

# 1 System.Xml.XmlTextWriter Class

```
2 [ILAsm]  
3 .class public XmlTextWriter extends System.Xml.XmlWriter  
4 [C#]  
5 public class XmlTextWriter: XmlWriter
```

## 6 Assembly Info:

- 7 • *Name:* System.Xml
- 8 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- 9 • *Version:* 2.0.x.x
- 10 • *Attributes:*
  - 11 ○ CLSCompliantAttribute(true)

## 12 Summary

13 Represents a writer that provides a fast, non-cached, forward-only way of generating  
14 streams or files containing XML data that conforms to the W3C Extensible Markup  
15 Language (XML) 1.0 and the Namespaces in XML recommendations.

## 16 Inherits From: System.Xml.XmlWriter

17

18 **Library:** XML

19

20 **Thread Safety:** All public static members of this type are safe for multithreaded operations.  
21 No instance members are guaranteed to be thread safe.

22

## 23 Description

24 This class maintains a namespace stack corresponding to all the namespaces defined in  
25 the current element stack. Namespaces can be declared manually to override the  
26 current namespace declaration. Prefixes can be specified to associate with a namespace.  
27 If there are multiple namespace declarations mapping different prefixes to the same  
28 namespace URI, this class walks the stack of namespace declarations backwards and  
29 picks the closest one.

30

31 If namespace conflicts occur inside an element, this class resolves the conflict by  
32 generating alternate prefixes. The generated prefixes are named *ni*, where *n* is the  
33 literal character 'n' and *i* is a number beginning at one. The number is reset to one for  
34 each element. See the example section for a demonstration of this behavior.

35

36 Attributes which are associated with a namespace URI must have a prefix (default  
37 namespaces do not apply to attributes). This conforms to section 5.2 of the W3C  
38 Namespaces in XML recommendation. If an attribute references a namespace URI, but  
39 does not specify a prefix, the writer generates a prefix for the attribute.

40

41 When writing an empty element, an additional space is added between tag name and

1 the closing tag, for example `<item />`. This provides compatibility with older browsers.  
2  
3 When a `System.String` is used as method parameter, `null` and `System.String.Empty`  
4 are equivalent. `System.String.Empty` follows the W3C rules.  
5  
6 This class implements the `System.Xml.XmlWriter` class.

## 7 **Example**

8 The following example demonstrates how this class resolves namespace conflicts inside  
9 an element. In the example, the writer writes an element that contains two attributes.  
10 The element and both attributes have the same prefix but different namespaces. The  
11 resulting XML fragment is written to the console.  
12

```
13 [C#]
14 using System;
15 using System.Xml;
16
17 public class WriteFragment
18 {
19     public static void Main()
20     {
21         XmlTextWriter xWriter = new XmlTextWriter(Console.Out);
22         xWriter.WriteStartElement("prefix", "Element1", "namespace");
23         xWriter.WriteStartAttribute("prefix", "Attr1", "namespacel");
24         xWriter.WriteString("value1");
25         xWriter.WriteStartAttribute("prefix", "Attr2", "namespace2");
26         xWriter.WriteString("value2");
27         xWriter.Close();
28     }
29 }
```

30  
31 The output is

```
32 <prefix:Element1 n1:Attr1="value1" n2:Attr2="value2" xmlns:n2="namespace2"
33 xmlns:n1="namespacel" xmlns:prefix="namespace" />
```

35

# XmlTextWriter(System.String, System.Text.Encoding) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(string filename,
class System.Text.Encoding encoding)

[C#]
public XmlTextWriter(string filename, Encoding encoding)
```

## Summary

Constructs and initializes a new instance of the System.Xml.XmlTextWriter class using the specified file.

## Parameters

Parameter	Description
<i>filename</i>	A System.String specifying the path and name of the file to write to.
<i>encoding</i>	The System.Text.Encoding to generate, or null.

## Description

- If *filename* exists, it is truncated and overwritten with the new content.
- If *encoding* is null, the file is written as UTF-8 and the encoding attribute is omitted from the processing instruction.
- The following properties are initialized to the specified values:
  - System.Xml.XmlTextWriter.Formatting to System.Xml.Formatting.None.
  - System.Xml.XmlTextWriter.Indentation to 2.
  - System.Xml.XmlTextWriter.IndentChar to the space character.
  - System.Xml.XmlTextWriter.Namespaces to true.
  - System.Xml.XmlTextWriter.QuoteChar to the double quote character.
  - System.Xml.XmlTextWriter.WriteState to System.Xml.WriteState.Start.

## Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>filename</i> is <code>System.String.Empty</code> , contains only white space, or contains one or more implementation-specific invalid characters.  -or-  The encoding is not supported.
<b>System.ArgumentNullException</b>	<i>filename</i> is null.
<b>System.IO.DirectoryNotFoundException</b>	The directory path specified in <i>filename</i> does not exist.
<b>System.IO.IOException</b>	<i>filename</i> includes an invalid syntax for the path or file name.
<b>System.IO.PathTooLongException</b>	The specified path, file name, or both exceeds the system-defined maximum length.
<b>System.Security.SecurityException</b>	The caller does not have the required permissions.
<b>System.UnauthorizedAccessException</b>	Write access is not permitted by the operating system for the path specified in <i>filename</i> .

1

2 **Permissions**

Permission	Description
<b>System.Security.Permissions.FileIOPermission</b>	Requires permission to write to files. See <code>System.Security.Permissions.FileIOPermissionAccess.Write</code> .

3

4

# 1 XmlTextWriter(System.IO.Stream, 2 System.Text.Encoding) Constructor

```
3 [ILAsm]  
4 public rtspecialname specialname instance void .ctor(class  
5 System.IO.Stream w, class System.Text.Encoding encoding)  
  
