XPath 1.0 Reference

Purpose:	A reference to the XML query language Xpath. Xpath is used to select nodes within an XML document. XSLT relies on Xpath. XSLT and XPATH are collectively known as XSL.
Document group:	Software Development Approach for XML

Reference Examples

Location path	Description
`/`	Locates the document root
`/books`	Locates all <books> children of the document root
`/descendant::title`	Locates all <title> elements in the document (i.e., descended from the document root)
`/comment() | /text()`	Locates all comments and text nodes in the document
`/processing-instruction("xml-stylesheet")`	Locates any processing instruction whose target is "xml-stylesheet".

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

Expression	Refers to
./author	All <author> elements within the current context. Note that this is equivalent to the expression in the next row.
author	All <author> elements within the current context.
first.name	All <first.name> elements within the current context.
/bookstore	The document element (<bookstore>) of this document.
//author	All <author> elements in the document.
book[/bookstore/@specialty = @style]	All <book> elements whose style attribute value is equal to the specialty attribute value of the <bookstore> element at the root of the document.
author/first-name	All <first-name> elements that are children of an <author> element.
bookstore//title	All <title> elements one or more levels deep in the <bookstore> element (arbitrary descendants). Note that this is different from the expression in the next row.
bookstore/*/title	All <title> elements that are grandchildren of <bookstore> elements.
bookstore//book/excerpt//emph	All <emph> elements anywhere inside

	<excerpt> children of <book> elements, anywhere inside the <bookstore> element.
.//title	All <title> elements one or more levels deep in the current context. Note that this situation is essentially the only one in which the period notation is required.
author/*	All elements that are the children of <author> elements.
book/*/last-name	All <last-name> elements that are grandchildren of <book> elements.
/	All grandchildren elements of the current context.
*[@specialty]	All elements with the specialty attribute.
@style	The style attribute of the current context.
price/@exchange	The exchange attribute on <price> elements within the current context.
price/@exchange/total	Returns an empty node set, because attributes do not contain element children. This expression is allowed by the XML Path Language (XPath) grammar, but is not strictly valid.
book[@style]	All <book> elements with style attributes, of the current context.
book/@style	The style attribute for all <book> elements of the current context.
@*	All attributes of the current element context.
./first-name	All <first-name> elements in the current context node. Note that this is equivalent to the expression in the next row.
first-name	All <first-name> elements in the current context node.
author[1]	The first <author> element in the current context node.
author[first-name][3]	The third <author> element that has a <first-name> child.
my:book	The <book> element from the my namespace.
my:*	All elements from the my namespace.
@my:*	All attributes from the my namespace (this does not include unqualified attributes on elements from the my namespace).
book[last()]	The last <book> element of the current context node.
book/author[last()]	The last <author> child of each <book> element of the current context node.
(book/author)[last()]	The last <author> element from the entire set of <author> children of <book> elements of the current context node.
book[excerpt]	All <book> elements that contain at least one <excerpt> element child.
book[excerpt]/title	All <title> elements that are children of <book> elements that also contain at least one <excerpt> element child.

book[excerpt]/author[degree]	All <author> elements that contain at least one <degree> element child, and that are children of <book> elements that also contain at least one <excerpt> element.
book[author/degree]	All <book> elements that contain <author> children that in turn contain at least one <degree> child.
author[degree][award]	All <author> elements that contain at least one <degree> element child and at least one <award> element child.
author[degree and award]	All <author> elements that contain at least one <degree> element child and at least one <award> element child.
author[(degree or award) and publication]	All <author> elements that contain at least one <degree> or <award> and at least one <publication> as the children
author[degree and not(publication)]	All <author> elements that contain at least one <degree> element child and that contain no <publication> element children.
author[not(degree or award) and publication]	All <author> elements that contain at least one <publication> element child and contain neither <degree> nor <award> element children.
author[last-name = "Bob"]	All <author> elements that contain at least one <last-name> element child with the value Bob.
author[last-name[1] = "Bob"]	All <author> elements where the first <last-name> child element has the value Bob. Note that this is equivalent to the expression in the next row.
author[last-name [position()=1]= "Bob"]	All <author> elements where the first <last-name> child element has the value Bob.
degree[@from != "Harvard"]	All <degree> elements where the from attribute is not equal to "Harvard".
author[. = "Matthew Bob"]	All <author> elements whose value is Matthew Bob.
author[last-name = "Bob" and ../price > 50]	All <author> elements that contain a <last-name> child element whose value is Bob, and a <price> sibling element whose value is greater than 50.
book[position() <= 3]	The first three books (1, 2, 3).
author[not(last-name = "Bob")]	All <author> elements that do no contain <last-name> child elements with the value Bob.
author[first-name = "Bob"]	All <author> elements that have at least one <first-name> child with the value Bob.
author[* = "Bob"]	all author elements containing any child element whose value is Bob.
author[last-name = "Bob" and first-name = "Joe"]	All <author> elements that has a <last-name> child element with the value Bob and

