# XmlStreamReader Class

The `XmlStreamReader` class provides methods for forward, read-only access to XML data. You can pull data from XML or skip unwanted events. You can parse nested XML content that's up to 50 nodes deep.

## Namespace

System

## Usage

The `XmlStreamReader` class is similar to the XMLStreamReader utility class from StAX (Streaming API for XML). StAX is an API to read and write XML documents, originating from the Java programming language community.

> **Note**
>
> The `XmlStreamReader` class in Apex is based on its counterpart in Java. See `Java XMLStreamReader class.`

- XmlStreamReader Constructors
- XmlStreamReader Methods

### See Also

- *Apex Developer Guide*: Reading XML Using Streams

## XmlStreamReader Constructors

The following are constructors for `XmlStreamReader.`

- **XmlStreamReader(xmlInput)**
  Creates a new instance of the `XmlStreamReader` class for the specified XML input.

### XmlStreamReader(xmlInput)

Creates a new instance of the `XmlStreamReader` class for the specified XML input.

#### Signature

```
public XmlStreamReader(String xmlInput)
```

#### Parameters

***xmlInput***

Type: String

The XML string input.

Returns the number of attributes on the start element, excluding namespace definitions.

- **getAttributeLocalName(index)**
  Returns the local name of the attribute at the specified index.
- **getAttributeNamespace(index)**
  Returns the namespace URI of the attribute at the specified index.
- **getAttributePrefix(index)**
  Returns the prefix of this attribute at the specified index.
- **getAttributeType(index)**
  Returns the XML type of the attribute at the specified index.
- **getAttributeValue(namespaceUri, localName)**
  Returns the value of the attribute in the specified *localName* at the specified URI.
- **getAttributeValueAt(index)**
  Returns the value of the attribute at the specified index.
- **getEventType()**
  Returns the type of XML event the cursor is pointing to.
- **getLocalName()**
  Returns the local name of the current event.
- **getLocation()**
  Return the current location of the cursor.
- **getNamespace()**
  If the current event is a start element or end element, this method returns the URI of the prefix or the default namespace.
- **getNamespaceCount()**
  Returns the number of namespaces declared on a start element or end element.
- **getNamespacePrefix(index)**
  Returns the prefix for the namespace declared at the index.
- **getNamespaceURI(prefix)**
  Return the URI for the given prefix.
- **getNamespaceURIAt(index)**
  Returns the URI for the namespace declared at the index.
- **getPIData()**
  Returns the data section of a processing instruction.
- **getPITarget()**
  Returns the target section of a processing instruction.
- **getPrefix()**
  Returns the prefix of the current XML event or `null` if the event does not have a prefix.
- **getText()**
  Returns the current value of the XML event as a string.
- **getVersion()**
  Returns the XML version specified on the XML declaration. Returns `null` if none was declared.
- **hasName()**
  Returns `true` if the current XML event has a name. Returns `false` otherwise.
- **hasNext()**
  Returns `true` if there are more XML events and `false` if there are no more XML events.
- **hasText()**
  Returns `true` if the current event has text, `false` otherwise.
- **isCharacters()**
  Returns `true` if the cursor points to a character data XML event. Otherwise, returns `false`.
- **isEndElement()**
  Returns `true` if the cursor points to an end tag. Otherwise, it returns `false`.

space. Otherwise it returns `false`.

- **next()**
  Reads the next XML event. A processor may return all contiguous character data in a single chunk, or it may split it into several chunks. Returns an integer which indicates the type of event.
- **nextTag()**
  Skips any white space (the `isWhiteSpace` method returns `true`), comment, or processing instruction XML events, until a start element or end element is reached. Returns the index for that XML event.
- **setCoalescing(returnAsSingleBlock)**
  If you specify `true` for *returnAsSingleBlock*, text is returned in a single block, from a start element to the first end element or the next start element, whichever comes first. If you specify it as `false`, the parser may return text in multiple blocks.
- **setNamespaceAware(isNamespaceAware)**
  If you specify `true` for *isNamespaceAware*, the parser recognizes namespace. If you specify it as `false`, the parser does not. The default value is `true`.
- **toString()**
  Returns a string containing the length of the input XML given to `XmlStreamReader` and the first 50 characters of the input XML.

## getAttributeCount()

Returns the number of attributes on the start element, excluding namespace definitions.

