SDTM-ETL 3.1 User Manual and Tutorial

Author: Jozef Aerts, XML4Pharma

Last update: 2014-07-14

CDISC Dataset-XML Generation

Sofar we have generated SDTM datasets using the option „Generate Transformation (XSLT) Code for SAS-XPT) although we did not create SAS datasets yet, and only generated the SDTM tables with the software itself.
In this part of the tutorial we will demonstrate the different output options which are:

- output of the SDTM datasets in the new CDISC Dataset-XML format
- output of the SDTM datasets in the classic SAS-XPORT (SAS Transport 5) format

The CDISC Dataset-XML format for electronic submissions has been developed to overcome all of the limitations of SAS Transport 5, such as:

- 8-character limitation for variable names and test codes
- 40-character limitation for variable labels
- 200-character limitation for variable values (forcing to split values over records)
- limited amount of publicly (and free) tools available
- not really vendor-neutral[1]

These limitations do not exist in Dataset-XML. Furthermore the new format has a number of additional advantages, such as:

- Perfect alignment with define.xml (both use the same base standard)
- Supplemental variables can remain in the parent domain, they just are flagged as being „non-standard“ in the define.xml
- It is not necessary anymore to have a COVAL, COVAL1, etc. variables in case there are comments of over 200 characters (which is often the case)

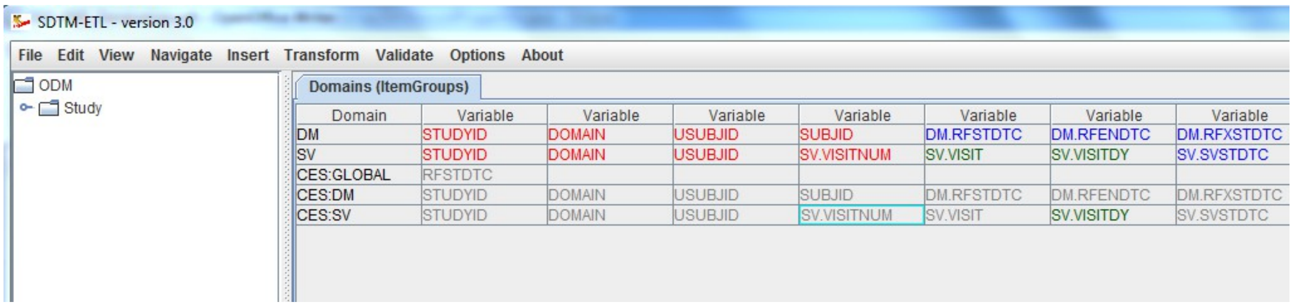
And there are some very interesting opportunities:

- incorporation of audit records on data points or records
- incorporation of data points from electronic health records (such as using HL7-v3 or HL7-FHIR as format), as Dataset-XML is extensible.

For further information, see the slides presented at the European CDISC Interchange 2014 (Paris).

If you do not have the DM and SV mappings loaded, do now, using „File – load Study define.xml“ (or use CTRL-D). After having done so, you can again filter the visible domains (see previous part of the tutorial „Creating mappings for the SV (Subject Visits) domain“) using „View – View/Hide Domains“. This e.g. results in:

1 Although the specification of the format has been published (the TS-140) document, it is extremely difficult implement in modern software that use freely available software languages (Java, C#, …). As such, new vendors start with a serious competition disadvantage.

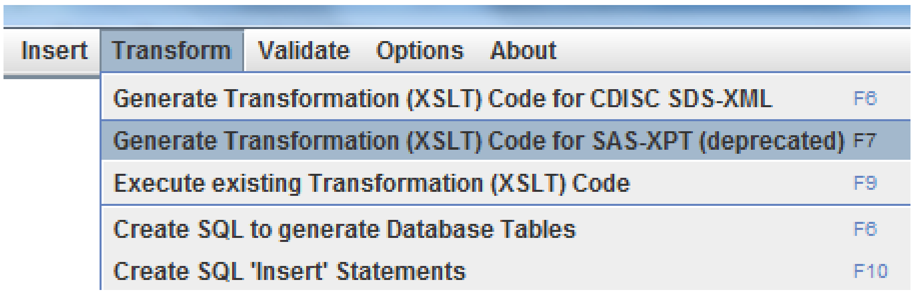

SDTM-ETL - version 3.0
File Edit View Navigate Insert Transform Validate Options About
ODM
Study
Domains (ItemGroups)

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

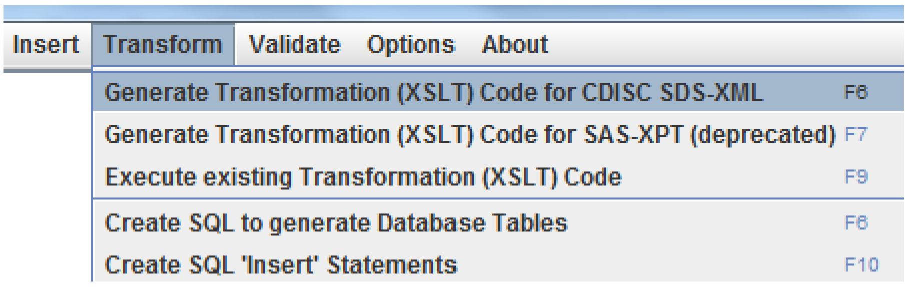
with the rows „CES:DM“ and „CES:SV“ containing the mappings for the DM and SV domain respectively, and the row „CES:GLOBAL“ containing mappings for global (reusable) variables such as RFSTDTC (Reference Start Date/Time)[2].

Variables that are displayed in grey do already contain a mapping. Variables in red are „required“, variables in blue „expected“ and variables in green „permissible“.

We will now again execute the mappings on our test clinical data. To do so, use the menu „Transform“. This results in:

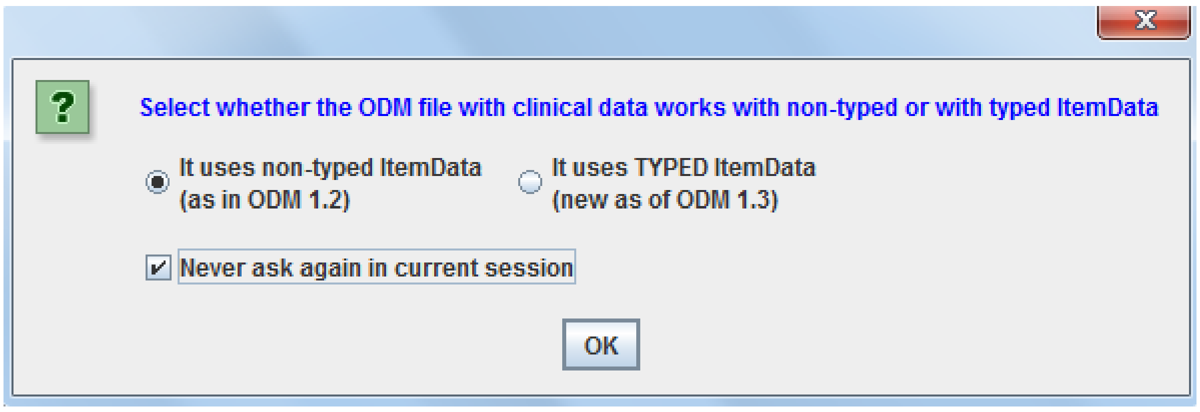

Insert Transform Validate Options About
Generate Transformation (XSLT) Code for CDISC SDS-XML F6
Generate Transformation (XSLT) Code for SAS-XPT (deprecated) F7
Execute existing Transformation (XSLT) Code F9
Create SQL to generate Database Tables F8
Create SQL 'Insert' Statements F10

We have the choice between generating the transformation code for generation of SAS Transport 5 (SAS-XPT) datasets, or for the new (modern) CDISC SDS-XML formatted datasets. We choose the latter option (SDS-XML):


Insert Transform Validate Options About
Generate Transformation (XSLT) Code for CDISC SDS-XML F6
Generate Transformation (XSLT) Code for SAS-XPT (deprecated) F7
Execute existing Transformation (XSLT) Code F9
Create SQL to generate Database Tables F8
Create SQL 'Insert' Statements F10

This results in the following dialog:

2 Later we will also add RFENDTC (Reference End Date(Time)

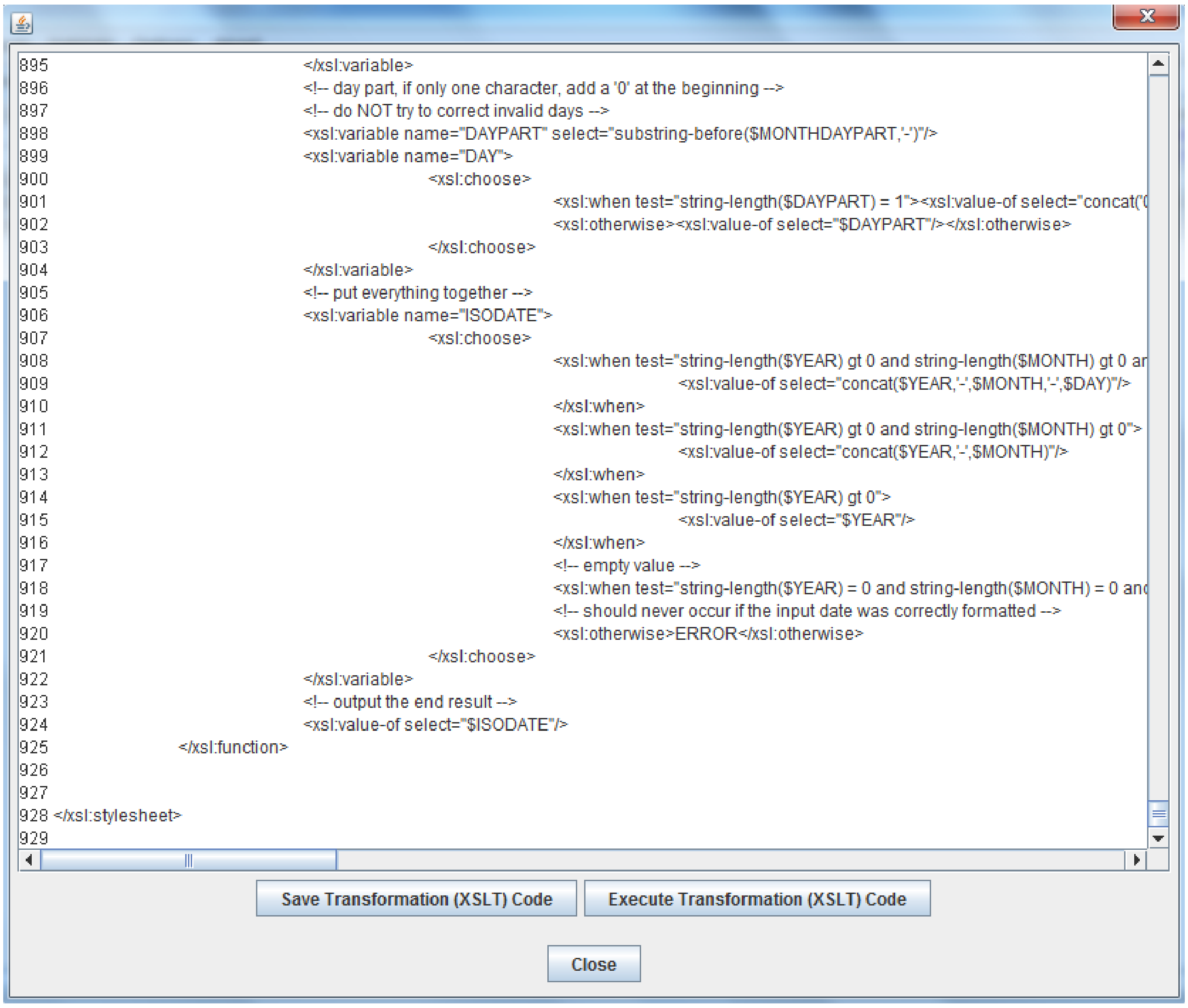


Select whether the ODM file with clinical data works with non-typed or with typed ItemData

[● 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]

We are asked whether our clinical data are „untyped“ (classic, using „ItemData“ with a „Value“ attribute) or „typed“ (i.e. ising e.g. ItemDataDateTime). If you are unsure, please ask your EDC vendor or just open an ODM file with clinical data – you will immediately find out.
In most cases, you will avoid that this question pops up again, so check the checkbutton „Never ask again in current session“. You are then asked to confirm your choice, and the following dialog is then displayed:



