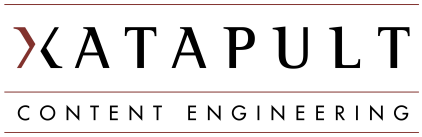
xtpxlib-common

Common code

Erik Siegel - Xatapult Content Engineering
2019-12-03

0 Table of Contents

0 Xatapult XML Library - Common code



xtpxlib library - component **xtpxlib-common** - v0.0 2019-11-28
Xatapult Content Engineering - http://www.xatapult.com - +31 6 53260792
Erik Siegel - erik@xatapult.com

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

`xtpxlib-common` TBD

Installation and usage information can be found on **xtpxlib**'s main website http://www.xatapult.nl.

Technical information:

Component documentation: http://www.xatapult.nl
License: GNU GENERAL PUBLIC LICENSE - Version 3, 29 June 2007
Git URI: `git@github.com:xatapult/xtpxlib-common.git`
Git site: https://github.com/xatapult/xtpxlib-common

1 Description

TBD "XSLT (2.0): general.mod.xsl" on page 8

2 XSLT Modules

2.1 XSLT (2.0): compare.mod.xsl

File: `xslmod/compare.mod.xsl`

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

- Comment and processing instructions are ignored
- Text nodes are normalized before comparison
- Empty text nodes (after normalization) are ignored
- The comparison stops after the first (set of) differences are encountered.
- The result is either:
  - An empty set in 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)

Module dependencies: general.mod.xsl, message.mod.xsl

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

Named template	Description
`xtlc:compare-documents`	
`xtlc:compare-elements`	
`xtlc:compare-node-lists`	

Function	Description
`xtlc:compare-attributes()`	

Named template: `xtlc:compare-documents as element(xtlc:message)*`

Parameter	Type	Rq?	Default	Description
`doc1`	`document-node()`	yes		
`doc2`	`document-node()`	yes		

Named template: `xtlc:compare-elements as element(xtlc:message)*`

Parameter	Type	Rq?	Default	Description
`elm1`	`element()`	yes		
`elm2`	`element()`	yes		
`path`	`xs:string`		`''`	

Named template: `xtlc:compare-node-lists as element(xtlc:message)*`

Parameter	Type	Rq?	Default	Description
`nodes1`	`node()*`	yes		
`nodes2`	`node()*`	yes		
`path`	`xs:string`	yes		

Function: `xtlc:compare-attributes() as element(xtlc:message)*`

Parameter	Type	Description
`path`	`xs:string`	
`attlist1`	`attribute()*`	
`attlist2`	`attribute()*`	

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, we only distinguish between Dutch and non-Dutch (usually English). Some functions will not work using Saxon HE (week-numbers for instance).

Module dependencies: general.mod.xsl

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

Variable	Type	Value	Description
`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
`xtlc:days-in-month()`	Computes the number of days in a particular month. If values are out of range it returns 0.
`xtlc:format-date-as-text()`	Formats a date as a string with month name in full.
`xtlc:format-date-as-text-short()`	Formats a date as a string with month name in short.
`xtlc:is-leap-year()`	Returns true when a given year is a leap year
`xtlc:month-name()`	Returns the name of a month.
`xtlc:month-name-short()`	Returns the name of a month in short (abbreviated to 3 characters).
`xtlc:to-date()`	Creates a date from its components.
`xtlc:unix-epoch()`	Computes the UNIX "epoch" code (number of seconds since 1-1-1970) for a given date/time.
`xtlc:week-number()`	Computes the week number for a given date. Watch out: I'm not completely sure that this uses the system used in The Netherlands...
`xtlc:weekday-name()`	Computes the name of the weekday for a given language. Will be capitalized.
`xtlc:weekday-number()`	The number of the weekday (1=Monday, 7=Sunday).

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).

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

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

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

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

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

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

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.

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 in.

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 in.

Function: `xtlc:to-date() as xs:date`

Creates a date from its components.

Parameter	Type	Description
`day`	`xs:integer`	
`month`	`xs:integer`	
`year`	`xs:integer`	

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.

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

Computes the week number for a given date. Watch out: I'm not completely sure that this uses the system used in The Netherlands...

Parameter	Type	Description
`date`	`xs:date`	Date to use.

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

Computes the name of the weekday for a given language. Will be capitalized.

Parameter	Type	Description
`date`	`xs:date`	Date to use.
`lang`	`xs:string`	The language you want the name in.

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).
Module dependencies: general.mod.xsl

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

Function	Description
`xtlc:duration2str()`	Turns a day/time duration into a more readable string
`xtlc:format-amount()`	Formats an amount by adding a € sign and always use double digits. For the Dutch language . and , are swapped.
`xtlc:format-double()`	Formats a double as a string with a given amount of digits. For the Dutch language . and , are swapped.
`xtlc:size2str()`	Turns an integer (e.g. a filesize) into a (rounded) number using the Kb/Mb/Gb suffix.

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

Turns a day/time duration into a more readable string

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

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. For the Dutch language . and , are swapped.

