New features in version 3.2

Version 3.2 is available since spring 2017
Table of Contents

Introduction........................................................................................................................................3
Support for SEND-IG v.3.1..................................................................................................................4
Wizards for easier generation and editing of ValueLists.................................................................6
Automated adaption of variable length from longest result value....................................................28
Retrieving page numbers from an annotated CRF in PDF format..................................................32
Skipping display of the generated XSLT........................................................................................39
Additional date and time functions..................................................................................................40
Progress messages...........................................................................................................................42
Extended messages for XSLT compilation and execution errors..................................................43
Post-execution derivation of the new SDTM 1.5 –LOXBFL baseline flag........................................44
**Introduction**

SDTM-ETL™ 3.2 is another "big step forward" in the development of software for the mapping between operational clinical data and the CDISC SDTM and Define-XML standard.

Although SDTM itself is under pressure, because being outdated (still using 2-dimensional tables and the "medieval" SAS-XPT format), and because more and more "derived" variables are added (see [here](#) and [here](#)), making it unnecessarily complex and error-prone, we will still need to live some time with it, also as the tools of the regulatory authorities are not very modern.

To cope with this (unnecessary) complexity, we are currently adding a good number of new features to SDTM-ETL™, making it even more user-friendly than it was, and at the same time adding new support for the new standards that are currently been developed and will soon be released by CDISC: SDTM 1.5/SDTM-IG 3.3 and Define-XML 2.1.

We are very proud to be able to present SDTM-ETL v.3.2.
We are convinced that this is the best tool on the market for mapping operational clinical data to submission data, with the lowest total cost of ownership (TCO).
Support for SEND-IG v.3.1

SDTM-ETL v.3.2 now fully supports SEND-IG v.3.1 which is based on the new SDTM v.1.5. No Implementation Guide for SDTM (SDTM-IG) has been published yet for this SDTM version. We have already made provisions for it and will implement it as soon as CDISC publishes the final version.

For SEND-IG v.3.1 everything however has been implemented, including the define.xml template, and all the rules (as far as they can be implemented in software) from the SEND-IG v.3.1.

When the user uses the menu “File – Create define.xml”, a new choice “SEND 3.1” is also displayed:

When the user chooses for “SEND 3.1”, also the latest SEND controlled terminology (2017-06-30) can be selected:
After clicking “OK”, the define.xml template for SEND 3.1, as well as the SEND controlled terminology is loaded.
Wizards for easier generation and editing of ValueLists

Generating and maintaining valuelists can be a lot of work. Essentially, one will need to generate a value-level variable for each test (or group of them) one has in each of the "Findings" domains. For example, if one has 50 different lab tests, each with their own metadata (datatype, max.length, number of figures after the decimal point, units used, or codelist for e.g. ordinal results), one may end with 50 different value-level variables, one for each test.

That looks like a lot of work, but we will show that this can be automated for a good part, without the user loosing control and oversight.

Usually, a ValueList is created started from a codelist. In SDTM-ETL™ this is done using the menu "Insert - Create ValueList from CodeList":

![SDTM-ETL Version 3.2 interface](image)

The user can then select one of the codelists that was loaded with the SDTM template. Usually (but not always) this will be CDISC CodeList:
The user can search for a specific codelist using the "Search" textfield and button. For example, for the "Vital Signs codes" codelist:
The checkbox "Create simple 'WhereClause' automatically" will allow to generate the "WhereClauses" for the valuelist items automatically. This is extremely useful, as in over 90% of the cases, the "WhereClause" will be very simple. For example for an item "PULSE", the "WhereClause" will typically be "WHERE VSTESTCD = DIABP". When this checkbox is checked, these simple "WhereClauses" will be generated automatically. This will often save an enormous amount of time.

In case the codelist contains many items, it will take some time to generate the valuelist. However, you will very often want to subset a codelist first. For example, it is very unusual that you will have all 30 vital sign tests (from the CDISC CodeList) in your study. For the lab test codes (LBTESTCD) there are over 1400 test codes, so you will definitely want to subset that codelist first, only containing the codes for the test that were actually planned or used in the study.

