

xtpxlib-common

Common code and IDE support

0 Table of Contents

| | |
|--|----|
| 0 Xatapult XML Library - Common code | 4 |
| 1 Description | 5 |
| 1.1 Contents | 5 |
| 1.2 Parameter handling in xtpxlib-common | 5 |
| 2 XSLT Modules | 7 |
| 2.1 XSLT (2.0): compare.mod.xsl | 7 |
| 2.1.1 Named template: xtlc:compare-documents as element(xtlc:message)* | 7 |
| 2.2 XSLT (2.0): date-time.mod.xsl | 7 |
| 2.2.1 Function: xtlc:day-in-year-number() as xs:integer | 8 |
| 2.2.2 Function: xtlc:days-in-month() as xs:integer | 8 |
| 2.2.3 Function: xtlc:format-date-as-text() as xs:string | 9 |
| 2.2.4 Function: xtlc:format-date-as-text-short() as xs:string | 9 |
| 2.2.5 Function: xtlc:is-leap-year() as xs:boolean | 9 |
| 2.2.6 Function: xtlc:month-name() as xs:string | 9 |
| 2.2.7 Function: xtlc:month-name-short() as xs:string | 9 |
| 2.2.8 Function: xtlc:to-date() as xs:date | 9 |
| 2.2.9 Function: xtlc:unix-epoch() as xs:decimal | 9 |
| 2.2.10 Function: xtlc:week-number() as xs:integer | 10 |
| 2.2.11 Function: xtlc:weekday-name() as xs:string | 10 |
| 2.2.12 Function: xtlc:weekday-number() as xs:integer | 10 |
| 2.3 XSLT (2.0): format-output.mod.xsl | 10 |
| 2.3.1 Function: xtlc:duration2str() as xs:string | 10 |
| 2.3.2 Function: xtlc:format-amount() as xs:string | 10 |
| 2.3.3 Function: xtlc:format-double() as xs:string | 10 |
| 2.3.4 Function: xtlc:size2str() as xs:string | 11 |
| 2.4 XSLT (2.0): general.mod.xsl | 11 |
| 2.4.1 Named template: xtlc:raise-error | 12 |
| 2.4.2 Function: xtlc:att2str() as xs:string | 12 |
| 2.4.3 Function: xtlc:capitalize() as xs:string | 12 |
| 2.4.4 Function: xtlc:char-repeat() as xs:string | 12 |
| 2.4.5 Function: xtlc:count-leading-whitespace() as xs:integer | 12 |
| 2.4.6 Function: xtlc:elm2str() as xs:string | 13 |
| 2.4.7 Function: xtlc:item2element() as element()? | 13 |
| 2.4.8 Function: xtlc:items2str() as xs:string | 13 |
| 2.4.9 Function: xtlc:prefix-to-length() as xs:string | 13 |
| 2.4.10 Function: xtlc:q() as xs:string | 13 |
| 2.4.11 Function: xtlc:str2bln() as xs:boolean | 13 |
| 2.4.12 Function: xtlc:str2filename-safe() as xs:string | 13 |
| 2.4.13 Function: xtlc:str2filename-safe() as xs:string | 14 |
| 2.4.14 Function: xtlc:str2id() as xs:string | 14 |
| 2.4.15 Function: xtlc:str2id() as xs:string | 14 |
| 2.4.16 Function: xtlc:str2int() as xs:integer | 14 |
| 2.4.17 Function: xtlc:str2regexp() as xs:string | 14 |
| 2.4.18 Function: xtlc:str2regexp() as xs:string | 14 |
| 2.4.19 Function: xtlc:str2seq() as xs:string* | 14 |
| 2.4.20 Function: xtlc:text2lines() as xs:string* | 15 |
| 2.5 XSLT (2.0): href.mod.xsl | 15 |
| 2.5.1 Function: xtlc:href-add-encoding() as xs:string | 15 |
| 2.5.2 Function: xtlc:href-canonical() as xs:string | 16 |
| 2.5.3 Function: xtlc:href-concat() as xs:string | 16 |
| 2.5.4 Function: xtlc:href-decode-uri() as xs:string | 16 |
| 2.5.5 Function: xtlc:href-ext() as xs:string | 16 |
| 2.5.6 Function: xtlc:href-is-absolute() as xs:boolean | 16 |
| 2.5.7 Function: xtlc:href-name() as xs:string | 16 |
| 2.5.8 Function: xtlc:href-name-noext() as xs:string | 17 |
| 2.5.9 Function: xtlc:href-noext() as xs:string | 17 |

| | | |
|----------|---|-----------|
| 2.5.10 | Function: xtlc:href-path() as xs:string | 17 |
| 2.5.11 | Function: xtlc:href-protocol() as xs:string | 17 |
| 2.5.12 | Function: xtlc:href-protocol() as xs:string | 17 |
| 2.5.13 | Function: xtlc:href-protocol-add() as xs:string | 17 |
| 2.5.14 | Function: xtlc:href-protocol-present() as xs:boolean | 18 |
| 2.5.15 | Function: xtlc:href-protocol-remove() as xs:string | 18 |
| 2.5.16 | Function: xtlc:href-relative() as xs:string | 18 |
| 2.5.17 | Function: xtlc:href-relative-from-path() as xs:string | 18 |
| 2.5.18 | Function: xtlc:href-result-doc() as xs:string | 18 |
| 2.6 | XSLT (3.0): macrodefs.mod.xsl | 19 |
| 2.6.1 | Named template: xtlc:expand-macro-definitions | 20 |
| 2.6.2 | Named template: xtlc:macrodefs-as-comment | 20 |
| 2.6.3 | Function: xtlc:expand-macrodefs() as xs:string | 20 |
| 2.6.4 | Function: xtlc:get-standard-macrodef-map() as map(xs:string, xs:string) | 21 |
| 2.6.5 | Function: xtlc:merge-macrodefs() as map(xs:string, xs:string) | 21 |
| 2.7 | XSLT (2.0): message.mod.xsl | 21 |
| 2.7.1 | Named template: xtlc:msg-create as element(xtlc:message) | 21 |
| 2.8 | XSLT (2.0): mimetypes.mod.xsl | 21 |
| 2.8.1 | Function: xtlc:ext2mimetype() as xs:string | 22 |
| 2.8.2 | Function: xtlc:mimetype2ext() as xs:string | 22 |
| 2.9 | XSLT (3.0): parameters.mod.xsl | 22 |
| 2.9.1 | Function: xtlc:expand-text-against-parameters() as xs:string | 22 |
| 2.9.2 | Function: xtlc:parameters-get() as map(xs:string, xs:string*) | 22 |
| 2.10 | XSLT (3.0): simple-macros.mod.xsl | 23 |
| 2.10.1 | Function: xtlc:expand-simple-macros() as xs:string | 23 |
| 2.10.2 | Function: xtlc:expand-simple-macros() as xs:string | 23 |
| 2.11 | XSLT (2.0): uuid.mod.xsl | 23 |
| 2.11.1 | Function: xtlc:get-uuid() as xs:string | 24 |
| 2.11.2 | Function: xtlc:is-real-uuid() as xs:boolean | 24 |
| 3 | XProc 1.0 Libraries | 25 |
| 3.1 | XProc (1.0) library: common.mod.xpl | 25 |
| 3.1.1 | Step: xtlc:copy-directory | 25 |
| 3.1.2 | Step: xtlc:copy-file | 25 |
| 3.1.3 | Step: xtlc:log | 25 |
| 3.1.4 | Step: xtlc:recursive-directory-list | 26 |
| 3.1.5 | Step: xtlc:remove-dir | 26 |
| 3.1.6 | Step: xtlc:tee | 26 |
| 3.1.7 | Step: xtlc:zip-directory | 27 |
| 4 | XProc 3.0 Support | 28 |
| 4.1 | oXygen XProc 3.0 support | 28 |
| 4.2 | XProc (3.0) pipeline: copy-dir.xpl | 28 |
| 4.3 | XProc (3.0) pipeline: create-clear-directory.xpl | 29 |
| 4.4 | XProc (3.0) pipeline: expand-macro-definitions.xpl | 29 |
| 4.5 | XProc (3.0) pipeline: recursive-directory-list.xpl | 29 |
| 4.6 | XProc (3.0) pipeline: subdir-list.xpl | 30 |
| 4.7 | XProc (3.0) pipeline: validate.xpl | 31 |
| 4.8 | XProc (3.0) pipeline: write-log.xpl | 31 |
| 4.9 | XProc (3.0) pipeline: zip-directory.xpl | 32 |
| 5 | XSLT Stylesheets | 33 |
| 5.1 | XSLT (3.0): expand-macro-definitions.xsl | 33 |
| 5.2 | XSLT (2.0): get-system-properties.xsl | 33 |
| 5.3 | XSLT (2.0): xslmod2xqmod-stub.xsl | 33 |
| 6 | XML Data Files | 34 |
| 6.1 | XML document: dummy.xml | 34 |
| 6.2 | XML document: fop-default-config.xml | 34 |
| 6.3 | XML document: mimetypes-table.xml | 34 |
| 7 | XML Schemas | 35 |

| | |
|--------------------------------------|----|
| 7.1 XML Schema: message.xsd | 35 |
| 7.2 XML Schema: mimetypes.xsd | 35 |
| 7.3 XML Schema: parameters.xsd | 35 |

0 Xatapult XML Library - Common code



xtpplib library - component **xtpplib-common** - v2.0.1 (2023-07-22)

Xatapult Content Engineering - <http://www.xatapult.com> - +31 6 53260792

Erik Siegel - erik@xatapult.com

xtpplib-common 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>).

xtpplib-common is **xtpplib**'s communal component. Most other **xtpplib** components rely on it. It contains:

- XSLT libraries, with functionality for handling parameters, manipulating filenames/URIs, MIME types, etc.
- Parts of the functionality of the XSLT libraries are translated into XQuery.
- XProc (1.0 and 3.0) steps, implementing things like recursive directory lists, creating ZIP files from directories, etc.
- Templates (empty XSLT, XProc, XQuery, etc. files) for use in the oXygen IDE.

