| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/XMLStreamReader.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/xml/stream/XMLStreamException.html)   [**NEXT CLASS**](http://docs.google.com/javax/xml/stream/XMLStreamWriter.html) | [**FRAMES**](http://docs.google.com/index.html?javax/xml/stream/XMLStreamReader.html)    [**NO FRAMES**](http://docs.google.com/XMLStreamReader.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#tyjcwt) | DETAIL: FIELD | CONSTR | [METHOD](#3dy6vkm) |

## **javax.xml.stream**

Interface XMLStreamReader

**All Superinterfaces:** [XMLStreamConstants](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html) **All Known Implementing Classes:** [StreamReaderDelegate](http://docs.google.com/javax/xml/stream/util/StreamReaderDelegate.html)

public interface **XMLStreamReader**extends [XMLStreamConstants](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html)

The XMLStreamReader interface allows forward, read-only access to XML. It is designed to be the lowest level and most efficient way to read XML data.

The XMLStreamReader is designed to iterate over XML using next() and hasNext(). The data can be accessed using methods such as getEventType(), getNamespaceURI(), getLocalName() and getText();

The [next()](#4d34og8) method causes the reader to read the next parse event. The next() method returns an integer which identifies the type of event just read.

The event type can be determined using [getEventType()](#3o7alnk).

Parsing events are defined as the XML Declaration, a DTD, start tag, character data, white space, end tag, comment, or processing instruction. An attribute or namespace event may be encountered at the root level of a document as the result of a query operation.

For XML 1.0 compliance an XML processor must pass the identifiers of declared unparsed entities, notation declarations and their associated identifiers to the application. This information is provided through the property API on this interface. The following two properties allow access to this information: javax.xml.stream.notations and javax.xml.stream.entities. When the current event is a DTD the following call will return a list of Notations List l = (List) getProperty("javax.xml.stream.notations"); The following call will return a list of entity declarations: List l = (List) getProperty("javax.xml.stream.entities"); These properties can only be accessed during a DTD event and are defined to return null if the information is not available.

The following table describes which methods are valid in what state. If a method is called in an invalid state the method will throw a java.lang.IllegalStateException.

| Valid methods for each state | |
| --- | --- |
| Event Type | Valid Methods |
| All States | getProperty(), hasNext(), require(), close(), getNamespaceURI(), isStartElement(), isEndElement(), isCharacters(), isWhiteSpace(), getNamespaceContext(), getEventType(),getLocation(), hasText(), hasName() |
| START\_ELEMENT | next(), getName(), getLocalName(), hasName(), getPrefix(), getAttributeXXX(), isAttributeSpecified(), getNamespaceXXX(), getElementText(), nextTag() |
| ATTRIBUTE | next(), nextTag() getAttributeXXX(), isAttributeSpecified(), |
| NAMESPACE | next(), nextTag() getNamespaceXXX() |
| END\_ELEMENT | next(), getName(), getLocalName(), hasName(), getPrefix(), getNamespaceXXX(), nextTag() |
| CHARACTERS | next(), getTextXXX(), nextTag() |
| CDATA | next(), getTextXXX(), nextTag() |
| COMMENT | next(), getTextXXX(), nextTag() |
| SPACE | next(), getTextXXX(), nextTag() |
| START\_DOCUMENT | next(), getEncoding(), getVersion(), isStandalone(), standaloneSet(), getCharacterEncodingScheme(), nextTag() |
| END\_DOCUMENT | close() |
| PROCESSING\_INSTRUCTION | next(), getPITarget(), getPIData(), nextTag() |
| ENTITY\_REFERENCE | next(), getLocalName(), getText(), nextTag() |
| DTD | next(), getText(), nextTag() |

**Since:** 1.6 **See Also:**[XMLEvent](http://docs.google.com/javax/xml/stream/events/XMLEvent.html), [XMLInputFactory](http://docs.google.com/javax/xml/stream/XMLInputFactory.html), [XMLStreamWriter](http://docs.google.com/javax/xml/stream/XMLStreamWriter.html)