6 [C#]  
7 public XmlTextWriter(Stream w, Encoding encoding)
```

## 8 Summary

9 Constructs and initializes a new instance of the `System.Xml.XmlTextWriter` class using  
10 the specified output stream.

## 11 Parameters

Parameter	Description
<i>w</i>	The <code>System.IO.Stream</code> to write to.
<i>encoding</i>	The <code>System.Text.Encoding</code> to generate, or null.

12

## 13 Description

14 If *encoding* is null, the stream is written as UTF-8 and the encoding attribute is omitted  
15 from the processing instruction.

16

17 The following properties are initialized to the specified values:

18

19 `System.Xml.XmlTextWriter.Formatting` to `System.Xml.Formatting.None`.

20

21 `System.Xml.XmlTextWriter.Indentation` to 2.

22

23 `System.Xml.XmlTextWriter.IndentChar` to the space character.

24

25 `System.Xml.XmlTextWriter.Namespaces` to true.

26

27 `System.Xml.XmlTextWriter.QuoteChar` to the double quote character.

28

29 `System.Xml.XmlTextWriter.WriteState` to `System.Xml.WriteState.Start`.

## 30 Exceptions

Exception	Condition
-----------	-----------

<b>System.ArgumentException</b>	<i>w</i> cannot be written to.  -or-  The encoding is not supported.
<b>System.ArgumentNullException</b>	<i>w</i> is null.

1

2

# 1 XmlTextWriter(System.IO.TextWriter)

## 2 Constructor

```
3 [ILAsm]  
4 public rtspecialname specialname instance void .ctor(class  
5 System.IO.TextWriter w)  
  
6 [C#]  
7 public XmlTextWriter(TextWriter w)
```

### 8 Summary

9 Constructs and initializes a new instance of the `System.Xml.XmlTextWriter` class.

### 10 Parameters

Parameter	Description
<code>w</code>	The <code>System.IO.TextWriter</code> to write to, initialized to the correct encoding.

### 11 Description

#### 12

13 The following properties are initialized to the specified values:  
14  
15 `System.Xml.XmlTextWriter.Formatting` to `System.Xml.Formatting.None`.  
16  
17 `System.Xml.XmlTextWriter.Indentation` to 2.  
18  
19 `System.Xml.XmlTextWriter.IndentChar` to the space character.  
20  
21 `System.Xml.XmlTextWriter.Namespaces` to true.  
22  
23 `System.Xml.XmlTextWriter.QuoteChar` to the double quote character.  
24  
25 `System.Xml.XmlTextWriter.WriteState` to `System.Xml.WriteState.Start`.  
26  
27 [*Note:* If a specific encoding is necessary, set the encoding using the constructor of `w`  
28 before instantiating the writer.  
29  
30 ]

31

# 1 XmlTextWriter.Close() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual void Close()  
4 [C#]  
5 public override void Close()
```

## 6 Summary

7 Closes the writer.

## 8 Description

9 This method closes all elements and attributes created by the  
10 `System.Xml.XmlTextWriter.WriteStartElement` and  
11 `System.Xml.XmlTextWriter.WriteStartAttribute` methods, respectively, that are  
12 open when the `System.Xml.XmlTextWriter.Close` method is called.

13  
14 This method calls the `System.Xml.XmlTextWriter.Flush` method to flush the  
15 underlying buffered stream and then closes the stream.

16  
17 This method sets the `System.Xml.XmlTextWriter.WriteState` to  
18 `System.Xml.WriteState.Closed`.

19

# 1 XmlTextWriter.Flush() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual void Flush()  
4 [C#]  
5 public override void Flush()
```

## 6 Summary

7 Clears all buffers and causes any buffered data to be written to the underlying stream.

## 8 Description

9 [Note: This method overrides System.Xml.XmlWriter.Flush.  
10 ]  
11 ]

12

# 1 XmlTextWriter.LookupPrefix(System.String)

## 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual string LookupPrefix(string ns)  
5 [C#]  
6 public override string LookupPrefix(string ns)
```

### 7 Summary

8 Returns the prefix defined in the current namespace scope for the specified namespace  
9 URI.

### 10 Parameters

Parameter	Description
<i>ns</i>	A <code>System.String</code> specifying the namespace URI.

### 11 Return Value

13 A `System.String` containing the corresponding prefix, or `System.String.Empty` if the  
14 prefix is not found and *ns* is the default namespace, or `null` if no matching namespace  
15 URI is found in the current scope.

### 16 Description

17 [Note: This method overrides `System.Xml.XmlWriter.LookupPrefix`.  
18 ]  
19 ]

### 20 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>ns</i> is null or <code>System.String.Empty</code> .

21

22

# 1 XmlTextWriter.WriteBase64(System.Byte[], 2 System.Int32, System.Int32) Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteBase64(class System.Byte[]  
5 buffer, int32 index, int32 count)  
  
6 [C#]  
7 public override void WriteBase64(byte[] buffer, int index, int count)
```

## 8 Summary

9 Encodes the specified binary bytes as Base64 and writes the resulting text.

## 10 Parameters

Parameter	Description
<i>buffer</i>	A <code>System.Byte</code> array containing the bytes to encode.
<i>index</i>	A <code>System.Int32</code> specifying the position within the array of the first byte to encode.
<i>count</i>	A <code>System.Int32</code> specifying the number of bytes to encode.

## 11 12 Description

13 [Note: Base64 encoding represents byte sequences in a text form comprised of the 65  
14 US-ASCII characters (A-Z, a-z, 0-9, +, /, =) where each character encodes 6 bits of the  
15 binary data.

16  
17 For more information on Base64 encoding, see RFC 2045  
18 (<http://www.ietf.org/rfc/rfc2045>).

19  
20 This method overrides `System.Xml.XmlWriter.WriteBase64`.

21 ]  
22

## 23 Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<i>buffer</i> is null.
<b>System.ArgumentException</b>	The buffer length minus <i>index</i> is less than

	<i>count</i> .
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> or <i>count</i> is less than zero.
<b>System.InvalidOperationException</b>	The <code>System.Xml.XmlTextWriter.WriteState</code> is <code>System.Xml.WriteState.Closed</code> .

1

2

# 1 XmlTextWriter.WriteBinHex(System.Byte[], 2 System.Int32, System.Int32) Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteBinHex(class System.Byte[]  
5 buffer, int32 index, int32 count)  
  
6 [C#]  
7 public override void WriteBinHex(byte[] buffer, int index, int count)
```

## 8 Summary

9 Encodes the specified binary bytes as BinHex and writes the resulting text.

## 10 Parameters

Parameter	Description
<i>buffer</i>	A System.Byte array containing the bytes to encode.
<i>index</i>	A System.Int32 specifying the position within the array of the first byte to encode.
<i>count</i>	A System.Int32 specifying the number of bytes to encode.

