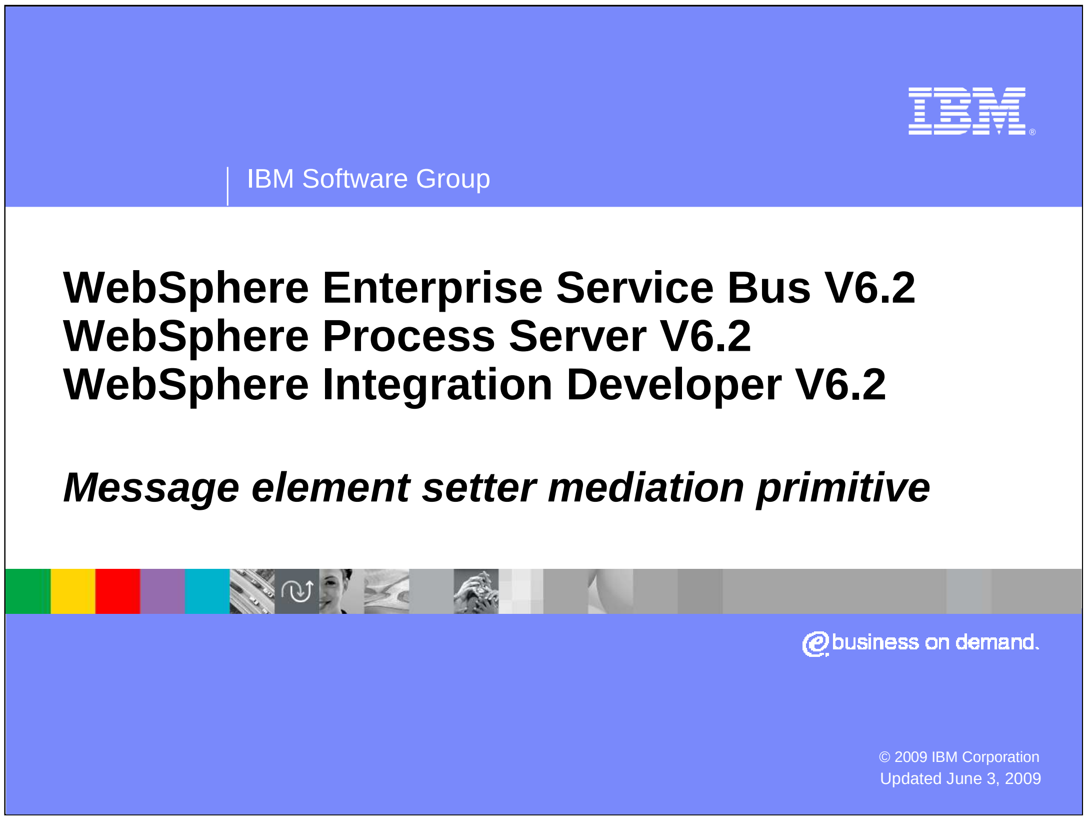


IBM Software Group

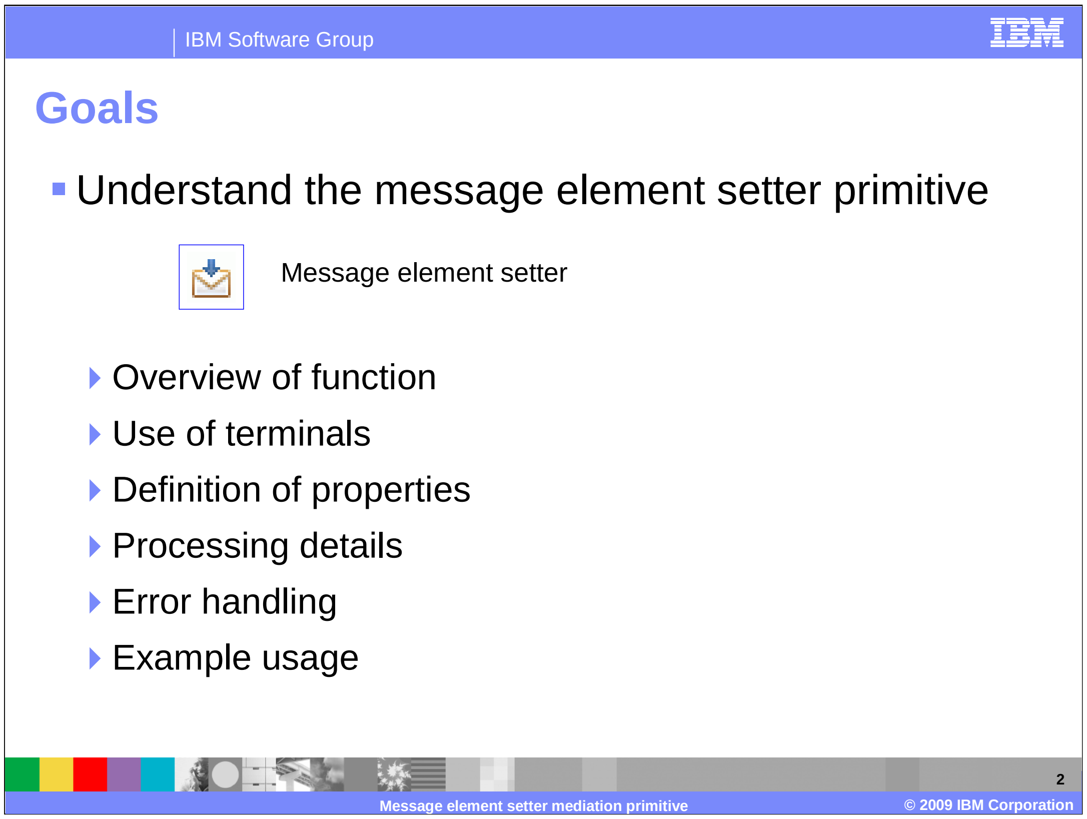
WebSphere Enterprise Service Bus V6.2
WebSphere Process Server V6.2
WebSphere Integration Developer V6.2

Message element setter mediation primitive

business on demand.

