xtpxlib-xoffice

Conversions for Word and Excel files

Erik Siegel - Xatapult Content Engineering
2024-12-12

0 Table of Contents

0 Xatapult XML Library - Conversions for Word and Excel files

Xtpxlib

xtpxlib library - component **xtpxlib-xoffice** - **v3.0** (2024-12-12)
Xatapult Content Engineering - http://www.xatapult.com - +31 6 53260792
Erik Siegel - erik@xatapult.com

xtpxlib-xoffice is part of the **xtpxlib** library. **xtpxlib** contains software for processing XML, using languages like XSLT and XProc. It consists of several separate components, all named `xtpxlib-*`. Everything can be found on GitHub (https://github.com/xatapult).

This component contains XProc (1.0 and 3.0) pipelines for converting Microsoft Office Word (`.docx`) and Excel (`.xlsx`) files to and from somewhat more manageable XML formats.

Installation and usage information can be found on **xtpxlib**'s main website https://www.xtpxlib.org.

Technical information:

Component documentation: https://xoffice.xtpxlib.org

Git URI: `git@github.com:xatapult/xtpxlib-xoffice.git`

Git site: https://github.com/xatapult/xtpxlib-xoffice

This component depends on:

- xtpxlib-container (Support for XML containers (multiple files wrapped into one))
- xtpxlib-common (Common component: Shared libraries and IDE support)

Release information:

v3.0 - 2024-12-12 (current)
Deprecation of XProc 1.0. Several fixes.

v2.0 - 2023-07-19
Added XProc 3.0 support.

v1.1.B - 2020-02-16
Added the option to insert dates into Excel sheets and a small library for converting dates between Excel and xs:date formats.

v1.1.A - 2020-02-16
New logo and minor fixes.

v1.1 - 2020-02-16
Added basic support for modifying Excel files and fixed some minor bugs.

(Abbreviated. Full release information in `README.md`)

1 Description

Microsoft Office files are actually zip files with a lot of XML and other stuff inside. It is remarkably difficult to get to the actual contents of them: What is in Excel cell A1B2 or what is written in this Word document. To help with this, the `xtpxlib-xoffice` component contains XProc (1.0 and 3.0) pipelines to extract contents from Excel (`.xlsx`) and Word (`.docx`) files.

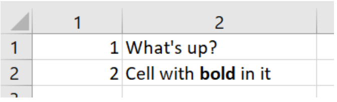
The namespace prefix `xtlxo:` is bound to the namespace `http://www.xtpxlib.nl/ns/xoffice` (`xmlns:xtlxo="http://www.xtpxlib.nl/ns/xoffice"`).

> **NOTE:**
> Especially the `.docx` (Word) conversions should be considered unfinished and experimental. Not everything is converted.

1.1 Converting from Excel (.xlsx)

The `xtlxo:extract-xlsx` pipeline takes an Excel `.xlsx` file and turns this into much more manageable XML. The schema for the resulting XML format is here.

Take for instance this simple Excel sheet:



	1	2
1	1	What's up?
2	2	Cell with **bold** in it

Figure 1-1 - Excel example sheet

Running this through the `xtlxo:extract-xlsx` pipeline returns something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<workbook xmlns="http://www.xtpxlib.nl/ns/xoffice"
          href="file:///path/to/excel.xlsx"
          timestamp="2019-12-11T12:50:20.252+01:00">

   <properties>
      … Sheet properties …
   </properties>

   <worksheet name="Sheet1">
      <row index="1">
         <cell index="1" ref="A1">
            <value>1</value>
         </cell>
         <cell index="2" ref="B1">
            <value>What's up?</value>
         </cell>
      </row>
      <row index="2">
         <cell index="1" ref="A2">
            <value>2</value>
            <formula>A1+1</formula>
         </cell>
         <cell index="2" ref="B2">
            <value>Cell with <span class="b">bold</span> in it</value>
         </cell>
      </row>
   </worksheet>

