



BEA WebLogic Workshop™ Help

Version 8.1 SP2
November 2003

Copyright

Copyright © 2003 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software–Restricted Rights Clause at FAR 52.227–19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227–7013, subparagraph (d) of the Commercial Computer Software—Licensing clause at NASA FAR supplement 16–52.227–86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Liquid Data for WebLogic, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E–Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

Table of Contents

XQuery Reference.....	1
XQuery Alphabetical Functions and Operators Reference.....	4
XQuery Type Conversion Functions Reference.....	8
xs:string.....	9
xs:decimal.....	11
xs:integer.....	12
xs:long.....	14
xs:int.....	16
xs:short.....	18
xs:byte.....	20
xs:float.....	22
xs:double.....	24
xf:number.....	27
bea-xf:integer-sequence.....	28
xs:boolean.....	29
xs:dateTime.....	30
xs:date.....	35
xs:time.....	38
xs:gYearMonth.....	41
xs:gYear.....	43
xs:gMonthDay.....	45
xs:gMonth.....	47
xs:gDay.....	49

Table of Contents

xs:duration.....	51
xs:anyURL.....	54
xs:Name.....	55
xs:QName.....	56
XQuery String Functions Reference.....	57
xf:concat.....	58
bea-xf:trim-left.....	59
bea-xf:trim-right.....	61
bea-xf:trim.....	63
xs:normalizedString.....	65
xs:token.....	66
xf:compare.....	68
xf:starts-with.....	70
xf:ends-with.....	71
xf:contains.....	72
xf:substring.....	73
xf:string-length.....	76
xf:substring-before.....	77
xf:substring-after.....	78
xf:normalize-space.....	79
xf:upper-case.....	81
xf:lower-case.....	82
xf:translate.....	83

Table of Contents

xf:string-pad.....	85
xf:matches.....	86
xf:replace.....	90
xf:tokenize.....	93
XQuery Numeric Function Reference.....	96
bea-xf:format-number.....	97
xf:floor.....	99
xf:ceiling.....	100
xf:round.....	101
XQuery URI Functions Reference.....	102
xf:escape-URL.....	103
xf:resolve-URL.....	105
XQuery Aggregate Function Reference.....	107
xf:count.....	108
xf:avg.....	109
xf:max.....	111
xf:min.....	113
xf:sum.....	115
XQuery Node Functions Reference.....	117
xf:node-kind.....	118
xf:node-name.....	120
xf:local-name.....	122
XQuery QName Functions Reference.....	123

Table of Contents

xf:get-local-name-from-QName.....	124
xf:get-namespace-from-QName.....	126
XQuery Date Functions Reference.....	128
xf:get-year-from-dateTime.....	129
xf:get-month-from-dateTime.....	130
xf:get-day-from-dateTime.....	131
xf:get-hours-from-dateTime.....	132
xf:get-minutes-from-dateTime.....	134
xf:get-seconds-from-dateTime.....	135
xf:get-timezone-from-dateTime.....	136
xf:get-year-from-date.....	138
xf:get-month-from-date.....	139
xf:get-day-from-date.....	140
xf:get-timezone-from-date.....	141
xf:get-hours-from-time.....	143
xf:get-minutes-from-time.....	145
xf:get-seconds-from-time.....	146
xf:get-timezone-from-time.....	147
xf:add-timezone-to-dateTime.....	149
xf:remove-timezone-from-dateTime.....	152
xf:add-timezone-to-date.....	154
xf:add-timezone-to-time.....	157
xf:remove-timezone-from-time.....	160

Table of Contents

xf:current-dateTime.....	162
xf:current-date.....	164
xf:current-time.....	165
XQuery Duration Functions Reference.....	166
xf:yearMonthDuration.....	167
xf:dayTimeDuration.....	169
xf:get-years-from-yearMonthDuration.....	172
xf:get-months-from-yearMonthDuration.....	174
xf:get-days-from-dayTimeDuration.....	175
xf:get-hours-from-dayTimeDuration.....	177
xf:get-minutes-from-dayTimeDuration.....	179
xf:get-seconds-from-dayTimeDuration.....	181
xf:get-yearMonthDuration-from-dateTimes.....	183
xf:get-dayTimeDuration-from-dateTimes.....	186
XQuery Numeric Operators Reference.....	188
op:decimal-add.....	189
op:float-add.....	190
op:double-add.....	191
op:decimal-subtract.....	192
op:float-subtract.....	193
op:double-subtract.....	194
op:decimal-multiply.....	195
op:float-multiply.....	196

Table of Contents

op:double-multiply.....	197
op:decimal-divide.....	198
op:float-divide.....	200
op:double-divide.....	202
op:numeric-integer-divide.....	204
op:decimal-mod.....	206
op:float-mod.....	208
op:double-mod.....	210
XQuery Boolean Operators Reference.....	212
op:boolean-equal.....	213
op:boolean-less-than.....	215
op:boolean-greater-than.....	217
XQuery Date and Time Operators Reference.....	219
op:subtract-dates.....	220
op:subtract-times.....	222
op:add-yearMonthDuration-to-dateTime.....	224
op:add-dayTimeDuration-to-dateTime.....	226
op:subtract-yearMonthDuration-from-dateTime.....	228
op:subtract-dayTimeDuration-from-dateTime.....	230
op:add-yearMonthDuration-to-date.....	232
op:add-dayTimeDuration-to-date.....	233
op:subtract-yearMonthDuration-from-date.....	234
op:subtract-dayTimeDuration-from-date.....	236

Table of Contents

op:add-dayTimeDuration-to-time.....	238
op:subtract-dayTimeDuration-from-time.....	239
XQuery Duration Operators Reference.....	241
op:add-yearMonthDurations.....	242
op:subtract-yearMonthDurations.....	243
op:multiply-yearMonthDuration.....	245
op:divide-yearMonthDuration.....	247
op:add-dayTimeDurations.....	249
op:subtract-dayTimeDurations.....	250
op:multiply-dayTimeDuration.....	252
op:divide-dayTimeDuration.....	253
XQuery Occurrence Indicators.....	254
XQuery Namespace Conventions.....	256
XQuery Data Types.....	258
XQuery Language and XML Reference.....	260
XQuery Prologs.....	261
XQuery Expressions.....	262
XQuery FLWR Expressions.....	263
XQuery Namespaces.....	264
XQuery Function Definitions.....	265
XQuery Function Calls and Arguments.....	266
XQuery Path Expressions.....	267
XQuery Literals.....	268

Table of Contents

XQuery Combining Sequences.....	269
XQuery Conditional Expressions.....	270
XQuery Logical Expressions.....	271
XQuery Comparison Expressions.....	272
XML Element Declarations.....	273
XML Attribute Declarations.....	274
XML CDATA Sections.....	275
XML Comments.....	276
XML Prolog and Document Type Declaration.....	277

XQuery Reference

This section provides XQuery reference information for queries (written in the XQuery language) you build with the mapper provided for transformations in business processes and web services. It provides descriptions of the XQuery operator and functions available from the mapper. In addition, it provides reference information on the occurrence indicators, namespace prefixes, and XML Schema data types used in the XQuery language.

To learn more about creating data transformations in business processes, see *Guide to Data Transformation*.

For more information on transformations in web services (where they are also called *XQuery maps*), see *Transforming XML Messages with XQuery Maps*.

Note: The queries generated using *WebLogic Workshop* for transformations in business and web services conform to the W3C Working Draft 16 August 2002 of XQuery 1.0. For more information about this specification, see the following URL available from the W3C Web site:

<http://www.w3.org/TR/2002/WD-xquery-20020816/>

The queries generated using *Liquid Data for WebLogic* conform to the W3C Working Draft 30 December 2001 of XQuery 1.0. For more information about this specification, see the following URL available from the W3C Web site:

<http://www.w3.org/TR/2001/WD-xquery-20011220/>

For Liquid Data XQuery Reference information, see the Liquid Data XQuery Reference Guide.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002 available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

Topics Included in This Section

XQuery Alphabetical Functions and Operators Reference

Provides an alphabetical listing of the functions and operators available from the mapper.

XQuery Type Conversion Functions Reference

Provides descriptions of the type conversion functions available from the mapper.

XQuery String Functions Reference

Provides descriptions of the string functions available from the mapper.

XQuery Numeric Function Reference

XQuery Reference

XQuery Reference

Provides descriptions of the numeric functions available from the mapper.

XQuery Aggregate Function Reference

Provides descriptions of the aggregate functions available from the mapper.

XQuery URI Functions Reference

Provides descriptions of the URI functions available from the mapper.

XQuery Node Functions Reference

Provides descriptions of the node functions available from the mapper.

XQuery QName Functions Reference

Provides descriptions of the QName functions available from the mapper.

XQuery Date Functions Reference

Provides descriptions of the date functions available from the mapper.

XQuery Duration Functions Reference

Provides descriptions of the duration functions available from the mapper.

XQuery Numeric Operators Reference

Provides descriptions of the numeric operators available from the mapper.

XQuery Boolean Operators Reference

Provides descriptions of the boolean operators available from the mapper.

XQuery Date and Time Operators Reference

Provides descriptions of the time and date operators available from the mapper.

XQuery Duration Operators Reference

Provides descriptions of the duration operators available from the mapper.

XQuery Occurrence Indicators

Describes the XQuery occurrence indicators.

XQuery Namespace Conventions

Provides information about XQuery namespace prefixes.

XQuery Data Types

XQuery Reference

XQuery Reference

Provides information on the XML Schema data types.

XQuery Language and XML Reference

Provides XQuery language and XML links to the W3C Working Draft 16 August 2002 XQuery language and W3C XML 1.0 documentation, respectively.

XQuery Alphabetical Functions and Operators Reference

This section provides an alphabetical list of all the XQuery functions and operators available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery functions and operators. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the *XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002* available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the all the functions and operators available from the mapper functionality sorted alphabetical by namespace and then function or operator name:

- bea-xf:
 - ◆ bea-xf:format-number
 - ◆ bea-xf:integer-sequence
 - ◆ bea-xf:trim
 - ◆ bea-xf:trim-left
 - ◆ bea-xf:trim-right
- op:
 - ◆ op:add-dayTimeDurations
 - ◆ op:add-dayTimeDuration-to-date
 - ◆ op:add-dayTimeDuration-to-dateTime
 - ◆ op:add-dayTimeDuration-to-time
 - ◆ op:add-yearMonthDurations
 - ◆ op:add-yearMonthDuration-to-date
 - ◆ op:add-yearMonthDuration-to-dateTime
 - ◆ op:boolean-equal
 - ◆ op:boolean-greater-than
 - ◆ op:boolean-less-than
 - ◆ op:decimal-add
 - ◆ op:decimal-divide
 - ◆ op:decimal-mod
 - ◆ op:decimal-multiply
 - ◆ op:decimal-subtract
 - ◆ op:divide-dayTimeDuration
 - ◆ op:divide-yearMonthDuration
 - ◆ op:double-add
 - ◆ op:double-divide
 - ◆ op:double-mod
 - ◆ op:double-multiply
 - ◆ op:double-subtract
 - ◆ op:float-add

- ◆ op:float-divide
- ◆ op:float-mod
- ◆ op:float-multiply
- ◆ op:float-subtract
- ◆ op:multiply-dayTimeDuration
- ◆ op:multiply-yearMonthDuration
- ◆ op:numeric-integer-divide
- ◆ op:subtract-dates
- ◆ op:subtract-dayTimeDuration-from-date
- ◆ op:subtract-dayTimeDuration-from-dateTime
- ◆ op:subtract-dayTimeDuration-from-time
- ◆ op:subtract-dayTimeDurations
- ◆ op:subtract-times
- ◆ op:subtract-yearMonthDuration-from-date
- ◆ op:subtract-yearMonthDuration-from-dateTime
- ◆ op:subtract-yearMonthDurations
- xf:
 - ◆ xf:add-timezone-to-date
 - ◆ xf:add-timezone-to-dateTime
 - ◆ xf:add-timezone-to-time
 - ◆ xf:avg
 - ◆ xf:ceiling
 - ◆ xf:compare
 - ◆ xf:concat
 - ◆ xf:contains
 - ◆ xf:count
 - ◆ xf:current-date
 - ◆ xf:current-dateTime
 - ◆ xf:current-time
 - ◆ xf:dayTimeDuration
 - ◆ xf:ends-with
 - ◆ xf:escape-URI
 - ◆ xf:floor
 - ◆ xf:get-day-from-date
 - ◆ xf:get-day-from-dateTime
 - ◆ xf:get-days-from-dayTimeDuration
 - ◆ xf:get-dayTimeDuration-from-dateTimes
 - ◆ xf:get-hours-from-dateTime
 - ◆ xf:get-hours-from-dayTimeDuration
 - ◆ xf:get-hours-from-time
 - ◆ xf:get-local-name-from-QName
 - ◆ xf:get-minutes-from-dateTime
 - ◆ xf:get-minutes-from-dayTimeDuration
 - ◆ xf:get-minutes-from-time
 - ◆ xf:get-month-from-date
 - ◆ xf:get-month-from-dateTime
 - ◆ xf:get-months-from-yearMonthDuration
 - ◆ xf:get-namespace-from-QName
 - ◆ xf:get-seconds-from-dateTime
 - ◆ xf:get-seconds-from-dayTimeDuration
 - ◆ xf:get-seconds-from-time

- ◆ xf:get-timezone-from-date
- ◆ xf:get-timezone-from-dateTime
- ◆ xf:get-timezone-from-time
- ◆ xf:get-year-from-date
- ◆ xf:get-year-from-dateTime
- ◆ xf:get-yearMonthDuration-from-dateTimes
- ◆ xf:get-years-from-yearMonthDuration
- ◆ xf:local-name
- ◆ xf:lower-case
- ◆ xf:matches
- ◆ xf:max
- ◆ xf:min
- ◆ xf:node-kind
- ◆ xf:node-name
- ◆ xf:normalize-space
- ◆ xf:number
- ◆ xf:remove-timezone-from-dateTime
- ◆ xf:remove-timezone-from-time
- ◆ xf:replace
- ◆ xf:resolve-URI
- ◆ xf:round
- ◆ xf:starts-with
- ◆ xf:string-length
- ◆ xf:string-pad
- ◆ xf:substring
- ◆ xf:substring-after
- ◆ xf:substring-before
- ◆ xf:sum
- ◆ xf:tokenize
- ◆ xf:translate
- ◆ xf:upper-case
- ◆ xf:yearMonthDuration
- xs:
 - ◆ xs:anyURI
 - ◆ xs:boolean
 - ◆ xs:byte
 - ◆ xs:date
 - ◆ xs:dateTime
 - ◆ xs:decimal
 - ◆ xs:double
 - ◆ xs:duration
 - ◆ xs:float
 - ◆ xs:gDay
 - ◆ xs:gMonth
 - ◆ xs:gMonthDay
 - ◆ xs:gYear
 - ◆ xs:gYearMonth
 - ◆ xs:int
 - ◆ xs:integer
 - ◆ xs:long
 - ◆ xs:Name

XQuery Reference

- ◆ xs:normalizedString
- ◆ xs:QName
- ◆ xs:short
- ◆ xs:string
- ◆ xs:time
- ◆ xs:token

XQuery Type Conversion Functions Reference

This section provides descriptions of the XQuery type conversion functions available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery functions. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions available in the mapper functionality of WebLogic Workshop, a larger set of functions is provided. You can manually add invocations to these functions to queries in the **Source View** of the mapper functionality. For a list of these additional functions, see the XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002 available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the type conversion functions available from the mapper functionality:

- xs:string
- xs:decimal
- xs:integer
- xs:long
- xs:int
- xs:short
- xs:byte
- xs:float
- xs:double
- xf:number
- bea-xf:integer-sequence
- xs:boolean
- xs:dateTime
- xs:date
- xs:time
- xs:gYearMonth
- xs:gYear
- xs:gMonthDay
- xs:gMonth
- xs:gDay
- xs:duration
- xs:anyURI
- xs:Name
- xs:QName
- xf:dayTimeDuration
- xf:yearMonthDuration

xs:string

Converts the value of \$item-var to a string.

If \$item-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xs:string(item* \$item-var) > xs:string

Arguments

Data Type	Argument	Description
item*	\$item-var	Represents an atomic value or an element.

Returns

Returns the representation of \$item-var as a string.

Examples

XML Node

When you invoke the following query:

```
{-- node example --}  
let $i := <book>The Toad</book>  
return <author>{xs:string($i)}</author>
```

The following result is generated:

```
<author>The Toad</author>
```

Integer

When you invoke the following query:

```
{-- integer example --}  
let $num := xs:integer("20")  
return (<integer>{xs:string($num)}</integer>)
```

The following result is generated:

```
<integer>20</integer>
```

Float

When you invoke the following query:

```
{-- float example --}  
let $num := xs:float("44.4")  
return (<float>{xs:string($num)}</float>)
```

The following result is generated:

```
<float>44.4</float>
```

Boolean

When you invoke the following query:

```
{-- boolean example --}  
let $boolean-var := xs:boolean("true")  
return (<boolean>{xs:string($boolean-var)}</boolean>)
```

The following result is generated:

```
<boolean>true</boolean>
```

Related Topics

[W3C string function description](#)

xs:decimal

Converts \$string-var (a string) to a decimal value.

If \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xs:decimal(xs:string \$string-var) > xs:decimal

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert to a decimal number.

Returns

Returns the decimal value of \$string-var.

Examples

Simple

Invoking decimal("2.2") returns the decimal value 2.2 as shown in the following example query:

```
<decimal>{xs:decimal("2.2")}</decimal>
```

The preceding query generates the following result:

```
<decimal>2.2</decimal>
```

Null

Invoking decimal(()) returns an empty sequence. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

For example, the following example query:

```
<decimal>{xs:decimal(())}</decimal>
```

The preceding query generates the following result:

```
<decimal/>
```

Related Topics

W3C decimal data type description

xs:integer

Converts \$string-var (a string) to an integer value.

If the value of \$string-var is greater than 9,223,372,036,854,775,807 or less than -9,223,372,036,854,775,808, the TransformException exception is raised with the SYS_LONG_OVERFLOW fault code. The following error message is displayed in the mapper:

```
Error occurred while executing XQuery: BigDecimal -> Long overflow!
```

Note: For performance reasons, the integer supported by this XQuery engine is equivalent to a Java long, which is smaller than the W3C XML Schema integer, which has an arbitrary length. To learn more, see the integer description.

Note: The value \$string-var must be specified without commas as shown in the following example invocation:

```
xs:integer("999999999")
```

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

To learn more about using fault codes, see [Getting the TransformException Fault Code Programmatically](#).

Signatures

```
xs:integer(xs:string $string-var) > xs:integer
```

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert to an integer value.

Returns

Returns an integer value of \$string-var.

Examples

Simple

Invoking integer("10402") returns the integer value 10402 as shown in the following example query:

```
<integer>{xs:integer("10402")}</integer>
```

The preceding query generates the following result:

```
<integer>10402</integer>
```

Error Decimal Point is Not Allowed

Invoking `integer("104.0")` outputs an error because the decimal point is not allowed in the argument.

For example, the following example query:

```
<integer>{xs:integer("104.0")}</integer>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "104.0" to type [integer@http://www.w3.org/2001/XMLSchema#integer]
```

Error Not a Number

Invoking `integer("foo")` outputs an error because "foo" is not a number.

For example, the following example query:

```
<integer>{xs:decimal("foo")}</integer>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "foo" to type [integer@http://www.w3.org/2001/XMLSchema#integer]
```

Null

Invoking `integer()` returns an empty sequence. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

For example, the following example query:

```
<integer>{xs:integer()}</integer>
```

The preceding query generates the following result:

```
<integer/>
```

XQuery Compliance

An integer value in this XQuery engine has a maximum size of 9,223,372,036,854,775,807 and a minimum size of -9,223,372,036,854,775,808. For performance reasons, the integer supported by this XQuery engine is equivalent to a Java long, which is smaller than the W3C XML Schema integer, which has an arbitrary length.

Related Topics

[integer data type description](#)

[Getting the TransformException Fault Code Programmatically](#)

xs:long

Converts \$string-var (a string) to a long value.

If the value of \$string-var is greater than 9,223,372,036,854,775,807 or less than -9,223,372,036,854,775,808, the following error message is displayed:

Error occurred while executing XQuery: Could not cast "9223372036854775809" to type [long@http:

Note: The value \$string-var must be specified without commas as shown in the following example invocation:

```
xs:long("999999999")
```

If \$string-var is the empty sequence, the empty sequence is returned.

The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

To learn more about using fault codes, see [Getting the TransformException Fault Code Programmatically](#).

Signatures

```
xs:long(xs:string $string-var) > xs:long
```

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert to a long value.

Returns

Returns the long value of \$string-var.

Examples

Simple

Invoking `long("10403")` returns the integer value 10403 as shown in the following example query:

```
<long>{xs:long("10403")}</long>
```

The preceding query generates the following result:

```
<long>10403</long>
```


Error Decimal Point Not Allowed

Invoking `long("104.0")` outputs an error because 104.0 is not a valid integer (decimal point is not allowed.)

For example, the following example query:

```
<long>{xs:long("104.0")}</long>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "104.0" to type [long@http://www.w3.org/2001/XMLSchema#long]
```

Error Not a Number

Invoking `long("foo")` outputs an error because "foo" is not a number.

For example, the following example query:

```
<long>{xs:long("foo")}</long>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "foo" to type [long@http://www.w3.org/2001/XMLSchema#long]
```

Null

Invoking `long()` returns an empty sequence. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

For example, the following example query:

```
<long>{xs:long()}</long>
```

The preceding query generates the following result:

```
<long/>
```

Related Topics

W3C long data type description

Getting the TransformException Fault Code Programmatically

xs:int

Converts \$string-var (a string) to an int value.

If the value of \$string-var is greater than 2,147,483,647 or less than -2,147,483,648, the following error is produced:

```
Error occurred while executing XQuery: Could not cast "2147483649" to type [int@http://www.w3.org/2001/XMLSchema#int]
```

Note: The value \$string-var must be specified without commas as shown in the following example invocation:

```
xs:int("999999999")
```

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

```
xs:int(xs:string $string-var) > xs:int
```

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert to an int value.

Returns

Returns the int value of \$string-var.

Examples

Simple

Invoking int("10403") returns the integer value 10403 as shown in the following example query:

```
<int>{xs:int("10403")}</int>
```

The preceding query generates the following result:

```
<int>10403</int>
```

Error Decimal Point Not Allowed

Invoking int("104.0") outputs an error because 104.0 not a valid integer (decimal point is not allowed.)

For example, the following example query:

```
<int>{xs:int("104.0")}</int>
```

xs:int

XQuery Reference

Produces the following error:

Error occurred while executing XQuery: Could not cast "104.0" to type [int@http://www.w3.org/2001

Error Not a Number

Invoking `int("foo")` outputs an error because "foo" is not a number.

For example, the following example query:

```
<int>{xs:int("foo")}</int>
```

Produces the following error:

Error occurred while executing XQuery: Could not cast "foo" to type [int@http://www.w3.org/2001

Null

Invoking `int()` returns an empty sequence. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

For example, the following example query:

```
<int>{xs:int()}</int>
```

The preceding query generates the following result:

```
<int/>
```

Related Topics

[W3C int data type description](#)

xs:short

Converts \$string-var (a string) to a short value.

If the value of \$string-var is greater than 32,767 or less than -32,768, the following error is produced:

Error occurred while executing XQuery: Could not cast "32769" to type [short@http://www.w3.org/

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xs:short(xs:string \$string-var) > xs:short

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert to a short value.

Returns

Returns the short value of \$string-var.

Examples

Simple

Invoking short("10403") returns the integer value 10403 as shown in the following example query:

```
<short>{xs:short("10403")}</short>
```

The preceding query generates the following result:

```
<short>10403</short>
```

Error Decimal Point Not Allowed

Invoking short("104.0") outputs the an error because 104.0 is not a valid integer (decimal point is not allowed.)

For example, the following example query:

```
<short>{xs:short("104.0")}</short>
```

Produces the following error:

Error occurred while executing XQuery: Could not cast "104.0" to type [short@http://www.w3.org/

Error Not a Number

Invoking `short("foo")` outputs an error because "foo" is not a number.

For example, the following example query:

```
<short>{xs:short("foo")}</short>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "foo" to type [short@http://www.w3.org/2001/XMLSchema#short]
```

Null

Invoking `short()` returns an empty sequence. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

For example, the following example query:

```
<short>{xs:short()}</short>
```

The preceding query generates the following result:

```
<short/>
```

Related Topics

[W3C short data type description](#)

xs:byte

Converts \$string-var (a string) to a byte value.

If the value of \$string-var is greater than 127 or less than -128, the following error is produced:

Error occurred while executing XQuery: Could not cast "129" to type [byte@http://www.w3.org/200

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xs:byte(xs:string \$string-var) > xs:byte

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert to a byte.

Returns

Returns the byte value of \$string-var.

Examples

Simple

Invoking byte("104") returns the integer value 104 as shown in the following example query:

```
<byte>{xs:byte("104")}</byte>
```

The preceding query generates the following result:

```
<byte>104</byte>
```

Error Decimal Point Not Allowed

Invoking byte("104.0") outputs an error because 104.0 is not a valid integer (decimal point is not allowed.)

For example, the following example query:

```
<byte>{xs:byte("104.0")}</byte>
```

Produces the following error:

Error occurred while executing XQuery: Could not cast "104.0" to type [byte@http://www.w3.org/200

Error Not a Number

Invoking `byte("foo")` outputs an error because "foo" is not a number.

For example, the following example query:

```
<byte>{xs:byte("foo")}</byte>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "foo" to type [byte@http://www.w3.org/200
```

Null

Invoking `byte()` returns an empty sequence. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

For example, the following example query:

```
<byte>{xs:byte()}</byte>
```

The preceding query generates the following result:

```
<byte/>
```

Related Topics

W3C byte data type description

xs:float

Converts \$string-var (a string) to a 32 bit floating point value. The data type float corresponds to the IEEE single-precision 32-bit floating point type (IEEE Std 754-1985).

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xs:float(xs:string \$string-var) > xs:float

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert to a float.

Returns

Returns the floating point value of \$string-var.

Examples

Simple

Invoking float("1.1") returns the floating point value of 1.1 as shown in the following example query:

```
<float>{xs:float("1.1")}</float>
```

The preceding query generates the following result:

```
<float>1.1</float>
```

Exponent

Invoking float("-104.345e2") returns the floating point value of -10434.5 as shown in the following example query:

```
<float>{xs:float("-104.345e2")}</float>
```

The preceding query generates the following error:

```
<float>-10434.5</float>
```

NaN

The string: NaN (Not a Number) is a legal argument, as shown in the following example query:

```
<float>{xs:float("NaN")}</float>
```

xs:float

The preceding query generates the following result:

```
<float>NaN</float>
```

INF and -INF

The strings: INF (positive infinity) and -INF (negative infinity) are legal arguments as shown in the following example query:

```
<infinity>
  <positive>{xs:float("INF")}</positive>
  <negative>{xs:float("-INF")}</negative>
</infinity>
```

The preceding query generates the following result:

```
<infinity>
  <positive>Infinity</positive>
  <negative>-Infinity</negative>
</infinity>
```

Error Invalid Exponent

Invoking float("10.1e2.1") outputs an error because 2.1 is not an integer. For example, the following example query:

```
<float>{xs:float("10.1e2.1")}</float>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "10.1e2.1" to type [float@http://www.w3.org/2001/XMLSchema#float]
```

Null

Invoking float(()) returns the empty sequence. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

For example, the following example query:

```
<float>{xs:float(())}</float>
```

The preceding query generates the following result:

```
<float/>
```

Related Topics

W3C float data type description

xs:double

Converts \$string-var (a string) to a double precision (64 bit) floating point value.

The data type double corresponds to IEEE double-precision 64-bit floating point type (IEEE 754-1985).

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xs:double(xs:string \$string-var) > xs:double

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert to a double.

Returns

Returns the double precision floating point value of \$string-var.

Examples

Simple

Invoking double("1.1") returns the double precision floating point value of 1.1 as shown in the following example query:

```
<double>{xs:double("1.1")}</double>
```

The preceding query generates the following result:

```
<double>1.1</double>
```

Exponent

Invoking double("-104.345e2") returns the double floating point value of -10434.5 as shown in the following example query:

```
<double>{xs:double("-104.345e2")}</double>
```

The preceding query generates the following result:

```
<double>-10434.5</double>
```

NaN

The string: NaN (Not a Number) is a legal argument, as shown in the following example query:

```
<double>{xs:double("NaN")}</double>
```

The preceding query generates the following result:

```
<double>NaN</double>
```

INF and -INF

The strings: INF (positive infinity) and -INF (negative infinity) are legal arguments as shown in the following example query:

```
<infinity>
  <positive>{xs:double("INF")}</positive>
  <negative>{xs:double("-INF")}</negative>
</infinity>
```

The preceding query generates the following result:

```
<infinity>
  <positive>Infinity</positive>
  <negative>-Infinity</negative>
</infinity>
```

Error Invalid Exponent

Invoking double("10.1e2.1") outputs an error because 2.1 is not an integer. For example, the following example query:

```
<double>{xs:double("10.1e2.1")}</double>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "10.1e2.1" to type [double@http://www.w3.
```

Error Null

Invoking double(()) returns the empty sequence. The empty sequence is a sequence containing zero items (), which is similar to null in SQL

For example, the following example query:

```
<double>{xs:double(())}</double>
```

The preceding query generates the following result:

```
<double/>
```

Related Topics

W3C double data type description

xf:number

Converts the value of \$node-var (XML element) to a double precision floating point value.

Signatures

xf:number(xf:node \$node-var) > xs:double

Arguments

Data Type	Argument	Description
xf:node	\$node-var	Represents an XML node.

Returns

Returns the double precision floating point value of \$node-var.

Examples

XML Node

When you invoke the following query:

```
{-- node example --}  
let $numnode := <a><b>12.1</b></a>  
return ( <num>{xf:number($numnode/b)}</num> )
```

The following result is generated:

```
<num>12.1</num>
```

Related Topics

W3C number function description

bea-xf:integer-sequence

Converts a sequence of XML nodes into a sequence of integers. This function is usually used in conjunction with the `xf:max` and `xf:min` functions.

Signatures

`bea-xf:integer-sequence(node* $node-var) > xs:integer*`

Arguments

Data Type	Argument	Description
node*	\$node-var	Represents a sequence of XML nodes.

Returns

Returns a sequence of integers converted from XML nodes.

Examples

Simple

Invoking the following query returns a sequence of integers, as shown in the following example query:

```
let $x := <a>100</a>
let $y := <b>2</b>
let $z := <c>50</c>
return <result>{bea-xf:integer-sequence(($x,$y,$z))}</result>
```

The preceding query generates the following result:

```
<result>100 2 50</result>
```

xs:boolean

Converts a \$string-var (a string) to a boolean value.

Signatures

xs:boolean(xs:string \$string-var) > xs:boolean

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert to a boolean value

Returns

Returns the boolean value of the passed in string.

Examples

Simple

Invoking xs:boolean("false") returns the boolean value false, as shown in the following example query:

```
<result>{xs:boolean("false")}</result>
```

The preceding query generates the following result:

```
<result>>false</result>
```

Numeric

Invoking xs:boolean("1") returns the boolean value true, as shown in the following example query:

```
<result>{xs:boolean("1")}</result>
```

The preceding query generates the following result:

```
<result>>true</result>
```

Related Topics

W3C boolean function description

xs:dateTime

Converts \$string-var (a string in the dateTime format) to the dateTime data type.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If the value of \$string-var is not valid to the dateTime format the following error is reported:

Could not cast "invalid_dateTime_string" to type [date@http://www.w3.org/2001/XMLSchema] is displayed.

Where *invalid_dateTime_string* is the string not valid to the date format, for example: "2003-08-16T21:10".

