

System.Xml.XmlReader Class

```
[ILASM]
.class public abstract XmlReader extends System.Object

[C#]
public abstract class XmlReader
```

Assembly Info:

- Name: System.Xml
- Public Key: [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- Version: 1.0.x.x
- Attributes:
 - CLSCompliantAttribute(true)

Type Attributes:

- DefaultMemberAttribute("Item") [Note: This attribute requires the RuntimeInfrastructure library.]

Summary

Represents a reader that provides non-cached, forward-only access to XML data.

Inherits From: System.Object

Library: XML

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

Description

This class provides forward-only, read-only access to a stream of XML data. This class enforces the rules of well-formed XML but does not perform data validation.

This class conforms to the W3C Extensible Markup Language (XML) 1.0 and the Namespaces in XML recommendations.

A given set of XML data is modeled as a tree of nodes. The different types of nodes are specified in the **System.Xml.XmlNodeType** enumeration. The reader is advanced to the next node using the **System.Xml.XmlReader.Read** method. The current node refers to the node on which the reader is positioned. The following table lists the node properties exposed for the current node.

Property	Description
AttributeCount	The number of attributes on the node.
BaseUri	The base URI of the node.
Depth	The depth of the node in the tree.
HasAttributes	Whether the node has attributes.
HasValue	Whether the node can have a text value.
IsDefault	Whether an Attribute node was generated from the default value defined in the DTD or schema.
IsEmptyElement	Whether an Element node is empty.
LocalName	The local name of the node.
Name	The qualified name of the node, equal to Prefix:LocalName .
NamespaceUri	The URI defining the namespace associated with the node.
NodeType	The System.Xml.XmlNodeType of the node.
Prefix	A shorthand reference to the namespace associated with the node.
QuoteChar	The quotation mark character used to enclose the value of an attribute.
Value	The text value of the node.
XmlLang	The <code>xml:lang</code> scope within which the node resides.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

This class does not expand default attributes or general entities. Any general entities encountered are returned as a single empty **EntityReference** node.

This class checks that a Document Type Definition (DTD) is well-formed, but does not validate using the DTD.

To read strongly typed data, use the **System.Xml.XmlConvert** class.

This class throws a **System.Xml.XmlException** on XML parse errors. After an exception is thrown, the state of the reader is not predictable. For example, the reported node type may be different than the actual node type of the current node.

[*Note:* This class is **abstract** and implemented in the **System.Xml.XmlTextReader** class.]

1 XmlReader() Constructor

```
2 [ILASM]  
3 family specialname instance void .ctor()  
4  
5 [C#]  
6 protected XmlReader()
```

6 Summary

7 Constructs a new instance of the **System.Xml.XmlReader** class.

8

1 XmlReader.Close() Method

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

6 Summary

7 Changes the **System.Xml.XmlReader.ReadState** to
8 **System.Xml.ReadState.Closed**.

9 Behaviors

10 This method releases any resources allocated by the current instance,
11 changes the **System.Xml.XmlReader.ReadState** to
12 **System.Xml.ReadState.Closed**, and calls the **Close** method of any
13 underlying **System.IO.Stream** or **System.IO.TextReader** instance.

14 How and When to Override

15 This method must be overridden in order to provide the functionality
16 described above, as there is no default implementation.

17

1 XmlReader.GetAttribute(System.Int32)

2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract string  
5 GetAttribute(int32 i)  
6  
7 [C#]  
8 public abstract string GetAttribute(int i)
```

8 Summary

9 Returns the value of the attribute with the specified index relative to
10 the containing element.

11 Parameters

12
13

Parameter	Description
<i>i</i>	A System.Int32 specifying the zero-based index of the attribute relative to the containing element.

14
15
16

15 Return Value

17 A **System.String** containing the value of the specified attribute.

18 Behaviors

19 This method does not move the reader.

20 How and When to Override

21 This method must be overridden in order to provide the functionality
22 described above, as there is no default implementation.

23 Exceptions

24
25

Exception	Condition
System.ArgumentOutOfRangeException	<i>i</i> is less than 0, or greater than or equal to the System.Xml.XmlReader.AttributeCount of the containing element.

26
27
28

27 Example

1 For an example demonstrating this method, see
2 **System.Xml.XmlTextReader.GetAttribute(System.String,**
3 **System.String).**

4

1 XmlReader.GetAttribute(System.String, 2 System.String) Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract string  
5 GetAttribute(string name, string namespaceURI)  
  
6 [C#]  
7 public abstract string GetAttribute(string name, string  
8 namespaceURI)
```

9 Summary

10 Returns the value of the attribute with the specified local name and
11 namespace URI.

12 Parameters

13
14

Parameter	Description
<i>name</i>	A System.String specifying the local name of the attribute.
<i>namespaceURI</i>	A System.String specifying the namespace URI of the attribute.

15
16
17

16 Return Value

18 A **System.String** containing the value of the specified attribute, or
19 **null** if the attribute is not found.

20 Behaviors

21 This method does not move the reader.

22 How and When to Override

23 This method must be overridden in order to provide the functionality
24 described above, as there is no default implementation.

25 Example

26

27 For an example demonstrating this method, see
28 **System.Xml.XmlTextReader.GetAttribute(System.String,
29 System.String)**.

30

1 XmlReader.GetAttribute(System.String)

2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract string  
5 GetAttribute(string name)  
  
6 [C#]  
7 public abstract string GetAttribute(string name)
```

8 Summary

9 Returns the value of the attribute with the specified qualified name.

10 Parameters

11
12

Parameter	Description
<i>name</i>	A System.String specifying the qualified name of the attribute.

13
14
15

Return Value

16 A **System.String** containing the value of the specified attribute, or
17 **null** if the attribute is not found.

18 Behaviors

19 This method does not move the reader.

20 How and When to Override

21 This method must be overridden in order to provide the functionality
22 described above, as there is no default implementation.

23 Example

24

25 For an example demonstrating this method, see
26 **System.Xml.XmlTextReader.GetAttribute(System.String,**
27 **System.String)**.

28

1 XmlReader.IsName(System.String)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static bool IsName(string str)  
5 [C#]  
6 public static bool IsName(string str)
```

7 Summary

8 Determines whether the specified string is a valid XML name.

9 Parameters

10
11

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

12
13
14

Return Value

15 A **System.Boolean** where **true** indicates the name is valid; otherwise,
16 **false**.

17 Description

18 [Note: This method uses the W3C XML 1.0 Recommendation
19 (<http://www.w3.org/TR/2000/REC-xml-20001006#NT-Name>) to
20 determine whether the name is valid.]

