

# **xtpxlib-container**

## **XML Container Handling**



## 0 Table of Contents

<b>0 Xatapult XML Library - XML Container Handling</b>	<b>2</b>
<b>1 Description</b>	<b>3</b>
1.1 Applications	3
1.2 Working with containers	4
<b>2 Container format</b>	<b>5</b>
2.1 XProc 1.0 container format	5
2.1.1 Internal documents	6
2.1.2 External documents	6
2.1.3 Document MIME type values	7
2.2 XProc 3.0 container format	7
2.2.1 Internal documents	8
2.2.2 External documents	9
<b>3 XProc 1.0 Libraries</b>	<b>10</b>
3.1 XProc (1.0) library: container.mod.xpl	10
3.1.1 Step: xtlcon:container-to-disk	10
3.1.2 Step: xtlcon:container-to-zip	10
3.1.3 Step: xtlcon:directory-to-container	11
3.1.4 Step: xtlcon:zip-to-container	11
<b>4 XProc 3.0 Support</b>	<b>12</b>
4.1 XProc (3.0) pipeline: container-to-disk.xpl	12
4.2 XProc (3.0) pipeline: container-to-zip.xpl	12
4.3 XProc (3.0) pipeline: directory-to-container.xpl	12
4.4 XProc (3.0) pipeline: zip-to-container.xpl	13
<b>5 XML Schemas</b>	<b>15</b>
5.1 XML Schema: container-xpl3.xsd	15
5.2 XML Schema: container.xsd	15

## 0 Xatapult XML Library - XML Container Handling



**xtpplib** library - component **xtpplib-container** - **v1.0.B** (2020-04-09)  
Xatapult Content Engineering - <http://www.xatapult.com> - +31 6 53260792  
Erik Siegel - [erik@xatapult.com](mailto:erik@xatapult.com)

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

XML Containers provide support for working with multiple related files by wrapping them into a single one. Binary files are referenced instead of included.

The container structure is standardized. Once contents is in a container it's easy to analyze, change and/or write back. It can also be used to create a whole file structure, in a container, and then write it out to disk or zip file.

The **xtpplib-container** component has XProc (1.0 and 3.0) pipelines for:

- Reading the contents of a zip file or directory structure into a container
- Writing a container out to a zip file or disk

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

### **Technical information:**

Component documentation: <https://container.xtpplib.org>

License: GNU GENERAL PUBLIC LICENSE - Version 3, 29 June 2007

Git URI: [git@github.com:xatapult/xtpplib-container.git](https://github.com/xatapult/xtpplib-container)

Git site: <https://github.com/xatapult/xtpplib-container>

This component depends on:

- **xtpplib-common** (Common component: Shared libraries and IDE support)

# 1 Description

In working with XML or HTML, it is often the case that you need to work with multiple documents *at the same time*. And in most cases that's rather awkward. For instance, an XSLT transformation processes a single document. And yes, of course you can get others by calling the `doc()` function or produce others using `<xsl:result-document>`. But, especially when there are a lot of relations between the documents, this requires careful and sometimes heavy programming. The idea of *XML containers* tries to make this more manageable.

An XML container (as handled/used by this module) is an XML structure that holds other XML documents and references to binary files. Here is a short example:

```
<document-container xmlns="http://www.xtpplib.nl/ns/container"
  timestamp="2019-12-12T12:11:10"
  href-target-path="/my/website/location">

  <document href-target="index.html">
    <html> ... </html>
  </document>

  <document href-target="page1.html">
    <html> ... </html>
  </document>

  <external-document href-source="/my/resources/image.jpg" href-target="resources/image.jpg"/>
</document-container>
```

Example 1-1 - Example of an XML container

This example shows a container, probably generated by some pipeline or XSLT stylesheet, that contains the contents of a simple website. All two pages and some image are there. Running this container through `xtlcon:container-to-disk` will write it to the path indicated in `/*/@href-target-path: /my/website/location`. The documents `index.html` and `page1.html` come from the container, the binary `image.jpg` is copied from the indicated source location. Because everything, every separate file, is in (or referenced in) a single encompassing document, lots of things get easier: creating or checking internal referencing, making classes consistent, etc. An XSLT stylesheet that gets this as its main document has access to *all* information.

