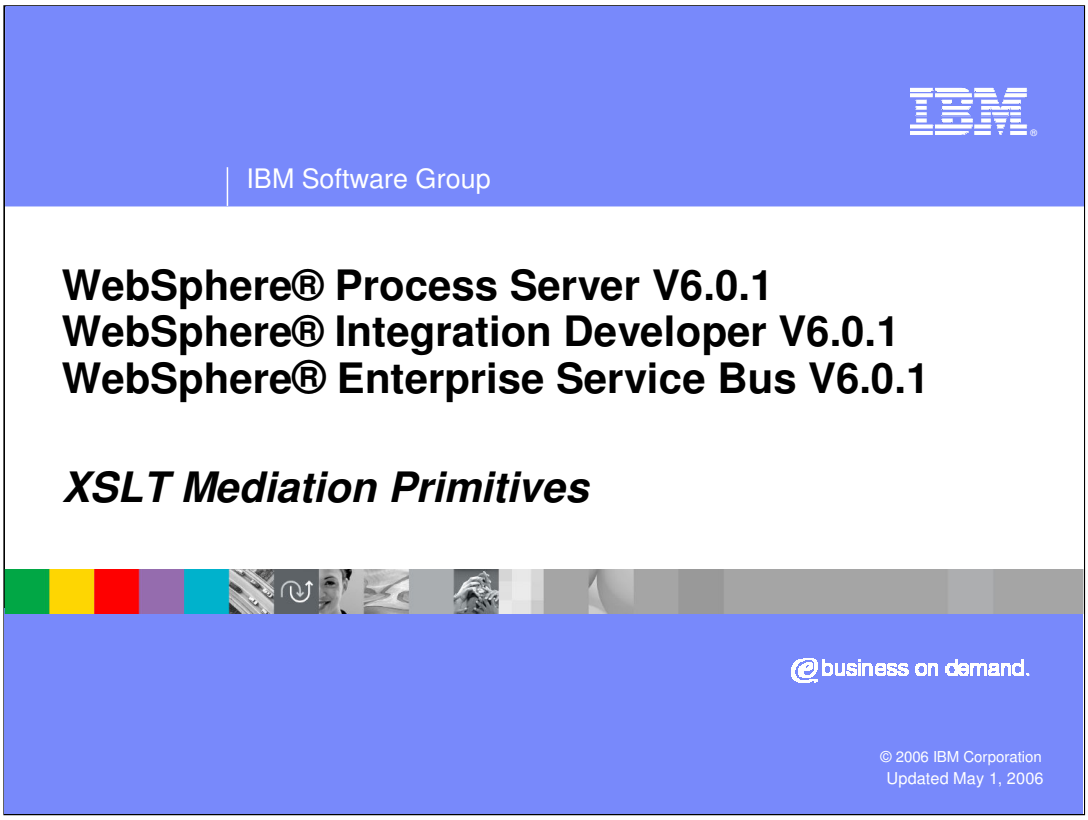


This presentation will provide a detailed look at the Extensible Stylesheet Language Transformation (XSLT) mediation primitive.

