User's Manual *and*

Translating Arelle to your language

Version 0.1 December 9th 2011

Notes:

- The official web site for documentation is arelle.org/documentation
- This handbook is available at www.openfiling.info => Arelle
- Comments about this handbook, please contact info@xbrl.es

This handbook has been made possible with the collaboration of:

REVISION SHEET

Release	Date	Author		Release description
Rev. 0.0	2011/08/03	Ignacio Amelivia	ignacio.amelivia@xbrl.es	Initial release
Rev.0.1	2011/12/09	Ignacio Boixo Colm O hAonghusa Allyson Ugarte	ignacio@boixo.com ircoha@gmail.com ugarte@pipeline.com	Revision

.

TABLE OF CONTENTS

1. WHAT IS ARELLE?

Arelle is a project to provide an easy-to-use open source platform and toolkit for XBRL.

The initial intent was to meet needs that are not commercially viable, such as to support extension modules under-development and test suite facilities, in a compact framework, and to support academic training and projects.

Support for XBRL versioning was an initial goal, to provide both a validation tool for versioning reports and a production tool to generate the basics of a versioning report that can be inferred by comparing two Discoverable Taxonomy Sets (DTSs). As the project evolved, Edgar and Global Filer Manual validation, Base Specification, Dimensions, Generic linkbase, Formula, and EuroFiling rendering linkbase, were added to the list of objectives. Additional features were added to allow formula experimentation with existing real filings, via an RSS Watch facility. Arelle fully integrates test cases with the object models for XBRL instances and DTSs.

Arelle allows for the continual verification of tool performance as it is extended and adapted by its users. Users can explore the functionality and features from either an interactive GUI or command line interface, and can develop their own controller interfaces, as needed.

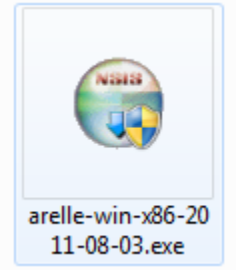
2. GETTING STARTED

2.1 DOWNLOADING AND INSTALLING

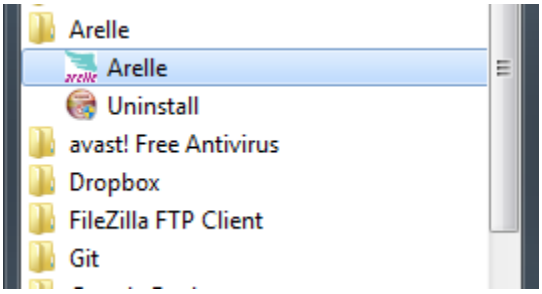
There are currently installable packages for Windows 32, Windows 64, and Mac OS. UNIX users can get the application running from the source code (see next section).

To get Arelle, select and download the appropriate installation package from the download section on its official website (http://arelle.org), click on the download button corresponding to your operating system and the Arelle version you need. The latest version is recommended.

After downloading the installer, double-click on it and follow the steps through the installation process. This documentation is also suitable for the Arelle.app on the Mac OS.



If everything goes OK, Arelle should now appear on the Windows Start Menu.



2.2 Installing from the Source Code

Arelle source code is available in the following GitHub repository:

https://github.com/Arelle/Arelle.git

There are several tools for working with this kind of repository. The official GitHub documentation has a section covering this kind of tool. In this example, we will use a command line tool called git to get the lxml branch of the source code. This branch is the latest version available at the moment.

```
foo@linux:~$ git clone https://github.com/Arelle/Arelle.git -b lxml
Cloning into Arelle...
remote: Counting objects: 1267, done.
remote: Compressing objects: 100% (686/686), done.
remote: Total 1267 (delta 743), reused 1071 (delta 574)
Receiving objects: 100% (1267/1267), 2.81 MiB | 236 KiB/s, done.
Resolving deltas: 100% (743/743), done.
```

Then, we can get into the application directory and execute it using python3

```
foo@linux:~$ ls
Arelle
foo@linux:~$ cd Arelle/
foo@linux:~/Arelle$ ls
arelle            buildRenameX64.bat  debugCmdLineEntry.py  runMacGUI.sh
arelleCmdLine.py  buildRenameX86.bat  installWin64.nsi      scripts
Arelle.egg-info   buildVersion.py     installWin86.nsi      setup.py
arelleGUI.pyw     buildWin64Dist.bat  License.txt           testParser2.py
arelle.pyw        buildWin86Dist.bat  MANIFEST.in           testParser.py
buildMacDist.sh   buildWinDists.bat   README.txt
foo@linux:~/Arelle$ python3 arelleGUI.pyw
```

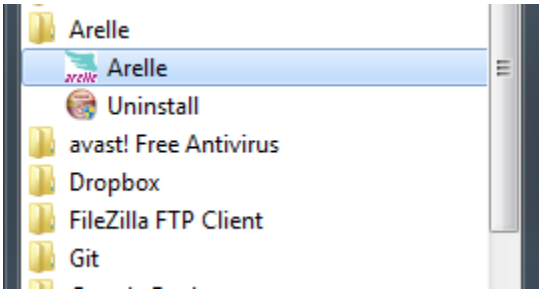
This version uses the lxml library so it should be installed in order to get the application working properly.

The Python's tool easy_install is the simplest way to do this, just by typing easy_install and the name of the library that we want to install.

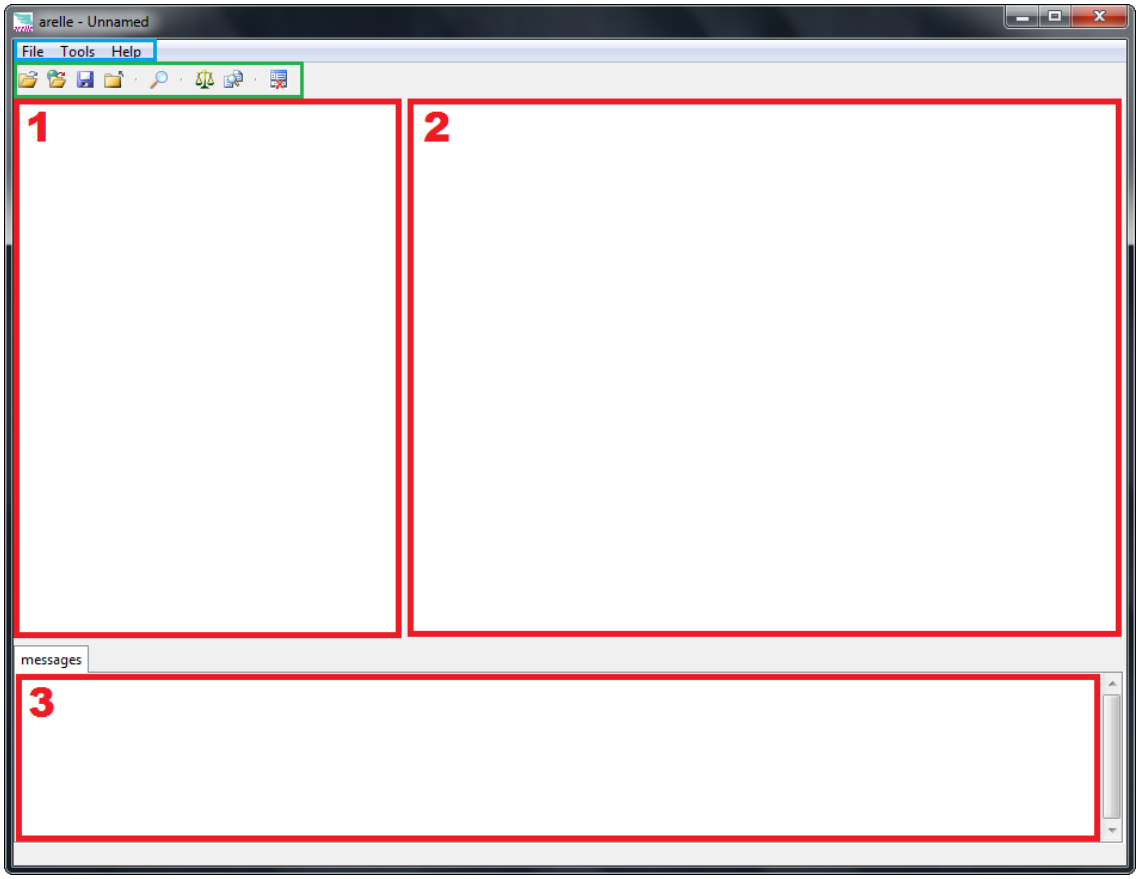
```
foo@linux:~$ sudo easy_install lxml
[sudo] password for foo:
Searching for lxml
Best match: lxml 2.3
Adding lxml 2.3 to easy-install.pth file

Using /usr/lib/python2.7/dist-packages
Processing dependencies for lxml
Finished processing dependencies for lxml
```

3. USING ARELLE (GUI)

To execute Arelle in GUI mode simply click on the Arelle shortcut from the Start Menu.