</workbook>
```

1.2 Converting to Excel (.xlsx)

The `xtlxo:modify-xlsx` pipeline takes a template Excel `.xlsx` file and changes this. The result will be written to a new Excel file.

It has the following features:

- You can change the individual worksheets in the Excel file. A worksheet is identified by its *name* (the name that is visible on its tab at the bottom of the Excel screen).
- You can identify a cell on a worksheet in three ways:
  - As a direct numeric row/column index
  - As identified by an Excel *name*. You can use this to identify a cell, by row, column, or both. An Excel name can reference an area (or even multiple areas) on a worksheet. To work around this the most upper-left cell in the named area(s) is used.
  - Using an Excel name (like above) and adding a numeric offset.
- You can insert a numeric or string value in a cell.
- You have to specify the type of the data to insert (so you can, for instance, insert a numeric value as a string if necessary)

There are some things you need to take care of creating the template Excel file:

- If you need formatting in a cell you're going to fill with this pipeline (like colors, borders, etc.) there *must* be some contents in the cell. Since this will be overwritten, it should not be a problem.
- The same is true for a cell you're referencing by name: It must contain some contents. If you need this contents to be invisible you can always use a single space character.
- Names of worksheets and cells are case-sensitive.

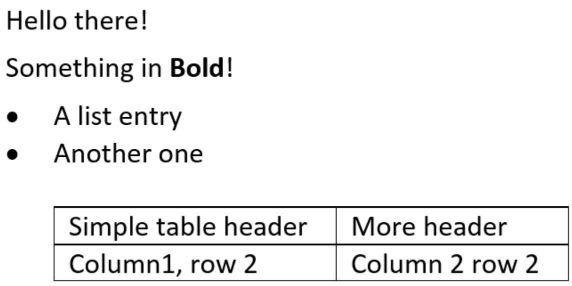
The XML for specifying the changes to the Excel file is quite simple. The schema can be found here. A simple example:

```
<xlsx-modifications xmlns="http://www.xtpxlib.nl/ns/xoffice">

  <worksheet name="TEST">

    <row name="NAMEDCELL" >
      <column name="NAMEDCELL" >
        <number>12345</number>
      </column>
      <column name="NAMEDCELL" offset="1">
        <string>One to the right</string>
      </column>
    </row>

    <row index="1">
      <column index="1">
        <string>Upper left-hand corner</string>
      </column>
      <column index="2">
        <number>6E3</number>
      </column>
    </row>
  </worksheet>

</xlsx-modifications>
```

1.3 Converting from Word (.docx)

The `xtlxo:extract-docx` pipeline takes a Word (`.docx`) file and turns this into an understandable XML format. This format is experimental, there is currently no schema for it.

As an example take this simple Word file:



Hello there!

Something in **Bold**!

- A list entry
- Another one

Simple table header	More header
Column1, row 2	Column 2 row 2

Figure 1-2 - Example Word document

Running this through the `xtlxo:extract-docx` pipeline returns something like:

```
<document xmlns="http://www.xtpxlib.nl/ns/xoffice"
          dref=""
          timestamp="2019-12-11T13:09:15.415+01:00">
   <properties>
      … document properties …
   </properties>

   <p xml:space="preserve">Hello there!</p>
   <p xml:space="preserve">Something in <span class="b">Bold</span>!</p>
   <p class="ListBullet" xml:space="preserve">A list entry</p>
   <p class="ListBullet" xml:space="preserve">Another one</p>
   <p class="ListBullet" indent-left="360" indent-level="0" xml:space="preserve"/>
   <table>
      <tr>
         <td>
            <p class="ListBullet" indent-level="0" xml:space="preserve">Simple table header</p>
         </td>
         <td>
            <p class="ListBullet" indent-level="0" xml:space="preserve">More header</p>
         </td>
      </tr>
      <tr>
         <td>
            <p class="ListBullet" indent-level="0" xml:space="preserve">Column1, row 2</p>
         </td>
         <td>
            <p class="ListBullet" indent-level="0" xml:space="preserve">Column 2 row 2</p>
         </td>
      </tr>
   </table>
   <p class="ListBullet" indent-left="360" indent-level="0" xml:space="preserve"/>