21

1 XmlReader.IsNameToken(System.String)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static bool IsNameToken(string  
5 str)  
  
6 [C#]  
7 public static bool IsNameToken(string str)
```

8 Summary

9 Determines whether the specified string is a valid XML name token
10 (Nmtoken).

11 Parameters

12
13

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

14
15
16

15 Return Value

17 A **System.Boolean** where **true** indicates the name is valid; otherwise
18 **false**.

19 Description

20 [Note: This method uses the W3C XML 1.0 Recommendation
21 (<http://www.w3.org/TR/2000/REC-xml-20001006#NT-Nmtoken>) to
22 determine whether the name token is valid.]

23

1 XmlReader.IsStartElement(System.String, 2 System.String) Method

```
3 [ILASM]  
4 .method public hidebysig virtual bool IsStartElement(string  
5 localname, string ns)  
  
6 [C#]  
7 public virtual bool IsStartElement(string localname, string  
8 ns)
```

9 Summary

10 Determines if a node containing content is an **Element** node with the
11 specified local name and namespace URI.

12 Parameters

13
14

Parameter	Description
<i>localname</i>	A System.String specifying the local name of an element.
<i>ns</i>	A System.String specifying the namespace URI associated with the element.

15
16
17

16 Return Value

18 A **System.Boolean** where **true** indicates the node is an **Element**
19 node with the specified local name and namespace URI; **false**
20 otherwise.

21 Behaviors

22 As described above.

23 Default

24 This method calls the **System.Xml.XmlReader.MoveToContent**
25 method, which determines whether the current node can contain
26 content and, if not, moves the reader to the next content node or the
27 end of the input stream. When the reader is positioned on a content
28 node, the node is checked to determine if it is an **Element** node with
29 **System.Xml.XmlReader.LocalName** and
30 **System.Xml.XmlReader.NamespaceURI** properties equal to
31 *localname* and *ns*, respectively.

32 How and When to Override

1 Override this method to customize the behavior of this method in
2 types derived from the **System.Xml.XmlReader** class.

3 **Usage**

4 Use this method to determine whether the node returned by the
5 **System.Xml.XmlReader.MoveToContent** method is an **Element**
6 node with the specified local name and namespace URI.

7 **Exceptions**

8
9

Exception	Condition
System.Xml.XmlException	An error occurred while parsing the XML.

10
11
12

1 XmlReader.IsStartElement(System.String 2) Method

```
3 [ILASM]  
4 .method public hidebysig virtual bool IsStartElement(string  
5 name)  
  
6 [C#]  
7 public virtual bool IsStartElement(string name)
```

8 Summary

9 Determines if a node containing content is an **Element** node with the
10 specified qualified name.

11 Parameters

12
13

Parameter	Description
<i>name</i>	A System.String specifying the qualified name of an element.

14
15
16

15 Return Value

17 A **System.Boolean** where **true** indicates the node is an **Element**
18 node with the specified name; **false** otherwise.

19 Behaviors

20 As described above.

21 Default

22 This method calls the **System.Xml.XmlReader.MoveToContent**
23 method, which determines whether the current node can contain
24 content and, if not, moves the reader to the next content node or the
25 end of the input stream. When the reader is positioned on a content
26 node, the node is checked to determine if it is an **Element** node with a
27 **System.Xml.XmlReader.Name** property equal to *name*.

28 How and When to Override

29 Override this method to customize the behavior of this method in
30 types derived from the **System.Xml.XmlReader** class.

31 Usage

1 Use this method to determine whether the node returned by the
2 **System.Xml.XmlReader.MoveToContent** method is an **Element**
3 node with the specified name.

4 **Exceptions**

5
6

Exception	Condition
System.Xml.XmlException	An error occurred while parsing the XML.

7
8
9

1 XmlReader.IsStartElement() Method

```
2 [ILASM]  
3 .method public hidebysig virtual bool IsStartElement()  
4 [C#]  
5 public virtual bool IsStartElement()
```

6 Summary

7 Determines if a node containing content is an **Element** node.

8 Return Value

9

10 A **System.Boolean** where **true** indicates the node is an **Element**
11 node; **false** otherwise.

12 Behaviors

13 As described above.

14 Default

15 This method calls the **System.Xml.XmlReader.MoveToContent**
16 method, which determines whether the current node can contain
17 content and, if not, moves the reader to the next content node or the
18 end of the input stream. When the reader is positioned on a content
19 node, the node is checked to determine if it is an **Element** node.

20 How and When to Override

21 Override this method to customize the behavior of this method in
22 types derived from the **System.Xml.XmlReader** class.

23 Usage

24 Use this method to determine whether the node returned by the
25 **System.Xml.XmlReader.MoveToContent** method is an **Element**
26 node.

27 Exceptions

28

29

Exception	Condition
System.Xml.XmlException	An error occurred while parsing the XML.

30

31

32

1 XmlReader.LookupNamespace(System.String) Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract string  
5 LookupNamespace(string prefix)  
  
6 [C#]  
7 public abstract string LookupNamespace(string prefix)
```

8 Summary

9 Resolves a namespace prefix in the scope of the current element.

10 Parameters

11
12

Parameter	Description
<i>prefix</i>	A System.String specifying the prefix whose namespace URI is to be resolved. To return the default namespace, specify System.String.Empty .

13
14
15

Return Value

16 A **System.String** containing the namespace URI to which the prefix
17 maps. If *prefix* is not in **System.Xml.XmlReader.NameTable** or no
18 matching namespace is found, **null** is returned.

19 Behaviors

20 As described above.

21 How and When to Override

22 This method must be overridden in order to provide the functionality
23 described above, as there is no default implementation.

24

1 XmlReader.MoveToAttribute(System.Int32) Method

```
3 [ILASM]
4 .method public hidebysig virtual abstract void
5 MoveToAttribute(int32 i)
6
7 [C#]
8 public abstract void MoveToAttribute(int i)
```

8 Summary

9 Moves the position of the current instance to the attribute with the
10 specified index relative to the containing element.

11 Parameters

12
13

Parameter	Description
<i>i</i>	A System.Int32 specifying the zero-based index of the attribute relative to the containing element.

14
15

16 Behaviors

17 After calling this method, the **System.Xml.XmlReader.Name**,
18 **System.Xml.XmlReader.NamespaceURI**, and
19 **System.Xml.XmlReader.Prefix** properties reflect the properties of
20 current attribute.

21 How and When to Override

22 This method must be overridden in order to provide the functionality
23 described above, as there is no default implementation.

24 Exceptions

25
26

Exception	Condition
System.ArgumentOutOfRangeException	<i>i</i> is less than 0 or greater than or equal to the System.Xml.XmlReader.AttributeCount of the containing element.

27
28
29

1 XmlReader.MoveToAttribute(System.String, System.String) Method

```
3 [ILASM]
4 .method public hidebysig virtual abstract bool
5 MoveToAttribute(string name, string ns)
6
7 [C#]
8 public abstract bool MoveToAttribute(string name, string ns)
```

9 Summary

10 Moves the position of the current instance to the attribute with the
11 specified local name and namespace URI.

12 Parameters

13
14

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

15
16
17

16 Return Value

18 A **System.Boolean** where **true** indicates the attribute was found;
19 otherwise, **false**. If **false**, the position of the current instance does not
20 change.