Then, the main window will appear on the screen.



The window is divided in various sections, as seen in the picture above. Marked with a blue rectangle, right below the Toolbar, there is the Menubar, and marked with red rectangles and divided into three sections, there is the main workspace.

3.1 Opening a file

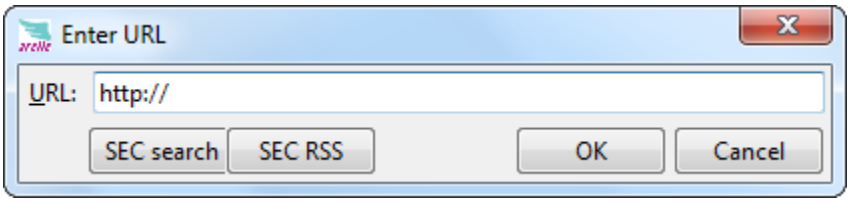
To start working, we have to get a file, and this can be done in two ways:

1. To open files from your computer, simply click on the Open local file icon situated in the Toolbar, or
2. Go to "File→Open File…" in the Menubar.

Then, select the desired xbrl, xsd or xml file and click the Open button.

There is also the possibility to open files hosted on a web server. To do this, click on the Open web file situated in the Toolbar or go to "File→Open Web…" in the Menubar.

Then this window will pop up:



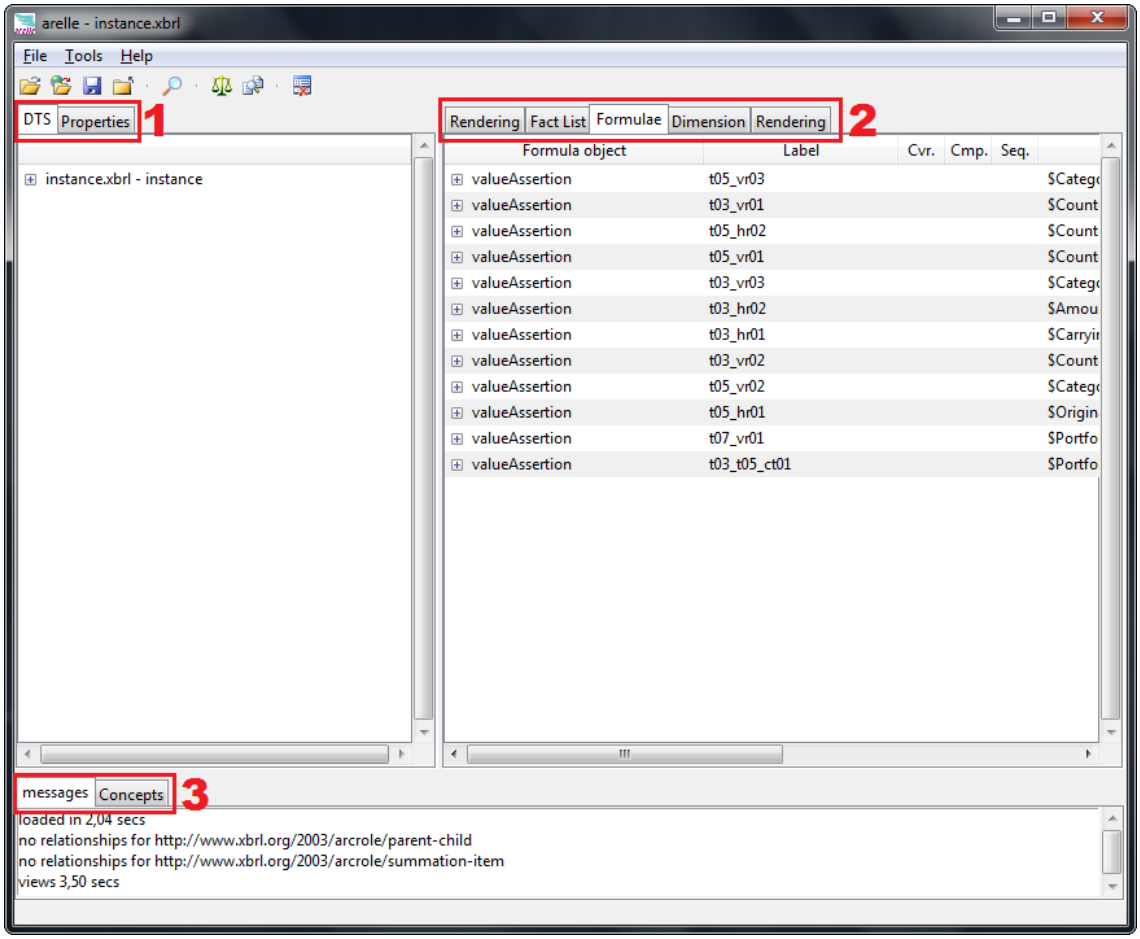
We are using the example of an "SEC search". In the URL field you can type the file URL you wish to work with and click the OK button.

The "SEC search" button opens the US SEC company search website in a browser window.

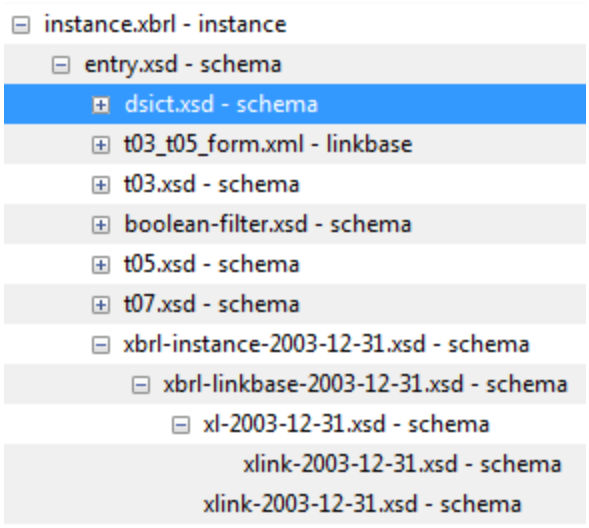
The "SEC RSS" button opens the RSS Feed that you can browse for an instance file.

