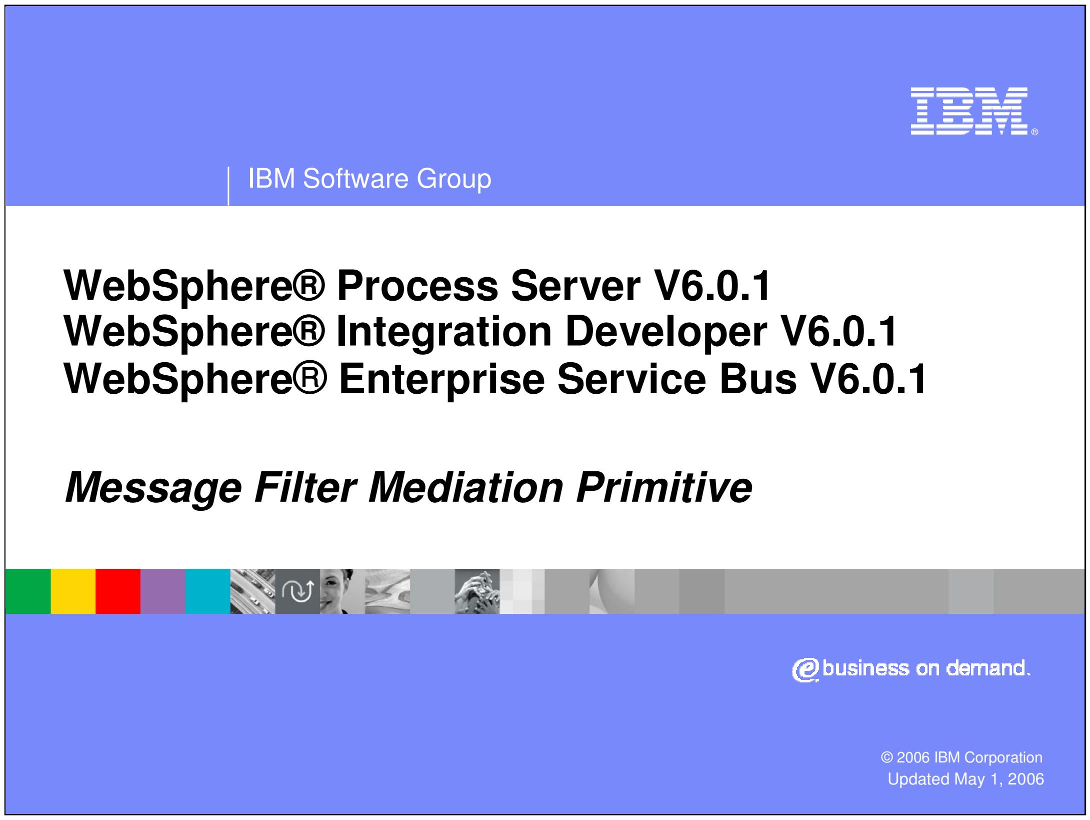
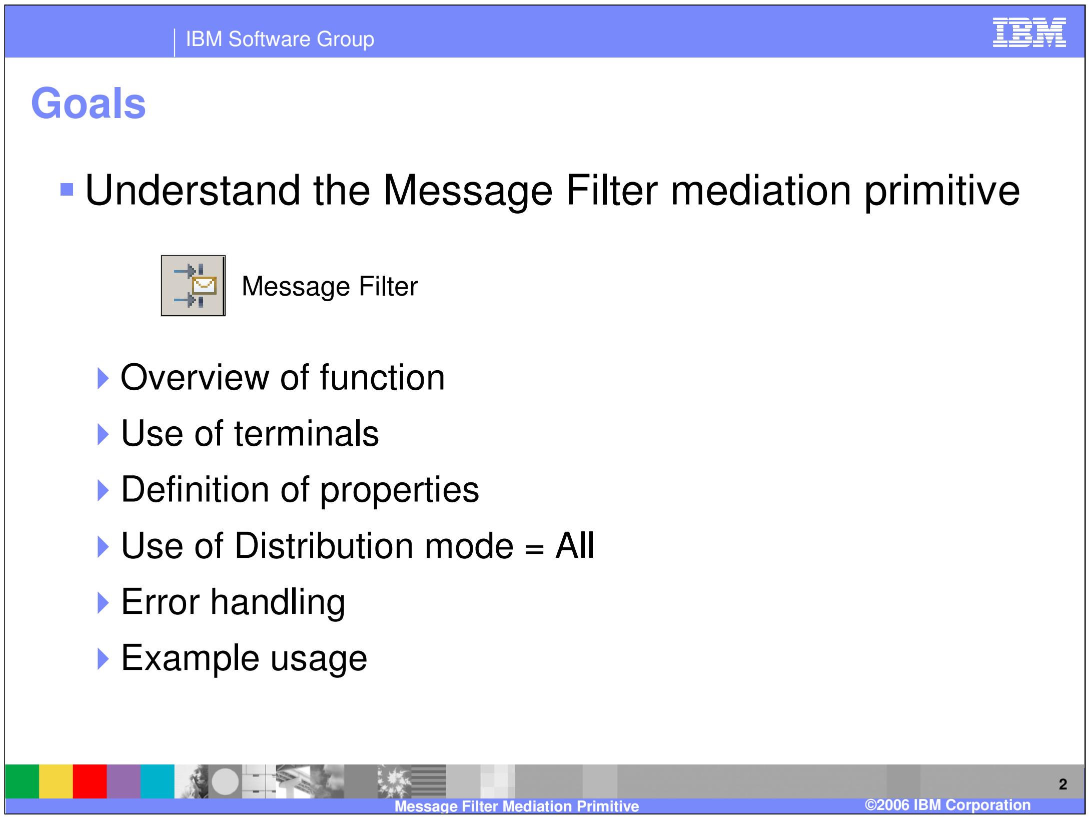