Updated June 3, 2009

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



Goals

- Understand the message element setter primitive

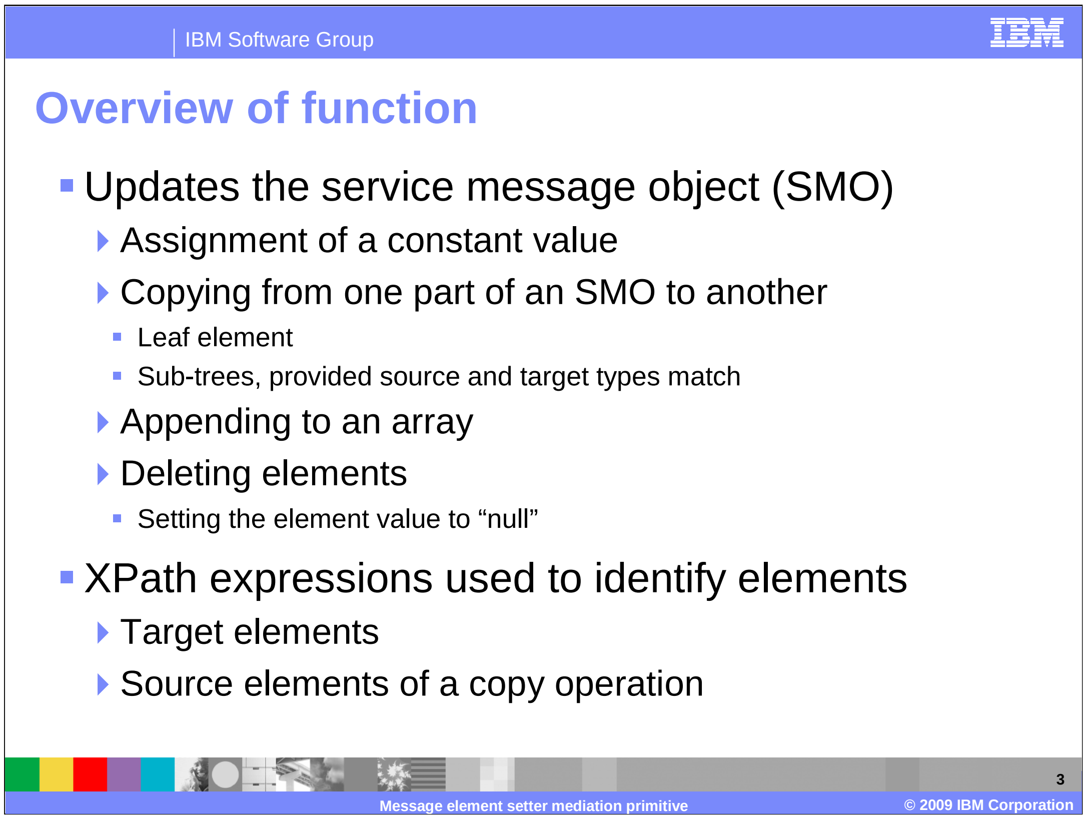
  Message element setter

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

The goal of this presentation is to provide you with a full understanding of the message element setter mediation 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 message element setter primitive specific material in this presentation.

An overview of the function provided by the message element setter primitive is presented, along with information about the primitive's use of terminals and its properties. Specific details of the processing behavior are described, followed by the error handling characteristics. Finally, a usage example of the message element setter primitive is provided.

