SDTM-ETL 4.4 User Manual and Tutorial

Author: Jozef Aerts, XML4Pharma

Last update: 2024-09-27

<u>Troubleshooting: Most common mistakes and errors</u>

Table of Contents

Table of Contents 1
Introduction 1
Most common error messages during "compilation" of the mapping scripts 1
- "assignment statement needs to end with a semicolon" 2
- "missing closing bracket in 'if' statement" 2
Most common error messages during execution of the mapping scripts 3
- "Typed" versus "Untyped" ODM clinical data. 3
- "Variable xxx has not been declared (or its declaration is not in scope)" 5
- "No mapping provided for a 'looping' variable" 6
- "A sequence of more than one item is not allowed as the first argument of fn:string()" 10
- "Invalid byte 1 of 1-byte UTF-8 sequence" 12
Further bad practices 12
- Sorting keys in the define.xml for variables that require post-processing 12

Introduction

The SDTM-ETL software makes development and execution of mappings between collected data (usually as CDISC-ODM) and SDTM or SEND easy. But still, users need to take decisions, and use the many wizards and dialogs in a wise way. In some cases, automatically scripts must be extended or adapted, or (very seldom) written from scratch.

Making mistakes is human, and also here, mistakes can be made, which lead to errors or unexpected results. The current document tries to make an inventory of most common mistakes and errors and how to either avoid or correct them.

Usually, the SDTM-ETL software will report such errors during execution of the mapping scripts, when using the menu "Transform - Generate Transformation (XSLT) Code for …", or later, after execution was started. We will explain most of the common error messages, and discuss measures to be taken to correct the scripts or the mapping strategy.

This document will regularly be updated with comments from our users.

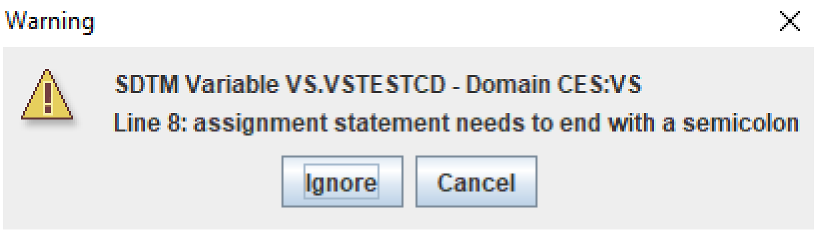
Most common error messages during "compilation" of the mapping scripts

The mapping scripts are, when the menu "Transform - Generate Transformation (XSLT) Code for

…" is used, transformed into XSLT (XML Transformation Language), which is then executed on the ODM-XML file with clinical data.
During generation of the XSLT, the software already checks a number of things, and the validity of the generated XSLT.

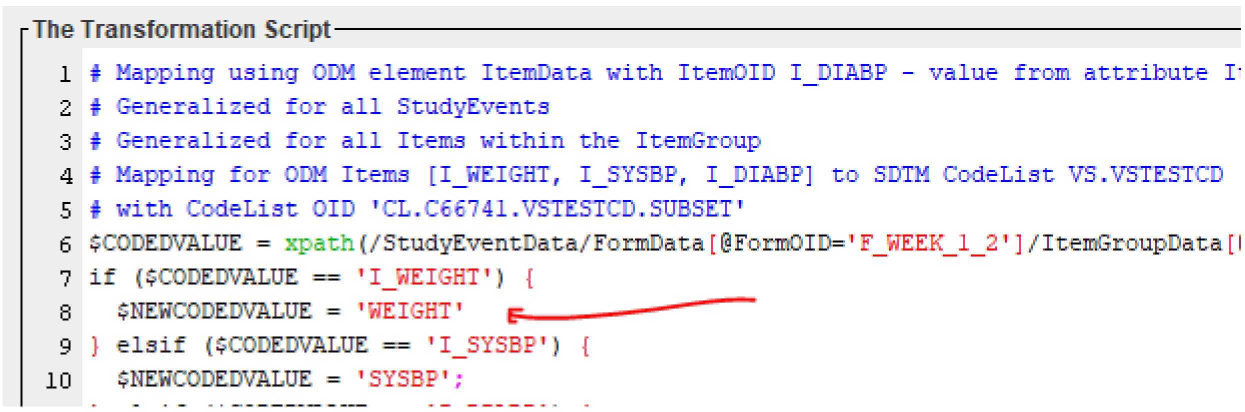
Here are the most common error messages generated:

- "assignment statement needs to end with a semicolon"



SDTM Variable VS.VSTESTCD - Domain CES:VS
Line 8: assignment statement needs to end with a semicolon

This is an obvious error that can occur when automatically generated mapping scripts have been edited, or written developed from scratch. When we then look into the script for VSTESTCD at line 8, we find:

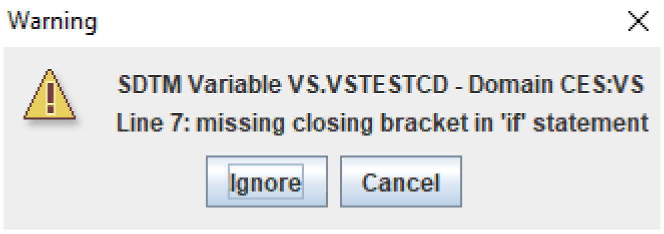


```
 1 # Mapping using ODM element ItemData with ItemOID I_DIABP - value from attribute I
 2 # Generalized for all StudyEvents
 3 # Generalized for all Items within the ItemGroup
 4 # Mapping for ODM Items [I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.VSTESTCD
 5 # with CodeList OID 'CL.C66741.VSTESTCD.SUBSET'
 6 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_WEEK_1_2']/ItemGroupData[
 7 if ($CODEDVALUE == 'I_WEIGHT') {
 8   $NEWCODEDVALUE = 'WEIGHT'
 9 } elsif ($CODEDVALUE == 'I_SYSBP') {
10   $NEWCODEDVALUE = 'SYSBP';
```

Where we see that a semicolon, which is needed to indicate the end of a programming statement is failing. One can e.g. see in line 10 what is needed.
In such a case, one can use the "Ignore" button, in which case the software will try to automatically correct the generated XSLT, but there is no guarantee that this will always work.

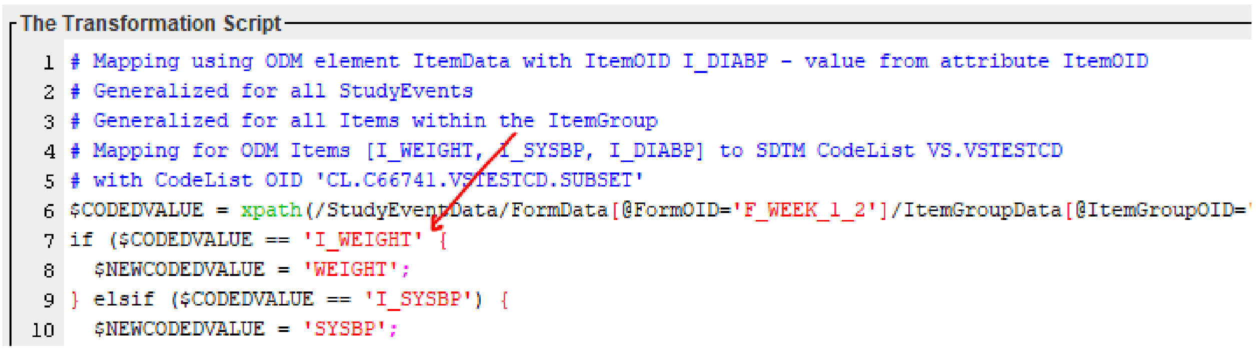
- "missing closing bracket in 'if' statement"

Is another obvious error, e.g.