IBM Software Group

WebSphere® Process Server V6.0.1
WebSphere® Integration Developer V6.0.1
WebSphere® Enterprise Service Bus V6.0.1

Message Filter Mediation Primitive

This presentation will provide a detailed look at the Message Filter mediation primitive.



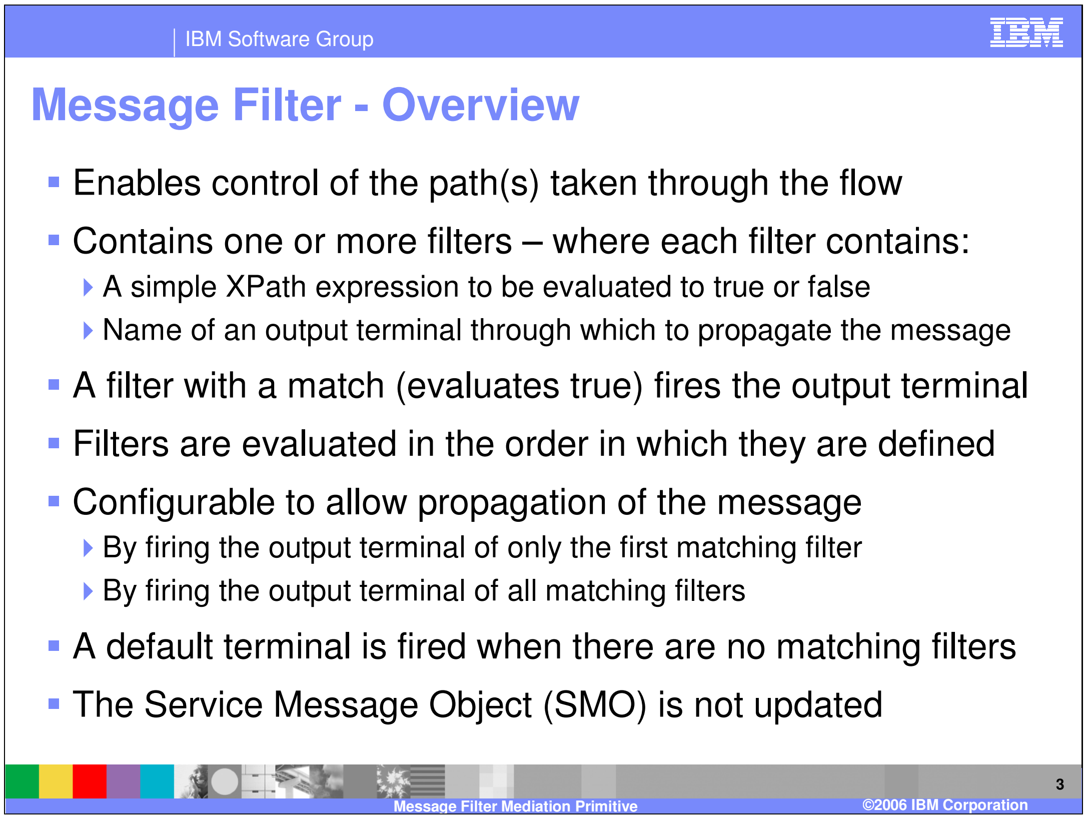
Goals

- 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. 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. 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.


IBM Software Group

Message Filter - Overview

- Enables control of the path(s) 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