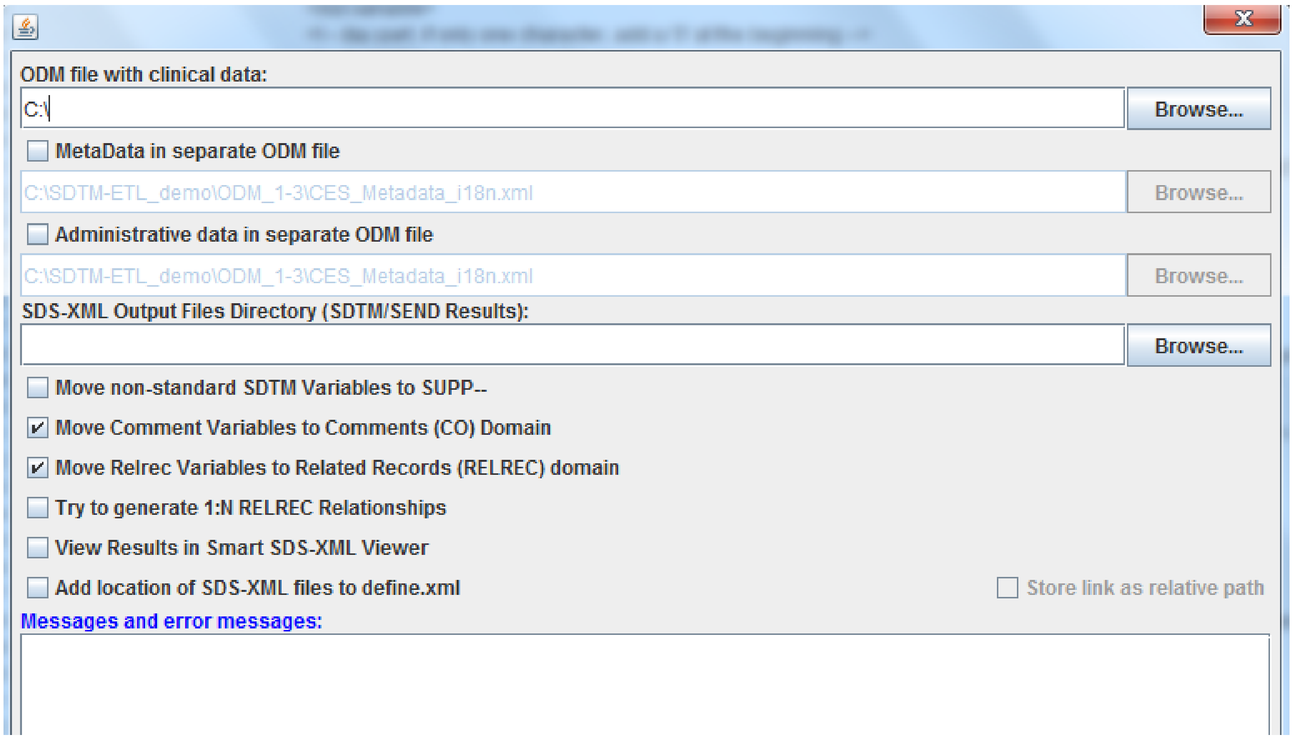
```
				</xsl:variable>
				<!-- day part, if only one character, add a '0' at the beginning -->
				<!-- do NOT try to correct invalid days -->
				<xsl:variable name="DAYPART" select="substring-before($MONTHDAYPART,'-')"/>
				<xsl:variable name="DAY">
					<xsl:choose>
						<xsl:when test="string-length($DAYPART) = 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
Close

The mapping code is displayed in XSLT (nice for debugging) and the dialog allows you to save the XSLT code for offline execution. You can however also immediately executing the mapping. To do so, click the button „Execute Transformation (XSLT) Code“. This results in a new dialog:


ODM file with clinical data:

C:\

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...

SDS-XML Output Files Directory (SDTM/SEND Results):

Browse...

☐ 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 Results in Smart SDS-XML Viewer

☐ Add location of SDS-XML files to define.xml ☐ Store link as relative path

Messages and error messages:

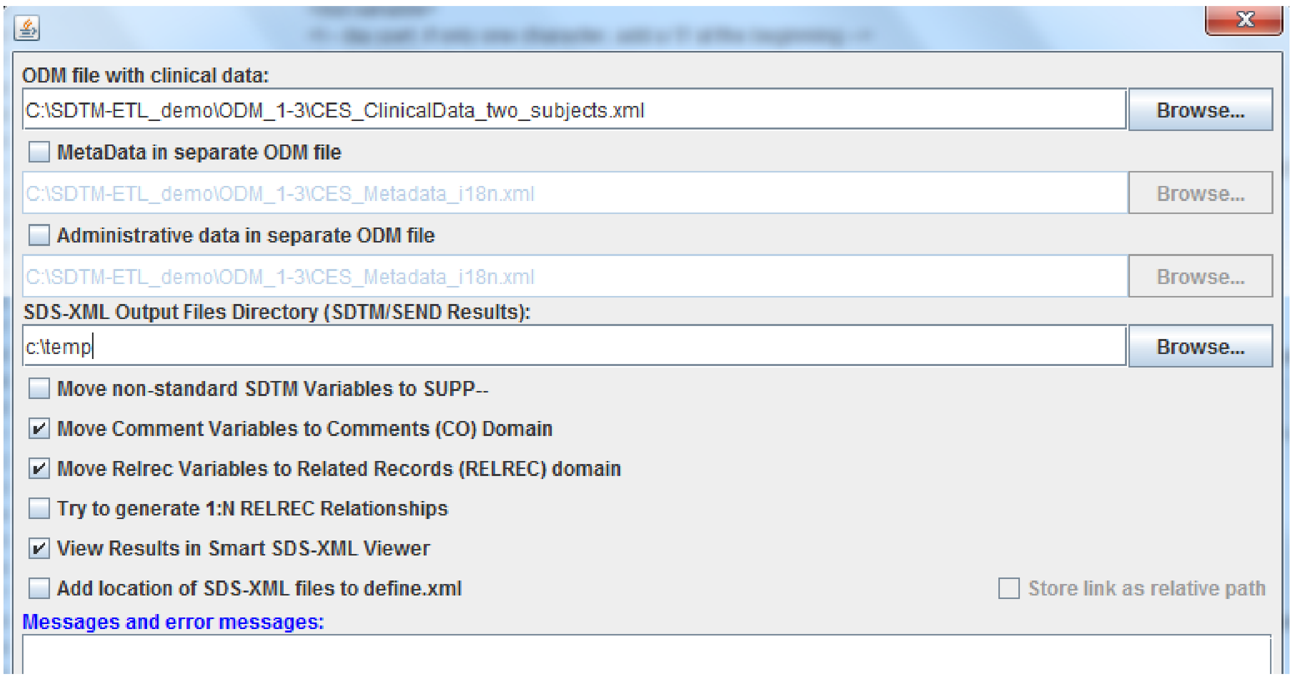
This dialog is very similar to the one obtained when wanting to use SAS-XPT as output, but with a few important differences:

- There is no checkbox anymore asking whether you would want to split values longer than 200 characters – in SDS-XML this is not necessary anymore
- The checkbox „View Result SDTM Tables“ has been replaced by a checkbox „View Results in Smart SDS-XML Viewer“. We will later see that this viewer is considerable „smarter“ than the SASViewer used for inspecting SAS-XPT files

The checkbox „Move non-standard SDTM variables to SUPP--“ is still available, but you are discouraged to check it, as keeping the „supplemental“ variables in the parent domain is exactly one of the advantages of SDS-XML[3].

At the moment, you are still encouraged to move „Comment“ variables into a separate „CO“ dataset, although also this might change in future.

3 It still has to be comfirmed by the FDA that they will accept submissions in the new format in the near future.


ODM file with clinical data:

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

SDS-XML Output Files Directory (SDTM/SEND Results):

c:\temp

☐ 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 Results in Smart SDS-XML Viewer

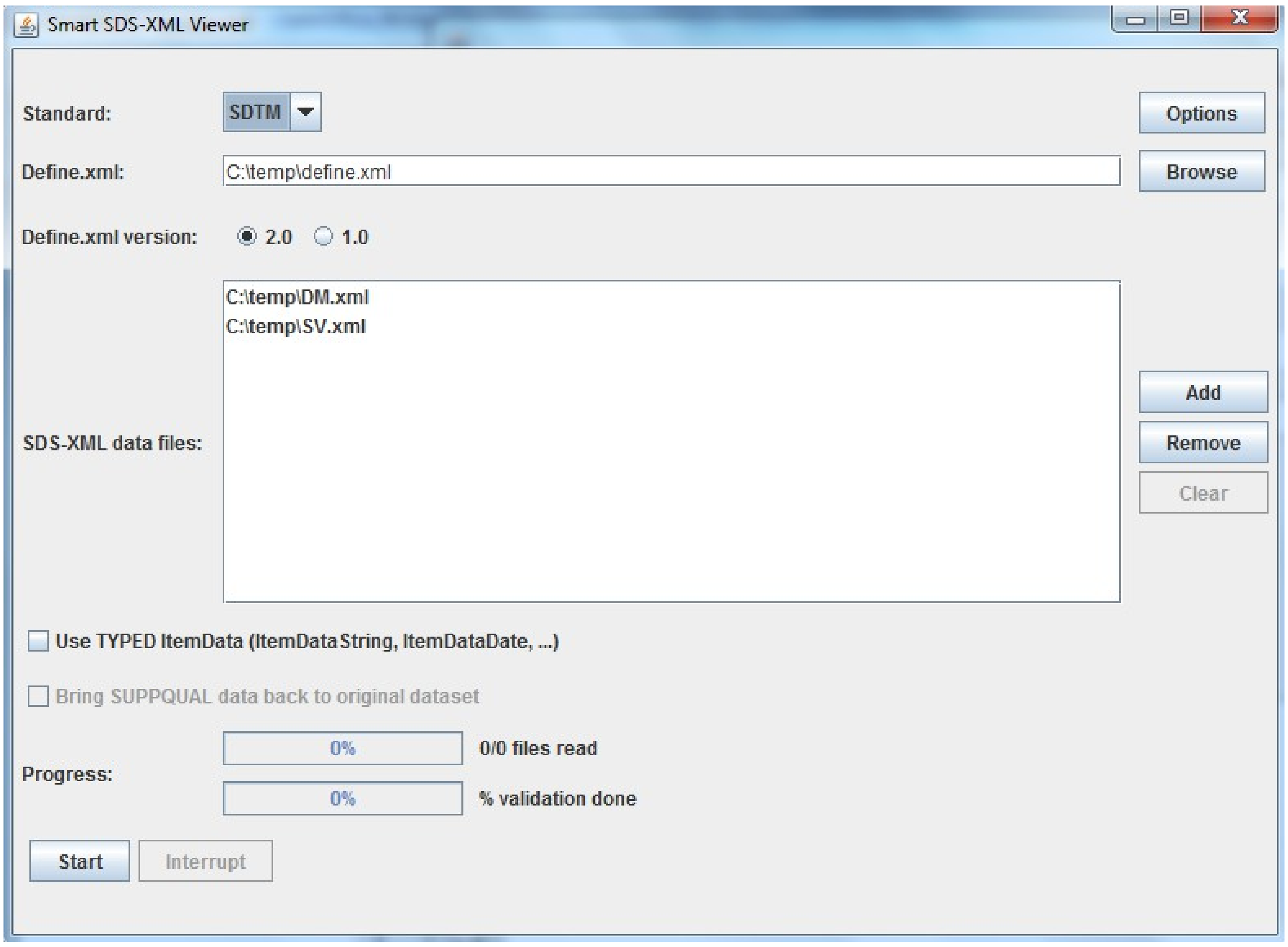
☐ Add location of SDS-XML files to define.xml

☐ Store link as relative path

Messages and error messages:

Now also check the checkbox „View results in Smart SDS-XML Viewer“:

I also already added the location of the file with ODM clinical data and of a directory where the SDS-XML files will be written to.
When now clicking the „Execute Transformation on Clinical Data“, the transformation executes and after a few seconds the „Smart SDS-XML Viewer“ pops up:


Smart SDS-XML Viewer

Standard: SDTM

Define.xml: C:\temp\define.xml

Define.xml version: [✓ 2.0] 1.0

SDS-XML data files:

C:\temp\DM.xml
C:\temp\SV.xml

☐ Use TYPED ItemData (ItemDataString, ItemDataDate, ...)

☐ Bring SUPPQUAL data back to original dataset

Progress: 0% 0/0 files read

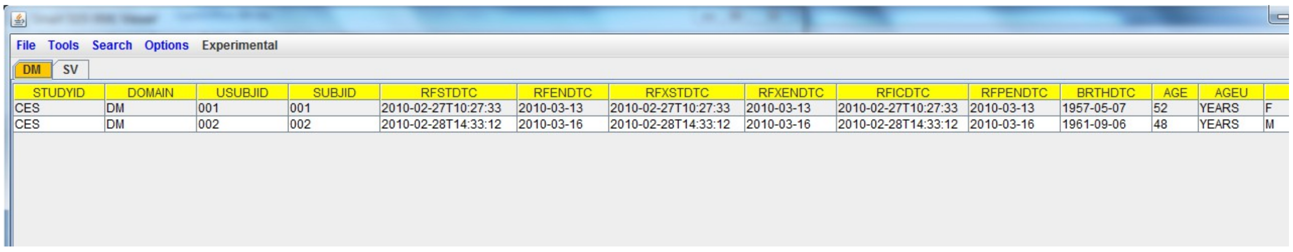
0% % validation done

A define.xml file has also be generated and written into the same directory as where the generated SDS-XML files reside.
You can now use all possible options of the „Smart SDS-XML Viewer". These are explained in a separate tutorial that you can download from the Open-Source „SourceForge" website:

http://sourceforge.net/projects/smart-sds-xml-viewer/

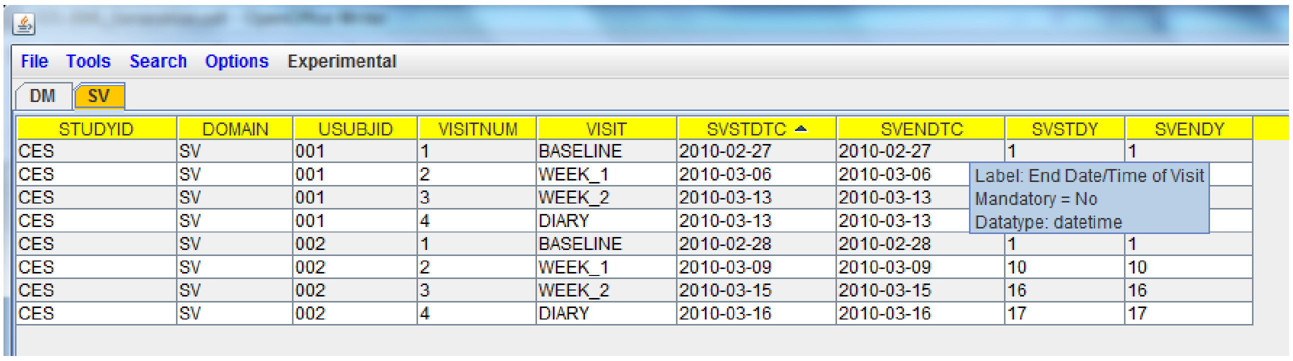
Also remark that the „Smart SDS-XML Viewer" is open source[4], so you can download the source code, extend it, change it, do anything you want to do with it. You can even replace the version that comes with SDTM-ETL with your own (e.g. extended) one, by replacing the files in the SDTM-ETL directory „Smart_SDS-XML_Viewer_software" (of course everything on your own responsibility!).

Starting the „Smart SDS-XML Viewer" in our case results in:



STUDYID	DOMAIN	USUBJID	SUBJID	RFSTDTC	RFENDTC	RFXSTDTC	RFXENDTC	RFICDTC	RFPENDTC	BRTHDTC	AGE	AGEU	
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	1957-05-07	52	YEARS	F
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	1961-09-06	48	YEARS	M

and for the SV dataset:



STUDYID	DOMAIN	USUBJID	VISITNUM	VISIT	SVSTDTC ▲	SVENDTC	SVSTDY	SVENDY
CES	SV	001	1	BASELINE	2010-02-27	2010-02-27	1	1
CES	SV	001	2	WEEK_1	2010-03-06	2010-03-06		
CES	SV	001	3	WEEK_2	2010-03-13	2010-03-13		
CES	SV	001	4	DIARY	2010-03-13	2010-03-13		
CES	SV	002	1	BASELINE	2010-02-28	2010-02-28	1	1
CES	SV	002	2	WEEK_1	2010-03-09	2010-03-09	10	10
CES	SV	002	3	WEEK_2	2010-03-15	2010-03-15	16	16
CES	SV	002	4	DIARY	2010-03-16	2010-03-16	17	17

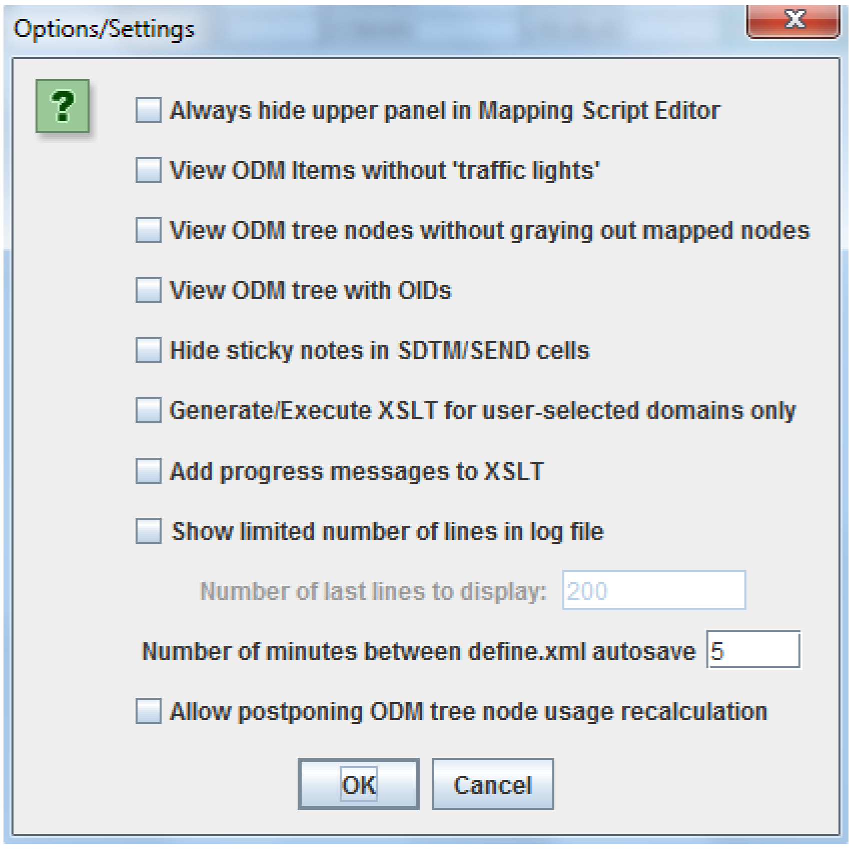
Label: End Date/Time of Visit
Mandatory = No
Datatype: datetime

Remark that unlike with the SASViewer, the „Smart SDS-XML Viewer" also displays the metadata (like the define.xml datatype) when hovering the mouse over the column header.
Much more is however possible with the „Smart SDS-XML Viewer" and you are invited to download the tutorial from the SourceForge site, but you can also look at a YouTuve demo video at:
https://www.youtube.com/watch?v=wRBktVtB47s&noredirect=1

Some options and settings for SDTM-ETL

There is a good number of options you can choose from when working with SDTM-ETL. You can change them using the menu „Options – Settings":

4 Once the CDISC specification is final, the source code will be made available through SourceForge.



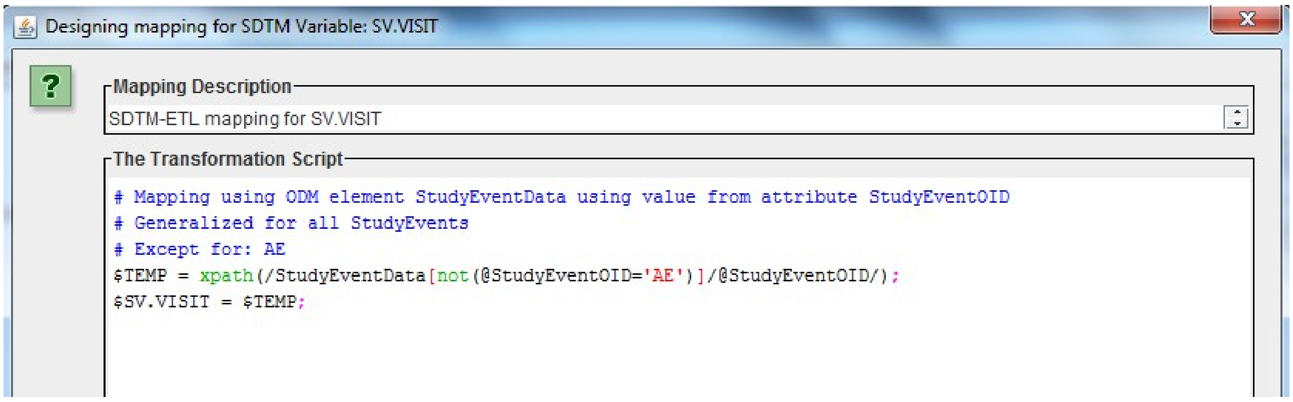
- [ ] Always hide upper panel in Mapping Script Editor
- [ ] View ODM Items without 'traffic lights'
- [ ] View ODM tree nodes without graying out mapped nodes
- [ ] View ODM tree with OIDs
- [ ] Hide sticky notes in SDTM/SEND cells
- [ ] Generate/Execute XSLT for user-selected domains only
- [ ] Add progress messages to XSLT
- [ ] Show limited number of lines in log file

Number of last lines to display: 200

Number of minutes between define.xml autosave 5

- [ ] Allow postponing ODM tree node usage recalculation

The number of options is steadily increased, so you might see a slightly different picture with your distribution.
The first option „Always hide upper panel in Mapping Script Editor“ always hides the upper panel in the mapping editor. So when it is checked, and editing a mapping, you will see:



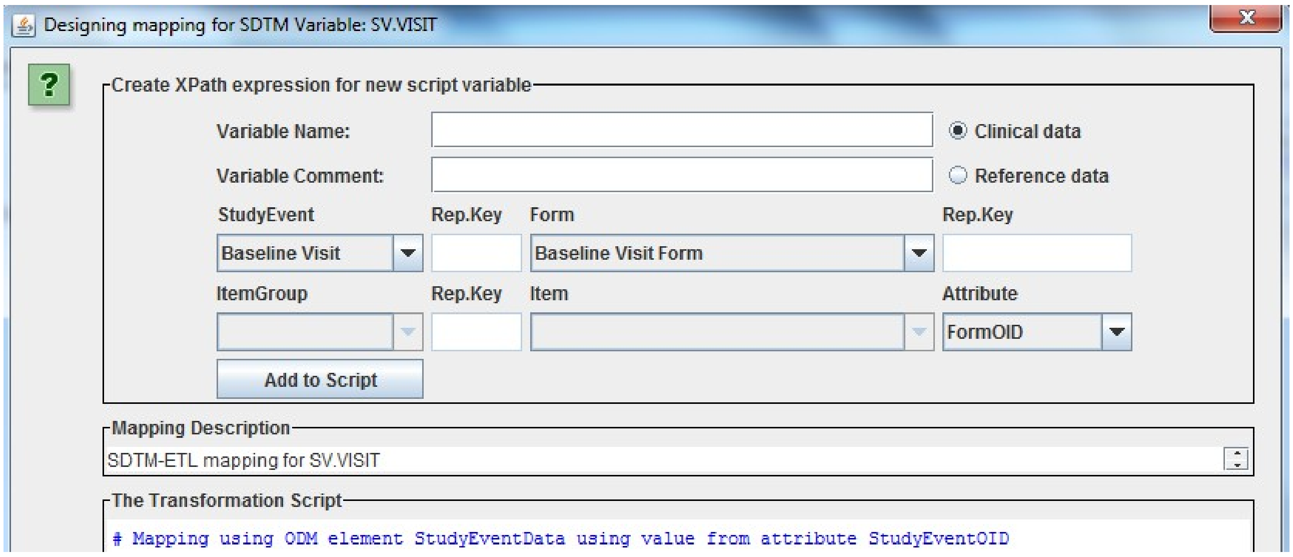
Mapping Description

SDTM-ETL mapping for SV.VISIT

The Transformation Script

