Cluster of Excellence
ASIA AND EUROPE
IN A GLOBAL CONTEXT

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

HRA – Visual Resources | Tutorial

Manual for
VRA Core 4 Transform Tool

(Version 0.0.4.)

Cluster of Excellence "Asia and Europe in a Global Context"
Karl Jaspers Centre for Advanced Transcultural Studies
Voßstraße 2, Building 4400, 69115 Heidelberg, Germany

Phone: +49 (0) 6221 - 54 40 06
info@asia-europe.uni-heidelberg.de
www.asia-europe.uni-heidelberg.de

Version: 20.08.2015 | Author: MA

Contents

1. Introduction and project history

The *VRA Core 4 XML Transform Tool* converts descriptive image metadata from flat tables (.csv) to structured VRA Core 4 XML. It makes use of a template with predefined headers. Users may work with the tool in a simple or advanced mode. The transform tool also offers XML validation and provides feedback in case of errors.

This manual covers whole the workflow of using the *VRA Core 4 XML Transform Tool* from preparation of data via uploading data, adjusting usage settings, data transformation, data validation and handling validation feedback throughout to downloading the XML files. It also covers the (experimental) generation of RDF XML and includes a number of appendices for further reference.

Development of the *VRA Core 4 XML Transform Tool* was based on an initiative by Susan Jane Williams and resulted in a successful application by Matthias Arnold for a Project Grant funded by the Visual Resources Association Foundation (VRAF) in Fall 2014. Additional support was given by the Heidelberg Research Architecture (HRA) at the Cluster of Excellence "Asia and Europe in a Global Context", Heidelberg University.

> "Matthias Arnold, Heidelberg Research Architecture, University of Heidelberg, will use the award to support the development of the VRA Core 4 XML Transform Tool. This tool will enable any user who can supply descriptive image metadata in a standardized CSV form (comma separated values, e.g. via Excel) to create validating VRA Core 4 XML. Useful on its own, this XML also represents an important step for further transformations to other XML schemas, like RDF to support output of data as LOD (Linked Open Data). The tool is being developed in consultation with both the VRA Data Standards and the Core OC Committees, will be shared with the image management community during beta-testing for feedback, and will be open-source and freely available upon completion. Susan Jane Williams (Independent Cataloging and Consulting Services) will serve as coordinator for the mapping templates and will help create documentation, demonstrations (sample records and mappings), and further mapping help in use of the tool. We are confident that this tool will have broad value and impact within the visual resources community, and for the VRA Core 4 standard."
> "2014-2015 VRA Foundation Project Grant Winners Named", VRAF News & Events, December 19, 2014

The template csv files and the mappings to XML were developed by Susan Jane Williams, Greg Reser, and Matthias Arnold. Implementation was coordinated by Heidelberg, programming was conducted by HRA's senior developer Matthias Guth.

Please note:
The tool is still in BETA version, currently v.0.0.4. Any feedback on issues, bugs or problems encountered is very much appreciated and should be directed to the developers (see section "Contact").

The *VRA Core 4 XML Transform Tool* is developed as Open Source, its source code can be found at: https://github.com/exc-asia-and-europe/csv2xml

Note: This user manual will be updated with every ne feature or functionality. The most current version can be found on GitHub:
https://github.com/exc-asia-and-europe/csv2xml/tree/master/doc.

2. Data preparation

Data transformation is based on templates. The *VRA Core 4 XML Transform Tool* uses **predefined headers** to identify data in the template and write the values to the correct elements or attributes in XML. To be able to transform your .csv data into XML you have to use these headers.

Any descriptive image metadata can be used for transformation, be it an export from your institution’s Digital Asset Management System, metadata embedded in images, or a personal image database. You only need to do two things for preparation:

1. Use the predefined template headers
2. Have the data exported as flat comma separated value (.csv) file

2.1. Predefined headers

Currently, the template offers 224 columns with fixed headers for your data.

These include references to your local system, like “IMAGE_Filename” or “LOCAL-REPO-NAME”. This also includes multiple values for a number of elements, like up to three agents and up to eight subjects for the WORK record. The template also allows including references to controlled vocabularies, like the name and Ref-ID of a subject in a vocabulary, and data types controlled by VRA Core 4 schema.

For example, each of the eight subjects of a WORK record can be expressed in four columns:

WORK_Subject1-8
WORK_SubjectType1-8
WORK_SubjectVocab1-8
WORK_SubjectRefid1-8

You do not need to fill each column with data. The sequence of columns (headers) may be changed and you can even delete columns you will not need. It is, however, essential **not to change the headers** themselves, i.e. not their “text” or “values”.

The complete list of headers can be found in "Appendix: Columns Full Template" below.

2.2. Repeated values

For a number of the eighteen elementSet’s of VRA Core 4 XML for WORK and IMAGE records elements can be repeated. These values usually are combined with other information, like references to authority files.

WORK
- Agent: 3
- CulturalContext: 2
- Date: 2
- Location: 2
- Material: 4
- Measurements: 2 sets
- Relation: 2 (plus 1 work-image)
- Style/Period: 4

- Subject: 8
- Technique: 4
- Title: 2 (preferred/alternative)
- Worktype: 3

IMAGE

- Subject: 3

An expanded list of repeated elements can be found below, in “Appendix: Elements, displays and repetitions.”

2.3. Display values

The template is designed to include data for each of the eighteen elementSet’s of VRA Core 4 XML for WORK and IMAGE records. For a large number of elementsSets structured (or “qualified”) information can be transformed. All elementSets have at least one display element.

For more information about which elementSet will only contain display values and which will also include other elements and attributes please refer to “Appendix: Elements, displays and repetitions” below.

2.3.1. *Auto-filling displays*

For The *VRA Core 4 XML Transform Tool* allows transforming both, structured data for sub-elements and attributes, and unstructured data for display values.

If users do NOT have data for display in their .csv file, *VRA Core 4 XML Transform Tool* will automatically construct the display element based on available data from the structured information.

For example, if the template contains

WORK_AgentDisplay	
WORK_Agent1NameType	personal
WORK_Agent1Name	Michelangelo Buonarroti
WORK_Agent1Role	painter
WORK_Agent1Attribution	school of

The transform tool will also create content for the display element:

<display>school of Michelangelo Buonarroti (painter)</display>

If users provide data for the display element (in the example “WORK_AgentDisplay”) these values will be used and not changed.

2.4. Controlled data

VRA Core 4.0 in the restricted version uses controlled type lists and date formats.

2.4.1. *Data type values*

For the **type values** please refer to the VRA Core 4.0 Restricted Schema Type Values document.

2.4.2. ***Date values***

For **date values** there is no pdf file, but the vra-strict.xsd states for dateValueType:

> “…Defines a date which follows the ISO 8601 date format, and allows right truncation. […] In brief, the following formats are allowed:
>
> present
> 2006 (2006)
> 2006-12 (December, 2006)
> 2006-12-31 (31 December 2006)
> -44 (44 BCE)
> -44-03 (March, 44 BCE)
> -44-03-15 (15 March, 44 BCE)
> -10000000 (10 Million Years Ago)
> -100000000000 (100 Billion Years Ago)”
>
> *Source: http://loc.gov/standards/vracore/vra-strict.xsd, last accessed August 3, 2015*

To be able to convert your data into valid VRA Core 4 XML these type values must be used and date format rules must be followed.

The transform tool will assist you as far as possible.

For example, in <date> the sub-element <latestDate> must not be empty. But if you only entered <earliestDate> the tool will automatically fill <latestDate> with the <earliestDate> value.

If you accidentally mistyped a controlled type value, the validation will point you to this error in the validation results (see chapter “Validation errors” below).

3. Introducing the template

3.1. Downloading the template

The **full template** for *VRA Core 4 XML Transform Tool* is available for download at

https://github.com/exc-asia-and-europe/csv2xml/tree/master/doc

in these formats:

- Comma-separated values textfile (.csv)
- Excel (.xslx)
- OpenDocument spreadsheet (.ods)

3.2. The template explained

The template consists of three worksheets.

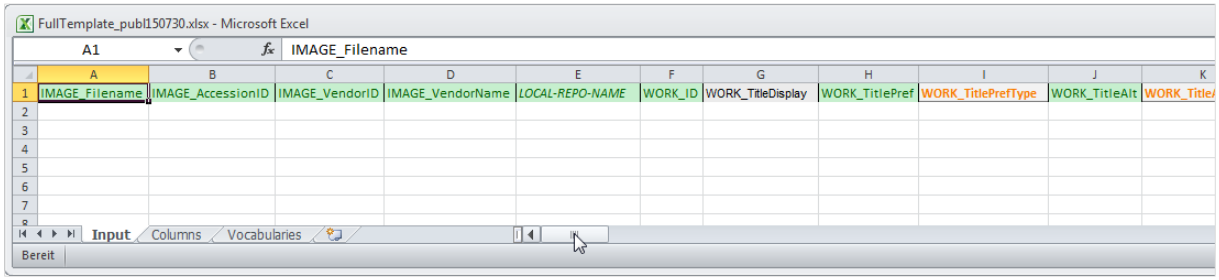


Figure 1: Full template - Input workbook (in Excel)

The "**Input**" worksheet contains all the columns with their respective header in horizontal layout (colour coded), so that it can be used for editing data.