| **Field Summary** | |
| --- | --- |

| **Fields inherited from interface javax.xml.stream.**[**XMLStreamConstants**](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html) |
| --- |
| [ATTRIBUTE](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#ATTRIBUTE), [CDATA](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#CDATA), [CHARACTERS](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#CHARACTERS), [COMMENT](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#COMMENT), [DTD](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#DTD), [END\_DOCUMENT](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#END_DOCUMENT), [END\_ELEMENT](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#END_ELEMENT), [ENTITY\_DECLARATION](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#ENTITY_DECLARATION), [ENTITY\_REFERENCE](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#ENTITY_REFERENCE), [NAMESPACE](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#NAMESPACE), [NOTATION\_DECLARATION](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#NOTATION_DECLARATION), [PROCESSING\_INSTRUCTION](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#PROCESSING_INSTRUCTION), [SPACE](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#SPACE), [START\_DOCUMENT](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#START_DOCUMENT), [START\_ELEMENT](http://docs.google.com/javax/xml/stream/XMLStreamConstants.html#START_ELEMENT) |

| **Method Summary** | |
| --- | --- |
| void | [**close**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#close())()            Frees any resources associated with this Reader. |
| int | [**getAttributeCount**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getAttributeCount())()            Returns the count of attributes on this START\_ELEMENT, this method is only valid on a START\_ELEMENT or ATTRIBUTE. |
| [String](http://docs.google.com/java/lang/String.html) | [**getAttributeLocalName**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getAttributeLocalName(int))(int index)            Returns the localName of the attribute at the provided index |
| [QName](http://docs.google.com/javax/xml/namespace/QName.html) | [**getAttributeName**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getAttributeName(int))(int index)            Returns the qname of the attribute at the provided index |
| [String](http://docs.google.com/java/lang/String.html) | [**getAttributeNamespace**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getAttributeNamespace(int))(int index)            Returns the namespace of the attribute at the provided index |
| [String](http://docs.google.com/java/lang/String.html) | [**getAttributePrefix**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getAttributePrefix(int))(int index)            Returns the prefix of this attribute at the provided index |
| [String](http://docs.google.com/java/lang/String.html) | [**getAttributeType**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getAttributeType(int))(int index)            Returns the XML type of the attribute at the provided index |
| [String](http://docs.google.com/java/lang/String.html) | [**getAttributeValue**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getAttributeValue(int))(int index)            Returns the value of the attribute at the index |
| [String](http://docs.google.com/java/lang/String.html) | [**getAttributeValue**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getAttributeValue(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) namespaceURI, [String](http://docs.google.com/java/lang/String.html) localName)            Returns the normalized attribute value of the attribute with the namespace and localName If the namespaceURI is null the namespace is not checked for equality |
| [String](http://docs.google.com/java/lang/String.html) | [**getCharacterEncodingScheme**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getCharacterEncodingScheme())()            Returns the character encoding declared on the xml declaration Returns null if none was declared |
| [String](http://docs.google.com/java/lang/String.html) | [**getElementText**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getElementText())()            Reads the content of a text-only element, an exception is thrown if this is not a text-only element. |
| [String](http://docs.google.com/java/lang/String.html) | [**getEncoding**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getEncoding())()            Return input encoding if known or null if unknown. |
| int | [**getEventType**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getEventType())()            Returns an integer code that indicates the type of the event the cursor is pointing to. |
| [String](http://docs.google.com/java/lang/String.html) | [**getLocalName**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getLocalName())()            Returns the (local) name of the current event. |
| [Location](http://docs.google.com/javax/xml/stream/Location.html) | [**getLocation**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getLocation())()            Return the current location of the processor. |
| [QName](http://docs.google.com/javax/xml/namespace/QName.html) | [**getName**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getName())()            Returns a QName for the current START\_ELEMENT or END\_ELEMENT event |
| [NamespaceContext](http://docs.google.com/javax/xml/namespace/NamespaceContext.html) | [**getNamespaceContext**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getNamespaceContext())()            Returns a read only namespace context for the current position. |
| int | [**getNamespaceCount**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getNamespaceCount())()            Returns the count of namespaces declared on this START\_ELEMENT or END\_ELEMENT, this method is only valid on a START\_ELEMENT, END\_ELEMENT or NAMESPACE. |
| [String](http://docs.google.com/java/lang/String.html) | [**getNamespacePrefix**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getNamespacePrefix(int))(int index)            Returns the prefix for the namespace declared at the index. |
| [String](http://docs.google.com/java/lang/String.html) | [**getNamespaceURI**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getNamespaceURI())()            If the current event is a START\_ELEMENT or END\_ELEMENT this method returns the URI of the prefix or the default namespace. |
| [String](http://docs.google.com/java/lang/String.html) | [**getNamespaceURI**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getNamespaceURI(int))(int index)            Returns the uri for the namespace declared at the index. |