```
# Mapping using ODM element StudyEventData using value from attribute StudyEventOID
# Generalized for all StudyEvents
# Except for: AE
$TEMP = xpath(/StudyEventData[not(@StudyEventOID='AE')]/@StudyEventOID/);
$SV.VISIT = $TEMP;
```

instead of:


Designing mapping for SDTM Variable: SV.VISIT

Create XPath expression for new script variable

Variable Name:		[✓ Clinical data]	
Variable Comment:		Reference data	
StudyEvent	Rep.Key	Form	Rep.Key
Baseline Visit		Baseline Visit Form	
ItemGroup	Rep.Key	Item	Attribute
			FormOID

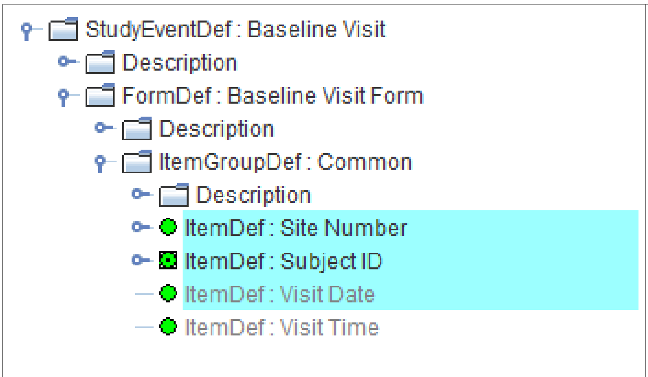
Mapping Description

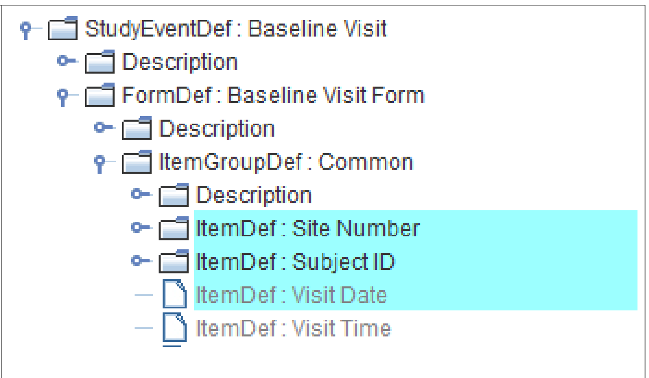
SDTM-ETL mapping for SV.VISIT

The Transformation Script

```
# Mapping using ODM element StudyEventData using value from attribute StudyEventOID
```

The second option „View ODM Items without „traffic lights“ allows you to switch off the „traffic lights“ in the ODM tree.