The "**Columns**" worksheet contains two columns. In the first, all headers are arranged vertically and colour coded, while in the second some hints to the content of the data are given.

The colour codes are included to provide visual feedback in case different types of data are expected for a "field".

Display values are shown in black on grey background.

Type values are shown in orange on light grey. Allowed values are provided in the second column.

References to external vocabularies are shown in dark blue on orange background.

Special content relevant for validation is shown in black on red background, together with the validation rules. This is used for dates.

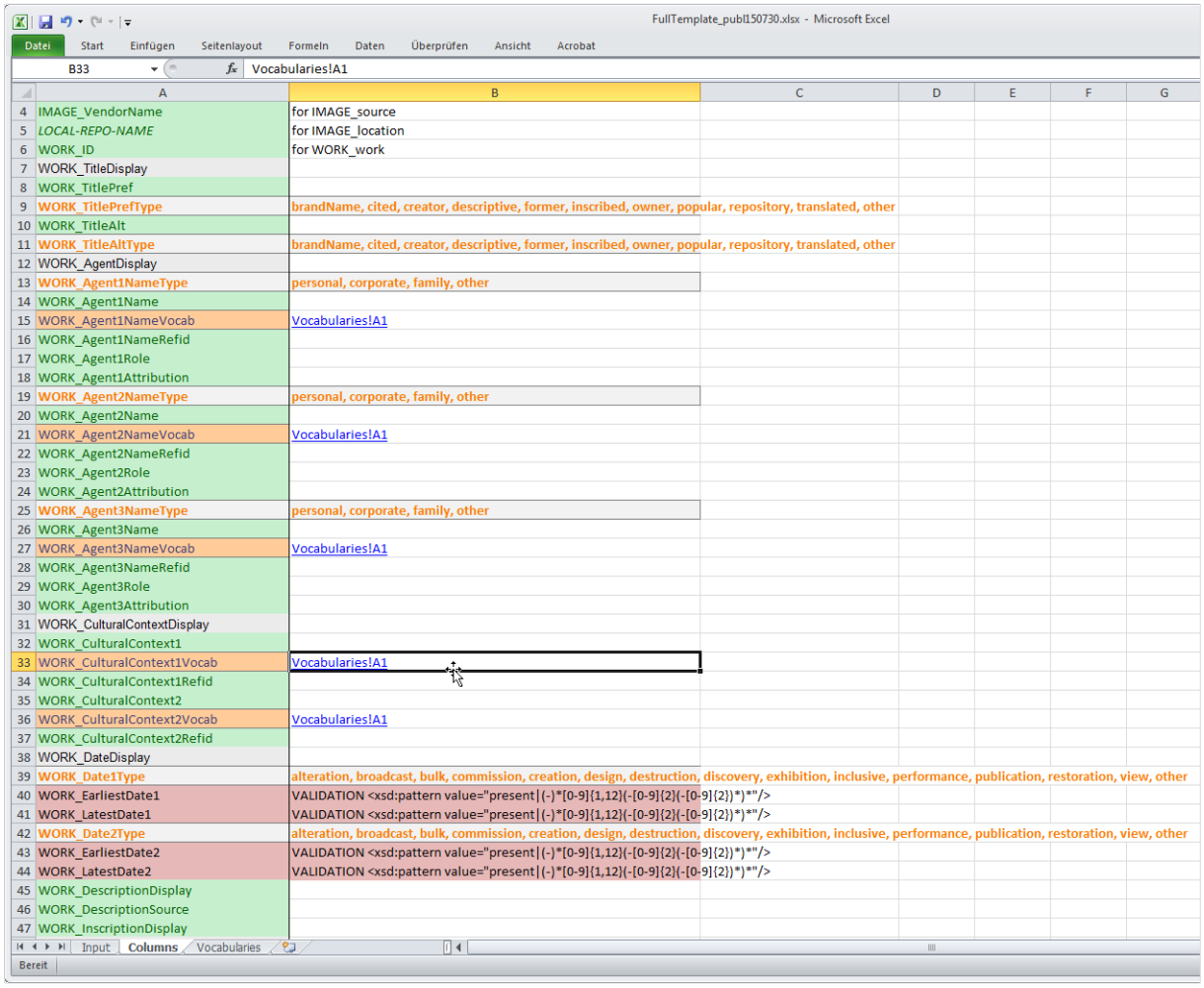


Figure 2: Full template - Columns workbook (in Excel)

The “**Vocabularies**” worksheet contains an (incomplete) list of external authorities or controlled vocabularies. These values are not controlled by the VRA Core 4 schema, but it is strongly recommended to make sure the content is consistent within your data set. This is especially of interest if you intend to transform your data to RDF and use it as Linked (Open) Data.

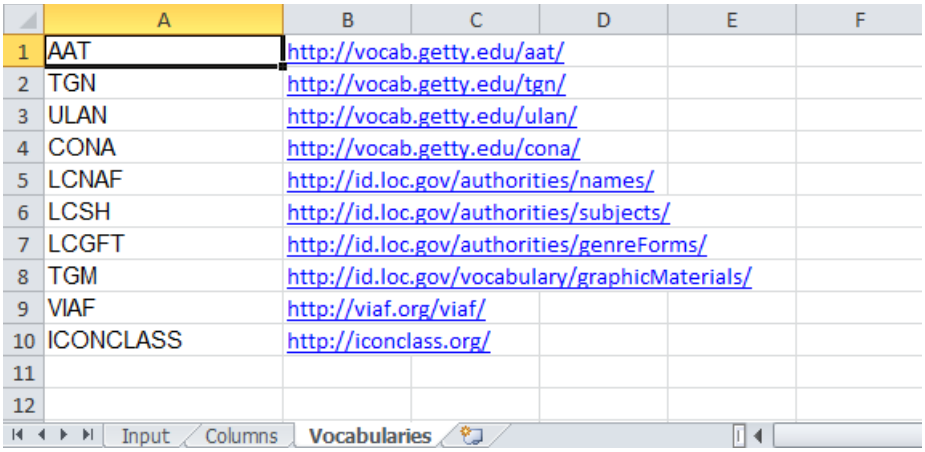


Figure 3: Full template - Vocabularies workbook (in Excel)

3.3. Creating a .csv file

Export records from your local system directly into .csv files if you can include the template headers.

Alternatively, use your preferred spreadsheet software (e.g. Microsoft Excel, OpenOffice Calc, or LibreOffice Calc) as intermediary to adjust the headers and export from there into .csv files. In the doc folder on GitHub an Excel (.xslx) version of the template is available.

Only .csv files can be transformed so make sure you convert spreadsheets (e.g. from Excel or Calc) to .csv text files before transformation. **Use UTF-8 as encoding** to conserve possible special characters in your data.

For a guide on how to convert Microsoft Excel files into comma separated value (.csv) files see “Appendix: Converting .xslx file to .csv using OpenOffice Calc” below.

To be covered:
Other data sources: EMWG export-import tool

4. Accessing the transform tool online

At the moment, the *VRA Core 4 XML Transform Tool* can be accessed online for testing at http://kjc-ws2.kjc.uni-heidelberg.de:8081/exist/apps/csv2xml/index.xq

Please note this is a server dedicated to testing software and developments.

4.1. The basic user interface

The *VRA Core 4 XML Transform Tool* interface offers a Simple and an Advanced Mode.

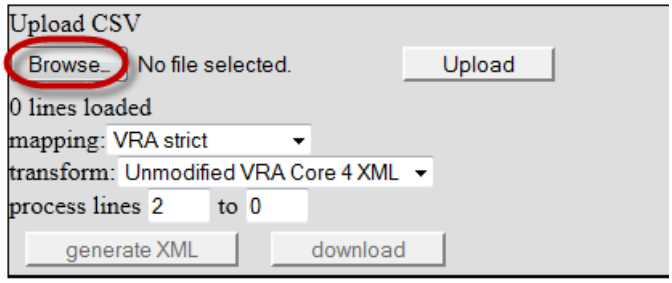
Figure 4: User interface - Simple mode.

In simple mode the XML generation and validation are processed in one step finishing with the optional download of the XML files.