Signatures

xs:dateTime(xs:string \$string-var) > xs:dateTime

Arguments

Data Type	Argument	Description	
xs:string	\$string-var	Represents a string with the date and time specified with one of the following formats: <ul style="list-style-type: none">• <i>YYYY-MM-DDThh:mm:ss.ssssss</i>• <i>YYYY-MM-DDThh:mm:ss.ssssssZ</i>• <i>YYYY-MM-DDThh:mm:ss.ssssss+hh:mm</i>• <i>YYYY-MM-DDThh:mm:ss.ssssss-hh:mm</i>	
		<i>YYYY</i>	Year.
		<i>MM</i>	Month (as a number).
		<i>DD</i>	Day.
		<i>T</i>	Date and Time separator.
		<i>hh</i>	Number of hours.
		<i>mm</i>	Number of minutes.
		<i>ss.ssssss</i>	Number of seconds.
			Seconds can be specified up to 7 digits after the decimal place. (Format <i>xx.xxxxxxx</i>)
		<i>:</i>	Separator between hours, minutes, or seconds.
		<i>+</i>	Positive time zone offset. This option is optional. If a plus or minus is not specified, + is assumed.
		<i>-</i>	Negative time zone offset. (Optional)
		<i>hh</i>	Number of hours that the time zone differs from UTC.
		<i>mm</i>	Number of minutes that the time zone differs from UTC.

	Z	Indicates that the time corresponds to the UTC time zone.
--	---	---

Returns

Returns the specified date and time in the `dateTime` data type.

Examples

Simple

Invoking `dateTime("2003-08-16T21:10:50")` returns a `dateTime` value corresponding to August 16th, 2003 at 9:10PM (21:10 in twenty four hour time) and 50 seconds in the current time zone, as shown in the following example query:

```
let $mydate := xs:dateTime("2003-08-16T21:10:50")
return
<components>
  <year>{xf:get-year-from-dateTime($mydate)}</year>
  <month>{xf:get-month-from-dateTime($mydate)}</month>
  <day>{xf:get-day-from-dateTime($mydate)}</day>
  <hour>{xf:get-hours-from-dateTime($mydate)}</hour>
  <minute>{xf:get-minutes-from-dateTime($mydate)}</minute>
  <second>{xf:get-seconds-from-dateTime($mydate)}</second>
</components>
```

The preceding query, generates the following XML result:

```
<components>
  <year>2003</year>
  <month>8</month>
  <day>16</day>
  <hour>21</hour>
  <minute>10</minute>
  <second>50</second>
</components>
```

Seconds with Decimal

Invoking `dateTime("2003-08-16T21:10:50.577")` returns a `dateTime` value corresponding to August 16th, 2003 at 9:10 PM (21:10 in twenty four hour time) and 50.577 seconds in the current time zone, as shown in the following example query:

```
let $mydate := xs:dateTime("2003-08-16T21:10:50.557")
return
<components>
  <year>{xf:get-year-from-dateTime($mydate)}</year>
  <month>{xf:get-month-from-dateTime($mydate)}</month>
  <day>{xf:get-day-from-dateTime($mydate)}</day>
  <hour>{xf:get-hours-from-dateTime($mydate)}</hour>
  <minute>{xf:get-minutes-from-dateTime($mydate)}</minute>
  <second>{xf:get-seconds-from-dateTime($mydate)}</second>
</components>
```

The preceding query, generates the following XML result:

```

<components>
  <year>2003</year>
  <month>8</month>
  <day>16</day>
  <hour>21</hour>
  <minute>10</minute>
  <second>50.5570000</second>
</components>

```

UTC Time Zone

Invoking `dateTime("2003-08-16T21:10:50Z")` returns a `dateTime` value corresponding to August 16th, 2003 at 9:10 PM (21:10 in twenty four hour time) and 50 seconds, in the UTC time zone, as shown in the following example query:

```

let $mydate := xs:dateTime("2003-08-16T21:10:50Z")
return
<components>
  <year>{xf:get-year-from-dateTime($mydate)}</year>
  <month>{xf:get-month-from-dateTime($mydate)}</month>
  <day>{xf:get-day-from-dateTime($mydate)}</day>
  <hour>{xf:get-hours-from-dateTime($mydate)}</hour>
  <minute>{xf:get-minutes-from-dateTime($mydate)}</minute>
  <second>{xf:get-seconds-from-dateTime($mydate)}</second>
</components>

```

The preceding query, generates the following XML result:

```

<components>
  <year>2003</year>
  <month>8</month>
  <day>16</day>
  <hour>21</hour>
  <minute>10</minute>
  <second>50</second>
</components>

```

Offset Time Zone

Invoking `dateTime("2003-08-16T13:10:50-07:00")` returns a `dateTime` value corresponding to August 16th, 2003 at 1:10 PM (13:10 in twenty four hour time) and 50 seconds, in the Pacific Daylight Savings (PDT) time zone that is offset by -7 hours from UTC (Universal Time, Coordinated), as shown in the following example query:

```

let $mydate := xs:dateTime("2003-08-16T13:10:50-07:00")
return
<components>
  <year>{xf:get-year-from-dateTime($mydate)}</year>
  <month>{xf:get-month-from-dateTime($mydate)}</month>
  <day>{xf:get-day-from-dateTime($mydate)}</day>
  <hour>{xf:get-hours-from-dateTime($mydate)}</hour>
  <minute>{xf:get-minutes-from-dateTime($mydate)}</minute>
  <second>{xf:get-seconds-from-dateTime($mydate)}</second>
</components>

```

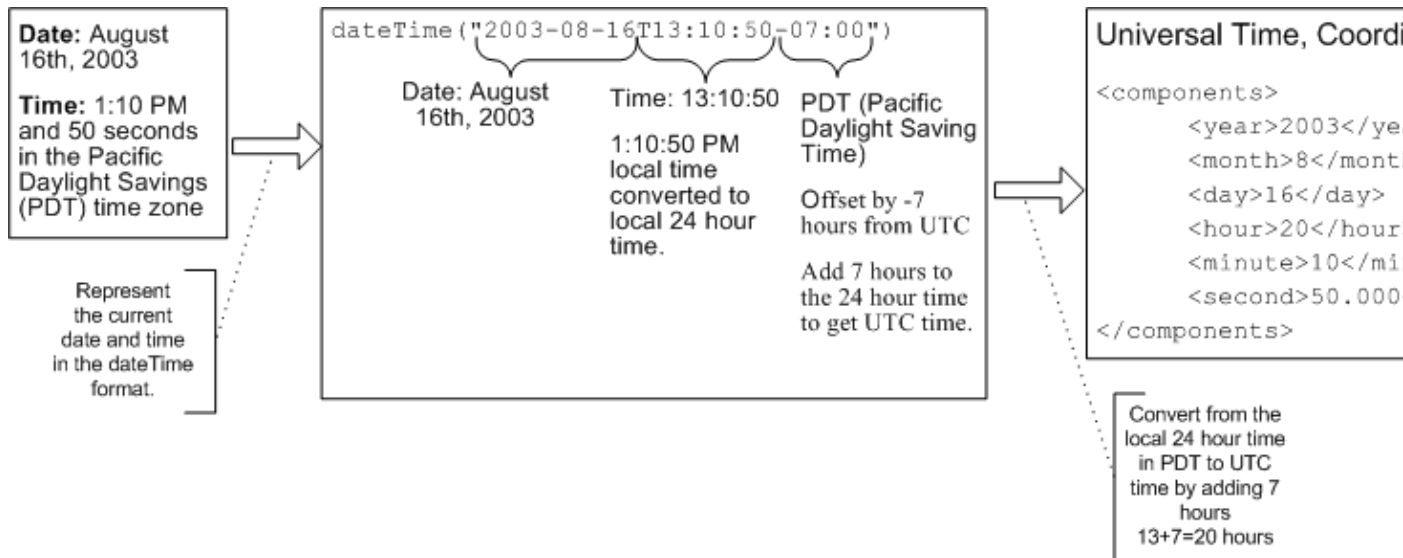
The preceding query, generates the following XML result:

```

<components>
  <year>2003</year>
  <month>8</month>
  <day>16</day>
  <hour>20</hour>
  <minute>10</minute>
  <second>50</second>
</components>

```

The conversion of the date and time is shown in the following figure.



Error No Seconds

Invoking `dateTime("2003-08-16T21:10")` outputs an error because seconds are not specified.

For example, the following example query:

```
<result>{xs:dateTime("2003-08-16T21:10")}</result>
```

Produces the following error:

Error occurred while executing XQuery: Could not cast "2003-08-16T21:10" to type [dateTime@http://www.w3.org/2001/XMLSchema#dateTime]

Error Incorrect Format

Invoking `dateTime("2003-8-16T21:10:50")` outputs an error because the month is not specified using two digits.

For example, the following example query:

```
<result>{xs:dateTime("2003-8-16T21:10:50")}</result>
```

Produces the following error:

Error occurred while executing XQuery: Could not cast "2003-8-16T21:10:50" to type [dateTime@http://www.w3.org/2001/XMLSchema#dateTime]

Related Topics

[W3C dateTime data type description](#)

xs:date

Converts \$string-var (a string in the date format) to the date data type.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If the value of \$string-var is not valid to the date format the following error is reported:

Could not cast "*invalid_date_string*" to type [date@http://www.w3.org/2001/XMLSchema] is displayed.

Where *invalid_date_string* is the string not valid to the date format, for example: "2003-04-31".

Signatures

xs:date(xs:string \$string-var) > xs:date

Arguments

Data Type	Argument	Description	
xs:string	\$string-var	Represents a string with the date specified with one of the following formats: <ul style="list-style-type: none">• YYYY-MM-DD• YYYY-MM-DDZ• YYYY-MM-DD+hh:mm• YYYY-MM-DD-hh:mm	
		YYYY	Year.
		MM	Month (as a number).
		DD	Day.
		–	Separator between year, month, and day.
		hh	Number of hours.
		mm	Number of minutes.
		:	Separator between hours and minutes.
		+	Positive time zone offset (If a plus or minus is not specified, + is assumed.)
		–	Negative time zone offset.
		hh	Number of hours that the time zone differs from UTC.
		mm	Number of minutes that the time zone differs from UTC.
		Z	Indicates that the time corresponds to the UTC time zone.

Returns

Returns the specified date in the date data type.

Examples

Simple

Invoking `date("2003-08-16")` returns a date value corresponding to August 16th, 2003 in the current time zone, as shown in the following example query:

```
let $mydate := xs:date("2003-08-16")
return
<components>
  <year>{xf:get-year-from-date($mydate)}</year>
  <month>{xf:get-month-from-date($mydate)}</month>
  <day>{xf:get-day-from-date($mydate)}</day>
</components>
```

The preceding query, generates the following XML result:

```
<components>
  <year>2003</year>
  <month>8</month>
  <day>16</day>
</components>
```

UTC Time Zone

Invoking `date("2003-08-16Z")` returns a date value corresponding to August 16th, 2003 in the UTC time zone, as shown in the following example query:

```
let $mydate := xs:date("2003-08-16Z")
return
<components>
  <year>{xf:get-year-from-date($mydate)}</year>
  <month>{xf:get-month-from-date($mydate)}</month>
  <day>{xf:get-day-from-date($mydate)}</day>
</components>
```

Note: The Z in the a date string, specifies that the date is specified in the UTC time zone.

The preceding query, generates the following XML result:

```
<components>
  <year>2003</year>
  <month>8</month>
  <day>16</day>
</components>
```

Offset Time Zone

Invoking `date("2003-08-16-02:00")` returns a date value corresponding to August 16th, 2003 in a time zone that is offset by -2 hours from UTC, as shown in the following example query:

```
let $mydate := xs:date("2003-08-16-02:00")
return
<components>
  <year>{xf:get-year-from-date($mydate)}</year>
```

XQuery Reference

```
<month>{xf:get-month-from-date($mydate)}</month>
<day>{xf:get-day-from-date($mydate)}</day>
</components>
```

The preceding query, generates the following XML result:

```
<components>
  <year>2003</year>
  <month>8</month>
  <day>16</day>
</components>
```

Error Not A Valid Date

Invoking `date("2003-04-31")` outputs an error because April 31, 2003 is not a valid date. (There are not 31 days in April.)

For example, the following example query:

```
<result>{xs:date("2003-04-31")}</result>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "2003-04-31" to type [date@http://www.w3.org/2001/XMLSchema#date]
```

Error Incorrect Format

Invoking `date("2003-8-16")` outputs an error because the month is not specified using two digits.

For example, the following example query:

```
<result>{xs:date("2003-8-16")}</result>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "2003-8-16" to type [date@http://www.w3.org/2001/XMLSchema#date]
```

Related Topics

[W3C date data type description](#)

xs:time

Converts \$string-var (a string in the time format) to the time data type.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If the value of \$string-var is not valid to the time format the following error is reported:

Could not cast "*invalid_time_string*" to type [time@http://www.w3.org/2001/XMLSchema] is displayed.

Where *invalid_time_string* is the string not valid to the time format, for example: "23:5:44".

Signatures

xs:time(xs:string \$string-var) > xs:time

Arguments

Data Type	Argument	Description	
xs:string	\$string-var	Represents a string with the time specified with one of the following formats: <ul style="list-style-type: none">• <i>hh:mm:ss.sssssss</i>• <i>hh:mm:ss.sssssssZ</i>• <i>hh:mm:ss+hh:mm</i>• <i>hh:mm:ss-hh:mm</i>	
		<i>YYYY</i>	Year.
		<i>MM</i>	Month (as a number).
		<i>DD</i>	Day.
		<i>hh</i>	Number of hours.
		<i>mm</i>	Number of minutes.
		<i>ss.sssssss</i>	Number of seconds.
			Seconds can be specified up to 7 digits after the decimal place. (Format <i>xx.xxxxxxx</i>)
		:	Separator between hours, minutes and seconds.
		+	Positive time zone offset. This option is optional. If a plus or minus is not specified, + is assumed.
		-	Negative time zone offset. (Optional)
		<i>hh</i>	Number of hours that the time zone differs from UTC.
		<i>mm</i>	Number of minutes that the time zone differs from UTC.
		Z	Indicates that the time corresponds to the UTC time zone.

Returns

Returns the specified time in the time data type.

Examples

Simple

Invoking `time("23:15:45")` returns a time value corresponding to 11:15 PM and 45 seconds in the current time zone, as shown in the following example query:

```
let $mytime := xs:time("23:15:45")
return
<components>
  <hours>{xf:get-hours-from-time($mytime)}</hours>
  <minutes>{xf:get-minutes-from-time($mytime)}</minutes>
  <seconds>{xf:get-seconds-from-time($mytime)}</seconds>
</components>
```

The preceding query, generates the following XML result:

```
<components>
  <hours>23</hours>
  <minutes>15</minutes>
  <seconds>45</seconds>
</components>
```

UTC Time Zone

Invoking `time("23:15:45Z")` returns a time value corresponding to 11:15 PM and 45 seconds in the UTC time zone, as shown in the following example query:

```
let $mytime := xs:time("23:15:45Z")
return
<components>
  <hours>{xf:get-hours-from-time($mytime)}</hours>
  <minutes>{xf:get-minutes-from-time($mytime)}</minutes>
  <seconds>{xf:get-seconds-from-time($mytime)}</seconds>
</components>
```

The preceding query, generates the following XML result:

```
<components>
  <hours>23</hours>
  <minutes>15</minutes>
  <seconds>45</seconds>
</components>
```

Error Incorrect Format

Invoking `time("23:5:44")` outputs an error because the minutes is not specified using two digits.

For example, the following example query:

```
<result>{xs:time("23:5:44")}</result>
```

XQuery Reference

Produces the following error:

Error occurred while executing XQuery: Could not cast "23:5:44" to type [time@http://www.w3.org

Related Topics

[W3C time data type description](#)

xs:gYearMonth

Converts \$string-var (a string in the gYearMonth format) to the gYearMonth data type.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If the value of \$string-var is not valid to the gYearMonth format, the following error is reported:

Could not cast "*invalid_gYearMonth_string*" to type [gYearMonth@http://www.w3.org/2001/XMLSchema] is displayed.

Where *invalid_gYearMonth_string* is the string not valid to the time format, for example: "2003-8".

Signatures

xs:gYearMonth(xs:string \$string-var) > xs:gYearMonth

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents a string with the month and year specified with one of the following formats: <ul style="list-style-type: none">• <i>YYYY-MM</i>• <i>YYYY-MMZ</i>• <i>YYYY-MM+hh:mm</i>• <i>YYYY-MM-hh:mm</i>
		<i>YYYY</i> Year.
		- Separator between year and months.
		<i>MM</i> Month (as a number).
		+ Positive time zone offset. This option is optional. If a plus or minus is not specified, + is assumed.
		- Negative time zone offset. (Optional)
		<i>hh</i> Number of hours that the time zone differs from UTC.
		<i>mm</i> Number of minutes that the time zone differs from UTC.
		Z Indicates that the time corresponds to the UTC time zone.

Returns

Returns the specified month and year in the gYearMonth data type.

Examples

Simple

Invoking `gYearMonth("2003-08")` returns a `gYearMonth` value corresponding to August 2003, as shown in the following example query:

```
<result>{xs:gYearMonth("2003-08")}</result>
```

The preceding query, generates the following XML result:

```
<result>2003-08</result>
```

UTC Time Zone

Invoking `gYearMonth("2003-08Z")` returns a `gYearMonth` value corresponding to August 2003 in the UTC time zone, as shown in the following example query:

```
<result>{xs:gYearMonth("2003-08Z")}</result>
```

The preceding query, generates the following XML result:

```
<result>2003-08Z</result>
```

Error Incorrect Format

Invoking `gYearMonth("2003-8")` outputs an error because the month is not specified using two digits.

For example, the following example query:

```
<result>{xs:gYearMonth("2003-8")}</result>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "2003-8" to type [gYearMonth@http://www.w
```

Related Topics

W3C `gYearMonth` data type description

xs:gYear

Converts \$string-var (a string in the gYear format) to the gYear data type.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If the value of \$string-var is not valid to the gYear format, the following error is reported:

Could not cast "*invalid_gYear_string*" to type [gYear@http://www.w3.org/2001/XMLSchema] is displayed.

Where *invalid_gYear_string* is the string not valid to the time format, for example: "20-2003".

Signatures

xs:gYear(xs:string \$string-var) > xs:gYear

Arguments

Data Type	Argument	Description	
xs:string	\$string-var	Represents a string with the year specified with one of the following formats: <ul style="list-style-type: none">• YYYY• YYYYZ• YYYY+hh:mm• YYYY-hh:mm	
		YYYY	Year.
		+	Positive time zone offset. This option is optional. If a plus or minus is not specified, + is assumed.
		-	Negative time zone offset. (Optional)
		hh	Number of hours that the time zone differs from UTC.
		mm	Number of minutes that the time zone differs from UTC.
		Z	Indicates that the time corresponds to the UTC time zone.

Returns

Returns the specified year in the gYear data type.

Examples

Simple

Invoking gYear("2003") returns a gYear value corresponding to the year 2003, as shown in the following example query:

```
<result>{xs:gYear("2003")}</result>
```

The preceding query, generates the following XML result:

```
<result>2003</result>
```

UTC Time Zone

Invoking `gYear("2003Z")` returns a `gYear` value corresponding to the year 2003 in the UTC time zone, as shown in the following example query:

```
<result>{xs:gYear("2003Z")}</result>
```

The preceding query, generates the following XML result:

```
<result>2003Z</result>
```

Related Topics

[W3C gYear data type description](#)

xs:gMonthDay

Converts \$string-var (a string in the gMonthDay format) to the gMonthDay data type.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If the value of \$string-var is not valid to the gMonthDay format, the following error is reported:

Could not cast "*invalid_gMonthDay_string*" to type [gMonthDay@http://www.w3.org/2001/XMLSchema] is displayed.

Where *invalid_gMonthDay_string* is the string not valid to the time format, for example: "08-15".

Signatures

xs:gMonthDay(xs:string \$string-var) > xs:gMonthDay

Arguments

Data Type	Argument	Description	
xs:string	\$string-var	Represents a string with the month and day specified with one of the following formats: <ul style="list-style-type: none">• <i>--MM-DD</i>• <i>--MM-DDZ</i>• <i>--MM-DD+hh:mm</i>• <i>--MM-DD-hh:mm</i>	
		<i>--</i>	Beginning prefix.
		<i>MM</i>	Month (as a number).
		<i>-</i>	Separator between months and days.
		<i>DD</i>	Day.
		<i>+</i>	Positive time zone offset. This option is optional. If a plus or minus is not specified, + is assumed.
		<i>-</i>	Negative time zone offset. (Optional)
		<i>hh</i>	Number of hours that the time zone differs from UTC.
		<i>mm</i>	Number of minutes that the time zone differs from UTC.
		<i>Z</i>	Indicates that the time corresponds to the UTC time zone.

Returns

Returns the specified month and day in the gMonthDay data type.

Examples

Simple

Invoking `gMonthDay("--08-15")` returns a `gMonthDay` value corresponding to August 15, as shown in the following example query:

```
<result>{xs:gMonthDay("--08-15")}</result>
```

The preceding query, generates the following XML result:

```
<result>--08-15</result>
```

UTC Time Zone

Invoking `gMonthDay("--08-15Z")` returns a `gMonthDay` value corresponding to August 15 in the UTC time zone, as shown in the following example query:

```
<result>{xs:gMonthDay("--08-15Z")}</result>
```

The preceding query, generates the following XML result:

```
<result>--08-15Z</result>
```

Error Incorrect Format

Invoking `gMonthDay("08-15")` outputs an error because the `--` prefix is missing.

For example, the following example query:

```
<result>{xs:gMonthDay("08-15")}</result>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "08-15" to type [gMonthDay@http://www.w3.
```

Related Topics

W3C `gMonthDay` data type description

xs:gMonth

Converts \$string-var (a string in the gMonth format) to the gMonth data type.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If the value of \$string-var is not valid to the gMonth format, the following error is reported:

Could not cast "*invalid_gMonth_string*" to type [gMonth@http://www.w3.org/2001/XMLSchema] is displayed.

Where *invalid_gMonth_string* is the string not valid to the time format, for example: "08".

Signatures

xs:gMonth(xs:string \$string-var) > xs:gMonth

Arguments

Data Type	Argument	Description	
xs:string	\$string-var	Represents a string with the month and day specified with one of the following formats: <ul style="list-style-type: none">• <i>--MM--</i>• <i>--MM--Z</i>• <i>--MM--+hh:mm</i>• <i>--MM---hh:mm</i>	
		<i>--</i>	Beginning prefix.
		<i>MM</i>	Month (as a number).
		<i>--</i>	Separator.
		<i>+</i>	Positive time zone offset. (Optional) If a plus or minus is not specified, + is assumed.
		<i>-</i>	Negative time zone offset. (Optional)
		<i>hh</i>	Number of hours that the time zone differs from UTC.
		<i>mm</i>	Number of minutes that the time zone differs from UTC.
		<i>Z</i>	Indicates that the time corresponds to the UTC time zone.

Returns

Returns the specified month in the gMonth data type.

Examples

Simple

Invoking `gMonth("--08--")` returns a `gMonth` value corresponding to the month of August, as shown in the following example query:

```
<result>{xs:gMonth("--08--")}</result>
```

The preceding query, generates the following XML result:

```
<result>--08--</result>
```

UTC Time Zone

Invoking `gMonth("--08--Z")` returns a `gMonth` value corresponding to the month of August in the UTC time zone, as shown in the following example query:

```
<result>{xs:gMonth("--08--Z")}</result>
```

The preceding query, generates the following XML result:

```
<result>--08--Z</result>
```

Error Incorrect Format

Invoking `gMonth("08")` outputs an error because the `--` prefix is missing.

For example, the following example query:

```
<result>{xs:gMonth("08")}</result>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "08" to type [gMonth@http://www.w3.org/2001/XMLSchema#gMonth]
```

Related Topics

W3C `gMonth` data type description

xs:gDay

Converts \$string-var (a string in the gDay format) to the gDay data type.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If the value of \$string-var is not valid to the gDay format, the following error is reported:

Could not cast "*invalid_gDay_string*" to type [gDay@http://www.w3.org/2001/XMLSchema] is displayed.

Where *invalid_gDay_string* is the string not valid to the time format, for example: "15".

Signatures

xs:gDay(xs:string \$string-var) > xs:gDay

Arguments

Data Type	Argument	Description	
xs:string	\$string-var	Represents a string with the day specified with one of the following formats: <ul style="list-style-type: none">• <i>DD</i>• <i>DDZ</i>• <i>DD+hh:mm</i>• <i>DD-hh:mm</i>	
		----	Beginning prefix.
		<i>DD</i>	Day.
		+	Positive time zone offset. This option is optional. If a plus or minus is not specified, + is assumed.
		-	Negative time zone offset. (Optional)
		<i>hh</i>	Number of hours that the time zone differs from UTC.
		<i>mm</i>	Number of minutes that the time zone differs from UTC.
		<i>Z</i>	Indicates that the time corresponds to the UTC time zone.

Returns

Returns the specified day in the gDay data type.

Examples

Simple

Invoking `gDay("----15")` returns a `gDay` value corresponding to the 15th of the month, as shown in the following example query:

```
<result>{xs:gDay( "----15" )}</result>
```

The preceding query, generates the following XML result:

```
<result>----15</result>
```

UTC Time Zone

Invoking `gDay("----15Z")` returns a `gDay` value corresponding to the 15th of the month in the UTC time zone, as shown in the following example query:

```
<result>{xs:gDay( "----15Z" )}</result>
```

The preceding query, generates the following XML result:

```
<result>----15Z</result>
```

Error Incorrect Format

Invoking `gDay("15")` outputs an error because the `----` prefix is missing.

For example, the following example query:

```
<result>{xs:gDay( "15" )}</result>
```

Produces the following error:

```
Error occurred while executing XQuery: Could not cast "15" to type [gDay@http://www.w3.org/2001
```

Related Topics

W3C `gDay` data type description

xs:duration

Converts \$string-var (a string in the duration format) to the duration data type.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xs:duration(xs:string \$string-var) > xs:duration

Arguments

Data Type	Argument	Description
xs:string	\$string-var	<p>Represents a string with the duration specified with one of the following formats:</p> <ul style="list-style-type: none">• PyYmMdDT<i>h</i>HiMsS• -PyYmMdDT<i>h</i>HiMsS <p>Note: Specifying the number of years, months, days, hours, minutes, or seconds is optional any of the substrings: <i>yY</i>, <i>mM</i>, <i>dD</i>, <i>hH</i>, <i>iM</i>, or <i>sS</i> do not have to be specified. For example, the following are valid durations: P10Y, P1D, -P11M, -P10Y7D, P2YT5H, or P6YT5H10S.</p>
	-	<p>Duration is a negative amount of time.</p> <p>Note: If - is not specified, duration is a positive amount of time.</p>
	P	The start of a duration string. The P must always be specified.
	yY	<p><i>y</i> number of years in the duration.</p> <p><i>Y</i> years are specified in the duration.</p>
	mM	<p><i>m</i> number of months in the duration.</p> <p><i>M</i> months are specified in the duration.</p>
	dD	<p><i>d</i> number of days in the duration.</p> <p><i>D</i> days are specified in the duration.</p>
	T	The start of the time part of the duration string. The T must be specified if any minutes, hours, or seconds (<i>hH</i> , <i>iM</i> , or <i>sS</i>) are specified.
	hH	<p><i>h</i> number of hours in the duration.</p> <p><i>H</i> years are specified in the duration.</p>
	iM	<p><i>i</i> number of minutes in the duration.</p> <p><i>M</i> minutes are specified in the duration.</p>
	sS	<p><i>s</i> number of seconds in the duration.</p> <p>Seconds can be specified up to 7 digits after the decimal place. (Format <i>xx.xxxxxxx</i>)</p> <p><i>S</i> seconds are specified in the duration.</p>

Returns

Returns a duration of time as a duration value.

Examples

duration with All Components

Invoking `duration("P1Y2M4DT9H8M20S")` returns a duration value corresponding to 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds, as shown in the following example query:

```
<result>{xs:duration("P1Y2M4DT9H8M20S")}</result>
```

The preceding query, generates the following XML result:

```
<result>P1Y2M4DT9H8M20S</result>
```

duration with Just Years

Invoking `duration("P9Y")` returns a duration value corresponding to 9 years, as shown in the following example query:

```
<result>{xs:duration("P9Y")}</result>
```

The preceding query, generates the following XML result:

```
<result>P9Y</result>
```

duration with Just Negative Months

Invoking `duration("-P10M")` returns a duration value corresponding to negative 10 months as shown in the following example query:

```
<result>{xs:duration("-P10M")}</result>
```

The preceding query, generates the following XML result:

```
<result>-P10M</result>
```

duration with Just Days and Seconds

Invoking `duration("P4DT20S")` returns a duration value corresponding to 4 days and 20 seconds, as shown in the following example query:

```
<result>{xs:duration("P4DT20S")}</result>
```

The preceding query, generates the following XML result:

```
<result>P4DT20S</result>
```

Related Topics

W3C duration data type description

xs:anyURI

Converts \$string-var that contains a URI (Uniform Resource Identifier Reference) to the anyURI data type.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xs:anyURI(xs:string \$string-var) > xs:anyURI

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the conversion string that contains the URI (Uniform Resource Identifier Reference)

Returns

Returns the \$string-var converted to anyURI data type.

Examples

Simple

When you invoke the following query:

```
<result>{xs:anyURI("http://www.acme.org/")}</result>
```

The preceding query generates the following result:

```
<result>http://www.acme.org/</result>
```

Related Topics

W3C anyURI data type description

xs:Name

Converts \$string-var to the Name data type.

If the value of \$string-var is the empty sequence the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xs:Name(xs:string \$string-var) > xs:Name

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert.

Returns

Returns the \$string-var converted to the Name data type.

Related Topics

[W3C Name data type description](#)

xs:QName

Creates a new QName with a local name specified \$string-var and no namespace.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xs:QName(xs:string \$string-var) > xs:QName

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to convert.

Returns

Returns the \$string-var converted to the QName data type.

Related Topics

W3C QName data type description

XQuery String Functions Reference

This section provides descriptions of the XQuery string functions available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and add invocations to these XQuery functions in these queries.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002 available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the type conversion functions available from the mapper functionality:

- xf:concat
- bea-xf:trim-left
- bea-xf:trim-right
- bea-xf:trim
- xs:normalizedString
- xs:token
- xf:compare
- xf:starts-with
- xf:ends-with
- xf:contains
- xf:substring
- xf:string-length
- xf:substring-before
- xf:substring-after
- xf:normalize-space
- xf:upper-case
- xf:lower-case
- xf:translate
- xf:string-pad
- xf:matches
- xf:replace
- xf:tokenize

xf:concat

Concatenates the string values of the passed in arguments.

Signatures

`xf:concat(xs:string $string-var1, xs:string $string-var2, ...) > xs:string`

Arguments

Data Type	Argument	Description
xs:string	\$string-var1	Represents the first string to concatenate together.
xs:string	\$string-var2	Represents the second string to concatenate together.
...		Represents more strings to concatenate together.

Returns

Returns a string made up of arguments to this function concatenated together.

If the value of any of the string arguments is the empty sequence, the argument is treated as a zero-length string ("").

Examples

This Query ...	Generates This Result ...
<code><r>{xf:concat("str1", "str2")}</r></code>	<code><r>str1str2</r></code>
<code><r>{xf:concat('str1', 'str2')}</r></code>	<code><r>str1str2</r></code>
<code><r>{xf:concat("str1")}</r></code>	<code><r>str1</r></code>
<code><r>{xf:concat()}</r></code>	<code><r></r></code>
<code><r>{xf:concat("str1", "str2", "str3", "str4", "str5", "str6")}</r></code>	<code><r>str1str2str3str4str5str6</r></code>

Related Topics

W3C concat function description.

bea-xf:trim-left

Removes the leading white space from \$string-var.