	a <first-name> child element with the value Joe.
price[@intl = "Canada"]	All <price> elements in the context node which have an intl attribute equal to "Canada".
degree[position() < 3]	The first two <degree> elements that are children of the context node.
p/text()[2]	The second text node in each <p> element in the context node.
ancestor::book[1]	The nearest <book> ancestor of the context node.
ancestor::book[author][1]	The nearest <book> ancestor of the context node and this <book> element has an <author> element as its child.
ancestor::author[parent::book][1]	The nearest <author> ancestor in the current context and this <author> element is a child of a <book> element.

Introduction

An XPath expression yields one of the following basic objects.

- node set
- Boolean
- number
- string

Syntax

Location Step

axis::nodetest[predicate]

```
child::person[attribute::birthdate &lt;= "1951-06-18"]
```

Each of the three components may or may not be present in a given expression.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

Compound Location Steps

Compound Location steps results in a node-set representing the union of all nodes retrieved by all of its constituent location paths

```
*[.='gibbon'] | marsupial[2] | insect
```

NodeTest

The *node-test* portion of a location step identifies the type of node(s), or the specific node(s), which make up the baseline collection of nodes from which to gather possible matches. It takes three possible forms:

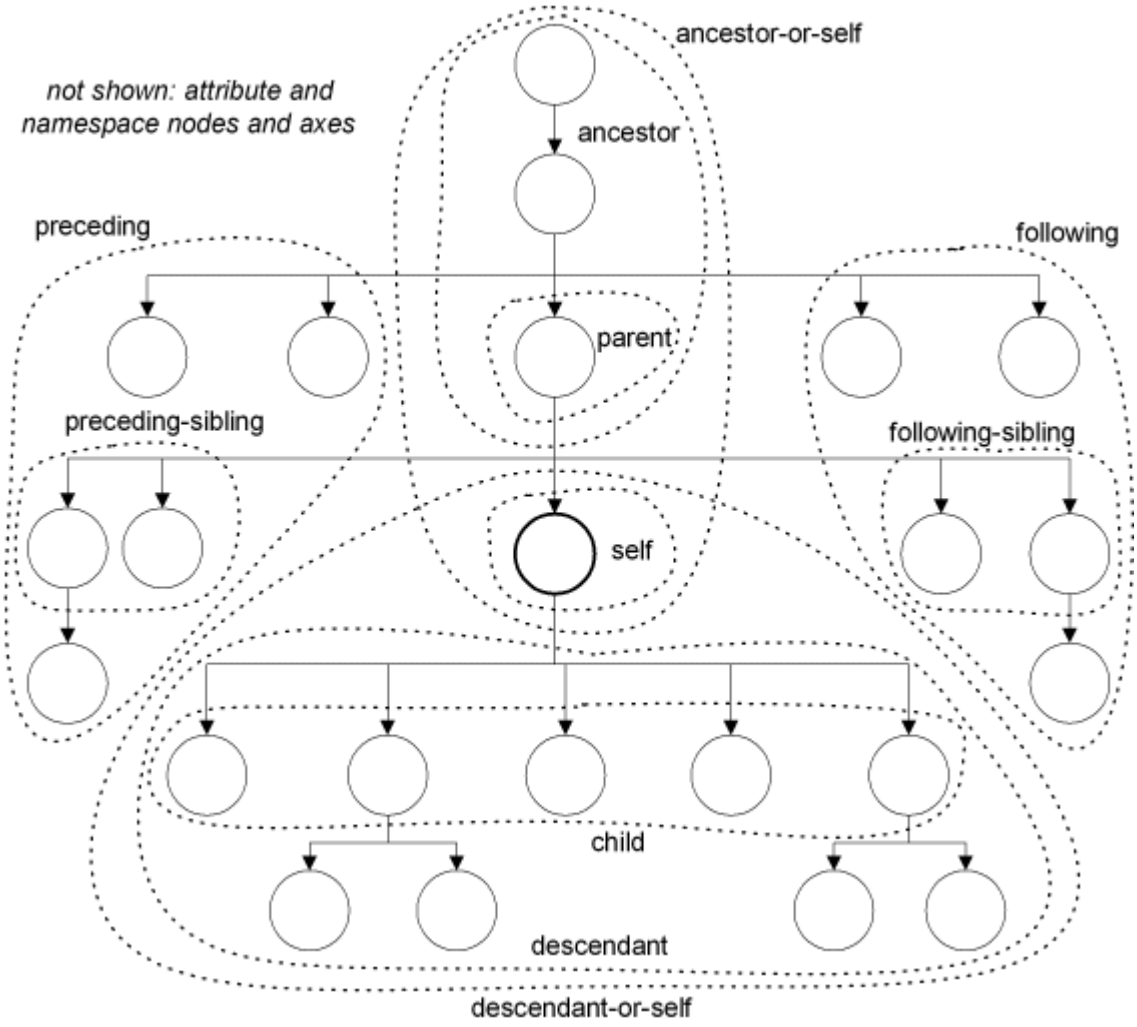
1. Use a node name to select nodes with that name.
2. Use a node type, followed by empty parentheses, (), to select nodes of that type. The following node types are available:
    - **comment()** selects element nodes;
    - **text()** selects attribute nodes;
    - **processing-instruction()** selects processing instruction nodes;
    - and **node()** selects nodes of any kind.
3. Use **processing-instruction("target")** to select processing instructions with a particular target.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

