Creating mappings for the AE domain

Now that we have created (and executed) mappings for several domains, let us create a mapping for the Adverse Events (AE) domain. If we take a look at our ODM study setup, we notice that there is an “Adverse Events” form, which is used (optionally) in the second visit. Usually, Adverse Event forms can be used in any visit, so we will do as if there were more visits defined in this study, and allow Adverse Event forms to have been used in any of the visits (StudyEvents).

A further look at the Adverse Event form shows that it is only partially suited for mapping with the SDTM. Ideally, the CDASH\textsuperscript{1} AE form should have been used, as it has an (almost) 1:1 mapping with the SDTM AE domain, but at the time of the current study design, this CDASH form was not available yet. This will happen often in the case of legacy studies.

To start, drag-and-drop the AE row in the SDTM table to the bottom of the table, this creating a study-specific instance of the AE domain. Accept the defaults for automatic generation of mappings for STUDYID, DOMAIN, USUBJID and AE.AESEQ:

The SDTM Implementation Guide tells us that we should have one record per adverse event per subject. We can translate this into “One record per AE.AETERM per USUBJID”, this as each ODM record has an “IT.AETERM” Item with the name “Conmed Indication” in the form „Adverse Events“. Please note that also other choices are possible, as long as the choice ensures that there is one record per AE form/subform used. We do however choose for AETERM as it is the SDTM topic variable.

Double click the first cell in our newly created row (“MyStudy:AE”). The Domain Editor panel shows up:

---

\textsuperscript{1} See the CDISC website at [http://www.cdisc.org/standards/cdash/index.html](http://www.cdisc.org/standards/cdash/index.html)
We select two levels of looping, the second being AE.AETERM. If we use define.xml 2.0, there is also a button „Set domain keys and sequence“. Clicking it pops up the dialog:

You can now add some variables using the „Add>>“ button and change the order with the „Move up“ and „Move down“ buttons. In earlier days, the SDTM-IG gave some suggestions for the keys, but most sponsors just copied these without further thinking\(^2\). You might e.g. choose for:

\[^2\text{It is however also not clear whether the FDA is doing anything with this information! The define.xml 2.0 specification states that adding domain keys in the context of a regulatory submission. To our knowledge, the FDA is not loading submission datasets in a database, so we wonder why these keys are necessary...}^\]
The next time you will edit the domain properties and hold the mouse over the button, you will find:

After having edited the domain properties, we can start with the mappings.

We do already have a mapping for USUBJID (automatically generated), so we must still generate a mapping for AE.AETERM. It is always good practice to first develop mappings for those SDTM variables that are looping variables (usually that is the topic variable), so let us do this one first. As already stated the ODM has an Item “Conmed Indication” with OID “IT.AETERM”. When we click in the SDTM cell for AE.AETERM, we even see that this Item is a “hot candidate” for the mapping (meaning that it has an annotation that its SDTM target\(^3\) is “AETERM”):

---

\(^3\) In the ODM, this is either accomplished by the „SDSVarName“ attribute on ItemDef, or by an „Alias“ element having „SDTM“ for the „Context“ attribute.
The square around the “traffic light” indicating that the item is a “hot candidate” for the SDTM variable “AETERM”:

If the item in the tree is not visible yet, we also can always find the “hot candidate“ using the menu “Navigate – Find hot SDTM Candidate“:

Just drag-and-drop the Item “Conmed Indication“ from the left side to the cell “AE.AETERM”. The following dialog shows up:
We accept the default that will retrieve the value from the clinical data („ItemData Value attribute“). As we want to take care that AE forms from all visits are taken into account (although in this study it is only used in the second visit, as there are only two visits), we need to check the checkbox “Generalize for all StudyEvents”. Later we will learn how to be more precise which visits exactly must be selected (unimportant in this case) using the „Except for ...“ and „Only for ...“ buttons. We are also warned that in the ODM, the value for „Conmed Indication“ may be up to 100 characters long, whereas our define.xml template has a default of max. 80 characters for this variable. When we check the checkbox „Set SDTM Variable Length to ODM ItemDef Length“, this will automatically be corrected, and the SDTM „Length“ will be set to 100 too.