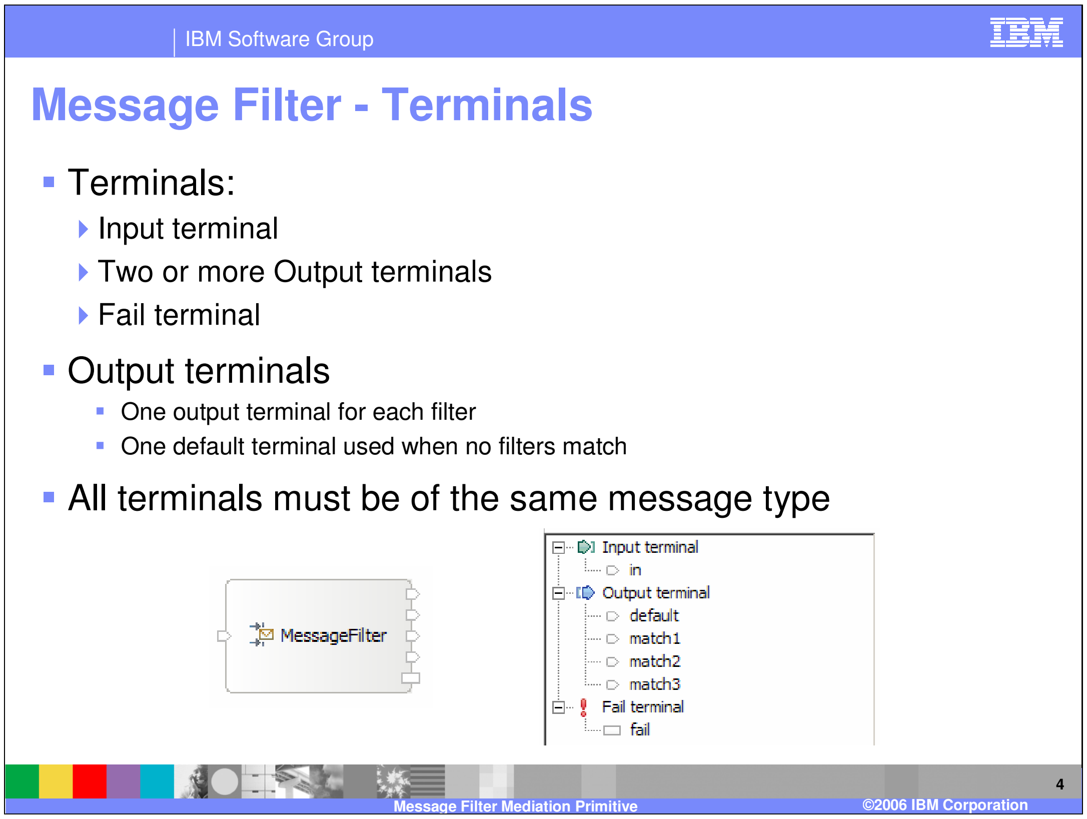
Message Filter Mediation Primitive 3 ©2006 IBM Corporation

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 will evaluate 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 define 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 will be 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.

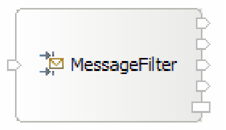
The SMO is not updated by the message filter.

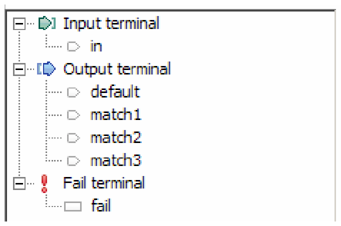


IBM Software Group

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





- Input terminal
  - in
- Output terminal
  - default
  - match1
  - match2
  - match3
- Fail terminal
  - fail

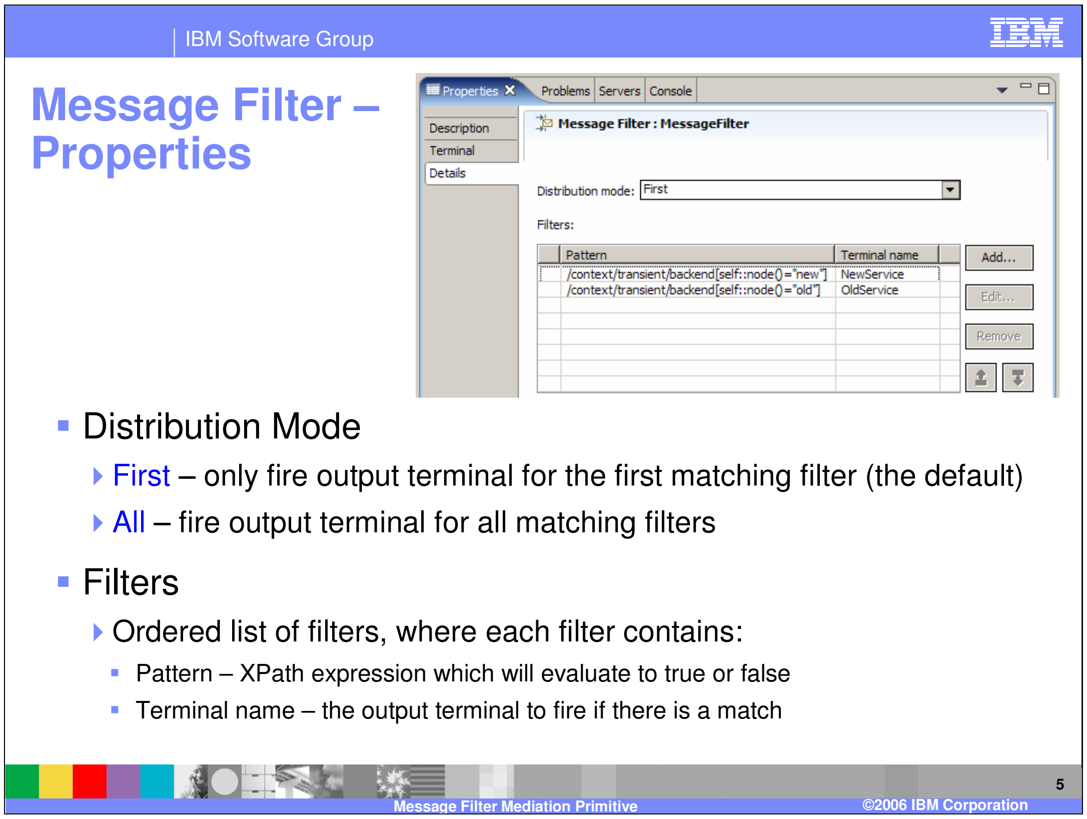
Message Filter Mediation Primitive

4

©2006 IBM Corporation

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 as the message filter primitive does not modify the message body. The slide shows a message logger primitive with its terminals and also the terminals as seen in the properties view.



