The Smart Dataset-XML Viewer: Adding own validation rules

Document last update: 2017-10-20

Introduction

The "Smart Dataset-XML Viewer" is not only used for displaying SDTM and ADaM datasets in Dataset-XML format, it is also used a lot for validation of SDTM datasets. As such almost all the SDTM, SEND and ADaM validation rules from as well CDISC, FDA and PMDA are already available - the "Open Rules for CDISC Standards" initiative.

This initiative is very interesting, as the validation software used by the FDA and PMDA is full of bugs and causes a very large of false positives. This is of course very disturbing for sponsors, who are often desperate about this, especially as there was no way so far to find out whether an error or warning is a false positive or whether it is a real error or warning. The software validation forum is full of such reports. So, as long as the FDA does not use other validation software, sponsors and service providers need a "second opinion" software that does not generate such false positives. The "Open Rules for CDISC Standards" provides such an alternative.

Also problematic is that bugs in the validation software used by FDA and PMDA are not fixed, or only after 2-3 years. Our validation rules RESTful webservices however allow to provide updates of the rules implementations within hours after a reported bug (which hasn't happened so far…)

Generating a company-specific rule in XQuery

If you do not know XQuery yet, it is easy to learn. An excellent source is W3Schools. My students at the university learn XQuery in just a few hours.
When you then also use a native XML database (and store your submissions in it) like eXist or Oracle Berkeley DB, or BaseX, you usually also get a graphical user interface allowing to develop and test the rule in a very user-friendly way. In our case, we used eXist native XML database with its eXide graphical user interface.

The rule we will develop here comes from a forum entry by user Manolya, who was asking the makers of the FDA/PMDA validation software how to add a new rule about IE-DM crosschecking. The answer was essentially "you can't do it and we won't do it". Very unsatisfactory indeed…

Essentially, what user Manolya was asking for was how to add a rule that when there is a record in IE for a subject, it is checked whether the subject was assigned to an arm in DM. This could either be an error, or it can be due that the subject was assigned a so-called "waiver" to participate in the study, although he or she did not meet one or more of the inclusion/exclusion criteria.

In our test submission, which resides in the container /db/fda_submissions/cdisc01 in the XML database, we look for the IE dataset (this can easily be done using the administrative features of the database). We find:

```
<ItemGroupData ItemGroupOID="IG.IE" data:ItemGroupDataSeq="1">
<ItemData ItemOID="IT.STUDYID" Value="CDISC01"/>
<ItemData ItemOID="IT.IE.DOMAIN" Value="IE"/>
<ItemData ItemOID="IT.USUBJID" Value="CDISC01.200005"/>
<ItemData ItemOID="IT.IE.IESEQ" Value="1"/>
<ItemData ItemOID="IT.IE.IETESTCD" Value="INCL03"/>
<ItemData ItemOID="IT.IE.IETEST" Value="Did not respond to a standard course of medication Abc"/>
<ItemData ItemOID="IT.IE.IECAT" Value="INCLUSION"/>
<ItemData ItemOID="IT.IE.IEORRES" Value="N"/>
<ItemData ItemOID="IT.IE.IESTRESC" Value="N"/>
<ItemData ItemOID="IT.IE.VISITNUM" Value="1"/>
<ItemData ItemOID="IT.IE.VISIT" Value="SCREEN"/>
<ItemData ItemOID="IT.IE.IEDTC" Value="2003-09-15"/>
</ItemGroupData>
```

