



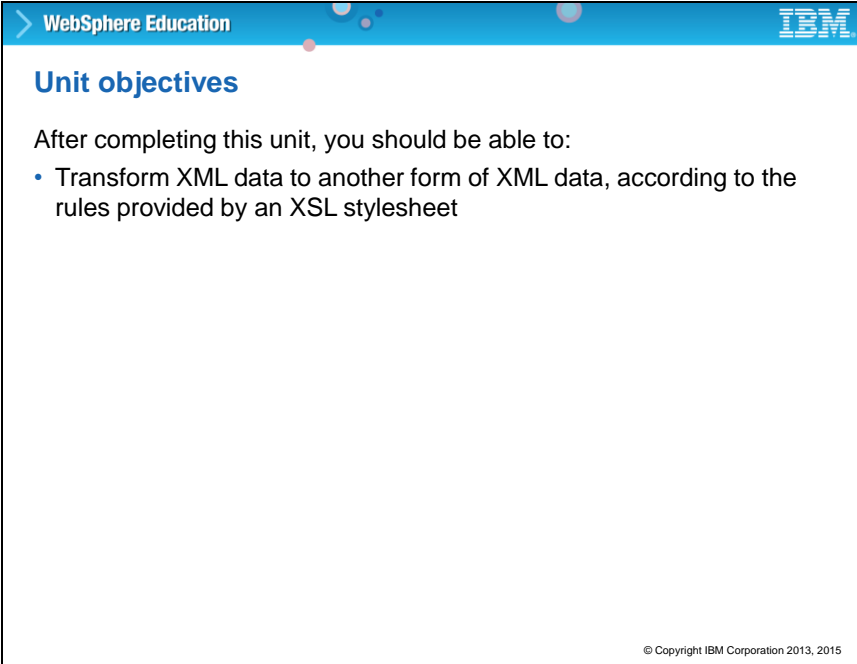
Slide 1


> WebSphere Education 

Transforming data with XSL stylesheets



© Copyright IBM Corporation 2013, 2015
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

A presentation slide with a blue header bar containing the text 'WebSphere Education' and the IBM logo. The main content area is white and contains the title 'Unit objectives' in blue, followed by a paragraph and a bulleted list. A small copyright notice is at the bottom right.

> WebSphere Education 

Unit objectives

After completing this unit, you should be able to:

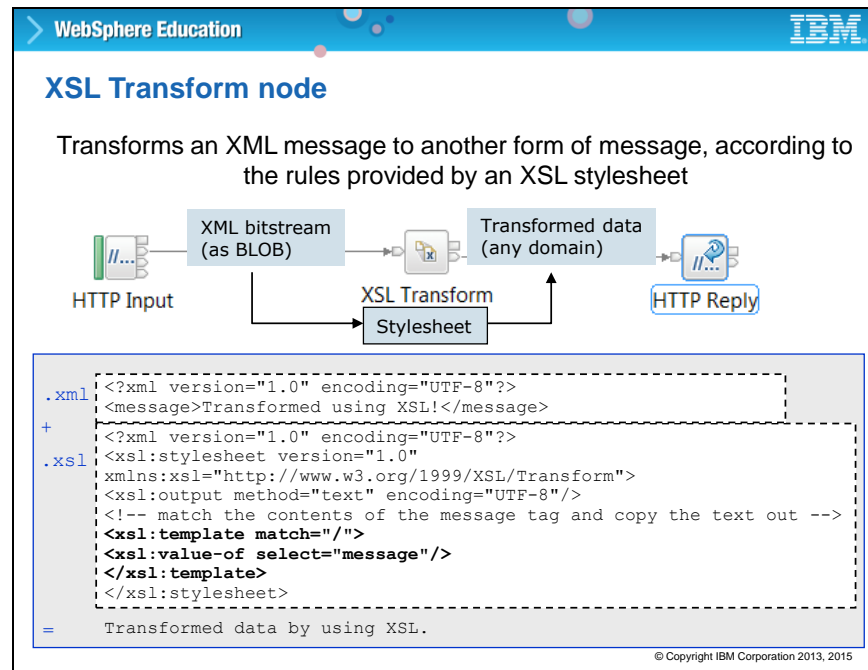
- Transform XML data to another form of XML data, according to the rules provided by an XSL stylesheet

© Copyright IBM Corporation 2013, 2015

Unit objectives

You can use the XSL Transform node to transform an XML message to another form of message, according to the rules provided by an Extensible Stylesheet Language (XSL) stylesheet. This unit describes how to use XSL stylesheets to transform data.

