



IBM Software Group

WebSphere® Enterprise Service Bus V6.0.2 WebSphere Process Server V6.0.2 WebSphere Integration Developer V6.0.2

XSLT mediation primitive



@business on demand.

© 2007 IBM Corporation
Updated April 12, 2007

This presentation provides a detailed look at the XSLT mediation primitive. The acronym XSLT stands for extensible style sheet language transformation, which is defined in a specification from the World Wide Web Consortium (W3C). The XSLT specification defines the syntax and semantics of XSLT, which is a language for transforming XML documents into other XML documents.

Goals

- Understand the XSLT mediation primitive



XSLT

- ▶ Overview of function
- ▶ Use of terminals
- ▶ Definition of properties
- ▶ Special considerations for properties
- ▶ Error handling
- ▶ Example usage

2

XSLT mediation primitive

© 2007 IBM Corporation

The goal of this presentation is to provide you with a full understanding of the XSLT mediation primitive.

The presentation assumes that you are already familiar with the material presented in the **Mediation primitive common details** presentation and the **Common details – Promoted properties** presentation. These two presentations serve 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. Starting in version 6.0.2, the handling of properties was enhanced. This was an overall improvement in behavior of the primitive, but it also introduced some special considerations in some circumstances. These special considerations are reviewed with pointers to more detailed information provided. Finally, error handling characteristics are presented followed by an example usage of an XSLT primitive.

Overview of function

- Modifies the service message object (SMO)
 - ▶ Uses extensible style sheet language (XSL) Transformation (XSLT)
 - ▶ Message type and message content can be changed
 - ▶ Transformed message is validated
- XPath used to defined the root (starting point) of the transformation
- Maps are used to define the transformations
 - ▶ Created with the XSLT mapping editor
 - ▶ XSL style sheets are generated from the XML maps

3

XSLT mediation primitive

© 2007 IBM Corporation

The purpose of the XSLT primitive is to modify the service message object (SMO). The XSLT primitive is capable of modifying the content of the SMO and of modifying the message type by restructuring the message body.

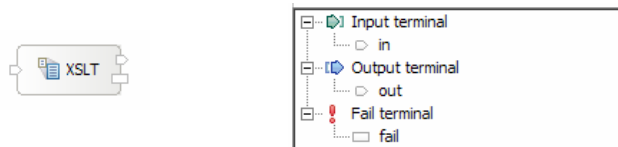
The transformed message is validated to ensure it conforms to the constraints specified for it. This validation always occurs and is not optionally selected as it is for input messages on this and other primitives.

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 XML maps created using the XSLT mapping editor. The XML maps are then generated into XSL style sheets, which are used by the runtime when performing the transformation.

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.

IBM Software Group

Properties

Properties x Problems Servers

Description

Terminal

Details

Promoted Properties

XSL Transformation : XForm1

Root: /body

Mapping file: * xslt/XSLTransformation1_req_1.xmx Browse... Edit... New...

Associated XSL: xslt/XSLTransformation1_req_1.xsl

Validate input

- Root
 - ▶ XPath expression defining portion of service message object to transform
 - ▶ One of: /, /body, /context, /headers
 - ▶ Not editable on this panel – it is specified in the new XSLT mapping dialog
- Mapping file
 - ▶ Identifies the XML map used to define the transformation
 - ▶ Browse... to select an existing map using the mapping file selection dialog
 - ▶ Edit... to edit the existing map using the XSLT mapping editor
 - ▶ New... to create a new map using the new XSLT mapping dialog
- Associated XSL
 - ▶ The XSL style sheet
 - ▶ File is automatically generated and updated when the mapping file is saved
- Validate input
 - ▶ Validates if incoming message is of the expected type
 - ▶ Ensures it meets any constraints defined
 - Such as minOccurs or maxValue

5

XSLT mediation primitive © 2007 IBM Corporation

In the upper right is a screen capture of the Details panel from the Properties view for an XSLT primitive.

The **Root** property contains an XPath expression defining the major portion of the SMO on which the transformation is performed. Valid values for this property are / (slash) meaning the entire SMO, or /body, /context or /header, referring to the corresponding section of the SMO. This property is not editable on this panel, rather its value is set in the New XSLT Mapping dialog.

The **Mapping file** property contains the file name of an XML map that is used to define the transformation. This field displays the file name and is not directly editable. It is set through use of the dialogs accessible using the buttons to the right of the field. The Browse... button opens the Mapping File Selection dialog that allows you to navigate and select an existing XML map. The Edit... button opens the XSLT Mapping Editor to allow you to edit the currently configured map. The New... button opens the New XSLT Mapping dialog which enables you to define the characteristics of the map, and then places you into the XSLT Mapping Editor to define the transformation.

