Tutorial: Generating baseline flags

Table of Contents
Introduction.................................................................................................................................2
The simple and easy way: Baseline flag based on the Visit Name or Visit Number..................2
Getting the date/time of first study treatment........................................................................2
Making RFXSTDTC a "global" variable..................................................................................11
Generating the EX dataset......................................................................................................13
Creating the Vital Signs (VS) domain......................................................................................17
Calculating the baseline flag....................................................................................................31
Calculating baseline flags with exact dates and times...........................................................34
Alternatives..............................................................................................................................40
Introduction

Until SDTM 1.4 (SDTM-IG 3.2) the concept of Baseline Flag (--BLFL variables) is poorly defined. The IG states "Indicator used to identify a baseline value. The value should be Y or null", but doesn't state how it should be derived or assigned. Essentially this means that sponsors can assign or derive baseline flags in any way they want.

As of SDTM 1.5, a new baseline flag variable --LOBXFL "Last Observation before first Exposure Flag" has been introduced with the definition "Operationally-derived indicator used to identify the last non-missing value prior to RFXSTDTC. Should be Y or null."

Essentially, this variable should not be in SDTM, as such observations can easily be derived and highlighted by review tools. See e.g. the article "Why --LOBXFL should not be in SDTM".

In this tutorial, we will explain how such baseline flags can be derived. As of the date of writing the SDTM-IG v.3.3 is not final yet, we will use demonstrate this by the VSBLFL variable.

The simple and easy way: Baseline flag based on the Visit Name or Visit Number

We see that in many submissions, the baseline flag (or last observation before medication flag) is simply based on the visit (StudyEvent) name or number. This however assumes that:

- there is a single visit (e.g. "Screening") that has measurements and ends with the first study treatment (e.g. first drug intake)
- It is clearly indicated in the protocol that no measurements may be performed anymore within this visit after the subject received the first study treatment (e.g. first drug intake)
- Alternatively, there is a clearly protocol-defined visit with measurements that is the last visit before first study treatment
- The investigator conforms to the protocol and does not perform any measurements after first study treatment within the same visit of first study treatment

In such cases, the baseline flag can simply be set by a "drag-and-drop" from the ODM "StudyEvent", setting "Generalize for all StudyEvents" and getting the identifier (OID) of the visit and then adding it to a little script like:

```bash
$VISITOID = xpath(/StudyEventData/@StudyEventOID);
if ($VISITOID == 'SE.SCREEN') {
    $VS.VSBLFL = 'Y';
} else {
    $VS.VSBLFL = '';
}
```

This is very similar to getting visit number and name, as explained in the other tutorials.

Getting the date/time of first study treatment

First of all, we need to generate RFXSTDTC in DM ("Date/Time of First Study Treatment") which needs to be delivered in ISO-8601 format. RFXSTDTC can either be a date or a datetime. In many clinical studies, we (still) see that first exposure is captured as a date.

---

1 One SDTM-IG 3.3 is final, one can then simply replace VSBLFL by VSLOBXFL
For this tutorial, we need the sample ODM 1.3 file "MyStudyNew_ODM_1_3.xml". When loaded into SDTM-ETL, we can see the following tree structure:

We see that this simple demo study consists of 3 (types of) visits: a pre-treatment visit, a treatment visit and a post-treatment visit. When clicking on the tree node "Treatment", we also see that the visit is "repeating", i.e. that there may be more than one occurrence of this visit:

Furthermore, we quickly see that the form "Drug Exposure Form" has a "Treatment Exposure Date" and a "Treatment Exposure Time". These will be very useful later to populate the EX (and/or EC) dataset and for RFXSTDTC.

Before starting any mappings, we want to see what data points are available for the exposure dates and times. Select "Treatment Exposure Date" and then use the menu "View - ODM Clinical Data". Select your ODM file with clinical data. As the whole visit is repeating, also check the checkbox "Also display RepeatKeys".
After clicking the "View ODM Clinical Data" button, one gets:

seeing that subject 001 has two exposures, one of May 1st 2006, and one on May 2nd 2006. Also notice the third column "RepeatKey" which is the repeat number of the visit.

For RFXSTDTC, we of course only need the first.

Now for generating the mapping to RFXSTDTC, there are several possibilities

A. Using the mapping wizard

After creating a study-specific copy of the DM domain (by dragging the DM row from the template to the bottom of the table):

double-click the cell "RFXSTDTC". The mapping window is displayed:
The upper part allows us to select a visit (StudyEvent), Form, ItemGroup (subform) and Item. For the date of the first exposure, we know we need to retrieve the data from the first "treatment visit", so we select:

![Selection of StudyEvent and Rep.Key](image)

We see that the field "Rep.Key" is highlighted, as the "treatment visit" is repeating. As we need the first treatment visit, we need to fill "1" in this field:

![Filling Rep.Key with 1](image)

Later we will see that there is another way of ensuring that the date of exposure from the first visit is selected.

We then continue with selecting the form, subform and item, and assign a variable name (e.g. "DATE") for this variable:
Clicking the "Add to Script" button leads to:

Notice the @StudyEventRepeatKey='1' in the XPath expression.

Now to the same for "Treatment Exposure Time" and name the variable "TIME":

leading to:

In the mapping script, we can now easily create a datetime in ISO-8601 format, using the function "createdatetime" (in the box with all function):

and then complete the script to:
Testing leads to:

<table>
<thead>
<tr>
<th>STUDYID</th>
<th>DOMAIN</th>
<th>USUBJID</th>
<th>DM.RFXSTDTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyStudy</td>
<td>DM</td>
<td>001</td>
<td>2006-05-01T12:57:04</td>
</tr>
</tbody>
</table>

B. By drag-and-drop

We can also use simple drag-and-drop for generating the mapping script for RFXSTDTC.