3.2 VIEWING A FILE

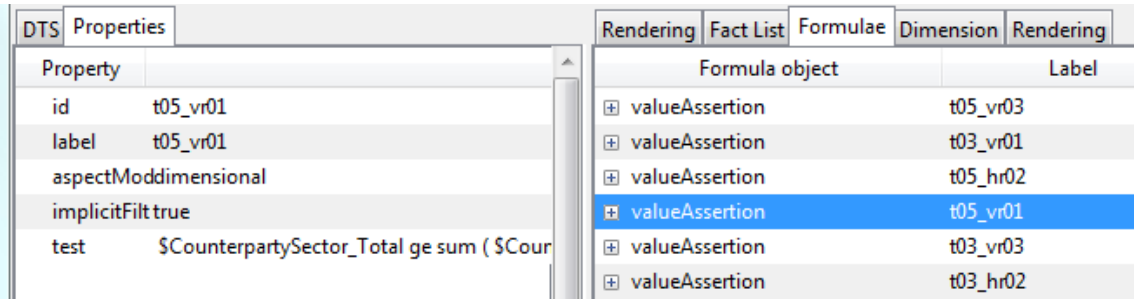
Once you open a file, the main workspace will look like this:



In the area marked with “Number 1”, there will be two tabs: DTS and Properties views. DTS offers an expandable tree view of the document DTS.

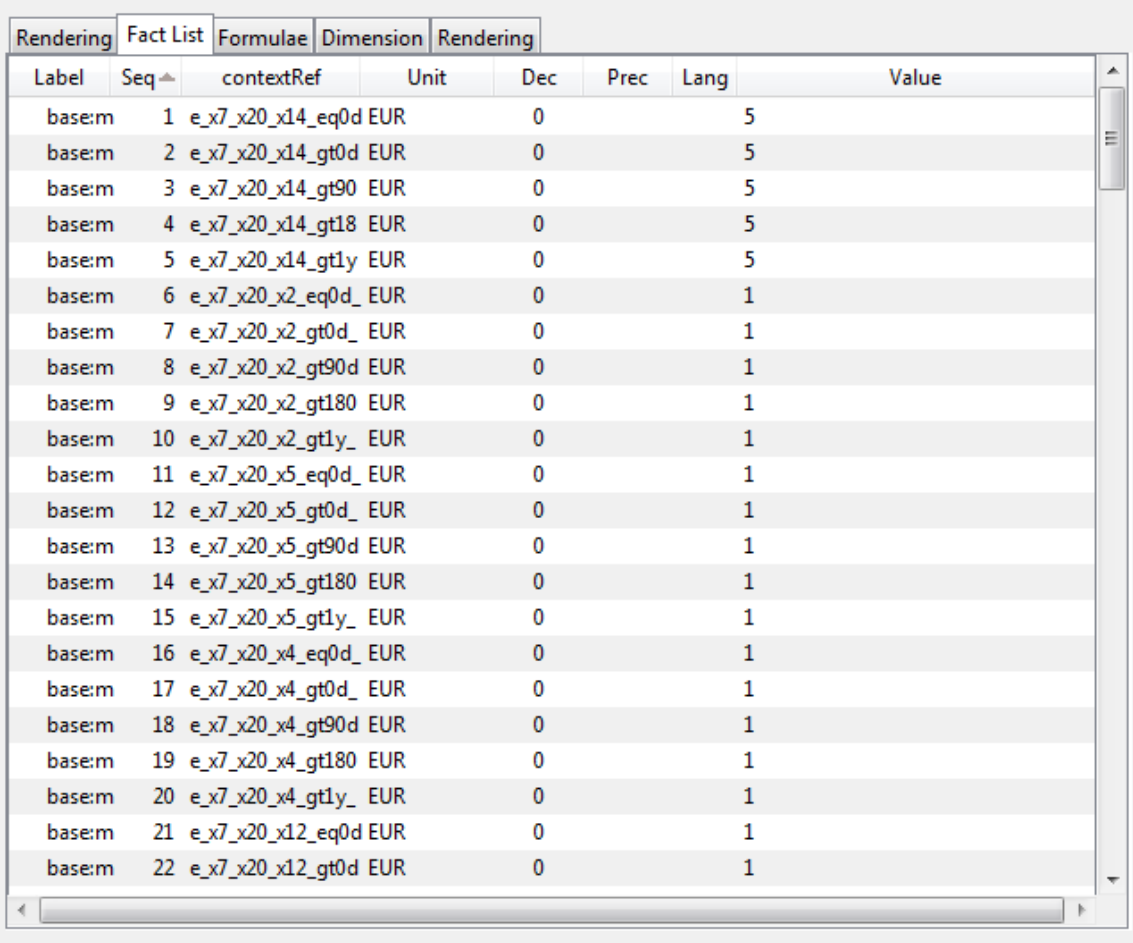


The properties tab shows the properties of the selected item from the main view, as seen in the following picture.



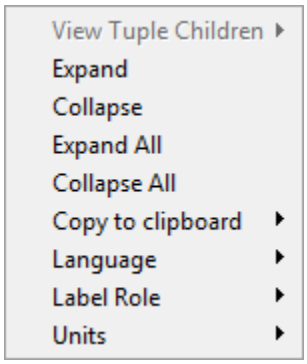
In the main view, marked with “Number 2”, there will be some other tabs. The number of tabs varies depending on what information is stored in the DTS. For example, if there is no rendering information in our DTS, there won’t be a rendering tab. These tabs show different views of our xbrl file.

Fact List



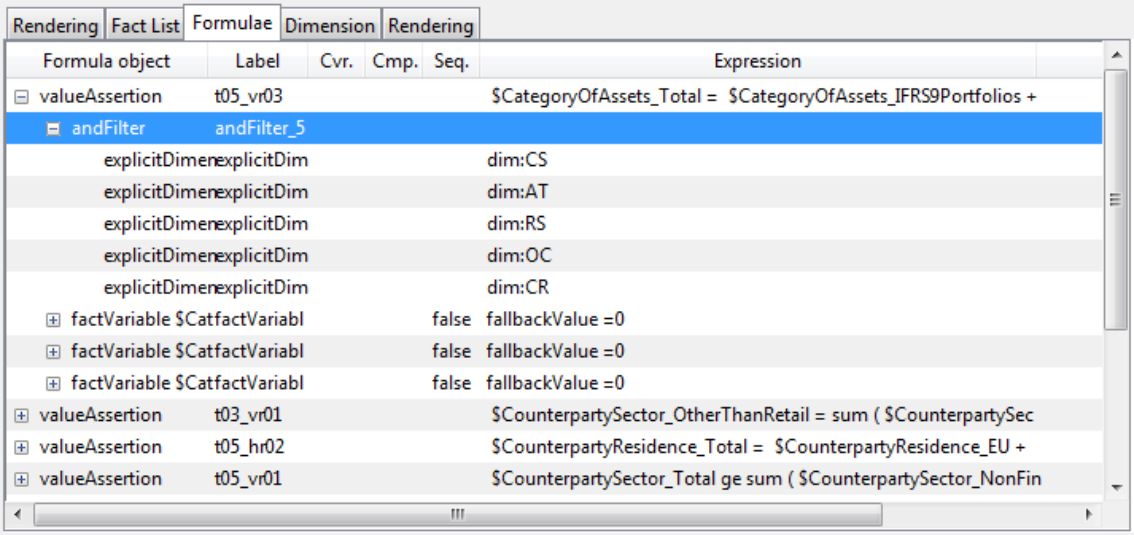
Label	Seq	contextRef	Unit	Dec	Prec	Lang	Value
base:m	1	e_x7_x20_x14_eq0d	EUR	0			5
base:m	2	e_x7_x20_x14_gt0d	EUR	0			5
base:m	3	e_x7_x20_x14_gt90	EUR	0			5
base:m	4	e_x7_x20_x14_gt18	EUR	0			5
base:m	5	e_x7_x20_x14_gt1y	EUR	0			5
base:m	6	e_x7_x20_x2_eq0d_	EUR	0			1
base:m	7	e_x7_x20_x2_gt0d_	EUR	0			1
base:m	8	e_x7_x20_x2_gt90d	EUR	0			1
base:m	9	e_x7_x20_x2_gt180	EUR	0			1
base:m	10	e_x7_x20_x2_gt1y_	EUR	0			1
base:m	11	e_x7_x20_x5_eq0d_	EUR	0			1
base:m	12	e_x7_x20_x5_gt0d_	EUR	0			1
base:m	13	e_x7_x20_x5_gt90d	EUR	0			1
base:m	14	e_x7_x20_x5_gt180	EUR	0			1
base:m	15	e_x7_x20_x5_gt1y_	EUR	0			1
base:m	16	e_x7_x20_x4_eq0d_	EUR	0			1
base:m	17	e_x7_x20_x4_gt0d_	EUR	0			1
base:m	18	e_x7_x20_x4_gt90d	EUR	0			1
base:m	19	e_x7_x20_x4_gt180	EUR	0			1
base:m	20	e_x7_x20_x4_gt1y_	EUR	0			1
base:m	21	e_x7_x20_x12_eq0d	EUR	0			1
base:m	22	e_x7_x20_x12_gt0d	EUR	0			1

This view shows all the facts that the xbrl instance file contains, with all its possible fields.

