Validating CDISC ODM files with the XML4Pharma ODM Checker v.1.3

Author: Jozef Aerts, XML4Pharma
Last update: October 20, 2009

Introduction

The CDISC ODM Checker v.1.3 is a validation tool, made freely available to CDISC members, to check ODM instance files against the ODM standard (v.1.2 or v.1.3).
Validation for ODM files version 1.1 or v.1.2 which use a DTD (Document Type Definition) is not supported anymore. For these, please use the old ODM Checker (v.0.7)

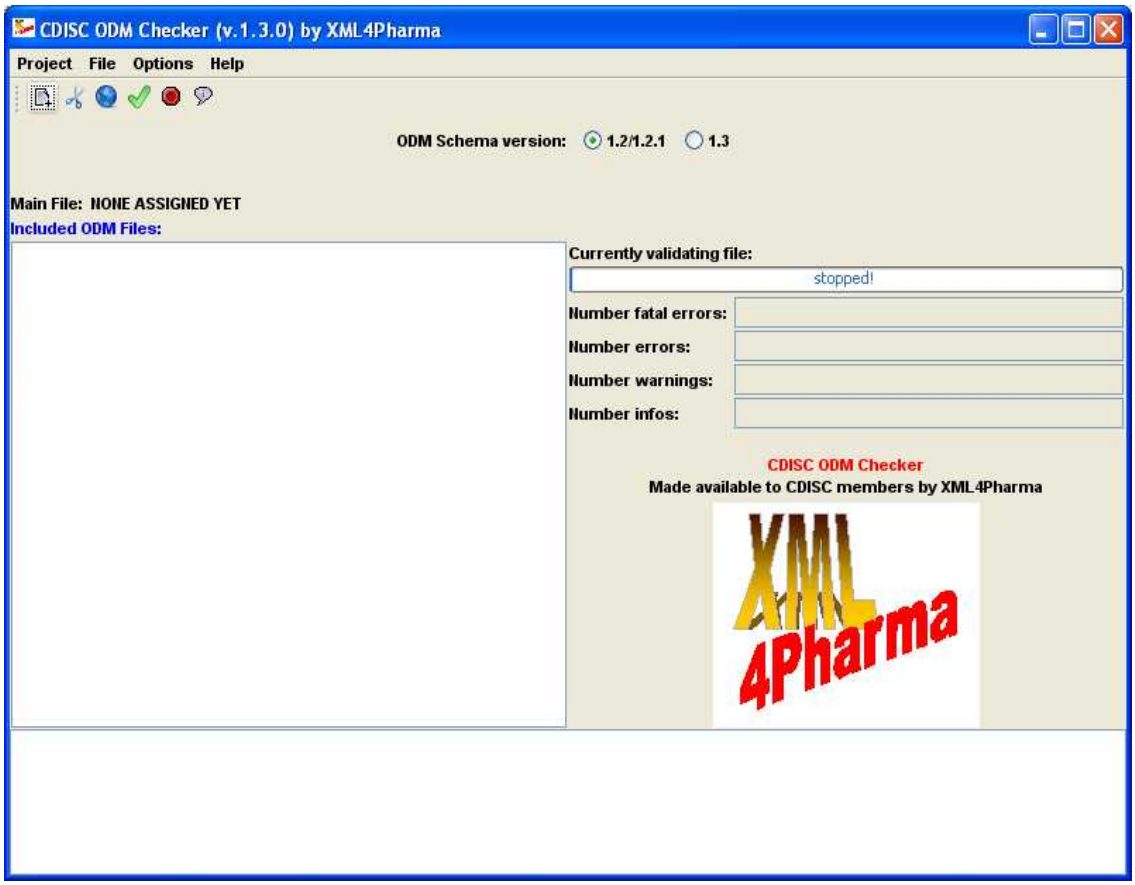
Installation

The CDISC ODM Checker is a Java application. For installation instructions, please see the installation manual or follow the instructions as you obtained by e-mail.

Starting the application

If the software has been correctly installed, a double-click on the entry “ODMChecker.bat” (Windows) or “ODMChecker.sh” (Linux, Unix) in your file explorer should start up the software. If nothing happens, please read the “troubleshooting” section in the installation manual first.

The following screen is presented:



Selecting an ODM file to validate

One can select a single ODM file to be validate either by using the menu “File – Open” or by using the “Add ODM File” icon on the toolbar:

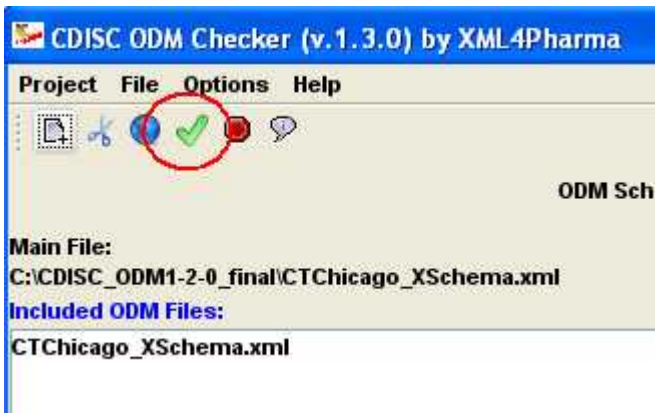
A filechooser then pops up, preselecting the files that have the “.xml” suffix.
If your ODM files use another suffix, please select “All files” under “File type”.
Selecting a file and clicking the OK button then loads the file. For example:

The filename of the selected file is added to the list of “Included ODM Files”, and it is also listed under the heading “Main File:”.

