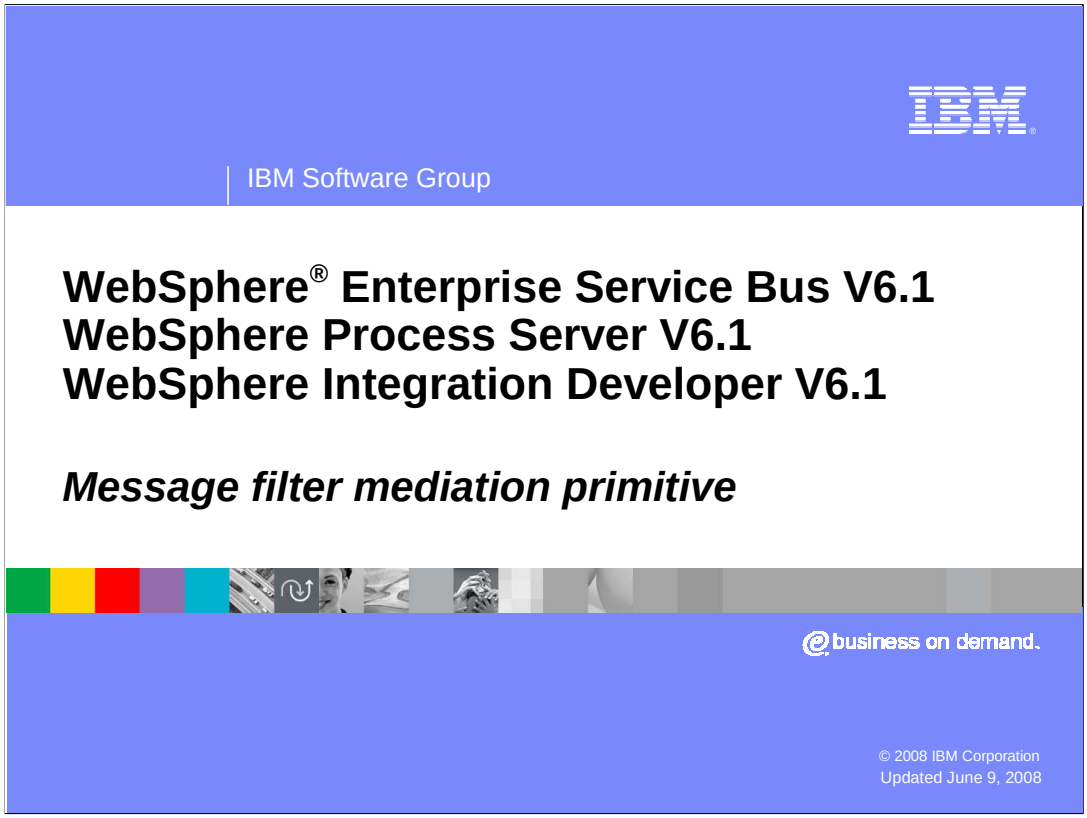


This presentation provides a detailed look at the message filter mediation primitive.

Goals

- Understand the message filter mediation primitive


Message filter

- Overview of function
- Use of terminals
- Definition of properties
- Use of distribution mode = All
- Error handling
- Example usage

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

In this presentation, an overview of the message filter mediation primitive is provided along with information about the primitive's use of terminals and its properties. There is also a discussion about the use of the Distribution mode property when it is set to All. The error handling characteristics are then covered and finally an example usage of a message filter is provided.

Overview

- Enables control of the paths taken through the flow
- Contains one or more filters – where each filter contains:
  - A simple XPath expression to be evaluated to true or false
  - Name of an output terminal through which to propagate the message
- A filter with a match (evaluates true) fires the output terminal
- Filters are evaluated in the order in which they are defined
- Configurable to allow propagation of the message
  - By firing the output terminal of only the first matching filter
  - By firing the output terminal of all matching filters
- A default terminal is fired when there are no matching filters
- The Service Message Object (SMO) is not updated

The purpose of the message filter is to enable flow of control logic within a mediation so that different paths can be taken based on the evaluation of values within the SMO.

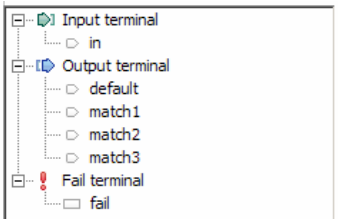
The primitive contains one or more filters. Each filter contains a conditional XPath expression that evaluates to true or false and an output terminal through which to propagate the message. When the expression evaluates to true it is called a match and the message is propagated through the terminal of the matching filter.