After completing this unit, you should be able to transform XML data to another form of XML data, according to the rules provided by an XSL stylesheet.



XSL Transform node

Extensible Stylesheet Language (XSL) is an XML-based language that is used to transform XML documents into other XML or “human-readable” documents.



The original document is not changed; rather, a new document is created based on the content of an existing one. The new document can be output by the processor in standard XML syntax or in another format.

The Integration Bus XSL Transform node transforms an XML message according to the rules provided by an XSLT stylesheet.

It is imperative that the data can be parsed into an XML message. The stylesheet, that uses the rules that are defined within it, can complete the following actions:

- Sort the data.
- Select data elements to include or exclude based on some criteria.
- Transform the data into another format.

The figure shows an example of an XSL Transform node in a message flow and an example of a stylesheet.

 WebSphere Education 

XSL Transform node properties

- **Stylesheet name** is the name of principal stylesheet
 - Click **Browse** to select stylesheet and automatically add it to the BAR file when you add a message flow to a BAR file
 - To identify a deployed stylesheet, set the **stylesheet name** property and leave the **stylesheet directory** property blank.
 - In the Message Flow editor, drag an `.xslt` file onto the XSLTransform node to set the stylesheet name automatically.
- **Advanced properties**
 - **Stylesheet directory** is the path where the stylesheet is located
 - **Stylesheet cache level** is the number of compiled or parsed stylesheets that are stored in this instance of the node
- To ensure that the XSLTransform node transforms the message correctly, set the **Retain mixed content**, **Retain comments**, and **Retain processing instructions** properties on the preceding node

© Copyright IBM Corporation 2013, 2015

XSL Transform node properties

The XSL Transform node properties identify the XSL stylesheet to use to transform the message and the message domain, message model, message, and physical format for the generated message.

In the XSL properties, you can specify the stylesheet name and directory by selecting it from the file system or dragging an `.xslt` file onto the XSL Transform node. If the stylesheet is already deployed or is deployed separately, set the **stylesheet name** property but leave the **stylesheet directory** property blank.

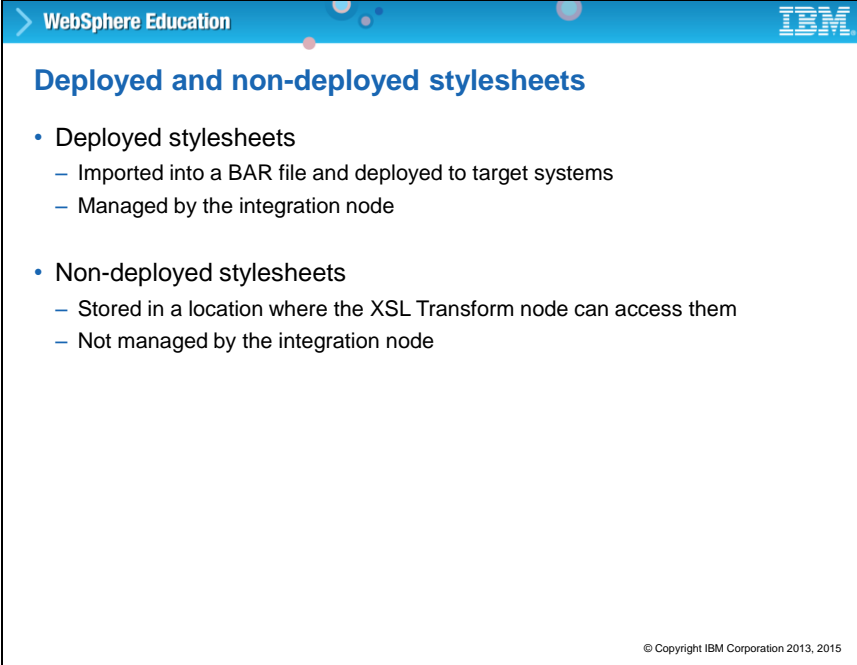
If the stylesheet is not embedded within the message, an XSLT compiler is used for the transformation. Compiled stylesheets can be stored in a stylesheet cache to improve performance.

The stylesheet cache is retained for the life of the node, and is cleared when the node is deleted from the flow, when the flow is deleted, or when the integration server is stopped. If you change a cached stylesheet by redeploying or replacing the file in the file system, the XSLTransform node that is holding the cache replaces the cached version with the most recent version before a new message is processed.