```
StudyEventDef : Baseline Visit
  Description
  FormDef : Baseline Visit Form
    Description
    ItemGroupDef : Common
      Description
      ● ItemDef : Site Number
      ● ItemDef : Subject ID
      ● ItemDef : Visit Date
      ● ItemDef : Visit Time
```
ODM tree with „traffic lights“ **on**

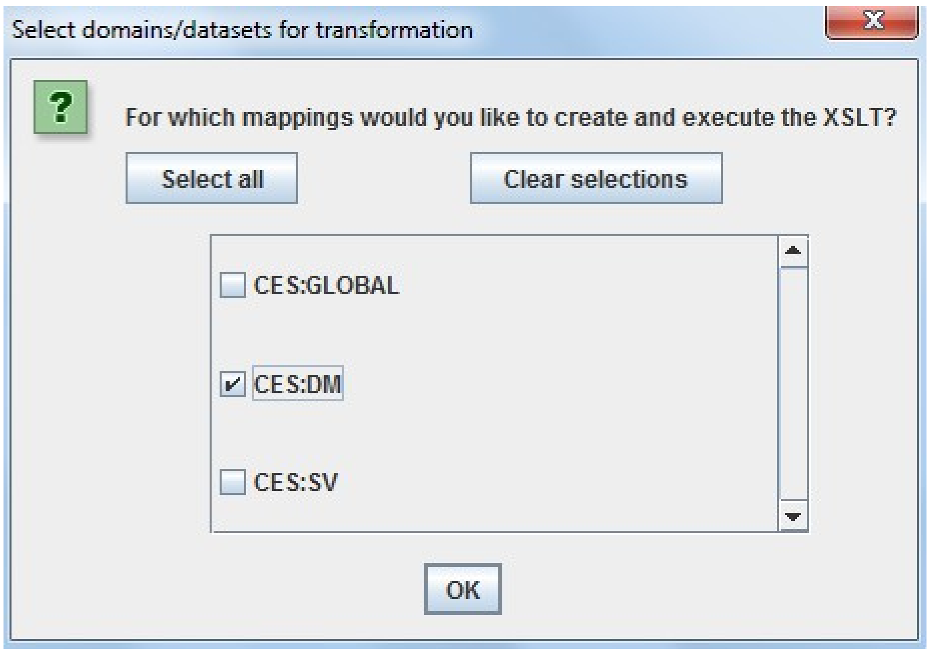

```
StudyEventDef : Baseline Visit
  Description
  FormDef : Baseline Visit Form
    Description
    ItemGroupDef : Common
      Description
      ItemDef : Site Number
      ItemDef : Subject ID
      ItemDef : Visit Date
      ItemDef : Visit Time
```
ODM tree with „traffic lights“ **off**

Some people love the „traffic lights“, other hate them.
Anyway, it is advised to switch them off, in case you see that your system slows down considerably when selecting another SDTM cell. This can happen when you have a very large or complicated ODM tree, as each time that a new SDTM cell is selected, the „traffic lights“ need to be recalculated.

The third option „View ODM tree nodes without graying out mapped nodes“ is similar.
You might have noticed that when an Item in the ODM tree has been used in a mapping, it is „grayed out“. Also here, some users like this feature, others don't.

The next option „Hide sticky notes“ will be explained later, when we discuss the feature of adding „sticky notes“ to SDTM variables.

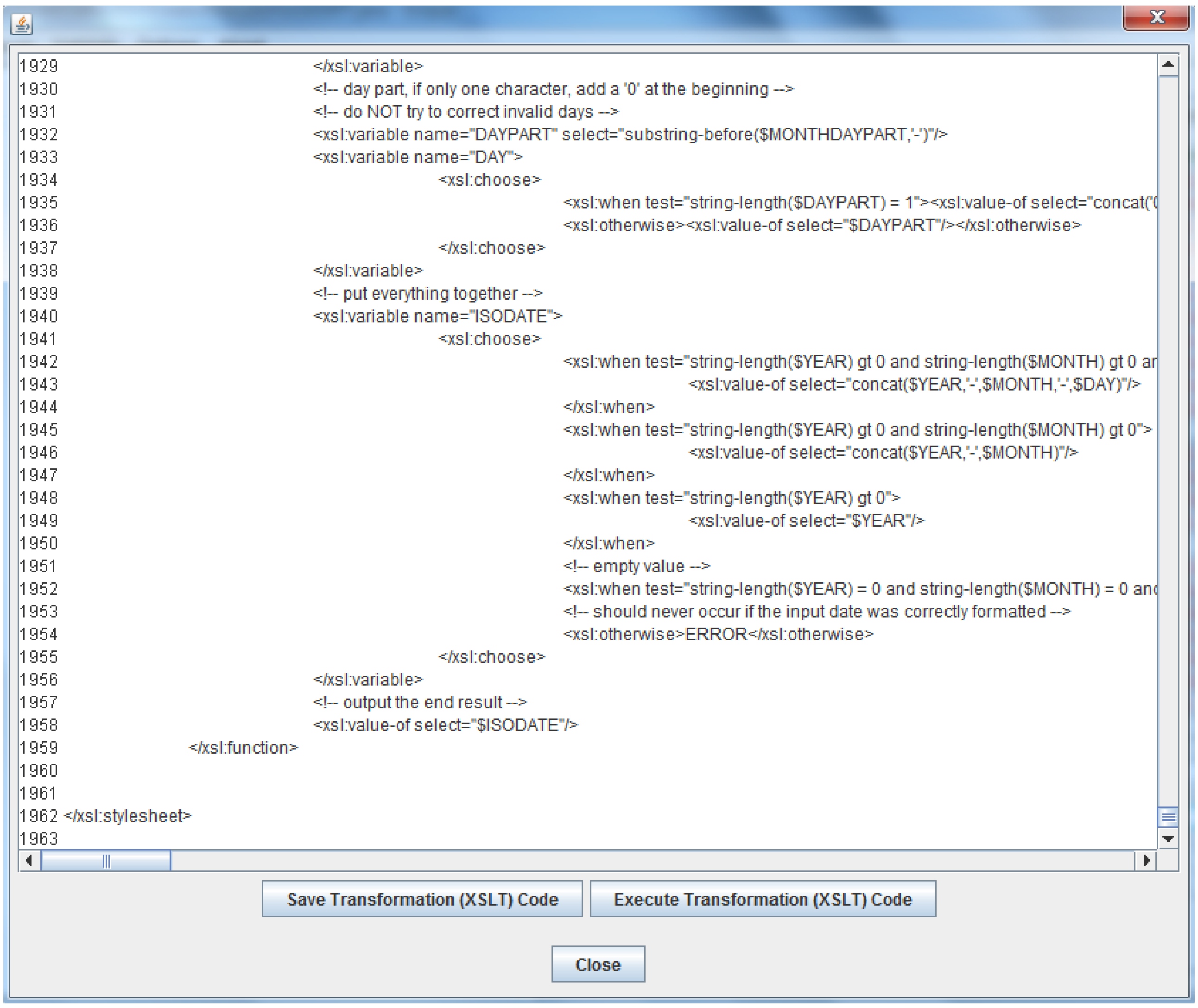
A new option as of SDTM-ETL 3.0 is the option „Generate/Execute XSLT for selected domains only“, and is very useful, especially when you have mappings for several domains, but only want to test the execution of the mappings for a single or a few selected domains only. In case this checkbox is checked, each time you want to execute a mapping, a dialog will be displayed asking you for which of the current domains you would like to execute the mappings. For example you may then see:


Select domains/datasets for transformation
For which mappings would you like to create and execute the XSLT?
Select all
Clear selections
OK

[ ] CES:GLOBAL
[✓] CES:DM
[ ] CES:SV

where it is requested that only the mappings for the DM domain are executed.
Selecting a single of many domains considerably increases the speed of execution, so this options can be very useful.

We now continue with the generation of the XSLT for generating Dataset-XML submission files.
The XSLT is generated and presented to the user:



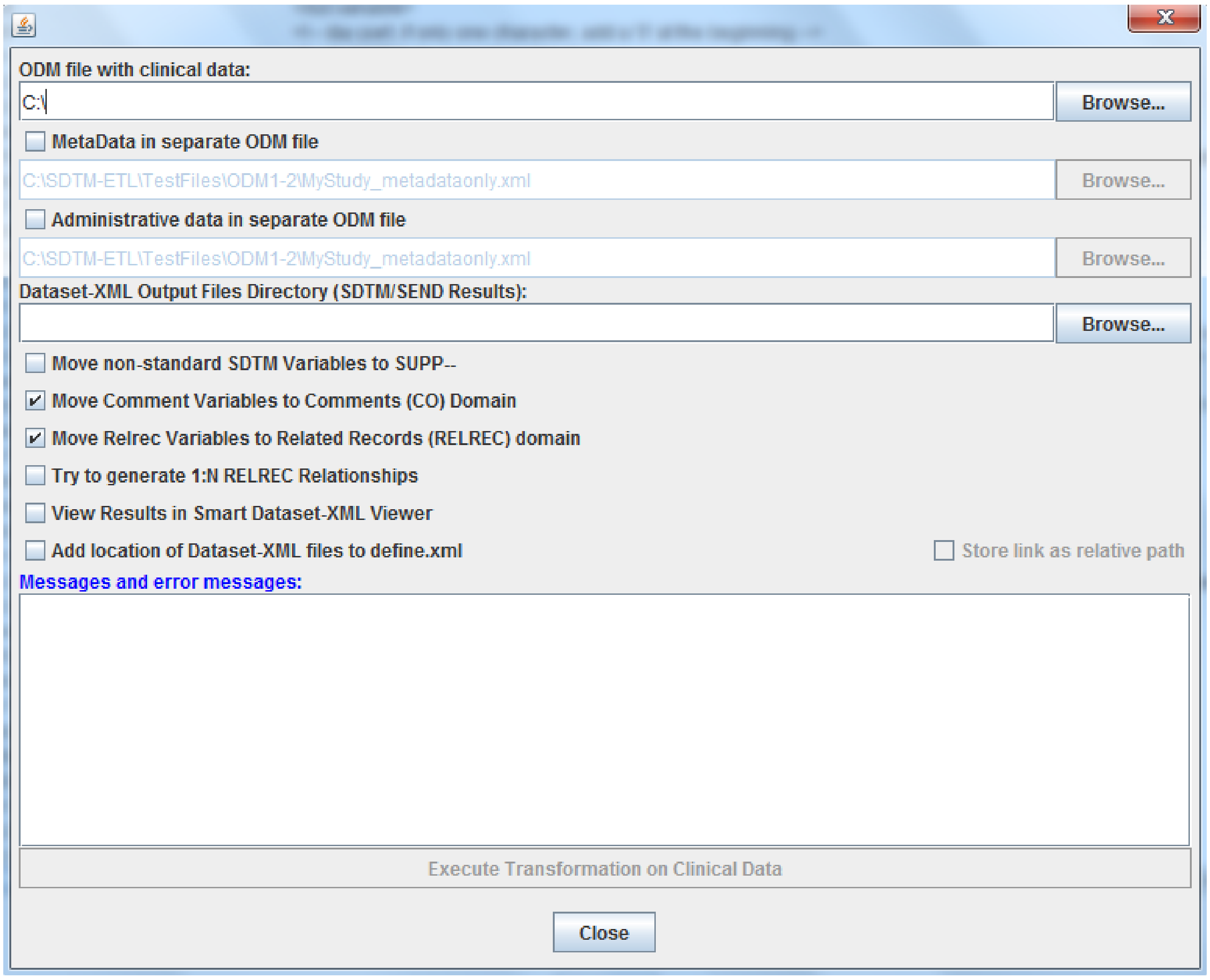
```
        </xsl:variable>
        <!-- day part, if only one character, add a '0' at the beginning -->
        <!-- do NOT try to correct invalid days -->
        <xsl:variable name="DAYPART" select="substring-before($MONTHDAYPART,'-')"/>
        <xsl:variable name="DAY">
            <xsl:choose>
                <xsl:when test="string-length($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 anc
                <!-- 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
Close

No worry! You do not need to understand XSLT to execute the mappings! However, some of our expert users want to see it and some want to save it to disc for offline execution.