If the value of \$string-var is the empty sequence, the following error is displayed in the mapper:

Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e

The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

bea-xf:trim-left(xs:string \$string-var) > xs:string

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to trim.

Returns

Returns \$string-var after the removal of the leading white space.

XQuery Compliance

Not a standard W3C XQuery function.

Examples

Remove Leading Spaces

Invoking trim-left(" abc ") returns the string "abc " as shown in the following example query:

```
<result>{bea-xf:trim-left("  abc  ")}</result>
```

The preceding query, generates the following XML result:

```
<result>abc  </result>
```

Error Null

Invoking trim-left(()) outputs an error. The string: () is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

For example, the following example query:

```
<result>{bea-xf:trim-left(())}</result>
```

XQuery Reference

Outputs the following error:

```
Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e
```

bea-xf:trim-right

Removes the trailing white space from \$string-var.

If the value of \$string-var is the empty sequence, the following error is displayed in the mapper:

Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e

The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

bea-xf:trim-right(xs:string \$string-var) > xs:string

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to trim.

Returns

Returns \$string-var after the removal of the trailing white space.

Examples

Remove Trailing Spaces

Invoking trim-right(" abc ") returns the string " abc" as shown in the following example query:

```
<result>{bea-xf:trim-right("  abc  ")}</result>
```

The preceding query, generates the following XML result:

```
<result>  abc</result>
```

Error Null

Invoking trim-right() outputs an error. The string: () is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

For example, the following example query:

```
<result>{bea-xf:trim-right(())}</result>
```

Outputs the following error:

Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e

XQuery Compliance

Not a standard W3C XQuery function.

bea-xf:trim

Removes the leading and trailing white space from \$string-var.

If the value of \$string-var is the empty sequence, the following error is displayed in the mapper:

Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e

The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

bea-xf:trim(xs:string \$string-var) > xs:string

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string to trim.

Returns

Returns \$string-var after the removal of the leading and trailing white space.

Examples

Remove Leading and Trailing Spaces

Invoking trim(" abc ") returns the string "abc" as shown in the following example query:

```
<result>{bea-xf:trim(" abc ")}</result>
```

The preceding query, generates the following XML result:

```
<result>abc</result>
```

Error Null

Invoking trim() outputs an error. The string: () is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

For example, the following example query:

```
<result>{bea-xf:trim()}</result>
```

Outputs the following error:

Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e

XQuery Compliance

Not a standard W3C XQuery function.

xs:normalizedString

Converts \$string-var (a string) to the normalizedString data type. As part of the conversion, all occurrences of tabs (#x9), line feeds (#xA) and carriage returns (#xD) are replaced with spaces (#x20).

Signatures

xs:normalizedString(xs:string \$string-var) > xs:normalizedString

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string for conversion.

Returns

Returns \$string-var after conversion to the normalizedString data type.

Examples

Remove Tabs

As part of the conversion to the normalizedString data type, tabs are replaced by spaces as shown in the following example query:

```
<result>{xf:normalizedString("  tab1    tab2    tab3    tab4")}</result>
```

The preceding query, generates the following XML result:

```
<result> tab1 tab2 tab3 tab4</result>
```

Remove Carriage Returns

As part of the conversion to the normalizedString data type, carriage returns are replaced by spaces as shown in the following example query:

```
<result>{xf:normalizedString("
CR1
CR2
")}</result>
```

The preceding query, generates the following XML result:

```
<result> CR1 CR2 </result>
```

Related Topics

W3C normalizedString data type description.

xs:token

Converts \$string-var (a string) to a token data type.

As part of the conversion, the following steps occur:

1. All occurrences of tabs (#x9), line feeds (#xA) and carriage returns (#xD) are replaced with spaces (#x20).
2. Contiguous sequences of spaces (#x20) are collapsed to a single space (#x20)
3. Leading and trailing spaces (#x20) are removed.

Signatures

xs:token(xs:string \$string-var) > xs:token

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the string for conversion.

Returns

Returns \$string-var after conversion to the token data type.

Examples

Remove Tabs

As part of the conversion to the token data type, tabs are replaced by spaces and then all leading and trailing spaces are removed as shown in the following example query:

```
<result>{xf:token("    tab1    tab2    tab3    tab4")}</result>
```

The preceding query, generates the following XML result:

```
<result>tab1 tab2 tab3 tab4</result>
```

Note: The tab before the string tab1 becomes a space and then is removed. (All trailing and leading spaces are removed as part of the conversion.)

Remove Carriage Returns

As part of the conversion to the token data type, carriage returns are replaced by spaces and then all trailing and leading spaces are removed, as shown in the following example query:

```
<result>{xf:token("
CR1
CR2
")}</result>
```

The preceding query, generates the following XML result:

```
<result>CR1 CR2</result>
```

Note: The carriage returns before the string CR1 and after CR2, become spaces and then are removed. (All trailing and leading spaces are removed as part of the conversion.)

Collapse Spaces

As part of the conversion to the token data type, contiguous sequences of spaces (#x20) are collapsed to a single space (#x20) and leading and trailing spaces (#x20) are removed, as shown in the following example query:

```
<result>{xf:token("  x  y  z  ")}</result>
```

The preceding query, generates the following XML result:

```
<result>x y z</result>
```

Related Topics

W3C token data type description.

xf:compare

Compares the value of \$string-var1 to \$string-var2.

Signatures

xf:compare(xs:string? \$string-var1, xs:string? \$string-var2) > xs:integer?

Arguments

Data Type	Argument	Description
xs:string	\$string-var1	Represents the first comparison string.
xs:string	\$string-var2	Represents the second comparison string.

Returns

Returns -1, 0, or 1, depending on whether the value of \$string-var1 is less than (-1), equal to (0), or greater than (1) the value of \$string-var2.

Examples

Less Than (-1)

Invoking compare('abc', 'abcde') returns an integer -1, because \$string-var1 is less than \$string-var2, as shown by the following example query:

```
<result>{xf:compare('abc', 'abcde')}
```

The preceding query, generates the following XML result:

```
<result>-1</result>
```

Equal (0)

Invoking compare('abc', 'abc') returns an integer 0, because the two string are equal, as shown by the following example query:

```
<result>{xf:compare('abc', 'abc')}
```

The preceding query, generates the following XML result:

```
<result>0</result>
```

Greater Than (+1)

Invoking compare('abcde', 'abc') returns an integer 1, because \$string-var1 is greater than \$string-var2, as shown by the following example query:

```
<result>{xf:compare('abcde', 'abc')}
```

The preceding query, generates the following XML result:

```
<result>1</result>
```

Related Topics

[W3C compare function description.](#)

xf:starts-with

Determines if the \$string-var1 starts with string specified in \$string-var2.

Signatures

xf:starts-with(xs:string? \$string-var1, xs:string? \$string-var2) > xs:boolean?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var1	Represents the string to compare against.
xs:string?	\$string-var2	Compare this string against \$string-var1 to see if \$string-var1 starts with this string.

Returns

Returns the boolean value of true, if \$string-var1 starts with a string that is equal to \$string-var2.

Returns the boolean value of false, if \$string-var1 does not starts with a string that is equal to \$string-var2.

Returns the boolean value true, if \$string-var2 is a zero-length string ("").

Returns the boolean value false, if \$string-var1 is a zero-length string ("") and \$string-var2 is not a zero-length string.

If the value of \$string-var1 or \$string-var2 is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Examples

This Query ...	Generates This Result ...
<r>{xf:starts-with("honeymoon", "honey")}</r>	<r>true</r>
<r>{xf:starts-with("honeymoon", "moon")}</r>	<r>false</r>
<r>{xf:starts-with("honeymoon", "hat")}</r>	<r>false</r>
<r>{xf:starts-with("honeymoon", "")}</r>	<r>true</r>
<r>{xf:starts-with("", "moon")}</r>	<r>false</r>
<r>{xf:starts-with("moon", ())}</r>	<r></r>

Related Topics

W3C starts-with function description.

xf:ends-with

Determines if the \$string-var1 ends with string specified in \$string-var2.

Signatures

xf:ends-with(xs:string? \$string-var1, xs:string? \$string-var2) > xs:boolean?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var1	Represents the string to compare against.
xs:string?	\$string-var2	Compare this string against \$string-var1 to see if \$string-var1 ends in this string.

Returns

Returns the boolean value of true, if \$string-var1 ends with a string that is equal to \$string-var2.

Returns the boolean value of false, if \$string-var1 does not end with a string that is equal to \$string-var2.

Returns the boolean value true, if \$string-var2 is a zero-length string ("").

Returns the boolean value false, if \$string-var1 is a zero-length string ("") and \$string-var2 is not a zero-length string.

If the value of \$string-var1 or \$string-var2 is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Examples

This Query . . .	Generates This Result . . .
<r>{xf:ends-with("honey-moon", "moon")}</r>	<r>true</r>
<r>{xf:ends-with("honey-moon", "honey")}</r>	<r>>false</r>
<r>{xf:ends-with("honey-moon", "hat")}</r>	<r>>false</r>
<r>{xf:ends-with("honey-moon", "")}</r>	<r>true</r>
<r>{xf:ends-with("", "moon")}</r>	<r>>false</r>
<r>{xf:ends-with("moon", ())}</r>	<r/>

Related Topics

W3C ends-with function description.

xf:contains

Determines if \$string-var1 contains the string specified in \$string-var2.

If the value of \$string-var1 or \$string-var2 is an empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:contains(xs:string? \$string-var1, xs:string? \$string-var2) > xs:boolean?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var1	Represents the string to compare against.
xs:string?	\$string-var2	Compare this string against \$string-var1 to see if this string is contained anywhere in \$string-var1.

Returns

Returns the boolean value of true, if \$string-var1 contains \$string-var2.

Returns the boolean value of false, if \$string-var1 does not contain \$string-var2.

Returns the boolean value true, if \$string-var2 is a zero-length string ("").

Returns the boolean value false, if \$string-var1 is a zero-length string ("") and \$string-var2 is not a zero-length string.

Examples

This Query . . .	Generates This Result . . .
<r>{xf:contains("supercalifragilistic", "fragil")}</r>	<r>true</r>
<r>{xf:contains("supercalifragilistic", "honey")}</r>	<r>false</r>
<r>{xf:contains("supercalifragilistic", "")}</r>	<r>true</r>
<r>{xf:contains("", "honey")}</r>	<r>false</r>
<r>{xf:contains("honey", ())}</r>	<r/>

Related Topics

W3C contains function description.

xf:substring

Get a substring of \$string-var at a particular index location.

If a positive integer is not passed into \$decimal-var or \$optional-decimal-var, the TransformException exception is raised with the RT_ILLEGAL_INDEX fault code. In the mapper the following error message is displayed:

```
Error occurred while executing XQuery: Index "-1" out of bounds (0, 5)
```

If the integer passed into \$decimal-var is greater than the length of the \$string-var, the TransformException exception is raised with the RT_ILLEGAL_INDEX fault code. In the mapper the following error message is displayed:

```
Error occurred while executing XQuery: Index "6" out of bounds (0, 5)
```

Where X is the out of bounds index, Y is the starting index, and Z is the ending index, for example Index "6" out of bounds (0, 5).

To learn more about using fault codes, see [Getting the TransformException Fault Code Programmatically](#).

Signatures

xf:substring(xs:string? \$string-var, xs:decimal? \$decimal-var) > xs:string?

xf:substring(xs:string? \$string-var, xs:decimal? \$decimal-var, xs:decimal? \$decimal-var) > xs:string?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var	Represents the source string.
xs:decimal?	\$decimal-var	Represents the starting location to start extracting the substring to return. Note: Use 1 to specify the first character of the string and not 0.
xs:decimal?	\$optional-decimal-var	Optional Represents the length of the extracted substring.

Returns

substring(\$string-var, \$decimal-var)

Returns the part of the \$string-var source string from the starting location specified by \$decimal-var.

If the value of any of the two arguments is an empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

substring(\$string-var, \$decimal-var, \$optional-decimal-var)

Returns the part of the \$string-var source string from the starting location specified by \$decimal-var and continuing for the number of characters specified by \$optional-decimal-var.

If the value of any of the three arguments is an empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Examples**Specify Just the Starting Position**

Invoking substring("supercalifragilistic", 15) returns the string: listic, as shown in the following example query:

```
<result>{xf:substring("supercalifragilistic", 15)}</result>
```

The preceding query generates the following result:

```
<result>listic</result>
```

Specify the Starting Position and Length

Invoking substring("supercalifragilistic", 1, 5) returns the string: super, as shown in the following example query:

```
<result>{xf:substring("supercalifragilistic", 1, 5)}</result>
```

The preceding query generates the following result:

```
<result>super</result>
```

Note: To specify the first character in a string, use 1 and not 0.

From the Middle of the String

Invoking substring("supercalifragilistic", 6, 4) returns the string: cali, as shown in the following example query:

```
<result>{xf:substring("supercalifragilistic", 6, 4)}</result>
```

The preceding query generates the following result:

```
<result>cali</result>
```

Pass in Null

Invoking substring((), 1) returns the null string as shown in the following example query:

```
<result>{xf:substring((), 1)}</result>
```

Note: The string: () is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

The preceding query generates the following result:

```
<result/>
```

Error Out of Bounds

Invoking `substring("super", 6)` outputs an error because the index specified (6) is longer than the string: `super`.

For example, the following example query:

```
<result>{xf:substring("super", 6)}</result>
```

Produces the following error:

```
Error occurred while executing XQuery: Index "6" out of bounds (0, 5)
```

Note: Indexing starts with the number 1 and not 0, so the index 6 is beyond the last character in the string: `super`. (The character `r` in the string: `super` is at index 5.)

Related Topics

W3C `substring` function description.

xf:string-length

Counts the length of \$string-var.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:string-length(xs:string? \$string-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var	Represents the string to count.

Returns

Returns an integer equal to the length of the \$string-var.

Examples

Simple

Invoking string-length("moo cow") returns the integer 7, as shown in the following example query:

```
<result>{xf:string-length("moo cow")}</result>
```

The preceding query generates the following result:

```
<result>7</result>
```

Pass in Null

Invoking string-length(()) returns the null string as shown in the following example query:

```
<result>{xf:string-length(())}</result>
```

Note: The string: () is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

The preceding query generates the following result:

```
<result/>
```

Related Topics

W3C string-length function description.

xf:substring-before

Finds the substring that precedes \$string-var2 in \$string-var2.

Signatures

xf:substring-before(xs:string? \$string-var1, xs:string? \$string-var2) > xs:string?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var1	Represents the source string.
xs:string?	\$string-var2	Represents the comparison string.

Returns

Returns the part of the \$string-var1 source string that precedes \$string-var2.

Returns the value of \$string-var1, if \$string-var2 is a zero-length string ("").

Returns a zero-length string (""), if \$string-var1 does not contain \$string-var2.

If the value of \$string-var1 or \$string-var2 is an empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Examples

This Query . . .	Generates This Result . . .
<r>{xf:substring-before("super", "p")}</r>	<r>su</r>
<r>{xf:substring-before("super", "")}</r>	<r>super</r>
<r>{xf:substring-before("super", "z")}</r>	<r/>
<r>{xf:substring-before("super", ())}</r>	<r/>

Related Topics

W3C substring-before function description.

xf:substring-after

Finds the substring that follows \$string-var2 in \$string-var2.

Signatures

xf:substring-after(xs:string? \$string-var1, xs:string? \$string-var2) > xs:string?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var1	Represents the source string.
xs:string?	\$string-var2	Represents the comparison string.

Returns

Returns the part of the \$string-var1 source string that follows \$string-var2.

Returns the value of \$string-var1, if \$string-var2 is a zero-length string ("").

Returns a zero-length string (""), if \$string-var1 does not contain \$string-var2.

If the value of \$string-var1 or \$string-var2 is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Examples

This Query . . .	Generates This Result . . .
<r>{xf:substring-after("super", "p")}</r>	<r>er</r>
<r>{xf:substring-after("super", "")}</r>	<r>super</r>
<r>{xf:substring-after("super", "z")}</r>	<r/>
<r>{xf:substring-after("super", ())}</r>	<r/>

Related Topics

W3C substring-after function description.

xf:normalize-space

Removes the leading and trailing white space and replaces the duplicate white space characters by a single space from \$string-var.

As part of the removal process, the following steps occur:

1. All occurrences of tabs (#x9), line feeds (#xA) and carriage returns (#xD) are replaced with spaces (#x20).
2. Contiguous sequences of spaces (#x20) are collapsed to a single space (#x20)
3. Leading and trailing spaces (#x20) are removed.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:normalize-space(xs:string? \$string-var) > xs:string?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var	Represents the string for conversion.

Returns

Returns the \$string-var after it has gone through the white space removal process (described above).

Examples

Collapse Spaces

As part of the conversion process, contiguous sequences of spaces (#x20) are collapsed to a single space (#x20) and leading and trailing spaces (#x20) are removed, as shown in the following example query:

```
<result>{xf:normalize-space("  x  y  z  ")}</result>
```

The preceding query, generates the following XML result:

```
<result>x y z</result>
```

Pass in Null

Invoking normalize-space() returns the null string as shown in the following example query:

```
<result>{xf:normalize-space()}</result>
```

Note: The string: () is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

The preceding query generates the following result:

```
<result/>
```

Related Topics

W3C `normalize-space` function description.

xf:upper-case

Converts all the lowercase characters of \$string-var to their uppercase form.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:upper-case(xs:string? \$string-var) > xs:string?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var	Represents the string to convert.

Returns

Returns the \$string-var source string converted to uppercase characters.

Examples

Simple

Invoking upper-case("cat") returns the string CAT as shown in the following example query:

```
<result>{xf:upper-case("cat")}</result>
```

The preceding query generates the following result:

```
<result>CAT</result>
```

Pass in Null

Invoking upper-case(()) returns the null string as shown in the following example query:

```
<result>{xf:upper-case(())}</result>
```

Note: The string: () is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

The preceding query generates the following result:

```
<result/>
```

Related Topics

W3C upper-case function description.

xf:lower-case

Converts all the lowercase characters of \$string-var (a string) to their lowercase form.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:lower-case(xs:string? \$string-var) > xs:string?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var	Represents the string to convert.

Returns

Returns the \$string-var source string converted to lowercase characters.

Examples

Simple

Invoking lower-case("CAT") returns the string cat as shown in the following example query:

```
<result>{xf:lower-case("CAT")}</result>
```

The preceding query generates the following result:

```
<result>cat</result>
```

Pass in Null

Invoking lower-case(()) returns the null string as shown in the following example query:

```
<result>{xf:lower-case(())}</result>
```

Note: The string: () is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

The preceding query generates the following result:

```
<result/>
```

Related Topics

W3C lower-case function description.

xf:translate

Replaces all occurrences of \$string-var2 with \$string-var3 in \$string-var1.

If the value of \$string-var1, \$string-var2, or \$string-var3 is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:translate(xs:string? \$string-var1, xs:string? \$string-var2, xs:string? \$string-var3) > xs:string?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var1	Represents the source string.
xs:string?	\$string-var2	Represents the search string.
xs:string?	\$string-var3	Represents the replacement string.

Returns

Returns the value of \$string-var1 after all occurrences of the \$string-var2 in \$string-var1 have been replaced with \$string-var3. If the length of \$string-var2 (the search string) is less than the length of \$string-var3 (replacement string), then only *N* characters are substituted, where *N* is length of \$string-var2. (See the third example below.)

Returns the value of \$string-var1, if \$string-var2 is a zero-length string ("").

Examples

Simple

Invoking translate("fghXfgh", "fgh", "yz") returns the string xyXyz as shown in the following example query:

```
<result>{xf:translate("fghXfgh", "fgh", "yz")}</result>
```

The preceding query generates the following result:

```
<result>yzXyz</result>
```

Replacement String Longer Than Source String

Invoking translate("abcdabc", "ab", "ABC") returns the string ABcdABc, as shown in the following example query:

```
<result>{xf:translate("abcdabc", "ab", "ABC")}</result>
```

The preceding query generates the following result:

```
<result>ABcdABc</result>
```

Note: Only the first two characters (AB) of the replacement string ABC are substituted because the length of ab (the search string) is less than the length of ABC (the replacement string). When the length of the search string is less than the length of the replacement string, the length of the search string determines the number characters substituted.

Replacement String Is Empty

Invoking `translate("abcdabc", "abc", "")` returns the string d because the replacement string is a zero-length string, as shown in the following example query:

```
<result>{xf:translate("abcdabc", "abc", "")}</result>
```

The preceding query generates the following result:

```
<result>d</result>
```

Search String Is Empty

Invoking `translate("abcdabc", "", "AB")` returns the string abcdabc because the search string is a zero-length string, as shown in the following example query:

```
<result>{xf:translate("abcdabc", "", "AB")}</result>
```

The preceding query generates the following result:

```
<result>abcdabc</result>
```

Pass in Null

Invoking `translate(), "ab", "ABC")` returns the null string as shown in the following example query:

```
<result>{xf:translate((), "ab", "ABC")}</result>
```

Note: The string: () is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

The preceding query generates the following result:

```
<result/>
```

Related Topics

W3C `translate` function description.

xf:string-pad

Returns a string of made up of \$decimal-var copies of \$string-var concatenated together.

Signatures

xf:string-pad(xs:string? \$string-var, xs:decimal? \$decimal-var) > xs:string?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var	Represents the string to replicate.
xs:decimal?	\$decimal-var	Represents the number of times to replicate \$string-var.

Returns

Returns a string of made up of \$decimal-var copies of \$string-var concatenated together.

Returns the value of \$string-var1, if \$string-var2 is a zero-length string ("").

Returns a zero-length string (""), if \$decimal-var is equal to 0.

If the value of \$string-var or \$decimal-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Examples

This Query . . .	Generates This Result . . .
<r>{xf:string-pad("cat", 2)}</r>	<r>catcat</r>
<r>{xf:string-pad("super", 0)}</r>	<r/>
<r>{xf:string-pad("super", ())}</r>	<r/>

Related Topics

W3C string-pad function description.

xf:matches

Compares \$string-var1 against the regular expression in string-var2.

If any character besides m or i is specified in \$string-var3 (the flags string), the TransformException exception is raised with the RT_REGEXP_FLAGS fault code. In the mapper, the following error is displayed:

```
Error occurred while executing XQuery: Invalid regular expression syntax (flags: "a")
```

If the value of \$string-var1, \$string-var2, or \$string-var3 is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

To learn more about using fault codes, see [Getting the TransformException Fault Code Programmatically](#).

If \$string-var2 (the regular expression string) has no anchors and is found anywhere in \$string-var1 (the source string), the regular expression string is considered to match and the boolean true is returned. (To learn more, see the following [String Matches Example](#).) However, if anchors (^ \$) are used in \$string-var2 (the regular expression string), the anchors must match the start/end of either the string (for string mode) or line (for multiline mode). (To learn more, see the following [Beginning Anchor Matches](#) and [Beginning Anchor Does Not Match Examples](#).)

Signatures

xf:matches(xs:string? \$string-var1, xs:string? \$string-var2) > xs:boolean?

xf:matches(xs:string? \$string-var1, xs:string? \$string-var2, xs:string? \$string-var3) > xs:boolean?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var1	Represents the source string.
xs:string?	\$string-var2	Represents the regular expression string. To learn more, see W3C Regular Expressions .
xs:string?	\$string-var3	Specifies flags that affect the comparison to the regular expression. (Optional)
	m	If specified the match operates in multiline mode. Otherwise, the match operates in string mode.
	i	If specified the match operates in case-insensitive mode. By default, the match operates in case-sensitive mode.

Returns

Returns the boolean true if \$string-var1 matches the regular expression supplied in \$string-var2.

Returns the boolean false if \$string-var1 *does not* matches the regular expression supplied in \$string-var2.

Examples

Regular Expression Matches

Invoking `matches("abc", "[cxy]")` returns the boolean `true` because the character `c` of the regular expression `[cxy]` is found in the source string `abc`, as shown in the following example query:

```
<result>{xf:matches("abc", "[cxy])}</result>
```

The preceding query generates the following result:

```
<result>true</result>
```

Regular Expression Does Not Match

Invoking `matches("abc", "[xy]")` returns the boolean `false` because none of the characters in the regular expression `[xy]` are in the source string `abc`, as shown in the following example query:

```
<result>{xf:matches("abc", "[xy])}</result>
```

The preceding query generates the following result:

```
<result>>false</result>
```

String Matches

Invoking `matches("uvwxyz", "xyz")` returns the boolean `true` because the regular expression string `xyz` is found in the source string `uvwxyz`, as shown in the following example query:

```
<result>{xf:matches("uvwxyz", "xyz")}</result>
```

The preceding query generates the following result:

```
<result>true</result>
```

Beginning Anchor Matches

Invoking `matches("uvwxyz", "^uv")` returns the boolean `true` because the source string `uvwxyz` does start with the string `uv`, as shown in the following example query:

```
<result>{xf:matches("uvwxyz", "^uv")}</result>
```

The preceding query generates the following result:

```
<result>true</result>
```

Note: The regular expression `^uv` specifies that source string must start with the string `uv` to be a successful match.

Beginning Anchor Does Not Match

Invoking `matches("uvwxyz", "^yz")` returns the boolean `false` because the source string `uvwxyz` does not start with the string `yz`, as shown in the following example query:

```
<result>{xf:matches("uvwxyz", "^yz")}</result>
```

The preceding query generates the following result:

```
<result>false</result>
```

Note: The regular expression `^yz` specifies that source string must start with the string `yz` to be a successful match.

Ending Anchor Matches

Invoking `matches("uvwxyz", "yz$")` returns the boolean `true` because the source string `uvwxyz` does end with the string `yz`, as shown in the following example query:

```
<result>{xf:matches("uvwxyz", "yz$")}</result>
```

The preceding query generates the following result:

```
<result>true</result>
```

Note: The regular expression `yz$` specifies that source string must end with the string `yz` to be a successful match.

Ending Anchor Does Not Match

Invoking `matches("uvwxyz", "vw$")` returns the boolean `false` because the source string `uvwxyz` does not end with the string `yz`, as shown in the following example query:

```
<result>{xf:matches("uvwxyz", "vw$")}</result>
```

The preceding query generates the following result:

```
<result>false</result>
```

Note: The regular expression `yz$` specifies that source string must end with the string `yz` to be a successful match.

Case-Insensitive Mode

Invoking `matches("aBc", "abc", "i")` returns the boolean `true` because the source string `aBc` does contain the string `abc`, if you ignore the case of the strings, as shown in the following example query:

```
<result>{xf:matches("aBc", "abc", "i")}</result>
```

The preceding query generates the following result:

```
<result>true</result>
```

Pass in Null

Invoking `matches("abc", ())` returns the null string as shown in the following example query:

```
<result>{xf:matches("abc", ())}</result>
```

Note: The string: `()` is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

The preceding query generates the following result:

```
<result/>
```

Related Topics

[W3C matches function description.](#)

[W3C Regular Expressions description.](#)

xf:replace

Substitutes all occurrences of the regular expression specified by \$string-var2 in \$string-var1 with \$string-var3.

If any character besides m or i is specified in \$string-var4 (the flags string), the TransformException exception is raised with the RT_REGEXP_FLAGS fault code. In the mapper the following error message is displayed:

```
Error occurred while executing XQuery: Invalid regular expression syntax (flags: "a")
```

To learn more about using fault codes, see [Getting the TransformException Fault Code Programmatically](#).

If the value of \$string-var1, \$string-var2, \$string-var3, or \$string-var4 is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:replace(xs:string? \$string-var1, xs:string? \$string-var2 xs:string? \$string-var3) > xs:string?

xf:replace(xs:string? \$string-var1, xs:string? \$string-var2, xs:string? \$string-var3, xs:string? \$string-var4) > xs:string?

Arguments

Data Type	Argument	Description
xs:string?	\$string-var1	Represents the source string.
xs:string?	\$string-var2	Represents the regular expression string. To learn more, see W3C Regular Expressions .
xs:string?	\$string-var3	Represents the replacement string.
xs:string?	\$string-var4	Specifies flags that affect the comparison to the regular expression. (Optional)
		m If specified the match operates in multiline mode. By default, the match operates in string mode.
		i If specified the match operates in case-insensitive mode. By default, the match operates in case-sensitive mode.

Returns

Returns the string after substitution has occurred. (All occurrences of the regular expression specified by \$string-var2 in \$string-var1 are replaced with \$string-var3.)

Examples

Simple

Invoking `replace("xyzaxyz", "yz", "o")` returns the string `xoaxo`, as shown in the following example query:

```
<result>{xf:replace("xyzaxyz", "yz", "o")}</result>
```

The preceding query generates the following result:

```
<result>xoaxo</result>
```

Greedy Qualifiers

Invoking `replace("xyzaxyz", "x.*z", "o")` returns the string `o` because the regular expression `x.*z` behaves as a greedy qualifier the entire source string is read in before the first match is attempted and only if the first match attempt (with the entire source string) fails, does the matcher reduce the size of the source string by one character and attempts to match again. This process is repeated until there are no more characters in the source string. In this case, the regular expression `x.*z` matches the entire source string `xyzaxyz`, so the entire source string is replaced by the string `o`, shown in the following example query:

```
<result>{xf:replace("xyzaxyz", "x.*z", "o")}</result>
```

The preceding query generates the following result:

```
<result>o</result>
```

Reluctant Qualifiers

Invoking `replace("xyzaxyz", "x.*?z", "o")` returns the string `oao` because the regular expression `x.*?z` behaves as a reluctant qualifier matching starts at the beginning of the input string, reluctantly reading in characters one at a time, until a match is found. Once a match is found, replacement occurs and the rest of the input string is searched for more matches. In this case, the regular expression `x.*?z` finds two matches in the source string `xyzaxyz`, so the two `xyz` strings in the source string are replaced with the string `o`, shown in the following example query:

```
<result>{xf:replace("xyzaxyz", "x.*?z", "o")}</result>
```

The preceding query generates the following result:

```
<result>oao</result>
```

Replacement String Is Empty

Invoking `replace("xyzaxyz", "x", "")` returns the string `yzayz` because the replacement string is a zero-length string, as shown in the following example query:

```
<result>{xf:replace("xyzaxyz", "x", "")}</result>
```

The preceding query generates the following result:

```
<result>yzayz</result>
```

Case-Insensitive Mode

Invoking `replace("Xax", "x", "o", "i")` returns the string `oao`. The `i` flag in `$string-var4` specifies to ignore the case during matching, so in this case two replacements occur (`X` and `x`), as shown in the following example query:

```
<result>{xf:replace("Xax", "x", "o", "i")}</result>
```

The preceding query generates the following result:

```
<result>oao</result>
```

Pass in Null

Invoking `replace("xyz", "xyz",())` returns the null string as shown in the following example query:

```
<result>{xf:replace("xyz", "xyz",())}</result>
```

Note: The string: `()` is the empty sequence (similar to a SQL null) which is a sequence containing zero items.

The preceding query generates the following result:

```
<result/>
```

Related Topics

W3C `replace` function description.

W3C Regular Expressions description.

xf:tokenize

Breaks up \$string-var1 into substrings based on the regular expression delimiter specified in \$string-var2.

If the value of \$string-var1, \$string-var2, or \$string-var3 is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If any character besides m or i is specified in \$string-var3 (the flags string), the TransformException exception is raised with the RT_REGEXP_FLAGS fault code. In the mapper the following error message is displayed:

```
Error occurred while executing XQuery: Invalid regular expression syntax (flags: "a")
```

To learn more about using fault codes, see [Getting the TransformException Fault Code Programmatically](#).

Signatures

xf:tokenize(xs:string? \$string-var1, xs:string? \$string-var2) > xs:string*

xf:tokenize(xs:string? \$string-var1, xs:string? \$string-var2 xs:string? \$string-var3) > xs:string*

Arguments

Data Type	Argument	Description
xs:string?	\$string-var1	Represents the source string.
xs:string?	\$string-var2	Represents the regular expression which determines how to break up the source string. To learn more, see W3C Regular Expressions .
xs:string?	\$string-var3	Specifies flags that affect how the regular expression is interpreted. (Optional)
	m	If specified the match operates in multiline mode. By default, the match operates in string mode.
	i	If specified the match operates in case-insensitive mode. By default, the match operates in case-sensitive mode.