When the system detects that you have more than 25 codes in your codelist, it will display the following dialog:

If we do subset our codelist first, using the menu "Insert - Create new SDTM CodeList from existing CodeList", we get the following dialog:

---

1 It is not understandable why CDISC has not switched yet to the LOINC system for lab tests. LOINC is used for lab coding in almost every hospital in the world.

2 Remind that you also need to include all tests that were planned, even when some of them were never executed, or for which no data was ever collected.

SDTM-ETL™: New features in v.3.2 p.8/44
also here, we can use the "Find" button to quickly find the desired one.

Once we made a selection for which codelist to subset, the system will make a proposal for a new OID and Name of the subset codelist, which you can of course always change:
For example, you might want to change the OID in "CL.MYVTESTCD" and the "Name" in "Vital Signs Test Code for my study":

We can then start removing test codes (using the button "Remove row"), but also add new ones (using the button "Insert row"). For example, we remove those that were never planned in our study and we add a new test code "BLPRATIO" ("blood pressure ratio"):

As one can see, when one adds a new term, the checkbox "ExtendedValue" is automatically selected. This information will then appear in the define.xml, meaning that the codelist has been extended with the value "BLPRATIO".

When removing items, and the codelist is a "test code" or "test name" codelist, you can always inspect the meaning (or code) by holding the mouse over an item, showing it as a tooltip. For example:

Unfortunately, there
Especially for test codes, the current SDTM 8 character limitation often makes it very difficult to immediately understand the meaning of a code. It is really high time that this limitation disappears.
After you have subsetted the codelist, you can now try to make a valuelist of it again:

![codelist selection window]

after clicking "OK", you will get the following message:

![warning message]

The reason is that, when you automatically want to create the "WhereClause" the system needs to know to what SDTM variable the codelist has been assigned. As we still have the old codelist "CL.C66741.VSTESTCD" assigned to the SDTM variable VSTESTCD, it is now a good time to update that.
In order to do so, select VSTESTCD in the main window, and use the menu "Edit - SDTM Variable" (or use CTRL-E), leading to:
If the checkbox "New CodeList" is selected, you can reassign the codelist, using the dropdown:

after clicking "OK", the subset codelist has been assigned to variable VSTESTCD, as one can easily see from the tooltip on the variable:

We can then try to create the new ValueList again, this time for the codelist "CL.MYVSTESTCD":

SDTM-ETL™: New features in v.3.2  p.13/44
For each test code, a value-level variable has been created, and one needs to provide the datatype, in most cases the maximal length and sometimes also the number of digits after the decimal point, and sometimes an associated codelist, like for "FRMSIZE" ("Frame Size": small, medium, large).

This looks like a lot of work! But fortunately, all the "WhereClauses" have already been generated automatically, as one easily sees when holding the mouse over one of them:

```
| WC.IT.WEIGHT |
| WC.IT.WSTCIR |
| WC.IT.BLPRATIO |
```

Or when clicking in the cell "WC.IT.BLPRATIO":
showing us the editor for the "WhereClause". We can still change anything here, for example to generate a single "WhereClause" for both diastolic blood pressure (DIABP) and systolic blood pressure (SYSBP). The "WhereClause" will then typically be like "WHERE VSTESTCD IN [DIABP,SYSBP].

In the ValueList table, we can now still remove rows in the table for which there was no test anyway in our study, or add new rows (tests). We can do this manually, but we can also do a lookup in the source ODM, to find out whether there was such a test.

In the lower part of the screen, we find 3 new buttons:

- Retrieve metadata from ODM source using SDTM annotations
- Retrieve metadata from ODM using (CRF) Item name
- Generate WhereClause automatically

First of all, let us look which vital signs were really present in the source ODM (study design). We can easily find out using the button "Retrieve metadata from ODM using (CRF) Item name". For example, when we select the first row containing "BMI", and then click the button, we are presented with the list of items (questions/datapoints) from the ODM:
and using the "Search" button:
So, the system did not find any item in which the wording "BMI" appears. We can of course also do a search on "body mass index" or similar, but will not find something like that either. So we can assume that this test was not in the study design (it might of course be that it was named completely differently, but then we have a semantic problem) and so can be removed from the valuclist.