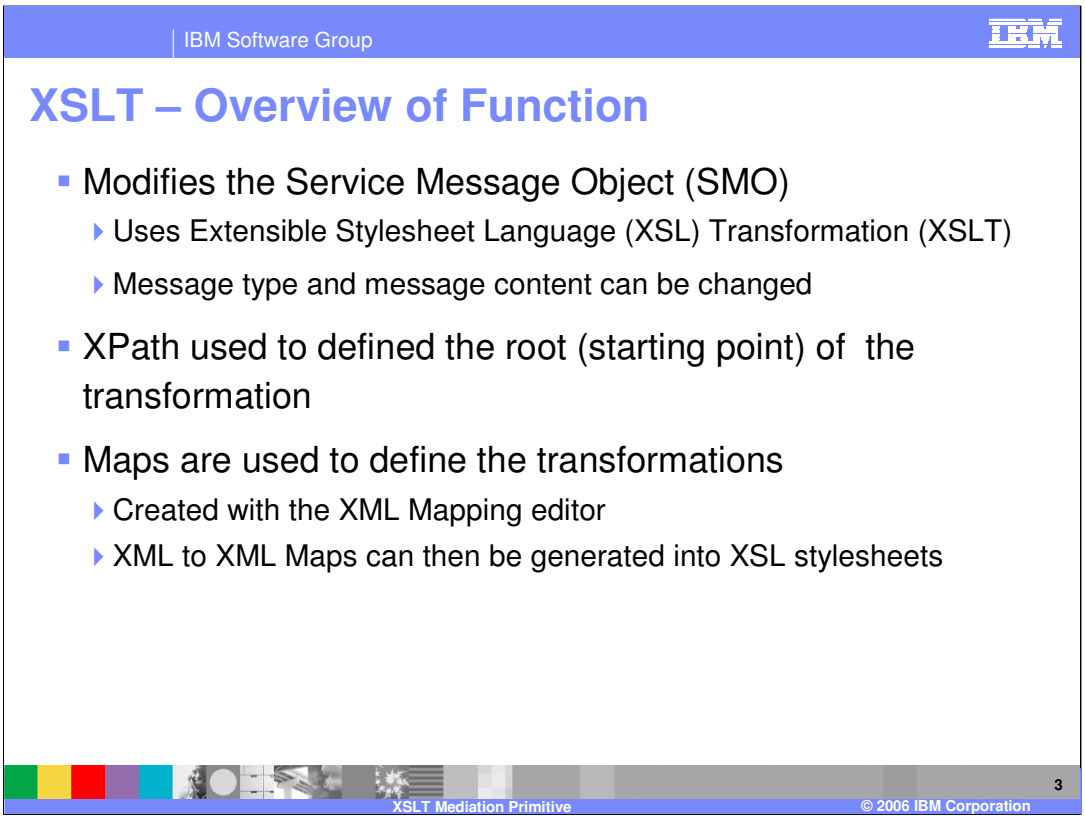
Goals

- Understand the XSLT mediation primitive

XSLT

- Overview of function
- Use of terminals
- Definition of properties
- Considerations for editing properties
- Error handling
- Example usage

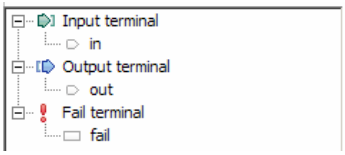
The goal of this presentation is to provide you with a full understanding of the XSLT mediation primitive. It is assumed that you are already familiar with the material presented in the **Mediation Primitive Common Details** presentation, which serves as a base for understanding mediation primitives in general. An overview of the XSLT primitive is presented along with information about the primitive's use of terminals and its properties. The editing of properties for the XSLT primitive is presented in some detail, along with some unique considerations that you must be familiar with. Finally, error handling characteristics are presented and an example usage of a XSLT primitive is provided.