Axis

- parent::
- child::
- ancestor::
- descendant::
- ancestor-or-self::
- descendant-or-self::
- preceding::
- following::
- preceding-sibling::
- following-sibling::
- self::
- attribute::
- namespace::

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide



Crane Softwrights

Axis tells the XPath-aware processor which direction to look, up, down, or sideways in a document tree, starting from a given point of origin-the context node. Using an axis as part of a location step is:

axisname::nodetest

Where *axisname* is one of the built-in axes defined by the XPath specification, selected from the list below, and *nodetest* indicates which node(s) along that axis are of interest.

```
child::*
indicates to select all children of the context node.

/child::*
```

is to select all children of the root node, which is made the context node by virtue of the opening slash.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

Predicate

The basic form of a predicate is:

`[something operator somevalue]`

where:

- the square brackets, [and], are required;
- *something* may be either a further location step, relative to the nodes matched so far by the node-test and axis, or a built-in XPath function;
- *operator* is one of several Boolean operators, such as an equals sign or "!=" for a "does not equal" condition; and
- *somevalue* is the value to which you want to compare *something*.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

You can specify multiple predicates

An XPath location step will usually include only a single predicate. Under certain circumstances, however, it may be necessary to further filter the node-set returned by the first predicate, within a single XPath location step.

For example:

```
chapter[@author="Niikkonen, Donna"][1]
```

locates the first <chapter> child of the context node for which the <chapter>'s author attribute has the value "Niikkonen, Donna" (i.e., the first chapter written by Donna Niikkonen).

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

When using multiple predicates, you must be careful to place them in the right order.

The above does not necessarily produce the same result as:

```
chapter[1][@author="Niikkonen, Donna"]
```

which always selects the first chapter, then tests whether that was or was not written by Donna Niikkonen and selects or rejects it accordingly.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

Operators

General

/	Child operator; selects immediate children of the left-side collection. When this path operator appears at the start of the pattern, it indicates that children should be selected from the root node.
//	Recursive descent; searches for the specified element at any depth. When this path operator appears at the start of the pattern, it indicates recursive descent from the root node.
.	Indicates the current context.
..	The parent of the current context node.
*	Wildcard; selects all elements regardless of the element name.
@	Attribute; prefix for an attribute name.
@*	Attribute wildcard; selects all attributes regardless of name.
:	Namespace separator; separates the namespace prefix from the element or attribute name.

