

## **Xalan-J's XSLT 3.0 specification implementation status**

Document modified : 2025-04-01

Document author : Apache Xalan-J team

### **(1) XSL Transformations (XSLT) 3.0 and XML Path Language (XPath) 3.1**

The XSLT 3.0 specification defines the following conformance features, and the level to which Xalan-J implements them.

- |                                   |  |
|-----------------------------------|--|
| a) Basic XSLT processor           | Supported  |
|                                   | XSLT 3.0 instructions and XPath language features, whose implementations are available are described in subsequent sections of this document, below.   |
| b) Schema aware XSLT processor    | Supported  |
|                                   | An XML Schema document can be imported into an XSL stylesheet using <code>xsl:import-schema</code> instruction, and schema's global type definitions and element & attribute declarations can be used within the stylesheet. |
|                                   | Schema aware feature where XML input document, resulting in node tree having detailed type annotations on all possible nodes is not supported. i.e, XPath processor is natively not schema aware.                            |
| c) Serialization feature          | Supported  |
|                                   | A new support for <code>xsl:output method="json"</code> is available, in addition to existing <code>xsl:output method</code> values.   |
| d) Streaming feature              | Not supported  |
| e) Dynamic evaluation feature     | Supported  |
| f) XPath 3.1 feature, for arrays  | Supported  |
| g) Higher-order functions feature | Supported  |

Following are details of XSL 3.0 family of language features, whose working implementation is available on Xalan-J's dev repos branch 'xalan-j\_xslt3.0\_mvn'.

#### **1.1) XSLT 3.0**

**XSLT version 3.0 specification :** <https://www.w3.org/TR/xslt-30/>

- 1) xsl:for-each-group instruction
- 2) xsl:analyze-string instruction
- 3) xsl:iterate instruction
- 4) xsl:for-each instruction implementation improvements, for new XSLT 3.0 requirements.  
Particularly, xsl:for-each instruction being able to iterate XPath atomic values in addition to nodes.
- 5) xsl:evaluate instruction
- 6) xsl:function instruction
- 7) xsl:sequence instruction
- 8) The following XSL stylesheet elements can now have attributes ‘type’ and ‘validation’ : xsl:element, literal result element (xsl:validation and xsl:attribute), xsl:attribute, xsl:copy-of, xsl:copy.
- 9) xsl:attribute element can now have both, "select" attribute and child sequence constructor. But only one of these is allowed to be present on xsl:attribute instruction as specified by XSLT 3.0 specification.
- 10) xsl:import-schema instruction
- 11) xsl:variable instruction evaluation to node set instead of result tree fragment (RTF). This XSLT specification change was first introduced in XSLT 2.0. With XSLT 1.0, if RTF has to be used as node set, then it has to be converted to node set using node-set extension function.
- 12) The sequence type expression "as" attribute on XSLT elements xsl:variable, xsl:template, xsl:function, xsl:param, xsl:with-param, xsl:evaluate.
- 13) XSL template tunnel parameters
- 14) xsl:value-of instruction can now produce result either via its “select” attribute, or by xsl:value-of instruction’s child sequence constructor. xsl:value-of instruction can now have an attribute named ‘separator’ as well.
- 15) xsl:merge instruction
- 16) xsl:fork instruction
- 17) xsl:source-document instruction
- 18) XSLT function implementations
  - a) New function implementations : fn:current-grouping-key, fn:current-group, fn:regex-group, fn:current-merge-group, fn:current-merge-key

b) Function implementation enhancements : fn:system-property

### **Support for following new Xalan-J XSL transformation properties:**

http://apache.org/xalan/validation (used to enable XML input document validation when  
xsl:import-schema instruction is used within an XSL stylesheet,  
with default value false)  
http://apache.org/xalan/xslevaluate (used to enable XSL stylesheet instruction xsl:evaluate, with  
default value false)

These new XSL transformation properties can be set, using Xalan-J's class TransformerImpl when XSL transformation is invoked via API, or via Xalan-J command line.

