Microsoft XML Diff 1.0 and XML Patch 1.0

Microsoft XML Diff 1.0 and XML Patch 1.0

The **XmlDiff** is a class used to compare two XML documents, detecting additions, deletions and other changes between XML documents. **XmlDiff** produces an **XML Diff Language Diffgram** that describes the differences between the two XML documents.

The Microsoft® XML Patch tool enables you to take a **XmlDiff Language Diffgram** produced from the **XmlDiff** class, and apply it against the source document to recreate the modified document.

The benefits of the **XmlDiff** class are:

- Detect equivalence between two XML documents, fragments, or nodes.
- Concisely report the differences between two documents, fragments, or nodes.
- Contains functionality to optionally ignore certain differences, such as comments or processing instructions.

The benefits of the **XmlPatch** class are:

- Apply an **XmlDiff Language Diffgram** to a document, fragment, or node, and re-create the changed document.
- Use as a source control or delta-encoding file. For example, if you have an array of servers that cache information as XML. Changes to one server can be sent over the network to the other servers. Rather than create network traffic collisions by trying to ship the entire cached document between servers, each server can use the XML Diff functionality, then send out the resulting **XDL Diffgram** as the patch.

Note This scenario assumes that the changes are propagated in a constant direction from a single master.

These two classes are contained in the **XmlDiffPatch** namespace.

In This Section

Microsoft XML Diff 1.0

Describes the **XmlDiff** class used to compare two XML documents, fragments, or nodes.

Microsoft XML Patch 1.0

Describes the **XmlPatch** class that applies a **XML Diff Language Diffgram** to an XML document to create a modified document.

Microsoft XML Diff 1.0 and XML Patch 1.0

XML Diff Functionality

XmlDiff is a class used to compare two XML documents. **XmlDiff** detects additions, deletions and other changes between the two XML documents. It also detects structural changes, for example when an XML subtree is moved. **XmlDiff** produces an **XDL Diffgram** written in **XML Diff Language Diffgram (XDL)**. The **XDL Diffgram** describes the differences between the two XML documents. The **XDL Diffgram** can be used to display differences, or to perform a patch operation using XmlPatch Functionality.

XDL Diffgrams contain information regarding additions, changes, or removals of document content, or content being moved from one place in the tree to another. The XDL Diffgram describes the changes by use of the **XML Diff Language (XDL)** and describes in detail what nodes in the source document were changed, and how they were changed.

XmlDiff class performs XML-based comparison of the XML documents as opposed to a common lexical comparison. This means that the **XmlDiff** class:

- Ignores the order attributes.
- Ignores insignificant white spaces.
- Does not distinguish between an empty element `<a/>` and element with no content `<a></a>`.
- Is not affected by the document encoding.

You can also set one or more properties on the **XmlDiff** class to direct what content it includes in the comparison. For example, you can set a flag to ignore comments that are different between the two documents, or ignore processing instructions. For more information, see Setting Options that Affect the Comparison.

There are also different algorithms available to use when comparing the data. The algorithm chosen affects the speed of the comparison, as well as the output in the **XDL Diffgram**. For more information, see Selecting the Algorithm for the Comparison.

See Also

Running Comparisons Between Documents, Fragments, or Nodes | XML Diff Language (Diffgram) | Extended Operations | Example of a Diffgram | XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Running Comparisons Between Documents, Fragments, or Nodes

The **XmlDiff** class can compare two document, two fragments, or two nodes that are stored in files. The **XmlDiff** class can also compare two **XmlReader** objects or two **XmlNode** objects.

For output, the tool returns a **true**/**false** answer; responding **true** when both inputs are the same, **false** if they are not the same. To enhance the **true/false** flag, the tool can also output an **XDL Diffgram** detailing what the differences are between the documents, fragments, or nodes.

There are two constructors for the **XmlDiff** class. One constructor initializes the class using the default option values. The other constructor enables you to customize the options applied when the documents are compared, using the **XmlDiffOptions**. The following code sample shows how to create the **XmlDiff** class with default options:

```
[Visual Basic]
Dim xmlDiff as New XmlDiff()
[C#]
XmlDiff xmlDiff = new XmlDiff();
```

To create the **XmlDiff** class with customized options, set the property value, then create the class with the options as parameters with an `or` keyword between each option:

```
[Visual Basic]
Dim xmlDiff As New XmlDiff(XmlDiffOptions.IgnoreComments Or XmlDiffO
[C#]
XmlDiff xmlDiff = new XmlDiff(XmlDiffOptions.IgnoreComments | XmlDif
```

For more information on the **XmlDiffOptions**, see Setting Options that Affect the Comparison.

There are several overloaded versions of the **Compare** method that allow you to compare documents of different input type.

The following illustration shows the comparison done with two files. The fragment flag, when **true**, indicates that the data is a fragment. The fragment flag has no default value, it must always be specified. If the overloaded **Compare** method that takes an **XmlWriter** as a last parameter is used, then it will additionally output an **XDL Diffgram** to the specified **XmlWriter**.

The following **Compare** methods are used to implement the preceding illustration:

```
[C#]
public bool Compare(string sourceFile, string changedFile, bool bFra
[Visual Basic]

Public Function Compare(sourceFile As String, changedFile As
String, bFragments As Boolean) As Boolean
```

This method compares two XML documents or fragments stored in files, and returns **true** if they are identical; otherwise returns **false**. The following overloaded method is used to additionally output the **XDL Diffgram**.

```
[C#]
public bool Compare(string sourceFile, string changedFile, bool bFra
[Visual Basic]
Public Function Compare(sourceFile As String, changedFile As String,
```

The following illustration shows the comparison done with **XmlReader** objects as input. One of the overloaded **Compare** methods takes an **XmlWriter** as a last parameter. When this method is used, the **Compare** method additionally outputs an **XDL Diffgram** to the specified **XmlWriter**.

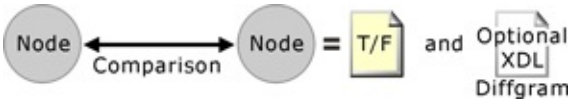
The following **Compare** methods are used to implement the preceding illustration:

```
[C#]
public bool Compare(XmlReader sourceReader, XmlReader changedReader)
[Visual Basic]
Public Function Compare(sourceReader As XmlReader, changedReader As
```

This method compares two XML documents or fragments parsed by the **XmlReader** objects and returns **true** if they are identical; otherwise returns **false**. The following overloaded method is used to additionally output the **XDL Diffgram**.

```
[C#]
public bool Compare(XmlReader sourceReader, XmlReader changedReader,
[Visual Basic]
Public Function Compare(sourceReader As XmlReader, changedReader As
```

The following illustration shows the comparison done with **XmlNode** objects as input. One of the overloaded **Compare** methods takes an **XmlWriter** as a last parameter. When this method is used, the **Compare** method additionally outputs an **XDL Diffgram** to the specified **XmlWriter**.



These are the **Compare** methods used to implement the preceding illustration:

```
[C#]
public bool Compare(XmlNode sourceNode, XmlNode changedNode);
[Visual Basic]
Public Function Compare(sourceNode As XmlNode, changedNode As XmlNod
```

This method compares two **XmlNode** objects and returns **true** if they are identical; otherwise returns **false**. The types of nodes that can be passed into the **Compare** method are any combination of the following:

- XmlDocument
- XmlElement
- XmlText
- XmlCDataSection
- XmlEntityReference
- XmlComment
- XmlDocumentType
- XmlProcessingInstruction

The **Compare** method cannot be used to compare **XmlAttribute**, **XmlEntity**, or **XmlNotation** node types. The following **Compare** overloaded method is used to additionally output the **XDL Diffgram**.

```
[C#]
public bool Compare(XmlNode sourceNode, XmlNode changedNode, XmlWrit
[Visual Basic]
Public Function Compare(sourceNode As XmlNode, changedNode As XmlNod
```

When using the **Compare** method, the results may differ when comparing documents loaded in an **XmlDocument** object, and documents passed in through files or as **XmlReader** objects, if the XML data contains entity references that get expanded by the XmlDocument. The **Patch** method in the XmlPatch class will reject a DOM document loaded into **XmlDocument**. That is, if it contains expanded entity references and if the **XDL Diffgram** supplied for patching has been generated by a **Compare** method that takes document files or **XmlReaders** as input parameters.

See Also

XML Diff Functionality | Selecting the Algorithm for the Comparison | Setting Options that Affect the Comparison | XML Diff Language (Diffgram) | Extended Operations | Example of a Diffgram | XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Setting Options that Affect the Comparison

The **XmlDiff** class allows you to set different options that affect the behavior of the comparison, as well as the resulting **XDL Diffgram**. The following list describes the enumeration properties that affect what items are included for consideration during the comparison. For more information see the XmlDiffOptions Enumeration.

- **IgnoreComments**: Comment nodes are not compared when **true**.
- **IgnorePI**: Processing instructions are not compared when **true**.
- **IgnoreXmlDecl**: The XML declaration is not compared when **true**.
- **IgnorePrefixes**: The prefixes of element and attribute names are not compared when **true**. When this option is selected, then two names that have the same local name and namespace URI, but have a different prefix, are treated as the same names.
- **IgnoreNamespaces**: The namespace URIs of the element and attribute names are not compared when **true**. This option also implies that the prefixes are ignored. When this option is selected, then two names with the same local name, but have a different namespace URI and prefix, are treated as the same names.
- **IgnoreChildOrder**: The order of child nodes of each element is ignored when **true**. When this option is selected, two nodes with the same value that differ only by their position among sibling child nodes are treated as the same nodes.
- **IgnoreWhitespace**: Significant white spaces are not compared when **true**, and all text nodes are normalized by discarding any leading and trailing white space characters (#x9, #x10, #x13, #x20), and by replacing sequences of white space characters by a single space (#x20) character.
- **IgnoreDtd**: The XML DTD is not compared when **true**.

> **Note** The order of attributes is always ignored.