## 1.1 Applications

As it turned out, the whole idea of working with multiple documents in an XML container had several applications:

- An important application of the zip format is its use as an overarching storage format for applications. For instance, most office suites do this: a Microsoft Word `.docx` or Excel `.xlsx` file is actually a zip file with many smaller files inside (most of them in XML format). There are many other examples. Trying to interpret such a zip file and get something meaningful out of it can be a nightmarish experience, especially if you want to follow the standard (and not rely on some file naming convention some engineer cooked up and might change). It takes following links through several files to the place the actual interesting information is stored. But if you run such a file through `xtlcon:zip-to-container` you get all files in a single encompassing one, making it much, much easier to follow internal links and find the right information. The `xtpplib-xoffice` component does exactly this: it contains pipelines to get the contents of Word and Excel files in an easier to interpret XML format.
- Going even further with this, it is now much easier to *change* or even *create* such a horribly complex Word, Excel or other kind of office zip file:
  - Read a (template) office document in using `xtlcon:zip-to-container`.
  - Change what you need to change (text, spreadsheet cell values, etc.). Leave the rest, with all this complex linking and other stuff you don't really need to understand, alone.
  - Write it to a resulting zip file using `xtlcon:container-to-zip`.
- A file structure that needs to end up on disk or in a zip file can be created easily using this XML container mechanism.

## 1.2 Working with containers

The container format is described [here](#) Working with XML container documents is done using [XProc 1.0](#) or [XProc 3.0](#) pipelines.

**WARNING:** The container formats and processing features differ between the 1.0 and the 3.0 version! More about this in the [container format description](#).

There are some notable missing features in the current container handling. These are not impossible to implement, the need for them just hasn't arisen yet.

- When writing a zip file you cannot control the compression (different ones, on or off). This means that this mechanism currently can't produce e-books (which require an uncompressed first file).
- You can't work with binary contents *inside* the container, for instance when its base64 encoded.

## 2 Container format

**WARNING:** The container formats and processing features differ between the XProc 1.0 and the 3.0 version!

- [Description of the container format for XProc 1.0](#)
- [Description of the container format for XProc 3.0](#)

### 2.1 XProc 1.0 container format

The schema for the XProc 1.0 container format can be found in `xsd/container.xsd`.

The root of an XProc 1.0 XML container document is an `<xtlcon:document-container>` element. The prefix `xtlcon:` must be bound to the namespace `http://www.xtpplib.nl/ns/container` (`xmlns:xtlcon="http://www.xtpplib.nl/ns/container"`).

```
<xtlcon:document-container timestamp = xs:dateTime
    href-source-zip? = xs:string
    href-target-zip? = xs:string
    href-target-zip-result? = xs:string
    href-target-zip-tmpdir? = xs:string
    href-source-path? = xs:string
    href-target-path? = xs:string
    href-target-result-path? = xs:string
    (any)? >
  ( <xtlcon:document> |
    <xtlcon:external-document> |
    <(any)> )*
</xtlcon:document-container>
```