### **1.2) XPath 3.1**

**XPath version 3.1 specification :** <https://www.w3.org/TR/xpath-31/>

- 1) Range "to" expression
- 2) Value comparison operators eq, ne, lt, le, gt, ge
- 3) Function item "inline function expression"
- 4) Dynamic function calls
- 5) "if" expression
- 6) "for" expression
- 7) Quantified expressions 'some', 'every'
- 8) "let" expression
- 9) Sequence constructor expression, using comma operator
- 10) String concatenation operator "||"
- 11) Node comparison operators "is", "<<", ">>"
- 12) Simple map operator '!'
- 13) Instance Of expression
- 14) Implementation of various new XML Schema built-in data types for use in XSLT 3.0 stylesheets and XPath 3.1 expressions. Implementation of, XPath constructor function calls (for e.g, xs:string('hello'), xs:date('2005-10-07') etc) for these supported XML Schema data types.

Following XML Schema built-in types are supported (depicted with XML Schema data type and subtype hierarchy as specified by W3C XML Schema data types specification) :

- xs:anyType
  - xs:anySimpleType
    - xs:anyAtomicType
      - xs:anyURI
      - xs:boolean
      - xs:decimal
      - xs:integer
        - xs:long
        - xs:int
        - xs:short
        - xs:byte
      - xs:nonNegativeInteger
        - xs:positiveInteger
        - xs:unsignedLong
        - xs:unsignedInt
        - xs:unsignedShort
        - xs:unsignedByte
      - xs:nonPositiveInteger
        - xs:negativeInteger
    - xs:double
    - xs:float
    - xs:QName
    - xs:string
      - xs:normalizedString
      - xs:token
      - xs:Name
      - xs:NCName
    - xs:date
    - xs:dateTime
    - xs:time
    - xs:duration
      - xs:dayTimeDuration
      - xs:yearMonthDuration
    - xs:gDay
    - xs:gMonth
    - xs:gMonthDay
    - xs:gYear
    - xs:gYearMonth

In addition to above mentioned XML Schema built-in data types, an XML Schema type xs:untyped specified by XPath 3.1 specification has also been implemented.

## 15) Collation support

Within the context of XSL languages, a collation is a method by which text information is compared and sorted.

As specified by XPath 3.1 F&O spec, implementations of following collations are available:

15.1) The Unicode Codepoint Collation

15.2) The Unicode Collation Algorithm

Support for following collation uri query parameters is available : 'fallback', 'lang', 'strength'

For the collation's query "lang" parameter, all languages as those supported by Java's 'java.util.Locale' class are available within Xalan-J's XSLT 3.0 implementation (ref, <https://docs.oracle.com/javase/8/docs/api/java/util/Locale.html>).

For the collation's query "strength" parameter, following values are supported : 'primary', 'secondary', 'tertiary', 'identical'.

15.3) The HTML ASCII Case-Insensitive Collation

16) Sequence type expression

17) Map expression

18) Array expression

19) Cast expression

20) Castable expression

21) Treat expression

22) Named function reference

23) Map and array lookup using function call syntax,  
Map and array lookup using unary lookup operator "?"

24) Arrow operator (=>)

25) Node combination operators union, intersect and except

### **1.3) XPath 3.1 functions**

**XPath version 3.1 F&O specification :** <https://www.w3.org/TR/xpath-functions-31/>

Implementation of XPath built-in default functions namespace : <http://www.w3.org/2005/xpath-functions>

Implementation of XPath built-in math functions namespace : <http://www.w3.org/2005/xpath-functions/math>

Implementation of XPath built-in map functions namespace : <http://www.w3.org/2005/xpath-functions/map>

Implementation of XPath built-in array functions namespace : <http://www.w3.org/2005/xpath-functions/array>

#### 1) Functions on numeric values

fn:abs

fn:round (implementation of an optional second argument, that's used to specify 'precision')

#### 2) Context functions

fn:current-dateTime

fn:current-date

fn:current-time

fn:implicit-timezone

fn:default-collation

#### 3) Functions giving access to external information

fn:doc

fn:doc-available

fn:collection

fn:unparsed-text

fn:unparsed-text-lines

#### 4) Functions on strings

fn:string-join

fn:upper-case

fn:lower-case

fn:codepoints-to-string

fn:string-to-codepoints

fn:compare (with support for collation argument)

fn:codepoint-equal

fn:contains-token (with support for collation argument)