Returns

Returns the strings after break up of \$string-var1 based on the pattern specified in \$string-var2 has occurred.

Examples

White Space Delimited List

Invoking tokenize("Jane fell down the hill", "\s") returns the following sequence of strings: ("Jane", "fell", "down", "the", "hill"). The regular expression \s specifies that the delimiter is white space, so in this case the source string is broken up by white space as shown in the following example query:

```
<1>{  
for $tok in xf:tokenize("Jane fell down the hill", "\s")
```

xf:tokenize

```

    return <i>{ $tok }</i>
  }</l>

```

The preceding query generates the following result:

```

<l>
  <i>Jane</i>
  <i>fell</i>
  <i>down</i>
  <i>the</i>
  <i>hill</i>
</l>

```

Comma Delimited List

Invoking `tokenize("3,20,,27,60", ",")` returns the following strings: ("3", "20", "", "27", "60"). In this case, the delimiter is a comma, so the source string is broken up by commas as shown in the following example query:

```

<l>{
for $tok in xf:tokenize("3,20,,27,60", ",")
  return <i>{ $tok }</i>
}</l>

```

The preceding query generates the following result:

```

<l>
  <i>3</i>
  <i>20</i>
  <i></i>
  <i>27</i>
  <i>60</i>
</l>

```

Comma and White Space Delimited List

Invoking `tokenize("3, 4, 27,67", ",\s")` returns the following strings: ("3", "4", "27,67"). The regular expression, `\s` specifies that the delimiter is a comma with white space, so in this case the source string is broken up by a comma with white space as shown in the following example query:

```

<l>{
for $tok in xf:tokenize("3, 4, 27,67", ",\s")
  return <i>{ $tok }</i>
}</l>

```

The preceding query generates the following result:

```

<l>
  <i>3</i>
  <i>4</i>
  <i>27,67</i>
</l>

```

Note: The numbers 27 and 67 are not broken up as tokens because there is no white space between the 27 and the 67 in the source string (just a comma).

Case-Insensitive Mode

Invoking `tokenize("1a2A3a4A5", "a", "i")` returns the following strings: ("1","2","3","4","5"). The `i` flag in `$string-var3` specifies to ignore the case of the characters during matching, so in this case the source string is broken up by both the capital A and the lowercase a characters, as shown in the following example query:

```
<1>{
for $tok in xf:tokenize("1a2A3a4A5", "a", "i")
  return <i>{ $tok }</i>
}</1>
```

The preceding query generates the following result:

```
<1>
  <i>1</i>
  <i>2</i>
  <i>3</i>
  <i>4</i>
  <i>5</i>
</1>
```

Related Topics

W3C `tokenize` function description.

W3C Regular Expressions description.

XQuery Numeric Function Reference

This section provides descriptions of the XQuery numeric functions available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery functions. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002 available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the numeric functions available from the Mapper tool:

- bea-xf:format-number
- xf:floor
- xf:ceiling
- xf:round

bea-xf:format-number

Converts \$double-var to a string using the format pattern specified by \$string-var.

If the value of \$double-var is the empty sequence, the following error is displayed in the mapper:

Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e

The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

bea-xf:format-number(xs:double \$double-var, xs:string \$string-var) > xs:string

Arguments

Data Type	Argument	Description
xs:double	\$double-var	Represents the double number to be converted to a string.
xs:string	\$string-var	Represents the pattern string. The format of this pattern is specified by the JDK 1.4.1 DecimalFormat class.

Returns

Returns \$double-var as a string based on the pattern specified by \$string-var.

Examples

Add Comma And More Decimal Places

Invoking format-number(xs:double(10002.45), "#,###0.000#") returns the string 1,0002.450 as shown in the following example query:

```
<result>{bea-xf:format-number(xs:double(10002.45), "#,###0.000#")}</result>
```

The preceding query generates the following result:

```
<result>1,0002.450</result>
```

XQuery Compliance

Not a standard W3C XQuery function. This is a standard XSLT function.

Note: Only the two argument version of the standard XSLT function is supported.

Related Topics

W3C XSLT format-number function description.

JDK 1.4.1 DecimalFormat class description.

xf:floor

Rounds \$double-var down to the *next* whole number.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:floor(xs:double? \$double-var) > xs:double?

Arguments

Data Type	Argument	Description
xs:double?	\$double-var	Represents the double to round down.

Returns

Returns a double equal to \$double-var rounded down to the *next* whole number.

Examples

This Query . . .	Generates This Result . . .
<r>{ xf:floor(xs:double("11.9")) }</r>	<r>11.0</r>
<r>{ xf:floor(xs:double("11.5")) }</r>	<r>11.0</r>
<r>{ xf:floor(xs:double("11.1")) }</r>	<r>11.0</r>
<r>{ xf:floor(xs:double("-11.9")) }</r>	<r>-12.0</r>
<r>{ xf:floor(xs:double("-11.5")) }</r>	<r>-12.0</r>
<r>{ xf:floor(xs:double("-11.1")) }</r>	<r>-12.0</r>

Related Topics

W3C floor function description.

xf:ceiling

Rounds \$double-var up to the *next* whole number.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:ceiling(xs:double? \$double-var) > xs:double?

Arguments

Data Type	Argument	Description
xs:double?	\$double-var	Represents the double to round down.

Returns

Returns a double equal to \$double-var rounded up to the *next* whole number.

Examples

This Query ...	Generates This Result ...
<r>{xf:ceiling(xs:double("11.9"))}</r>	<r>12.0</r>
<r>{xf:ceiling(xs:double("11.5"))}</r>	<r>12.0</r>
<r>{xf:ceiling(xs:double("11.1"))}</r>	<r>12.0</r>
<r>{xf:ceiling(xs:double("-11.9"))}</r>	<r>-11.0</r>
<r>{xf:ceiling(xs:double("-11.5"))}</r>	<r>-11.0</r>
<r>{xf:ceiling(xs:double("-11.1"))}</r>	<r>-11.0</r>

Related Topics

W3C ceiling function description.

xf:round

Rounds \$double-var to the *nearest* whole number.

If the value of \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:round(xs:double? \$double-var) > xs:double?

Arguments

Data Type	Argument	Description
xs:double?	\$double-var	Represents the double to round.

Returns

Returns a double equal to \$double-var rounded to the *nearest* whole number.

Examples

This Query . . .	Generates This Result . . .
<r>{ xf:round(xs:double("11.9")) }</r>	<r>12.0</r>
<r>{ xf:round(xs:double("11.5")) }</r>	<r>12.0</r>
<r>{ xf:round(xs:double("11.1")) }</r>	<r>11.0</r>
<r>{ xf:round(xs:double("-11.9")) }</r>	<r>-12.0</r>
<r>{ xf:round(xs:double("-11.5")) }</r>	<r>-11.0</r>
<r>{ xf:round(xs:double("-11.1")) }</r>	<r>-11.0</r>

Related Topics

W3C round function description.

XQuery URI Functions Reference

This section provides descriptions of the XQuery URI functions available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery functions. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002 available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the URI functions available from the mapper functionality:

- xf:escape-URI
- xf:resolve-URI

xf:escape-URI

Applies the URI escaping rules to \$string-var that contains a URI (Uniform Resource Identifier Reference).

If the value of \$string-var or \$boolean-var is the empty sequence, the following error is displayed in the mapper:

```
Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e
```

The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:escape-URI(xs:string \$string-var, xs:boolean \$boolean-var) > xs:string

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents the conversion string that contains the URI (Uniform Resource Identifier Reference)
xs:boolean	\$boolean-var	Represents the boolean that determines the URI escaping rules to apply.
	true	<p>All characters are escaped except the following:</p> <ul style="list-style-type: none">• Lower case letters a–z• Upper case letters A–Z• Digits 0–9• The following characters: "-" "_" "." "!" "~" "*" "'" "(" ")"". (These characters are referred to as <i>marks</i> in RFC 2396.) <p>The "%" character is escaped only if it is not followed by two hexadecimal digits (0–9, a–f, or A–F).</p> <p>Typically this option is used to escape part of a URI.</p>
	false	<p>In addition to the characters defined in the <i>true</i> section of this table as not being escaped, the following reserved characters are not escaped: ";" "/" "?" ":" "@" "&" "=" "+" "\$" ",". (As defined by RFC 2396 .)</p> <p>Typically, this option is used to escape an entire URI or URI reference.</p>

Returns

Returns the \$string-var after the URI escaping rules have been applied. The rules applied are different depending on the value of \$boolean-var.

Examples

True

The following example query shows the `escape-URI` function called with the `$boolean-var` argument set to `true`:

```
<true>{xf:escape-URI("http://www.acme.org/", xs:boolean("true"))}</true>
```

The preceding query generates the following result:

```
<true>http%3A%2F%2Fwww.acme.org%2F</true>
```

False

The following example query shows the `escape-URI` function called with the `$boolean-var` argument set to `false`:

```
<false>{xf:escape-URI("http://www.acme.org/widget/201/#specs",xs:boolean("false"))}</false>
```

The preceding query generates the following result:

```
<false>http://www.acme.org/widget/201/%23specs</false>
```

Related Topics

W3C `escape-uri` function description.

xf:resolve-URI

Resolves the relative URI \$anyURI-var1 against the base URI \$anyURI-var2. To learn about URI resolution, see the javadoc for the URI class.

If the value of \$anyURI-var1 or \$anyURI-var2 is the empty sequence, the following error is displayed in the mapper:

Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e

The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:resolve-URI(xs:anyURI \$anyURI-var1, xs:anyURI \$anyURI-var2) > xs:anyURI

Arguments

Data Type	Argument	Description
xs:anyURI	\$anyURI-var1	Represents the base URI.
xs:anyURI	\$anyURI-var2	Represents the relative URI.

Returns

Returns the absolute URI (of data type anyURI) which is result of the relative URI \$anyURI-var1 being resolved against the base URI \$anyURI-var2.

Returns the relative URI \$anyURI-var2, if \$anyURI-var2 is an absolute URI.

Example

Resolve-URI Example 1

The first example of an query invoking the resolve-URI function:

```
<resolve-URI-1>
{xs:string(xf:resolve-URI("http://www.ics.uci.edu/pub/ietf/uri/#Related", "priv#internal"))}
</resolve-URI-1>
```

The preceding query generates the following result:

```
<resolve-URI-1>http://www.ics.uci.edu/pub/ietf/uri/priv#internal</resolve-URI-1>
```

Resolve-URI Example 2

The second example of an query calling the resolve-URI function:

```
<resolve-URI-2>
{xs:string(xf:resolve-URI("http://www.ics.uci.edu/pub/", "../priv/../../priv#internal"))}
```

```
</resolve-URI-2>
```

The preceding query generates the following result:

```
<resolve-URI-2>http://www.ics.uci.edu/priv#internal</resolve-URI-2>
```

Resolve-URI Example 3

The third example of an query calling the resolve-URI function:

```
<resolve-URI-3>
  {xs:string(xf:resolve-URI("http://www.ics.uci.edu/", "priv"))}
</resolve-URI-3>
```

The preceding query generates the following result:

```
<resolve-URI-3>http://www.ics.uci.edu/priv</resolve-URI-3>
```

Related Topics

W3C resolve-URI function description.

JDK 1.4.1 URI class description.

XQuery Aggregate Function Reference

This section provides descriptions of the XQuery aggregate functions available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery functions. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the *XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002* available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the aggregate functions available from the Mapper tool:

- xf:count
- xf:avg
- xf:max
- xf:min
- xf:sum

xf:count

Counts the number items in the sequence.

Signatures

xf:count(item* \$item-var) > xs:unsignedInt

Arguments

Data Type	Argument	Description
item*	\$item-var	Represents the sequence to be counted.

Returns

Returns the number of items in a sequence passed into \$item-var as an unsignedInt value.

Returns 0 if \$item-var is the empty sequence. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Examples

Simple

Invoking count(("a","b","c")) returns the unsigned integer value of 3, as shown in the following example query:

```
<result>{xf:count(("a","b","c"))}</result>
```

The preceding query generates the following result:

```
<result>3</result>
```

Empty Sequence

Invoking count(()) returns the unsigned integer value of 0 because () is an empty sequence (contains no elements), as shown in the following example query:

```
<result>{xf:count(())}</result>
```

The preceding query generates the following result:

```
<result>0</result>
```

Related Topics

W3C count function description.

xf:avg

Determines the average of all the numbers in a sequence.

If the value of \$item-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:avg(item* \$item-var) > xs:double?

Arguments

Data Type	Argument	Description
item*	\$item-var	Represents the sequence of numbers to add together.

Returns

Returns the average of all the numbers in a sequence passed into \$item-var.

Examples

Simple

Invoking xf:avg(("3","1","2")) returns the value of 2.0, as shown in the following example query:

```
<result>{xf:avg(("3","1","2"))}</result>
```

The preceding query generates the following result:

```
<result>2.0</result>
```

Empty Sequences

Invoking xf:avg((1,(),6,2,9)) returns the value of 4.5, as shown in the following example query:

```
<result>{xf:avg((1,(),6,2,9))}</result>
```

The preceding query generates the following result:

```
<result>4.5</result>
```

Note: Instances of the empty sequence () are ignored.

Nodes

Invoking the following query returns the value of 50, as shown in the following example query:

```
let $x := <a>100</a>
```

xf:avg

XQuery Reference

```
let $y := <b>50</b>
let $z := <c>0</c>
return <result>{xf:avg(($x,$y,$z))}</result>
```

The preceding query generates the following result:

```
<result>50.0</result>
```

In this example, the value of the nodes are extracted before the numbers are averaged together as if the data function had been invoked on each node in the sequence before being averaged together.

Related Topics

W3C avg function description.

xf:max

Finds the maximum value in a sequence.

If the value of \$item-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:max(integer* \$integer-var) > xs:anySimpleType?

Arguments

Data Type	Argument	Description
integer*	\$integer-var	Represents the sequence to evaluate.

Returns

Returns the number with the highest value from the sequence passed into \$item-var.

Examples

Simple

Invoking xf:max(("3","1","2")) returns the value of 3, as shown in the following example query:

```
<result>{xf:max(("3","1","2"))}</result>
```

The preceding query generates the following result:

```
<result>3</result>
```

Empty Sequences

Invoking xf:max((1,(),6,2,9)) returns the value of 9, as shown in the following example query:

```
<result>{xf:max((1,(),6,2,9))}</result>
```

The preceding query generates the following result:

```
<result>9</result>
```

Note: Instances of the empty sequence () are ignored.

Nodes

Invoking the following query returns the value of 100, as shown in the following example query:

```
let $x := <a>100</a>
```

xf:max

XQuery Reference

```
let $y := <b>2</b>
let $z := <c>50</c>
return <result>{xf:max(($x,$y,$z))}</result>
```

The preceding query generates the following result:

```
<result>50</result>
```

In this example, the value of the nodes are extracted before the comparison to determine the maximum number is done as if the data function had been invoked on each node in the sequence before the comparison. However, the string values of the nodes are extracted and compared and not the integer values of the node. In the preceding query, the string: 50 is reported as the maximum value because the ASCII value of the character 5 is greater than the ASCII value of the character 1, the first character in the string: 100. To compare the integer values of the nodes instead, first convert the node sequences to integers using the `bea-xf:integer-sequence` function and then invoke the `xf:max` function, as shown in the following example query:

```
let $x := <a>100</a>
let $y := <b>2</b>
let $z := <c>50</c>
return <result>{xf:max(bea-xf:integer-sequence(($x,$y,$z)))}</result>
```

The preceding query generates the following result:

```
<result>100</result>
```

Related Topics

W3C max function description.

BEA `bea-xf:integer-sequence` function description.

xf:min

Finds the minimum value in a sequence.

If the value of \$item-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:min(integer* \$integer-var) > xs:anySimpleType?

Arguments

Data Type	Argument	Description
integer*	\$integer-var	Represents the sequence to evaluate.

Returns

Returns the number with the lowest value from the sequence passed into \$item-var.

Examples

Simple

Invoking xf:min(("3","1","2")) returns the value of 1, as shown in the following example query:

```
<result>{xf:min(("3","1","2"))}</result>
```

The preceding query generates the following result:

```
<result>1</result>
```

Empty Sequences

Invoking xf:min((1,(),6,2,9)) returns the value of 1, as shown in the following example query:

```
<result>{xf:min((1,(),6,2,9))}</result>
```

The preceding query generates the following result:

```
<result>1</result>
```

Note: Instances of the empty sequence () are ignored.

Nodes

Invoking the following query returns the value of 2, the lowest number in the sequence, as shown in the following example query:

XQuery Reference

```
let $x := <a>100</a>
let $y := <b>2</b>
let $z := <c>50</c>
return <result>{xf:min(($x,$y,$z))}</result>
```

The preceding query generates the following result:

```
<result>100</result>
```

In this example, the value of the nodes are extracted before the comparison to determine the minimum number is done as if the data function had been invoked on each node in the sequence before the comparison. However, the string values of the nodes are extracted and compared and not the integer values of the node. In the preceding query, the string: 100 is reported as the minimum value because the ASCII value of the character 1 is less than the ASCII value of the character 5, the first character in the string: 50. To compare the integer values of the nodes instead, first convert the node sequences to integers using the `bea-xf:integer-sequence` function and then invoke the `xf:min` function, as shown in the following example query:

```
let $x := <a>100</a>
let $y := <b>2</b>
let $z := <c>50</c>
return <result>{xf:min(bea-xf:integer-sequence(($x,$y,$z)))}</result>
```

The preceding query generates the following result:

```
<result>2</result>
```

Related Topics

W3C min function description.

BEA `bea-xf:integer-sequence` function description.

xf:sum

Determines the sum of all the items in a sequence.

Signatures

xf:sum(item* \$item-var) > xs:double?

Arguments

Data Type	Argument	Description
item*	\$item-var	Represents the sequence to evaluate.

Returns

Returns the sum of all the numbers in a sequence passed into \$item-var.

Returns 0.0 if \$item-var is the empty sequence. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Examples

Simple

Invoking xf:sum(("3","1","2")) returns the value of 6, as shown in the following example query:

```
<result>{xf:sum(("3","1","2"))}</result>
```

The preceding query generates the following result:

```
<result>6.0</result>
```

Empty Sequences

Invoking xf:sum((1,(),6,2,9)) returns the value of 18, as shown in the following example query:

```
<result>{xf:sum((1,(),6,2,9))}</result>
```

The preceding query generates the following result:

```
<result>18.0</result>
```

Note: Instances of the empty sequence () are ignored.

Nodes

Invoking the following query returns the value of 152, as shown in the following example query:

```
let $x := <a>100</a>
```

xf:sum

XQuery Reference

```
let $y := <b>2</b>
let $z := <c>50</c>
return <result>{xf:sum(($x,$y,$z))}</result>
```

The preceding query generates the following result:

```
<result>152.0</result>
```

In this example, the value of the nodes are extracted before the numbers are added to together as if the data function had been invoked on each node in the sequence before being added together.

Related Topics

W3C sum function description.

XQuery Node Functions Reference

This section provides descriptions of the XQuery node functions available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery functions. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002 available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the node functions available from the mapper functionality:

- xf:node-kind
- xf:node-name
- xf:local-name

xf:node-kind

If a \$node-var is not a node, the following error is displayed in the mapper:

```
Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method:
Type error in function node-kind invocation: expected type node, given type
[string@http://www.w3.org/2001/XMLSchema]
```

If the value of \$node-var is the empty sequence, the following error is displayed in the mapper:

```
Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method:
Type error in function node-kind invocation: expected type node, given type empty
```

The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:node-kind(xf:node \$node-var) > xs:string

Arguments

Data Type	Argument	Description
node	\$node-var	Represents the node from which to determine type.

Returns

Returns a string which represents the kind of \$node-var. One of the following strings will be returned: document, element, attribute, text, namespace, processing-instruction, or comment.

Examples

Element

When you invoke the following query:

```
let $x := <e a="b">c</e>
return <type>{xf:node-kind($x)}</type>
```

The following result is generated:

```
<type>element</type>
```

Attribute

When you invoke the following query:

```
let $x := <e a="b">c</e>
return <type>{xf:node-kind($x/@a)}</type>
```


The following result is generated:

```
<type>attribute</type>
```

Text

When you invoke the following query:

```
let $x := <e a="b">c</e>  
return <type>{xf:node-kind($x/text())}</type>
```

The following result is generated:

```
<type>text</type>
```

Related Topics

W3C node-kind function description.

xf:node-name

Returns the expanded QName of \$node-var. An expanded QName contains a namespace URI and a local name.

If the value of \$node-var is the empty sequence, the following error message is displayed:

```
Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method:
Type error in function node-name invocation: expected type node, given type empty
```

The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If a \$node-var is not a node, the following error is displayed in the mapper:

```
Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method:
Type error in function node-name invocation: expected type node, given type
[string@http://www.w3.org/2001/XMLSchema]
```

Signatures

xf:node-name(xf:node \$node-var) > xs:QName?

Arguments

Data Type	Argument	Description
node	\$node-var	Represents the nod to extract the QName from.

Returns

Returns the expanded QName of \$node-var. An expanded QName contains a namespace URI and a local name.

Returns the empty sequence for nodes that do not have names (document, text, processing-instruction, comment). The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Examples

Simple

When you invoke the following query:

```
let $x := <elem xmlns:foo="http://www.acme.org/"><foo:subelem/></elem>,
    $name := xf:node-name(children($x))
return
    <result>
        <uri> {
            xf:get-namespace-from-QName($name)
        }</uri>
        <local>{
            xf:get-local-name-from-QName($name)
```

```
    }</local>  
</result>
```

The preceding query generates the following result:

```
<?xml version="1.0" encoding="UTF-8"?>  
<result>  
  <uri>http://www.acme.org/</uri>  
  <local>subelem</local>  
</result>
```

Related Topics

W3C node-name function description.

xf:local-name

Returns the local name (as a string) of \$node-var.

Signatures

xf:local-name(xf:node? \$node-var) > xs:string

Arguments

Data Type	Argument	Description
node?	\$node-var	Represents the node to get the local name from.

Returns

Returns the local name (as a string) of \$node-var.

Examples

Simple

The following XML document defines the namespace called xacme:

```
<?xml version='1.0'?>
<mydoc xmlns:xacme="http://www.acme.com/foo"> <xacme:n/> </mydoc>
```

The preceding XML Document associates the namespace prefix xacme to the URI: <http://www.acme.com/foo>. The string xacme:n is qualified name for the URI/local name pair: ("<http://www.acme.com/foo>", "n").

The following example query returns the local part of the URI/local name pair as shown in the following XQuery code:

```
let $a := <mydoc xmlns:xacme="http://www.acme.com/foo"> <xacme:n/> </mydoc>
return <name>{xf:local-name($a/*[1])}</name>
```

The preceding query generates the following result:

```
<name>n</name>
```

Related Topics

W3C local-name function description.

XQuery QName Functions Reference

This section provides descriptions of the XQuery QName functions available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery functions. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002 available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the QName functions available from the mapper functionality:

- `xf:get-local-name-from-QName`
- `xf:get-namespace-from-QName`

xf:get-local-name-from-QName

Extracts the local part of the QName from \$QName-var.

A QName (qualified name) is made up of a namespace name and a local part. The namespace name is the URI associated with the prefix of a namespace declaration.

If the value of \$QName-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If a QName is not passed into \$QName-var, the following error is displayed in the mapper:

```
Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method:
Type error in function get-local-name-from-QName invocation:
expected type [QName@http://www.w3.org/2001/XMLSchema?],
given type [string@http://www.w3.org/2001/XMLSchema]
```

Signatures

xf:get-local-name-from-QName(xs:QName? \$QName-var) > xs:string?

Arguments

Data Type	Argument	Description
xs:QName	\$QName-var	Represents the QName to extract from.

Returns

Returns the a string which contains the local part of the QName from \$QName-var.

Examples

Simple

The following XML document defines the namespace called xacme:

```
<?xml version='1.0'?>
<mydoc xmlns:xacme="http://www.acme.com/foo">
  <xacme:n/>
</mydoc>
```

The preceding XML Document associates the namespace prefix xacme to the URI: http://www.acme.com/foo. The string xacme:n is qualified name for the URI/local name pair:("http://www.acme.com/foo", "n").

The following example query calls the get-local-name-from-QName function:

```
<qname_example>
{xs:string(xf:get-local-name-from-QName(xf:expanded-QName("http://www.acme.com/foo", "n")))}
</qname_example>
```

The preceding query generates the following result:

```
<qname_example>n</qname_example>
```

Related Topics

W3C [get-local-name-from-QName](#) function description.

xf:get-namespace-from-QName

Extracts the namespace URI from \$QName-var.

A QName (qualified name) is made up of a namespace name and a local part. The namespace name is the URI associated with the prefix of a namespace declaration.

If the value of \$QName-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

If a QName is not passed into \$QName-var, the following error is displayed in the mapper:

```
Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method:
Type error in function get-namespace-from-QName invocation:
expected type [QName@http://www.w3.org/2001/XMLSchema?],
given type [string@http://www.w3.org/2001/XMLSchema]
```

Signatures

xf:get-namespace-from-QName(xs:QName? \$QName-var) > xs:anyURI?

Arguments

Data Type	Argument	Description
xs:QName	\$QName-var	Represents the QName to extract from.

Returns

Returns the URI (of data type anyURI) of the namespace in \$QName-var.

Examples

Simple

The following XML document defines the namespace called xacme:

```
<?xml version='1.0'?>
<mydoc xmlns:xacme="http://www.acme.com/foo">
  <xacme:n/>
</mydoc>
```

The preceding XML Document associates the namespace prefix xacme to the URI: http://www.acme.com/foo. The string xacme:n is qualified name for the URI/local name pair: ("http://www.acme.com/foo", "n").

The following example query calls the get-namespace-from-QName function:

```
let $a := <mydoc xmlns:xacme="http://www.acme.com/foo"> <xacme:n/> </mydoc>
return
  <name>{
```


XQuery Reference

```
      xf:get-namespace-from-QName(xf:node-name($a/*[1]))  
    }</name>
```

The preceding query generates the following result:

```
<name>http://www.acme.com/foo</name>
```

Related Topics

W3C `get-namespace-from-QName` function description.

XQuery Date Functions Reference

This section provides descriptions of the XQuery date functions available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery functions. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the *XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002* available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the date functions available from the mapper functionality:

- `xf:get-year-from-dateTime`
- `xf:get-month-from-dateTime`
- `xf:get-day-from-dateTime`
- `xf:get-hours-from-dateTime`
- `xf:get-minutes-from-dateTime`
- `xf:get-seconds-from-dateTime`
- `xf:get-timezone-from-dateTime`
- `xf:get-year-from-date`
- `xf:get-month-from-date`
- `xf:get-day-from-date`
- `xf:get-timezone-from-date`
- `xf:get-hours-from-time`
- `xf:get-minutes-from-time`
- `xf:get-seconds-from-time`
- `xf:get-timezone-from-time`
- `xf:add-timezone-to-dateTime`
- `xf:remove-timezone-from-dateTime`
- `xf:add-timezone-to-date`
- `xf:add-timezone-to-time`
- `xf:remove-timezone-from-time`
- `xf:current-dateTime`
- `xf:current-date`
- `xf:current-time`

xf:get-year-from-dateTime

Extracts the year from \$dateTime-var.

If the value of \$dateTime-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-year-from-dateTime(xs:dateTime? \$dateTime-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:dateTime?	\$dateTime-var	Contains a representation of the date and time.

Returns

Returns the year as an integer from \$dateTime-var.

Examples

Simple

When you invoke the following query:

```
<year>{xf:get-year-from-dateTime("2002-08-30T22:21:01")}</year>
```

The preceding query generates the following result:

```
<year>2002</year>
```

Related Topics

W3C get-year-from-dateTime function description.

W3C dateTime data type description.

xf:get-month-from-dateTime

Extracts the month from \$dateTime-var.

If the value of \$dateTime-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-month-from-dateTime(xs:dateTime? \$dateTime-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:dateTime?	\$dateTime-var	Contains a representation of the date and time.

Returns

Returns the month as an integer from \$dateTime-var.

Examples

Simple

When you invoke the following query:

```
<month>{xf:get-month-from-dateTime("2002-08-30T22:21:01")}</month>
```

The preceding query generates the following result:

```
<month>8</month>
```

Related Topics

W3C get-month-from-dateTime function description.

W3C dateTime data type description.

xf:get-day-from-dateTime

Extracts the day from \$dateTime-var.

If the value of \$dateTime-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-day-from-dateTime(xs:dateTime? \$dateTime-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:dateTime?	\$dateTime-var	Contains a representation of the date and time.

Returns

Returns the day as an integer from \$dateTime-var.

Examples

Simple

When you invoke the following query:

```
<day>{xf:get-day-from-dateTime("2002-08-30T22:21:01")}</day>
```

The preceding query generates the following result:

```
<day>30</day>
```

Related Topics

W3C get-day-from-dateTime function description.

W3C dateTime data type description.

xf:get-hours-from-dateTime

Extracts the hours from \$dateTime-var.

If the value of \$dateTime-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-hours-from-dateTime(xs:dateTime? \$dateTime-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:dateTime?	\$dateTime-var	Contains a representation of the date and time.

Returns

Returns the hours as an integer from \$dateTime-var.

Examples

Simple

When you invoke the following query:

```
<hours>{xf:get-hours-from-dateTime("2002-08-30T22:21:01")}</hours>
```

The preceding query generates the following result:

```
<hours>22</hours>
```

Timezone

The following example uses the get-hours-from-dateTime function with a timezone:

```
<hours>{xf:get-hours-from-dateTime("2002-08-30T14:21:01-05:00")}</hours>
```

The preceding query generates the following result:

```
<hours>19</hours>
```

In the preceding query, the -05:00 string specifies the time is specified in the Eastern Standard timezone which is 5 hours behind Coordinated Universal Time (UTC). The hours are reported in Coordinated Universal Time (UTC), so 5 hours are added to 14 hours of the Eastern Standard timezone, resulting in 19 hours in Coordinated Universal Time (UTC).

Related Topics

W3C `get-hours-from-dateTime` function description.

W3C `dateTime` data type description.

xf:get-minutes-from-dateTime

Extracts the minutes from \$dateTime-var.

If the value of \$dateTime-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-minutes-from-dateTime(xs:dateTime? \$dateTime-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:dateTime?	\$dateTime-var	Contains a representation of the date and time.

Returns

Returns the minutes as an integer from \$dateTime-var.

Examples

Simple

When you invoke the following query:

```
<minutes>{xf:get-minutes-from-dateTime("2002-08-30T22:21:01")}</minutes>
```

The preceding query generates the following result:

```
<minutes>21</minutes>
```

Related Topics

W3C get-minutes-from-dateTime function description.

W3C dateTime data type description.

xf:get-seconds-from-dateTime

Extracts the seconds from \$dateTime-var.

If the value of \$dateTime-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-seconds-from-dateTime(xs:dateTime? \$dateTime-var) > xs:xs:decimal?

Arguments

Data Type	Argument	Description
xs:dateTime?	\$dateTime-var	Contains a representation of the date and time.

Returns

Returns the seconds as an decimal from \$dateTime-var.

Examples

Simple

When you invoke the following query:

```
<seconds>{xf:get-seconds-from-dateTime("2002-08-30T22:21:01")}</seconds>
```

The preceding query generates the following result:

```
<seconds>1</seconds>
```

Related Topics

W3C get-seconds-from-dateTime function description.

W3C dateTime data type description.

xf:get-timezone-from-dateTime

Extracts the current timezone from \$dateTime-var and returns it as a string.

If \$dateTime-var does not contain a timezone, the empty sequence is returned.

If the value of \$dateTime-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-timezone-from-dateTime(xs:dateTime? \$dateTime-var) > xs:string?

