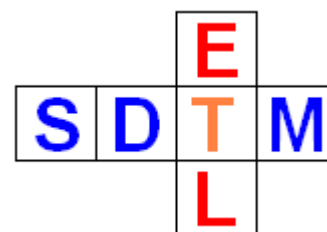


SDTM-ETL 5.0/5.1 User Manual and Tutorial

Author: Jozef Aerts, XML4Pharma

Last update: 2025-12-28



Troubleshooting: Most common mistakes and errors

Table of Contents

Table of Contents	1
Introduction.....	1
Most common error messages during "compilation" of the mapping scripts	2
- "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	4
- "Typed" versus "Untyped" ODM clinical data.	4
- "Variable xxx has not been declared (or its declaration is not in scope)"	5
- "No mapping provided for a 'looping' variable"	7
- "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
- Crash during XSLT Transformation due to invalid characters	12
- XSLT Transformer fatal error: Cannot convert string "" to double	15
Other common issues	16
- VISITNUM column is empty for "Findings" dataset	16
Further bad practices.....	19
- Sorting keys in the define.xml for variables that require post-processing.....	19

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.

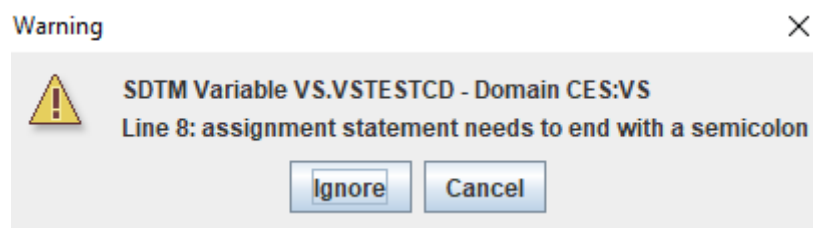
Most common error messages during "compilation" of the mapping scripts

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"



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:

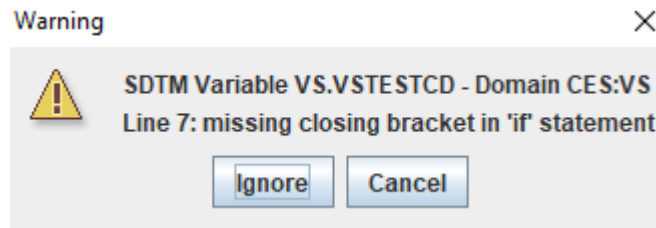
```
The Transformation Script
1 # Mapping using ODM element ItemData with ItemOID I_DIABP - value from attribute I
2 # Generalized for all StudyEvents
3 # Generalized for all Items within the ItemGroup
4 # Mapping for ODM Items [I_WEIGHT, I_SYSBP, I_DIABP] to SDTM CodeList VS.VSTESTCD
5 # with CodeList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

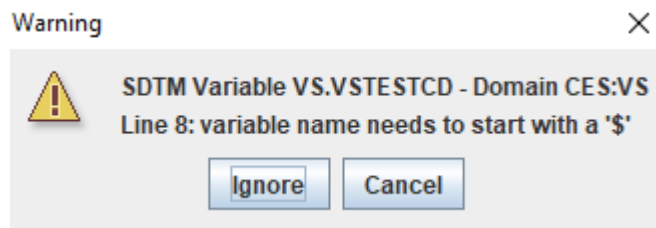
Is another obvious error, e.g.



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   $NEWCODEDVALUE = 'WEIGHT';
9 } elseif ($CODEDVALUE == 'I_SYSBP') {
10  $NEWCODEDVALUE = 'SYSBP';
```

- "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.V
5 # with CodeList OID 'CL.C66741.VSTESTCD.SUBSET'
6 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_WEEK_1_2']/ItemGr
7 if ($CODEDVALUE == 'I_WEIGHT') {
8   NEWCODEDVALUE = 'WEIGHT';
9 } elseif ($CODEDVALUE == 'I_SYSBP') {
10  $NEWCODEDVALUE = 'SYSBP';
11 } elseif ($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 are two "flavors" of ODM "ClinicalData", i.e. "untyped" and "typed".