Attribute	#	Type	Description
timestamp	1	xs:dateTime	The timestamp when this container was initially created/generated.
href-source-zip	?	xs:string	When the container was read from a zip file (using <a href="#">xtlcon:zip-to-container</a> ), this attribute holds the href of this zip file.
href-target-zip	?	xs:string	Holds the name of the zip file for writing the container to (using <a href="#">xtlcon:container-to-zip</a> ).
href-target-zip-result	?	xs:string	After the container is written to a zip file using <a href="#">xtlcon:container-to-zip</a> , this attribute will hold the full canonical filename of the zip file.
href-target-zip-tmpdir	?	xs:string	After the container is written to a zip file using <a href="#">xtlcon:container-to-zip</a> , this attribute will hold the full canonical name of the temporary directory used for this process (probably not very useful).
href-source-path	?	xs:string	When the container was read from a directory (using <a href="#">xtlcon:directory-to-container</a> ), this attribute holds the href of this directory.
href-target-path	?	xs:string	Holds the name of the directory for writing the container to (using <a href="#">xtlcon:container-to-disk</a> ).
href-target-result-path	?	xs:string	After the container is written to a directory file using <a href="#">xtlcon:container-to-disk</a> , this attribute will hold the full canonical name of the directory.
(any)	?		Any other attributes are allowed, so additional information can be added for use during processing.

Child element	#	Description
xtlcon:document	*	A document inside the container structure. See "Internal documents" on page 6.
xtlcon:external-document	*	An external document, referenced from the container structure. See "External documents" on page 6.
(any)	*	Any other elements are allowed, so additional information can be added for use during processing.

### 2.1.1 Internal documents

An *internal* document is a document whose contents is inside the container document. This will in most cases be XML documents, but text is also possible. It must be surrounded by an `<xtlcon:document>` element:

```
<xtlcon:document href-source? = xs:string
  href-source-result? = xs:string
  href-target? = xs:string
  href-target-result? = xs:string
  mime-type? = xs:string
  (any)? >
  <(any)>
</xtlcon:document>
```

Attribute	#	Type	Description
href-source	?	xs:string	href of the source for this document. When this document comes from a zip file, it holds the href of the file <i>in</i> the zip.
href-source-result	?	xs:string	After processing holds the full canonical name of the source file.
href-target	?	xs:string	href of the target for this document.
href-target-result	?	xs:string	After processing holds the full canonical name of the target file.
mime-type	?	xs:string	Some specific values for this attribute trigger special conversions on output. See "Document MIME type values" on page 7.
(any)	?		Any other attributes are allowed, so additional information can be added for use during processing.

Child element	#	Description
(any)	1	Root element + contents of the document.

### 2.1.2 External documents

An *external* document is a document that is only referenced from the container. Usually binary files but anything goes. The referencing is done using an `<xtlcon:external-document>` element:

```
<xtlcon:external-document (attributes-from-internal-document)?
  href-source-zip? = xs:string
  href-source-zip-result? = xs:string
  not-in-global-source-zip? = xs:boolean >
  <(any)>
</xtlcon:external-document>
```

All attributes of an [internal document](#) plus the following:

Attribute	#	Type	Description
(attributes-from-internal-document)	?		See <a href="#">internal documents</a> .
href-source-zip	?	xs:string	Reference to the source zip file for this document. If present overrides <code>*/@href-source-zip</code>
href-source-zip-result	?	xs:string	After processing holds the full canonical name of the source zip file.
not-in-global-source-zip	?	xs:boolean	Default: <code>false</code> When set to <code>true</code> , the global zip file <code>*/@href-source-zip</code> is not used. This is necessary to allow references to external files that, when a global zip file is used, come from elsewhere.



Child element	#	Description
(any)	1	Any other elements are allowed, so additional information can be added for use during processing.

### 2.1.3 Document MIME type values

The `xtlcon:document/@mime-type` attribute (see [Internal documents](#)) can trigger some special treatment of the contents of the element:

**`mime-type="application/pdf"` and the root element is `<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">`**

The document is assumed to be an XSL-FO document and the FOP XSL-FO processor is called to create a PDF document. The pipelines (both `xtlcon:container-to-disk` and `xtlcon:container-to-zip`) allow you to specify an FOP configuration file.

**`mime-type="text/plain"`**

The contents of the `<xtlcon:document>` element will be stringified and stored as text. The schema requires some root element to be present as child of `<xtlcon:document>` but this is ignored:

```
<xtlcon:document mime-type="text/plain" ...>
  <dummy-root>Text to be stored
  another line...</dummy-root>
</xtlcon:document>
```

## 2.2 XProc 3.0 container format

The container format for XProc 3.0 allows documents in XML, HTML, text and JSON format. It can also process JSON as XML.