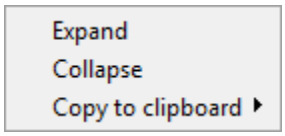


The context menu for this view is accessed right-clicking anywhere in the view, and shows some common options such as Language, Label Role and Units options; you can switch among the available representation formats for these fields.

Formulae

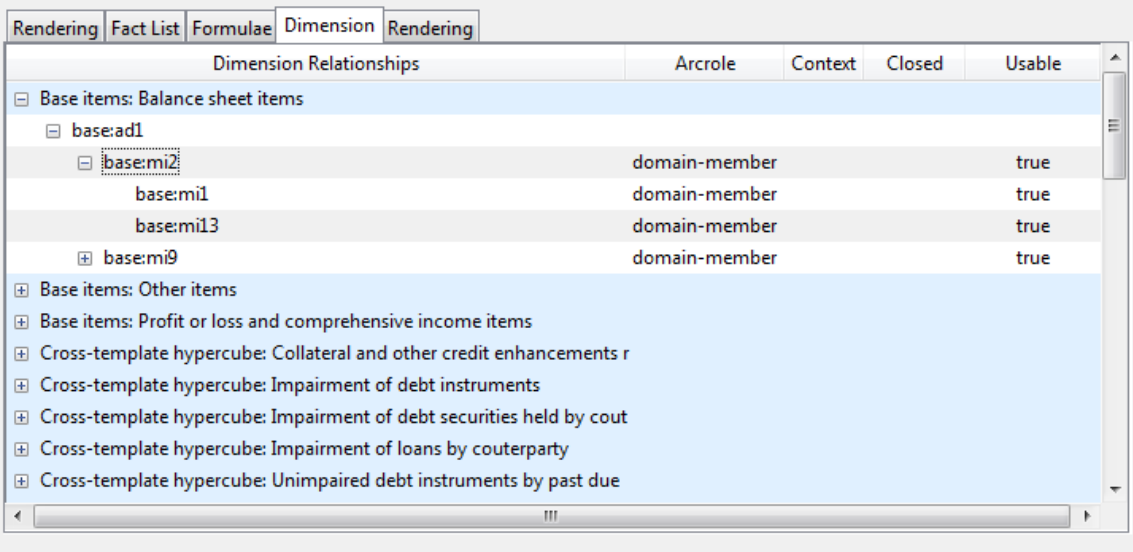


This view shows the formulae information contained in the DTS; it is expandable and you can also check all its properties.

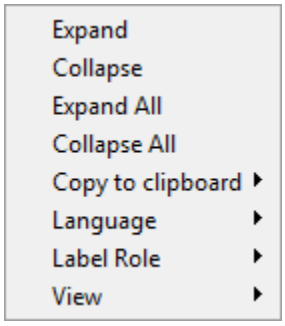


The context menu in this case shows just some simple and common options.

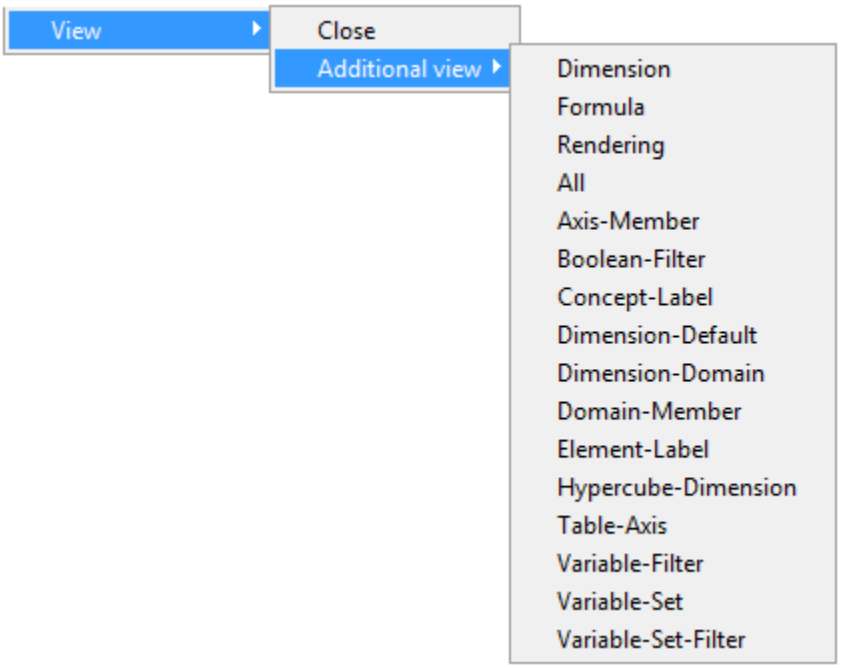
Dimension



This view shows an expandable view of the dimensional information. You can check dimensions, domain members, hypercubes etc…

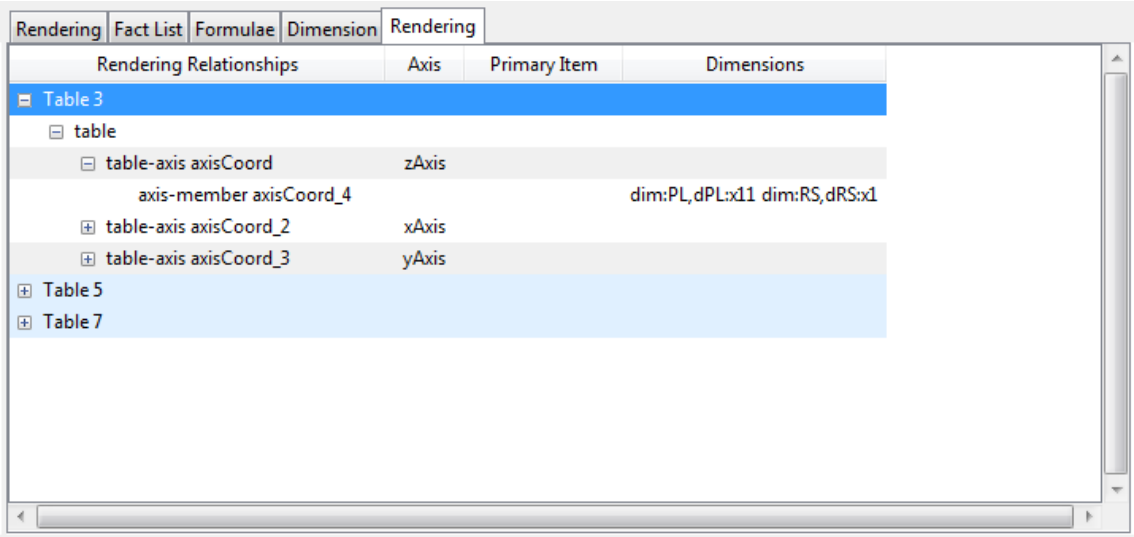


The context menu for this view offers a similar functionality, as in the other views.