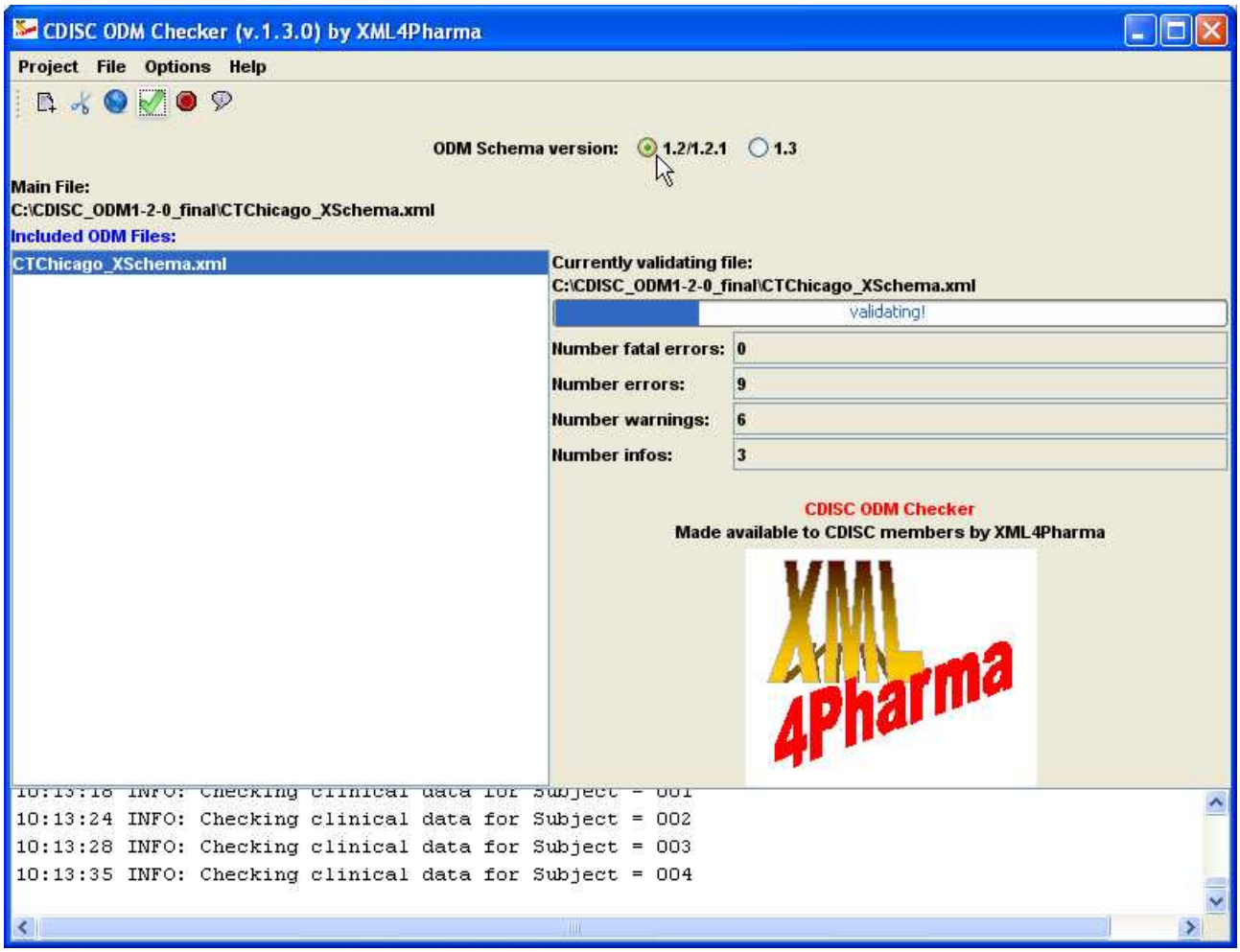
Performing a first validation

If you know that your ODM file is an ODM file that has the ODM namespace assigned, leave the radiobutton “1.2” for “ODM Schema Version” checked. If you know your ODM file is a v.1.3 ODM file, select “1.3” for the “ODM Schema Version”.
Please remark that old ODM files v.1.1 or v.1.2 with a DTD (Document Type Definition) cannot be validated anymore using the current version of the ODMChecker.

To start validation, either use the menu “Project – Start Validation” or click the “Start Validation” icon on the toolbar.



Validation is started.
The course of the validation can easily be followed in the console which is located near the bottom of the application window:

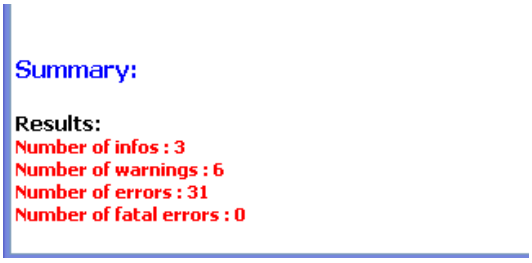


During validation, the number of errors and warnings found sofar is constantly updated on the right side of the screen.

When the validation is ready, a new window (the report window) is presented:

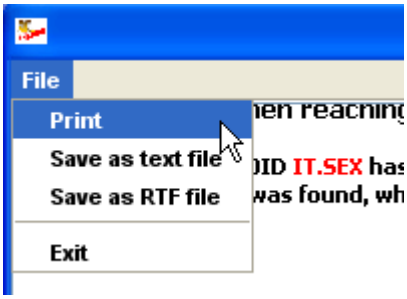
For each warning or error, the line number in the ODM file is given, as well as a description of the error or warning, and a reference to the section of the ODM standard[1] that is applicable.

Scrolling down to the bottom of the report window shows a short summary:



Printing and Saving the validation report to file

The contents of the validation report can either be printed or saved to file.
To print the contents, use the “File – Print” menu.



1 In all cases, the reference is to the ODM 1.3 standards document

To save the contents of the report to a Windows RTF file, choose “File – Save a s RTF file”.
One can also save the file as a simple text file, using “File – Save as text file”.

To close the window, use the menu “File – Exit”.

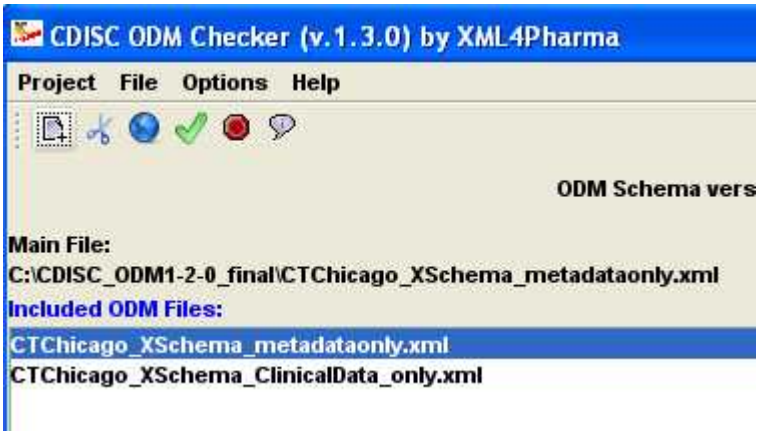
Loading several ODM files

You can load several ODM files for validation.
The system however expects that there is a “Main File” containing at least the metadata for the study for which the other ODM files will be validated.