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. For the Dutch language . and , are swapped.

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

Turns an integer (e.g. a filesize) into a (rounded) number using the 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('')/*)	Name of the xtpxlib common namespace.
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.
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 the 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. Useful for easy creation of messages consisting of multiple parts and pieces.
xtlc:prefix-to-length()	Prefixes a string with a given character so it will get at least a given length.
xtlc:q()	Returns the input string quoted ("$in")
xtlc:str2bln()	Safe conversion of a string into a boolean. When $in is empty or not convertible into a boolean, $default is returned.
xtlc:str2id()	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 extra prefix id- is added.

Function	Description
`xtlc:str2id()`	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 extra prefix `id-` is added.
`xtlc:str2int()`	Safe conversion of a string into an integer. When `$in` is empty or not convertible into an integer, `$default` is returned.
`xtlc:str2seq()`	Converts a string with a list of words into a sequence of words.
`xtlc:text2lines()`	Converts text into separate lines (using the LF as separator, CRs are removed).

Named template: `xtlc:raise-error`

Stops any processing by raising an error.

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

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

Turns an attribute into a string representation, suitable for display.

Parameter	Type	Description
`att`	`attribute()?`	Attribute to convert.

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

Capitalizes a string (makes the first character uppercase).

Parameter	Type	Description
`in`	`xs:string`	The string to work on.

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

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

Parameter	Type	Description
`char`	`xs:string`	The first character of this string is the character to repeat. If empty, an empty string is returned.
`repeat`	`xs:integer`	The number of repeats. If <= 0, an empty string is returned.

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

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

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

Parameter	Type	Description
`elm`	`element()?`	Element to convert

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 `$item` could not be resolved. Otherwise, the function will return `()`.

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

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, a `*` is used.
`length`	`xs:integer`	The length to reach.

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

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

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

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 $in is empty or cannot be converted.

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 extra prefix `id-` is added.

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

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 extra prefix `id-` is added.

Parameter	Type	Description
`in`	`xs:string`	String to convert.

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

Safe conversion of a string into an integer. When `$in` is empty or not convertible into an integer, `$default` is returned.

Parameter	Type	Description
in	xs:string?	String to convert.
default	xs:integer	Default value to return when $in is empty or cannot be converted.

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.

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

Converts text into separate lines (using 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 true any empty (containing whitespace only) lines at the beginning and end are removed.
normalize-indents	xs:boolean	When true 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).
Module dependencies: None

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

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

Function	Description
xtlc:href-add-encoding()	Encodes all "strange" characters with %xx. Any existing %xx parts will be kept as is.
xtlc:href-canonical()	Makes an href canonical (remove any .. and . directory specifiers).
xtlc:href-concat()	Performs a safe concatenation of href components:
xtlc:href-ext()	Returns the extension part of an href.
xtlc:href-is-absolute()	Returns true if the href can be considered absolute.
xtlc:href-name()	Returns the (file)name part of an href.
xtlc:href-name-noext()	Returns the (file)name part of an href, but without its extension.
xtlc:href-noext()	Returns the complete href path but without its extension.
xtlc:href-path()	Returns the path part of an href.
xtlc:href-protocol()	Returns the protocol part of an href (without the ://).
xtlc:href-protocol()	Returns the protocol part of an href (without the ://) or a default value when none present.
xtlc:href-protocol-add()	Adds a protocol part (written without the trailing ://) to an href.
xtlc:href-protocol-present()	Returns true when an href has a protocol specifier (e.g. file:// or http://).
xtlc:href-protocol-remove()	Removes the protocol part from an href.
xtlc:href-relative()	Computes a relative href from one document to another.

Function	Description
`xtlc:href-relative-from-path()`	Computes a relative href from a path to a document.
`xtlc:href-result-doc()`	Transforms an href into something <xsl:result-document> can use.

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

Encodes all "strange" characters with %xx. Any existing %xx parts will be kept as is.

Parameter	Type	Description
`href`	`xs:string`	href to work on.

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.

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 that will be concatenated into a full href.

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.

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

Returns true if the href can be 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.

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.

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

Returns the (file)name part of an href, but 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.

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

Returns the complete href path but 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.

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.

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

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

Parameter	Type	Description
`href`	`xs:string`	href to work on.

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

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

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

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

Adds a protocol part (written without the trailing ://) to an href.

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

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.

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

Removes the protocol part from an href.

Examples (it is tricky and inconsistent!)

- 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.

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.

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

Computes a relative href from a 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.

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

Transforms an href into something <xsl:result-document> can use.

<xsl:result-document> instruction always needs a file:// in front and has some strict rules about the formatting. Make sure the input is an absolute href!

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

2.6 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. Message schema: ../xsd/message.xsd

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

Named template	Description
`xtlc:msg-create`	Generates a standard xtlc:message construct.

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

Generates a standard xtlc:message construct.