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

Technical information:

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

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

Git URI: `git@github.com:xatapult/xtpplib-common.git`

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

Release information:

v2.0.1 - 2023-07-22 (current)

Weekday-number and week-number calculations now also work with Saxon HE.

v2.0 - 2023-07-19

Added XProc 3.0 support.

v1.3.2 - 2022-03-24

Added indent option to `xtlc:tee`

v1.3.1 - 2020-08-18

Some bugfixes for `xtlc:log-write`

v1.3 - 2020-08-18

Added `xtlc:write-log` XProc 3.0 step

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

1 Description

xtpxlib-common is xtpxlib's communal component. Most other components in xtpxlib are dependent on it. If you start using xtpxlib, you'll also use it a lot yourself.

1.1 Contents

xtpxlib-common consists of the following parts (by subdirectory):

| Directory | Contents |
|-----------|--|
| data | XML data files. |
| doc | Sources for the generation of the component's documentation. Internal use only. |
| docs | GitHub pages site for this component. |
| etc | Auxiliary files, mainly for use in the oXygen IDE. |
| template | Template files. These files contain XSLT, XQuery, XProc, etc. files with the main structure and headers filled in. Contain macros for use in the oXygen IDE. To install/use these files in oXygen, open its preferences dialogue (Options > Preferences...) and add the xtpxlib-common/template subdirectory to its Document templates section. |
| xpl | General purpose XProc (1.0) pipelines. . |
| xplmod | General purpose XProc (1.0) modules. |
| xpl3 | General purpose XProc (3.0) pipelines. . |
| xpl3mod | General purpose XProc (3.0) modules. |
| xqmod | General purpose XQuery modules. This is a partial translation of the XSLT module's functionality (especially from href.mod.xsl) into XQuery. |
| xsd | Schemas for some of the document types used in Xatapult XML Library. |
| xsl | Some general purpose XSLT stylesheets. |
| xslmod | General purpose XSLT modules. |

1.2 Parameter handling in xtpxlib-common

Parameters, as referred to here, are name/value pairs meant for customizing software's behavior. Things like prompts, URIs, etc. The xtpxlib-common component's parameters have the following characteristics:

- Parameters in this component are handled by the XSLT module [parameteres.mod.xsl](#). This includes:
 - Reading them from an XML document, either a document on its own or embedded into a bigger XML document. The result will be an XPath map (`xs:string, xs:string*`), which can be inspected and used.
 - Expanding parameter references in strings. Parameter references are constructions like `{parameter-name}` (or `${parameter-name}`, both will yield the same results).
- Parameters are specified within an XML element called `<parameters>`, the namespace does not matter. This element can be the root of a document on its own or embedded in a bigger (XML) document. For instance:

```
<parameters>
  <parameter name="greeting">
    <value>Hello!</value>
  </parameter>
</parameters>
```

There is a [schema](#) available for this.

- A single parameter is specified using a `<parameter name="...">` child element. The value of the name attribute will be normalized (whitespace collapsed to a single space character, leading/trailing whitespace removed) and space characters are replaced with an underscore (`_`). So `name=" a b "` will become `parameter a_b`.
- Values for a parameter are specified using `<value>` child element. A parameter can have multiple values. Parameter references inside values (either written as `{parameter-name}` or `${parameter-name}`) are expanded into their values (for multi-valued parameters only the first value is used).

- It is often useful to specify values for parameters based on different circumstances. For instance based on language (Hello in English or Bonjour in French), or system type (<https://www...> for production, <http://test...> for test). This is implemented as follows:
 - When initially reading the parameters you can specify a filter map (`map(xs:string, xs:string*)`).
 - The `<value>` elements can have any attributes. These attributes are handled as whitespace separated lists of values.
 - The name of such an attribute is held against the entries in filter map. If a filter entry with this name exists, one of the values of the attribute must be present in the filter map.

For instance, assume the parameters look like this:

```
<parameters>
  <parameter name="greeting">
    <value lang="en">Hello!</value>
    <value lang="nl de">Hallo!</value>
    <value lang="fr">Bonjour!</value>
  </parameter>
  <parameter name="number">
    <value>123</value>
  </parameter>
</parameters>
```

- Reading this with an empty (or absent) filter map, or a filter map that does not have a `lang` entry, will result in a `greeting` parameter with multiple values, `Hello!`, `Hallo!` and `Bonjour!`.
- Reading this with a filter map `map{ 'lang': 'en' }` will return the `greeting` parameter with value `Hello!`.
- Reading this with a filter map `map{ 'lang': 'fr' }` will return the `greeting` parameter with value `Bonjour!`.
- Reading this with a filter map `map{ 'lang': ('en', 'de') }` (not particularly useful) will return the `greeting` parameter with values `Hello!` *and* `Hallo!`.
- In all cases the `number` parameter will get value `123` (since it has no filtering attributes on its `<value>` element).

It is possible to combine multiple filter attributes on a `<value>` element.

- Another thing that is often useful in specifying parameters is to *group* them. For this you can put a number of `<parameter>` elements inside a `<group name="...">` element. The name of the group is used as a prefix (with a dot (.) separator) for the parameters in the group. For instance:

```
<parameters>
  <group name="important">
    <parameter name="greeting">
      <value>Hello!</value>
    </parameter>
  </group>
</parameters>
```

This will result in a parameter called `important.greeting`.

2 XSLT Modules

The xtpxlib-common component contains the following XSLT modules. The ones used most frequently are [general.mod.xsl](#) and [href.mod.xsl](#).

| Module/Pipeline | Description |
|---------------------------------------|--|
| compare.mod.xsl | XSL library module with support for comparing XML documents/elements: |
| date-time.mod.xsl | XSLT library module containing functions for working with dates and times. |
| format-output.mod.xsl | XSLT library with functions for formatting output/strings. |
| general.mod.xsl | XSLT library module with general constants and code. |
| href.mod.xsl | XSLT library module with functions for the generic handling of href-s (filenames/paths). |
| macrodefs.mod.xsl | Module for handling macro definitions. |
| message.mod.xsl | Message related templates. |
| mimetypes.mod.xsl | MIME type conversion related functions. |
| parameters.mod.xsl | Takes an XML document with parameters and turns this into a parameter map. |
| simple-macros.mod.xsl | |
| uuid.mod.xsl | UUID related functions. |

Table 2-1 - Module overview

2.1 XSLT (2.0): compare.mod.xsl

File: xslmod/compare.mod.xsl

XSL library module with support for comparing XML documents/elements:

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

| Named template | Description |
|--|---|
| xtlc:compare-documents | Compares two XML documents with each other: |

2.1.1 Named template: [xtlc:compare-documents](#) as `element(xtlc:message)*`

Compares two XML documents with each other:

- Comments and processing instructions are ignored
- Text nodes are normalized before comparison
- Empty text nodes (after normalization) are ignored
- The comparison stops after the first difference is encountered.
- The result is either:
 - An empty set, when no differences found
 - One or more `xtlc:message` elements, `status="error"` when differences were found (you can only get more than one message on attribute differences)

| Parameter | Type | Rq? | Default | Description |
|-----------|-------------------------------|-----|---------|-----------------------------|
| doc1 | <code>document-node ()</code> | yes | | First document to compare. |
| doc2 | <code>document-node ()</code> | yes | | Second document to compare. |

2.2 XSLT (2.0): date-time.mod.xsl

File: xslmod/date-time.mod.xsl

XSLT library module containing functions for working with dates and times.

When language based, it only distinguishes between Dutch and non-Dutch (which now means: English).

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

| Variable | Type | Value | Description |
|---------------------|------------|--|--|
| xtlc:day-names-en | xs:string+ | ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday') | Sequence with the names of the days in English |
| xtlc:day-names-nl | xs:string+ | ('maandag', 'dinsdag', 'woensdag', 'donderdag', 'vrijdag', 'zaterdag', 'zondag') | Sequence with the names of the days in Dutch |
| xtlc:month-names-en | xs:string+ | ('January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December') | Sequence with the names of the months in English |
| xtlc:month-names-nl | xs:string+ | ('januari', 'februari', 'maart', 'april', 'mei', 'juni', 'juli', 'augustus', 'september', 'oktober', 'november', 'december') | Sequence with the names of the months in Dutch |

| Function | Description |
|---|---|
| <code>xtlc:day-in-year-number()</code> | Computes the day number in the year: January 1 is 1, December 31 is 365 (or 366 in leap years). |
| <code>xtlc:days-in-month()</code> | Computes the number of days in a particular month. If values are out of range it returns 0. |
| <code>xtlc:format-date-as-text()</code> | Formats a date as a string with the month name in full. |
| <code>xtlc:format-date-as-text-short()</code> | Formats a date as a string with the month name in short. |
| <code>xtlc:is-leap-year()</code> | Returns true when a given year is a leap year |
| <code>xtlc:month-name()</code> | Returns the name of a month. |
| <code>xtlc:month-name-short()</code> | Returns the name of a month in short (abbreviated to 3 characters). |
| <code>xtlc:to-date()</code> | Creates a date from its components. |
| <code>xtlc:unix-epoch()</code> | Computes the UNIX "epoch" code (number of seconds since 1-1-1970) for a given date/time. |
| <code>xtlc:week-number()</code> | Computes the ISO week number for a given date. |
| <code>xtlc:weekday-name()</code> | Returns the name of a month. |
| <code>xtlc:weekday-number()</code> | The number of the weekday (1=Monday, 7=Sunday). |