So, after removing the first row, the valuclist reduces a little bit as:
Finding out which tests were in the study design goes pretty fast in this way. We can then remove rows until we only have the vital signs that were really present in our study design.

In our valuelist table, there is a second way to find out whether a test was in the study design or not. This method is based on the annotations that were added to the ODM study design, usually already at study design time. These annotations use the "Alias" element in the ODM, usually with the "Context" being "SDTM". For example:

As the annotations (or better, their way of writing) is not standardized, some variations of it can be found, like "VSORRES where VTESTCD = 'SYSBP'".

In our ValueList table editor, we can make advantage of this, by using the button "Retrieve Metadata from ODM Source using Annotations":

For example, if we first select the row "WSTCIR" and then click the button, we get:

TODO: replace image by the one for WSTCIR

5 This is an extremely good practice, explained in our "ODM Wiki"

SDTM-ETL™: New features in v.3.2
We can still edit the search expression, and also use a "wildcard" ("*" or "/") if we do not know part of the annotation. However, the system is very smart, it will look for sentence similarity in all the ODM-SDTM annotations, and rank them according to the similarity. For the above search term (without any editing) we get:

indicating that none of the data points in the study design was annotated as being to be used later with VSTESTCD=WSTCIR. So we can remove that row too.

If we do the same however for the row with entry "DIABP":

leading to:
and then executing the search, leading to:

the items in the list being ordered by similarity with the search string, and showing that there is indeed an item in the study design that was annotated and that is a "diastolic blood pressure".

We can than copy its metadata from the ODM to the SDTM, by selecting the item and clicking the button "Copy Metadata to ValueList":

---

SDTM-ETL™: New features in v.3.2  

p.20/44
and immediately, we see in our valuelist table:

<table>
<thead>
<tr>
<th>OID</th>
<th>Name</th>
<th>Data Type</th>
<th>Length</th>
<th>Sign.Digits</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>T.DIABP</td>
<td>DIABP</td>
<td>integer</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T.FARMCIR</td>
<td>FARMCIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T.FRMSIZE</td>
<td>FRMSIZE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T.HDCIRC</td>
<td>HDCIRC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T.HEIGHT</td>
<td>HEIGHT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T.HHR</td>
<td>HHR</td>
<td>integer</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T.PULSE</td>
<td>PULSE</td>
<td>integer</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T.SYSEBP</td>
<td>SYSEBP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T.TEMP</td>
<td>TEMP</td>
<td>float</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T.WEIGHT</td>
<td>WEIGHT</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T.WSTCIR</td>
<td>WSTCIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T.BLPRATIO</td>
<td>BLPRATIO</td>
<td>float</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

that the datatype (in this case "integer") maximal length (in this case "3") have been copied. If a codelist would be involved, also the codelist reference would be copied in the cell "CodeList". If that codelist is not present in the loaded controlled terminology (in the underlying define.xml), the user is invited to copy the codelist from the ODM into the SDTM, or to choose another one that is already in the SDTM. The latter can easily be done by using the dropdown, e.g. for "FRMSIZE":

With these two new features, one can easily and relatively quickly find out which tests were in the study design, for which value-level metadata need to provided, and copy them from the study design automatically⁶. The ones in the list that were not in the study design can then easily removed from the valuelist table.

Doing so in our case, quickly leads to:

We decided not to make this a "bulk" or "blind" automatism, as that can easily lead to errors, like removing tests that were known under a (more or less slightly) different name or code in the study design. We consider it to be important that the user understands what he/she is doing, unlike in "black box" tools.

---

⁶ SDTM-ETL™: New features in v.3.2
with datatype and maximal length being copied from the source ODM study design.

The third button under the table states "Generate WhereClause Automatically".

Most of our "WhereClauses" will be similar to "where VSTESTCD=DIABP" for the value-level variable "SYSBP". So, when one has added new items to the valuelist after it has been created, one can still automatically generate the "simple" "WhereClause".