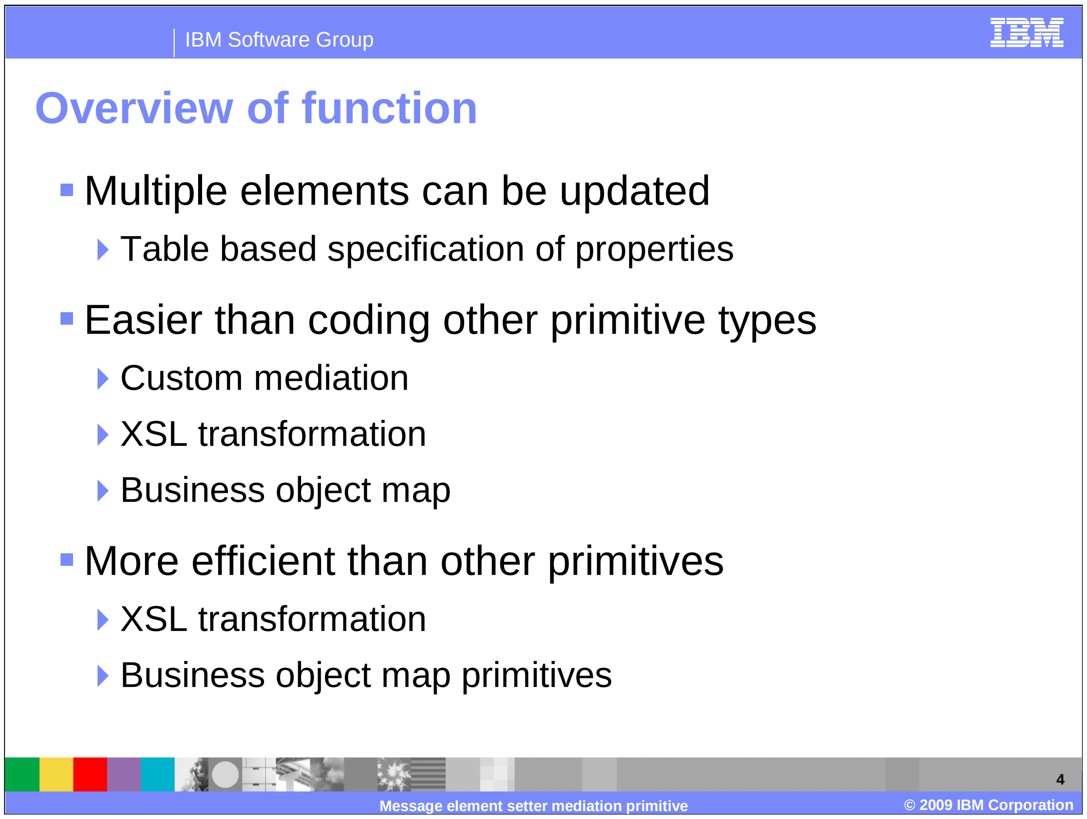


Overview of function

- Updates the service message object (SMO)
  - Assignment of a constant value
  - Copying from one part of an SMO to another
    - Leaf element
    - Sub-trees, provided source and target types match
  - Appending to an array
  - Deleting elements
    - Setting the element value to "null"
- XPath expressions used to identify elements
  - Target elements
  - Source elements of a copy operation

The function of the message element setter primitive is to enable an easy and efficient mechanism to make updates to the service message object (SMO). There are four different types of updates that can be made. The first capability is the assignment of a constant value to a leaf element of the SMO. Secondly, a copy capability is provided which allows you to copy from one part of the SMO to another. The copy might be for a leaf element or for a sub-tree, provided that the source and target sub-trees have a matching structure. Similar to the copy operation is the append, which enables you to add an element to the end of an array, providing the array target and source types match. Finally, an element can be deleted. This does not actually delete the element completely from the SMO, but rather sets the value of the element to null.

XPath expressions are used to identify the target elements in the SMO that are updated by the primitive. The source elements of copy and append operations are also identified using XPath expressions.

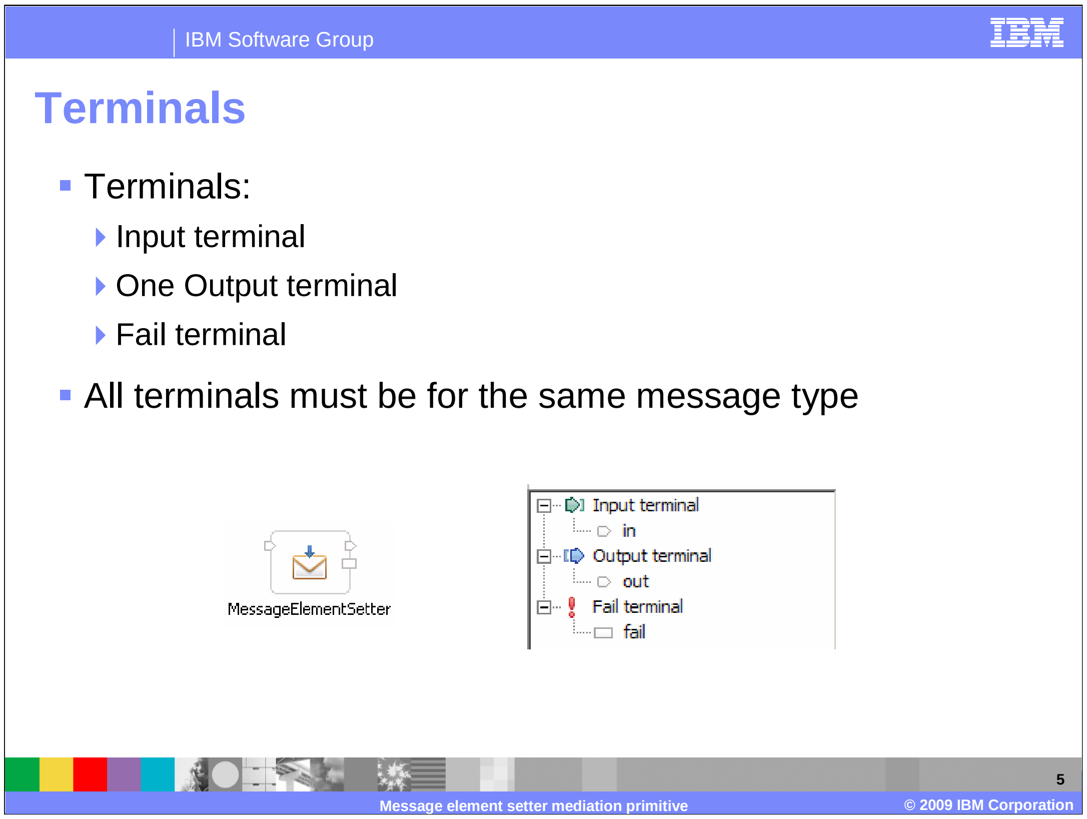


Overview of function

- Multiple elements can be updated
  - Table based specification of properties
- Easier than coding other primitive types
  - Custom mediation
  - XSL transformation
  - Business object map
- More efficient than other primitives
  - XSL transformation
  - Business object map primitives

The message element setter primitive allows multiple elements to be updated. It makes use of a table property where each row of the table defines a single update.

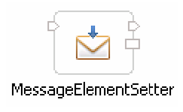
The other primitives that can be used to perform the same kind of function are the custom mediation, business object map and the XSL transformation primitives. The message element setter primitive provides an easier mechanism to define the updates than these other primitives. In addition, the updates made by a message element setter are done in place rather than making a totally new copy of the SMO. Therefore, it is much more efficient at runtime than the XSL transformation or business object map primitives.