Message Filter – Properties

Properties | Problems | Servers | Console

Description | Terminal | Details

Message Filter : MessageFilter

Distribution mode: First

Filters:

Pattern	Terminal name
/context/transient/backend[self::node()="new"]	NewService
/context/transient/backend[self::node()="old"]	OldService

Add... Edit... Remove

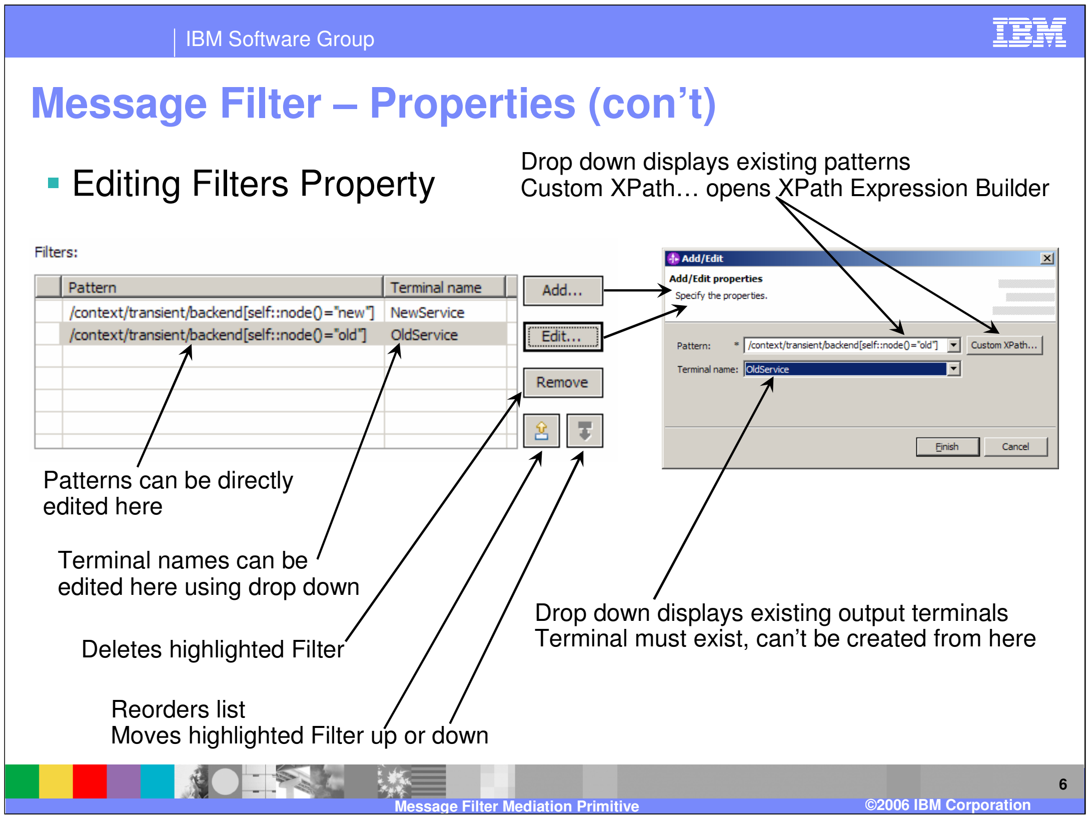
- Distribution Mode
  - First – only fire output terminal for the first matching filter (the default)
  - All – fire output terminal for all matching filters
- Filters
  - Ordered list of filters, where each filter contains:
    - Pattern – XPath expression which will evaluate to true or false
    -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

In the upper right corner of the slide is a screen capture of the Details tab from the Properties view for a Message Filter and shows the following 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.

The Filters property will be examined in more detail on the next slide.



Message Filter – Properties (con't)

- Editing Filters Property

Drop down displays existing patterns
Custom XPath… opens XPath Expression Builder

Filters:

Pattern	Terminal name
/context/transient/backend[self::node()="new"]	NewService
/context/transient/backend[self::node()="old"]	OldService

Pattern: /context/transient/backend[self::node()="old"]
Terminal name: OldService

Patterns can be directly edited here

Terminal names can be edited here using drop down

Deletes highlighted Filter

Reorders list
Moves highlighted Filter up or down

Drop down displays existing output terminals
Terminal must exist, can't be created from here

