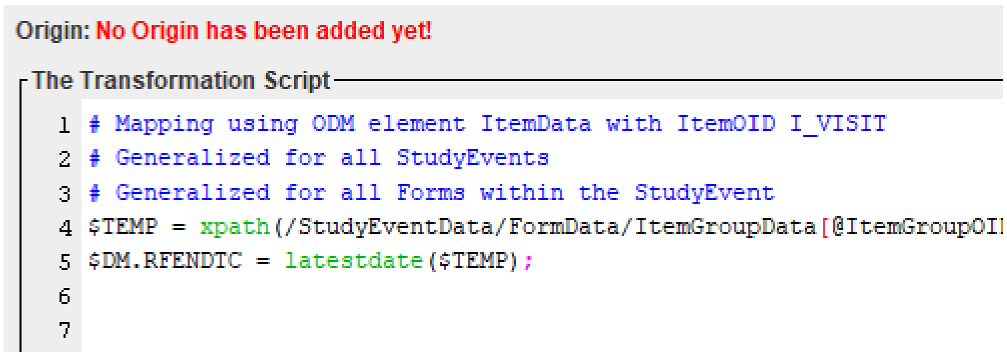
```
# Mapping using ODM element ItemData with ItemOID I_VISIT
# Generalized for all StudyEvents
# Generalized for all Forms within the StudyEvent
$TEMP = xpath(/StudyEventData/FormData/ItemGroupData[@ItemGroupOID='IG_COMMON
$DM.RFENDTC = $TEMP[last()];


```

i.e. By generating a temporary variable, and then adding a condition [last()] to the expression[18].

[17]Later we will see how to work with the buttons "Except for ...“ and "Only for ...“
[18]"taking the first one available" is written as "[1]“

Another possibility is to use one of the many "datetime" functions, in this case "latestdate()":



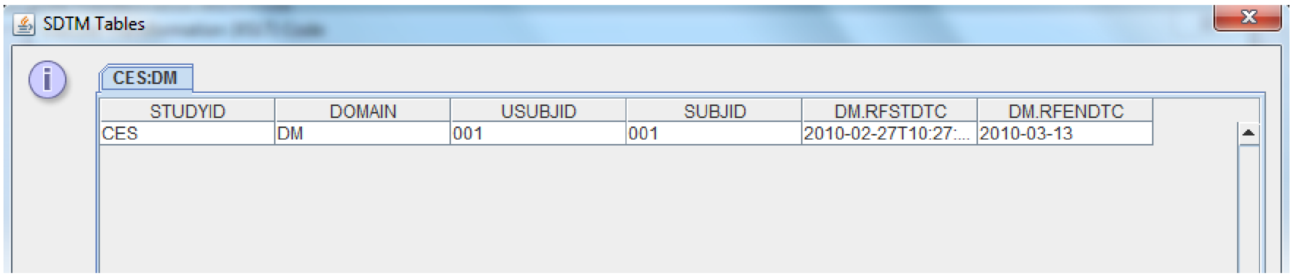
Origin: No Origin has been added yet!

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_VISIT
# Generalized for all StudyEvents
# Generalized for all Forms within the StudyEvent
$TEMP = xpath(/StudyEventData/FormData/ItemGroupData[@ItemGroupOI
$DM.RFENDTC = latestdate($TEMP);
```

Which may be a more safe method.
The use of functions (and how they can be extended by the user) will be explained further on and in other tutorials.

Then executing the mapping script leads to:

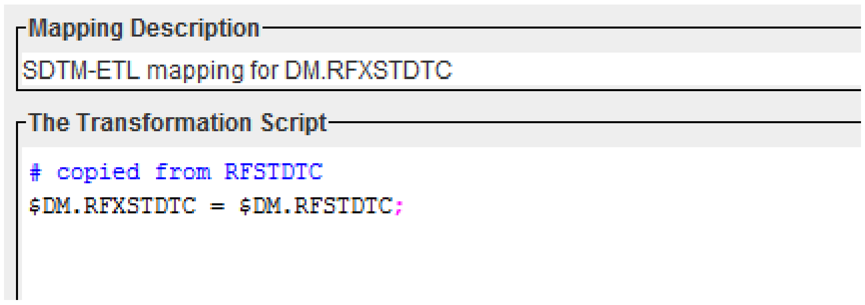


SDTM Tables

CES:DM

STUDYID	DOMAIN	USUBJID	SUBJID	DM.RFSTDTC	DM.RFENDTC
CES	DM	001	001	2010-02-27T10:27:...	2010-03-13

In a good number of cases, earlier defined mappings (i.e. for variables more to the left in the same domain) can easily be reused. For example, for the next variable "RFXSTDTC“ in the DM domain, we can write:



Mapping Description

SDTM-ETL mapping for DM.RFXSTDTC

The Transformation Script

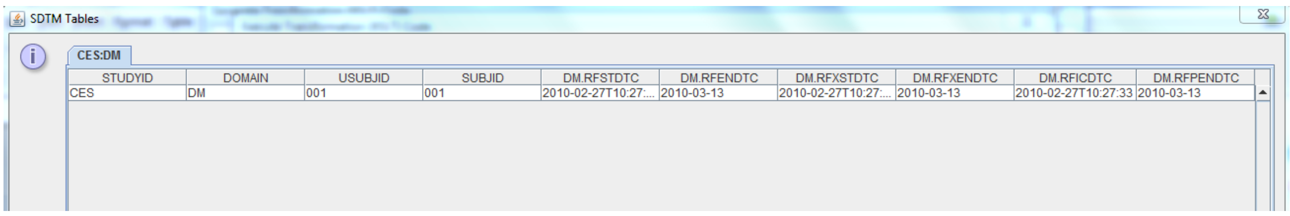
```
# copied from RFSTDTC
$DM.RFXSTDTC = $DM.RFSTDTC;
```

Similarly, we can set for the next DM variables:

$DM.RFXENDTC = $DM.RFENDTC;
$DM.RFICDTC = $DM.RFSTDTC;
$DM.RFPENDTC = $DM.RFENDTC;

but of course only in the case dates were really identical to the first and last visit date correspondingly.

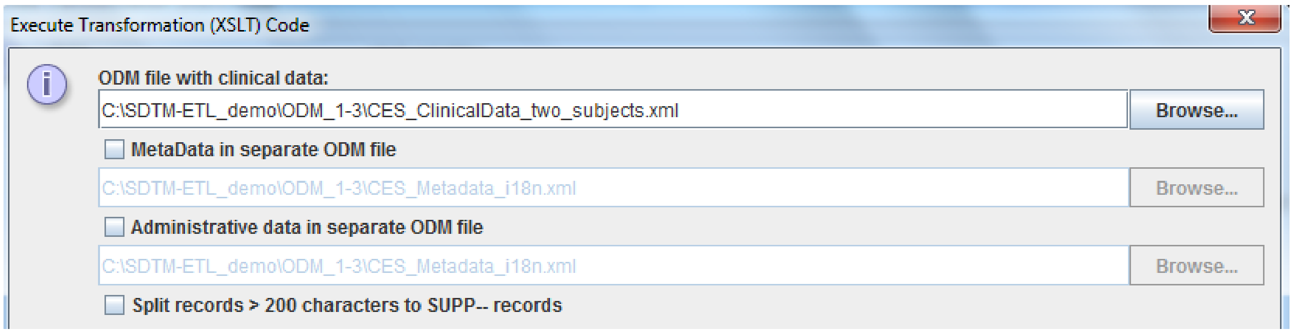
This leads to the following result:


SDTM Tables

CES:DM

STUDYID	DOMAIN	USUBJID	SUBJID	DM.RFSTDTC	DM.RFENDTC	DM.RFXSTDTC	DM.RFXENDTC	DM.RFICDTC	DM.RFPENDTC
CES	DM	001	001	2010-02-27T10:27:...	2010-03-13	2010-02-27T10:27:...	2010-03-13	2010-02-27T10:27:33	2010-03-13

Meanwhile we have received the data of a second subject, so we can test our mapping again:


Execute Transformation (XSLT) Code

ODM file with clinical data:

C:\SDTM-ETL_demo\ODM_1-3\CES_ClinicalData_two_subjects.xml

Browse...

☐ MetaData in separate ODM file

C:\SDTM-ETL_demo\ODM_1-3\CES_Metadata_i18n.xml

Browse...

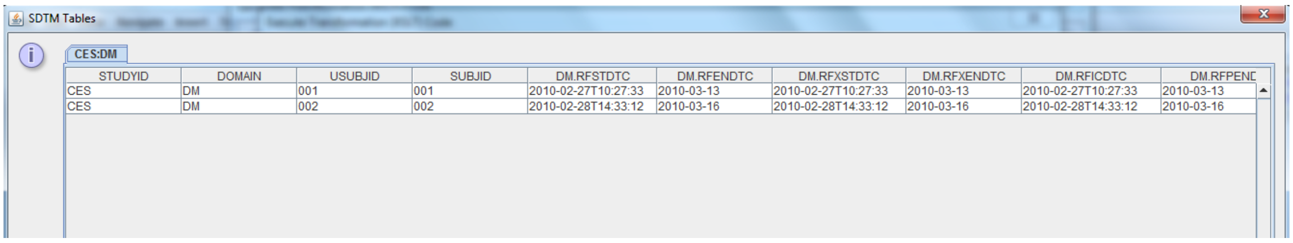
☐ Administrative data in separate ODM file

C:\SDTM-ETL_demo\ODM_1-3\CES_Metadata_i18n.xml

Browse...

☐ Split records > 200 characters to SUPP-- records

Resulting in:

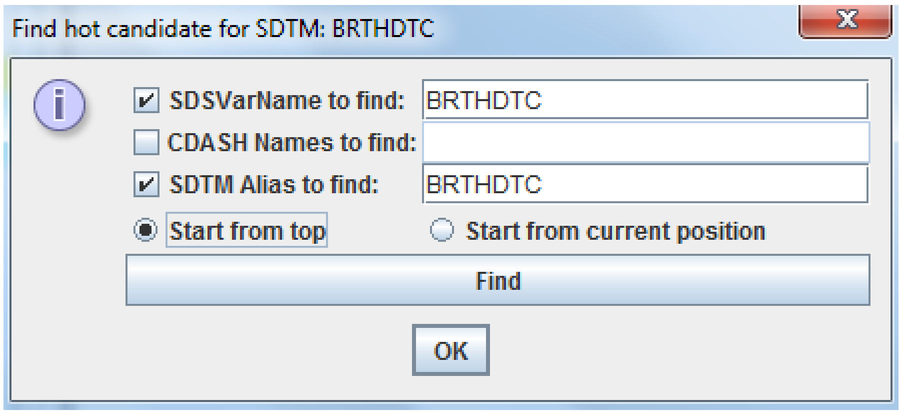

SDTM Tables

CES:DM

STUDYID	DOMAIN	USUBJID	SUBJID	DM.RFSTDTC	DM.RFENDTC	DM.RFXSTDTC	DM.RFXENDTC	DM.RFICDTC	DM.RFPENDTC
CES	DM	001	001	2010-02-27T10:27:33	2010-03-13	2010-02-27T10:27:33	2010-03-13	2010-02-27T10:27:33	2010-03-13
CES	DM	002	002	2010-02-28T14:33:12	2010-03-16	2010-02-28T14:33:12	2010-03-16	2010-02-28T14:33:12	2010-03-16

In the tutorial "Creating and working with Subject Global Variables", we will learn how to make RFSTDTC and RFENDTC as "global" variable that can used (in read-mode) over and over again, as it is used in many domains for the calculation of the "visit day" (-DY) variable and for variables that are about "relative to start ..." or " relative to end of study participation".

Let us now concentrate on two other important SDTM variables in the SDTM domain: BRTHDTC and AGE. Again we first try to find a "hot candidate" in our ODM tree.


Find hot candidate for SDTM: BRTHDTC

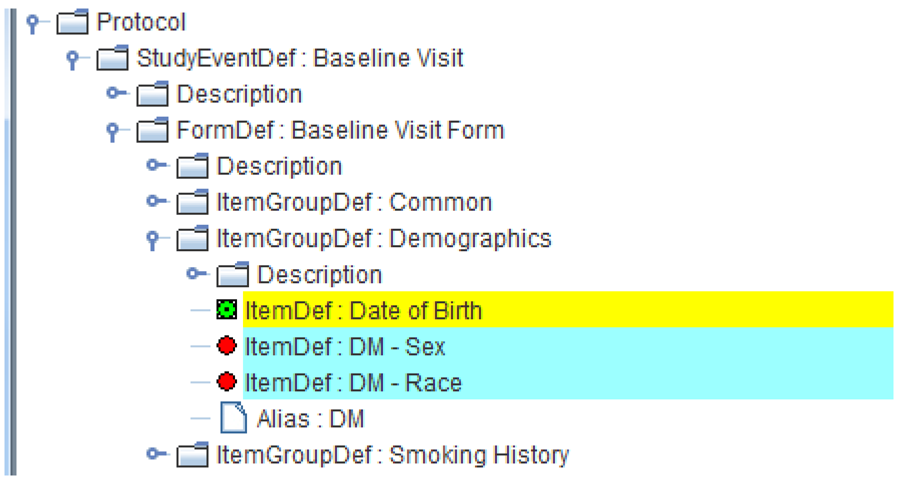
☑ SDSVarName to find: BRTHDTC

☐ CDASH Names to find:

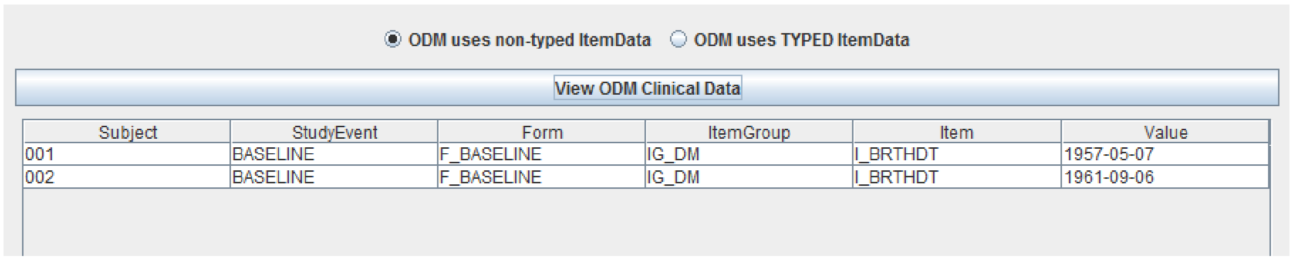
☑ SDTM Alias to find: BRTHDTC

◉ Start from top ○ Start from current position

With the result:

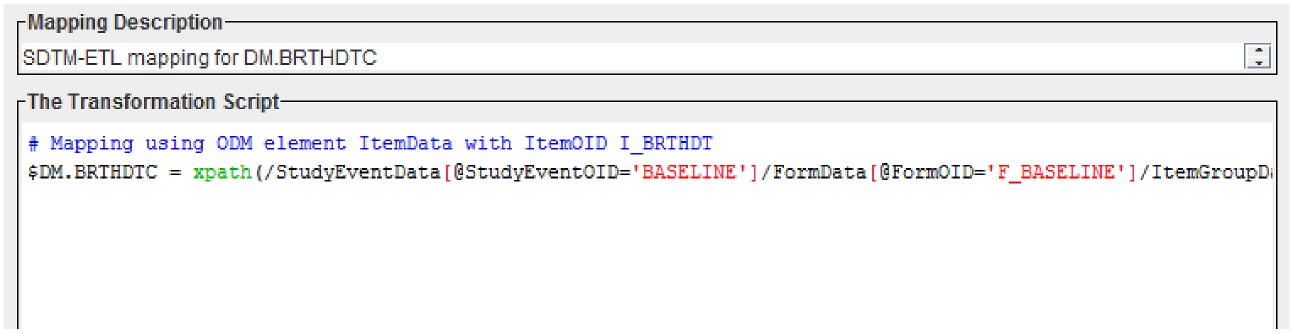


A view in the clinical data for this item (using "View – ODM Clinical Data") results in:



Subject	StudyEvent	Form	ItemGroup	Item	Value
001	BASELINE	F_BASELINE	IG_DM	I_BRTHDT	1957-05-07
002	BASELINE	F_BASELINE	IG_DM	I_BRTHDT	1961-09-06

Dragging and dropping the item from the tree into the cell "DM.BRTHDTC" results in the mapping:

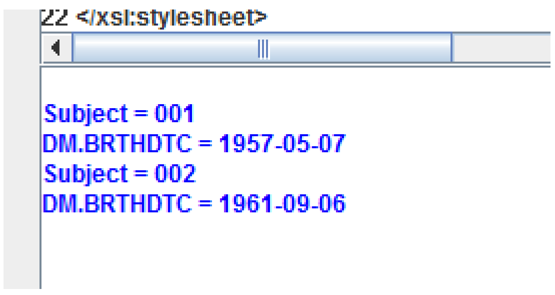

Mapping Description

SDTM-ETL mapping for DM.BRTHDTC

The Transformation Script

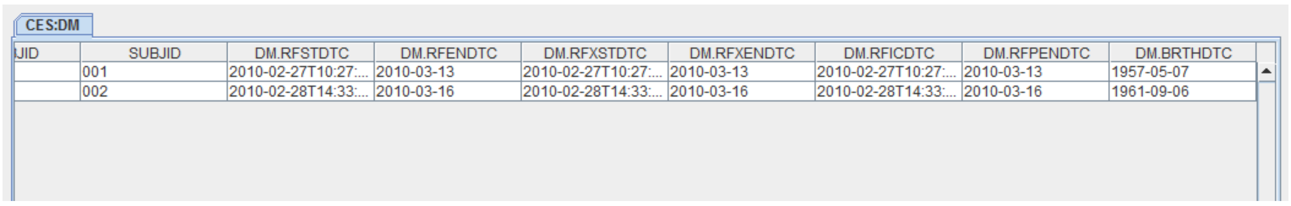
```
# Mapping using ODM element ItemData with ItemOID I_BRTHDT
$DM.BRTHDTC = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupD
```

and doing a "local" quick test of this mapping results in:


```
22 </xsl:stylesheet>
```

```
Subject = 001
DM.BRTHDTC = 1957-05-07
Subject = 002
DM.BRTHDTC = 1961-09-06
```

Or executing the mapping for all SDTM variables in the DM that we mapped sofar in:

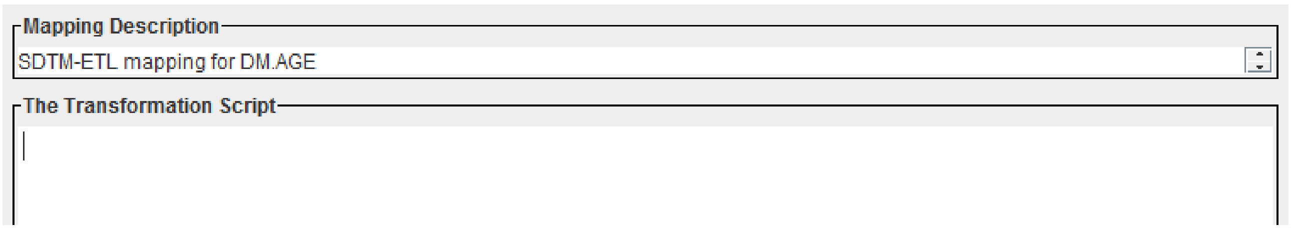


CES:DM

JID	SUBJID	DM.RFSTDTC	DM.RFENDTC	DM.RFXSTDTC	DM.RFXENDTC	DM.RFICDTC	DM.RFPENDTC	DM.BRTHDTC
	001	2010-02-27T10:27:...	2010-03-13	2010-02-27T10:27:...	2010-03-13	2010-02-27T10:27:...	2010-03-13	1957-05-07
	002	2010-02-28T14:33:...	2010-03-16	2010-02-28T14:33:...	2010-03-16	2010-02-28T14:33:...	2010-03-16	1961-09-06

The next variable that needs to be mapped is "AGE". However, it looks as the age of the subject was not collected directly, so we need to calculate it from the birth date and the reference start date.

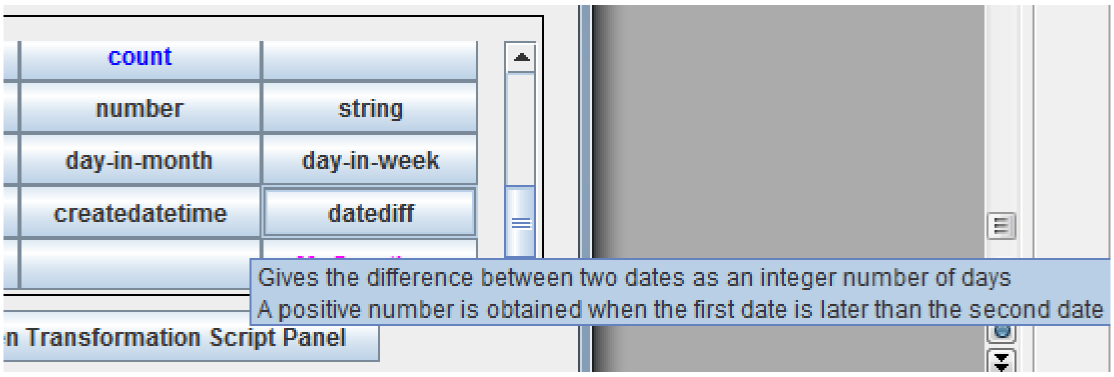
Just double-click the cell "AGE" to start the mapping process:



Mapping Description
SDTM-ETL mapping for DM.AGE

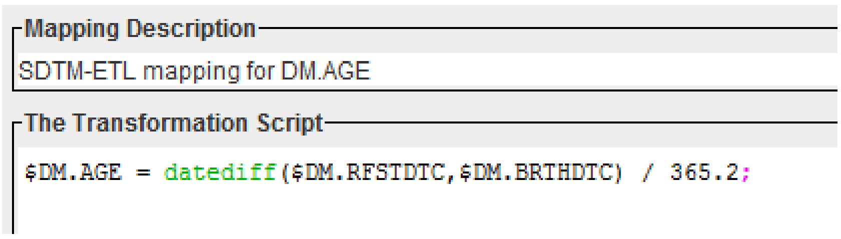
The Transformation Script

As the birth date ($DM.BRTHDTC) and the reference start date ($DM.RFSTDTC) were already mapped before, we can reuse them, but in only in "read mode".
Now look into the lower part of the mapping screen, where the "Scripting Language Functions" are displayed. If we scroll down, we find:



Gives the difference between two dates as an integer number of days
A positive number is obtained when the first date is later than the second date

So we can use the function "datediff()" to calculate the difference (in number of days) between reference start date and the birth date. If the result is then divided by 365.2 (the average number of days in a year), then the age in years is obtained. So the mapping script becomes:



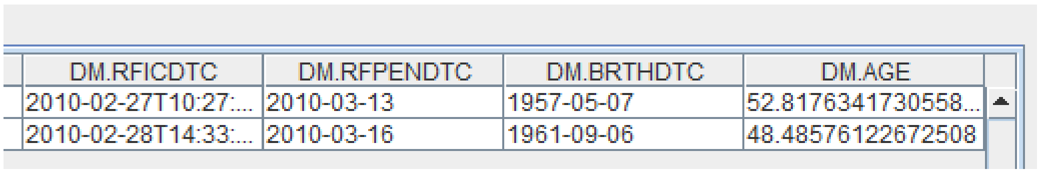
Mapping Description
SDTM-ETL mapping for DM.AGE

The Transformation Script