Filters are defined in a table and are evaluated in the order in which they appear in the table. A configuration option is used to specify if the message is propagated through only the first matching filter or through all matching filters. 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 message 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

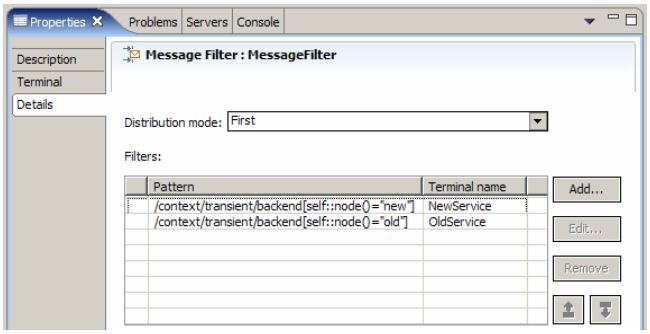


The message filter primitive has one input terminal, two or more output terminals and a fail terminal. There must be one output terminal for every filter defined and there is also a default terminal used when there are no matching filters. Two or more filters cannot reference the same output terminal.

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

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

Properties



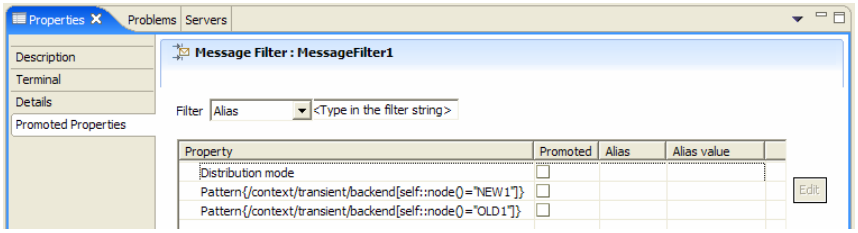
- Distribution mode
  - First – only fire output terminal for the first matching filter (the default)
  - All – fire output terminal for all matching filters
- Filters
  - Ordered list of filters, where each filter contains:
    - Pattern – XPath expression which evaluates to true or false
    - Terminal name – the output terminal to fire if there is a match

In the upper right corner of the slide is a screen capture showing the Details panel of the Properties view for a message filter primitive which shows the primitive's two properties.

The **Distribution mode** property is used to specify how to handle the matching filters. The value **First** indicates that the message should only be propagated through the terminal for the first matching filter whereas the value **All** indicates that the message should be propagated through the terminal for all of the matching filters.

The **Filters** property is a table which contains an ordered list of filters. The table has two columns. The **Pattern** column contains the XPath expression that is to be evaluated and the **Terminal name** column contains the terminal associated with that filter. A terminal must have been added to the message filter primitive before you are able to specify it in this table.

Promoted properties



- Promotable
  - Distribution mode
  - Filters (pattern column)

This slide shows the Promoted Properties panel. As you can see, there are no properties which are not promotable.

For table based properties only one column is allowed to be promotable. It is the Pattern column that is promotable for the Filters property, whereas the Terminal column is not.

Although Distribution mode is promotable, care should be taken when considering if it should be promoted. Typically the overall flow logic is dependent upon which value for Distribution mode has been specified, and changing that value would normally require a corresponding change in the flow logic.

Distribution mode = All

- Caution on the use of distribution mode = All
  - Can potentially create multiple active paths through the flow
  - Request/response operations can only have one response
  - The response flow fails if multiple callouts return a response
  - Therefore, flow logic shouldn't allow multiple request/response callouts
- Possible uses of distribution mode = All
  - Selectively broadcast a one-way operation to multiple services
  - Check SMO for multiple conditions to log
    - Define a filter for each condition requiring a log
    - Each terminal wired to a message logger and then to a stop primitive
    - Define a filter that evaluates true and wire its terminal to a flow with a callout

In this slide the use of **Distribution mode** equals **All** is examined. When using Distribution mode equals All there is the possibility of creating multiple active paths through the flow. If more than one of the active paths results in a callout to a service provider there will be multiple responses coming back, but the mediation flow can only handle one response. If more than one response comes back it results in exceptions on all but the first response. Therefore, when making use of Distribution mode equals All you must construct your flow logic so that there is only one callout to a service provider.

With this behavior in mind, what are some of the possible uses of Distribution mode equals All? One possible use is for a one-way operation where you need to selectively broadcast to multiple services based on the content of the message. Another possible usage is if you want to check for particular conditions in the SMO content and write a log for the occurrence of each condition. There can be filter checking for each condition and the terminal for each filter is wired to a Message Logger primitive and then wired to a Stop primitive. You have as many filters as the number of conditions you needed to check for. In addition, there is one additional filter that always results in a match and specifies a terminal that is wired to a flow, which results in a callout to a service provider.

These are only a couple example use cases for Distribution mode equals All. The key is for you to be aware of the potential issue with multiple callouts and avoid designing flows that might result in that situation.

Error processing

- MediationRuntimeException thrown for:
  - Invalid syntax of an XPath expression in a filter pattern
  - Two or more filters with the same terminal name
  - A Filter with a null pattern or a null terminal
- Default terminal fired for:
  - An empty filters property
  - This is not considered an error condition
- Filter evaluates as no match for:
  - XPath expression syntactically valid but incorrect
    - For example, a spelling error in the name of a field

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

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

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.

It is possible to have an XPath expression that is syntactically correct but contains a reference to an element not in the SMO. For example, this might occur if an element name was misspelled in the filter pattern. This is not considered an error condition and the filter results in no match.