So for example, if the metadata for a study are located in one file, and the clinical data come as separate files, one will usually first load the file with metadata, and then add the other files.

To load several files, use the menu “File – Add” several times, or use the “Add ODM File” icon on the toolbar, until all necessary files are listed in the “Included ODM files” list.

For example:



First, the file with only metadata has been loaded, followed by a file with only clinical data.
The first file that was loaded is automatically set to be the “Main File”.
This means that the metadata will be taken from that file, and that for all other files, the clinical data will be validated against the metadata from the “Main File”.

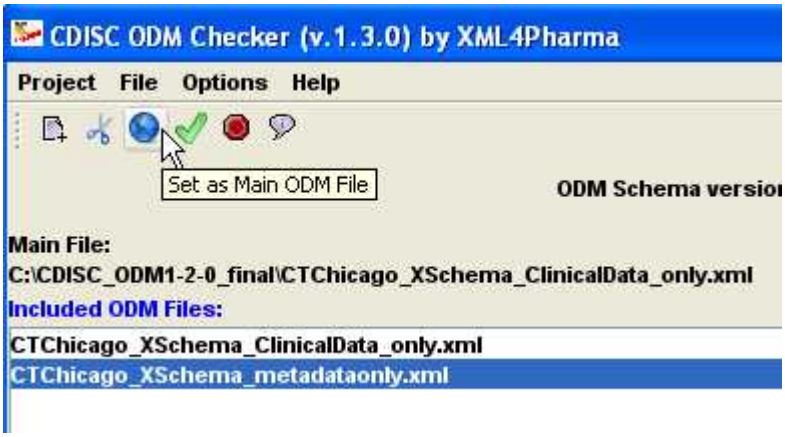
Removing files from the list

Files can always be removed from the list by selecting the file in the list and using the menu “File – Remove” or by clicking the “cissor” icon on the menu bar.
If there are still files in the list after the “Remove” operation, then the first file in the list is automatically being set to be the “Main File”.

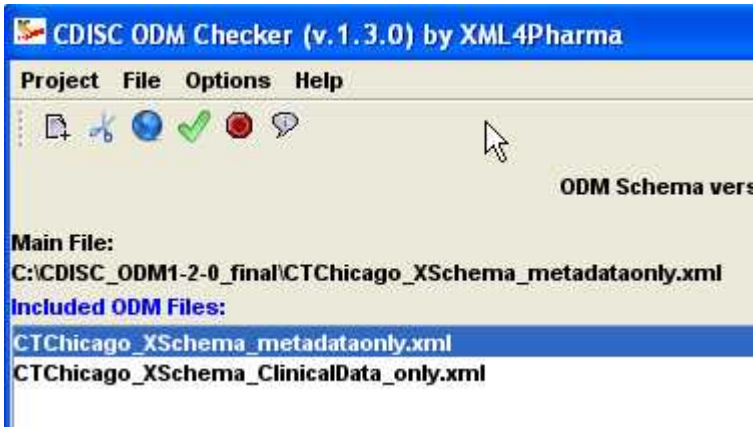
Setting a specific file as the “Main File”

One can set a specific file in the list to be the “Main File”.

To do so, select the file in the list, and click the “Set as Main ODM File” icon on the toolbar, or use the menu “File – Define Main File”.



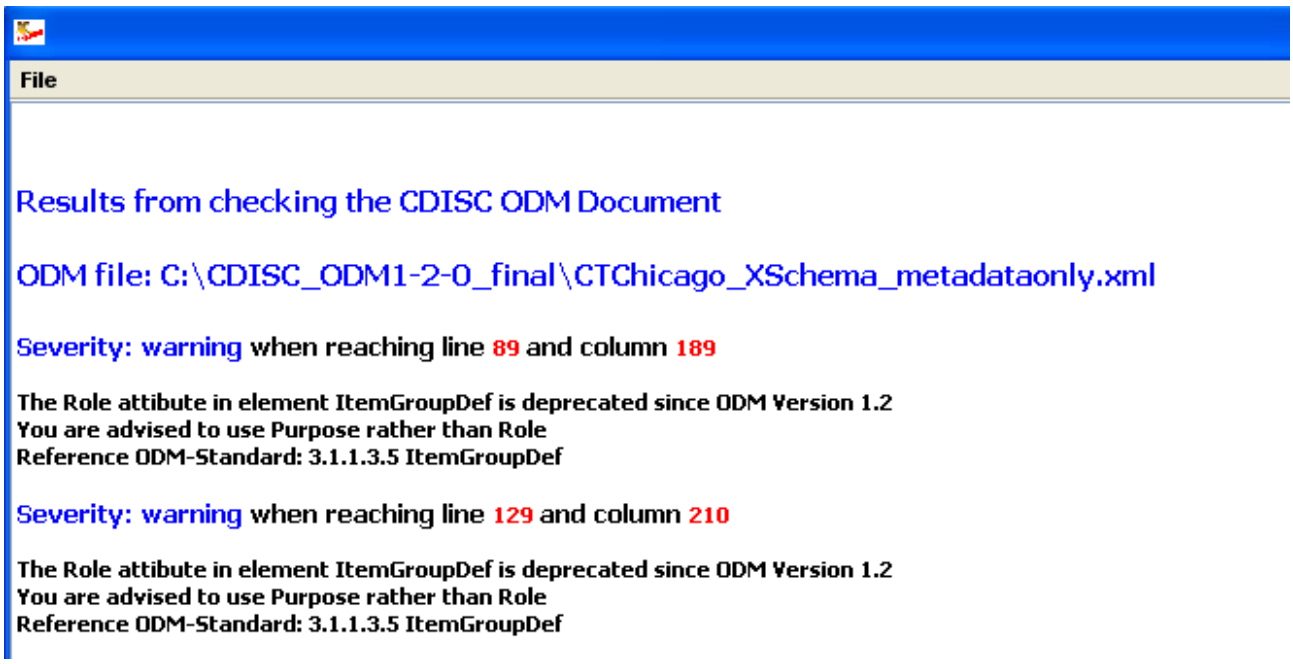
The selected file is moved to the top of the list, and is then listed as the “Main File”:



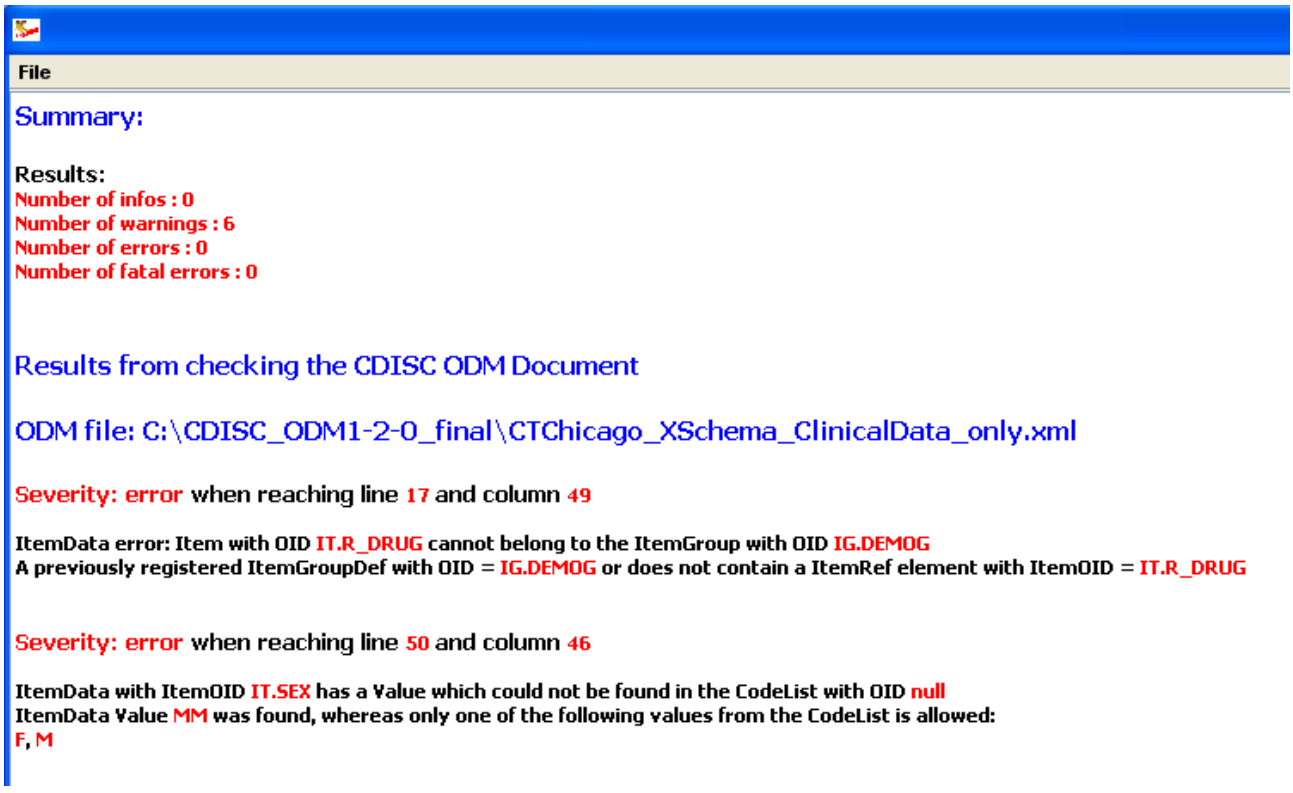
One can now starting validating, as the file containing the metadata is now correctly assigned.
The validator will first validate the metadata in the first file (the “Main File”), and then will validate all other information from all (including the first file) against the metadata[2].

The resulting report looks like:

2 Essentially, the system will build a so-called DOM document of metadata in memory, and validate all data against that DOM document with metadata.



giving the results for the first file, followed by the results for the second file, using the metadata fro the first file:



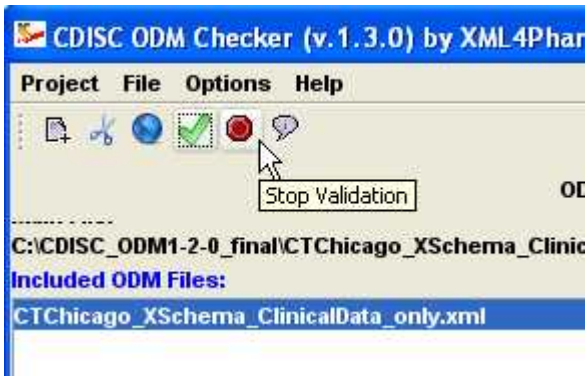
Of course one can add more files with clinical data.

If the files that have not been designated as the main file contain study metadata, these metadata will undergo a basic check against the standard (so not a complete check), and they will not be used to serve as a base for validation of any clinical data.

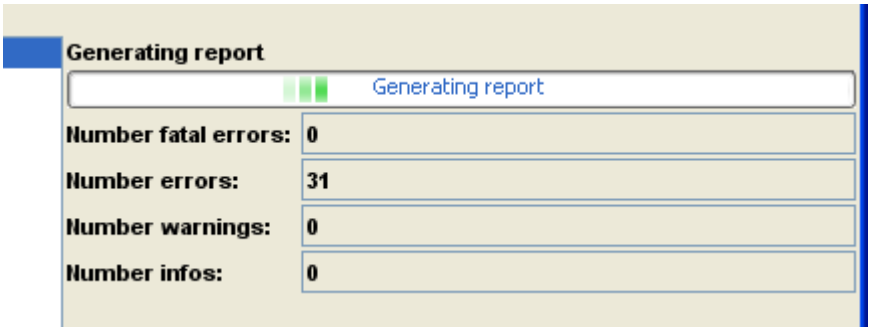
So, if one would like complete validation of the metadata in these files, one is enouraged to do so separately, by setting each of them once as the “Main File”.

Interrupting the validation

If one observes that (very) many errors are generated, one can decide to interrupt the validation.
In order to do so, either use the menu “Project – Stop Validation” or click the icon “Stop Validation” on the toolbar.



The validator then halts the validation and starts creating the report with all the errors and warnings obtained sofar.