The **stylesheet cache level** property sets the number of stylesheets that are stored in this instance of the node. If you set this property to zero, no stylesheet is cached, and stylesheets are interpreted rather than compiled.

Consider performance when you set this property. Typically, when you set this property to a number greater than the default value of 5, performance is faster because stylesheet recompilation is less likely. However, cached stylesheets use Java virtual machine (JVM) heap space; if you keep too many cached stylesheets, the JVM is likely to slow. In addition, if the cached stylesheets are not used regularly and you set this property to a large number, performance can be affected because compiling stylesheets increases processor usage. To decide on the most suitable value for this property, you must balance how many stylesheets you use with the size of the stylesheets, the size of the JVM heap space, and your usage pattern.

If the input to the XSL Transform node is generated from the XMLNSC parser, the XMLNSC parser discards certain information in XML documents, such as processing instructions and comments. However, you can set properties to retain this information in a preceding node. To ensure that the XSLTransform node transforms the message correctly, set the **Retain mixed content**, **Retain comments**, and **Retain processing instructions** properties correctly on the node that sends the message to the XSL Transform node.



The slide is titled "Deployed and non-deployed stylesheets" and is part of a WebSphere Education presentation. It contains a bulleted list with two main categories: "Deployed stylesheets" and "Non-deployed stylesheets". Each category has two sub-points. The slide also features the IBM logo in the top right corner and a copyright notice at the bottom right.

- Deployed stylesheets
 - Imported into a BAR file and deployed to target systems
 - Managed by the integration node
- Non-deployed stylesheets
 - Stored in a location where the XSL Transform node can access them
 - Not managed by the integration node

© Copyright IBM Corporation 2013, 2015

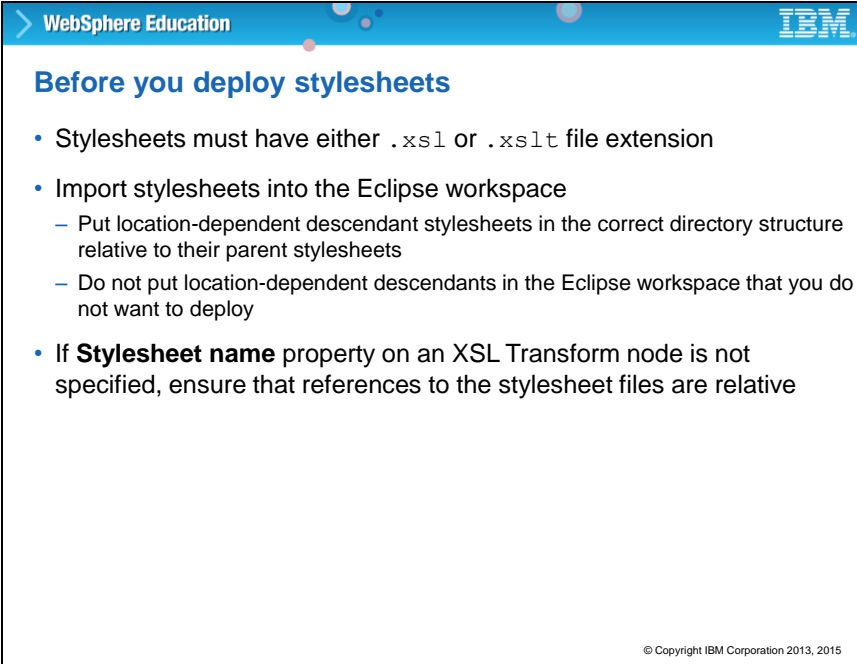
Deployed and non-deployed stylesheets

You can use stylesheets in two different ways with the XSLTransform node.


Deployed stylesheets are stylesheets that you import into a BAR file and deploy to target systems. The integration node manages deployed stylesheets.

Non-deployed stylesheets are stylesheets that you store in a location where the XSL Transform node can access them. The integration node does not manage non-deployed stylesheets.

To change a stylesheet that is deployed in a BAR file, you must redeploy the BAR file. If you are changing several stylesheets, stop the relevant message flows before you change it. If you do not stop the relevant message flows before you change the stylesheet, the order of the changes cannot be guaranteed by running message flows, which might cause an incompatibility between the stylesheets that are changed. Use the `mqxsireload` command to reload a stylesheet.