| [String](http://docs.google.com/java/lang/String.html) | [**getNamespaceURI**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getNamespaceURI(java.lang.String))([String](http://docs.google.com/java/lang/String.html) prefix)            Return the uri for the given prefix. |
| [String](http://docs.google.com/java/lang/String.html) | [**getPIData**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getPIData())()            Get the data section of a processing instruction |
| [String](http://docs.google.com/java/lang/String.html) | [**getPITarget**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getPITarget())()            Get the target of a processing instruction |
| [String](http://docs.google.com/java/lang/String.html) | [**getPrefix**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getPrefix())()            Returns the prefix of the current event or null if the event does not have a prefix |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getProperty**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getProperty(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Get the value of a feature/property from the underlying implementation |
| [String](http://docs.google.com/java/lang/String.html) | [**getText**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getText())()            Returns the current value of the parse event as a string, this returns the string value of a CHARACTERS event, returns the value of a COMMENT, the replacement value for an ENTITY\_REFERENCE, the string value of a CDATA section, the string value for a SPACE event, or the String value of the internal subset of the DTD. |
| char[] | [**getTextCharacters**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getTextCharacters())()            Returns an array which contains the characters from this event. |
| int | [**getTextCharacters**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getTextCharacters(int,%20char%5B%5D,%20int,%20int))(int sourceStart, char[] target, int targetStart, int length)            Gets the the text associated with a CHARACTERS, SPACE or CDATA event. |
| int | [**getTextLength**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getTextLength())()            Returns the length of the sequence of characters for this Text event within the text character array. |
| int | [**getTextStart**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getTextStart())()            Returns the offset into the text character array where the first character (of this text event) is stored. |
| [String](http://docs.google.com/java/lang/String.html) | [**getVersion**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#getVersion())()            Get the xml version declared on the xml declaration Returns null if none was declared |
| boolean | [**hasName**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#hasName())()            returns true if the current event has a name (is a START\_ELEMENT or END\_ELEMENT) returns false otherwise |
| boolean | [**hasNext**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#hasNext())()            Returns true if there are more parsing events and false if there are no more events. |
| boolean | [**hasText**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#hasText())()            Return true if the current event has text, false otherwise The following events have text: CHARACTERS,DTD ,ENTITY\_REFERENCE, COMMENT, SPACE |
| boolean | [**isAttributeSpecified**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#isAttributeSpecified(int))(int index)            Returns a boolean which indicates if this attribute was created by default |
| boolean | [**isCharacters**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#isCharacters())()            Returns true if the cursor points to a character data event |
| boolean | [**isEndElement**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#isEndElement())()            Returns true if the cursor points to an end tag (otherwise false) |
| boolean | [**isStandalone**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#isStandalone())()            Get the standalone declaration from the xml declaration |
| boolean | [**isStartElement**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#isStartElement())()            Returns true if the cursor points to a start tag (otherwise false) |
| boolean | [**isWhiteSpace**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#isWhiteSpace())()            Returns true if the cursor points to a character data event that consists of all whitespace |
| int | [**next**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#next())()            Get next parsing event - a processor may return all contiguous character data in a single chunk, or it may split it into several chunks. |
| int | [**nextTag**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#nextTag())()            Skips any white space (isWhiteSpace() returns true), COMMENT, or PROCESSING\_INSTRUCTION, until a START\_ELEMENT or END\_ELEMENT is reached. |
| void | [**require**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#require(int,%20java.lang.String,%20java.lang.String))(int type, [String](http://docs.google.com/java/lang/String.html) namespaceURI, [String](http://docs.google.com/java/lang/String.html) localName)            Test if the current event is of the given type and if the namespace and name match the current namespace and name of the current event. |
| boolean | [**standaloneSet**](http://docs.google.com/javax/xml/stream/XMLStreamReader.html#standaloneSet())()            Checks if standalone was set in the document |

| **Method Detail** |
| --- |

### getProperty

[Object](http://docs.google.com/java/lang/Object.html) **getProperty**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html)

Get the value of a feature/property from the underlying implementation

**Parameters:**name - The name of the property, may not be null **Returns:**The value of the property **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if name is null

### next

int **next**()  
 throws [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html)

Get next parsing event - a processor may return all contiguous character data in a single chunk, or it may split it into several chunks. If the property javax.xml.stream.isCoalescing is set to true element content must be coalesced and only one CHARACTERS event must be returned for contiguous element content or CDATA Sections. By default entity references must be expanded and reported transparently to the application. An exception will be thrown if an entity reference cannot be expanded. If element content is empty (i.e. content is "") then no CHARACTERS event will be reported.