Parameter	Type	Rq?	Default	Description
`extra-attributes`	`attribute()*`		`()`	Any extra attributes to add to the message.
`extra-contents`	`element()*`		`()`	Any extra elements to add to the message.
`msg-parts`	`item()+`	yes		Message to show (in parts, all parts will be concatenated).
`status`	`xs:string`	yes		The status of the message. Must be one of the $xtlc:status-* constants.

2.7 XSLT (2.0): mimetypes.mod.xsl

File: `xslmod/mimetypes.mod.xsl`

MIME type conversion related functions.

Module dependencies: None

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

Function	Description
`xtlc:ext2mimetype()`	Turns a dref extension (e.g. 'xml') into the correct MIME type ('text/xml'). When it cannot find the extension, it returns the empty string.
`xtlc:mimetype2ext()`	Turns a MIME type (e.g. 'text/xml') into a corresponding dref extension ('xml'). When it cannot find the MIME type, it returns the empty string.

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

Turns a dref extension (e.g. 'xml') into the correct MIME type ('text/xml'). When it cannot find the extension, it returns the empty string.

This conversion works with an external MIME type/extension table in data/mime-types-table.xml

Parameter	Type	Description
`ext`	`xs:string`	The extension to convert.

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

Turns a MIME type (e.g. 'text/xml') into a corresponding dref extension ('xml'). When it cannot find the MIME type, it returns the empty string.

This conversion works with an external MIME type/extension table in data/mime-types-table.xml

Parameter	Type	Description
`mimetype`	`xs:string`	The MIME type to convert.

2.8 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 in the README.md file.

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

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

Function	Description
xtlc:expand-text-against-parameters()	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.
xtlc:parameters-get()	Tries to locate a <parameters> element underneath $root-item and processes the <parameter> elements in here into a parameters map. The <value> elements are filtered according to the entries in $filters. Parameter references (either {$...} or ${...}). are expanded. If a parameter has multiple values, only the first one is used.

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	xs:string	Text to expand.
parameter-map	map(xs:string, xs:string*)	Map with parameter values.

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

Tries to locate a <parameters> element underneath $root-item and processes the <parameter> elements in here into a parameters map. The <value> elements are filtered according to the entries in $filters. Parameter references (either {$...} 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 a URI, a document node or an element. See xtlc:item2element() on how this is processed.
filters	map(xs:string, xs:string*)?	Any filters for the parameter's <value> elements. See module header for more information.

2.9 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.

Module dependencies: None

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

Function	Description
xtlc:get-uuid()	Returns a random unique UUID (by calling an underlying Java function)
xtlc:is-real-uuid()	Checks whether a string contains a "real" UUID (conforms to the UUID formatting rules).

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

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

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

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

Parameter	Type	Description
`id`	`xs:string?`	UUID to check.

3 XProc Libraries

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

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

XProc library with generic steps.

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

Step	Description
`xtlc:copy-directory`	Copies a full directory structure.
`xtlc:copy-file`	Copies a file, if necessary from inside a zip file.
`xtlc:log`	Writes a message to a log file.
`xtlc:recursive-directory-list`	Returns the contents of a directory, going into sub-directories recursively. When the requested directory does not exist, it returns only a c:directory root element with @error="true".
`xtlc:remove-dir`	Removes a full directory (since the normal processing using pxf:delete does not work properly some older Calabash versions... :-( ) 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.

Step declaration: `xtlc:copy-directory`

Copies a full directory structure.

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

Option	Rq?	Default	Description
`href-source-dir`	yes		Reference to the directory to copy from.
`href-target-dir`	yes		Reference to the directory to copy to.

Step declaration: `xtlc:copy-file`

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

IMPORTANT: For older versions of Calabash (before January 2017) there is a huge bug in this step: A file inside a zip file must have a straight filename without any characters that normally would have been escaped (like, the most inportant one, spaces).

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

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

Step declaration: `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 is done at all.
href-log	yes		Name of the file to write the logmessages 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
status		'ok'	Status of the message. Must be ok, warning, error or debug.

Step declaration: 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 @error="true".

Adapated from Norman Walsh example code at https://github.com/xquery/xquerydoc/blob/master/deps/xmlcalabash/recursive-directory-list.xpl

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 regexp exclude filter.
flatten		false()	When true, the list will be "flattened": Only c:file children within the root c:directory element. All c:file elements have a @name, @href-abs (absolute filename) and @href-rel (relative filename) attribute.
include-filter			An optional regexp include filter.
path	yes		The path to get the directory listing from.

Step declaration: xtlc:remove-dir

Removes a full directory (since the normal processing using pxf:delete does not work properly some older Calabash versions... :-( ) 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.

Step declaration: 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 $root-attribute-href was specified).

Option	Rq?	Default	Description
enable		true()	Whether to actually do the write. When false nothing happens.
href	yes		Name of the file to write to (must have a leading file:// specifier!)
root-attribute-href		''	If filled, $href is recorded as an attribute with this name on the root element of the original input. Must be a valid attribute name.

Step declaration: xtlc:zip-directory

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

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

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