The **Associated XSL** property contains the name of the XSL style sheet that is used by the runtime when performing the transformation. The XSL style sheet is automatically generated when the XML map is saved. This field is not editable and does not appear if a mapping file has not yet been configured.

The **Validate input** property is a check box used to indicate if incoming messages to the XSLT primitive are to be validated before processing. This ensures that the incoming message is of the expected type and that any constraints defined are not violated.

The screenshot shows the IBM Software Group interface. On the left, a panel titled 'XSL Transformation : XForm1' displays an error message: 'Mapping file: cannot be empty.' Below this, the 'Root' is set to '<not specified>' and the 'Mapping file' is '<Select a mapping file>'. A 'New...' button is highlighted with a red arrow. On the right, the 'New XSLT Mapping' dialog box is open, titled 'Specify Message Types'. It contains the following fields: 'Message Root' (a dropdown menu with 'body' selected), 'Input Message Body' (a text field with 'getRealtimeQuoteRequest' and a 'Browse...' button), and 'Output Message Body' (a text field with 'getQuoteRequestMsg' and a 'Browse...' button). Below these are 'Defined Contexts' with 'Correlation Context' set to '{http://StockMediationMod}CorrelationContext' and 'Transient Context' with a warning icon and the text 'No transient context is set for this flow. Contexts can be set on the input node of the flow.' At the bottom of the dialog are '< Back', 'Next >', 'Finish', and 'Cancel' buttons.

- New XSLT mapping
 - ▶ Root specified using a drop down box
 - ▶ Input/output message types
 - Values pre-filled if message type of terminals is already defined
 - Browse... accesses change message type dialog
 - Changing message type here modifies message type of terminal
 - ▶ Mapping file is created with a generated name

6

XSLT mediation primitive

© 2007 IBM Corporation


This slide contains a screen capture of the New XSLT Mapping dialog, shown in the upper right. On the left is the Details panel as it appears before a mapping file has been configured for the XSLT primitive. The New... button is used to access the dialog.

The **Message Root** field is a drop down box that allows you to specify the Root property described on the previous slide.

The **Input Message Body** and **Output Message Body** are the message types for the input and output messages. These types correspond to the message types associated with the in and out terminals of the XSLT primitive. If a terminal already has a message type defined, it is pre-filled in the dialog, otherwise the field is left blank. The Browse... button is used to access the Change Message Type dialog, which enables you to originally define the message type or change an existing message type. Whatever you define here is propagated to the message type definition for the terminal, ensuring that they remain synchronized.

Hitting the Finish button creates an XML map corresponding to the root and message type definitions provided in this dialog. The name of the file is generated for you. You are placed into the XSLT Mapping Editor so that you can begin to define the transformations between the input and output messages.

Special considerations for properties

- Background on XSLT primitive enhancement 
 - ▶ Before V6.0.2
 - Configure with XML map or XSL style sheet, generation of style sheet a manual step
 - ▶ Starting with V6.0.2
 - Directly configure only an XML map, generation of style sheet is automatic
- Special considerations
 - ▶ Information center articles contain detailed instructions
 - Link: <http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp>
 - ▶ Migration of XSLT primitives built before V6.0.2
 - Automatic generation turned off for XSLT primitives built before V6.0.2
 - To enable automatic generation, see: **migrating an XSL transformation primitive**
 - ▶ Using an XSL style sheet without an XML map
 - In some cases, it may be better to use an XSL style sheet directly
 - To do this, see: **using an existing XSL style sheet**



In WebSphere Integration Developer before version 6.0.2, the interface and behavior for configuring an XSLT mediation primitive was different than it currently is. You were able to configure the primitive with an XML map or with an XSL style sheet. If it was configured with an XML map, the generation of the associated XSL style sheet was a manual step. Starting in version 6.0.2, the XSLT primitive can only be configured with an XML map and the generation of the XSL style sheet is automatically done for you. This change in behavior is an overall improvement for almost all usages of the XSLT primitive. However, there are a couple of situations which require special handling because of this change.


This slide only identifies the situations requiring special handling and does not provide the details. For the details you are referred to the Information Center, accessible through the link shown in the slide.

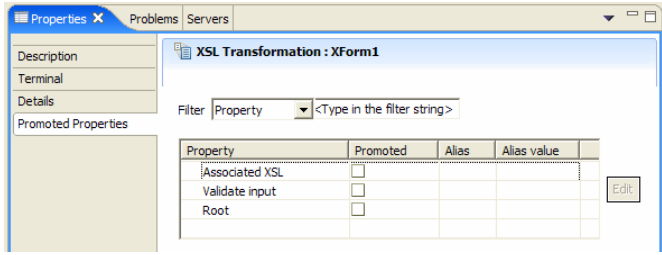
The first situation is when you have a mediation flow which contains an XSLT primitive that was created in WebSphere Integration Developer before version 6.0.2. These primitives retain their old behavior and the XSL style sheets are not automatically generated when the XML map is saved. In most cases, you would want to enable the automatic generation. See the Information Center page entitled “Migrating an XSL Transformation primitive” for details on how to do this.