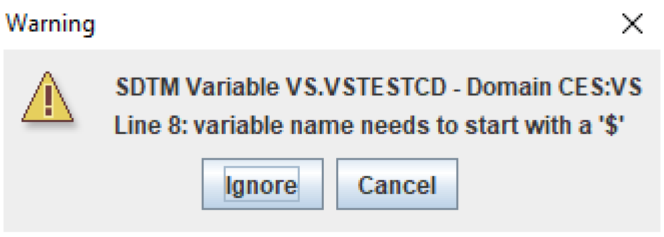
SDTM Variable VS.VSTESTCD - Domain CES:VS
Line 7: missing closing bracket in 'if' statement

Which can easily be retraced in the mapping script:


The Transformation Script

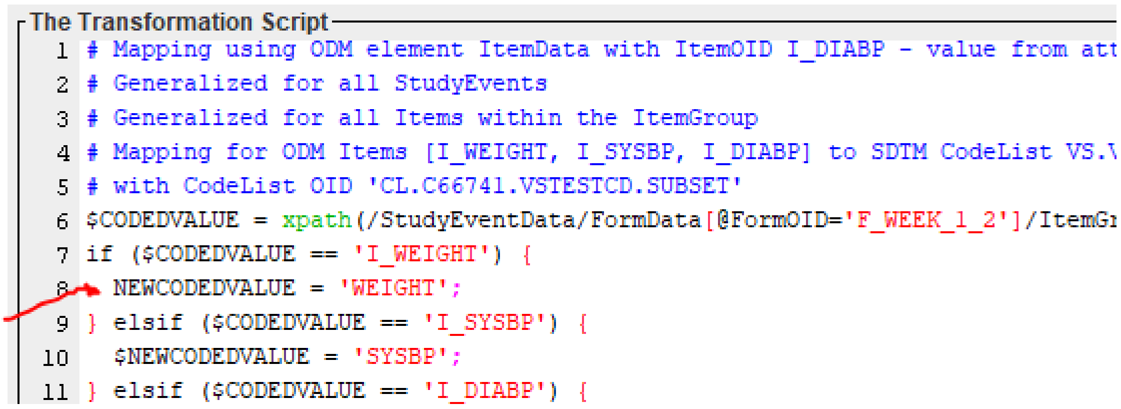
```
 1 # Mapping using ODM element ItemData with ItemOID I_DIABP - value from attribute ItemOID
 2 # Generalized for all StudyEvents
 3 # Generalized for all Items within the ItemGroup
 4 # Mapping for ODM Items [I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.VSTESTCD
 5 # with CodeList OID 'CL.C66741.VSTESTCD.SUBSET'
 6 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_WEEK_1_2']/ItemGroupData[@ItemGroupOID='
 7 if ($CODEDVALUE == 'I_WEIGHT' {
 8   $NEWCODEDVALUE = 'WEIGHT';
 9 } elsif ($CODEDVALUE == 'I_SYSBP') {
10   $NEWCODEDVALUE = 'SYSBP';
```

- "variable name needs to start with a '$'"


Warning

SDTM Variable VS.VSTESTCD - Domain CES:VS
Line 8: variable name needs to start with a '$'

Just like in Perl, PHP and JavaScript, variables are denoted by starting with a dollar ($) character. This is a consequence of our mapping script language being "untyped", meaning that variables are not explicitly assigned a data type (like "string", "integer", …), like e.g. in Java. See here for some explanation.
Also here, as the line number is provided, this error is easy to trace back:


The Transformation Script

```
 1 # Mapping using ODM element ItemData with ItemOID I_DIABP - value from att
 2 # Generalized for all StudyEvents
 3 # Generalized for all Items within the ItemGroup
 4 # Mapping for ODM Items [I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.\
 5 # with CodeList OID 'CL.C66741.VSTESTCD.SUBSET'
 6 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_WEEK_1_2']/ItemGı
 7 if ($CODEDVALUE == 'I_WEIGHT') {
 8   NEWCODEDVALUE = 'WEIGHT';
 9 } elsif ($CODEDVALUE == 'I_SYSBP') {
10   $NEWCODEDVALUE = 'SYSBP';
11 } elsif ($CODEDVALUE == 'I_DIABP') {
```

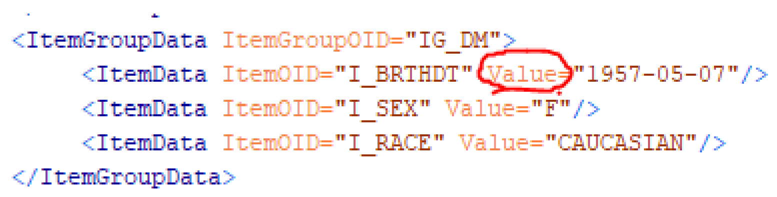
-

Most common error messages during execution of the mapping scripts

- "Typed" versus "Untyped" ODM clinical data.

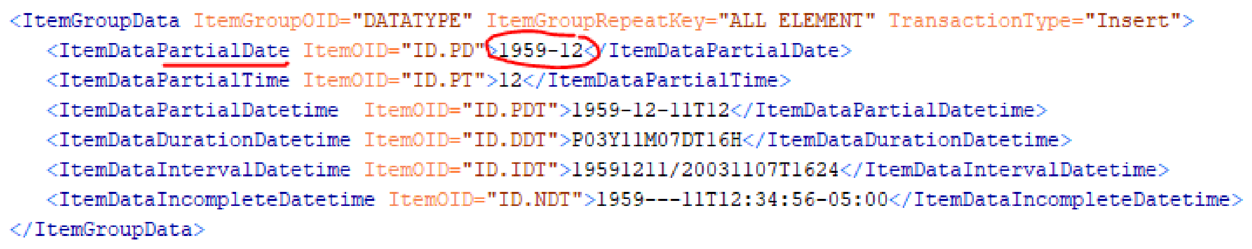
Once the XSLT generated, it can be applied to a file with clinical data in ODM format.
One must remark here that there a two "flavors" of ODM "ClinicalData", I.e. "untyped" and "typed".

Most EDC systems export ODM clinical data in the "untyped" flavor, which is characterized by having the value of the data point in a "Value" attribute. For example:



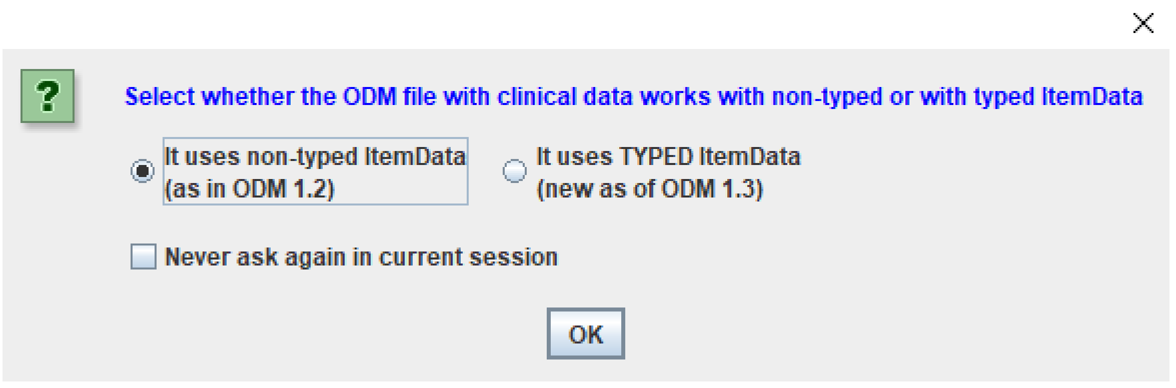
```
<ItemGroupData ItemGroupOID="IG_DM">
     <ItemData ItemOID="I_BRTHDT" Value="1957-05-07"/>
     <ItemData ItemOID="I_SEX" Value="F"/>
     <ItemData ItemOID="I_RACE" Value="CAUCASIAN"/>
</ItemGroupData>
```