For example, when we add a new entry "EARSIZE", select it, and then click the button "Generate WhereClause Automatically" is clicked, a dialog is displayed with:

```
WhereClause for: EARSIZE

WhereClause will be:
WHERE VSTESTCD='EARSIZE'

OK   Cancel
```

You can still cancel this now. If "OK" is clicked, the "WhereClause Wizard" shows up, allowing to still make changes to our "WhereClause":

Also here, we can validate whether the underlying machine-readable expression is correct, using the "Show 'Where' Clause" button. We can however also add a comment and/or add a link to a comment to an external document (like a reviewers guide).

If you feel fine, just click the "OK" button, returning to the valuelist table. We then see that the cell on the right is highlighted (blue color). When holding the mouse over the cell, we also see the "WhereClause" as a human-readable expression:
Just for the exercise, let us know make a single "WhereClause" for as well the systolic blood pressure. This makes sense, as both are very similar properties, both are measured as an integer with a maximal length of 3.

First, we remove one of both entries. Let's says we just keep the first which is "DIABP", and remove "SYSBP":

Then we edit the OID and the Name to reflect that this is about blood pressure. For example:

Remember that (as long as we are forced to use SAS-XPT), the value for "Name" may not be longer than 8 characters. For the "Description", we write "Systolic and diastolic blood pressure". Due again to the SAS-XPT limitation, the "Description" may not be longer than 40 characters:

The one thing we still need to adapt is the "WhereClause". So we click the corresponding cell on the right, and the "WhereClause Wizard" is displayed:

Well, isn't it time the FDA moves away from SAS-XPT?

SDTM-ETL™: New features in v.3.2  p.23/44
using the "Show 'Where' Clause" button, we see that currently, only DIABP is covered:
but we of course also want it to cover SYSBP. Such an "OR" statement is essentially accomplished by providing a list, and then stating that the value must be in the list. So we set the "Comparator" to "IN" (using the dropdown), leading to:

"VSTESTCD" is still OK, and we still need to add "SYSBP" and "DIABP" to our list. We can just do by typing into the "Add to or remove from list", and then click "Add to list", like:

and again for "DIABP":

SDTM-ETL™: New features in v.3.2
When then clicking the "Show 'Where' Clause again, we get:

which is exactly what we want.

We also change the OID (although it is nothing else than an identifier, so it is not mandatory, but we like our OIDs to be "mnenomic", so we make it:
Clicking "OK" then brings us back to the valuelist table:

<table>
<thead>
<tr>
<th>OID</th>
<th>Name</th>
<th>Data Type</th>
<th>Length</th>
<th>Description</th>
<th>WhereClause</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT BLOODPRESSURE</td>
<td>BLP</td>
<td>integer</td>
<td>3</td>
<td>Systolic and diastolic blood pressure</td>
<td>WC IT BLOODPRESSURE</td>
</tr>
<tr>
<td>IT HEIGHT</td>
<td>HEIGHT</td>
<td>integer</td>
<td>3</td>
<td>HEIGHT</td>
<td>WC IT HEIGHT</td>
</tr>
<tr>
<td>IT WEIGHT</td>
<td>WEIGHT</td>
<td>integer</td>
<td>3</td>
<td>WEIGHT</td>
<td>WC IT WEIGH where VSTBOD IN [SBP, DBP]</td>
</tr>
</tbody>
</table>

again showing the "human-readable" expression as a tooltip.

We should not forget to validate the table, using the "Validate" button which is near the bottom. Using it leads to … nothing, meaning that our "ValueList" table is technically valid.

We could still add the "Origin", but we could also do that at the higher level, when all vital signs where collected on the same CRF "page"\(^8\). We will describe how to semi-automatically add CRF page numbers in one of the next sections.