i.e. there is a record for subject with USUBJI=CDISC01.200005

The record in DM for this subject is:

```
<ItemGroupData ItemGroupOID="IG.DM" data:ItemGroupDataSeq="5">
<ItemData ItemOID="IT.STUDYID" Value="CDISC01"/>
<ItemData ItemOID="IT.DM.DOMAIN" Value="DM"/>
<ItemData ItemOID="IT.USUBJID" Value="CDISC01.200005"/>
<ItemData ItemOID="IT.DM.SUBJID" Value="200005"/>
<ItemData ItemOID="IT.DM.SITEID" Value="200"/>
<ItemData ItemOID="IT.DM.BRTHDTC" Value="1937-02-22"/>
<ItemData ItemOID="IT.DM.AGE" Value="66"/>
<ItemData ItemOID="IT.DM.AGEU" Value="YEARS"/>
<ItemData ItemOID="IT.DM.SEX" Value="F"/>
<ItemData ItemOID="IT.DM.RACE" Value="WHITE"/>
<ItemData ItemOID="IT.DM.ETHNIC" Value="NOT HISPANIC OR LATINO"/>
<ItemData ItemOID="IT.DM.ARMCD" Value="SCRNFAIL"/>
<ItemData ItemOID="IT.DM.ARM" Value="Screen Failure"/>
<ItemData ItemOID="IT.DM.COUNTRY" Value="USA"/>
</ItemGroupData>
```

With assigned arm code being "SCRNFAIL" (Screen Failure)

For testing purposes, we change this into:

```
<ItemGroupData ItemGroupOID="IG.DM" data:ItemGroupDataSeq="5">
<ItemData ItemOID="IT.STUDYID" Value="CDISC01"/>
<ItemData ItemOID="IT.DM.DOMAIN" Value="DM"/>
<ItemData ItemOID="IT.USUBJID" Value="CDISC01.200005"/>
<ItemData ItemOID="IT.DM.SUBJID" Value="200005"/>
<ItemData ItemOID="IT.DM.SITEID" Value="200"/>
<ItemData ItemOID="IT.DM.BRTHDTC" Value="1937-02-22"/>
<ItemData ItemOID="IT.DM.AGE" Value="66"/>
<ItemData ItemOID="IT.DM.AGEU" Value="YEARS"/>
<ItemData ItemOID="IT.DM.SEX" Value="F"/>
<ItemData ItemOID="IT.DM.RACE" Value="WHITE"/>
<ItemData ItemOID="IT.DM.ETHNIC" Value="NOT HISPANIC OR LATINO"/>
<!--ItemData ItemOID="IT.DM.ARMCD" Value="SCRNFAIL"/-->
<ItemData ItemOID="IT.DM.ARMCD" Value="WONDER10"/>
<ItemData ItemOID="IT.DM.ARM" Value="Screen Failure"/>
<ItemData ItemOID="IT.DM.COUNTRY" Value="USA"/>
</ItemGroupData>
```

i.e. we assign the arm code to be "WONDER10" which is a medication arm. This essentially means that the subject should not be in the study, as he/she did not pass the inclusion criterium.

In our graphical editor, we can now start generating the rule that in such case, an informational message is generated. The first lines are:

```
(: My own company rule -
 Check IE records for which the subject was assigned to an arm in DM (not SCRNFAIL, not NOTASSGN) :)
xquery version "3.0";
declare namespace def = "http://www.cdisc.org/ns/def/v2.0";
declare namespace odm="http://www.cdisc.org/ns/odm/v1.3";
declare namespace data="http://www.cdisc.org/ns/Dataset-XML/v1.0";
declare namespace xlink="http://www.w3.org/1999/xlink";
(: "declare variable ... external" allows to pass $base and $define from an external programm :)
declare variable $base external;
declare variable $define external;
(: get the define.xml :)
let $base := '/db/fda_submissions/cdisc01/'
let $define := 'define2-0-0-example-sdtm.xml'
let $definedoc := doc(concat($base,$define))
```

Where we declare the namespaces used (as we work with XML), and declare where the define.xml document can be found in the database.
I also added two lines with "declare variable … external" which will later, when using the rule in the "Smart Dataset-XML Viewer", allow us to pass the location from the define.xml from within an external system (our viewer).