21 Behaviors

22 After calling this method, the **System.Xml.XmlReader.Name**,
23 **System.Xml.XmlReader.NamespaceURI**, and
24 **System.Xml.XmlReader.Prefix** properties reflect the properties of
25 current attribute.

26 How and When to Override

27 This method must be overridden in order to provide the functionality
28 described above, as there is no default implementation.

29

1 XmlReader.MoveToAttribute(System.String) Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract bool  
5 MoveToAttribute(string name)  
  
6 [C#]  
7 public abstract bool MoveToAttribute(string name)
```

8 Summary

9 Moves the position of the current instance to the attribute with the
10 specified qualified name.

11 Parameters

12
13

Parameter	Description
<i>name</i>	A System.String specifying the qualified name of the attribute.

14
15
16

15 Return Value

17 A **System.Boolean** where **true** indicates the attribute was found;
18 otherwise, **false**. If **false**, the reader's position does not change.

19 Behaviors

20 After calling this method, the **System.Xml.XmlReader.Name**,
21 **System.Xml.XmlReader.NamespaceURI**, and
22 **System.Xml.XmlReader.Prefix** properties reflect the properties of
23 current attribute.

24 How and When to Override

25 This method must be overridden in order to provide the functionality
26 described above, as there is no default implementation.

27

1 XmlReader.MoveToContent() Method

```
2 [ILASM]  
3 .method public hidebysig virtual valuetype  
4 System.Xml.XmlNodeType MoveToContent()  
  
5 [C#]  
6 public virtual XmlNodeType MoveToContent()
```

7 Summary

8 Determines whether the current node can contain content and, if not,
9 moves the position of the current instance to the next content node or
10 the end of the input stream.

11 Return Value

12

13 The **System.Xml.XmlNodeType** of the content node, or
14 **System.Xml.XmlNodeType.None** if the position of the reader has
15 reached the end of the input stream.

16 Description

17 [Note: The following members of **System.Xml.XmlNodeType** can
18 contain content: **Attribute**, **CDATA**, **Element**, **EndElement**,
19 **EntityReference**, **EndEntity**, and **Text**.]

20 Behaviors

21 As described above.

22 Default

23 If the current node is an **Attribute** node, this method moves the
24 position of the reader back to the **Element** node that owns the
25 attribute.

26 How and When to Override

27 Override this method to customize the behavior of this method in
28 types derived from the **System.Xml.XmlReader** class.

29 Usage

30 Use this method to determine whether the current node can contain
31 content and, if not, move the position of the reader to the next content
32 node.

1 **Exceptions**

2

3

Exception	Condition
System.Xml.XmlException	An error occurred while parsing the XML.

4

5

6

1 XmlReader.MoveToElement() Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract bool  
4 MoveToElement()  
5 [C#]  
6 public abstract bool MoveToElement()
```

7 Summary

8 Moves the position of the current instance to the node that contains
9 the current **Attribute** node.

10 Return Value

11

12 A **System.Boolean** where **true** indicates the position of the reader
13 was moved; **false** indicates the reader was not positioned on an
14 **Attribute** node and therefore the position of the reader was not
15 moved.

16 Description

17 [*Note:* The **DocumentType**, **Element**, and **XmlDeclaration**
18 members of **System.Xml.XmlNodeType** can contain attributes.]

19 Behaviors

20 As described above.

21 How and When to Override

22 This method must be overridden in order to provide the functionality
23 described above, as there is no default implementation.

24

1 XmlReader.MoveToFirstAttribute()

2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract bool  
5 MoveToFirstAttribute()  
6 [C#]  
7 public abstract bool MoveToFirstAttribute()
```

8 Summary

9 Moves the position of the current instance to the first attribute
10 associated with the current node.

11 Return Value

12

13 A **System.Boolean** where **true** indicates the current node contains at
14 least one attribute; otherwise, **false**.

15 Behaviors

16 If **System.Xml.XmlReader.AttributeCount** is non-zero, the position
17 of the reader moves to the first attribute; otherwise, the position of
18 the reader does not change.

19 How and When to Override

20 This method must be overridden in order to provide the functionality
21 described above, as there is no default implementation.

22

1 XmlReader.MoveNextAttribute() 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract bool  
5 MoveNextAttribute()  
6 [C#]  
7 public abstract bool MoveNextAttribute()
```

8 Summary

9 Moves the position of the current instance to the next attribute
10 associated with the current node.

11 Return Value

12

13 A **System.Boolean** where **true** indicates the position of the reader
14 moved to the next attribute; **false** if there were no more attributes.

15 Behaviors

16 As described above.

17 How and When to Override

18 This method must be overridden in order to provide the functionality
19 described above, as there is no default implementation.

20

1 XmlReader.Read() Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract bool Read()  
4 [C#]  
5 public abstract bool Read()
```

6 Summary

7 Moves the position of the current instance to the next node in the
8 stream, exposing its properties.

9 Return Value

10

11 A **System.Boolean** where **true** indicates the node was read
12 successfully, and **false** indicates there were no more nodes to read.

13 Behaviors

14 As described above.

15 How and When to Override

16 This method must be overridden in order to provide the functionality
17 as described herein, as there is no default implementation.

18 Usage

19 When a reader is first created and initialized, there is no information
20 available. Calling this method is required to read the first node.

21 Exceptions

22

23

Exception	Condition
System.Xml.XmlException	An error occurred while parsing the XML.

24

25

26

1 XmlReader.ReadAttributeValue() Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract bool  
4 ReadAttributeValue()  
5 [C#]  
6 public abstract bool ReadAttributeValue()
```

7 Summary

8 Parses an attribute value into one or more **Text**, **EntityReference**,
9 and **EndElement** nodes.

10 Return Value

11

12 A **System.Boolean** where **true** indicates the attribute value was
13 parsed, and **false** indicates the reader was not positioned on an
14 attribute node or all the attribute values have been read.

15 Description

16 [*Note:* To parse an **EntityReference** node, call the
17 **System.Xml.XmlReader.ResolveEntity** method. After the node is
18 parsed into child nodes, call the
19 **System.Xml.XmlReader.ReadAttributeValue** method again to read
20 the value of the entity.
21

22 The **System.Xml.XmlReader.Depth** of an attribute value node is one
23 plus the depth of the attribute node. When general entity references
24 are stepped into or out of, the **System.Xml.XmlReader.Depth**
25 increments or decrements by one, respectively.]

26 Behaviors

27 As described above.

28 How and When to Override

29 Implementations that cannot expand general entities should return
30 general entities as a single empty (**System.Xml.XmlReader.Value**
31 equals **System.String.Empty**) **EntityReference** node.