The slide is titled "Before you deploy stylesheets" and is part of a WebSphere Education presentation. It contains a bulleted list of three main points regarding stylesheet deployment prerequisites. The first point states that stylesheets must have a .xsl or .xslt file extension. The second point, "Import stylesheets into the Eclipse workspace", has two sub-points: one about directory structure and another about not putting location-dependent descendants in the workspace if they are not to be deployed. The third point states that if the "Stylesheet name" property on an XSL Transform node is not specified, references to stylesheet files must be relative. The IBM logo is in the top right corner, and a copyright notice for IBM Corporation 2013, 2015 is at the bottom right.

WebSphere Education 

Before you deploy stylesheets

- Stylesheets must have either `.xsl` or `.xslt` file extension
- Import stylesheets into the Eclipse workspace
 - Put location-dependent descendant stylesheets in the correct directory structure relative to their parent stylesheets
 - Do not put location-dependent descendants in the Eclipse workspace that you do not want to deploy
- If **Stylesheet name** property on an XSL Transform node is not specified, ensure that references to the stylesheet files are relative

© Copyright IBM Corporation 2013, 2015



Before you deploy stylesheets

If you are going to deploy the stylesheets in the BAR file, you have some prerequisite steps to complete.

First, make sure that the files have the correct file name extensions. Deployed stylesheets must have either `.xsl` or `.xslt` as their file extension.

Next, import all stylesheets that are going to be deployed into the Toolkit workspace. Put location-dependent descendant stylesheets in the correct directory structure relative to their parent stylesheets.

Finally, ensure that any reference to a principal stylesheet is relative to the root of the relevant Toolkit workspace. The only exception is when you specify a principal stylesheet as the **stylesheet name** property on an XSL Transform node. You can use an absolute path that points to the correct directory structure in the workspace. If the principal stylesheet is found, the system resets the node property automatically to the correct relative value.

 WebSphere Education 

Stylesheet search order

XSL Transform node searches for the XSL file in the following order:

1. Dynamically from the input message
Example: `<?xml-stylesheet type="text/xsl" href="abc.xsl"?>`
2. Dynamically from the `XSL.StyleSheetName` local environment variable
 - Allows for dynamic selection of the stylesheet from within the message flow
 - Can apply multiple stylesheets to one message
3. Statically from **Stylesheet name** and **Stylesheet directory** in the XSL Transform node properties

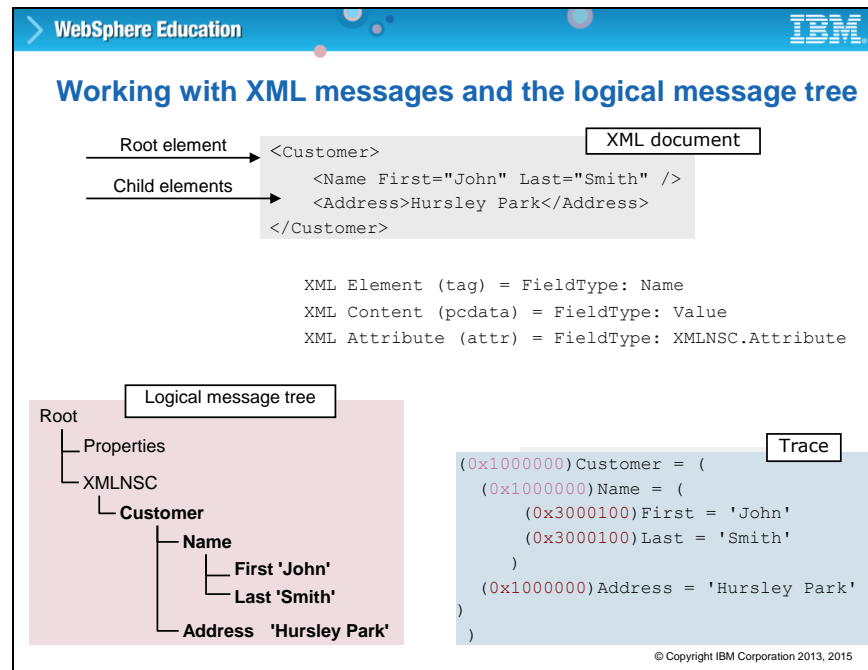
© Copyright IBM Corporation 2013, 2015

Stylesheet search order

You can specify the location of the stylesheet to apply to this transformation in three ways.

- Use the content of the XML data within the message itself to transform the message according to a stylesheet that the message itself defines.
- Set a value in the LocalEnvironment folder. You must set this value in a node that precedes the XSLTransform node. With this technique, you can use various inputs to determine which stylesheet to use for this message, such as the content of the message data, or a value in a database.
- Use node properties to ensure that the transformation this single stylesheet defines is applied to every message that this node processes.