There are several ways to set the options. The following example shows how to set the **IgnorePI** and **IgnoreComments** enumerations by using the **XmlDiff.Options** property explicitly.

```
[Visual Basic]
Imports System
Imports System.Xml
Imports System.IO
Imports Microsoft.XmlDiffPatch

Namespace TestCompare

   Class Class1

      Shared Sub Main()
        Dim myDiff As New XmlDiff()
        Dim diffgramWriter = New XmlTextWriter(New StreamWriter("dif
        myDiff.Options = XmlDiffOptions.IgnorePI Or XmlDiffOptions.I
        Dim bSame As Boolean = myDiff.Compare("Source.xml", "Changed
        diffgramWriter.Close()
      End Sub                                 'Main
   End Class                                  'Class1
End Namespace                                 'TestCompare
[C#]
using System;
using System.Xml;
using System.IO;
using Microsoft.XmlDiffPatch;

namespace TestCompare
{
  class Class1
    {
        static void Main()
          {
            XmlDiff myDiff = new XmlDiff();
            XmlWriter diffgramWriter = new XmlTextWriter( new Stream
            myDiff.Options = XmlDiffOptions.IgnorePI | XmlDiffOption
            bool bSame = myDiff.Compare("Source.xml", "Changed.xml",
            diffgramWriter.Close();
            }
    }
}
```

The following example shows how to run the compare with the **XmlDiffOptions** enumerations of **IgnorePI** and **IgnoreComments** set to **true** without using the **XmlDiff.Options** property. Instead, it sets them as a property of the **XmlDiff** class.

```
[Visual Basic]
```

```
Imports System
Imports System.Xml
Imports System.IO
Imports Microsoft.XmlDiffPatch

Namespace TestCompare

   Class Class1

      Shared Sub Main()
        Dim myDiff As New XmlDiff()
        Dim diffgramWriter = New XmlTextWriter(New StreamWriter("dif
        myDiff.IgnorePI = True
        myDiff.IgnoreComments = True
        Dim bSame As Boolean = myDiff.Compare("Source.xml", "Changed
        diffgramWriter.Close()
      End Sub
   End Class
End Namespace
[C#]
using System;
using System.Xml;
using System.IO;
using Microsoft.XmlDiffPatch;

namespace TestCompare
{
  class Class1
    {
        static void Main()
            {
              XmlDiff myDiff = new XmlDiff();
              XmlWriter diffgramWriter = new XmlTextWriter( new Stre
              myDiff.IgnorePI = true;
              myDiff.IgnoreComments = true;
              bool bSame = myDiff.Compare("Source.xml", "Changed.xml
              diffgramWriter.Close();
            }
    }
}
```

This last example shows the options being set inside the constructor of the **XmlDiff** class, using the **XmlDiffOptions**, as viewed in the first line of code.

```
[Visual Basic]
Imports System
```

```
Imports System.Xml
Imports System.IO
Imports Microsoft.XmlDiffPatch

Namespace TestCompare

   Class Class1

      Shared Sub Main()
        Dim myDiff As New XmlDiff(XmlDiffOptions.IgnorePI Or XmlDiff
        Dim diffgramWriter = New XmlTextWriter(New StreamWriter("dif
        Dim bSame As Boolean = myDiff.Compare("Source.xml", "Changed
        diffgramWriter.Close()
      End Sub
   End Class
End Namespace
[C#]
using System;
using System.Xml;
using System.IO;
using Microsoft.XmlDiffPatch;

namespace TestCompare
{
  class Class1
    {
        static void Main()
            {
XmlDiff myDiff = new XmlDiff( XmlDiffOptions.IgnorePI | XmlDiffOptio
XmlWriter diffgramWriter = new XmlTextWriter( new StreamWriter( "dif
bool bSame = myDiff.Compare("Source.xml", "Changed.xml", false, diff
diffgramWriter.Close();
            }
    }
}
```

For information on running the code samples, see Running XmlDiff and XmlPatch Class Code Samples.

See Also

XML Diff Functionality | Running Comparisons Between Documents, Fragments, or Nodes | Selecting the Algorithm for the Comparison | Limitations | XML Diff Language (Diffgram) | Extended Operations | Example of a Diffgram | XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Selecting the Algorithm for the Comparison

The **XmlDiff** class has implemented two algorithms for comparing XML documents:

- **XmlDiffAlgorithm.Precise:** Based on an algorithm for finding editing distance between trees, also known as Zhang-Shasha algorithm. This algorithm gives very precise results but it may be very slow on large XML documents with many changes.
- **XmlDiffAlgorithm.Fast:** Compares the two XML documents by traversing the XML tree and comparing each node. This algorithm is very fast, but may produce less precise results. For example, it may detect an `xd:add` and `xd:remove` operation on a node instead of a `change` operation. The Fast algorithm is as accurate as the Precise algorithm. For example, the results in the **XDL Diffgram** are a non-empty **XDL Diffgram** whenever the files are different, and the algorithm will produce an empty **XDL Diffgram** when the files are equivalent, with regard to the options set.
- **XmlDiffAlgorithm.Auto**: Chooses the comparison algorithm for you depending on the size and assumed number of changes in the compared documents.

To select the comparison algorithm you want to use, set the **Algorithm** property of the **XmlDiff** class before calling **Compare**. The default value of this property is **XmlDiffAlgorithm.Auto**.

For more information, see XmlDiffAlgorithm Enumeration.

See Also

XML Diff Functionality | Running Comparisons Between Documents, Fragments, or Nodes | Setting Options that Affect the Comparison | Limitations | XML Diff Language (Diffgram) | Extended Operations | Example of a Diffgram | XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Limitations

The current version of the **XmlDiff** class has the following limitations:

- The XML Diff tool uses the DOM content model.
- The XML Diff tool compares DTDs, but it only compares the DTD name, the System ID, Public ID, and the text content of the internal DTD subset.

See Also

XML Diff Functionality | Running Comparisons Between Documents, Fragments, or Nodes | Setting Options that Affect the Comparison | Selecting the Algorithm for the Comparison | XML Diff Language (Diffgram) | Extended Operations | Example of a Diffgram | XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

XML Diff Language (Diffgram)

The **XML Diff Language (XDL)** is a proprietary XML-based language for describing differences between two XML documents. Changes between two XML documents are described in a document called an **XDL Diffgram**.

The root element of the **XDL Diffgram** is `xd:xmldiff`. It contains the version of the **XDL Diffgram** in the `version` attribute, the XDL namespace declaration for the `xd:` prefix, and the XML Diff options selected in the `options` attribute when the **XDL Diffgram** was created. It also contains a `srcDocHash` attribute with a number that is calculated from the source document, and is called a hash value. This number allows checking if an XML document is the correct source document the **XDL Diffgram** was created on. This check is performed by XML Patch tool. Additionally, it contains a `fragments` attribute with value of `yes` or `no`. A value of `yes` indicates that XML fragments were compared. A value of `no` indicates that XML documents were compared. The child nodes of the `xd:xmldiff` root element are elements describing the particular differences between the two XML documents or fragments.

The following example shows an original source document, a changed document, and the **XDL Diffgram** that results when a **Compare** method is run against the **Source XML** and **Target XML** documents, shown below.

Source XML

```
<b>
    <a>Some text 1</a>
    <b>Some text 2</b>
    <c>Some text 3</c>
    <z> Another text
        <fob/>
    </z>
</b>
```

Target XML

```
<b>
    <yy>Some text 1</yy>
    <b>Some text 2</b>
```

```
    <c>Some text 3</c>
    <d>Some text 4</d>
    <z>Changed text</z>
</b>
```

XDL Diffgram

```
<?xml version="1.0" encoding="utf-16" ?>
  <xd:xmldiff version="1.0" srcDocHash="12250381522878755 77" options
    <xd:node match="1">
        <xd:change match="1" name="yy" />
      <xd:node match="3" />
        <xd:add>
        <d>Some text 4</d>
        </xd:add>
    <xd:node match="4">
        <xd:change match="1">Changed text</xd:change>
        <xd:remove match="2" />
    </xd:node>
  </xd:node>
</xd:xmldiff>
```

See Also

XML Diff Functionality | Example of a Diffgram | Path Descriptors | Extended Operations | XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Path Descriptors

The XML Diff Language (XDL) uses path descriptors to identify the nodes in the source XML document. Path descriptors work on DOM data model, and use the node position as the core identifier for the nodes. XDL does not use XPath because the XPath data model differs from DOM.

All path descriptors refer to the original source XML tree before any changes are applied. So when the path descriptor applies to the first node of the source tree, which has been changed to be the third node in the changed tree, the path descriptor for this node is "1" as the node is first in the source document (the source document is a base).

The **XDL Diffgram** shows what nodes have been modified, added, or removed. It uses the words change, add, or remove inside the XDL to describe the change. Next, it describes what node the changes took place on, by giving a path descriptor, and then indicates what the changes were.

Here is the formal grammar used by the path descriptors.

Attribute :== '@'Name

NodePosition ::= Digit+

AttributeList :== Attribute ('|' Attribute)*

PathDescriptor ::= NodeList | AttributeList

NodeList ::= RelativeNodeList | AbsoluteNodeList

RelativeNodeList ::= NodeInterval ('|' NodeInterval)*

AbsoluteNodeList ::= '/' (NodePosition'/')* RelativeNodeList

NodeInterval ::= NodePosition | NodePosition '-' NodePosition

The following table shows some examples of path descriptors, and provides an explanation of the path descriptors.

Example path descriptor	Explanation
3	Means the third child node of the current node.
1-2	Means the first and second child of the current node.
1-2|5	Means the first, second, and fifth child of the current node.
/1/2	Means the second child of the first node at the root level.
1/2/3-6	Means the third, fourth, fifth, and sixth child of the second child of the first node at root level.
@value	Means the attribute named `value`.
@value | @type	Means the attribute named `value` and the attribute named `type`.

The path descriptors are used in the **XDL Diffgram** to identify nodes in the original document. The **XDL Diffgram** contains elements qualified with the `xd:` prefix associated with the namespace http://schemas.microsoft.com/xmltools/2002/xmldiff. The XDL Diffgram element names indicate what action has occurred (for example `add`, `remove`, `change`). The attributes of the XDL Diffgram element and the element content then specifies the details of the operation, for example which nodes are affected, new values of the nodes, and so on.

See Also

XML Diff Functionality | XML Diff Language (Diffgram) | XmlDiff Operation xd:node | XmlDiff Operation xd:add | XmlDiff Operation xd:remove | XmlDiffOperation xd:change | Extended Operations | Example of a Diffgram |XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Operation xd:node

The **XDL Diffgram** contains `xd:node` element if it is describing changes on the child nodes of a node in the source document, or if it is describing the position at which new nodes were added. The `xd:node` element contains the **match** attribute with a path descriptor identifying the referenced node.

Here is an example of what this element may look like:

```
<xd:node match="4"/>
```

Here is an example of `xd:node` element that is used to identify the fourth node of the current source element. The changes on the child nodes of the fourth node are described in the child nodes of the `xd:node` element.

```
<xd:node match="4">
  <xd:change match="1">Changed text</change>
  <xd:remove match="2" />
</xd:node>
```

See Also

XML Diff Functionality | XML Diff Language (Diffgram) | Path Descriptors | XmlDiff Operation xd:add | XmlDiff Operation xd:remove | XmlDiffOperation xd:change | Extended Operations | Example of a Diffgram |XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Operation xd:add

An `xd:add` element occurs when a new node or XML fragment is found in the changed document that was not found in the source document.