32 Usage

33 Use this method after calling
34 **System.Xml.XmlReader.MoveToAttribute** to read through the
35 **Text**, **EntityReference**, or **EndElement** nodes that make up the

1 attribute value. Call the **System.Xml.XmlReader.ResolveEntity**
2 method to resolve the **EntityReference** nodes.

3

1 XmlReader.ReadElementString(System.String, System.String) Method

```
3 [ILASM]  
4 .method public hidebysig virtual string  
5 ReadElementString(string localname, string ns)  
  
6 [C#]  
7 public virtual string ReadElementString(string localname,  
8 string ns)
```

9 Summary

10 Reads the contents of a text-only element with the specified local
11 name and namespace URI.

12 Parameters

13
14

Parameter	Description
<i>localname</i>	A System.String specifying the local name of an element.
<i>ns</i>	A System.String specifying the namespace URI associated with the element.

15
16
17

16 Return Value

18 A **System.String** containing the contents of the element.

19 Behaviors

20 As described above.

21 Default

22 This method calls the **System.Xml.XmlReader.MoveToContent**
23 method. If the returned node is an **Element** node, this method
24 compares the **System.Xml.XmlReader.LocalName** and
25 **System.Xml.XmlReader.NamespaceURI** properties of the node to
26 *localname* and *ns*, respectively. If they are equal, this method calls the
27 **System.Xml.XmlReader.ReadString** method to read the contents of
28 the element.

29 How and When to Override

30 Override this method to customize the behavior of this method in
31 types derived from the **System.Xml.XmlReader** class.

1 **Usage**

2 Use this method to read the contents of a text-only element with the
3 specified local name and namespace URI.

4 **Exceptions**

5
6

Exception	Condition
System.Xml.XmlException	The node is not an Element node, the System.Xml.XmlReader.LocalName property of the Element node does not equal <i>localname</i> , or the System.Xml.XmlReader.NamespaceURI property of the Element node does not equal <i>ns</i> , the element does not contain a simple text value, or an error occurred while parsing the XML.

7
8
9

1 XmlReader.ReadElementString(System.String) Method

```
3 [ILASM]
4 .method public hidebysig virtual string
5 ReadElementString(string name)
6
7 [C#]
8 public virtual string ReadElementString(string name)
```

8 Summary

9 Reads the contents of a text-only element with the specified qualified
10 name.

11 Parameters

12
13

Parameter	Description
<i>name</i>	A System.String specifying the qualified name of an element.

14
15
16

15 Return Value

17 A **System.String** containing the contents of the element.

18 Behaviors

19 As described above.

20 Default

21 This method calls the **System.Xml.XmlReader.MoveToContent**
22 method and, if the returned node is an **Element** node, compares the
23 **System.Xml.XmlReader.Name** property of the node to *name*. If
24 they are equal, this method calls the
25 **System.Xml.XmlReader.ReadString** method to read the contents of
26 the element.

27 How and When to Override

28 Override this method to customize the behavior of this method in
29 types derived from the **System.Xml.XmlReader** class.

30 Usage

1 Use this method to read the contents of a text-only element with the
2 specified qualified name.

3 **Exceptions**

4
5

Exception	Condition
System.Xml.XmlException	The node is not an Element node, the System.Xml.XmlReader.Name property of the Element node does not equal <i>name</i> , the element does not contain a simple text value, or an error occurred while parsing the XML.

6
7
8

1 XmlReader.ReadElementString() Method

```
2 [ILASM]  
3 .method public hidebysig virtual string ReadElementString()  
4 [C#]  
5 public virtual string ReadElementString()
```

6 Summary

7 Reads the contents of a text-only element.

8 Return Value

9

10 A **System.String** containing the contents of the element.

11 Behaviors

12 As described above.

13 Default

14 This method calls the **System.Xml.XmlReader.MoveToContent**
15 method and, if the returned node is an **Element** node, calls the
16 **System.Xml.XmlReader.ReadString** method to read the contents.

17 How and When to Override

18 Override this method to customize the behavior of this method in
19 types derived from the **System.Xml.XmlReader** class.

20 Usage

21 Use this method to read the contents of a text-only element.

22 Exceptions

23

24

Exception	Condition
System.Xml.XmlException	The node is not an Element node, the element does not contain a simple text value, or an error occurred while parsing the XML.

25

26

27

1 XmlReader.ReadEndElement() Method

```
2 [ILASM]  
3 .method public hidebysig virtual void ReadEndElement()  
4 [C#]  
5 public virtual void ReadEndElement()
```

6 Summary

7 Reads an **EndElement** node and advances the reader to the next
8 node.

9 Behaviors

10 As described above.

11 Default

12 This method calls the **System.Xml.XmlReader.MoveToContent**
13 method, which determines whether the current node can contain
14 content and, if not, moves the reader to the next content node or the
15 end of the input stream. The node the reader ends up positioned on is
16 checked to determine if it is an **EndElement** node. If so, the node is
17 read and the reader is moved to the next node.

18 How and When to Override

19 Override this method to customize the behavior of this method in
20 types derived from the **System.Xml.XmlReader** class.

21 Usage

22 Use this method to read an **EndElement** node and advance the reader
23 to the next node.

24 Exceptions

25
26

Exception	Condition
System.Xml.XmlException	The node is not an EndElement node or an error occurred while parsing the XML.

27
28
29

1 XmlReader.ReadInnerXml() Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract string  
4 ReadInnerXml()  
5 [C#]  
6 public abstract string ReadInnerXml()
```

7 Summary

8 Reads the contents of the current node, including child nodes and
9 markup.

10 Return Value

11

12 A **System.String** containing the XML content, or
13 **System.String.Empty** if the current node is neither an element nor
14 attribute, or has no child nodes.

15 Behaviors

16 The current node and corresponding end node are not returned.

17

18 If the current node is an element, after the call to this method, the
19 reader is positioned after the corresponding end element.

20

21 If the current node is an attribute, the position of the reader is not
22 changed.

23

24 [Note: For a comparison between this method and the
25 **System.Xml.XmlReader.ReadOuterXml** method, see
26 **System.Xml.XmlTextReader.ReadInnerXml**.]

27 How and When to Override

28 This method must be overridden in order to provide the functionality
29 described above, as there is no default implementation.

30 Exceptions

31

32

Exception	Condition
System.Xml.XmlException	The XML was not well-formed, or an error occurred while parsing the XML.

33

34

35

1 XmlReader.ReadOuterXml() Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract string  
4 ReadOuterXml()  
5 [C#]  
6 public abstract string ReadOuterXml()
```

7 Summary

8 Reads the current node and its contents, including child nodes and
9 markup.

10 Return Value

11

12 A **System.String** containing the XML content, or
13 **System.String.Empty** if the current node is neither an element nor
14 attribute.