Given the following XML:

<foo><!--description-->content text<![CDATA[<greeting>Hello</greeting>]]>other content</foo>

The behavior of calling next() when being on foo will be:

1- the comment (COMMENT)

2- then the characters section (CHARACTERS)

3- then the CDATA section (another CHARACTERS)

4- then the next characters section (another CHARACTERS)

5- then the END\_ELEMENT

**NOTE:** empty element (such as <tag/>) will be reported with two separate events: START\_ELEMENT, END\_ELEMENT - This preserves parsing equivalency of empty element to <tag></tag>. This method will throw an IllegalStateException if it is called after hasNext() returns false.

**Returns:**the integer code corresponding to the current parse event **Throws:** NoSuchElementException - if this is called when hasNext() returns false [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html) - if there is an error processing the underlying XML source**See Also:**[XMLEvent](http://docs.google.com/javax/xml/stream/events/XMLEvent.html)

### require

void **require**(int type,  
 [String](http://docs.google.com/java/lang/String.html) namespaceURI,  
 [String](http://docs.google.com/java/lang/String.html) localName)  
 throws [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html)

Test if the current event is of the given type and if the namespace and name match the current namespace and name of the current event. If the namespaceURI is null it is not checked for equality, if the localName is null it is not checked for equality.

**Parameters:**type - the event typenamespaceURI - the uri of the event, may be nulllocalName - the localName of the event, may be null **Throws:** [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html) - if the required values are not matched.

### getElementText

[String](http://docs.google.com/java/lang/String.html) **getElementText**()  
 throws [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html)

Reads the content of a text-only element, an exception is thrown if this is not a text-only element. Regardless of value of javax.xml.stream.isCoalescing this method always returns coalesced content.

Precondition: the current event is START\_ELEMENT.

Postcondition: the current event is the corresponding END\_ELEMENT.

The method does the following (implementations are free to optimized but must do equivalent processing):

if(getEventType() != XMLStreamConstants.START\_ELEMENT) {  
 throw new XMLStreamException(  
 "parser must be on START\_ELEMENT to read next text", getLocation());  
 }  
 int eventType = next();  
 StringBuffer content = new StringBuffer();  
 while(eventType != XMLStreamConstants.END\_ELEMENT ) {  
 if(eventType == XMLStreamConstants.CHARACTERS  
 || eventType == XMLStreamConstants.CDATA  
 || eventType == XMLStreamConstants.SPACE  
 || eventType == XMLStreamConstants.ENTITY\_REFERENCE) {  
 buf.append(getText());  
 } else if(eventType == XMLStreamConstants.PROCESSING\_INSTRUCTION  
 || eventType == XMLStreamConstants.COMMENT) {  
 // skipping  
 } else if(eventType == XMLStreamConstants.END\_DOCUMENT) {  
 throw new XMLStreamException(  
 "unexpected end of document when reading element text content", this);  
 } else if(eventType == XMLStreamConstants.START\_ELEMENT) {  
 throw new XMLStreamException(  
 "element text content may not contain START\_ELEMENT", getLocation());  
 } else {  
 throw new XMLStreamException(  
 "Unexpected event type "+eventType, getLocation());  
 }  
 eventType = next();  
 }  
 return buf.toString();

**Throws:** [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html) - if the current event is not a START\_ELEMENT or if a non text element is encountered

### nextTag

int **nextTag**()  
 throws [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html)

Skips any white space (isWhiteSpace() returns true), COMMENT, or PROCESSING\_INSTRUCTION, until a START\_ELEMENT or END\_ELEMENT is reached. If other than white space characters, COMMENT, PROCESSING\_INSTRUCTION, START\_ELEMENT, END\_ELEMENT are encountered, an exception is thrown. This method should be used when processing element-only content seperated by white space.

Precondition: none

Postcondition: the current event is START\_ELEMENT or END\_ELEMENT and cursor may have moved over any whitespace event.