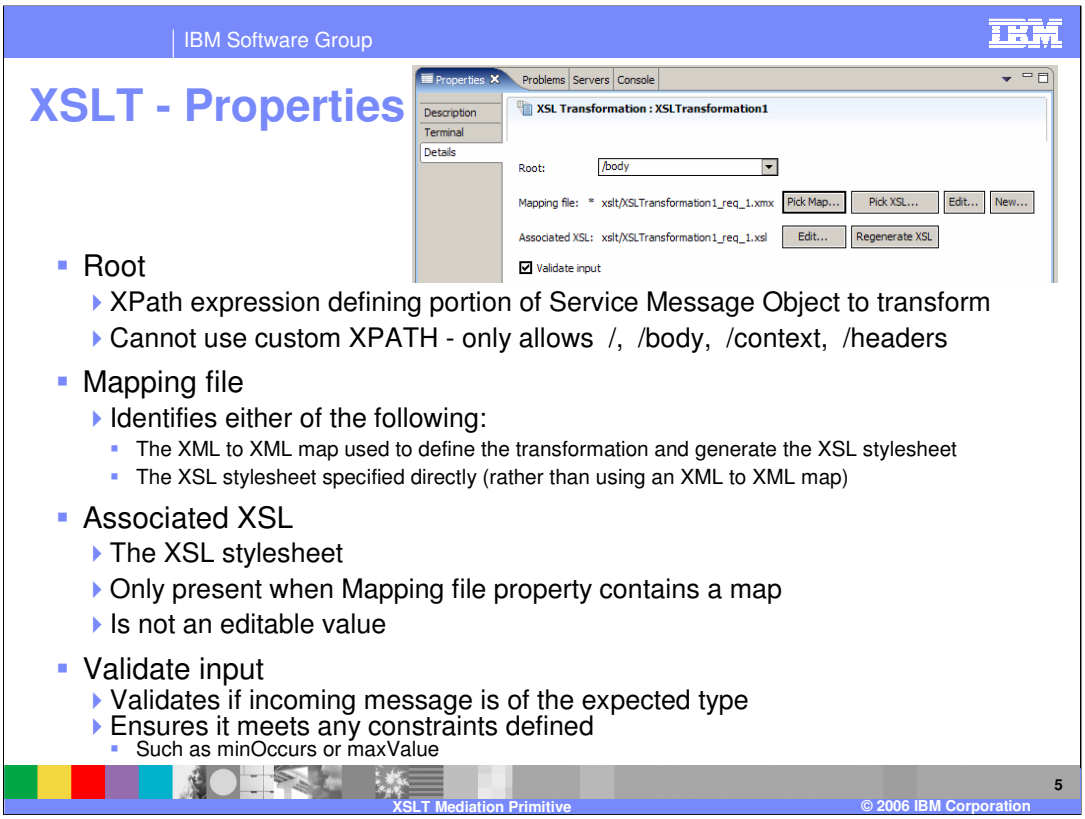
The purpose of the XSLT primitive is to modify the Service Message Object, using Extensible Stylesheet Language Transformation (XSL Transformation or simply XSLT). The XSLT primitive is capable of modifying the content of the SMO and is also able to modify the message type by restructuring the message body. There is a root property, which uses an XPath expression to define the starting point within the SMO for the transformation. Transformations are defined by means of maps created using the XML Mapping Editor. The XML to XML maps are then generated into XSL style sheets, which are used by the runtime when performing the transformation.

XSLT - Terminals

- Terminals:
  - Input terminal
  - One Output terminal
  - Fail terminal
- Message type of Input and Output terminal
  - Same type – for manipulation of values within a message
  - Different type – for changing format of the message body



The XSLT primitive has one input terminal, one output terminal and a fail terminal. The output terminal can be for the same message type as the input terminal or for a different message type. When the message type is different, the transformation modifies the structure of the body of the message. Shown here is an XSLT primitive with its terminals and the terminals as seen in the properties view.



In the upper right is a screen capture of the Details tab from the Properties view for an XSLT primitive showing the following properties:

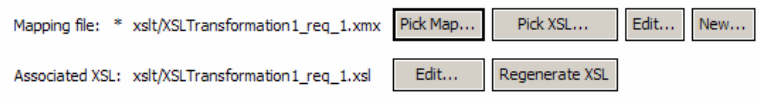
•**Root** contains an XPath expression defining the portion of the SMO on which to perform the transformation. This is restricted to the major sections of the SMO and can be set to / (slash) meaning the entire SMO or /**body**, /**context** or /**header** referring to the associated section of the SMO.

•**Mapping file** contains the name of a file. This could be the name of an XML to XML map, which defines the transformation and will be used to generate an XSL stylesheet. Alternatively, it could be the name of the XSL stylesheet to be used to perform the transformation.

•**Associated XSL** contains the name of the XSL stylesheet that will be used to perform the transformation. This property is only present when the Mapping file property contains an XML to XML mapping file and will not be present when the Mapping file property contains an XSL stylesheet. This property value is not editable.

•**Validate input** is a checkbox used to indicate if incoming messages to the XSLT primitive are to be validated prior to processing. This will ensure that the incoming message is of the expected type and that any constraints defined are not violated.