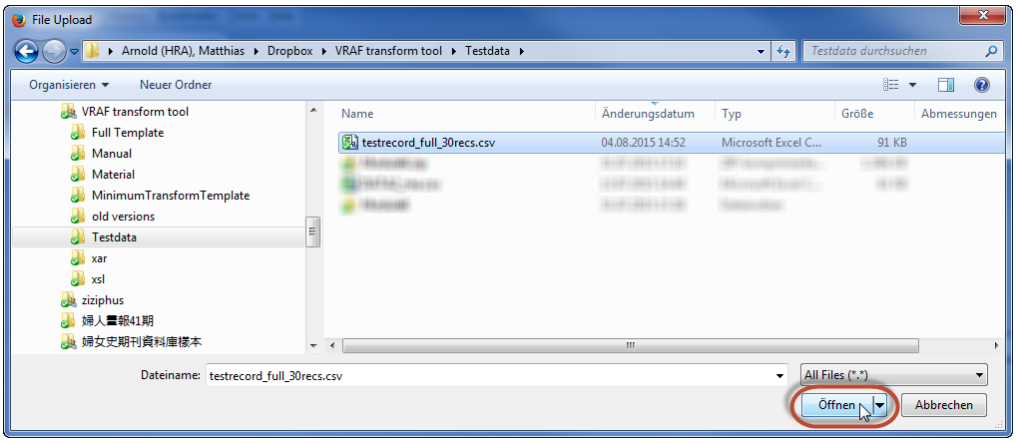
The advanced interface provides additional functionalities for the advanced user including the possibility to apply additional transformations.

5. Uploading a file

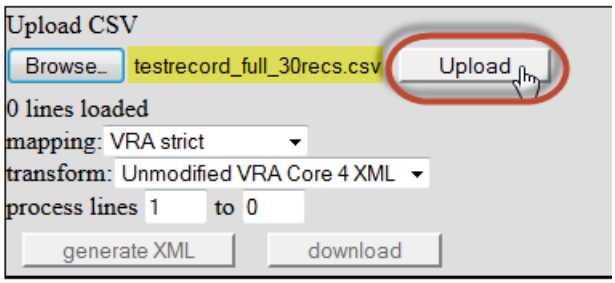
To transform data you need to upload it first. Click the “Browse” button:



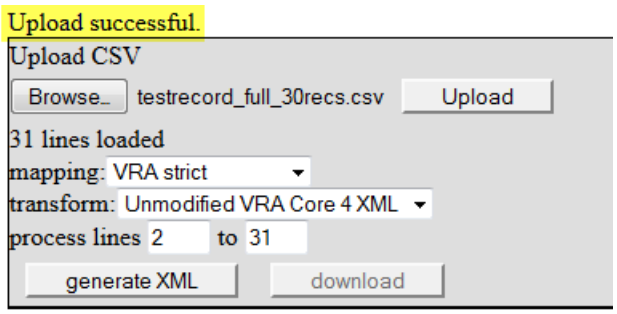
…then select the .csv file on your computer and click “Open”.



The name of the .csv file will be displayed in the interface (here: “testrecord_full_30recs.csv”). Now click on “Upload”.



You will see an “Upload successful” message.



6. Generating VRA Core 4 XML

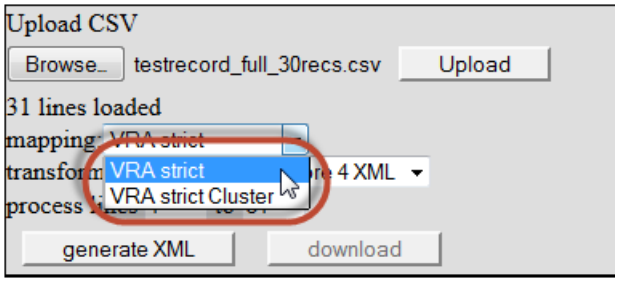
After uploading a .csv file to the *VRA Core 4 XML Transform Tool* you can just hit the "generate XML" button. This will proces all records into VRA Core 4 XML using the default settings.

However, even in Simple Mode you can manipulate how the XML is generated. You have three options:

- Select another mapping
- Select another transform
- Define the range of records to be processed

6.1. Selecting a mapping

The *VRA Core 4 XML Transform Tool* supports generating XML based on different mappings. For VRA Core, two variants are available: "VRA strict" and "VRA strict Cluster".



6.1.1. *VRA strict*

This is the **default mapping**. It is based on the VRA Core 4 restricted version (http://www.loc.gov/standards/vracore/vra-strict.xsd). It uses controlled values for types and date formats.

For more information please refer to http://www.loc.gov/standards/vracore/schemas.html.

6.1.2. *VRA strict Cluster*

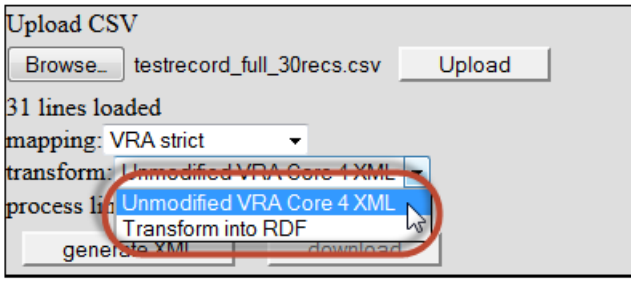
This mapping is an expansion of the restricted VRA Core 4 version developed at the Cluster of Excellence "Asia and Europe in a Global Context", University of Heidelberg. It adds, for example, attributes for multilingual data, role attributes for agents, and an element for geo-coordinates (http://cluster-schemas.uni-hd.de/vra-strictCluster.xsd).

For more information please refer to the document "VRA Core – Extensions".

6.2. Selecting a transformation

The tool supports different transformations.

In the current version of *VRA Core 4 XML Transform Tool* (0.0.4) additional .xsl files need to be added in the source code to become available in the interface. This may be changed in a future version.



Currently, two transformations are implemented:

- Unmodified VRA Core 4 XML
- Transform into RDF (explanations see below)

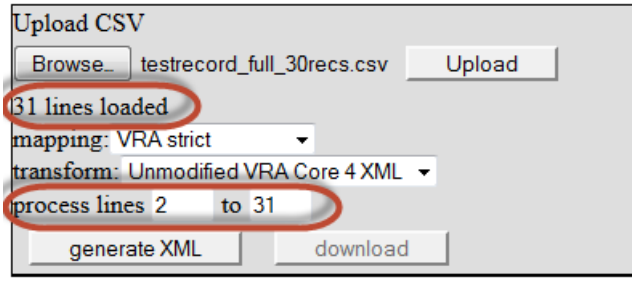
To transform the .csv data into VRA Core 4 XML choose "Unmodified VRA Core 4 XML".

6.3. Defining the records to be processed

By default, all records of an uploaded .csv file will be processed.

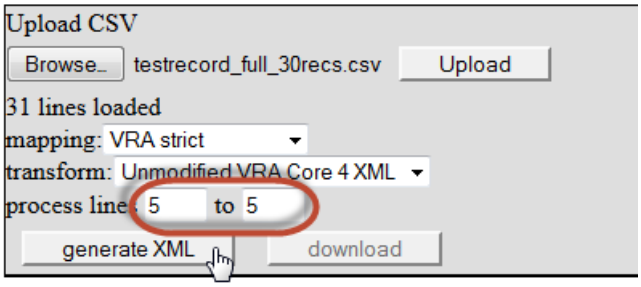
6.3.1. *Calculating records for processing*

Within the .csv file a record equals a line. You may have noticed, that in the example .csv the 30 records result in 31 lines. This is because the first record (first line) always contains the column headers. The *VRA Core 4 XML Transform Tool* "knows" that and will per default only process data beginning with line #2 (i.e. record #1).



6.3.2. *Defining a range of records*

The *VRA Core 4 XML Transform Tool* also allows you to define the range of records to be processed. This is very helpful if you want to check a smaller range of records or even individual ones, look at their xml and test if they validate.

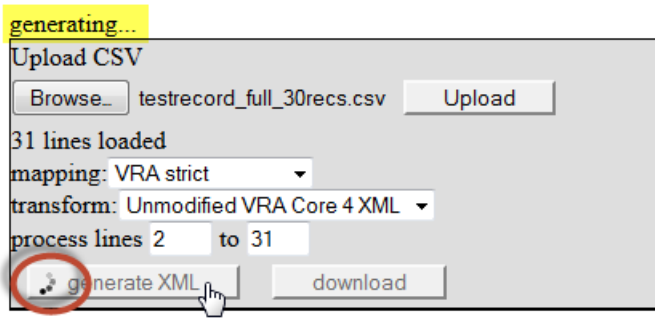


In this example, only line 5 (i.e. record #4) will be processed.

6.4. Generating XML

If all settings are made, click the “generate XML” button.