The schema for the XProc 3.0 container format can be found in `xsd/container-xpl3.xsd`.

The root of an XProc 3.0 XML container document is an `<xtlcon:document-container>` element. The prefix `xtlcon:` must be bound to the namespace `http://www.xtpplib.nl/ns/container` (`xmlns:xtlcon="http://www.xtpplib.nl/ns/container"`).

#### NOTE:

When you get the container back after processing by one of the pipelines, you'll find most href related attributes (attributes that start with `href-`) copied to an attribute with the same name plus an underscore in front. The values of these added attributes will be the fully expanded and canonicalized references.

For instance: When the container has an attribute `href-target-zip="../out.zip"`, after processing you'll find an attribute like `_href-target-zip="file:///ful/path/to/out.zip"` on the result.

```
<xtlcon:document-container timestamp = xs:dateTime
  href-source-path? = xs:string
  href-target-path? = xs:string
  href-source-zip? = xs:string
  href-target-zip? = xs:string
  (any)? >
  ( <xtlcon:document> |
    <xtlcon:external-document> |
    <(any)> )*
</xtlcon:document-container>
```

Attribute	#	Type	Description
timestamp	1	xs:dateTime	The timestamp when this container was initially created/generated.
href-source-path	?	xs:string	When the container was read from a directory (using <a href="#">xtlcon:directory-to-container</a> ), this attribute holds the href of this directory.
href-target-path	?	xs:string	Contains the path for the directory when writing the container to (using <a href="#">xtlcon:container-to-disk</a> ).

Attribute	#	Type	Description
href-source-zip	?	xs:string	When the container was read from a zip file (using <a href="#">xtlcon:zip-to-container</a> ), this attribute holds the href of this zip file.
href-target-zip	?	xs:string	Holds the path for the zip file when zipping the container's contents (using <a href="#">xtlcon:container-to-zip</a> ).
(any)	?		Any other attributes are allowed, so additional processing information can be added.

Child element	#	Description
xtlcon:document	*	A document inside the container structure. See "Internal documents" on page 8.
xtlcon:external-document	*	An external document, referenced from the container structure. See "External documents" on page 9.
(any)	*	Any other elements are allowed (in a different namespace), so additional processing information can be added.

## 2.2.1 Internal documents

An *internal* document is a document whose contents is *inside* the container document. It must be surrounded by an `<xtlcon:document>` element.

```
<xtlcon:document href-source? = xs:string
  href-target? = xs:string
  href-source-zip? = xs:string
  content-type? = xs:string
  serialization? = xs:string
  (any)? >
  <(any)>
</xtlcon:document>
```

Attribute	#	Type	Description
href-source	?	xs:string	href of the source for this document. Will be filled by <a href="#">xtlcon:directory-to-container</a> and the <a href="#">xtlcon:zip-to-container</a> pipelines. When this document comes from disk, it holds the relative filename against the value of the container's <code>/*/@href-source-path</code> . When this document comes from a zip file, it holds the (relative) href of the file <i>in</i> the zip.
href-target	?	xs:string	href of the target location for this document. Used by the <a href="#">xtlcon:container-to-disk</a> and <a href="#">xtlcon:container-to-zip</a> pipelines. When writing to disk, a relative filename is made absolute against the container's <code>/*/@href-target-path</code> . When writing to a zip, it <i>must</i> be a relative filename. This will become the path of the file <i>in</i> the zip file.
href-source-zip	?	xs:string	If present, overrides the value of the container's <code>/*/@href-source-zip</code>
content-type	?	xs:string	Content (MIME) type of the document. Will be filled by <a href="#">xtlcon:directory-to-container</a> and the <a href="#">xtlcon:zip-to-container</a> pipelines. If not present when writing a container, <code>text/xml</code> is assumed for internal documents

Attribute	#	Type	Description
serialization	?	xs:string	Serialization settings for this document, expressed as a JSON map. To allow the usual double quoted attribute values ("..."), all single quotes will be converted to double quotes before JSON parsing is done. For instance: <code>serialization="{ 'indent': true}"</code>
(any)	?		Any other attributes are allowed, so additional processing information can be added.

