

Web.config Transformation Syntax for Web Project Deployment Using Visual Studio

.NET Framework 4.5

This topic provides reference information about the syntax of **xdt:Transform** and **xdt:Locator** attributes that you use in Web.config transform files.

Note

This topic applies to Visual Studio 2012 and Visual Studio Express 2012 for Web. The topic covers features that are included in the latest [Visual Studio Web Publish Update](#) available as of June, 2013. Most of these features are also available in Visual Studio 2010 and Visual Web Developer 2010 Express when you install the Web Publish Update.

A transform file is an XML file that specifies how the Web.config file should be changed when it is deployed. Transformation actions are specified by using XML attributes that are defined in the **XML-Document-Transform** namespace, which is mapped to the **xdt** prefix. The **XML-Document-Transform** namespace defines two attributes: **Locator** and **Transform**. The **Locator** attribute specifies the Web.config element or set of elements that you want to change in some way. The **Transform** attribute specifies what you want to do to the elements that the **Locator** attribute finds.

The following example shows the contents of a transform file that changes a connection string and replaces the **customErrors** element:

Xml

```
<?xml version="1.0"?>
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
  <connectionStrings>
    <add name="MyDB"
        connectionString="value for the deployed Web.config file"
        xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>
  </connectionStrings>
  <system.web>
    <customErrors defaultRedirect="GenericError.htm"
        mode="RemoteOnly" xdt:Transform="Replace">
      <error statusCode="500" redirect="InternalError.htm"/>
    </customErrors>
  </system.web>
</configuration>
```

The root element of a transform file must specify the **XML-Document-Transform** namespace in its opening tag, as shown in the preceding example. The **Locator** and **Transform** elements themselves are not reproduced in the deployed Web.config file.

If you are deploying to a Windows Azure Web Site, and if you want to transform settings in the **appSettings** or **connectionStrings** elements, an alternative is to store the values for the destination site in Windows Azure. For more information, see [Windows Azure Web Sites: How Application Strings and Connection Strings Work](#) on the Windows Azure blog.

For more information, see [Using Web.config transformations to change settings in the destination Web.config file or app.config file during deployment](#) on the ASP.NET site.

Locator Attribute Syntax

Each of the following sections explains the syntax for one **Locator** attribute.

Condition

Specifies an XPath expression that is appended to the current element's XPath expression. Elements that match the combined XPath expression are selected.

Syntax

```
Locator="Condition(XPath expression)"
```

Example

The following example shows how to select connection string elements whose **name** attribute value is **oldname** or a **providerName** attribute whose value is **oldprovider**. In the deployed Web.config file, the selected elements are replaced with the element that is specified in the transform file.

Xml

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add name="AWLT" connectionString="newstring"
        providerName="newprovider"
        xdt:Transform="Replace"
        xdt:Locator="Condition(@name='oldname'
            or @providerName='oldprovider')"/>
  </connectionStrings>
</configuration>
```

The effective XPath expression that is applied to the development Web.config file as a result of the specified **Condition** expression is the following:

```
configuration/connectionStrings/add[@name='AWLT' or @providerName='System.Data.SqlClient']
```

This expression is a result of combining the implicit XPath condition for the current element (**configuration/connectionStrings**) with the expression that is specified explicitly.

Match

Selects the element or elements that have a matching value for the specified attribute or attributes. If multiple attribute names are specified, only elements that match all the specified attributes are selected.

Syntax

```
Locator="Match(comma-delimited list of one or more attribute names)"
```

Example

The following example shows how to select the connection string **add** element that has **AWLT** in the **name** attribute in the development Web.config file. In the deployed Web.config file, the selected element is replaced with the **add** element that is specified in the transform file.

Xml

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add name="AWLT" connectionString="newstring"
        providerName="newprovider"
        xdt:Transform="Replace"
        xdt:Locator="Match(name)" />
  </connectionStrings>
</configuration>
```

XPath

Specifies an absolute XPath expression that is applied to the development Web.config file. (Unlike **Condition**, the expression that you specify is not appended to the implicit XPath expression that corresponds to the current element.)

Syntax

```
Locator="XPath(XPath expression)"
```

Example

The following example shows how to select the same elements that are selected by the preceding example for the **Condition** keyword.

Xml

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add name="AWLT" connectionString="newstring"
        providerName="newprovider"
        xdt:Transform="Replace"
        xdt:Locator="XPath(configuration/connectionStrings[@name='AWLT'
            or @providerName='System.Data.SqlClient'])" />
  </connectionStrings>
</configuration>
```

Transform Attribute Syntax

Each of the following sections explains the syntax for one **Transform** attribute.

Replace

Replaces the selected element with the element that is specified in the transform file. If more than one element is selected, only the first selected element is replaced. For an example of how to use the **Replace** keyword, see the examples for the **Locator** attributes.

▲ Syntax

```
Transform="Replace"
```

▲ Insert

Adds the element that is defined in the transform file as a sibling to the selected element or elements. The new element is added at the end of any collection.

▲ Syntax

```
Transform="Insert"
```

▲ Example

The following example shows how to select all the connection strings in the development Web.config file. In the deployed Web.config file, the specified connection string is added to the end of the collection.

Xml

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add name="AWLT" connectionString="newstring"
        providerName="newprovider"
        xdt:Transform="Insert" />
  </connectionStrings>
</configuration>
```

▲ InsertBefore

Inserts the element that is defined in the transform XML directly before the element that is selected by the specified XPath expression. The XPath expression must be an absolute expression, because it is applied to the development Web.config file as a whole; it is not appended only to the current element's implicit XPath expression.