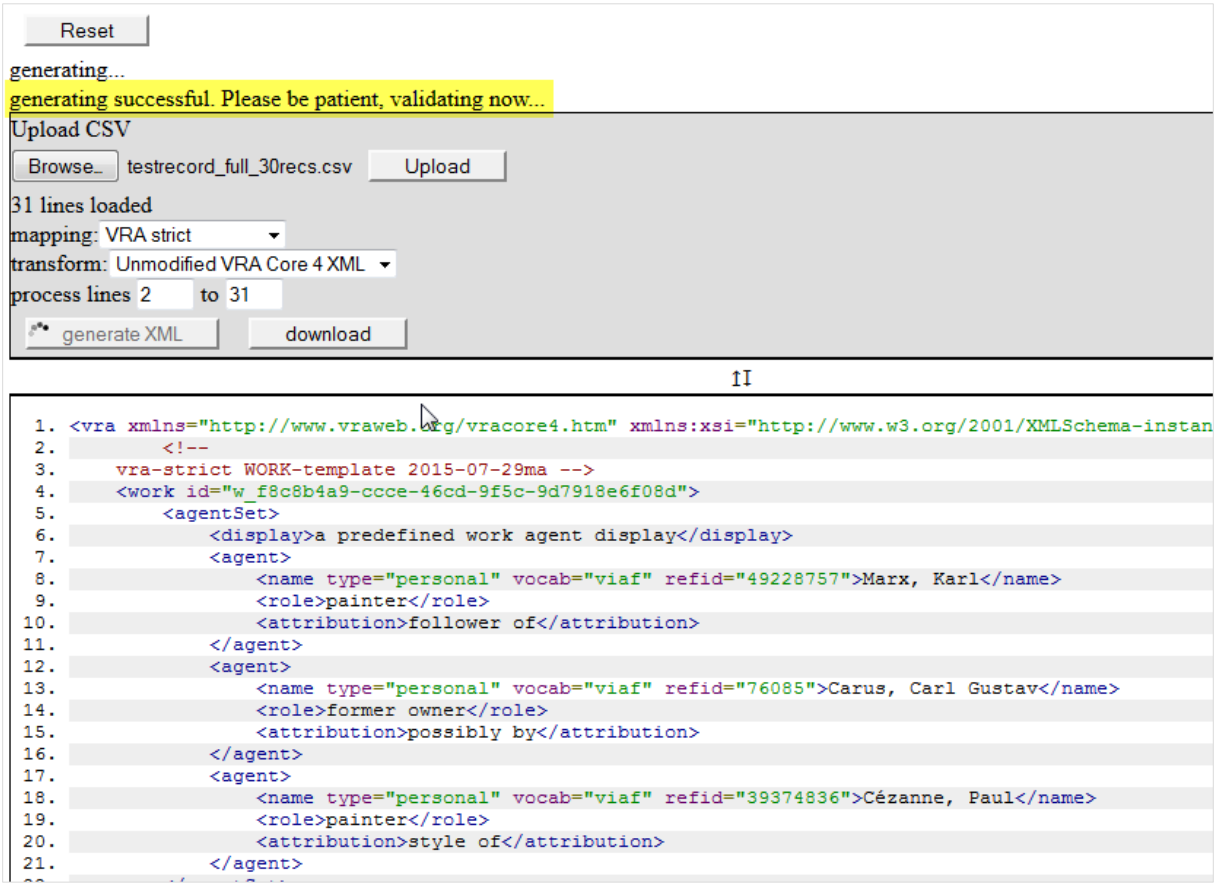
The tool will start generating the XML.



Please note: Processing a large number of records may take some time for XML generation and validation, so please be patient.

If the XML is generated, the code will be displayed below the button. To the top another message will be displayed:

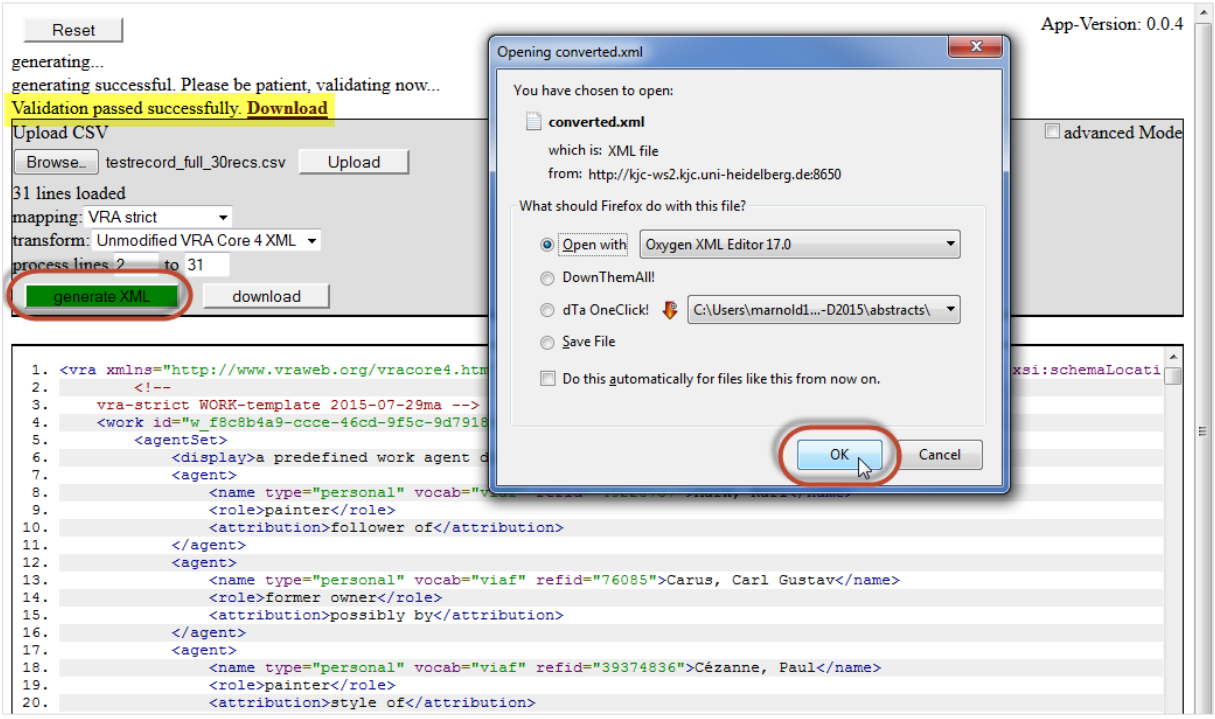
“Generating successful. Please be patient, validating now…”



At the end of the validation another message will be shown:

“Validation passed successfully,” together with a “Download” link. The “Generate XML” button is now green and the download dialog “Opening converted.xml” opens.

The default file name will be “converted.xml”.

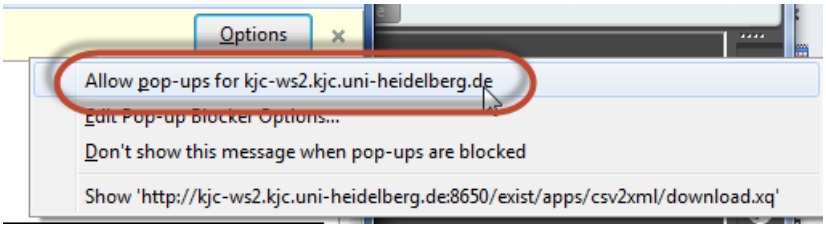


You can save the file or open it in your preferred XML editor.

6.5. Allow pop-ups

If you do not see the download dialog, you might need to check any pop-up blocker. In Firefox it may look like this:

Choos “Allow pop-ups” from the “Options” menue.



7. The interface in Advanced Mode

In the Advanced mode it is possible to apply XSL transforms, choose a schema against which you wish to validate your XML file, and upload your own Schema file. It also separates XML generation from validation.