Child element	#	Description
(any)	1	Document contents (For text and JSON does not need to be XML).

## 2.2.2 External documents

An *external* document is a document that is only referenced from the container. Usually binary files but anything goes. The referencing is done using an `<xtlcon:external-document>` element:

```
<xtlcon:external-document href-source? = xs:string
                           href-target? = xs:string
                           href-source-zip? = xs:string
                           not-in-zip? = xs:boolean
                           (any)? >
  <(any)>*
</xtlcon:external-document>
```

Attribute	#	Type	Description
href-source	?	xs:string	href of the source for this document. Will be filled by <code>xtlcon:directory-to-container</code> and the <code>xtlcon:zip-to-container</code> pipelines. When this document comes from disk, it holds the the relative filename against the value of the container's <code>/*/@href-source-path</code> . When this document comes from a zip file, it holds the (relative) href of the file <i>in</i> the zip.
href-target	?	xs:string	href of the target location for this document. Used by the <code>xtlcon:container-to-disk</code> and <code>xtlcon:container-to-zip</code> pipelines. When writing to disk, a relative filename is made absolute against the container's <code>/*/@href-target-path</code> . When writing to a zip, it <i>must</i> be a relative filename. This will become the path of the file <i>in</i> the zip file.
href-source-zip	?	xs:string	If present, overrides the value of the container's <code>/*/@href-source-zip</code>
not-in-zip	?	xs:boolean	Forces this document to load from disk, even when a zip file reference is specified.
(any)	?		Default: false Any other attributes are allowed, so additional processing information can be added.

Child element	#	Description
(any)	*	Any contents allowed for additional processing purposes.

## 3 XProc 1.0 Libraries

The xtpxlib-container component contains the following XProc (1.0) library module:

Module/Pipeline	Description
<code>container.mod.xpl</code>	XProc library with steps for handling xtpxlib containers.

Table 3-1 - Module overview

### 3.1 XProc (1.0) library: container.mod.xpl

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

XProc library with steps for handling xtpxlib containers.

Prefix	Namespace URI
<code>xtlcon</code>	<code>http://www.xtpxlib.nl/ns/container</code>

Step	Description
<code>xtlcon:container-to-disk</code>	Writes the contents of a container to disk.
<code>xtlcon:container-to-zip</code>	Writes the contents of a container to a zip file.
<code>xtlcon:directory-to-container</code>	Reads a directory into a container. All XML files will be read into the container, all other files will be included/referenced as external contents.
<code>xtlcon:zip-to-container</code>	Reads a zip file into a container. All XML files will be read into the container, all other files will be included/referenced as external contents.

#### 3.1.1 Step: xtlcon:container-to-disk

Writes the contents of a container to disk.

Port	Type	Primary?	Description
<code>source</code>	<code>in</code>	yes	The container to process.
<code>result</code>	<code>out</code>	yes	The input container, but with changes that reflect the writing process.

Option	Rq?	Default	Description
<code>href-fop-config</code>		<code>resolve-uri('.././.././xtpxlib-common/data/fop-default-config.xml', static-base-uri())</code>	Optional reference to an Apache FOP configuration file. Must be absolute! When not present a default file will be used.
<code>href-target</code>		<code>' '</code>	Base path where to write the container. When you specify this it will have precedence over a <code>*/@href-target-path</code> .
<code>indent-xml</code>		<code>false()</code>	Whether to indent the XML we create or not.
<code>remove-target</code>		<code>true()</code>	Whether to attempt to remove the target directory before writing.

#### 3.1.2 Step: xtlcon:container-to-zip

Writes the contents of a container to a zip file.