In the next lines, we locate the DM dataset from the information in the define.xml ("def:leaf"), and also the ODM OIDs of the variables "USUBJID" and "ARMCD" in the DM dataset. This looks a bit like more complicated code, but even I do usually copy-paste it and adapt it for the variable I need.

```
(: get the DM dataset :)
let $dmitemgroupdef := $definedoc//odm:ItemGroupDef[@Name='DM']
let $dmdatasetlocation := $dmitemgroupdef/def:leaf/@xlink:href
let $dmdatasetdoc := doc(concat($base,$dmdatasetlocation))
(: we also need the OID of the USUBJID and ARMCD variables in DM :)
let $dmusubjidoid := (
    for $a in $definedoc//odm:ItemDef[@Name='USUBJID']/@OID
    where $a = $dmitemgroupdef/odm:ItemRef/@ItemOID
    return $a
)
let $dmarmcdoid := (
    for $a in $definedoc//odm:ItemDef[@Name='ARMCD']/@OID
    where $a = $dmitemgroupdef/odm:ItemRef/@ItemOID
    return $a
)
```

In the next lines, we query for records in the IE dataset, and iterate over them:

```
(: get the IE dataset :)
for $itemgroupdef in $definedoc//odm:ItemGroupDef[@Name='IE']
    (: get the location of the IE dataset from the define.xml :)
    let $datasetlocation := $itemgroupdef/def:leaf/@xlink:href
    let $datasetdoc := doc(concat($base,$datasetlocation))
    (: we also need the OID of the USUBJID variable in IE :)
    let $usubjidoid := (
        for $a in $definedoc//odm:ItemDef[@Name='USUBJID']/@OID
        where $a = $itemgroupdef/odm:ItemRef/@ItemOID
        return $a
    )
    (: iterate over all the records in the IE dataset :)
    for $record in $datasetdoc//odm:ItemGroupData
```

We first iterate over all "IE" datasets defined in the define.xml (you never know whether there will be more than one – "splitted" datasets), and then get the dataset location. We then get the ODM OID of the variable USUBJID.
When we have it, we start iterating over all records in the IE dataset:

```
    (: iterate over all the records in the IE dataset :)
    for $record in $datasetdoc//odm:ItemGroupData
        let $recnum := $record/@data:ItemGroupDataSeq
        (: get the value of USUBJID in IE :)
        let $ieusubjid := $record/odm:ItemData[@ItemOID=$usubjidoid]/@Value
        (: now look up this subject in DM :)
        let $dmrecord := $dmdatasetdoc//odm:ItemGroupData[odm:ItemData[@ItemOID=$dmusubjidoid and @Value=$ieusubjid]]
        let $dmrecnum := $dmrecord/@data:ItemGroupDataSeq
        (: check the value of ARMCD in that record.
        If it is NOT 'SCRNFAIL' and NOT 'NOTASSGN' give an informational message :)
        let $dmarmcd := $dmrecord/odm:ItemData[@ItemOID=$dmarmcdoid]/@Value
```

We get the record number in the IE dataset (for later reporting), and then search for the record for that subject in the DM dataset. We then retrieve the value for that subject for ARMCD. In normal cases (subject not included in the study) the value for ARMCD should be "SCRNFAIL" (Screen Failure) or maybe "NOTASSGN" (Not Assigned)

In the final lines, we write an informal message when the value for ARMCD is <u>not</u> "SCRNFAIL" and <u>not</u> "NOTASSGN". In this case, I choose to make it an "info" message, as we do not know whether there was a "waiver" for that subject.

```
    (: iterate over all the records in the IE dataset :)
    for $record in $datasetdoc//odm:ItemGroupData
        let $recnum := $record/@data:ItemGroupDataSeq
        (: get the value of USUBJID in IE :)
        let $ieusubjid := $record/odm:ItemData[@ItemOID=$usubjidoid]/@Value
        (: now look up this subject in DM :)
        let $dmrecord := $dmdatasetdoc//odm:ItemGroupData[odm:ItemData[@ItemOID=$dmusubjidoid and @Value=$ieusubjid]]
        let $dmrecnum := $dmrecord/@data:ItemGroupDataSeq
        (: check the value of ARMCD in that record.
        If it is NOT 'SCRNFAIL' and NOT 'NOTASSGN' give an informational message :)
        let $dmarmcd := $dmrecord/odm:ItemData[@ItemOID=$dmarmcdoid]/@Value
        where not($dmarmcd='SCRNFAIL') and not($dmarmcd='NOTASSGN')
        return <info rule="MyCompanyRule001" dataset="IE"
                rulelastupdate="2017-10-20" recordnumber="{data($recnum)}">Subject {data($ieusubjid)} found in IE dataset
who was assigned to arm {data($dmarmcd)} (record {data($dmrecnum)} in DM dataset)</info>
```