</document>
```

There's an experimental pipeline `xtlxo:create-docx` to create Word documents (using a template Word document for things like styles, margins, etc.). If you feed this the same kind of XML you get from `xtlxo:extract-docx`, the result *should* be a valid, useable Word document with the new text in it. It's currently incomplete (it doesn't do tables for instance). Use at your own risk.

2 XProc 3.0 Support

The xtpxlib-xoffice component contains the following XProc 3.0 pipelines:

Module/Pipeline	Description
create-docx.xpl	Takes as input the same kind of (unspecified) XML as create by `docx-to-xml.xpl` and tries to turn this into a Word file. Unfinished and experimental (for instance: tables are not (yet) supported)!
docx-to-xml.xpl	Extracts the contents of a Word (`.docx`) file in a more useable XML format (unspecified). Somewhat experimental and unfinished!
modify-xlsx.xpl	Takes an input/template Excel (`.xlsx`) and a modification specification and from this creates a new modified Excel file that merges these two sources.
xlsx-to-xml.xpl	Extracts the contents of an Excel (`.xlsx`) file in a more useable XML format.

Table 2-1 - Module overview

2.1 XProc (3.0) pipeline: create-docx.xpl

File: `xpl3/create-docx.xpl`

Type: `xtlxo:create-docx`

Takes as input the same kind of (unspecified) XML as create by `docx-to-xml.xpl` and tries to turn this into a Word file. Unfinished and experimental (for instance: tables are not (yet) supported)!

Port	Type	Primary?	Description
source	in	yes	The XML to convert into `.docx`.
result	out	yes	The output is identical to the input but with `@timestamp`, `@docx-href-in` and `@docx-href-out` added to the root element.

Option	Type	Rq?	Default	Description
docx-href-in	xs:string	yes		URI of the input (template) `.docx` file to process
docx-href-out	xs:string	yes		URI of the output `.docx` file.

2.2 XProc (3.0) pipeline: docx-to-xml.xpl

File: `xpl3/docx-to-xml.xpl`

Type: `xtlxo:docx-to-xml`

Extracts the contents of a Word (`.docx`) file in a more useable XML format (unspecified). Somewhat experimental and unfinished!

Port	Type	Primary?	Description
result	out	yes	The resulting XML document.

Option	Type	Rq?	Default	Description
xlsx-href	xs:string	yes		Document reference of the `.docx` file to process (must have `file://` in front).

2.3 XProc (3.0) pipeline: modify-xlsx.xpl

File: `xpl3/modify-xlsx.xpl`

Type: `xtlxo:modify-xlsx`

Takes an input/template Excel (`.xlsx`) and a modification specification and from this creates a new modified Excel file that merges these two sources.

Port	Type	Primary?	Description
`source`	in	yes	The modification specification.
`result`	out	yes	The output is identical to the input but with `@timestamp`, `@xlsx-href-in` and `@xlsx-href-out` added to the root element.

Option	Type	Rq?	Default	Description
`xlsx-href-in`	`xs:string`	yes		URI of the input (template) `.xlsx` file to process
`xlsx-href-out`	`xs:string`	yes		URI of the output `.xlsx` file.

2.4 XProc (3.0) pipeline: xlsx-to-xml.xpl

File: `xpl3/xlsx-to-xml.xpl`

Type: `xtlxo:xlsx-to-xml`

Extracts the contents of an Excel (`.xlsx`) file in a more useable XML format.

Port	Type	Primary?	Description
`result`	out	yes	The resulting XML document.

Option	Type	Rq?	Default	Description
`xlsx-href`	`xs:string`	yes		Document reference of the `.xlsx` file to process (must have `file://` in front).

3 XProc 1.0 Support

The xtpxlib-xoffice component contains the following XProc 1.0 library modules:

WARNING: XProc 1.0 support is considered deprecated and will be removed in the near future!