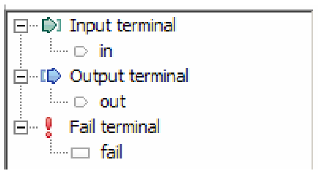


IBM Software Group

Terminals

- Terminals:
  - Input terminal
  - One Output terminal
  - Fail terminal
- All terminals must be for the same message type





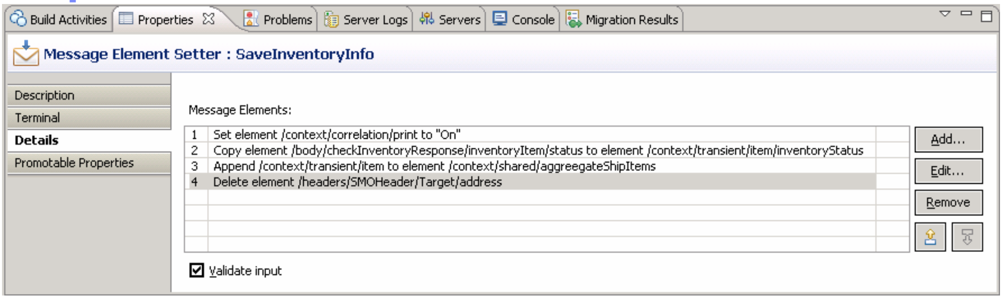
5

Message element setter mediation primitive

© 2009 IBM Corporation

The message element setter primitive has one input terminal, one output terminal and a fail terminal. The output terminal must be for the same message type as the input terminal, since the message element setter primitive does not modify the type of the message body. Shown here is a message element setter primitive with its terminals and the terminals as seen in the properties view.

Properties



Message Element Setter : SaveInventoryInfo

Message Elements:

1	Set element /context/correlation/print to "On"
2	Copy element /body/checkInventoryResponse/inventoryItem/status to element /context/transient/item/inventoryStatus
3	Append /context/transient/item to element /context/shared/aggreegateShipItems
4	Delete element /headers/SMOHeader/Target/address

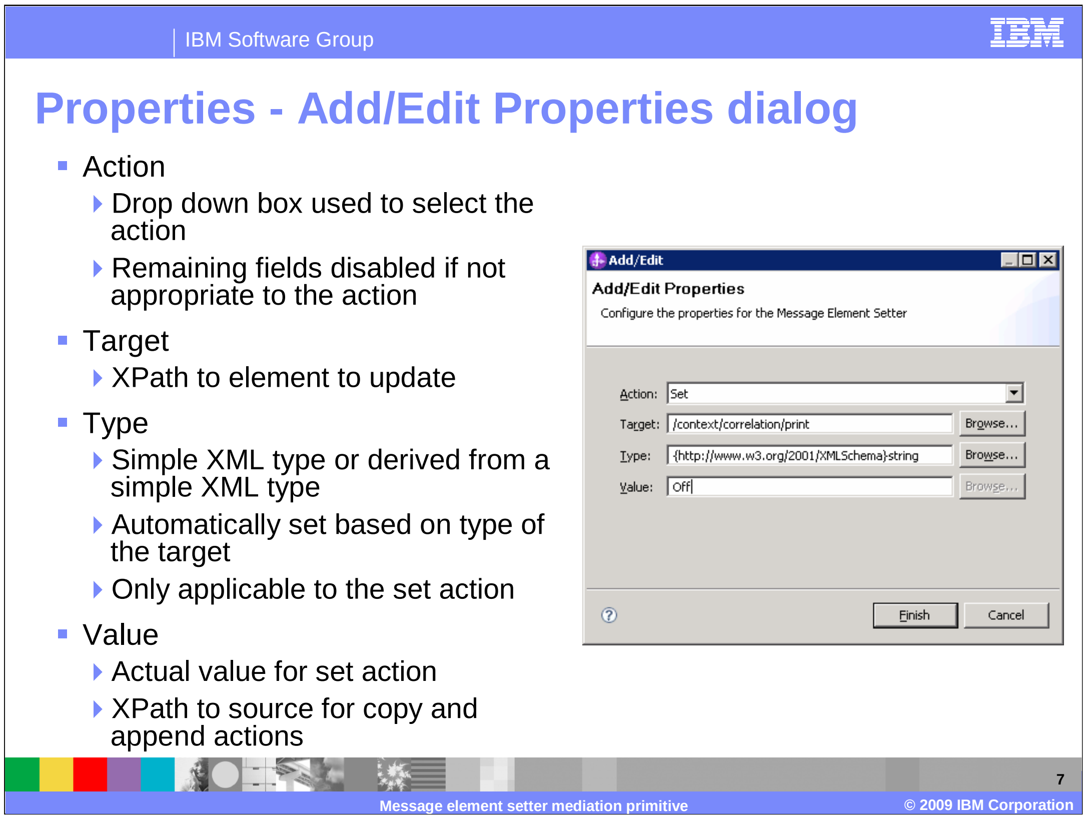
☑ Validate input

- Message Elements
  - Table defining an ordered list of actions to take
    - Set element <target XPath> to <value>
    - Copy element <source XPath> to element <target XPath>
    - Append <source XPath> to element <target XPath>
    - Delete element <target XPath>
  - The table is not directly editable
    - All editing done through the Add/Edit Properties dialog

The primary property of the message element setter primitive is the Message Elements table. It provides an ordered list of actions to be performed on elements in the SMO. There are four possible actions, which are set, copy, append and delete. Each action has the additional information needed to perform the action. All of them have a target XPath expression identifying the target of the action. The set action has a value that is placed into the target. The copy and append actions each have a source XPath expression that identifies the location of the value to be copied or appended to the target. The delete action has nothing additional specified.

The rows in the table are not directly editable. You must use the Add/Edit Properties dialog to configure each row.

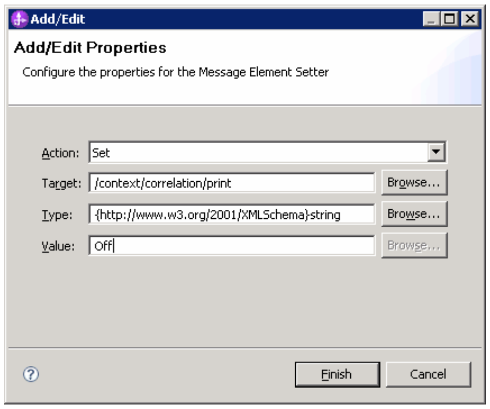
Since the table is an ordered list, the actions are performed in the same order as they appear in the table. As with most table properties, the order of the rows can be modified by changing the position of a selected row using the up arrow or down arrow buttons.



