Frequently Asked Questions about XSLT

The following are frequently asked questions about XSLT.

Why do my transformations fail using Internet Explorer?

You might have an earlier version of MSXML running with Internet Explorer on your machine. MSXML versions 2.6 and earlier only support the XSL standard. MSXML versions 3.0 and later support XSLT 1.0. To use XSLT transformations, update the system by installing MSXML 3.0 or later.

You can install MSXML 3.0 in replace mode so that it becomes the default XML/XSLT processor for Internet Explorer 5.0 or 5.5.

You cannot, however, install MSXML 4.0 or later in such a way that it becomes the default XML/XSLT processor, because MSXML 4.0 and later can only be installed in side-by-side mode. Until Internet Explorer ships with MSXML 4.0 or later as its default XML parser, you can use MSXML 4.0 and later with Internet Explorer in the following ways only:

- **In Windows XP** - Although you cannot install MSXML 4.0 or later as the default parser, in Windows XP you can set MSXML as the default XML/XSLT processor by using a manifest. For more information, see MSXML and Windows [http://msdn.microsoft.com/en-us/library/ms760418(VS.85).aspx] .
- **For earlier versions of Windows** - The MSXML 4.0 and later features are available in Internet Explorer only via scripting. Embedded linking will not work for MSXML 4.0 or later with any current versions of Internet Explorer. For an example, see Initiate XSLT in a Script [http://msdn.microsoft.com/en-us/library/ms762796(VS.85).aspx] .

How do I run XSLT transformations programmatically?

XSLT transformations can be initiated programmatically using any language that supports COM interfaces. Such programming languages include Microsoft JScript, VBScript, Visual Basic, C++, and even Perl. To run XSLT transformations programmatically, you must create two XML DOM objects, one for the XML source document and the other for the XSLT style sheet. Then, you call the

```
transformNode()
```

function on the XML source document with the XSLT style sheet XML DOM object as the argument. The following example in JScript illustrates these points.

JScript File (test.js)

J# Copy Code

```
var xmlstr = "<object><name>apple</name><color>Red</color></object>";
var xsltstr = '<?xml version="1.0"?>' +
      '<xsl:stylesheet ' +
      '      version="1.0" ' +
      '      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> '+
      '   <xsl:output method="text"/> '+
      '   <xsl:template match="/object"> '+
      '      <xsl:value-of select="color"/> '+
      '      <xsl:text> </xsl:text> '+
      '      <xsl:value-of select="name"/> '+
      '   </xsl:template> '+
      '</xsl:stylesheet>';

var xmldom, xsltdom;
try {
   xmldom  = new ActiveXObject("Msxml2.DOMDocument.3.0");
```

```
   xmldom.validateOnParse = true;
   xmldom.async = false;
   xmldom.loadXML(xmlstr);

   xsltdom = new ActiveXObject("Msxml2.DOMDocument.3.0");
   xsltdom.validateOnParse = true;
   xsltdom.async = false;
   xsltdom.loadXML(xsltstr);

   output = xmldom.transformNode(xsltdom);
   WScript.echo(output);
}
catch(err) {
   WScript.echo(err.description);
}
```

Try It!

1. Copy and paste the code into a file, and save it as test.js.
2. Type the "test.js" command from a command window.

Output

The output is "Red apple".

Do I need to use a different XSLT namespace with Internet Explorer?

No. Use the standard

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

syntax. If Internet Explorer returns an error when you use this namespace, it is likely that you are using an earlier version of MSXML that does not support XSLT. If you have older XSL files and do not want to convert them to XSLT files, you can still use the namespace declaration

```
xmlns:xsl="http://www.w3.org/TR/WD-xsl"
```

. To convert XSL files to XSLT files, use the XSL to XSLT Converter1.1, available from MSDN Online Downloads at http://msdn.microsoft.com/downloads/.

I have installed a later version of Microsoft XML Core Services (MSXML). Why isn't my application using it?

If you embed your XSLT style sheet in your source XML document, later versions of MSXML will not be used without additional programming being used to access them by version-specific ProgIDs. For more information, see the FAQ question "Why do my transformations fail in Internet Explorer?" and its answer as discussed above.

If you call your transformation from a programming language, it is likely that you are not using the correct version-specific ProgID. You must use the version-dependent ProgID. The following Visual Basic code shows the correct syntax to create a

```
DOMDocument
```

object into which you load an XSLT file. The code uses the version-dependent ProgID.

Copy Code

```
Dim xsltDoc
Set xsltDoc = CreateObject("Msxml2.DOMDocument.6.0")
```

For more information about ProgIDs and syntax, see GUID and ProgID Information [http://msdn.microsoft.com/en-us/library/ms757837(VS.85).aspx] .

