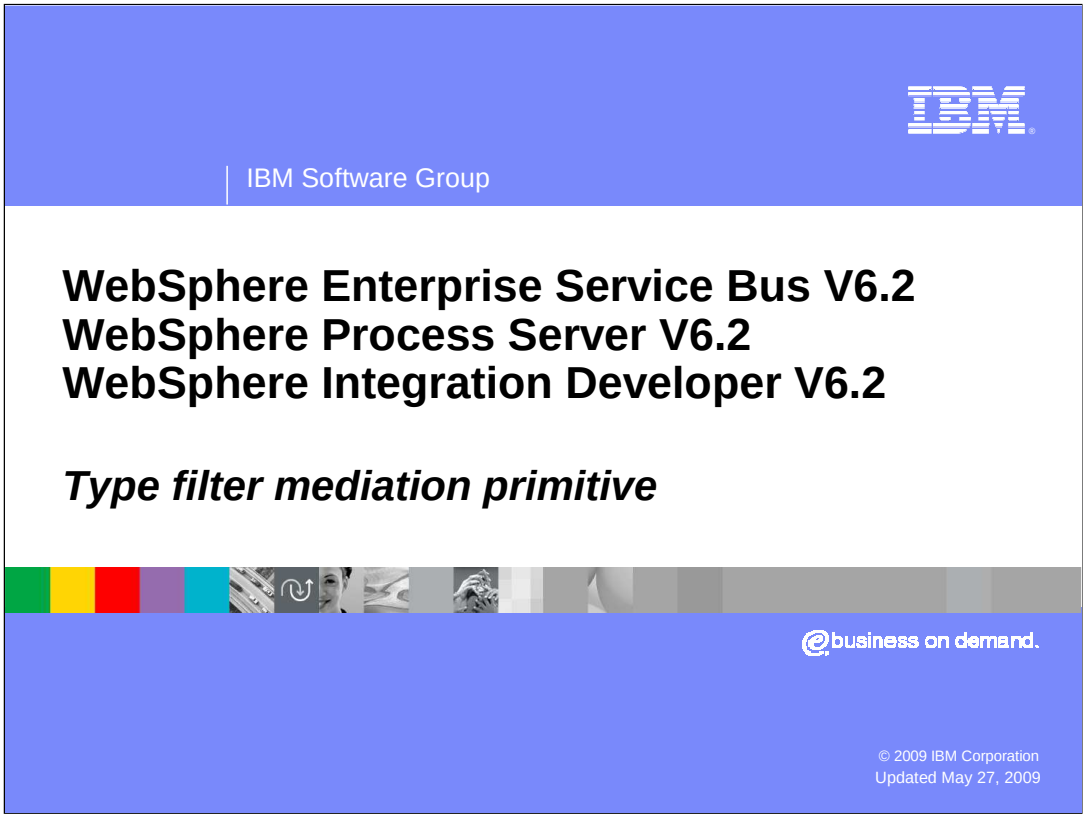


This presentation provides a detailed look at the type filter primitive, which is a new primitive introduced in version 6.2.

Goals

- Understand the type filter mediation primitive

Type filter

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

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

The presentation assumes that you are already familiar with the material presented in the presentations that cover common elements of all mediation primitives, such as properties, terminals, wiring and the use of promoted properties. The general knowledge of mediation primitives they provide is needed to understand the type filter primitive specific material in this presentation.

The presentation contains an overview of the function provided by the type filter primitive, along with information about the primitive's use of terminals and its properties. The error handling characteristics are then covered and finally an example usage of a type filter primitive is provided.

Overview

- Enables control of the paths taken through the flow based on type of message elements
- Contains one or more filters – where each filter contains:
  - A simple XPath expression identifying an element in the SMO
  - A type to compare with the type of the element
  - Name of an output terminal through which to propagate the message
- A filter with matching types fires the output terminal
  - Filters are evaluated in the order in which they are defined
  - Only the first filter with a matching type is fired
  - An element that is a derived type of the filter type is a match
- A default terminal is fired when there are no matching filters
- The Service Message Object (SMO) is not updated

The purpose of the type filter is to provide logic to control the flow within a mediation so that different paths can be taken based on the evaluation of element types within the SMO.