When clicking OK, this correction is again confirmed:

The transformation script is now automatically generated:

A „mapping description“ is automatically created, which you might want to adapt. The information

---

4 It is NOT a good idea to set the length for all variables to the SAS-XPT maximum of 200, as this will generate extremely large SAS-XPT data sets that the FDA cannot handle. Also, setting a correct maximum length will help generating an optimized database (see „Generating an SDTM database“)
in it will go into the „Description“ element of the corresponding „MethodDef“ element in the underlying define.xml (in case define.xml 2.0 is used).

We can test it using the button “Test – Transform to XSLT”. We get something like:

![Test XSLT on ODM Clinical Data](image)

after having used the „Test XSLT on ODM Clinical Data“ button and having provided a file with clinical data for this study. We also see that some subjects have not have any adverse events (e.g. subjects 005, 006, 007). When later executing the mappings, no AE records will be generated for these subjects.

Define.xml 2.0 also requires to provide the „Origin“ of each SDTM variable when the context is a regulatory submission. To add it for AETERM, use the menu „Edit – SDTM Variable Properties“ and then look for the checkbox „New Origin“:

![Edit SDTM Variable Properties](image)

You can now add the „Origin“ by checking the checkbox followed by clicking the „Edit“ button. This gives:

If you use define.xml 1.0, you will see that „Origin“ is free text, and you will find a text field.
In this case, it is clear that the origin is „CRF“, so we need to check that checkbox:

We must associate a define.xml def:leaf with the annotated CRF, but we haven't got one yet, so we need to create one first by returning to the main window, and use the menu „Insert – Link to annotated CRF“:

6 If you have annotated your ODM study design well, this should essentially serve the purpose of the „annotated CRF“. However, the FDA does not accept this yet – you still need to provide an oldfashioned PDF, even in the case no paper was used in the study (e.g. in the case of eCRFs)
and use the „Browse“ button to add the location of your aCRF.
You can then add this information again to the „AETERM“ variable using the „Edit – SDTM Variable“ again:

and also add the page number to the field „Page list“

If you do not have an annotated CRF in PDF format yet, you can of course skip this step, and add the information later.

The second SDTM variable we want to develop a mapping for is AE.AEDECOD („Dictionary-derived term“, usually the MedDRA preferred term). As we haven't got anything better, we just use the same mapping as for AE.AETERM for the moment.

The “CDISC Notes” tell us that the sponsor need to give the dictionary name and version in the “Comments” column in the define.xml document. This can easily be done by selecting the AE.AEDECOD cell, and then use the menu “Edit -> SDTM Variable Properties”. We can then add the information in the input field for “Comment”:

7 If you have a MedDRA database or tool, we can help you generating a connection to it for mapping purposes.
The next SDTM variable we can develop a mapping for is the variable AE.AESEV (Severity/Intensity). When we select the cell, we see that there is an ODM “hot candidate” IT.AESEV („Severity“), which is however governed by a codelist.

As indicated by the square around the „traffic light“ and the blue color of it (indicating there is an associated codelist). To see the details of the codelist at the ODM side, select the Item „Severity“ and then use the menu “Navigate -> Select associated CodeList” (Ctrl-F2). The selection jumps to the associated CodeList in the ODM tree and expands its tree node:

We observe that there are four possibilities with coded values “1”, “2”, “3” or “4” (these can be seen in the status field at the bottom when selecting a “CodeListItem”, and decoded values “Mild”, “Moderate”, “Severe” and “Life Threatening”.

The SDTM-IG tells us that there is controlled terminology for AESEV and provides a reference to the NCI website. There we can find that there is a codelist „CL.C66769.AESEV“ which however only has 3 allowed values: „MILD“, „MODERATE“ and „SEVERE“. It also states that the codelist is non-extensible. Unfortunately the SDTM-IG is also not very clear on whether one must use that codelist. It states: „The severity or intensity of the event. Examples: MILD, MODERATE, SEVERE“. Especially, the wording „Examples“ is confusing.
We now have the choice between two approaches:

a) map the 4 possible values from the study design to the three values of the „recommended“ SDTM-IG codelist with only 3 values. If we do so, we should document this in the „reviewers guide“ but also in the „annotated CRF“

b) copy our own „SEVERITY“ codelist from the ODM (study design) to the SDTM, and use that.