Simply drag-and-drop "Treatment Exposure Date" to the RFXSTDTC cell. This leads to:
and clicking "OK" to the following mapping script:

```
# Mapping using ODM element ItemData with ItemOID II.EXPDATE
DM.RFXSTDTC = xpath://StudyEventData[@StudyEventOID='SE.Visit1']/FormData[@FormOID='FORM.EXPDATE']/ItemGroupData[]
```

When we however test this, we get:

```
Messages and error messages:
Messages:

Error Messages:
A sequence of more than one item is not allowed as the first argument of string() (@Value, @Value)
```

The reason is that the XPath retrieves ALL treatment dates, but we only want the first one, or better said, the one from the first visit.

We can easily solve this in several ways:

- add [@StudyEventRepeatKey='1'] to the XPath in the mapping script. This however already requires some knowledge of ODM and of XPath
- Manually edit the XPath expression like:
Notice the [1] at the end of the "StudyEventData" part

- Use the following construct:

```
$DATE = xpath('StudyEventData[@studyEventOID='SE.VISITA']/FormData[@FormOID='FORM.EXPOSURE']/ItemData[ItemOID][1]/IT.EXPDATE);
$FIRSTDATE = $DATE[1];
```

where first all treatment dates are retrieved, and then the first one encountered.

We can then do the same for the time of the first exposure, finally leading to the following script:

```
$TIME = xpath('StudyEventData[@studyEventOID='SE.VISITA']/FormData[@FormOID='FORM.EXPOSURE']/ItemData[ItemOID][1]/IT.EXPTIM);
$FIRSTTIME = $TIME[1];
```

Remark that this will only be correct when both date and time have been collected for the first exposure.

For RFXENDTDC (last date/time) of exposure, we can use the construct:

```
$LASTDATE = xpath('StudyEventData[@studyEventOID='SE.VISITA']/FormData[@FormOID='FORM.EXPOSURE']/ItemData[ItemOID][last()]/IT.EXPDATE);
$LASTTIME = xpath('StudyEventData[@studyEventOID='SE.VISITA']/FormData[@FormOID='FORM.EXPOSURE']/ItemData[ItemOID][last()]/IT.EXPTIM);
$LASTDATE = $LASTDATE[1];
$LASTTIME = $LASTTIME[1];
```

Notice the "[last()]" predicates. This construct is especially useful when we do not know the number of repeats of study treatments, for example when these differ between subjects.

However, …, suppose that we have subjects in the database (and so in the ODM) that have been included yet, but for which no exposure has been captured yet. If we keep the script as it is now, this will lead to a result of "T" for RFXSTDTC.

So we better do a check whether data is available, and if not, put an empty value.

This can e.g. be done by:
Remark the "print" statement which will generate a message during execution, and the usage of $USUBJID. The latter is possible as $USUBJID has been defined before $DM.RFXSTDTC.

The result is then e.g.: 

<table>
<thead>
<tr>
<th>STUDYID</th>
<th>DOMAIN</th>
<th>USUBJID</th>
<th>DM.RFXSTDTC</th>
<th>DM.RFXENDTC</th>
</tr>
</thead>
</table>

We see that for subjects 002 and 003, no first nor last exposure date/time have been found yet. In the messages box we indeed find:

<table>
<thead>
<tr>
<th>Messages and error messages:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messages.</td>
</tr>
<tr>
<td>no first exposure date found yet for subject 002</td>
</tr>
<tr>
<td>no last exposure date found yet for subject 002</td>
</tr>
<tr>
<td>no first exposure date found yet for subject 003</td>
</tr>
<tr>
<td>no last exposure date found yet for subject 003</td>
</tr>
</tbody>
</table>
Making RFXSTDTC a "global" variable

As we will need the values of RFXSTDC for deriving baseline flags, we need to make it a "global" variable\(^2\). For this, use the menu "Insert - Global Subject Variables Domain"

![Menu Screenshot]

leading to:

A "special" dataset has been added, allowing to define variables and mappings that can be reused in any other domain mapping. However, when executing the mappings, no "GLOBAL" dataset will be seen in the output, it is just for temporary usage.

We can now generate variables in "MyStudy:GLOBAL" like for any other domain, using the menu "Insert - New SDTM Variable", leading to:

![Variable Screenshot]

We give the new variable the name "RFXSTDTC". Do not name it "DM.RFXSTDTC" as the latter

\(^2\) The reason for this is that SDTM-ETL cannot "look" in datasets that have already been created and populated, it can only look in the source data.
is only for DM, whereas ours is global.
After filling "RFXSTDTC" in the field and clicking "OK" we get:

![Image of inserted global subject variable]

We can now just add a mapping for RFXSTDTC just like for any other variable. As we do already have a very similar mapping for DM.RFXSTDTC we can copy that one, and adapt it for the different variable name RFXSTDTC, i.e.:

```
and then repeat that for the global variable RFXENDTC.
```

![Image of designing mapping for SDTM variable]

It doesn't matter here whether RFXENDTC comes before or after RFXSTDTC.
Generating the EX dataset

In order to generate the EX dataset, drag-and-drop the EX row from the template to after the "MyStudy:DM" row. This leads to:

asking us whether the mappings for STUDYD, DOMAIN, USUBJID and EXSEQ can already be automatically generated. As these are obvious, we accept. We then see that these cells in the table are already "grayed", meaning that a mapping is available:

The most important variables in EX are "EXTRT" (the topic variable), EXDOSE (the numeric dose), and of course EXSTDTC (The date/time when administration of the treatment indicated by EXTRT and EXDOSE began) and EXENDTC (The date/time when administration of the treatment indicated by EXTRT and EXDOSE ended). The usual structure is "One record per protocol-specified study treatment, constant-dosing interval, per subject".