#### 5) String functions that use regular expressions

fn:matches

fn:replace

fn:tokenize

fn:analyze-string

#### 6) Functions that compare values in sequences

fn:distinct-values (with support for collation argument)

fn:index-of (with support for collation argument)  
fn:deep-equal (with support for collation argument)

#### 7) Maths trigonometric and exponential functions

math:pi  
math:exp  
math:exp10  
math:log  
math:log10  
math:pow  
math:sqrt  
math:sin  
math:cos  
math:tan  
math:asin  
math:acos  
math:atan  
math:atan2

#### 8) Component extraction functions on durations

fn:years-from-duration  
fn:months-from-duration  
fn:days-from-duration  
fn:hours-from-duration  
fn:minutes-from-duration  
fn:seconds-from-duration

#### 9) Constructing xs:dateTime value

fn:dateTime

#### 10) Component extraction functions on dates and times

fn:year-from-dateTime  
fn:month-from-dateTime  
fn:day-from-dateTime  
fn:hours-from-dateTime  
fn:minutes-from-dateTime  
fn:seconds-from-dateTime  
fn:timezone-from-dateTime  
fn:year-from-date  
fn:month-from-date  
fn:day-from-date  
fn:timezone-from-date  
fn:hours-from-time  
fn:minutes-from-time  
fn:seconds-from-time

fn:timezone-from-time

## 11) Built-in higher-order functions

fn:for-each

fn:filter

fn:fold-left

fn:fold-right

fn:for-each-pair

fn:sort (with support for collation argument)

fn:apply

Dynamic loading and execution, of XSLT stylesheets:

fn:transform

## 12) Functions on sequences

### 12.1 General functions on sequences

fn:empty

fn:exists

fn:head

fn:tail

fn:insert-before

fn:remove

fn:reverse

fn:subsequence

fn:unordered

### 12.2 Aggregate functions

fn:avg

fn:max

fn:min

## 13) Parsing and serializing

fn:parse-xml

fn:parse-xml-fragment

## 14) Accessors

fn:node-name

fn:string

fn:data

fn:base-uri

fn:document-uri

## 15) Functions related to QNames

fn:resolve-QName



fn:QName

#### 16) Functions related to maps

map:merge  
map:size  
map:keys  
map:contains  
map:get  
map:find  
map:put  
map:entry  
map:remove  
map:for-each

#### 17) Functions related to arrays

array:size  
array:get  
array:put  
array:append  
array:subarray  
array:remove  
array:insert-before  
array:head  
array:tail  
array:reverse  
array:join  
array:for-each  
array:filter  
array:fold-left  
array:fold-right  
array:for-each-pair  
array:sort (with support for collation argument)  
array:flatten

#### 18) Functions on JSON data

fn:parse-json  
fn:json-doc  
fn:json-to-xml  
fn:xml-to-json

**Other than the above mentioned newly implemented XPath 3.1 functions, all the functions that are specified for XPath 1.0 are available with Xalan-J's XPath 3.1 implementation as well.**

Please refer to the link <https://www.w3.org/TR/1999/REC-xpath-19991116/> (section “4 Core Function Library”) for the details about XPath 1.0 functions.

## **(2) Xalan-J XSLT 3.0 and XPath 3.1 test suite**

Xalan-J's XSLT 3.0 and XPath 3.1 test suite is available at the location :

[https://github.com/apache/xalan-java/tree/xalan-j\\_xslt3.0\\_mvn/src/test](https://github.com/apache/xalan-java/tree/xalan-j_xslt3.0_mvn/src/test) and the results of these XSL tests are available at [https://xalan.apache.org/xalan-j/xsl3/tests/xalan\\_j\\_xslt3\\_test\\_results.xml](https://xalan.apache.org/xalan-j/xsl3/tests/xalan_j_xslt3_test_results.xml).

Apache Xalan-J site

<https://xalan.apache.org/xalan-j/>

Copyright © 1999-2025 The Apache Software Foundation