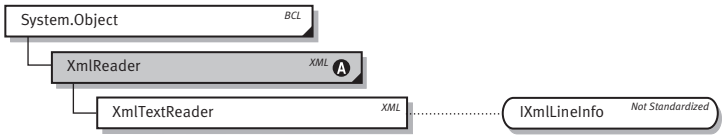
System.Xml
XmlReader

XML Library



Summary

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

Type Summary

```
public abstract class XmlReader
    {
    // Constructors
       protected XmlReader();
    // Properties
       public abstract int AttributeCount { get; }
       public abstract string BaseURI { get; }
       public virtual bool CanResolveEntity { get; }
       public abstract int Depth { get; }
       public abstract bool EOF { get; }
       public virtual bool HasAttributes { get; }
       public abstract bool HasValue { get; }
       public abstract bool IsDefault { get; }
       public abstract bool IsEmptyElement { get; }
       public abstract string LocalName { get; }
       public abstract string Name { get; }
       public abstract string NamespaceURI { get; }
       public abstract XmlNameTable NameTable { get; }
       public abstract XmlNodeType NodeType { get; }
       public abstract string Prefix { get; }
       public abstract char QuoteChar { get; }
       public abstract ReadState ReadState { get; }
       public abstract string Value { get; }
       public abstract string XmlLang { get; }
       public abstract XmlSpace XmlSpace { get; }
       public abstract string this[int i] { get; }
       public abstract string this[string name] { get; }
       public abstract string this[string name, string namespaceURI] { get; }
    // Methods
       public abstract void Close();
       public abstract string GetAttribute(int i);
       public abstract string GetAttribute(string name);
       public abstract string GetAttribute(string name, string namespaceURI);
       public static bool IsName(string str);
       public static bool IsNameToken(string str);
       public virtual bool IsStartElement();
       public virtual bool IsStartElement(string name);
       public virtual bool IsStartElement(string localname, string ns);
       public abstract string LookupNamespace(string prefix);
       public abstract void MoveToAttribute(int i);
       public abstract bool MoveToAttribute(string name);
       public abstract bool MoveToAttribute(string name, string ns);
       public virtual XmlNodeType MoveToContent();
       public abstract bool MoveToElement();
       public abstract bool MoveToFirstAttribute();
       public abstract bool MoveToNextAttribute();
       public abstract bool Read();
       public abstract bool ReadAttributeValue();
       public virtual string ReadElementString();
```

```
    public virtual string ReadElementString(string name);
    public virtual string ReadElementString(string localname, string ns);
    public virtual void ReadEndElement();
    public virtual string ReadInnerXml();
    public virtual string ReadOuterXml();
    public virtual void ReadStartElement();
    public virtual void ReadStartElement(string name);
    public virtual void ReadStartElement(string localname, string ns);
    public virtual string ReadString();
    public abstract void ResolveEntity();
    public virtual void Skip();
}
```

MF

The `XmlReader` was a significant innovation in XML parsing due to its pull model approach and caused a significant amount of acclaim in the industry. Until then developers had to endure the SAX push model, which although great for XML parser implementers (just push out an event when something happens), was painful for developers to use. The `XmlReader` reversed this position allowing you to do simple, procedural code where you could get the `XmlReader` to do the heavy lifting for you, like skipping elements that you did not care about or throwing away white space from the document. The `XmlReader` was inspired by seeing the use of other readers in the .NET Framework such as the `StringReader` class and applying the same principles to XML.

MF

There are aspects of the `XmlReader` design where we had to choose between usability and performance. For example, due to the attribute indexer methods all the attributes for an element have to be cached, which does not allow for a complete streaming API. On the whole the majority of XML documents have small numbers of attributes so improved usability was the best design.

CL

In addition to the reason Mark just noted, you have to cache all the attributes anyway in order to know the namespace of the element. SAX had the exact same problem—only worse. In SAX, an `Array` object is created with all of the attributes that is then passed to your handler. At least here we avoid that array creation.

JM

The `Read` and `Skip` methods are, of course, different beasts. But they are more similar than you might think. If the reader is on any node besides a non-empty element node, `Read` and `Skip` behave the same. Otherwise, `Skip` positions the reader following the corresponding `EndElement` node, not exposing any properties along the way (while `Read` will expose the properties).

CL

In general the `XmlReader` makes recursive descent parsing of a given XML document a snap. In fact, it is so easy that the `XmlSerializer` generates IL code for you that calls the `XmlReader` to parse the XML while it builds your own custom objects from what it finds in the stream. Hence, `XmlSerialization` is probably the #1 customer of the `XmlReader` class in terms of overall volume of XML parsed.

Notice that `XmlReader` is abstract. If you want to just parse XML text, you need the concrete subclass `XmlTextReader` discussed later. Why did we bother with an abstract base class? Because we envisioned the creation of lots of different kinds of XML readers that took their data from other sources besides text. It also means you can plug in additional behavior into the XML processing pipeline. Suppose someone consumes an XML reader. Well, you can wrap the `XmlTextReader` with an `XmlValidatingReader` and pass that instead, and now you are causing your consumer to validate while they parse without them even knowing it.

HK
Although the `XmlReader` was designed as an abstract class, it has a few properties that are bound to the text representation of XML, such as `QuoteChar` or `IsEmptyElement`. As was pointed out quite a few times by our users, it is annoying that the `XmlReader` reports empty and non-empty elements differently. For non-empty element <a>… the reader returns `Element` and `EndElement` events, but for empty element <a/> it only returns a single `Element` event. The user has to check the `IsEmptyElement` property to see whether he should expect a closing `EndElement` or not. In many cases this causes two similar code paths in the handling code, one for empty elements and one for non-empty ones.

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 `XmlNodeType` enumeration. The reader is advanced to the next node using the `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 `xml:lang` scope within which the node resides.

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 `XmlConvert` class.

This class throws an `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 `XmlTextReader` class.]

Example

```
using System;
using System.Xml;


public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.XmlDeclaration
                || r.NodeType == XmlNodeType.Element)
            {
                Console.WriteLine("NodeType {0} ('{1}') is at Depth: '{2}'",
                    r.NodeType, r.Name, r.Depth);
                if (r.HasValue)
                {
                    Console.Write(" Value: '{0}'", r.Value);
                }


                Console.WriteLine("Element <{0}> has {1} attribute(s)",
                    r.Name, r.AttributeCount);
                if (r.AttributeCount > 0)
                {
                    string fn = String.Empty;
                    string ln = String.Empty;
                    string ns = String.Empty;
                    while (r.MoveToNextAttribute())
                    {
                        fn = r.Name;
                        ln = r.LocalName;
                        ns = r.NamespaceURI;
                        Console.WriteLine("- Attribute: '{0}', "
                            + "LocalName: '{1}', Value: '{2}'",
                            fn, ln, r.GetAttribute(ln, ns));
                    }

                }
            }
```

```
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();

    }
}
```

The output is

```
NodeType XmlDeclaration ('xml') is at Depth: '0'
 Value: 'version="1.0"'Element <xml> has 1 attribute(s)
- Attribute: 'version', LocalName: 'version', Value: '1.0'
NodeType Element ('root') is at Depth: '0'
Element <root> has 1 attribute(s)
- Attribute: 'xmlns:test', LocalName: 'test', Value: 'http://mysite/prefixes'
NodeType Element ('test:subelement') is at Depth: '1'
Element <test:subelement> has 2 attribute(s)
- Attribute: 'test:first', LocalName: 'first', Value: 'one'
- Attribute: 'test:second', LocalName: 'second', Value: 'two'


Press Enter to continue
```

XmlReader() Constructor

```
[ILASM]
family rtspecialname specialname instance void .ctor()
[C#]
protected XmlReader()
```

Summary

Constructs a new instance of the `XmlReader` class.

Example

```
using System;
using System.Xml;

namespace Samples
{
    public class XmlReaderSample
    {
        private static void ReadXmlDocument(XmlReader r)
        {
            Console.WriteLine();
            Console.WriteLine("Reading XML document...");
            while (r.Read())
                Console.WriteLine("NodeType: {0} ({1})",
                    r.NodeType, r.Name);
            r.Close();
        }
        public static void Main()
        {
            XmlTextReader r = new XmlTextReader("sample.xml");
            ReadXmlDocument(r);
            Console.WriteLine();
            Console.WriteLine();
            Console.WriteLine("Press Enter to continue");
            Console.ReadLine();
        }
```

```
    }
}
```

The output is

```
Reading XML document...
NodeType: XmlDeclaration (xml)
NodeType: Element (root)
NodeType: Element (subelement)
NodeType: Text ()
NodeType: EndElement (subelement)
NodeType: EndElement (root)


Press Enter to continue
```

XmlReader.AttributeCount Property

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

Summary

Gets the number of attributes on the current node.