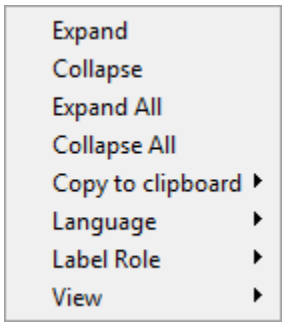


The View option offers some additional views and the possibility to close the current view tab.

Rendering (information)

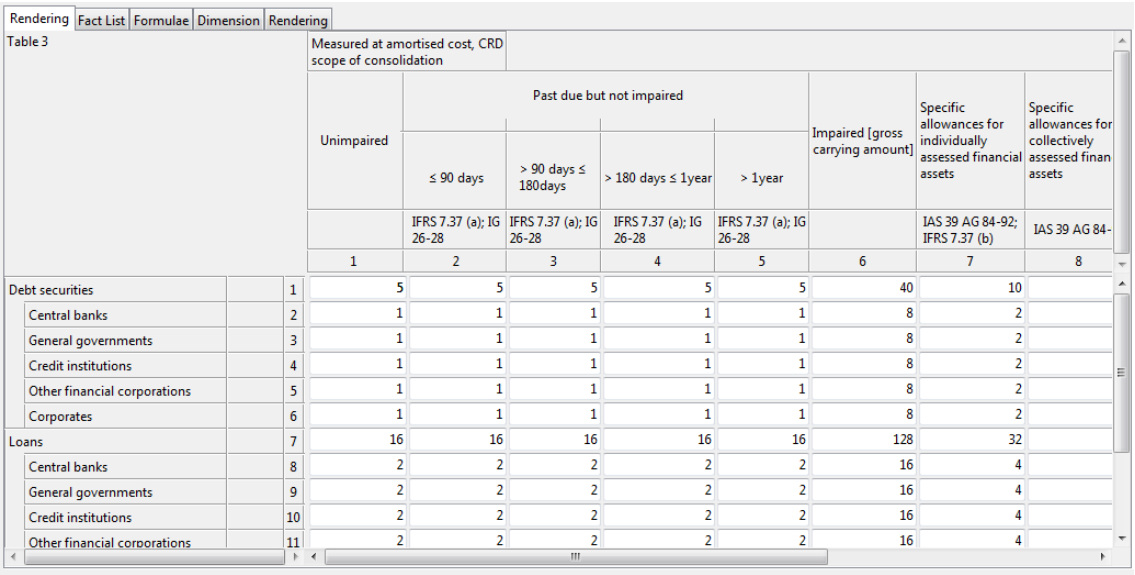


This view shows the rendering information; this is the information needed to render the file, not the rendering itself. You can expand the items to see all the relationships and attributes of every item needed to render the file.



The context menu is the same as in the previous view.

Rendering (render)

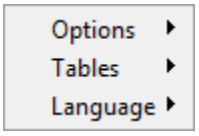


Rendering | Fact List | Formulae | Dimension | Rendering

Table 3

		Measured at amortised cost, CRD scope of consolidation							
		Unimpaired	Past due but not impaired				Impaired [gross carrying amount]	Specific allowances for individually assessed financial assets	Specific allowances for collectively assessed finan assets
			≤ 90 days	> 90 days ≤ 180days	> 180 days ≤ 1year	> 1year			
			IFRS 7.37 (a); IG 26-28	IFRS 7.37 (a); IG 26-28	IFRS 7.37 (a); IG 26-28	IFRS 7.37 (a); IG 26-28		IAS 39 AG 84-92; IFRS 7.37 (b)	IAS 39 AG 84-
		1	2	3	4	5	6	7	8
Debt securities	1	5	5	5	5	5	40	10	
Central banks	2	1	1	1	1	1	8	2	
General governments	3	1	1	1	1	1	8	2	
Credit institutions	4	1	1	1	1	1	8	2	
Other financial corporations	5	1	1	1	1	1	8	2	
Corporates	6	1	1	1	1	1	8	2	
Loans	7	16	16	16	16	16	128	32	
Central banks	8	2	2	2	2	2	16	4	
General governments	9	2	2	2	2	2	16	4	
Credit institutions	10	2	2	2	2	2	16	4	
Other financial corporations	11	2	2	2	2	2	16	4	

This last view shows the rendered file, if there is rendering information contained in the DTS.



This time the context menu looks a little different. The “Options” menu allows activating some view options.

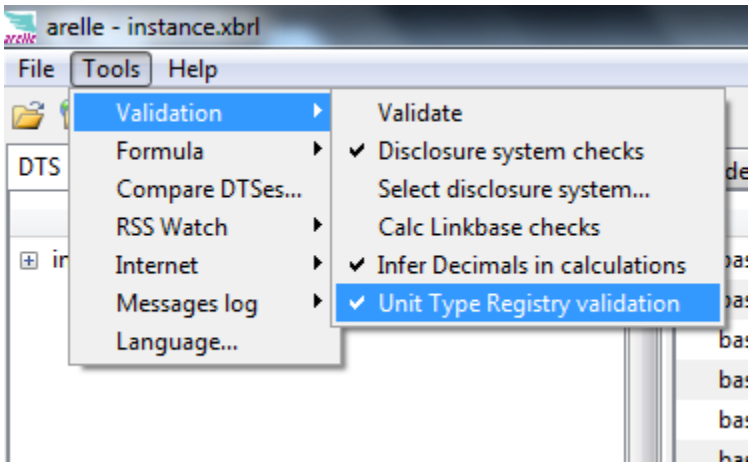
“Tables” switches the table rendered in the screen. To visualize a certain table on the screen, you have to select it from this menu.

“Language” allows selecting the language of the text appearing on the rendering. ***Warning!*** Arelle’s language is set by default to your system language. Depending on the taxonomy, the rendering will show no text if that language is not supported by that taxonomy. If that happens, just switch the view’s language using this option.

3.3 File Validation

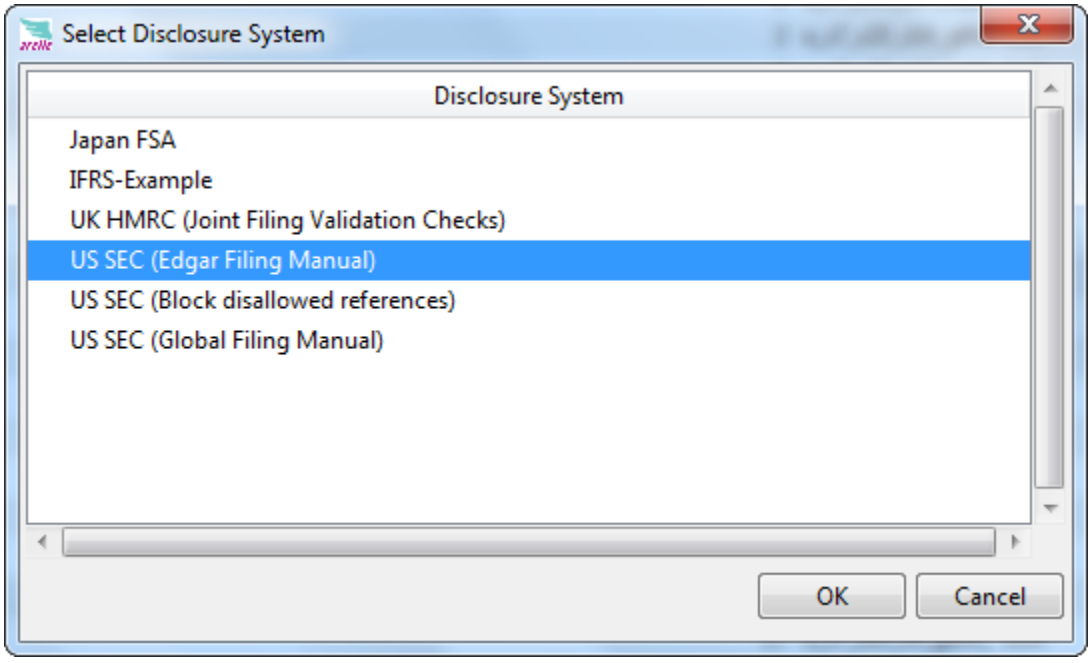
For validating a file, some options must be set before.