## 11 12 Description

13 [Note: For information on BinHex encoding, see RFC 1741  
14 (<http://www.ietf.org/rfc/rfc1741>).

15  
16 This method overrides System.Xml.XmlWriter.WriteBinHex.

17  
18 ]

## 19 Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<i>buffer</i> is null.
<b>System.ArgumentException</b>	The buffer length minus <i>index</i> is less than <i>count</i> .
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> or <i>count</i> is less than zero.

**System.InvalidOperationException**

The System.Xml.XmlTextWriter.WriteState is System.Xml.WriteState.Closed.

1

2

# 1 XmlTextWriter.WriteCData(System.String)

## 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteCData(string text)  
5 [C#]  
6 public override void WriteCData(string text)
```

### 7 Summary

8 Writes out a CDATA block containing the specified text.

### 9 Parameters

Parameter	Description
<i>text</i>	A <code>System.String</code> specifying the text to place inside the CDATA block.

### 10 11 Description

12 This method writes `<![CDATA[ text ]>`.

13  
14 If *text* is null or `System.String.Empty`, this method writes an empty CDATA block,  
15 `<![CDATA[ ]>`.

16  
17 [*Note:* This method overrides `System.Xml.XmlWriter.WriteCData`.

18  
19 ]

### 20 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	The text would result in a non-well formed XML document.
<b>System.InvalidOperationException</b>	The <code>System.Xml.XmlTextWriter.WriteState</code> is <code>System.Xml.WriteState.Closed</code> .

21  
22

# 1 XmlTextWriter.WriteCharEntity(System.Char) 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteCharEntity(valuetype  
5 System.Char ch)  
  
6 [C#]  
7 public override void WriteCharEntity(char ch)
```

## 8 Summary

9 Forces the generation of a character entity for the specified Unicode character value.

## 10 Parameters

Parameter	Description
<i>ch</i>	The <code>System.Char</code> for which to generate the entity.

## 11 12 Description

13 This method writes the Unicode character in hexadecimal character entity reference  
14 format.

15  
16 [*Note:* This method overrides `System.Xml.XmlWriter.WriteCharEntity`.

17  
18 ]

## 19 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	The character is in the surrogate pair character range, 0xd800 - 0xdfff, or the text would result in a non-well formed XML document.
<b>System.InvalidOperationException</b>	The <code>System.Xml.XmlTextWriter.WriteState</code> is <code>System.Xml.WriteState.Closed</code> .

20

21

# 1 XmlTextWriter.WriteChars(System.Char[], 2 System.Int32, System.Int32) Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteChars(char[] buffer, int32  
5 index, int32 count)  
  
6 [C#]  
7 public override void WriteChars(char[] buffer, int index, int count)
```

## 8 Summary

9 Writes text a buffer at a time.

## 10 Parameters

Parameter	Description
<i>buffer</i>	A System.Char array containing the text to write.
<i>index</i>	A System.Int32 specifying the position within the array of the start of the text to write.
<i>count</i>	A System.Int32 specifying the number of characters to write.

## 11 12 Description

13 [Note: This method can be used to write large amounts of text a buffer at a time.

14  
15 An exception is thrown if surrogate pair characters would be split across multiple buffer  
16 writes. This exception must be caught in order to continue writing the next surrogate  
17 pair characters. The XML specification defines the valid ranges for surrogate pairs.

18  
19 This method overrides System.Xml.XmlWriter.WriteChars.

20  
21 ]

## 22 Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<i>buffer</i> is null.
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> or <i>count</i> is less than zero. - or -

	The buffer length minus <i>index</i> is less than <i>count</i> .
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState is System.Xml.WriteState.Closed.

1

2

1  
2 **XmlTextWriter.WriteComment(System.String**  
3 **) Method**

```
4 [ILAsm]  
5 .method public hidebysig virtual void WriteComment(string text)  
6 [C#]  
7 public override void WriteComment(string text)
```

8 **Summary**

9 Writes out a comment containing the specified text.

10 **Parameters**

Parameter	Description
<i>text</i>	A System.String containing the text to place inside the comment.

11  
12 **Description**

13 This method writes `<!--text-->`.

14  
15 If *text* is null or System.String.Empty, this method writes a comment with no content,  
16 `<!-->`.

17  
18 [*Note:* This method overrides System.Xml.XmlWriter.WriteComment.

19  
20 ]

21 **Exceptions**

Exception	Condition
<b>System.ArgumentException</b>	The text would result in a non-well formed XML document
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState is System.Xml.WriteState.Closed.

22

23

# 1 XmlTextWriter.WriteDocType(System.String, 2 System.String, System.String, System.String) 3 Method

```
4 [ILAsm]  
5 .method public hidebysig virtual void WriteDocType(string name, string  
6 pubid, string sysid, string subset)  
  
7 [C#]  
8 public override void WriteDocType(string name, string pubid, string sysid,  
9 string subset)
```

## 10 Summary

11 Writes the document type declaration with the specified name and optional attributes.

## 12 Parameters

Parameter	Description
<i>name</i>	A System.String specifying the name of the document type.
<i>pubid</i>	A System.String specifying the public identifier, which is an alternative to the system identifier.
<i>sysid</i>	A System.String specifying the system identifier, which is the URI of the DTD (document type definition) for the document.
<i>subset</i>	A System.String specifying a URI that contains markup declarations.

13

## 14 Description

15 The optional attributes, *pubid*, *sysid*, and *subset*, are not checked for invalid characters.

16

17 [Note: A document type declaration is of the following form:

18

19 <!DOCTYPE *name* PUBLIC "*pubid*" "*sysid*" [*subset*]>

20

21 This method overrides System.Xml.XmlWriter.WriteDocType.

22

23 ]

## 24 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<p><i>name</i> is null or <code>System.String.Empty</code>.</p> <p>-or-</p> <p>The value for <i>name</i> would result in invalid XML.</p>
<b>System.InvalidOperationException</b>	<p>This method was called outside the prolog (after the root element).</p>

1

2

# 1 XmlTextWriter.WriteEndElement() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual void WriteEndElement()  
4 [C#]  
5 public override void WriteEndElement()
```

## 6 Summary

7 Closes the attribute started with the  
8 System.Xml.XmlTextWriter.WriteStartElement method.

## 9 Description

10 *[Note: The System.Xml.XmlTextWriter.WriteStartElement and*  
11 *System.Xml.XmlTextWriter.WriteEndElement methods also will close an open*  
12 *attribute if one exists when they are called.*

13 This method overrides System.Xml.XmlWriter.WriteEndElement.  
14  
15  
16 ]

## 17 Exceptions

Exception	Condition
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState is not System.Xml.WriteState.Attribute.