Arguments

Data Type	Argument	Description
xs:dateTime?	\$dateTime-var	Contains a representation of the date and time.

Returns

Returns the current timezone as a string from \$dateTime-var.

Returns an empty sequence, if \$dateTime-var does not contain a timezone. (See examples below.)

Examples

Timezone

The following example uses the get-timezone-from-dateTime function with a timezone:

```
<tz>{xf:get-timezone-from-dateTime("2002-08-30T14:21:01-05:00")}</tz>
```

The preceding query generates the following result:

```
<tz>-05:00</tz>
```

In the preceding query, the -05:00 string specifies the time is specified in the Eastern Standard time zone which is 5 hours behind Universal Time, Coordinated (UTC).

UTC Timezone

The following example uses the get-timezone-from-dateTime function with a timezone:

```
<tz>{xf:get-timezone-from-dateTime("2002-08-30T14:21:01Z")}</tz>
```

The preceding query generates the following result:

```
<tz>00:00</tz>
```

In this example, the Z means the time is being specified in Universal Coordinated, Time (UTC), so no timezone is reported.

No Timezone

The following is an example of using the `get-timezone-from-dateTime` function with no timezone specified:

```
<tz>{xf:get-timezone-from-dateTime( "2002-08-30T22:21:01" )}</tz>
```

The preceding query generates the following result:

```
<tz/>
```

Related Topics

W3C `get-timezone-from-dateTime` function description.

W3C `dateTime` data type description.

xf:get-year-from-date

Extracts the year from \$date-var.

If the value of \$date-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-year-from-date(xs:date? \$date-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:date?	\$date-var	Contains a representation of the date.

Returns

Returns the year as an integer from \$date-var.

Examples

Simple

When you invoke the following query:

```
<year>{xf:get-year-from-date("2002-08-30")}</year>
```

The preceding query generates the following result:

```
<year>2002</year>
```

Related Topics

W3C get-year-from-date function description.

W3C date data type description.

xf:get-month-from-date

Extracts the month from \$date-var.

If the value of \$date-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-month-from-date(xs:date? \$date-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:date?	\$date-var	Contains a representation of the date.

Returns

Returns the month as an integer from \$date-var.

Examples

Simple

When you invoke the following query:

```
<month>{get-month-from-date("2002-08-30")}</month>
```

The preceding query generates the following result:

```
<month>8</month>
```

Related Topics

W3C get-month-from-date function description.

W3C date data type description.

xf:get-day-from-date

Extracts the day from \$date-var.

If the value of \$date-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-day-from-date(xs:date? \$date-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:date?	\$date-var	Contains a representation of the date.

Returns

Returns the day as an integer from \$date-var.

Examples

Simple

When you invoke the following query:

```
<day>{xf:get-day-from-date("2002-08-30")}</day>
```

The preceding query generates the following result:

```
<day>30</day>
```

Related Topics

W3C get-day-from-date function description.

W3C date data type description.

xf:get-timezone-from-date

Extracts the current timezone from \$date-var.

If \$date-var does not contain a timezone, the empty sequence is returned.

If the value of \$date-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-timezone-from-date(xs:date? \$date-var) > xs:string?

Arguments

Data Type	Argument	Description
xs:date?	\$date-var	Contains a representation of the date.

Returns

Returns the current timezone as a string from \$date-var.

Returns an empty sequence, if \$date-var does not contain a timezone. (See examples below.)

Examples

Timezone

The following example uses the get-timezone-from-date function with a timezone:

```
<tz>{xf:get-timezone-from-date(xs:date("2002-08-30-05:00"))}</tz>
```

The preceding query generates the following result:

```
<tz>-05:00</tz>
```

In the preceding query, the -05:00 string specifies the time is specified in the Eastern Standard timezone which is 5 hours behind Coordinated Universal Time (UTC).

UTC Timezone

The following example uses the get-timezone-from-date function with a timezone:

```
<tz>{xf:get-timezone-from-date(xs:date("2002-08-30Z"))}</tz>
```

The preceding query generates the following result:

```
<tz>00:00</tz>
```

In this preceding query, the Z means the time is being specified in Coordinated Universal Time (UTC), so no timezone is reported.

No Timezone

The following is an example of using the `get-timezone-from-date` function with no timezone specified:

```
<tz>{xf:get-timezone-from-date(xs:date("2002-08-30"))}</tz>
```

The preceding query generates the following result:

```
<tz/>
```

Related Topics

W3C `get-timezone-from-date` function description.

W3C date data type description.

xf:get-hours-from-time

Extracts the hours from \$time-var.

If the value of \$time-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-hours-from-time(xs:time? \$time-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:time?	\$time-var	Contains a representation of time.

Returns

Returns the hours as an integer from \$time-var.

Examples

Simple

The following is a simple example using the get-hours-from-time function:

```
<hours>{xf:get-hours-from-time("22:21:01")}</hours>
```

The preceding query generates the following result:

```
<hours>22</hours>
```

Timezone

The following example uses the get-hours-from-time function with a timezone:

```
<hours>{xf:get-hours-from-time("14:21:01-05:00")}</hours>
```

The preceding query generates the following result:

```
<hours>19</hours>
```

In the preceding query, the -05:00 string specifies the time is specified in the Eastern Standard timezone which is 5 hours behind Coordinated Universal Time (UTC). The hours are reported in Coordinated Universal Time (UTC), so 5 hours are added to 14 hours of the Eastern Standard timezone, resulting in 19 hours in Coordinated Universal Time (UTC).

Related Topics

W3C `get-hours-from-time` function description.

W3C time data type description.

xf:get-minutes-from-time

Extracts the minutes from \$time-var.

If the value of \$time-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-minutes-from-time(xs:time? \$time-var) > xs:integer?

Arguments

Data Type	Argument	Description
xs:time?	\$time-var	Contains a representation of time.

Returns

Returns the minutes as an integer from \$time-var.

Examples

Simple

The following is a simple example using the get-minutes-from-time function:

```
<minutes>{xf:get-minutes-from-time("22:21:01")}</minutes>
```

The preceding query generates the following result:

```
<minutes>21</minutes>
```

Timezone

The following example uses the get-minutes-from-time function with a timezone:

```
<minutes>{xf:get-minutes-from-time("14:21:01-05:00")}</minutes>
```

The preceding query generates the following result:

```
<minutes>21</minutes>
```

Related Topics

W3C get-minutes-from-time function description.

W3C time data type description.

xf:get-seconds-from-time

Extracts the seconds from \$time-var.

If the value of \$time-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-seconds-from-time(xs:time? \$time-var) > xs:decimal?

Arguments

Data Type	Argument	Description
xs:time?	\$time-var	Contains a representation of time.

Returns

Returns the seconds as an decimal from \$time-var.

Examples

Simple

The following is a simple example using the get-seconds-from-time function:

```
<seconds>{xf:get-seconds-from-time("22:21:01")}</seconds>
```

The preceding query generates the following result:

```
<seconds>1</seconds>
```

Timezone

The following example uses the get-seconds-from-time function with a timezone:

```
<seconds>{xf:get-seconds-from-time("14:21:01-05:00")}</seconds>
```

The preceding query generates the following result:

```
<seconds>1</seconds>
```

Related Topics

W3C get-seconds-from-time function description.

W3C time data type description.

xf:get-timezone-from-time

Extracts the current timezone from \$time-var.

If \$time-var does not contain a timezone, the empty sequence is returned.

If the value of \$time-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-timezone-from-time(xs:time? \$time-var) > xs:string?

Arguments

Data Type	Argument	Description
xs:time?	\$time-var	Contains a representation of the date and time.

Returns

Returns the current timezone as a string from \$time-var.

Returns an empty sequence, if \$time-var does not contain a timezone. (See examples below.)

Examples

Timezone

The following example uses the get-timezone-from-time function with a timezone:

```
<tz>{xf:get-timezone-from-time(xs:time("14:21:01-05:00"))}</tz>
```

The preceding query generates the following result:

```
<tz>-05:00</tz>
```

In the preceding query, the -05:00 string specifies the time is specified in the Eastern Standard timezone which is 5 hours behind Coordinated Universal Time (UTC).

UTC Timezone

The following example uses the get-timezone-from-time function with a timezone:

```
<tz>{get-timezone-from-time(xs:time("14:21:01Z"))}</tz>
```

The preceding query generates the following result:

```
<tz>00:00</tz>
```

In this preceding query, the Z means the time is being specified in Coordinated Universal Time (UTC), so no timezone is reported.

No Timezone

The following is an example of using the `get-timezone-from-time` function with no timezone specified:

```
<tz>{xf:get-timezone-from-time(xs:time("22:21:01"))}</tz>
```

Returns an empty sequence, because `$time-var` does not contain a timezone.

Related Topics

W3C `get-timezone-from-time` function description.

W3C time data type description.

xf:add-timezone-to-dateTime

Adds a time zone to a dateTime value. The timezone added to the dateTime value is either the implicit time zone or passed in with the optional argument: \$dayTimeDuration-var.

Note: The implicit time zone is obtained from the current environment. For example, if the time zone specified on your machine is the PSD (Pacific Daylight Savings), the implicit time zone would be -07:00.

Adds a time zone to a dateTime value by invoking the following steps:

1. If the specified dateTime value already has a time zone value, that time zone is removed from the dateTime value. Since the purpose of the add-timezone-to-dateTime function is to add a timezone to a dateTime value without a timezone, in most cases the passed in dateTime value (argument: \$dateTime-var) will have not have a time zone, so this step will be ignored as shown in the preceding examples.
2. Adds the time zone, specified by either the optional argument \$dayTimeDuration-var or the implicit time, to the dateTime value (argument: \$dateTime-var).
3. Converts the new dateTime value generated by the previous step to a Universal Coordinated Time (UTC) time zone value.

Signatures

xf:add-timezone-to-dateTime(xs:dateTime \$dateTime-var) > xs:dateTime

xf:add-timezone-to-dateTime(xs:dateTime \$dateTime-var, xf:dayTimeDuration, \$dayTimeDuration-var) > xs:dateTime

Arguments

Data Type	Argument	Description
xs:dateTime?	\$dateTime-var	Contains a representation of the date and time.
xf:dayTimeDuration	\$dayTimeDuration-var	Time zone to add to \$dateTime-var. (Optional, if not specified the implicit time zone is used).

Returns

Returns a dateTime with a time zone.

If \$dateTime-var contains a time zone and the optional argument \$dayTimeDuration-var is not specified, the passed in date and time are returned with no conversion.

Examples

Adding the Specified Timezone to the dateTime Value

The following example adds the specified time zone to the specified dateTime value. In this example, the specified dateTime value: 2002-12-06T12:00:00, initially does not have a timezone zone associated with it.

XQuery Reference

Adding the specified time zone to the specified dateTime value, by invoking `xf:add-timezone-to-dateTime(xs:dateTime("2002-12-06T12:00:00", dayTimeDuration("-PT7H")))`, causes the following internal conversion steps:

1. In this example, the time zone is specified with the dayTimeDuration string: `-PT7H`, which is the Pacific Daylight Savings Time (PST) time zone (`-07:00`). This specified time zone: `-07:00` is added to the specified dateTime value: `2002-12-06T12:00:00` resulting in the following internal dateTime value: `2002-12-06T12:00:00-07:00`.
2. This new dateTime value: `2002-12-06T19:00:00-07:00` is converted to Universal Coordinated Time (UTC), resulting in the following dateTime value: `2002-12-06T19:00:00Z`, as shown in the following example query:

```
let $dateTimeWithNoTimezone := xs:dateTime("2002-12-06T12:00:00")
return
<result>
  <dateTimeWithNoTimezone>{$dateTimeWithNoTimezone}</dateTimeWithNoTimezone>
  <dateTimeWithTimezoneAdded>{ xf:add-timezone-to-dateTime(xs:dateTime($dateTimeWithNoTimezone,
    xf:dayTimeDuration("-PT7H")) ) }</dateTimeWithTimezoneAdded>
</result>
```

The preceding query produces the following result:

```
<result>
  <dateTimeWithNoTimezone>2002-12-06T12:00:00</dateTimeWithNoTimezone>
  <dateTimeWithTimezoneAdded>2002-12-06T19:00:00Z</dateTimeWithTimezoneAdded>
</result>
```

Note: The Z at the end of the `2002-12-06T19:00:00Z` value specifies that the dateTime value is in UTC time.

Adding the Implicit Timezone to the dateTime Value

The following example adds the implicit time zone to the specified dateTime value. In this example, the specified dateTime value: `2002-12-06T12:00:00`, initially does not have a timezone zone associated with it.

In this example, the implicit time zone is assumed to be `-07:00`, the Pacific Daylight Savings Time (PST) time zone. (The implicit time zone is the default time zone for the current machine.)

Adding the implicit time zone to the specified dateTime value, by invoking `xf:add-timezone-to-dateTime(xs:dateTime("2002-12-06T12:00:00"))`, causes the following internal conversion steps:

1. The implicit time zone: `-07:00` is added to the specified dateTime value: `2002-12-06T12:00:00` resulting in the following internal dateTime value: `2002-12-06T12:00:00-07:00`.
2. This new dateTime value: `2002-12-06T19:00:00-07:00` is converted to Universal Coordinated Time (UTC), resulting in the following dateTime value: `2002-12-06T19:00:00Z`, as shown in the following example query:

```
let $dateTimeWithNoTimezone := xs:dateTime("2002-12-06T12:00:00")
return
<result>
  <dateTimeWithNoTimezone>{$dateTimeWithNoTimezone}</dateTimeWithNoTimezone>
  <dateTimeWithTimezoneAdded>{ xf:add-timezone-to-dateTime(xs:dateTime($dateTimeWithNoTimezone)) }
  </dateTimeWithTimezoneAdded>
```



```
</result>
```

The preceding query produces the following result:

```
<result>
  <dateTimeWithNoTimezone>2002-12-06T12:00:00</dateTimeWithNoTimezone>
  <dateTimeWithTimeZoneAdded>2002-12-06T19:00:00Z</dateTimeWithTimeZoneAdded>
</result>
```

Note: The Z at the end of the 2002-12-06T19:00:00Z value specifies that the dateTime value is in UTC time.

Note: The results of this query maybe be different for your machine because the implicit time zone may be different.

Related Topics

W3C add-timezone-to-dateTime function description.

W3C dateTime data type description.

W3C dayTimeDuration operator description.

xf:remove-timezone-from-dateTime

Removes a time zone from a dateTime value. The dateTime value with a time zone is localized to either the implicit time zone or to time zone specified by the argument: \$dayTimeDuration-var and then the timezone indicator is removed.

If the specified dateTime value has no time zone associated with it, then the implicit time zone is added to dateTime value before the localization described above is done. Since the purpose of the remove-timezone-to-dateTime function is to remove a timezone from a dateTime value with a timezone, in most cases the passed in dateTime value (argument: \$dateTime-var) will have a time zone, so this step will be ignored as shown in the preceding examples.

Note: The implicit time zone is obtained from the current environment. For example, if the time zone specified on your machine is the PSD (Pacific Daylight Savings), the implicit time zone would be -07:00.

Signatures

xf:remove-timezone-from-dateTime(xs:dateTime \$dateTime-var) > xs:dateTime

xf:remove-timezone-from-dateTime(xs:dateTime \$dateTime-var, xf:dayTimeDuration, \$dayTimeDuration-var) > xs:dateTime

Arguments

Data Type	Argument	Description
xs:dateTime?	\$dateTime-var	Contains a representation of the date and time.
xf:dayTimeDuration	\$dayTimeDuration-var	Time zone to remove from \$dateTime-var. (Optional, if not specified the implicit time zone is used).

Returns

Returns a dateTime without a time zone.

If \$dateTime-var does not contain a time zone and optional argument \$dayTimeDuration-var is not specified, the passed in date and time are returned with no conversion.

Examples

Removing the Specified Timezone From a dateTime Value

The following example removes the specified time zone from the specified dateTime value.

In this example, the passed in dateTime value: 2002-12-06T12:00:00Z is specified with the UTC time zone specified by the time zone indicator: Z.

The time zone is specified with the dayTimeDuration string: -PT7H, in the preceding query which is the Pacific Daylight Savings Time (PST) time zone (-07:00).

XQuery Reference

The passed in `dateTime` value: 2002-12-06T12:00:00Z specified with the UTC time zone is localized to PST time zone and the time zone indicator (Z) is removed, as shown in the following example query:

```
let $dateTimeWithUTCtimezone := xs:dateTime("2002-12-06T12:00:00Z")
return
<result>
  <dateTimeWithUTCtimezone>{$dateTimeWithUTCtimezone}</dateTimeWithUTCtimezone>
  <dateTimeWithTimeZoneAdded>{ xf:remove-timezone-from-dateTime(xs:dateTime($dateTimeWith
    xf:dayTimeDuration("-PT7H"))) }</dateTimeWithTimeZoneAdded>
</result>
```

The preceding query produces the following result:

```
<result>
  <dateTimeWithUTCtimezone>2002-12-06T12:00:00Z</dateTimeWithUTCtimezone>
  <dateTimeWithTimeZoneAdded>2002-12-06T05:00:00</dateTimeWithTimeZoneAdded>
</result>
```

Removing the Implicit Timezone From the `dateTime` Value

The following example removes the implicit time zone from the specified `dateTime` value. In this example, the passed in `dateTime` value: 2002-12-06T12:00:00Z is specified with the UTC time zone specified by the time zone indicator: Z.

In this example, the implicit time zone is assumed to be -07:00, the Pacific Daylight Savings Time (PST) time zone. (The implicit time zone is the default time zone for the current machine.)

The passed in `dateTime` value: 2002-12-06T12:00:00Z specified with the UTC time zone is localized to PST time zone and the time zone indicator (Z) is removed, as shown in the following example query:

```
let $dateTimeWithUTCtimezone := xs:dateTime("2002-12-06T12:00:00Z")
return
<result>
  <dateTimeWithUTCtimezone>{$dateTimeWithUTCtimezone}</dateTimeWithUTCtimezone>
  <dateTimeWithTimeZoneAdded>{ xf:remove-timezone-from-dateTime(xs:dateTime($dateTimeWith
    </dateTimeWithTimeZoneAdded>
</result>
```

The preceding query produces the following result:

```
<result>
  <dateTimeWithUTCtimezone>2002-12-06T12:00:00Z</dateTimeWithUTCtimezone>
  <dateTimeWithTimeZoneAdded>2002-12-06T05:00:00</dateTimeWithTimeZoneAdded>
</result>
```

Related Topics

W3C `remove-timezone-from-dateTime` function description.

W3C `dateTime` data type description.

W3C `dayTimeDuration` operator description.

xf:add-timezone-to-date

Adds a time zone to a date value. The timezone added to the date value is either the implicit time zone or passed in with the optional argument: \$dayTimeDuration-var.

Note: The implicit time zone is obtained from the current environment. For example, if the time zone specified on your machine is the PSD (Pacific Daylight Savings), the implicit time zone would be -07:00.

Adds a time zone to a date value by invoking the following steps:

1. If the specified date value already has a time zone value, that time zone is removed from the date value. Since the purpose of the add-timezone-to-date function is to add a timezone to a date value without a timezone, in most cases the passed in date value (argument: \$date-var) will have not have a time zone, so this step will be ignored as shown in the preceding examples.
2. Adds the time zone, specified by either the optional argument \$dayTimeDuration-var or the implicit time, to the date value (argument: \$date-var).
3. Converts the new date value generated by the previous step to a Universal Coordinated Time (UTC) time zone value.

Signatures

xf:add-timezone-to-date(xs:date \$date-var) > xs:date

xf:add-timezone-to-date(xs:date \$date-var, xf:dayTimeDuration \$dayTimeDuration-var) > xs:date

Arguments

Data Type	Argument	Description
xs:date?	\$date-var	Contains a representation of the date.
xf:dayTimeDuration	\$dayTimeDuration-var	Time zone to add to \$date-var. (Optional, if not specified the implicit time zone is used).

Returns

Returns a date with a time zone.

If \$date-var contains a time zone and optional argument \$dayTimeDuration-var is not specified, the passed in date and time are returned with no conversion.

Examples

Adding the Specified Timezone to the date Value

The following example adds the specified time zone to the specified date value. In this example, the specified date value: 2002-12-06, initially does not have a timezone zone associated with it.

Adding the specified time zone to the specified date value, by invoking
xf:add-timezone-to-date(xs:date("2002-12-06", dayTimeDuration("-PT7H"))), causes the following

xf:add-timezone-to-date

internal conversion steps:

1. In this example, the time zone is specified with the `dayTimeDuration` string: `–PT7H`, which is the Pacific Daylight Savings Time (PST) time zone (`–07:00`). This specified time zone: `–07:00` is added to the specified date value: `2002–12–06` resulting in the following internal date value: `2002–12–06–07:00`.
2. This new date value: `2002–12–06–07:00` is converted to Universal Coordinated Time (UTC), resulting in the following date value: `2002–12–06Z`, as shown in the following example query:

```
let $dateWithNoTimezone := xs:date("2002-12-06")
return
<result>
  <dateWithNoTimezone>{$dateWithNoTimezone}</dateWithNoTimezone>
  <dateWithTimeZoneAdded>{ xf:add-timezone-to-date(xs:date($dateWithNoTimezone),
    xf:dayTimeDuration("–PT7H")) }</dateWithTimeZoneAdded>
</result>
```

The preceding query produces the following result:

```
<result>
  <dateWithNoTimezone>2002-12-06</dateWithNoTimezone>
  <dateWithTimeZoneAdded>2002-12-06Z</dateWithTimeZoneAdded>
</result>
```

Note: The `Z` at the end of the `2002–12–06Z` value specifies that the date value is in UTC time.

Adding the Implicit Timezone to the date Value

The following example adds the implicit time zone to the specified date value. In this example, the specified date value: `2002–12–06`, initially does not have a time zone zone associated with it.

In this example, the implicit time zone is assumed to be `–07:00`, the Pacific Daylight Savings Time (PST) time zone. (The implicit time zone is the default time zone for the current machine.)

Adding the implicit time zone to the specified date value, by invoking `xf:add-timezone-to-date(xs:date("2002–12–06"))`, causes the following internal conversion steps:

1. The implicit time zone: `–07:00` is added to the specified date value: `2002–12–06` resulting in the following internal date value: `2002–12–06–07:00`.
2. This new date value: `2002–12–06–07:00` is converted to Universal Coordinated Time (UTC), resulting in the following date value: `2002–12–06Z`, as shown in the following example query:

```
let $dateWithNoTimezone := xs:date("2002-12-06")
return
<result>
  <dateWithNoTimezone>{$dateWithNoTimezone}</dateWithNoTimezone>
  <dateWithTimeZoneAdded>{ xf:add-timezone-to-date(xs:date($dateWithNoTimezone)) }
  </dateWithTimeZoneAdded>
</result>
```

The preceding query produces the following result:

```
<result>
  <dateWithNoTimezone>2002-12-06</dateWithNoTimezone>
```

XQuery Reference

```
<dateWithTimeZoneAdded>2002-12-06Z</dateWithTimeZoneAdded>  
</result>
```

Note: The Z at the end of the 2002-12-06Z value specifies that the date value is in UTC time.

Note: The results of this query maybe be different for your machine because the implicit time zone may be different.

Related Topics

W3C add-timezone-to-date function description.

W3C date data type description.

W3C dayTimeDuration operator description.

xf:add-timezone-to-time

Adds a time zone to a time value. The timezone added to the time value is either the implicit time zone or passed in with the optional argument: \$dayTimeDuration-var.

Note: The implicit time zone is obtained from the current environment. For example, if the time zone specified on your machine is the PSD (Pacific Daylight Savings), the implicit time zone would be -07:00.

Adds a time zone to a time value by invoking the following steps:

1. If the specified time value already has a time zone value, that time zone is removed from the time value. Since the purpose of the add-timezone-to-time function is to add a timezone to a time value without a timezone, in most cases the passed in time value (argument: \$time-var) will have not have a time zone, so this step will be ignored as shown in the preceding examples.
2. Adds the time zone, specified by either the optional argument \$dayTimeDuration-var or the implicit time, to the time value (argument: \$time-var).
3. Converts the new time value generated by the previous step to a Universal Coordinated Time (UTC) time zone value.

Signatures

xf:add-timezone-to-time(xs:time \$time-var) > xs:time

xf:add-timezone-to-time(xs:time \$time-var, xf:dayTimeDuration, \$dayTimeDuration-var) > xs:time

Arguments

Data Type	Argument	Description
xs:time	\$time-var	Contains a representation of the time.
xf:dayTimeDuration	\$dayTimeDuration-var	Time zone to remove to \$time-var. (Optional, if not specified the implicit time zone is used).

Returns

Returns a time with a time zone.

If \$time-var contains a time zone and optional argument \$dayTimeDuration-var is not specified, the passed in date and time are returned with no conversion.

Examples

Adding the Specified Timezone to the time Value

The following example adds the specified time zone to the specified time value. In this example, the specified time value: 12:00:00, initially does not have a timezone zone associated with it.

Adding the specified time zone to the specified time value, by invoking
xf:date-timezone-to-time(xs:time("12:00:00", dayTimeDuration("-PT7H"))), causes the following internal

xf:add-timezone-to-time

conversion steps:

1. In this example, the time zone is specified with the `dayTimeDuration` string: `-PT7H`, which is the Pacific Daylight Savings Time (PST) time zone (`-07:00`). This specified time zone: `-07:00` is added to the specified time value: `12:00:00` resulting in the following internal time value: `12:00:00-07:00`.
2. This new time value: `12:00:00-07:00` is converted to Universal Coordinated Time (UTC), resulting in the following time value: `19:00:00Z`, as shown in the following example query:

```
let $timeWithNoTimezone := xs:time("12:00:00")
return
<result>
  <timeWithNoTimezone>{$timeWithNoTimezone}</timeWithNoTimezone>
  <timeWithTimeZoneAdded>{ xf:add-timezone-to-time(xs:time($timeWithNoTimezone),
    xf:dayTimeDuration("-PT7H")) }</timeWithTimeZoneAdded>
</result>
```

The preceding query produces the following result:

```
<result>
  <timeWithNoTimezone>12:00:00</timeWithNoTimezone>
  <timeWithTimeZoneAdded>19:00:00Z</timeWithTimeZoneAdded>
</result>
```

Note: The Z at the end of the `19:00:00Z` value specifies that the time value is in UTC time.

Adding the Implicit Timezone to the time Value

The following example adds the implicit time zone to the specified time value. In this example, the specified time value: `12:00:00`, initially does not have a timezone zone associated with it.

In this example, the implicit time zone is assumed to be `-07:00`, the Pacific Daylight Savings Time (PST) time zone. (The implicit time zone is the default time zone for the current machine.)

Adding the implicit time zone to the specified time value, by invoking `xf:add-timezone-to-time(xs:time("12:00:00"))`, causes the following internal conversion steps:

1. The implicit time zone: `-07:00` is added to the specified time value: `12:00:00` resulting in the following internal time value: `12:00:00-07:00`.
2. This new time value: `12:00:00-07:00` is converted to Universal Coordinated Time (UTC), resulting in the following time value: `19:00:00Z`, as shown in the following example query:

```
let $timeWithNoTimezone := xs:time("12:00:00")
return
<result>
  <timeWithNoTimezone>{$timeWithNoTimezone}</timeWithNoTimezone>
  <timeWithTimeZoneAdded>{ xf:add-timezone-to-time(xs:time($timeWithNoTimezone)) }
  </timeWithTimeZoneAdded>
</result>
```

The preceding query produces the following result:

```
<result>
  <timeWithNoTimezone>12:00:00</timeWithNoTimezone>
  <timeWithTimeZoneAdded>19:00:00Z</timeWithTimeZoneAdded>
</result>
```


XQuery Reference

Note: The Z at the end of the 19:00:00Z value specifies that the time value is in UTC time.

Note: The results of this query maybe be different for your machine because the implicit time zone may be different.

Related Topics

W3C add-timezone-to-time function description.

W3C time data type description.

W3C dayTimeDuration operator description.

xf:remove-timezone-from-time

Removes a time zone from a time value. The time value with a time zone is localized to either the implicit time zone or to time zone specified by the argument: \$dayTimeDuration-var and then the timezone indicator is removed.

If the specified time value has no time zone associated with it, then the implicit time zone is added to time value before the localization described above is done. Since the purpose of the remove-timezone-to-time function is to remove a timezone from a time value with a timezone, in most cases the passed in time value (argument: \$time-var) will have a time zone, so this step will be ignored as shown in the following examples.

Note: The implicit time zone is obtained from the current environment. For example, if the time zone specified on your machine is the PSD (Pacific Daylight Savings), the implicit time zone would be -07:00.

Signatures

xf:remove-timezone-from-time(xs:time \$time-var) > xs:time

xf:remove-timezone-from-time(xs:time \$time-var, xf:dayTimeDuration, \$dayTimeDuration-var) > xs:time

Arguments

Data Type	Argument	Description
xs:time	\$time-var	Contains a representation of the time.
xf:dayTimeDuration	\$dayTimeDuration-var	Time zone to add to \$time-var as a duration. (Optional, if not specified the implicit time zone is used).

Returns

Returns a time without a time zone.

If \$time-var does not contain a time zone and optional argument \$dayTimeDuration-var is not specified, the passed in time is returned with no conversion.

Examples

Removing the Specified Timezone From a time Value

The following example removes the specified time zone from the specified time value.

In this example, the passed in time value: 12:00:00Z is specified with the UTC time zone as specified by the time zone indicator: Z.

The time zone is specified with the dayTimeDuration string: -PT7H, in the preceding query which is the Pacific Daylight Savings Time (PST) time zone (-07:00).

XQuery Reference

The passed in time value: 12:00:00Z specified with the UTC time zone is localized to PST time zone and the time zone indicator (Z) is removed, as shown in the following example query:

```
let $timeWithUTCtimezone := xs:time("12:00:00Z")
return
<components>
  <timeWithUTCtimezone>{$timeWithUTCtimezone}</timeWithUTCtimezone>
  <timeWithTimeZoneAdded>{ xf:remove-timezone-from-time(xs:time($timeWithUTCtimezone),
    xf:dayTimeDuration("-PT7H")) }</timeWithTimeZoneAdded>
</components>
```

The preceding query produces the following result:

```
<components>
  <timeWithUTCtimezone>12:00:00Z</timeWithUTCtimezone>
  <timeWithTimeZoneAdded>05:00:00</timeWithTimeZoneAdded>
</components>
```

Removing the Implicit Timezone From the time Value

The following example removes the implicit time zone from the specified time value. In this example, the passed in time value: 12:00:00Z is specified with the UTC time zone as specified by the time zone indicator: Z.

In this example, the implicit time zone is assumed to be -07:00, the Pacific Daylight Savings Time (PST) time zone. (The implicit time zone is the default time zone for the current machine.)

The passed in time value: 12:00:00Z specified with the UTC time zone is localized to PST time zone and the time zone indicator (Z) is removed, as shown in the following example query:

```
let $timeWithUTCtimezone := xs:time("12:00:00Z")
return
<components>
  <timeWithUTCtimezone>{$timeWithUTCtimezone}</timeWithUTCtimezone>
  <timeWithTimeZoneAdded>{ xf:remove-timezone-from-time(xs:time($timeWithUTCtimezone)) }
  </timeWithTimeZoneAdded>
</components>
```

The preceding query produces the following result:

```
<components>
  <timeWithUTCtimezone>12:00:00Z</timeWithUTCtimezone>
  <timeWithTimeZoneAdded>05:00:00</timeWithTimeZoneAdded>
</components>
```

Related Topics

W3C remove-timezone-from-time function description.

W3C time data type description.

W3C dayTimeDuration operator description.

xf:current-dateTime

Gets the current date and time.

Signatures

xf:current-dateTime() > xs:dateTime?

Returns

Returns the current date and time as a dateTime value.

All invocations of this function in a single outermost XQuery or XPATH expression will report the same date and time.