Metadata files using the MetaDataVersion “Include” mechanism

In many cases, there are several versions of a study design, often using an earlier version as a base for the new version.
In ODM, this is accomplished through the “Include” mechanism.
The directory “testfiles/Include” that comes with the distribution contains a number of sample files:

- file CTChicago_MV_1_0_0.xml contains the first metadata version, only containing “ItemDef”s and “CodeList”s. It can be regarded as a base library
- file CTChicago_MV_1_1_0.xml contains definitions for the StudyEvents, Forms and ItemGroups. It does not have definitions for the Items, but these are referenced by an “Include” element using:

```
- <MetaDataVersion OID="v1.1.0" Name="Version 1.1.0">
    <Include StudyOID="123-456-789" MetaDataVersionOID="v1.0.0" />
```

referencing the first metadata version to be included
- file CTChicago_MV_1_2_0_ClinicalData.xml is the third version of the metadata for the study. It contains the "Protocol" element referencing the StudyEvents from the second metadata version, using an Include statement:

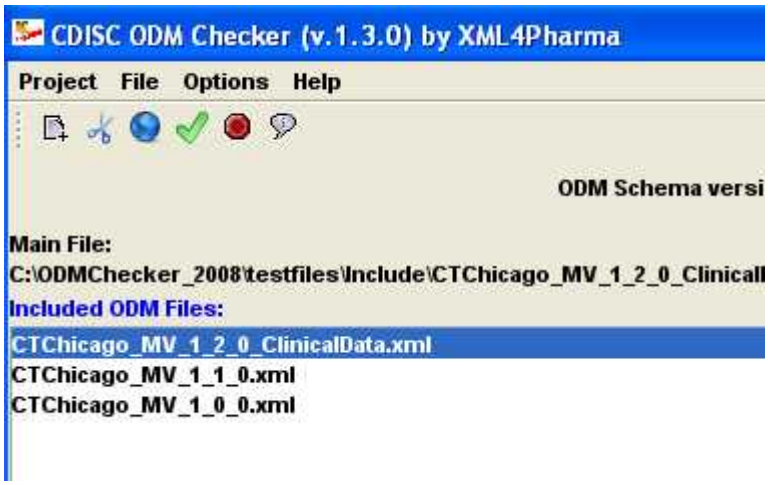
```
- <MetaDataVersion OID="v1.2.0" Name="Version 1.2.0">
    <Include StudyOID="123-456-789" MetaDataVersionOID="v1.1.0" />
  - <Protocol>
      <StudyEventRef StudyEventOID="SE.VISIT0" OrderNumber="1" Mandatory="Yes" />
      <StudyEventRef StudyEventOID="SE.VISIT1" OrderNumber="2" Mandatory="Yes" />
    </Protocol>
    <!--  and we do REDEFINE some ItemDef from the earliest MetaDataVersion (were Datatype 'text')   -->
    <ItemDef OID="IT.AEENDT" Name="Derived Stop Date AS DATE" DataType="date" />
    <ItemDef OID="IT.STRT_DT" Name="Derived Start Date" DataType="date" SASFieldName="STRT_DT" />
    <ItemDef OID="IT.STOP_DT" Name="Derived Stop Date" DataType="date" SASFieldName="STOP_DT" />
  </MetaDataVersion>
```

It also redefines a number of Items from the first metadata version[3] for the data type.

As such, the third metadata version ("v1.2.0") contains a full study description.

Additionally, it contains a set of reference data and clinical data for the third version of the metadata.

Essentially, all the metadata is distributed over 3 files, with the file **CTChicago_MV_1_2_0_ClinicalData.xml** containing the latest version of the metadata. So it is this file that we need to set as "Main File".
We also need to add the two other files (CTChicago_MV_1_1_0.xml and CTChicago_MV_1_0_0.xml) to the list so that the system can "execute" the "Include" statements from the metadata in the main file. The order of these two additional files in the list is unimportant. So we may have:

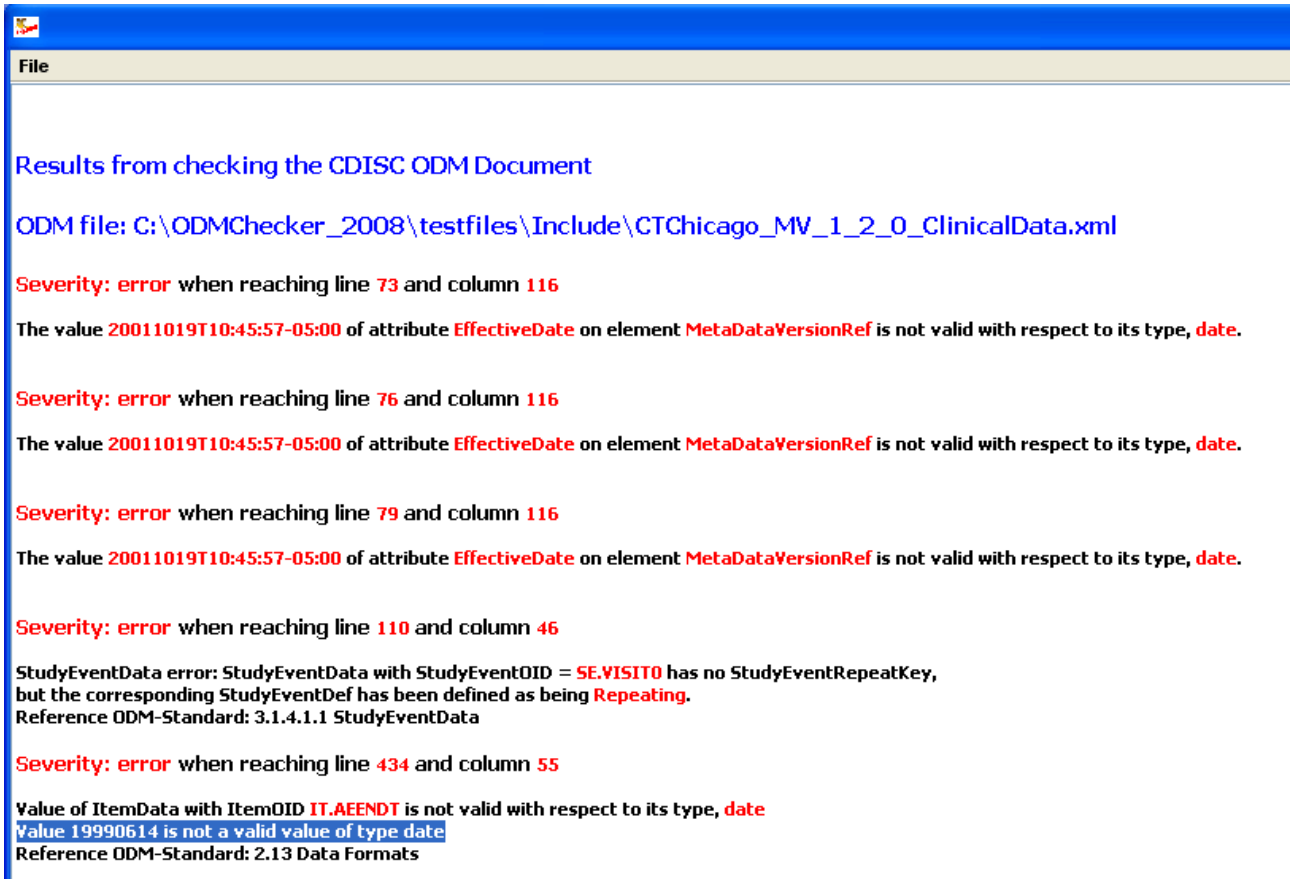


The system will first build a full set of the metadata by using/executing the "Include" statements, and searching in as well in the main file as in the other files for the to-be-included previous metadata versions.
Once the full metadata set build, it starts again with the first file, and validates the reference data and the clinical data against the full metadata set.