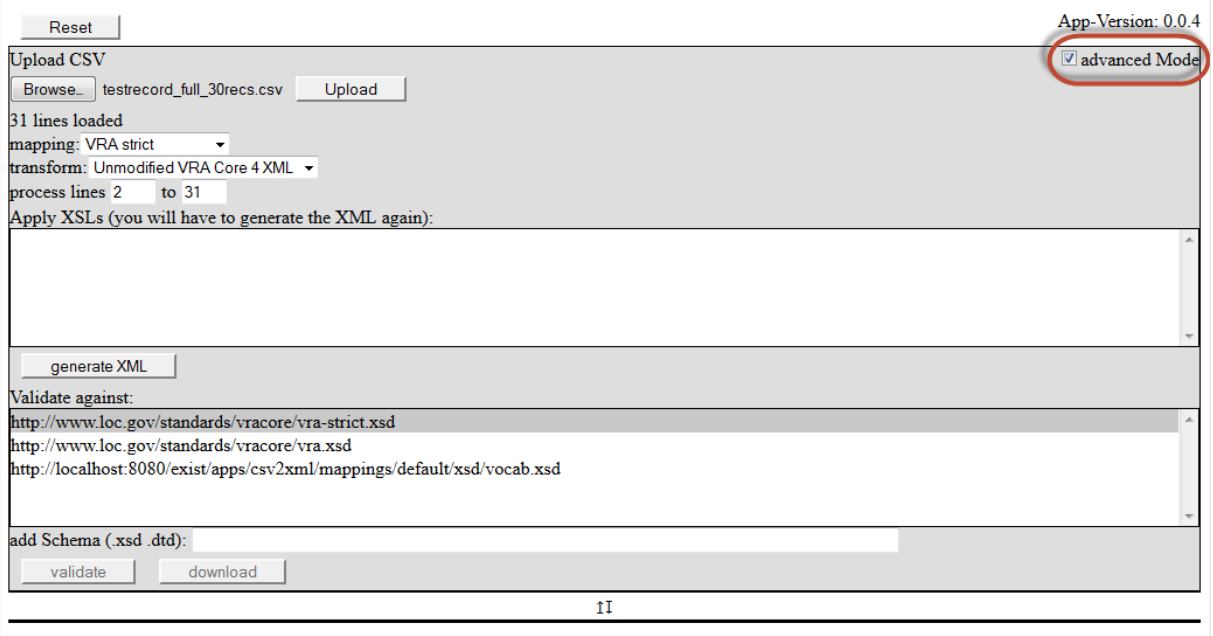


Figure 5: User interface - Advanced mode.

7.1. Applying XSLs

For the default transformation to VRA Core 4 XML this field will remain empty.

If you choose “Transform into RDF” the “VRA2RDF.xsl” will be loaded and applied. For more information about RDF transformation see chapter “Generating RDF” below.

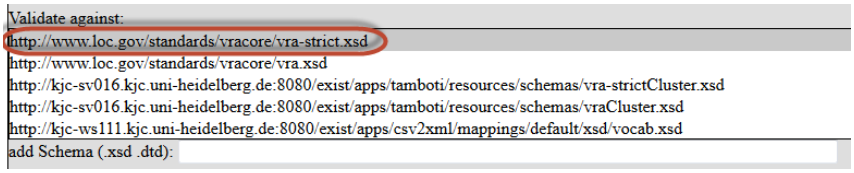
In the current version of *VRA Core 4 XML Transform Tool* (0.0.4) additional .xsl files need to be added in the source code to become available in the interface. This may be changed in a future version.

7.2. Validating the XML

Just by applying a transformation the tool cannot be sure the resulting xml files are valid. The restricted VRA Core schema defines a number of fixed values. If the values in the .csv document do not match the values expected by the schema, validation will fail. We therefore strongly recommend validating the generated data.

7.2.1. *Choosing a schema for validation*

In the “Validate against” box you can choose a schema for validation. Depending on the selected mapping (see above) the respective default validation schema is automatically set.

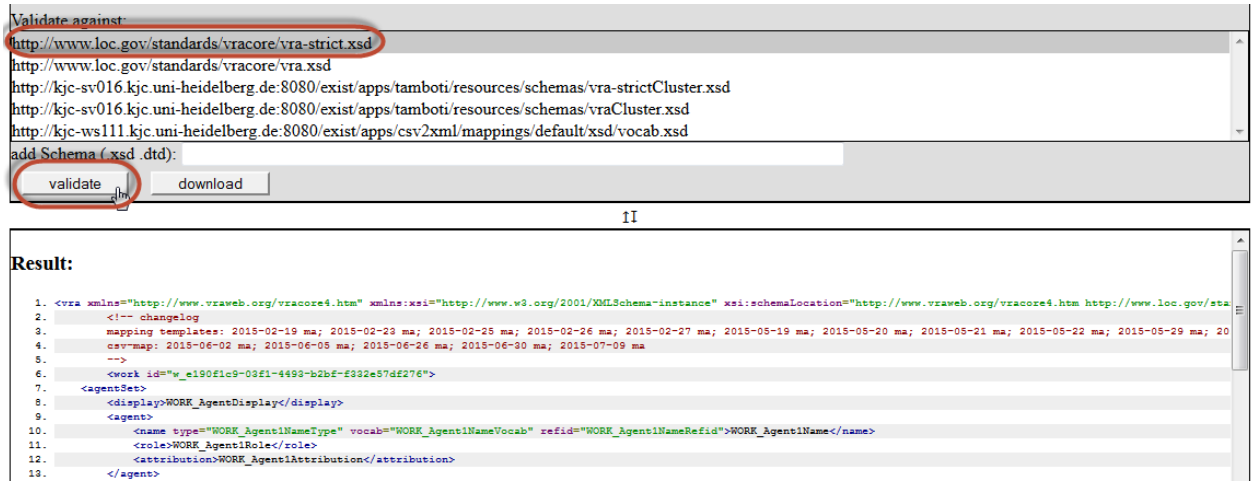


For example, if you selected "VRA strict" as mapping, the corresponding schema http://www.loc.gov/standards/vracore/vra-strict.xsd is automatically set.

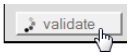
It is possible to choose different validation schemas. Users may also upload their own schema (ALPHA!).

7.2.2. *Starting the validation*

To start the validation process, click “validate”.



Note: Validating a large XML file may take some time, please be patient.



7.2.3. *Getting the validation result*

Once the validation is finished, a new browser tab will be opened with the validation result. In addition, the button’s color will change:

Result NOT VALID

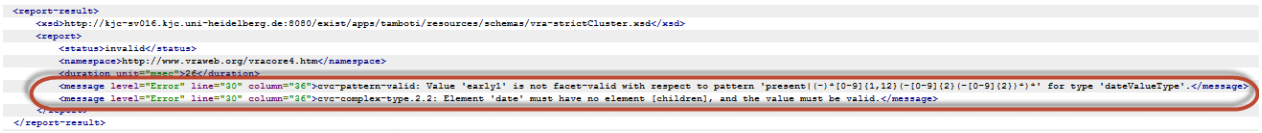
Resul VALID

8. If validation fails

VRA Core 4 in the restricted version uses controlled type lists and date formats. If your data does not match the prescribed rules, your XML will not validate.

In case of problems with the validation, the error messages will be displayed in a separate tab. The messages will indicate both line and column of the respective generated XML document. This is displayed in the interface.

Error messages usually come in pairs. The first line will name the invalid value based on an enumeration in the schema (e.g. type values defined by vra-strict). The second line will indicate the parent element.



8.1. Example 1 (element type):


```
<message level="Error" line="134" column="52">cvc-enumeration-valid: Value 'illustrative' is not facet-valid with respect to enumeration '[brandName, cited, creator, descriptive, former, generalView, inscribed, owner, partialView, popular, repository, translated, other]'. It must be a value from the enumeration.</message>

<message level="Error" line="134" column="52">cvc-attribute.3: The value 'illustrative' of attribute 'type' on element 'title' is not valid with respect to its type, 'titleTypeType'.</message>
```

The error message refers to `line="134" column="52"` in the XML:

```
131.        </textrefSet>
132.        <titleSet>
133.            <display>a_predefined_work_title_display</display>
134.            <title pref="true" type="illustrative">some_w_prefTitle</title>
135.            <title pref="false" type="translated">some_altTitle</title>
136.        </titleSet>
137.        <worktypeSet>
```

The first line of the message states that “illustrative” is not valid because it is not included in the list of possible values: “brandName, cited, creator, descriptive, former, generalView, inscribed, owner, partialView, popular, repository, translated, other”.

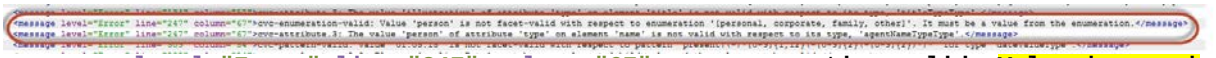
The second line adds that the error occurred within attribute “type” of element <title>, and the provided value for “type” was not valid.

Result:

Values of **`<title type="">`** are controlled by the schema.

You need to delete “illustrative” and use of one of the allowed values, i.e. “brandName, cited, creator, descriptive, former, generalView, inscribed, owner, partialView, popular, repository, translated, other”.