Port	Type	Primary?	Description
<code>source</code>	<code>in</code>	yes	The container to process.
<code>result</code>	<code>out</code>	yes	The input container, but with all the changes in links, paths, etc.

Option	Rq?	Default	Description
<code>href-fop-config</code>		<code>' '</code>	Optional reference to an Apache FOP configuration file. Must be absolute! When not present a default file will be used.
<code>href-target-zip</code>			Base path where to write the container. When you specify this it will have precedence over <code>*/@href-target-zip</code> .
<code>indent-xml</code>		<code>false()</code>	Whether to indent the XML we create or not.

### 3.1.3 Step: xtlcon:directory-to-container

Reads a directory into a container. All XML files will be read into the container, all other files will be included/referenced as external contents.

Port	Type	Primary?	Description
result	out	yes	The output container.

Option	Rq?	Default	Description
add-document-target-paths		true()	Adds (relative) source paths as the target paths to the individual documents.
href-source-directory	yes		Reference to the directory to read.
href-target-path		' '	Optional target path to record on the container.

### 3.1.4 Step: xtlcon:zip-to-container

Reads a zip file into a container. All XML files will be read into the container, all other files will be included/referenced as external contents.

Port	Type	Primary?	Description
result	out	yes	The output container.

Option	Rq?	Default	Description
add-document-target-paths		true()	Adds source paths as the target paths to the individual documents.
href-source-zip	yes		Reference to the zip file to read.
href-target-path		' '	Optional target path to record on the container.

## 4 XProc 3.0 Support

The xtpxlib-container component contains the following support (pipelines and/or libraries) for XProc 3.0:

Module/Pipeline	Description
<a href="#">container-to-disk.xpl</a>	Writes an xtpxlib container structure to disk. A base path must be provided, either as option <code>\$href-target</code> path or as <code>/*/@href-target</code> .
<a href="#">container-to-zip.xpl</a>	Writes an xtpxlib container structure to a zip file. A name for the output zip can be specified in either option <code>\$href-target-zip</code> or on the container in <code>/*/@href-target-zip</code> .
<a href="#">directory-to-container.xpl</a>	Loads a directory (with optional sub-directories) into an xtpxlib container structure.
<a href="#">zip-to-container.xpl</a>	Loads the contents of a zip file (directory depth can be set) into an xtpxlib container structure.

Table 4-1 - Module overview

### 4.1 XProc (3.0) pipeline: container-to-disk.xpl

File: `xpl3mod/container-to-disk/container-to-disk.xpl`

Type: `xtlcon:container-to-disk`

Writes an xtpxlib container structure to disk. A base path must be provided, either as option `$href-target` path or as `/*/@href-target`.

Port	Type	Primary?	Description
source	in	yes	The container to process.
result	out	yes	The input container structure with additional shadow attributes filled.

Option	Type	Rq?	Default	Description
<code>href-target-path</code>	<code>xs:string?</code>		<code>()</code>	Base path where to write the container. When you specify this it will have precedence over a <code>/*/@href-target-path</code> .
<code>remove-target</code>	<code>xs:boolean</code>		<code>true()</code>	Whether to attempt to remove the target directory before writing.

### 4.2 XProc (3.0) pipeline: container-to-zip.xpl

File: `xpl3mod/container-to-zip/container-to-zip.xpl`

Type: `xtlcon:container-to-zip`

Writes an xtpxlib container structure to a zip file. A name for the output zip can be specified in either option `$href-target-zip` or on the container in `/*/@href-target-zip`.

Port	Type	Primary?	Description
source	in	yes	The container to process.
result	out	yes	The input container structure with additional shadow attributes filled.

Option	Type	Rq?	Default	Description
<code>href-target-zip</code>	<code>xs:string?</code>		<code>()</code>	Name of the zip file to write. When you specify this it will have precedence over a <code>/*/@href-target-zip</code> .