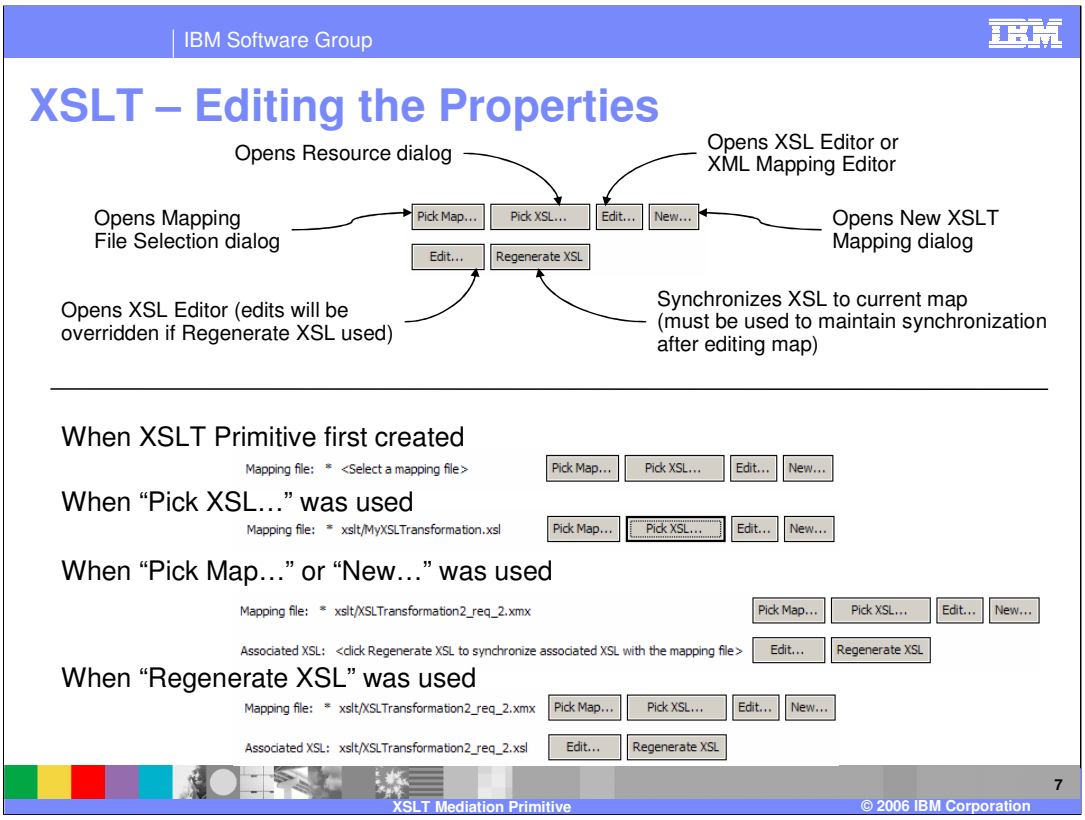
XSLT – Editing the Properties



- Editing Mapping file and Associated XSL properties
  - Handling of these properties needs to be understood to avoid problems
  - The Mapping file property can contain either:
    - an XML to XML map
    - an XSL stylesheet
  - Associated XSL won't be present if Mapping file is an XSL stylesheet
  - Synchronization of the XML to XML map and generated XSL stylesheet
    - The map and XSL stylesheet are not automatically synchronized
    - The XSL stylesheet must be explicitly regenerated after mapping changes
    - The XSL stylesheet can be directly edited after it is generated

The next few slides examine considerations for editing the properties of an XSLT primitive, in particular the Mapping file and Associated XSL properties. You must understand how these properties are manipulated and managed to avoid encountering problems when the mapping definition used at runtime is different from the one you believe you have defined.

As previously discussed, the Mapping file property might contain an XML to XML map or an XSL stylesheet. The Associated XSL property will not be present when the Mapping file is an XSL stylesheet. When using an XML to XML map, the XSL stylesheet is generated from the map, but the generation step is not done automatically. This can easily lead to the map and the stylesheet being out of synch. You must explicitly regenerate the XSL stylesheet whenever you make a mapping change. If you don't regenerate, the mapping change will not be reflected in the XSL stylesheet or used at runtime to perform the transformation. Another potential problem can arise as a result of the capability to edit the generated XSL stylesheet directly without making an update to the map. This change will be reflected at runtime, however, if you later make a map change and regenerate the stylesheet, the direct edit of the stylesheet will be lost.



The top portion of this slide shows the buttons related to the **Mapping file** and **Associated XSL** properties. The top row of buttons is used with the Mapping file property, while the bottom row is used with the Associated XSL property.

The **New…** button will open the New XSLT Mapping dialog, which leads you through the creation of an XML to XML map and puts you into the XML Mapping Editor for defining the map.