Examples

Invoke Once

The following simple example query, calls the current-dateTime function a single time:

```
<current>{xf:current-dateTime()}</current>
```

If the current date is November 11, 2002 and the current time is 4:23, the preceding XQuery generates the following result:

```
<current>2002-11-11T04:23:14.9030000</current>
```

Invoke Multiple Times

The following example query, invokes the current-dateTime function twice:

```
<current>
  <first>{xf:current-dateTime()}</first>
  <second>{xf:current-dateTime()}</second>
</current>
```

If the current date is November 11, 2002 and the current time is 4:23, the preceding XQuery generates the following result:

```
<current>
  <first>2002-11-11T04:23:34.5430000</first>
  <second>2002-11-11T04:23:34.5430000</second>
</current>
```

Note: The same exact time (even seconds) is returned for both invocations of the function.

Related Topics

W3C current-dateTime function description.

W3C dateTime data type description.

xf:current-date

Gets the current date.

Signatures

xf:current-date() > xs:date?

Returns

Returns the current date as a date value.

All invocation of this function in a single outermost XQuery or XPATH expression will report the same date.

Examples

The following example query, invokes the current-date function:

```
<current>{xf:current-date()}</current>
```

If the current date is November 11, 2002 and the current time is 4:23, The preceding query generates the following result:

```
<current>2002-11-11</current>
```

Related Topics

W3C current-date function description.

W3C date data type description.

xf:current-time

Gets the current time.

Signatures

xf:current-time() > xs:time?

Returns

Returns the current date and time as a time value.

All invocation of this function in a single outermost XQuery or XPATH expression will report the same time.

Examples

Invoke Once

The following example query, invokes the current-time function a single time:

```
<current>{xf:current-time()}</current>
```

If the current date is November 11, 2002 and the current time is 4:23, The preceding query generates the following result:

```
<current>04:23:14.9030000</current>
```

Invoke Multiple Times

The following example query, invokes the current-time function twice:

```
<current>
<first>{xf:current-time()}</first>
<second>{xf:current-time()}</second>
</current>
```

If the current date is November 11, 2002 and the current time is 4:23, The preceding query generates the following result:

```
<current>
<first>04:23:34.5430000</first>
<second>04:23:34.5430000</second>
</current>
```

Note: The same exact time (even seconds) is returned for both invocations of the function.

Related Topics

W3C current-time function description.

W3C time data type description.

XQuery Duration Functions Reference

This section provides descriptions of the XQuery duration functions available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery functions. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the *XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002* available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the duration functions available from the mapper functionality:

- `xf:yearMonthDuration`
- `xf:dayTimeDuration`
- `xf:get-years-from-yearMonthDuration`
- `xf:get-months-from-yearMonthDuration`
- `xf:get-days-from-dayTimeDuration`
- `xf:get-hours-from-dayTimeDuration`
- `xf:get-minutes-from-dayTimeDuration`
- `xf:get-seconds-from-dayTimeDuration`
- `xf:get-yearMonthDuration-from-dateTimes`
- `xf:get-dayTimeDuration-from-dateTimes`

xf:yearMonthDuration

Converts \$string-var (a string in the yearMonthDuration format) to the yearMonthDuration data type.

If \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:yearMonthDuration(xs:string \$string-var) > xf:yearMonthDuration

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents a string with the duration specified with one of the following formats: <ul style="list-style-type: none">• PyYmM• -PyYmM Note: Either one or both of the following substrings: yY or mM must be specified. For example, the following are valid durations: P10Y, -P11M, or P6Y4M.
		- Duration is a negative amount of time. Note: If - is not specified, duration is a positive amount of time.
		P The start of a duration string. The P must always be specified.
		yY y number of years in the duration.
		y number of years in the duration.
		mM m number of months in the duration.
		M months are specified in the duration.

Returns

Returns a duration of time as a yearMonthDuration value.

Examples

yearMonthDuration with Year and Month

Invoking yearMonthDuration("P1Y2M") returns a yearMonthDuration value corresponding to 1 year and 2 months, as shown in the following example query:

```
let $mydur := xf:yearMonthDuration("P1Y2M")
return
<components>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
```

XQuery Reference

```
<months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</components>
```

The preceding query, generates the following XML result:

```
<components>
  <years>1</years>
  <months>2</months>
</components>
```

yearMonthDuration with Year

Invoking `yearMonthDuration("P9Y")` returns a `yearMonthDuration` value corresponding to 9 years, as shown in the following example query:

```
let $mydur := xf:yearMonthDuration("P9Y")
return
<components>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
  <months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</components>
```

The preceding query, generates the following XML result:

```
<components>
  <years>9</years>
  <months>0</months>
</components>
```

yearMonthDuration with Negative Month

Invoking `yearMonthDuration("-P10M")` returns a `yearMonthDuration` value corresponding to a negative 10 months, as shown in the following example query:

```
let $mydur := xf:yearMonthDuration("-P10M")
return
<components>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
  <months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</components>
```

The preceding query, generates the following XML result:

```
<components>
  <years>0</years>
  <months>-10</months>
</components>
```

Related Topics

W3C `yearMonthDuration` data type description.

xf:dayTimeDuration

Converts \$string-var (a string in the dayTimeDuration format) to the dayTimeDuration data type.

If \$string-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:dayTimeDuration(xs:string \$string-var) > xf:dayTimeDuration

Arguments

Data Type	Argument	Description
xs:string	\$string-var	Represents a string with the duration specified with one of the following formats: <ul style="list-style-type: none">• <i>dDT<h>h</h>iM<i>s</i>S</i>• <i>-dDT<h>h</h>iM<i>s</i>S</i> Note: Specifying the number days, hours, minutes, or seconds is optional any of the substrings: <i>dD</i> , <i>hH</i> , <i>iM</i> , or <i>sS</i> do not have to be specified. For example, the following are valid durations: P10Y, P1D, -P11M, -P10Y7D, P2YT5H, or P6YT5H10S.
		<div>-</div> Duration is a negative amount of time. Note: If - is not specified, duration is a positive amount of time.
		PThe start of a duration string. The P must always be specified.
		dD <i>d</i> number of days in the duration.
		D <i>D</i> days are specified in the duration.
		TThe start of the time part of the duration string. The T must be specified if any minutes, hours, or seconds (<i>hH</i> , <i>iM</i> , or <i>sS</i>) are specified.
		hH <i>h</i> number of hours in the duration.
		H <i>H</i> years are specified in the duration.
		iM <i>i</i> number of minutes in the duration.
		M <i>M</i> minutes are specified in the duration.
		sS <i>s</i> number of seconds in the duration.
		S <i>S</i> seconds are specified in the duration.

Returns

Returns a duration of time as a dayTimeDuration value.

Examples

dayTimeDuration with All Components

Invoking `dayTimeDuration("P4DT9H8M20S")` returns a `dayTimeDuration` value corresponding to 4 days, 9 hours, 8 minutes, and 20 seconds, as shown in the following example query:

```
let $mydur := xf:dayTimeDuration("P4DT9H8M20S")
return
<components>
  <days>{xf:get-days-from-dayTimeDuration($mydur)}</days>
  <hours>{xf:get-hours-from-dayTimeDuration($mydur)}</hours>
  <minutes>{xf:get-minutes-from-dayTimeDuration($mydur)}</minutes>
  <seconds>{xf:get-seconds-from-dayTimeDuration($mydur)}</seconds>
</components>
```

The preceding query, generates the following XML result:

```
<components>
  <days>4</days>
  <hours>9</hours>
  <minutes>8</minutes>
  <seconds>20</seconds>
</components>
```

dayTimeDuration with Just Hours and Seconds

Invoking `dayTimeDuration("PT2H20S")` returns a `dayTimeDuration` value corresponding to 2 hours and 20 seconds, as shown in the following example query:

```
let $mydur := xf:dayTimeDuration("PT2H20S")
return
<components>
  <days>{xf:get-days-from-dayTimeDuration($mydur)}</days>
  <hours>{xf:get-hours-from-dayTimeDuration($mydur)}</hours>
  <minutes>{xf:get-minutes-from-dayTimeDuration($mydur)}</minutes>
  <seconds>{xf:get-seconds-from-dayTimeDuration($mydur)}</seconds>
</components>
```

The preceding query, generates the following XML result:

```
<components>
  <days>0</days>
  <hours>2</hours>
  <minutes>0</minutes>
  <seconds>20</seconds>
</components>
```

dayTimeDuration with Just Negative Days

Invoking `dayTimeDuration("-P10D")` returns a `dayTimeDuration` value corresponding to negative 10 days, as shown in the following example query:

```
let $mydur := xf:dayTimeDuration("-P10D")
return
<components>
  <days>{xf:get-days-from-dayTimeDuration($mydur)}</days>
  <hours>{xf:get-hours-from-dayTimeDuration($mydur)}</hours>
  <minutes>{xf:get-minutes-from-dayTimeDuration($mydur)}</minutes>
```

XQuery Reference

```
<seconds>{xf:get-seconds-from-dayTimeDuration($mydur)}</seconds>  
</components>
```

The preceding query, generates the following XML result:

```
<components>  
  <days>-10</days>  
  <hours>0</hours>  
  <minutes>0</minutes>  
  <seconds>0</seconds>  
</components>
```

Related Topics

W3C dayTimeDuration data type description.

xf:get-years-from-yearMonthDuration

Extracts the number of years from the years component of \$yearMonthDuration-var.

If the value of \$yearMonthDuration-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-years-from-yearMonthDuration(xf:yearMonthDuration? \$yearMonthDuration-var) > xs:integer?

Arguments

Data Type	Argument	Description
xf:yearMonthDuration?	\$yearMonthDuration-var	Contains a representation of a time duration which can contain years and months.

Returns

Returns the number of year as an integer from years component of \$yearMonthDuration-var.

Examples

Get Years from yearMonthDuration with Years and Months

```
<years>{xf:get-years-from-yearMonthDuration(xf:yearMonthDuration("P2Y13M"))}</years>
```

The preceding query generates the following result:

```
<years>2</years>
```

Note: Even though 13 months is specified in the month component (13M) of the yearMonthDuration creation string (adding up to an additional year), only 2 years is returned as originally specified by the years component (2Y).

Get Years from yearMonthDuration with Just Negative Years

```
<years>{xf:get-years-from-yearMonthDuration(xf:yearMonthDuration("-P5Y"))}</years>
```

The preceding query generates the following result:

```
<years>-5</years>
```

Get Years from yearMonthDuration with Just Months

```
<years>{xf:get-years-from-yearMonthDuration(xf:yearMonthDuration("P10M"))}</years>
```

The preceding query generates the following result:

`<years>0</years>`

Related Topics

W3C `get-years-from-yearMonthDuration` function description.

W3C `yearMonthDuration` data type description.

xf:get-months-from-yearMonthDuration

Extracts the number of months from the months component of \$yearMonthDuration-var.

If the value of \$yearMonthDuration-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-months-from-yearMonthDuration(xf:yearMonthDuration? \$yearMonthDuration-var) > xs:integer?

Arguments

Data Type	Argument	Description
xf:yearMonthDuration?	\$yearMonthDuration-var	Contains a representation of a time duration which can contain years and months.

Returns

Returns the number of months as an integer from months component of \$yearMonthDuration-var.

Examples

Get Months from yearMonthDuration with Years and Months

```
<months>{xf:get-months-from-yearMonthDuration(xf:yearMonthDuration("P2Y10M"))}</months>
```

The preceding query generates the following result:

```
<months>10</months>
```

Get Months from yearMonthDuration with Just Negative Months

```
<months>{xf:get-months-from-yearMonthDuration(xf:yearMonthDuration("-P5M"))}</months>
```

The preceding query generates the following result:

```
<months>-5</months>
```

Related Topics

W3C get-months-from-yearMonthDuration function description.

W3C yearMonthDuration data type description.

xf:get-days-from-dayTimeDuration

Extracts the number of days from the days component of \$dayTimeDuration-var.

If the value of \$dayTimeDuration-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-days-from-dayTimeDuration(xf:dayTimeDuration? \$dayTimeDuration-var) > xs:integer?

Arguments

Data Type	Argument	Description
xf:dayTimeDuration?	\$dayTimeDuration-var	Contains a representation of a time duration which can contain days, hours, minutes, and seconds.

Returns

Returns the number of days as an integer from days component of \$dayTimeDuration-var.

Examples

Get Days from dayTimeDuration with All Components

```
<days>{xf:get-days-from-dayTimeDuration(xf:dayTimeDuration("P7DT25H8M20S"))}</days>
```

The preceding query generates the following result:

```
<days>7</days>
```

Note: Even though 25 hours is specified in the hours component (25H) of the dayTimeDuration creation string (adding up to an additional day), only 7 days are returned as originally specified by the days component (7D).

Get Days from dayTimeDuration with Just Negative Days

```
<days>{xf:get-days-from-dayTimeDuration(xf:dayTimeDuration("-PT4D"))}</days>
```

The preceding query generates the following result:

```
<days>-4</days>
```

Get Days from dayTimeDuration with No Days

```
<days>{xf:get-days-from-dayTimeDuration(xf:dayTimeDuration("PT2H20S"))}</days>
```

The preceding query generates the following result:

`<days>0</days>`

Related Topics

W3C `get-days-from-dayTimeDuration` function description.

W3C `dayTimeDuration` data type description.

xf:get-hours-from-dayTimeDuration

Extracts the number of hours from the hours component of \$dayTimeDuration-var.

If the value of \$dayTimeDuration-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-hours-from-dayTimeDuration(xf:dayTimeDuration? \$dayTimeDuration-var) > xs:integer?

Arguments

Data Type	Argument	Description
xf:dayTimeDuration?	\$dayTimeDuration-var	Contains a representation of a time duration which can contain days, hours, minutes, and seconds.

Returns

Returns the number as hours as an integer from hours component of \$dayTimeDuration-var.

Examples

Get Hours from dayTimeDuration with All Components

```
<hours>{xf:get-hours-from-dayTimeDuration(xf:dayTimeDuration("P7DT9H65M20S"))}</hours>
```

The preceding query generates the following result:

```
<hours>9</hours>
```

Note: Even though 65 minutes is specified in the minutes component (65M) of the dayTimeDuration creation string (adding up to an additional hour), only 9 hours are returned as originally specified by the hours component (9H).

Get Hours from dayTimeDuration with Just Negative Hours

```
<hours>{xf:get-hours-from-dayTimeDuration(xf:dayTimeDuration("-PT3H"))}</hours>
```

The preceding query generates the following result:

```
<hours>-3</hours>
```

Get Hours from dayTimeDuration with No Hours

```
<hours>{xf:get-hours-from-dayTimeDuration(xf:dayTimeDuration("P2DT20S"))}</hours>
```

The preceding query generates the following result:

```
<hours>0</hours>
```

Related Topics

W3C `get-hours-from-dayTimeDuration` function description.

W3C `dayTimeDuration` data type description.

xf:get-minutes-from-dayTimeDuration

Extracts the number of minutes from the minutes component of \$dayTimeDuration-var.

If the value of \$dayTimeDuration-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-minutes-from-dayTimeDuration(xf:dayTimeDuration? \$dayTimeDuration-var) > xs:integer?

Arguments

Data Type	Argument	Description
xf:dayTimeDuration	\$dayTimeDuration-var	Contains a representation of a time duration which can contain days, hours, minutes, and seconds.

Returns

Returns the number as minutes as an integer from minutes component of \$dayTimeDuration-var.

Examples

Get Minutes from dayTimeDuration with All Components

```
<minutes>{xf:get-minutes-from-dayTimeDuration(xf:dayTimeDuration("P7DT9H12M65S"))}</minutes>
```

The preceding query generates the following result:

```
<minutes>12</minutes>
```

Note: Even though 65 seconds is specified in the seconds component (65S) of the dayTimeDuration creation string (adding up to an additional minute), only 12 minutes are returned as originally specified by the minutes component (12M).

Get Minutes from dayTimeDuration with Just Negative Minutes

```
<minutes>{xf:get-minutes-from-dayTimeDuration(xf:dayTimeDuration("-PT3M"))}</minutes>
```

The preceding query generates the following result:

```
<minutes>-3</minutes>
```

Get Minutes from dayTimeDuration with No Minutes

```
<minutes>{xf:get-minutes-from-dayTimeDuration(xf:dayTimeDuration("P2DT20S"))}</minutes>
```

The preceding query generates the following result:

```
<minutes>0</minutes>
```

Related Topics

W3C `get-minutes-from-dayTimeDuration` function description.

W3C `dayTimeDuration` data type description.

xf:get-seconds-from-dayTimeDuration

Extracts the number of seconds from the seconds component of \$dayTimeDuration-var.

If the value of \$dayTimeDuration-var is the empty sequence, the empty sequence is returned. The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

Signatures

xf:get-seconds-from-dayTimeDuration(xf:dayTimeDuration? \$dayTimeDuration-var) > xs:decimal?

Arguments

Data Type	Argument	Description
xf:dayTimeDuration?	\$dayTimeDuration-var	Contains a representation of a time duration which can contain days, hours, minutes, and seconds.

Returns

Returns the number as seconds as an decimal from seconds component of \$dayTimeDuration-var.

Examples

Get Seconds from dayTimeDuration with All Components

```
<seconds>{xf:get-seconds-from-dayTimeDuration(xf:dayTimeDuration("P7DT9H12M14S"))}</seconds>
```

The preceding query generates the following result:

```
<seconds>14</seconds>
```

Get Seconds from dayTimeDuration with Just Negative Seconds

```
<seconds>{xf:get-seconds-from-dayTimeDuration(xf:dayTimeDuration("-PT7S"))}</seconds>
```

The preceding query generates the following result:

```
<seconds>-7</seconds>
```

Get Seconds from dayTimeDuration with No Seconds

```
<seconds>{xf:get-seconds-from-dayTimeDuration(xf:dayTimeDuration("P2DT6M"))}</seconds>
```

The preceding query generates the following result:

```
<seconds>0</seconds>
```

Related Topics

W3C `get-seconds-from-dayTimeDuration` function description.

W3C `dayTimeDuration` data type description.

xf:get-yearMonthDuration-from-dateTimes

Computes the time difference between \$dateTime-var1 and \$dateTime-var2 and returns it as a yearMonthDuration value.

If the value of \$date-var1 follows in time the value of \$date-var2, then the returned value is a negative duration. (The date specified in \$dateTime-var1 comes before the date specified in \$dateTime-var2.) To learn more, see the following Negative Difference example.

The time difference between the values of two dateTime arguments could include years, months, minutes, and seconds but a yearMonthDuration value can only contain years and months, so the time difference is rounded to nearest month. If the time difference between the two dateTime arguments is greater than or equal to 15.5 days (15 days and 12 hours), the month duration is rounded up. To learn more, see the following Round Up and Not Enough to Round Up examples.

If the value of \$date-var1 or \$date-var2 is the empty sequence, the following error is displayed:

Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e

Signatures

xf:get-yearMonthDuration-from-dateTimes(xs:dateTime \$dateTime-var1, xs:dateTime \$dateTime-var2)
> xf:yearMonthDuration

Arguments

Data Type	Argument	Description
xs:dateTime	\$dateTime-var1	Contains a representation of the date and time.
xs:dateTime	\$dateTime-var2	Contains a representation of the date and time.

Returns

Returns the time difference between \$dateTime-var1 and \$dateTime-var2 as a yearMonthDuration value.

Returns a negative yearMonthDuration value if \$dateTime-var1 follows in time \$dateTime-var2. (The date specified in \$dateTime-var1 comes before the date specified in \$dateTime-var2.) To learn more, see the following Negative Difference example.

Examples

Positive Difference

The following example query returns a positive years and months duration because the date specified in \$datetime-var1 comes after the date specified in \$datetime-var2:

```
let $dateTime-var1 := xs:dateTime("2002-12-26T00:00:01")
return
let $dateTime-var2 := xs:dateTime("2001-11-26T00:00:01")
return
```

XQuery Reference

```
let $mydur := xf:get-yearMonthDuration-from-dateTimes($dateTime-var1, $dateTime-var2)
return
<result>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
  <months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</result>
```

The preceding query generates the following result:

```
<result>
  <years>1</years>
  <months>1</months>
</result>
```

Negative Difference

The following example query returns a negative years and month duration because the date specified in \$dateTime-var1 comes before the date specified in \$dateTime-var2:

```
let $dateTime-var1 := xs:dateTime("2001-11-26T00:00:01")
return
let $dateTime-var2 := xs:dateTime("2002-12-26T00:00:01")
return
let $mydur := xf:get-yearMonthDuration-from-dateTimes($dateTime-var1, $dateTime-var2)
return
<result>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
  <months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</result>
```

The preceding query generates the following result:

```
<result>
  <years>-1</years>
  <months>-1</months>
</result>
```

Round Up

The following example query returns a 1 month yearMonthDuration because there are 15.5 days (15 days and 12 hours) between \$dateTime-var1 and \$dateTime-var2:

```
let $dateTime-var1 := xs:dateTime("2002-11-16T12:00:00")
return
let $dateTime-var2 := xs:dateTime("2002-11-01T00:00:00")
return
let $mydur := xf:get-yearMonthDuration-from-dateTimes($dateTime-var1, $dateTime-var2)
return
<result>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
  <months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</result>
```

The preceding query generates the following result:

```
<result>
  <years>0</years>
```

```

    <months>1</months>
</result>

```

Not Enough to Round Up

The following example query returns a 0 months yearMonthDuration because there are only 15 days, 11 hours, 59 minutes, and 59 seconds (1 second from 15.5 days) between \$dateTime-var1 and \$dateTime-var2:

```

let $dateTime-var1 := xs:dateTime("2002-11-16T11:59:59")
return
let $dateTime-var2 := xs:dateTime("2002-11-01T00:00:00")
return
let $mydur := xf:get-yearMonthDuration-from-dateTimes($dateTime-var1, $dateTime-var2)
return
<result>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
  <months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</result>

```

The preceding query generates the following result:

```

<result>
  <years>0</years>
  <months>0</months>
</result>

```

Related Topics

W3C `get-yearMonthDuration-from-dayTimes` function description.

W3C `yearMonthDuration` data type description.

W3C `dateTime` data type description.

xf:get-dayTimeDuration-from-dateTimes

Computes the time difference between \$dateTime-var1 and \$dateTime-var2 and returns it as a dayTimeDuration value.

If the value of \$date-var1 follows in time the value of \$date-var2, then the returned value is a negative duration.

If the value of \$date-var1 or \$date-var2 is the empty sequence, the following error is displayed:

Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method: Type e

Signatures

xf:get-dayTimeDuration-from-dateTimes(xs:dateTime \$dateTime-var1, xs:dateTime \$dateTime-var2) >
xf:dayTimeDuration

Arguments

Data Type	Argument	Description
xs:dateTime	\$dateTime-var1	Contains a representation of the date and time.
xs:dateTime	\$dateTime-var2	Contains a representation of the date and time.

Returns

Returns the time difference between \$dateTime-var1 and \$dateTime-var2 as a dayTimeDuration value.

Returns a negative dayTimeDuration value if \$dateTime-var1 follows in time \$dateTime-var2. (The date specified in \$dateTime-var1 comes before the date specified in \$dateTime-var2.) To learn more, see the following Negative Difference example.

Examples

Positive Difference

The following example query returns a positive days, hours, minutes, and seconds duration because the date specified in \$datetime-var1 comes after the date specified in \$datetime-var2:

```
let $dateTime-var1 := xs:dateTime("2002-12-26T01:01:01")
return
let $dateTime-var2 := xs:dateTime("2001-11-26T00:00:00")
return
let $mydur := xf:get-dayTimeDuration-from-dateTimes($dateTime-var1, $dateTime-var2)
return
<result>
  <days>{xf:get-days-from-dayTimeDuration($mydur)}</days>
  <hours>{xf:get-hours-from-dayTimeDuration($mydur)}</hours>
  <minutes>{xf:get-minutes-from-dayTimeDuration($mydur)}</minutes>
  <seconds>{xf:get-seconds-from-dayTimeDuration($mydur)}</seconds>
</result>
```

The preceding query generates the following result:

```
<result>
  <days>395</days>
  <hours>1</hours>
  <minutes>1</minutes>
  <seconds>1</seconds>
</result>
```

Negative Difference

The following example query returns a negative days, hours, minutes, and seconds duration because the date specified in \$dateTime-var1 comes before the date specified in \$dateTime-var2:

```
let $dateTime-var1 := xs:dateTime("2001-11-26T00:00:00")
return
let $dateTime-var2 := xs:dateTime("2002-12-26T01:01:01")
return
let $mydur := xf:get-dayTimeDuration-from-dateTimes($dateTime-var1, $dateTime-var2)
return
<result>
  <days>{xf:get-days-from-dayTimeDuration($mydur)}</days>
  <hours>{xf:get-hours-from-dayTimeDuration($mydur)}</hours>
  <minutes>{xf:get-minutes-from-dayTimeDuration($mydur)}</minutes>
  <seconds>{xf:get-seconds-from-dayTimeDuration($mydur)}</seconds>
</result>
```

The preceding query generates the following result:

```
<result>
  <days>-395</days>
  <hours>-1</hours>
  <minutes>-1</minutes>
  <seconds>-1</seconds>
</result>
```

Related Topics

W3C `get-dayTimeDuration-from-dateTimes` function description.

W3C `dayTimeDuration` data type description.

W3C `dateTime` data type description.

XQuery Numeric Operators Reference

This section provides descriptions of the XQuery numeric operators available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery operators. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the *XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002* available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the numeric operators available from the mapper functionality:

- op:decimal-add
- op:float-add
- op:double-add
- op:decimal-subtract
- op:float-subtract
- op:double-subtract
- op:decimal-multiply
- op:float-multiply
- op:double-multiply
- op:decimal-divide
- op:float-divide
- op:double-divide
- op:numeric-integer-divide
- op:decimal-mod
- op:float-mod
- op:double-mod

op:decimal-add

Add \$decimal-var1 to \$decimal-var2. The + operator invokes the decimal-add operator.

Signatures

op:decimal-add(xs:decimal \$decimal-var1, xs:decimal \$decimal-var2) > xs:decimal

Arguments

Data Type	Argument	Description
xs:decimal	\$decimal-var1	Represents a decimal number, for example: 1.1.
xs:decimal	\$decimal-var2	Represents a decimal number, for example: 1.1.

Returns

Returns the decimal value of adding \$decimal-var1 to \$decimal-var2.

Examples

Simple

Invoking decimal-add("1.1","2.2") returns the decimal value 3.3 as shown in the following example query:

```
<decimal-add>{op:decimal-add("1.1","2.2")}</decimal-add>
```

The preceding query generates the following result:

```
<decimal-add>3.3</decimal-add>
```

Related Topics

W3C decimal data type description.

W3C numeric-add operator description.

op:float-add

Add \$float-var1 to \$float-var2. The + operator invokes the float-add operator.

Signatures

op:float-add(xs:float \$float-var1, xs:float \$float-var2) > xs:float

Arguments

Data Type	Argument	Description
xs:float	\$float-var1	Represents a 32 bit floating point number, for example: 1.1.
xs:float	\$float-var2	Represents a 32 bit floating point number, for example: 1.1.

Returns

Returns the float value of adding \$float-var1 to \$float-var2.

Examples

Simple

Invoking float-add("1.0","2.0") returns the floating point value: 3.0 as shown in the following example query:

```
<float-add>{op:float-add("1.0","2.0")}</float-add>
```

The preceding query generates the following result:

```
<float-add>3.0</float-add>
```

Related Topics

W3C float data type description.

W3C numeric-add operator description.

op:double-add

Add \$double-var1 to \$double-var2. The + operator invokes the double-add operator.

Signatures

op:double-add(xs:double \$double-var1, xs:double \$double-var2) > xs:double

Arguments

Data Type	Argument	Description
xs:double	\$double-var1	Represents a double precision (64 bit) floating point number, for example: 1.1.
xs:double	\$double-var2	Represents a double precision (64 bit) floating point number, for example: 1.1.

Returns

Returns the double value of adding \$double-var1 to \$double-var2.

Examples

Simple

Invoking double-add("1.0","2.0") returns the double precision floating point value: 3.0 as shown in the following example query:

```
<double-add>{op:double-add("1.0","2.0")}</double-add>
```

The preceding query generates the following result:

```
<double-add>3.0</double-add>
```

Related Topics

W3C double data type description.

W3C numeric-add operator description.

op:decimal-subtract

Subtracts \$decimal-var2 from \$decimal-var1. The – operator invokes the decimal-subtract operator.

Signatures

op:decimal-subtract(xs:decimal \$decimal-var1, xs:decimal \$decimal-var2) > xs:decimal

Arguments

Data Type	Argument	Description
xs:decimal	\$decimal-var1	Represents a decimal number, for example: 1.1.
xs:decimal	\$decimal-var2	Represents a decimal number, for example: 1.1.

Returns

Returns the decimal value of subtracting \$decimal-var2 from \$decimal-var1.

Examples

Simple

Invoking decimal-subtract("2.2","1.1") returns the decimal value 1.1 as shown in the following example query:

```
<decimal-subtract>{op:decimal-subtract("2.2","1.1")}</decimal-subtract>
```

The preceding query generates the following result:

```
<decimal-subtract>1.1</decimal-subtract>
```

Related Topics

W3C decimal data type description.

W3C numeric-subtract operator description.

op:float-subtract

Subtracts \$float-var2 from \$float-var1. The – operator invokes the float-subtract operator.

Signatures

op:float-subtract(xs:float \$float-var1, xs:float \$float-var2) > xs:float

Arguments

Data Type	Argument	Description
xs:float	\$float-var1	Represents a 32 bit floating point number, for example: 1.1.
xs:float	\$float-var2	Represents a 32 bit floating point number, for example: 1.1.

Returns

Returns the floating point value of subtracting \$float-var2 from \$float-var1.

Examples

Simple

Invoking float-subtract("2.2","1.1") returns the floating point value: 1.1 as shown in the following example query:

```
<float-subtract>{op:float-subtract("2.2","1.1")}</float-subtract>
```

The preceding query generates the following result:

```
<float-subtract>1.1</float-subtract>
```

Related Topics

W3C float data type description.

W3C numeric-subtract operator description.

op:double-subtract

Subtracts \$double-var2 from \$double-var1. The – operator invokes the double-subtract operator.

Signatures

op:double-subtract(xs:double \$double-var1, xs:double \$double-var2) > xs:double

Arguments

Data Type	Argument	Description
xs:double	\$double-var1	Represents a double precision (64 bit) floating point number, for example: 1.1.
xs:double	\$double-var2	Represents a double precision (64 bit) floating point number, for example: 1.1.

Returns

Returns the double precision (64 bit) floating point value of subtracting \$double-var2 from \$double-var1.

Examples

Simple

Invoking double-subtract("2.2","1.1") returns the double precision (64 bit) floating point value: 1.1 as shown in the following example query:

```
<double-subtract>{op:double-subtract("2.2","1.1")}</double-subtract>
```

The preceding query generates the following result:

```
<double-subtract>1.1</double-subtract>
```

Related Topics

W3C double data type description.

W3C numeric-subtract operator description.

op:decimal-multiply

Multiplies \$decimal-var1 by \$decimal-var2. The * operator invokes the decimal-multiply operator.

Signatures

op:decimal-multiply(xs:decimal \$decimal-var1, xs:decimal \$decimal-var2) > xs:decimal

Arguments

Data Type	Argument	Description
xs:decimal	\$decimal-var1	Represents a decimal number, for example: 1.1.
xs:decimal	\$decimal-var2	Represents a decimal number, for example: 1.1.

Returns

Returns the decimal value of multiplying \$decimal-var1 by \$decimal-var2.

Examples

Simple

Invoking decimal-multiply("2.0","1.0") returns the decimal value: 2 as shown in the following example query:

```
<decimal-multiply>{op:decimal-multiply("2.0","1.0")}</decimal-multiply>
```

The preceding query generates the following result:

```
<decimal-multiply>2</decimal-multiply>
```

Related Topics

W3C decimal data type description.