3 This as there is an "Include" for the second metadata version, which has an "Include" for the first metadata version.

It then performs a basic validation of the metadata in the other files, and validates the reference data and the clinical data against the full metadata version set, if reference data and/or clinical data is present (which is not the case here).

Some results are:



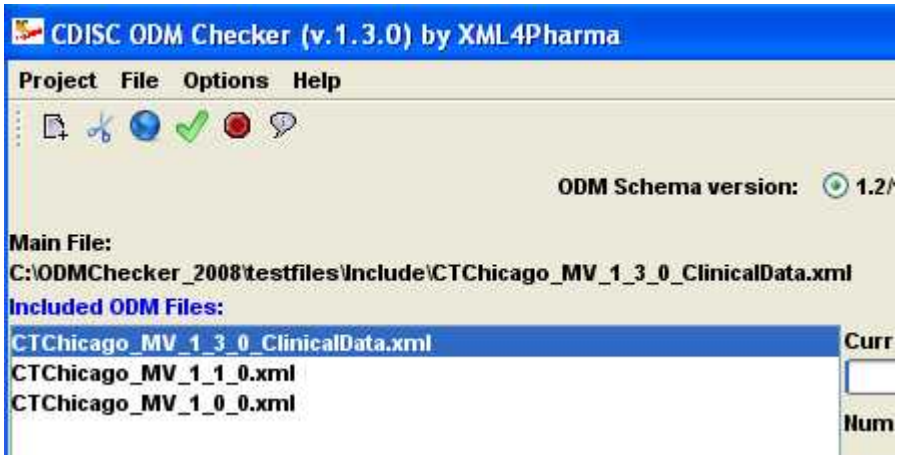
Originally, in the first version of the metadata, the “Item” with OID “IT.AEENDT” was defined as being of type “text”, but in the third version, it was redefined to be of type “date”.
So when in the clinical data, a value “19990614” is found, an error is generated as the latter value is not a valid XML-date.

Different metadata versions do not have to be in the same file. This can be seen in the sample file **CTChicago_MV_1_3_0_ClinicalData.xml**.
It also contains the third version of the metadata (v1.2.0) including the second one (which includes the first one), but also a fourth version of the metadata, defining a new visit with a new form, and adding the new visit to the protocol:

```
- <MetaDataVersion OID="v1.2.0" Name="Version 1.2.0">
    <Include StudyOID="123-456-789" MetaDataVersionOID="v1.1.0" />
  - <Protocol>
      <StudyEventRef StudyEventOID="SE.VISIT0" OrderNumber="1" Mandatory="Yes" />
      <StudyEventRef StudyEventOID="SE.VISIT1" OrderNumber="2" Mandatory="Yes" />
    </Protocol>
    <!-- and we do REDEFINE some ItemDef from the earliest MetaDataVersion (were Datatype 'text')  -->
    <ItemDef OID="IT.AEENDT" Name="Derived Stop Date AS DATE" DataType="date" />
    <ItemDef OID="IT.STRT_DT" Name="Derived Start Date" DataType="date" SASFieldName="STRT_DT" />
    <ItemDef OID="IT.STOP_DT" Name="Derived Stop Date" DataType="date" SASFieldName="STOP_DT" />
    <!-- DELIBERATE ERROR: DataType 'datetest' does not exist AND IT.STOP_DT already defined  -->
    <ItemDef OID="IT.STOP_DT" Name="Derived Stop Date" DataType="datetest" SASFieldName="STOP_DT" />
  </MetaDataVersion>
+ <!--  -->
- <MetaDataVersion OID="v1.3.0" Name="Version 1.3.0">
    <Include StudyOID="123-456-789" MetaDataVersionOID="v1.2.0" />
  - <!--
        If we add a new Visit, we must also relist the old visits,
                                        otherwise, it is meant that the old visits MUST be removed
    -->
  - <Protocol>
      <StudyEventRef StudyEventOID="SE.VISIT0" OrderNumber="1" Mandatory="Yes" />
      <StudyEventRef StudyEventOID="SE.VISIT1" OrderNumber="2" Mandatory="Yes" />
      <!-- new visit  -->
      <StudyEventRef StudyEventOID="SE.NEWVISIT" OrderNumber="3" Mandatory="Yes" />
      <!-- DELIBERATE ERROR: StudyEvent SE.NOTEXIST does not exist  -->
      <StudyEventRef StudyEventOID="SE.NOTEXIST" OrderNumber="3" Mandatory="Yes" />
    </Protocol>
    <!-- DELIBERATE ERROR: should have Repeating attribute, AND a Type attribute  -->
  - <StudyEventDef OID="SE.NEWVISIT" Name="Follow Up">
      <FormRef FormOID="FORM.FOLLOWUP" Mandatory="Yes" />
    </StudyEventDef>
  - <FormDef OID="FORM.FOLLOWUP" Name="Follow Up" Repeating="No">
```

It also contains a set of reference data and clinical data for the second version of the metadata, and a small set of clinical data for the third version of the metadata.

We can now validate the file, but need to add the files "CTChicago_MV_1_1_0.xml" and "CTChicago_MV_1_0_0.xml" to the list, as metadata from them are being referenced:



If we also add another file with clinical data (i.e. file CTChicago_MV_1_2_0_ClinicalData.xml), but not as the main file, its clinical data will be checked against the third version of the metadata (v1.2.0), but the metadata themselves will not be used to define the full set of metadata.

Saving a project

It may be useful to save all the settings and the file list for future use, e.g. for the case one regularly receives updates of the files with clinical data.