The **Edit…** button in the top row will open the XML Mapping Editor if the Mapping file property contains a map, or the XSL Editor if it contains an XSL stylesheet.

The **Pick XSL…** button will open a Resource dialog, which is a generic dialog for finding any resource and can be used to select an XSL stylesheet.

The **Pick Map…** button will open a Mapping File Selection dialog, which enables you to select an XML to XML map.

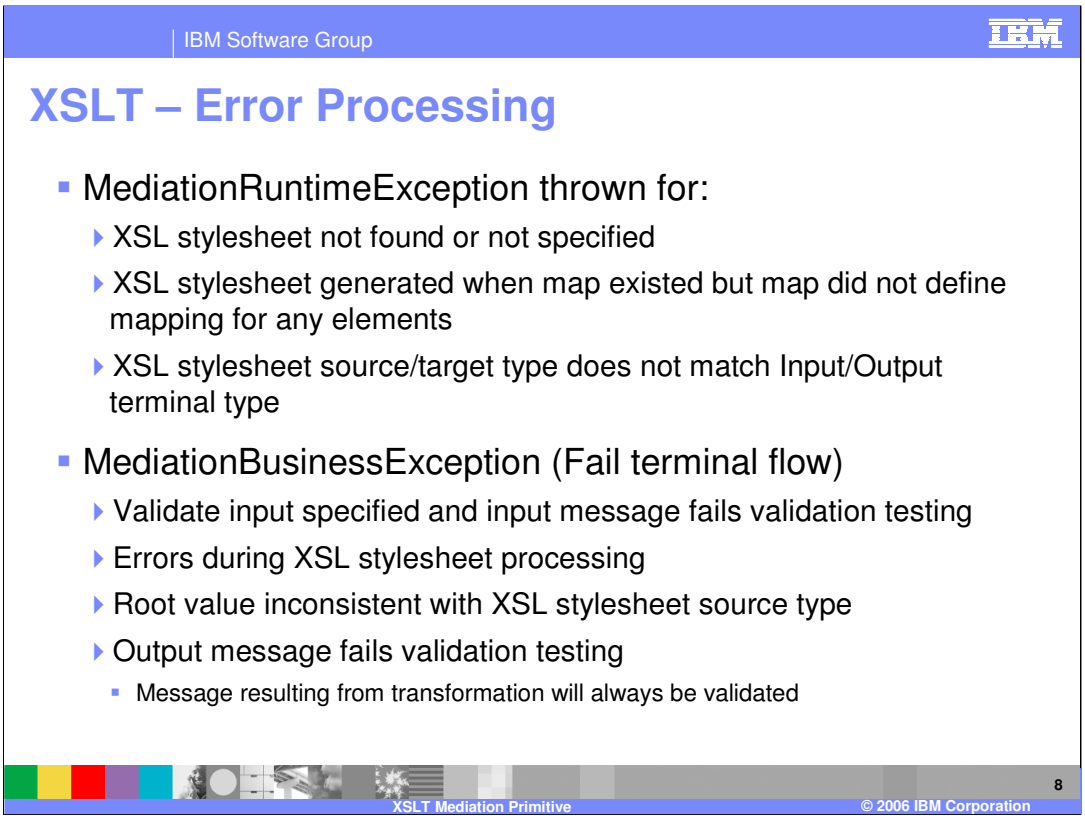
The **Regenerate XSL** button is used to regenerate the XSL stylesheet based on the current definition of the XML to XML map and must be used after editing a map to maintain synchronization between the map and the stylesheet

The **Edit…** button in the bottom row opens the XSL Editor and allows you to edit an XSL stylesheet that was previously generated from the specified XML to XML map.

In order to better illustrate the handling of these properties, the bottom portion of the slide steps through a scenario of using these buttons to manipulate the properties.

When an XSLT primitive is first created, only the Mapping file property shows along with its 4 buttons and the Mapping file value indicates you need to select a mapping file.

Assuming that the **Pick XSL…** button was then used, it is still only the Mapping file property that shows, but its value is now set to the XSL stylesheet that you selected



The error processing details and considerations are examined in this slide.