We will use the first approach (only as an example).

Now just drag-and-drop the ItemDef “Severity” to the SDTM cell “AE.AESEV”. The following dialog shows up:

This time, we do NOT check the button „Set SDTM Variable Length to ODM ItemDef Length“ as we will use a different codelist on the SDTM site. Do not forget to check the box „Generalize for all Study Events“ if it was not automatically checked yet. Clicking „OK“ leads to:

---

8 In such cases, the mapper should decide himself / herself which of the 2 approaches is most appropriate
Where we select „Use CodeList from SDTM variable“ (which is the CDISC-NCI codelist). Then clicking „OK“ leads to:

which allows us to do the mapping between both the codelists.
As we know that „1“ stands for „Mild“, „2“ for „Moderate“ and „3“ for „Severe“, we just use the dropdowns leading to:
where we map „life threatening“ to „severe“.
For „Missing“ value, we just map to „blank“ - this will give a null value in the later result.
So we have:

The button „Attempt 1:1 mapping based on coded values“ can be used in case of long codelists. The system will then try to compare both and come with a suitable proposal for the mappings. After clicking „OK“, the „mapping editor“ shows up and the result is:
So the system has automatically generated a mapping script – no need for typing!
Of course one can always edit/adapt the mapping script, but in this case, this is not necessary at all.

Also note that comment lines (all starting with `#` have automatically been generated).

The next SDTM variable on our list is AE.AESER ("Is this a serious event?). The possible values are “Y” (Yes) and “N” (No). We see that there is no corresponding item in the ODM, so we will (arbitrarily) add “N” values for mild and moderate adverse events, and “Y” for severe and life threatening adverse events.

Drag-and-drop from IT.AESEV (ItemDef: Severity) to the SDTM cell AE.AESER. Check again to generalize for all visits (StudyEvents). We get:
We better choose the codelist from the SDTM Variable. Check the radiobutton and then click the button “Show Codelist Details”:

Remark that dependent on which version of CDISC-CT you have chosen when loading the template, you might see slightly different things here.

Again, the codelist mapping tool is then displayed. We map to:
Which leads to the automatically created mapping script:

```
# Mapping using ODM element ItemData with ItemOID IT.AESEV
# Generalized for all StudyEvents
# Using SDTM CodeList CL.NY
# Using a CodeList mapping between ODM CodeList CL.AESEV and SDTM CodeList CL.NY
$CODEVALUE = xpath('/StudyEventData/FormData[$FormID='FORM.AE']/ItemGroupData[$ItemGroupOID='IS.AE']/Ite
$NEWCODEVALUE = '';
if ($CODEVALUE == 1) {
   $NEWCODEVALUE = 'N';
} elsif ($CODEVALUE == 2) {
   $NEWCODEVALUE = 'N';
} elsif ($CODEVALUE == 3) {
   $NEWCODEVALUE = 'Y';
} elsif ($CODEVALUE == 4) {
   $NEWCODEVALUE = 'Y';
} else {
   $NEWCODEVALUE = 'U';
}
$AE.AESER = $NEWCODEVALUE;
```

Also remark the automatically created comment line, allowing us later to remember how the mapping script was generated. In case such an arbitrary decision is made, it is surely not a bad habit to explain that decision in the “Comments” field of the define.xml. You can do so using the menu “Edit => SDTM Variable Properties”, and add your comment in the field “Comment”:

For a regulatory submission, define.xml 2.0 also requires that the „Origin“ is provided, either on the variable level or on the valuelist level. We can easily add it using the „New Origin“ checkbox followed by „Edit“:

---

9 And which can also be read by the FDA reviewer when inspecting the “Methods” in the submitted define.xml file.
and in this case set it to „Derived“.

Note: when having finished mapping a domain / dataset, it is surely a good idea to check whether each variable has been assigned an „Origin“. The absence of it is NOT marked as an error even when using OpenCDISC, as the latter cannot know whether the context of the define.xml is a regulatory submission.