### 4.3 XProc (3.0) pipeline: directory-to-container.xpl

File: `xpl3mod/directory-to-container/directory-to-container.xpl`

Type: `xtlcon:directory-to-container`

Loads a directory (with optional sub-directories) into an xtpxlib container structure.

Port	Type	Primary?	Description
result	out	yes	The resulting container structure.

Option	Type	Rq?	Default	Description
add-document-target-paths	xs:boolean		false()	Copies the relative source path as the target path @target-path for the individual documents.  The idea behind this is that in some cases you want to write almost the same structure back to disk. Recording the relative source path as the target path makes this easier: you don't have to set it explicitly.
depth	xs:integer		-1	The sub-directory depth to go. When lt 0, all sub-directories are processed.
exclude-filter	xs:string*		'\.git/'	Optional regular expression exclude filters. By default, .git directories are excluded.
href-source-directory	xs:string	yes		URI of the directory to read.
href-target-path	xs:string?		()	Optional target path to record on the container.  The idea behind this is that this makes it easier to write the container back to another location on disk, the target path is already there.
include-filter	xs:string*		()	Optional regular expression include filters.
json-as-xml	xs:boolean		false()	When JSON files are loaded (option \$load-json is true): whether to add them to the container as XML or as JSON text. It will set the entry's content type to application/json+xml.
load-html	xs:boolean		false()	Whether to load HTML files.
load-json	xs:boolean		false()	Whether to load JSON files.
load-text	xs:boolean		false()	Whether to load text files.
override-content-types	array(array(xs:string))?		()	Override content types specification (see description of p:directory-list).

## 4.4 XProc (3.0) pipeline: zip-to-container.xpl

File: xpl3mod/zip-to-container/zip-to-container.xpl

Type: xtlcon:zip-to-container

Loads the contents of a zip file (directory depth can be set) into an xtpplib container structure.

Port	Type	Primary?	Description
result	out	yes	The resulting container structure

Option	Type	Rq?	Default	Description
add-document-target-paths	xs:boolean		false()	Copies the relative source path as the target path @target-path for the individual documents.  The idea behind this is that in some cases you want to write almost the same structure back to disk or zip. Recording the relative source path as the target path makes this easier: you don't have to set it explicitly.
depth	xs:integer		-1	The sub-directory depth to go. When lt 0, all sub-directories are processed.
exclude-filter	xs:string*		'\ .git/ '	Optional regular expression exclude filters. By default, .git directories are excluded.
href-source-zip	xs:string	yes		URI of the directory to read.
href-target-path	xs:string?		()	Optional target path to record on the container.  The idea behind this is that this makes it easier to write the container back to another location on disk, the target path is already there.
include-filter	xs:string*			Optional regular expression include filters.
json-as-xml	xs:boolean		false()	When JSON files are loaded (option \$load-json is true): whether to add them to the container as XML or as JSON text. It will set the entry's content type to application/json+xml.
load-html	xs:boolean		false()	Whether to load HTML files.
load-json	xs:boolean		false()	Whether to load JSON files.
load-text	xs:boolean		false()	Whether to load text files.
override-content-types	array(array(xs:string))?		()	Override content types specification (see description of p:archive-manifest).



## 5 XML Schemas

The xtpplib-container component contains the following XML Schemas:

Module/Pipeline	Description
<a href="#">container-xpl3.xsd</a>	Schema for an XML container (XProc 3 based pipelines)
<a href="#">container.xsd</a>	Schema for an XML container.

Table 5-1 - Module overview

### 5.1 XML Schema: container-xpl3.xsd

File: xsd/container-xpl3.xsd

Target namespace: <http://www.xtpplib.nl/ns/container>

Schema for an XML container (XProc 3 based pipelines)

Element	Description
document-container	Root element for a document container.

### 5.2 XML Schema: container.xsd

File: xsd/container.xsd

Target namespace: <http://www.xtpplib.nl/ns/container>

Schema for an XML container.

Element	Description
document-container	Root element for a document container.