Addition of a single node

The following is an example of what an `xd:add` element looks like when new nodes are found:

```
<xd:add type="1" name="Customer">
    <xd:add type="2" name="id">1001</xd:add>
</xd:add>
```

When a single node has been added, the `xd:add` element has the attribute **type** that indicates what type of node was added, using the W3C DOM Node Types enumeration. The `xd:add` element can then have a **name** attribute, which is the local name for the new element, attribute, DTD, or entity reference. The element can also contain an **ns** attribute, indicating the namespace, and a **prefix** attribute indicating the prefix for the new element or attribute name. It can also have the **opid** attribute which contains the ID number of the add operation. If the new node type is a DTD, the `xd:add` element may contain attributes **systemId** and **publicId** specifying the system and public identifiers for the DTD.

The value of the added node, that is, the value of an attribute or xml declaration, is specified as the content of the `xd:add` element.

The preceding example shows that a new **Customer** element was added. The two lines below show an addition of a type 2 node, which is a new attribute, with an attribute name of **id**, and an attribute value of *1001*.

Currently, the `xd:add` element for showing the addition of a single node is used only when a new element, attribute, xml declaration, DTD, or entity reference has been added. For other node types, the `xd:add` element used when adding fragments is shown. For more information, see Adding Fragments and Multiple Nodes.

The following table describes the attributes that the `xd:add` element might have:

Attribute name	Data type	Description
type	unsigned int	Type of the new node according to the values of the DOM node type enumeration (1=element, 2=attribute, and so on).
name	NCName	Local name for the new element or attribute, or a name for the new processing instruction, DTD, or entity reference.
ns	Namespace URI	Namespace for the new element or attribute.
prefix	NCName	Prefix for the new element or attribute.
systemId	string	System identifier for the new DTD node.
publicId	string	Public identifier for the new DTD.
opid	unsigned int	Operation ID.

The following table describes the content that the `xd:add` element might have:

Content type	Description
text value	The value for the new attribute, or the value for the new xml declaration.
CDATA section	Internal subset of the new DTD enclosed in a CDATA section.
*(xd:add)**	Child nodes of the new node. This `xd:add` child nodes are only available for an add of type=1 (element).

See Also

XML Diff Functionality | XML Diff Language (Diffgram) | Path Descriptors | XmlDiff Operation xd:node | XmlDiff Operation xd:remove | XmlDiffOperation

xd:change | Extended Operations | Example of a Diffgram |XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

W3C DOM Node Types

The following table shows the node type and its numeric equivalent. These numeric values are used in the XDL when describing what type of node has been added in an `xd:add` element.

Node type	Value
Element	1
Attribute	2
Text	3
CDATA	4
EntityReference	5
Entity	6
ProcessingInstruction	7
Comment	8
Document	9
DocumentType	10
DocumentFragment	11
Notation	12
Whitespace	13
SignificantWhitespace	14
XmlDeclaration	18

Note Values 15 and 16 are **EndElement** and **EndEntity**, respectively. These are part of the Microsoft DOM but not W3C DOM.

See Also

XML Diff Functionality | XML Diff Language (Diffgram) | Path Descriptors | XmlDiff Operation xd:node | XmlDiff Operation xd:add | Adding Fragments and Multiple Nodes | Copying an Existing Node or Fragment | Extended Operations | XmlDiff Operation xd:remove | XmlDiff Operation xd:change | Example of a Diffgram | XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Adding Fragments and Multiple Nodes

The `xd:add` element is also used to indicate that a fragment or multiple nodes were added. When fragments or multiple nodes are added, there are no additional attributes on the `xd:add` element, except an **opid** attribute that can contain the ID number of the operation when multiple nodes are added. The new fragment is specified as the content of the `xd:add` element between the `<xd:add>` and `</xd:add>` tags. The `xd:add` element that has no attributes, is also used to indicate the addition of single text nodes, CDATA sections, processing instructions, or comments.

Here is an example of what an `xd:add` element looks like when a new fragment is found:

```
<xd:add>
    <Customer>James Brown</Customer>
</xd:add>
```

The following table describes the attributes of the `xd:add` element when a fragment is added.

Attribute name	Data type	Description
opid	unsigned int	Operation ID

If content is found in an `xd:add` element when a fragment is added, then the content represents the XML fragment that was added.

For information on an `xd:add` for single node additions, see XmlDiff Operation xd:add.

See Also

XML Diff Functionality | XML Diff Language (Diffgram) | Path Descriptors | XmlDiff Operation xd:node | XmlDiff Operation xd:add | W3C DOM Node Types | Copying an Existing Node or Fragment | Extended Operations | XmlDiff

Operation xd:remove | XmlDiff Operation xd:change | Example of a Diffgram | XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Copying an Existing Node or Fragment to a New Position in the Tree

The **XmlDiff** tool distinguishes when a node has simply moved to a new position in the tree. When this occurs, the **XDL Diffgram** reports this as a pair of an `add` and a `remove` operation. The `xd:remove` operation is shown where the nodes have been moved from and the `xd:add` operation is where the nodes have been moved to. In this case, XDL has a special `xd:add` element with the **match** attribute specifying from where the nodes have been copied. It may also have a **subtree** attribute specifying whether whole subtree or just the root node has been copied.

For example, the following `xd:add` element indicates that the fifth and sixth child nodes of the first node, at the root level, have been copied at the current position in the source tree.

```
<xd:add match="/1/5-6"/>
```

The following table describes the attributes of the `xd:add` element when a fragment or node is moved to a new position in the tree.

Attribute name	Data type	Description
match	path descriptor	Absolute path descriptor evaluating to a set of nodes in the source XML document. The node set cannot contain attributes.
subtree	bool	An option indicating whether to copy the descendants of the nodes in the selected node set (copy whole subtrees) or not. Default is **yes**. If the

		value is **no**, then the **match** attribute must evaluate to a single node.
opid	unsigned int	Operation ID. The corresponding `xd:remove` operation has the same operation ID.

The following table describes the attributes of the `xd:add` element when a fragment or node is moved to a new position in the tree.

Content	Description
*(xd:add)**	Child nodes of the new node (for element only) when the **subtree** option is **no**. Otherwise the content of the node must be empty.

See Also

XML Diff Functionality | XML Diff Language (Diffgram) | Path Descriptors | XmlDiff Operation xd:node | XmlDiff Operation xd:add | W3C DOM Node Types | Adding Fragments and Multiple Nodes | Extended Operations | XmlDiff Operation xd:remove | XmlDiff Operation xd:change | Example of a Diffgram | XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Operation xd:remove

An `xd:remove` operation occurs when a node or an XML fragment has been removed from the original source document, and no longer appears in the changed document. The following is an example of what a `xd:remove` element might look like:

```
<xd:remove match="3"/ >
```

The preceding shows that the third child of the current element has been removed.

The following table describes the attributes of the `xd:remove` element.

Attribute name	Data Type	Description
match	path descriptor	Relative path descriptor evaluating to a list of nodes or a list of attributes that have been removed.
subtree	bool	An option indicating whether the descendants of the nodes have been removed as well (whole subtrees removed) or not. Default is **yes**. If the value is **no**, then the `match` attribute must evaluate to a single node.
opid	unsigned int	Operation ID.

The following table describes the content of the `xd:remove` element.

Content	Description
(xd:node, xd:add, xd:remove,	Diff operations for the child nodes of the removed node (for element only) when the subtree options is set to **false**.

*xd:change)**	The child nodes of the removed node become child nodes of the parent node of the removed node.

See Also

XML Diff Functionality | XML Diff Language (Diffgram) | Path Descriptors | XmlDiff Operation xd:node | XmlDiff Operation xd:add | XmlDiffOperation xd:change | Extended Operations | Example of a Diffgram |XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Operation xd:change

An `xd:change` operation occurs when a node in the source document has had its name or value modified, or both. The following is an example of what a `change` element might look like:

```
<xd:change match="1" ns="http://New.Namespace.Uri"/>
```

The preceding example shows that the namespace URI of the first child of the current node has changed to "http://New.Namespace.Uri".

The following table describes the attributes of the `xd:change` element.

Attribute name	Data type	Description
match	path descriptor	Relative path descriptor evaluating to a single child node or a single attribute of the current parent element.
name	NCName	Changed local name for the element or attribute, or changed target name for the processing instruction.
ns	Namespace URI	Changed namespace URI for the element or attribute.
prefix	NCName	Changed prefix for the element or attribute.
systemId	string	System identifier for the changed DTD node.
publicId	string	Public identifier for the changed DTD.
opid	unsigned int	Operation ID.

The following table describes the content of the `xd:change` element.

Content	Description
text value	Text value of the new node for a text node or attribute value, or xml declaration.
CDATA section	Changed CDATA section, or the new value of the internal subset of the changed DTD.
processing instruction	Changed processing instruction.
comment	Changed comment.
*(xd:node, xd:add, xd:remove, xd:change)**	Diff operations for the child nodes of the changed node (for element only).

See Also

XML Diff Functionality | XML Diff Language (Diffgram) | Path Descriptors | XmlDiff Operation xd:node | XmlDiff Operation xd:add | XmlDiffOperation xd:remove | Extended Operations | Example of a Diffgram |XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Extended Operations

Diff operations can be grouped together and form an extended operation. Each extended operation has an ID number, and all the related XML Diff operations carry this number in their **opid** attribute. The extended operation also has a descriptor that further describes the operation.

For example, a move operation is considered an extended operation. It consists of two types of operations; one remove operation and one add operation. Both of these operations are indicated in the **XDL Diffgram** in the places where the nodes have been removed and added. The same ID number connects these two operations together with the operation descriptor, and indicates that the remove and add pair are in fact a move operation.

The descriptors of the extended operations appear as elements named `xd:descriptor`. The following table describes the mandatory attributes of all the `xd:descriptor` elements:

Attribute name	Data type	Description
opid	unsigned int	Operation ID.
type	string	Type of the descriptor.

See Also

XML Diff Functionality | Running Comparisons Between Documents, Fragments, or Nodes | Setting Options that Affect the Comparison | Selecting the Algorithm for the Comparison | Limitations | XML Diff Language (Diffgram) | Path Descriptors | Example of a Diffgram |XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Move Operation

Identifies a move operation of a node or XML fragment. There can be one `xd:add` operation and one `xd:remove` operation with the same corresponding **opid**.

The following is an example of a move operation descriptor:

```
<xd:descriptor type="move" opid="2"/>
```

The preceding example shows that all operations with an **opid** attribute with the value 2 from a move operation.

There are no other attributes for the move descriptor element. Its content must be empty.

See Also

XML Diff Functionality | Running Comparisons Between Documents, Fragments, or Nodes | Setting Options that Affect the Comparison | Selecting the Algorithm for the Comparison | Limitations | XML Diff Language (Diffgram) | Path Descriptors | Extended Operations | Namespace Change Operation | Prefix Change Operation | XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Namespace Change Operation

This extended operation identifies a change of namespace declaration. It consists of one `xd:add` operation, to show the addition of the new namespace declaration, and one `xd:remove` operation, to show the removal of the old namespace declaration. Additionally, the **XDL Diffgram** may contain any number of the `xd:change` operations with the same **opid** for every element or attribute name that has its namespace URI changed due to this action.