15 Behaviors

16 The current node and corresponding end node are returned.

17

18 If the current node is an element, after the call to this method, the
19 reader is positioned after the corresponding end element.

20

21 If the current node is an attribute, the position of the reader is not
22 changed.

23

24 [Note: For a comparison between this method and the
25 **System.Xml.XmlReader.ReadOuterXml** method, see
26 **System.Xml.XmlTextReader.ReadInnerXml**.]

27 How and When to Override

28 This method must be overridden in order to provide the functionality
29 described above, as there is no default implementation.

30 Exceptions

31

32

Exception	Condition
System.Xml.XmlException	The XML was not well-formed, or an error occurred while parsing the XML.

33

34

35

1 XmlReader.ReadStartElement(System.String, System.String) Method

```
3 [ILASM]
4 .method public hidebysig virtual void
5 ReadStartElement(string localname, string ns)
6
7 [C#]
8 public virtual void ReadStartElement(string localname,
9 string ns)
```

9 Summary

10 Reads an **Element** node with the specified local name and namespace
11 URI and advances the reader to the next node.

12 Parameters

13
14

Parameter	Description
<i>localname</i>	A System.String specifying the local name of an element.
<i>ns</i>	A System.String specifying the namespace URI associated with the element.

15
16

17 Behaviors

18 As described above.

19 Default

20 This method calls the **System.Xml.XmlReader.MoveToContent**
21 method. If the returned node is an **Element** node, this method
22 compares the **System.Xml.XmlReader.LocalName** and
23 **System.Xml.XmlReader.NamespaceURI** properties of the node to
24 *localname* and *ns*, respectively. If they are equal, this method calls the
25 **System.Xml.XmlReader.Read** method to read the element and
26 move to the next node.

27 How and When to Override

28 Override this method to customize the behavior of this method in
29 types derived from the **System.Xml.XmlReader** class.

30 Usage

1 Use this method to read an **Element** node with the specified local
2 name and namespace URI, and advance the reader to the next node.

3 **Exceptions**

4
5

Exception	Condition
System.Xml.XmlException	The node is not an Element node, the System.Xml.XmlReader.LocalName property of the Element node does not equal <i>localname</i> , the System.Xml.XmlReader.NamespaceURI property of the Element node does not equal <i>ns</i> , or an error occurred while parsing the XML.

6
7
8

1 XmlReader.ReadStartElement(System.String) Method

```
3 [ILASM]
4 .method public hidebysig virtual void
5 ReadStartElement(string name)
6
7 [C#]
8 public virtual void ReadStartElement(string name)
```

8 Summary

9 Reads an **Element** node with the specified qualified name and
10 advances the reader to the next node.

11 Parameters

12
13

Parameter	Description
<i>name</i>	A System.String specifying the qualified name of an element.

14
15

16 Behaviors

17 As described above.

18 Default

19 This method calls the **System.Xml.XmlReader.MoveToContent**
20 method and, if the returned node is an **Element** node, compares the
21 **System.Xml.XmlReader.Name** property of the node to *name*. If
22 they are equal, this method calls the **System.Xml.XmlReader.Read**
23 method to read the element and move to the next node.

24 How and When to Override

25 Override this method to customize the behavior of this method in
26 types derived from the **System.Xml.XmlReader** class.

27 Usage

28 Use this method to read an **Element** node with the specified qualified
29 name, and advance the reader to the next node.

1 **Exceptions**
2
3

Exception	Condition
System.Xml.XmlException	The node is not an Element node, the System.Xml.XmlReader.Name property of the Element node does not equal <i>name</i> , or an error occurred while parsing the XML.

4
5
6

1 XmlReader.ReadStartElement() Method

```
2 [ILASM]  
3 .method public hidebysig virtual void ReadStartElement()  
4 [C#]  
5 public virtual void ReadStartElement()
```

6 Summary

7 Reads an **Element** node and advances the reader to the next node.

8 Behaviors

9 As described above.

10 Default

11 This method calls the **System.Xml.XmlReader.MoveToContent**
12 method, which determines whether the current node can contain
13 content and, if not, moves the reader to the next content node or the
14 end of the input stream. The node the reader ends up positioned on is
15 checked to determine if it is an **Element** node. If so, the node is read
16 and the reader is moved to the next node.

17 How and When to Override

18 Override this method to customize the behavior of this method in
19 types derived from the **System.Xml.XmlReader** class.

20 Usage

21 Use this method to read an **Element** node and advance the reader to
22 the next node.

23 Exceptions

24
25

Exception	Condition
System.Xml.XmlException	The node is not an Element node or an error occurred while parsing the XML.

26
27
28

1 XmlReader.ReadString() Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract string  
4 ReadString()  
5 [C#]  
6 public abstract string ReadString()
```

7 Summary

8 Reads the contents of an element or text node as a string.

9 Return Value

10

11 A **System.String** containing the contents of the **Element** or **Text**
12 node, or **System.String.Empty** if the reader is positioned on any
13 other type of node.

14 Behaviors

15 If positioned on an **Element** node, this method concatenates all **Text**,
16 **SignificantWhitespace**, **Whitespace**, and **CDATA** node types, and
17 returns the concatenated data as the element content. If none of these
18 node types exist, **System.String.Empty** is returned. Concatenation
19 stops when any markup is encountered, which can occur in a mixed
20 content model or when an element end tag is read.

21

22 If positioned on an element **Text** node, this method performs the
23 same concatenation from the **Text** node to the element end tag. If the
24 reader is positioned on an attribute **Text** node, this method has the
25 same functionality as if the reader were position on the element start
26 tag.

27 How and When to Override

28 This method must be overridden in order to provide the functionality
29 described above, as there is no default implementation.

30 Exceptions

31

32

Exception	Condition
System.Xml.XmlException	An error occurred while parsing the XML.

33

34

35

1 XmlReader.ResolveEntity() Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract void  
4 ResolveEntity()  
5 [C#]  
6 public abstract void ResolveEntity()
```

7 Summary

8 Resolves the entity reference for **EntityReference** nodes.

9 Behaviors

10 This method parses the entity reference into child nodes. When the
11 parsing is finished a new **System.Xml.XmlNodeType.EndEntity**
12 node is placed in the stream to close the **EntityReference** scope. To
13 step into the entity after this method has been called, call the
14 **System.Xml.XmlReader.ReadAttributeValue** method if the entity
15 is part of an attribute value, or the **System.Xml.XmlReader.Read**
16 method if the entity is part of element content.

17
18 If this method is not called, the parser moves to the next node past
19 the entity (child nodes are bypassed).

20 How and When to Override

21 This method must be overridden in order to provide the functionality
22 as described in the Behaviors and Usage sections, as there is no
23 default implementation.