Some (but less common) EDC systems however export the clinical data in the "typed" flavor, which is characterized by that the name of the "ItemData…" element has the data type in it, and that the value of the data point comes as XML "text content". For example:



```
<ItemGroupData ItemGroupOID="DATATYPE" ItemGroupRepeatKey="ALL ELEMENT" TransactionType="Insert">
   <ItemDataPartialDate ItemOID="ID.PD">1959-12</ItemDataPartialDate>
   <ItemDataPartialTime ItemOID="ID.PT">12</ItemDataPartialTime>
   <ItemDataPartialDatetime  ItemOID="ID.PDT">1959-12-11T12</ItemDataPartialDatetime>
   <ItemDataDurationDatetime ItemOID="ID.DDT">P03Y11M07DT16H</ItemDataDurationDatetime>
   <ItemDataIntervalDatetime ItemOID="ID.IDT">19591211/20031107T1624</ItemDataIntervalDatetime>
   <ItemDataIncompleteDatetime ItemOID="ID.NDT">1959---11T12:34:56-05:00</ItemDataIncompleteDatetime>
</ItemGroupData>
```

Therefore, when one starts the execution, the system will request to indicate which "flavor" is used in your dataset with clinical data:

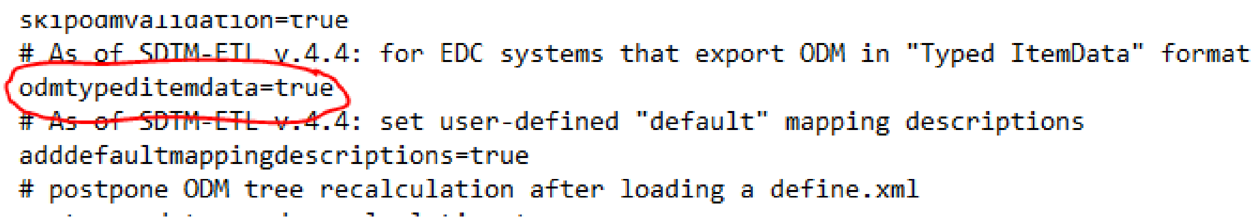


Select whether the ODM file with clinical data works with non-typed or with typed ItemData

[✓ 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

One should then select the correct "flavor". If one is 100% sure and does not want that the systems asks for it each time the mappings are executed, check the checkbox "Never ask again in current session".
One can also preset the choice in the "properties.dat" file when one will always work with files from the same (type of) EDC system, e.g.

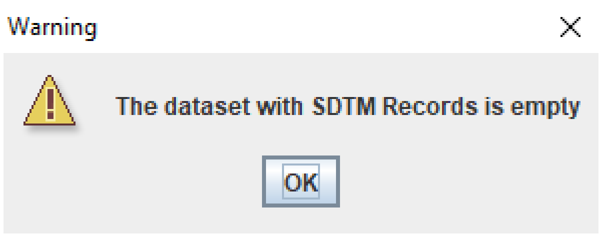


```
skipodmvalidation=true
# As of SDTM-ETL v.4.4: for EDC systems that export ODM in "Typed ItemData" format
odmtypeditemdata=true
# As of SDTM-ETL v.4.4: set user-defined "default" mapping descriptions
adddefaultmappingdescriptions=true
# postpone ODM tree recalculation after loading a define.xml
```

indicating that the "default" is "typed ItemData".

Now, what happens if one makes the false choice?

In such a case, nothing special will happen, no errors will be generated, the mappings will just execute, but as, at least for the variables for which the value is extracted from the ODM file with clinical data (all these where an "xpath(…)" statement occurs in the mapping script), no SDTM/SEND values will be generated. In the best case, a message:


Warning

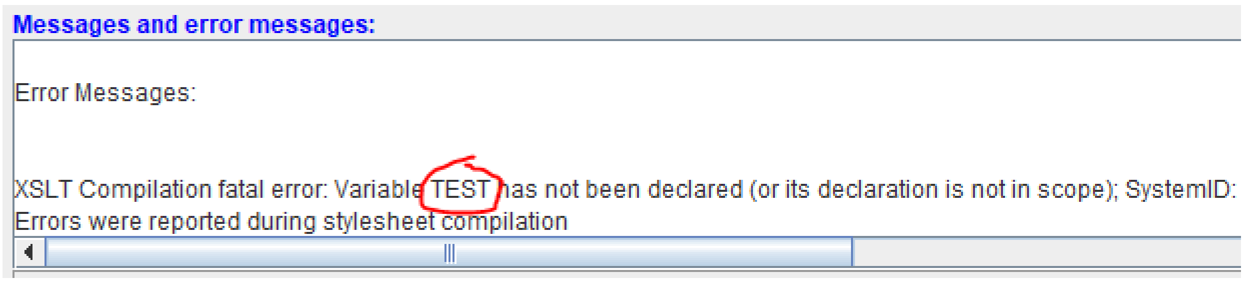
The dataset with SDTM Records is empty

will appear.

So, in the case that this message appears, one should first always ask oneself "is my clinical data in the 'untyped' or in the 'typed' flavor?" and "did I make the right choice when the system asked be about it?". There can of course also be other reasons why the generated datasets are empty.

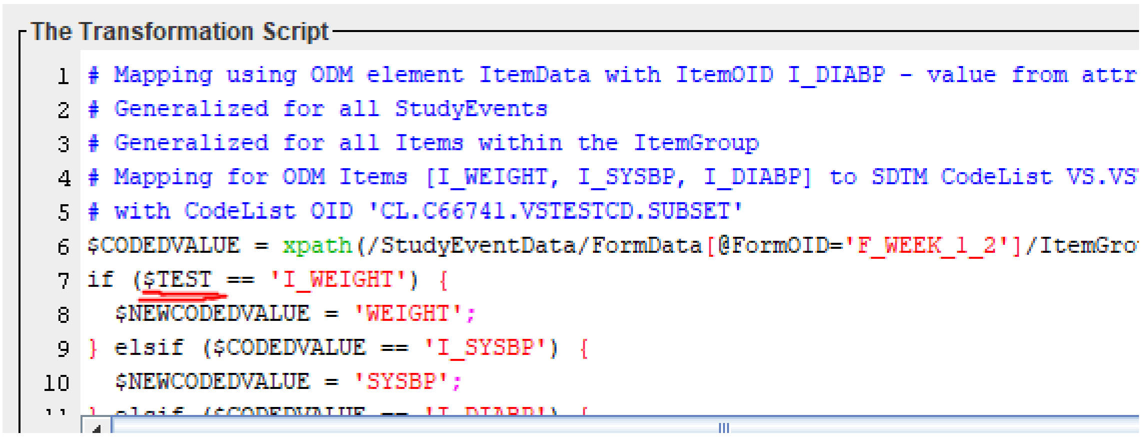
- "Variable xxx has not been declared (or its declaration is not in scope)"