Property Value

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

Description

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

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.XmlDeclaration
                || r.NodeType == XmlNodeType.Element)
                Console.WriteLine("Element <{0}> has {1} attribute(s)",
                    r.Name, r.AttributeCount);
        }
        r.Close();
```

```
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element <xml> has 1 attribute(s)
Element <root> has 0 attribute(s)
Element <subelement> has 2 attribute(s)


Press Enter to continue
```

XmlReader.BaseURI Property

```
[ILASM]
.property string BaseURI { public hidebysig virtual abstract specialname string
get_BaseURI() }
[C#]
public abstract string BaseURI { get; }
```

Summary

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

Property Value

The base URI of the current node.

Description

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

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            Console.WriteLine("NodeType: {0} has BaseURI: '{1}'",
                r.NodeType, r.BaseURI);
        }
```

```
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType: XmlDeclaration has BaseURI:'file:///C:/System.Xml/XmlReader/BaseURI/sa
mple.xml'
NodeType: Element has BaseURI: 'file:///C:/System.Xml/XmlReader/BaseURI/sample.x
ml'
NodeType: Element has BaseURI: 'file:///C:/System.Xml/XmlReader/BaseURI/sample.x
ml'
NodeType: Text has BaseURI: 'file:///C:/System.Xml/XmlReader/BaseURI/sample.xml'
NodeType: EndElement has BaseURI: 'file:///C:/System.Xml/XmlReader/BaseURI/sampl
e.xml'
NodeType: EndElement has BaseURI: 'file:///C:/System.Xml/XmlReader/BaseURI/sampl
e.xml'
NodeType: Whitespace has BaseURI: 'file:///C:/System.Xml/XmlReader/BaseURI/sampl
e.xml'


Press Enter to continue
```

XmlReader.CanResolveEntity Property

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

Summary

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

Property Value

A `System.Boolean` equal to false.

Behaviors

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

Default

This property always returns `false`.

How and When to Override

Override this property to return `true` for implementations that support schema or DTD information.

Usage

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        XmlValidatingReader vr = new XmlValidatingReader(r);
        vr.ValidationType = ValidationType.None;
        vr.EntityHandling = EntityHandling.ExpandCharEntities;
        while (vr.Read())
        {
            Console.Write("NodeType: {0}, Name: '{1}'", vr.NodeType, vr.Name);
            if (vr.HasValue)
                Console.Write(", Value: '{0}'", vr.Value);
            Console.WriteLine();
            if (vr.NodeType == XmlNodeType.EntityReference)
            {
                Console.WriteLine("- CanResolveEntity: {0}",
                    vr.CanResolveEntity);
                vr.ResolveEntity();
                Console.WriteLine("- Executed ResolveEntity() method, "
                    + " value inserted as next text node");
            }
        }
        vr.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType: XmlDeclaration, Name: 'xml', Value: 'version="1.0"'
NodeType: DocumentType, Name: 'sample', Value: ''
NodeType: Element, Name: 'sample'
NodeType: Element, Name: 'book'
NodeType: Text, Name: '', Value: 'Microsoft .NET by '
NodeType: EntityReference, Name: 'pub'
- CanResolveEntity: True
- Executed ResolveEntity() method,  value inserted as next text node
NodeType: Text, Name: '', Value: 'Addison Wesley'
NodeType: EndEntity, Name: 'pub'
NodeType: EndElement, Name: 'book'
NodeType: EndElement, Name: 'sample'
NodeType: Whitespace, Name: '', Value: '
'


Press Enter to continue
```

XmlReader.Depth Property

```
[ILASM]
.property int32 Depth { public hidebysig virtual abstract specialname int32
get_Depth() }
[C#]
public abstract int Depth { get; }
```

Summary

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

Property Value

A `System.Int32` containing the depth of the current node in the XML document.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            Console.WriteLine("NodeType {0} ('{1}') is at Depth: '{2}'",
                r.NodeType, r.Name, r.Depth);
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType XmlDeclaration ('xml') is at Depth: '0'
NodeType Element ('root') is at Depth: '0'
NodeType Element ('subelement') is at Depth: '1'
NodeType Element ('subsubelement') is at Depth: '2'
NodeType Text ('') is at Depth: '3'
NodeType EndElement ('subsubelement') is at Depth: '2'
NodeType EndElement ('subelement') is at Depth: '1'
NodeType EndElement ('root') is at Depth: '0'
NodeType Whitespace ('') is at Depth: '0'


Press Enter to continue
```

XmlReader.EOF Property

```
[ILASM]
.property bool EOF { public hidebysig virtual abstract specialname bool
get_EOF() }
[C#]
public abstract bool EOF { get; }
```

Summary

Gets a value indicating whether the `ReadState` is `ReadState.EndOfFile`, signifying the reader is positioned at the end of the stream.

Property Value

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

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            Console.WriteLine("NodeType {0} ('{1}'), EOF: '{2}'",
                r.NodeType, r.Name, r.EOF);
        }
        Console.WriteLine("After Read()=False, EOF: '{0}'", r.EOF);
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType XmlDeclaration ('xml'), EOF: 'False'
NodeType Element ('root'), EOF: 'False'
NodeType Element ('subelement'), EOF: 'False'
NodeType Text (''), EOF: 'False'
NodeType EndElement ('subelement'), EOF: 'False'
NodeType EndElement ('root'), EOF: 'False'
NodeType Whitespace (''), EOF: 'False'
After Read()=False, EOF: 'True'


Press Enter to continue
```

XmlReader.HasAttributes Property

```
[ILASM]
.property bool HasAttributes { public hidebysig virtual specialname bool
get_HasAttributes() }
[C#]
public virtual bool HasAttributes { get; }
```

Summary

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

Property Value

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

Default

This property returns `true` if the `AttributeCount` property of the current node is greater than zero.

How and When to Override

Override this property to customize the behavior of this property in types derived from the `XmlReader` class.

Usage

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

Example

```
using System;
using System.Xml;


public class XmlReaderSample
{
    private static void ReadXmlDocument(XmlReader r)
    {
        Console.WriteLine();
        Console.WriteLine("Reading XML document...");
        while (r.Read())
        {
            Console.WriteLine("NodeType: {0} ({1})", r.NodeType, r.Name);
            if (r.HasAttributes)
            {
                while (r.MoveToNextAttribute())
                    Console.WriteLine("- Attribute '{0}', Value: '{1}'",
                        r.Name, r.Value);
            }
        }
        r.Close();
    }
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        ReadXmlDocument(r);
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Reading XML document...
NodeType: XmlDeclaration (xml)
- Attribute 'version', Value: '1.0'
NodeType: Element (root)
- Attribute 'sample', Value: 'value'
NodeType: Element (subelement)
NodeType: Text ()
NodeType: EndElement (subelement)
```

```
NodeType: EndElement (root)


Press Enter to continue
```

XmlReader.HasValue Property

```
[ILASM]
.property bool HasValue { public hidebysig virtual abstract specialname bool
get_HasValue() }
[C#]
public abstract bool HasValue { get; }
```

Summary

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

Property Value

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

Description

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

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    private static void ReadXmlDocument(XmlReader r)
    {
        Console.WriteLine();
        Console.WriteLine("Reading XML document...");
        while (r.Read())
        {
            Console.Write("NodeType: {0} ({1})", r.NodeType, r.Name);
            if (r.HasValue)
                Console.Write(" Value: '{0}'", r.Value);
            Console.WriteLine();
        }
        r.Close();
    }
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        ReadXmlDocument(r);
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
```

```
        Console.ReadLine();
    }
}
```

The output is

```
Reading XML document...
NodeType: XmlDeclaration (xml) Value: 'version="1.0"'
NodeType: Element (root)
NodeType: Element (subelement)
NodeType: Text () Value: 'sample'
NodeType: EndElement (subelement)
NodeType: EndElement (root)


Press Enter to continue
```

XmlReader.IsDefault Property

```
[ILASM]
.property bool IsDefault { public hidebysig virtual abstract specialname bool
get_IsDefault() }
[C#]
public abstract bool IsDefault { get; }
```

Summary

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

Property Value

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

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    private static void ReadXmlDocument(XmlReader r)
    {
        while (r.Read())
        {
            if (r.HasAttributes)
                while (r.MoveToNextAttribute())
                    Console.WriteLine("Attribute Name: '{0}', "
                        + "Value: '{0}', IsDefault: {2}",
                        r.Name, r.Value, r.IsDefault.ToString());
        }
        r.Close();
    }
    public static void Main()
    {
```

```
        XmlTextReader r = new XmlTextReader("sample.xml");
        Console.WriteLine();
        Console.WriteLine("Reading XML with an XmlTextReader:");
        ReadXmlDocument(r);
        r = new XmlTextReader("sample.xml");
        XmlValidatingReader vr = new XmlValidatingReader(r);
        Console.WriteLine();
        Console.WriteLine("Reading XML with an XmlValidatingReader:");
        ReadXmlDocument(vr);
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Reading XML with an XmlTextReader:
Attribute Name: 'version', Value: 'version', IsDefault: False
Attribute Name: 'SYSTEM', Value: 'SYSTEM', IsDefault: False