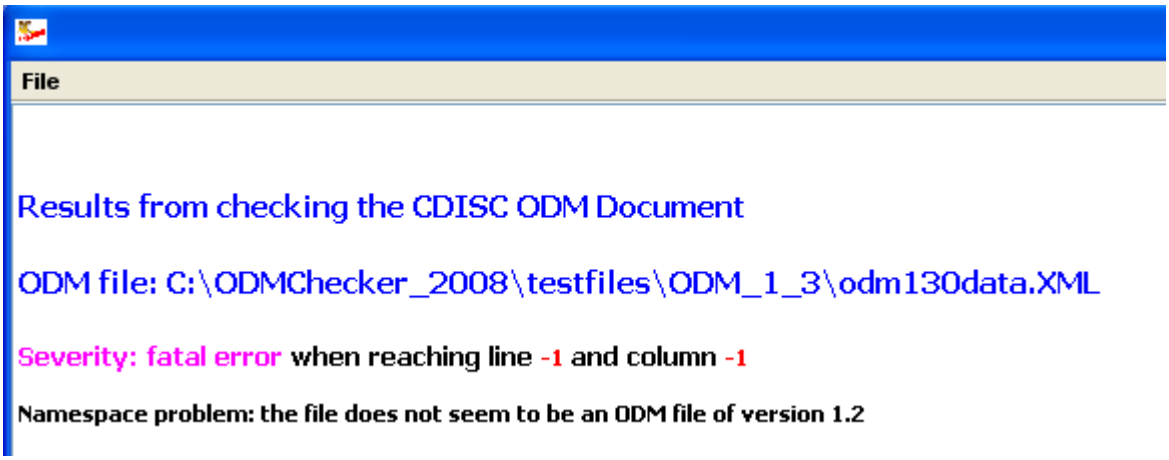
In order to save the project, use the menu “Project – Save”. A filechooser is then displayed, allowing the settings to be saved to a file. The file suffix for project files is “.prj”.

Similarly, one can load previously saved project files using the menu “Project – Open”.
One is then prompted to select a file having the suffix “.prj”.

Working with CDISC ODM 1.3 files

Until now, we only worked with ODM 1.2 files.
The distribution contains a small set of ODM 1.3 files.
Validation of ODM 1.3 files exactly works the same way as for ODM 1.2 files, one only needs to take care that the radiobutton “1.3” for “Schema Version” is selected.

If this is not done, the schema validation cannot work correctly, and a fatal error is thrown:



The reason is that the elements of the ODM file “live” in a namespace, which is different for each of the ODM versions, and that the namespace of the instance file must always match that of the XML-Schema file.

A similar error may be obtained when the file is not an ODM file at all, or when the ODM file is still an ODM 1.0 or 1.1 file, or when it is an old ODM file that still references a DTD (Document Type Definition). The latter do not have a namespace assigned.

One of the main features of v.1.3 of the standard is that it allows the use of “typed” ItemData, e.g.:

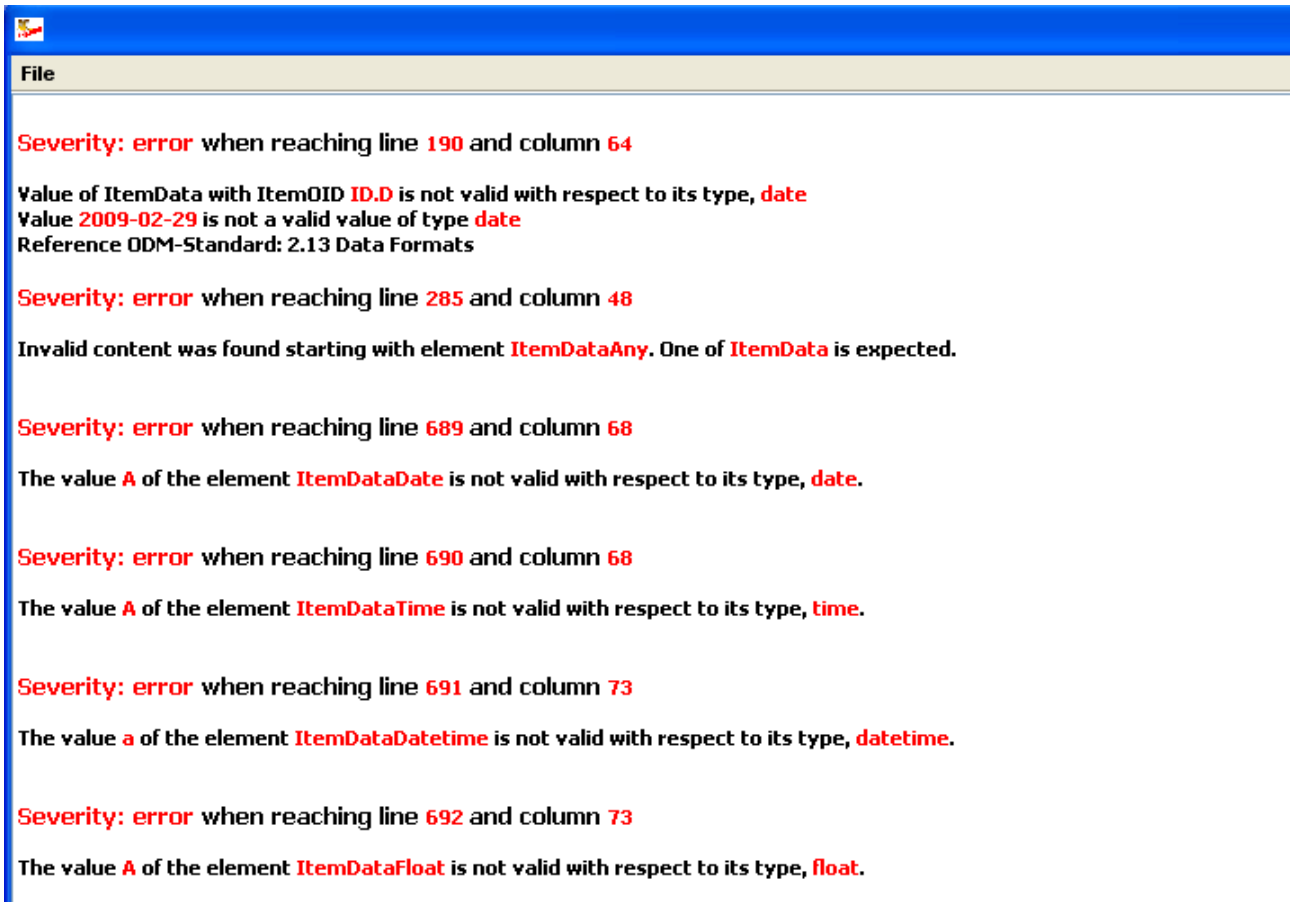
```
- <SubjectData SubjectKey="002">
  - <StudyEventData StudyEventOID="StudyEventOID" StudyEventRepeatKey="1">
    - <FormData FormOID="FormOID" FormRepeatKey="1">
      - <ItemGroupData ItemGroupOID="DATATYPE" ItemGroupRepeatKey="ALL ELEMENT" TransactionType="Insert">
          <ItemDataPartialDate ItemOID="ID.PD">1959-12</ItemDataPartialDate>
          <ItemDataPartialTime ItemOID="ID.PT">12</ItemDataPartialTime>
          <ItemDataPartialDatetime ItemOID="ID.PDT">1959-12-11T12</ItemDataPartialDatetime>
          <ItemDataDurationDatetime ItemOID="ID.DDT">P03Y11M07DT16H</ItemDataDurationDatetime>
          <ItemDataIntervalDatetime ItemOID="ID.IDT">19591211/20031107T1624</ItemDataIntervalDatetime>
          <ItemDataIncompleteDatetime ItemOID="ID.NDT">1959---11T12:34:56-05:00</ItemDataIncompleteDatetime>
        </ItemGroupData>
      </FormData>
    </StudyEventData>
  </SubjectData>
```