This error message means that a variable was used (i.e. read in an operation), but the variable was not declared before. E.g.:


Messages and error messages:

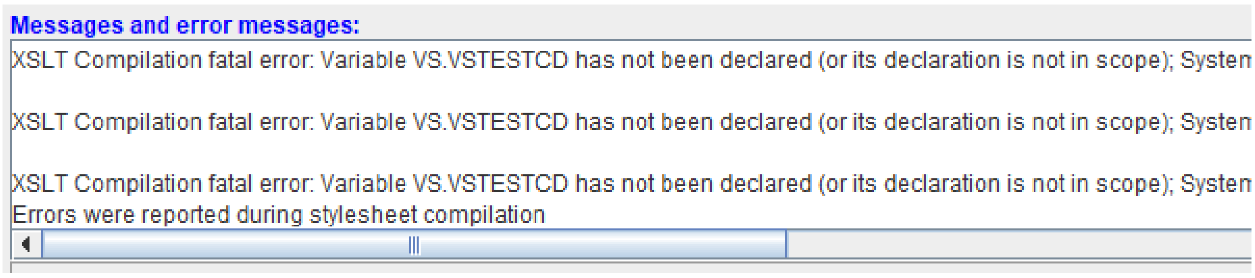
```
Error Messages:

XSLT Compilation fatal error: Variable TEST has not been declared (or its declaration is not in scope); SystemID:
Errors were reported during stylesheet compilation
```

where in the mapping script we find:

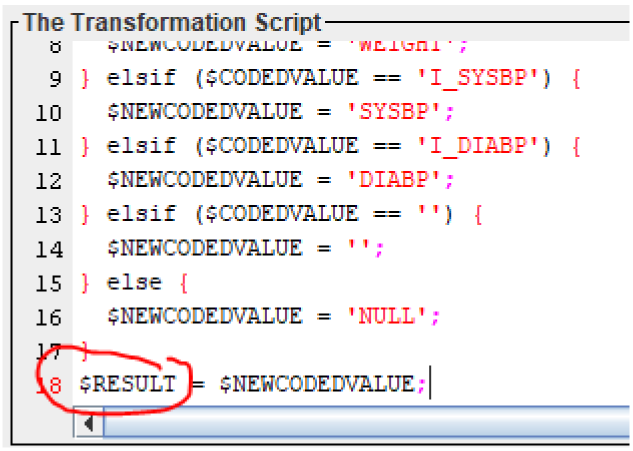

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_DIABP - value from attr
# Generalized for all StudyEvents
# Generalized for all Items within the ItemGroup
# Mapping for ODM Items [I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.VS
# with CodeList OID 'CL.C66741.VSTESTCD.SUBSET'
$CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_WEEK_1_2']/ItemGro
if ($TEST == 'I_WEIGHT') {
  $NEWCODEDVALUE = 'WEIGHT';
} elsif ($CODEDVALUE == 'I_SYSBP') {
  $NEWCODEDVALUE = 'SYSBP';
```

where we see that the variable "$TEST" is used in an "if" statement, but was never declared before. In this case, it is obvious that "$TEST" must be replaced by "$CODEDVALUE".
Rather often, this error is at the end of the script, in the last assignment, e.g.:


```
Messages and error messages:
XSLT Compilation fatal error: Variable VS.VSTESTCD has not been declared (or its declaration is not in scope); Systen

XSLT Compilation fatal error: Variable VS.VSTESTCD has not been declared (or its declaration is not in scope); Systen

XSLT Compilation fatal error: Variable VS.VSTESTCD has not been declared (or its declaration is not in scope); Systen
Errors were reported during stylesheet compilation
```

Which is less obvious as the script is:


```
The Transformation Script
  $NEWCODEDVALUE = 'WEIGHT';
} elsif ($CODEDVALUE == 'I_SYSBP') {
  $NEWCODEDVALUE = 'SYSBP';
} elsif ($CODEDVALUE == 'I_DIABP') {
  $NEWCODEDVALUE = 'DIABP';
} elsif ($CODEDVALUE == '') {
  $NEWCODEDVALUE = '';
} else {
  $NEWCODEDVALUE = 'NULL';
}
$RESULT = $NEWCODEDVALUE;
```

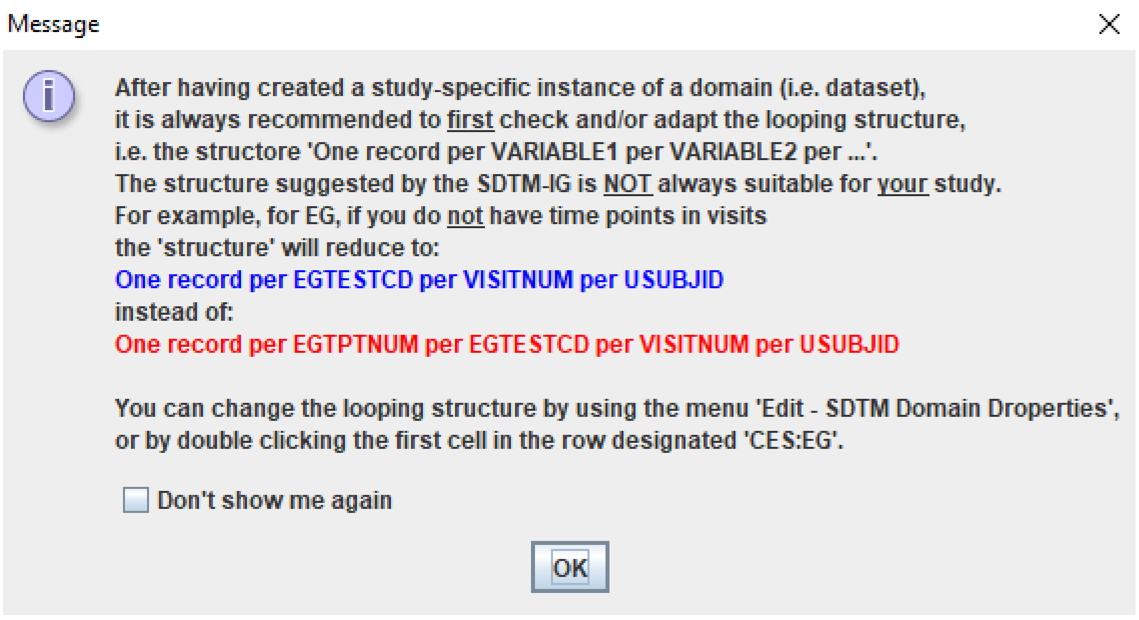
Which seems to be correct ...
Reason is that it is expected that the last statement in the mapping script always does the assignment to the SDTM/SEND variable that the script is meant for. In this case, the script is for VS.VSTESTCD, so the last assignment should be:
$VS.VSTESTCD = $NEWCODEDVALUE;

P.S.: there is no need that variable names are all uppercase, except for the SDTM/SEND variable itself, it is just a coding style …

The source for this kind of errors is often more difficult to find out, as the error message does not state in which script the error was made (as the script was already transformed to XSLT). Therefore, it is good custom to always test a mapping immediately after it was developed. If the error then occurs, it is clear that the source of it is the last mapping script that was developed.

- "No mapping provided for a 'looping' variable"

When one "instantiates" a domain for adding mappings for it, I.e. Drag-and-drop it from a template row to the bottom, a message appears, e.g.:


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 EG, if you do not have time points in visits
the 'structure' will reduce to:
One record per EGTESTCD per VISITNUM per USUBJID
instead of:
One record per EGTPTNUM per EGTESTCD per VISITNUM per USUBJID

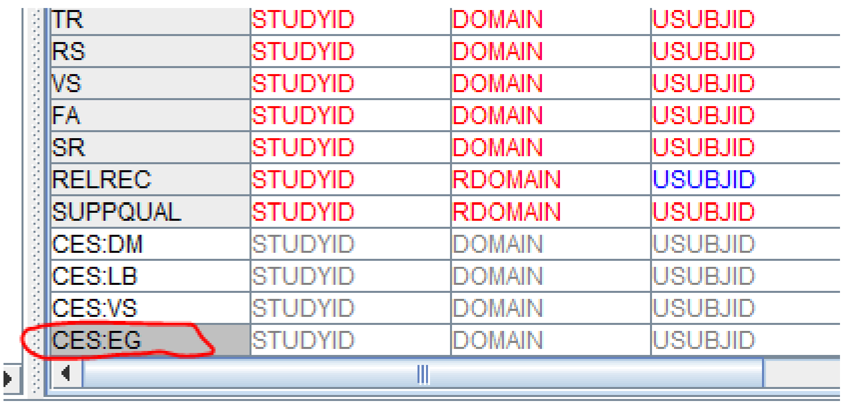
You can change the looping structure by using the menu 'Edit - SDTM Domain Droperties',
or by double clicking the first cell in the row designated 'CES:EG'.

Stating that one should first decide on the "structure" of the dataset, i.e. over which SDTM/SEND variables the system will iterate when generating the dataset.
In most cases, it will be sufficient to have following "structures" for the following SDTM/SEND "classes":

Class	"Looping" variables	Structure as displayed
Findings	1. USUBJID 2. xxTESTCD	One record per xxTESTCD per USUBJID
Events	1. USUBJID 2. xxTERM	One record per xxTERM per USUBJID
Interventions	1. USUBJID 2. xxTRT	One record per xxTRT per USUBJID

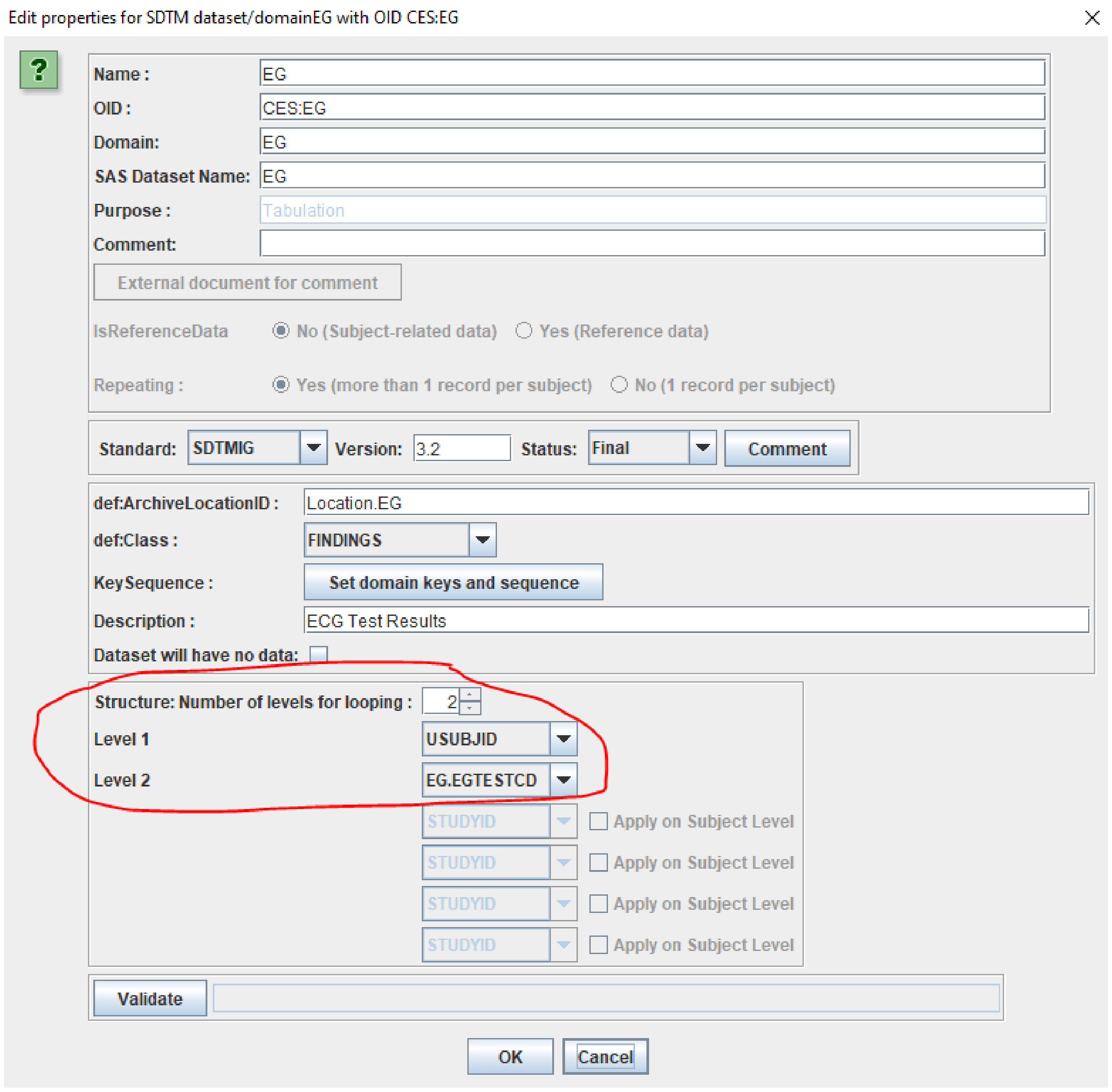
where "xx" designates the domain code.

So, the first step after "instantiating" a domain, one should always at least check the "structure". This can be done by a double-click on the first cell (defining the dataset/domain) of the new generated row. E.g. for "EG":



TR	STUDYID	DOMAIN	USUBJID
RS	STUDYID	DOMAIN	USUBJID
VS	STUDYID	DOMAIN	USUBJID
FA	STUDYID	DOMAIN	USUBJID
SR	STUDYID	DOMAIN	USUBJID
RELREC	STUDYID	RDOMAIN	USUBJID
SUPPQUAL	STUDYID	RDOMAIN	USUBJID
CES:DM	STUDYID	DOMAIN	USUBJID
CES:LB	STUDYID	DOMAIN	USUBJID
CES:VS	STUDYID	DOMAIN	USUBJID