`( )`	Groups operations to explicitly establish precedence.
`[ ]`	Applies a filter pattern.
`[ ]`	Subscript operator; used for indexing within a collection.

Namespaced Elements and Attributes

Expression	Example	Description
namespacePrefix:element	`my:book`	The `<book>` element from the `my` namespace.
namespacePrefix:*	`my:*`	All elements from the `my` namespace.
`@namespacePrefix:*`	`@my:*`	All attributes from the `my` namespace. This does not include unqualified attributes on elements from the `my` namespace.

Boolean

Operator	Description
=	If values to left and right of the "=" are equal, the result is true; otherwise the result is false.
!=	If values to left and right of the "=" are not equal, the result is true; otherwise the result is false. (Can also use XPath `not()` function).
<	If value to the left of the "<" is less than value to the right, the result is true; otherwise the result is false.
>	If value to the left of the ">" is greater than value to the right, the result is true; otherwise the result is false.
<=	If value to the left of the "<=" is less than or equal to value to the right, the result is true; otherwise the result is false.
>=	If value to the left of the ">=" is greater than or equal to value to the right, the result is true; otherwise the result is false.
test1 and test2	If the results of both test1 and test2 are true, then the result of the test as a whole is true; otherwise the result is false.
test1 or test2	If the result of either test1 or test2 is true, then the result of the test as a whole is true; otherwise the result is false.

The "<" and ">" characters are significant in markup, you will typically have to escape them using the `<` and `>` entity references.

```
[attribute::empid &lt;= "69999"]
```

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

Maths

Operator	Description
+	Performs addition.
-	Performs subtraction.
`div`	Performs floating-point division according to IEEE 754.
*	Performs multiplication.
`mod`	Returns the remainder from a truncating division.

Precedence

1	()	Grouping
2	[ ]	Filters
3	/ //	Path operations
4	< <= > >=	Comparisons
5	= !=	Comparisons
6	|	Union
7	not()	Boolean not
8	and	Boolean and
9	or	Boolean or

Context Node

In lengthy XPath expressions, the context node implicitly changes as the processor hits various steps in moving along the expression from left to right.

```
attribute::scale/ancestor::weather/forecast[position()=2]/attribute::day
```

- When evaluation of the expression begins, the context node is the parent element of a scale attribute.
- The /ancestor::weather term is evaluated in terms of the scale attribute.
- /forecast[position()=2] assumes the context node to be the result of the previous location step, that is, in the case of the sample document, the second <forecast> child of the <weather> element.
- /attribute::day locates the day attribute of the elements resulting from the previous step.

For XSLT, understanding the context node, and knowing what it is at any given point, is critical to successful use of the standard.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

The template rule matches a single element type in the source tree, `<temperature>`, and the corresponding match pattern (`match="temperature"`) is what establishes the context node within the body of the template rule.

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<!-- suppress text nodes not covered in subsequent template rule -->
<xsl:template match="text()"/>

<xsl:template match="temperature">
    Today<xsl:choose>
        <xsl:when test="name(..)='today'">'s </xsl:when>
        <xsl:otherwise ><xsl:value-of select="../@day" />'s
</xsl:otherwise>
    </xsl:choose>
temperature: <xsl:value-of select="." /><br />
```

```
</xsl:template>

</xsl:stylesheet>
```

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

Absolute V Relative Location Steps

An absolute location step in an XPath expression assumes that the content will be located "starting from scratch" within the target document. It always begins with a "/" delimiter, whose presence is interpreted to mean, "Start at the document's root node."

The starting point for a relative location step is always considered to be some previously established context node.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

Common Context Operators

Context	Expression	Example
Current	./ {or Omit}	./author author
Document Root	/	/bookstore
Root Element	/* {or * if Document root is context}	
Recursive descent	//	//author { relative to the root of the document} .// prefix {relative to the current context }
Specific Element	An expression that starts with an element name refers to a query of the specific element, starting from the current context node	images/background.jpg

Context Count

The nodes set is 1 based. That is, the position() function starts counting at 1.

Processing Tree Fragments as Trees

To treat a result tree fragment as a tree or sub-tree

1. Declare the `msxsl:` namespace. For more information, see Declaring the msxsl: Namespace Prefix.

2. Establish the result tree fragment, using a variable or parameter.