The XSL Transform node searches for reference to a stylesheet in this order until a stylesheet is found.





Working with XML messages and the logical message tree

The XSL Transform node serializes the message body, gets the XML data from the message, and applies the stylesheet to the XML data to produce new data.

This slide shows an example of the data and the logical message tree that describes it. It also contains a portion of a trace file.

If you copy a message tree from input to output without changing the domain, most of the syntax elements are created by the parser and the field types are correct. However, if you construct your message tree from a database query or in a Compute node for example, you must ensure that you identify each syntax element correctly by using its field type.

 WebSphere Education 

Determining the output message format

- XSL Transform node searches for the message domain, message model, message type, and message format to use for the output message by interrogating, in the following order:
 1. Dynamically from `XSL.MessageDomain`, `XSL.MessageSet`, `XSL.MessageType`, and `XSL.MessageFormat`. local environment variables
 2. XSL Transform **Output Message Parsing** node properties
- BLOB domain is used if the node cannot determine the message domain from the `XSL.MessageDomain` environment variable or the **Message domain** property

© Copyright IBM Corporation 2013, 2015

Determining the output message format

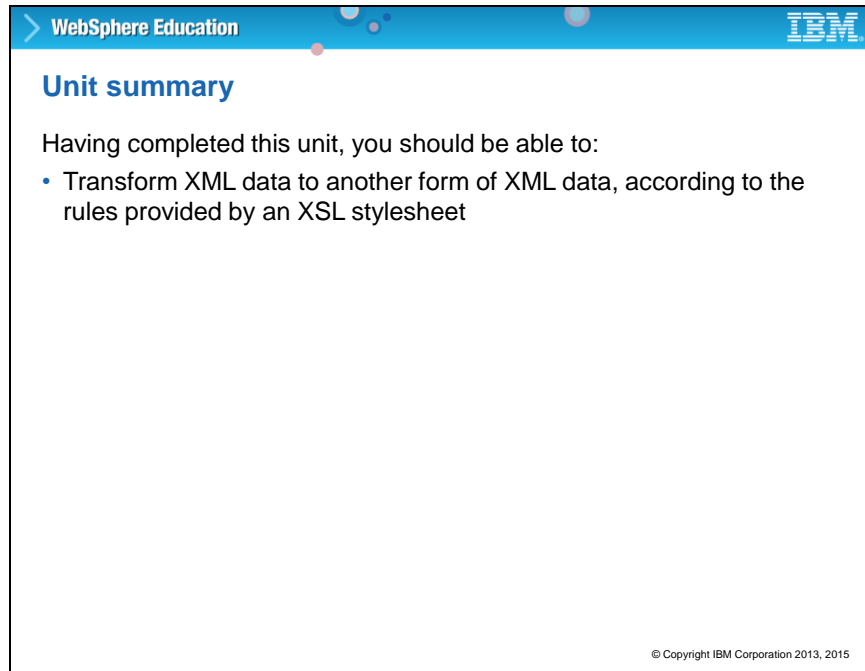
When determining the format for the output message, the XSL Transform node searches for the message domain, message model, message type, and message format.

First, the XSL Transform node searches the local environment variables in the message tree.

If the local environment does not identify the message property, the XSL Transform node uses the properties that are configured in the node.

If either the local environment variables or the XSL Transform node properties cannot determine the domain, the BLOB domain is used.

An example of a message flow with the XSL Transform node is provided in the Integration Toolkit Tutorials Gallery.

The slide features a blue header bar with a white right-pointing arrow, the text 'WebSphere Education', and the IBM logo. The main content area is white with a black border. It contains the title 'Unit summary' in blue, followed by a paragraph 'Having completed this unit, you should be able to:' and a bulleted list item. A small copyright notice is at the bottom right.

> WebSphere Education IBM

Unit summary

Having completed this unit, you should be able to:

- Transform XML data to another form of XML data, according to the rules provided by an XSL stylesheet

© Copyright IBM Corporation 2013, 2015

Unit summary

You can use the XSL Transform node to transform an XML message to another form of message, according to the rules provided by an Extensible Stylesheet Language (XSL) stylesheet. This unit described how to use XSL stylesheets to transform data.

Having completed this unit, you should be able to transform XML data to another form of XML data, according to the rules provided by an XSL stylesheet.