W3C numeric-multiply operator description.

op:float-multiply

Multiplies \$float-var1 by \$float-var2. The * operator invokes the float-multiply operator.

Signatures

op:float-multiply(xs:float \$float-var1, xs:float \$float-var2) > xs:float

Arguments

Data Type	Argument	Description
xs:float	\$float-var1	Represents a 32 bit floating point number, for example: 1.1.
xs:float	\$float-var2	Represents a 32 bit floating point number, for example: 1.1.

Returns

Returns the 32 bit floating point value of multiplying \$float-var1 by \$float-var2.

Examples

Simple

Invoking float-multiply("2.0","3.0") returns the floating point value: 6.0 as shown in the following example query:

```
<float-multiply>{op:float-multiply("2.0","3.0")}</float-multiply>
```

The preceding query generates the following result:

```
<float-multiply>6.0</float-multiply>
```

Related Topics

W3C float data type description.

W3C numeric-multiply operator description.

op:double-multiply

Multiplies \$double-var1 by \$double-var2. The * operator invokes the double-multiply operator.

Signatures

op:double-multiply(xs:double \$double-var1, xs:double \$double-var2) > xs:double

Arguments

Data Type	Argument	Description
xs:double	\$double-var1	Represents a double precision (64 bit) floating point number, for example: 1.1.
xs:double	\$double-var2	Represents a double precision (64 bit) floating point number, for example: 1.1.

Returns

Returns the double precision (64 bit) floating point value of multiplying \$double-var1 by \$double-var2.

Examples

Simple

Invoking double-multiply("2.0","3.0") returns the double precision floating point value: 6.0 as shown in the following example query:

```
<double-multiply>{op:double-multiply("2.0","3.0")}</double-multiply>
```

The preceding query generates the following result:

```
<double-multiply>6.0</double-multiply>
```

Related Topics

W3C double data type description.

W3C numeric-multiply operator description.

op:decimal-divide

Divides \$decimal-var1 by \$decimal-var2. The div operator invokes the decimal-divide operator.

If the value of \$decimal-var2 is equal to zero, the TransformException exception is raised with the RT_DIV_ZERO fault code. The following error message is displayed in the mapper:

```
Error occurred while executing XQuery: division by zero
```

Signatures

op:decimal-divide(xs:decimal \$decimal-var1, xs:decimal \$decimal-var2) > xs:decimal

Arguments

Data Type	Argument	Description
xs:decimal	\$decimal-var1	Represents a decimal number, for example: 1.1.
xs:decimal	\$decimal-var2	Represents a decimal number, for example: 1.1.

Returns

Returns the decimal value of dividing \$decimal-var1 by \$decimal-var2.

Examples

Simple

Invoking decimal-divide("2.2","1.1") returns the decimal value 2 as shown in the following example query:

```
<decimal-divide>{op:decimal-divide("2.2","1.1")}</decimal-divide>
```

The preceding query generates the following result:

```
<decimal-divide>2.0</decimal-divide>
```

Error Divide by Zero

Invoking decimal-divide("2.2","0")throws the TransformException exception with the RT_DIV_ZERO fault code as shown in the following example query:

```
<decimal-divide>{op:decimal-divide("2.2","0")}</decimal-divide>
```

The following error message is displayed in the mapper:

```
Error occurred while executing XQuery: division by zero
```

Related Topics

W3C decimal data type description.

W3C numeric-divide operator description.

op:float-divide

Divides \$float-var1 by \$float-var2. The div operator invokes the float-divide operator .

If the value of \$float-var2 is equal to zero, the value of Infinity is returned. To learn more see [Divide by Zero](#).

Signatures

op:float-divide(xs:float \$float-var1, xs:float \$float-var2) > xs:float

Arguments

Data Type	Argument	Description
xs:float	\$float-var1	Represents a 32 bit floating point number, for example: 1.1.
xs:float	\$float-var2	Represents a 32 bit floating point number, for example: 1.1.

Returns

Returns the floating point value of dividing \$float-var1 by \$float-var2.

Examples

Simple

Invoking float-divide("4.4","2.2") returns the floating point value of 2.0 as shown in the following example query:

```
<float-divide>{op:float-divide("4.4","2.2")}</float-divide>
```

The preceding query generates the following result:

```
<float-divide>2.0</float-divide>
```

Divide by Zero

Invoking float-divide("2.2","0") returns the Infinity value as shown in the following example query:

```
<float-divide>{op:float-divide("2.2","0")}</float-divide>
```

The preceding query generates the following result:

```
<float-divide>Infinity</float-divide>
```

To learn more see the [IEEE Standard for Binary Floating-Point Arithmetic](#).

Related Topics

W3C float data type description.

W3C numeric-divide operator description.

IEEE Standard for Binary Floating-Point Arithmetic.

op:double-divide

Divides \$double-var1 by \$double-var2. The div operator invokes the double-divide operator.

If the value of \$double-var2 is equal to zero, the value of Infinity is returned. To learn more see [Divide by Zero](#).

Signatures

op:double-divide(xs:double \$double-var1, xs:double \$double-var2) > xs:double

Arguments

Data Type	Argument	Description
xs:double	\$double-var1	Represents a double precision (64 bit) floating point number, for example: 1.1.
xs:double	\$double-var2	Represents a double precision (64 bit) floating point number, for example: 1.1.

Returns

Returns the double precision floating point value of dividing \$double-var1 by \$double-var2.

Examples

Simple

Invoking double-divide("4.4","2.2") returns the double precision floating point value of 2.0 as shown in the following example query:

```
<double-divide>{op:double-divide("4.4","2.2")}</double-divide>
```

The preceding query generates the following result:

```
<double-divide>2.0</double-divide>
```

Divide by Zero

Invoking double-divide("2.2","0") returns the Infinity value as shown in the following example query:

```
<double-divide>{op:double-divide("2.2","0")}</double-divide>
```

The preceding query generates the following result:

```
<double-divide>Infinity</double-divide>
```

To learn more see the [IEEE Standard for Binary Floating-Point Arithmetic](#).

Related Topics

W3C double data type description.

W3C numeric-divide operator description.

IEEE Standard for Binary Floating-Point Arithmetic.

op:numeric-integer-divide

Divides \$integer-var1 by \$integer-var2 and returns the resulting integer. If \$integer-var2 is not divided evenly by \$integer-var1, the remainder is dropped no rounding occurs. The idiv operator is invoked by the numeric-integer-divide operator.

If the value of \$decimal-var2 is equal to zero, the TransformException exception is raised with the RT_DIV_ZERO fault code. The following error message is displayed in the mapper:

Error occurred while executing XQuery: division by zero

Signatures

op:numeric-integer-divide(xs:integer \$integer-var1, xs:integer \$double-var2) > xs:integer

Arguments

Data Type	Argument	Description
xs:integer	\$integer-var1	Represents a integer number, for example: 1.
xs:integer	\$integer-var2	Represents a integer number, for example: 1.

Returns

Returns the integer result of dividing \$integer-var1 by \$integer-var2. If \$integer-var2 is not divided evenly by \$integer-var1, the remainder is dropped no rounding occurs.

Examples

No Remainder

Invoking op:numeric-integer-divide("4","2") returns the integer: 2 as shown in the following example query:

```
<numeric-integer-divide>{
  op:numeric-integer-divide{"4","2"}
}</numeric-integer-divide>
```

The preceding query generates the following result:

```
<numeric-integer-divide>2</numeric-integer-divide>
```

Note: The integer: 4 is divided eventually by the integer: 2 with no remainder.

Throw Away Remainder

Invoking op:numeric-integer-divide("5","2") returns the integer: 2 as shown in the following example query:

```
<numeric-integer-divide>{
  op:numeric-integer-divide{"5","2"}
}</numeric-integer-divide>
```

The preceding query generates the following result:

```
<numeric-integer-divide>2</numeric-integer-divide>
```

Note: The remainder is discarded.

Error Divide by Zero

Invoking `op:numeric-integer-divide("2","0")` throws the `TransformException` exception with the `RT_DIV_ZERO` fault code as shown in the following example query:

```
<numeric-integer-divide>{  
    op:numeric-integer-divide{  
    }</numeric-integer-divide>  
}</numeric-integer-divide>
```

The following error message is displayed in the mapper:

```
Error occurred while executing XQuery: division by zero
```

Related Topics

W3C integer data type description.

W3C `numeric-integer-divide` operator description.

op:decimal-mod

Return the remainder of dividing \$decimal-var1 by \$decimal-var2. The mod operator invokes the decimal-mod operator.

If the value of \$decimal-var2 is equal to zero, the TransformException exception is raised with the RT_DIV_ZERO fault code. The following error message is displayed in the mapper:

```
Error occurred while executing XQuery: division by zero
```

Signatures

op:decimal-mod(xs:decimal \$decimal-var1, xs:decimal \$decimal-var2) > xs:decimal

Arguments

Data Type	Argument	Description
xs:decimal	\$decimal-var1	Represents a decimal number, for example: 1.1.
xs:decimal	\$decimal-var2	Represents a decimal number, for example: 1.1.

Returns

Returns the remainder of dividing \$decimal-var1 by \$decimal-var2.

Examples

Simple

Invoking decimal-mod("2.1","2.0") returns the decimal value .1 because 2.1 divided by 2.0 results in 1 with the remainder: .1. The remainder: .1 is returned as shown in the following example query:

```
<decimal-mod>{op:decimal-mod("2.1","2.0")}</decimal-mod>
```

The preceding query generates the following result:

```
<decimal-mod>0.1</decimal-mod>
```

Error Divide by Zero

Invoking decimal-mod("2.2","0")throws the TransformException exception with the RT_DIV_ZERO fault code as shown in the following example query:

```
<decimal-mod>{op:decimal-mod("2.2","0")}</decimal-mod>
```

The following error message is displayed in the mapper:

```
Error occurred while executing XQuery: division by zero
```


Related Topics

[W3C decimal data type description.](#)

[W3C numeric-mod operator description.](#)

op:float-mod

Return the remainder of dividing \$float-var1 by \$float-var2. The mod operator invokes the float-mod operator.

To learn more about using this operator with NaN, positive infinity, or negative infinity operands, see the W3C numeric-mod operator description.

Signatures

op:float-mod(xs:float \$float-var1, xs:float \$float-var2) > xs:float

Arguments

Data Type	Argument	Description
xs:float	\$float-var1	Represents a 32 bit floating point number, for example: 1.1.
xs:float	\$float-var2	Represents a 32 bit floating point number, for example: 1.1.

Returns

Returns the remainder of dividing \$float-var1 by \$float-var2.

Examples

Simple

Invoking float-mod("1.25","1.0") returns the floating point value: .25 because 1.25 divided by 1.0 results in 1 with the remainder: .25. The remainder: .25 is returned as shown in the following example query:

```
<float-mod>{op:float-mod("1.25","1.0")}</float-mod>
```

The preceding query generates the following result:

```
<float-mod>0.25</float-mod>
```

Divide by Zero

Invoking float-mod("2.2","0") returns a NaN (Not a Number) value as shown in the following example query:

```
<float-mod>{op:float-mod("2.2","0")}</float-mod>
```

The preceding query generates the following result:

```
<float-mod>NaN</float-mod>
```

Related Topics

W3C float data type description.

W3C numeric-mod operator description.

op:double-mod

Return the remainder of dividing \$double-var1 by \$double-var2. The mod operator invokes the double-mod operator.

To learn more about using this operator with NaN, positive infinity, or negative infinity operands, see the W3C numeric-mod operator description.

Signatures

op:double-mod(xs:double \$double-var1, xs:double \$double-var2) > xs:double

Arguments

Data Type	Argument	Description
xs:double	\$double-var1	Represents a double precision (64 bit) floating point number, for example: 1.1.
xs:double	\$double-var2	Represents a double precision (64 bit) floating point number, for example: 1.1.

Returns

Returns the remainder of dividing \$double-var1 by \$double-var2.

Examples

Simple

Invoking double-mod("1.25","1.0") returns the double precision (64 bit) floating point value: .25 because 1.25 divided by 1.0 results in 1 with the remainder: .25. The remainder: .25 is returned as shown in the following example query:

```
<double-mod>{op:double-mod("1.25","1.0")}</double-mod>
```

The preceding query generates the following result:

```
<double-mod>0.25</double-mod>
```

Divide by Zero

Invoking double-mod("2.2","0") returns a NaN value as shown in the following example query:

```
<double-mod>{op:double-mod("2.2","0")}</double-mod>
```

The preceding query generates the following result:

```
<double-mod>NaN</double-mod>
```

Related Topics

W3C double data type description.

W3C numeric-mod operator description.

XQuery Boolean Operators Reference

This section provides descriptions of the XQuery boolean operators available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery operators. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002 available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the boolean operators available from the mapper functionality:

- op:boolean-equal
- op:boolean-less-than
- op:boolean-greater-than

op:boolean-equal

If \$boolean-var1 has the same boolean value as \$boolean-var2, the boolean value: true is returned. If \$boolean-var1 does not have the same boolean value as \$boolean-var, the boolean value: false is returned. For example: op:boolean-equal(xf:true(), xf:false()) returns the boolean value: false.

This operator is equivalent to the eq operator with boolean values.

Signatures

op:boolean-equal(xs:boolean \$boolean-var1, xs:boolean \$boolean-var2) > xs:boolean

Arguments

Data Type	Argument	Description
xs:boolean	\$boolean-var1	Represents a boolean value.
xs:boolean	\$boolean-var2	Represents a boolean value.

Returns

Returns the boolean value: true if \$boolean-var1 is equal to the boolean value of \$boolean-var2.

Returns the boolean value: false if \$boolean-var1 is not equal to the boolean value of \$boolean-var2.

Examples

Not Equal Returns false

Invoking op:boolean-equal(xf:false(),xf:true()) returns the boolean value: false, as shown in the following example query:

```
<boolean-equal>{
  op:boolean-equal(xf:false(),xf:true())
}</boolean-equal>
```

The preceding query generates the following result:

```
<boolean-equal>false</boolean-equal>
```

Equal Returns true

Invoking op:boolean-equal(xf:false(),xf:false()) returns the boolean value: true, as shown in the following example query:

```
<boolean-equal>{
  op:boolean-equal(xf:false(),xf:false())
}</boolean-equal>
```

The preceding query generates the following result:

op:boolean-equal

```
<boolean-equal>true</boolean-equal>
```

Related Topics

W3C `boolean-equal` operator description.

W3C `boolean` function description

`xs:boolean`

op:boolean-less-than

If \$boolean-var1 equals the boolean value: false and \$boolean-var2 equals the boolean value: true, the boolean value: true is returned. For all other cases, the boolean value: false is returned as shown in the following table.

If boolean-var1 equals ...	And boolean-var2 equals ...	The boolean-less-than Operator Returns ...
false	true	true The boolean value of \$boolean-var1 is less than the boolean value of \$boolean-var2. (The boolean value: false is less than the boolean value: true.)
true	false	false The boolean value of \$boolean-var1 is greater than the boolean value of \$boolean-var2.
true	true	false The boolean value of \$boolean-var1 is the same as the boolean value of \$boolean-var2. \$boolean-var1 is not less than \$boolean-var2.
false	false	false The boolean value of \$boolean-var1 is the same as the boolean value of \$boolean-var2. \$boolean-var1 is not less than \$boolean-var2.

This operator is equivalent to the lt operator with boolean values.

Signatures

op:boolean-less-than(xs:boolean \$boolean-var1, xs:boolean \$boolean-var2) > xs:boolean

Arguments

Data Type	Argument	Description
xs:boolean	\$boolean-var1	Represents a boolean value.
xs:boolean	\$boolean-var2	Represents a boolean value.

Returns

Returns the boolean value: true if \$boolean-var1 is less than boolean value of \$boolean-var2. For example: if \$boolean-var1 is equal to false and \$boolean-var2 is equal to true, the boolean value true is returned.

Returns the boolean value: false for all other cases.

Examples

boolean-less-than(false, true) Returns true

Invoking `op:boolean-less-than(xf:false(),xf:true())` returns the boolean value: `true`, as shown in the following example query:

```
<boolean-less-than>{
  op:boolean-less-than(xf:false(),xf:true())
}</boolean-less-than>
```

The preceding query generates the following result:

```
<boolean-less-than>true</boolean-less-than>
```

boolean-less-than(true, false) Returns false

Invoking `op:boolean-less-than(xf:true(),xf:false())` returns the boolean value: `false`, as shown in the following example query:

```
<boolean-less-than>{
  op:boolean-less-than(xf:true(),xf:false())
}</boolean-less-than>
```

The preceding query generates the following result:

```
<boolean-less-than>false</boolean-less-than>
```

Related Topics

W3C `boolean-less-than` operator description.

W3C `boolean` function description

`xs:boolean`

op:boolean-greater-than

If \$boolean-var1 equals the boolean value: true and \$boolean-var2 equals the boolean value: false, the boolean value: true is returned. For all other cases, the boolean value: false is returned as shown in the following table.

If boolean-var1 equals ...	And boolean-var2 equals ...	The boolean-greater-than Operator Returns ...
true	false	true The boolean value of \$boolean-var1 is greater than the boolean value of \$boolean-var2. (The boolean value: true is greater than the boolean value: false.)
false	true	false The boolean value of \$boolean-var1 is less than the boolean value of \$boolean-var2.
true	true	false The boolean value of \$boolean-var1 is the same as the boolean value of \$boolean-var2. \$boolean-var1 is not greater than \$boolean-var2.
false	false	false The boolean value of \$boolean-var1 is the same as the boolean value of \$boolean-var2. \$boolean-var1 is not greater than \$boolean-var2.

This operator is equivalent to the gt operator with boolean values.

Signatures

op:boolean-greater-than(xs:boolean \$boolean-var1, xs:boolean \$boolean-var2) > xs:boolean

Arguments

Data Type	Argument	Description
xs:boolean	\$boolean-var1	Represents a boolean value.
xs:boolean	\$boolean-var2	Represents a boolean value.

Returns

Returns the boolean value: true if \$boolean-var1 is greater than boolean value of \$boolean-var2. For example: if \$boolean-var1 is equal to true and \$boolean-var2 is equal to false, the boolean value true is returned.

Returns the boolean value: false for all other cases.

Examples

boolean-greater-than(true, false) Returns true

Invoking `op:boolean-greater-than(xf:true(), xf:false())` returns the boolean value: `true`, as shown in the following example query:

```
<boolean-greater-than>{
  op:boolean-greater-than(xf:true(),xf:false())
}</boolean-greater-than>
```

The preceding query generates the following result:

```
<boolean-greater-than>true</boolean-greater-than>
```

boolean-greater-than(false, true) Returns false

Invoking `op:boolean-greater-than(xf:false(),xf:true())` returns the boolean value: `false`, as shown in the following example query:

```
<boolean-greater-than>{
  op:boolean-greater-than(xf:false(), xf:true())
}</boolean-greater-than>
```

The preceding query generates the following result:

```
<boolean-greater-than>false</boolean-greater-than>
```

Related Topics

W3C `boolean-greater-than` operator description.

W3C boolean function description

`xs:boolean`

XQuery Date and Time Operators Reference

This section provides descriptions of the XQuery date and time operators available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery operators. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002 available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the date and time operators available from the mapper functionality:

- op:subtract-dates
- op:subtract-times
- op:add-yearMonthDuration-to-dateTime
- op:add-dayTimeDuration-to-dateTime
- op:subtract-yearMonthDuration-from-dateTime
- op:subtract-dayTimeDuration-from-dateTime
- op:add-yearMonthDuration-to-date
- op:add-dayTimeDuration-to-date
- op:subtract-yearMonthDuration-from-date
- op:subtract-dayTimeDuration-from-date
- op:add-dayTimeDuration-to-time
- op:subtract-dayTimeDuration-from-time

op:subtract-dates

Computes the time difference between \$date-var1 and \$date-var2. Subtracts the value of \$date-var2 from \$date-var1.

If the value of \$date-var2 follows in time the value of \$date-var1, then the returned value is a negative duration.

If the value of \$date-var1 or \$date-var2 is the empty sequence, the following error is displayed:

```
Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method:
Type error in function subtract-dates invocation: expected type
[date@http://www.w3.org/2001/XMLSchema],
given type empty
```

Signatures

op:subtract-dates(xs:date \$date-var1, xs:date \$date-var2) > xf:dayTimeDuration

Arguments

Data Type	Argument	Description
xs:date	\$date-var1	Contains a representation of the date.
xs:date	\$date-var2	Contains a representation of the date.

Returns

Returns the time difference between \$date-var1 and \$date-var2 as dayTimeDuration.

Examples

Positive Difference

The following example query returns a positive number of days because \$date-var2 precedes in time \$date-var1:

```
<num_days>{xf:get-days-from-dayTimeDuration(op:subtract-dates(xs:date("2002-08-30"),
xs:date("2001-08-30")))}</num_days>
```

The preceding query generates the following result:

```
<num_days>365</num_days>
```

Negative Difference

The following example query returns a negative number of days because \$date-var2 follows in time \$date-var1:

```
<num_days>{xf:get-days-from-dayTimeDuration(op:subtract-dates(xs:date("2001-08-30"),
```

```
xs:date("2002-08-30"))}</num_days>
```

The preceding query generates the following result:

```
<num_days>-365</num_days>
```

Related Topics

W3C subtract-dates operator description.

W3C date data type description.

W3C dayTimeDuration operator description.

op:subtract-times

Computes the time difference between \$time-var1 and \$time-var2. Subtracts the value of \$time-var2 from \$time-var1.

If the value of \$time-var2 follows in time the value of \$time-var1, then the returned value is a negative duration.

If the value of \$date-var1 or \$date-var2 is the empty sequence, the following error is displayed:

Error occurred while executing XQuery: Error loading the XQuery or XSLT for this method:
Type error in function subtract-dates invocation: expected type [date@http://www.w3.org/2001/XMLSchema#date] given type empty

Signatures

op:subtract-times(xs:time \$time-var1, xs:time \$time-var2) > xs:dayTimeDuration

Arguments

Data Type	Argument	Description
xs:time	\$time-var1	Contains a representation of time.
xs:time	\$time-var2	Contains a representation of time.

Returns

Returns the time difference between \$time-var1 and \$time-var2 as dayTimeDuration value.

Examples

Positive

The following example returns a positive number of minutes because \$time-var2 precedes in time \$time-var1:

```
<minutes>{xf:get-minutes-from-dayTimeDuration(op:subtract-times(xf:time("08:01:00"),  
xf:time("08:00:00")))}</minutes>
```

The preceding query generates the following result:

```
<minutes>1</minutes>
```

Negative

The following example returns a negative number of days because \$time-var2 follows in time \$time-var1:

```
<minutes>{xf:get-minutes-from-dayTimeDuration(op:subtract-times(xf:time("08:00:00"),  
xf:time("08:01:00")))}</minutes>
```


The preceding query generates the following result:

```
<minutes>-1</minutes>
```

Related Topics

W3C `subtract-times` operator description.

W3C date data type description.

W3C `dayTimeDuration` operator description.

op:add-yearMonthDuration-to-dateTime

Adds \$yearMonthDuration-var to the date and time specified by \$dateTime-var.

Signatures

op:add-yearMonthDuration-to-dateTime(xs:dateTime \$dateTime-var, xf:yearMonthDuration \$yearMonthDuration-var) > xs:dateTime

Arguments

Data Type	Argument	Description
xs:dateTime	\$dateTime-var	Contains a representation of the date and time.
xf:yearMonthDuration	\$yearMonthDuration-var	Contains a time duration that can contain years and months.

Returns

Returns the dateTime result of adding the year and month specified by \$yearMonthDuration-var to date and time specified by \$dateTime-var. The returned dateTime value has the same timezone as \$dateTime-var. If \$dateTime-var has no timezone, then the returned dateTime value has the no timezone.

Examples

Simple

The following example query adds a yearMonthDuration value equal to 1 year and 1 month to a dateTime value equal to the date: January 1, 2003 and time: 1:00 AM as shown in the following query:

```
<dateTime>{
  op:add-yearMonthDuration-to-dateTime(xs:dateTime("2003-01-01T01:00:00"), xf:yearMonthDuration)
}</dateTime>
```

The resulting dateTime value equal to the date: February 1, 2004 and the time: 1:00 AM is returned as shown in the following result:

```
<dateTime>2004-02-01T01:00:00</dateTime>
```

Related Topics

W3C add-yearMonthDuration-to-dateTime operator description.

W3C yearMonthDuration description.

xf:yearMonthDuration constructor description.

W3C dateTime data type description

xs:dateTime constructor description.

op:add-dayTimeDuration-to-dateTime

Adds \$dayTimeDuration-var to the date and time specified by \$dateTime-var.

Signatures

op:add-dayTimeDuration-to-dateTime(xs:dateTime \$dateTime-var, xf:dayTimeDuration \$dayTimeDuration-var) > xs:dateTime

Arguments

Data Type	Argument	Description
xs:dateTime	\$dateTime-var	Contains a representation of the date and time.
xf:dayTimeDuration	\$dayTimeDuration-var	Contains a time duration that can contain days, hours, minutes, and seconds.

Returns

Returns the dateTime result of adding the date and time specified by \$dayTimeDuration-var to date and time specified by \$dateTime-var. The returned dateTime value has the same timezone as \$dateTime-var. If \$dateTime-var has no timezone, then the returned dateTime value has the no timezone.

Examples

Simple

The following example query adds a dayTimeDuration value equal to 1 day, 2 hours, 30 minutes, and 5 seconds to a dateTime value equal to the date: January 1, 2003 and time: 1:00 AM as shown in the following query:

```
<dateTime>{
  op:add-dayTimeDuration-to-dateTime(xs:dateTime("2003-01-01T01:00:00"), xf:dayTimeDuration
}</dateTime>
```

The resulting dateTime value equal to the date: January 2, 2003 and the time: 3:30:05 AM is returned as shown in the following result:

```
<dateTime>2003-01-02T03:30:05</dateTime>
```

Related Topics

W3C add-dayTimeDuration-to-dateTime operator description.

W3C dayTimeDuration description.

xf:dayTimeDuration constructor description.

W3C dateTime data type description

xs:dateTime constructor description.

op:subtract-yearMonthDuration-from-dateTime

Subtracts the time duration specified by \$yearMonthDuration-var from the date and time specified by \$dateTime-var.

Signatures

op:subtract-yearMonthDuration-from-dateTime(xs:dateTime \$dateTime-var, xf:yearMonthDuration \$yearMonthDuration-var) > xs:dateTime

Arguments

Data Type	Argument	Description
xs:dateTime	\$dateTime-var	Contains a representation of the date and time.
xf:yearMonthDuration	\$yearMonthDuration-var	Contains a time duration which can contain years and months.

Returns

Returns the dateTime value of subtracting the time duration specified by \$yearMonthDuration-var from the date and time specified by \$dateTime-var.

Examples

Subtracting a Positive yearMonthDuration

The following example query subtracts a positive yearMonthDuration value equal to 1 year and 1 month from a dateTime value equal to the date: February 2, 2003 and time: 1:01 AM as shown in the following query:

```
<positive>{
  op:subtract-yearMonthDuration-from-dateTime(xs:dateTime("2003-02-02T01:01:00"), xf:yearMonthDuration("P1Y1M"))
}</positive>
```

The resulting dateTime value equal to the date: January 2, 2002 and time: 1:01 AM is returned as shown in the following result:

```
<positive>2002-01-02T01:01:00</positive>
```

Subtracting a Negative yearMonthDuration

The following example query subtracts a negative yearMonthDuration value equal to 1 year and 1 month from a dateTime value equal to the date: January 2, 2003 and time: 1:01 AM as shown in the following query:

```
<negative>{
  op:subtract-yearMonthDuration-from-dateTime(xs:dateTime("2003-01-02T01:01:00"), xf:yearMonthDuration("P1Y1M"))
}</negative>
```

The resulting dateTime value equal to the date: February 2, 2004 and time: 1:01 AM is returned as shown in the following result:

`<negative>2004-02-02T01:01:00</negative>`

Related Topics

W3C `subtract-yearMonthDuration-from-dateTime` operator description.

W3C `yearMonthDuration` description.

`xf:yearMonthDuration` constructor description.

W3C `dateTime` data type description

`xs:dateTime` constructor description.

op:subtract-dayTimeDuration-from-dateTime

Subtracts the time duration specified by \$dayTimeDuration-var from the date and time specified by \$dateTime-var.

Signatures

op:subtract-dayTimeDuration-from-dateTime(xs:dateTime \$dateTime-var, xf:dayTimeDuration \$dayTimeDuration-var) > xs:dateTime

Arguments

Data Type	Argument	Description
xs:dateTime	\$dateTime-var	Contains a representation of the date and time.
xf:dayTimeDuration	\$dayTimeDuration-var	Contains a time duration that can contain days, hours, minutes, and seconds.

Returns

Returns the dateTime value of subtracting the time duration specified by \$dayTimeDuration-var from the date and time specified by \$dateTime-var.

Examples

Subtracting a Positive dayTimeDuration

The following example query subtracts a positive dayTimeDuration value equal to 1 day and 1 minute from a dateTime value equal to the date: January 2, 2003 and time: 1:01 AM as shown in the following query:

```
<positive>{  
  op:subtract-dayTimeDuration-from-dateTime(xs:dateTime("2003-01-02T01:01:00"), xf:dayTimeDuration("P1DT01M"))  
}</positive>
```

The resulting dateTime value equal to the date: January 1, 2003 and time: 1:00 AM is returned as shown in the following result:

```
<positive>2003-01-01T01:00:00</positive>
```

Subtracting a Negative dayTimeDuration

The following example query subtracts a negative dayTimeDuration value equal to 1 day and 1 minute from a dateTime value equal to the date: January 2, 2003 and time: 1:01 AM as shown in the following query:

```
<negative>{  
  op:subtract-dayTimeDuration-from-dateTime(xs:dateTime("2003-01-02T01:01:00"), xf:dayTimeDuration("P1DT01M"))  
}</negative>
```

The resulting dateTime value equal to the date: January 3, 2003 and time: 1:02 AM is returned as shown in the following result:

op:subtract-dayTimeDuration-from-dateTime

`<negative>2003-01-03T01:02:00</negative>`

Related Topics

[W3C subtract-dayTimeDuration-from-dateTime operator description.](#)

[W3C dayTimeDuration description.](#)

[xf:dayTimeDuration constructor description.](#)

[W3C dateTime data type description](#)

[xs:dateTime constructor description.](#)

op:add-yearMonthDuration-to-date

Adds \$yearMonthDuration-var to the date specified by \$date-var.

Signatures

op:add-yearMonthDuration-to-date(xs:date \$date-var, xf:yearMonthDuration \$yearMonthDuration-var)
> xs:date

Arguments

Data Type	Argument	Description
xs:date	\$date-var	Contains a representation of the date.
xf:yearMonthDuration	\$yearMonthDuration-var	Contains a time duration that can contain years and months.

Returns

Returns the date result of adding the year and month specified by \$yearMonthDuration-var to date specified by \$date-var. The returned date value has the same timezone as \$date-var. If \$date-var has no timezone, then the returned date value has the no timezone.

Examples

Simple

The following example query adds a yearMonthDuration value equal to 1 year and 1 month to a date value equal to the date: January 1, 2003 as shown in the following query:

```
<date>{  
  op:add-yearMonthDuration-to-date(xs:date("2003-01-01"), xf:yearMonthDuration("P1Y1M"))  
}</date>
```

The resulting date value equal to the date: February 1, 2004 is returned as shown in the following result:

```
<date>2004-02-01</date>
```

Related Topics

W3C add-yearMonthDuration-to-date operator description.

W3C yearMonthDuration description.

xf:yearMonthDuration constructor description.

W3C date data type description

xs:date constructor description.

op:add-dayTimeDuration-to-date

Adds \$dayTimeDuration-var to the date specified by \$date-var.