Now just click the „Execute Transformation (XSLT) Code“ button. The following dialog is now displayed:


ODM file with clinical data:

C:\

☐ MetaData in separate ODM file

C:\SDTM-ETL\TestFiles\ODM1-2\MyStudy_metadataonly.xml

☐ Administrative data in separate ODM file

C:\SDTM-ETL\TestFiles\ODM1-2\MyStudy_metadataonly.xml

Dataset-XML Output Files Directory (SDTM/SEND Results):

☐ 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 Results in Smart Dataset-XML Viewer

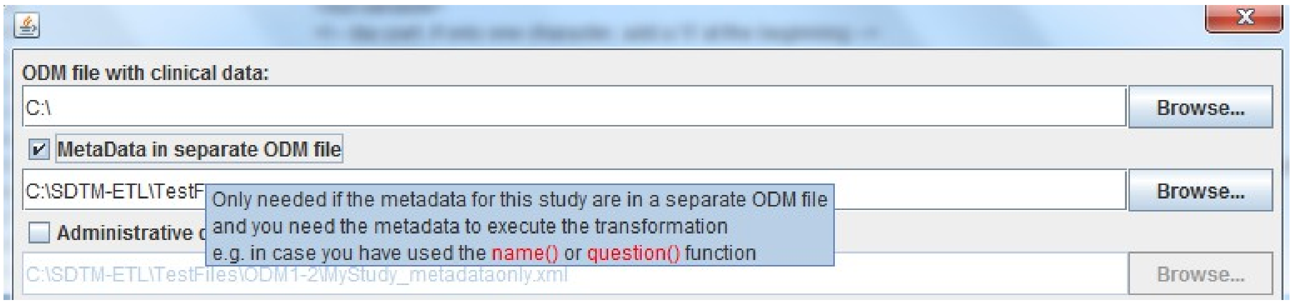
☐ Add location of Dataset-XML files to define.xml

☐ Store link as relative path

Messages and error messages:

You are requested to locate the file with the clinical data.

If you use special functions in your script that requires the metadata (like the „name()" or „question()" function, and your file with clinical data does <u>not</u> contain the metadata, you also need to select the checkbox „", and provide the location of the file with the metadata (study design):


ODM file with clinical data:

C:\

☑ MetaData in separate ODM file

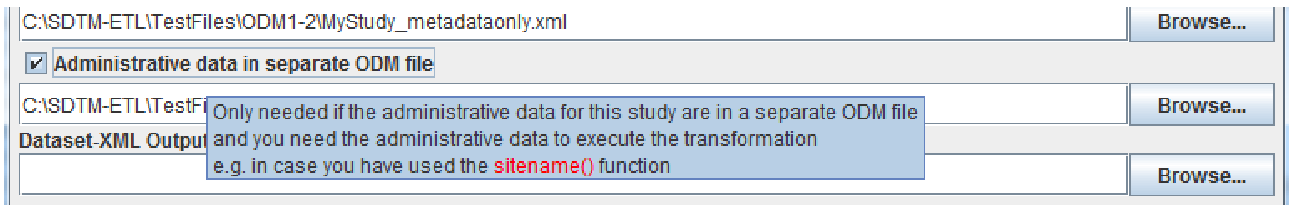
C:\SDTM-ETL\TestF

☐ Administrative d

C:\SDTM-ETL\TestFiles\ODM1-2\MyStudy_metadataonly.xml

Only needed if the metadata for this study are in a separate ODM file
and you need the metadata to execute the transformation
e.g. in case you have used the name() or question() function

Also, if you need to retrieve some information from your „AdminData" (e.g. site information), and this data is in a separate file, you also need to check the checkbox „" and provide the location of the file with administrative data:


C:\SDTM-ETL\TestFiles\ODM1-2\MyStudy_metadataonly.xml [Browse...]

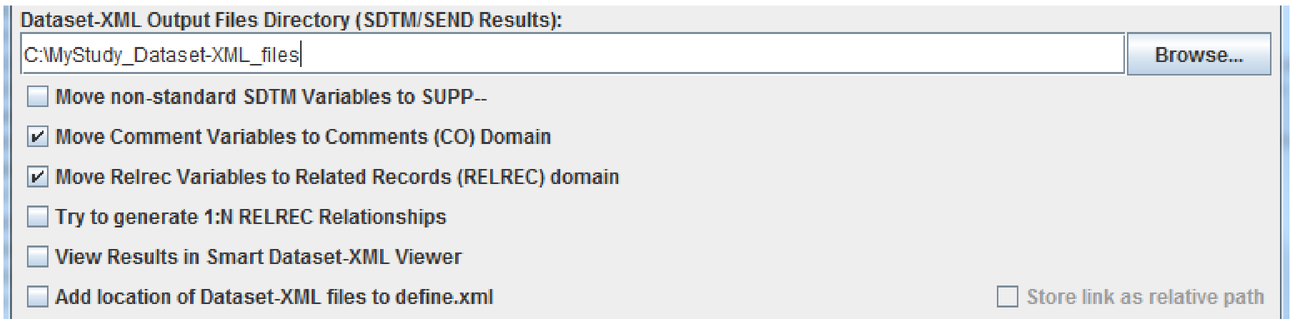
[✓] Administrative data in separate ODM file

C:\SDTM-ETL\TestFi

Dataset-XML Output

Only needed if the administrative data for this study are in a separate ODM file and you need the administrative data to execute the transformation e.g. in case you have used the sitename() function

Now also provide the location of an existing directory where the output of the transformation (the Dataset-XML files) will be written to, e.g.:


Dataset-XML Output Files Directory (SDTM/SEND Results):

C:\MyStudy_Dataset-XML_files [Browse...]

[ ] 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 Results in Smart Dataset-XML Viewer

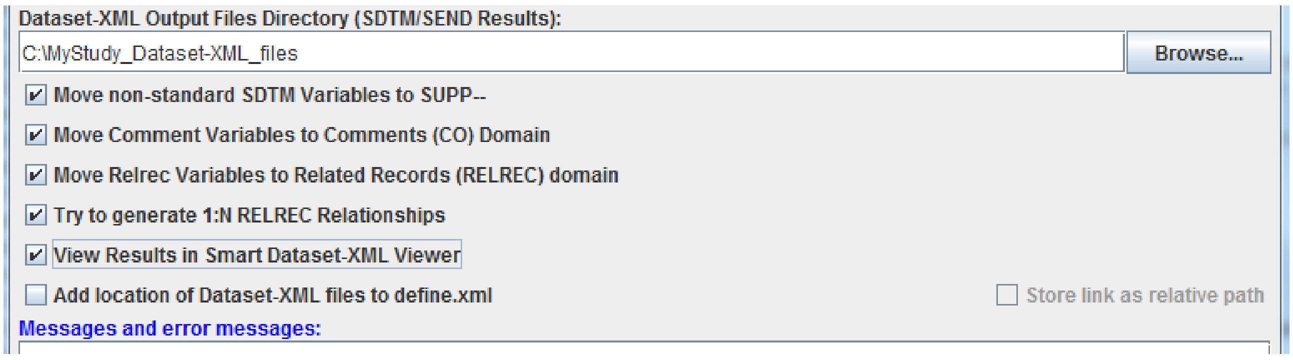
[ ] Add location of Dataset-XML files to define.xml [ ] Store link as relative path

You can now also decide whether non-standard variables should be moved to SUPP-- datasets (which is unfortunately still required by the FDA). Also, if you want to move „comment" variables to a CO dataset (also still required by SDTM), check the corresponding checkbox. Similarly for RELREC records. You can also select whether the system should try to generate 1:N RELREC records. This will be treated in detail later.

Finally, if you want to see the results of your transformation using the open source „Smart Dataset-XML Viewer"[5], check the corresponding checkbox.

The checkbox „Add location of Dataset-XML files to define.xml" will take care of writing the def:leaf elements in the define.xml file. This is only required in the final stages of developing a submission.

Based on the current (July 2014) requirements of the SDTM and the FDA, the typical set iof options will look like:


Dataset-XML Output Files Directory (SDTM/SEND Results):

C:\MyStudy_Dataset-XML_files [Browse...]

[✓] 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

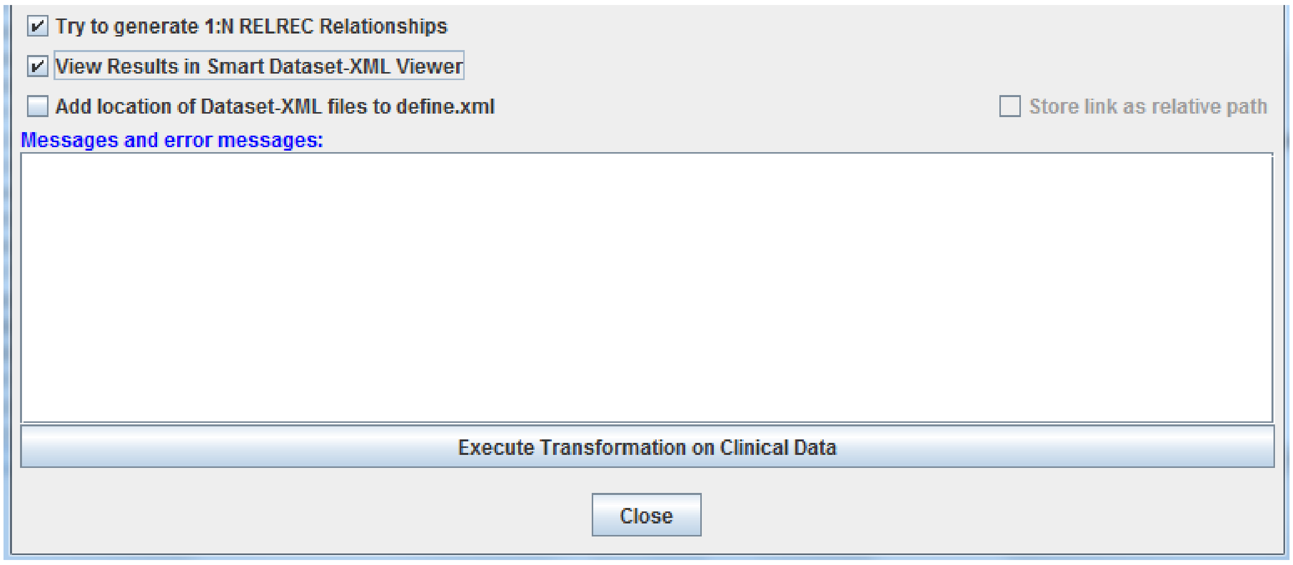
[✓] View Results in Smart Dataset-XML Viewer

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

Messages and error messages:

Now it is time to click the „Execute Transformation on Clinical Data" button:

5 The „Smart Dataset-XML Viewer" comes as a part of the „SDTM-ETL" software. The system has however been designed in such a way, that you can replace an older version of the viewer by a more recent one, or even by your own one (as the „Smart Datase-XML Viewer" is open source, you might have extended it).