---

\(^8\) When using EDC, this essentially does not make sense. As we start from a study design in ODM, our mapping script IS the description of the origin. However, the FDA still want sponsors to generate a paper-like annotated CRF artificially, with page numbers, although the EDC screens do not have such at all.
Automated adaption of variable length from longest result value

The FDA requires that the variable lengths in the SAS-XPT files exactly match the length of the longest value in the variable column. So they introduced the famous rule FDAC036 "Variable length should be assigned based on actual stored data to minimize file size. Datasets should be resized to the maximum length of actual data used...".

Essentially, this means that one would need a postprocessing step when generating the SAS-XPT files, as one cannot know the "maximal length" in advance. We did however find a smarter way.

When generating the XSLT transformation script and then executing it, the following dialog is displayed:

![Image of a dialog box](image.png)

Remark the new checkbox "Adapt Variable Length for longest result value". When it is checked, the data will be generated from the mapping scripts, but not stored immediately into the SAS-XPT files, but instead they will be parsed again, and for each variable, the largest value length will be retained. This then leads to a new dialog:

This is a ridiculous requirement: the reason behind it is the unfounded fear of the FDA for large file sizes and the fact that SAS-XPT is a "constant record length" format, where blanks are inserted after the text value itself to have each value (per variable) the same length, and thus all records the same length. So SAS-XPT uses a format that is wasting an enormous amount of bytes. So SAS-XPT is a very inefficient format. Although this and the fact that SAS-XPT is a completely outdated format (stemming from the IBM mainframe era), the FDA requires us to use this outdated format for submissions. On the other hand, the FDA has over and over again added new derived variables in SDTM, introducing dangerous data redundancy and further enormously increasing the file sizes. In order to somewhat counteract these (by the FDA (unfounded) feared) large file sizes, they set a requirement in force that states that the length of a variable must exactly be the length of the longest value for that variable.

Modern information technology looks different...
displaying all the variables where the maximal length found during generation of the data deviates from the information in our underlying define.xml. For example, our define.xml still contained a (from the template) maximal length of 40, whereas the STUDYID always has the value "MyStudy" which is only 7 characters. Similarly, our "SUBJID" values are "001","002", etc., so never longer than 3 characters.

The user can now decide for which variables the lengths in the define.xml will be adapted, and as the define.xml is steering the generation of the SDTM files (as it should be), these will then also be the variable lengths in the SAS-XPT files. For text variables, this will usually make sense, for numeric variables, it will usually not be, as a numeric variable in SAS-XPT will always take 8 bytes\(^\text{10}\) anyway. The "Length" in the define.xml for numeric values is the length in characters for the number anyway.

\(^\text{10}\) Another waste of bytes ...

SDTM-ETL™: New features in v.3.2
So, let us select which variable lengths we want to adapt automatically, e.g.:

After executing the transformation, we can easily see that the length have indeed been adapted in the underlying define.xml. For example for the variable AETERM:

and in the SAS-XPT file (using the SAS-Viewer):
When using Dataset-XML as the output format, the same checkbox is found again, but in this case the issue is very less critical, as XML is not a fixed-length record, and is much more efficient with storing data than SAS-XPT is\(^{11}\). In the case of Dataset-XML however, the "Length" instructions in the define.xml can e.g. be used to generate a relational database (i.e. create SQL "CREATE TABLE" statements), where the "Length" is used to make a "VARCHAR\(length\)" statement. As "VARCHAR" is a very efficient storage format, it is absolutely not necessary to set the length to the maximum of the encountered values, at the contrary, one can then better set it somewhat larger than the longest encountered variable value length.

Anyway, this new feature easily allows to obey the (although ridiculous) FDAC036 rule, without needing to do a postprocessing step, as is usually needed when using other software packages.

Important remark: when more data become available during the study, you might need to re-use this feature, as the new data can contain values with a larger length then when you first used this new feature.

\(^{11}\) This doesn't mean that XML files are always smaller in size. XML usually uses UTF-8 (allowing non-ASCII) characters, and XML also contains the tags. In the past, we found that, for the same data, the XML files for normal domains are usually larger when using XML with respect to SAS-XPT, whereas datasets for "special" domains such as CO and supplemental qualifier domains, the SAS-XPT are considerably larger than the SAS-XPT files. Filesize shouldn't be an issue anyway however, as the amount of storage in a database is independent from the transport format, and the cost of storage of even the largest submissions is typically below 1 US$.
Retrieving page numbers from an annotated CRF in PDF format

Retrieving page number information from annotated Case Report Forms (aCRFs) has now become very easy using SDTM-ETL.