```
$DM.AGE = datediff($DM.RFSTDTC,$DM.BRTHDTC) / 365.2;
```

and executing the mapping for the whole domain[19] results in:

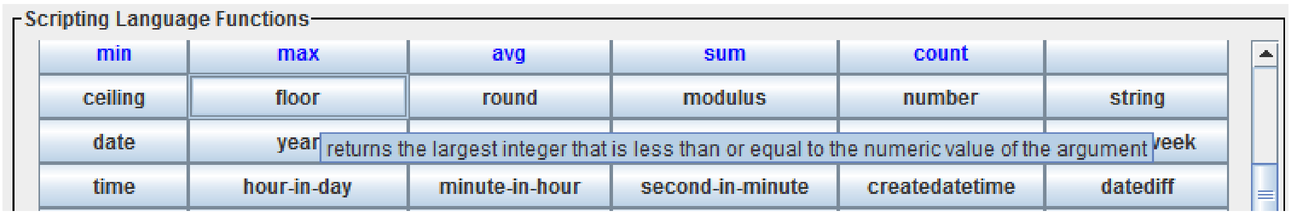
[19]We can not do a "local" testing, as the variables "DM.RFSTDTC" and "DM.BRTHDTC" are out of scope, as they have been defined in previous mappings.



DM.RFICDTC	DM.RFPENDTC	DM.BRTHDTC	DM.AGE
2010-02-27T10:27:...	2010-03-13	1957-05-07	52.8176341730558...
2010-02-28T14:33:...	2010-03-16	1961-09-06	48.48576122672508

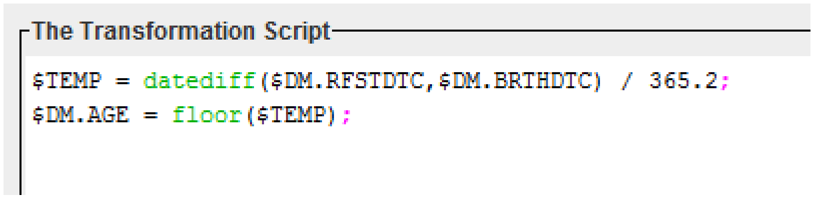
which is … not entirely what we want, as we would like to obtain an integer number.

If we look again to the available functions, we find:



returns the largest integer that is less than or equal to the numeric value of the argument

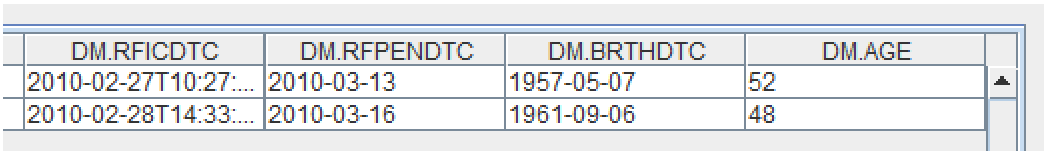
with the "floor()“ function delivering what we want. So the mapping is adapted to:



The Transformation Script

```
$TEMP = datediff($DM.RFSTDTC,$DM.BRTHDTC) / 365.2;
$DM.AGE = floor($TEMP);
```

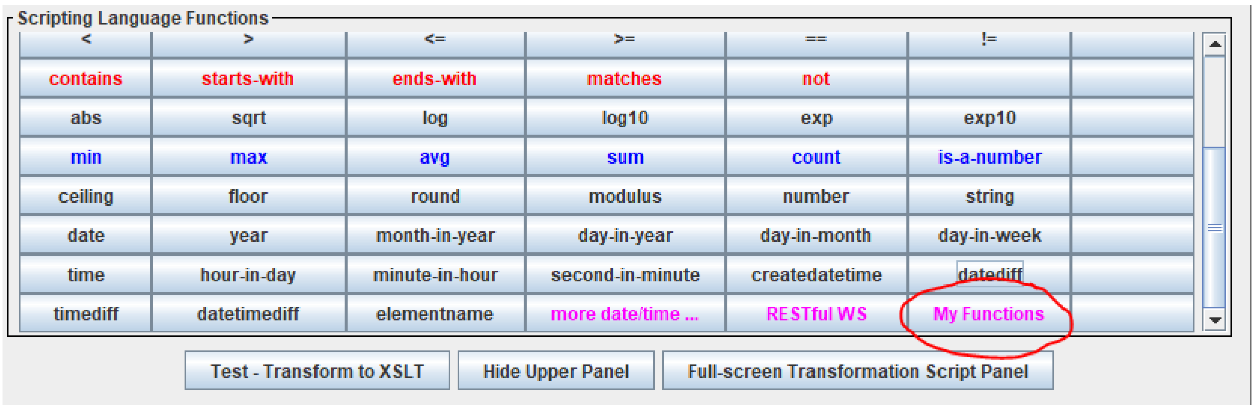
Resulting in:



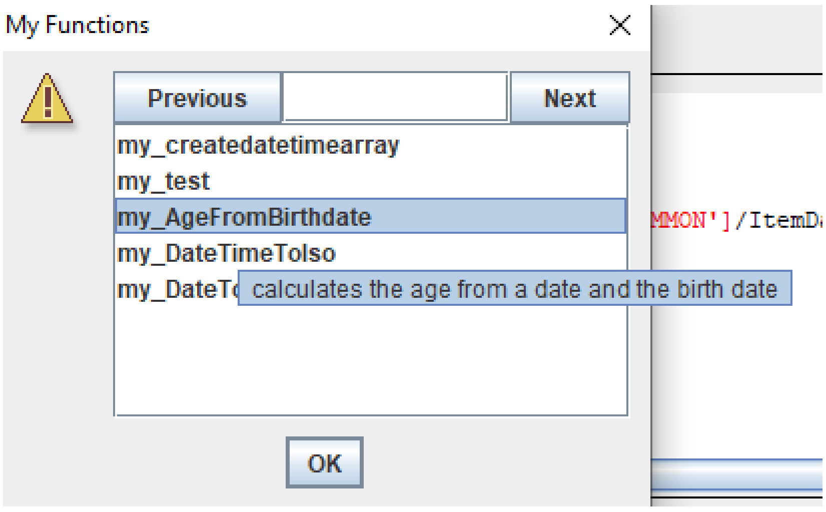
DM.RFICDTC	DM.RFPENDTC	DM.BRTHDTC	DM.AGE
2010-02-27T10:27:...	2010-03-13	1957-05-07	52
2010-02-28T14:33:...	2010-03-16	1961-09-06	48

which is exactly what we want.

This is where we come to "user functions". How to develop these is explained in another tutorial. These functions reside in the file "functions.xsl" in the folder "stylesheets" and need to start with "my_". The distribution already comes with an example user function "my_AgeFromBirthdate". It can be found under the button "My Functions" in the mapping editor":

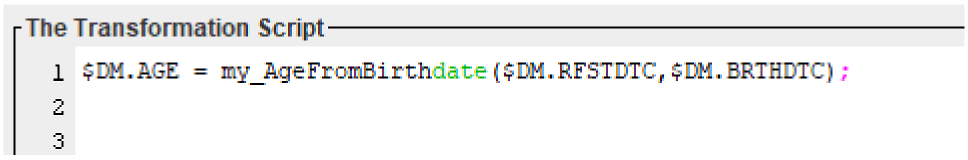

Scripting Language Functions
< > <= >= == !=
contains starts-with ends-with matches not
abs sqrt log log10 exp exp10
min max avg sum count is-a-number
ceiling floor round modulus number string
date year month-in-year day-in-year day-in-month day-in-week
time hour-in-day minute-in-hour second-in-minute createdatetime datediff
timediff datetimediff elementname more date/time ... RESTful WS My Functions
Test - Transform to XSLT
Hide Upper Panel
Full-screen Transformation Script Panel

when clicked, we find:



calculates the age from a date and the birth date

And we can use it in the mapping script, e.g.:



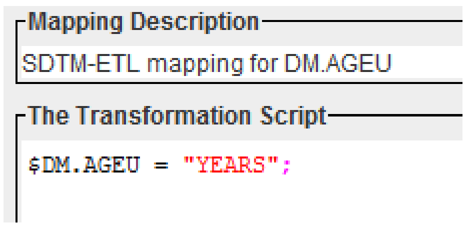
```
$DM.AGE = my_AgeFromBirthdate($DM.RFSTDTC,$DM.BRTHDTC);
```

Development of user functions is relatively easy, and is very frequently done by our customers[20].

This kind of calculations should be the exception in SDTM, as SDTM is about collected data and not about derived data. Unfortunately, derivations have sneaked in in SDTM in the last years, as the tools of the regulatory authorities are not able to calculate them "on the fly“ from the already available data. A typical example are all the --DY variables.

The next SDTM variable is "AGEU“. In our case it just is the string "YEARS". So the mapping is:

20 We do of course deliver services for development of user functions.

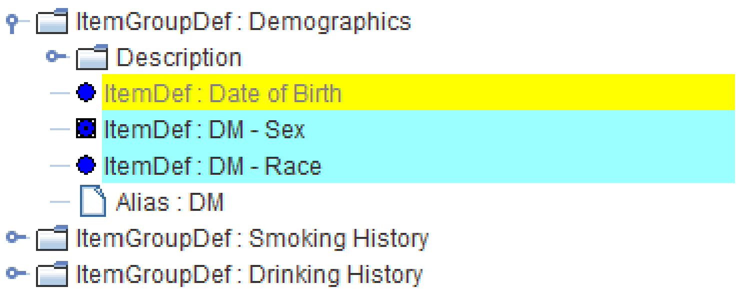

Mapping Description

SDTM-ETL mapping for DM.AGEU

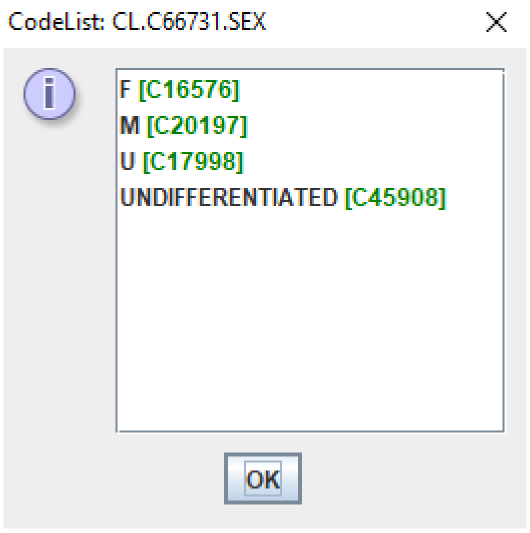
The Transformation Script