18

19

# 1 XmlTextWriter.WriteEndElement() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual void WriteEndElement()  
4 [C#]  
5 public override void WriteEndElement()
```

## 6 Summary

7 Closes open elements and attributes and sets the  
8 System.Xml.XmlTextWriter.WriteState back to the System.Xml.WriteState.Start  
9 state.

## 10 Description

11 This method closes all elements and attributes created by the  
12 System.Xml.XmlTextWriter.WriteStartElement and  
13 System.Xml.XmlTextWriter.WriteStartAttribute methods, respectively, that are  
14 open when the System.Xml.XmlTextWriter.WriteEndElement method is called.

15  
16 [Note: After calling this method, the current instance can be used to write a new XML  
17 document.

18  
19 This method overrides System.Xml.XmlWriter.WriteEndElement.  
20

21 ]

## 22 Exceptions

Exception	Condition
<b>System.InvalidOperationException</b>	The current instance is in the wrong System.Xml.WriteState, or the document does not have a root element.

23

24

# 1 XmlTextWriter.WriteEndElement() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual void WriteEndElement()  
4 [C#]  
5 public override void WriteEndElement()
```

## 6 Summary

7 Closes an open element and pops the corresponding namespace scope.

## 8 Description

9 If the open element does not contain content, it is closed as an empty element using "  
10 />"; otherwise an end element is written.

11  
12 [*Note:* This method overrides `System.Xml.XmlWriter.WriteEndElement`.  
13

14 ]

## 15 Exceptions

Exception	Condition
<b>System.InvalidOperationException</b>	No element was open, or the <code>System.Xml.XmlTextWriter.WriteState</code> is <code>System.Xml.WriteState.Closed</code> .

16

17

# 1 XmlTextWriter.WriteEntityRef(System.String)

## 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteEntityRef(string name)  
5 [C#]  
6 public override void WriteEntityRef(string name)
```

### 7 Summary

8 Writes an entity reference with the specified name.

### 9 Parameters

Parameter	Description
<i>name</i>	A <code>System.String</code> specifying the name of the entity reference.

### 10 Description

12 This method writes *%name*;

13  
14 [Note: This method overrides `System.Xml.XmlWriter.WriteEntityRef`.

15  
16 ]

### 17 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	A <code>System.String</code> containing the text would result in a non-well formed XML document, or <i>name</i> is either null or <code>System.String.Empty</code> .
<b>System.InvalidOperationException</b>	The <code>System.Xml.XmlTextWriter.WriteState</code> is <code>System.Xml.WriteState.Closed</code> .

18

19

# 1 XmlTextWriter.WriteEndElement() 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteEndElement()  
5 [C#]  
6 public override void WriteEndElement()
```

## 7 Summary

8 Closes an open element and pops the corresponding namespace scope.

## 9 Description

10 This method writes an end element regardless of whether there is any content in the  
11 element.

12  
13 [*Note:* This method overrides `System.Xml.XmlWriter.WriteEndElement`.  
14  
15 ]

## 16 Exceptions

Exception	Condition
<b>System.InvalidOperationException</b>	No start tag was open, or the <code>System.Xml.XmlTextWriter.WriteState</code> is <code>System.Xml.WriteState.Closed</code> .

17

18

# 1 XmlTextWriter.WriteName(System.String)

## 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteName(string name)  
5 [C#]  
6 public override void WriteName(string name)
```

### 7 Summary

8 Writes out the specified name, ensuring it is a valid name according to the W3C XML 1.0  
9 recommendation (<http://www.w3.org/TR/1998/REC-xml-19980210#NT-Name>).

### 10 Parameters

Parameter	Description
<i>name</i>	A System.String specifying the name to write.

### 11 Description

13 If System.Xml.XmlTextWriter.Namespaces is set to true, this method checks that  
14 *name* is also valid according to the W3C Namespaces in XML recommendation  
15 (<http://www.w3.org/TR/REC-xml-names>).

16 [Note: This method overrides System.Xml.XmlWriter.WriteName.

17 ]  
18  
19 ]

### 20 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>name</i> is null or System.String.Empty; or <i>name</i> is not a valid XML Name.
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState is System.Xml.WriteState.Closed.

21  
22

1  
2 **XmlTextWriter.WriteNmToken(System.String)**  
3 **Method**

```
4 [ILAsm]  
5 .method public hidebysig virtual void WriteNmToken(string name)  
6 [C#]  
7 public override void WriteNmToken(string name)
```

8 **Summary**

9 Writes out the specified name, ensuring it is a valid name token (Nmtoken) according to  
10 the W3C XML 1.0 recommendation (<http://www.w3.org/TR/1998/REC-xml-19980210#NT-Name>).  
11

12 **Parameters**

Parameter	Description
<i>name</i>	A System.String specifying the name to write.

13  
14 **Description**

15 [Note: This method overrides System.Xml.XmlWriter.WriteNmToken.  
16 ]  
17

18 **Exceptions**

Exception	Condition
<b>System.ArgumentException</b>	<i>name</i> is null or System.String.Empty; or <i>name</i> is not a valid XML Nmtoken.
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState is System.Xml.WriteState.Closed.

19  
20

# 1 2 XmlTextWriter.WriteProcessingInstruction(System.String, System.String) Method 3