In the case we have only one constant dosing interval and one treatment (as in our study), this essentially reduces to "One record per subject". We need to indicate that in our domain structure by double-clicking the cell "MyStudy:EX"3:

We can decrease the number of iteration levels to 1, which leaves us with iterating only over the subjects (i.e. one record per subject)4:

---

3 Or by using the menu "Edit - SDTM Domain Properties"
4 In case of multiple treatment and dosing intervals, we will probably need the looping variables USUBJID,
Click "OK" to confirm.

We can now start adding the mappings. EXTRT and EXDOSE (and EXDOSEU) or EXDOSETXT are obvious as they come from the protocol and are fix (text) values. You can fill in the values simply e.g. as $EX.EXTRT = 'My Study Drug';

For EX.EXFREQ, there is an associated codelist, and we can ask for the values using the menu "View - SDTM associated CodeList5":

In our case, the protocol stated "daily" so we use "QD"6 leading to the mapping $EX.EXDOSFRQ = 'QD';

Let us now concentrate on EXSTDC (first exposure date/time) and EXENDTC (last exposure date/time). We will do the exercise for 2 use cases:

- only the exposure dates were captured
- both exposure dates and times were captured.

---

5 In this case, we used CDISC SDTM controlled terminology 2016-06-24, codelist CL.71113.FREQ.
6 Unfortunately, the CDISC controlled terminology comes without explanation of what "QD" means, so mappers do either need medical knowledge, or need to look up what these terms mean.
Of course, in some cases the database already contain first and last date(time) of exposure, but we want to do the exercise where all the exposure dates or datetimes are known\(^7\).

The first case (only exposure dates captured) is very common, but may lead to problems with assigning baseline flags. For example, suppose the first exposure date is 2006-05-02 and vital signs have been captured that same day, once a 09:02:55 and once at 23:12:55. Was the last vital signs test before or after first exposure? No way to find out from the data themselves. And if no times were given for the vitals signs, but only the date, the problem even becomes more difficult. We will demonstrate this in this tutorial.

For the first case, we will just use the dates. For EX.EXSTDC we look into our ODM tree:

and just drag-and-drop "Treatment Exposure Date" to EX.EXSDTC, leading to:

Executing the mappings then results in an error:

```
Messages and error messages:
no first exposure date found yet for subject 012
no last exposure date found yet for subject 012

Error Messages:
A sequence of more than one item is not allowed as the first argument of string() (@Value, @Value)
```

as there is more than 1 exposure date, i.e. the value of EX.EXSTDTC is an array instead of a single value. We can solve this in 2 ways.

The first is that we trust that the exposure dates in the ODM file are in chronological order. This is a requirement of the ODM standard: "Section 2.10: The elements in the data portion of an ODM file that apply to a single entity must occur in temporal order".

In that case we can simply adapt the script to:

\(^7\) Remark that this may mean that also an EC dataset (Exposure as collected) needs to be populated. This is especially important when treatments were missed or deviations from the constant dosing interval have been collected.
As of version 3.1 of the software, we also have new (experimental⁸) functions for selecting the earliest and latest date in a series of dates. These are:

- earliestdate(...)  
- latestdate(...)  
- earliestdatetime(...)  
- latestdatetime(...)

each taking an array of dates or datetimes respectively.

They do select the earliest or latest date or datetime in a series (array) of date or datetimes, independent of the order in which these appear.

So for example, for $EX.EXSTDTC we can use:

---

⁸ So you won't find them in the "buttons" at the bottom of the mapping editor yet. They are however fully supported. But they are not very well tested yet.
leading to exactly the same result as before, but now ensuring we really have the earliest date.

Later, we will see that some other useful new functions have been defined, like "latest date before or equal to a reference date", which will help us to generate baseline flags.

Creating the Vital Signs (VS) domain

We can start creating mappings for the VS domain by drag-and-drop from the template, this time to after the "MyStudy:EX" row.

Looking at the structure by double-clicking "MyStudy:VS" we find:

This is sufficient for now. Later we might need to add an additional iteration level.
Looking at our ODM, we find:

We see that no vital signs are taken during the "Pre-treatment" visit (at least not using the "Vitzal Signs Form"), but we also know that the "Treatment" visit is repeating. We also see however, when clicking "ItemGroupDef: Vital Signs", that this group is repeating, i.e. we can have repeats for all our vital signs within one and the same "Vital Signs Form".

We also see things like "Systolic Blood Pressure" and "Pulse" which are clearly vitals signs data points, but in the same group we also find "Vital Signs Date" which will probably later come in VSDTC.

First we will start doing the mapping for VS.VSTESTCD, as this is a "looping variable".9

So we need to take a bit of attention when using "drag and drop". Let us for example "drag and drop" "Systolic Blood Pressure" to VS.VSTESTCD:

---

9 It is advised to always first perform the mappings for "looping variables" as this makes testing much easier.
We do not want the value from the collected data, but we want to get the identifier of the test, and map that to CDISC controlled terminology for VSTESTCD. So we check the radiobutton "Import XPath expression for another ItemData attribute/subelement):

"ItemOID" seems to be a good choice, as it is the identifier for the data point.