Module/Pipeline	Description
`common.mod.xpl`	XProc (1.0) library with generic steps.

Table 3-1 - Module overview

3.1 XProc (1.0) library: common.mod.xpl

File: `xplmod/common.mod/common.mod.xpl`

XProc (1.0) library with generic steps.

Prefix	Namespace URI
`xtlc`	`http://www.xtpxlib.nl/ns/common`

Step	Description
`xtlc:copy-directory`	Copies a full directory structure.
`xtlc:copy-file`	Copies a file, if necessary from inside a zip file.
`xtlc:log`	Writes a message to a log file.
`xtlc:recursive-directory-list`	Returns the contents of a directory, going into sub-directories recursively. When the requested directory does not exist, it returns only a `c:directory` root element with an `error="true"` attribute.
`xtlc:remove-dir`	Removes a full directory When the directory does not exist, everything continues without error.
`xtlc:tee`	Tees the input to a file and passes it unchanged (like the Unix tee command).
`xtlc:zip-directory`	Zips a directory and its sub-directories into a single zip file.

3.1.1 Step: xtlc:copy-directory

Copies a full directory structure.

Port	Type	Primary?	Description
`source`	in	yes	Input, will be passed unchanged.
`result`	out	yes	The input unchanged.

Option	Rq?	Default	Description
`href-source-dir`	yes		Reference to the directory to copy from (must have a leading `file:/` specifier!).
`href-target-dir`	yes		Reference to the directory to copy to (must have a leading `file:/` specifier!). If it does not exist the step will try to create it.

3.1.2 Step: xtlc:copy-file

Copies a file, if necessary from inside a zip file.

Port	Type	Primary?	Description
`source`	in	yes	Input, will be passed unchanged.
`result`	out	yes	The input unchanged.

Option	Rq?	Default	Description
`enable`		`true()`	Whether the copying is done at all.
`href-source`	yes		Reference to the source file to copy (must have a leading `file:/` specifier!).
`href-source-zip`		`''`	Document reference to a zip file (must have a leading `file:/` specifier!). When filled, `$href-source` is assumed to be a path inside this zip.
`href-target`	yes		Reference to the target.

3.1.3 Step: xtlc:log

Writes a message to a log file.

Port	Type	Primary?	Description
`source`	in	yes	Input to the logging, will be passed unchanged to the output
`result`	out	yes	The input unchanged.

Option	Rq?	Default	Description
`enable`		`true()`	Whether the logging will be done at all.
`href-log`	yes		Name of the file to write the log messages to (must have a leading `file:/` specifier!).
`keep-messages`		`100`	The number of messages to keep in the logfile. If le `0`, all messages are kept. Set by default to `100` to prevent overflowing files…
`message`	yes		The actual log message to write.
`status`		`'ok'`	Status of the message. Must be `ok`, `warning`, `error` or `debug`.

3.1.4 Step: xtlc:recursive-directory-list

Returns the contents of a directory, going into sub-directories recursively. When the requested directory does not exist, it returns only a `c:directory` root element with an `error="true"` attribute.

Adapted from Norman Walsh's example code.

Port	Type	Primary?	Description
`result`	out	yes	The resulting directory structure listing in XML format.

Option	Rq?	Default	Description
`depth`		`-1`	The sub-directory depth to go. When le `0`, all sub-directories are processed.
`exclude-filter`			An optional regular expression exclude filter.
`flatten`		`false()`	When `true`, the list will be "flattened": All `c:file` children will be direct children of the root's `c:directory` element. These `c:file` elements get a `@name`, `@href-abs` (absolute filename) and `@href-rel` (relative filename) attribute.