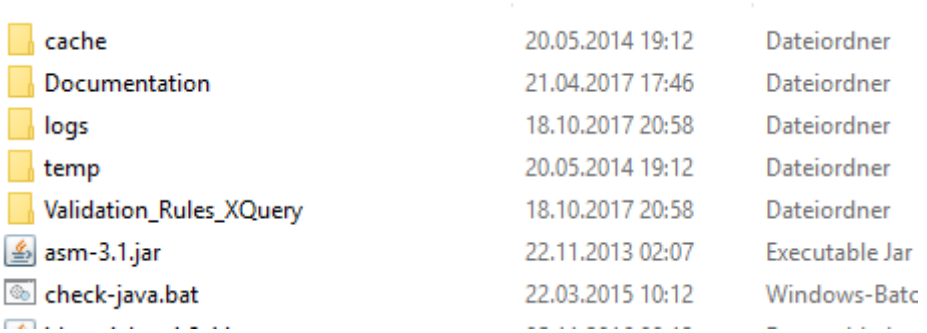
All together, it took me about 20 minutes to write this rule and test it.
When I run the XQuery rule, the output is:

```
<info rule="MyCompanyRule001" dataset="IE" rulelastupdate="2017-10-20"
recordnumber="1">Subject CDISC01.200005 found in IE dataset who was
assigned to arm WONDER10 (record 5 in DM dataset)</info>
```

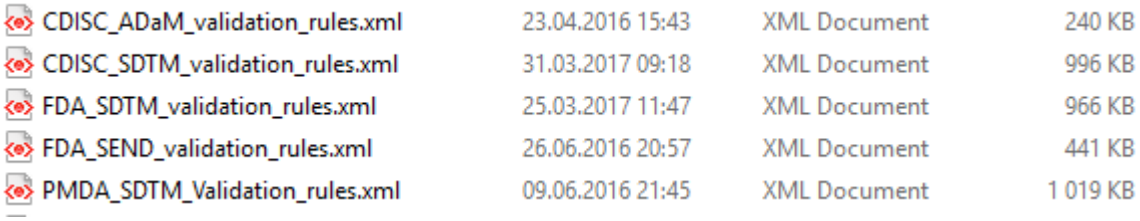
Making the rule available to the "Smart Dataset-XML" software

In order to make your own company validation rules available to the "Smart Dataset-XML Viewer", only a few more steps are necessary.

First navigate to the directory "Validation_Rules_XQuery":



Within that directory, you will already find the following files:



These are all XML files, each containing a set of rules. By convention, if the file name contains "SDTM", the "Smart Dataset-XML Viewer" software will regard it as a file with SDTM rules. Similar conventions apply for files with SEND and ADaM rules.