CES:EG	STUDYID	DOMAIN	USUBJID

leading to the dialog:



Edit properties for SDTM dataset/domainEG with OID CES:EG

Name :	EG
OID :	CES:EG
Domain:	EG
SAS Dataset Name:	EG
Purpose :	Tabulation
Comment:	

IsReferenceData [✓ No (Subject-related data)] Yes (Reference data)

Repeating : [✓ Yes (more than 1 record per subject)] No (1 record per subject)

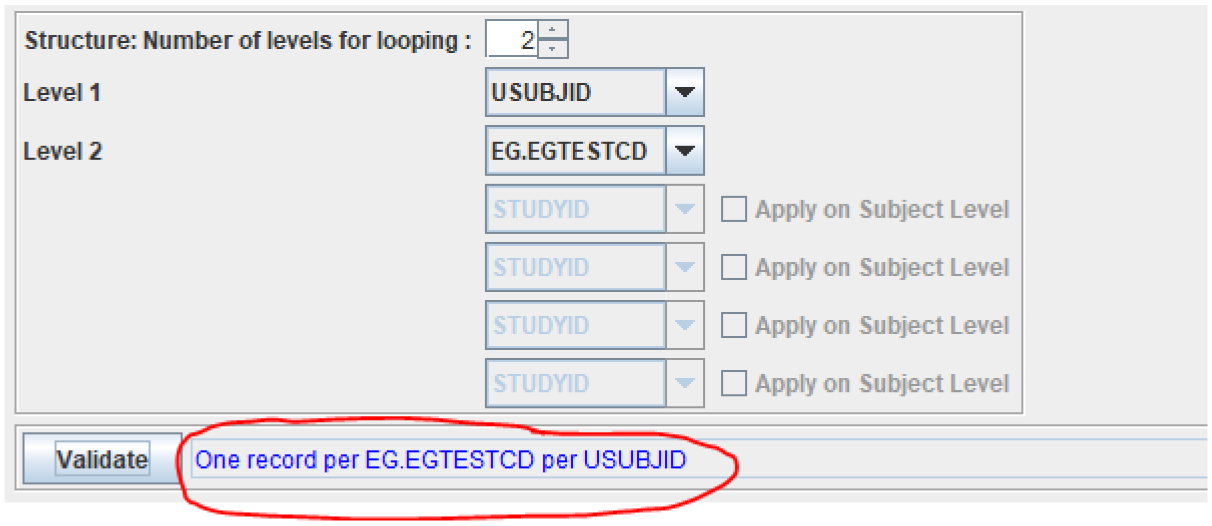
Standard: SDTMIG Version: 3.2 Status: Final

def:ArchiveLocationID :	Location.EG
def:Class :	FINDINGS
KeySequence :	Set domain keys and sequence
Description :	ECG Test Results
Dataset will have no data:	☐

Structure: Number of levels for looping : 2

Level 1	USUBJID	
Level 2	EG.EGTESTCD	
	STUDYID	☐ Apply on Subject Level
	STUDYID	☐ Apply on Subject Level
	STUDYID	☐ Apply on Subject Level
	STUDYID	☐ Apply on Subject Level

where, in the lower part, the "looping variables" are defined.
When one than clicks the button "Validate" one sees:

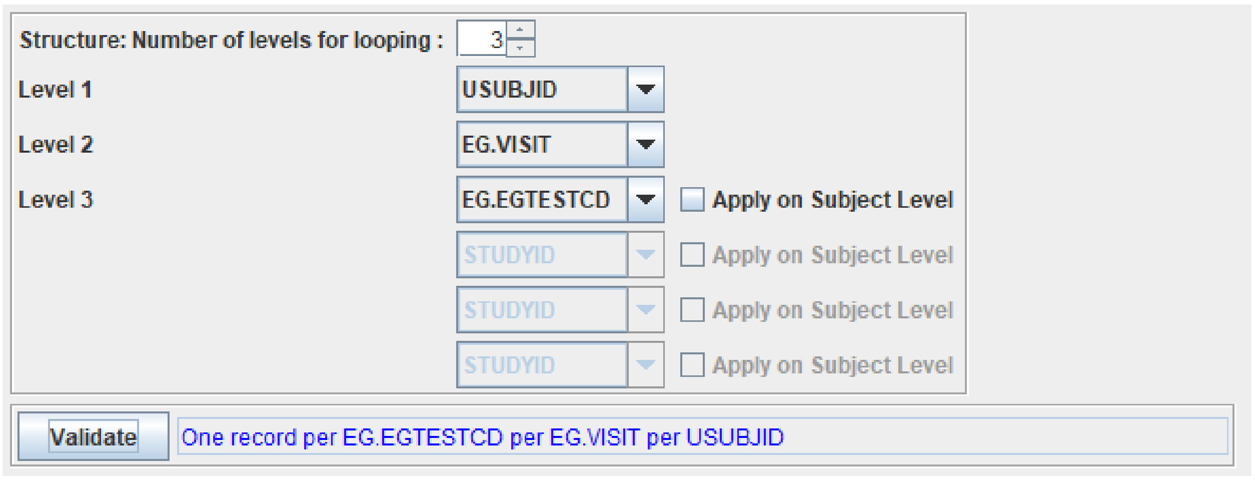


Structure: Number of levels for looping : 2

Level 1	USUBJID	
Level 2	EG.EGTESTCD	
	STUDYID	☐ Apply on Subject Level
	STUDYID	☐ Apply on Subject Level
	STUDYID	☐ Apply on Subject Level
	STUDYID	☐ Apply on Subject Level

One record per EG.EGTESTCD per USUBJID

Usually, for a "Findings" domain, this will be sufficient, as when one then uses "Generalize for all StudyEvents" (i.e. "visits) when developing the mapping for EGTESTCD, the system will automatically know that it needs to iterate over the visits anyway.
Some users however want to be more explicit, e.g. as they develop "special" mapping scripts for "VISIT"[1], and then change the "structure" e.g. to:

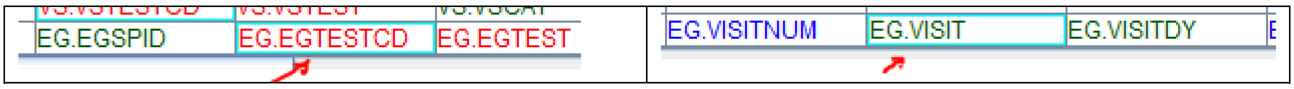

Structure: Number of levels for looping : 3

Level 1	USUBJID	
Level 2	EG.VISIT	
Level 3	EG.EGTESTCD	☐ Apply on Subject Level
	STUDYID	☐ Apply on Subject Level
	STUDYID	☐ Apply on Subject Level
	STUDYID	☐ Apply on Subject Level

Validate | One record per EG.EGTESTCD per EG.VISIT per USUBJID

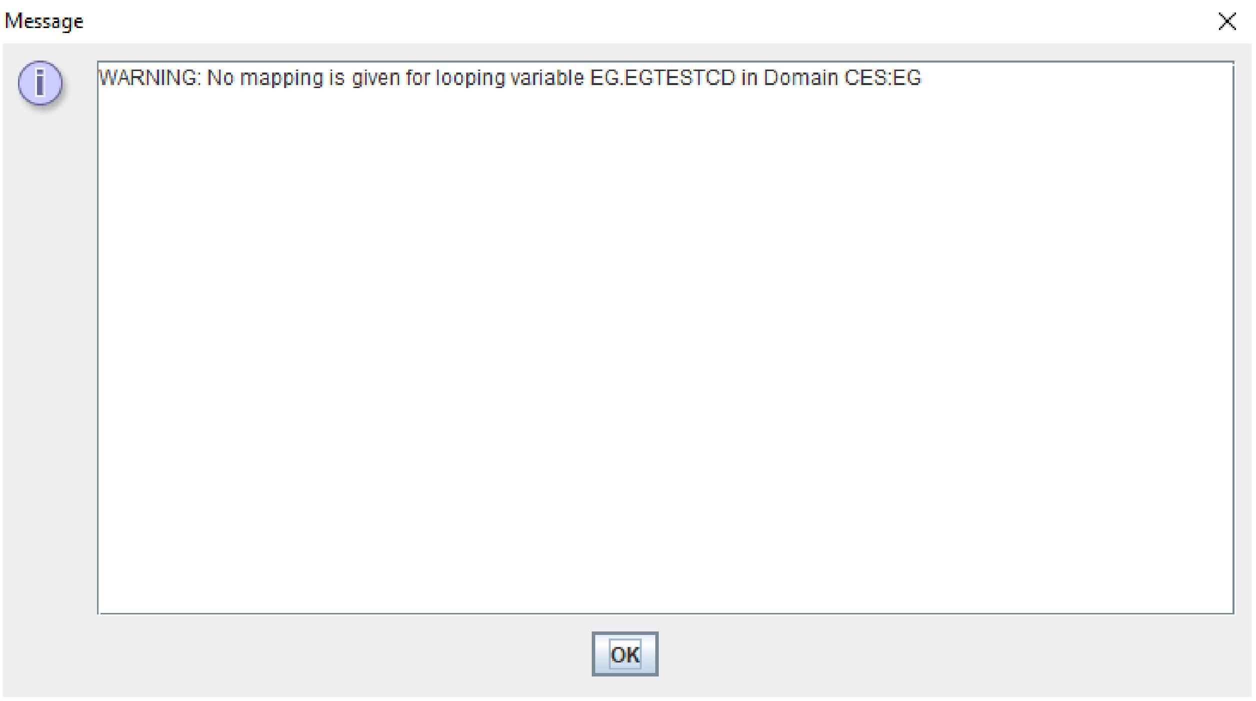
In such a case, it is important that the chosen "structure" is logical and hierarchical correct. For example it doesn't make sense to have EG.EGTESTCD as the first level, and USUBJID as the second.

In the SDTM/SEND table on the right, one can easily recognize the looping variables, as they have a blue-cyan border, e.g.:

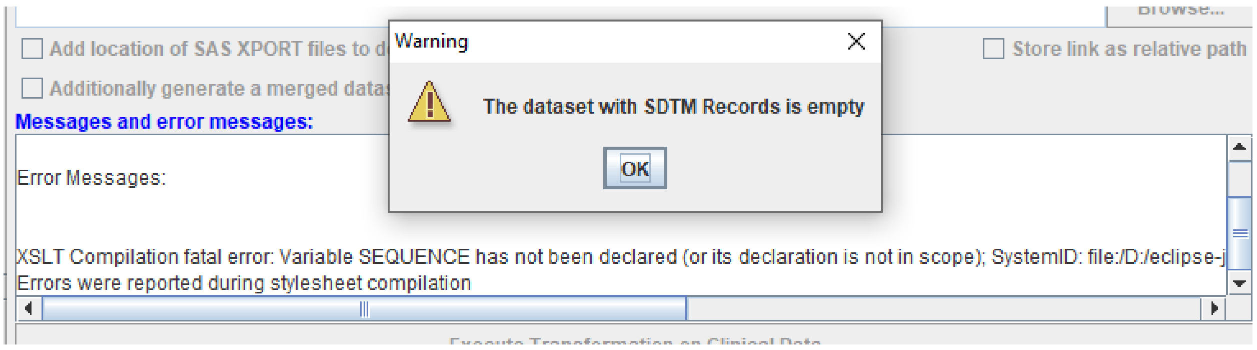


Important now is that one always has a mapping for each of the "looping" variables when executing the mappings on clinical data. Suppose for example that we added a mapping for "EG.VISIT" (usually derived from the ODM "StudyEventOID"), but <u>not</u> yet for EG.EGTESTCD.
When one then executes the mappings, the following warning will be be displayed:

[1] This may e.g. happen when several "StudyEvents" represent a single "visit" (in the sense of SDTM/SEND), i.e. several "StudyEvents" are contracted into a single "visit".