First, one of course needs to provide the location of the aCRF, which can be done using the menu "Insert - Link to annotated CRF":

leading to:

One gives the link a title (e.g. "Annotated CRF"), an ID (like "LF.aCRF"), and can then use the "Browse" button to add the location of the annotated CRF.

One can then later always change the information using the menu "Edit - Links to External Documents".
We have already learned how to add an "Origin" to an SDTM/SEND/ADaM variable, by selecting the cell in the table, and then using the menu "Edit - SDTM Variable Properties". For example for "STUDYID":

When then checking the "Edit Origin" checkbox, followed by clicking the "Edit" button, the "Origin" wizard will appear:

12 In case of using SEND, the menu is automatically adapted to "SEND Variable Properties".

SDTM-ETL™: New features in v.3.2
Depending on the origin type (Assigned, Protocol, Derived, Electronic Data Transfer, CRF), a number of fields will become disabled/enabled and to be filled or not. For example, when we state that the STUDYID is taken from the protocol, we just select "Protocol" and all other fields get disabled:

![Designing/Updating Origin for Item STUDYID](image1)

This also allows to add page numbers in the case of the CRF being the origin, and this in a very user-friendly way. Doing so for each variable (we have hundreds of them) is still a cumbersome task, which we want to automate when possible.

If we do already have an annotated CRF, the page numbers can be semi-automatically retrieved and the "Origin" elements populated and assigned to the variables, this as well to the normal domain variables as well as to value-level variables.

To do so, use the menu "Insert - CRF Page Numbers to Variable Origin". The following information is displayed:

![Message](image2)

Unlike other (black box) tools on the market, you will be guided through the process, allowing you to decide yourself which page numbers will be copied to the variables, and which not. Normally, PDF page numbers will only be retrieved for variables for which you already provided a
mapping. If your mapping is not complete yet, you can later repeat the process if you want - the software will tell you for which variables you do have already the page numbers assigned and for which not.

After clicking OK, the system is retrieving all information (this may take a few seconds), and the following dialog is displayed:

![Screen shot of dialog](image)

In each row, one sees the suggested variable, the page number or numbers, and the PDF annotation itself. Rows with a red text indicate that there is an "Origin" assignment yet, and one can easily find out what assignment is already present, by hovering the mouse over the row. A tooltip is then displayed:

SDTM-ETL™: New features in v.3.2  p.35/44
If the assignment were a CRF page or pages, also the latter will be displayed in the tooltip (see later).

As a user, you can now decide yourself which page numbers must be copied to the variable "Origin". For example, an annotation of "STUDYID" was found on page 6, but you want to keep the old assignment, stating that it comes from the protocol, so you would in this case not check the checkbox.

Suggestions are also generated for value-level variables. For example, near the bottom of the list we find:

stating that annotations were found on page 11 about value-level variables "HEIGHT" and "WEIGHT". It is also indicated that an "Origin" assignment was already present, so if we check these checkboxes, the old assignments will be overwritten.

One can now decide for which variables (domain or value-level variables) the page numbers are copied. For example, we check all of them, except for "STUDYID". The assignments are then automatically performed, but only for the variables for which the checkbox was checked.

If we later than inspect the "Origin" for e.g. the variable "QSORRES" (using the menu "Edit - SDTM Variable Properties" again, we find:
We can then still change this, e.g. add page numbers, of changing the list into a "first page - last page" in this case (just as an example) set it to pages 13-17:

When one later then repeats the whole process, all the variables that already were assigned an origin are marked as such (red color), for example:
As one can see, this new feature can save enormous amounts of time when fine-tuning the define.xml.
**Skipping display of the generated XSLT**

When starting executing the transformation (menus "Transform - Generate Transformation (XSL) Code for SAS-XPT" and "Transform - Generate Transformation (XSL) Code for Dataset-XML" the system will generate XSLT (Extensible Stylesheet Language Transformations) code that is then applied to the ODM with clinical data. Until now, this transformation code was always displayed to the user, also allowing to save the XSLT transformation code to file, e.g. for batch processing. Some users however do not always want to see this code, and asked us to introduce a new feature to allow to skip the display of the XSLT transformation code and immediately proceed to the dialog "Execute Transformation Code".