```
4 [ILAsm]  
5 .method public hidebysig virtual void WriteProcessingInstruction(string  
6 name, string text)  
  
7 [C#]  
8 public override void WriteProcessingInstruction(string name, string text)
```

## 9 Summary

10 Writes out a processing instruction with the specified name and text.

## 11 Parameters

Parameter	Description
<i>name</i>	A System.String specifying the name of the processing instruction.
<i>text</i>	A System.String specifying the text to include in the processing instruction.

## 12 13 Description

14 This method writes `<?name?text?>`.

15  
16 If *text* is null or System.String.Empty, this method writes a processing instruction  
17 with no text content, `<?name?>`.

18  
19 [Note: This method overrides System.Xml.XmlWriter.WriteProcessingInstruction.  
20

21 ]

## 22 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	The text would result in a non-well formed XML document.  - or -  <i>name</i> is null or System.String.Empty.

	<p>- or -</p> <p>This method is being used to create an XML declaration after <code>System.Xml.XmlTextWriter.WriteStartDocument</code> has already been called.</p>
<b>System.InvalidOperationException</b>	<p>The <code>System.Xml.XmlTextWriter.WriteState</code> is <code>System.Xml.WriteState.Closed</code>.</p>

1

2

# XmlTextWriter.WriteQualifiedName(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig virtual void WriteQualifiedName(string localName,
string ns)

[C#]
public override void WriteQualifiedName(string localName, string ns)
```

## Summary

Writes out the qualified name.

## Parameters

Parameter	Description
<i>localName</i>	A <code>System.String</code> specifying the local name to write.
<i>ns</i>	A <code>System.String</code> specifying the namespace URI to associate with <i>localname</i> .

## Description

If *ns* maps to the current default namespace, no prefix is generated.

When writing attribute values, this method generates a prefix if *ns* is not found. When writing element content, this method throws an exception if *ns* is not found.

If the current instance supports namespaces (`System.Xml.XmlTextWriter.Namespaces` is set to `true`), this method looks up the prefix that is in scope for the given namespace and checks that the name is valid according to the W3C Namespaces in XML recommendation (<http://www.w3.org/TR/REC-xml-names>).

[*Note:* This method overrides `System.Xml.XmlWriter.WriteQualifiedName`.

]

## Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>localName</i> is null or <code>System.String.Empty</code> . -or-

	<p><code>System.Xml.XmlTextWriter.Namespaces</code> is false, and <code>ns</code> is neither null nor <code>System.String.Empty</code>.</p> <p>-or-</p> <p><code>localName</code> is not a valid XML name.</p>
<b>System.InvalidOperationException</b>	The <code>System.Xml.XmlTextWriter.WriteState</code> is <code>System.Xml.WriteState.Closed</code> .

1

2

# 1 XmlTextWriter.WriteRaw(System.Char[], 2 System.Int32, System.Int32) Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteRaw(char[] buffer, int32 index,  
5 int32 count)  
  
6 [C#]  
7 public override void WriteRaw(char[] buffer, int index, int count)
```

## 8 Summary

9 Writes raw text from a character array.

## 10 Parameters

Parameter	Description
<i>buffer</i>	A <code>System.Char</code> array containing the text to write.
<i>index</i>	A <code>System.Int32</code> specifying the position within the array of the start of the text to write.
<i>count</i>	A <code>System.Int32</code> specifying the number of characters to write.

## 11 12 Description

13 This method does not encode any characters.

14  
15 [Note: This method overrides `System.Xml.XmlWriter.WriteRaw`.

16  
17 ]

## 18 Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<i>buffer</i> is null.
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> or <i>count</i> is less than zero.
	- or - The buffer length minus <i>index</i> is less than

	<i>count.</i>
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState is System.Xml.WriteState.Closed.

1

2

# 1 XmlTextWriter.WriteRaw(System.String)

## 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteRaw(string data)  
5 [C#]  
6 public override void WriteRaw(string data)
```

### 7 Summary

8 Writes raw text from a string.

### 9 Parameters

Parameter	Description
<i>data</i>	A <code>System.String</code> specifying the text to write.

### 10 11 Description

12 If *data* is null, `System.String.Empty` is written.

13  
14 This method does not encode any characters.

15  
16 [Note: This method overrides `System.Xml.XmlWriter.WriteRaw`.

17  
18 ]

### 19 Exceptions

Exception	Condition
<b>System.InvalidOperationException</b>	The <code>System.Xml.XmlTextWriter.WriteState</code> is <code>System.Xml.WriteState.Closed</code> .

20  
21

1  
2 **XmlTextWriter.WriteStartAttribute(System.String, System.String, System.String) Method**  
3

```
4 [ILAsm]  
5 .method public hidebysig virtual void WriteStartAttribute(string prefix,  
6 string localName, string ns)  
  
7 [C#]  
8 public override void WriteStartAttribute(string prefix, string localName,  
9 string ns)
```

10 **Summary**

11 Writes the start of an attribute with the specified prefix and name, and associates the  
12 prefix with the specified namespace URI.

13 **Parameters**

Parameter	Description
<i>prefix</i>	A System.String specifying the namespace prefix of the attribute.
<i>localName</i>	A System.String specifying the local name of the attribute.
<i>ns</i>	A System.String specifying the namespace URI associated with the attribute.

14  
15 **Description**

16 If any of the input parameters are null or System.String.Empty, the start attribute is  
17 written with that parameter missing.

18  
19 [Note: This method overrides System.Xml.XmlWriter.WriteStartAttribute.  
20  
21 ]

22 **Exceptions**

Exception	Condition
<b>System.ArgumentException</b>	System.Xml.XmlTextWriter.Namespaces is false for the writer, and <i>prefix</i> and <i>ns</i> are not both null or System.String.Empty.
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState is not

one of the following:  
System.Xml.WriteState.Attribute or  
System.Xml.WriteState.Element.