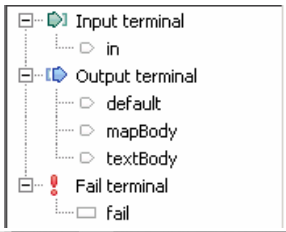
The primitive contains one or more filters contained in a table. Each filter contains a simple XPath expression that identifies an element within the SMO, specification of a type used to compare with the element and an output terminal through which to propagate the message. When the types match, the message is propagated through the terminal of the matching filter. It is considered a match if the element is the same type or a derived type of the type specified in the filter.

Filters are defined in a table and are evaluated in the same order that they appear in the table. The message is propagated through only the terminal of the first matching filter. In the case where none of the filters results in a match, there is a default terminal through which the message is propagated.

The SMO is not updated by the type filter.

Terminals

- Terminals:
  - Input terminal
  - Two or more Output terminals
  - Fail terminal
- Output terminals
  - One output terminal for each filter
  - One default terminal used when no filters match
- All terminals must be of the same message type

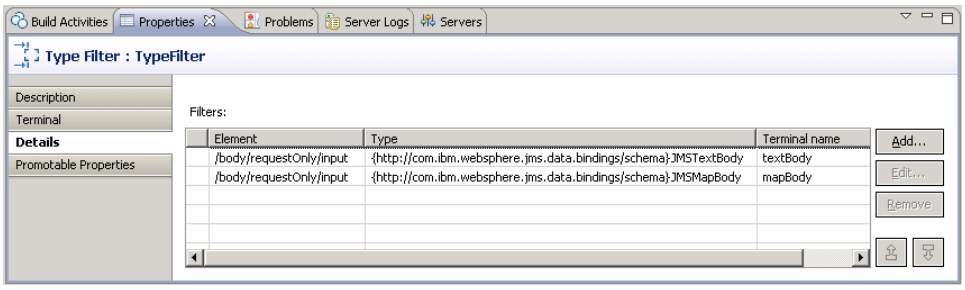


The type filter primitive has one input terminal, two or more output terminals and a fail terminal. There must be one output terminal for every filter defined and there is also a default terminal used when there are no matching filters. Two or more filters cannot reference the same output terminal.

The output terminals must all be for the same message type as the input terminal because the type filter primitive does not modify the message body.

Shown here is a type filter primitive with its terminals and the terminals as seen in the properties view.

Properties

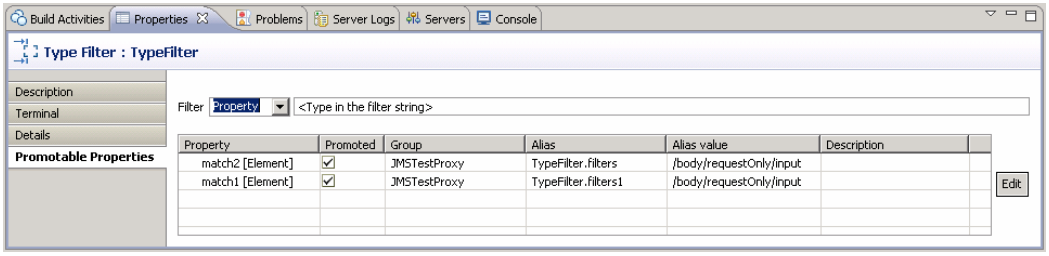


- Filters
  - Ordered list of filters, where each filter contains:
    - Element – XPath expression identifying an element in the SMO
    - Type – type against which to compare the element's type
    - Terminal name – the output terminal to fire if the types match

This slide shows a screen capture of the Details panel of the Properties view for a type filter primitive.

The only property is the Filters property, a table containing an ordered list of filters. The table has three columns. The Element column contains an XPath expression identifying an element within the SMO. The Type column contains a type used to compare to the type of the specified element. The Terminal name column contains the terminal associated with that filter, through which the message is propagated when the element type and specified type match. As previously mentioned, it is considered a match if the element type is the exact or a derived type of the specified type.

Promotable properties



- Promotable
  - Filters (Element column)
- Considerations
  - Does promotion of element make sense in your scenario
  - Default terminal names used rather than actual terminal names
  - Can't distinguish promoted rows with same element value

This slide shows the Promotable Properties panel. For table based properties such as the Filter property, only one column is allowed to be promotable. In this case it is the Element column that is promotable, whereas the Type and Terminal name columns are not promotable.

When working with the promotable properties of a type filter primitive, there are a few things you should be aware of. The first consideration is if it makes sense, in your scenario, to promote the element of a filter. If so, then the next thing you need to be aware of is that the Property column contains a default terminal name that was assigned to the filter rather than the actual terminal name you configured. Finally, the alias value is indistinguishable for filters that identify the same SMO element. The combination of these two make it difficult to identify which row in the promotable properties table is associated with a particular row in the filters table. The best way to proceed in this situation is to select only one row at a time in the promotable properties table and then look at the Details panel to see which row is shown as promoted.