This is now possible.

In order to skip the presentation of the generate XSLT, use the menu "Options - Setting" and look for the new item "Skip display of generated XSLT":

![Options/Settings](image)

When this checkbox is checked, and the transformation code is generated, the system will directly proceed with the dialog "Execute Transformation Code".

SDTM-ETL<sup>TM</sup>: New features in v.3.2
**Additional date and time functions**

In SDTM, a lot of “timing variables” is present. About half of them are in principle unnecessary as they can easily be derived (also “on the fly”), and appear in the SDTM as redundant information (databases should not have any redundancy). It looks however as reviewers (or their tools) at the regulatory authorities are not capable of doing these “on the fly” derivations, which is the reason why they have been added.

To even make these derivations easier, a number of additional date and time functions have been added to SDTM-ETL v3.2. The complete list of all date and time functions (over 20 of them) is described in the document “Specification for the “SDTM -ETL” mini-scripting language”. We limit ourselves here to the ones that are new of them in SDTM-ETL 3.2:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>latestdate()</td>
<td>Returns the latest date in an array or series (array variable) of dates</td>
<td>$d = xpath(…) ; # returns an array $lastdate = latestdate($d);</td>
</tr>
<tr>
<td>earliestdate()</td>
<td>Returns the earliest date in an array or series (array variable) of dates</td>
<td>$d = xpath(…) ; # returns an array $early = earliestdata($d);</td>
</tr>
<tr>
<td>latestorequaldatebefore()</td>
<td>Returns the latest date before or equal to a reference date. Very useful in combination with RFSTDTC, RFXSTDTC, RFENDTC, ...</td>
<td>$lastdatebeforereferencedata = latestorequaldatebefore($d, $DM.RFSTDTC);</td>
</tr>
<tr>
<td>earliestorequaldateafter()</td>
<td>Returns the earliest date after or equal to a reference date. Very useful in combination with RFSTDTC, RFXSTDTC, RFENDTC...</td>
<td>$firststudyaedate = earliestorequaldateafter($aedates, $DM.RFSTDTC);</td>
</tr>
<tr>
<td>latestdatetime()</td>
<td>Returns the latest datetime in an array or series (array variable) of datetimes</td>
<td>$dt = xpath(…) ; # array of datetimes $lastdatetime = latestdatetime($dt);</td>
</tr>
<tr>
<td>earliestdatetime()</td>
<td>Returns the earliest datetime in an array or series (array variable) of datetimes</td>
<td># array of datetimes $dt = xpath(…) ; $early = earliestdatetime($dt);</td>
</tr>
<tr>
<td>latestorequaldatetimeafter()</td>
<td>Returns the latest datetime after or equal to a reference datetime. Very useful in combination with RFSTDTC, RFXSTDTC, RFENDTC...</td>
<td>$earliestdatetime = latestorequaldatetimeafter($d, $DM.RFSTDTC);</td>
</tr>
<tr>
<td>latestorequaldatetimebefore()</td>
<td>Returns the latest datetime before or equal to a reference datetime. Very useful in combination with RFSTDTC, RFXSTDTC, RFENDTC...</td>
<td>$latestdatetime = latestorequaldatetimebefore($d, $DM.RFSTDTC)</td>
</tr>
</tbody>
</table>
These functions also appear when the user clicks the button “More date and time ...” in the lower part of the script editor:

![Scripting Language Functions](image1.png)

Also remember that any XSLT2 date/time function (see e.g. [http://www.xsltfuntions.com/xsl/c0002.html](http://www.xsltfuntions.com/xsl/c0002.html) for a complete list) can always be used in our scripting language.
**Progress messages**