2.2.1 Function: `xtlc:day-in-year-number()` as `xs:integer`

Computes the day number in the year: January 1 is 1, December 31 is 365 (or 366 in leap years).

| Parameter | Type | Description |
|-----------|---------|--------------|
| date | xs:date | Date to use. |

2.2.2 Function: `xtlc:days-in-month()` as `xs:integer`

Computes the number of days in a particular month. If values are out of range it returns 0.

| Parameter | Type | Description |
|--------------|------------|--|
| month-number | xs:integer | The month to calculate the number of days for. |
| year | xs:integer | The year this month is in (important because of leap years). |

2.2.3 Function: **xtlc:format-date-as-text()** as xs:string

Formats a date as a string with the month name in full.

| Parameter | Type | Description |
|-----------|-----------|----------------------------------|
| date | xs:date | The date to format. |
| lang | xs:string | The language for the conversion. |

2.2.4 Function: **xtlc:format-date-as-text-short()** as xs:string

Formats a date as a string with the month name in short.

| Parameter | Type | Description |
|-----------|-----------|----------------------------------|
| date | xs:date | The date to format. |
| lang | xs:string | The language for the conversion. |

2.2.5 Function: **xtlc:is-leap-year()** as xs:boolean

Returns true when a given year is a leap year

| Parameter | Type | Description |
|-----------|------------|--------------------|
| year | xs:integer | The year to check. |

2.2.6 Function: **xtlc:month-name()** as xs:string

Returns the name of a month.

| Parameter | Type | Description |
|--------------|------------|--|
| month-number | xs:integer | The month number (1-12). |
| lang | xs:string | The language you want the month name in. |

2.2.7 Function: **xtlc:month-name-short()** as xs:string

Returns the name of a month in short (abbreviated to 3 characters).

| Parameter | Type | Description |
|--------------|------------|--|
| month-number | xs:integer | The month number (1-12). |
| lang | xs:string | The language you want the month name in. |

2.2.8 Function: **xtlc:to-date()** as xs:date

Creates a date from its components.

| Parameter | Type | Description |
|-----------|------------|----------------------|
| day | xs:integer | Day number to use. |
| month | xs:integer | Month number to use. |
| year | xs:integer | Year to use. |

2.2.9 Function: **xtlc:unix-epoch()** as xs:decimal

Computes the UNIX "epoch" code (number of seconds since 1-1-1970) for a given date/time.

| Parameter | Type | Description |
|-----------|-------------|--|
| datetime | xs:dateTime | The date/time to compute the epoch code for. |

2.2.10 Function: xtlc:week-number() as xs:integer

Computes the ISO week number for a given date.

| Parameter | Type | Description |
|-----------|---------|--------------|
| date | xs:date | Date to use. |

2.2.11 Function: xtlc:weekday-name() as xs:string

Returns the name of a month.

| Parameter | Type | Description |
|------------|------------|--|
| day-number | xs:integer | The day number (1-7). |
| lang | xs:string | The language you want the month name in. |

2.2.12 Function: xtlc:weekday-number() as xs:integer

The number of the weekday (1=Monday, 7=Sunday).

| Parameter | Type | Description |
|-----------|---------|--------------|
| date | xs:date | Date to use. |

2.3 XSLT (2.0): format-output.mod.xsl

File: xslmod/format-output.mod.xsl

XSLT library with functions for formatting output/strings.

When language based, we only distinguish between Dutch and non-Dutch (usually English).

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

| Function | Description |
|-----------------------------------|--|
| <code>xtlc:duration2str()</code> | Turns a day/time duration into a more readable string, e.g. 1d3h40m12s |
| <code>xtlc:format-amount()</code> | Formats an amount by adding a € sign and always use double digits. |
| <code>xtlc:format-double()</code> | Formats a double as a string with a given amount of digits. |
| <code>xtlc:size2str()</code> | Turns an integer (e.g. a file size) into a (rounded) number using a Kb/Mb/Gb suffix. |

2.3.1 Function: xtlc:duration2str() as xs:string

Turns a day/time duration into a more readable string, e.g. 1d3h40m12s

| Parameter | Type | Description |
|---------------|--------------------|---|
| duration | xs:dayTimeDuration | The duration to convert. |
| round-seconds | xs:boolean | Whether the seconds part must be rounded. |

2.3.2 Function: xtlc:format-amount() as xs:string

Formats an amount by adding a € sign and always use double digits.

For the Dutch language, . and , are swapped.

| Parameter | Type | Description |
|-----------|-----------|----------------------------------|
| amount | xs:double | The amount to format. |
| lang | xs:string | The language for the conversion. |

2.3.3 Function: xtlc:format-double() as xs:string

Formats a double as a string with a given amount of digits.

For the Dutch language, . and , are swapped.

| Parameter | Type | Description |
|-----------|------------|--|
| dbl | xs:double | Number to convert |
| digits | xs:integer | The number of digits to use. When < 0 this is left open. |
| lang | xs:string | The language for the conversion. |

2.3.4 Function: xtlc:size2str() as xs:string

Turns an integer (e.g. a file size) into a (rounded) number using a Kb/Mb/Gb suffix.

| Parameter | Type | Description |
|-----------|------------|----------------------|
| size | xs:integer | The size to convert. |

2.4 XSLT (2.0): general.mod.xsl

File: xslmod/general.mod.xsl

XSLT library module with general constants and code.

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

| Variable | Type | Value | Description |
|----------------------------|------------|---|--|
| xtlc:default-dt-format | xs:string | '[Y]-[M01]-[D01] [H01]:[m01]:[s01]' | Default date/time format string (yyyy-mm-dd ...). |
| xtlc:default-dt-format-en | xs:string | '[M01]-[D01]-[Y] [H01]:[m01]:[s01]' | Date/time format string (English: mm-dd-yyyy ...). |
| xtlc:default-dt-format-nl | xs:string | '[D01]-[M01]-[Y] [H01]:[m01]:[s01]' | Date/time format string (Dutch: dd-mm-yyyy ...). |
| xtlc:internal-error-prompt | xs:string | 'Internal error: ' | Add this in front of any internal error raised. |
| xtlc:language-en | xs:string | 'en' | Language code for English |
| xtlc:language-nl | xs:string | 'nl' | Language code for Dutch |
| xtlc:namespace-xtlc-common | xs:string | namespace-uri-for-prefix('xtlc', doc('')/*) | Namespace used for xtpxlib-common. |
| xtlc:status-codes | xs:string+ | (\$xtlc:status-info, \$xtlc:status-warning, \$xtlc:status-error, \$xtlc:status-debug) | Sequence with all valid status codes. |
| xtlc:status-debug | xs:string | 'debug' | Generic debug status/severity code. |
| xtlc:status-error | xs:string | 'error' | Generic error status/severity code. |
| xtlc:status-info | xs:string | 'info' | Generic info (a.k.a. OK) status/severity code. |
| xtlc:status-warning | xs:string | 'warning' | Generic warning status/severity code. |

| Named template | Description |
|------------------|---|
| xtlc:raise-error | Stops any processing by raising an error. |

| Function | Description |
|---------------------------------|--|
| xtlc:att2str() | Turns an attribute into a string representation, suitable for display (e.g. name="value"). |
| xtlc:capitalize() | Capitalizes a string (makes the first character uppercase). |
| xtlc:char-repeat() | Returns a string with a single character repeated a given number of times. |
| xtlc:count-leading-whitespace() | Counts the number of whitespace characters at the beginning of a string |
| xtlc:elm2str() | Turns an element into a descriptive string (the element with all its attributes, excluding schema references). |
| xtlc:item2element() | Tries to find the element belonging to a given item. |
| xtlc:items2str() | Creates a string from a sequence of items. |

| Function | Description |
|---------------------------------------|---|
| <code>xtlc:prefix-to-length()</code> | Prefixes a string with a given character so it will get at least a given length. |
| <code>xtlc:q()</code> | Returns the input string quoted ("\$in") |
| <code>xtlc:str2bln()</code> | Safe conversion of a string into a boolean. |
| <code>xtlc:str2filename-safe()</code> | Replaces all characters in a string that are not allowed in filenames with another character. |
| <code>xtlc:str2filename-safe()</code> | Replaces all characters in a string that are not allowed in filenames with an underscore. |
| <code>xtlc:str2id()</code> | Turns a string into a valid identifier, adding a prefix. |
| <code>xtlc:str2id()</code> | Turns a string into a valid identifier. |
| <code>xtlc:str2int()</code> | Safe conversion of a string to an integer. |
| <code>xtlc:str2regexp()</code> | Turns a string into a regular expression that matches the input exactly. Optionally anchors the regular expression so the match will be on this string <i>only</i> (result starts with ^ and ends with \$). |
| <code>xtlc:str2regexp()</code> | Turns a string into a regular expression that matches the input exactly. |
| <code>xtlc:str2seq()</code> | Converts a string with a list of words into a sequence of words. |
| <code>xtlc:text2lines()</code> | Converts text into separate lines. |

2.4.1 Named template: `xtlc:raise-error`

Stops any processing by raising an error.