A MediationRuntimeException will be thrown for problems accessing the XSL stylesheet, such as when the stylesheet cannot be found or has not been specified. This exception will also be thrown for an XSL stylesheet that does not contain any transformation operation. This can occur if it was generated from a map that did not specify the mapping for any elements. The MediationRuntimeException is thrown if the XSL stylesheet source and target types do not match the terminal message types.

A MediationBusinessException occurs for several different problems including the following:

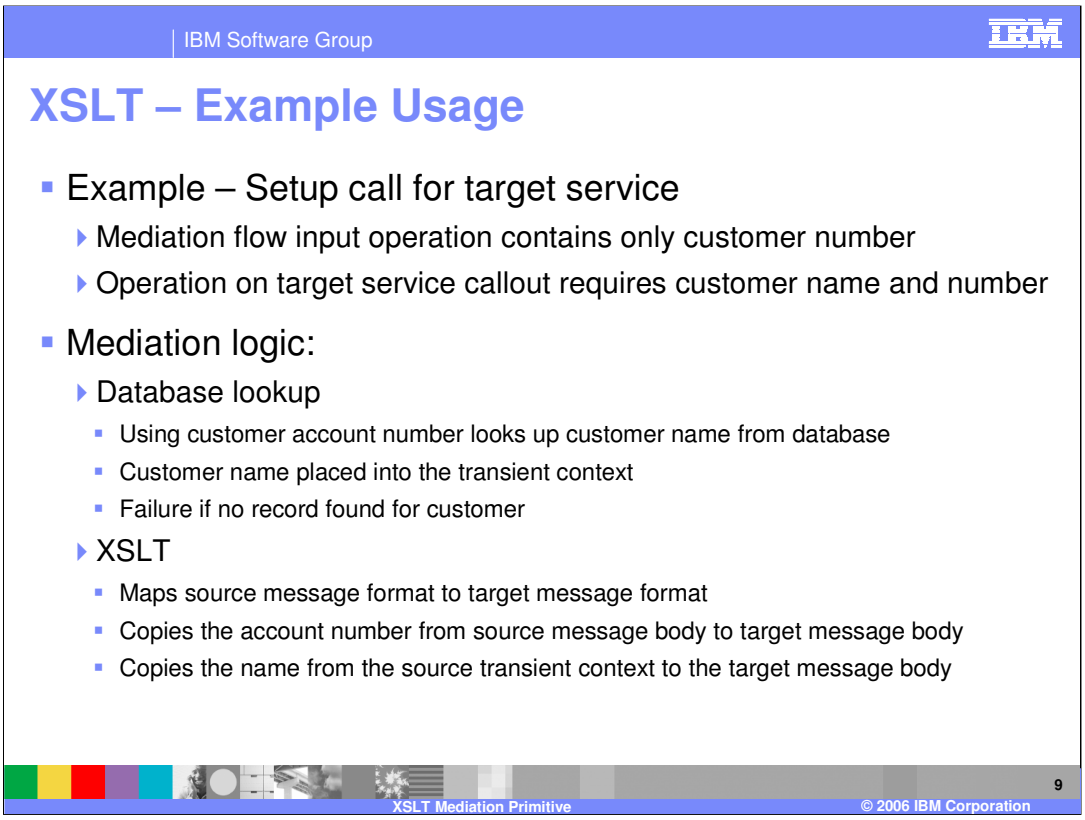
•Validate input has been specified and the message fails the validation processing.

•Errors during the processing of the XSL stylesheet.