The following is an example of what a `namespace change` element looks like:

```
<xd:descriptor type="namespace change" opid="3" oldNs="http://some.u
```

This shows that a namespace change has occurred, and shows the old namespace and new namespace. All operations with an **opid** attribute with the value *3* are part of this action.

The following table describes additional attributes of the `xd:descriptor` element for a namespace change operation.

Attribute name	Description
oldNs	Old namespace URI.
newNs	New namespace URI

See Also

XML Diff Functionality | Running Comparisons Between Documents, Fragments, or Nodes | Setting Options that Affect the Comparison | Selecting the Algorithm for the Comparison | Limitations | XML Diff Language (Diffgram) | Path Descriptors | Extended Operations | Move Operation | Prefix Change Operation |XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Prefix Change Operation

This extended operation identifies a change of prefix. It consists of one `xs:add` operation, which show the addition of the new prefix, and one `xd:remove` operation, which show the removal of the prefix. There can also be any number of the `xd:change` operations with the same **opid** for each element or attribute name that is affected by the changed prefix.

The following is an example of what a `prefix change` element looks like:

```
<xd:descriptor type="prefix change" opid="3" oldPrefix="xslt" newPre
```

The preceding example shows that a prefix change has occurred, and shows the old and new prefixes. All operations with an **opid** attribute with value *3* are part of this action.

The following table describes additional attributes of the `xd:descriptor` element for a prefix change operation.

Attribute name	Description
oldPrefix	Old prefix.
newPrefix	New prefix

See Also

XML Diff Functionality | Running Comparisons Between Documents, Fragments, or Nodes | Setting Options that Affect the Comparison | Selecting the Algorithm for the Comparison | Limitations | XML Diff Language (Diffgram) | Path Descriptors | Extended Operations | Move Operation | Namespace Change Operation |XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

Example of a Diffgram

The following table shows two XML documents: one is the source and the other is the changed document. These documents are the input to a **Compare** method that creates an XDL Diffgram. The line number shown is added for readability purposes.

Line number	Source XML document	Line number	Changed XML document
1	<?xml version="1.0"?>	1	<?xml version="1.0"?>
2		2	
3	<a>Some text 1	3	<yy>Some text 1</yy>
4	Some text 2	4	Some text 2
5	<c>Some text 3</c>	5	<c>Some text 3</c>
6	<d>	6	<e>Some text 4</e>
7	Another text	7	<f>Some text 5</f>
8	<fob/>	8	<d>Changed text</d>
9	</d>	9	<x firstAttr="changed attribute value" newAttr="new value"/>
10	<x firstAttr="value1" secondAttr="value2"/>	10	
11	<y>	11	<q>
12	<!--Any comments?-->	12	<y>
13	<z id="10">Just another text</z>	13	<!--Any comments?-->
14	</y>	14	<z id="10">Just another text</z>
15		15	</y>
16		16	</q>
17		17	
18		18	
19		19	

Following is the **XDL Diffgram** produced when a **Compare** method is run against the two documents shown in the preceding table. The line number is added for readability purposes.

Line number	XDL Diffgram output
1	<?xml version="1.0" encoding="utf-16"?>
2	<xd:xmldiff version="1.0" srcDocHash="5346998544451918424" options="None" xmlns:xd="http://schemas.microsoft.com/xmltools/2002/xmldiff">
3	<xd:node match="2">
4	<xd:change match="1" name="yy" />
5	<xd:node match="3" />
6	<xd:add>
7	<e>Some text 4</e>
8	<f>Some text 5</f>
9	</xd:add>
10	<xd:node match="4">
11	<xd:change match="1">Changed text</xd:change>
12	<xd:remove match="2" />
13	</xd:node>
14	<xd:node match="5">
15	<xd:remove match="@secondAttr" />
16	<xd:add type="2" name="newAttr">new value</xd:add>
17	<xd:change match="@firstAttr">changed attribute value</xd:change>
18	</xd:node>
19	<xd:remove match="6" opid="1" />
20	<xd:add type="1" name="p">
21	<xd:add type="1" name="q">
22	<xd:add match="/2/6" opid="1" />
23	</xd:add>
24	</xd:add>
25	</xd:node>

```
26          <xd:descriptor opid="1" type="move" />
27        </xd:xmldiff>
```

The following list shows each line in the **XDL Diffgram** output from the preceding table. Here you can see why each line was generated.

Line 1: is required to create a well-formed XML document.

Line 2: is the root element of the **XDL Diffgram** `xd:xmldiff`. It contains the version of the **XDL Diffgram** in the version attribute, the XDL namespace declaration for the `xd:` prefix, and the XML Diff options selected when the **XDL Diffgram** was created in the options attribute. It also contains **srcDocHash** attribute with a number that is calculated from the source document and is called a hash value. This number allows checking if an XML document is the correct source document the **XDL Diffgram** was created on. This check is performed by XML Patch tool.

Line 3: is an `xd:node` element with a **match** attribute containing a path descriptor pointing to the second element at the source tree root level, which is the `<b>` element on line 2 on the source document. It indicates that there will be some changes described on either the child nodes of the `<b>` element, or on the sibling nodes following the `<b>` element, or both. In this particular case, the child nodes of `xd:node` describes the changes on the child nodes of the `<b>` element.

Line 4: indicates a change has occurred. The changed node is identified with a path descriptor "1" which points to `<a>`, the first child node of the `<b>` element. The **name** attribute is the new local name for the node. In this particular case, the `xd:change` is indicating that element `<a>` in the source document has been changed to an element with a name of "yy" in the new document.

Line 5: is an `xd:node` element with a **match** attribute containing a path descriptor pointing to the third child node of the `<b>` element, which is the `<c>` element. The path descriptor is relative to the `xd:node` **match** element that it is nested under, so it is pointing to the third child node (`<c>` element) under the second node at root level (`<b>` element, see line 3). In this case, the `xd:node` has no child nodes and it is indicating that there will be some changes described on the following sibling nodes. In other words, it indicates that something is happening, starting at this location.

Line 6: indicates an add operation, which is occurring under after the <c> element of the source document. The <c> element has been specified by the previous <xd:node> operation (line 5). Note that there are no attributes associated with the xd:add, so the content of this element is the new fragment that has been added in the changed document.

Line 7-8: shows the XML fragment that has been added in the changed document after the <c> element. It adds the <e> and <f> element from lines 6-7 of the changed document.

Line 9: is an end element, indicating the end of the add operation.

Line 10: is a match operation, with its path descriptor pointing to the fourth child node. They are still nested under the match operation from line 3, which is a match of element 2, the <b> element, so the fourth child node under the <b> element is the <d> element. It is now pointing at the <d> element. The child nodes of the xd:node element will describe changes of the child nodes of the <d> element.

Line 11: xd:change element that indicates that a change has occurred. The changed node is identified with a path descriptor "1" that points to the first child node of the <d> element, which is a text node, "Another text" (line 7 of the source document). The content of the xd:change element is the new value of the text node.

Line 12: xd:remove element that indicates a node has been removed. The path descriptor points to the second child node of the <d> element, which is the <fob/> element. So this indicates that the <fob/> element has been removed and is no longer present in the changed document.

Line 13: is the end element for the xd:node element from line 10. Now you see they are nested under the <b> element again.

Line 14: is an xd:node element with the path descriptor pointing to the fifth child node of the <b> element, which is the <x> element. The child nodes of this xd:node element will describe changes of the child nodes of the <x> element.

Line 15: is a remove operation, and is using a path descriptor that contains the "@" symbol. The "@" symbol indicates that the patch descriptor points to an attribute. The attribute name follows the "@" symbol and it is "**secondAttr**". So

this `xd:remove` element is showing the removal of the **secondAttr** attribute from the <x> element.

Line 16: shows an add operation on a node type of 2, which is the addition of an attribute node. Its name is *newAttr* and the value is "*new value*". So in the changed document the <x> element has a new attribute with the name **newAttr** and value of "*new value*". This is shown in the changed document on line 9.

Line 17: shows a change operation. The **match** attribute indicates that the change is occurring to the attribute named "**firstAttr**", and the content shows the new value of the attribute. This indicates that in the changed document the "**firstAttr**" attribute has its value changed to "changed attribute value". The old attribute value of "value1" has been replaced. The old attribute value is seen in the original document at line 10. The new, modified value is shown in the changed document at line 9.

Line 18: is the end of the `xd:node` element from line 14. Now you can see that they are nested under the <b> element again.

Line 19: is a `remove` operation of the sixth child node of the <b> element, which is the <y> element (and all its descendant nodes). Notice that there is an **opid**, so this remove operation is part of an extended operation. The extended operation will be found later in the **XDL Diffgram** under an `xd:descriptor` element with the same **opid**. Line 26 is the `xd:descriptor` with the matching **opid**, and shows that this remove operation is part of a larger move operation.

Line 20: is an `add` operation, adding a node of type 1, which is an element node. The new element name is "p". This new element is shown in the modified document at line 10. The child nodes of this <xd:add> element describes the child nodes of the new <p> element. So you can see that they now nested under the <p> element.

Line 21: is an `add` operation, adding a node of type 1, which is an element node. The new element name is "q". This new element is shown in the modified document at line 11. The <q> element is a child of the <p> element. The child nodes of this <xd:add> element describes the child nodes of the <q> element. Now they are nested under the <q> element.

Line 22: is an add operation with a path descriptor /2/6, which is an absolute path descriptor. It points to the sixth child node of the second node at root level,

which is the <y> element shown in the original document at line 11. Note that the `xd:add` operation has an **opid** attribute, so this operation is part of an extended operation. The extended operation will be found later in the **XDL Diffgram** under an `xd:descriptor` element with the same **opid**. Line 26 is the `xd:descriptor` with the matching **opid**, and shows that this add operation is part of a larger move operation.

Line 23: is an end element, ending the `xd:add` operation from line 21. Now they are nested under the <p> element.

Line 24: is an end element, ending the `xd:add` operation from line 20. Now they are nested under the <b> element.

Line 25: is an end element, ending the `xd:node` from line 3. Now they are nested under the original document root node.

Line 26: is an `xd:descriptor` describing an extended operation. It shows an operation ID of "1" in the **opid** attribute and the type of the operation "move" in the type attribute. It means that there is a remove and add operation somewhere in the **XDL Diffgram**, both with an **opid** of "1", and it indicates that an existing node was removed from the original document and added back into a different spot in the changed document, which is basically a move operation. Reviewing the **XDL Diffgram**, the moved node was the <y> node (and all its descendants) at line 11 of the original document. In the **XDL Diffgram**, there is a remove operation with an **opid** attribute value of 1 back at line 18, and an add operation back at line 22. The <y> element has been moved to line 12 in the changed document.