```
<xsl:variable name="tree">
```

```
   <em>Emphasized and <b>bold</b></em> words
</xsl:variable>
```

3. Convert the result tree fragment to a node-set, using the `msxsl:node-set()` function.

```
<xsl:value-of select="msxsl:node-set($emprecs)/employee"/>
```

4. Use the converted node-set as you would use any other node-set, in XPath expressions, function calls, and so on.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XSLT Developer's Guide

Defaults

The `child::` axis is the default, if no axis is supplied in a location step.

```
equivalent:
child::name
name
```

Abbreviations

In predicates, when testing for position, you can more simply use the number of the position you want to test.

```
chapter[position()=1]

' contracts to
chapter[1]
```

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

The value of the predicate need not be a literal number. It can also be a reference to a variable which has a numeric value, for example, or the value of the `number()` function applied to some node or text string. Therefore, both of the forms below also demonstrate acceptable uses of this shortcut:

```
chapter[$xref]
chapter[number("12")]
```

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

Full	Abbreviation	Example	Description
`child::`	{Omit}	child::name name	The default axis if none is provided in a given location step.
`attribute::`	@	director/attribute::empdate ddirector/@empdate	
`/descendant-or-self::node()/`	//	name[.="Kim Akers"]/descendant-or-self::node()/name name[.="Kim Akers"]//name	

`self::node()`	.	self::node()/attribute::empdate ./attribute::empdate	
`parent::node()`	..	parent::node()[self::node()="Josh Barnett"] ..[self::node()="Josh Barnett"]	
	*	attribute::empID | attribute::empdate	wild-card character, used in a name node test to indicate that selected nodes may have any names at all
attribute::*	@*		
	/	name /name	While not really a shortcut, a single leading slash, representing the document root in an XPath expression always makes it an absolute path, relative to the document root.

Key Points/Ephiphanies

XPath returns a node or node set of type:

1. Elements
2. Attributes
3. Processing Instructions
4. Comments
5. Textual Content
6. Namespace
7. Document itself

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

The Root Node is not the root element.

> Processing instructions and comments in the document prolog or following the root element are considered children of the root node, as is the root element.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

All XSLT patterns are also XML Path Language (XPath) expressions. However, the reverse is not always true: not all XPath expressions are XSLT patterns.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

In `<xsl:for-each>` or `<xsl:apply-templates>`, the select returns a node-set, and each node in this set becomes the current node for further queries within it.

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

When used within an attribute assignment "{path}" has the exact same effect as <xsl:value-of select="path"/> used outside of attribute assignments.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
  <xsl:template match="/">
    <html>
      <head>
        <title>Welcome</title>
      </head>
      <body>
        <font bgcolor="{member/favoriteColor}">
          Welcome <xsl:value-of select="member/name"/>!
        </font>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

http://www.xfront.com/rescuing-xslt.html

Notes

"Although CSS can be linked directly to an XML source document, it is more effective to use CSS as a compliment within XSLT"

Microsoft XML Core Services (MSXML) 4.0 - XSLT Developer's Guide> Comparing XSLT to Cascading Style Sheets

Sources

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XSLT Developer's Guide

Microsoft

Microsoft XML Core Services (MSXML) 4.0 - XSLT Developer's Guide — Last Dated: 2003

ms-help://MS.MSDNQTR.2003FEB.1033/xmlsdk/htm/xsl_intro_7yw5.htm

MSDN 2003 > Microsoft XML Core Services (MSXML) 4.0 > XPath Developer's Guide

Microsoft

Microsoft XML Core Services (MSXML) 4.0 - XPath Developer's Guide Last Dated: 2003

ms-help://MS.MSDNQTR.2003FEB.1033/xmlsdk/htm/xpath_devguide_overview_86gn.htm

Document Licence