Performance

The CDISC ODM has a large number of rules, so validating large files, especially when containing lots of clinical data, can take a considerable amount of time.

Showing sample data for some of the new data types of ODM 1.3.

The rules concerning typed ItemData have also been implemented in the ODM Checker, and the distribution contains a sample test file (containing a good amount of deliberate errors) in directory "testfiles/ODM_1_3/odm130data.xml".



For example, one sees an error message for line 285, because the standard does not allow mixing

"classic" ItemData elements (usually having a "Value" attribute) with typed ItemData elements (such as ItemDataAny).
The error for line 190 may seem strange first, but we must recognize that there hasn't been a February 29th in 2009. Because the standard uses a native XML datatype here, it immediately recognizes non-existing dates. This is one of the major advantages of using typed ItemData.

Options and Settings

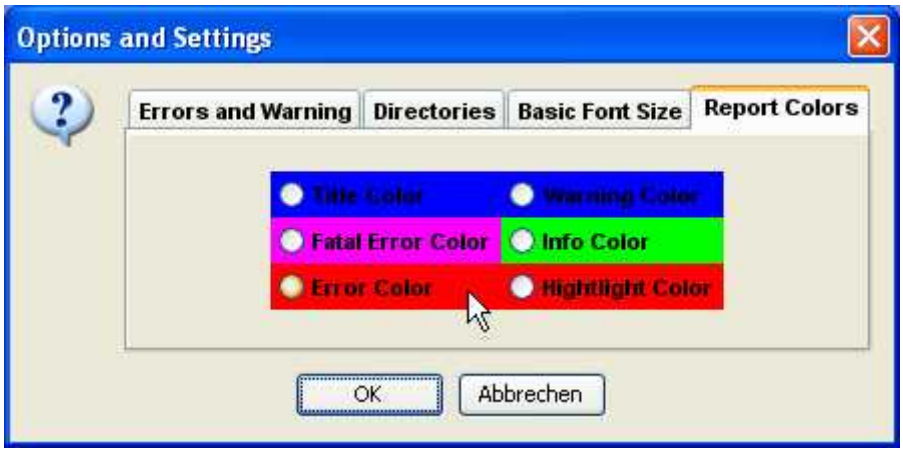
A number of options and settings can be changed using the menu "Options – Settings". The following dialog appears:

First of all, one can choose whether the errors and warnings are sorted by linenumber (default = yes). One can disable sorting in case one expects a large number of errors and warnings, as the sorting process can be a time-consuming one in the case of a large amount of errors.

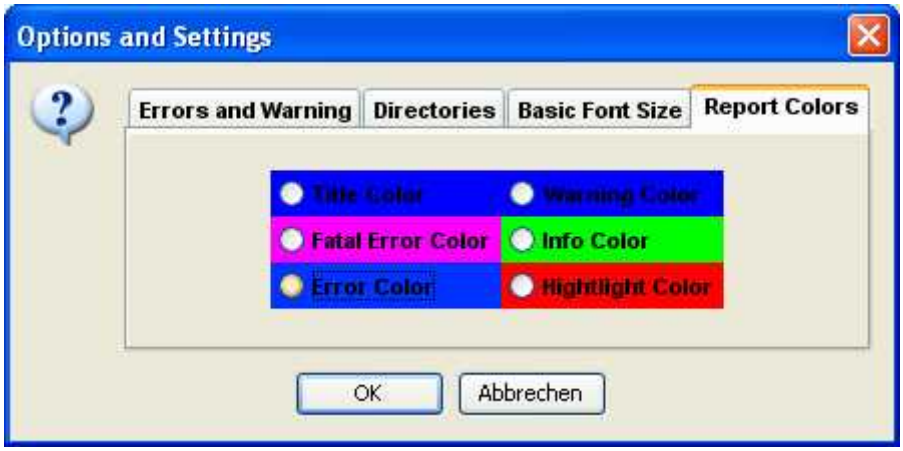
Normally, the system remembers the directory from where the last file was loaded. This can also be switched off.

The basic font site for all the labels in the graphical user interface (GUI) can be changed. The default value is "11". Setting it to a larger value can especially be interesting for people with a visual handicap.

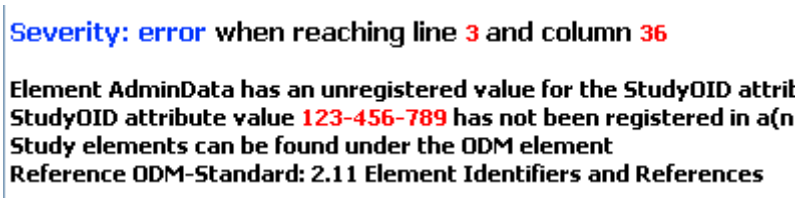
The colors used to make up the report with warnings and errors can be changed.



We can change the color for e.g. “Error Color” by clicking the radiobutton. A colorchooser shows up. If we e.g. select a blue color, the background color in the dialog is changed:



Clicking OK brings us back. If we now do a validation, we see that the red color for errors in the report has been changed into blue:



Limitations

Currently the validation does not cope very well with a situation where more than one Study is being defined in the main file with metadata: only the first Study from that file is taken.