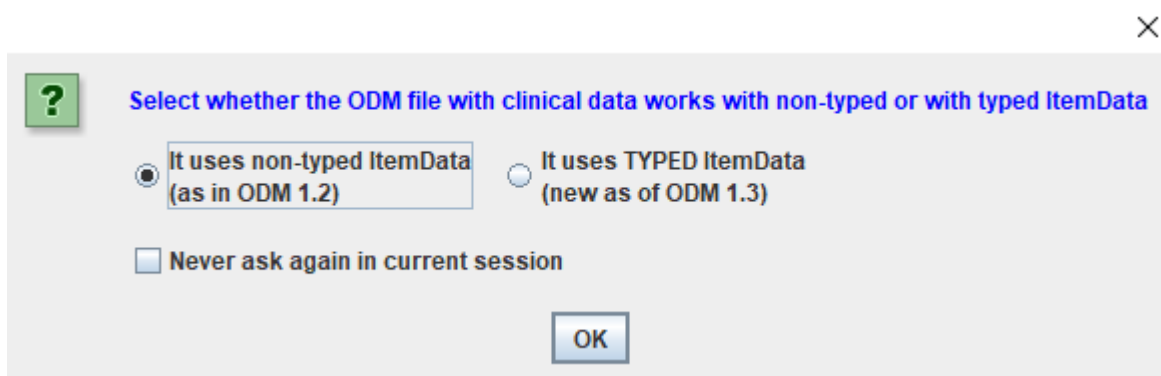
Most EDC systems export ODM clinical data in the "untyped" flavor, which is characterized by having the value of the data point in a "Value" attribute. For example:

```
<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:



The screenshot shows a dialog box with a question mark icon and a close button (X) in the top right corner. The title bar text is "Select whether the ODM file with clinical data works with non-typed or with typed ItemData". There are two radio button options: "It uses non-typed ItemData (as in ODM 1.2)" which is selected, and "It uses TYPED ItemData (new as of ODM 1.3)". Below these is a checkbox labeled "Never ask again in current session" which is currently unchecked. An "OK" button is located at the bottom center of the dialog.

One should then select the correct "flavor". If one is 100% sure and does not want that the system asks for it each time the mappings are executed, check the checkbox "Never ask again in current session".

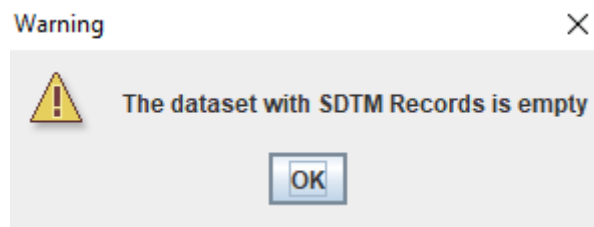
One can also preset the choice in the "properties.dat" file when one will always work with files from the same (type of) EDC system, e.g.

```
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
adefaultmappingdescriptions=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:

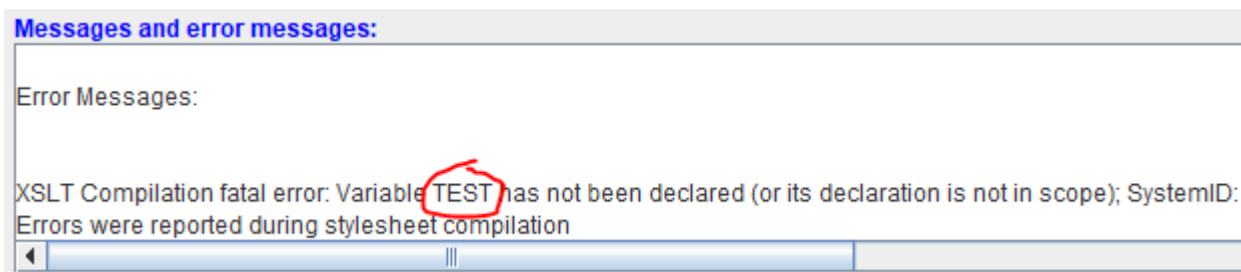


will appear.

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.:



where in the mapping script we find:

The Transformation Script

```
1 # Mapping using ODM element ItemData with ItemOID I_DIABP - value from attr
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.VS
5 # with CodeList OID 'CL.C66741.VSTESTCD.SUBSET'
6 $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_WEEK_1_2']/ItemGro
7 if ($TEST == 'I_WEIGHT') {
8     $NEWCODEDVALUE = 'WEIGHT';
9 } elseif ($CODEDVALUE == 'I_SYSBP') {
10    $NEWCODEDVALUE = 'SYSBP';
11 } elseif ($CODEDVALUE == 'I_DIABP') {
```

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

Which is less obvious as the script is:

The Transformation Script

```
8     $NEWCODEDVALUE = 'WEIGHT';
9 } elseif ($CODEDVALUE == 'I_SYSBP') {
10    $NEWCODEDVALUE = 'SYSBP';
11 } elseif ($CODEDVALUE == 'I_DIABP') {
12    $NEWCODEDVALUE = 'DIABP';
13 } elseif ($CODEDVALUE == '') {
14    $NEWCODEDVALUE = '';
15 } else {
16    $NEWCODEDVALUE = 'NULL';
17 }
18 $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:

$\$VS.VSTESTCD = \$NEWCODEDVALUE;$

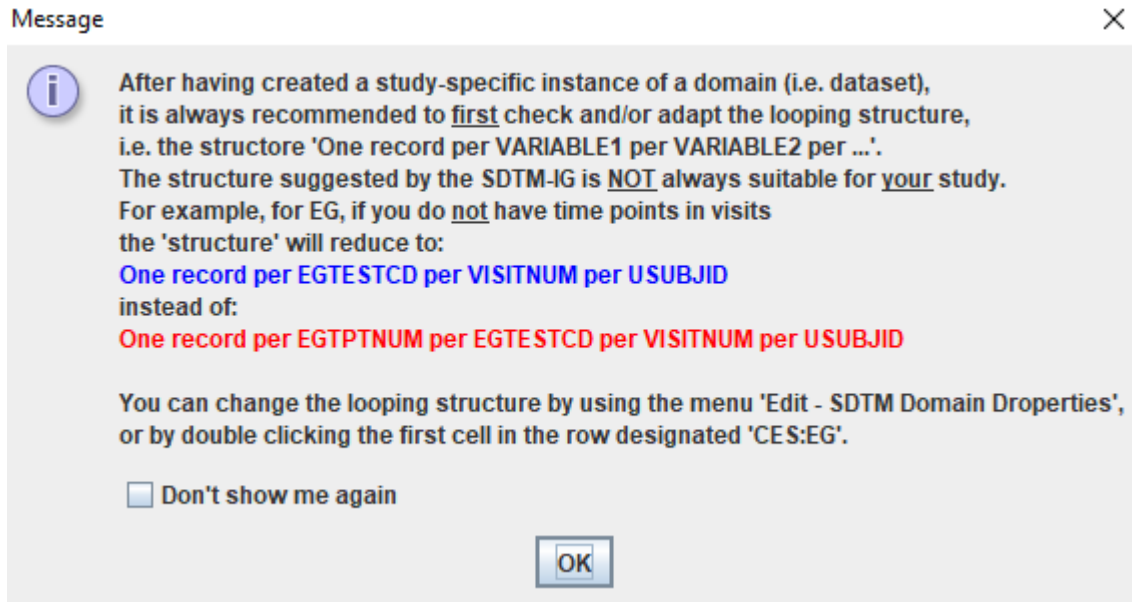
P.S.: there is no need that variable names are all uppercase, except for the SDTM/SEND variable itself, it is just a coding style ...

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.:



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:
 External document for 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 Comment

def:ArchiveLocationID : Location.EG
 def:Class : FINDINGS
 Key Sequence : 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

Validate

OK Cancel

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

Validate 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

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

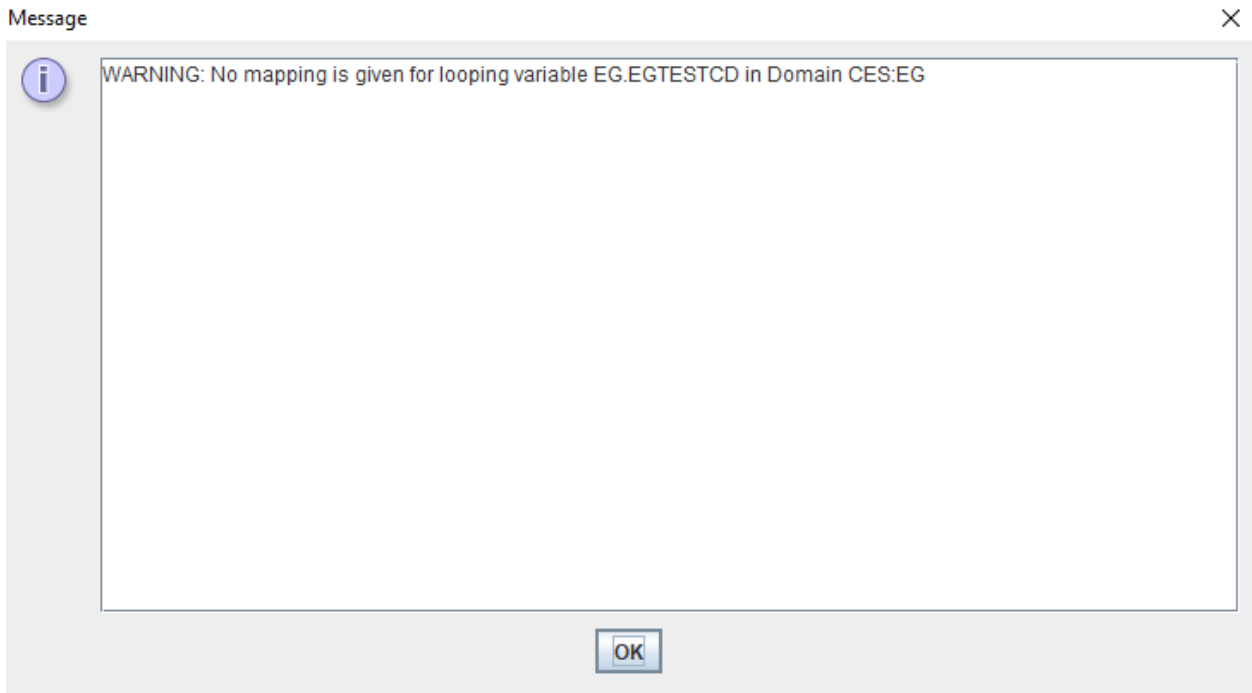
In such a case, it is important that the chosen "structure" is logical and hierarchical correct. For example it doesn't make sense to have EG.EGTESTCD as the first level, and USUBJID as the second.

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

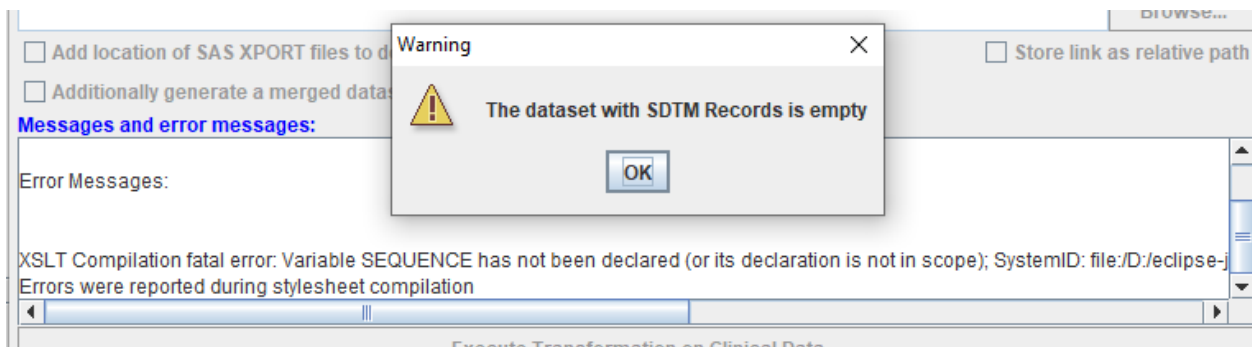
EG.EGSPID	EG.EGTESTCD	EG.EGTEST	EG.VISITNUM	EG.VISIT	EG.VISITDY
-----------	-------------	-----------	-------------	----------	------------

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 not yet for EG.EGTESTCD. When one then executes the mappings, the following warning will 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".



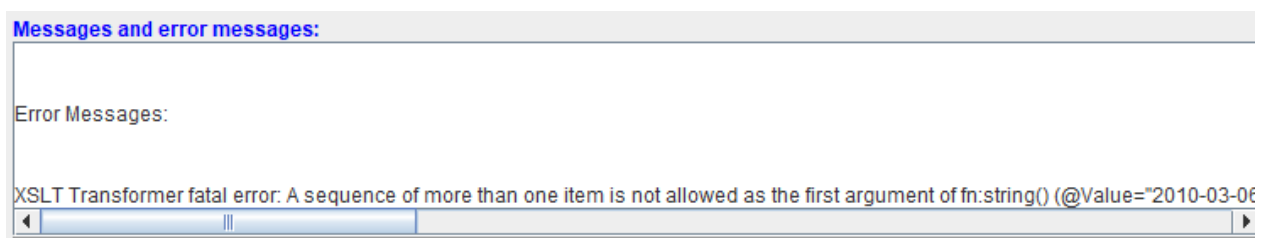
Some users however choose to ignore this warning and do continue (which is possible), leading to e.g.:



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()"



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
1 # Mapping using ODM element ItemData with ItemOID I_VISIT
2 # Generalized for all StudyEvents
3 $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 all 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 not using the button "Generalize for all StudyEvents". This would e.g. lead to:

```
The Transformation Script
1 # Mapping using ODM element ItemData with ItemOID I_VISIT
2 $DM.DMDTC = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGr
3
```

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

```
The Transformation Script
1 # Mapping using ODM element ItemData with ItemOID I_VISIT
2 # Generalized for all StudyEvents
3 $DATES = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_COMMON']/Item
4 $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
1 # Mapping using ODM element ItemData with ItemOID I_VISIT
2 # Generalized for all StudyEvents
3 $DATES = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_COMMON']/It
4 $DM.DMDTC = earliestdate($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:

```
15 } elseif($ITEM = 'ML.CAL_MLOCCUR_DEC') {
16 $FA.FAOBJ = 'Was the solid meal < 500 calories or ≥ 500 calories?';
17 } elseif($ITEM = 'ML.FAT_MLOCCUR_DEC') {
18 $FA.FAOBJ = 'Was the meal low fat or high fat?';
19 } elseif($ITEM = 'ML.LIQUID_MLOCCUR_DEC') {
20 $FA.FAOBJ = 'Did the patient receive a liquid meal?';
21 } elseif($ITEM = 'ML.MLPAIN_DEC') {
22 $FA.FAOBJ = 'An increase in the patient's 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.

- Crash during XSLT Transformation due to invalid characters

SDTM-ETL uses a lot of XSLT transformations as these are highly efficient and effective, and the transformation XSLT can be generated by SDTM-ETL "on the fly". XSLT however supposes that [Unicode](#) is used for the character representation².

We have observed that in some seldom cases, SDTM-ETL crashes during XSLT transformation without warning, when the source data comes from systems like Excel (or even Word) that was then transformed to ODM e.g. using the "[ODMGenerator](#)" software. We haven't observed this at all when the ODM came from an EDC system.

For example, in the console, we see:

² Remark that ASCII is a subset of Unicode, so if your data is ASCII-coded, you are still fine.