Essentially it does the following (implementations are free to optimized but must do equivalent processing):

int eventType = next();  
 while((eventType == XMLStreamConstants.CHARACTERS && isWhiteSpace()) // skip whitespace  
 || (eventType == XMLStreamConstants.CDATA && isWhiteSpace())   
 // skip whitespace  
 || eventType == XMLStreamConstants.SPACE  
 || eventType == XMLStreamConstants.PROCESSING\_INSTRUCTION  
 || eventType == XMLStreamConstants.COMMENT  
 ) {  
 eventType = next();  
 }  
 if (eventType != XMLStreamConstants.START\_ELEMENT && eventType != XMLStreamConstants.END\_ELEMENT) {  
 throw new String XMLStreamException("expected start or end tag", getLocation());  
 }  
 return eventType;

**Returns:**the event type of the element read (START\_ELEMENT or END\_ELEMENT) **Throws:** [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html) - if the current event is not white space, PROCESSING\_INSTRUCTION, START\_ELEMENT or END\_ELEMENT NoSuchElementException - if this is called when hasNext() returns false

### hasNext

boolean **hasNext**()  
 throws [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html)

Returns true if there are more parsing events and false if there are no more events. This method will return false if the current state of the XMLStreamReader is END\_DOCUMENT

**Returns:**true if there are more events, false otherwise **Throws:** [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html) - if there is a fatal error detecting the next state

### close

void **close**()  
 throws [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html)

Frees any resources associated with this Reader. This method does not close the underlying input source.

**Throws:** [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html) - if there are errors freeing associated resources

### getNamespaceURI

[String](http://docs.google.com/java/lang/String.html) **getNamespaceURI**([String](http://docs.google.com/java/lang/String.html) prefix)

Return the uri for the given prefix. The uri returned depends on the current state of the processor.

**NOTE:**The 'xml' prefix is bound as defined in [Namespaces in XML](http://www.w3.org/TR/REC-xml-names/#ns-using) specification to "http://www.w3.org/XML/1998/namespace".

**NOTE:** The 'xmlns' prefix must be resolved to following namespace <http://www.w3.org/2000/xmlns/>

**Parameters:**prefix - The prefix to lookup, may not be null **Returns:**the uri bound to the given prefix or null if it is not bound **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the prefix is null

### isStartElement

boolean **isStartElement**()

Returns true if the cursor points to a start tag (otherwise false)

**Returns:**true if the cursor points to a start tag, false otherwise

### isEndElement

boolean **isEndElement**()

Returns true if the cursor points to an end tag (otherwise false)

**Returns:**true if the cursor points to an end tag, false otherwise

### isCharacters

boolean **isCharacters**()

Returns true if the cursor points to a character data event

**Returns:**true if the cursor points to character data, false otherwise

### isWhiteSpace

boolean **isWhiteSpace**()

Returns true if the cursor points to a character data event that consists of all whitespace

**Returns:**true if the cursor points to all whitespace, false otherwise

### getAttributeValue

[String](http://docs.google.com/java/lang/String.html) **getAttributeValue**([String](http://docs.google.com/java/lang/String.html) namespaceURI,  
 [String](http://docs.google.com/java/lang/String.html) localName)

Returns the normalized attribute value of the attribute with the namespace and localName If the namespaceURI is null the namespace is not checked for equality

**Parameters:**namespaceURI - the namespace of the attributelocalName - the local name of the attribute, cannot be null **Returns:**returns the value of the attribute , returns null if not found **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT or ATTRIBUTE

### getAttributeCount

int **getAttributeCount**()

Returns the count of attributes on this START\_ELEMENT, this method is only valid on a START\_ELEMENT or ATTRIBUTE. This count excludes namespace definitions. Attribute indices are zero-based.

**Returns:**returns the number of attributes **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT or ATTRIBUTE

### getAttributeName

[QName](http://docs.google.com/javax/xml/namespace/QName.html) **getAttributeName**(int index)

Returns the qname of the attribute at the provided index

**Parameters:**index - the position of the attribute **Returns:**the QName of the attribute **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT or ATTRIBUTE

### getAttributeNamespace

[String](http://docs.google.com/java/lang/String.html) **getAttributeNamespace**(int index)

Returns the namespace of the attribute at the provided index

**Parameters:**index - the position of the attribute **Returns:**the namespace URI (can be null) **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT or ATTRIBUTE

### getAttributeLocalName

[String](http://docs.google.com/java/lang/String.html) **getAttributeLocalName**(int index)

Returns the localName of the attribute at the provided index

**Parameters:**index - the position of the attribute **Returns:**the localName of the attribute **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT or ATTRIBUTE