Reading XML with an XmlValidatingReader:
Attribute Name: 'version', Value: 'version', IsDefault: False
Attribute Name: 'SYSTEM', Value: 'SYSTEM', IsDefault: False
Attribute Name: 'publisher', Value: 'publisher', IsDefault: True


Press Enter to continue
```

XmlReader.IsEmptyElement Property

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

Summary

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

Property Value

A `System.Boolean` where `true` indicates the current node is an element (`NodeType` equals `XmlNodeType.Element`) that ends with "`/>`", otherwise, `false`.

Behaviors

A corresponding `EndElement` node is not generated for empty elements.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.Element)
                Console.WriteLine("NodeType {0} ('{1}'), "
                    + "IsEmptyElement: {2}",
                    r.NodeType, r.Name,
                    r.IsEmptyElement);
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType Element ('root'), IsEmptyElement: False
NodeType Element ('subelement1'), IsEmptyElement: True
NodeType Element ('subelement2'), IsEmptyElement: False


Press Enter to continue
```

XmlReader.LocalName Property

```
[ILASM]
.property string LocalName { public hidebysig virtual abstract specialname
string get_LocalName() }
[C#]
public abstract string LocalName { get; }
```

Summary

Gets the local name of the current node.

Property Value

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

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
```

```
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.XmlDeclaration
                || r.NodeType == XmlNodeType.Element)
            {
                Console.WriteLine("Element: <{0}> LocalName: '{1}' "
                    + "has {2} attribute(s)",
                    r.Name, r.LocalName,
                    r.AttributeCount);
                if (r.AttributeCount > 0)
                {
                    string fn = String.Empty;
                    string ln = String.Empty;
                    string ns = String.Empty;
                    while (r.MoveToNextAttribute())
                    {
                        fn = r.Name;
                        ln = r.LocalName;
                        ns = r.NamespaceURI;
                        Console.WriteLine("- Attribute: '{0}', "
                            + "LocalName: '{1}', Value: '{2}'",
                            fn, ln, r.GetAttribute(ln, ns));
                    }

                }
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element: <xml> LocalName: 'xml' has 1 attribute(s)
- Attribute: 'version', LocalName: 'version', Value: '1.0'
Element: <root> LocalName: 'root' has 1 attribute(s)
- Attribute: 'xmlns:test', LocalName: 'test', Value: 'http://mysite/prefixes'
Element: <test:subelement> LocalName: 'subelement' has 2 attribute(s)
- Attribute: 'test:first', LocalName: 'first', Value: 'one'
- Attribute: 'test:second', LocalName: 'second', Value: 'two'


Press Enter to continue
```

XmlReader.Name Property

```
[ILASM]
.property string Name { public hidebysig virtual abstract specialname string
get_Name() }
[C#]
public abstract string Name { get; }
```

Summary

Gets the qualified name of the current node.

Property Value

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

Behaviors

The qualified name is equivalent to the `LocalName` prefixed with `Prefix` and the ':' character. For example, `Name` is "bk:book" for the element `<bk:book>`.

The name returned is dependent on the `NodeType` of the node. The following node types return the listed values. All other node types return an empty 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".

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            Console.WriteLine("NodeType: {0} has Name: '{1}'",
                r.NodeType, r.Name);
            if (r.HasAttributes)
                while (r.MoveToNextAttribute())
                    Console.WriteLine("- Attribute Name: '{0}'",  r.Name);
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType: XmlDeclaration has Name: 'xml'
- Attribute Name: 'version'
NodeType: Element has Name: 'root'
NodeType: Element has Name: 'subelement'
- Attribute Name: 'first'
- Attribute Name: 'second'
NodeType: Text has Name: ''
NodeType: EndElement has Name: 'subelement'
NodeType: EndElement has Name: 'root'
NodeType: Whitespace has Name: ''


Press Enter to continue
```

XmlReader.NamespaceURI Property

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

Summary

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

Property Value

A `String` containing the namespace URI of the current node or, if no namespace URI is associated with the current node, `String.Empty`.

Behaviors

This property is relevant to `Element` and `Attribute` nodes only.

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

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.XmlDeclaration
                || r.NodeType == XmlNodeType.Element)
            {
                Console.WriteLine("Element <{0}> has NamespaceURI: '{1}'",
                    r.Name, r.NamespaceURI);
```

```
                if (r.AttributeCount > 0)
                {
                    string fn = String.Empty;
                    string ln = String.Empty;
                    string ns = String.Empty;
                    while (r.MoveToNextAttribute())
                    {
                        fn = r.Name;
                        ln = r.LocalName;
                        ns = r.NamespaceURI;
                        Console.WriteLine("- Attribute '{0}' "
                            + "has NamespaceURI: '{1}'",
                            r.Name, r.NamespaceURI);
                    }
                }
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element <xml> has NamespaceURI: ''
- Attribute 'version' has NamespaceURI: ''
Element <root> has NamespaceURI: ''
- Attribute 'xmlns:test' has NamespaceURI: 'http://www.w3.org/2000/xmlns/'
Element <test:subelement> has NamespaceURI: 'http://mysite/prefixes'
- Attribute 'test:first' has NamespaceURI: 'http://mysite/prefixes'
- Attribute 'test:second' has NamespaceURI: 'http://mysite/prefixes'


Press Enter to continue
```

XmlReader.NameTable Property

```
[ILASM]
.property class System.Xml.XmlNameTable NameTable { public hidebysig virtual
abstract specialname class System.Xml.XmlNameTable get_NameTable() }
[C#]
public abstract XmlNameTable NameTable { get; }
```

Summary

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

Property Value

The `XmlNameTable` used by the current instance.

Behaviors

Element and attribute names, prefixes, and namespaces are stored as individual `String` objects when a document is read.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read());
        NameTable n = (NameTable) r.NameTable;
        n.Add("another");
        Console.WriteLine("String 'child' is in NameTable: {0}",
            n.Get("child") == "child");
        Console.WriteLine("String 'not-there' is in NameTable: {0}",
            n.Get("not-there") == "not-there");
        Console.WriteLine("String 'another' is in NameTable: {0}",
            n.Get("another") == "another");
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
String 'child' is in NameTable: True
String 'not-there' is in NameTable: False
String 'another' is in NameTable: True


Press Enter to continue
```

XmlReader.NodeType Property

```
[ILASM]
.property valuetype System.Xml.XmlNodeType NodeType { public hidebysig virtual
abstract specialname valuetype System.Xml.XmlNodeType get_NodeType() }
[C#]
public abstract XmlNodeType NodeType { get; }
```

Summary

Gets the type of the current node.

Property Value

One of the members of the `XmlNodeType` enumeration representing the type of the current node.

Behaviors

This property does not return the following `XmlNodeType` members: `Document`, `DocumentFragment`, `Entity`, `EndEntity`, and `Notation`.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    static void ShowNodeType(XmlTextReader r)
    {
        Console.WriteLine("Node name: {0}, Node type: XmlNodeType.{1}", r.Name, r.
NodeType);
    }
    public static void Main()
    {
        string s = "sample.xml";
        Console.WriteLine("Reading file '{0}'", s);
        XmlTextReader r = new XmlTextReader(s);
        while(r.Read())
            ShowNodeType(r);
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Reading file 'sample.xml'
Node name: xml, Node type: XmlNodeType.XmlDeclaration
Node name: root, Node type: XmlNodeType.Element
Node name: subelement, Node type: XmlNodeType.Element
Node name: , Node type: XmlNodeType.Text
Node name: subelement, Node type: XmlNodeType.EndElement
Node name: root, Node type: XmlNodeType.EndElement


Press Enter to continue
```

XmlReader.Prefix Property

```
[ILASM]
.property string Prefix { public hidebysig virtual abstract specialname string
get_Prefix() }
[C#]
public abstract string Prefix { get; }
```

Summary

Gets the namespace prefix associated with the current node.

Property Value

A `String` containing the namespace prefix associated with the current node.

Description

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

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.Element)
            {
                Console.WriteLine("Element: <{0}> Prefix: '{1}', "
                    + "LocalName: '{2}'",
                    r.Name, r.LocalName, r.Prefix);
                if (r.AttributeCount > 0)
                {
                    while (r.MoveToNextAttribute())
                    {
                        Console.WriteLine("- Attribute: '{0}', "
                            + "LocalName: '{1}', Prefix: '{2}'",
                            r.Name, r.LocalName, r.Prefix);
                    }
                }
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element: <root> Prefix: 'root', LocalName: ''
- Attribute: 'xmlns:test', LocalName: 'test', Prefix: 'xmlns'
Element: <test:subelement> Prefix: 'subelement', LocalName: 'test'
- Attribute: 'test:first', LocalName: 'first', Prefix: 'test'
- Attribute: 'test:second', LocalName: 'second', Prefix: 'test'


Press Enter to continue
```