Error processing

- MediationRuntimeException thrown for:
  - Syntax error in element XPath expression
  - Two or more filters with the same terminal name
  - A filter row with a null element, type or terminal name
- Default terminal fired for:
  - An empty filters property
  - This is not considered an error condition
- Filter row evaluates as no match:
  - Element XPath not found in the SMO

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

A MediationRuntimeException is thrown for incorrect conditions in the definitions of your filters. These include an XPath expression for a filter element with an incorrect syntax, two or more filters that specify the same terminal name or a filter with a null value for the element, type or terminal name.

Having a Filters property with no filters is not considered an error condition. The behavior is the same as when there are no matching filters and the default terminal is used to propagate the message.

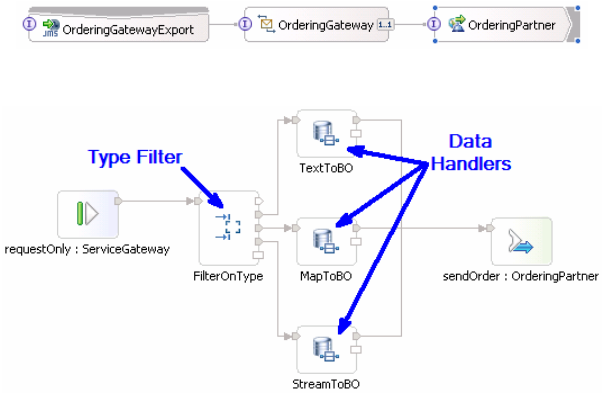
In the case where the element identified by a filter is not found in the SMO, that filter results in no match.

Example usage

- Example using the static service gateway pattern
  - Ordering system
    - Back end ordering system with a Web service interface
    - Service gateway used to forward messages to ordering system
  - Input to gateway
    - JMS message with order information originating from various sources
    - JMS message can be JMS text, map or stream format
  - Function of gateway
    - Receive input message
    - Type filter – controls flow based on type of JMS message
    - Data handlers – transform from input message format to order business object
    - Send Web service request to back end ordering system

This slide introduces an example usage of a type filter primitive in the context of a flow that implements a static service gateway pattern. The example is of an ordering system on the backend which has a Web service interface and a static service gateway that is used to forward orders to it. The input to the gateway is JMS messages. There are different systems which originate these messages and each uses a different JMS body format, either JMS text, JMS map or JMS stream. The gateway receives the message and uses a type filter primitive to determine what type of JMS message body was sent and propagates the message on the appropriate path for that type. There is a data handler primitive for each type that transforms the JMS body into the appropriate order business object which is then forwarded to the ordering system using the Web service interface.

Example usage (continued)



Filters: **Type Filter properties**

Element	Type	Terminal name
/body/message	{http://com.ibm.websphere.jms.data.bindings/schema}JMSTextBody	JMSTextBody
/body/message	{http://com.ibm.websphere.jms.data.bindings/schema}JMSMapBody	JMSMapBody
/body/message	{http://com.ibm.websphere.jms.data.bindings/schema}JMSStreamBody	JMSStreamBody

This slide contains screen captures illustrating the example described on the previous slide. The top portion shows the SCA assembly containing a JMS export supporting the ServiceGateway interface, the mediation flow component which also supports the ServiceGateway interface and a Web service import supporting the Ordering interface.

The middle of the slide shows the implementation of the service gateway mediation. You can see that the type filter primitive controls the flow, propagating the message to the appropriate data handler primitive. Each data handler transforms the message from a specific JMS body type into a business object needed to invoke the ordering system.

The bottom of the slide shows the definition of the filters needed by the type filter to control the flow.

Summary

- Examined the type filter mediation primitive

Type filter

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

In summary, this presentation provided details regarding the type filter mediation primitive. It presented an overview of the type filter primitive's function, along with information about the primitive's use of terminals and its properties. Error handling characteristics were then presented and finally an example usage of a type filter 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:

mailto:iea@us.ibm.com?subject=Feedback_about_WBPMv62_TypeFilterPrimitive.ppt

This module is also available in PDF format at: ../WBPMv62_TypeFilterPrimitive.pdf

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

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Other company, product, or 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 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.

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 (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.

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.