1

2

# 1 XmlTextWriter.WriteStartDocument() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual void WriteStartDocument()  
4 [C#]  
5 public override void WriteStartDocument()
```

## 6 Summary

7 Writes the XML declaration with the version "1.0" and no standalone attribute.

## 8 Description

9 *[Note:* When `System.Xml.XmlTextWriter` is instantiated, the  
10 `System.Xml.XmlTextWriter.WriteState` is set to `System.Xml.WriteState.Start`. All  
11 the "write" methods change the `System.Xml.XmlTextWriter.WriteState` to a value  
12 other than `Start`. Thus, if this method is not the first "write" method called, a  
13 `System.InvalidOperationException` is thrown.

14  
15 If `System.Xml.XmlTextWriter.WriteStartDocument` has been called and then the  
16 `System.Xml.XmlTextWriter.WriteProcessingInstruction` method is used to create  
17 another XML declaration, a `System.ArgumentException` will be thrown.

18  
19 The output of this method using an encoding equal to `System.Text.Encoding.Unicode`  
20 and the default `System.Xml.XmlTextWriter.QuoteChar` is

```
21 <?xml version="1.0" encoding="utf-16"?>
```

22  
23 Character encoding is set when the writer is instantiated.

24  
25 This method overrides `System.Xml.XmlWriter.WriteStartDocument`.

26  
27 ]

## 28 Exceptions

Exception	Condition
<code>System.InvalidOperationException</code>	The <code>System.Xml.XmlTextWriter.WriteState</code> is not <code>System.Xml.WriteState.Start</code> .

29

30  
31

# 1 2 XmlTextWriter.WriteStartDocument(System.Boolean) Method 3

```
4 [ILAsm]  
5 .method public hidebysig virtual void WriteStartDocument(bool standalone)  
6 [C#]  
7 public override void WriteStartDocument(bool standalone)
```

## 8 Summary

9 Writes the XML declaration with the version "1.0" and the standalone attribute.

## 10 Parameters

Parameter	Description
<i>standalone</i>	A System.Boolean where true indicates to write "yes" as the value for the standalone attribute, and false indicates to write "no".

## 11 12 Description

13 [Note: When System.Xml.XmlTextWriter is instantiated, the  
14 System.Xml.XmlTextWriter.WriteState is set to System.Xml.WriteState.Start. All  
15 the "write" methods change the System.Xml.XmlTextWriter.WriteState to a value  
16 other than Start. Thus, if this method is not the first "write" method called, a  
17 System.InvalidOperationException is thrown.

18  
19 If System.Xml.XmlTextWriter.WriteStartDocument has been called and then the  
20 System.Xml.XmlTextWriter.WriteProcessingInstruction method is used to create  
21 another XML declaration, a System.ArgumentException will be thrown.

22  
23 The output of this method with *standalone* equal to true, an encoding equal to  
24 System.Text.Encoding.Unicode, and using the default  
25 System.Xml.XmlTextWriter.QuoteChar is:

```
26  
27 <?xml version="1.0" encoding="utf-16" standalone="yes"?>
```

28  
29 Character encoding is set when the writer is instantiated.

30  
31 This method overrides System.Xml.XmlWriter.WriteStartDocument.

32 ]  
33

## 34 Exceptions

Exception	Condition
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState is not System.Xml.WriteState.Start.

1

2

# 1 2 XmlTextWriter.WriteStartElement(System.String, System.String) Method 3

```
4 [ILAsm]  
5 .method public hidebysig virtual void WriteStartElement(string prefix,  
6 string localName, string ns)  
  
7 [C#]  
8 public override void WriteStartElement(string prefix, string localName,  
9 string ns)
```

## 10 Summary

11 Writes a start element with the specified name, and associates it with the given  
12 namespace and prefix.

## 13 Parameters

Parameter	Description
<i>prefix</i>	A System.String specifying the namespace prefix of the element.
<i>localName</i>	A System.String specifying the local name of the element.
<i>ns</i>	A System.String specifying the namespace URI to associate with the element.

## 14 15 Description

16 If *ns* is already in scope and has an associated prefix, the current instance will  
17 automatically write that prefix also.

18  
19 If any of the input parameters are null or System.String.Empty, the start element is  
20 written with that parameter missing.

21  
22 [Note: Write any attributes using the  
23 System.Xml.XmlTextWriter.WriteStartAttribute,  
24 System.Xml.XmlTextWriter.WriteString, and  
25 System.Xml.XmlTextWriter.WriteEndAttribute methods, then close the element  
26 using the System.Xml.XmlTextWriter.WriteEndElement method.

27  
28 This method overrides System.Xml.XmlWriter.WriteStartElement.  
29

30 ]

## 31 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	System.Xml.XmlTextWriter.Namespaces is false for the writer, and <i>prefix</i> and <i>ns</i> are not both null.
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState is System.Xml.WriteState.Closed.

1

## 2 Example

3 This example demonstrates the System.Xml.XmlTextWriter.WriteStartElement  
4 method, writing the XML to the console.

5

6 [C#]

7 using System;

8 using System.Xml;

9

10 public class WriteXml

11 {

12 public static void Main()

13 {

14 XmlTextWriter xWriter =

15 new XmlTextWriter(Console.Out);

16 xWriter.WriteStartDocument();

17 xWriter.WriteStartElement("prefix", "element", "namespace");

18 xWriter.WriteEndElement();

19 }

20 }

21 The output is

22

23 <?xml version="1.0" encoding="someencoding"?>

24

25 <prefix:element xmlns:prefix="namespace" />

26

27 The value of the encoding attribute is the encoding of the output stream of the console.

28

# 1 XmlTextWriter.WriteString(System.String) 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void WriteString(string text)  
5 [C#]  
6 public override void WriteString(string text)
```

## 7 Summary

8 Writes the specified text.

## 9 Parameters

Parameter	Description
<i>text</i>	A System.String specifying the text to write.

## 10 11 Description

12 This method performs the following conversions before writing the text:

- 13 • The characters '&', '<', and '>' are replaced with "&amp;", "&lt;", and "&gt;",  
14 respectively.
- 15 • Character values in the range 0x-0x1F (excluding the white space characters 0x9,  
16 0x10, and 0x13) are replaced with numeric character entities ("&#0;" through  
17 "&#0x1F").
- 18 • If called in the context of an attribute value, double and single quotes are replaced  
19 with "&quot;" and "&apos;" respectively.