```

ERROR IOException in class XSLTCreatorExecutor during XSLT transformation of clinical data
DEBUG After running SDTMRecordsSAXContentHandler: XML output file = D:\SDTM-ETL_5_0_1\temp\temp_2025_4_5_8-6-13.xml
DEBUG doPostIOBXMLGeneration = true
DEBUG Will now rename file = D:\SDTM-ETL_5_0_1\temp\temp_2025_4_5_8-6-13.xml - size = 401408 to = temp\temp_2025_4_5_8-6-13_intermediate.xml
ERROR Postprocessing for --LOBXFL: Could not move file temp_2025_4_5_8-6-13.xml to temp\temp_2025_4_5_8-6-13_intermediate.xml

```

which is indeed not very helpful on first sight.

In such cases, it is a good idea to have a look at the last generated file in the "temp" directory of where the software is installed, and sort the files by creation date/time. In our case:

SDTM-ETL_5_0_1 > temp

Name	Änderungsdatum	Typ	Größe
tempdefine.xml	05.04.2025 08:04	XML-Datei	9,583 KB
testdefine.xml	05.04.2025 08:06	XSL-Stylesheet	696 KB
temp_resort.xml	05.04.2025 08:08	XSL-Stylesheet	48 KB
temp_resort_corrected.xml	05.04.2025 08:08	XSL-Stylesheet	48 KB
tempSDTM.xml	05.04.2025 08:08	XML-Datei	5,397 KB
tempSDTM_resorted.xml	05.04.2025 08:08	XML-Datei	5,120 KB
tempVisitNumUnscheduled.xml	05.04.2025 08:08	XML-Datei	4,321 KB
temp_2025_4_5_8-6-13.xml	05.04.2025 08:08	XML-Datei	392 KB

where the temporary file "temp_2025_4_5_8-6-13.xml" was the last one generated, and for which the system has complained it cannot be moved to temp_2025_4_5_8-6-13_intermediate" (see console screenshot).

When we then open this file in e.g. NotePad++, or using an XML-editor, and scroll to the bottom, we find:

```

7976 <ItemData ItemOID="CM.CMENRTP" IsNull="Yes"/>
7977 <ItemData ItemOID="CM.CMENTPT" IsNull="Yes"/>
7978 </ItemGroupData>
7979 <ItemGroupData ItemGroupOID="[REDACTED]CM" TransactionType="Insert">
7980 <ItemData ItemOID="STUDYID" Value="[REDACTED]"/>
7981 <ItemData ItemOID="DOMAIN" Value="CM"/>
7982 <ItemData ItemOID="USUBJID" Value="207-020-001"/>
7983 <ItemData ItemOID="CM.CMSEQ" Value="51"/>
7984 <ItemData ItemOID="CM.CMTRT" Value="clonidine"/>
7985 <ItemData ItemOID="CM.CMDECOD" IsNull="Yes"/>
7986 <ItemData ItemOID="CM.CMINDC" Value="Medical History"/>
7987 <ItemData ItemOID="CM.CMCLAS" IsNull="Yes"/>
7988 <ItemData ItemOID="CM.CMCLASCD" IsNull="Yes"/>
7989 <ItemData ItemOID="CM.CMDOSE" Value="1"/>
7990 <ItemData ItemOID="CM.CMDOSTXT" IsNull="Yes"/>
7991 <ItemData ItemOID="CM.CMDOSU" Value="OTHER"/>
7992 <ItemData ItemOID="CM.CMDOSFRQ" Value="ONCE"/>
7993 <ItemData ItemOID="CM.CMROUTE" Value="TRANSDERMAL"/>
7994 <ItemData ItemOID="CM.CMSTDTC" Value="2025-02-21"/>
7995 <ItemData ItemOID="CM.CMENDTC" Value="2025-02-21"/>
7996 <ItemData ItemOID="CM.CMSTDY" Value="4"/>
7997 <ItemData ItemOID="CM.CMENDY" Value="4"/>
7998 <ItemData ItemOID="CM.CMENRTP" IsNull="Yes"/>
7999 <Item

```

and see that it is incomplete. So it looks as the XSLT transformation crashed during generating data for the CM domain. At least this then narrows the problem down. It also looks as the issue occurred with the ODM data for subject 207-020-001, but this is not necessarily always the case ...

In this case, the issue only occurred with the option "automated assignment of VISITNUM for

unscheduled visits" on, so one could in principle run once without that feature, and then inspect the generated XPT file for "unexpected" characters.

At least, we know it looks as the error occurred with treating the CM data.

What can one then do?

The first thing one can do is to go back to the source, in this case an Excel file, and look for "non-Unicode" characters. Often these are "[typical Microsoft characters](#)" which are not covered by Unicode, or have another meaning in Unicode.

The second may be to use a "hex editor" such as [Neo](#), and then try to find something that is unusual. This however may be "looking for the needle in the haystack"!

In our case, we found:

00000280	54 22	20 56	61 6c	75 65	3d 22	32 30	32 35	2d 30	T" Value="2025-0
00000290	33 2d	30 36	54 31	36 3a	35 36	3a 35	34 22	2f 3e	3-06T16:56:54"/>
000002a0	0a 20	20 20	20 3c	49 74	65 6d	44 61	74 61	20 49	. <ItemData I
000002b0	74 65	6d 4f	49 44	3d 22	43 4d	2e 43	4d 54	45 52	temOID="CM.CMTER
000002c0	4d 22	0a 20	20 20	20 20	20 20	20 56	61 6c	75 65	M". Value
000002d0	3d 22	6d 61	67 6e	65 73	69 75	6d 20	73 75	6c 66	="magnesium sulf
000002e0	61 74	65 20	32 20	67 72	61 6d	2f 35	30 20	6d 4c	ate [REDACTED]
000002f0	20 28	34 25	29 20	69 6e	20 53	74 65	72 69	6c 65	[REDACTED]
00000300	20 57	61 74	65 72	20 49	56 50	42 20	70 72	65 6d	[REDACTED]
00000310	69 78	20 32	20 67	20 c2	a0 22	2f 3e	0a 20	20 20	[REDACTED]
00000320	20 3c	49 74	65 6d	44 61	74 61	20 49	74 65	6d 4f	<ItemData ItemO

There are also websites that at least try to find such non-conformant "characters", such as <https://www.socisurvey.de/tools/view-chars.php>.

When we put the source XML for our source data there, and click "Show me the characters", we e.g. find:

```
....<ItemData·ItemOID="CM.PAGESEQ"·Value="75"/>CR LF
....<ItemData·ItemOID="CM.STATUSID"·Value="5"/>CR LF
....<ItemData·ItemOID="CM.STATUSID_DEC"·Value="Entered"/>CR LF
....<ItemData·ItemOID="CM.PAGELMDT"·Value="2025-03-06T16:56:54"/>CR LF
....<ItemData·ItemOID="CM.DATALMDT"·Value="2025-03-06T16:56:54"/>CR LF
....<ItemData·ItemOID="CM.CMTERM"·Value="magnesium·sulfate·2·gram/50·mL·(4%)·in·Sterile·Water·IVPB·premix·2·g·U+A0"/>CR LF
....<ItemData·ItemOID="CM.CMREAS"·Value="PRO"/>CR LF
....<ItemData·ItemOID="CM.CMREAS_DEC"·Value="Prophylaxis"/>CR LF
....<ItemData·ItemOID="CM.CMDOSE"·Value="2"/>CR LF
....<ItemData·ItemOID="CM.CMUNIT"·Value="GM"/>CR LF
....<ItemData·ItemOID="CM.CMUNIT_DEC"·Value="Gram"/>CR LF
```

and the non-conformant character is displayed. In this case it looks to be the "no-break space" which is a "Latin-1 extension"

Anyway, such issues are nasty, and not easy to resolve³.

Of course, one could say that sponsors and CROs should take care that the the investigators only use modern tools that are compatible with international (and non-propriety) standards for characters, but this can be hard to enforce. Essentially, tools that are based on "Latin-1" should be avoided.

What one can do to reduce the risk, is when exporting data from Excel into CSV, take care that the export is done using Unicode (UTF-8). The way this is done in Excel may however be dependent on the version of Office, the language settings, and some other. Welcome in the jungle!

³ Remark that computers do not know what a "character" is: the only thing they know are bits and bytes. It is standards like Unicode that makes them to characters.

- XSLT Transformer fatal error: Cannot convert string "" to double

This error currently only occurs when the ODM clinical data is of type "Typed ItemData" which is used by a few EDC systems, and when the absence of a data point is given by the "IsNull" attribute. For example:

```
<ItemDataAny ItemOID="IT.sex" IsNull="Yes"></ItemDataAny>
```

and when the value for the mapped variable (here "Sex") is coded in the ODM as an integer. When then using the "mapping wizard", the mapping code then looks like e.g.;

```
$CODEDVALUE = xpath(...../ItemData[@ItemOID='IT.sex']/@Value);
# Sex is coded as an integer
if ($CODEDVALUE == 1) {
    $NEWCODEDVALUE = 'M';
} elseif ($CODEDVALUE == 2) {
    $NEWCODEDVALUE = 'F';
} elseif ($CODEDVALUE == 99) {
    $NEWCODEDVALUE = 'U';
} elseif ($CODEDVALUE == 3) {
    $NEWCODEDVALUE = 'INTERSEX';
} else {
    $NEWCODEDVALUE = "";
}
$DM.SEX = $NEWCODEDVALUE;
```

When then executed, leading to the XSLT error:

Cannot convert string "" to double

This can currently be solved by putting the values representing the code for "sex" (1, 2, 99, 3) in single quotes, essentially making them strings, i.e.