▲ Syntax

```
Transform="InsertBefore(XPath expression)"
```

▲ Example

The following example shows how to select the **deny** element that denies access to all users. It then inserts an **allow** element before the **deny** element in order to grant access to administrators.

Xml

```
<configuration xmlns:xdt="...">
  <authorization>
    <allow roles="Admins"
        xdt:Transform="InsertBefore(/configuration/system.web/authorization/deny[@users='*'])" />
  </authorization>
</configuration>
```

▲ InsertAfter

Inserts the element that is defined in the transform XML directly after the element that is selected by the specified XPath expression. The XPath expression must be an absolute expression, because it is applied to the development Web.config file as a whole; it is not appended to the current element's implicit XPath expression.

▲ Syntax

```
Transform="InsertAfter(XPath expression)"
```

Example

The following example shows how to select the **allow** element that grants access to administrators, and inserts a **deny** element after it that denies access to a specified user.

Xml

```
<configuration xmlns:xdt="...">
  <authorization>
    <deny users="UserName"
      xdt:Transform="InsertAfter
        (/configuration/system.web/authorization/allow[@roles='Admins'])" />
  </authorization>
</configuration>
```

Remove

Removes the selected element. If multiple elements are selected, removes the first element.

Syntax

```
Transform="Remove"
```

Example

The following example shows how to select all the connection string **add** elements in the development Web.config file. In the deployed Web.config file, only the first connection string element is removed.

Xml

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add xdt:Transform="Remove" />
  </connectionStrings>
</configuration>
```

RemoveAll

Removes the selected element or elements.

Syntax

```
Transform="RemoveAll"
```

Example

The following example shows how to select all the connection strings in the development Web.config file. In the deployed Web.config file, all the elements are removed.

Xml

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add xdt:Transform="RemoveAll" />
  </connectionStrings>
</configuration>
```

RemoveAttributes

Removes specified attributes from the selected elements.

▲ Syntax

```
Transform="RemoveAttributes(comma-delimited list of one or more attribute names)"
```

▲ Example

The following example shows how to select all the **compilation** elements in the development Web.config file. (Because there can be only one **compilation** element in the configuration file, you do not have to specify a **Locator** attribute.) In the deployed Web.config file, the **debug** and **batch** attributes are removed from the **compilation** element.

Xml

```
<configuration xmlns:xdt="...">
  <compilation
    xdt:Transform="RemoveAttributes(debug,batch)">
  </compilation>
</configuration>
```

▲ SetAttributes

Sets attributes for selected elements to the specified values. The **Replace** transform attribute replaces an entire element including all of its attributes. In contrast, the **SetAttributes** attribute enables you to leave the element as it is but change selected attributes. If you do not specify which attributes to change, all of the attributes that are present in the element in the transform file are changed.

The **SetAttributes** transform affects all selected elements. This behavior is different from the **Replace** transform attribute, which affects only the first selected element if multiple elements are selected.

▲ Syntax

```
Transform="SetAttributes(comma-delimited list of one or more attribute names)"
```

▲ Example

The following example shows how to select all the **compilation** elements in the development Web.config file. (Because there can be only one **compilation** element in the configuration file, you do not have to specify a **Locator** attribute.) In the deployed Web.config file, the value of the **compilation** element's **batch** attribute is set to **false**.

Xml

```
<configuration xmlns:xdt="...">
  <compilation
    batch="false"
    xdt:Transform="SetAttributes(batch)">
  </compilation>
</configuration>
```

▲ Omitting Locator Attributes

Locator attributes are optional. If you do not specify a **Locator** attribute, the element to be changed is specified implicitly by the element that the **Transform** attribute is specified for. In the following example, the entire **system.web** element is replaced, because no **Locator** attribute is specified to indicate otherwise.

Xml

```
<?xml version="1.0"?>
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
  <system.web xdt:Transform="Replace">
    <customErrors defaultRedirect="GenericError.htm"
      mode="RemoteOnly">
      <error statusCode="500" redirect="InternalServerError.htm"/>
    </customErrors>
  </system.web>
</configuration>
```

▲ Using Transform and Locator Attributes on Separate Elements

A **Transform** attribute does not have to be set in the same element as a **Locator** element. You can specify a **Locator** element on a parent element in order to select elements whose child elements you want to work with. You can then specify a **Transform** attribute in a child element in order to apply a change to the children.

The following example shows how to use the **Locator** attribute to select **location** elements for the specified path. However, only elements that are children of the selected **location** elements are transformed.

Xml

```
<configuration xmlns:xdt="...">
  <location path="C:\MySite\Admin" xdt:Locator="Match(path)">
    <system.web>
      <pages viewStateEncryptionMode="Always"
        xdt:Transform="SetAttributes(viewStateEncryptionMode)" />
    </system.web>
  </location>
</configuration>
```

If you specify a **Locator** attribute but you do not specify a **Transform** attribute in the same element or in a child element, no changes are made.

Note

A **Transform** attribute on a parent element can affect child elements even if no **Transform** is specified for them. For example, if you put the attribute `xdt:Transform="Replace"` in the **system.web** element, all the elements that are children of the **system.web** element are replaced with the content from the transform file.

See Also

Concepts

[Web Deployment Overview for Visual Studio and ASP.NET](#)

Other Resources

[Web Deployment Content Map for Visual Studio and ASP.NET](#)

[Web.config File Transformations \(tutorial on the ASP.NET site\)](#)

[SlowCheetah - XML Transforms for app.config and other XML files](#)