The next SDTM variable is AE.AEACN (Action taken with study treatment). When clicking the cell, we see that there is an ODM “hot candidate” “Actions take re study drug”. Alternatively, we can search for a „hot candidate“ on the ODM side using the menu „Navigate – Find hot SDTM Candidate“. We find:

Drag-and-drop, and generalizing for all StudyEvents gives us the dialog asking us which codelist to use. Again we can use the already associated SDTM codelist:
Clicking “OK” gives us the codelist mapping wizard, which easily leads us to the following mapping:

and again, the mapping script is automatically generated:
At this point, let us try to execute the full mapping that we have so far on a set of clinical data, just as a check that we are doing the right things. For this, use the menu “Transform - Generate Transformation (XSLT) Code for CDISC Dataset-XML” (alternatively you can also use „Generate Transformation (XSLT) Code for SAS-XPT“) followed by “Execute Transformation (XSLT) Code”:

```
The Transformation Script:

# Mapping using ODM element ItemData with ItemOID IT.AEACTIRI
# Generalized for all StudyEvents
# Using SDTM CodeList CL.ACN
# Using a CodeList mapping between ODM CodeList CL.AEACTTR and SDTM CodeList CL.ACN
$CODEVALUE = xpath://StudyEventData/FormData[@FormOID='FORM.AE']/ItemGroupData[@ItemGroupOID='IG.AE']/Item
$NEWCODEVALUE = '';
if ($CODEVALUE == '0') {
  $NEWCODEVALUE = 'DOSE NOT CHANGED';
} else if ($CODEVALUE == '1') {
  $NEWCODEVALUE = 'DRUG WITHDRAWN';
} else if ($CODEVALUE == '2') {
  $NEWCODEVALUE = 'DOSE REDUCED';
} else if ($CODEVALUE == '3') {
  $NEWCODEVALUE = 'DRUG INTERRUPTED';
} else {
  $NEWCODEVALUE = 'UNKNOWN';
}
$AE.AEACN = $NEWCODEVALUE;
```

Note that there is no checkbox „Split records > 200 characters to SUPP-- records“ as this is not necessary anymore when using Dataset-XML instead of SAS-XPT. We select a file with clinical data. As we want to inspect the results, we check the checkbox „View Results in Smart Dataset-XML Viewer“.

Then click “Execute Transformation on Clinical Data”. The mapping scripts are executed and soon the „Smart Dataset-XML Viewer“ shows up:
leading to:

There are a few records only, but the results seem to be OK.

We can compare directly by selecting the “ItemDef: Conmed Indication” in the ODM tree, and then use the menu “View - Clinical Data” which gives us all records for which an adverse event was reported:
The next SDTM variable we can tackle is AE.AEREL (Causality). The SDTM-IG states that there currently is no controlled terminology for this variable. For AEREL there is once again an ODM “hot candidate” (Relationship to study drug), so this is an easy mapping. Just for the exercise, we will this time use the ODM values, but as these are coded (coded values 0-3) we want to use the ODM decoded values (“None”, “Unlikely”, “Possible”, “Probable”). So in the dialog, we choose for “Use Study CodeList (from ODM) – Decoded Values”:

As our ODM has decoded values for different languages, the system now asks us for which language we want to retrieve the decoded values:

---

10 The SDTM-IG 1.4 states: „Records the investigator's opinion as to the causality of the event to the treatment. ICH E2A and E2B examples include NOT RELATED, UNLIKELY RELATED, POSSIBLY RELATED, RELATED. Controlled Terminology may be defined in the future. Check with regulatory authority for population of this variable“. Does this mean that the latter essentially depends on which reviewer you have?
We choose “english” (“en”). Again, the mapping script is automatically created:

leading to the following mapping script:

```python
if ($CODEVALUE == '0') {
    $TEMP = 'None';
} elsif ($CODEVALUE == '1') {
    $TEMP = 'Unlikely';
} elsif ($CODEVALUE == '2') {
    $TEMP = 'Possible';
} elsif ($CODEVALUE == '3') {
    $TEMP = 'Probable';
} else {
    $TEMP = '';  
}
$AE.AEVEL = $TEMP;
```

Now do the same for the SDTM variable AE.AEACNOTH (“Other action taken”). It can be mapped to ItemDef “Actions Taken – Other”. Once again, use decoded values from the ODM codelist.