```
$CODEDVALUE = xpath(...../ItemData[@ItemOID='IT.sex']/@Value);
# Sex is coded as an integer
if ($CODEDVALUE == '1') {
    $NEWCODEDVALUE = 'M';
} elseif ($CODEDVALUE == '2') {
    $NEWCODEDVALUE = 'F';
} elseif ($CODEDVALUE == '99') {
    $NEWCODEDVALUE = 'U';
} elseif ($CODEDVALUE == '3') {
    $NEWCODEDVALUE = 'INTERSEX';
} else {
    $NEWCODEDVALUE = "";
}
$DM.SEX = $NEWCODEDVALUE;
```

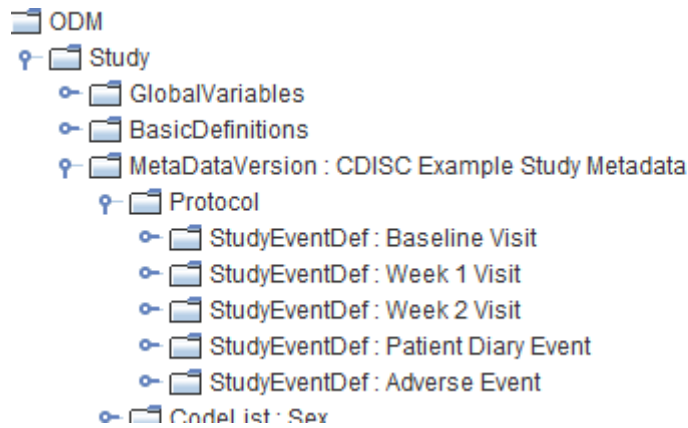
As said, this issue only occurs when "typed ItemData" is used and missing values are represented are provided using the "IsNull" attribute. The better practice in ODM is to have no "ItemData" at all for missing data points.

We are currently investigating how this can be handled in a better way in the software and will correct this in the new version.

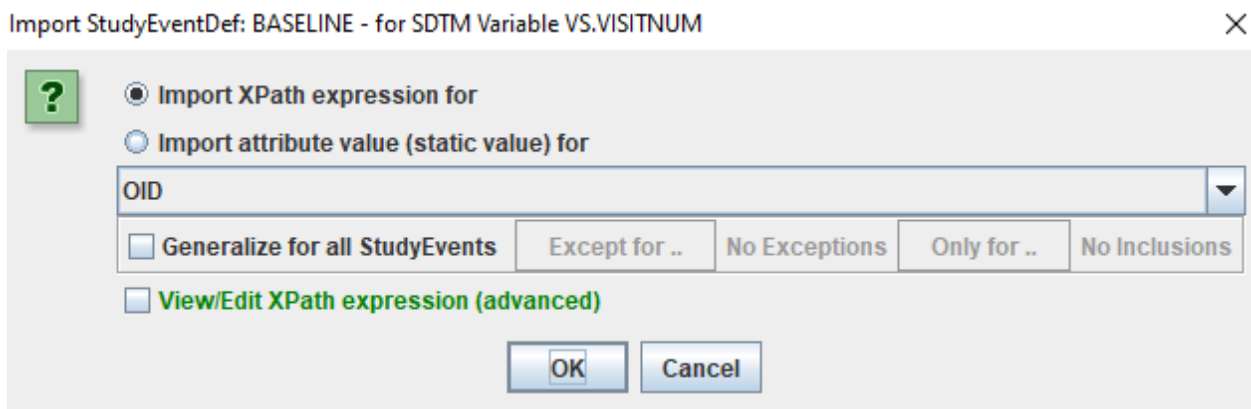
Other common issues

- VISITNUM column is empty for "Findings" dataset

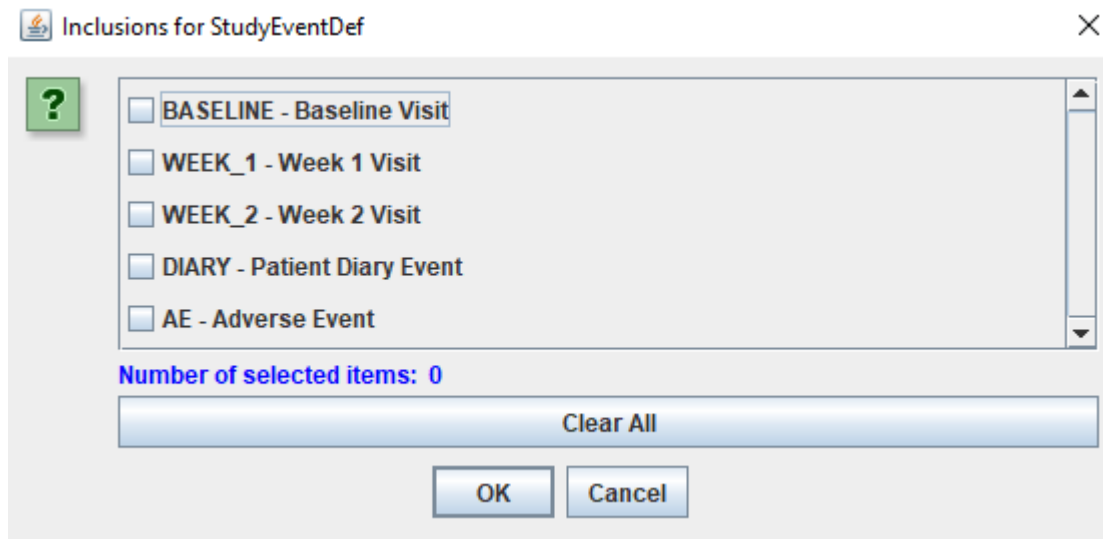
Very often, VISITNUM for "Findings" datasets will be generated from the "StudyEventDef" item in the ODM tree on the left. For example:



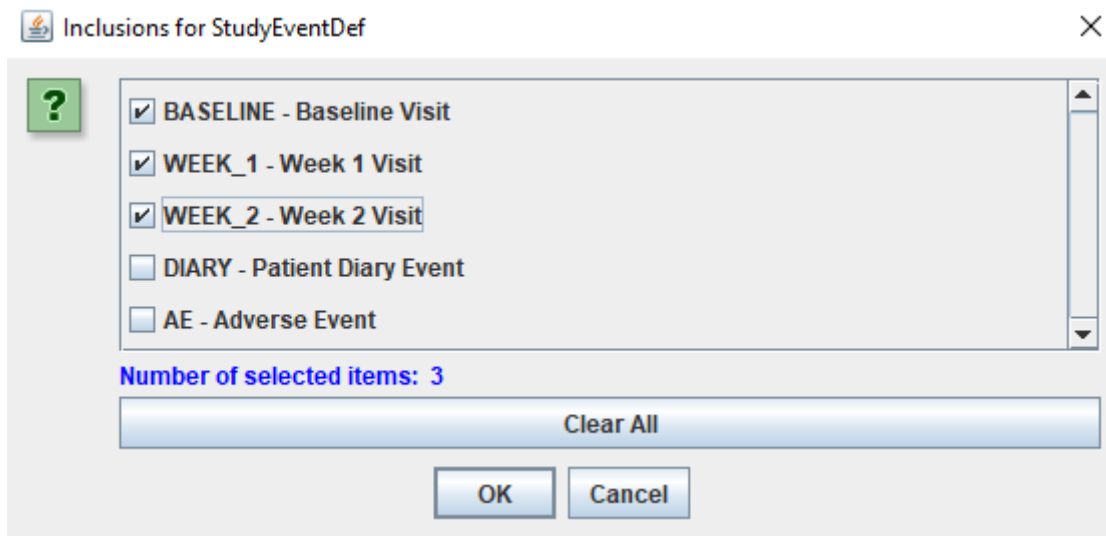
When then using drag-and-drop to the cell for VISITNUM, the following wizard is displayed:



If the type of measurements is done in each visit, one will typically check the checkbox "Generalize for all StudyEvents". When however, the type of measurements is only done in a single or limited set of visits (e.g. Vital Signs) one will additionally use "Only for ...". For example, when our vital signs measurements are only done in the "Baseline Visit" and the "Week 1 Visit" and in the "Week 2 Visit", additionally clicking "Only for ..." leads to:



and we select these 3 visits:



After "OK" twice, leading to the mapping script:

```

The Transformation Script
1 # Mapping using ODM element StudyEventData using value from attribute StudyEventOID
2 # Generalized for all StudyEvents
3 $VS.VISITNUM = xpath(/StudyEventData[@StudyEventOID='BASELINE' or @StudyEventOID='WEEK_1' or @StudyEventOID='WEEK_2']/@StudyEventOID/);
4

```

In some very seldom cases, especially when the selection for the "topic variable", which is the -TESTCD variable in the case of "Findings", this may lead to an empty VISITNUM column anyway. If this happens, one can always use the "alternative XPath" in the mapping script (which then need to be typed):

```

The Transformation Script
1 # Using the relative path up to "StudyEventData"
2 $VS.VISITNUM = xpath(../../../../../@StudyEventOID);
3

```

What happens here is that the engine will go three levels up (three time "../"), starting from the "ItemData" of the "looping/topic" variable, and then arriving at the "StudyEventData" level, and then taking the value of attribute "StudyEventOID".

However, one will not want the OID (identifier) to be taken for "VISITNUM" as the latter needs to be a number, so one will take a derivative of it, mostly using an "if-elseif-else" structure, like:

```
The Transformation Script
1 # Using the relative path up to "StudyEventData"
2 $TEMP = xpath../../../../@StudyEventOID);
3 if($TEMP = 'BASELINE') {
4     $VS.VISITNUM = 0;
5 } elseif($TEMP = 'WEEK_1') {
6     $VS.VISITNUM = 1;
7 } elseif($TEMP = 'WEEK_2') {
8     $VS.VISITNUM = 1;
9 } else {
10    $VS.VISITNUM = -999;
```

Assigning "-999" for the "else" case is not a bad idea, as it will immediately catch the eyes in the results when one has made an incorrect selection or assignment.

One can then use the same approach for the "VISIT" (visit name) variable, e.g.:

```
The Transformation Script
1 # Using the relative path up to "StudyEventData"
2 $TEMP = xpath../../../../@StudyEventOID);
3 if($TEMP == 'BASELINE') {
4     $VS.VISIT = 'Baseline Visit';
5 } elseif($TEMP == 'WEEK_1') {
6     $VS.VISIT = 'Week 1 Visit';
7 } elseif($TEMP == 'WEEK_2') {
8     $VS.VISIT = 'Week 2 Visit';
9 } else {
10    $VS.VISIT = 'TODO';
11 }
```

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.