To check and set those options, go to Tools->Validation in the Menubar.



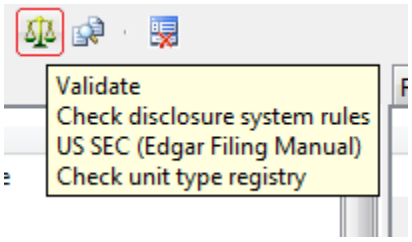
This screen will allow you to select the specific checks that the validation process will carry out.

If you select “Disclosure system checks”, you will need to select a disclosure system, which can also be changed with the “Select Disclosure System” option, and then by selecting one of the disclosure systems offered in the available window.

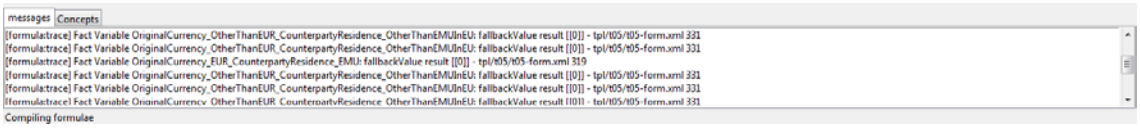


Once these options have been set, you can start the Validation process by clicking “Validate” on the Validation menu, or you can click on the Validation icon placed in the Toolbar. If you

highlight that icon, you can preview the options configuration before performing the validation process.



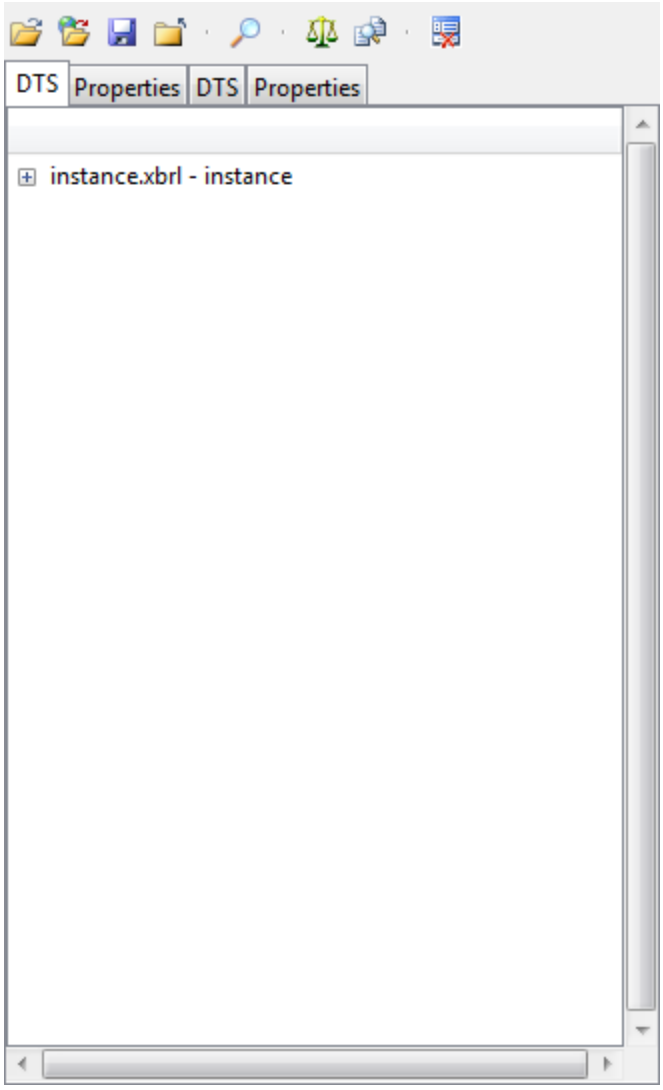
Then the validation process will start and all the log messages will appear on the messages area from the main workspace.



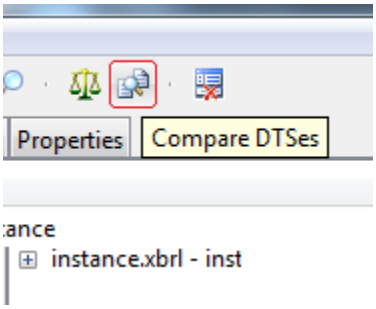
After the validation process, the result will appear in this area.

3.4 DTS COMPARISON

Arelle also allows performing a DTS comparison. To do this, it is necessary to open two files, using the previous procedure to open those files sequentially. Both files will appear in the explorer, in the left-hand workspace area.



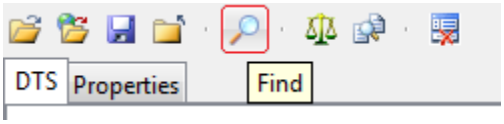
Once those two files are open, select Compare DTSes, or click on the corresponding icon from the Toolbar. This will generate a report file containing the result of the comparison, and you will be asked where to save it.



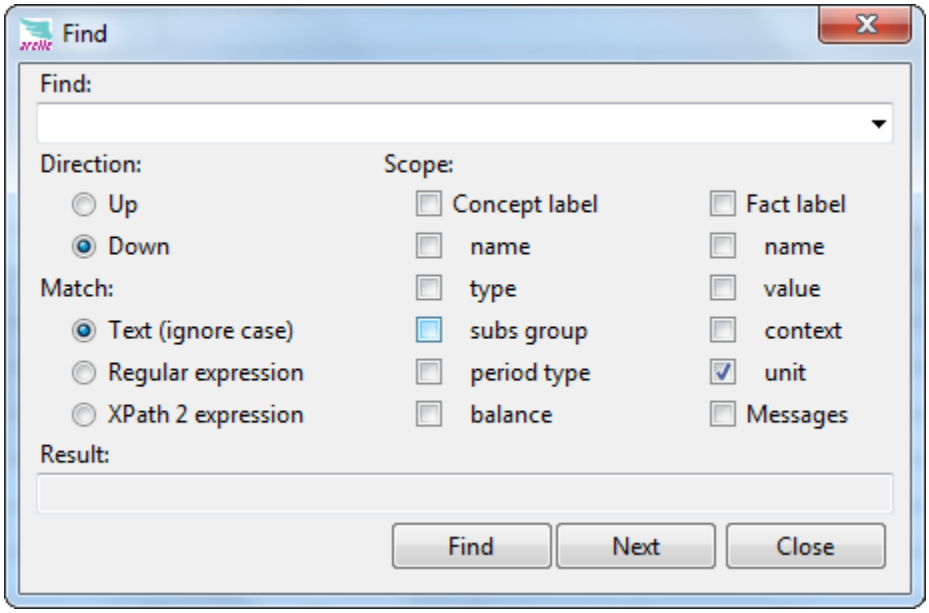
3.5 Other operations and options

Find

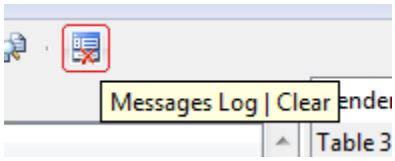
Click on the “Find” button from the Toolbar



to search through the document. The window that appears sets the options for searching.



Clear Messages Log



The last icon from the Toolbar just clears the messages area.

RSS Watch: RSS Watch on the Tools section from the Menubar implements the RSS Feed Processing Control.

Internet: Internet sets some connection options.

Formula: Allows for formula tracing.

Messages Log: Clear Log messages and save them to an external file.