IBM Software Group

Properties - Add/Edit Properties dialog

- Action
  - Drop down box used to select the action
  - Remaining fields disabled if not appropriate to the action
- Target
  - XPath to element to update
- Type
  - Simple XML type or derived from a simple XML type
  - Automatically set based on type of the target
  - Only applicable to the set action
- Value
  - Actual value for set action
  - XPath to source for copy and append actions



Add/Edit

Add/Edit Properties

Configure the properties for the Message Element Setter

Action: Set

Target: /context/correlation/print

Type: {http://www.w3.org/2001/XMLSchema}string

Value: Off

7

Message element setter mediation primitive

© 2009 IBM Corporation

The Add/Edit Properties dialog is shown here. This dialog is used to define a row in the Message Elements table.

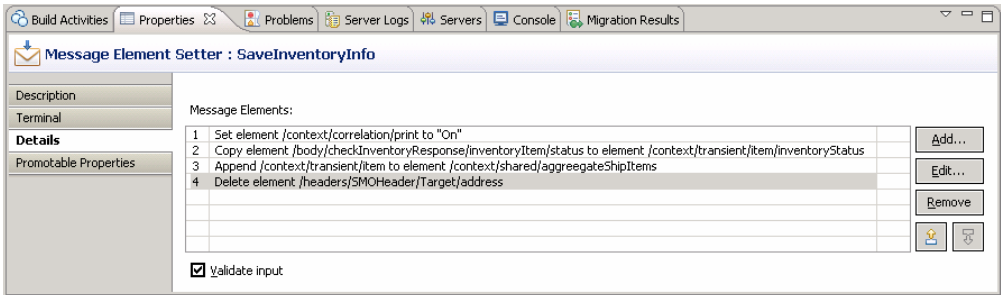
The first field is Action and you set it by selecting an action from a drop down box. Which of the remaining fields are applicable to an action vary, and those that are not applicable to the selected action are disabled.

The Target field contains an XPath expression defining the element in the SMO that is to be updated. The Browse… button opens the XPath expression builder dialog used to construct the XPath. This field is applicable to all of the actions.

The Type field is only applicable to the set action. It contains the type of the target element, which must be a simple XML type or a type derived from a simple XML type. When you set the target field, the type field is automatically set to the type of the target element.

The Value field defines what is to be set in the target element. For the set action, it contains the actual value. For the append and copy actions, it contains an XPath expression to the source location in the SMO. The type of the source element must be compatible with the type of the target element, and can be either a simple or complex type.

Properties



Message Element Setter : SaveInventoryInfo

Message Elements:

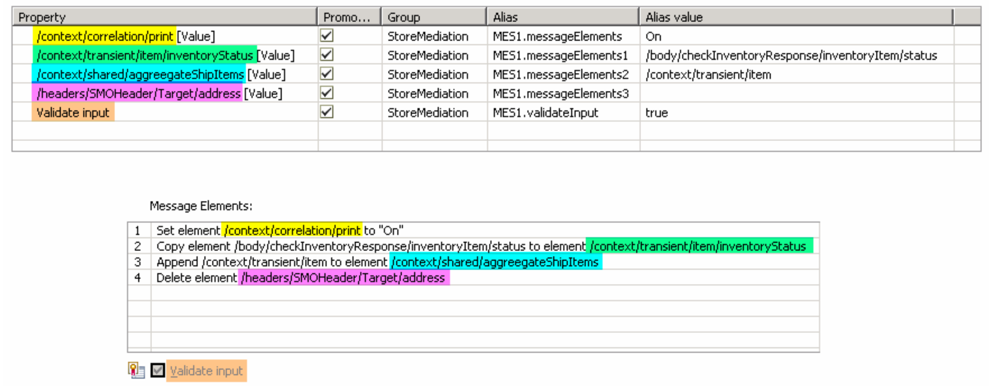
1	Set element /context/correlation/print to "On"
2	Copy element /body/checkInventoryResponse/inventoryItem/status to element /context/transient/item/inventoryStatus
3	Append /context/transient/item to element /context/shared/aggreegateShipItems
4	Delete element /headers/SMOHeader/Target/address

☑ Validate input

- Validate input
  - Validate incoming message is of the expected type
  - Ensure it meets constraints defined by its type
    - For example: minOccurs, maxValue

The message element setter also has a Validate input property. It performs a validation of the incoming SMO to ensure it is of the expected type and that it meets the constraints defined by its type. Performing the validation involves runtime processing overhead. Therefore, this should only be selected where there is a possibility that the input SMO might not conform to the specified type.

Promotable properties



Property	Promo...	Group	Alias	Alias value
/context/correlation/print [Value]	✓	StoreMediation	MES1.messageElements	On
/context/transient/item/inventoryStatus [Value]	✓	StoreMediation	MES1.messageElements1	/body/checkInventoryResponse/inventoryItem/status
/context/shared/aggreegateShipItems [Value]	✓	StoreMediation	MES1.messageElements2	/context/transient/item
/headers/SMOHeader/Target/address [Value]	✓	StoreMediation	MES1.messageElements3	
Validate input	✓	StoreMediation	MES1.validateInput	true

Message Elements:

1	Set element /context/correlation/print to "On"
2	Copy element /body/checkInventoryResponse/inventoryItem/status to element /context/transient/item/inventoryStatus
3	Append /context/transient/item to element /context/shared/aggreegateShipItems
4	Delete element /headers/SMOHeader/Target/address