### getAttributePrefix

[String](http://docs.google.com/java/lang/String.html) **getAttributePrefix**(int index)

Returns the prefix of this attribute at the provided index

**Parameters:**index - the position of the attribute **Returns:**the prefix of the attribute **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT or ATTRIBUTE

### getAttributeType

[String](http://docs.google.com/java/lang/String.html) **getAttributeType**(int index)

Returns the XML type of the attribute at the provided index

**Parameters:**index - the position of the attribute **Returns:**the XML type of the attribute **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT or ATTRIBUTE

### getAttributeValue

[String](http://docs.google.com/java/lang/String.html) **getAttributeValue**(int index)

Returns the value of the attribute at the index

**Parameters:**index - the position of the attribute **Returns:**the attribute value **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT or ATTRIBUTE

### isAttributeSpecified

boolean **isAttributeSpecified**(int index)

Returns a boolean which indicates if this attribute was created by default

**Parameters:**index - the position of the attribute **Returns:**true if this is a default attribute **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT or ATTRIBUTE

### getNamespaceCount

int **getNamespaceCount**()

Returns the count of namespaces declared on this START\_ELEMENT or END\_ELEMENT, this method is only valid on a START\_ELEMENT, END\_ELEMENT or NAMESPACE. On an END\_ELEMENT the count is of the namespaces that are about to go out of scope. This is the equivalent of the information reported by SAX callback for an end element event.

**Returns:**returns the number of namespace declarations on this specific element **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT, END\_ELEMENT or NAMESPACE

### getNamespacePrefix

[String](http://docs.google.com/java/lang/String.html) **getNamespacePrefix**(int index)

Returns the prefix for the namespace declared at the index. Returns null if this is the default namespace declaration

**Parameters:**index - the position of the namespace declaration **Returns:**returns the namespace prefix **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT, END\_ELEMENT or NAMESPACE

### getNamespaceURI

[String](http://docs.google.com/java/lang/String.html) **getNamespaceURI**(int index)

Returns the uri for the namespace declared at the index.

**Parameters:**index - the position of the namespace declaration **Returns:**returns the namespace uri **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT, END\_ELEMENT or NAMESPACE

### getNamespaceContext

[NamespaceContext](http://docs.google.com/javax/xml/namespace/NamespaceContext.html) **getNamespaceContext**()

Returns a read only namespace context for the current position. The context is transient and only valid until a call to next() changes the state of the reader.

**Returns:**return a namespace context

### getEventType

int **getEventType**()

Returns an integer code that indicates the type of the event the cursor is pointing to.

### getText

[String](http://docs.google.com/java/lang/String.html) **getText**()

Returns the current value of the parse event as a string, this returns the string value of a CHARACTERS event, returns the value of a COMMENT, the replacement value for an ENTITY\_REFERENCE, the string value of a CDATA section, the string value for a SPACE event, or the String value of the internal subset of the DTD. If an ENTITY\_REFERENCE has been resolved, any character data will be reported as CHARACTERS events.

**Returns:**the current text or null **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this state is not a valid text state.

### getTextCharacters

char[] **getTextCharacters**()

Returns an array which contains the characters from this event. This array should be treated as read-only and transient. I.e. the array will contain the text characters until the XMLStreamReader moves on to the next event. Attempts to hold onto the character array beyond that time or modify the contents of the array are breaches of the contract for this interface.

**Returns:**the current text or an empty array **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this state is not a valid text state.

### getTextCharacters

int **getTextCharacters**(int sourceStart,  
 char[] target,  
 int targetStart,  
 int length)  
 throws [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html)

Gets the the text associated with a CHARACTERS, SPACE or CDATA event. Text starting a "sourceStart" is copied into "target" starting at "targetStart". Up to "length" characters are copied. The number of characters actually copied is returned. The "sourceStart" argument must be greater or equal to 0 and less than or equal to the number of characters associated with the event. Usually, one requests text starting at a "sourceStart" of 0. If the number of characters actually copied is less than the "length", then there is no more text. Otherwise, subsequent calls need to be made until all text has been retrieved. For example: int length = 1024; char[] myBuffer = new char[ length ]; for ( int sourceStart = 0 ; ; sourceStart += length ) { int nCopied = stream.getTextCharacters( sourceStart, myBuffer, 0, length ); if (nCopied < length) break; } XMLStreamException may be thrown if there are any XML errors in the underlying source. The "targetStart" argument must be greater than or equal to 0 and less than the length of "target", Length must be greater than 0 and "targetStart + length" must be less than or equal to length of "target".