XmlReader.QuoteChar Property

```
[ILASM]
.property valuetype System.Char QuoteChar { public hidebysig virtual abstract
specialname valuetype System.Char get_QuoteChar() }
[C#]
public abstract char QuoteChar { get; }
```

Summary

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

Property Value

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

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.Element)
            {
                while (r.MoveToNextAttribute())
                {
                    Console.Write("Attribute Name: {0}, ", r.Name);
                    char q = r.QuoteChar;
                    Console.WriteLine("QuoteChar is {0} and Value is {1}",
                        q, q + r.Value + q);
                }
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Attribute Name: first, QuoteChar is ' and Value is 'one'
Attribute Name: second, QuoteChar is " and Value is "two"


Press Enter to continue
```

XmlReader.ReadState Property

```
[ILASM]
.property valuetype System.Xml.ReadState ReadState { public hidebysig virtual
abstract specialname valuetype System.Xml.ReadState get_ReadState() }
[C#]
public abstract ReadState ReadState { get; }
```

Summary

Gets the read state of the reader.

Property Value

One of the members of the `ReadState` enumeration.

How and When to Override

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

Example

```
using System;
using System.Xml;


public class XmlReaderSample
{
    static void ShowReadState(XmlTextReader r)
    {
        Console.WriteLine("Node {0} {1}, ReadState is {2}", r.NodeType, r.Name,
           r.ReadState);
    }
    public static void Main()
    {
        string s = "sample.xml";
        Console.WriteLine("Reading file '{0}'", s);
        XmlTextReader r = new XmlTextReader(s);
        Console.WriteLine("Before first Read, ReadState is {0}", r.ReadState);
        while(r.Read())
            ShowReadState(r);
        r.Close();
        Console.WriteLine("After Close, ReadState is {0}", r.ReadState);
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Reading file 'sample.xml'
Before first Read, ReadState is Initial
Node XmlDeclaration xml, ReadState is Interactive
Node Element root, ReadState is Interactive
Node Text , ReadState is Interactive
Node EndElement root, ReadState is Interactive
Node Whitespace , ReadState is Interactive
After Close, ReadState is Closed


Press Enter to continue
```

XmlReader.Value Property

```
[ILASM]
.property string Value { public hidebysig virtual abstract specialname string
get_Value() }
[C#]
public abstract string Value { get; }
```

Summary

Gets the text value of the current node.

Property Value

A `String` containing the text value of the current node.

Behaviors

The value returned depends on the `NodeType`. The following table lists node types that have a value to return. All other node types return `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 `xml:space = "preserve"`.
Text	The content of the text node.
Whitespace	The white space between markup.
XmlDeclaration	The content of the declaration.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    private static void ReadXmlDocument(XmlReader r)
    {
        Console.WriteLine();
        Console.WriteLine("Reading XML document...");
        while (r.Read())
        {
            Console.Write("NodeType: {0} ({1})", r.NodeType, r.Name);
            if (r.HasValue)
                Console.Write(" Value: '{0}'", r.Value);
            Console.WriteLine();
            if (r.HasAttributes)
            {
                while (r.MoveToNextAttribute())
                    Console.WriteLine("- Attribute '{0}', Value: '{1}'",
```

```
                        r.Name, r.Value);
            }
        }
        r.Close();
    }
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        ReadXmlDocument(r);
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Reading XML document...
NodeType: XmlDeclaration (xml) Value: 'version="1.0"'
- Attribute 'version', Value: '1.0'
NodeType: Element (root)
- Attribute 'sample', Value: 'value'
NodeType: Element (subelement)
NodeType: Text () Value: 'sample'
NodeType: EndElement (subelement)
NodeType: EndElement (root)


Press Enter to continue
```

XmlReader.XmlLang Property

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

Summary

Gets the current `xml:lang` scope.

Property Value

A `String` containing the current `xml:lang` scope.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
```

```
        r.WhitespaceHandling = WhitespaceHandling.None;
        r.MoveToContent();
        r.Read();
        Console.WriteLine("XmlLang value is '{0}'", r.XmlLang);
        r.Close();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
XmlLang value is 'en-gb'


Press Enter to continue
```

XmlReader.XmlSpace Property

```
[ILASM]
.property valuetype System.Xml.XmlSpace XmlSpace { public hidebysig virtual
abstract specialname valuetype System.Xml.XmlSpace get_XmlSpace() }
[C#]
public abstract XmlSpace XmlSpace { get; }
```

Summary

Gets the current `xml:space` scope.

Property Value

One of the members of the `XmlSpace` enumeration. If no `xml:space` scope exists, this property defaults to `XmlSpace.None`.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    static void ReadXmlDocument(string s, XmlSpace wspace)
    {
        XmlTextReader r = new XmlTextReader(s);
        while(r.Read())
            if(r.NodeType == XmlNodeType.Element)
                Console.WriteLine("Element <{0}> in '{1}',"
                    + "XmlSpace is {2}",
                    r.Name, s, r.XmlSpace);
        r.Close();
    }
    public static void Main()
    {
        ReadXmlDocument("default.xml", XmlSpace.Default);
        ReadXmlDocument("none.xml", XmlSpace.None);
```

```
        ReadXmlDocument("preserve.xml", XmlSpace.Preserve);
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element <root> in 'default.xml',XmlSpace is Default
Element <subelement> in 'default.xml',XmlSpace is Default
Element <root> in 'none.xml',XmlSpace is None
Element <subelement> in 'none.xml',XmlSpace is None
Element <root> in 'preserve.xml',XmlSpace is Preserve
Element <subelement> in 'preserve.xml',XmlSpace is Preserve


Press Enter to continue
```

XmlReader.Item Property

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

Summary

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

Parameters

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

Property Value

A `String` containing the value of the attribute.

Behaviors

This property does not move the reader.

How and When to Override

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

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.XmlDeclaration
                || r.NodeType == XmlNodeType.Element)
            {
                Console.WriteLine("Element <{0}> has {1} attribute(s)",
                    r.Name, r.AttributeCount);
                for (int i = 0; i < r.AttributeCount; i++)
                    Console.WriteLine("XmlReader[{0}] Value: '{1}'",
                        i, r[i]);
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element <xml> has 1 attribute(s)
XmlReader[0] Value: '1.0'
Element <root> has 0 attribute(s)
Element <subelement> has 2 attribute(s)
XmlReader[0] Value: 'one'
XmlReader[1] Value: 'two'


Press Enter to continue
```

XmlReader.Item Property

```
[ILASM]
.property string Item[string name] { public hidebysig virtual abstract
specialname string get_Item(string name) }
[C#]
public abstract string this[string name] { get; }
```

Summary

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

Parameters

Parameter	Description
name	A `System.String` specifying the qualified name of the attribute.

Property Value

A `String` containing the value of the specified attribute, or `null` if the attribute is not found.

Behaviors

This property does not move the reader.

If the reader is positioned on a `DocumentType` node, this method can be used to get the PUBLIC and SYSTEM literals.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.XmlDeclaration
                || r.NodeType == XmlNodeType.Element)
            {
                Console.WriteLine("Element <{0}> has {1} attribute(s)",
                    r.Name, r.AttributeCount);
                if (r.AttributeCount > 0)
                {
                    string s = String.Empty;
                    while (r.MoveToNextAttribute())
                    {
                        s = r.Name;
                        Console.WriteLine("XmlReader[\"{0}\"] Value: '{1}'",
                            s, r[s]);
                    }
                }
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element <xml> has 1 attribute(s)
XmlReader["version"] Value: '1.0'
Element <root> has 0 attribute(s)
Element <subelement> has 2 attribute(s)
XmlReader["first"] Value: 'one'
XmlReader["second"] Value: 'two'


Press Enter to continue
```

XmlReader.Item Property

```
[ILASM]
.property string Item[string name, string namespaceURI] { public hidebysig
virtual abstract specialname string get_Item(string name, string namespaceURI) }
[C#]
public abstract string this[string name, string namespaceURI] { get; }
```

Summary

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

Parameters

Parameter	Description
name	A `System.String` specifying the local name of the attribute.
namespaceURI	A `System.String` specifying the namespace URI of the attribute.

Property Value

A `String` containing the value of the specified attribute, or `null` if the attribute is not found.