8.2. Example 2 (element type):



```
<message level="Error" line="247" column="67">cvc-enumeration-valid: Value 'person'
is not facet-valid with respect to enumeration '[personal, corporate, family,
other]'. It must be a value from the enumeration.</message>

<message level="Error" line="247" column="67">cvc-attribute.3: The value 'person'
of attribute 'type' on element 'name' is not valid with respect to its type,
'agentNameTypeType'.</message>
```

The error message refers to `line="247" column="67"` in the XML:

```
244.        <agentSet>
245.            <display>school of Michelangelo Buonarroti (painter); formerly attributed to Paul, Priya (collector);
246.            <agent>
247.                <name type="person" vocab="viaf" refid="24585191">Michelangelo Buonarroti</name>
248.                <role>painter</role>
249.                <attribution>school of</attribution>
250.            </agent>
251.            <agent>
252.                <name type="personal" vocab="viaf" refid="277848046">Paul, Priya</name>
```

The first line of the message states that "person" is not valid because it is not included in the list of possible values: "personal, corporate, family, other".

The second line adds that the error occurred within attribute "type" of element <name>, and the provided value for "type" was not valid.

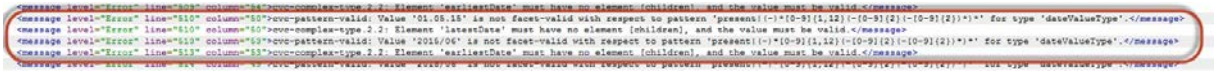
Result:

Values of **`<name type="">`** are controlled by the schema.

You need to delete "person" and use of one of the allowed values, i.e. "personal, corporate, family, other".

8.3. Example 3 (date pattern):

This example consists of two similar problems:



```
<message level="Error" line="510" column="50">cvc-pattern-valid: Value '01.05.15'
is not facet-valid with respect to pattern 'present|(-)*[0-9]{1,12}(-[0-9]{2}(-[0-
9]{2})*)*' for type 'dateValueType'.</message>
<message level="Error" line="510" column="50">cvc-complex-type.2.2: Element
'latestDate' must have no element [children], and the value must be valid.</message>

<message level="Error" line="513" column="53">cvc-pattern-valid: Value '2015/06' is
not facet-valid with respect to pattern 'present|(-)*[0-9]{1,12}(-[0-9]{2}(-[0-
9]{2})*)*' for type 'dateValueType'.</message>
<message level="Error" line="513" column="53">cvc-complex-type.2.2: Element
'earliestDate' must have no element [children], and the value must be
valid.</message>
```

The error message refers to `line="510" column="50" and line="513" column="53"` in the XML:

```
506.        <dateSet>
507.            <display>a predefined work date display</display>
508.            <date type="creation">
509.                <earliestDate>01.05.15</earliestDate>
510.                <latestDate>01.05.15</latestDate>
511.            </date>
512.            <date type="alteration">
513.                <earliestDate>2015/06</earliestDate>
514.                <latestDate>2015/06</latestDate>
515.            </date>
516.        </dateSet>
```

The first lines of each message sat state that "01.05.15" and "2015/06" are not valid because they do not follow the predefined pattern for dateValues.

The second lines add that the error occurred within element <latestDate> (respective <earliestDate>), and the element must be valid.

Result:

Values of **<earliestDate>** and **<latestDate>** are pattern-controlled by the schema.

You need to write the dates in the correct format, i.e. "2015-05-01" or "2015-06".

8.4. Example 4 (work/image ID):

```
<message level="Error" line="239" column="18">cvc-datatype-valid.1.2.1: '2' is not a valid value for 'NCName'.</message>
<message level="Error" line="239" column="18">cvc-attribute.3: The value '2' of attribute 'id' on element 'work' is not valid with respect to its type, 'ID'.</message>
```

The error message refers to `line="239" column="18"` in the XML:

```
238. </image>
239.     <work id="2">
240.       <agentSet>
241.           <display>Carlo
```

The first line of the message states that "2" is not a valid NCName, which stands for "Non-colonized" Names (cf. http://www.w3.org/TR/xmlschema-2/#NCName).

The second line adds that the error occurred within attribute "id" of element <work>, and the provided value was not valid.

Result:

Values of **<work id="">** are regulated by the schema. The "VRA Core4 Element Description" explains in a note: "The XML *id* attribute must begin with a character. The convention used here prefixes a numeric value with w_ for works, c_ for collections, and i_ for images."

You need to provide a work ID that begins with "w_".

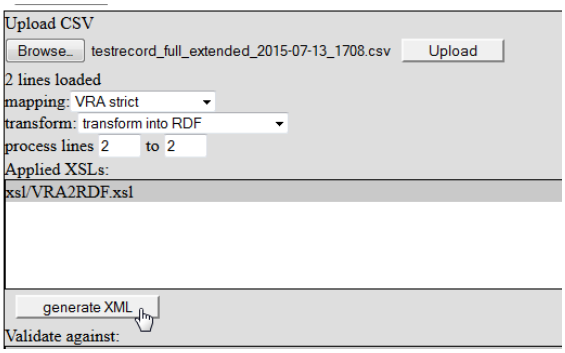
9. Generating RDF XML

In addition to generating VRA Core 4 XML it is possible to transform the data to RDF. At the moment, this feature is still experimental.

The tool makes use of the XSLT stylesheet as provided by the VRA-RDF-Project https://github.com/mixterj/VRA-RDF-Project.

To transform your data to RDF first make sure the .csv data will validate in VRA Core.

Then go back and select “transform into RDF” in the “transform” line. Note that the entry “xsl/VRA2RDF.xsl” appears in the “Applied XSL’s” box.



Now click “generate XML”.

Once generated, the RDF XML will be displayed in the “Result” box and can be downloaded.

10. Contact

If you have problems or questions please contact Matthias Arnold at marnold@asia-europe.uni-heidelberg.de

Contact details
arnold@asia-europe.uni-heidelberg.de
Phone: +49 (0) 6221 - 54 4094
Fax: +49 (0) 6221 - 54 4012
Skype: matz-skype

Office address
Karl Jaspers Centre, Room 005b

MediaLab
Karl Jaspers Centre, Room 005c

Further links
HRA Portal
http://hra.uni-hd.de/

MediaLab Tutorials in Sharepoint
https://sharepoint.urz.uni-heidelberg.de/vjc/kjc/hra/medialab/

11. Appendix: Elements, displays and repetitions

WORK
[workID]

Agent display
3x nameType – name – nameVocab – nameRefid – role

CulturalContext display
2x culturalContext – vocab – refid

Date display
2x type – earliestDate – latestDate

Description display [display = description]
descriptionSource

Inscription display [display = text]

Location display
Location notes
2x type – Name – NameType – NameVocab – NameRefID – ObjType – ObjRefID – Geo – GeoVocab – GeoRefid

Material display
4x material – vocab – refid

Measurements display
2x extent – unit – value1-4 – type1-4

Relation display
2x relatedWork – type
[transform adds work-image link]

Rights display [display = text]

Source display

StateEdition display

Style/Period display
4x stylePeriod – vocab – refid

Subject display
8x subject – vocab – refid – type

Technique display
4x technique – vocab – refid

TextRef display

Title display
titlePref, titlePrefType, titleAlt, titleAltType

Worktype display
3x worktype – vocab – refid

IMAGE
[Filename (-> href); accessionID (-> refid)]

Agent display
1x nameType – name – nameVocab – nameRefid – role

CulturalContext display

Date display

Description display [display = description]

Inscription display [display = text]

Location display
[localRepoName, accession-id]

Material display

Measurements display

Relation display
[transform adds image-work link]

Rights display
Rights notes
1x type – holder – text

Source display
[VendorName, VendorID]
1x sourceType, sourceValue

StateEdition display

Style/Period display

Subject display
3x subject – vocab – refid – type

Technique display
[Technique: <technique vocab="AAT" refid="300237903">digital imaging</technique>]

TextRef display

Title display
1x type, title

WorkType display
[Worktype: <worktype vocab="AAT" refid="300215302">digital images</worktype>]