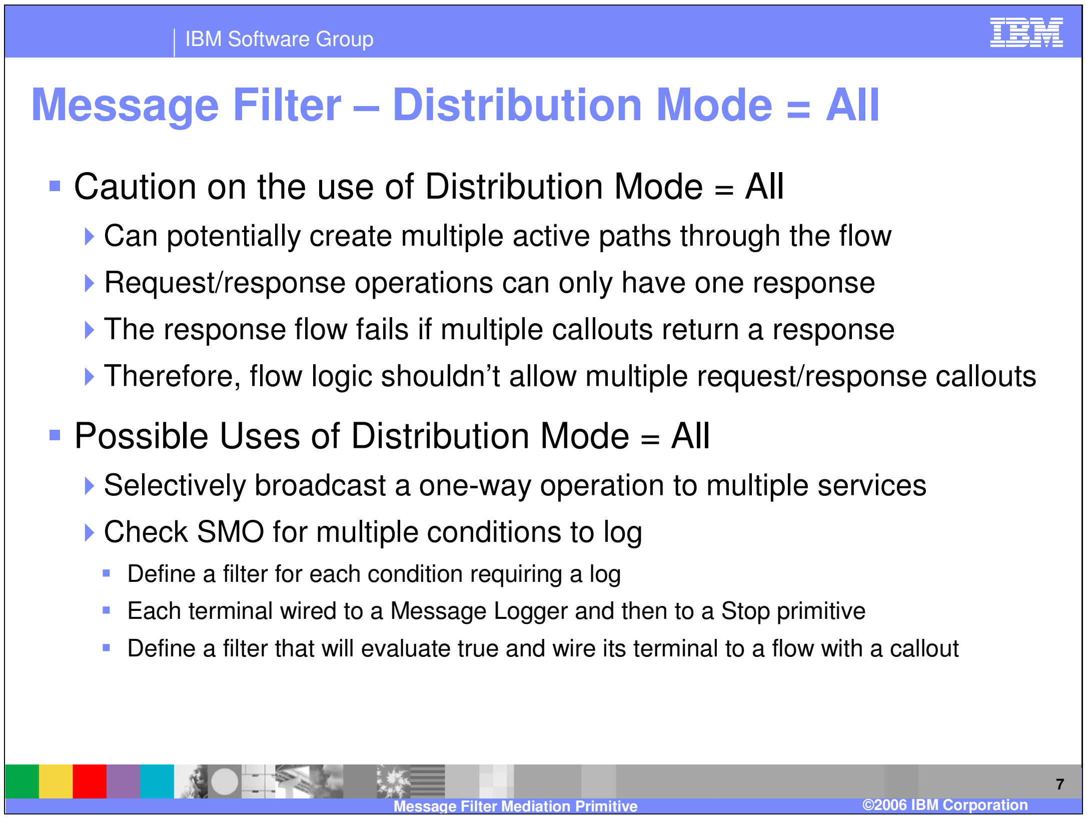
This slide looks at how the table for the Filters property is edited.

The **Add/Edit properties** dialog is used to edit a single filter. It is accessed by hitting the **Add…** button or selecting an existing filter from the table and hitting the **Edit…** button. In the dialog the value for the **Pattern** is entered by hitting the **Custom XPath…** button and using the XPath Expression Builder dialog to define the expression to be evaluated. Also patterns that have been previously entered can be selected from the dropdown box rather than using the XPath Expression Builder. The **Terminal name** is set using the drop down box which lists all existing terminals defined on the Message Filter primitive. There is no facility provided to create the terminal from this dialog so the terminal must have been previously defined.

The filters can also be edited directly within the table rather than using the **Add/Edit properties** dialog. The **Pattern** is simply edited within the table cell and the **Terminal name** is set by using a drop down box within the cell.

The **Remove** button will delete the selected filter from the table.

Since the filters are evaluated in order there must be a way to reorder the filters in the table. The up and down arrows can be used to move the selected filter within the table.


IBM Software Group

Message Filter – 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 will evaluate true and wire its terminal to a flow with a callout