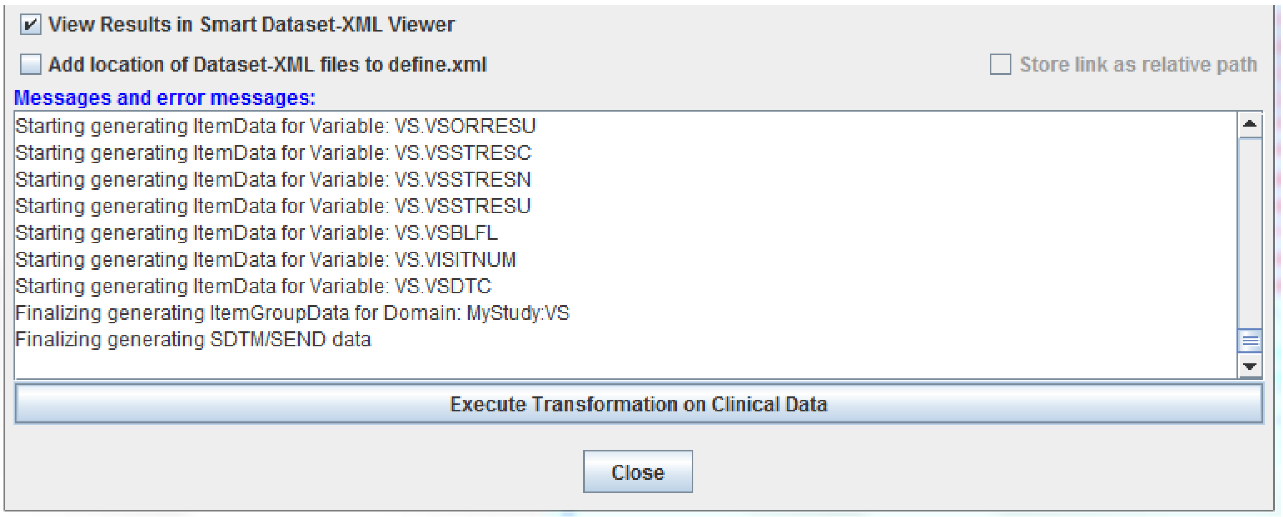
☑ Try to generate 1:N RELREC Relationships

☑ View Results in Smart Dataset-XML Viewer

☐ Add location of Dataset-XML files to define.xml ☐ Store link as relative path

Messages and error messages:

Some messages might show up in the „messages and error messages“ console. For example, if you have used the option in „Options - „ also a message will be printed each time a new dataset has been generated. This console will also show you all the messages from your „print“ statements in your mappings, which is very useful when debugging mappings.
For example:



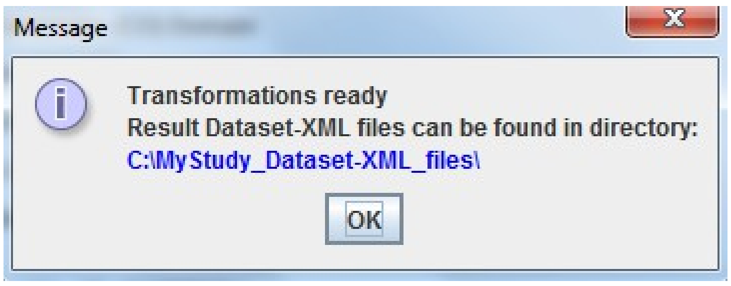
☑ View Results in Smart Dataset-XML Viewer

☐ Add location of Dataset-XML files to define.xml ☐ Store link as relative path

Messages and error messages:

```
Starting generating ItemData for Variable: VS.VSORRESU
Starting generating ItemData for Variable: VS.VSSTRESC
Starting generating ItemData for Variable: VS.VSSTRESN
Starting generating ItemData for Variable: VS.VSSTRESU
Starting generating ItemData for Variable: VS.VSBLFL
Starting generating ItemData for Variable: VS.VISITNUM
Starting generating ItemData for Variable: VS.VSDTC
Finalizing generating ItemGroupData for Domain: MyStudy:VS
Finalizing generating SDTM/SEND data
```

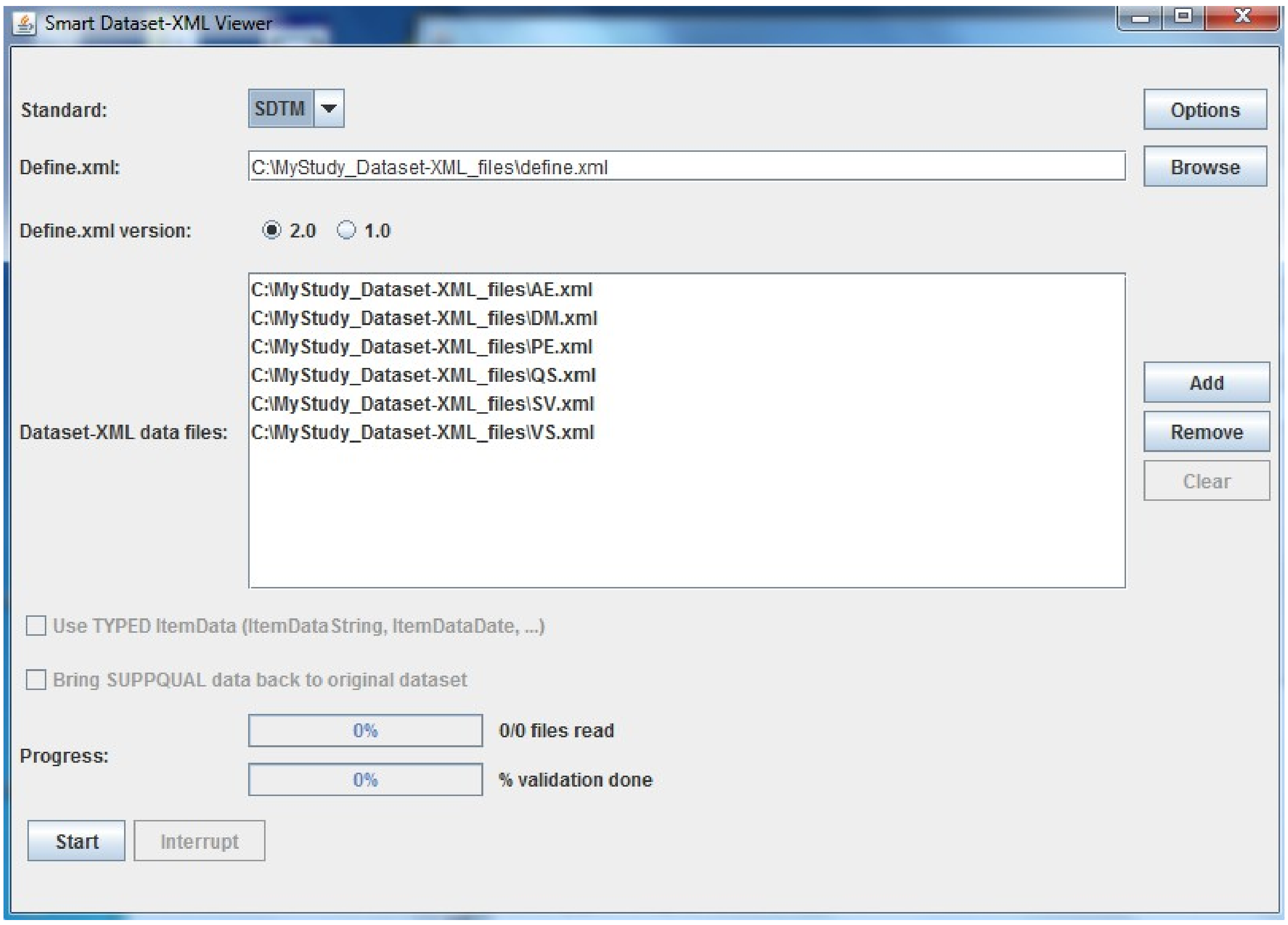
When everything is finished, the following message is displayed:



Message

Transformations ready
Result Dataset-XML files can be found in directory:
C:\MyStudy_Dataset-XML_files\

and if you choose to use the „Smart Dataset-XML Viewer“ for inspecting the results, a new window is displayed already containing all the necessary information, e.g.:



Standard: SDTM

Define.xml: C:\MyStudy_Dataset-XML_files\define.xml

Define.xml version: (● 2.0) ○ 1.0

Dataset-XML data files:

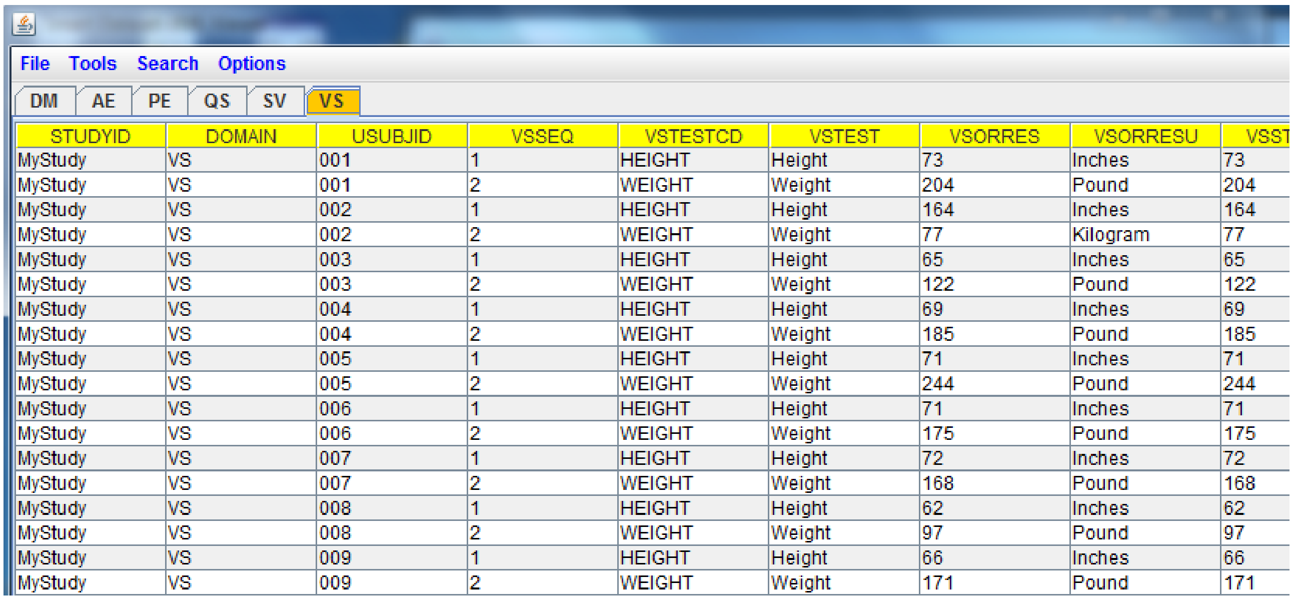
C:\MyStudy_Dataset-XML_files\AE.xml
C:\MyStudy_Dataset-XML_files\DM.xml
C:\MyStudy_Dataset-XML_files\PE.xml
C:\MyStudy_Dataset-XML_files\QS.xml
C:\MyStudy_Dataset-XML_files\SV.xml
C:\MyStudy_Dataset-XML_files\VS.xml

☐ Use TYPED ItemData (ItemDataString, ItemDataDate, ...)

☐ Bring SUPPQUAL data back to original dataset

Progress: 0% 0/0 files read; 0% % validation done

and after clicking the „Start“ button resulting in:



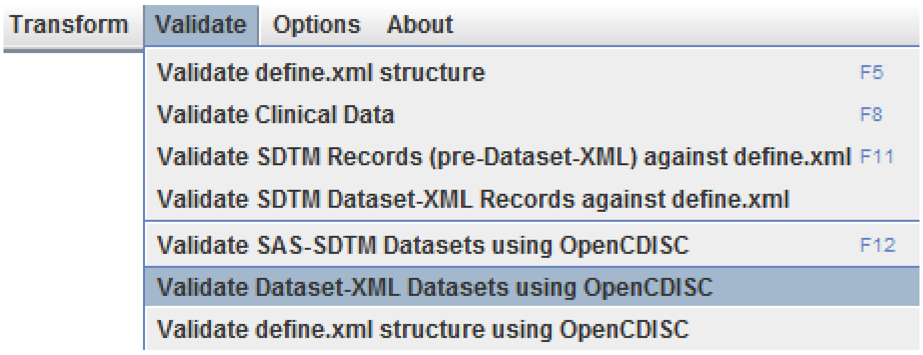
STUDYID	DOMAIN	USUBJID	VSSEQ	VSTESTCD	VSTEST	VSORRES	VSORRESU	VSST
MyStudy	VS	001	1	HEIGHT	Height	73	Inches	73
MyStudy	VS	001	2	WEIGHT	Weight	204	Pound	204
MyStudy	VS	002	1	HEIGHT	Height	164	Inches	164
MyStudy	VS	002	2	WEIGHT	Weight	77	Kilogram	77
MyStudy	VS	003	1	HEIGHT	Height	65	Inches	65
MyStudy	VS	003	2	WEIGHT	Weight	122	Pound	122
MyStudy	VS	004	1	HEIGHT	Height	69	Inches	69
MyStudy	VS	004	2	WEIGHT	Weight	185	Pound	185
MyStudy	VS	005	1	HEIGHT	Height	71	Inches	71
MyStudy	VS	005	2	WEIGHT	Weight	244	Pound	244
MyStudy	VS	006	1	HEIGHT	Height	71	Inches	71
MyStudy	VS	006	2	WEIGHT	Weight	175	Pound	175
MyStudy	VS	007	1	HEIGHT	Height	72	Inches	72
MyStudy	VS	007	2	WEIGHT	Weight	168	Pound	168
MyStudy	VS	008	1	HEIGHT	Height	62	Inches	62
MyStudy	VS	008	2	WEIGHT	Weight	97	Pound	97
MyStudy	VS	009	1	HEIGHT	Height	66	Inches	66
MyStudy	VS	009	2	WEIGHT	Weight	171	Pound	171

For more information about working with the „Smart Dataset-XML Viewer“, please download the manual / tutorial from the Sourceforge site http://sourceforge.net/projects/smart-sds-xml-viewer/files/.