Line 27: indicates the end of the `xd:xmldiff` element, and the end of the **XDL Diffgram**.

See Also

XML Diff Functionality | Running Comparisons Between Documents, Fragments, or Nodes | Setting Options that Affect the Comparison | Selecting the Algorithm for the Comparison | Limitations | XML Diff Language (Diffgram) | Path Descriptors | Extended Operations |XmlDiff Class

Microsoft XML Diff 1.0 and XML Patch 1.0

XML Patch Functionality

The Microsoft XML Patch enables you to take an **XDL Diffgram** produced from the Microsoft XML Diff tool, and apply it to the original source document to recreate the changed document.

This is useful if you have multiple source documents at various physical locations, and one of these documents changes. You may use the **XDL Diffgram** and XML Patch tool to propagate the change to all the other source documents by applying the **XDL Diffgram** to them.

A custom application could use the **XDL Diffgram** as a file that shows modifications made to a source document, similar to change tracking for audit purposes. Whatever your need, the XML Patch tool allows the use of the **XDL Diffgram** to create the changed document, fragment or node from the original source document, fragment or node.

The **XmlPatch** class is the class that performs the document, fragment or node modification. It uses the **Patch** method to apply the **XDL Diffgram** to a source document to create a patched document. There are several overloaded **Patch** methods offered, each reading in the source document from different formats, and then saving the patched document in different formats. For more information on the overloaded **Patch** methods, see XmlPatch.Patch Method.

The following code sample loads a source document and an **XDL Diffgram**, and saves the changed **sourceDoc** into a new file called **changed_doc.xml**.

```
[Visual Basic]
Imports System
Imports System.IO
Imports System.Xml
Imports Microsoft.XmlDiffPatch

Public Class Sample

   Public Shared Sub Main()
      Dim sourceDoc As New XmlDocument()
      sourceDoc.Load("source.xml")
      Dim myPatch As New XmlPatch()
```

```
      Dim myRdr As New XmlTextReader("diffgram.xml")
      myPatch.Patch(sourceDoc, myRdr)
      sourceDoc.Save("changed_doc.xml")
   End Sub 'Main
End Class 'Sample
[C#]
using System;
using System.IO;
using System.Xml;
using Microsoft.XmlDiffPatch;

public class Sample
{

    public static void Main()
    {
        XmlDocument sourceDoc = new XmlDocument();
        sourceDoc.Load("source.xml");
        XmlPatch myPatch = new XmlPatch();
        XmlTextReader myRdr = new XmlTextReader("diffgram.xml");
        myPatch.Patch(sourceDoc, myRdr);
        sourceDoc.Save("changed_doc.xml");
    }
}
```

For more information on running the code sample, see Running XmlDiff and XmlPatch Class Code Samples.

One **Patch** method takes an **XmlDocument** as the source document, and applies the **XDL Diffgram** that is parsed by a derived class of the **XmlReader**. The **Patch** method will reject a document with entity references that is loaded into the **XmlDocument**, and whose **XDL Diffgram** has been generated by a **Compare** that compared documents parsed by an **XmlTextReader**, as that **XDL Diffgram** will not have expanded entity references. The **Patch** method in the XmlPatch class will reject a DOM document loaded into **XmlDocument**. That is, if it contains expanded entity references and if the **XDL Diffgram** supplied for patching has been generated by a **Compare** method that takes document files or **XmlReaders** as input parameters.

For more information on the **XmlPatch** methods and properties, see XmlPatch Class.

See Also

Xml Diff Functionality | XML Diff Language (Diffgram) | Microsoft.XMLDiffPatch Namespace

Microsoft XML Diff 1.0 and XML Patch 1.0

Running XmlDiff and XmlPatch Class Code Samples

The code samples shown in this documentation assumes that the common language runtime has been downloaded from http://msdn.microsoft.com/netframework/downloads/howtoget.asp.

For full code samples, copy and paste the code into Notepad, and save the file as CompareNodes.txt. Next, open up a command prompt. The following examples show that two different commands are used for running code, depending on the language of the code sample.

The following command compiles the code for Visual Basic code samples. This command assumes that the code is stored on the C drive and is called CompareNodes.txt.

Visual Basic Samples

```
C:\>vbc /r:system.xml.dll /r:system.dll /r:xmldiffpatch.dll C:\Compa
```

When the code compiles successfully, the executable version of the code is saved in another file that has the same name, however the extension of the file is .exe. You can now run the executable by issuing the command:

```
C:\>C:\CompareNodes
```

The following command compiles the code for C# code samples. This command assumes that the code is stored on the C drive and is called CompareNodes.txt.

C# Samples

```
C:\>csc /r:system.xml.dll;system.dll; C:\CompareNodes.txt
```

When the code compiles successfully, the executable version of the code is saved in another file that has the same name, however the extension of the file is .exe. You can now run the executable by issuing the command:

```
C:\>C:\CompareNodes
```

Microsoft XML Diff 1.0 and XML Patch 1.0

Microsoft.XmlDiffPatch Namespace

The **Microsoft.XmlDiffPatch** namespace contains two classes, the Microsoft XML Diff and Patch 1.0 tool. The XML diff functionality is used to compare two documents, fragments, or nodes for differences; it creates an **XDL Diffgram** describing the differences.

The XML patch functionality takes an **XDL Diffgram** and applies it to a document or nodes to recreate the changed document.

Classes

Class	Description
XmlDiff	Represents the class that performs a comparison of XML documents, fragments, or nodes.
XmlPatch	Applies an **XDL Diffgram** to a source document to create a modified document in place in memory.

Enumerations

Enum	Description
XmlDiffOptions	Specifies what options to include in the comparison.
XmlDiffAlgorithm	Specifies which algorithm the XML diff should use for the comparison.

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Class

Represents the class that performs a comparison of XML documents, fragments, or nodes.

For a list of all members of this type, see XmlDiff Members.

Object

  XmlDiff

```
[Visual Basic]
Public Class XmlDiff
[C#]
public class XmlDiff;
```

Requirements

Namespace: Microsoft.XmlDiffPatch

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

Assembly: Microsoft.XmlDiffPatch (in XmlDiffPatch.dll)

See Also

XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Members

XmlDiff overview

Public Instance Constructors

XmlDiff Constructor	Overloaded. Initializes a new instance of the **XmlDiff** class.

Public Instance Properties

IgnoreChildOrder	The order of child nodes of each element is ignored when **true**. When this option is selected, two nodes with the same value that differ only by their position among sibling child nodes are treated as the same nodes.
IgnoreComments	Comment nodes are not compared when **true**.
IgnoreDtd	The XML Document Type Declaration (DTD) is not compared when **true**.
IgnoreNamespaces	The namespace URIs of the element and attribute names are not compared when **true**. This option also implies that the prefixes are ignored. When this option is selected, then two names with the same local name, but have a different namespace URI and prefix, are treated as the same names.
IgnorePI	Processing instructions are not compared when **true**.
	The prefixes of element and attribute names are not compared when **true**. When this option is selected, then two

IgnorePrefixes	names that have the same local name and namespace URI, but have a different prefix, are treated as the same names.
IgnoreWhitespace	Significant white spaces are not compared when true, and all text nodes are normalized by discarding any leading and trailing white space characters (#x9, #x10, #x13, #x20), and by replacing sequences of white space characters by a single space (#x20) character.
IgnoreXmlDecl	The XML declaration is not compared when **true**.
Options	Set all the properties of the **XmlDiffOptions** enumeration.
Algorithm	Specifies the algorithm that the will be used for comparing the documents. The type is XmlDiffAlgorithm Enumeration.

Public Instance Methods

Compare	Overloaded. Performs a comparison of XML documents, fragments, or nodes.
Equals (Inherited from **Object**)	Determines whether two **Object** instances are equal.
GetHashCode (Inherited from **Object**)	Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
GetType (Inherited from **Object**)	Gets the **Type** of the current instance.
ParseOptions	Parses the **options** attribute from an existing **XDL Diffgram** root element and returns an **XmlDiffOptions** enumeration.

ToString (Inherited from **Object**)	Creates and returns a string representation of the current **Object**.
VerifySource	Returns true if the source document is the document, node, or fragment that the **XDL Diffgram** was generated from.

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Constructor

Initializes a new instance of the **XmlDiff** class.

Overload List

Initializes the **XmlDiff** class with default options.

[Visual Basic] Overloads Public Sub New();

[C#] public XmlDiff();

Initializes the **XmlDiff** class with the specified options.

[Visual Basic] Overloads Public Sub New(XmlDiffOptions);

[C#] public XmlDiff(XmlDiffOptions);

Example

The following example creates the **XmlDiff** class with no options.

```
[Visual Basic]
Dim xmlDiff As New XmlDiff()
[C#]
XmlDiff xmlDiff = new XmlDiff();
```

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Constructor ()

Initializes the **XmlDiff** class with default options.

```
[Visual Basic]
Overloads Public Sub New()
[C#]
public XmlDiff();
```

Example

The following example creates the **XmlDiff** class with no options.

```
[Visual Basic]
Dim xmlDiff As New XmlDiff()
[C#]
XmlDiff xmlDiff = new XmlDiff();
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Constructor (XmlDiffOptions)

Initializes the **XmlDiff** class with the specified options.

```
[Visual Basic]
Overloads Public Sub New(XmlDiffOptions options)
[C#]
public XmlDiff(XmlDiffOptions options);
```

Parameters

options

A pipe (|) delineated list (`or` in Visual Basic) of the **XmlDiffOptions** enumerations to use, or the enumeration **None** to run the comparison with all of the **XmlDiffOptions** set to **false**.

Example

```
[Visual Basic]
Dim xmlDiff As New XmlDiff(XmlDiffOptions.IgnoreComments Or XmlDiffO
[C#]
XmlDiff xmlDiff = new XmlDiff(XmlDiffOptions.IgnoreComments | XmlDif
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Properties

The properties of the **XmlDiff** class are listed here. For a complete list of **XmlDiff** class members, see the XmlDiff Members topic.