Language: Changes the application language.

4. USING ARELLE (COMMAND LINE)

For executing Arelle from the command line in Windows, the file to be executed from a terminal window is “arelleCmdLine.exe”, located in the Arelle installation directory.

Note the terminal directory should be the folder of Arelle. Example: Change Directory to Arelle

```
C:\>cd C:\Program Files\Arelle
```

and the type “arelleCmdLine.exe”

```
C:\Program Files\Arelle> arelleCmdLine.exe
```

Other option is to write the complete route to the program “arelleCmdLine.exe”,

```
C:\>C:\Program Files\Arelle\arelleCmdLine.exe
```

If the application has been downloaded from the sources, the Python3 script that starts the command line mode is “arelleCmdLine.py”, located in the Arelle installation directory.

The usage for this command is: arelleCmdLine.exe [Options]

*(NOTE: Please check the updated list of options executing “**arelleCmdLine.exe -h**”, or reading the official website at http://arelle.org/documentation/command-line-operation/)*

Options

-h, --help: show this help message and exit

-a, --about: Show product version, copyright, and license

--version: (Note: two hyphens) show program's version number and exit.

-f FILENAME, --file=FILENAME: FILENAME is an entry point, which may be an XBRL instance, schema, linkbase file, inline XBRL instance, testcase file, testcase index file. FILENAME may be a local file or a URI to a web located file.

-d DIFFFILENAME, --diff=DIFFFILENAME: FILENAME is a second entry point when comparing (differentiating) two DTSes producing a versioning report.

-r VERSREPORTFILENAME, --report=VERSREPORTFILENAME: FILENAME is the filename to “save as” the versioning report.

-v, --validate: Validate the file according to the entry file type. An XBRL file is validated according to XBRL validation 2.1, calculation linkbase validation, if either --calcDecimals or --calcPrecision are specified, and SEC Edgar Filing Manual (if --efm selected) or Global Filer Manual disclosure system validation (if --gfm=XXX selected). In the case of a test suite or testcase, the test case variations are individually so validated.

--calcDecimals: Specify calculation linkbase validation inferring decimals.

--calcPrecision: Specify calculation linkbase validation inferring precision.

--efm: Select Edgar Filer Manual (U.S. SEC) disclosure system validation.

--gfm=GFMNAME: Specify a Global Filer Manual disclosure system name and select disclosure system validation.

--utr: Select validation with respect to Unit Type Registry.

--csvDTS=CSVDTS: Write DTS tree into CSVFILE.

--csvFacts=CSVFACTLIST: Write fact list into CSVFILE.

--csvFactCols=CSVFACTLISTCOLS: Columns for CSVFILE.

--csvConcepts=CSVCONCEPTS: Write concepts into CSVFILE.

--csvPre=CSVPRE: Write presentation linkbase into CSVFILE.

--csvCal=CSVCAL: Write calculation linkbase into CSVFILE.

--csvDim=CSVDIM: Write dimensions (of definition) linkbase into CSVFILE.

--csvTestReport=CSVTESTREPORT: Write test report of validation (of test cases) into CSVFILE.

--logFile=LOGFILE: Write log messages into file, otherwise they go to standard output.

--formulaParamExprResult: Specify formula tracing.
--formulaParamInputValue: Specify formula tracing.
--formulaCallExprSource: Specify formula tracing.
--formulaCallExprCode: Specify formula tracing.
--formulaCallExprEval: Specify formula tracing.
--formulaCallExprResult: Specify formula tracing.
--formulaVarSetExprEval: Specify formula tracing.
--formulaVarSetExprResult: Specify formula tracing.
--formulaAsserResultCounts: Specify formula tracing.
--formulaFormulaRules: Specify formula tracing.
--formulaVarsOrder: Specify formula tracing.
--formulaVarExpressionSource: Specify formula tracing.
--formulaVarExpressionCode: Specify formula tracing.
--formulaVarExpressionEvaluation: Specify formula tracing.
--formulaVarExpressionResult: Specify formula tracing.
--formulaVarFiltersResult: Specify formula tracing.

5. USING ARELLE (API)

Arelle can also be used as an API.

The API requires a Controller (subclass of "Cntlr.py") to initiate modelManager operations. The controller must have two feedback methods, "addToLog" (for messages) and "showStatus" (for status messages, if so desired). "CntlrCmdLine.py" will be explained to demonstrate the minimum API:

modelManager = ModelManager.initialize(self)

Initializes the model manager, and provides the calling instance (self), needed for the callbacks to "addToLog" and "showStatus".

filesource = FileSource.FileSource("c:\test\abc.xsd")

A filesource instance is a wrapper for files that may be ordinary files or zip/xfd/frm archive files. If the file is not an archive, the filesource just retains the file path, but if it is an archive, then contents of the archive (such as an instance document to read first) can be specified by the "filesource.select" ("instance.xml") method.

self.modelManager.validateInferDecimals

Select calculation linkbase checks, inferring decimals, to be performed by the "Validate" operation.

self.modelManager.validateInferPrecision

Select calculation linkbase checks, inferring precision, to be performed by the "Validate" operation.

modelManager.validateDisclosureSystem = True

Used to inform the modelManager when Disclosure System rules validation (such as U.S. SEC Edgar or Japan FSA) is desired. Must be set before loading an entry point document. Edgar Filing Manual and Global Filing Manual are currently implemented.

modelManager.disclosureSystem.select("efm")

Used to select which disclosure system rules to apply (e.g., U.S. SEC Edgar, Japan FSA). The parameter must match one of the disclosure system names in the file lib/disclosuresystems.xml. For example, the file entry names="Japan FSA|jp-fsa|fsa" means that any of these three name alternatives would specify the Japan FSA rule set.

self.modelManager.validateUtr

Select Unit Type Registry validation is to be performed by the “Validate” operation.

modelManager.load(filesource, “subsequent action”)

Asks the modelManager to load the entry point specified by the filesource. If successful, the prompt specified by the second argument will be displayed to the “showStatus” callback method.

modelManager.validate()

Performs validation applicable to the entry point (including that specified by the “validateSEC" option, if the entry file is an instance or DTS). Any errors and other messages are provided to the “addToLog” callback method.

To compare two DTSes and produce a versioning report, please replace modelManager.validate(), above, with these:

modelManager.load(diffFilesource, _(“views loading”))

modelManager.compareDTSes (versioningReportFileName)

The first .load represents the “fromDTS”, the second .load represents the “toDTS”. The file name provided by “compareDTSes” is that used to save the versioning report.

modelManager.close()

Closes the last-opened entry point and releases resources.

Note that two entries have been loaded, so there should be two modelManager.close() calls, which act in LIFO manner to close the corresponding .load()’ed entry.