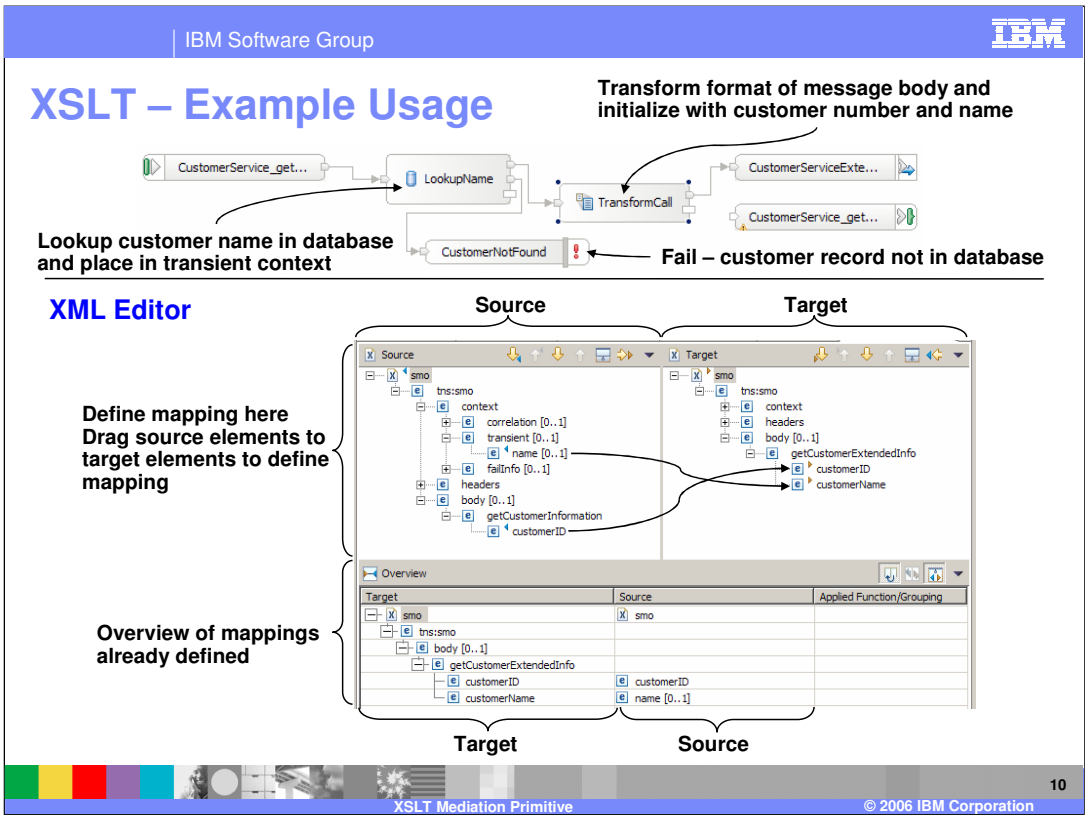
•Root property value is inconsistent with the XSL stylesheet source type.

•Output message fails validation testing. Unlike the optional validation of the input message, which is based on a user specified property, the validation of the output message will always occur.

In all of these MediationBusinessException cases, if the Fail terminal is wired, the flow from the Fail terminal will be followed rather than the exception being thrown.



The next couple of slides provide an example usage of the XSLT primitive. In this scenario, a call to a service provider requires more information than the call coming in from the service requestor. Specifically, the request contains only a customer number, and the service provider has in interface requiring both a customer name and customer number. This is done using an XSLT primitive in conjunction with a Database Lookup primitive. The mediation flow logic starts with the Database Lookup, which uses the customer number to look up a customer name from a database. The customer name is placed into the transient context of the SMO by the Database Lookup. Since the service provider always needs both the customer name and number it is considered an error condition if the lookup does not find the customer record. Following the Database Lookup, the XSLT primitive maps the source message to the target message, changing the message type to match the operation being called on the service provider. The account number is moved from the body of the source SMO to the body of the target SMO, while the name is moved from the transient context of the source SMO to the body of the target SMO.



The top portion of this slide shows the mediation flow logic for the example scenario. The input node is wired to a Database Lookup, which looks up the customer name using the customer number as the key. The keyNotFound terminal is wired to a Fail primitive which will throw an exception. If the lookup is successful, the flow goes to an XSLT primitive, which modifies the SMO to the format required by the service provider. The bottom portion of the slide shows the XML Mapping editor. The top portion of the editor is where the transformations are defined and the bottom portion shows the existing mappings. Notice that the customer ID has been moved from the body of the source to the body of the target, whereas the customer name has been moved from the transient context of the source to the body of the target, and that the source and target bodies contain different elements.

Summary

- Examined the XSLT mediation primitive

XSLT

  - Overview of function
  - Use of terminals
  - Definition of properties
  - Considerations for editing properties
  - Error handling
  - Example usage

In summary, this presentation provided an overview of the XSLT primitive along with information about the primitive's use of terminals and its properties. The editing of properties for the XSLT primitive were presented in some detail. Finally, error handing characteristics were presented and an example usage of an XSLT primitive was provided.

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.