Public Instance Properties

IgnoreChildOrder	The order of child nodes of each element is ignored when **true**. When this option is selected, two nodes with the same value that differ only by their position among sibling child nodes are treated as the same nodes.
IgnoreComments	Comment nodes are not compared when **true**.
IgnoreDtd	The XML DTD is not compared when **true**.
IgnoreNamespaces	The namespace URIs of the element and attribute names are not compared when **true**. This option also implies that the prefixes are ignored. When this option is selected, then two names with the same local name, but have a different namespace URI and prefix, are treated as the same names.
IgnorePI	Processing instructions are not compared when **true**.
IgnorePrefixes	The prefixes of element and attribute names are not compared when **true**. When this option is selected, then two names that have the same local name and namespace URI, but have a different prefix, are treated as the same names.
	Significant white spaces are not

IgnoreWhitespace	compared when true, and all text nodes are normalized by discarding any leading and trailing white space characters (#x9, #x10, #x13, #x20), and by replacing sequences of white space characters by a single space (#x20) character.
IgnoreXmlDecl	The XML declaration is not compared when **true**.
Options	Set all the properties of the **XmlDiffOptions** enumeration.
Algorithm	Specifies the algorithm that the will be used for comparing the documents. The type is XmlDiffAlgorithm Enumeration.

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.IgnoreChildOrder Property

The order of child nodes of each element is ignored when **true**. When this option is selected, two nodes with the same value that differ only by their position among sibling child nodes are treated as the same nodes.

```
[Visual Basic]
Public Property IgnoreChildOrder() As Boolean
[C#]
public Boolean IgnoreChildOrder {get; set;};
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.IgnoreComments Property

Comment nodes are not compared when **true**.

```
[Visual Basic]
Public Property IgnoreComments() As Boolean
[C#]
public Boolean IgnoreComments {get; set;}
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.IgnoreDtd Property

The XML Document Type Declaration (DTD) is not compared when **true**.

```
[Visual Basic]
Public Property IgnoreDtd() As Boolean
[C#]
public Boolean IgnoreDtd {get; set;}
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.IgnoreNamespaces Property

The namespace URIs of the element and attribute names are not compared when **true**. This option also implies that the prefixes are ignored.

When this option is selected, then two names that have the same local name, but have a different namespace URI and prefix, are treated as the same.

```
[Visual Basic]
Public Property IgnoreNamespaces() As Boolean
[C#]
public Boolean IgnoreNamespaces {get; set;}
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.IgnorePI Property

Processing instructions are not compared when **true**.

```
[Visual Basic]
Public Property IgnorePI() As Boolean
[C#]
public Boolean IgnorePI {get; set;}
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.IgnorePrefixes Property

The prefixes of element and attribute names are not compared when **true**. When this option is selected, then two names that have the same local name and namespace URI, but have a different prefix, are treated as the same.

```
[Visual Basic]
Public Property IgnorePrefixes() As Boolean
[C#]
public Boolean IgnorePrefixes {get; set;}
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.IgnoreWhitespace Property

Significant white spaces are not compared when **true**, and all text nodes are normalized by discarding any leading and trailing white space characters (#x9, #x10, #x13, #x20), and by replacing sequences of white space characters by a single space (#x20) character.

```
[Visual Basic]
Public Property IgnoreWhitespace() As Boolean
[C#]
public Boolean IgnoreWhitespace {get; set;}
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.IgnoreXmlDecl Property

The XML declaration is not compared when **true**.

```
[Visual Basic]
Public Property IgnoreXmlDecl() As Boolean
[C#]
public Boolean IgnoreXmlDecl {get; set;}
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.Options Property

Sets all the properties of the **XmlDiffOptions** enumeration.

```
[Visual Basic]
Public Property Options() As XmlDiffOptions
[C#]
public XmlDiffOptions Options {set;}
```

Example

The following example shows how to set the **IgnorePI** and **IgnoreComments** enumerations by using the **XmlDiff.Options** property explicitly.

```
[Visual Basic]
Imports System
Imports System.Xml
Imports System.IO
Imports Microsoft.XmlDiffPatch

Namespace TestCompare
   Class Class1
      Shared Sub Main()
        Dim myDiff As New XmlDiff()
        Dim diffgramWriter = New XmlTextWriter(New StreamWriter("dif
        myDiff.Options = XmlDiffOptions.IgnorePI Or XmlDiffOptions.I
        Dim bSame As Boolean = myDiff.Compare("Source.xml", "Changed
        diffgramWriter.Close()
      End Sub
   End Class
End Namespace
[C#]
using System;
using System.Xml;
using System.IO;
using Microsoft.XmlDiffPatch;

namespace TestCompare
{
  class Class1
    {
        static void Main()
```

```
          {
            XmlDiff myDiff = new XmlDiff();
            XmlWriter diffgramWriter = new XmlTextWriter( new Stream
myDiff.Options = XmlDiffOptions.IgnorePI | XmlDiffOptions.IgnoreComm
            bool bSame = myDiff.Compare("Source.xml", "Changed.xml",
            diffgramWriter.Close();
            }
    }
}
```

For information on running the code samples, see Running XmlDiff and XmlPatch Class Code Samples.

There are several other ways to set the comparison options. For examples, see Setting Options that Affect the Comparison.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.Agorithm Property

Specifies the algorithm that the will be used for comparing the documents. The type is XmlDiffAlgorithm Enumeration. The default is **Auto**.

```
[Visual Basic]
Public Property Algorithm() As XmlDiffAlgorithmn [C#]
public XmlDiffAlgorithm Options {set;}
```

Remarks

The following code sample shows how to use the **Precise** algorithm when comparing two documents.

```
[Visual Basic]
Imports System
Imports System.Xml
Imports Microsoft.XmlDiffPatch

Namespace TestCompare
   Class Class1
      Shared Sub Main()
        Dim diffWriter = New XmlTextWriter("diffgram.xml", New Syste
        Dim myDiff As New XmlDiff()
        myDiff.Algorithm = XmlDiffAlgorithm.Precise
        Dim bSame As Boolean = myDiff.Compare("source.xml", "changed
        Console.WriteLine("The answer is {0} ", bSame)
      End Sub
   End Class
End Namespace

[C#]
using System;
using System.Xml;
using Microsoft.XmlDiffPatch;

namespace TestCompare
{
  class Class1
    {
        static void Main()
            {
```

```
            XmlWriter diffWriter = new XmlTextWriter("diffgram.xml",
            XmlDiff myDiff = new XmlDiff();
            myDiff.Algorithm = XmlDiffAlgorithm.Precise;
            bool bSame = myDiff.Compare("source.xml", "changed.xml",
        Console.WriteLine("The answer is {0} ", bSame);
            }
    }
}
```

For information on running the code samples, see Running XmlDiff and XmlPatch Class Code Samples.

For more information on the enumerations, see XmlDiffAlgorithm Enumeration.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff Methods

The methods of the **XmlDiff** class are listed here. For a complete list of **XmlDiff** class members, see the XmlDiff Members topic.

Public Instance Methods

Compare	Overloaded. Performs a comparison of XML documents, fragments, or nodes.
Equals (Inherited from **Object**)	Determines whether two **Object** instances are equal.
GetHashCode (Inherited from **Object**)	Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
GetType (Inherited from **Object**)	Gets the **Type** of the current instance.
ParseOptions	Parses the *options* attribute from an existing **XDL Diffgram** and returns an **XmlDiffOptions** enumeration.
ToString (Inherited from **Object**)	Creates and returns a string representation of the current **Object**.
VerifySource	Returns true if the source document is the document, node, or fragment that the **XDL Diffgram** was generated from.

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.Compare Method

Performs a comparison of XML documents, fragments, or nodes.

Overload List

Performs a comparison of XML documents or fragments.

[Visual Basic] Overloads Public Function Compare(String, String, Boolean) As Boolean

[C#] public Boolean Compare(String, String, Boolean)

Performs a comparison of XML documents or fragments and outputs an **XDL Diffgram** describing the differences.

[Visual Basic] Overloads Public Function Compare(String, String, Boolean, XmlWriter) As Boolean

[C#] public Boolean Compare(String, String, Boolean, XmlWriter)

Performs a comparison of XML nodes.

[Visual Basic] Overloads Public Function Compare(XmlNode, XmlNode) As Boolean

[C#] public Boolean Compare(XmlNode, XmlNode)

Performs a comparison of XML nodes and outputs an **XDL Diffgram** describing the differences.

[Visual Basic] Overloads Public Function Compare(XmlNode, XmlNode, XmlWriter) As Boolean

[C#] public Boolean Compare(XmlNode, XmlNode, XmlWriter)

Performs a comparison of XML documents parsed by an **XmlReader**.

[Visual Basic] Overloads Public Function Compare(XmlReader, XmlReader) As Boolean

[C#] public Boolean Compare(XmlReader, XmlReader)

Performs a comparison of XML documents parsed by an **XmlReader** and outputs an **XDL Diffgram** describing the differences.

[Visual Basic] Overloads Public Function Compare(XmlReader, XmlReader, XmlWriter) As Boolean

[C#] public Boolean Compare(XmlReader, XmlReader, XmlWriter)

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.Compare Method (String, String, Boolean)

Performs a comparison of XML documents or fragments.

```
[Visual Basic]
Overloads Public Function Compare( _
  ByVal sourceFile As String, _
  ByVal changedFile As String, _
  ByVal bFragments As Boolean _
) As Boolean
[C#]
public Boolean Compare(
  String sourceFile,
  String changedFile,
  Boolean bFragments
);
```

Parameters

sourceFile
: A file name or URL containing the original source XML document or fragment to be used in the comparison.

changedFile
: A file name or URL containing the changed XML document or fragment to be used in the comparison.

bFragments
: A flag indicating whether the *sourceFile* and *changedFile* are well-formed XML documents (**false**) or whether they are fragments (**true**).

Example

```
[Visual Basic]
Imports System
Imports System.Xml
Imports System.IO
Imports Microsoft.XmlDiffPatch
```

```
Namespace XDConsole_CS
   Class Class1
      Public Overloads Shared Sub Main()
         Main(System.Environment.GetCommandLineArgs())
      End Sub

      Overloads Shared Sub Main(args() As String)
         Dim xmlDiff As New XmlDiff()
         Dim bSame As Boolean = xmlDiff.Compare("Source.xml", "Chang
      End Sub
   End Class
End Namespace
[C#]
using System;
using System.Xml;
using System.IO;
using Microsoft.XmlDiffPatch;

namespace XDConsole_CS
{
    class Class1
    {
        static void Main(string[] args)
        {
            XmlDiff xmlDiff = new XmlDiff();
            bool bSame = xmlDiff.Compare("Source.xml", "Changed.xml"
        }
    }
}
```

For information on running the code samples, see Running XmlDiff and XmlPatch Class Code Samples.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.Compare Method (String, String, Boolean, XmlWriter)

Performs a comparison of XML documents or fragments and outputs an **XDL Diffgram** describing the differences.

```
[Visual Basic]
Overloads Public Function Compare( _
  ByVal sourceFile As String, _
  ByVal changedFile As String, _
  ByVal bFragments As Boolean, _
  ByVal diffgramWriter As XmlWriter _
) As Boolean
[C#]
public Boolean Compare(
  String sourceFile,
  String changedFile,
  Boolean bFragments,
  XmlWriter diffgramWriter
);
```

Parameters

sourceFile
: A file name or URL containing the original source XML document or fragment to be used in the comparison.

changedFile
: A file name or URL containing the changed XML document or fragment to be used in the comparison.

bFragments
: A flag indicating whether the *sourceFile* and *changedFile* are well-formed XML documents (**false**) or whether they are fragments (**true**).

diffgramWriter
: The **XmlWriter** to which you want to output the **XDL Diffgram**.