| Parameter | Type | Rq? | Default | Description |
|-------------------------|------------------------|-----|----------------------------------|---|
| <code>error-name</code> | <code>xs:string</code> | | <code>\$xtlc:status-error</code> | The (optional) name of the error. Must be an NCName. |
| <code>msg-parts</code> | <code>item()+</code> | yes | | Error message to show (in parts, all parts will be concatenated by <code>xtlc:items2str()</code>). |

2.4.2 Function: `xtlc:att2str()` as `xs:string`

Turns an attribute into a string representation, suitable for display (e.g. `name="value"`).

| Parameter | Type | Description |
|------------------|---------------------------|-----------------------|
| <code>att</code> | <code>attribute()?</code> | Attribute to convert. |

2.4.3 Function: `xtlc:capitalize()` as `xs:string`

Capitalizes a string (makes the first character uppercase).

| Parameter | Type | Description |
|-----------------|------------------------|------------------------|
| <code>in</code> | <code>xs:string</code> | The string to work on. |

2.4.4 Function: `xtlc:char-repeat()` as `xs:string`

Returns a string with a single character repeated a given number of times.

| Parameter | Type | Description |
|---------------------|-------------------------|---|
| <code>char</code> | <code>xs:string</code> | The first character of this string is the character to repeat. If empty, an empty string is returned. |
| <code>repeat</code> | <code>xs:integer</code> | The number of repeats. If ≤ 0 , an empty string is returned. |

2.4.5 Function: `xtlc:count-leading-whitespace()` as `xs:integer`

Counts the number of whitespace characters at the beginning of a string

| Parameter | Type | Description |
|-----------|-----------|------------------|
| text | xs:string | Text to work on. |

2.4.6 Function: xtlc:elm2str() as xs:string

Turns an element into a descriptive string (the element with all its attributes, excluding schema references).

| Parameter | Type | Description |
|-----------|-------------|--------------------|
| elm | element() ? | Element to convert |

2.4.7 Function: xtlc:item2element() as element()?

Tries to find the element belonging to a given item.

- When the item is of type `xs:string` or `xs:anyURI`, it is assumed to be a document reference. The root element of this is returned.
- When the item is of type `document-node()`, the root element of this document is returned
- When the item is of type `element()`, this is returned

You can choose whether to produce an error message or `()` when the item cannot be resolved.

| Parameter | Type | Description |
|----------------------|------------|--|
| item | item() | The item to work on |
| error-on-non-resolve | xs:boolean | Whether to generate an error when <code>\$item</code> could not be resolved. Otherwise, the function will return <code>()</code> . |

2.4.8 Function: xtlc:items2str() as xs:string

Creates a string from a sequence of items.

Useful for easy creation of messages consisting of multiple parts and pieces.

| Parameter | Type | Description |
|-----------|----------|------------------------------|
| items | item() * | The message parts to combine |

2.4.9 Function: xtlc:prefix-to-length() as xs:string

Prefixes a string with a given character so it will get at least a given length.

| Parameter | Type | Description |
|-------------|------------|--|
| in | xs:string | String to prefix |
| prefix-char | xs:string | String to prefix with. Only first character is used. If empty, <code>*</code> is used. |
| length | xs:integer | The length to reach. |

2.4.10 Function: xtlc:q() as xs:string

Returns the input string quoted ("`$in`")

| Parameter | Type | Description |
|-----------|------------|--------------------|
| in | xs:string? | String to convert. |

2.4.11 Function: xtlc:str2bln() as xs:boolean

Safe conversion of a string into a boolean.

When `$in` is empty or not convertible into a boolean, `$default` is returned.

| Parameter | Type | Description |
|-----------|------------|---|
| in | xs:string? | String to convert. |
| default | xs:boolean | Default value to return when <code>\$in</code> is empty or cannot be converted. |

2.4.12 Function: xtlc:str2filename-safe() as xs:string

Replaces all characters in a string that are not allowed in filenames with another character.

| Parameter | Type | Description |
|--------------|------------|---|
| in | xs:string? | String to convert |
| replace-char | xs:string? | String to replace invalid characters with. Only first character is used. If empty, _ is used. |

2.4.13 Function: xtlc:str2filename-safe() as xs:string

Replaces all characters in a string that are not allowed in filenames with an underscore.

| Parameter | Type | Description |
|-----------|------------|-------------------|
| in | xs:string? | String to convert |

2.4.14 Function: xtlc:str2id() as xs:string

Turns a string into a valid identifier, adding a prefix.

All characters that are not allowed in an identifier are converted into underscores.

When the result does not start with a letter or underscore, the prefix `id-` is added.

| Parameter | Type | Description |
|-----------|------------|--------------------|
| in | xs:string | String to convert. |
| prefix | xs:string? | Prefix to apply. |

2.4.15 Function: xtlc:str2id() as xs:string

Turns a string into a valid identifier.

All characters that are not allowed in an identifier are converted into underscores.

When the result does not start with a letter or underscore, the prefix `id-` is added.

| Parameter | Type | Description |
|-----------|-----------|--------------------|
| in | xs:string | String to convert. |

2.4.16 Function: xtlc:str2int() as xs:integer

Safe conversion of a string to an integer.

When `$in` is empty or not convertible to an integer, `$default` is returned.

| Parameter | Type | Description |
|-----------|------------|---|
| in | xs:string? | String to convert. |
| default | xs:integer | Default value to return when <code>\$in</code> is empty or cannot be converted. |

2.4.17 Function: xtlc:str2regexp() as xs:string

Turns a string into a regular expression that matches the input exactly. Optionally anchors the regular expression so the match will be on this string *only* (result starts with `^` and ends with `$`).

| Parameter | Type | Description |
|-----------|------------|---|
| in | xs:string? | String to convert |
| anchor | xs:boolean | If true, the resulting string will be anchored (start with <code>^</code> and ends with <code>\$</code>) |

2.4.18 Function: xtlc:str2regexp() as xs:string

Turns a string into a regular expression that matches the input exactly.

| Parameter | Type | Description |
|-----------|------------|-------------------|
| in | xs:string? | String to convert |

2.4.19 Function: xtlc:str2seq() as xs:string*

Converts a string with a list of words into a sequence of words.

| Parameter | Type | Description |
|-----------|------------|--------------------|
| in | xs:string? | String to convert. |

2.4.20 Function: `xtlc:text2lines()` as `xs:string*`

Converts text into separate lines.

Uses the LF as separator; CRs are removed.

| Parameter | Type | Description |
|------------------------------|------------|--|
| text | xs:string? | The text to convert. |
| remove-empty-start-end-lines | xs:boolean | When <code>true</code> any empty (containing whitespace only) lines at the beginning and end are removed. |
| normalize-indents | xs:boolean | When <code>true</code> the indents of the lines are normalized: the indent of the non-whitespace line with the minimum leading whitespace is removed from all other lines. Lines that contain only whitespace will become zero length. |

2.5 XSLT (2.0): `href.mod.xsl`

File: `xslmod/href.mod.xsl`

XSLT library module with functions for the generic handling of href-s (filenames/paths).

| Prefix | Namespace URI |
|--------|--|
| xtlc | <code>http://www.xtpxlib.nl/ns/common</code> |

| Variable | Type | Value | Description |
|--------------------|-----------|--------|--------------------------|
| xtlc:protocol-file | xs:string | 'file' | File protocol specifier. |

| Function | Description |
|---|---|
| <code>xtlc:href-add-encoding()</code> | Percent encodes all "strange" characters (%xx). Any existing percentage encodings will be kept as is. |
| <code>xtlc:href-canonical()</code> | Makes an href canonical (remove any .. and . directory specifiers). |
| <code>xtlc:href-concat()</code> | Performs a safe concatenation of href components: |
| <code>xtlc:href-decode-uri()</code> | Reverse function of <code>encode-for-uri()</code> . Translates percent encodings (%xx) into their actual characters. |
| <code>xtlc:href-ext()</code> | Returns the extension part of an href. |
| <code>xtlc:href-is-absolute()</code> | Returns <code>true</code> if the href is considered absolute. |
| <code>xtlc:href-name()</code> | Returns the (file)name part of an href. |
| <code>xtlc:href-name-noext()</code> | Returns the (file)name part of an href without its extension. |
| <code>xtlc:href-noext()</code> | Returns the complete href path without its extension. |
| <code>xtlc:href-path()</code> | Returns the path part of an href. |
| <code>xtlc:href-protocol()</code> | Returns the protocol part of an href (without the <code>://</code>). |
| <code>xtlc:href-protocol()</code> | Returns the protocol part of an href (without the <code>://</code>) or a default value when none present. |
| <code>xtlc:href-protocol-add()</code> | Adds a protocol specifier (written without the trailing <code>://</code> , e.g. <code>http</code>) to an href. |
| <code>xtlc:href-protocol-present()</code> | Returns <code>true</code> when an href has a protocol specifier (e.g. <code>file://</code> or <code>http://</code>). |
| <code>xtlc:href-protocol-remove()</code> | Removes the protocol part from an href. |
| <code>xtlc:href-relative()</code> | Computes a relative href from one document to another. |
| <code>xtlc:href-relative-from-path()</code> | Computes a relative href from a directory path to a document. |
| <code>xtlc:href-result-doc()</code> | Transforms an href into something <code>xsl:result-document/@href</code> can use. |

2.5.1 Function: `xtlc:href-add-encoding()` as `xs:string`

Percent encodes all "strange" characters (%xx). Any existing percentage encodings will be kept as is.

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.2 Function: xtlc:href-canonical() as xs:string

Makes an href canonical (remove any .. and . directory specifiers).