There are cases where it is better to use an existing XSL style sheet rather than generating one through use of an XML map. See the Information Center page entitled “Using an existing XSL style sheet” for the details on how this can be done.

IBM Software Group IBM

Promoted properties





- Promotable
 - ▶ Associated XSL
 - ▶ Validate input
 - ▶ Root
- Not promotable
 - ▶ Mapping file

XSLT mediation primitive © 2007 IBM Corporation

This slide shows the Promoted Properties panel and lists those properties which are promotable and which are not.

Starting with the **Mapping file** property, it is not promotable because it is not involved in the runtime aspects of the XSLT primitive. It is a development time only artifact and therefore it would not make sense to promote it for administrative control.

Promotion of the **Associated XSL** and **Root** properties would need to be considered together. The XSL style sheet is built specifically for a particular value of root. Therefore, if the root property were to change, the associated XSL property would also need to be changed. Conversely, it is possible to change the associated XSL without changing the root, assuming the two XSL style sheets involved were built for the same root value. In addition, any change to the XSL style sheet would require that the message types being transformed by the style sheet conformed to the message types of the terminals.

Promoting the **Validate input** property allows an administrator to turn validation of the SMO off and on. This allows the performance advantage of not doing validation of the input SMO. However, if the need arises to debug a problem, the administrator can turn on validation while the problem is being investigated.

Error processing

- **MediationRuntimeException** thrown for:
 - ▶ XSL style sheet not found or not specified
 - ▶ XSL style sheet generated when map existed but map did not define mapping for any elements
 - ▶ XSL style sheet source/target type does not match input/output terminal type
- **MediationBusinessException** (fail terminal flow)
 - ▶ Validate input specified and input message fails validation testing
 - ▶ Errors during XSL style sheet processing
 - ▶ Root value inconsistent with XSL style sheet source type
 - ▶ Output message fails validation testing
 - The message resulting from the transformation is always validated

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

A `MediationRuntimeException` is thrown for problems accessing the XSL style sheet, such as when the style sheet cannot be found or has not been specified. This exception is also thrown for an XSL style sheet 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 style sheet source and target types do not match the terminal message types.

A `MediationBusinessException` occurs for several different problems, such as if validate input has been specified and the input message fails the validation processing.

It also occurs if there are errors encountered during the processing of the XSL style sheet, or if the root property value is inconsistent with the XSL style sheet source type.

Output messages are always validated and the `MediationBusinessException` occurs when the message fails validation testing.

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

Example usage

- Example – Setup call for target service
 - ▶ Mediation flow input operation contains only customer number
 - ▶ Operation on target service callout requires customer name and number
- Mediation logic:
 - ▶ Database lookup
 - Using customer account number looks up customer name from database
 - Customer name placed into the transient context
 - Failure if no record found for customer
 - ▶ XSLT
 - Maps source message format to target message format
 - Copies the account number from source message body to target message body
 - Copies the name from the source transient context to the target message body

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 an 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 source SMO body to the target SMO body and the name is moved from the source SMO transient context to the body of the target SMO.

IBM Software Group IBM

Example usage (cont.)

Transform format of message body and initialize with customer number and name

Lookup customer name in database and place in transient context

Fail – customer record not in database

XML editor

**Define mapping here
Drag source elements to target elements to define mapping**

Overview of mappings already defined

Target	Source	Applied Function/Grouping
tns:smo	tns:smo	
tns:smo	tns:smo	
body [0..1]	body [0..1]	
getCustomerExtendedInfo	getCustomerExtendedInfo	
customerID	customerID	
customerName	name [0..1]	

Target **Source**

XSLT mediation primitive © 2007 IBM Corporation

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 throws 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 is moved from the source body to the target body, whereas the customer name is moved from the source transient context to the target body.

Summary

- Examined the XSLT mediation primitive



XSLT

- ▶ Overview of function
- ▶ Use of terminals
- ▶ Definition of properties
- ▶ Special considerations for 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. Special considerations introduced by enhancements starting with version 6.0.2 were examined. Finally, error handling characteristics were presented and an example usage of an XSLT primitive was provided.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

[Click to send e-mail feedback](#)



You can help improve the quality of IBM Education Assistant content by providing feedback.

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 WebSphere

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 or changes in the products or programs 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 NON-INFRINGEMENT. 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 (for example, 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.

© Copyright International Business Machines Corporation 2007. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.