20 If *text* is null or System.String.Empty, this method writes a text node with no data  
21 content.

22  
23 [Note: This method overrides System.Xml.XmlWriter.WriteString.

24  
25 ]

## 26 Exceptions

Exception	Condition
System.InvalidOperationException	The System.Xml.XmlTextWriter.WriteState is

	System.Xml.XmlWriterState.Closed and <i>text</i> is neither null nor System.String.Empty.
--	---

1

## 2 **Example**

3 The following example demonstrates the conversions performed by this method.

4

5 [C#]

6 using System;

7 using System.Xml;

8

9 public class WriteFrag {

10

11 public static void Main() {

12

13 XmlTextWriter xtWriter =

14 new XmlTextWriter(Console.Out);

15 xtWriter.WriteString("<1 & 2 = 3>");

16 }  
17 }

18

19

20 The output is

21

22 &lt;1 & 2 = 3&gt;

23

# XmlTextWriter.WriteSurrogateCharEntity(System.Char, System.Char) Method

```
[ILAsm]
.method public hidebysig virtual void WriteSurrogateCharEntity(valuetype
System.Char lowChar, valuetype System.Char highChar)

[C#]
public override void WriteSurrogateCharEntity(char lowChar, char highChar)
```

## Summary

Generates and writes the surrogate character entity for the surrogate character pair.

## Parameters

Parameter	Description
<i>lowChar</i>	A System.Char containing the low surrogate. This must be a value between 0xDC00 and 0xDFFF.
<i>highChar</i>	A System.Char containing the high surrogate. This must be a value between 0xD800 and 0xDBFF.

## Description

This method only applies to a writer that uses the UTF-16 encoding type.

The surrogate character entity is written in hexadecimal format. The range for surrogate characters is #x10000 to #x10FFFF. The following formula is used to generate the surrogate character entity:  $(highChar - 0xD800) * 0x400 + (lowChar - 0xDC00) + 0x10000$ .

[Note: For both HTML and XML, the document character set (and therefore the notation of numeric character references) is based on UCS [ISO-10646]. A single numeric character reference in a source document might therefore in some cases correspond to two 16-bit units in a string (a high surrogate and a low surrogate). These 16-bit units are referred to as a surrogate pair.

For more information regarding surrogates or characters, refer to section 3.7 of the Unicode 3.0/Unicode 2.0 standard located at <http://www.unicode.org>, or section 2.2 of the W3C XML 1.0 Recommendation located at <http://www.w3.org/TR/REC-xml#charsets>.

This method overrides System.Xml.XmlWriter.WriteSurrogateCharEntity.

1

2 ]

### 3 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	An invalid surrogate character pair was passed.
<b>System.InvalidOperationException</b>	The <code>System.Xml.XmlTextWriter.WriteState</code> is <code>System.Xml.WriteState.Closed</code> .

4

5

1  
2 **XmlTextWriter.WriteWhitespace(System.String)**  
3 **Method**

```
4 [ILAsm]  
5 .method public hidebysig virtual void WriteWhitespace(string ws)  
6 [C#]  
7 public override void WriteWhitespace(string ws)
```

8 **Summary**

9 Writes the given white space.

10 **Parameters**

Parameter	Description
<i>ws</i>	A System.String containing the white space characters.

11  
12 **Description**

13 [Note: This method is used to manually format a document. Use the  
14 System.Xml.XmlTextWriter.Formatting property to have the current instance format  
15 the output automatically.

16 This method overrides System.Xml.XmlWriter.WriteWhitespace.

17 ]  
18  
19

20 **Exceptions**

Exception	Condition
<b>System.ArgumentException</b>	<i>ws</i> is null or System.String.Empty or contains non-white space characters.
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState is System.Xml.WriteState.Closed.

21  
22

# 1 XmlTextWriter.BaseStream Property

```
2 [ILAsm]  
3 .property class System.IO.Stream BaseStream { public hidebysig specialname  
4 instance class System.IO.Stream get_BaseStream() }  
5 [C#]  
6 public Stream BaseStream { get; }
```

## 7 Summary

8 Gets the underlying stream used by the writer.

## 9 Property Value

10 A `System.IO.Stream`, or `null` if the current instance does not use an underlying stream.

## 11 Description

12 This property is read-only.

13

14 If the current instance was constructed using a `System.IO.TextWriter` that is a  
15 subclass of the `System.IO.StreamWriter` class, this property is equivalent to the  
16 `System.IO.StreamWriter.BaseStream` property.

17

18 If the writer was constructed using a `System.IO.Stream`, this property returns the  
19 `Stream` passed to the constructor.

20

21 If the writer was constructed using a file name, this property returns the `Stream`  
22 representing the file.

23

# 1 XmlTextWriter.Formatting Property

```
2 [ILAsm]  
3 .property valuetype System.Xml.Formatting Formatting { public hidebysig  
4 specialname instance valuetype System.Xml.Formatting get_Formatting()  
5 public hidebysig specialname instance void set_Formatting(valuetype  
6 System.Xml.Formatting value) }  
7 [C#]  
8 public Formatting Formatting { get; set; }
```

## 9 Summary

10 Indicates how the output is formatted.

## 11 Property Value

12 One of the members of the `System.Xml.Formatting` enumeration. The default is  
13 `System.Xml.Formatting.None` (no special formatting).

## 14 Description

15 If this property is set to `System.Xml.Formatting.Indented`, child elements are  
16 indented using the `System.Xml.XmlTextWriter.Indentation` and  
17 `System.Xml.XmlTextWriter.IndentChar` properties. Only element content will be  
18 indented.

19  
20 [*Note:* Writing any text content, including `System.String.Empty`, puts that element into  
21 mixed content mode. Child elements do not inherit this "mixed" mode status. A child  
22 element of a "mixed" element will do indenting, unless it is also contains "mixed"  
23 content. Element content ([http://www.w3.org/TR/1998/REC-xml-19980210#sec-](http://www.w3.org/TR/1998/REC-xml-19980210#sec-element-content)  
24 [element-content](http://www.w3.org/TR/1998/REC-xml-19980210#sec-element-content)) and mixed content ([http://www.w3.org/TR/1998/REC-xml-](http://www.w3.org/TR/1998/REC-xml-19980210#sec-mixed-content)  
25 [19980210#sec-mixed-content](http://www.w3.org/TR/1998/REC-xml-19980210#sec-mixed-content)) are defined according to the XML 1.0 definitions of these  
26 terms.