As vital signs have only been collected during treatment visits, we do not need to check "Generalize for all StudyEvents" (in case of doubt, one can check it, this does not harm). We do however have that different items contain the test name and value, but with exceptions. So we check "Generalize for all Items".
and then use "Except for …" or "Only for …" to select which ones we do really want. Using "Only for …" this leads to:

We do not want to have "IT.VSDATE", "IT.VSTIM" and "IT.VSDATETIM" as test codes, and also not "IT.WTUNITS", but we want to generate test codes for all the others ("IT.SYSBP" to "IT.BMI"). So we leave the first three unchecked, and check all other ones:
Then clicking "OK" twice leads to:

![User Interface Image]

This will retrieve the identifiers (ItemOIDs), but of course, these are not the test codes themselves. The system knows this and asks us whether we want to use the mapping wizard for generating the 1:1 mappings. When clicking "Yes", the wizard is displayed:

![CodeList mapping Image]

with dropdowns on the right, which make it easy to do the mapping, e.g.:
The correct mapping can be easily achieved:

Clicking "OK" then leads to the mapping script:

which one can still edit manually.

Remark the third-last line "$NEWCODEDVALUE = 'NULL'" which is for identifiers that were

---

10 When using the dropdown, typing the first character often already leads to the correct VSTESCD - no need to scroll down ...
somehow missed. One can then of course still change the text 'NULL' in something else, like 'TO DO'.

Executing the mappings then gives:

![SDTM Tables](image)

looking pretty good for the moment\(^\text{11}\).

Let us now get some results "as captured" (VSORRES).

Just drag-and-drop from "Systolic Blood Pressure" again, but this time to VS.VSORRES, and keep the first choice "", as we now want the value of the data point, not the identifier:

![Import ItemDef: Systolic Blood Pressure - for SDTM Variable VS.VSORRES](image)

\(^\text{11}\) Also observe that the VSSEQ values are assigned automatically, as we use the default mechanism to generate them automatically.
we see that the "Generalize for all Items" is still checked and the "Only for" still shows 5 inclusions. Clicking "OK" then leads to:

<table>
<thead>
<tr>
<th>The Transformation Script</th>
</tr>
</thead>
<tbody>
<tr>
<td><code># Mapping using ODM element ItemDate with ItemOID IT.SYSBP</code></td>
</tr>
<tr>
<td><code># Generalized for all Items within the ItemGroup</code></td>
</tr>
<tr>
<td><code>@VS.VSORRES := xpath(/StudyEventDate[@@EventID='3E.VISIT2']/FormDef[@FormID='FORM.VS']/ItemGrou</code></td>
</tr>
</tbody>
</table>

ending with "@Value".

Execution of the mappings then shows:

![SDTM Tables](image)

Let us now also add "VSTEST" and "VSORRESU".

For the first, we can simply repeat the mapping as we did for VSTESTCD (using the "ItemOID"), and use the mapping wizard:

![CodeList mapping between a set of ODM items and SDTM CodeList Vital Signs Test N...](image)

For the second (Vital Signs Original Results Unit), we need to be a little bit more careful, as we only have units in the source data for "weight" (which can be either "pounds" or "kilograms"), but we of course also need to add them for "pulse rate" and for the "blood pressure". Probably these
were preprinted on the CRF.

Drag-and-drop "Weight Units" to VS.VSORRESU. The mapping wizard again helps us:

![Import ItemDef: Weight Units - for SDTM Variable VS.VSORRESU](image)

But we don't want to "generalize" anymore, as we will use "Weight units" only. So we uncheck the "Generalize for all Items" button:

![Import ItemDef: Weight Units - for SDTM Variable VS.VSORRESU](image)

and clicking "OK" leads to the following dialog:
The reason is that (unfortunately!) CDISC decided to associate an (although extensible) codelist to VSORRESU, making "original" "not original" anymore. As such, we need to map what was on the CRF as unit to a CDISC unit.

This leads to:

We can however reuse the value of VS.VSTESTCD, so that the script can be rewritten to:

```
# Mapping using ODM element ItemData with Item OID IT.WTUNITIS
# Using categorization as a CodeList is associated with the SDTM CodeList
# but no CodeList is associated with the ODM data
$CODEDVALUE = xpath(/StudyEventData[@StudyEventID='SE.VISITA']/FormDef[@FormOID='59.5DF']/ItemGroupData/ItemGroupDef[@OID='49.75DF']/ItemDef[@OID='IT.WTUNITIS']/@value); 
if($VS.VSTESTCD == 'PULSE' ) { 
  $VS.VSORRESU = 'beats/min';
} elsif($VS.VSTESTCD == 'SYSBP' or $VS.VSTESTCD == 'DIABP' ) { 
  $VS.VSORRESU = 'mmHg';
} elsif($VS.VSTESTCD == 'WEIGHT' ) { 
  $VS.VSORRESU = $CODEDVALUE;
} elsif($VS.VSTESTCD == 'BMI' ) { 
  $VS.VSORRESU = 'kg/m2';
} else { 
  $VS.VSORRESU = 'UNKNOWN';
}
```

UCUM is an international notation for units and mandatory for the use in electronic health records.
in the fourth line we just retrieve the "weight units" from the ODM and store it in $CODEDV ALUE.
Then we test on the test code, and when the value of VSTESTCD is "WEIGHT", then we pick up the value from "$CODEDV ALUE" again. In all other cases, we just assign the value from the CDISC controlled terminology, based on VSTESTCD. In case we cannot do such an assignment, we set "UNKNOWN". We could also assign "'", meaning that in any other case, there is supposed to be no unit for the test code\(^\text{13}\).