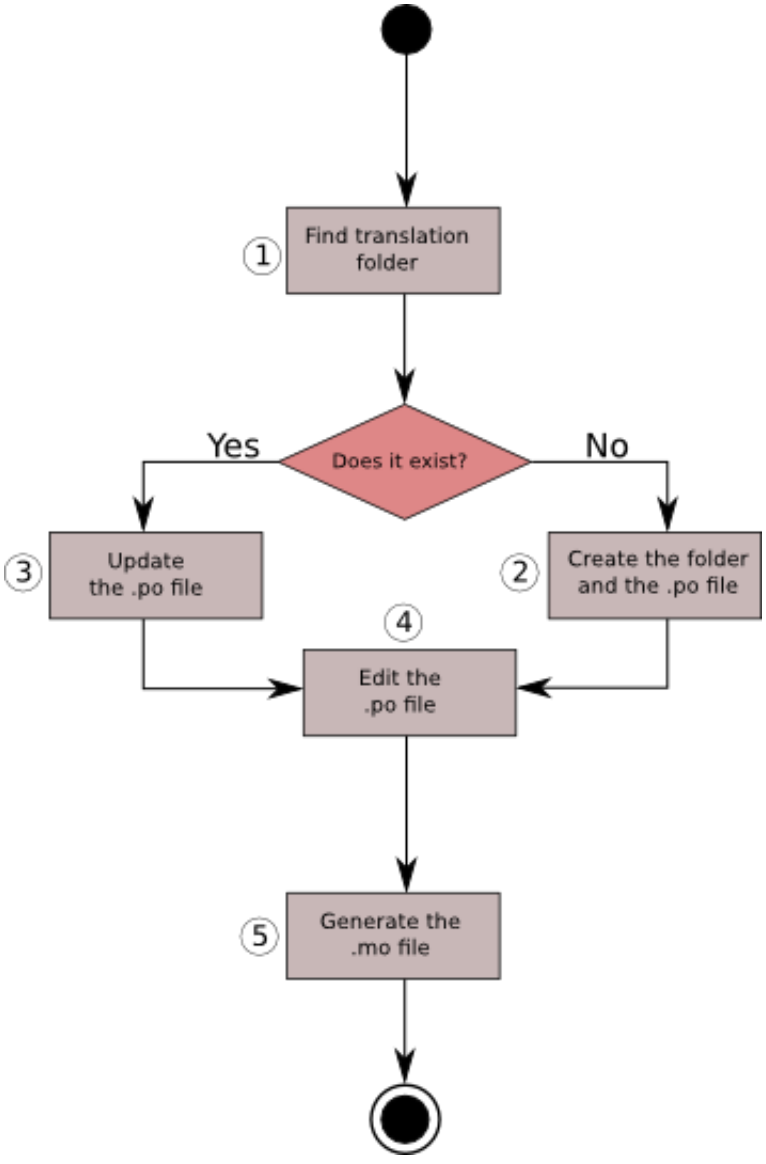
ANNEX: TRANSLATING ARELLE TO YOUR LANGUAGE

Note: Once you created an Arelle translation to your language, please contribute it to www.arelle.org project for the benefit of your linguistic community. Thanks!

How to translate Arelle to your local language

Arelle uses the "gettext" GNU internationalization and localization library (i18n).

For translating the application, translators will follow the standard procedure for translating applications using such library.



1.- Find the translation folder

The first step will be to see if there is an existing translation, which you can see by accessing the *"locale"* folder under the Arelle directory and check if there is a folder named with the corresponding language code for the translation desired (e.g. *es* for Spanish, *fr* for French, etc..). Those language codes are specified in the ISO 639-1 standard.

2.- Create the folder and the .po file

If there is no folder, you will create it, and also create another folder inside of it called *"LC_MESSAGES"* and copy inside the *"messages.pot"* file situated in the *"locale"* directory, rename it with the language code followed by the *".po"* extension (e.g. *es.po* for Spanish, *fr.po* for French, etc...).

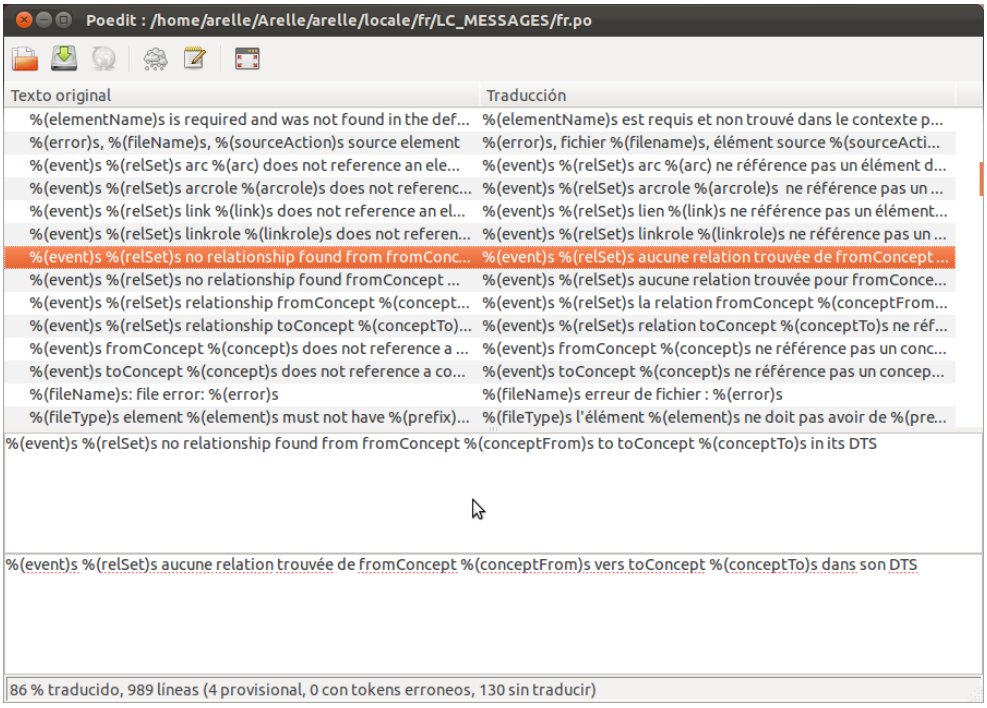
For example, the Spanish translation file will be placed like this:*"\locale\es\LC_MESSAGES\es.po"*

3.- Update the .po file

If the folder already exists, and contains a *".po"* file, some strings could have been added to the source code after the last modification. For adding these new strings to the existing *".po"* file, copy the *"messages.pot"* file from *"locale"* to the directory where the *".po"* file is placed and execute: *"msgmerge LANGCODE.po messages.pot -o LANGCODE.po"* for merging both files into a new one, keeping the strings already translated. After doing that, delete the *"messages.pot"* copy you have just made, as you don't need it anymore.

4.- Edit the .po file

Once the ".po" file is ready, you can start editing it and complete the msgstr entries with the translation of the msgid strings.



This process can be made by using a text editor, but it is recommended to use some software intended to edit those kinds of files. There are plenty of tools for doing the job, and there is a section dedicated to that in the "gettext" documentation. In the image above, you can see one of them, ***poedit***.

5.- Generate the .mo file

Once the translation is done, it is necessary to generate the *".mo"* file, which is a compiled version of the *".po"*.

Some tools like "poedit" do this task automatically; if not, the command to execute it will be

"pocompile -i LANGCODE.po -o LANGCODE.mo", as the output file must be named with the same language code.

If the process has been done correctly, the next time you start Arelle, the language you just generated will be available.

.po and .pot files basic format

".po" and *".pot"* files are composed of a header with this format:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR ORGANIZATION
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
msgid ""
msgstr ""
"Project-Id-Version: arelle\n"
"POT-Creation-Date: 2011-10-10 14:41+Pacific Daylight Time\n"
"PO-Revision-Date: 2011-10-18 14:51+0200\n"
"Last-Translator: Name <email@domain.com>\n"
"Language-Team: Team Name <email@domain.com>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n != 1);\n"
```

where information about translation and translator is included. (Check the "gettext" documentation to fill in the Plural-Forms entry for your language.)

And now the rest of the set of entries are included according to this format:

```
#: arelle\CntlrCmdLine.py:24
msgid ""
"FILENAME is an entry point, which may be an XBRL instance, schema,
linkbase "
"file, inline XBRL instance, testcase file, testcase index file.
FILENAME "
"may be a local file or a URI to a web located file."
msgstr ""
```

Each of these entries is divided into three parts. In the first one, marked in blue, there is a commentary referring to the source code file and the line where such string is located.

Marked in green, there is the "msgid", which is the original string located in the source code, original language.

Marked in red, there is the "msgstr", which is what the translator has to fill in with the corresponding translation for the msgid string above.

If you use a *".po"* file editor, the tool will hide the entire format's complexity to the translator. It will show the contents and allows performing the translation process in a friendly manner.