Example usage

- Example - Account number based routing
  - Route requests to different services based on account number
  - 7 digit account numbers
  - 1000000 to 6999999 processed by one service
  - 7000000 to 9999999 processed by the other service
  - The services have different interfaces
  - Raise an error for anything that is out of range
  - Logic:

Distribution Mode = First

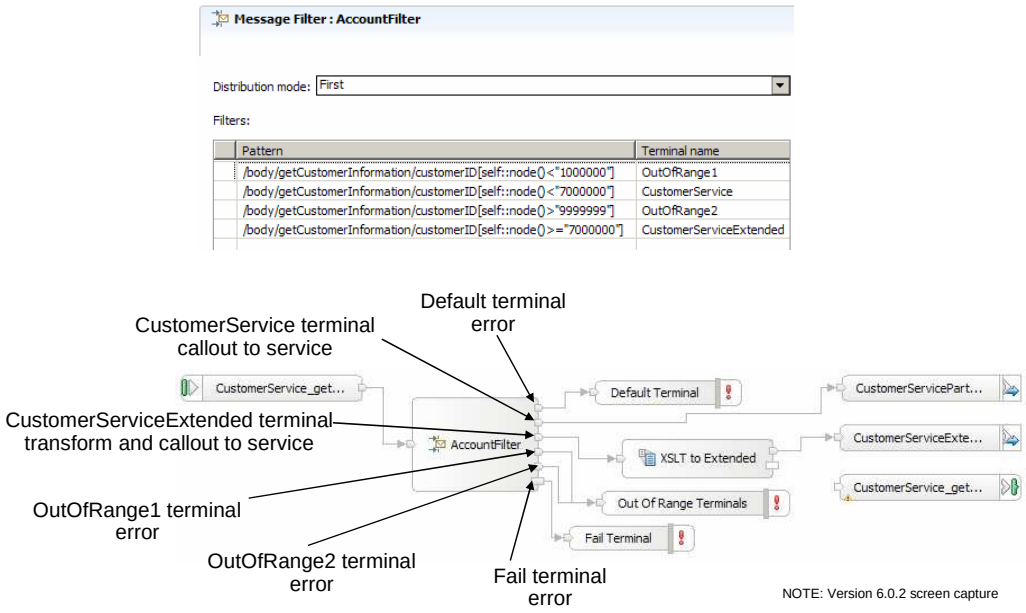
IF account # < 1000000 fire OutOfRange terminal
IF account # < 7000000 fire Service1 terminal
IF account # > 9999999 fire OutOfRange terminal
IF account # >= 7000000 fire Service2 terminal
fire Default terminal

This slide introduces an example usage of a message filter primitive. The requirement being addressed is to provide the capability to use different service providers based on the value of a customer's account number contained in the request. In the scenario, there are seven digit account numbers. Those in the range from 1 million to less than 7 million are to be handled by one service provider and the range from 7 million to less than 10 million to be handled by a different service provider. Any account numbers outside of these two ranges are considered an error condition. The two services support different interfaces, one being the same interface as the request.

The logic in the message filter is to use distribution mode equals First with the filters performing the checks in this order. First, if the account number is less than 1 million use an out of range terminal. Then if the account number is less than 7 million use the service 1 terminal. If the account number is greater than 9 million, 999 thousand, 999 use an out of range terminal and finally if the account number is greater than or equal to 7 million use the service 2 terminal.

The default terminal is used if none of the filters results in a match. However, the set of conditions being check for in the filters should cover all possibilities, and therefore this is considered an error condition.

Example usage (continued)



This slide contains screen captures for the example described on the previous slide. The top portion shows the properties view for the message filter. You can see that the Distribution mode is set to first and that the filters correspond to the logic previously described.

The bottom portion of the slide shows the mediation flow containing this message filter. You can see that the input node is wired directly to the message filter.

Looking at the terminals for the message filter you can see that the default terminal is wired to a Fail primitive which results in throwing an exception. This is done because the filters should cover all possible cases and therefore the default terminal should never be used unless there is some kind of a processing error.

The CustomerService terminal is wired directly to a callout to the CustomerService provider which supports the same interface as the original request.

The CustomerServiceExtended terminal is wired to an XSLT primitive which modifies the SMO so that it can then be passed to the CustomerServiceExtended callout.

There are two out of range terminals which are each wired to the same Fail primitive, which creates an exception because the account number is outside of the valid ranges. Although these terminals result in the identical processing, two terminals are needed because you cannot have two filters using the same terminal in a message filter primitive.

The Fail terminal is wired to a Fail primitive which creates an exception.

Summary

- Examined the message filter mediation primitive

Message filter

- Overview of function
- Use of terminals
- Definition of properties
- Use of distribution mode = All
- Error handling
- Example usage

In summary, this presentation provided details regarding the message filter mediation primitive. It presented an overview of the message filter primitive's function, along with information about the primitive's use of terminals and its properties. There was a discussion regarding the use of the Distribution mode property when it is set to All. Error handling characteristics were then presented and finally an example usage of a message 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_WBIV61_MessageFilterPrimitive.ppt

This module is also available in PDF format at: ../WBIV61_MessageFilterPrimitive.pdf

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

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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 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.