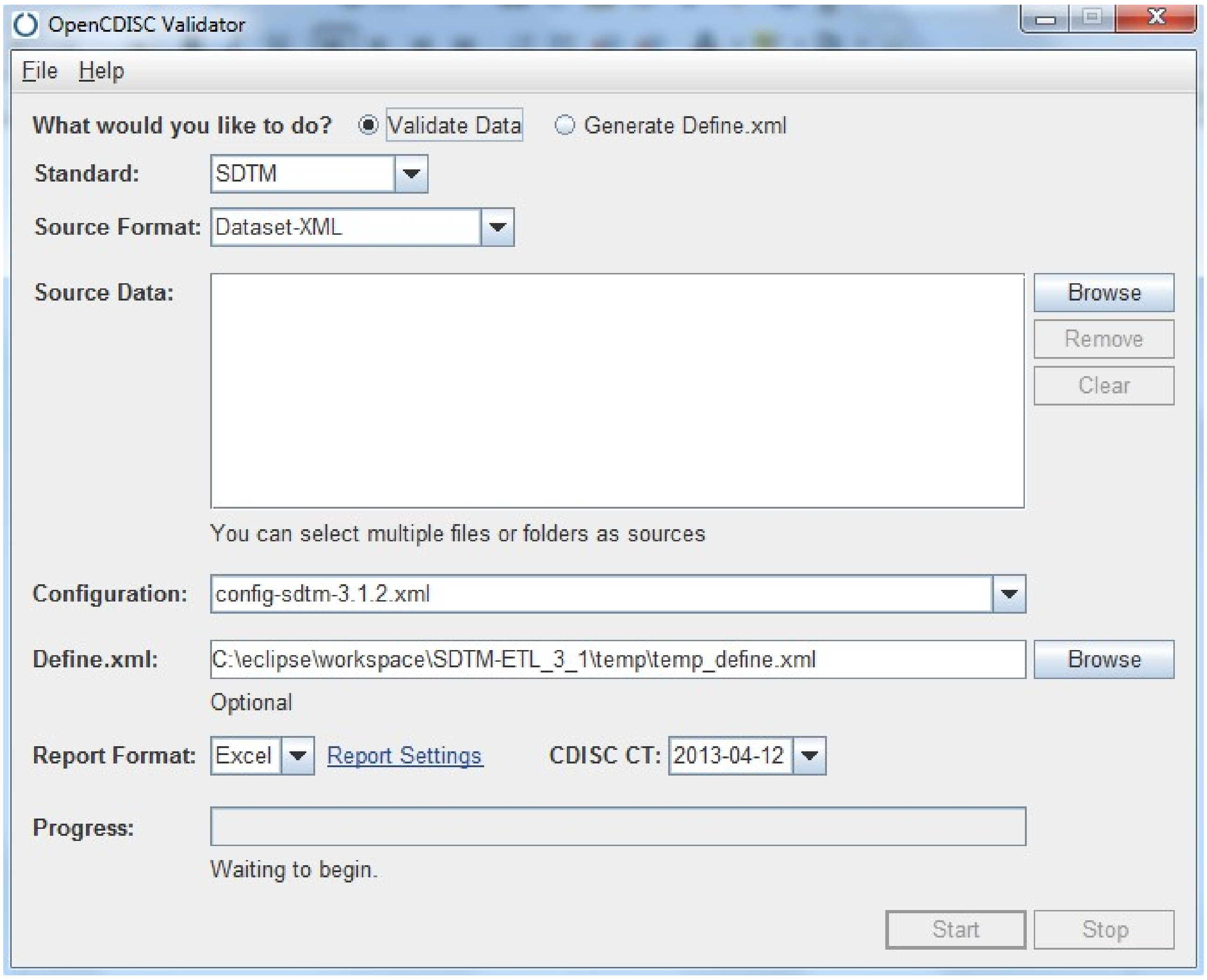
After inspection, you can close all but the main window, and continue with the development of new mappings.

Validating your generated Dataset-XML files using OpenCDISC

The newest version of OpenCDISC (v.1.5) supports Dataset-XML and has been integrated into the SDTM-ETL software[6]. So if you use the menu „Validate – Validate SDTM Dataset-XML Records against define.xml“:


Transform
Validate
Options
About
Validate define.xml structure F5
Validate Clinical Data F8
Validate SDTM Records (pre-Dataset-XML) against define.xml F11
Validate SDTM Dataset-XML Records against define.xml
Validate SAS-SDTM Datasets using OpenCDISC F12
Validate Dataset-XML Datasets using OpenCDISC
Validate define.xml structure using OpenCDISC

OpenCDISC will be started automatically and most of the necessary fields will already be filled:



What would you like to do? [✓ Validate Data] Generate Define.xml

Standard: SDTM

Source Format: Dataset-XML

Source Data:

You can select multiple files or folders as sources

Configuration: config-sdtm-3.1.2.xml

Define.xml: C:\eclipse\workspace\SDTM-ETL_3_1\temp\temp_define.xml

Optional

Report Format: Excel

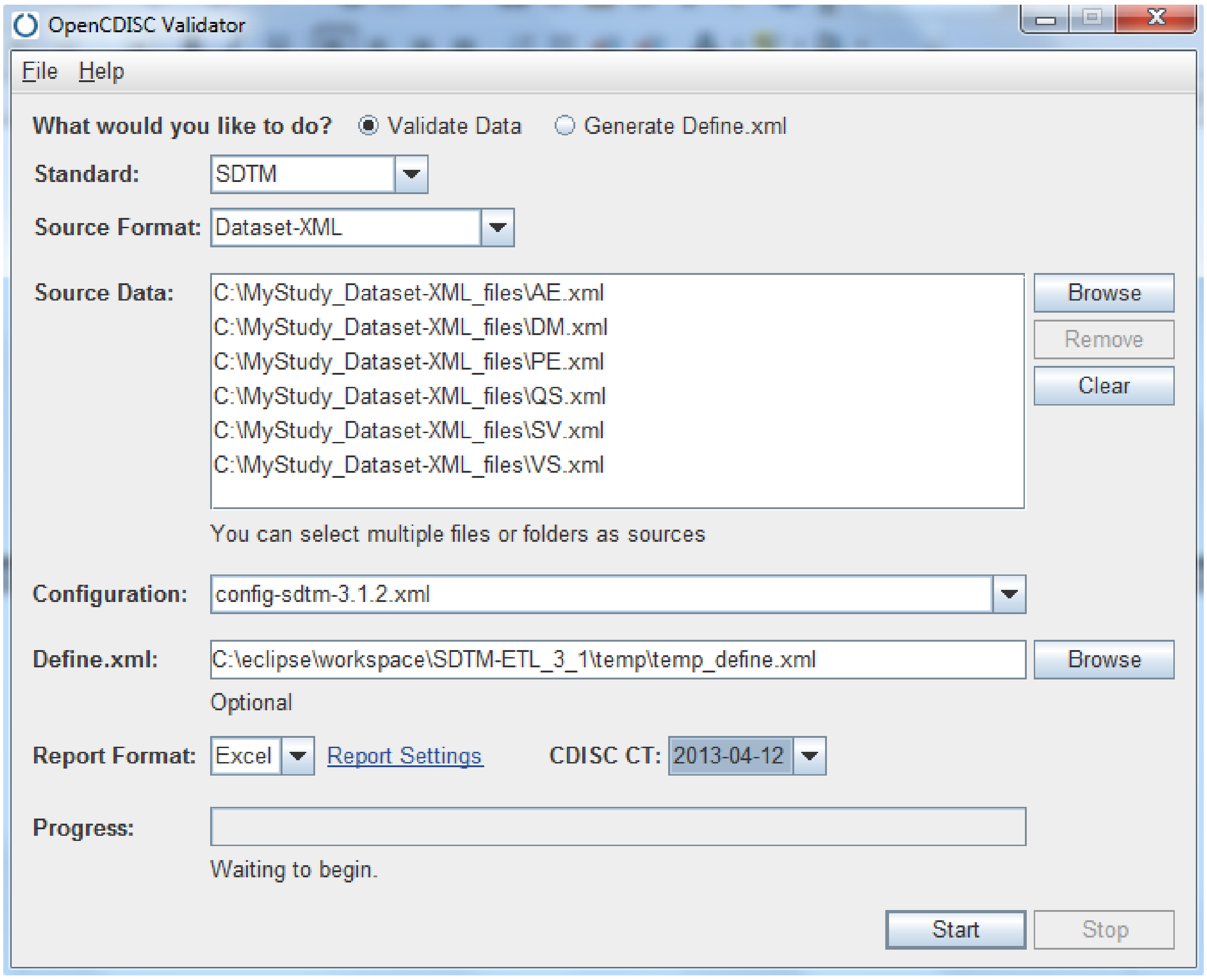
CDISC CT: 2013-04-12

Progress:

Waiting to begin.

You will still need to provide a list of generated Dataset-XML files, e.g. the ones we created in the previous step:

6 Remark that we found some bugs in the Dataset-XML implementation which we have corrected in the SDTM-ETL integration. These bugs have been reported to OpenCDISC but no formal update has been provided by OpenCDISC since.


OpenCDISC Validator

File Help

What would you like to do? [✓ Validate Data] Generate Define.xml

Standard: SDTM

Source Format: Dataset-XML

Source Data:
C:\MyStudy_Dataset-XML_files\AE.xml
C:\MyStudy_Dataset-XML_files\DM.xml
C:\MyStudy_Dataset-XML_files\PE.xml
C:\MyStudy_Dataset-XML_files\QS.xml
C:\MyStudy_Dataset-XML_files\SV.xml
C:\MyStudy_Dataset-XML_files\VS.xml

You can select multiple files or folders as sources

Configuration: config-sdtm-3.1.2.xml

Define.xml: C:\eclipse\workspace\SDTM-ETL_3_1\temp\temp_define.xml

Optional

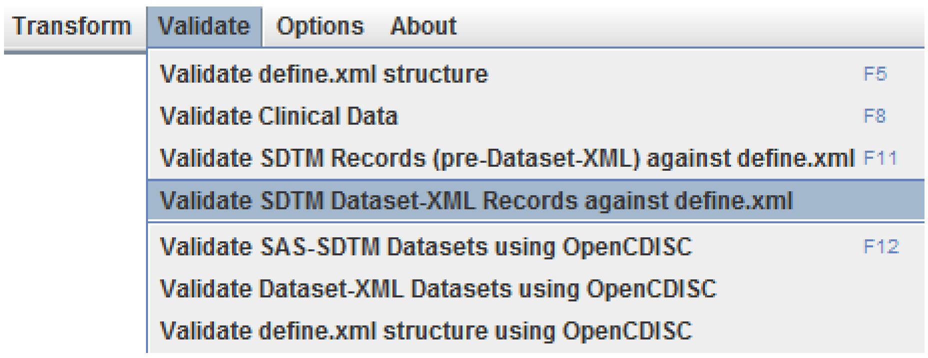
Report Format: Excel Report Settings CDISC CT: 2013-04-12

Progress:

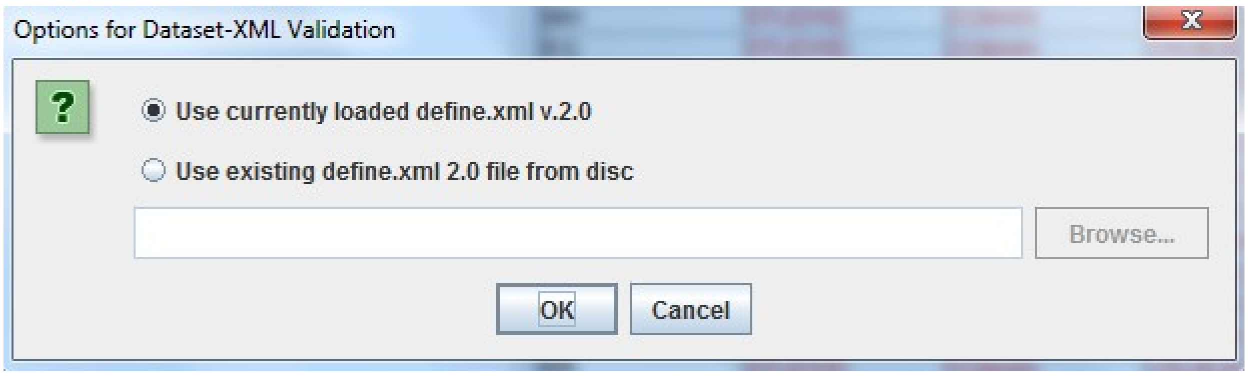
Waiting to begin.

For more information of validating Dataset-XML files using OpenCDISC, please refer to the OpenCDISC website. Please also note that the validation rules of OpenCDISC is the „OpenCDISC interpretation“ of the SDTM/SEND standard, so not the XML4Pharma interpretation and even not the „CDISC interpretation“. So if you get validation errors that you do not understand, or disagree with, please use the „OpenCDISC Forum“ to discuss these with the OpenCDISC people.