Examples:

- `href-canonical('a/b/..c') ==> 'a/c'`

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.3 Function: xtlc:href-concat() as xs:string

Performs a safe concatenation of href components:

- Translates all backslashes into slashes
- Makes sure that all components are separated with a single slash
- If somewhere in the list is an absolute path, the concatenation stops.

Examples:

- `xtlc:href-concat(('a', 'b', 'c')) ==> 'a/b/c'`
- `xtlc:href-concat(('a', '/b', 'c')) ==> '/b/c'`

| Parameter | Type | Description |
|----------------------|------------|--|
| href-path-components | xs:string* | The path components to concatenate into a full href. |

2.5.4 Function: xtlc:href-decode-uri() as xs:string

Reverse function of encode-fo-uri(). Translates percent encodings (%xx) into their actual characters.

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.5 Function: xtlc:href-ext() as xs:string

Returns the extension part of an href.

Examples:

- `xtlc:href-ext('a/b/c.xml') ==> 'xml'`
- `xtlc:href-ext('a/b/c') ==> ''`

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.6 Function: xtlc:href-is-absolute() as xs:boolean

Returns true if the href is considered absolute.

An href is considered absolute when it starts with a / or \, contains a protocol specifier (e.g. file://) or starts with a Windows drive letter (e.g. C:).

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.7 Function: xtlc:href-name() as xs:string

Returns the (file)name part of an href.

Examples:

- `xtlc:href-name('a/b/c') ==> 'c'`
- `xtlc:href-name('c') ==> 'c'`

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.8 Function: xtlc:href-name-noext() as xs:string

Returns the (file)name part of an href without its extension.

Examples:

- `xtlc:href-name-noext('a/b/c.xml') ==> 'c'`
- `xtlc:href-name-noext('a/b/c') ==> 'c'`

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.9 Function: xtlc:href-noext() as xs:string

Returns the complete href path without its extension.

Examples:

- `xtlc:href-noext('a/b/c.xml') ==> 'a/b/c'`
- `xtlc:href-noext('a/b/c') ==> 'a/b/c'`

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.10 Function: xtlc:href-path() as xs:string

Returns the path part of an href.

Examples:

- `xtlc:href-path('a/b/c') ==> 'a/b'`
- `xtlc:href-path('c') ==> ''`

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.11 Function: xtlc:href-protocol() as xs:string

Returns the protocol part of an href (without the ://).

Examples:

- `xtlc:href-protocol('http://...') ==> 'http'`

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.12 Function: xtlc:href-protocol() as xs:string

Returns the protocol part of an href (without the ://) or a default value when none present.

Examples:

- `xtlc:href-protocol('http://...', 'file') ==> 'http'`
- `xtlc:href-protocol('/a/b/c', 'file') ==> 'file'`

| Parameter | Type | Description |
|------------------|-----------|--|
| href | xs:string | href to work on. |
| default-protocol | xs:string | Default protocol to return when \$ref contains none. |

2.5.13 Function: xtlc:href-protocol-add() as xs:string

Adds a protocol specifier (written without the trailing ://, e.g. http) to an href.

| Parameter | Type | Description |
|-----------|------------|---|
| href | xs:string | href to work on. |
| protocol | xs:string | The protocol to add, without a leading <code>://</code> part (e.g. just <code>file</code> or <code>http</code>). |
| force | xs:boolean | When true an existing protocol is removed first. When false, a reference with an existing protocol is left unchanged. |

2.5.14 Function: `xtlc:href-protocol-present()` as `xs:boolean`

Returns true when an href has a protocol specifier (e.g. `file://` or `http://`).

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.15 Function: `xtlc:href-protocol-remove()` as `xs:string`

Removes the protocol part from an href.

Examples:

- `xtlc:protocol-remove('file:///a/b/c') ==> '/a/b/c'`

Weird exceptions:

- `xtlc:protocol-remove('file:/a/b/c') ==> '/a/b/c'`
- `xtlc:protocol-remove('file:/C:/a/b/c') ==> 'C:/a/b/c'`

| Parameter | Type | Description |
|-----------|-----------|------------------|
| href | xs:string | href to work on. |

2.5.16 Function: `xtlc:href-relative()` as `xs:string`

Computes a relative href from one document to another.

Examples:

- `href-relative('a/b/c/from.xml', 'a/b/to.xml') ==> '../to.xml'`
- `href-relative('a/b/c/from.xml', 'a/b/d/to.xml') ==> '../d/to.xml'`

| Parameter | Type | Description |
|-----------|-----------|---|
| from-href | xs:string | href (of a document) of the starting point. |
| to-href | xs:string | href (of a document) of the target. |

2.5.17 Function: `xtlc:href-relative-from-path()` as `xs:string`

Computes a relative href from a directory path to a document.

Examples:

- `href-relative-from-path('a/b/c', 'a/b/to.xml') ==> '../to.xml'`
- `href-relative-from-path('a/b/c', 'a/b/d/to.xml') ==> '../d/to.xml'`

| Parameter | Type | Description |
|----------------|-----------|--|
| from-href-path | xs:string | href (of a directory) of the starting point. |
| to-href | xs:string | href (of a document) of the target. |

2.5.18 Function: `xtlc:href-result-doc()` as `xs:string`

Transforms an href into something `xsl:result-document/@href` can use.

`xsl:result-document/@href` needs a `file://` in front and has some strict rules about the formatting. The input to this function *must* be an absolute href!

| Parameter | Type | Description |
|-----------|-----------|------------------------------------|
| href | xs:string | href to work on. Must be absolute! |

2.6 XSLT (3.0): macrodefs.mod.xsl

File: xslmod/macrodefs.mod.xsl

Module for handling macro definitions.

A macro definition is a simple name=value construct. They are passed around in maps (map(xs:string, xs:string)).

The `xtlc:expand-macrodefs()` function expands macro definition references within strings by using `{...}` or `$...`. To prevent the expansion of these constructions, simply double the opening curly brace. All referenced macro definitions must exist, otherwise an error will be raised.

Macro definitions can reference other macro definitions.

Additionally, you can modify the value of a macro definition reference by appending one or more flags. For more information on the available flags, refer to the `$macrodef-flag-*` global variables.

| Prefix | Namespace URI |
|--------|---|
| xtlc | http://www.xtpplib.nl/ns/common |

| Variable | Type | Value | Description |
|---|-----------|--------------------|--|
| xtlc:macrodef-flag-capitalize | xs:string | 'cap' | Macro definition reference flag: capitalize (upper-case first character) |
| xtlc:macrodef-flag-compact | xs:string | 'compact' | Macro definition reference flag: remove all whitespace |
| xtlc:macrodef-flag-filename-safe | xs:string | 'fns' | Macro definition reference flag: make filename safe (replace all characters forbidden in file/directory names with underscores) |
| xtlc:macrodef-flag-filename-safe-extra | xs:string | 'fnsx' | Macro definition reference flag: make filename safe, extended (replace all characters forbidden in file/directory names and all whitespace with underscores) |
| xtlc:macrodef-flag-lower-case | xs:string | 'lc' | Macro definition reference flag: lower-case |
| xtlc:macrodef-flag-normalize | xs:string | 'normalize' | Macro definition reference flag: normalize space |
| xtlc:macrodef-flag-upper-case | xs:string | 'uc' | Macro definition reference flag: upper-case |
| xtlc:macrodef-standard-date | xs:string | 'DATE' | Standard macro definition: date only (YYYY-MM-DD) |
| xtlc:macrodef-standard-date-compact | xs:string | 'DATECOMPACT' | Standard macro definition: date only, compact (YYMMDD) |
| xtlc:macrodef-standard-datetimeiso | xs:string | 'DATETIMEISO' | Standard macro definition: date/time in ISO format |
| xtlc:macrodef-standard-time | xs:string | 'TIME' | Standard macro definition: time only (hh:mm:ss) |
| xtlc:macrodef-standard-time-compact | xs:string | 'TIMECOMPACT' | Standard macro definition: time only, compact (hhmmss) |
| xtlc:macrodef-standard-time-short | xs:string | 'TIMESHORT' | Standard macro definition: time only without seconds (hh:mm) |
| xtlc:macrodef-standard-time-short-compact | xs:string | 'TIMESHORTCOMPACT' | Standard macro definition: time only without seconds, compact (hhmm) |
| xtlc:macrodef-start-character | xs:string | '\$' | Character that starts a macro definition reference. |

| Named template | Description |
|---|--|
| <code>xtlc:expand-macro-definitions</code> | Expands macro definitions in text nodes and/or attribute values. |
| <code>xtlc:macrodefs-as-comment</code> | Outputs a simple comment showing the contents of <code>\$macrodef-map</code> . |
| Function | Description |
| <code>xtlc:expand-macrodefs()</code> | Expands macro definition references in a string against the macro definitions in <code>\$macrodef-map</code> . Checks for circular references. |
| <code>xtlc:get-standard-macrodef-map()</code> | Returns a map with standard macro definitions. See the <code>\$xtlc:macrodef-standard-*</code> global variable definitions. |
| <code>xtlc:merge-macrodefs()</code> | Merges multiple macro definition maps, taking care that newer definitions override existing ones. Will return an empty map if the input is the empty sequence. |

2.6.1 Named template: `xtlc:expand-macro-definitions`

Expands macro definitions in text nodes and/or attribute values.

The template checks for `<:macrodefs>` elements that are the first child of any element. If so, any `<:macrodef>` children are used to define (or override) macro definitions. These elements can be in any namespace.

See also `xsdmod/macrodefs.mod.xsd`.