```
$DM.AGEU = "YEARS";
```

For "SEX“, we once again first look for a "hot candidate“ and find:


ItemGroupDef : Demographics
Description
ItemDef : Date of Birth
ItemDef : DM - Sex
ItemDef : DM - Race
Alias : DM
ItemGroupDef : Smoking History
ItemGroupDef : Drinking History

It is seen that the "traffic light“ is blue, meaning that the variable is under controlled terminology. The information about the SDTM controlled terminology can be obtained using the menu "View – SDTM associated codelist“ which delivers:

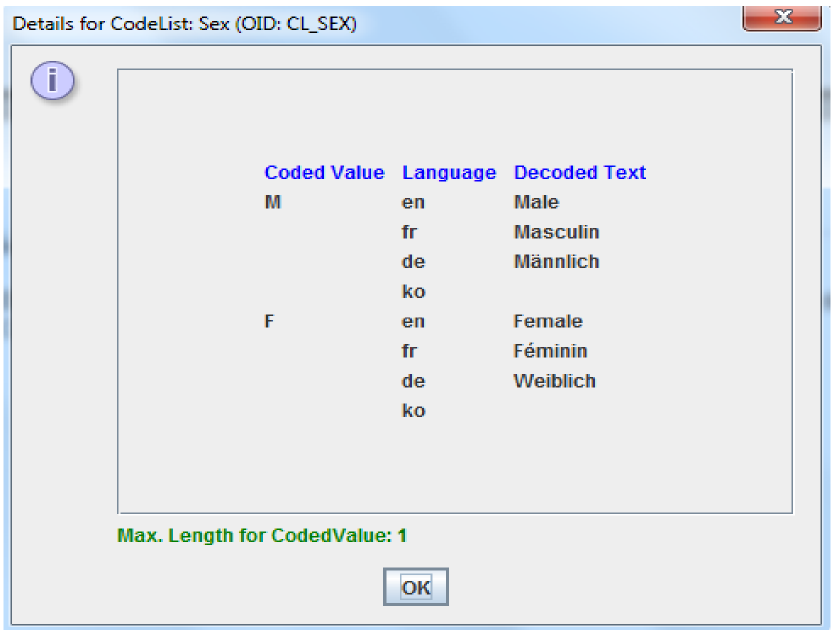

CodeList: CL.C66731.SEX

F [C16576]
M [C20197]
U [C17998]
UNDIFFERENTIATED [C45908]

standing for "female“, "male“, "unknown“ and "undifferentiated“ (intersex)[21].
The codes in brackets in green provide the CDISC/NCI codes, which become ever more important in how CDISC maintains its codelists.

Also on the ODM side, there is an associated codelist. Selecting the item "Sex“ and using the menu "View – Item CodeList details“ provides a dialog:

[21]See the published CDISC controlled terminology lists published by NCI



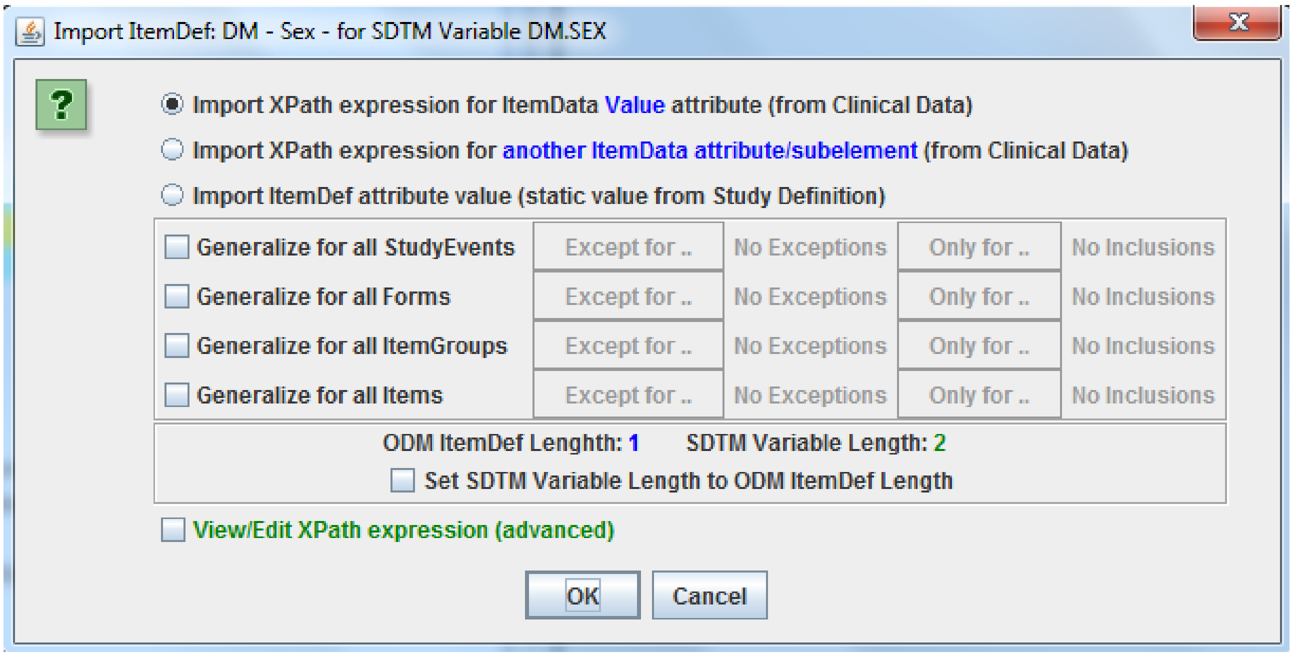
Details for CodeList: Sex (OID: CL_SEX)

Coded Value	Language	Decoded Text
M	en	Male
	fr	Masculin
	de	Männlich
	ko	
F	en	Female
	fr	Féminin
	de	Weiblich
	ko	

Max. Length for CodedValue: 1

stating that in the ODM, only the values "M“ and "F“ are foreseen.

Drag-and-drop from the item "Sex“ in the study design tree to the SDTM cell "DM.SEX“ displays the wizard:



Import ItemDef: DM - Sex - for SDTM Variable DM.SEX

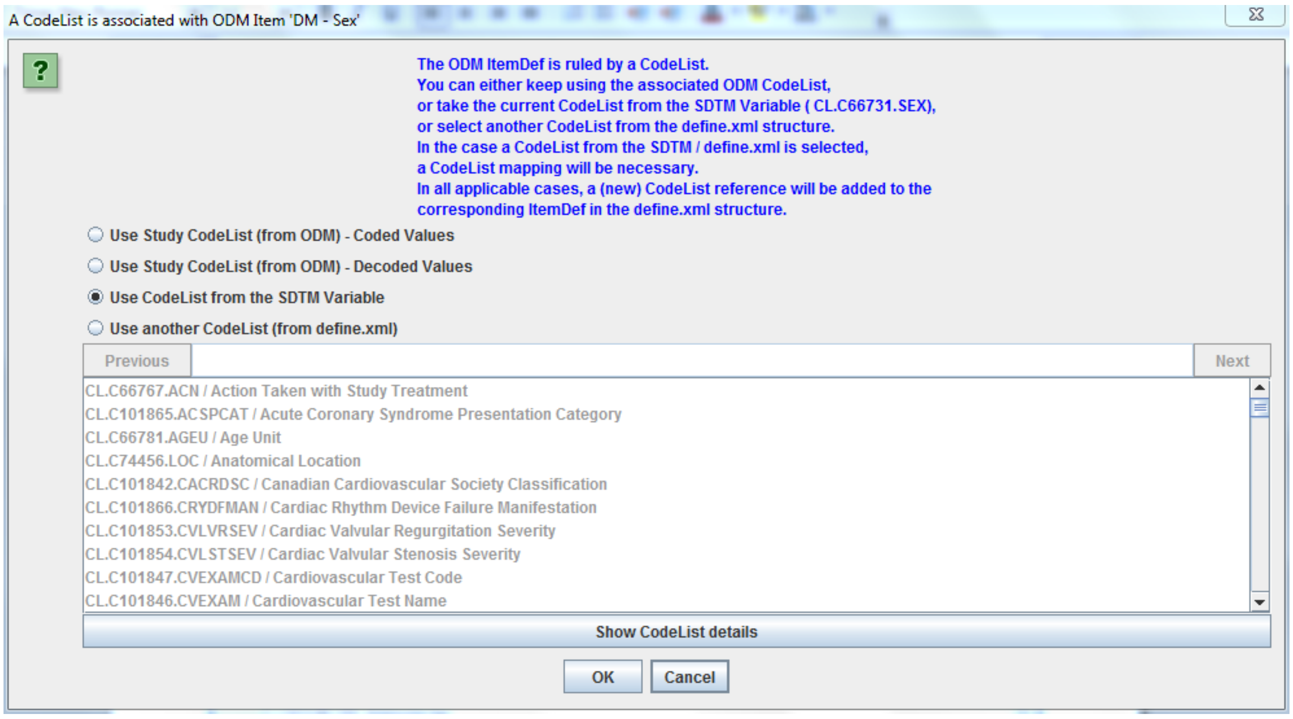
[✓ Import XPath expression for ItemData Value attribute (from Clinical Data)]
Import XPath expression for another ItemData attribute/subelement (from Clinical Data)
Import ItemDef attribute value (static value from Study Definition)

☐ Generalize for all StudyEvents	Except for ..	No Exceptions	Only for ..	No Inclusions
☐ Generalize for all Forms	Except for ..	No Exceptions	Only for ..	No Inclusions
☐ Generalize for all ItemGroups	Except for ..	No Exceptions	Only for ..	No Inclusions
☐ Generalize for all Items	Except for ..	No Exceptions	Only for ..	No Inclusions

ODM ItemDef Lenghth: 1 SDTM Variable Length: 2
☐ Set SDTM Variable Length to ODM ItemDef Length

☐ View/Edit XPath expression (advanced)

and then clicking "OK“ leads to the following dialog:

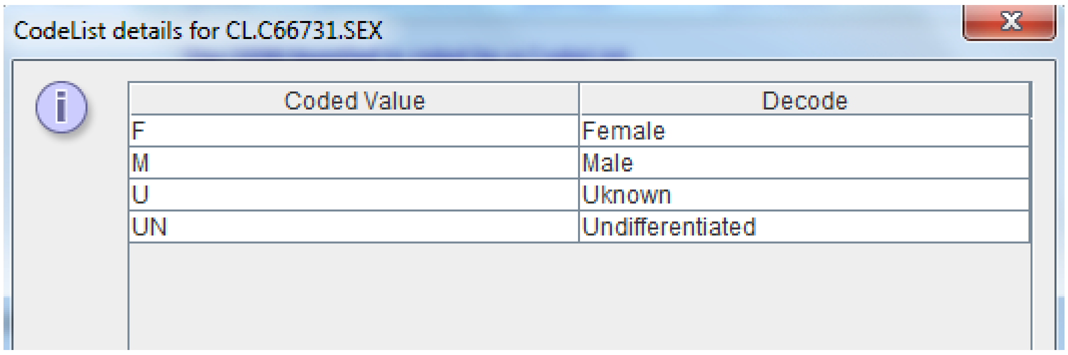

A CodeList is associated with ODM Item 'DM - Sex'

The ODM ItemDef is ruled by a CodeList.
You can either keep using the associated ODM CodeList,
or take the current CodeList from the SDTM Variable (CL.C66731.SEX),
or select another CodeList from the define.xml structure.
In the case a CodeList from the SDTM / define.xml is selected,
a CodeList mapping will be necessary.
In all applicable cases, a (new) CodeList reference will be added to the
corresponding ItemDef in the define.xml structure.

- ○ Use Study CodeList (from ODM) - Coded Values
- ○ Use Study CodeList (from ODM) - Decoded Values
- ● Use CodeList from the SDTM Variable
- ○ Use another CodeList (from define.xml)

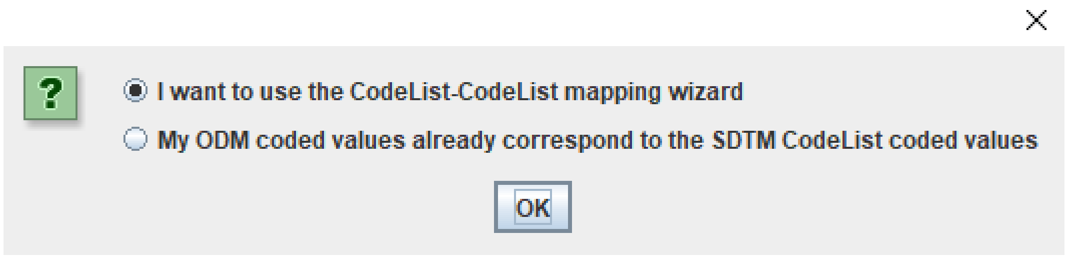
CL.C66767.ACN / Action Taken with Study Treatment
CL.C101865.ACSPCAT / Acute Coronary Syndrome Presentation Category