12. Appendix: Columns Full Template

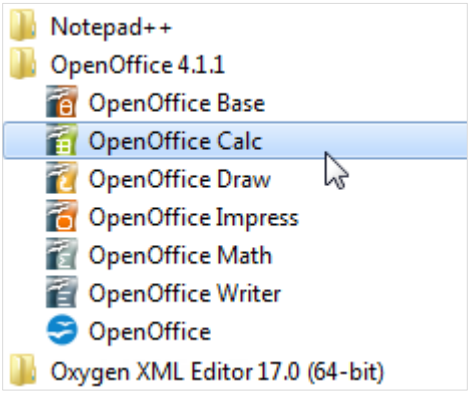
IMAGE_Filename
IMAGE_AccessionID
IMAGE_VendorID
IMAGE_VendorName
LOCAL-REPO-NAME
WORK_ID
WORK_TitleDisplay
WORK_TitlePref
WORK_TitlePrefType
WORK_TitleAlt
WORK_TitleAltType
WORK_AgentDisplay
WORK_Agent1NameType
WORK_Agent1Name
WORK_Agent1NameVocab
WORK_Agent1NameRefid
WORK_Agent1Role
WORK_Agent1Attribution
WORK_Agent2NameType
WORK_Agent2Name
WORK_Agent2NameVocab
WORK_Agent2NameRefid
WORK_Agent2Role
WORK_Agent2Attribution
WORK_Agent3NameType
WORK_Agent3Name
WORK_Agent3NameVocab
WORK_Agent3NameRefid
WORK_Agent3Role
WORK_Agent3Attribution
WORK_CulturalContextDisplay
WORK_CulturalContext1
WORK_CulturalContext1Vocab
WORK_CulturalContext1Refid
WORK_CulturalContext2
WORK_CulturalContext2Vocab
WORK_CulturalContext2Refid
WORK_DateDisplay
WORK_Date1Type
WORK_EarliestDate1
WORK_LatestDate1
WORK_Date2Type
WORK_EarliestDate2
WORK_LatestDate2
WORK_DescriptionDisplay
WORK_DescriptionSource
WORK_InscriptionDisplay
WORK_LocationDisplay
WORK_LocationNotes
WORK_Location1Type
WORK_Location1Name
WORK_Location1NameType
WORK_Location1NameVocab
WORK_Location1NameRefID
WORK_Location1ObjType
WORK_Location1ObjRefID
WORK_Location1Geo
WORK_Location1GeoVocab
WORK_Location1GeoRefid
WORK_Location2Type
WORK_Location2Name
WORK_Location2NameType
WORK_Location2NameVocab
WORK_Location2NameRefID
WORK_Location2ObjType
WORK_Location2ObjRefID
WORK_Location2Geo
WORK_Location2GeoVocab
WORK_Location2GeoRefid
WORK_MaterialDisplay
WORK_Material1
WORK_MaterialVocab1
WORK_MaterialRefid1
WORK_Material2
WORK_MaterialVocab2
WORK_MaterialRefid2
WORK_Material3
WORK_MaterialVocab3
WORK_MaterialRefid3
WORK_Material4
WORK_MaterialVocab4
WORK_MaterialRefid4
WORK_MeasurementsDisplay
WORK_Measurements1Extent
WORK_Measurements1Unit
WORK_Measurements1Value1
WORK_Measurements1Type1
WORK_Measurements1Value2
WORK_Measurements1Type2
WORK_Measurements1Value3
WORK_Measurements1Type3
WORK_Measurements1Value4
WORK_Measurements1Type4
WORK_Measurements2Extent
WORK_Measurements2Unit
WORK_Measurements2Value1
WORK_Measurements2Type1
WORK_Measurements2Value2
WORK_Measurements2Type2
WORK_Measurements2Value3
WORK_Measurements2Type3
WORK_Measurements2Value4
WORK_Measurements2Type4
WORK_RelationDisplay
WORK_RelatedWork1
WORK_RelationType1
WORK_RelatedWork2
WORK_RelationType2
WORK_RightsDisplay
WORK_SourceDisplay
WORK_StateEditionDisplay
WORK_StylePeriodDisplay
WORK_StylePeriod1
WORK_StylePeriodVocab1

WORK_StylePeriodRefid1
WORK_StylePeriod2
WORK_StylePeriodVocab2
WORK_StylePeriodRefid2
WORK_StylePeriod3
WORK_StylePeriodVocab3
WORK_StylePeriodRefid3
WORK_StylePeriod4
WORK_StylePeriodVocab4
WORK_StylePeriodRefid4
WORK_SubjectDisplay
WORK_Subject1
WORK_SubjectType1
WORK_SubjectVocab1
WORK_SubjectRefid1
WORK_Subject2
WORK_SubjectType2
WORK_SubjectVocab2
WORK_SubjectRefid2
WORK_Subject3
WORK_SubjectType3
WORK_SubjectVocab3
WORK_SubjectRefid3
WORK_Subject4
WORK_SubjectType4
WORK_SubjectVocab4
WORK_SubjectRefid4
WORK_Subject5
WORK_SubjectType5
WORK_SubjectVocab5
WORK_SubjectRefid5
WORK_Subject6
WORK_SubjectType6
WORK_SubjectVocab6
WORK_SubjectRefid6
WORK_Subject7
WORK_SubjectType7
WORK_SubjectVocab7
WORK_SubjectRefid7
WORK_Subject8
WORK_SubjectType8
WORK_SubjectVocab8
WORK_SubjectRefid8
WORK_TechniqueDisplay
WORK_Technique1
WORK_TechniqueVocab1
WORK_TechniqueRefid1
WORK_Technique2
WORK_TechniqueVocab2
WORK_TechniqueRefid2
WORK_Technique3
WORK_TechniqueVocab3
WORK_TechniqueRefid3
WORK_Technique4
WORK_TechniqueVocab4
WORK_TechniqueRefid4
WORK_TextrefDisplay
WORK_WorktypeDisplay
WORK_Worktype1
WORK_WorktypeVocab1
WORK_WorktypeRefid1
WORK_Worktype2
WORK_WorktypeVocab2
WORK_WorktypeRefid2
WORK_Worktype3
WORK_WorktypeVocab3
WORK_WorktypeRefid3
IMAGE_AgentDisplay
IMAGE_Agent1NameType
IMAGE_Agent1Name
IMAGE_Agent1NameVocab
IMAGE_Agent1NameRefid
IMAGE_Agent1Role
IMAGE_CulturalContextDisplay
IMAGE_DateDisplay
IMAGE_DescriptionDisplay
IMAGE_InscriptionDisplay
IMGAE_LocationDisplay
IMAGE_MaterialDisplay
IMAGE_MeasurementsDisplay
IMAGE_RelationDisplay
IMAGE_RightsDisplay
IMAGE_RightsNotes
IMAGE_RightsType
IMAGE_RightsHolder
IMAGE_RightsText
IMAGE_SourceDisplay
IMAGE_SourceValue
IMAGE_SourceType
IMAGE_StateEditionDisplay
IMAGE_StylePeriodDisplay
IMAGE_SubjectDisplay
IMAGE_Subject1
IMAGE_SubjectRefid1
IMAGE_SubjectVocab1
IMAGE_SubjectType1
IMAGE_Subject2
IMAGE_SubjectRefid2
IMAGE_SubjectVocab2
IMAGE_SubjectType2
IMAGE_Subject3
IMAGE_SubjectRefid3
IMAGE_SubjectVocab3
IMAGE_SubjectType3
IMAGE_TechniqueDisplay
IMAGE_TextrefDisplay
IMAGE_TitleDisplay
IMAGE_Title
IMAGE_TitleType
IMAGE_WorktypeDisplay

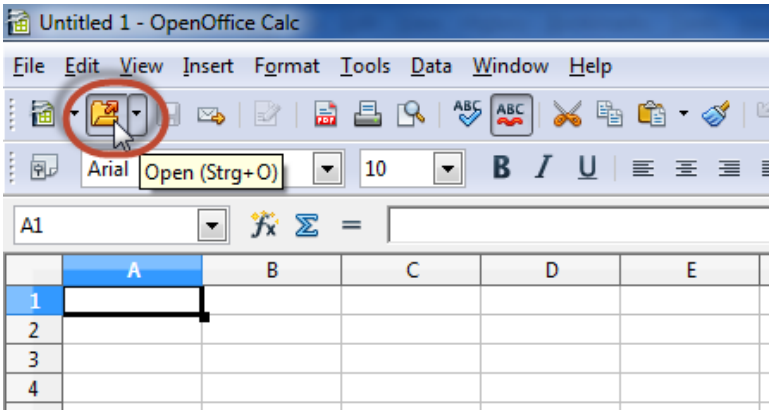
13. Appendix: Converting .xslx file to .csv using OpenOffice Calc

Save your Excel spreadsheet as usual.