Behaviors

This property does not move the reader.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.XmlDeclaration
                || r.NodeType == XmlNodeType.Element)
            {
                Console.WriteLine("Element <{0}> has {1} attribute(s)",
                    r.Name, r.AttributeCount);
                if (r.AttributeCount > 0)
                {
                    string fn = String.Empty;
                    string ln = String.Empty;
                    string ns = String.Empty;
                    while (r.MoveToNextAttribute())
                    {
                        fn = r.Name;
                        ln = r.LocalName;
                        ns = r.NamespaceURI;
                        Console.WriteLine("XmlReader[\"{0}\", \"{1}\"] Value: '{2}'",
```

```
                            ln, ns, r[ln, ns]);
                    }
                }
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element <xml> has 1 attribute(s)
XmlReader["version", ""] Value: '1.0'
Element <root> has 1 attribute(s)
XmlReader["test", "http://www.w3.org/2000/xmlns/"] Value: 'http://mysite'
Element <test:subelement> has 2 attribute(s)
XmlReader["first", "http://mysite"] Value: 'one'
XmlReader["second", "http://mysite"] Value: 'two'


Press Enter to continue
```

XmlReader.Close() Method

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

Summary

Changes the `ReadState` to `ReadState.Closed`.

Behaviors

This method releases any resources allocated by the current instance, changes the `ReadState` to `Closed`, and calls the `Close` method of any underlying `System.IO.Stream` or `System.IO.TextReader` instance.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
            Console.WriteLine("NodeType: {0} ({1})",
                r.NodeType, r.Name);
        r.Close();
```

```
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType: XmlDeclaration (xml)
NodeType: Element (root)
NodeType: Element (subelement)
NodeType: Text ()
NodeType: EndElement (subelement)
NodeType: EndElement (root)


Press Enter to continue
```

XmlReader.GetAttribute(System.Int32) Method

```
[ILASM]
.method public hidebysig virtual abstract string GetAttribute(int32 i)
[C#]
public abstract string GetAttribute(int i)
```

Summary

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

Parameters

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

Return Value

A `String` containing the value of the specified attribute.

Behaviors

This method does not move the reader.

How and When to Override

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

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.XmlDeclaration
                || r.NodeType == XmlNodeType.Element)
            {
                Console.WriteLine("Element <{0}> has {1} attribute(s)",
                    r.Name, r.AttributeCount);
                for (int i = 0; i < r.AttributeCount; i++)
                    Console.WriteLine("- Attribute {0} has value '{1}'",
                        i.ToString(), r.GetAttribute(i));
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element <xml> has 1 attribute(s)
- Attribute 0 has value '1.0'
Element <root> has 0 attribute(s)
Element <subelement> has 2 attribute(s)
- Attribute 0 has value 'one'
- Attribute 1 has value 'two'


Press Enter to continue
```

XmlReader.GetAttribute(System.String) Method

```
[ILASM]
.method public hidebysig virtual abstract string GetAttribute(string name)
[C#]
public abstract string GetAttribute(string name)
```

Summary

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

Parameters

Parameter	Description
name	A `System.String` specifying the qualified name of the attribute.

Return Value

A `String` containing the value of the specified attribute, or `null` if the attribute is not found.

Behaviors

This method does not move the reader.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.XmlDeclaration
                || r.NodeType == XmlNodeType.Element)
            {
                Console.WriteLine("Element <{0}> has {1} attribute(s)",
                    r.Name, r.AttributeCount);
                if (r.AttributeCount > 0)
                {
                    string s = String.Empty;
                    while (r.MoveToNextAttribute())
                    {
                        s = r.Name;
                        Console.WriteLine("- Attribute '{0}' has value '{1}'",
                            s, r.GetAttribute(s));
                    }

                }
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element <xml> has 1 attribute(s)
- Attribute 'version' has value '1.0'
Element <root> has 0 attribute(s)
Element <subelement> has 2 attribute(s)
- Attribute 'first' has value 'one'
- Attribute 'second' has value 'two'


Press Enter to continue
```

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

```
[ILASM]
.method public hidebysig virtual abstract string GetAttribute(string name,
string namespaceURI)
[C#]
public abstract string GetAttribute(string name, string namespaceURI)
```

Summary

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

Parameters

Parameter	Description
name	A `System.String` specifying the local name of the attribute.
namespaceURI	A `System.String` specifying the namespace URI of the attribute.

Return Value

A `String` containing the value of the specified attribute, or `null` if the attribute is not found.

Behaviors

This method does not move the reader.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.NodeType == XmlNodeType.XmlDeclaration
                || r.NodeType == XmlNodeType.Element)
            {
                Console.WriteLine("Element <{0}> has {1} attribute(s)",
                    r.Name, r.AttributeCount);
                if (r.AttributeCount > 0)
                {
                    string fn = String.Empty;
                    string ln = String.Empty;
                    string ns = String.Empty;
                    while (r.MoveToNextAttribute())
                    {
                        fn = r.Name;
                        ln = r.LocalName;
                        ns = r.NamespaceURI;
                        Console.WriteLine("- Attribute: '{0}', "
```

```
                            + "LocalName: '{1}', Value: '{2}'",
                            fn, ln, r.GetAttribute(ln, ns));
                    }

                }
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element <xml> has 1 attribute(s)
- Attribute: 'version', LocalName: 'version', Value: '1.0'
Element <root> has 1 attribute(s)
- Attribute: 'xmlns:test', LocalName: 'test', Value: 'http://mysite/prefixes'
Element <test:subelement> has 2 attribute(s)
- Attribute: 'test:first', LocalName: 'first', Value: 'one'
- Attribute: 'test:second', LocalName: 'second', Value: 'two'


Press Enter to continue
```

XmlReader.IsName(System.String) Method

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

Summary

Determines whether the specified string is a valid XML name.

Parameters

Parameter	Description
str	A `System.String` specifying the name to validate.

Return Value

A `System.Boolean` where `true` indicates the name is valid; otherwise, `false`.

Description

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

Example

```
using System;
using System.Xml;


public class XmlReaderSample
{
    public static void Main()
    {
        string s = "myelement";
        Console.WriteLine("XmlReader.IsName '{0}' = {1}",
            s, XmlReader.IsName(s));
        s = "2ndelement";
        Console.WriteLine("XmlReader.IsName '{0}' = {1}",
            s, XmlReader.IsName(s));
        s = "test:element";
        Console.WriteLine("XmlReader.IsName '{0}' = {1}",
            s, XmlReader.IsName(s));
        s = "al&dave";
        Console.WriteLine("XmlReader.IsName '{0}' = {1}",
            s, XmlReader.IsName(s));
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
XmlReader.IsName 'myelement' = True
XmlReader.IsName '2ndelement' = False
XmlReader.IsName 'test:element' = True
XmlReader.IsName 'al&dave' = False


Press Enter to continue
```

XmlReader.IsNameToken(System.String) Method

```
[ILASM]
.method public hidebysig static bool IsNameToken(string str)
[C#]
public static bool IsNameToken(string str)
```

Summary

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

Parameters

Parameter	Description
str	A `System.String` specifying the name to validate.

Return Value

A `System.Boolean` where `true` indicates the name is valid; otherwise `false`.

Description

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        string s = "mytoken";
        Console.WriteLine("XmlReader.IsNameToken '{0}' = {1}",
            s, XmlReader.IsNameToken(s));
        s = "2ndtoken";
        Console.WriteLine("XmlReader.IsNameToken '{0}' = {1}",
            s, XmlReader.IsNameToken(s));
        s = "test:token";
        Console.WriteLine("XmlReader.IsNameToken '{0}' = {1}",
            s, XmlReader.IsNameToken(s));
        s = "al&dave";
        Console.WriteLine("XmlReader.IsNameToken '{0}' = {1}",
            s, XmlReader.IsNameToken(s));
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
XmlReader.IsNameToken 'mytoken' = True
XmlReader.IsNameToken '2ndtoken' = True
XmlReader.IsNameToken 'test:token' = True
XmlReader.IsNameToken 'al&dave' = False


Press Enter to continue
```

XmlReader.IsStartElement() Method

```
[ILASM]
.method public hidebysig virtual bool IsStartElement()
[C#]
public virtual bool IsStartElement()
```

Summary

Determines if a node containing content is an `Element` node.

Return Value

A `System.Boolean` where `true` indicates the node is an `Element` node; `false` otherwise.

Default

This method calls the `MoveToContent` method, which determines whether the current node can contain content and, if not, moves the reader to the next content node or the end of the input

stream. When the reader is positioned on a content node, the node is checked to determine if it is an `Element` node.

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

Use this method to determine whether the node returned by the `MoveToContent` method is an `Element` node.

Exceptions

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

Example

```
using System;
using System.Xml;


public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            Console.WriteLine("NodeType {0} ('{1}'), "
                + "IsStartElement: {2}",
                r.NodeType, r.Name,
                r.IsStartElement());
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType XmlDeclaration ('xml'), IsStartElement: True
NodeType Element ('subelement1'), IsStartElement: True
NodeType Element ('subelement2'), IsStartElement: True
NodeType Text (''), IsStartElement: False
NodeType EndElement ('subelement2'), IsStartElement: False
NodeType EndElement ('root'), IsStartElement: False
NodeType Whitespace (''), IsStartElement: False


Press Enter to continue
```

XmlReader.IsStartElement(System.String) Method

```
[ILASM]
.method public hidebysig virtual bool IsStartElement(string name)
[C#]
public virtual bool IsStartElement(string name)
```

Summary

Determines if a node containing content is an `Element` node with the specified qualified name.