✓ Validate input

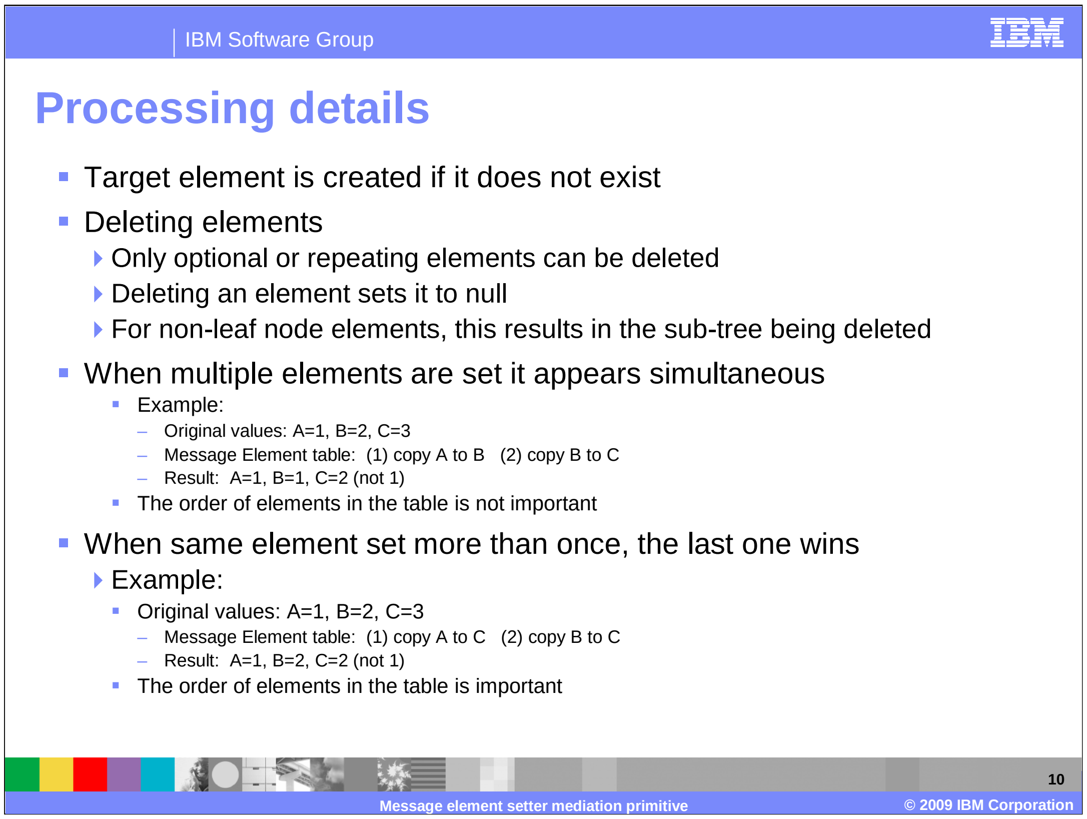
- Promotable
  - Message elements table
    - It is the value that is promoted
    - Row in table identified by the target XPath
  - Validate Input

This slide examines the promotable properties panel for the message element setter primitive. The slide shows the promotable properties panel at the top and the details panel underneath it. Color coding is used to correlate the relationship between how the properties are represented and relate to each other on the two panels.

In the message elements table property, it is the value from each row that can be promoted. In the above screen captures, the four types of actions are shown in the message elements table. They are also shown in the promotable properties panel. It is the target XPath that is used to identify a row from the message elements table in the promotable properties panel. The color coding is used to show the relationships between the rows in each panel and the location of the target XPath for each type of action.

Promoting a set, copy or append can be useful, depending upon your application scenario. Since the delete does not contain a value, it does not make sense to promote a row for a delete. However, there is still an entry in the promotable properties table for the delete.

Promoting the Validate input property allows an administrator to turn validation of the SMO on and off. This enables the performance advantage realized by not doing validation of the input SMO, while at the same time enabling the administrator to turn on validation for problem determination if the need arises to debug a problem.



Processing details

- Target element is created if it does not exist
- Deleting elements
  - Only optional or repeating elements can be deleted
  - Deleting an element sets it to null
  - For non-leaf node elements, this results in the sub-tree being deleted
- When multiple elements are set it appears simultaneous
  - Example:
    - Original values: A=1, B=2, C=3
    - Message Element table: (1) copy A to B (2) copy B to C
    - Result: A=1, B=1, C=2 (not 1)
  - The order of elements in the table is not important
- When same element set more than once, the last one wins
  - Example:
  - Original values: A=1, B=2, C=3
    - Message Element table: (1) copy A to C (2) copy B to C
    - Result: A=1, B=2, C=2 (not 1)
  - The order of elements in the table is important