7

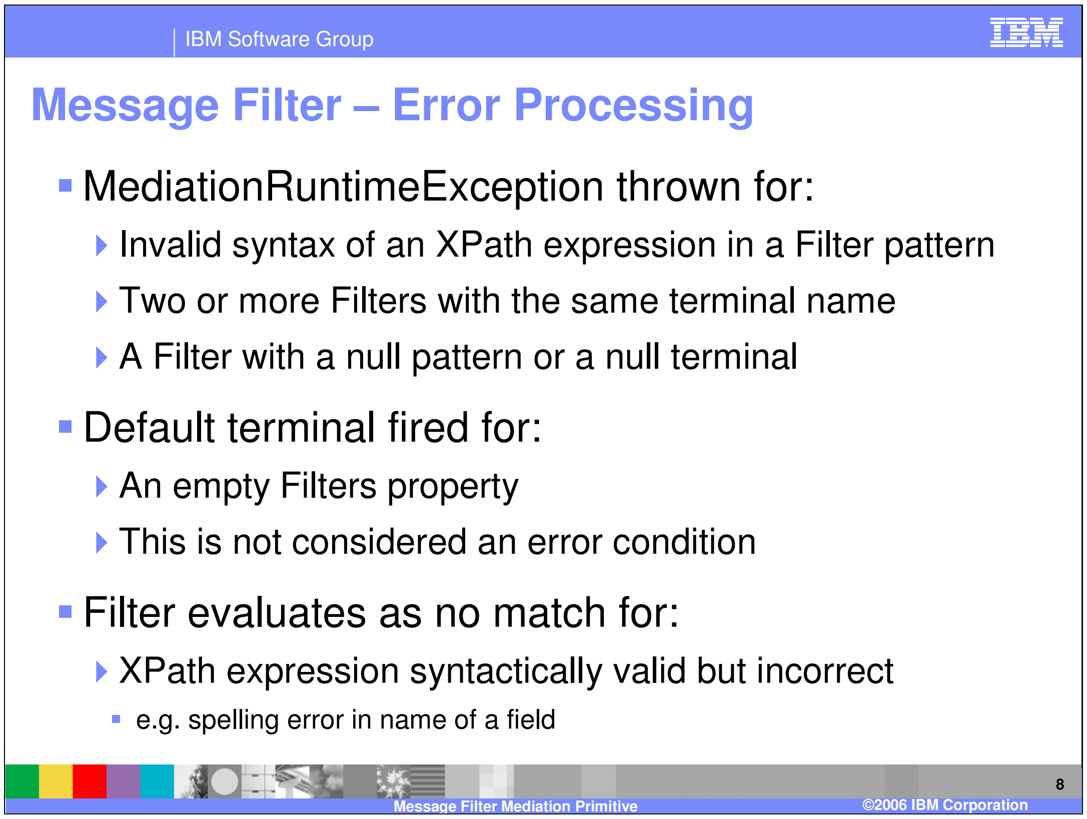
Message Filter Mediation Primitive

©2006 IBM Corporation

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 for a request response operation can only handle one response. If more than one response comes back it will result 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 would be for a one-way operation where you needed to selectively broadcast to multiple services based on the content of the message. Another possible usage would be if you wanted to check for particular conditions in the SMO content and write a log for the occurrence of each condition. There could be a filter checking for each condition and the terminal for each filter would be wired to a Message Logger primitive and then wired to a Stop primitive. You would have as many filters as the number of conditions you needed to check for, and then have one additional filter that would always result in a match which was wired to a flow that eventually resulted in the callout to a 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.



Message Filter – Error Processing

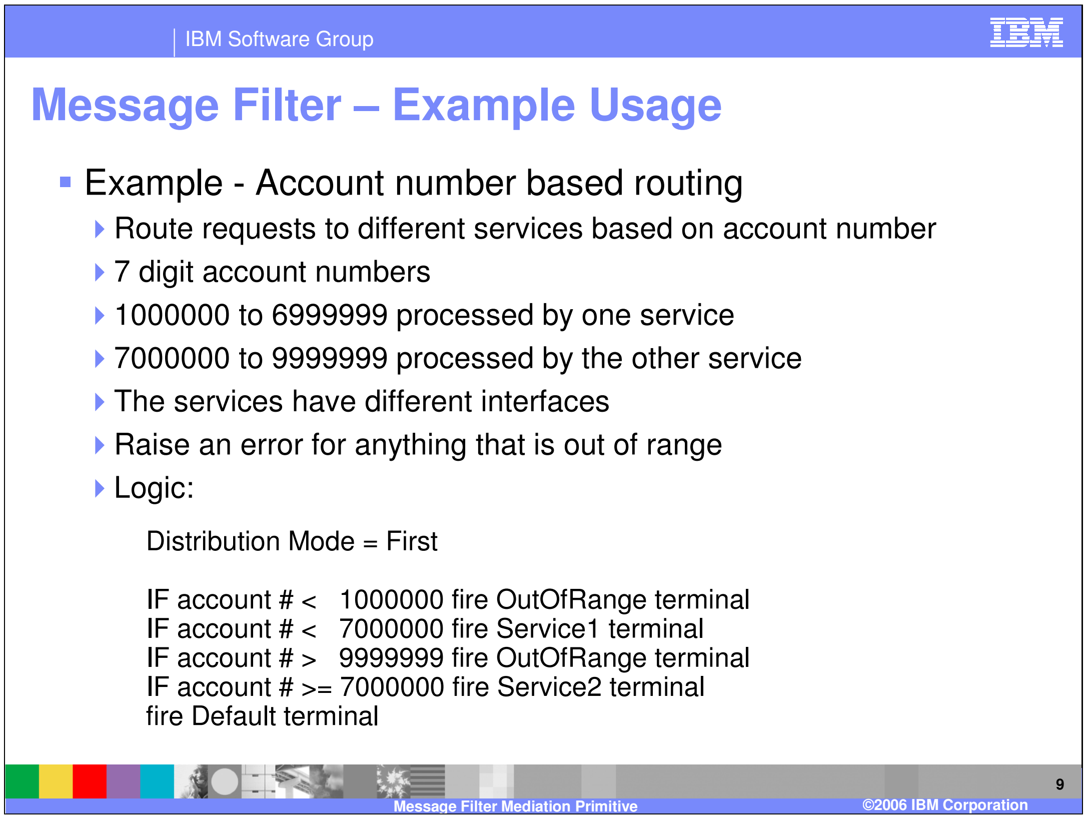
- MediationRuntimeException thrown for:
  - Invalid syntax of an XPath expression in a Filter pattern
  - Two or more Filters with the same terminal name
  - A Filter with a null pattern or a null terminal
- Default terminal fired for:
  - An empty Filters property
  - This is not considered an error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

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

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

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 could occur if an element name was misspelled in the filter pattern. This is not considered an error condition and will simple result in the filter resulting in no match.