For the SDTM variable AE.AEOUT, there is again an ODM “hot candidate” (ItemDef: Outcome). There is an ODM codelist associated, and at the SDTM side, there is already an associated codelist CL.OUT. The usual process leads us to the codelist mapping wizard:

```
<table>
<thead>
<tr>
<th>ODM CodeList Item</th>
<th>SDTM CodeList Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - Resolved, no residual effects</td>
<td>RECOVERED/RESOLVED</td>
</tr>
<tr>
<td>2 - Continuing</td>
<td>NOT RECOVERED/NOT RESOLVED</td>
</tr>
<tr>
<td>3 - Resolved, residual effects</td>
<td>RECOVERED/RESOLVED WITH SE...</td>
</tr>
<tr>
<td>4 - Death</td>
<td>FATAL</td>
</tr>
<tr>
<td>MISSING/INVALID VALUE</td>
<td>UNKNOWN</td>
</tr>
</tbody>
</table>
```

and the automatically created mapping script:
You can now also easily make similar mappings for the following SDTM variables: AE.AESDTH (Results in Death), AE.AEHOSP (Hospitalization), AE.AESLIFE (Life Threatening), and AE.AECONTRT (Concomitant or additional treatment given). In each case, the associated codelist is CL.NY.

Some hints for the ODM items that can be used and the ODM and SDTM codelists involved, are given in the following table:

<table>
<thead>
<tr>
<th>SDTM Variable</th>
<th>ODM Item</th>
<th>ODM Codelist</th>
<th>Define.xml codelist</th>
</tr>
</thead>
<tbody>
<tr>
<td>AESDTH (Results in Death)</td>
<td>Outcome (IT.AEOUT)</td>
<td>CL.AEOUT</td>
<td>CL.NY</td>
</tr>
<tr>
<td>AEHOSP (Requires or Prolongs Hospitalization)</td>
<td>Actions taken, other (IT.AECONTRT)</td>
<td>CL.AECONTRT</td>
<td>CL.NY</td>
</tr>
<tr>
<td>AESLIFE (Is Life Threatening)</td>
<td>Severity (IT.AESEV)</td>
<td>CL.AESEV</td>
<td>CL.NY</td>
</tr>
<tr>
<td>AECONTRT (Concomitant or Additional Trtmnt Given)</td>
<td>Actions taken, other (IT.AECONTRT)</td>
<td>CL.AECONTRT</td>
<td>CL.NY</td>
</tr>
</tbody>
</table>

For example, for AECONTRT (Concomitant or Additional Trtmnt Given) the mapping to the ODM „Actions taken, other“, can look like:

![Image of code list mapping between ODM and SDTM codelists](image-url)
Note that it depends on the structure and content of your CRF whether absence of additional treatment should be mapped to either „N“ or null for AECONTRT.

Executing the already available mappings using the CDISC Dataset-XML format then leads to:

**Timing variables**

The SDTM variables AESTDTC (AE start date), AEENDTC (AE end date), AESTDY (Study day of start AE), AEENDY (Study day of end AE), and AEDUR (AE duration) require somewhat more attention. For the first two, the information is available in the ODM (“Derived Start Date” and “Derived Stop Date”). The next two need to be calculated using e.g. the date of the first visit as study day 1. The last (AE duration) should only be submitted (according to the SDTM-IG) in case it was collected as a duration, so it should not be derived. However, just for the exercise, we will calculate it anyway.

In SDTM, dates should be given in ISO-8601 format (YYYY-MM-DD). If we look however at the ODM clinical data for e.g. “Derived Start Date”, we see that these are not in the required ISO-8601 format:

The ISO-format is YYYY-MM-DD, e.g. for the first date it should be: 2006-04-10.