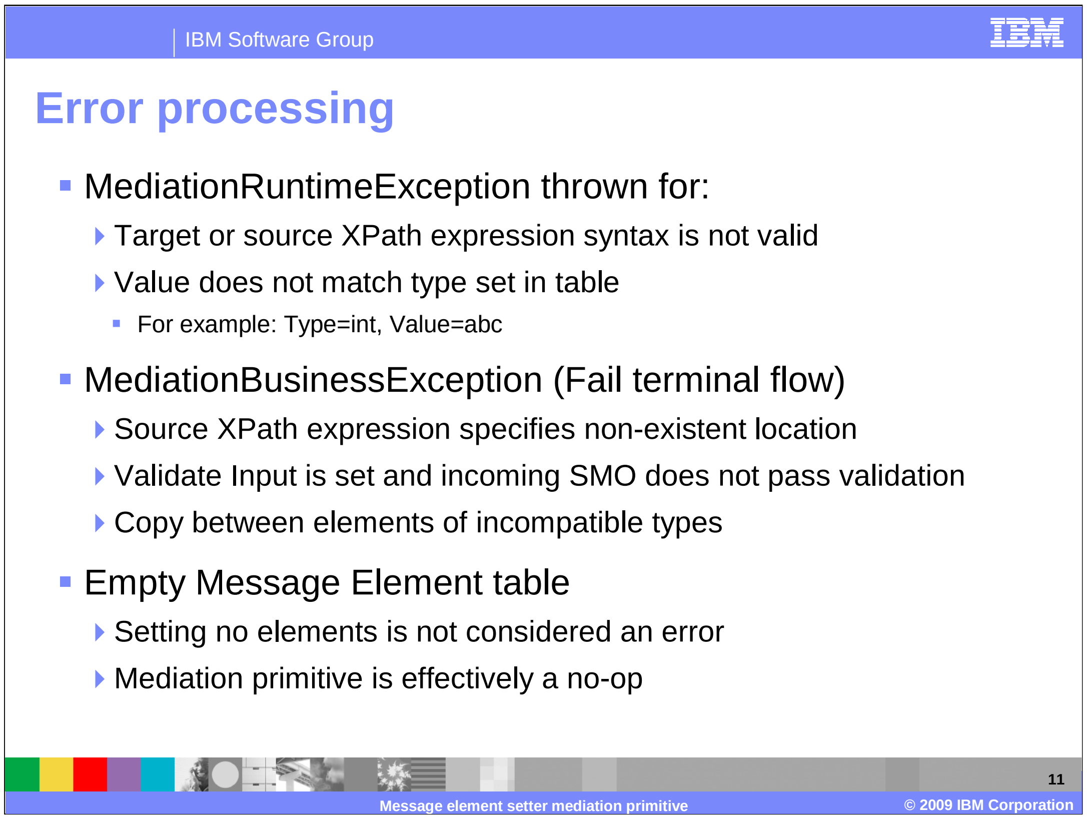
For some cases, it can be important to understand the nuances of behavior exhibited by the message element setter primitive. Several of the processing details for the primitive are provided here.

When the target XPath expression identifies an element in the SMO that does not currently exist in the SMO, it is created.

When deleting elements, there are a few things to be considered. First of all, only optional or repeating elements can be deleted. When an element is deleted, it is not removed from the SMO, rather it is set to null. Also, if the element is not a leaf node element, setting it to null results in the sub-tree for that element being deleted.

Although the table is an ordered list of updates, the results of processing updates to multiple elements appears to have occurred simultaneously. For example, suppose the table specifies to copy A to B, and then specifies to copy B to C. The result is that C is set to the original value for B, not to the value for A.

Order is important when the same element is updated more than once. In this case, the last update is the effective one. For example, suppose the table specifies to copy A to C, and then specifies to copy B to C. The result is C contains the original value for B since that was the last update to C.



Error processing

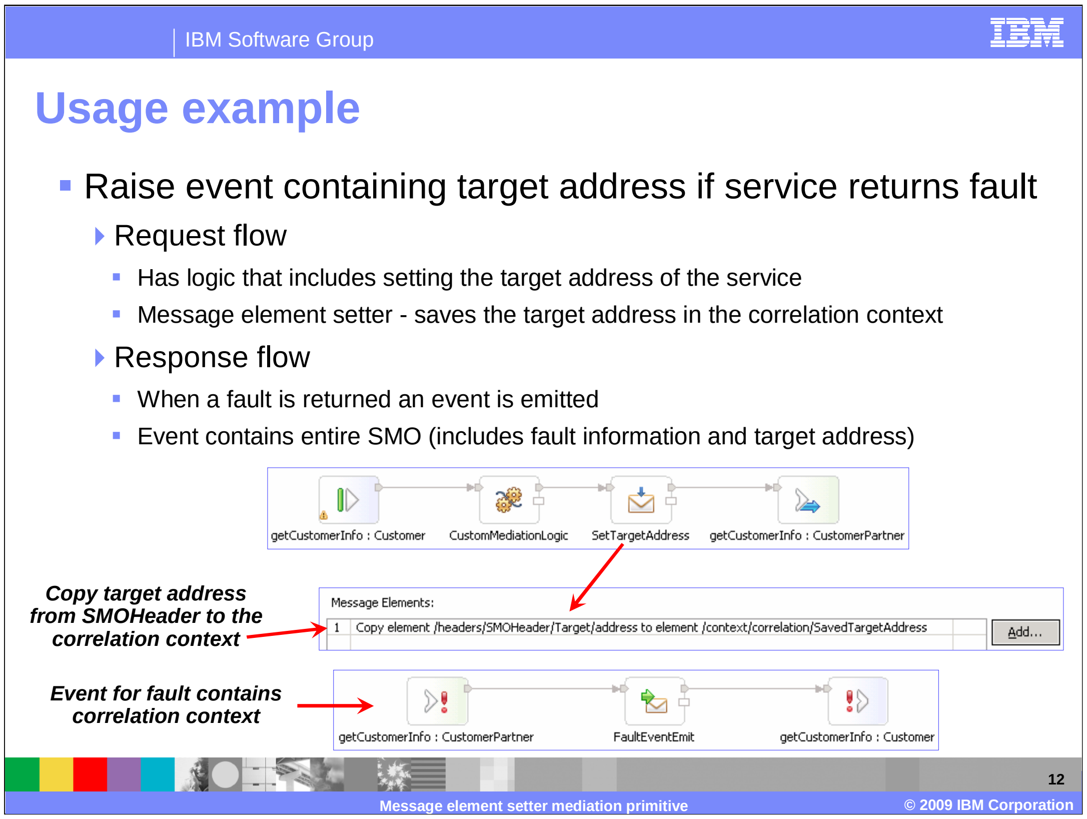
- MediationRuntimeException thrown for:
  - Target or source XPath expression syntax is not valid
  - Value does not match type set in table
    - For example: Type=int, Value=abc
- MediationBusinessException (Fail terminal flow)
  - Source XPath expression specifies non-existent location
  - Validate Input is set and incoming SMO does not pass validation
  - Copy between elements of incompatible types
- Empty Message Element table
  - Setting no elements is not considered an error
  - Mediation primitive is effectively a no-op

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

A MediationRuntimeException is thrown when either the source or target XPath expression syntax is not valid. However, WebSphere® Integration Developer should in most cases prevent you from configuring a message element setter with an XPath that is not valid. The MediationRuntimeException is also thrown when the value in the table does not match the type. An example of a type mismatch is when the type is xsd:integer and the value is an alphabetic string, such as “abc”. This problem is also identified by a validation error in WebSphere Integration Developer.