Message Filter – 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 ordered set of filters performing the following checks:

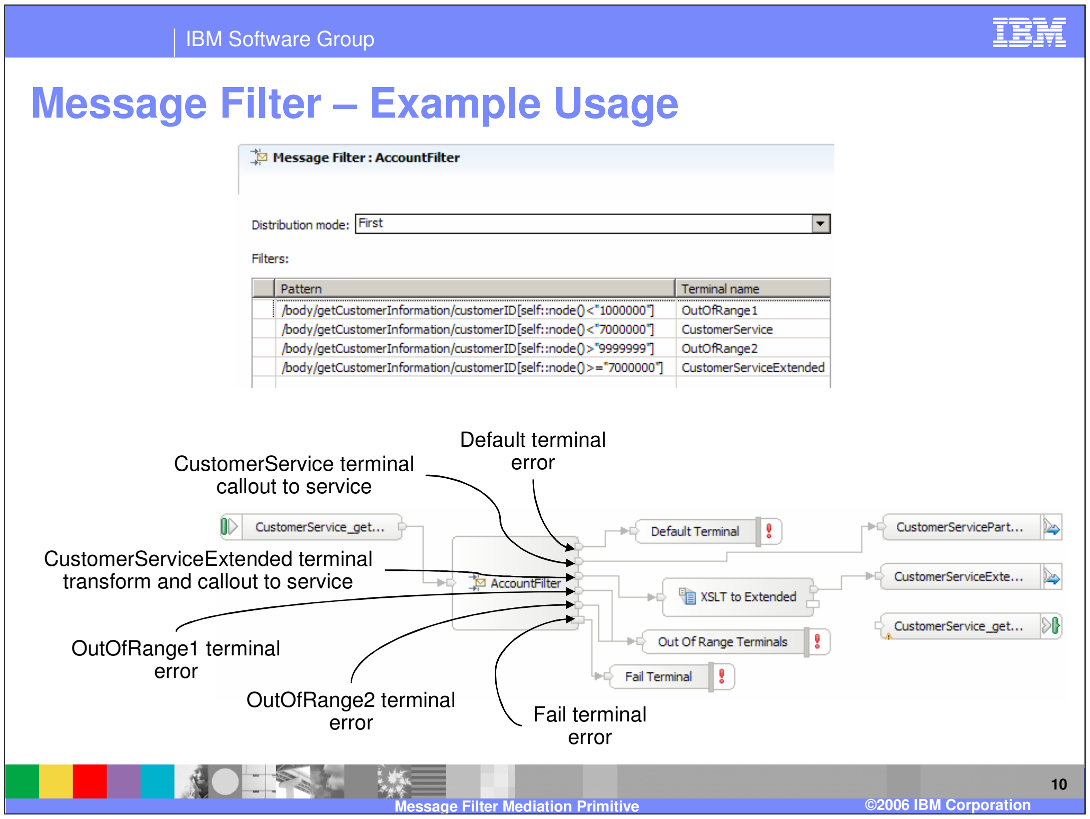
•If the account number is less than 1 million use an out of range terminal

•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

•If the account number is greater than or equal to 7 million use the service 2 terminal

The default terminal will be 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 would be considered an error condition.



IBM Software Group

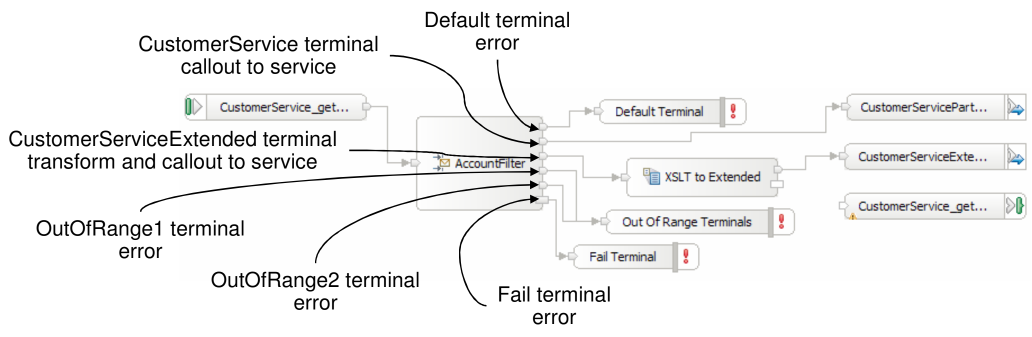
Message Filter – Example Usage

Message Filter : AccountFilter

Distribution mode: First

Filters:

Pattern	Terminal name
/body/getCustomerInformation/customerID[self::node()<"1000000"]	OutOfRange1
/body/getCustomerInformation/customerID[self::node()<"7000000"]	CustomerService
/body/getCustomerInformation/customerID[self::node()>"9999999"]	OutOfRange2
/body/getCustomerInformation/customerID[self::node()>="7000000"]	CustomerServiceExtended



10

Message Filter Mediation Primitive

©2006 IBM Corporation

This slide contains screen shots 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 the following:

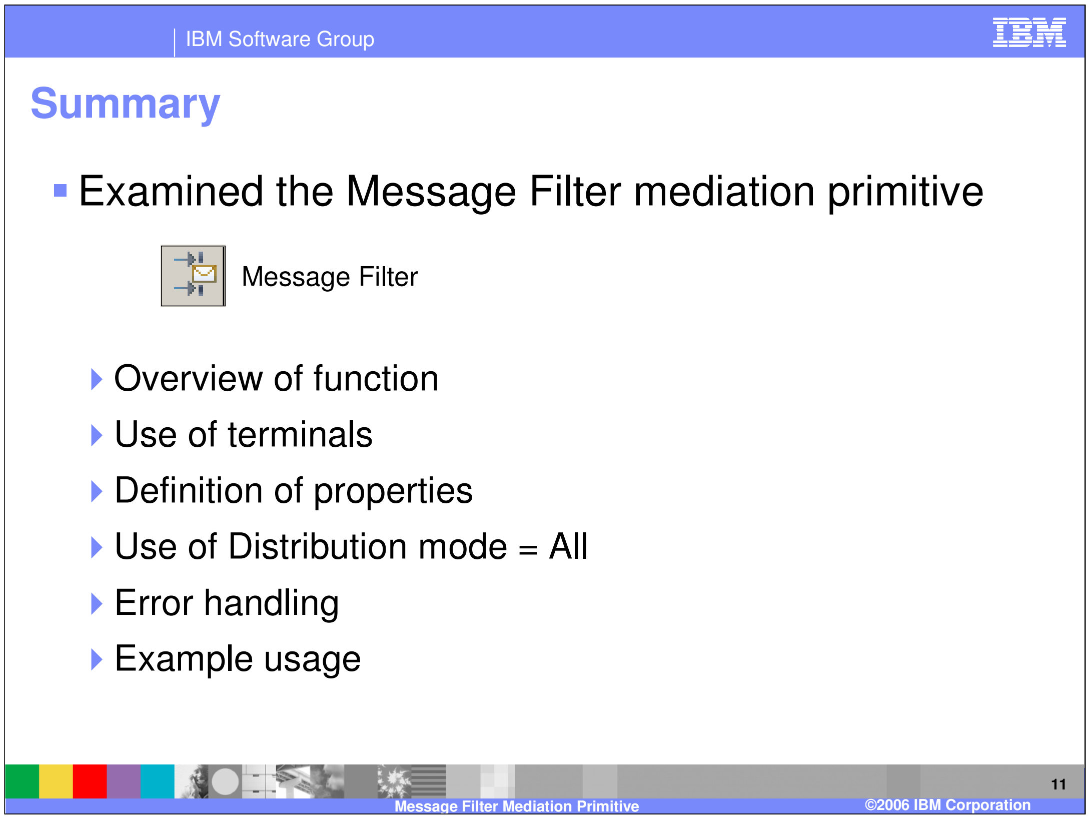
•The default terminal is wired to a Fail primitive which will result in an exception being thrown. This is done because the filters should cover all possible cases and therefore the default terminal should never be used unless there is some kind of a processing error.

•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 will throw 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 will throw an exception.



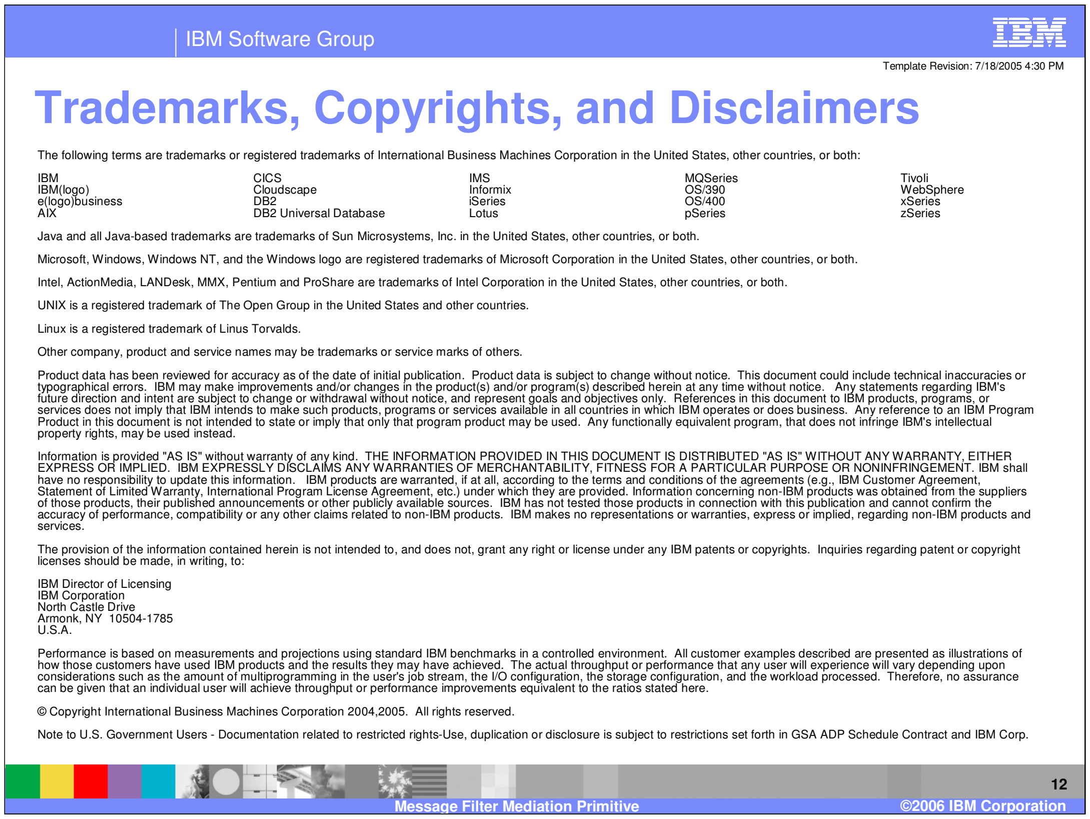
Summary

- 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 this presentation details were provided regarding the Message Filter mediation primitive and an overview of the Message Filter, along with information about the primitive’s use of terminals and its properties was provided. 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.



Template Revision: 7/18/2005 4:30 PM

Trademarks, Copyrights, and Disclaimers