27  
28 ]

29

# 1 XmlTextWriter.Indentation Property

```
2 [ILAsm]  
3 .property int32 Indentation { public hidebysig specialname instance int32  
4 get_Indentation() public hidebysig specialname instance void  
5 set_Indentation(int32 value) }  
  
6 [C#]  
7 public int Indentation { get; set; }
```

## 8 Summary

9 Gets or sets how many indentation characters to write for each level in the hierarchy  
10 when `System.Xml.XmlTextWriter.Formatting` is set to  
11 `System.Xml.Formatting.Indented`.

## 12 Property Value

13 A `System.Int32` specifying the number of `System.Xml.XmlTextWriter.IndentChar`  
14 characters to use for each level. The default is 2.

## 15 Description

16 Indentation is performed on the following members of `System.Xml.XmlNodeType`:  
17 `DocumentType`, `Element`, `Comment`, `ProcessingInstruction`, and `CDATA`. All other node  
18 types are not affected. The `System.Xml.XmlTextWriter` class does not indent the  
19 internal DTD subset.

## 20 Exceptions

Exception	Condition
<code>System.ArgumentException</code>	The value to be set is less than zero.

21

22

# 1 XmlTextWriter.IndentChar Property

```
2 [ILAsm]  
3 .property valuetype System.Char IndentChar { public hidebysig specialname  
4 instance valuetype System.Char get_IndentChar() public hidebysig  
5 specialname instance void set_IndentChar(valuetype System.Char value) }  
6 [C#]  
7 public char IndentChar { get; set; }
```

## 8 Summary

9 Gets or sets the character to use for indenting when  
10 System.Xml.XmlTextWriter.Formatting is set to System.Xml.Formatting.Indented.

## 11 Property Value

12 A System.Char specifying the character to use for indenting. The default is space  
13 (character code 0x20).

## 14 Description

15 [*Note:* This property can be set to any character. To ensure valid XML, set this property  
16 to a valid white space character: 0x9, 0x10, 0x13, or 0x20.

17 ]  
18 ]

19

# 1 XmlTextWriter.Namespaces Property

```
2 [ILAsm]  
3 .property bool Namespaces { public hidebysig specialname instance bool  
4 get_Namespaces() public hidebysig specialname instance void  
5 set_Namespaces(bool value) }  
  
6 [C#]  
7 public bool Namespaces { get; set; }
```

## 8 Summary

9 Gets or sets a value indicating whether the writer supports namespaces.

## 10 Property Value

11 A System.Boolean where true indicates the writer supports namespaces; otherwise,  
12 false. The default is true.

## 13 Description

14 This property determines whether the writer supports the XML Namespaces specification  
15 (<http://www.w3.org/TR/REC-xml-names>).

16  
17 If an attempt is made to set this property after a write operation has occurred, a  
18 System.InvalidOperationException is thrown.

## 19 Exceptions

Exception	Condition
<b>System.InvalidOperationException</b>	The System.Xml.XmlTextWriter.WriteState of the current instance is not System.Xml.WriteState.Start.

20

21

# 1 XmlTextWriter.QuoteChar Property

```
2 [ILAsm]  
3 .property valuetype System.Char QuoteChar { public hidebysig specialname  
4 instance valuetype System.Char get_QuoteChar() public hidebysig  
5 specialname instance void set_QuoteChar(valuetype System.Char value) }  
6 [C#]  
7 public char QuoteChar { get; set; }
```

## 8 Summary

9 Gets or sets the character used to quote the value of an attribute.

## 10 Property Value

11 A System.Char specifying the quotation mark character (" or ') used to enclose the  
12 value of an attribute. The default is the double quote.

## 13 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	The value to be set is not the single quote or double quote character.

14

15

# 1 XmlTextWriter.WriteState Property

```
2 [ILAsm]  
3 .property valuetype System.Xml.WriteState WriteState { public hidebydef  
4 virtual specialname valuetype System.Xml.WriteState get_WriteState() }  
5 [C#]  
6 public override WriteState WriteState { get; }
```

## 7 Summary

8 Gets the write state of the writer.

## 9 Property Value

10 One of the members of the `System.Xml.WriteState` enumeration.

## 11 Description

12 This property is read-only.

13

14 [*Note:* This property overrides `System.Xml.XmlWriter.WriteState`.

15

16 ]

17

# 1 XmlTextWriter.XmlLang Property

```
2 [ILAsm]  
3 .property string XmlLang { public hidebysig virtual specialname string  
4 get_XmlLang() }  
5 [C#]  
6 public override string XmlLang { get; }
```

## 7 Summary

8 Gets the language attribute, `xml:lang`, specifying the language in which the content and  
9 attribute values of the current element are written.

## 10 Property Value

11 A `System.String` containing the language attribute, or `null` if the language attribute is  
12 not specified for the element.

## 13 Description

14 This property is read-only.

15  
16 [*Note:* This property overrides `System.Xml.XmlWriter.XmlLang`.  
17

18 ]  
19

# 1 XmlTextWriter.XmlSpace Property

```
2 [ILAsm]  
3 .property valuetype System.Xml.XmlSpace XmlSpace { public hideby sig  
4 virtual specialname valuetype System.Xml.XmlSpace get_XmlSpace() }  
5 [C#]  
6 public override XmlSpace XmlSpace { get; }
```

## 7 Summary

8 Gets the white space attribute, `xml:space`, specifying how white space is handled in the  
9 current element.

## 10 Property Value

11 One of the members of the `System.Xml.XmlSpace` enumeration, or  
12 `System.Xml.XmlSpace.None` if the white space attribute is not specified for the element.

## 13 Description

14 This property is read-only.

15  
16 [*Note:* This property overrides `System.Xml.XmlWriter.XmlSpace`.  
17

18 ]

19