Parameters

Parameter	Description
name	A `System.String` specifying the qualified name of an element.

Return Value

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

Default

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

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

Use this method to determine whether the node returned by the `MoveToContent` method is an `Element` node with the specified name.

Exceptions

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

Example

```
using System;
using System.Xml;


public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
```

```
        while (r.Read())
        {
            Console.WriteLine("NodeType {0} ('{1}'), "
                + "IsStartElement(\"subelement1\"): {2}",
                r.NodeType, r.Name,
                r.IsStartElement("subelement1"));
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType XmlDeclaration ('xml'), IsStartElement("subelement1"): False
NodeType Element ('subelement1'), IsStartElement("subelement1"): True
NodeType Element ('subelement2'), IsStartElement("subelement1"): False
NodeType Text (''), IsStartElement("subelement1"): False
NodeType EndElement ('subelement2'), IsStartElement("subelement1"): False
NodeType EndElement ('root'), IsStartElement("subelement1"): False
NodeType Whitespace (''), IsStartElement("subelement1"): False


Press Enter to continue
```

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

```
[ILASM]
.method public hidebysig virtual bool IsStartElement(string localname, string ns)
[C#]
public virtual bool IsStartElement(string localname, string ns)
```

Summary

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

Parameters

Parameter	Description
localname	A `System.String` specifying the local name of an element.
ns	A `System.String` specifying the namespace URI associated with the element.

Return Value

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

Default

This method calls the `MoveToContent` method, which determines whether the current node can contain content and, if not, moves the reader to the next content node or the end of the input stream. When the reader is positioned on a content node, the node is checked to determine if it is an `Element` node with `LocalName` and `NamespaceURI` properties equal to *localname* and *ns*, respectively.

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

Use this method to determine whether the node returned by the `MoveToContent` method is an `Element` node with the specified local name and namespace URI.

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            Console.WriteLine();
            Console.WriteLine("NodeType {0} ('{1}'), ",r.NodeType, r.Name);
            Console.WriteLine("- IsStartElement(\"subelement1\", \"http://test\"): {0}",
                r.IsStartElement("subelement1", "http://test").ToString());
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType XmlDeclaration ('xml'),
- IsStartElement("subelement1", "http://test"): False

NodeType Element ('dummy:subelement1'),
- IsStartElement("subelement1", "http://test"): False

NodeType Element ('test:subelement1'),
- IsStartElement("subelement1", "http://test"): True

NodeType Text (''),
- IsStartElement("subelement1", "http://test"): False

NodeType EndElement ('test:subelement1'),
- IsStartElement("subelement1", "http://test"): False

NodeType EndElement ('root'),
- IsStartElement("subelement1", "http://test"): False

NodeType Whitespace (''),
- IsStartElement("subelement1", "http://test"): False


Press Enter to continue
```

XmlReader.LookupNamespace(System.String) Method

```
[ILASM]
.method public hidebysig virtual abstract string LookupNamespace(string prefix)
[C#]
public abstract string LookupNamespace(string prefix)
```

Summary

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

Parameters

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

Return Value

A `String` containing the namespace URI to which the prefix maps. If *prefix* is not in `NameTable` or no matching namespace is found, `null` is returned.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.IsStartElement("test:subelement"))
                Console.WriteLine("Element <{0}>, Prefix: '{1}', "
                    + "Namespace: '{2}'",
                    r.Name, r.Prefix,
                    r.LookupNamespace(r.Prefix));
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element <test:subelement>, Prefix: 'test', Namespace: 'http://mysite/prefixes'


Press Enter to continue
```

XmlReader.MoveToAttribute(System.Int32) Method

```
[ILASM]
.method public hidebysig virtual abstract void MoveToAttribute(int32 i)
[C#]
public abstract void MoveToAttribute(int i)
```

Summary

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

Parameters

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

Behaviors

After calling this method, the `Name`, `NamespaceURI`, and `Prefix` properties reflect the properties of the current attribute.

How and When to Override

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

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.IsStartElement("subelement"))
            {
                Console.WriteLine("Element Name: '{0}'", r.Name);
                r.MoveToAttribute(0);
                Console.WriteLine("Attribute Name: '{0}', Value: '{1}'",
                    r.Name, r.Value);
                r.MoveToAttribute(1);
                Console.WriteLine("Attribute Name: '{0}', Value: '{1}'",
                    r.Name, r.Value);
            }
```

```
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element Name: 'subelement'
Attribute Name: 'first', Value: 'one'
Attribute Name: 'second', Value: 'two'


Press Enter to continue
```

XmlReader.MoveToAttribute(System.String) Method

```
[ILASM]
.method public hidebysig virtual abstract bool MoveToAttribute(string name)
[C#]
public abstract bool MoveToAttribute(string name)
```

Summary

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

Parameters

Parameter	Description
name	A `System.String` specifying the qualified name of the attribute.

Return Value

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

Behaviors

After calling this method, the `Name`, `NamespaceURI`, and `Prefix` properties reflect the properties of current attribute.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
```

```
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.IsStartElement("subelement"))
            {
                Console.WriteLine("Element Name: '{0}'", r.Name);
                r.MoveToAttribute(0);
                Console.WriteLine("Attribute Name: '{0}', Value: '{1}'",
                    r.Name, r.Value);
                r.MoveToAttribute(1);
                Console.WriteLine("Attribute Name: '{0}', Value: '{1}'",
                    r.Name, r.Value);
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element Name: 'subelement'
Attribute Name: 'first', Value: 'one'
Attribute Name: 'second', Value: 'two'


Press Enter to continue
```

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

```
[ILASM]
.method public hidebysig virtual abstract bool MoveToAttribute(string name,
string ns)
[C#]
public abstract bool MoveToAttribute(string name, string ns)
```

Summary

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

Parameters

Parameter	Description
name	A `System.String` specifying the local name of the attribute.
ns	A `System.String` specifying the namespace URI of the attribute.

Return Value

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

Behaviors

After calling this method, the `Name`, `NamespaceURI`, and `Prefix` properties reflect the properties of current attribute.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.IsStartElement("subelement", "http://mysite/prefixes"))
            {
                Console.WriteLine("Element Name: '{0}'", r.Name);
                r.MoveToAttribute("first", "http://mysite/prefixes");
                Console.WriteLine("Attribute Name: '{0}', Value: '{1}'",
                    r.Name, r.Value);
                r.MoveToAttribute("second", "http://mysite/prefixes");
                Console.WriteLine("Attribute Name: '{0}', Value: '{1}'",
                    r.Name, r.Value);
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Element Name: 'test:subelement'
Attribute Name: 'test:first', Value: 'one'
Attribute Name: 'test:second', Value: 'two'


Press Enter to continue
```

XmlReader.MoveToContent() Method

```
[ILASM]
.method public hidebysig virtual valuetype System.Xml.XmlNodeType MoveToContent()
[C#]
public virtual XmlNodeType MoveToContent()
```

Summary

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

Return Value

The `XmlNodeType` of the content node, or `XmlNodeType.None` if the position of the reader has reached the end of the input stream.

Description

[*Note:* The following members of `XmlNodeType` can contain content: `Attribute`, `CDATA`, `Element`, `EndElement`, `EntityReference`, `EndEntity`, and `Text`.]

Default

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

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

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

Exceptions

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

Example

```
using System;
using System.Xml;


public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.MoveToContent();
        Console.WriteLine("NodeType: {0}, Name: '{1}'",
            r.NodeType, r.Name);
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType: Element, Name: 'root'


Press Enter to continue
```

XmlReader.MoveToElement() Method

```
[ILASM]
.method public hidebysig virtual abstract bool MoveToElement()
[C#]
public abstract bool MoveToElement()
```

Summary

Moves the position of the current instance to the node that contains the current `Attribute` node.

Return Value

A `System.Boolean` where `true` indicates the position of the reader was moved; `false` indicates the reader was not positioned on an `Attribute` node and therefore the position of the reader was not moved.

Description

[*Note:* The `DocumentType`, `Element`, and `XmlDeclaration` members of `XmlNodeType` can contain attributes.]

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.IsStartElement("subelement"))
            {
                Console.WriteLine("Starting at Element '{0}'", r.Name);
                r.MoveToAttribute(1);
                Console.WriteLine("Moved to Attribute '{0}'", r.Name);
                r.MoveToElement();
                Console.WriteLine("Moved back to Element '{0}'", r.Name);
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Starting at Element 'subelement'
Moved to Attribute 'second'
Moved back to Element 'subelement'


Press Enter to continue
```

XmlReader.MoveToFirstAttribute() Method

```
[ILASM]
.method public hidebysig virtual abstract bool MoveToFirstAttribute()
[C#]
public abstract bool MoveToFirstAttribute()
```

Summary

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

Return Value

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

Behaviors

If `AttributeCount` is non-zero, the position of the reader moves to the first attribute; otherwise, the position of the reader does not change.