For validating the generated Dataset-XML files, you can also use the menu „Validate -


Transform
Validate
Options
About
Validate define.xml structure F5
Validate Clinical Data F8
Validate SDTM Records (pre-Dataset-XML) against define.xml F11
Validate SDTM Dataset-XML Records against define.xml
Validate SAS-SDTM Datasets using OpenCDISC F12
Validate Dataset-XML Datasets using OpenCDISC
Validate define.xml structure using OpenCDISC

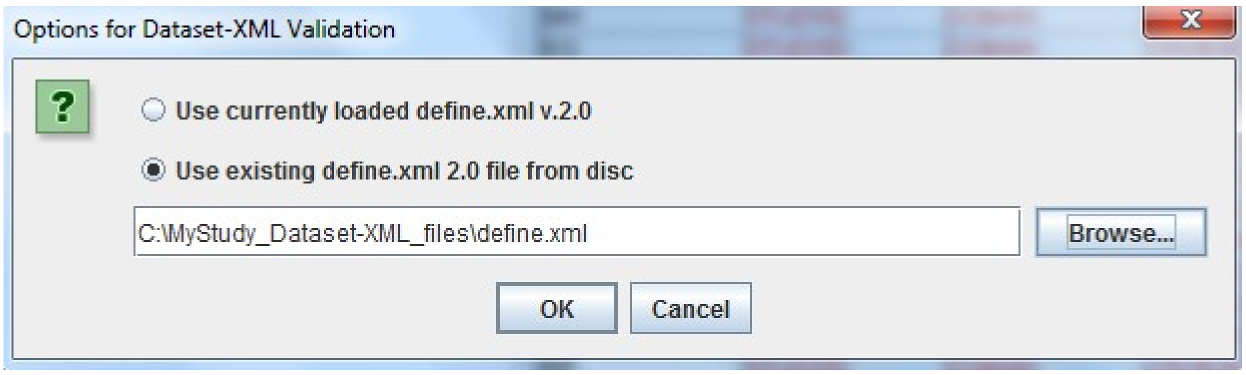
The following dialog is displayed:



(● selected) Use currently loaded define.xml v.2.0

Use existing define.xml 2.0 file from disc

You can either use the currently loaded define.xml (a copy will be made by the software), or if you do already have one (as in our case), you can select to use the one that you have already on disc:

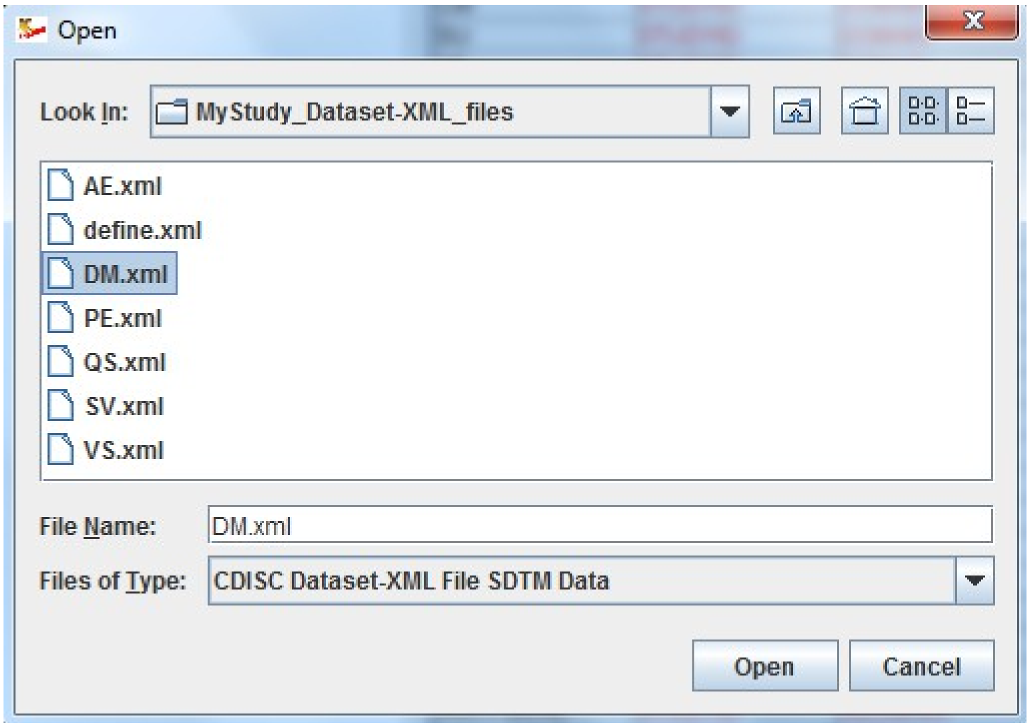


Use currently loaded define.xml v.2.0

(● selected) Use existing define.xml 2.0 file from disc

C:\MyStudy_Dataset-XML_files\define.xml

after clicking OK, you can start selecting a Dataset-XML file, e.g.:



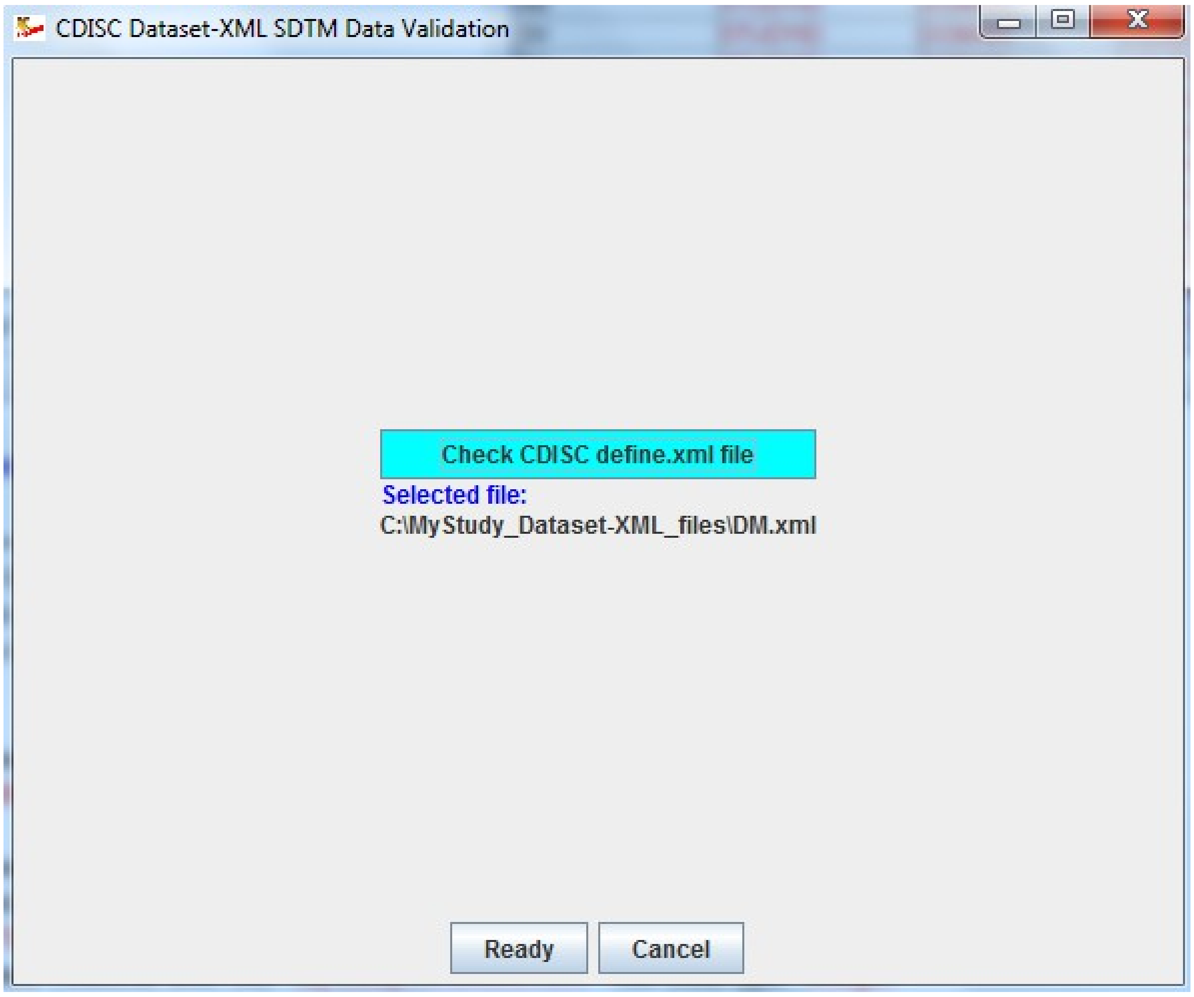
Look In: MyStudy_Dataset-XML_files

AE.xml
define.xml
DM.xml
PE.xml
QS.xml
SV.xml
VS.xml

File Name: DM.xml

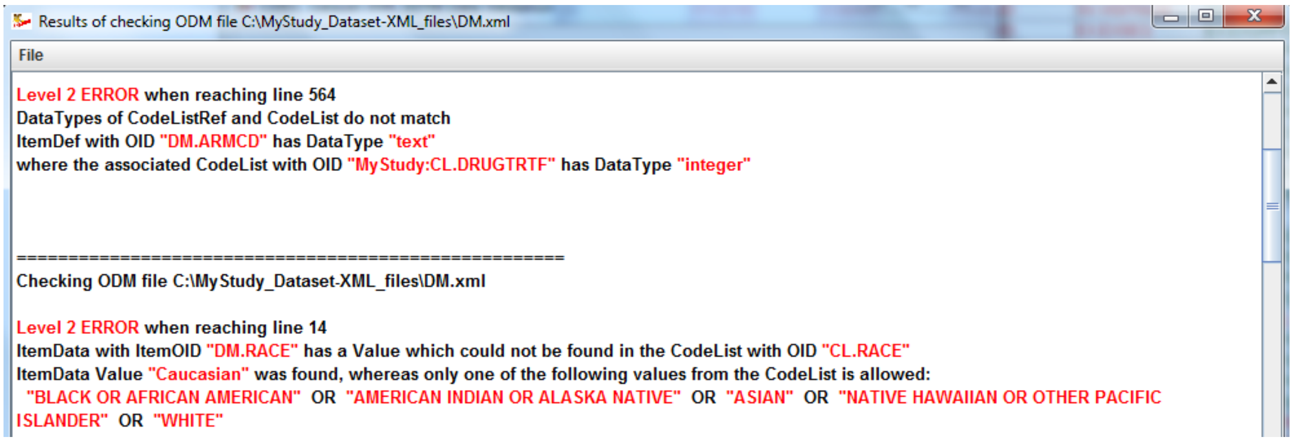
Files of Type: CDISC Dataset-XML File SDTM Data

Unlike with OpenCDISC, you can only validate one Dataset-XML file at the time, leading to the following dialog:


CDISC Dataset-XML SDTM Data Validation
Check CDISC define.xml file
Selected file:
C:\MyStudy_Dataset-XML_files\DM.xml
Ready
Cancel

TODO: text on button is not correct.

And e.g. as a result of the validation:


Results of checking ODM file C:\MyStudy_Dataset-XML_files\DM.xml
File

```
Level 2 ERROR when reaching line 564
DataTypes of CodeListRef and CodeList do not match
ItemDef with OID "DM.ARMCD" has DataType "text"
where the associated CodeList with OID "MyStudy:CL.DRUGTRTF" has DataType "integer"



=====================================================
Checking ODM file C:\MyStudy_Dataset-XML_files\DM.xml

Level 2 ERROR when reaching line 14
ItemData with ItemOID "DM.RACE" has a Value which could not be found in the CodeList with OID "CL.RACE"
ItemData Value "Caucasian" was found, whereas only one of the following values from the CodeList is allowed:
 "BLACK OR AFRICAN AMERICAN"  OR  "AMERICAN INDIAN OR ALASKA NATIVE"  OR  "ASIAN"  OR  "NATIVE HAWAIIAN OR OTHER PACIFIC
ISLANDER"  OR  "WHITE"
```

There is an important difference in validation approach between the OpenCDISC validator and the one from SDTM-ETL:
Essentially, OpenCDISC validates the contents of the files against the SDTM rules and official CDISC controlled terminology. For example, if you developed your own codelist for an SDTM variable (which is stored in the define.xml) instead of using the CDISC one, OpenCDISC will report an error.
The SDTM-ETL validator for Dataset-XML files however only validates the data against the metadata in the define.xml. So it does not try to interprete the SDTM-IG, nor does it do a look up in

any published CDISC-CT. Leading in the validation is only information in the define.xml.
This does thus also mean that the SDTM-ETL validator for Dataset-XML does not e.g. validate that an end date/time (--ENDTC) does not come before the corresponding start date/time (--STDTC).

So validating your Dataset-XML files with OpenCDISC and with the build-in validator WILL give different results, which should be seen as complimentary.

Generating SAS-XPT (Transport 5) SDTM files

You can still generate SAS Transport 5 SDTM files using SDTM-ETL 3.0/3.1. This is probably the safe option as long as there is no official announcement that the FDA accepts submissions in the new Dataset-XML format (or it must be that you take part in the FDA pilot project).

The process for generating SDTM/SEND datasets is very similar to what is described here (you will see many of the dialogs already displayed here), and is explained in a separate document.