```
WARNING: No mapping is given for looping variable EG.EGTESTCD in Domain CES:EG
```

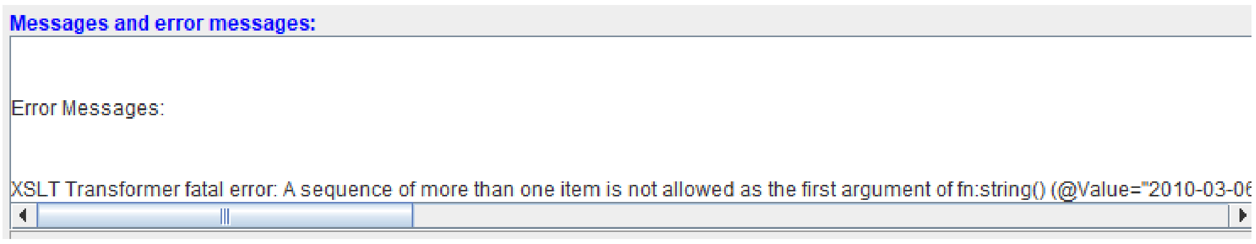
Some users however choose to ignore this warning and do continue (which is possible), leading to e.g.:


```
The dataset with SDTM Records is empty
```
```
Error Messages:

XSLT Compilation fatal error: Variable SEQUENCE has not been declared (or its declaration is not in scope); SystemID: file:/D:/eclipse-j
Errors were reported during stylesheet compilation
```
Add location of SAS XPORT files to d
Additionally generate a merged datas
Messages and error messages:
Warning
OK
Store link as relative path
Browse...

for which it is not immediately obvious what the source of the problem is.

P.S. Users have reported to us that in the case the "looping variables" have not been selected adequately, and for one of more of them no mapping has been provided, this had led to a "Fatal Error" during execution of the mapping.

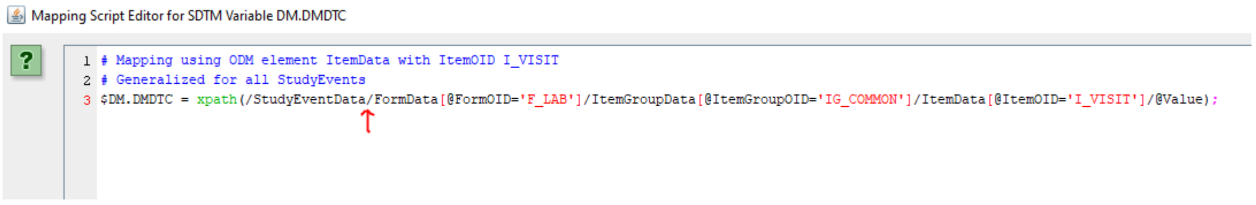
- "A sequence of more than one item is not allowed as the first argument of fn:string()"


```
Error Messages:

XSLT Transformer fatal error: A sequence of more than one item is not allowed as the first argument of fn:string() (@Value="2010-03-06
```

This is a typical error that often occurs and that users make desperate …

What does it mean?
It means that for a variable (in this case DMDTC), a list of variables has been retrieved, whereas a single value is expected. For DM, which has a structure of "one record per subject", one can of course only have one DM collection date, not a list of them.

So, for this case, let us have a look at our mapping script:



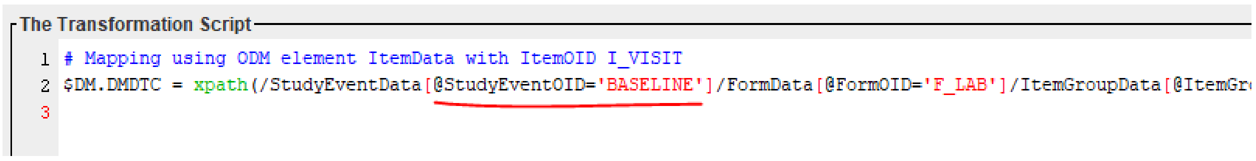
Mapping Script Editor for SDTM Variable DM.DMDTC