You can customize its functionality by using the template parameters.

| Parameter | Type | Rq? | Default | Description |
|-------------------------------------|---|-----|----------------------|---|
| <code>add-macrodef-comments</code> | <code>xs:boolean</code> | | <code>false()</code> | Whether to add a macro definition comment (summarizing all macro definitions) when a <code><*:macrodefs></code> element is processed. |
| <code>expand-in-attributes</code> | <code>xs:boolean</code> | | <code>true()</code> | Whether to expand the macro definitions in attributes. |
| <code>expand-in-text</code> | <code>xs:boolean</code> | | <code>true()</code> | Whether to expand the macro definitions in text nodes. |
| <code>in</code> | <code>node()</code> | | <code>.</code> | The node for which to expand the macro definitions. Must be an element or a document node. |
| <code>macrodefs</code> | <code>map(xs:string, xs:string)*</code> | | <code>()</code> | Any initial macro definitions. |
| <code>use-local-macrodefs</code> | <code>xs:boolean</code> | | <code>true()</code> | Check for <code><*:macrodefs></code> element as first child and process accordingly |
| <code>use-standard-macrodefs</code> | <code>xs:boolean</code> | | <code>true()</code> | Whether to use the standard macro definitions. |

2.6.2 Named template: `xtlc:macrodefs-as-comment`

Outputs a simple comment showing the contents of `$macrodef-map`.

| Parameter | Type | Rq? | Default | Description |
|---------------------------|--|-----|---------|---|
| <code>macrodef-map</code> | <code>map(xs:string, xs:string)</code> | yes | | The macro definitions to show in the comment. |

2.6.3 Function: `xtlc:expand-macrodefs()` as `xs:string`

Expands macro definition references in a string against the macro definitions in `$macrodef-map`. Checks for circular references.

| Parameter | Type | Description |
|--------------|------------------------------|-------------|
| text | xs:string | |
| macrodef-map | map(xs:string, xs:string) | |

2.6.4 Function: `xtlc:get-standard-macrodef-map()` as `map(xs:string, xs:string)`

Returns a map with standard macro definitions. See the `$xtlc:macrodef-standard-*` global variable definitions.

2.6.5 Function: `xtlc:merge-macrodefs()` as `map(xs:string, xs:string)`

Merges multiple macro definition maps, taking care that newer definitions override existing ones. Will return an empty map if the input is the empty sequence.

| Parameter | Type | Description |
|-----------|--------------------------------|-------------------------------------|
| macrodefs | map(xs:string, xs:string) * | The macro definition maps to merge. |

2.7 XSLT (2.0): `message.mod.xsl`

File: `xslmod/message.mod.xsl`

Message related templates.

A [message](#) is a standardized piece of XML used for inserting (error, debug, etc.) messages into XML documents.

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

| Named template | Description |
|---------------------------------|---|
| xtlc:msg-create | Generates a standard <code>xtlc:message</code> element. |

2.7.1 Named template: `xtlc:msg-create` as `element(xtlc:message)`

Generates a standard `xtlc:message` element.

| Parameter | Type | Rq? | Default | Description |
|------------------|----------------------------|-----|-----------------|--|
| extra-attributes | <code>attribute() *</code> | | <code>()</code> | Any extra attributes to add to the message. |
| extra-contents | <code>element() *</code> | | <code>()</code> | Any extra elements to add to the message. |
| msg-parts | <code>item() +</code> | yes | | Message to show (parts will be concatenated by <code>xtlc:items2str()</code>). |
| status | xs:string | yes | | The status of the message. Must be one of the <code>\$xtlc:status-*</code> constants as defined in general.mod.xsl . |

2.8 XSLT (2.0): `mimetypes.mod.xsl`

File: `xslmod/mimetypes.mod.xsl`

MIME type conversion related functions.

These conversions work with an [external MIME type/extension table](#).

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

| Function | Description |
|-------------------------------------|---|
| xtlc:ext2mimetype() | Turns an href extension (e.g. <code>xml</code>) into the correct MIME type (<code>'text/xml'</code>). |
| xtlc:mimetype2ext() | Turns a MIME type (e.g. <code>'text/xml'</code>) into a corresponding href extension (<code>'xml'</code>). |

2.8.1 Function: `xtlc:ext2mimetype()` as `xs:string`

Turns an href extension (e.g. `xml`) into the correct MIME type (`'text/xml'`).

When it cannot find an appropriate MIME type it returns the empty string.

| Parameter | Type | Description |
|-----------|------------------------|---------------------------|
| ext | <code>xs:string</code> | The extension to convert. |

2.8.2 Function: `xtlc:mimetype2ext()` as `xs:string`

Turns a MIME type (e.g. `'text/xml'`) into a corresponding href extension (`'xml'`).

When it doesn't recognize the MIME type it returns the empty string.

| Parameter | Type | Description |
|-----------|------------------------|---------------------------|
| mimetype | <code>xs:string</code> | The MIME type to convert. |

2.9 XSLT (3.0): `parameters.mod.xsl`

File: `xslmod/parameters.mod.xsl`

Takes an XML document with parameters and turns this into a parameter map.

More information [here](#).

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

| Variable | Type | Value | Description |
|--|------------------------|-------------------|---|
| <code>xtlc:parameter-group-separator</code> | <code>xs:string</code> | <code>'.'</code> | When a <code><group></code> element is encountered, this character is used as a separator after the group's name. |
| <code>xtlc:parameter-main-trigger-character</code> | <code>xs:string</code> | <code>'\$'</code> | Use this variable for a quick check on whether something might contain a parameter: <code>contains(..., \$xtlc:parameter-main-trigger-character)</code> |

| Function | Description |
|---|---|
| xtlc:expand-text-against-parameters() | Expands parameter references in <code>\$text</code> (either <code>{...}</code> or <code>\${...}</code>) against the parameters in <code>\$parameter-map</code> . If a parameter has multiple values, only the first one is used. |
| xtlc:parameters-get() | Tries to locate a <code><parameters></code> element (in any namespace) underneath <code>\$root-item</code> and processes the child <code><parameter></code> and <code><group></code> elements in here into a parameter map. |

2.9.1 Function: `xtlc:expand-text-against-parameters()` as `xs:string`

Expands parameter references in `$text` (either `{...}` or `${...}`) against the parameters in `$parameter-map`. If a parameter has multiple values, only the first one is used.

| Parameter | Type | Description |
|---------------|---|----------------------------|
| text | <code>xs:string</code> | Text to expand. |
| parameter-map | <code>map(xs:string, xs:string*)</code> | Map with parameter values. |

2.9.2 Function: `xtlc:parameters-get()` as `map(xs:string, xs:string*)`

Tries to locate a `<parameters>` element (in any namespace) underneath `$root-item` and processes the child `<parameter>` and `<group>` elements in here into a parameter map.

The <value> elements are filtered according to the entries in \$filters.

Parameter references in values (either {\$. . .} or \${. . .}). are expanded. If a parameter has multiple values, only the first one is used.

| Parameter | Type | Description |
|-----------|-----------------------------|--|
| root-item | item() | Root item under which the first <parameters> element is processed. Can be an href, a document node or an element. See <code>xtlc:item2element()</code> on how this is processed. |
| filters | map(xs:string, xs:string*)? | Any filters for the parameter's <value> elements. |

2.10 XSLT (3.0): simple-macros.mod.xsl

File: xslmod/simple-macros.mod.xsl

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

| Variable | Type | Value | Description |
|-----------------------------------|-----------|-------|-------------|
| xtlc:simple-macro-start-character | xs:string | '\$' | |

| Function | Description |
|--|--|
| <code>xtlc:expand-simple-macros()</code> | Expands simple macro's in a string with values. All macros to expand must start with <code>\$xtlc:simple-macro-start-character (\$)</code> , for instance: <code>\$DATE</code> . |
| <code>xtlc:expand-simple-macros#3</code> | Expands simple macro's in a string with values. See <code>xtlc:expand-simple-macros#3</code> |

2.10.1 Function: xtlc:expand-simple-macros() as xs:string

Expands simple macro's in a string with values. All macros to expand must start with `$xtlc:simple-macro-start-character ($)`, for instance: `$DATE`.

The substitution values are in a map. The keys must be the macro strings. For instance: `map{'DATE': '2023-04-04', 'TIME': '16:04:35'}`

| Parameter | Type | Description |
|---------------|---------------------------|--|
| in | xs:string | The string to convert. |
| macros-map | map(xs:string, xs:string) | The map with the macro/substitution values. |
| filename-safe | xs:boolean | Whether to make all substitutions "filename safe", replacing all invalid characters for a file/directory name with an underscore. Use this when replacing macros in file/directory name strings. |

2.10.2 Function: xtlc:expand-simple-macros() as xs:string

Expands simple macro's in a string with values. See `xtlc:expand-simple-macros#3`

| Parameter | Type | Description |
|------------|---------------------------|---|
| in | xs:string | The string to convert. |
| macros-map | map(xs:string, xs:string) | The map with the macro/substitution values. |

2.11 XSLT (2.0): uuid.mod.xsl

File: xslmod/uuid.mod.xsl

UUID related functions.

Works only in Saxon PE or EE (not in the free HE), because we are calling an underlying Java function.

| Prefix | Namespace URI |
|----------------------------------|---|
| xtlc | http://www.xtpxlib.nl/ns/common |
| Function | Description |
| <code>xtlc:get-uuid()</code> | Returns a random unique UUID (by calling an underlying Java function) |
| <code>xtlc:is-real-uuid()</code> | Checks whether a string contains a "real" UUID (conforms to the UUID formatting rules). |

2.11.1 Function: `xtlc:get-uuid()` as `xs:string`

Returns a random unique UUID (by calling an underlying Java function)

2.11.2 Function: `xtlc:is-real-uuid()` as `xs:boolean`

Checks whether a string contains a "real" UUID (conforms to the UUID formatting rules).