CL.C66781.AGEU / Age Unit
CL.C74456.LOC / Anatomical Location
CL.C101842.CACRDSC / Canadian Cardiovascular Society Classification
CL.C101866.CRYDFMAN / Cardiac Rhythm Device Failure Manifestation
CL.C101853.CVLVRSEV / Cardiac Valvular Regurgitation Severity
CL.C101854.CVLSTSEV / Cardiac Valvular Stenosis Severity
CL.C101847.CVEXAMCD / Cardiovascular Test Code
CL.C101846.CVEXAM / Cardiovascular Test Name

asking whether we want to use the ODM codelist (coded or decoded values), the currently to DM.SEX asscociated codelist, or another list from the SDTM define.xml list. We want to use the SDTM codelist, so we select "Use codelist from the SDTM Variable“. We can quickly inspect that codelist using the "Show CodeList Details“ button:


CodeList details for CL.C66731.SEX

Coded Value	Decode
F	Female
M	Male
U	Uknown
UN	Undifferentiated

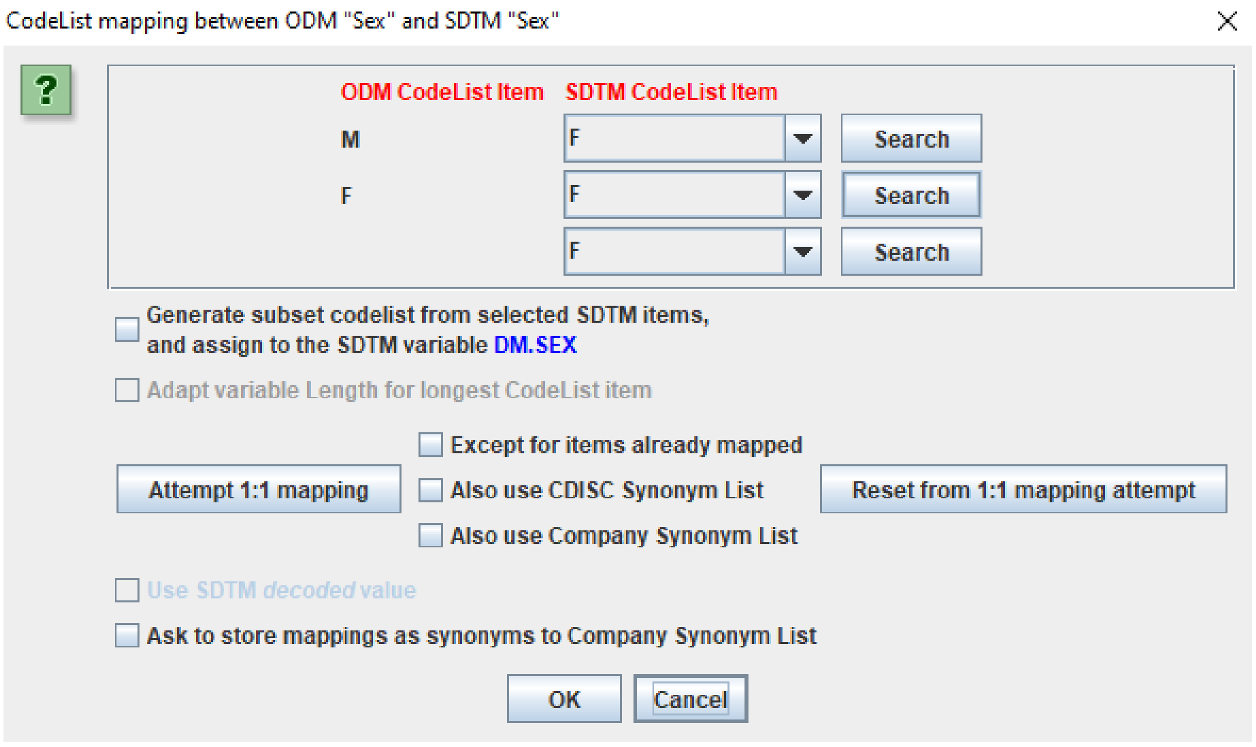
After clicking the "OK“ button in the "A CodeList is associated“ wizard, a choice dialog[22] is displayed:


- ● I want to use the CodeList-CodeList mapping wizard
- ○ My ODM coded values already correspond to the SDTM CodeList coded values

Mostly, you will want to use the "mapping wizard", unless your (coded) values in the ODM are already "F", "M", etc.. This will especially the case when CDASH forms have been used with CDISC controlled terminology already in the study design.

22 This is new as of SDTM-ETL version 4.1.

Not only just for the exercise, but as we also want to have "U" (Unknown) in the result, which is not in the ODM, we want to use the "mapping wizard", leading to:

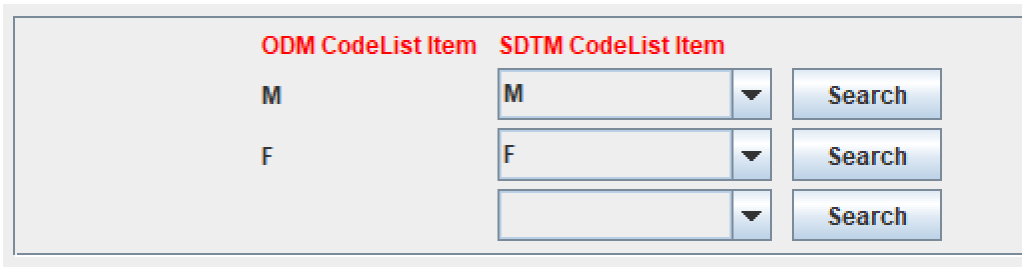


CodeList mapping between ODM "Sex" and SDTM "Sex"

ODM CodeList Item	SDTM CodeList Item
M	F
F	F
	F

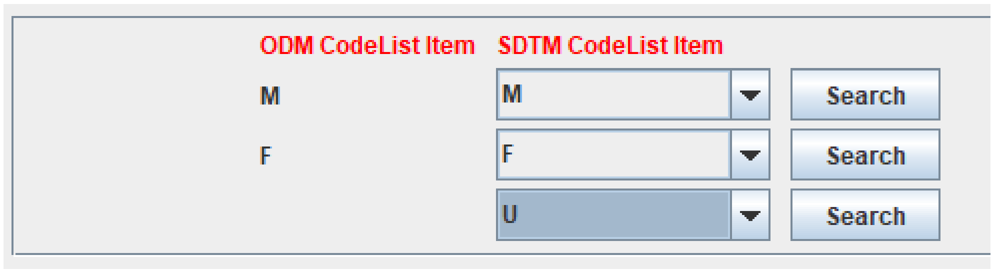
- [ ] Generate subset codelist from selected SDTM items, and assign to the SDTM variable DM.SEX
- [ ] Adapt variable Length for longest CodeList item
- [ ] Except for items already mapped
- [ ] Also use CDISC Synonym List
- [ ] Also use Company Synonym List
- [ ] Use SDTM *decoded* value
- [ ] Ask to store mappings as synonyms to Company Synonym List

In this case it is easy, and even the wizard will have an easy task finding out – so click the "Attempt 1:1 mapping" button, resulting in a proposal mapping:



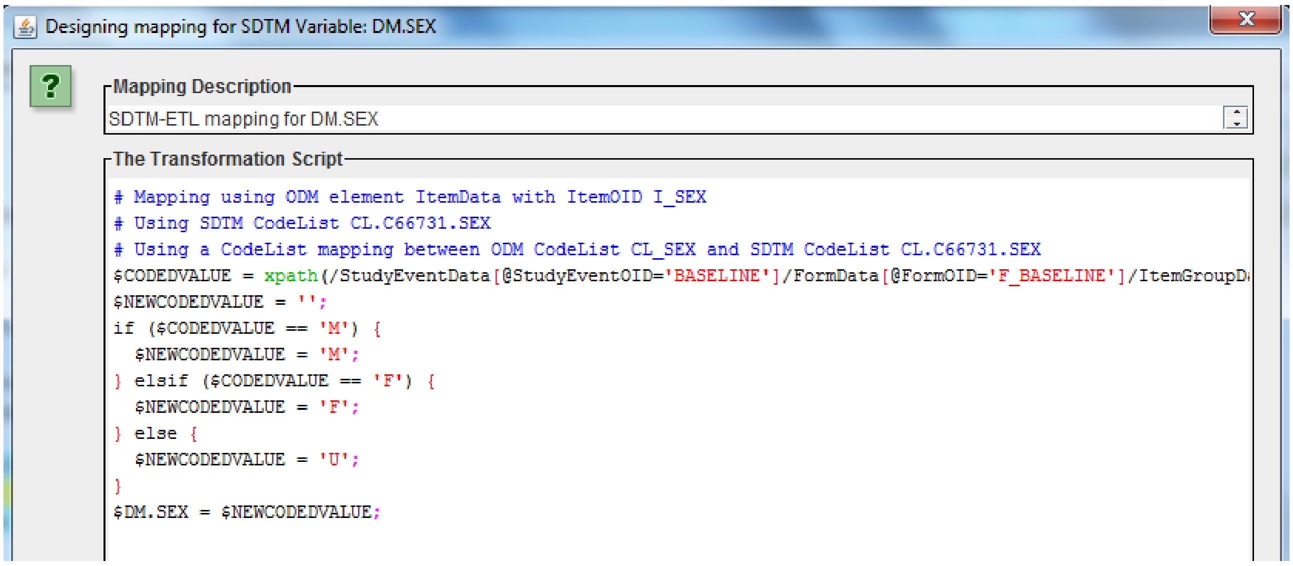
ODM CodeList Item	SDTM CodeList Item
M	M
F	F

For the empty value, we can add that we want to map this to "U" (for "Unknown"), so we add that by using the dropbox:



ODM CodeList Item	SDTM CodeList Item
M	M
F	F
	U

After clicking "OK“ the mapping script is completely automatically created:


Designing mapping for SDTM Variable: DM.SEX

Mapping Description

SDTM-ETL mapping for DM.SEX

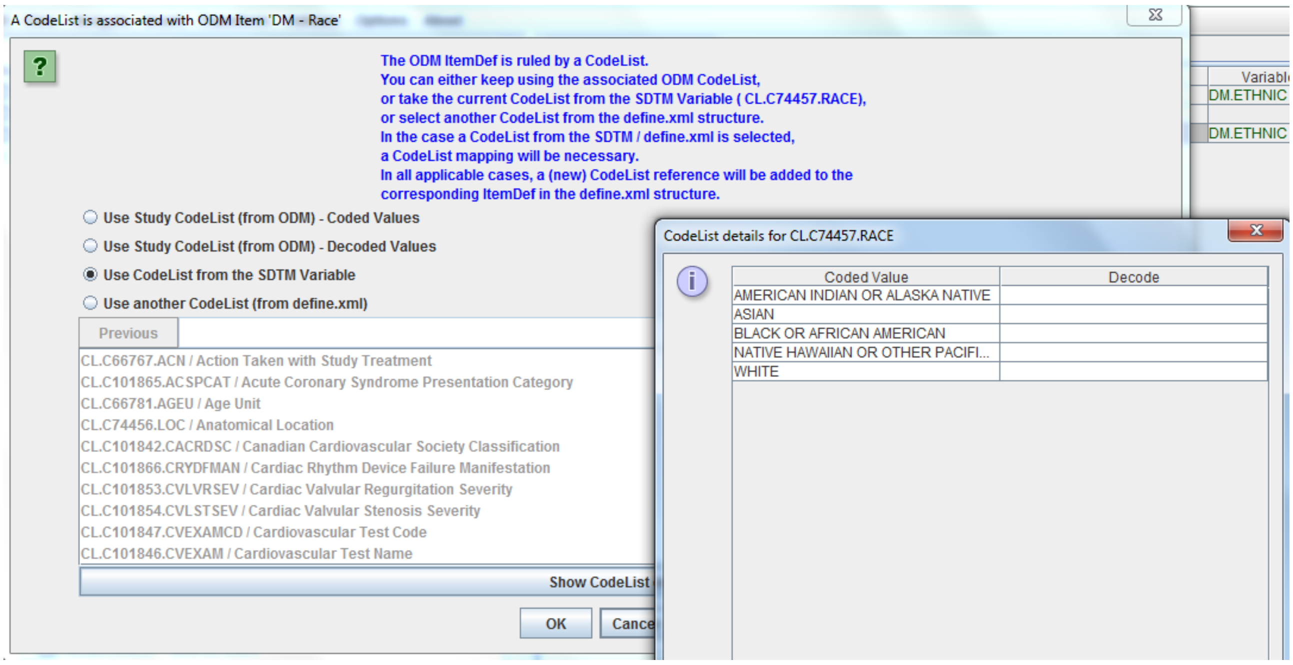
The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_SEX
# Using SDTM CodeList CL.C66731.SEX
# Using a CodeList mapping between ODM CodeList CL_SEX and SDTM CodeList CL.C66731.SEX
$CODEDVALUE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupD
$NEWCODEDVALUE = '';
if ($CODEDVALUE == 'M') {
  $NEWCODEDVALUE = 'M';
} elsif ($CODEDVALUE == 'F') {
  $NEWCODEDVALUE = 'F';
} else {
  $NEWCODEDVALUE = 'U';
}
$DM.SEX = $NEWCODEDVALUE;
```