24
25 This method is required to throw an exception for implementations
26 that do not support schema or DTD information. In this case, the
27 **System.Xml.XmlReader.CanResolveEntity** property is required to
28 return **false**.

29 Usage

30 Use this method to resolve the entity reference for **EntityReference**
31 nodes. Before calling this method, determine whether the reader can
32 resolve an entity by checking the
33 **System.Xml.XmlReader.CanResolveEntity** property.

34 Exceptions

35
36

Exception	Condition
System.InvalidOperationException	The reader is not positioned on a

1
2
3

	System.Xml.XmlNodeType.EntityReference node.
--	--

1 XmlReader.Skip() Method

```
2 [ILASM]  
3 .method public hidebysig virtual void Skip()  
4 [C#]  
5 public virtual void Skip()
```

6 Summary

7 Skips over the current element and moves the position of the current
8 instance to the next node in the stream.

9 Behaviors

10 If the reader is positioned on a non-empty **Element** node
11 (**System.Xml.XmlReader.IsEmptyElement** equals **false**), the
12 position of the reader is moved to the node following the
13 corresponding **EndElement** node. The properties of the nodes that are
14 skipped over are not exposed. If the reader is positioned on any other
15 node type, the position of the reader is moved to the next node, in this
16 case behaving like the **System.Xml.XmlReader.Read** method.

17 Default

18 This method calls the **System.Xml.XmlReader.MoveToElement**
19 method before skipping to the next node.

20 How and When to Override

21 Override this method to customize the behavior of this method in
22 types derived from the **System.Xml.XmlReader** class.

23 Usage

24 Use this method to skip over the current node.

25 Exceptions

26
27

Exception	Condition
System.Xml.XmlException	The XML was not well-formed, or an error occurred while parsing the XML.

28
29
30

1 XmlReader.AttributeCount Property

```
2 [ILASM]  
3 .property int32 AttributeCount { public hidebysig virtual  
4 abstract specialname int32 get_AttributeCount() }  
5 [C#]  
6 public abstract int AttributeCount { get; }
```

7 Summary

8 Gets the number of attributes on the current node.

9 Property Value

10

11 A **System.Int32** containing the number of attributes on the current
12 node, or zero if the current node does not support attributes.

13 Description

14 [*Note:* This property is only relevant to the **DocumentType**,
15 **Element**, and **XmlDeclaration** node types of the
16 **System.Xml.XmlNodeType** enumeration. Other node types do not
17 have attributes.]

18 Behaviors

19 As described above.

20

21 This property is read-only.

22 How and When to Override

23 This property must be overridden in order to provide the functionality
24 described above, as there is no default implementation.

25

1 XmlReader.BaseURI Property

```
2 [ILASM]
3 .property string BaseURI { public hidebysig virtual
4 abstract specialname string get_BaseURI() }
5
6 [C#]
7 public abstract string BaseURI { get; }
```

7 Summary

8 Gets the base Uniform Resource Identifier (URI) of the current node.

9 Property Value

10

11 The base URI of the current node.

12 Description

13 [Note: A networked XML document is comprised of chunks of data
14 aggregated using various W3C standard inclusion mechanisms and
15 therefore contains nodes that come from different places. DTD entities
16 are an example of this, but this is not limited to DTDs. The base URI
17 tells where these nodes come from. If there is no base URI for the
18 nodes being returned (for example, they were parsed from an in-
19 memory string), **System.String.Empty** is returned.]

20 Behaviors

21 As described above.

22

23 This property is read-only.

24 How and When to Override

25 This property must be overridden in order to provide the functionality
26 described above, as there is no default implementation.

27

1 XmlReader.CanResolveEntity Property

```
2 [ILASM]  
3 .property bool CanResolveEntity { public hidebysig virtual  
4 specialname bool get_CanResolveEntity() }  
5 [C#]  
6 public virtual bool CanResolveEntity { get; }
```

7 Summary

8 Gets a value indicating whether this reader can parse and resolve
9 entities.

10 Property Value

11

12 A **System.Boolean** equal to **false**.

13 Behaviors

14 This property returns **true** to indicate the reader can parse and resolve
15 entities; otherwise, **false**.

16

17 This property is read-only.

18 Default

19 This property always returns **false**.

20 How and When to Override

21 Override this property to return **true** for implementations that support
22 schema or DTD information.

23 Usage

24 Use this property to determine whether the reader can parse and
25 resolve entities.

26

1 XmlReader.Depth Property

```
2 [ILASM]
3 .property int32 Depth { public hidebysig virtual abstract
4 specialname int32 get_Depth() }
5
6 [C#]
7 public abstract int Depth { get; }
```

7 Summary

8 Gets the depth of the current node in the XML document.

9 Property Value

10

11 A **System.Int32** containing the depth of the current node in the XML
12 document.

13 Behaviors

14 As described above.

15

16 This property is read-only.

17 How and When to Override

18 This property must be overridden in order to provide the functionality
19 described above, as there is no default implementation.

20

1 XmlReader.EOF Property

```
2 [ILASM]
3 .property bool EOF { public hidebysig virtual abstract
4 specialname bool get_EOF() }
5
6 [C#]
7 public abstract bool EOF { get; }
```

7 Summary

8 Gets a value indicating whether the
9 **System.Xml.XmlReader.ReadState** is
10 **System.Xml.ReadState.EndOfFile**, signifying the reader is
11 positioned at the end of the stream.

12 Property Value

13

14 A **System.Boolean** where **true** indicates the reader is positioned at
15 the end of the stream; otherwise, **false**.

16 Behaviors

17 As described above.

18

19 This property is read-only.

20 How and When to Override

21 This property must be overridden in order to provide the functionality
22 described above, as there is no default implementation.

23

1 XmlReader.HasAttributes Property

```
2 [ILASM]
3 .property bool HasAttributes { public hidebysig virtual
4 specialname bool get_HasAttributes() }
5
6 [C#]
7 public virtual bool HasAttributes { get; }
```

7 Summary

8 Gets a value indicating whether the current node has any attributes.

9 Property Value

10

11 A **System.Boolean** where **true** indicates the current node has
12 attributes; otherwise, **false**.

13 Behaviors

14 As described above.

15

16 This property is read-only.

17 Default

18 This property returns **true** if the
19 **System.Xml.XmlReader.AttributeCount** property of the current
20 node is greater than zero.

21 How and When to Override

22 Override this property to customize the behavior of this property in
23 types derived from the **System.Xml.XmlReader** class.

24 Usage

25 Use this property to determine whether the current node has any
26 attributes.

27

1 XmlReader.HasValue Property

```
2 [ILASM]
3 .property bool HasValue { public hidebysig virtual abstract
4 specialname bool get_HasValue() }
5
6 [C#]
7 public abstract bool HasValue { get; }
```

7 Summary

8 Gets a value indicating whether the current node can have an
9 associated text value.

10 Property Value

11

12 A **System.Boolean** where **true** indicates the node on which the
13 reader is currently positioned can have an associated text value;
14 otherwise, **false**.