One can now use one of these files as a template for the new file with our own rules. In our case I will name it "MyCompany_SDTM_validation_rules.xml". The "root" element of the file is "sdsrules", with an attribute "last-update" providing the last update date in ISO-8601 format.
For each company-internal rule, now add an element "sdsrule" (you can use NotePad++ or an XML editor) and provide the attributes as follows:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <sdsrules last-update="2017-10-20">
3       <sdsrule originator="MyCompany" id="MC001" standard="SDTM" last-update="2017-10-20">
4           <ruledescription>Record in IE with DM.ARMCD not being SCRNFAIL nor NOTASSGN</ruledescription>
5           <domain>IE</domain>
6           <rulexquery><![CDATA[
7
8   ]]></rulexquery>
9       </sdsrule>
10  </sdsrules>
11
```

Add an "originator" attribute pair (e.g. containing your company name or department), the date of the last update of the rule in the "last-update" attribute, the name of the standard ("SDTM", "SEND" or "ADaM").

Then add an element "ruledescription" with a description of the rule, the domain the rule is about in the "domain" element (you can have several).
The next element then is "rulexquery" which will contain the XQuery itself. As the XQuery is essentiall plain text that can contain special charactes, be sure that the whole script starts with "<![CDATA[" and ends with "]]>", at best on a separate line.

Then just copy the XQuery in the XML structure as follows:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <sdsrules last-update="2017-10-20">
3      <sdsrule originator="MyCompany" id="MC001" standard="SDTM" last-update="2017-10-20">
4          <ruledescription>Record in IE with DM.ARMCD not being SCRNFAIL nor NOTASSGN</ruledescription>
5          <domain>IE</domain>
6          <rulexquery><![CDATA[
7  (: My own company rule -
8   Check IE records for which the subject was assigned to an arm in DM (not SCRNFAIL, not NOTASSGN) :)
9  xquery version "3.0";
10 declare namespace def = "http://www.cdisc.org/ns/def/v2.0";
11 declare namespace odm="http://www.cdisc.org/ns/odm/v1.3";
12 declare namespace data="http://www.cdisc.org/ns/Dataset-XML/v1.0";
13 declare namespace xlink="http://www.w3.org/1999/xlink";
14 (: "declare variable ... external" allows to pass $base and $define from an external programm :)
15 declare variable $base external;
16 declare variable $define external;
17 (: get the define.xml :)
18 let $base := '/db/fda_submissions/cdisc01/'
19 let $define := 'define2-0-0-example-sdtm.xml'
20 let $definedoc := doc(concat($base,$define))
21 (: get the DM dataset :)
```

Almost ready!

In our script, the location of the define.xml was set to be in the native XML database, but for use with the "Smart Dataset-XML Viewer" it will be passed from within the software. So all we need to do is to "comment out" the lines 18 and 19 that reference the database.
Comments in XQuery start with "(:" and end with ":)", so:

```
15 declare variable $base external;
16 declare variable $define external;
17 (: get the define.xml :)
18 (: let $base := '/db/fda_submissions/cdisc01/' :)
19 (: let $define := 'define2-0-0-example-sdtm.xml' :)
20 let $definedoc := doc(concat($base,$define))
21 (: get the DM dataset :)
```

Lines 15 and 16 now define that $base and $define are "external", so will be passed to the script by the "Smart Dataset-XML" software.

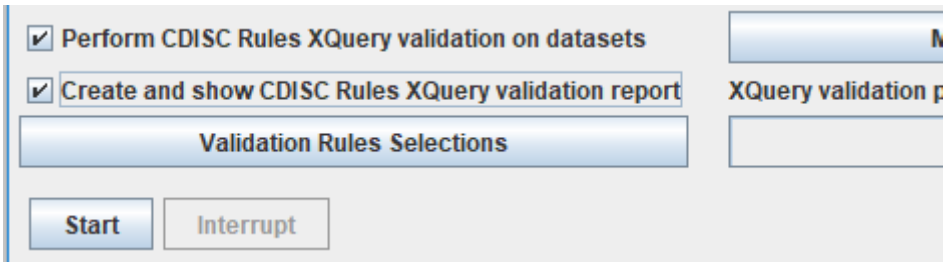
The bottom of the rule embedded in the XML then looks like:

```
56         If it is NOT 'SCRNFAIL' and NOT 'NOTASSGN' give an informational message :)
57         let $dmarmcd := $dmrecord/odm:ItemData[@ItemOID=$dmarmcdoid]/@Value
58         where not($dmarmcd='SCRNFAIL') and not($dmarmcd='NOTASSGN')
59         return <info rule="MyCompanyRule001" dataset="IE"
60                 rulelastupdate="2017-10-20" recordnumber="{data($recnum)}">Subject {data
61 ]]></rulexquery>
62     </sdsrule>
63 </sdsrules>
```