The result of the mapping execution is:

Looking very good.

"VISIT" and "VISITNUM" can easily be generated from the "StudyEventRepeatKey" of the StudyEvent. For example for VISITNUM, drag-and-drop "StudyEvent: Treatment" to the cell VISITNUM, and select to use "RepeatKey":

This will lead to:

\(\text{# Mapping using ODM element StudyEventData using value from attribute StudyEventRepeatKey}\)
\(\text{VS.VISITNUM = xPath}(/\text{StudyEventData}[\text{@StudyEventOID}=\text{"SE.VISITA"}]/\text{StudyEventRepeatKey})\)
and the result is:

<table>
<thead>
<tr>
<th>SUBJID</th>
<th>VS.VSSEQ</th>
<th>VS.VSTESTCD</th>
<th>VS.VSORRES</th>
<th>VS.VSORRESU</th>
<th>VS.VISITNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>1</td>
<td>SYSBP</td>
<td>101</td>
<td>mmHg</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>2</td>
<td>DIABP</td>
<td>67</td>
<td>mmHg</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>3</td>
<td>PULSE</td>
<td>63</td>
<td>beats/min</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>4</td>
<td>WEIGHT</td>
<td>88.1</td>
<td>Kg</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>5</td>
<td>BMI</td>
<td>25.5</td>
<td>Kg/m2</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>6</td>
<td>SYSBP</td>
<td>100</td>
<td>mmHg</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>7</td>
<td>DIABP</td>
<td>70</td>
<td>mmHg</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>8</td>
<td>PULSE</td>
<td>62</td>
<td>beats/min</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>9</td>
<td>WEIGHT</td>
<td>88</td>
<td>Kg</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>10</td>
<td>BMI</td>
<td>25.6</td>
<td>Kg/m2</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>11</td>
<td>SYSBP</td>
<td>108</td>
<td>mmHg</td>
<td>2</td>
</tr>
</tbody>
</table>

if we do not want to have the "Pre-treatment" visit have VISITNUM=0 (there are essentially no rules for this), we can change our script into:

```python
# Mapping using ODM element StudyEventData using value from attribute StudyEventRepeatKey
$TEMP = xpath(/StudyEventData[@StudyEventOID='SE.VISITA']/@StudyEventRepeatKey);
$VS.VISITNUM = number($TEMP) + 1;
```

Do not forget the "number(..)" as the outcome of $TEMP is a text, not a number yet.

This leads to:

<table>
<thead>
<tr>
<th>SUBJID</th>
<th>VS.VSSEQ</th>
<th>VS.VSTESTCD</th>
<th>VS.VSORRES</th>
<th>VS.VSORRESU</th>
<th>VS.VISITNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>1</td>
<td>SYSBP</td>
<td>101</td>
<td>mmHg</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>2</td>
<td>DIABP</td>
<td>67</td>
<td>mmHg</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>3</td>
<td>PULSE</td>
<td>63</td>
<td>beats/min</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>4</td>
<td>WEIGHT</td>
<td>88.1</td>
<td>Kg</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>5</td>
<td>BMI</td>
<td>25.5</td>
<td>Kg/m2</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>6</td>
<td>SYSBP</td>
<td>100</td>
<td>mmHg</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>7</td>
<td>DIABP</td>
<td>70</td>
<td>mmHg</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>8</td>
<td>PULSE</td>
<td>62</td>
<td>beats/min</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>9</td>
<td>WEIGHT</td>
<td>88</td>
<td>Kg</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>10</td>
<td>BMI</td>
<td>25.6</td>
<td>Kg/m2</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>11</td>
<td>SYSBP</td>
<td>108</td>
<td>mmHg</td>
<td>2</td>
</tr>
</tbody>
</table>

Unfortunately, CDISC has not published any recommendations yet for repeating visits (as e.g. very usual in oncology studies with cycles). So you will to develop your own strategy. Important to know is that the visit numbers do not need to be subsequent, so when your pre-treatment study is assigned VISITNUM=1, you are free to assign VISITNUM=101 to the first treatment visit.

For VISIT (Visit Name) we can do something similar, like:

```python
# Mapping using ODM element StudyEventData using value from attribute StudyEventRepeatKey
$TEMP = xpath(/StudyEventData[@StudyEventOID='SE.VISITA']/@StudyEventRepeatKey);
$VS.VISIT = concat('TREATMENT VISIT ', $TEMP);
```

leading to:
Remark that you will later also need to add the visit numbers and names to the trial visit dataset (TV), which might be some work in the case of repeating visits and e.g. there was one subject who had 67 repeats. For SDTM, these are still 67 different visits!

Also the value for VSDTC (date/time of collection) can easily be obtained, as it was on the CRF (item "Vital Signs Date"). In this part of the exercise, we will just use the date part, demonstrating the data quality issues this can bring.

Just drag-and-drop the Item "Vital Signs Date" to the cell "VS.VSDTC". Do not check any of the "Generalization" checkboxes. This leads to the script:

```
  Mapping using DM element ItemData with ItemOID IT.VSDATE
  @VS.VSDTC = xpath://StudyEventData[@StudyEventOID='ST.VISITA']/formData[@formOID='FORM.VS']/ItemGroupData[
    ItemGroupDef: Vital Signs
    ItemDef: Vital Signs Date
    ItemDef: Vital Signs Time
    ItemDef: Vital Signs Date Time
    ItemDef: Systolic Blood Pressure
  ]/@ItemData[@ItemOID='IT.VSDATE']/@Value);
```

and the execution result being:
When looking at DM, we also see:

with the first date of exposure being 2006-05-01.
So it will already be clear that the measurements on 2006-04-30 will not be baseline values.
Calculating the baseline flag

Let us now start the calculation of the baseline flag. For this we need to know all measurement dates for vital signs measurements, and compare these with the "date of first study treatment" for which we developed a mapping in the global variable RFXSTDTC.