Example

```
[Visual Basic]
Imports System
Imports System.Xml
Imports Microsoft.XmlDiffPatch

Namespace TestCompare

   Class Class1

      Shared Sub Main()
        Dim diffgramWriter = New XmlTextWriter("diffgram.xml", New S
        Dim xmlDiff As New XmlDiff()
        Dim bSame As Boolean = xmlDiff.Compare("Source.xml", "Change
        diffgramWriter.Flush()
        diffgramWriter.Close()
      End Sub   End Class
End Namespace
[C#]
using System;
using System.Xml;
using Microsoft.XmlDiffPatch;

namespace TestCompare
{
  class Class1
    {
        static void Main()
            {
                XmlWriter diffgramWriter = new XmlTextWriter("diffgr
                XmlDiff xmlDiff = new XmlDiff();
                bool bSame = xmlDiff.Compare("Source.xml", "Changed.
                diffgramWriter.Flush();
                diffgramWriter.Close();
            }
    }
}
```

For information on running the code samples, see Running XmlDiff and XmlPatch Class Code Samples.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.Compare Method (XmlNode, XmlNode)

Performs a comparison of XML nodes.

```
[Visual Basic]
Overloads Public Function Compare( _
  ByVal sourceNode As XmlNode, _
  ByVal changedNode As XmlNode _
) As Boolean
[C#]
public Boolean Compare(
  XmlNode sourceNode,
  XmlNode changedNode
);
```

Parameters

sourceNode
: An **XmlNode** containing the original source node to be used in the comparison.

changedNode
: An **XmlNode** containing the changed node to be used in the comparison.

Remarks

The types of nodes that can be passed into the **Compare** method are any combination of the following:

- XmlDocument
- XmlElement
- XmlText
- XmlCDataSection
- XmlEntityReference
- XmlComment
- XmlDocumentType
- XmlProcessingInstruction

The **Compare** method cannot be used to compare **XmlAttribute**, **XmlEntity**, or **XmlNotation** node types.

Example

```
[Visual Basic]
Imports System
Imports System.Xml
Imports Microsoft.XmlDiffPatch

Namespace TestCompare
   Class Class1
      Shared Sub Main()
         Dim sourceDoc As New XmlDocument()
         sourceDoc.Load("source.xml")
         Dim changedDoc As New XmlDocument()
         sourceDoc.Load("target.xml")
         Dim xmlDiff As New XmlDiff()
         Dim bSame As Boolean = xmlDiff.Compare(sourceDoc, changedDo
         Console.WriteLine("The answer is {0} ", bSame)
      End Sub
   End Class
End Namespace
[C#]
using System;
using System.Xml;
using Microsoft.XmlDiffPatch;

namespace TestCompare
{
  class Class1
    {
        static void Main()
            {
                XmlDocument sourceDoc = new XmlDocument();
                sourceDoc.Load("source.xml");
                XmlDocument changedDoc = new XmlDocument();
                sourceDoc.Load("target.xml");
                XmlDiff xmlDiff = new XmlDiff();
                bool bSame = xmlDiff.Compare(sourceDoc, changedDoc);
                Console.WriteLine("The answer is " + bSame);
            }
    }
}
```

For information on running the code samples, see Running XmlDiff and XmlPatch Class Code Samples.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.Compare Method (XmlNode, XmlNode, XmlWriter)

Performs a comparison of XML nodes and outputs an **XDL Diffgram** describing the differences.

```
[Visual Basic]
Overloads Public Function Compare( _
  ByVal sourceNode As XmlNode, _
  ByVal changedNode As XmlNode, _
  ByVal diffgramWriter As XmlWriter _
) As Boolean
[C#]
public Boolean Compare(
  XmlNode sourceNode,
  XmlNode changedNode,
  XmlWriter diffgramWriter
);
```

Parameters

sourceFile
: An **XmlNode** containing the original source node to be used in the comparison.

changedFile
: An **XmlNode** containing the changed node to be used in the comparison.

diffgramWriter
: The **XmlWriter** to which you want to output the **XDL Diffgram**.

Remarks

The types of nodes that can be passed into the **Compare** method are any combination of the following:

- XmlDocument
- XmlElement
- XmlText

- XmlCDataSection
- XmlEntityReference
- XmlComment
- XmlDocumentType
- XmlProcessingInstruction

The **Compare** method cannot be used to compare **XmlAttribute**, **XmlEntity**, or **XmlNotation** node types.

Example

```
[Visual Basic]
Imports System
Imports System.Xml
Imports Microsoft.XmlDiffPatch

Namespace TestCompare
   Class Class1

      Shared Sub Main()
         Dim sourceDoc As New XmlDocument()
         sourceDoc.Load("source.xml")
         Dim changedDoc As New XmlDocument()
         sourceDoc.Load("changed.xml")
         Dim diffgramWriter = New XmlTextWriter("testdiff.xml", New
         Dim xmlDiff As New XmlDiff()
         Dim bSame As Boolean = xmlDiff.Compare(sourceDoc, changedDo
         diffgramWriter.Flush()
         diffgramWriter.Close()
      End Sub
   End Class
End Namespace
[C#]
using System;
using System.Xml;
using Microsoft.XmlDiffPatch;
namespace TestCompare
{
  class Class1
    {
        static void Main()
            {
                XmlDocument sourceDoc = new XmlDocument();
                sourceDoc.Load("source.xml");
```

```
                XmlDocument changedDoc = new XmlDocument();
                sourceDoc.Load("changed.xml");
                XmlWriter diffgramWriter = new XmlTextWriter("testdi
                XmlDiff xmlDiff = new XmlDiff();
                bool bSame = xmlDiff.Compare(sourceDoc, changedDoc,
                diffgramWriter.Flush();
                diffgramWriter.Close();
            }
    }
}
```

For information on running the code samples, see Running XmlDiff and XmlPatch Class Code Samples.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.Compare Method (XmlReader, XmlReader)

Performs a comparison of two XML documents or fragments parsed by **XmlReader** objects.

```
[Visual Basic]
Overloads Public Function Compare( _
  ByVal sourceReader As XmlReader, _
  ByVal changedReader As XmlReader _
) As Boolean
[C#]
public Boolean Compare(
  XmlReader sourceReader,
  XmlReader changedReader
);
```

Parameters

sourceReader
: An **XmlReader** parsing the original source XML document or fragment to be used in the comparison.

changedReader
: An **XmlReader** parsing the changed XML document or fragment to be used in the comparison.

Remarks

The result of **Compare** method may differ when comparing documents loaded in DOM and documents parsed by **XmlTextReader**. That is, if the document contains entity references that are expanded by the DOM.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.Compare Method (XmlReader, XmlReader, XmlWriter)

Performs a comparison of two XML documents or fragments parsed by **XmlReader** objects and outputs an **XDL Diffgram** describing the differences.

```
[Visual Basic]
Overloads Public Function Compare( _
  ByVal sourceReader As XmlReader, _
  ByVal changedReader As XmlReader, _
  ByVal diffgramWriter As XmlWriter _
) As Boolean
[C#]
public Boolean Compare(
  XmlReader sourceReader,
  XmlReader changedReader,
  XmlWriter diffgramWriter
);
```

Parameters

sourceReader
: An **XmlReader** parsing the original source XML document or fragment to be used in the comparison.

changedReader
: An **XmlReader** parsing the changed XML document or fragment to be used in the comparison.

diffgramWriter
: The **XmlWriter** to which you want to output the **XDL Diffgram**.

Remarks

The result of **Compare** method may differ when comparing documents loaded in DOM and documents parsed by **XmlTextReader**. That is, if the document contains entity references that are expanded by the DOM.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.ParseOptions Method (String)

Takes an **options** attribute from an existing **XDL Diffgram** root element `xd:xmldiff`, parses it, and returns the **XmlDiffOptions** enumeration.

```
[Visual Basic]
Public Function ParseOptions( _
  ByVal options As String _
) As XmlDiffOptions
[C#]
public XmlDiffOptions ParseOptions(
  String options
);
```

Parameters

options
: The *options* attribute from the root element `xd:xmldiff` of the **XDL Diffgram**.

Remarks

This method is useful when consuming the **XDL Diffgram**. The **options** attribute on the `xd:xmldiff` element contains a space-separated list of options used when the **XDL Diffgram** was generated. The **ParseOptions** method translates this list back into the **XmlDiffOptions** enumeration.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiff.VerifySource Method (XmlNode, UInt64, XmlDiffOptions)

Returns **true** if the passed document, node or fragment is the source document, node, or fragment the **XDL Diffgram** was generated from.

```
[Visual Basic]
Public Function VerifySource( _
  ByVal node As XmlNode, _
  ByVal hashValue As UInt64, _
  ByVal options As XmlDiffOptions _
) As Boolean
[C#]
public Boolean VerifySource(
  XmlNode node,
  UInt64 hashValue,
  XmlDiffOptions options
);
```

Parameters

node

The document, node, or fragment to be verified.

hashValue

The value of the **sourceXmlHash** attribute from the `xd:xmldiff` element that identifies the source document that the **XDL Diffgram** was generated from. This is an example of the `xd:xmldiff` element with the **sourceXmlHash** attribute:

```
<xmldiff
xmlns="http://schemas.microsoft.com/xmltools/2002/xmldiff"
sourceXmlHash=" 3299133317929493637">
```

options

The **XmlDiffOption** enumeration specifying the options used when the **XDL Diffgram** was generated. The enumeration can be obtained by passing the **options** attribute from the `xd:xmldiff` element of an **XDL Diffgram** into the **ParseOptions** method.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlDiff Class | XmlDiff Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiffOptions Enumeration

The **XmlDiff** class allows the setting of different options that affects the behavior of the comparison, as well as the resulting **XDL Diffgram**. The following table describes the enumeration property that affects what items are included for consideration during the comparison. The default values of the options are **false**.