Does MSXML provide a 100% compliant XSLT processor?

Yes. MSXML is fully compliant with the XSLT specification. For more information, see Supported XSLT Features [http://msdn.microsoft.com/en-us/library/ms764682(VS.85).aspx] .

Does Internet Explorer 5.5 include a fully compliant XSLT processor?

Internet Explorer 5.5 is shipped with a version of MSXML that supports only XSL—not XSLT. However, you can use MSXML 6.0 with the help of client-side or server-side scripting. For more information, see Workarounds to Version Independence [http://msdn.microsoft.com/en-us/library/ms757057(VS.85).aspx] .

Why was the MSDN online documentation for XSL so different from the XSLT recommendation?

Because XSLT is a different technology from XSL.

Where can I find documentation about MSXML compliance?

Each new release of MSXML has a Bug List Page that describes known problems, such as coding mistakes or features that are not fully implemented. To find this page, search MSDN. For full conformance disclosure, see Supported XSLT Features [http://msdn.microsoft.com/en-us/library/ms764682(VS.85).aspx] .

If you find a bug or implementation point that is not clearly documented, please send feedback to the XML documentation team by using the XML Documentation Feedback form. To use this form, click the Feedback icon (the envelope) at the top-right corner of any page of this documentation.

How can I call MSXML from the command prompt to do batch processing of XSLT?

Because MSXML is a COM object, you can write VBScript, JScript, or other Windows Script Host (WSH) files to launch MSXML from the command prompt. Microsoft provides an XSLT command line utility, msxsl.exe, that performs command-line XSL transformations using the Microsoft XSL processor. Msxsl.exe is a small (~11K) command line utility that invokes Msxml5.dll to perform the actual work of the transformation. Msxsl.exe is available for download at the XML Developer Center at MSDN.

What can I do if I wrote lots of XSL using the old Internet Explorer version of XSL?

If you would like to upgrade XSL files so that they are compliant with the XSLT recommendation, you can use the XSL to XSLT 1.1 Converter available from MSDN Downloads at http://msdn.microsoft.com/downloads/.

I updated a style sheet I wrote previously to transform ADO-persisted XML recordsets. I now get an "undeclared reference to namespace prefix" error. What do I need to do to fix this?

The reason for this error involves namespaces that ActiveX Data Objects (ADO) 2.6 uses when you use the

```
adPersistXML
```

formatting option with the

```
Save
```

method to persist an ADO recordset as XML.

For example, the following line shows the syntax used to persist the current ADO recordset as XML in a Visual Basic application.

```
rs.Save "c:\temp\nwind.xml ", adPersistXML
```

The persisted XML output includes several namespace prefixes: "s", "dt", "rs", and "z". When you use XSLT style sheets to transform ADO-persisted XML, these namespaces must be declared for the transformation to succeed.

When using ADO to persist data as XML, you might encounter the following error text when you attempt to apply XSLT to your persisted XML.

Copy Code

```
  'undeclared reference to namespace prefix'
```

For example, a style sheet might reference the XSLT namespace URI as follows:

Xml Copy Code

```
<xsl:stylesheet version="1.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <xsl:element name="catalog">
            <xsl:value-of select="//z:row/@isbnnumber"/>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

This example, though valid as XSLT, produces the undeclared namespace error described above. To fix the problem, style sheets that specify the XSLT namespace URI must also declare any of the ADO namespaces that are used as input.

The following example shows how to update the previous style sheet code to declare these namespaces.

Xml Copy Code

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:s='uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882'
        xmlns:dt='uuid:C2F41010-65B3-11d1-A29F-00AA00C14882'
        xmlns:rs='urn:schemas-microsoft-com:rowset'
        xmlns:z='#RowsetSchema' version='1.0'>
    <xsl:template match="/">
        <xsl:element name="catalog">
            <xsl:value-of select="//z:row/@isbnnumber"/>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

By adding the ADO namespace declarations above to your style sheet, you can correct the error and permit it to successfully transform ADO-persisted XML.

Style sheets that reference the older XSL namespace (like the following sample) do not produce this error.

Xml Copy Code

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
    <xsl:template match="/">
        <xsl:element name="catalog">
            <xsl:value-of select="//z:row/@isbnnumber"/>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

Therefore, these style sheets are not affected and do not require any changes for this issue.

Will MSXML support XSLT 2.0?

No. MSXML versions 4.0 and later fully implement and support XSL Transformations (XSLT) Version 1.0 (W3C Recommendation 16 November 1999). If your XML application requires a later version of XSLT, Microsoft strongly recommends moving to the newer System.Xml framework classes, because all future XML development efforts will be focused there.