So we need to find all measurements, and this for each different test code, the last measurement before the first treatment date, or on the first treatment date.

One can already see the data quality problem here: if our datetime of first exposure is 2006-05-01, we do not whether measurements on the same date were before first exposure. Of course it can be stated that all measurements on the date of first exposure need to be done before the first exposure, but this will not be clear from the data themselves.

For VSSTDTC we had the mapping:

```xml
# Mapping using ODM element ItemData with ItemOID IT.VSDATE  
$VS.VSDTC =  
    xpath(/StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']  
        /ItemGroupData[@ItemGroupOID='IG.VS']/ItemData[@ItemOID='IT.VSDATE']/@Value);  
```

as we iterate over all tests, this gives one value per row.

In order to find ALL measurement dates however, we need to adapt this XPath expression to:

```xml
xpath((//SubjectData[@SubjectKey=$USUBJID]/StudyEventData[@StudyEventOID='SE.VISITA']  
        /FormData[@FormOID='FORM.VS']/ItemGroupData[@ItemGroupOID='IG.VS']/ItemData[@ItemOID='IT.VSDATE']/@Value);  
```

Remark the `//SubjectData[@SubjectKey=$USUBJID]` in front. It selects all data for which the subject ID is equal to our value of USUBJID. As it starts with `//`, this time this is an absolute path in the ODM, and not relative to our looping variable.

We can now use this in our script for VS.VSBLFL:

```xml
<Mapping Description and Link to external Document>
   SCDM-ETL mapping for VS.VSBLFL
</Mapping>
```

we can check that this delivers more than one date, e.g. using:

14 In case we did not use the subject ID from the ODM for USUBJID, the expression will be a little bit more difficult, we could then e.g. create a "global" variable SUBJID for the ID in the ODM, and use that in the XPath expression instead of USUBJID.
and upon executing the mappings, we see the following message coming by:

```
Messages and error messages:
Messages:

   number of vital sign dates = 7
```

But in order to assign a baseline flag, we need to get the latest measurement or measurements before or on the first treatment date (which we stored in RFXSTDTC).

For this, a new function was introduced in SDTM-ETL v.3.1:

```
function latestorequaldatebefore($dates, $referencedate).
```

It takes a series of dates (array) as the first argument, and a reference date as the second argument, and returns a single date that is the lastest date before or on the reference date.

So let us use this in the mapping script:

```
The Transformation Script:
$ALLVSDATES = xpath://SubjectData[@SubjectKey='$SUBJID']/StudyEventData[@StudyEventOID='SE.1']
$COUNT = count($ALLVSDATES);
print 'number of vital sign dates = ' $COUNT;
print 'vital sign dates = ' $ALLVSDATES;
$VS.VSBLFL = $COUNT;
```

and when executing the script, we see the message:

```
Messages and error messages:
Messages:

   number of vital sign dates = 7
```

Unfortunately, CDISC has put the VSBLFL variable before VSDTC, not taking into account that one needs the second to calculate the first. So we cannot simply use $VS.VSDTC but need to retrieve it again. This can easily be done using "drag-and-drop" from "Vital Signs Date" again.

We then compare the value of VSDTC with the latest measurement date before first exposure:

15 See the document "Scripting Language Specification" v.3.1
Remark that we need the function "date()" in 5th line, as RFXSTDTC is a datetime, and not a date, and datetimes can essentially not be compared with dates without making assumptions for the case that the date part is equal.

The result is:

As one sees, all the measurements that were on the date of first exposure, have been marked with VSBLFL=Y, all earlier and later ones do not have the flag set.

One already sees the danger of such an approach: we here assume that all measurements on the date of exposure were done before the first treatment. If this was not the case, we have a serious data quality problem. Suppose e.g. that we have a blood pressure lowering agent, and there was a systolic blood pressure measurement with a value of 130 one hour after first study treatment. With our approach, this value would be marked with a baseline flag, whereas the reality is that the blood pressure was raised after the first treatment, maybe because of the treatment with the blood pressure lowering agent!

Sadly, this quality of clinical data is still often found in studies.
So, in order to have high quality data, one should not only capture the first exposure as a datetime (including the time part), but also capture the time part (and not only the date part) of any measurement, especially vital sign measurements.
Calculating baseline flags with exact dates and times

Especially when measurements are performed on the first date of treatment or exposure, it is of utmost importance to capture the date and time of the measurement exactly. Also the first treatment has to be captured exactly, so with date and time.

This section will demonstrate how baseline flags can be generated when such exact information is available. The final results will then be compared with those of the case that only dates are used.

As we already have done a lot of mapping work, we just create a second instance of our VS domain (for demonstration purposes) by copy-and-paste the row for "MyStudy:VS". This can either be done by drag-and-dropping this row, or by selecting it and then use the menu "Edit - Copy Domain/Dataset" followed by "Edit - Paste Domain/Dataset".

In both cases, the following dialog is displayed:

leading to an additional row in our SDTM table:

We will now start changing/editing some scripts for the new use case that we want to use datetime-s for all our measurements.

The first script we change is the one for VS.VSSTDTC. Instead of using item "Vital Signs Date" we use "Vital Signs DateTime".