the "if-elsif-else“ construct being generated automatically.

In many cases, wizards will create mapping scripts completely automatically, but the user can always further enhance or change the mapping script manually.

A similar mapping needs to be done for "RACE“. Using the menu "Navigate – Find hot SDTM Candidate“, the ODM item "Race“ is quickly found in the study design tree“, and the subsequent drag-and-drop leads to:


A CodeList is associated with ODM Item 'DM - Race'

The ODM ItemDef is ruled by a CodeList.
You can either keep using the associated ODM CodeList,
or take the current CodeList from the SDTM Variable (CL.C74457.RACE),
or select another CodeList from the define.xml structure.
In the case a CodeList from the SDTM / define.xml is selected,
a CodeList mapping will be necessary.
In all applicable cases, a (new) CodeList reference will be added to the
corresponding ItemDef in the define.xml structure.

- Use Study CodeList (from ODM) - Coded Values
- Use Study CodeList (from ODM) - Decoded Values
- [✓ Use CodeList from the SDTM Variable]
- Use another CodeList (from define.xml)

Previous

CL.C66767.ACN / Action Taken with Study Treatment
CL.C101865.ACSPCAT / Acute Coronary Syndrome Presentation Category
CL.C66781.AGEU / Age Unit
CL.C74456.LOC / Anatomical Location
CL.C101842.CACRDSC / Canadian Cardiovascular Society Classification
CL.C101866.CRYDFMAN / Cardiac Rhythm Device Failure Manifestation
CL.C101853.CVLVRSEV / Cardiac Valvular Regurgitation Severity
CL.C101854.CVLSTSEV / Cardiac Valvular Stenosis Severity
CL.C101847.CVEXAMCD / Cardiovascular Test Code
CL.C101846.CVEXAM / Cardiovascular Test Name

Show CodeList

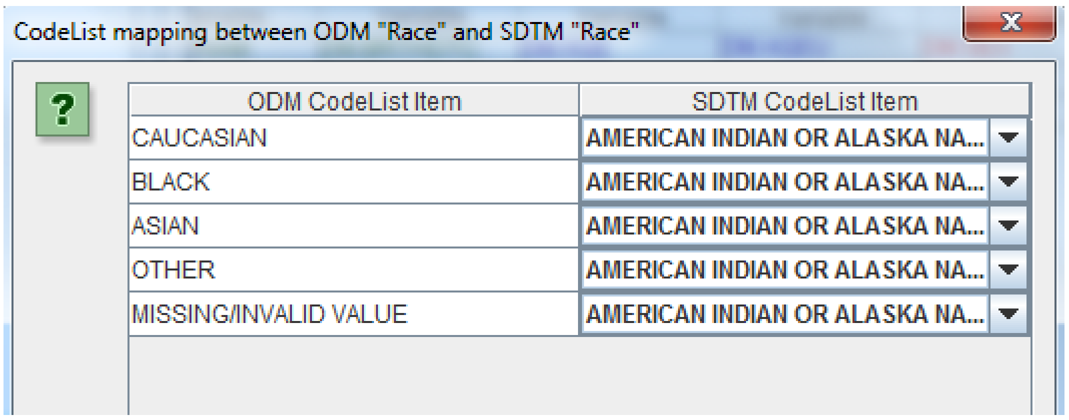
OK Cance

Variabl
DM.ETHNIC
DM.ETHNIC

CodeList details for CL.C74457.RACE

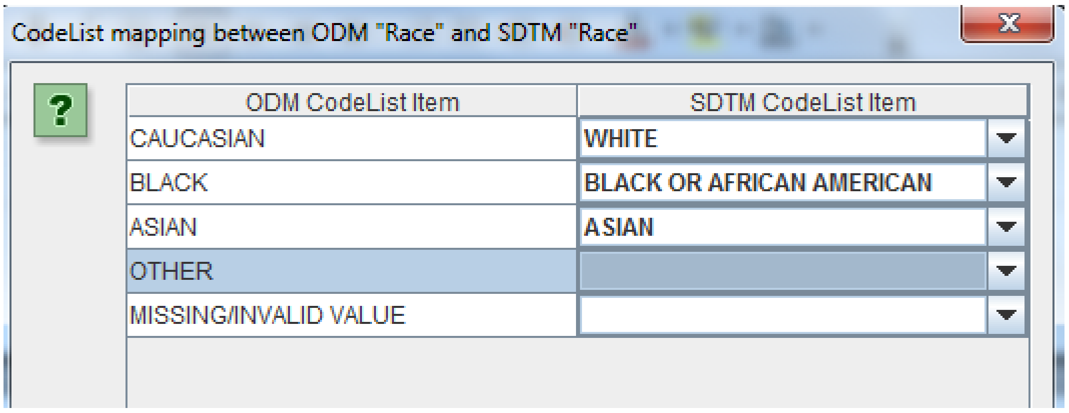
Coded Value	Decode
AMERICAN INDIAN OR ALASKA NATIVE	
ASIAN	
BLACK OR AFRICAN AMERICAN	
NATIVE HAWAIIAN OR OTHER PACIFI...	
WHITE	

and the codelist mapping wizard:


CodeList mapping between ODM "Race" and SDTM "Race"

ODM CodeList Item	SDTM CodeList Item
CAUCASIAN	AMERICAN INDIAN OR ALASKA NA...
BLACK	AMERICAN INDIAN OR ALASKA NA...
ASIAN	AMERICAN INDIAN OR ALASKA NA...
OTHER	AMERICAN INDIAN OR ALASKA NA...
MISSING/INVALID VALUE	AMERICAN INDIAN OR ALASKA NA...