However, due to the numerous script functions that we have available, it is pretty easy to make a transformation, after having performed drag-and-drop from the ODM item „Derived Start Date„ to the SDTM cell AESTDTC. For example:

11 It is named „derived“ as it comes from the combination of „start year“ (Item OID IT.AESTYR), „start month“ (IT.AESTMON) and „start day“ (IT.AESTDAY)
The first line retrieves the value from the ODM clinical data. The second line takes the first four characters from the retrieved value, the third line the next two characters, and the fourth line the last two characters. If you do not understand how the “substring” function works, just look for the button near the bottom of the screen, and hold the mouse over it. A tooltip shows up:

The fifth line in the script then sets up an if-else structure. In case there is a value for the date, it is reassembled from the individual parts using the dash as separator between year, month and day. The “else” part is necessary as otherwise we would get “--” in case there is a missing value for the date.

For the “Study day of AE start” (AE.AESTDY), we need to do some date handling. We will take the visit date of the first visit as study day 1 (see remark 12), and then calculate the value for AE.AESTDY from this and the AE start date. For the former, we can use drag-and-drop from the ODM, this although the datatypes of source and target do not match (red traffic light). We then get the following warning message:

In this case, we should have the checkbox “Generalize for all StudyEvents” unchecked! We can however also easily use the mapping selectors in the mapping editor:

12 In SDTM study days (relative to the reference start date) can either be positive values, or negative values, but NEVER 0. This always needs to be taken into account when calculating study days.
So we defined the first visit date as our reference day.

Now we define the AE start date, using the already defined variable $AE.AESTDTC. This may look strange at first, but we may always use previously defined (i.e. in a cell before the current one in the same row) scripting variable in “read” mode. So we write:

```
# calculate from first visit date and start of AE start date
# $FIRSTVISITDATE = xpath('/StudyEventData[@StudyEventOID='SE.VISIT0']/FormData[@FormOID='FORM.VISIT']/Items')
print 'FIRSTVISITDATE = ' $FIRSTVISITDATE;
# For the AE start date, we use the already defined variable!
$AESTARTDATE = $AE.AESTDTC;
print 'AESTARTDATE = ' $AESTARTDATE;
```

Also remark that we added some “print” statements for testing.

We know that both dates are in ISO-8601 format, so we can now use the “datediff” function to calculate the number of days between both dates, and from that, retrieve the study day of AE start:

```
print 'AESTARTDATE = ' $AESTARTDATE;
# calculate the difference in days
if($FIRSTVISITDATE != '' and $AESTARTDATE != '')
  $DIFF = datediff($AESTARTDATE,$FIRSTVISITDATE);
  $AE.AESTDY = $DIFF + 1;
else
  $AE.AESTDY = 1;
```

We need to take into account missing values for the AE start date. If we don't, the “datediff” function will fail and raise an exception (i.e. the execution of the script will crash). Also we need to add 1 to the result, as the first study day is day 1, and not day 0.
We need to take into account missing values for the AE start date. If we don't, the “datediff” function will fail and raise an exception (i.e. the execution of the script will crash). Also we need to add 1 to the result, as the first study day is day 1, and not day 0.

We can do similarly for the “Study day of AE end” (AE.AENDY):

Here again, we reuse (but read-only) SAE.AEENDTC, for the calculation of the study day.

There is even a simpler way that we will learn about a bit later. We will need „reference start date“ (from DM-RFSTDTC) and „reference end date“ (from DM-RFENDTC) very often (in almost each domain), so we will create some „subject global variables“ that we can then use over and over again in „read only“ mode. For more details, see the tutorial „creating and working with global variables“.

The last SDTM variable we want to provide a mapping for is AE.AEDUR (duration of the AE event). According to the SDTM, it should only be submitted if it was collected as a duration on the CRF. However, for the sake of the exercise, we will derive it here. As we do already have the start and end date, this is very easy, and we can use the datediff() function again. We only need to take missing values into account.

The script then looks like:
The three last lines take care that the date difference (in days) is put into ISO-8601 “duration” format.

Now, let us once again execute the mapping on a file with clinical data. We again use the menu “Transform -> Generate Transformation Code for CDISC Dataset-XML -> Execute Transformation Code”. The result as inspected using the „Smart Dataset-XML Viewer“ is:

Remark again that AEDUR should normally not be calculated during the mapping, but only be provided when it was collected on the CRF.
We did calculate it here just for the sake of the exercise.