Example: 5EAE5C68-7394-48d7-A50B-1669E8D3A6C9 (upper/lower-case both admitted)

| Parameter | Type | Description |
|-----------|-------------------------|----------------|
| id | <code>xs:string?</code> | UUID to check. |

3 XProc 1.0 Libraries

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

| 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 |
|-------------------|--|
| <code>xtlc</code> | <code>http://www.xtpxlib.nl/ns/common</code> |

| 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 <code>c:directory</code> root element with an <code>error="true"</code> 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 |
|---------------------|------------------|----------|----------------------------------|
| <code>source</code> | <code>in</code> | yes | Input, will be passed unchanged. |
| <code>result</code> | <code>out</code> | yes | The input unchanged. |

| Option | Rq? | Default | Description |
|------------------------------|-----|---------|--|
| <code>href-source-dir</code> | yes | | Reference to the directory to copy from (must have a leading <code>file:/</code> specifier!). |
| <code>href-target-dir</code> | yes | | Reference to the directory to copy to (must have a leading <code>file:/</code> 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 |
|---------------------|------------------|----------|----------------------------------|
| <code>source</code> | <code>in</code> | yes | Input, will be passed unchanged. |
| <code>result</code> | <code>out</code> | yes | The input unchanged. |

| Option | Rq? | Default | Description |
|------------------------------|-----|----------------------|---|
| <code>enable</code> | | <code>true ()</code> | Whether the copying is done at all. |
| <code>href-source</code> | yes | | Reference to the source file to copy (must have a leading <code>file:/</code> specifier!). |
| <code>href-source-zip</code> | | <code>' '</code> | Document reference to a zip file (must have a leading <code>file:/</code> specifier!). When filled, <code>\$href-source</code> is assumed to be a path inside this zip. |
| <code>href-target</code> | 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 <code>c:file</code> children will be direct children of the root's <code>c:directory</code> element. These <code>c:file</code> elements get a <code>@name</code> , <code>@href-abs</code> (absolute filename) and <code>@href-rel</code> (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 <code>\$root-attribute-href</code> 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, <code>\$href</code> 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 <code>file:/</code> specifier!) |
| href-target-zip | yes | | Document reference for the zip file to produce (must have a leading <code>file:/</code> specifier!) |
| include-base | | true () | When true, the last part of <code>\$base-path</code> (e.g. <code>a/b/c ==> c</code>) is used as the root directory in the zip file. |

4 XProc 3.0 Support

4.1 oXygen XProc 3.0 support

The component contains a framework for oXygen that enables it to validate XProc 3.0 documents. To use this:

- Add the framework to the oXygen configuration:
 - Menu: Options / Preferences...
 - Navigate to: Document Type Association / Locations
 - Add the full path to `xtpxlib-common/frameworks`
 - Navigate on up to: Document Type Association
 - Check that the XProc 3.0 framework is enabled
- Disable the use of the XProc 1.0 support in oXygen. To do this:
 - Menu: Options / Preferences...
 - Navigate to: File types
 - Associate the file types that you use for XProc 3.0 (in my case `.xpl` files) with the plain XML editor

| Module/Pipeline | Description |
|---|--|
| <code>copy-dir.xpl</code> | This step copies a directory and all its contents from one location to the other. |
| <code>create-clear-directory.xpl</code> | This step does two things: |
| <code>expand-macro-definitions.xpl</code> | This is an XProc driver for the <code>xtlc:expand-macro-definitions</code> template in <code>xslmod/macrodets.mod.xsl</code> . |
| <code>recursive-directory-list.xpl</code> | Extension of standard the <code>p:directory</code> list step. Returns the contents of a directory, going into sub-directories recursively. Adds the possibility to "flatten" the list. |
| <code>subdir-list.xpl</code> | Returns an XML document with the sub-directories of a given directory. |
| <code>validate.xpl</code> | This step performs validation using a W3C Schema and/or Schematron. It breaks the processing if something is wrong. |
| <code>write-log.xpl</code> | Writes an entry to a log file. |
| <code>zip-directory.xpl</code> | Zips a directory into a single zip file. |

Table 4-1 - Module overview

4.2 XProc (3.0) pipeline: `copy-dir.xpl`

File: `xpl3mod/copy-dir/copy-dir.xpl`

Type: `xtlc:copy-dir`

This step copies a directory and all its contents from one location to the other.

- If `$clear-target` is true (default), before copying the target directory is cleared/emptied.
- If the source directory is empty, it simply creates an empty target directory.
- It can do include/exclude filtering, like `p:directory-list`

The step itself acts as an identity step.

| Port | Type | Primary? | Description |
|--------|------|----------|-------------|
| source | in | yes | |
| result | out | yes | |

| Option | Type | Rq? | Default | Description |
|-----------------------------|-------------------------|-----|------------------------|--|
| <code>clear-target</code> | <code>xs:boolean</code> | | <code>true ()</code> | Whether to clear the target before copying. |
| <code>depth</code> | <code>xs:integer</code> | | <code>-1</code> | The sub-directory depth to go. When lt 0, all sub-directories are processed. |
| <code>exclude-filter</code> | <code>xs:string*</code> | | <code>'\..git/'</code> | Regular expression(s) for files to be excluded from the copy. By default, git directories are excluded |
| <code>href-source</code> | <code>xs:string</code> | yes | | The full path/URI of the source directory. If the directory does not exist, nothing will happen. |

| Option | Type | Rq? | Default | Description |
|----------------|------------|-----|---------|--|
| href-target | xs:string | yes | | The full path/URI of the target directory. Any non-existing parent directories leading up to this directory will be automatically created. |
| include-filter | xs:string* | | () | Regular expression(s) files to be included in the copy. |

4.3 XProc (3.0) pipeline: create-clear-directory.xpl

File: xpl3mod/create-clear-directory/create-clear-directory.xpl

Type: xtlc:create-clear-directory

This step does two things:

- When \$clear is true, it removes an (optionally) existing directory
- Then it makes sure the directory always exists

It doesn't matter whether the directory exists beforehand.

The step itself acts as an identity step.

| Port | Type | Primary? | Description |
|--------|------|----------|-------------|
| source | in | yes | |
| result | out | yes | |

| Option | Type | Rq? | Default | Description |
|----------|------------|-----|---------|--|
| clear | xs:boolean | | true () | Whether or not to empty an existing directory. |
| href-dir | xs:string | yes | | The full path/URI of the directory to delete. |

4.4 XProc (3.0) pipeline: expand-macro-definitions.xpl

File: xpl3mod/expand-macro-definitions/expand-macro-definitions.xpl

Type: xtlc:expand-macro-definitions

This is an XProc driver for the xtlc:expand-macro-definitions template in xslmod/macrodefs.mod.xsl.

| Port | Type | Primary? | Description |
|--------|------|----------|---|
| source | in | yes | The document to expand the macro definition references in |
| result | out | yes | The resulting document with the macro definitions expanded. |

| Option | Type | Rq? | Default | Description |
|------------------------|----------------------------|-----|----------|--|
| add-macrodef-comments | xs:boolean | | false () | Whether to add a macro definition comment (summarizing all macro definitions) when a <*:macrodefs> element is processed. |
| expand-in-attributes | xs:boolean | | true () | Whether to expand the macro definitions in attributes. |
| expand-in-text | xs:boolean | | true () | Whether to expand the macro definitions in text nodes. |
| macrodefs | map(xs:string, xs:string)* | | () | Any initial macro definitions. |
| use-local-macrodefs | xs:boolean | | true () | Check for <*:macrodefs> element as first child and process accordingly |
| use-standard-macrodefs | xs:boolean | | true () | Whether to use the standard macro definitions. |

4.5 XProc (3.0) pipeline: recursive-directory-list.xpl

File: xpl3mod/recursive-directory-list/recursive-directory-list.xpl

Type: xtlc:recursive-directory-list

Extension of standard the `p:directory` list step. Returns the contents of a directory, going into sub-directories recursively. Adds the possibility to "flatten" the list.

This step will also *not* throw an error when the directory does not exist. Instead it will simply return an empty result (with an `error="true"` attribute).

| Port | Type | Primary? | Description |
|--------|------|----------|---|
| result | out | yes | The resulting directory structure in XML format. See the standard <code>p:directory-list</code> step for a more detailed description. |

| Option | Type | Rq? | Default | Description |
|------------------------|--------------------------|-----|------------|--|
| add-decoded | xs:boolean | | false() | When true and <code>\$flatten</code> is true, attributes <code>@href-rel-decoded</code> and <code>@href-abs-decoded</code> are added in which any percent encoded characters are decoded. |
| depth | xs:integer | | -1 | The sub-directory depth to go. When lt 0, all sub-directories are processed. |
| detailed | xs:boolean | | false() | Whether to add detailed information. |
| exclude-filter | xs:string* | | '\ .git/ ' | Optional regular expression exclude filters. By default, git directories are excluded. |
| flatten | xs:boolean | | false() | When true, the list will be "flattened": All <code>c:file</code> children will be direct children of the root's <code>c:directory</code> element. These <code>c:file</code> elements get a <code>@name</code> , <code>@href-abs</code> (absolute filename) and <code>@href-rel</code> (relative filename) attribute. |
| include-filter | xs:string* | | | Optional regular expression include filters. |
| override-content-types | array(array(xs:string))? | | () | Override content types specification (see description of <code>p:directory-list</code>). |
| path | xs:string | yes | | The path to get the directory listing from. |

4.6 XProc (3.0) pipeline: subdir-list.xpl

File: `xpl3mod/subdir-list/subdir-list.xpl`

Type: `xtlc:subdir-list`

Returns an XML document with the sub-directories of a given directory.

```
<subdir-list href="...">
  <subdir href="..." name="..." />
  ...
</subdir-list>
```

If an error occurs, it will only return the root element with an additional `error="true"` attribute. Will not recurse!

| Port | Type | Primary? | Description |
|--------|------|----------|---|
| result | out | yes | The sub-directory listing (as described above). |

| Option | Type | Rq? | Default | Description |
|--------|-----------|-----|---------|--|
| path | xs:string | yes | | The path to get the sub-directories from. Always use an absolute path! |

4.7 XProc (3.0) pipeline: validate.xpl

File: xpl3mod/validate/validate.xpl

Type: xtlc:validate

This step performs validation using a W3C Schema and/or Schematron. It breaks the processing if something is wrong.