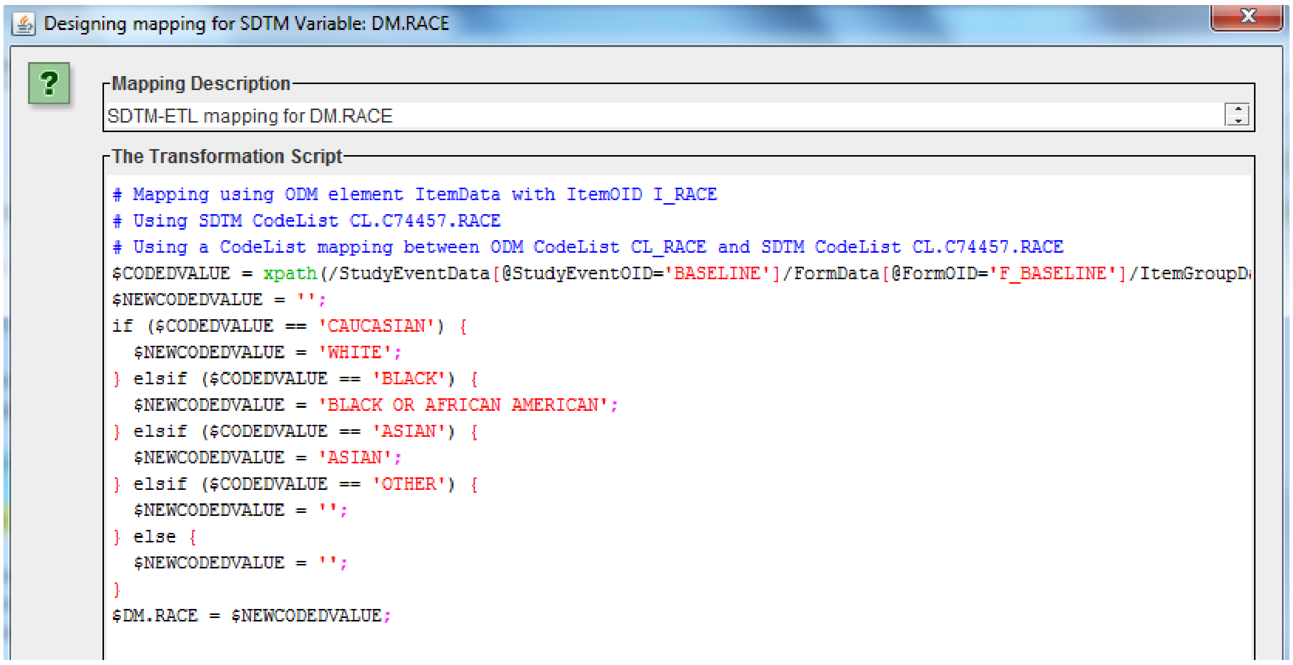
which is easily mapped to:


CodeList mapping between ODM "Race" and SDTM "Race"

ODM CodeList Item	SDTM CodeList Item
CAUCASIAN	WHITE
BLACK	BLACK OR AFRICAN AMERICAN
ASIAN	ASIAN
OTHER	
MISSING/INVALID VALUE	

The two ODM entries mapped to "NULL“ (empty).

This leads to the automatically generated mapping script:


Designing mapping for SDTM Variable: DM.RACE

Mapping Description

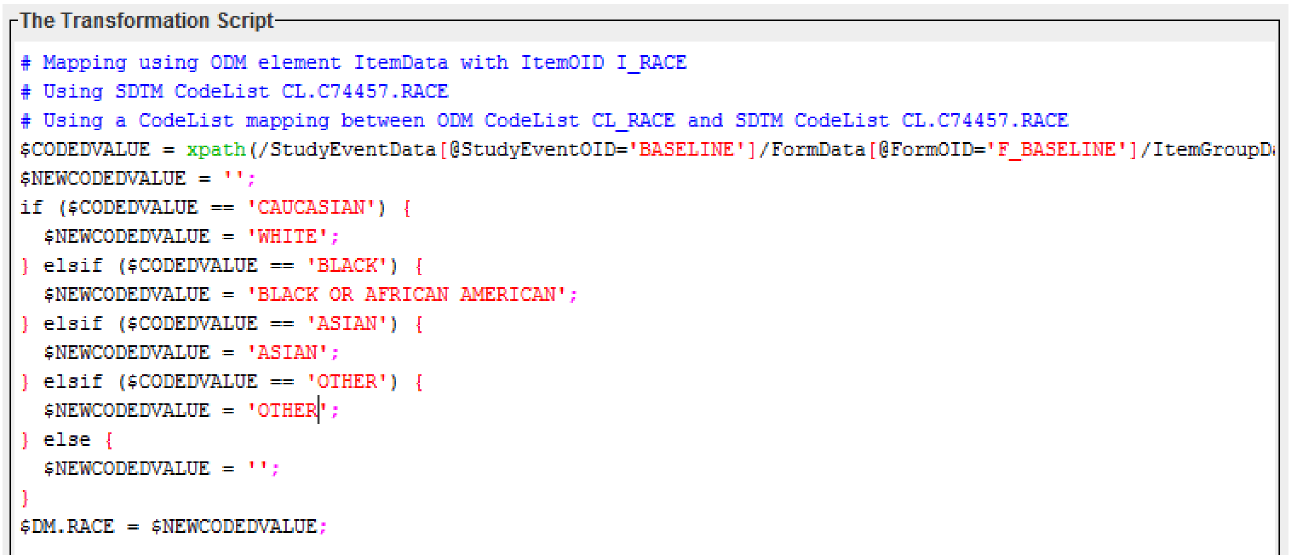
SDTM-ETL mapping for DM.RACE

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_RACE
# Using SDTM CodeList CL.C74457.RACE
# Using a CodeList mapping between ODM CodeList CL_RACE and SDTM CodeList CL.C74457.RACE
$CODEDVALUE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupD
$NEWCODEDVALUE = '';
if ($CODEDVALUE == 'CAUCASIAN') {
  $NEWCODEDVALUE = 'WHITE';
} elsif ($CODEDVALUE == 'BLACK') {
  $NEWCODEDVALUE = 'BLACK OR AFRICAN AMERICAN';
} elsif ($CODEDVALUE == 'ASIAN') {
  $NEWCODEDVALUE = 'ASIAN';
} elsif ($CODEDVALUE == 'OTHER') {
  $NEWCODEDVALUE = '';
} else {
  $NEWCODEDVALUE = '';
}
$DM.RACE = $NEWCODEDVALUE;
```

"Other“ is not part of the official CDISC codelist, but we could of course add it (later we will see how), and add it to the mapping script. In that case, depending on whether the study design had also a "please specify“ field, one should also add a supplemental qualifier (RACEOTH)[23] to provide the information on the "other race“.

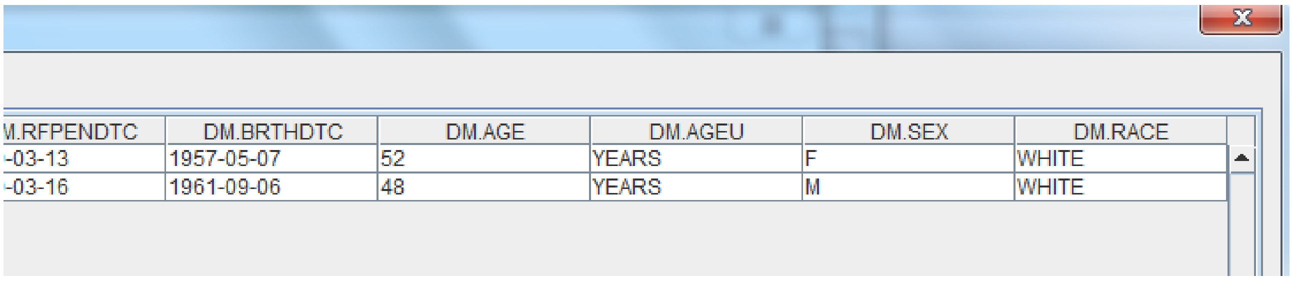
If we change the mapping script to:



The Transformation Script

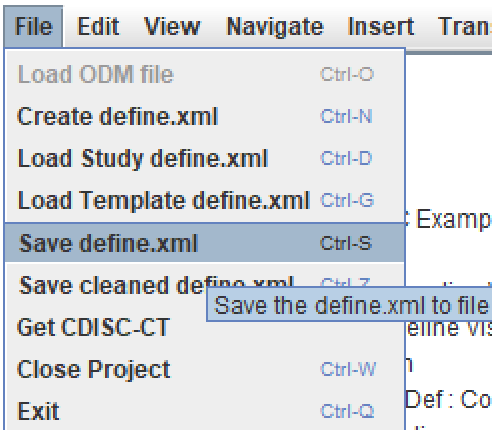
```
# Mapping using ODM element ItemData with ItemOID I_RACE
# Using SDTM CodeList CL.C74457.RACE
# Using a CodeList mapping between ODM CodeList CL_RACE and SDTM CodeList CL.C74457.RACE
$CODEDVALUE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupD
$NEWCODEDVALUE = '';
if ($CODEDVALUE == 'CAUCASIAN') {
  $NEWCODEDVALUE = 'WHITE';
} elsif ($CODEDVALUE == 'BLACK') {
  $NEWCODEDVALUE = 'BLACK OR AFRICAN AMERICAN';
} elsif ($CODEDVALUE == 'ASIAN') {
  $NEWCODEDVALUE = 'ASIAN';
} elsif ($CODEDVALUE == 'OTHER') {
  $NEWCODEDVALUE = 'OTHER';
} else {
  $NEWCODEDVALUE = '';
}
$DM.RACE = $NEWCODEDVALUE;
```

and test the mapping for the whole domain, the following result is obtained (partial view):



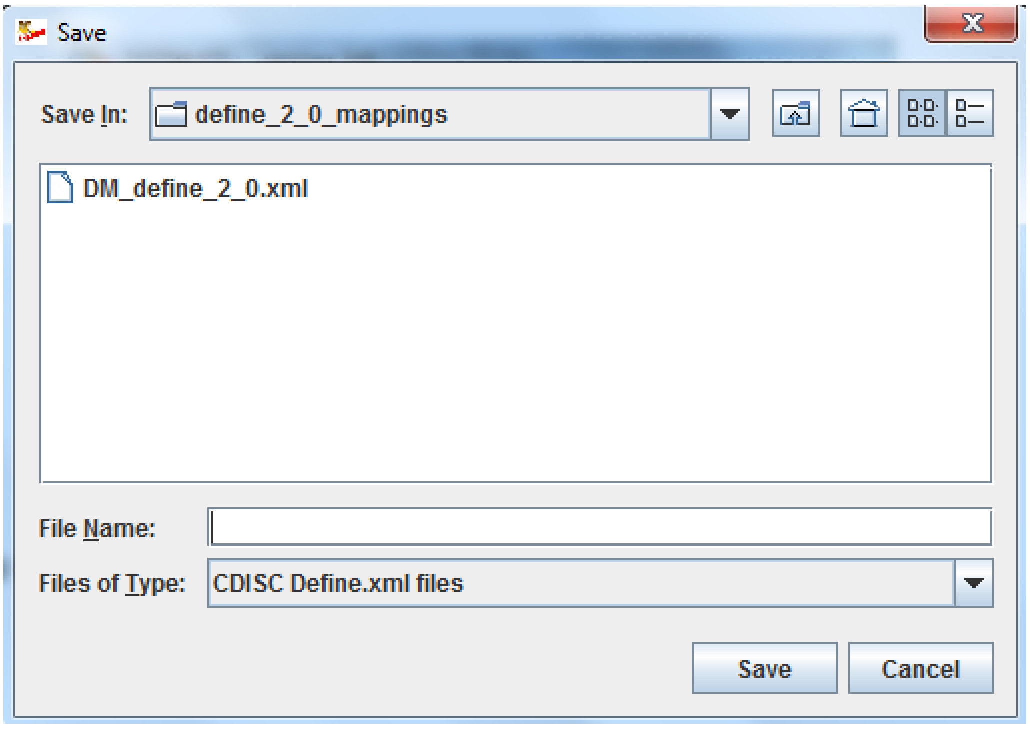
M.RFPENDTC	DM.BRTHDTC	DM.AGE	DM.AGEU	DM.SEX	DM.RACE
-03-13	1957-05-07	52	YEARS	F	WHITE
-03-16	1961-09-06	48	YEARS	M	WHITE

It's not a bad idea to save all the work done sofar. This is accomplished by using the menu "File – Save define.xml“ (or using CTRL-S):


File Edit View Navigate Insert Tran
Load ODM file Ctrl-O
Create define.xml Ctrl-N
Load Study define.xml Ctrl-D
Load Template define.xml Ctrl-G
Save define.xml Ctrl-S
Save cleaned define.xml Ctrl-Z
Save the define.xml to file
Get CDISC-CT
Close Project Ctrl-W
Exit Ctrl-Q

23 The way "other race" is handled may depend on the regulatory authority, and has been changed a few times in the history of SDTM. So please consult the latest regulatory requirements.

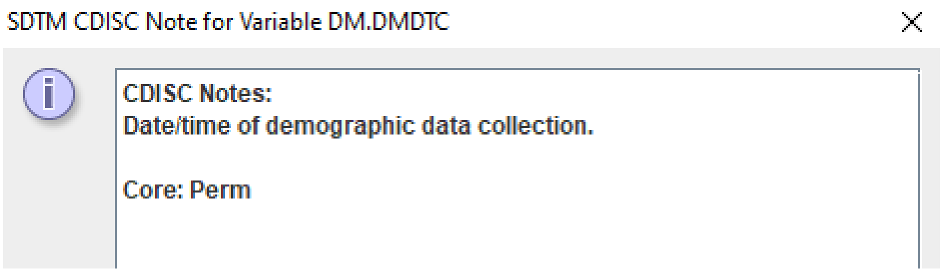
and selecting a location and name for our file, e.g. "DM_define_2_0.xml“:



Remark that all the current work is automatically saved every 5 minutes as a define.xml in the folder "autosave". The amount of time between such "backups" can be changed by using the "Options" menu. The auto-saved define.xml files can help in recovering e.g. after a computer crash, or when one made mistakes and wants to revert to a prior situation.

The "Country" is fixed in this study. So one can just add $DM.COUNTRY = 'xxx' where 'xxx' is the three character code in ISO-3166 notation. Examples are: USA (United States of America), CAN (Canada), GER (Germany), AUT (Austria), AUS (Australia).

The next variable is DMTDC. When using CTRL-H, more information is displayed:

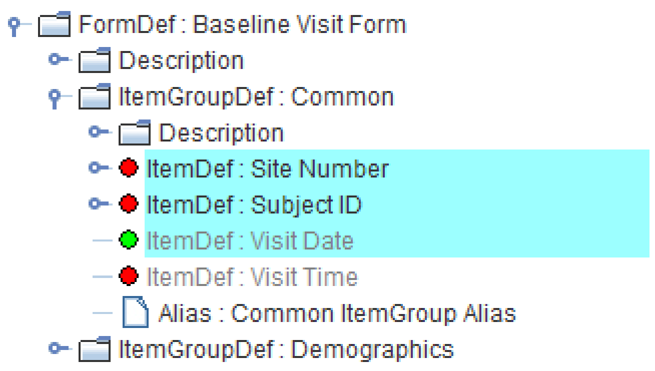


SDTM CDISC Note for Variable DM.DMDTC

CDISC Notes:
Date/time of demographic data collection.

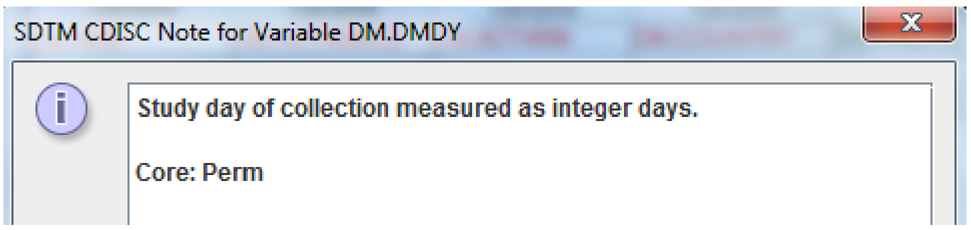
Core: Perm

We can just take it as the "Visit Date“ for the form where also the demographics data was collected:


FormDef : Baseline Visit Form
Description
ItemGroupDef : Common
Description
ItemDef : Site Number
ItemDef : Subject ID
ItemDef : Visit Date
ItemDef : Visit Time
Alias : Common ItemGroup Alias
ItemGroupDef : Demographics

In this case, a simple drag-and-drop from the item "Visit date" is all is needed.

The next one is DMDY:

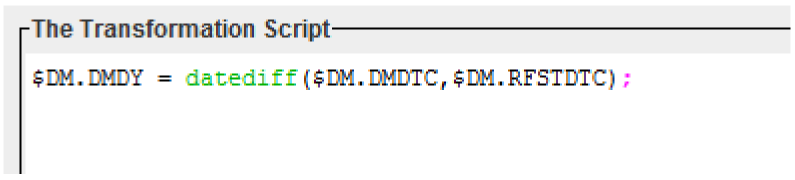

SDTM CDISC Note for Variable DM.DMDY

Study day of collection measured as integer days.

Core: Perm

There is something very special here (see the SDTM-IG). In SDTM, the day the study starts for a specific subject has xxDY = 1 (and not 0 as one might think). The day before the study starts however is then not day 0, as one might think, but day -1. So in SDTM, there is no day "0", and xxDY can never have the value "0". Logical, isn't it[24]?

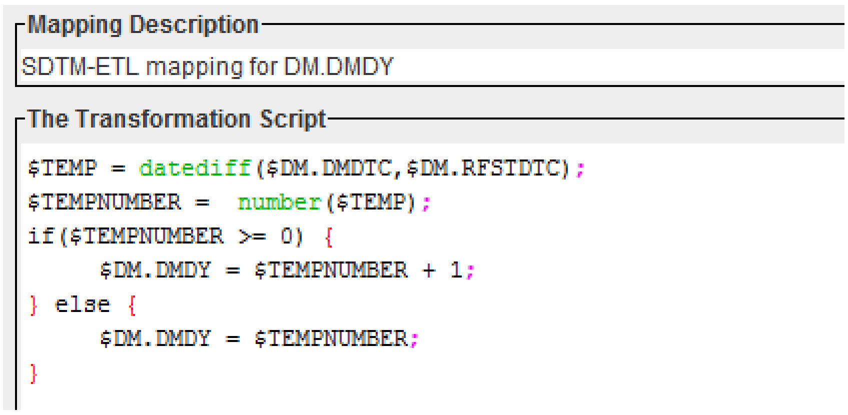
So when calculating xxDY, we must always add logic in our script to avoid that a value "0" is given as the result. In this case, it is pretty simple – we can even reuse variables that were defined before. For DMDY, we write the mapping:


The Transformation Script

```
$DM.DMDY = datediff($DM.DMDTC,$DM.RFSTDTC);
```

DM.DMDTC and DM.RFSTDTC have been defined before (i.e. more to the left), so we can reuse them in "read only" mode. The "datediff()" function delivers the difference in days. In case the first parameter value is later than the second, a positive (or better said, non-negative) result is obtained. One immediately sees that this can lead to a DMDY=0 result when DMDTC and RFSTDTC are identical (as is the case)[25]. So we adapt the mapping to:

[24]That was meant sceptically ...
[25]Essentially, DMDY should never appear in SDTM, as SDTM is about collected data, not about derived data. The tools of the FDA should do these kind of calculations.

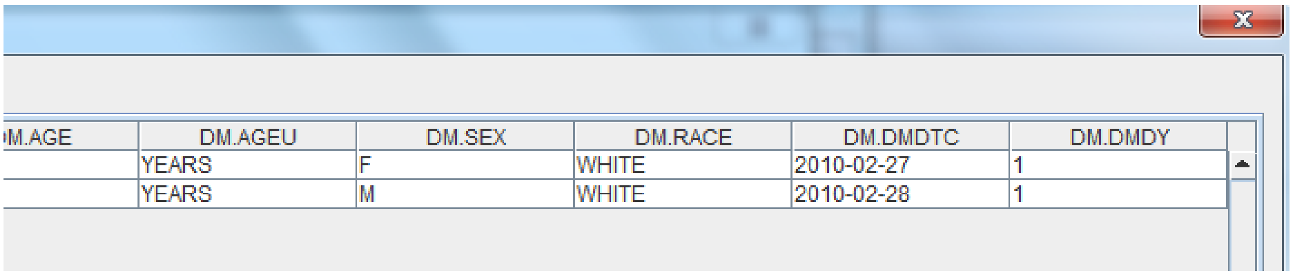


SDTM-ETL mapping for DM.DMDY

```
$TEMP = datediff($DM.DMDTC,$DM.RFSTDTC);
$TEMPNUMBER =  number($TEMP);
if($TEMPNUMBER >= 0) {
     $DM.DMDY = $TEMPNUMBER + 1;
} else {
     $DM.DMDY = $TEMPNUMBER;
}
```

There is one peculiarity in this script: the "datediff“ function essentially returns a string[26], which need to be transformed into a number (kind of casting) in order to do mathematical calculations with it.

The result for our two subjects is:



M.AGE	DM.AGEU	DM.SEX	DM.RACE	DM.DMDTC	DM.DMDY
	YEARS	F	WHITE	2010-02-27	1
	YEARS	M	WHITE	2010-02-28	1

In the next tutorial, we will work on the SV (subject visits) domain, and also introduce a new output format, and an alternative (better) "smart" viewer for inspecting the resulting records.

[26]The reason for this is that in XSLT, a datediff returns a duration, e.g. "P1D“ meaning a period of 1 day.