In the case of large input files, or complicated mappings, transformations can take some time. If the user wants to follow the progress of the transformation, he/she can choose to have intermediate messages shown during the transformation to see what the progress is. In order to do so, use “Options – Settings” and check the checkbox “Add progress messages to XSLT”:

![Progress messages checkbox](image)

The generated XSLT will then contain additional messaging statements that are executed during execution of the transformation. For example, one may find:

```
Messages and error messages:
Messages:
Starting iterating over variable USUBJID
Starting generating ItemGroupData for Domain: MyStudy:DM - Structure: One record per USUBJID
Starting generating ItemData for Variable: STUDYID
Finalizing generating Data for Variable: STUDYID
```

This will e.g. also help when the transformation fails, as one can then see until what stage the transformation went well.
**Extended messages for XSLT compilation and execution errors**

Mapping scripts may fail during execution. It is always important to understand why they failed in order to then be able to correct the scripts (debugging).

In order to make debugging easier, an additional option has been added in SDTM-ETL v.3.2. You can switch it on using the menu “Options – Settings” and then check “Show extended XSLT error messages”. A new field is then enabled allowing to define how many lines around the XSLT line that caused the error will be shown in the output:

![Check box for Show extended XSLT error messages](image)

If the value is set to “2”, this means that the line causing the error will be displayed, together with 2 lines before and 2 lines after the line causing the error. For example, one may see the following in the output when execution failed:

```
Messages and error messages:
The warning/error was caused by line 135 in the XSLT, in the following XSLT snippet:
133: <!-- Add the value calculated from the ODM2SDTM scripting language -->
134: <xsl:choose>
135: <xsl:when test="string($SUBJID) ne "">
136: <xsl:attribute name="Value">
137: <xsl:value-of select="$SUBJID"/>
```

Line 135 caused the execution to fail, and lines 133-137 are displayed. In case the option “Show extended XSLT error messages” has been checked, the system will also not only display the error message, but also try to explain it. In our case, the following line is:

```
Messages and error messages:
135: <xsl:when test="string($SUBJID) ne "">
136: <xsl:attribute name="Value">
137: <xsl:value-of select="$SUBJID"/>
Explanation: It was tried to put multiple values (array) into a single value string
```

stating that $SUBJID is not a single value (as was expected), but an array of values. As this error occurred during the generation of the DM dataset, the probable reason is that the subject ID was taken from a form that was repeating during the visit from which the demographics data was retrieved.

The original XSLT message however does not well explain the error. It is a well known error for XSLT specialists, but not well suitable for normal users.
Post-execution derivation of the new SDTM 1.5 –LOXBFL baseline flag

A new SDTM/SEND v.1.5 variable that has been added is the –LOXBFL “last observation before first exposure flag”. Again, this variable is essentially unnecessary, as any “smart” viewer tool such as the “Smart Dataset-XML Viewer” (also coming with the SDTM-ETL distribution) software can mark such baseline records “on the fly”, at least when observation dates and times and treatment exposure dates and times have been correctly (which unfortunately is not always the case).

When generating SDTM however, in some cases the derivation of –LOXBFL will require a postprocessing step, as all observation data need to be generated first before the system can decide which of them (by –TESTCD) is the last one before the first study treatment exposure.

When generating the SDTM datasets (using the menu “Transform – Generate transformation” followed by “Execute transformation”) one will also see that an extra checkbox has been added to the graphical user interface:

In case the checkbox is checked, a post-processing step will be performed, sorting all observations and for each unique test code (based on –TESTCD), the last observation (based on –DTC) before the first exposure (based on DM-RFXSTDTC) will be marked, and the value for –LOBXFL set to “Y”.

Remark that this is a feature that is meant for the future, as –LOBXFL is a new variable for SDTM v.1.5, and there currently is no Implementation Guide for that version. In SEND-IG v.3.1, –LOBXFL is not used at all.

Also remark that although this feature is available, the user can still decide to write his own mapping scripts for –LOBXFL.