We could also drag-and-drop both "Vital Signs Date" and "Vital Signs Time" and then combine them using the "createdatetime" function, but we will see later that for getting all measurement datetimes, we need "Vital Signs DataTime" anyway.

So, just drag-and-drop from "Vital Signs DateTime" to VS.VSDTC of row "VS.1" leads to:
where we select to "overwrite".
We will now have datetimes for the VSDTC instead of dates only.

As we do already have RFXSTDTC as global variable (datetime of first study treatment) as a datetime, we do not need to change anything there.

For the baseline flag, we need to make a few changes, as we do want to base it on datetimes instead of on dates alone.

First of all, we need to get all the measurement datetimes for the test corresponding to the current value of TESTCD. We will store these datetimes in an array variable $ALLVSDATETIMES.

So we first need to retrieve the value of TESTCD. The easiest is to just copy the first line of the script for VS.VSTESTCD into VS.VSBLFL and give the variable another name, e.g. "CUROID" (as it is the current test OID):

SCUROID = xpath(/StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']/ItemGroupData[@ItemGroupOID='IG.VS']/ItemData[@ItemOID='IT.SYSBP' or @ItemOID='IT.DIABP' or @ItemOID='IT.PULSE' or @ItemOID='IT.WT' or @ItemOID='IT.BMI']/@ItemOID);

Remark that we select the OID (@ItemOID) at the end, and that we again only accept codes for SYSBP, DIABP, PULSE, WT and HT and BMI.

For the variable $ALLVSDATETIMES (all vital sign datetimes for the current test), we drag-and-drop "Vital Signs DateTime" to VS.VSBLFL, and select to append it to the script with another name:
after clicking OK, this leads to:

Ensure that the "Generalize for all Items" is not checked as we really only want to retrieve information from "Vital Signs DateTime" and not from any other source variable.

However, this would not filter on the current test code: we only want to get the datetimes for the current observation, not from other types of observations. So we need to do something special. There is a simple wizard for this, which is enabled by checking "View/Edit XPath expressions":

This then leads to:
where the filters are again displayed. We do however require that we only get datetimes for which there really was a measurement for the current test (identified by CUROID). We can add this condition by editing the filter for "ItemGroupData":

or a bit cleared:

```
[@ItemGroupOID='IG.VS'](ItemData[@ItemOID=$CUROID and @Value]]
```

what we state here is that there MUST be a data point for the current test (CUROID) in the current group, and that there MUST be a value for it\(^{16}\).

After clicking OK, we get:

\[
\begin{align*}
\$CUROID &= \text{xpath}(:,:,StudyEventData[@StudyEventOID='SE.VISITA'])/FormData[@FormOID='FORM.VS']
\$ALLVSDATETIMES &= \text{xpath}(:,:,StudyEventData[@StudyEventOID='SE.VISITA'])/FormData[@FormOID=
\end{align*}
\]

\$ALLVSDATETIMES would now only give one date as it will be interpreted against the current looping variable (which is VS.VSTEMPCD). We do however want to have all datetimes for the current test and subject. We already added the condition for "the current test", but not yet for "the current subject". This can be easily done by adding "//SubjectData[@SubjectKey=$USUBJID]" at the beginning of the XPath expression\(^{17}\):

---

\(^{16}\) Remark that ODM does not allow the construct Value="" (i.e. the empty value). In such a case the whole ItemData must be absent.

\(^{17}\) In case one cannot use $USUBJID as it has been transformed from the original one in the ODM (e.g. study-site-comcatenation), one can always define a subject-global variable, e.g. "$ODMSUBJECTKEY", assign the ODM subject key to it, and then use that is
This is an important step, as one can otherwise obtain false positives.

Not a bad idea either to add some "print" statements for during testing:

<table>
<thead>
<tr>
<th>The Transformation Script</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&amp;CUROID = xpath(/StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']/ItemGroupData/@Item)</code></td>
</tr>
<tr>
<td><code>&amp;ALLVSDATETIMES = xpath(/SubjectData[@SubjectKey=$SUBJID]/StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']/ItemGroupData/@Item)</code></td>
</tr>
</tbody>
</table>

We can now also add the datetime we will be comparing to. As we made RFXSTDTC a global variable, we can read it out here:

<table>
<thead>
<tr>
<th>The Transformation Script</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print 'number of vital sign dates for this test = ' &amp;COUNT;</code></td>
</tr>
<tr>
<td><code>print 'vital sign dates = ' &amp;ALLVSDATETIMES;</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The Transformation Script</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$CUROID = xpath(/StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']/ItemGroupData/@Item)</code></td>
</tr>
<tr>
<td><code>$ALLVSDATETIMES = xpath(/SubjectData[@SubjectKey=$SUBJID]/StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']/ItemGroupData/@Item)</code></td>
</tr>
<tr>
<td><code>print 'number of vital sign dates = ' &amp;COUNT;</code></td>
</tr>
<tr>
<td><code>print 'vital sign dates = ' &amp;ALLVSDATETIMES;</code></td>
</tr>
<tr>
<td><code>$REFDATE = $RFXSTDTC;</code></td>
</tr>
</tbody>
</table>

So what we will be doing is compare the reference datetime (which is the first exposure datetime) with all the datetimes of observation for the current test, and retrieve the latest one from the array that is before or on the reference datetime.