15 Description

16 [Note: The following members of the **System.Xml.XmlNodeType**
17 enumeration can have an associated value: **Attribute**, **CDATA**,
18 **Comment**, **DocumentType**, **ProcessingInstruction**,
19 **SignificantWhitespace**, **Text**, **Whitespace**, and **XmlDeclaration**.]

20 Behaviors

21 As described above.

22

23 This property is read-only.

24 How and When to Override

25 This property must be overridden in order to provide the functionality
26 described above, as there is no default implementation.

27

1 XmlReader.IsDefault Property

```
2 [ILASM]  
3 .property bool IsDefault { public hidebysig virtual  
4 abstract specialname bool get_IsDefault() }  
5  
6 [C#]  
7 public abstract bool IsDefault { get; }
```

7 Summary

8 Gets a value indicating whether the current node is an attribute that
9 was generated from the default value defined in the DTD or schema.

10 Property Value

11

12 A **System.Boolean** where **true** indicates the current node is an
13 attribute whose value was generated from the default value defined in
14 the DTD or schema; **false** indicates the attribute value was explicitly
15 set.

16 Behaviors

17 As described above.

18

19 This property is read-only.

20 How and When to Override

21 This property should return **false** for implementations that do not
22 support schema or DTD information.

23

1 XmlReader.IsEmptyElement Property

```
2 [ILASM]
3 .property bool IsEmptyElement { public hidebysig virtual
4 abstract specialname bool get_IsEmptyElement() }
5 [C#]
6 public abstract bool IsEmptyElement { get; }
```

7 Summary

8 Gets a value indicating whether the current node is an empty element
9 (for example, <MyElement />).

10 Property Value

11

12 A **System.Boolean** where **true** indicates the current node is an
13 element (**System.Xml.XmlReader.NodeType** equals
14 **System.Xml.XmlNodeType.Element**) that ends with ">",
15 otherwise, **false**.

16 Behaviors

17 A corresponding **EndElement** node is not generated for empty
18 elements.

19

20 This property is read-only.

21 How and When to Override

22 This property must be overridden in order to provide the functionality
23 described above, as there is no default implementation.

24

1 XmlReader.Item Property

```
2 [ILASM]  
3 .property string Item[int32 i] { public hidebysig virtual  
4 abstract specialname string get_Item(int32 i) }  
5 [C#]  
6 public abstract string this[int i] { get; }
```

7 Summary

8 Retrieves the value of the attribute with the specified index relative to
9 the containing element.

10 Parameters

11
12

Parameter	Description
<i>i</i>	A System.Int32 specifying the zero-based index of the attribute relative to the containing element.

13
14
15

14 Property Value

16 A **System.String** containing the value of the attribute.

17 Behaviors

18 This property does not move the reader.

19 How and When to Override

20 This property must be overridden in order to provide the functionality
21 described above, as there is no default implementation.

22 Exceptions

23
24

Exception	Condition
System.ArgumentOutOfRangeException	<i>i</i> is less than 0 or greater than or equal to the System.Xml.XmlReader.AttributeCount of the containing element.

25
26
27

1 XmlReader.Item Property

```
2 [ILASM]
3 .property string Item[string name] { public hideby sig
4 virtual abstract specialname string get_Item(string name) }
5 [C#]
6 public abstract string this[string name] { get; }
```

7 Summary

8 Retrieves the value of the attribute with the specified qualified name.

9 Parameters

10
11

Parameter	Description
<i>name</i>	A System.String specifying the qualified name of the attribute.

12
13
14

13 Property Value

15 A **System.String** containing the value of the specified attribute, or
16 **null** if the attribute is not found.

17 Behaviors

18 This property does not move the reader.
19
20 If the reader is positioned on a **DocumentType** node, this method can
21 be used to get the PUBLIC and SYSTEM literals.

22 How and When to Override

23 This property must be overridden in order to provide the functionality
24 described above, as there is no default implementation.

25

1 XmlReader.Item Property

```
2 [ILASM]
3 .property string Item[string name, string namespaceURI] {
4 public hidebysig virtual abstract specialname string
5 get_Item(string name, string namespaceURI) }
6
7 [C#]
8 public abstract string this[string name, string
9 namespaceURI] { get; }
```

9 Summary

10 Retrieves the value of the attribute with the specified local name and
11 namespace URI.

12 Parameters

13
14

Parameter	Description
<i>name</i>	A System.String specifying the local name of the attribute.
<i>namespaceURI</i>	A System.String specifying the namespace URI of the attribute.

15
16
17

16 Property Value

18 A **System.String** containing the value of the specified attribute, or
19 **null** if the attribute is not found.

20 Behaviors

21 This property does not move the reader.

22 How and When to Override

23 This property must be overridden in order to provide the functionality
24 described above, as there is no default implementation.

25

1 XmlReader.LocalName Property

```
2 [ILASM]
3 .property string LocalName { public hidebysig virtual
4 abstract specialname string get_LocalName() }
5
6 [C#]
7 public abstract string LocalName { get; }
```

7 Summary

8 Gets the local name of the current node.

9 Property Value

10

11 A **System.String** containing the local name of the current node or, for
12 node types that do not have a name (like **Text**, **Comment**, and so
13 on), **System.String.Empty**.

14 Behaviors

15 As described above.

16 How and When to Override

17 This property must be overridden in order to provide the functionality
18 described above, as there is no default implementation.

19

1 XmlReader.Name Property

```
2 [ILASM]
3 .property string Name { public hidebysig virtual abstract
4 specialname string get_Name() }
5
6 [C#]
7 public abstract string Name { get; }
```

7 Summary

8 Gets the qualified name of the current node.

9 Property Value

10

11 A **System.String** containing the qualified name of the current node
12 or, for node types that do not have a name (like **Text**, **Comment**, and
13 so on), **System.String.Empty**.

14 Behaviors

15 The qualified name is equivalent to the
16 **System.Xml.XmlReader.LocalName** prefixed with
17 **System.Xml.XmlReader.Prefix** and the ':' character. For example,
18 **System.Xml.XmlReader.Name** is "bk:book" for the element
19 <bk:book>.

20

21 The name returned is dependent on the
22 **System.Xml.XmlReader.NodeType** of the node. The following node
23 types return the listed values. All other node types return an empty
24 string.

Node Type	Name
Attribute	The name of the attribute.
DocumentType	The document type name.
Element	The tag name.
EntityReference	The name of the entity referenced.
ProcessingInstruction	The target of the processing instruction.
XmlDeclaration	The literal string "xml".

25

26 This property is read-only.

27 How and When to Override

1 This property must be overridden in order to provide the functionality
2 described above, as there is no default implementation.