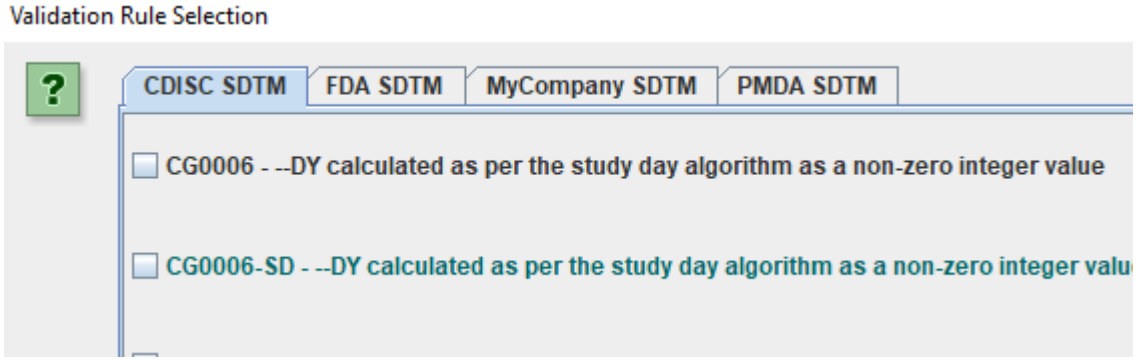
Save your work and ensure the XML file with your own company rules is in the directory " Validation_Rules_XQuery "

Running the rules from within the "Smart Dataset-XML Viewer"

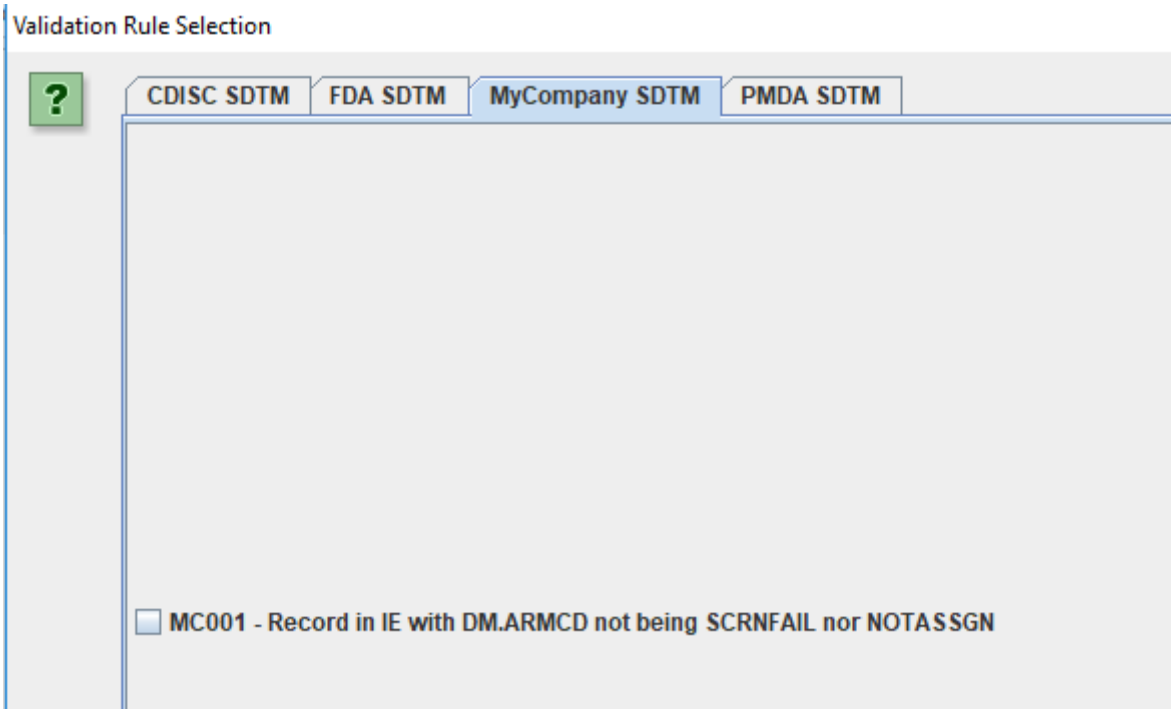
When starting the "Smart Dataset-XML Viewer" and first checking the checkbox "Perform CDISC Rules XQuery validation on datasets". Also check the checkbox "Create and show CDISC Rules XQuery validation report".
To select which rules to activate, click the button "Validation Rules Selection":