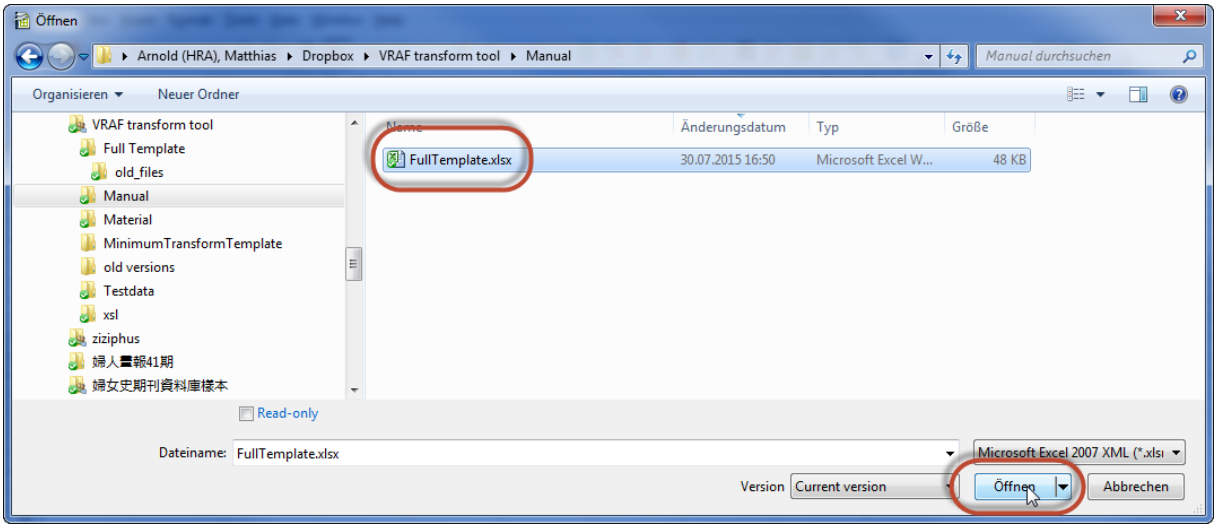
Start OpenOffice Calc

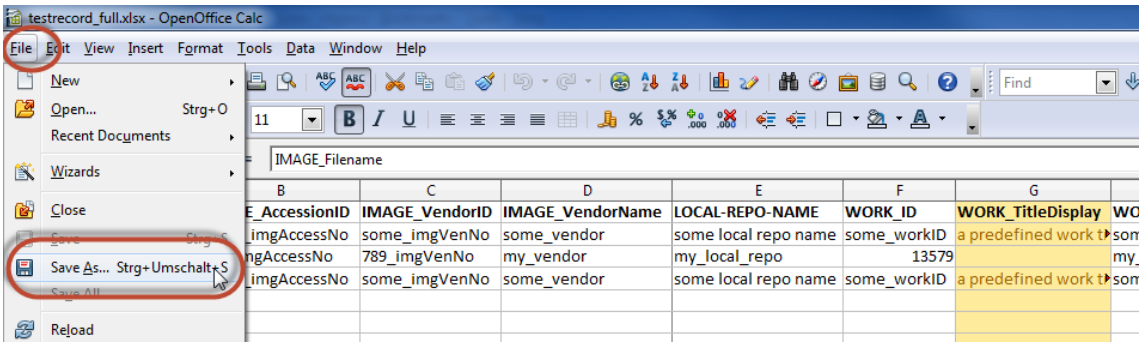


Click "Open"…

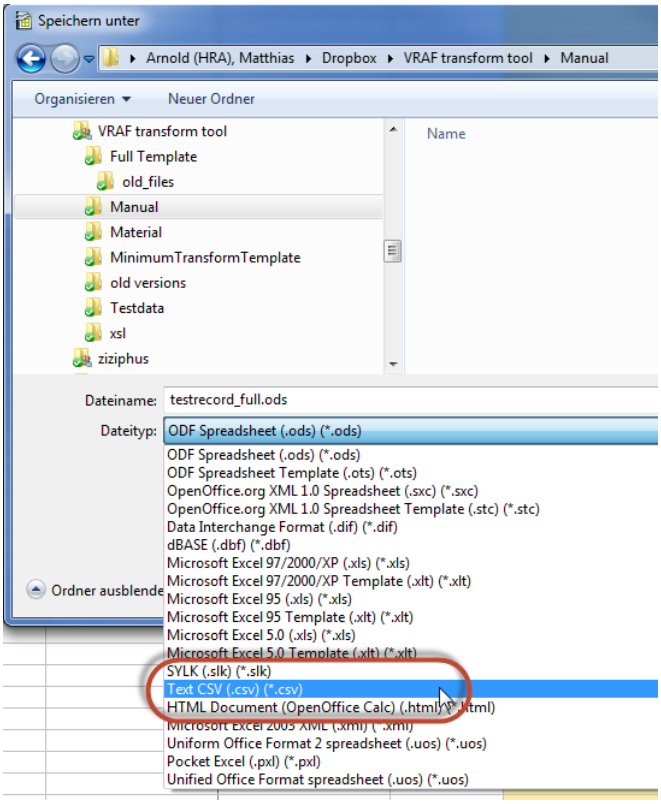


…and select your spreadsheet.



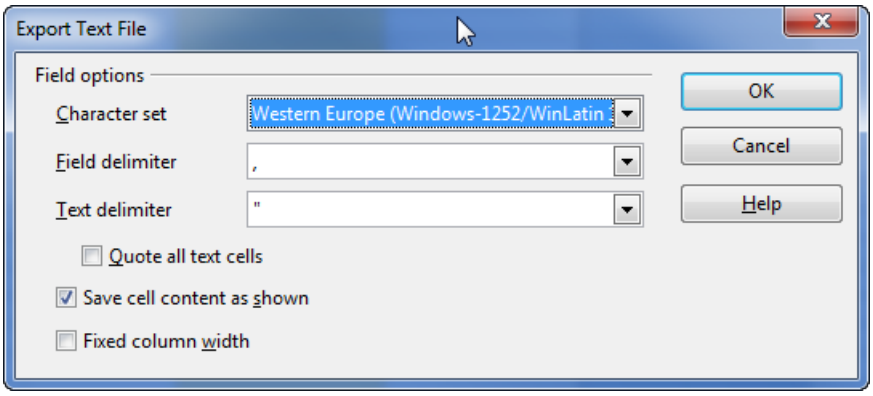


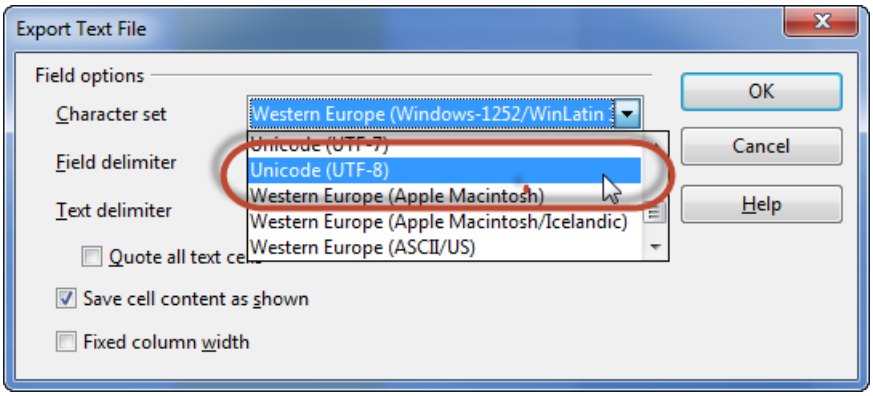
Once the document is opened go to the “File menu” and select “Save As…”



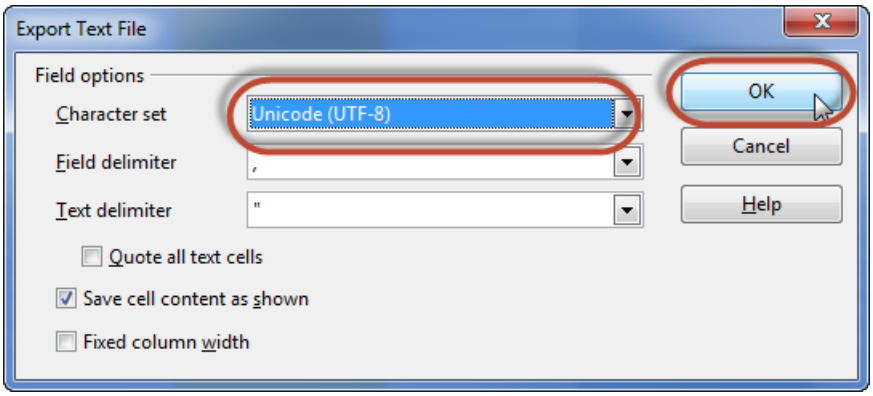
Choose “Text CSV (.csv) (*.csv)” and hit the “Save” button.

This will open the “Export Text File” dialog.

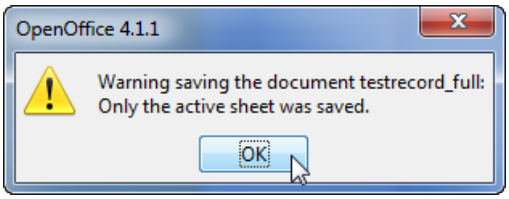




Change the “Character set” to “Unicode (UTF-8)”.



Then click “OK”.



If the workbook contained more than one spreadsheets, a message will be displayed, informing you that only the active sheet was saved to .csv format. Click “OK” to confirm.

You can now close Calc.

To double-check if your data is correctly encoded, open the .csv file in an editor like NotePad++. If it displays correctly, you can proceed to XML transformation.