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.IsStartElement("subelement"))
            {
                r.MoveToFirstAttribute();
                Console.WriteLine("Attribute Name: '{0}', Value: '{1}'",
                    r.Name, r.Value);
                while (r.MoveToNextAttribute())
                {
                    Console.WriteLine("Attribute Name: '{0}', Value: '{1}'",
                        r.Name, r.Value);
                }
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Attribute Name: 'first', Value: 'one'
Attribute Name: 'second', Value: 'two'


Press Enter to continue
```

XmlReader.MoveToNextAttribute() Method

```
[ILASM]
.method public hidebysig virtual abstract bool MoveToNextAttribute()
[C#]
public abstract bool MoveToNextAttribute()
```

Summary

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

Return Value

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

How and When to Override

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.IsStartElement("subelement"))
            {
                r.MoveToFirstAttribute();
                Console.WriteLine("Attribute Name: '{0}', Value: '{1}'",
                    r.Name, r.Value);
                while (r.MoveToNextAttribute())
                {
                    Console.WriteLine("Attribute Name: '{0}', Value: '{1}'",
                        r.Name, r.Value);
                }
            }
        }
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Attribute Name: 'first', Value: 'one'
Attribute Name: 'second', Value: 'two'


Press Enter to continue
```

XmlReader.Read() Method

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

Summary

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

Return Value

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

How and When to Override

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

Usage

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

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
            Console.WriteLine("NodeType: {0} ({1})",
                r.NodeType, r.Name);
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType: XmlDeclaration (xml)
NodeType: Element (root)
NodeType: Element (subelement)
NodeType: Text ()
NodeType: EndElement (subelement)
NodeType: EndElement (root)


Press Enter to continue
```

XmlReader.ReadAttributeValue() Method

```
[ILASM]
.method public hidebysig virtual abstract bool ReadAttributeValue()
[C#]
public abstract bool ReadAttributeValue()
```

Summary

Parses an attribute value into one or more `Text`, `EntityReference`, and `EndEntity` nodes.

Return Value

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

Description

[*Note:* To parse an `EntityReference` node, call the `ResolveEntity` method. After the node is parsed into child nodes, call the `ReadAttributeValue` method again to read the value of the entity.The `Depth` of an attribute value node is one plus the depth of the attribute node. When general entity references are stepped into or out of, the `Depth` increments or decrements by one, respectively.]

How and When to Override

Implementations that cannot expand general entities should return general entities as a single empty (`Value` equals `String.Empty`) `EntityReference` node.

Usage

Use this method after calling `MoveToAttribute` to read through the `Text`, `EntityReference`, or `EndEntity` nodes that make up the attribute value. Call the `ResolveEntity` method to resolve the `EntityReference` nodes.

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        while (r.Read())
        {
            if (r.IsStartElement("book"))
            {
                r.MoveToFirstAttribute();
                Console.WriteLine("Reading Attribute '{0}'...", r.Name);
                while (r.ReadAttributeValue())
                {
                    Console.WriteLine("Item: {0}, Name: '{1}', Value: '{2}'",
                        r.NodeType, r.Name, r.Value);
                }
            }
        }
        r.Close();
        Console.WriteLine();
```

```
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Reading Attribute 'publisher'...
Item: Text, Name: '', Value: 'Published by '
Item: EntityReference, Name: 'pub', Value: ''
Item: Text, Name: '', Value: ' Inc.'


Press Enter to continue
```

XmlReader.ReadElementString() Method

```
[ILASM]
.method public hidebysig virtual string ReadElementString()
[C#]
public virtual string ReadElementString()
```

Summary

Reads the contents of a text-only element.

Return Value

A `String` containing the contents of the element.

Default

This method calls the `MoveToContent` method and, if the returned node is an `Element` node, calls the `ReadString` method to read the contents.

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

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

Exceptions

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.

Example

```
using System;
using System.Xml;

public class XmlReaderSample
```

```
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.MoveToContent();
        r.Read();
        string s = r.ReadElementString();
        Console.WriteLine("Value of Element <test:subelement1> is '{0}'",
            s);
        s = r.ReadElementString();
        Console.WriteLine("Value of Element <test:subelement2> is '{0}'",
            s);
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Value of Element <test:subelement1> is 'subvalue1'
Value of Element <test:subelement2> is 'subvalue2'


Press Enter to continue
```

XmlReader.ReadElementString(System.String) Method

```
[ILASM]
.method public hidebysig virtual string ReadElementString(string name)
[C#]
public virtual string ReadElementString(string name)
```

Summary

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

Parameters

Parameter	Description
name	A `System.String` specifying the qualified name of an element.

Return Value

A `String` containing the contents of the element.

Default

This method calls the `MoveToContent` method and, if the returned node is an `Element` node, compares the `Name` property of the node to *name*. If they are equal, this method calls the `ReadString` method to read the contents of the element.

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

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

Exceptions

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 *name*, the element does not contain a simple text value, or an error occurred while parsing the XML.

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.MoveToContent();
        r.Read();
        string s = r.ReadElementString("subelement1");
        Console.WriteLine("Value of Element <test:subelement1> is '{0}'",
            s);
        s = r.ReadElementString("subelement2");
        Console.WriteLine("Value of Element <test:subelement2> is '{0}'",
            s);
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Value of Element <test:subelement1> is 'subvalue1'
Value of Element <test:subelement2> is 'subvalue2'


Press Enter to continue
```

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

```
[ILASM]
.method public hidebysig virtual string ReadElementString(string localname,
string ns)
[C#]
public virtual string ReadElementString(string localname, string ns)
```

Summary

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

Parameters

Parameter	Description
localname	A `System.String` specifying the local name of an element.
ns	A `System.String` specifying the namespace URI associated with the element.

Return Value

A `String` containing the contents of the element.

Default

This method calls the `MoveToContent` method. If the returned node is an `Element` node, this method compares the `LocalName` and `NamespaceURI` properties of the node to *localname* and *ns*, respectively. If they are equal, this method calls the `ReadString` method to read the contents of the element.

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

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

Exceptions

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 *localname*, or the `System.Xml.XmlReader.NamespaceURI` property of the Element node does not equal *ns*, the element does not contain a simple text value, or an error occurred while parsing the XML.

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        string s = String.Empty;
        r.MoveToContent();
        r.Read();
        s = r.ReadElementString("subelement1", "http://myns/test");
        Console.WriteLine("Value of Element <test:subelement1> is '{0}'",
            s);
        s = r.ReadElementString("subelement2", "http://myns/test");
        Console.WriteLine("Value of Element <test:subelement2> is '{0}'",
            s);
        r.Close();
```

```
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Value of Element <test:subelement1> is 'subvalue1'
Value of Element <test:subelement2> is 'subvalue2'


Press Enter to continue
```

XmlReader.ReadEndElement() Method

```
[ILASM]
.method public hidebysig virtual void ReadEndElement()
[C#]
public virtual void ReadEndElement()
```

Summary

Reads an `EndElement` node and advances the reader to the next node.

Default

This method calls the `MoveToContent` method, which determines whether the current node can contain content and, if not, moves the reader to the next content node or the end of the input stream. The node the reader ends up positioned on is checked to determine if it is an `EndElement` node. If so, the node is read and the reader is moved to the next node.

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

Use this method to read an `EndElement` node and advance the reader to the next node.

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
```

```
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.MoveToContent();
        r.Read();
        r.ReadStartElement("subelement1");
        Console.WriteLine("Read Start of Element");
        String s = r.ReadString();
        Console.WriteLine("String Value of Element '{0}' is '{1}'", r.Name, s);
        r.ReadEndElement();
        Console.WriteLine("Read End of Element");
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Read Start of Element
String Value of Element 'subelement1' is 'subvalue1'
Read End of Element


Press Enter to continue
```

XmlReader.ReadInnerXml() Method

```
[ILASM]
.method public hidebysig virtual abstract string ReadInnerXml()
[C#]
public virtual string ReadInnerXml()
```

Summary

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

Return Value

A `String` containing the XML content, or `String.Empty` if the current node is neither an element nor attribute, or has no child nodes.

Behaviors

The current node and corresponding end node are not returned.

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

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

[*Note:* For a comparison between this method and the `ReadOuterXml` method, see `ReadInnerXml`.]

How and When to Override

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

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.WhitespaceHandling = WhitespaceHandling.None;
        r.MoveToContent();
        r.Read();
        String s = r.ReadInnerXml();
        Console.WriteLine("InnerXml of Element 'subelement' is '{0}'", s);
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
InnerXml of Element 'subelement' is 'subvalue'


Press Enter to continue
```

XmlReader.ReadOuterXml() Method

```
[ILASM]
.method public hidebysig virtual abstract string ReadOuterXml()
[C#]
public virtual string ReadOuterXml()
```

Summary

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

Return Value

A `String` containing the XML content, or `String.Empty` if the current node is neither an element nor attribute.