This might seem superfluous (there are already `p:validate-with...` steps), but often these steps *change* the document. This step performs like a real identity step.

| Port | Type | Primary? | Description |
|--------|------|----------|---------------------------------|
| source | in | yes | Document to validate. |
| result | out | yes | The same as the input document. |

| Option | Type | Rq? | Default | Description |
|-----------------|------------|-----|---------|---|
| href-schema | xs:string? | | () | Optional reference to an W3C Schema to validate the document with. If (), no schema validation will be performed. |
| href-schematron | xs:string? | | () | Optional reference to a Schematron Schema to validate the document with. If (), no Schematron validation will be performed. |
| schema-version | xs:string | | '1.0' | The W3C Schema version to use. |

4.8 XProc (3.0) pipeline: write-log.xpl

File: xpl3mod/write-log/write-log.xpl

Type: xtlc:write-log

Writes an entry to a log file.

With regards to documents flowing through, acts like a `p:identity` step.

| Port | Type | Primary? | Description |
|--------|------|----------|---|
| source | in | yes | Documents will be passed unchanged to the <code>result</code> port. |
| result | out | yes | Documents coming from the <code>source</code> port, unchanged. |

| Option | Type | Rq? | Default | Description |
|-----------------------|---------------------------|-----|---------|---|
| additional-attributes | map(xs:QName, xs:string)? | | () | A map with additional attributes to add to the log entry's entry element. |
| additional-elements | element()* | | () | Elements with additional information to add to this log entry. |
| enable | xs:boolean | | true() | Whether the logging will be done at all. |
| enable-debug-messages | xs:boolean | | true() | Whether messages with debug status will be written as well. |
| href-log | xs:string | yes | | URI of the file to write the log entries to. |
| keep-entries | xs:integer | | 0 | The number of entries to keep in the logfile. If ≤ 0 , all messages are kept. |
| log-comments | xs:string* | | () | Any comments to write as file header when creating a new log file. Ignored on an existing log file. |

| Option | Type | Rq? | Default | Description |
|----------|------------|-----|---------|---|
| messages | xs:string+ | yes | | The actual texts/lines of the log entry to write. All will become a separate message element. |
| status | xs:string | | 'info' | Status of the entry. Must be info, warning, error or debug. |

4.9 XProc (3.0) pipeline: zip-directory.xpl

File: xpl3mod/zip-directory/zip-directory.xpl

Type: xtlc:zip-directory

Zips a directory into a single zip file.

| Port | Type | Primary? | Description |
|--------|------|----------|---|
| result | out | yes | The archive manifest of the created zip file. |

| Option | Type | Rq? | Default | Description |
|-----------------|------------|-----|-----------|---|
| base-path | xs:string | yes | | URI of the directory which contents will be stored in the zip. |
| 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-target-zip | xs:string | yes | | URI for the zip file to produce. |
| include-base | xs:boolean | | true() | When true, the last part of \$base-path (for instance a/b/c ==> c) is used as root directory for entries in the zip file. |
| include-filter | xs:string* | | | Optional regular expression include filters. |

5 XSLT Stylesheets

The xtpxlib-common component contains the following XSLT Stylesheets:

| Module/Pipeline | Description |
|--|---|
| expand-macro-definitions.xsl | This is a driver for the <code>xtlc:expand-macro-definitions</code> template in <code>xslmod/macrodefs.mod.xsl</code> . |
| get-system-properties.xsl | Gets all the XSLT available system properties (as returned by <code>system-property()</code>). |
| xslmod2xqmod-stub.xsl | This stylesheet translates an XSLT module (in xtpxlib "style") into a stub for an XQuery Module. After this you still need to hand-edit it to make it all work. |

Table 5-1 - Module overview

5.1 XSLT (3.0): expand-macro-definitions.xsl

File: `xsl/expand-macro-definitions.xsl`

This is a driver for the `xtlc:expand-macro-definitions` template in `xslmod/macrodefs.mod.xsl`.

| Parameter | Type | Rq? | Default | Description |
|-------------------------------------|---|-----|----------------------|---|
| <code>add-macrodef-comments</code> | <code>xs:boolean</code> | | <code>false()</code> | Whether to add a macro definition comment (summarizing all macro definitions) when a <code><*:macrodefs></code> element is processed. |
| <code>expand-in-attributes</code> | <code>xs:boolean</code> | | <code>true()</code> | Whether to expand the macro definitions in attributes. |
| <code>expand-in-text</code> | <code>xs:boolean</code> | | <code>true()</code> | Whether to expand the macro definitions in text nodes. |
| <code>in</code> | <code>node()</code> | | <code>.</code> | The node for which to expand the macro definitions. Must be an element or a document node. |
| <code>macrodefs</code> | <code>map(xs:string, xs:string)*</code> | | <code>()</code> | Any initial macro definitions. |
| <code>use-local-macrodefs</code> | <code>xs:boolean</code> | | <code>true()</code> | Check for <code><*:macrodefs></code> element as first child and process accordingly |
| <code>use-standard-macrodefs</code> | <code>xs:boolean</code> | | <code>true()</code> | Whether to use the standard macro definitions. |

5.2 XSLT (2.0): get-system-properties.xsl

File: `xsl/get-system-properties.xsl`

Gets all the XSLT available system properties (as returned by `system-property()`).

5.3 XSLT (2.0): xslmod2xqmod-stub.xsl

File: `xsl/xslmod2xqmod-stub.xsl`

This stylesheet translates an XSLT module (in xtpxlib "style") into a stub for an XQuery Module. After this you still need to hand-edit it to make it all work.

See as an example [href.mod.xsl](#). Large parts of this module were turned into XQuery by this stylesheet. The result (edited after this initial conversion) is in the `xqmod` directory of this component.

6 XML Data Files

The xtpxlib-common component contains the following XML data files:

| Module/Pipeline | Description |
|--|---|
| dummy.xml | Dummy file to use as input for processes that require an XML input document but the input is ignored. |
| fop-default-config.xml | Default configuration file for the FOP XSL-FO renderer. |
| mimetypes-table.xml | Table used for transforming file extensions into a MIME type and vice versa. |

Table 6-1 - Module overview

6.1 XML document: dummy.xml

File: data/dummy.xml

Root element: <dummy>

Dummy file to use as input for processes that require an XML input document but the input is ignored.

6.2 XML document: fop-default-config.xml

File: data/fop-default-config.xml

Root element: <fop>

Default configuration file for the FOP XSL-FO renderer.

The only thing this configuration file does is set the font handling to "auto-detect" (meaning it will try the use the system fonts).

6.3 XML document: mimetypes-table.xml

File: data/mimetypes-table.xml

Root element: <mimetypes> (namespace: <http://www.xtpxlib.nl/ns/mimetypes>)

Table used for transforming file extensions into a MIME type and vice versa.

Follows the [mimetypes.xsd](#) schema. Used internally by the [mimetypes.mod.xml](#) module, but might also be useful in other situations.

7 XML Schemas

The xtpxlib-common component contains the following XML Schemas:

| Module/Pipeline | Description |
|--------------------------------|---|
| message.xsd | Schema for messages used and created by this component. |
| mimetypes.xsd | Schema for the MIME type association datafile. |
| parameters.xsd | Schema for sets of parameters as used by this library. |

Table 7-1 - Module overview

7.1 XML Schema: message.xsd

File: `xsd/message.xsd`

Target namespace: `http://www.xtpxlib.nl/ns/common`

Schema for messages used and created by this component.

See also [message.mod.xsl](#).

| Element | Description |
|---------|--|
| message | A message generated by this component. |

7.2 XML Schema: mimetypes.xsd

File: `xsd/mimetypes.xsd`

Target namespace: `http://www.xtpxlib.nl/ns/mimetypes`

Schema for the MIME type association datafile.

See also [mimetypes-table.xml](#) and [mimetypes.mod.xsl](#).

| Element | Description |
|-----------|---|
| mimetypes | Root element of the MIME types associaton list. |

7.3 XML Schema: parameters.xsd

File: `xsd/parameters.xsd`

Schema for sets of parameters as used by this library.

Use [parameters.mod.xsl](#) for turning these lists into maps. An explanation of the parameter mechanism can be found [here](#).

Although this is schema for no namespace, parameters can be in *any* namespace (if you use [parameters.mod.xsl](#) for processing them).

| Element | Description |
|------------|---|
| parameters | Root element for a set of parameters (either in a document on its own or embedded). |