`include-filter`			An optional regular expression include filter.
`path`	yes		The path to get the directory listing from.

3.1.5 Step: xtlc:remove-dir

Removes a full directory When the directory does not exist, everything continues without error.

Port	Type	Primary?	Description
`source`	in	yes	Input, will be passed unchanged.
`result`	out	yes	The input unchanged.

Option	Rq?	Default	Description
`enable`		`true()`	Whether the removal is done at all.
`href-dir`	yes		Reference to the directory to remove (must have a leading `file:/` specifier!).

3.1.6 Step: xtlc:tee

Tees the input to a file and passes it unchanged (like the Unix tee command).

Port	Type	Primary?	Description
`source`	in	yes	Input to the tee.
`result`	out	yes	The input unchanged (unless a `$root-attribute-href` was specified).

Option	Rq?	Default	Description
`enable`		`true()`	Whether to actually do the write. When `false`, nothing happens.
`href`	yes		Name of the file to write to (must have a leading `file:/` specifier!)
`indent`		`true()`	Whether or not to indent the tee-d output.
`root-attribute-href`		`''`	If filled, `$href` is recorded as an attribute with this name on the root element of the original input. Must be a valid attribute name.

3.1.7 Step: xtlc:zip-directory

Zips a directory and its sub-directories into a single zip file.

Port	Type	Primary?	Description
`result`	out	yes	The output of the actual zip step, listing all the files that went in.

Option	Rq?	Default	Description
`base-path`	yes		Directory which contents will be stored in the zip (must have a leading `file:/` specifier!)
`href-target-zip`	yes		Document reference for the zip file to produce (must have a leading `file:/` specifier!)
`include-base`		`true()`	When true, the last part of `$base-path` (e.g. `a/b/c` ==> `c`) is used as the root directory in the zip file.

4 XML Schemas

The xtpxlib-xoffice component contains the following XML Schemas:

Module/Pipeline	Description
xlsx-extract.xsd	Schema for the result of an Excel (.xlsx) data extraction to XML. Format produced by the **[**** Referenced linkend id "excel.mod.xpl-xtlxo_extract-xlsx" not found (phase: inline)]** pipeline.
xlsx-modify.xsd	Schema for the modification spefication of Excel (.xlsx) files. Format used by the **[**** Referenced linkend id "excel.mod.xpl-xtlxo_modify-xlsx" not found (phase: inline)]** pipeline.

Table 4-1 - Module overview

4.1 XML Schema: xlsx-extract.xsd

File: xsd/xlsx-extract.xsd

Target namespace: http://www.xtpxlib.nl/ns/xoffice

Schema for the result of an Excel (.xlsx) data extraction to XML. Format produced by the **[**** Referenced linkend id "excel.mod.xpl-xtlxo_extract-xlsx" not found (phase: inline)]** pipeline.

Element	Description
workbook	Root element of the Excel workbook extraction XML result.

4.2 XML Schema: xlsx-modify.xsd

File: xsd/xlsx-modify.xsd

Target namespace: http://www.xtpxlib.nl/ns/xoffice

Schema for the modification spefication of Excel (.xlsx) files. Format used by the **[**** Referenced linkend id "excel.mod.xpl-xtlxo_modify-xlsx" not found (phase: inline)]** pipeline.

Element	Description
xlsx-modifications	Root element of the Excel modifications specification.

5 XSLT Modules

The xtpxlib-xoffice component contains the following XSLT modules.

Module/Pipeline	Description
excel-conversions.mod.xsl	Excel data specific conversions
xoffice.mod.xsl	Library with support code for the MS Office file handling.

Table 5-1 - Module overview

5.1 XSLT (3.0): excel-conversions.mod.xsl