Behaviors

The current node and corresponding end node are returned.

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

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

[*Note:* For a comparison between this method and the `ReadOuterXml` method, see `ReadInnerXml`.]

How and When to Override

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

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.WhitespaceHandling = WhitespaceHandling.None;
        r.MoveToContent();
        r.Read();
        String s = r.ReadOuterXml();
        Console.WriteLine("OuterXml of Element 'subelement' is:");
        Console.WriteLine(s);
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
OuterXml of Element 'subelement' is:
<subelement myattr="attrvalue">subvalue</subelement>


Press Enter to continue
```

XmlReader.ReadStartElement() Method

```
[ILASM]
.method public hidebysig virtual void ReadStartElement()
[C#]
public virtual void ReadStartElement()
```

Summary

Reads an `Element` node and advances the reader to the next node.

Default

This method calls the `MoveToContent` method, which determines whether the current node can contain content and, if not, moves the reader to the next content node or the end of the input stream. The node the reader ends up positioned on is checked to determine if it is an `Element` node. If so, the node is read and the reader is moved to the next node.

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

Use this method to read an `Element` node and advance the reader to the next node.

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.MoveToContent();
        r.Read();
        r.ReadStartElement();
        Console.WriteLine("Read Start of Element");
        String s = r.ReadString();
        Console.WriteLine("String Value of Element '{0}' is '{1}'", r.Name, s);
        r.ReadEndElement();
        Console.WriteLine("Read End of Element");
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Read Start of Element
String Value of Element 'subelement1' is 'subvalue1'
Read End of Element


Press Enter to continue
```

XmlReader.ReadStartElement(System.String) Method

```
[ILASM]
.method public hidebysig virtual void ReadStartElement(string name)
[C#]
public virtual void ReadStartElement(string name)
```

Summary

Reads an `Element` node with the specified qualified name and advances the reader to the next node.

Parameters

Parameter	Description
name	A `System.String` specifying the qualified name of an element.

Default

This method calls the `MoveToContent` method and, if the returned node is an `Element` node, compares the `Name` property of the node to *name*. If they are equal, this method calls the `Read` method to read the element and move to the next node.

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

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

Exceptions

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 *name*, or an error occurred while parsing the XML.

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.MoveToContent();
        r.Read();
        r.ReadStartElement("subelement1");
        Console.WriteLine("Reading Start of Element");
        String s = r.ReadString();
        Console.WriteLine("String Value of Element '{0}' is '{1}'", r.Name, s);
        r.ReadEndElement();
        Console.WriteLine("Reading End of Element");
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Reading Start of Element
String Value of Element 'subelement1' is 'subvalue1'
Reading End of Element


Press Enter to continue
```

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

```
[ILASM]
.method public hidebysig virtual void ReadStartElement(string localname, string
ns)
[C#]
public virtual void ReadStartElement(string localname, string ns)
```

Summary

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

Parameters

Parameter	Description
localname	A `System.String` specifying the local name of an element.
ns	A `System.String` specifying the namespace URI associated with the element.

Default

This method calls the `MoveToContent` method. If the returned node is an `Element` node, this method compares the `LocalName` and `NamespaceURI` properties of the node to *localname* and *ns*, respectively. If they are equal, this method calls the `Read` method to read the element and move to the next node.

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

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

Exceptions

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 *localname*, the `System.Xml.XmlReader.NamespaceURI` property of the Element node does not equal *ns*, or an error occurred while parsing the XML.

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
```

```
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.MoveToContent();
        r.Read();
        r.ReadStartElement("subelement1", "http://myns/test");
        Console.WriteLine("Reading Start of Element");
        string s = r.ReadString();
        Console.WriteLine("String Value of Element '{0}'",r.LocalName);
        Console.WriteLine("in namespace '{0}'", r.NamespaceURI);
        Console.WriteLine("is '{0}'", s);
        r.ReadEndElement();
        Console.WriteLine("Reading End of Element");
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Reading Start of Element
String Value of Element 'subelement1'
in namespace 'http://myns/test'
is 'subvalue1'
Reading End of Element


Press Enter to continue
```

XmlReader.ReadString() Method

```
[ILASM]
.method public hidebysig virtual abstract string ReadString()
[C#]
public virtual string ReadString()
```

Summary

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

Return Value

A `String` containing the contents of the `Element` or `Text` node, or `String.Empty` if the reader is positioned on any other type of node.

Behaviors

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

If positioned on an element `Text` node, this method performs the same concatenation from the `Text` node to the element end tag. If the reader is positioned on an attribute `Text` node, this method has the same functionality as if the reader were positioned on the element start tag.

How and When to Override

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

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.MoveToContent();
        r.Read();
        r.ReadStartElement();
        Console.WriteLine("Read Start of Element");
        String s = r.ReadString();
        Console.WriteLine("String Value of Element '{0}' is '{1}'", r.Name, s);
        r.ReadEndElement();
        Console.WriteLine("Read End of Element");
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Read Start of Element
String Value of Element 'subelement1' is 'subvalue1'
Read End of Element


Press Enter to continue
```

XmlReader.ResolveEntity() Method

```
[ILASM]
.method public hidebysig virtual abstract void ResolveEntity()
[C#]
public abstract void ResolveEntity()
```

Summary

Resolves the entity reference for `EntityReference` nodes.

Behaviors

This method parses the entity reference into child nodes. When the parsing is finished a new `XmlNodeType.EndEntity` node is placed in the stream to close the `EntityReference` scope.

To step into the entity after this method has been called, call the `ReadAttributeValue` method if the entity is part of an attribute value, or the `Read` method if the entity is part of element content.

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

How and When to Override

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

This method is required to throw an exception for implementations that do not support schema or DTD information. In this case, the `CanResolveEntity` property is required to return `false`.

Usage

Use this method to resolve the entity reference for `EntityReference` nodes. Before calling this method, determine whether the reader can resolve an entity by checking the `CanResolveEntity` property.

Exceptions

Exception	Condition
`System.InvalidOperationException`	The reader is not positioned on a `System.Xml.XmlNodeType.EntityReference` node.

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        XmlValidatingReader vr = new XmlValidatingReader(r);
        vr.ValidationType = ValidationType.None;
        vr.EntityHandling = EntityHandling.ExpandCharEntities;
        while (vr.Read())
        {
            Console.Write("NodeType: {0}, Name: '{1}'", vr.NodeType, vr.Name);
            if (vr.HasValue)
                Console.Write(", Value: '{0}'", vr.Value);
            Console.WriteLine();
            if (vr.NodeType == XmlNodeType.EntityReference)
            {
                Console.WriteLine("- CanResolveEntity: {0}",
                    vr.CanResolveEntity);
                vr.ResolveEntity();
                Console.WriteLine("- Executed ResolveEntity() method, "
                    + " value inserted as next text node");
            }
        }
        vr.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
NodeType: XmlDeclaration, Name: 'xml', Value: 'version="1.0"'
NodeType: DocumentType, Name: 'sample', Value: ''
NodeType: Element, Name: 'sample'
NodeType: Element, Name: 'book'
NodeType: Text, Name: '', Value: 'Microsoft .NET by '
NodeType: EntityReference, Name: 'pub'
- CanResolveEntity: True
- Executed ResolveEntity() method,  value inserted as next text node
NodeType: Text, Name: '', Value: 'Addison Wesley'
NodeType: EndEntity, Name: 'pub'
NodeType: EndElement, Name: 'book'
NodeType: EndElement, Name: 'sample'
NodeType: Whitespace, Name: '', Value: '
'


Press Enter to continue
```

XmlReader.Skip() Method

```
[ILASM]
.method public hidebysig virtual void Skip()
[C#]
public virtual void Skip()
```

Summary

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

Behaviors

If the reader is positioned on a non-empty `Element` node (`IsEmptyElement` equals `false`), the position of the reader is moved to the node following the corresponding `EndElement` node. The properties of the nodes that are skipped over are not exposed. If the reader is positioned on any other node type, the position of the reader is moved to the next node, in this case behaving like the `Read` method.

Default

This method calls the `MoveToElement` method before skipping to the next node.

How and When to Override

Override this method to customize the behavior of this method in types derived from the `XmlReader` class.

Usage

Use this method to skip over the current node.

Exceptions

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

Example

```
using System;
using System.Xml;

public class XmlReaderSample
{
    public static void Main()
    {
        XmlTextReader r = new XmlTextReader("sample.xml");
        r.WhitespaceHandling = WhitespaceHandling.None;
        r.MoveToContent();
        r.Read();
        Console.WriteLine("Positioned on Element <{0}>", r.Name);
        r.Skip();
        Console.WriteLine("Positioned on Element <{0}>", r.Name);
        r.Close();
        Console.WriteLine();
        Console.WriteLine();
        Console.WriteLine("Press Enter to continue");
        Console.ReadLine();
    }
}
```

The output is

```
Positioned on Element <element1>
Positioned on Element <element2>


Press Enter to continue
```