### Signature

```
public Integer getAttributeCount()
```

### Return Value

Type: Integer

### Usage

This method is only valid on a start element or attribute XML events. The count for the number of attributes for an attribute XML event starts with zero.

## getAttributeLocalName(index)

Returns the local name of the attribute at the specified index.

### Signature

```
public String getAttributeLocalName(Integer index)
```

### Parameters

*index*

Type: Integer

### Return Value

Type: String

### Usage

If there is no name, an empty string is returned. This method is only valid with start element or attribute XML events.

## getAttributeNamespace(index)

#### Parameters

*index*

Type: Integer

#### Return Value

Type: String

#### Usage

If no namespace is specified, `null` is returned. This method is only valid with start element or attribute XML events.

## getAttributePrefix(index)

Returns the prefix of this attribute at the specified index.

#### Signature

```
public String getAttributePrefix(Integer index)
```

#### Parameters

*index*

Type: Integer

#### Return Value

Type: String

#### Usage

If no prefix is specified, `null` is returned. This method is only valid with start element or attribute XML events.

## getAttributeType(index)

Returns the XML type of the attribute at the specified index.

#### Signature

```
public String getAttributeType(Integer index)
```

#### Parameters

*index*

Type: Integer

#### Return Value

Type: String

#### Usage

For example, `id` is an attribute type. This method is only valid with start element or attribute XML events.

## getAttributeValue(namespaceUri, localName)

Returns the value of the attribute in the specified *localName* at the specified URI.

*namespaceUri*

Type: String

*localName*

Type: String

#### Return Value

Type: String

#### Usage

Returns `null` if the value is not found. You must specify a value for *localName*. This method is only valid with start element or attribute XML events.

## getAttributeValueAt(index)

Returns the value of the attribute at the specified index.

#### Signature

```
public String getAttributeValueAt(Integer index)
```

#### Parameters

*index*

Type: Integer

#### Return Value

Type: String

#### Usage

This method is only valid with start element or attribute XML events.

## getEventType()

Returns the type of XML event the cursor is pointing to.

#### Signature

```
public System.XmlTag getEventType()
```

#### Return Value

Type: System.XmlTag

#### XmlTag Enum

The values for `XmlTag` are:

- `ATTRIBUTE`
- `CDATA`
- `CHARACTERS`
- `COMMENT`
- `DTD`
- `END_DOCUMENT`
- `END_ELEMENT`

- PROCESSING_INSTRUCTION
- SPACE
- START_DOCUMENT
- START_ELEMENT

## getLocalName()

Returns the local name of the current event.

### Signature

```
public String getLocalName()
```

### Return Value

Type: String

### Usage

For start element or end element XML events, it returns the local name of the current element. For the entity reference XML event, it returns the entity name. The current XML event must be start element, end element, or entity reference.

## getLocation()

Return the current location of the cursor.

### Signature

```
public String getLocation()
```

### Return Value

Type: String

### Usage

If the location is unknown, returns -1. The location information is only valid until the `next` method is called.

## getNamespace()

If the current event is a start element or end element, this method returns the URI of the prefix or the default namespace.

### Signature

```
public String getNamespace()
```

### Return Value

Type: String

### Usage

Returns `null` if the XML event does not have a prefix.

## getNamespaceCount()

Returns the number of namespaces declared on a start element or end element.

Type: Integer

#### Usage

This method is only valid on a start element, end element, or namespace XML event.

## getNamespacePrefix(index)

Returns the prefix for the namespace declared at the index.

#### Signature

```
public String getNamespacePrefix(Integer index)
```

#### Parameters

***index***

Type: Integer

#### Return Value

Type: String

#### Usage

Returns `null` if this is the default namespace declaration. This method is only valid on a start element, end element, or namespace XML event.

## getNamespaceURI(prefix)

Return the URI for the given prefix.

#### Signature

```
public String getNamespaceURI(String prefix)
```

#### Parameters

***prefix***

Type: String

#### Return Value

Type: String

#### Usage

The returned URI depends on the current state of the processor.

## getNamespaceURIAt(index)

Returns the URI for the namespace declared at the index.

#### Signature

```
public String getNamespaceURIAt(Integer index)
```

#### Parameters

***index***

Type: Integer

This method is only valid on a start element, end element, or namespace XML event.