File: `xslmod/excel-conversions.mod.xsl`

Excel data specific conversions

Prefix	Namespace URI
xtlxo	http://www.xtpxlib.nl/ns/xoffice

Variable	Type	Value	Description
xtlxo:excel-start-date	xs:date	xs:date('1900-01-01')	

Function	Description
xtlxo:excel-date-to-xs-date()	Converts an Excel date integer into an xs:date.
xtlxo:xs-date-to-excel-date()	Converts an xs:date into an Excel date integer.

5.1.1 Function: xtlxo:excel-date-to-xs-date() as xs:date

Converts an Excel date integer into an xs:date.

Parameter	Type	Description
excel-value	xs:integer	The Excel date integer to convert.

5.1.2 Function: xtlxo:xs-date-to-excel-date() as xs:integer

Converts an xs:date into an Excel date integer.

Parameter	Type	Description
date	xs:date	The xs:date to convert.

5.2 XSLT (3.0): xoffice.mod.xsl

File: `xslmod/xoffice.mod.xsl`

Library with support code for the MS Office file handling.

Depends on the following XSLT modules from the xtpxlib-common component:

- general.mod.xsl
- href.mod.xsl

Yet largely undocumented. Use at your own risk.

Prefix	Namespace URI
xtlxo	http://www.xtpxlib.nl/ns/xoffice

Variable	Type	Value	Description
xtlxo:relationship-type-comments	xs:string	'http://schemas.openxmlformats.org/officeDocument/2006/relationships/comments'	
xtlxo:relationship-type-core-properties	xs:string	'http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties'	
xtlxo:relationship-type-custom-properties	xs:string	'http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties'	
xtlxo:relationship-type-extended-properties	xs:string	'http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-properties'	
xtlxo:relationship-type-main-document	xs:string	'http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument'	
xtlxo:relationship-type-shared-strings	xs:string	'http://schemas.openxmlformats.org/officeDocument/2006/relationships/sharedStrings'	

Named template	Description
xtlxo:get-properties	

Function	Description
xtlxo:doc-href()	
xtlxo:get-file-root()	
xtlxo:get-file-root-from-relationship-id()	
xtlxo:get-file-root-from-relationship-type()	
xtlxo:get-file-root-relationship()	
xtlxo:get-href()	
xtlxo:get-rels-href()	

5.2.1 Named template: xtlxo:get-properties

Parameter	Type	Rq?	Default	Description
`extracted-office-xml`	`element(xtlcon:document-container)`			

5.2.2 Function: xtlxo:doc-href() as xs:string

Parameter	Type	Description
`href-parts`	`xs:string+`	

5.2.3 Function: xtlxo:get-file-root() as element()?

Parameter	Type	Description
`extracted-office-xml`	`element(xtlcon:document-container)`	
`href-parts`	`xs:string+`	
`is-mandatory`	`xs:boolean`	

5.2.4 Function: xtlxo:get-file-root-from-relationship-id() as element()?

Parameter	Type	Description
`extracted-office-xml`	`element(xtlcon:document-container)`	
`basefile-href`	`xs:string`	
`relationship-id`	`xs:string`	
`is-mandatory`	`xs:boolean`	

5.2.5 Function: xtlxo:get-file-root-from-relationship-type() as element()?

Parameter	Type	Description
`extracted-office-xml`	`element(xtlcon:document-container)`	
`basefile-href`	`xs:string`	
`relationship-type`	`xs:string`	
`is-mandatory`	`xs:boolean`	

5.2.6 Function: xtlxo:get-file-root-relationship() as element(mso-rels:Relationships)?

Parameter	Type	Description
`extracted-office-xml`	`element(xtlcon:document-container)`	
`basefile-href`	`xs:string`	
`is-mandatory`	`xs:boolean`	

5.2.7 Function: xtlxo:get-href() as xs:string

Parameter	Type	Description
`elm`	`element()`	

5.2.8 Function: xtlxo:get-rels-href() as xs:string

Parameter	Type	Description
`basefile-href`	`xs:string`	