3

1 XmlReader.NamespaceURI Property

```
2 [ILASM]
3 .property string NamespaceURI { public hidebysig virtual
4 abstract specialname string get_NamespaceURI() }
5 [C#]
6 public abstract string NamespaceURI { get; }
```

7 Summary

8 Gets the namespace URI associated with the node on which the reader
9 is positioned.

10 Property Value

11

12 A **System.String** containing the namespace URI of the current node
13 or, if no namespace URI is associated with the current node,
14 **System.String.Empty**.

15 Behaviors

16 This property is relevant to **Element** and **Attribute** nodes only.

17

18 Namespaces conform to the W3C "Namespaces in XML"
19 recommendation, REC-xml-names-19990114.

20

21 This property is read-only.

22 How and When to Override

23 This property must be overridden in order to provide the functionality
24 described above, as there is no default implementation.

25

1 XmlReader.NameTable Property

```
2 [ILASM]
3 .property class System.Xml.XmlNameTable NameTable { public
4 hidebysig virtual abstract specialname class
5 System.Xml.XmlNameTable get_NameTable() }
6
7 [C#]
8 public abstract XmlNameTable NameTable { get; }
```

8 Summary

9 Gets the name table used by the current instance to store and look up
10 element and attribute names, prefixes, and namespaces.

11 Property Value

12

13 The **System.Xml.XmlNameTable** used by the current instance.

14 Behaviors

15 Element and attribute names, prefixes, and namespaces are stored as
16 individual **System.String** objects when a document is read.

17

18 This property is read-only.

19 How and When to Override

20 This property must be overridden in order to provide the functionality
21 described above, as there is no default implementation.

22

1 XmlReader.NodeType Property

```
2 [ILASM]
3 .property valuetype System.Xml.XmlNodeType NodeType {
4 public hidebysig virtual abstract specialname valuetype
5 System.Xml.XmlNodeType get_NodeType() }
6
7 [C#]
8 public abstract XmlNodeType NodeType { get; }
```

8 Summary

9 Gets the type of the current node.

10 Property Value

11

12 One of the members of the **System.Xml.XmlNodeType** enumeration
13 representing the type of the current node.

14 Behaviors

15 This property does not return the following
16 **System.Xml.XmlNodeType** members: **Document**,
17 **DocumentFragment**, **Entity**, **EndEntity**, and **Notation**.

18
19 This property is read-only.

20 How and When to Override

21 This property must be overridden in order to provide the functionality
22 described above, as there is no default implementation.

23

1 XmlReader.Prefix Property

```
2 [ILASM]
3 .property string Prefix { public hidebysig virtual abstract
4 specialname string get_Prefix() }
5
6 [C#]
7 public abstract string Prefix { get; }
```

7 Summary

8 Gets the namespace prefix associated with the current node.

9 Property Value

10

11 A **System.String** containing the namespace prefix associated with the
12 current node.

13 Description

14 [*Note:* A namespace prefix is used as a reference for a namespace URI
15 and is defined in an element declaration. For example, <someElement
16 xmlns:bk="someURL">, defines a prefix name "bk".]

17 Behaviors

18 As described above.

19

20 This property is read-only.

21 How and When to Override

22 This property must be overridden in order to provide the functionality
23 described above, as there is no default implementation.

24

1 XmlReader.QuoteChar Property

```
2 [ILASM]  
3 .property valuetype System.Char QuoteChar { public  
4 hidebysig virtual abstract specialname valuetype  
5 System.Char get_QuoteChar() }  
6  
7 [C#]  
8 public abstract char QuoteChar { get; }
```

8 Summary

9 Gets the quotation mark character used to enclose the value of an
10 attribute.

11 Property Value

12

13 A **System.Char** specifying the quotation mark character (" or ') used
14 to enclose the value of an attribute.

15 Behaviors

16 As described above.

17

18 This property is read-only.

19 How and When to Override

20 This property must be overridden in order to provide the functionality
21 described above, as there is no default implementation.

22

1 XmlReader.ReadState Property

```
2 [ILASM]
3 .property valuetype System.Xml.ReadState ReadState { public
4 hidebysig virtual abstract specialname valuetype
5 System.Xml.ReadState get_ReadState() }
6
7 [C#]
8 public abstract ReadState ReadState { get; }
```

8 Summary

9 Gets the read state of the reader.

10 Property Value

11

12 One of the members of the **System.Xml.ReadState** enumeration.

13 Behaviors

14 As described above.

15

16 This property is read-only.

17 How and When to Override

18 This property must be overridden in order to provide the functionality
19 described above, as there is no default implementation.

20

1 XmlReader.Value Property

```
2 [ILASM]  
3 .property string Value { public hidebysig virtual abstract  
4 specialname string get_Value() }  
5  
6 [C#]  
7 public abstract string Value { get; }
```

7 Summary

8 Gets the text value of the current node.

9 Property Value

10

11 A **System.String** containing the text value of the current node.

12 Behaviors

13 The value returned depends on the
14 **System.Xml.XmlReader.NodeType**. The following table lists node
15 types that have a value to return. All other node types return
16 **System.String.Empty**.

Node Type	Value
Attribute	The value of the attribute.
CDATA	The content of the CDATA section.
Comment	The content of the comment.
DocumentType	The internal subset.
ProcessingInstruction	The entire content, excluding the target.
SignificantWhitespace	The white space between markup in a mixed content model, or in the scope of <code>xml:space = "preserve"</code> .
Text	The content of the text node.
Whitespace	The white space between markup.
XmlDeclaration	The content of the declaration.

17

18 This property is read-only.

19 How and When to Override

20 This property must be overridden in order to provide the functionality
21 described above, as there is no default implementation.

22

1 XmlReader.XmlLang Property

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

7 Summary

8 Gets the current `xml:lang` scope.

9 Property Value

10

11 A **System.String** containing the current `xml:lang` scope.

12 Behaviors

13 As described above.

14

15 This property is read-only.

16 How and When to Override

17 This property must be overridden in order to provide the functionality
18 described above, as there is no default implementation.

19

1 XmlReader.XmlSpace Property

```
2 [ILASM]
3 .property valuetype System.Xml.XmlSpace XmlSpace { public
4 hidebysig virtual abstract specialname valuetype
5 System.Xml.XmlSpace get_XmlSpace() }
6
7 [C#]
8 public abstract XmlSpace XmlSpace { get; }
```

8 Summary

9 Gets the current `xml:space` scope.

10 Property Value

11

12 One of the members of the **System.Xml.XmlSpace** enumeration. If
13 no `xml:space` scope exists, this property defaults to
14 **System.Xml.XmlSpace.None**.

15 Behaviors

16 As described above.

17

18 This property is read-only.

19 How and When to Override

20 This property must be overridden in order to provide the functionality
21 described above, as there is no default implementation.

22