## getPIData()

Returns the data section of a processing instruction.

### Signature

```
public String getPIData()
```

### Return Value

Type: String

## getPITarget()

Returns the target section of a processing instruction.

### Signature

```
public String getPITarget()
```

### Return Value

Type: String

## getPrefix()

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

### Signature

```
public String getPrefix()
```

### Return Value

Type: String

## getText()

Returns the current value of the XML event as a string.

### Signature

```
public String getText()
```

### Return Value

Type: String

### Usage

The valid values for the different events are:

- The string value of a character XML event
- The string value of a comment
- The replacement value for an entity reference. For example, assume `getText` reads the following XML snippet:

The `getText` method returns `Salesforce for Dummies`, not `&Title`.

- The string value of a CDATA section
- The string value for a space XML event
- The string value of the internal subset of the DTD

## getVersion()

Returns the XML version specified on the XML declaration. Returns `null` if none was declared.

### Signature

```
public String getVersion()
```

### Return Value

Type: String

## hasName()

Returns `true` if the current XML event has a name. Returns `false` otherwise.

### Signature

```
public Boolean hasName()
```

### Return Value

Type: Boolean

### Usage

This method is only valid for start element and stop element XML events.

## hasNext()

Returns `true` if there are more XML events and `false` if there are no more XML events.

### Signature

```
public Boolean hasNext()
```

### Return Value

Type: Boolean

### Usage

This method returns `false` if the current XML event is end document.

## hasText()

Returns `true` if the current event has text, `false` otherwise.

### Signature

```
public Boolean hasText()
```

### Return Value

Type: Boolean

### isCharacters()

Returns `true` if the cursor points to a character data XML event. Otherwise, returns `false`.

#### Signature

```
public Boolean isCharacters()
```

#### Return Value

Type: Boolean

## isEndElement()

Returns `true` if the cursor points to an end tag. Otherwise, it returns `false`.

#### Signature

```
public Boolean isEndElement()
```

#### Return Value

Type: Boolean

## isStartElement()

Returns `true` if the cursor points to a start tag. Otherwise, it returns `false`.

#### Signature

```
public Boolean isStartElement()
```

#### Return Value

Type: Boolean

## isWhiteSpace()

Returns `true` if the cursor points to a character data XML event that consists of all white space. Otherwise it returns `false`.

#### Signature

```
public Boolean isWhiteSpace()
```

#### Return Value

Type: Boolean

## next()

Reads the next XML event. A processor may return all contiguous character data in a single chunk, or it may split it into several chunks. Returns an integer which indicates the type of event.

#### Signature

```
public Integer next()
```

#### Return Value

Type: Integer

### Signature

```
public Integer nextTag()
```

### Return Value

Type: Integer

### Usage

This method throws an error if elements other than white space, comments, processing instruction, start elements or stop elements are encountered.

## setCoalescing(returnAsSingleBlock)

If you specify `true` for *returnAsSingleBlock*, text is returned in a single block, from a start element to the first end element or the next start element, whichever comes first. If you specify it as `false`, the parser may return text in multiple blocks.

### Signature

```
public Void setCoalescing(Boolean returnAsSingleBlock)
```

### Parameters

***returnAsSingleBlock***

Type: Boolean

### Return Value

Type: Void

## setNamespaceAware(isNamespaceAware)

If you specify `true` for *isNamespaceAware*, the parser recognizes namespace. If you specify it as `false`, the parser does not. The default value is `true`.

### Signature

```
public Void setNamespaceAware(Boolean isNamespaceAware)
```

### Parameters

***isNamespaceAware***

Type: Boolean

### Return Value

Type: Void

## toString()

Returns a string containing the length of the input XML given to `XmlStreamReader` and the first 50 characters of the input XML.

### Signature

```
public String toString()
```

### Return Value

Type: String



**DEVELOPER CENTERS**

- Heroku
- MuleSoft
- Tableau
- Commerce Cloud
- Lightning Design System
- Einstein
- Quip

**POPULAR RESOURCES**

- Documentation
- Component Library
- APIs
- Trailhead
- Sample Apps
- Podcasts
- AppExchange

**COMMUNITY**

- Trailblazer Community
- Events and Calendar
- Partner Community
- Blog
- Salesforce Admins
- Salesforce Architects