Member Name	Description
IgnoreChildOrder	The order of child nodes of each element is ignored when **true**. When this option is selected, two nodes with the same value that differ only by their position among sibling child nodes, are treated as the same nodes.
IgnoreComments	Comment nodes are not compared when **true**.
IgnoreDtd	Document Type Declaration (DTD) is not compared when **true**.
IgnoreNamespaces	The namespace URIs of the element and attribute names are not compared when **true**. This option also implies that the name prefixes are ignored. When this option is selected, then two names with the same local name, but having a different namespace URI and prefix, are treated as the same names.
IgnorePI	Processing instructions are not compared when **true**.
IgnorePrefixes	The prefixes of element and attribute names are not compared when **true**. When this option is selected, then two names that have the same local name

	and namespace URI, but have a different prefix, are treated as the same names.
IgnoreWhitespace	Significant white spaces are not compared when true, and all text nodes are normalized by discarding any leading and trailing white space characters (#x9, #x10, #x13, #x20), and by replacing sequences of white space characters by a single space (#x20) character.
IgnoreXmlDecl	The XML declaration is not compared when **true**.
None	When passed into the **XmlDiff** constructor as its argument, specifies that all the options in the enumeration are **false**.

See Also

Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlDiffAlgorithm Enumeration

Specifies which algorithm to use when comparing XML documents.

```
[Visual Basic]
Public property Algorithm(ByVal value As XmlDiffAlgorithm)
[C#]
public XmlDiffAlborithm Algorithm {get; set;}
```

Members

Member name	Description
Auto	Default. Chooses the comparison algorithm for you depending on the size and assumed number of changes in the compared documents.
Fast	Compares the two XML documents by traversing the XML tree and comparing it node-by-node. This algorithm is very fast but may produce less precise results. For example it may detect an `add` and `remove` operation on a node instead of a `move` operation.
Precise	Is based on an algorithm for finding editing distance between trees, also known as Zhang-Shasha algorithm. This algorithm gives very precise results but it may be very slow on large XML documents with many changes.

Example

```
[Visual Basic]
Imports System
Imports System.Xml
Imports Microsoft.XmlDiffPatch
```

```
Namespace TestCompare
   Class Class1
      Shared Sub Main()
        Dim diffWriter = New XmlTextWriter("diffgram.xml", New Syste
        Dim myDiff As New XmlDiff()
        myDiff.Algorithm = XmlDiffAlgorithm.Precise
        Dim bSame As Boolean = myDiff.Compare("source.xml", "changed
        Console.WriteLine("The answer is {0} ", bSame)
      End Sub       'Main
   End Class       'Class1
End Namespace       'TestCompare
[C#]
using System;
using System.Xml;
using Microsoft.XmlDiffPatch;

namespace TestCompare
{
  class Class1
    {
        static void Main()
            {
            XmlWriter diffWriter = new XmlTextWriter("diffgram.xml",
            XmlDiff myDiff = new XmlDiff();
            myDiff.Algorithm = XmlDiffAlgorithm.Precise;
            bool bSame = myDiff.Compare("source.xml", "changed.xml",
    Console.WriteLine("The answer is {0} ", bSame);

            }
    }
}
```

Remarks

To select the comparison algorithm you want to use, set the **Algorithm** property of the **XmlDiff** class before calling the **Compare** method. The default value of this property is **XmlDiffAlgorithm.Auto**, which means the comparison algorithm will be automatically selected depending on the size and assumed number of changes in the compared documents.

Requirements

Namespace: Microsoft.XmlDiffPatch

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

Assembly: Microsoft.XmlDiffPatch (in XmlDiffPatch.dll)

See Also

Microsoft.XmlDiffPatch | Microsoft XML Diff 1.0 | Microsoft XML Patch 1.0

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlPatch Class

Applies an **XDL Diffgram** to a source document to create a modified document.

For a list of all members of this type, see XmlPatch Members.

Object

XmlPatch

```
[Visual Basic]
Public Class XmlPatch
[C#]
public class XmlPatch;
```

Requirements

Namespace: Microsoft.XmlDiffPatch

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

Assembly: Microsoft.XmlDiffPatch (in XmlDiffPatch.dll)

See Also

XmlPatch Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlPatch Members

XmlPatch overview

Public Instance Constructors

XmlPatch Constructor	Initializes a new instance of the **XmlPatch** class.

Public Instance Methods

Equals (Inherited from **Object**)	Determines whether two **Object** instances are equal.
GetHashCode (Inherited from **Object**)	Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
GetType (Inherited from **Object**)	Gets the **Type** of the current instance.
Patch	Overloaded. Applies the **XDL Diffgram** to a source document to create the modified document.
ToString (Inherited from **Object**)	Creates and returns a string representation of the current **Object**.

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlPatch Constructor ()

Initializes a new instance of the **XmlPatch** class.

```
[Visual Basic]
Public Sub New()
[C#]
public XmlPatch();
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlPatch Class | XmlPatch Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlPatch Methods

The methods of the **XmlPatch** class are listed here. For a complete list of **XmlPatch** class members, see the XmlPatch Members topic.

Public Instance Methods

Equals (Inherited from **Object**)	Determines whether two **Object** instances are equal.
GetHashCode (Inherited from **Object**)	Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
GetType (Inherited from **Object**)	Gets the **Type** of the current instance.
Patch	Overloaded. Applies the **XDL Diffgram** to a source document to create a modified document.
ToString (Inherited from **Object**)	Creates and returns a string representation of the current **Object**.

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlPatch.Patch Method

Applies the **XDL Diffgram** to a source document to create a modified document.

Overload List

Modifies the document in the **XmlDocument** by applying the **XDL Diffgram** parsed by an **XmlReader**.

[Visual Basic] Overloads Public Sub Patch(XmlDocument, XmlReader)

[C#] public void Patch(XmlDocument, XmlReader)

Modifies the node tree in the **XmlNode** by applying the **XDL Diffgram** parsed by an **XmlReader**.

[Visual Basic] Overloads Public Sub Patch(XmlNode, XmlReader)

[C#] public void Patch(XmlNode, XmlReader)

Modifies the document or fragment identified by the file name or URL by applying the **XDL Diffgram** parsed by an **XmlReader**. The resulting patched document or fragment is outputted to a Stream.

[Visual Basic] Overloads Public Sub Patch(String, Stream, XmlReader)

[C#] public void Patch(String, Stream, XmlReader)

Modifies the document or fragment parsed by an **XmlReader** by applying the **XDL Diffgram** that is also parsed by an **XmlReader**. The resulting patched document is outputted to a Stream.

[Visual Basic] Overloads Public Sub Patch(XmlReader, Stream, XmlReader)

[C#] public void Patch(XmlReader, Stream, XmlReader)

See Also

XmlPatch Class | XmlPatch Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlPatch.Patch Method (XmlDocument, XmlReader)

Modifies the document in the **XmlDocument** by applying the **XDL Diffgram** parsed by an **XmlReader**.

```
[Visual Basic]
Overloads Public Sub Patch( _
  ByVal sourceDoc As XmlDocument, _
  ByVal diffgram As XmlReader _
)
[C#]
public void Patch(
  XmlDocument sourceDoc,
  XmlReader diffgram
);
```

Parameters

sourceDoc
: An **XmlDocument** containing the source document to be modified.

diffgram
: The **XDL Diffgram** that is parsed by the **XmlReader** to apply to the source document.

Example

The following code sample loads a source document and an **XDL Diffgram**, and saves the changed **sourceDoc** into a new file called **changed_doc.xml**.

```
[Visual Basic]
Imports System
Imports System.IO
Imports System.Xml
Imports Microsoft.XmlDiffPatch

Public Class Sample
```

```
   Public Shared Sub Main()
      Dim sourceDoc As New XmlDocument()
      sourceDoc.Load("source.xml")
      Dim myPatch As New XmlPatch()
      Dim myRdr As New XmlTextReader("diffgram.xml")
      myPatch.Patch(sourceDoc, myRdr)
      sourceDoc.Save("changed_doc.xml")
   End Sub 'Main
End Class 'Sample
[C#]
using System;
using System.IO;
using System.Xml;
using Microsoft.XmlDiffPatch;

public class Sample
{
    public static void Main()
    {
        XmlDocument sourceDoc = new XmlDocument();
        sourceDoc.Load("source.xml");
        XmlPatch myPatch = new XmlPatch();
        XmlTextReader myRdr = new XmlTextReader("diffgram.xml");
        myPatch.Patch(sourceDoc, myRdr);
        sourceDoc.Save("changed_doc.xml");
    }
}
```

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlPatch Class | XmlPatch Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlPatch.Patch Method (XmlNode, XmlReader)

Modifies the node tree that is rooted in the **XmlNode** by applying the **XDL Diffgram** parsed by an **XmlReader**.

```
[Visual Basic]
Overloads Public Sub Patch( _
  ByRef sourceNode As XmlNode, _
  ByVal diffgram As XmlReader _
)
[C#]
public void Patch(
  ref XmlNode sourceNode,
  XmlReader diffgram
);
```

Parameters

sourceNode
: An **XmlNode** containing the source node to be modified. The parameter must be passed by reference.

diffgram
: The **XDL Diffgram** that is parsed by the **XmlReader** to apply to the source document.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlPatch Class | XmlPatch Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlPatch.Patch Method (String, Stream, XmlReader)

Modifies the document or fragment indicated by the file name or URL by applying the **XDL Diffgram** parsed by an **XmlReader**. The resulting patched document or fragment is outputted to a Stream.

```
 [Visual Basic]
Overloads Public Sub Patch( _
  ByVal sourceFile As String, _
  ByVal outputStream As Stream, _
  ByVal diffgramReader As XmlReader _
)
[C#]
public void Patch(
  String sourceFile,
  Stream outputStream,
  XmlReader diffgramReader
);
```

Parameters

sourceFile
: A file name or URL containing the original source XML document or fragment to be modified.

outputStream
: The stream to which you want to output the resulting patched document or fragment.

diffgramReader
: The **XDL Diffgram** parsed by the **XmlReader** that is applied to the source XML document or fragment.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlPatch Class | XmlPatch Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

XmlPatch.Patch Method (XmlReader, Stream, XmlReader)

Modifies the document or fragment parsed by an **XmlReader** by applying the **XDL Diffgram**, which is also parsed by an **XmlReader**. The resulting patched document or fragment is outputted to a Stream.

```
 [Visual Basic]
Overloads Public Sub Patch( _
  ByVal sourceReader As XmlReader, _
  ByVal outputStream As Stream, _
  ByVal diffgramReader As XmlReader _
)
[C#]
public void Patch(
  XmlReader sourceReader,
  Stream outputStream,
  XmlReader diffgramReader
);
```

Parameters

sourceReader
: The **XmlReader** parsing the original source XML document or fragment that will be patched.

outputStream
: The stream to which you want to output the resulting patched document or fragment.

diffgramReader
: The **XDL Diffgram** parsed by the **XmlReader** that is applied to the source XML document or fragment.

Requirements

Platforms: Windows 98, Windows NT 4.0, Windows Millennium Edition, Windows 2000, Windows XP, Windows .NET Server

See Also

XmlPatch Class | XmlPatch Members | Microsoft.XmlDiffPatch

Microsoft XML Diff 1.0 and XML Patch 1.0

Copyright and Legal Information

© 2002 Microsoft Corporation. All rights reserved.