Signatures

op:add-dayTimeDuration-to-date(xs:date \$date-var, xf:dayTimeDuration \$dayTimeDuration-var) > xs:date

Arguments

Data Type	Argument	Description
xs:date	\$date-var	Contains a representation of the date.
xf:dayTimeDuration	\$dayTimeDuration-var	Contains a time duration that can contain days, hours, minutes, and seconds.

Returns

Returns the date result of adding the date and time specified by \$dayTimeDuration-var to date specified by \$date-var. The returned date value has the same timezone as \$date-var. If \$date-var has no timezone, then the returned date value has the no timezone.

Examples

Simple

The following example query adds a dayTimeDuration value equal to 1 day, 2 hours, 30 minutes, and 5 seconds to a date value equal to the date: January 1, 2003 as shown in the following query:

```
<date>{  
  op:add-dayTimeDuration-to-date(xs:date("2003-01-01"), xf:dayTimeDuration("P1DT2H30M5S"))  
}</date>
```

The resulting date value equal to the date: January 2, 2003 is returned as shown in the following result:

```
<date>2003-01-02</date>
```

Related Topics

W3C add-dayTimeDuration-to-date operator description.

W3C dayTimeDuration description.

xf:dayTimeDuration constructor description.

W3C date data type description

xs:date constructor description.

op:subtract-yearMonthDuration-from-date

Subtracts the time duration specified by \$yearMonthDuration-var from the date specified by \$date-var.

Signatures

op:subtract-yearMonthDuration-from-date(xs:date \$date-var, xf:yearMonthDuration \$yearMonthDuration-var) > xs:date

Arguments

Data Type	Argument	Description
xs:date	\$date-var	Contains a representation of the date.
xf:yearMonthDuration	\$yearMonthDuration-var	Contains a time duration which can contain years and months.

Returns

Returns the date value of subtracting the time duration specified by \$yearMonthDuration-var from the date specified by \$date-var.

Examples

Subtracting a Positive yearMonthDuration

The following example query subtracts a positive yearMonthDuration value equal to 1 year and 1 month from a date value equal to the date: February 2, 2003 as shown in the following query:

```
<positive>{
  op:subtract-yearMonthDuration-from-date(xs:date("2003-02-02"), xf:yearMonthDuration("P1
"}</positive>
```

The resulting date value equal to the date: January 2, 2002 is returned as shown in the following result:

```
<positive>2002-01-02</positive>
```

Subtracting a Negative yearMonthDuration

The following example query subtracts a negative yearMonthDuration value equal to 1 year and 1 month from a date value equal to the date: January 2, 2003 as shown in the following query:

```
<negative>{
  op:subtract-yearMonthDuration-from-date(xs:date("2003-01-02"), xf:yearMonthDuration("-P
"}</negative>
```

The resulting date value equal to the date: February 2, 2004 is returned as shown in the following result:

```
<negative>2004-02-02</negative>
```

Related Topics

W3C `subtract-yearMonthDuration-from-date` operator description.

W3C `yearMonthDuration` description.

`xf:yearMonthDuration` constructor description.

W3C date data type description

`xs:date` constructor description.

op:subtract-dayTimeDuration-from-date

Subtracts the time duration specified by \$dayTimeDuration-var from the date specified by \$date-var.

Signatures

op:subtract-dayTimeDuration-from-date(xs:date \$date-var, xf:dayTimeDuration \$dayTimeDuration-var)
> xs:date

Arguments

Data Type	Argument	Description
xs:date	\$date-var	Contains a representation of the date.
xf:dayTimeDuration	\$dayTimeDuration-var	Contains a time duration that can contain days, hours, minutes, and seconds.

Returns

Returns the date value of subtracting the time duration specified by \$dayTimeDuration-var from the date specified by \$date-var.

Examples

Subtracting a Positive dayTimeDuration

The following example query subtracts a positive dayTimeDuration value equal to 1 day and 1 minute from a date value equal to the date: January 2, 2003 as shown in the following query:

```
<positive>{
  op:subtract-dayTimeDuration-from-date(xs:date("2003-01-02"), xf:dayTimeDuration("P1DT1M"))
}</positive>
```

The resulting date value equal to the date: January 1, 2003 is returned as shown in the following result:

```
<positive>2003-01-01</positive>
```

Subtracting a Negative dayTimeDuration

The following example query subtracts a negative dayTimeDuration value equal to 1 day and 1 minute from a date value equal to the date: January 2, 2003 as shown in the following query:

```
<negative>{
  op:subtract-dayTimeDuration-from-date(xs:date("2003-01-02"), xf:dayTimeDuration("-P1DT1M"))
}</negative>
```

The resulting date value equal to the date: January 3, 2003 is returned as shown in the following result:

```
<negative>2003-01-03</negative>
```

Related Topics

W3C `subtract-dayTimeDuration-from-date` operator description.

W3C `dayTimeDuration` description.

`xf:dayTimeDuration` constructor description.

W3C date data type description

`xs:date` constructor description.

op:add-dayTimeDuration-to-time

Adds \$dayTimeDuration-var to the time specified by \$time-var.

Signatures

op:add-dayTimeDuration-to-time(xs:time \$time-var, xf:dayTimeDuration \$dayTimeDuration-var) > xs:time

Arguments

Data Type	Argument	Description
xs:time	\$time-var	Contains a representation of the time.
xf:dayTimeDuration	\$dayTimeDuration-var	Contains a time duration that can contain days, hours, minutes, and seconds.

Returns

Returns the time result of adding the date and time specified by \$dayTimeDuration-var to time specified by \$time-var. The returned time value has the same timezone as \$time-var. If \$time-var has no timezone, then the returned time value has the no timezone.

Examples

Simple

The following example query adds a dayTimeDuration value equal to 1 day, 2 hours, 30 minutes, and 5 seconds to a time value equal to the time: 1:00 AM as shown in the following query:

```
<time>{  
  op:add-dayTimeDuration-to-time(xs:time("01:00:00"), xf:dayTimeDuration("P1DT2H30M5S"))  
}</time>
```

The resulting time value equal to the time: 3:30:05 AM is returned as shown in the following result:

```
<time>03:30:05</time>
```

Related Topics

W3C add-dayTimeDuration-to-time operator description.

W3C dayTimeDuration description.

xf:dayTimeDuration constructor description.

W3C time data type description

xs:time constructor description.

op:subtract-dayTimeDuration-from-time

Subtracts the time duration specified by \$dayTimeDuration-var from the time specified by \$time-var.

Signatures

op:subtract-dayTimeDuration-from-time(xs:time \$time-var, xf:dayTimeDuration \$dayTimeDuration-var)
> xs:time

Arguments

Data Type	Argument	Description
xs:time	\$time-var	Contains a representation of the time.
xf:dayTimeDuration	\$dayTimeDuration-var	Contains a time duration that can contain days, hours, minutes, and seconds.

Returns

Returns the time value of subtracting the time duration specified by \$dayTimeDuration-var from the date and time specified by \$time-var.

Examples

Subtracting a Positive dayTimeDuration

The following example query subtracts a positive dayTimeDuration value equal to 1 day and 1 minute from a time value equal to the time: 1:01 AM as shown in the following query:

```
<positive>{  
  op:subtract-dayTimeDuration-from-time(xs:time("01:01:00"), xf:dayTimeDuration("P1DT1M"))  
}</positive>
```

The resulting time value equal to the time: 1:00 AM is returned as shown in the following result:

```
<positive>01:00:00</positive>
```

Subtracting a Negative dayTimeDuration

The following example query subtracts a negative dayTimeDuration value equal to 1 day and 1 minute from a time value equal to the time: 1:01 AM as shown in the following query:

```
<negative>{  
  op:subtract-dayTimeDuration-from-time(xs:time("01:01:00"), xf:dayTimeDuration("-P1DT1M"))  
}</negative>
```

The resulting time value equal to the time: 1:02 AM is returned as shown in the following result:

```
<negative>01:02:00</negative>
```

Related Topics

W3C `subtract-dayTimeDuration-from-time` operator description.

W3C `dayTimeDuration` description.

`xf:dayTimeDuration` constructor description.

W3C time data type description

`xs:time` constructor description.

XQuery Duration Operators Reference

This section provides descriptions of the XQuery duration operators available in the mapper functionality of WebLogic Workshop. You use the mapper functionality to generate queries and to edit these queries to add invocations to these provided XQuery operators. To learn more, see *Invoking Functions or Operators in a Query*.

In addition to the XQuery functions and operators available in the mapper functionality, a larger set functions and operators is provided. You can manually add invocations to these functions and operators to queries in the **Source View** of the mapper functionality. For a list of these additional functions and operators, see the *XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Working Draft 16 August 2002* available from the W3C Web site at the following URL:

<http://www.w3.org/TR/2002/WD-xquery-operators-20020816>

This section lists the duration operators available from the mapper functionality:

- op:add-yearMonthDurations
- op:subtract-yearMonthDurations
- op:multiply-yearMonthDuration
- op:divide-yearMonthDuration
- op:add-dayTimeDurations
- op:subtract-dayTimeDurations
- op:multiply-dayTimeDuration
- op:divide-dayTimeDuration

op:add-yearMonthDurations

Adds \$yearMonthDuration-var1 to \$yearMonthDuration-var2.

Signatures

op:add-yearMonthDurations(xf:yearMonthDuration \$yearMonthDuration-var1, xf:yearMonthDuration \$yearMonthDuration-var2) > xf:yearMonthDuration

Arguments

Data Type	Argument	Description
xf:yearMonthDuration	\$yearMonthDuration-var1	Contains a time duration that can contain years and months.
xf:yearMonthDuration	\$yearMonthDuration-var2	Contains a time duration that can contain years and months.

Returns

Returns the yearMonthDuration value of adding \$yearMonthDuration-var1 to \$yearMonthDuration-var2.

Examples

Simple

The following example query adds a yearMonthDuration value equal to 1 year and 2 months to a yearMonthDuration value equal to 2 years and 11 months as shown in the following query:

```
<yearMonthDuration>{  
  op:add-yearMonthDurations(xf:yearMonthDuration("P1Y2M"), xf:yearMonthDuration("P2Y11M"))  
}</yearMonthDuration>
```

The resulting yearMonthDuration value of 4 years and 1 month is returned as shown in the following result:

```
<yearMonthDuration>P4Y1M</yearMonthDuration>
```

Related Topics

W3C add-yearMonthDurations operator description.

W3C yearMonthDuration description.

xf:yearMonthDuration constructor description.

op:subtract-yearMonthDurations

Subtracts the value of \$yearMonthDuration-var2 from \$yearMonthDuration-var1.

Signatures

op:subtract-yearMonthDurations(xf:yearMonthDuration \$yearMonthDuration-var1, xf:yearMonthDuration \$yearMonthDuration-var2) > xf:yearMonthDuration

Arguments

Data Type	Argument	Description
xf:yearMonthDuration	\$yearMonthDuration-var1	Contains a time duration that can contain years and months.
xf:yearMonthDuration	\$yearMonthDuration-var2	Contains a time duration that can contain years and months.

Returns

Returns the yearMonthDuration value of subtracting \$yearMonthDuration-var2 from \$yearMonthDuration-var1.

Examples

Positive

The following example query subtracts a yearMonthDuration value equal to 1 year and 1 month from a yearMonthDuration value equal to 3 years and 3 months as shown in the following query:

```
<positive>{  
    op:subtract-yearMonthDurations(xf:yearMonthDuration("P3Y3M"), xf:yearMonthDuration("P1Y1M"))  
}</positive>
```

The resulting positive yearMonthDuration value of 2 years and 2 months is returned as shown in the following result:

```
<positive>P2Y2M</positive>
```

Negative

The following example query subtracts a yearMonthDuration value equal to 3 years and 3 months from a yearMonthDuration value equal to 1 year and 1 month as shown in the following query:

```
<negative>{  
    op:subtract-yearMonthDurations(xf:yearMonthDuration("P1Y1M"), xf:yearMonthDuration("P3Y3M"))  
}</negative>
```

The resulting negative yearMonthDuration value of 2 years and 2 months is returned as shown in the following result:

`<negative>-P2Y2M</negative>`

Related Topics

W3C `subtract-yearMonthDurations` operator description.

W3C `yearMonthDuration` description.

`xf:yearMonthDuration` constructor description.

op:multiply-yearMonthDuration

Multiplies \$yearMonthDuration-var with \$decimal-var.

Signatures

op:multiply-yearMonthDuration(xf:yearMonthDuration \$yearMonthDuration-var, xs:decimal \$decimal-var)
> xf:yearMonthDuration

Arguments

Data Type	Argument	Description
xf:yearMonthDuration	\$yearMonthDuration-var	Contains a time duration that can contain years and months.
xs:decimal	\$decimal-var	Contains a decimal value, for example: 1.1.

Returns

Returns the yearMonthDuration value of dividing \$yearMonthDuration-var with \$decimal-var. The return value is rounded up to the nearest month and year as shown in the following example.

Examples

Round Up

Multiplying the 1 year value of \$yearMonthDuration-var by 1.5 results in 1 year and 6 months. Multiplying the 1 month value of \$yearMonthDuration-var by 1.5 results in 1.5 months which is rounded up to 2 months. Adding 1 year and 6 months to 2 months results in a yearMonthDuration of 1 year and 8 months as shown in the following example query:

```
let $yearMonthDuration-var := xf:yearMonthDuration("P1Y1M")
return
let $decimal-var := 1.5
return
let $mydur := op:multiply-yearMonthDuration($yearMonthDuration-var, $decimal-var)
return
<result>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
  <months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</result>
```

The preceding query generates the following result:

```
<result>
  <years>1</years>
  <months>8</months>
</result>
```

Not Enough to Round Up

Multiplying a 1 month yearMonthDuration by 0.49 results in 0.49 months which is rounded down to zero months as shown in the following example query:

```
let $yearMonthDuration-var := xf:yearMonthDuration("P1M")
return
let $decimal-var := 0.49
return
let $mydur := op:multiply-yearMonthDuration($yearMonthDuration-var, $decimal-var)
return
<result>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
  <months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</result>
```

The preceding query generates the following result:

```
<result>
  <years>0</years>
  <months>0</months>
</result>
```

Related Topics

W3C multiply-yearMonthDuration operator description.

W3C yearMonthDuration description.

xf:yearMonthDuration constructor description.

op:divide-yearMonthDuration

Divides \$yearMonthDuration-var by \$decimal-var.

Signatures

op:divide-yearMonthDuration(xf:yearMonthDuration \$yearMonthDuration-var, xs:decimal \$decimal-var)
> xf:yearMonthDuration

Arguments

Data Type	Argument	Description
xf:yearMonthDuration	\$yearMonthDuration-var	Contains a time duration that can contain years and months.
xs:decimal	\$decimal-var	Contains a decimal value, for example: 1.1.

Returns

Returns the yearMonthDuration value of dividing \$yearMonthDuration-var by \$decimal-var. The return value is rounded up to the nearest year and month as shown in following example.

Examples

Round Up

Dividing the 1 year value of \$yearMonthDuration-var by 2.0 results in 6 months. Dividing the 1 month value of \$yearMonthDuration-var by 2.0 results in .5 months which is rounded up to 1 month. Adding 6 months to 1 month results in a yearMonthDuration of 7 months as shown in the following example query:

```
let $yearMonthDuration-var := xf:yearMonthDuration("P1Y1M")
return
let $decimal-var := 2
return
let $mydur := op:divide-yearMonthDuration($yearMonthDuration-var, $decimal-var)
return
<result>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
  <months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</result>
```

The preceding query generates the following result:

```
<result>
  <years>0</years>
  <months>7</months>
</result>
```

Not Enough to Round Up

Dividing a 1 month yearMonthDuration by 2.04 results in 0.4901... months which is rounded down to zero months as shown in the following example query:

```
let $yearMonthDuration-var := xf:yearMonthDuration("P1M")
return
let $decimal-var := 2.04
return
let $mydur := op:divide-yearMonthDuration($yearMonthDuration-var, $decimal-var)
return
<result>
  <years>{xf:get-years-from-yearMonthDuration($mydur)}</years>
  <months>{xf:get-months-from-yearMonthDuration($mydur)}</months>
</result>
```

The preceding query generates the following result:

```
<result>
  <years>0</years>
  <months>0</months>
</result>
```

Related Topics

W3C divide-yearMonthDuration operator description.

W3C yearMonthDuration description.

xf:yearMonthDuration constructor description.

op:add-dayTimeDurations

Adds \$dayTimeDuration-var1 to \$dayTimeDuration-var2.

Signatures

op:add-dayTimeDurations(xf:dayTimeDuration \$dayTimeDuration-var1, xf:dayTimeDuration \$dayTimeDuration-var2) > xf:dayTimeDuration

Arguments

Data Type	Argument	Description
xf:dayTimeDuration	\$dayTimeDuration-var1	Contains a time duration that can contain days, hours, minutes, and seconds.
xf:dayTimeDuration	\$dayTimeDuration-var2	Contains a time duration that can contain days, hours, minutes, and seconds.

Returns

Returns the dayTimeDuration value of adding \$dayTimeDuration-var1 to \$dayTimeDuration-var2.

Examples

Simple

The following example query adds a dayTimeDuration value equal to 1 day, 2 hours, 30 minutes, and 5 seconds to a dayTimeDuration value equal to 31 minutes as shown in the following query:

```
<dayTimeDuration>{  
  op:add-dayTimeDurations(xf:dayTimeDuration("P1DT2H30M5S"), xf:dayTimeDuration("PT31M"))  
}</dayTimeDuration>
```

The resulting dayTimeDuration value of 1 day, 3 hours, 1 minute, and 5 seconds is returned as shown in the following result:

```
<dayTimeDuration>P1DT3H1M5S</dayTimeDuration>
```

Related Topics

W3C add-dayTimeDurations operator description.

W3C dayTimeDuration description.

xf:dayTimeDuration constructor description.

op:subtract-dayTimeDurations

Subtracts the value of \$dayTimeDuration-var2 from \$dayTimeDuration-var1.

Signatures

op:subtract-dayTimeDurations(xf:dayTimeDuration \$dayTimeDuration-var1, xf:dayTimeDuration \$dayTimeDuration-var2) > xf:dayTimeDuration

Arguments

Data Type	Argument	Description
xf:dayTimeDuration	\$dayTimeDuration-var1	Contains a time duration that can contain days, hours, minutes, and seconds.
xf:dayTimeDuration	\$dayTimeDuration-var2	Contains a time duration that can contain days, hours, minutes, and seconds.

Returns

Returns the dayTimeDuration value of subtracting \$dayTimeDuration-var2 from \$dayTimeDuration-var1.

Examples

Positive

The following example query subtracts a dayTimeDuration value equal to 1 day and 1 minute from a dayTimeDuration value equal to 3 days, 3 hours, 3 minutes, and 3 seconds as shown in the following query:

```
<positive>{  
    op:subtract-dayTimeDurations(xf:dayTimeDuration("P3DT3H3M3S"), xf:dayTimeDuration("P1DT1M"))  
}</positive>
```

The resulting positive dayTimeDuration value of 2 days, 3 hours, 2 minutes and 3 seconds is returned as shown in the following result:

```
<positive>P2DT3H2M3S</positive>
```

Negative

The following example query subtracts a dayTimeDuration value equal to 3 days, 3 hours, 3 minutes, and 3 seconds from a dayTimeDuration value equal to 1 day and 1 minute as shown in the following query:

```
<negative>{  
    op:subtract-dayTimeDurations(xf:dayTimeDuration("P1DT1M"), xf:dayTimeDuration("P3DT3H3M3S"))  
}</negative>
```

The resulting negative dayTimeDuration value of 2 years and 2 months is returned as shown in the following result:

`<negative>-P2DT3H2M3S</negative>`

Related Topics

W3C `subtract-dayTimeDurations` operator description.

W3C `dayTimeDuration` description.

`xf:dayTimeDuration` constructor description.

op:multiply-dayTimeDuration

Multiplies \$dayTimeDuration-var with \$decimal-var.

Signatures

op:multiply-dayTimeDuration(xf:dayTimeDuration \$dayTimeDuration-var, xs:decimal \$decimal-var) > xf:dayTimeDuration

Arguments

Data Type	Argument	Description
xf:dayTimeDuration	\$dayTimeDuration-var	Contains a time duration that can contain days, hours, minutes, and seconds.
xs:decimal	\$decimal-var	Contains a decimal value, for example: 1.1.

Returns

Returns the dayTimeDuration value of dividing \$dayTimeDuration-var with \$decimal-var.

Examples

Simple

Multiplying the 1 day value of \$dayTimeDuration-var by 1.5 results in 1 day and 12 hours. Multiplying the 1 hour value of \$dayTimeDuration-var by 1.5 results in 1 hour and 30 minutes. Adding 1 day and 12 hours to 1 hour and 30 minutes results in a dayTimeDuration of 1 day, 13 hours, and 30 minutes as shown in the following example query:

```
<result>{  
  op:multiply-dayTimeDuration(xf:dayTimeDuration("P1DT1H"), 1.5)  
}</result>
```

The preceding query generates the following result:

```
<result>P1DT13H30M</result>
```

Related Topics

W3C multiply-dayTimeDuration operator description.

W3C dayTimeDuration description.

xf:dayTimeDuration constructor description.

op:divide-dayTimeDuration

Divides \$dayTimeDuration-var by \$decimal-var.

Signatures

op:divide-dayTimeDuration(xf:dayTimeDuration \$dayTimeDuration-var, xs:decimal \$decimal-var) > xf:dayTimeDuration

Arguments

Data Type	Argument	Description
xf:dayTimeDuration	\$dayTimeDuration-var	Contains a time duration that can contain days, hours, minutes, and seconds.
xs:decimal	\$decimal-var	Contains a decimal value, for example: 1.1.

Returns

Returns the dayTimeDuration value of dividing \$dayTimeDuration-var by \$decimal-var.

Examples

Simple

Dividing the 1 day value of \$dayTimeDuration-var by 2.0 results in 12 hours. Dividing the 1 hour value of \$dayTimeDuration-var by 2.0 results in 30 minutes. Adding these values together results in a dayTimeDuration of 12 hours and 30 minutes as shown in the following example query:

```
<result>{
  op:divide-dayTimeDuration(xf:dayTimeDuration("P1DT1H"), 2.0)
}</result>
```

The preceding query generates the following result:

```
<result>PT12H30M</result>
```

Related Topics

W3C divide-dayTimeDuration operator description.

W3C dayTimeDuration description.

xf:dayTimeDuration constructor description.

XQuery Occurrence Indicators

The following section defines the supported XQuery occurrence indicators. An occurrence indicator defines how many times an item may occur. Occurrence indicators are used in XQuery function signatures to define how many items can be passed into a argument and how many items can be returned from a function.

None

If no occurrence indicator is specified, than the item must appear once and only once as shown in the following example function signature:

```
bea-xf:trim-left(xs:string $string-var) > xs:string
```

In the preceding example, just a single item must be specified for the \$string-var1 argument of the trim-left function. For example, the following would be a *valid* invocation of the normalizedString function:

```
<result>{bea-xf:trim-left("  abc  ")}</result>
```

The following are *invalid* invocations of the normalizedString function, as specified by the occurrence indicator of the \$string-var argument:

```
<result>{bea-xf:trim-left(())}</result>
```

```
<result>{bea-xf:trim-left("  abc  ", "  def  ")}</result>
```

?

The ? occurrence indicator defines that the item can occur once or zero times. An item appearing zero times means that the empty sequence is passed in as an argument. (See the following example for details.) The empty sequence is a sequence containing zero items (), which is similar to null in SQL.

The ? occurrence indicator is specified for both the \$string-var1 and \$string-var2 arguments of the following example XQuery function signature:

```
xf:tokenize(xs:string? $string-var1, xs:string? $string-var2) > xs:string*
```

The ? occurrence indicator for the \$string-var1 argument defines that the item can occur once or zero times, as shown in the following table.

Occurs This Many Times . . .	Example
Zero	tokenize((),"\s")
One	tokenize("Jane fell down the hill", "\s")

XQuery Reference

*

The * occurrence indicator defines that the item can contain zero or more items. The * occurrence indicator is specified in the return value of the following example XQuery function signature:

```
xf:tokenize(xs:string? $string-var1, xs:string? $string-var2) > xs:string*
```

An * occurrence indicator as part of the return value of a function defines that zero or more items may be returned from that function, as shown in the examples of the following table.

Examples	How Many Items Are Returned?	What is Returned?
<code>tokenize("", "\s")</code>	Zero	
<code>tokenize("Jane", "\s")</code>	One	Jane
<code>tokenize("Jane fell down", "\s")</code>	Many (In this case, three)	("Jane", "fell", "down")

+

The + occurrence indicator defines that the item must appear one or more times.

XQuery Namespace Conventions

Namespaces provide a mechanism to uniquely distinguish names used in XML documents. The W3C uses by convention the following namespace prefixes to identity W3C XQuery data types (xs:), functions (xf:), and operations (op:). In addition, BEA has defined the bea-xf: namespace to uniquely identify additional BEA supplied functions and data types. These XQuery namespace prefixes are used in the XQuery function signatures as shown in the examples provided in the following sections.

This section provides information about the following XQuery namespace prefixes:

- xs:
- xf:
- op:
- bea-xf:

xs:

XML Schema datatypes and the constructors that construct these datatypes belong to the XML Schema namespace. The XML Schema namespace is identified by the following URI:

<http://www.w3.org/2001/XMLSchema>

The xs: namespace prefix is associated (by convention) with this namespace URI.

For example, when specifying the XML Schema data type string in a function signature, the xs: namespace prefix precedes string in the \$string-var argument definition, as shown in the following example function signature:

```
bea-xf:trim-left(xs:string $string-var) > xs:string
```

xf:

W3C XQuery functions belong to the xquery functions namespace. The xquery functions namespace is identified by the following URI:

<http://www.w3.org/2002/08/xquery-functions>

The xf: namespace prefix is associated (by convention) with this namespace URI.

For example when invoking a W3C function in an query, prefix the function name with the string xf: as shown in the following example query:

```
<1>{
for $tok in xf:tokenize("Jane fell down the hill", "\s")
  return <i>{ $tok }<i>
}</1>
```

op:

W3C XQuery operators belong to the xquery operators namespace. The xquery operators namespace is identified by the following URI:

```
http://www.w3.org/2002/08/xquery-operators
```

The op: namespace prefix is associated (by convention) with this namespace URI.

For example when invoking a W3C operation in an query, prefix the function name with the string op: as shown in the following example query:

```
<num_days>{get-days-from-dayTimeDuration(op:subtract-dates(date("2002-08-30"),date("2001-08-30"))}
```

bea-xf:

BEA XQuery functions belong to the bea xquery functions namespace. The bea xquery namespace is identified by the following URI:.

```
http://www.bea.com/2002/xquery-functions
```

The bea-xf: namespace prefix is associated (by convention) with this namespace URI.

For example when invoking a BEA function in an query, prefix the function name with the string bea-xf: as shown in the following example query:

```
<result>{bea-xf:trim-left("  abc  ")}</result>
```

XQuery Data Types

The following are the XML Schema data types used in the XQuery language:

- xs:anyURI
- xs:boolean
- xs:byte
- xs:date
- xs:dateTime
- xs:decimal
- xf:dayTimeDuration
- xs:double
- xs:duration
- xs:float
- xs:gDay
- xs:gMonthDay
- xs:gYear
- xs:gYearMonth
- xs:int
- xs:long
- xs:Name
- xs:QName
- xs:short
- xs:string
- xs:time
- xf:yearMonthDuration

Note: For performance reasons, the integer supported by this XQuery engine is equivalent to a Java long, which is smaller than the W3C XML Schema integer, which has an arbitrary length. To learn more, see the integer description.

item

The item data type is an atomic value or a node. The item data type is an XQuery specific data type.

Related Topics

XQuery item description.

XQuery SequenceType description.

integer

A sequence of decimal digits (0–9) within the range of 9,223,372,036,854,775,807 to –9,223,372,036,854,775,808. For performance reasons, the integer supported by this XQuery engine is equivalent to a Java long, which is smaller than the W3C XML Schema integer, which has an arbitrary length.

node

The node data type is either an element, an attribute, a document, a text–node, a processing instruction, or a comment. The node data type is an XQuery specific data type.

Related Topics

[XQuery Node description.](#)

[XQuery SequenceType description.](#)

XQuery Language and XML Reference

This section provides XQuery language and XML links to the W3C Working Draft 16 August 2002 XQuery language and W3C XML 1.0 documentation, respectively.

This section contains the following topics:

- XQuery Prologs
- XQuery Expressions
- XQuery FLWR Expressions
- XQuery Namespaces
- XQuery Function Definitions
- XQuery Function Calls and Arguments
- XQuery Path Expressions
- XQuery Literals
- XQuery Combining Sequences
- XQuery Conditional Expressions
- XQuery Logical Expressions
- XQuery Comparison Expressions
- XML Element Declarations
- XML Attribute Declarations
- XML CDATA Sections
- XML Comments
- XML Prolog and Document Type Declaration

XQuery Prologs

To learn more about XQuery prologs, see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#id-query-prolog>

XQuery Expressions

To learn more about XQuery expressions, see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#id-expressions>

XQuery FLWR Expressions

To learn more about XQuery FLWR expressions (for, let, where, and return), see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#id-flwr-expressions>

To learn more about using the for and where clauses, see Step 4: Mapping a Repeating Element (Join) in *Tutorial: Building Your First Data Transformation*.

XQuery Namespaces

To learn more about the syntax of XQuery Namespaces, see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#id-namespace-decls>

To learn more about XQuery namespaces, see Understanding the Transformation in the *Tutorial: Building Your First Data Transformation*.

To learn more about the namespace prefixes used in WebLogic Workshop, see XQuery Namespace Conventions.

XQuery Function Definitions

To learn more about defining functions, see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#FunctionDefns>

XQuery Function Calls and Arguments

To learn more about function calls and arguments, see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#id-function-calls>

To learn more about adding function calls to queries using the mapper, see "Invoking Functions or Operators in a Query" in Modifying Links Using the Target Expression Tab in the *Guide to Data Transformation*.

To learn more about the XQuery functions and operators available from the mapper, see the XQuery function and operator chapters in the XQuery Reference.

XQuery Path Expressions

To learn more about path expressions, see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#id-path-expressions>

XQuery Literals

To learn more about XQuery string and numeric literals, see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#id-literals>

XQuery Combining Sequences

To learn more about the operators for combining sequences of nodes, see the following URL:

http://www.w3.org/TR/2002/WD-xquery-20020816/#combining_seq

XQuery Conditional Expressions

To learn more about conditional expressions based on the if, then, and else keywords, see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#id-conditionals>

XQuery Logical Expressions

To learn more about the and and or logical expressions, see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#id-logical-expressions>

XQuery Comparison Expressions

To learn more comparison expressions, see the following URL:

<http://www.w3.org/TR/2002/WD-xquery-20020816/#id-comparisons>

XML Element Declarations

To learn more about XML element declarations, see the following URL:

<http://www.w3.org/TR/2004/REC-xml-20040204/#elemdecls>

WebLogic Workshop uses XML Schemas (and not DTDs) for the validation of XML. To learn more about XML element type declarations with XML Schemas, see the following URL:

http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/#cElement_Declarations

XML Attribute Declarations

To learn more about XML attribute declarations, see the following URL:

<http://www.w3.org/TR/2004/REC-xml-20040204/#attdecls>

WebLogic Workshop uses XML Schemas (and not DTDs) for the validation of XML. To learn more about XML attribute declarations with XML Schemas, see the following URL:

http://www.w3.org/TR/xmlschema-1/#cAttribute_Declarations

XML CDATA Sections

To learn more about XML CDATA sections, see the following URL:

<http://www.w3.org/TR/2004/REC-xml-20040204/#sec-cdata-sect>

XML Comments

To learn more about XML comments, see the following URL:

<http://www.w3.org/TR/2004/REC-xml-20040204/#sec-comments>

XML Prolog and Document Type Declaration

To learn more about XML prolog and document type declarations, see the following URL:

<http://www.w3.org/TR/2004/REC-xml-20040204/#sec-prolog-dtd>