There is a special function for this: `latestorequaldatetimebefore()`. It has two arguments, the first being the array with all datetimes, the second one being the reference datetime. So we add:

<table>
<thead>
<tr>
<th>The Transformation Script</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$LASTESTDATETIMEBEFOREFIRSTEXPOSURE = latestorequaldatetimebefore($ALLVSDATETIMES, $REFDATE);</code></td>
</tr>
<tr>
<td><code>print 'latest measurement date before or on first exposure date = ' &amp;LASTESTDATETIMEBEFOREFIRSTEXPOSURE;</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The Transformation Script</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print 'number of vital sign dates = ' &amp;COUNT;</code></td>
</tr>
<tr>
<td><code>print 'vital sign dates = ' &amp;ALLVSDATETIMES;</code></td>
</tr>
<tr>
<td><code>$REFDATE = $RFXSTDTC;</code></td>
</tr>
<tr>
<td><code>$LASTESTDATETIMEBEFOREFIRSTEXPOSURE = latestorequaldatetimebefore($ALLVSDATETIMES, $REFDATE);</code></td>
</tr>
</tbody>
</table>

We still need the current datetime of the observation, which is equal to the one from VS.VSTESTCD, so we can simply copy that mapping into our script. E.g.:
Just for good reference, it is:

\[ VSDTC = \text{xpath}(/\text{StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']/ItemGroupData[@ItemGroupOID='IG.VS']/ItemData[@ItemOID='IT.VSDATETIM']/@Value}); \]

When the last datetime of exposure (for current subject and test) is equal to our datetime (from VSDTC), then we need to set the baseline flag to "Y", in all other cases to null. So:

The Transformation Script:

```plaintext
$REFDATE = $RX5STIDC;
$LATESTDATETIMEBEFOREFIRSTEXPOSURE = latestequaldatetimebefore($ALLVSDATETIMES, $REFDATE);
print 'last test measurement datetime before or on first exposure date = ' $LATESTDATETIMEBEFOREFIRSTEXPOSURE;
$VSDTC = xpath//SubjectData[@SubjectKey=$USUBJID]/StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']/ItemGroupData[@ItemGroupOID='IG.VS']/ItemData[@ItemOID='IT.VSDATETIM']/$Value;
if($VSDTC == $LATESTDATETIMEBEFOREFIRSTEXPOSURE) {
$VS.VSBFLF = 'Y';
} else {
$VS.VSBFLF = '';
}
```

The final result when executing the script on the clinical data is:

<table>
<thead>
<tr>
<th>USUBJID</th>
<th>VS.VSEQ</th>
<th>VS.VSTIDC</th>
<th>VS.VSTERROR</th>
<th>VS.VSTRESU</th>
<th>VS.VSBFLF</th>
<th>VS.VSDATETIM</th>
<th>VS.VEVISIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>25</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>26</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>27</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>28</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>29</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>30</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>31</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>32</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>33</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>34</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>35</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>36</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>37</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>38</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>39</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>40</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>41</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>42</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>43</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>44</td>
<td>101</td>
<td>mmHg</td>
<td>2006-04-30T12:48:33</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(the VS.VSDTC column has been moved to the left for better clarity).

The first datetime of exposure for subject 001 was 2006-05-01T12:57:04 so at 12:57 on the first of May 2006. As one sees, for each of the unique tests SYSBP, DIAPB, WEIGHT PULSE and BMI, a single baseline value has been generated, which all (in this case) are on the same date of first exposure, but before the time of first exposure. All other measurements after 12:57 have not been marked with the baseline flag.

For good reference, here is the whole script again:

```plaintext
$CUROID = xpath(/StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']/ItemGroupData[@ItemGroupOID='IG.VS']/ItemData[@ItemOID='IT.SYSBP' or @ItemOID='IT.DIABP' or @ItemOID='IT.PULSE' or @ItemOID='IT.WT' or @ItemOID='IT.BMI']/@ItemOID);
print ' $CUROID = ' $CUROID;
$ALLVSDATETIMES = xpath//SubjectData[@SubjectKey=$USUBJID]/StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']/ItemGroupData[@ItemGroupOID='IG.VS']/ItemData[@ItemOID='IT.VSDATETIM' and @Value]/ItemData[@ItemOID='IT.VSDATETIM']/@Value);
```
$COUNT = count($ALLVSDATETIMES);
print 'number of vital sign dates = ' $COUNT;
print 'vital sign dates = ' $ALLVSDATETIMES;
$REFDATE = $RFXSTDTC;
$LATESTDATETIMEBEFOREFIRSTEXPOSURE = latestorequaldatetimebefore($ALLVSDATETIMES, $REFDATE);
print 'lastest measurement datetime before or on first exposure date = '
$LATESTDATETIMEBEFOREFIRSTEXPOSURE;
$VSDTC = xpath(/StudyEventData[@StudyEventOID='SE.VISITA']/FormData[@FormOID='FORM.VS']/ItemGroupData[@ItemGroupOID='IG.VS']/ItemData[@ItemOID='IT.VSDATETIM']/@Value);
# when the visit date equals the last measurement date before first exposure
if($VSDTC == $LATESTDATETIMEBEFOREFIRSTEXPOSURE) {
    $VS.VSBLFL = 'Y';
} else {
    $VS.VSBLFL = ''; 
}

Alternatives

An alternative for programming the baseline flag as explained here is to do a post-SDTM processing step, i.e. generate all the SDTM datasets without the baseline flags, and then do a post-processing step in which the datasets are "corrected", and the baseline flags calculated from the generated datasets.

Such a post-processing step is of course completely against the spirit of SDTM ("source data - no derived variables"), but this is sad truth.

We will implement such a post-processing step (as an optional feature) in SDTM-ETL 3.2. The current tutorial however shows that a correct baseline derivation does not really need such a post-processing step. Some users will however find it easier to execute such a step after dataset generation.