**Parameters:**sourceStart - the index of the first character in the source array to copytarget - the destination arraytargetStart - the start offset in the target arraylength - the number of characters to copy **Returns:**the number of characters actually copied **Throws:** [XMLStreamException](http://docs.google.com/javax/xml/stream/XMLStreamException.html) - if the underlying XML source is not well-formed [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if targetStart < 0 or > than the length of target [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if length < 0 or targetStart + length > length of target [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this method is not supported [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - is if target is null

### getTextStart

int **getTextStart**()

Returns the offset into the text character array where the first character (of this text event) is stored.

**Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this state is not a valid text state.

### getTextLength

int **getTextLength**()

Returns the length of the sequence of characters for this Text event within the text character array.

**Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this state is not a valid text state.

### getEncoding

[String](http://docs.google.com/java/lang/String.html) **getEncoding**()

Return input encoding if known or null if unknown.

**Returns:**the encoding of this instance or null

### hasText

boolean **hasText**()

Return true if the current event has text, false otherwise The following events have text: CHARACTERS,DTD ,ENTITY\_REFERENCE, COMMENT, SPACE

### getLocation

[Location](http://docs.google.com/javax/xml/stream/Location.html) **getLocation**()

Return the current location of the processor. If the Location is unknown the processor should return an implementation of Location that returns -1 for the location and null for the publicId and systemId. The location information is only valid until next() is called.

### getName

[QName](http://docs.google.com/javax/xml/namespace/QName.html) **getName**()

Returns a QName for the current START\_ELEMENT or END\_ELEMENT event

**Returns:**the QName for the current START\_ELEMENT or END\_ELEMENT event **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this is not a START\_ELEMENT or END\_ELEMENT

### getLocalName

[String](http://docs.google.com/java/lang/String.html) **getLocalName**()

Returns the (local) name of the current event. For START\_ELEMENT or END\_ELEMENT returns the (local) name of the current element. For ENTITY\_REFERENCE it returns entity name. The current event must be START\_ELEMENT or END\_ELEMENT, or ENTITY\_REFERENCE

**Returns:**the localName **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this not a START\_ELEMENT, END\_ELEMENT or ENTITY\_REFERENCE

### hasName

boolean **hasName**()

returns true if the current event has a name (is a START\_ELEMENT or END\_ELEMENT) returns false otherwise

### getNamespaceURI

[String](http://docs.google.com/java/lang/String.html) **getNamespaceURI**()

If the current event is a START\_ELEMENT or END\_ELEMENT this method returns the URI of the prefix or the default namespace. Returns null if the event does not have a prefix.

**Returns:**the URI bound to this elements prefix, the default namespace, or null

### getPrefix

[String](http://docs.google.com/java/lang/String.html) **getPrefix**()

Returns the prefix of the current event or null if the event does not have a prefix

**Returns:**the prefix or null

### getVersion

[String](http://docs.google.com/java/lang/String.html) **getVersion**()

Get the xml version declared on the xml declaration Returns null if none was declared

**Returns:**the XML version or null

### isStandalone

boolean **isStandalone**()

Get the standalone declaration from the xml declaration

**Returns:**true if this is standalone, or false otherwise

### standaloneSet

boolean **standaloneSet**()

Checks if standalone was set in the document

**Returns:**true if standalone was set in the document, or false otherwise

### getCharacterEncodingScheme

[String](http://docs.google.com/java/lang/String.html) **getCharacterEncodingScheme**()

Returns the character encoding declared on the xml declaration Returns null if none was declared

**Returns:**the encoding declared in the document or null

### getPITarget

[String](http://docs.google.com/java/lang/String.html) **getPITarget**()

Get the target of a processing instruction

**Returns:**the target or null

### getPIData

[String](http://docs.google.com/java/lang/String.html) **getPIData**()

Get the data section of a processing instruction

**Returns:**the data or null

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/XMLStreamReader.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/xml/stream/XMLStreamException.html)   [**NEXT CLASS**](http://docs.google.com/javax/xml/stream/XMLStreamWriter.html) | [**FRAMES**](http://docs.google.com/index.html?javax/xml/stream/XMLStreamReader.html)    [**NO FRAMES**](http://docs.google.com/XMLStreamReader.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#tyjcwt) | DETAIL: FIELD | CONSTR | [METHOD](#3dy6vkm) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).