```
# Mapping using ODM element ItemData with ItemOID I_VISIT
# Generalized for all StudyEvents
$DM.DMDTC = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_COMMON']/ItemData[@ItemOID='I_VISIT']/@Value);
```

We see that in the selection path, there is no condition for "StudyEventData" (selection conditions are always in square brackets), so that the script will retrieve <u>all</u> values of "I_VISIT" (which represents the visit date) from the lab form (that is however used in all or at least several visits). Maybe in the case, the date from the lab form was used because there was nothing better …

What can one do in such a case?

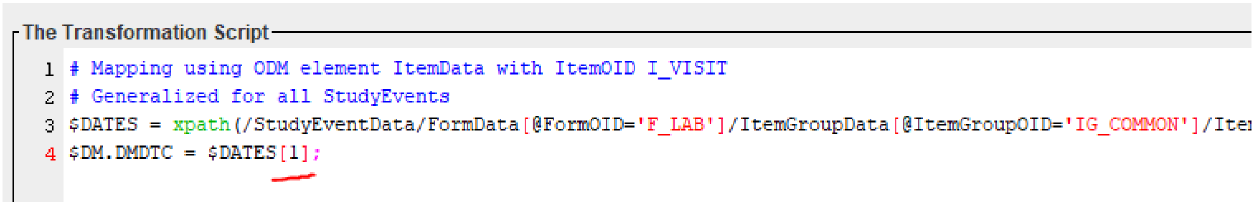
* take care that only the visit in which the demographics was collected, which can be taken care of by <u>not</u> using the button "Generalize for all StudyEvents". This would e.g. lead to:



The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_VISIT
$DM.DMDTC = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGr
```

* take the first date that is in the list. E.g.:

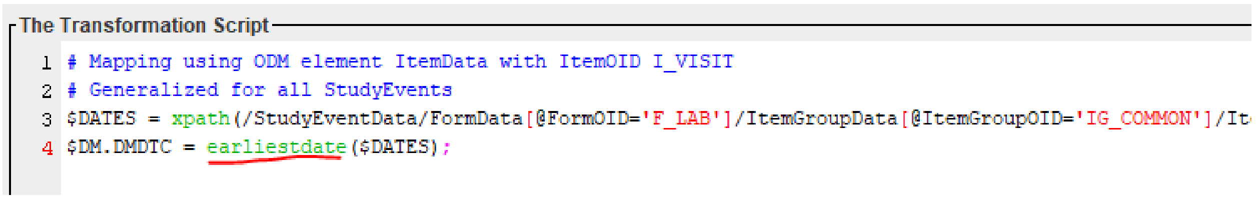


The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_VISIT
# Generalized for all StudyEvents
$DATES = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_COMMON']/Ite
$DM.DMDTC = $DATES[1];
```

This assumes however that in the source (the ODM) the visits are in chronological order

* select the earliest date from the list. E.g.:



The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_VISIT
# Generalized for all StudyEvents
$DATES = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_COMMON']/It
$DM.DMDTC = earliestdate($DATES);
```

Using the function "earliestdate". This however assumes that all the dates as retrieved from the ODM source is in ISO-8601 format.

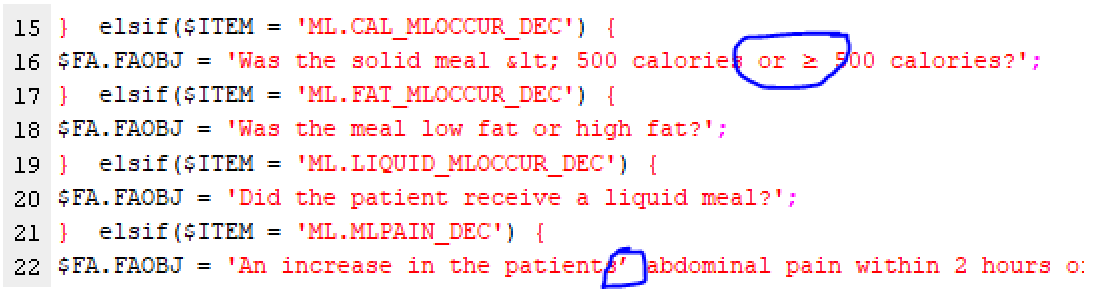
- "Invalid byte 1 of 1-byte UTF-8 sequence"

Java works with Unicode / UTF-8 encoding, SAS Transport 5 (XPT format) with US-ASCII (a subset of UTF-8 encoding). These are rather different from the (mostly default) Microsoft "Codepoint 1255" or "Windows 1255" encoding, which is incompatible …

We have seen this error occurring when users have either:

a) a source XML file that is not completely UTF-8 encoded
This can occur when the extraction from the database was not done in UTF-8 encoding, or when the XML was generated from Excel file without using the option "Tools - Web options - Encoding - Unicode (UTF-8)". See e.g. https://www.youtube.com/watch?v=jdCEXU-9GHE.
We have also seen this when users copied text from a Word or Excel document (which uses "Windows 1255" into a correctly encoded ODM-XML file or into a correctly CSV export from Excel (or anything else) to generate the ODM-XML file. In such cases, the result is an XML file with a mixture of encodings, which, as can be expected, causes problems.

b) an SDTM-ETL script in which text from e.g. a Word-document (typically using "Windows 1255" encoding) was copy-pasted. For example:



```
} elsif($ITEM = 'ML.CAL_MLOCCUR_DEC') {
$FA.FAOBJ = 'Was the solid meal &lt; 500 calories or ≥ 500 calories?';
} elsif($ITEM = 'ML.FAT_MLOCCUR_DEC') {
$FA.FAOBJ = 'Was the meal low fat or high fat?';
} elsif($ITEM = 'ML.LIQUID_MLOCCUR_DEC') {
$FA.FAOBJ = 'Did the patient receive a liquid meal?';
} elsif($ITEM = 'ML.MLPAIN_DEC') {
$FA.FAOBJ = 'An increase in the patients' abdominal pain within 2 hours o:
```

Where the characters "≥" and the skew quote "’" are "Windows 1255" characters that are not only not supported by US-ASCII (when generating XPT files), but also can cause problems during the transformation. In such a case, one can better use e.g. ">=" and the "straight" quote.

Essentially, one should always be extremely careful when using copy-paste from MS files in applications that either use UTF-8 encoding or US-ASCII.

Further bad practices

- Sorting keys in the define.xml for variables that require post-processing

When using the checkbox "Resort records using define.xml keys", one should take care to not have assigned a variable as a key for which the value is calculated or generated in a post-processing step.
This always applies to --SEQ (Sequence Number) as this is a "surrogate key", and is assigned in the very last step of the process, after any resorting: one should not try to sort on something that is not there yet.
The same however applies to --LOBXFL (Last Observation before First Exposure Flag), and especially for VISITNUM when also the checkbox "Perform post-processing unscheduled VISITNUM". Also here, sorting is first done to ensure that the data comes in the right order (as well

for SV as for any other "Findings" dataset), at which time there will not be a VISITNUM for the unscheduled visits. We found that having VISITNUM as a key for sorting, and there are unscheduled visits for which VISITNUM needs to be generated, can lead to serious side effects such as duplicate records in as well SV as in the other "Findings" datasets for which there are unscheduled visits.
Usually in such cases, the keys can be limited to STUDYID, USUBJID, --DTC, and in some cases, --TESTCD and/or --CAT.

Essentially, when one wants to use the "resorting using define.xml" keys, one should try to limit the number of keys anyway, also for the virtue of processing time. We have seen cases where people assigned up to 10 keys for a dataset definition in SDTM-ETL, including for variables for which the value is always empty (leading to an empty column). This of course doesn't make sense.