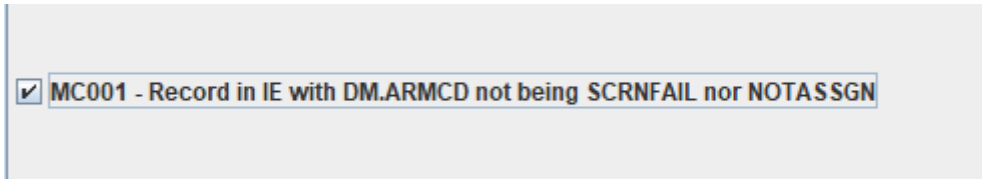
a new screen is displayed with the sets of rules:



also containing a tab "MyCompany SDTM", which, when selected shows:



One can then set the company-internal developed rule to "active" by checking the checkbos "MC001":

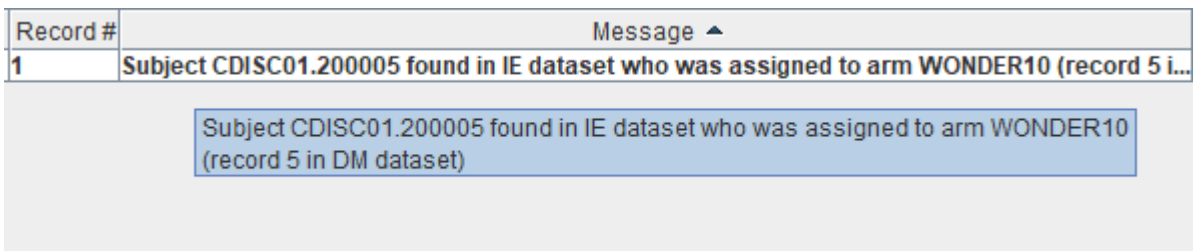


Also take care that the IE and DM datasets are really loaded:

And now click the "Start" button.

After the tables with the data is displayed, also a "validation report" table shows up:

Detailed view:



The validation report table can also be saved to an XML file – use the button "Store messages to file". We choose for XML as it can be read by other software tools as well as spreadsheet programs. The contents look like:

```
<XQueryValidationMessages CreationDateTime="2017-10-20T15:06:20.120+02:00">
<info rule="MyCompanyRule001"
      dataset="IE"
      rulelastupdate="2017-10-20"
      recordnumber="1">Subject CDISC01.200005 found in IE dataset who was assigned to arm WONDER10 (record 5 in DM dataset)</info>
</XQueryValidationMessages>
```

Conclusion

As shown, it is very easy to generate company-specific SDTM rules and make them available to the "Smart Dataset-XML Viewer". In this tutorial, we demonstrated this for an SDTM rule, but the same of course also applies to company-internal SEND and ADaM rules.

The only thing you need to learn is the XQuery language (but it is pretty easy). It is a W3C standard and as such, each young informatician nowadays learn it. So in case you do not know XQuery and are do not want to learn it, just grep a young informatician in your company. He or she will surely know XQuery and will help you further.

You can of course always ask us to develop some company-specific rules for you. Just drop us a mail at info@XML4Pharma.com.