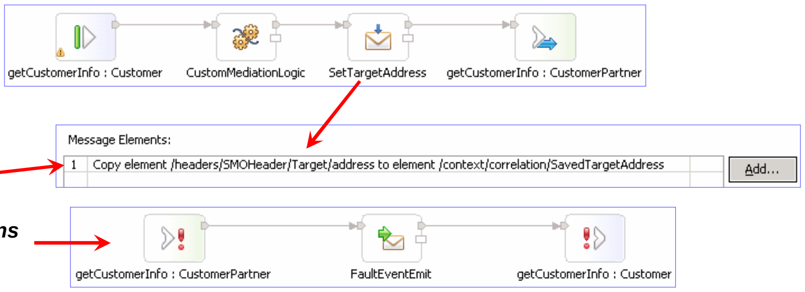
When a MediationBusinessException occurs, the flow passes through the fail terminal for the message element setter if it is wired. This exception occurs if the source XPath expression for a copy or append operation specifies a location that does not exist in the SMO. It also occurs when the Validate input property is set and the SMO does not pass validation. Another reason for the occurrence of a MediationBusinessException is when the source and target elements of a copy or append operation are not of compatible types.

It is not considered an error condition when the message element table property contains no rows. In this case, the message element setter primitive is effectively a no-op. The SMO is not updated and no error is raised.



Usage example

- Raise event containing target address if service returns fault
  - Request flow
    - Has logic that includes setting the target address of the service
    - Message element setter - saves the target address in the correlation context
  - Response flow
    - When a fault is returned an event is emitted
    - Event contains entire SMO (includes fault information and target address)

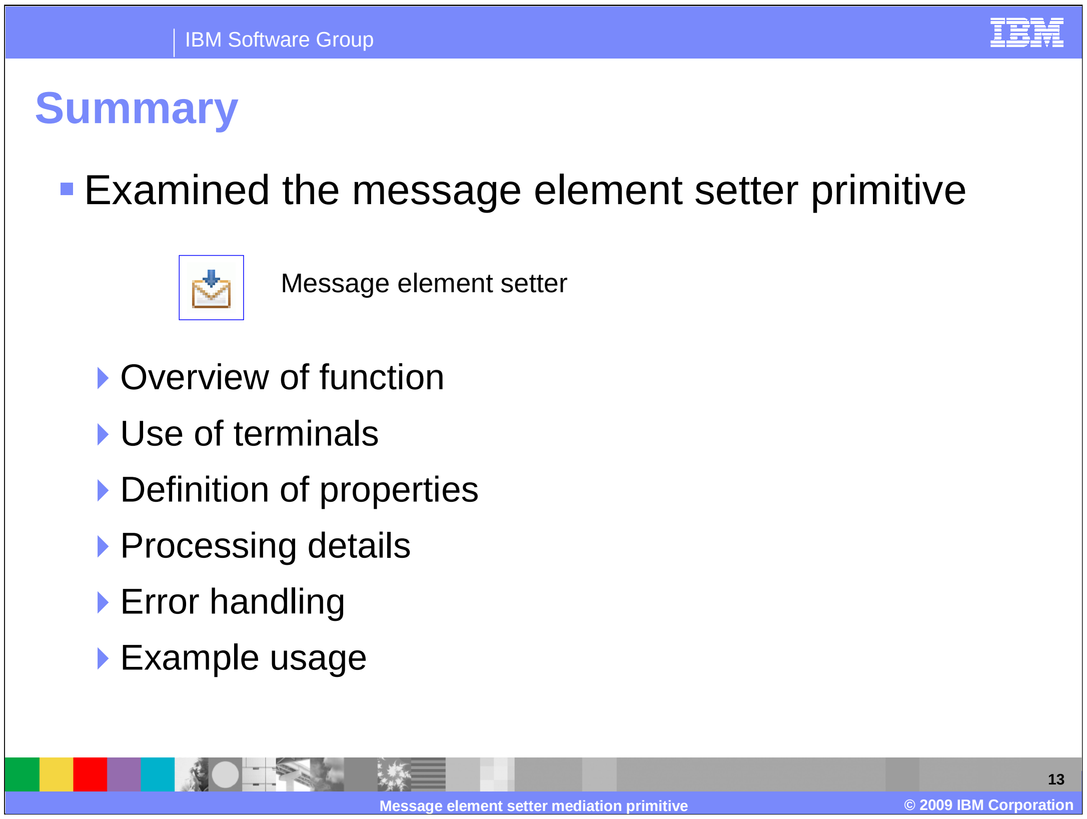

getCustomerInfo : Customer
CustomMediationLogic
SetTargetAddress
getCustomerInfo : CustomerPartner
Message Elements:
Add...
getCustomerInfo : CustomerPartner
FaultEventEmit
getCustomerInfo : Customer

1 Copy element /headers/SMOHeader/Target/address to element /context/correlation/SavedTargetAddress

Copy target address from SMOHeader to the correlation context

Event for fault contains correlation context

This slide describes an example usage of the message element setter primitive. The purpose of the scenario is to be able to raise an event that contains the target address of the service provider if the service returns a fault. In the scenario, the callout to the service provider is a dynamic callout, making use of the target address set into the SMO by logic in the mediation flow. This might be a custom mediation with logic to set the target address or possibly the result of an endpoint lookup primitive. In any case, the target address needs to be preserved across the call in case the service provider returns a fault. This is where the message element setter comes in, copying the target address from the SMO header to an element in the correlation context. Then on the response flow, the callout fault node is wired to an event emitter which places the entire SMO into the event. This produces an event that contains the target address and the fault information.



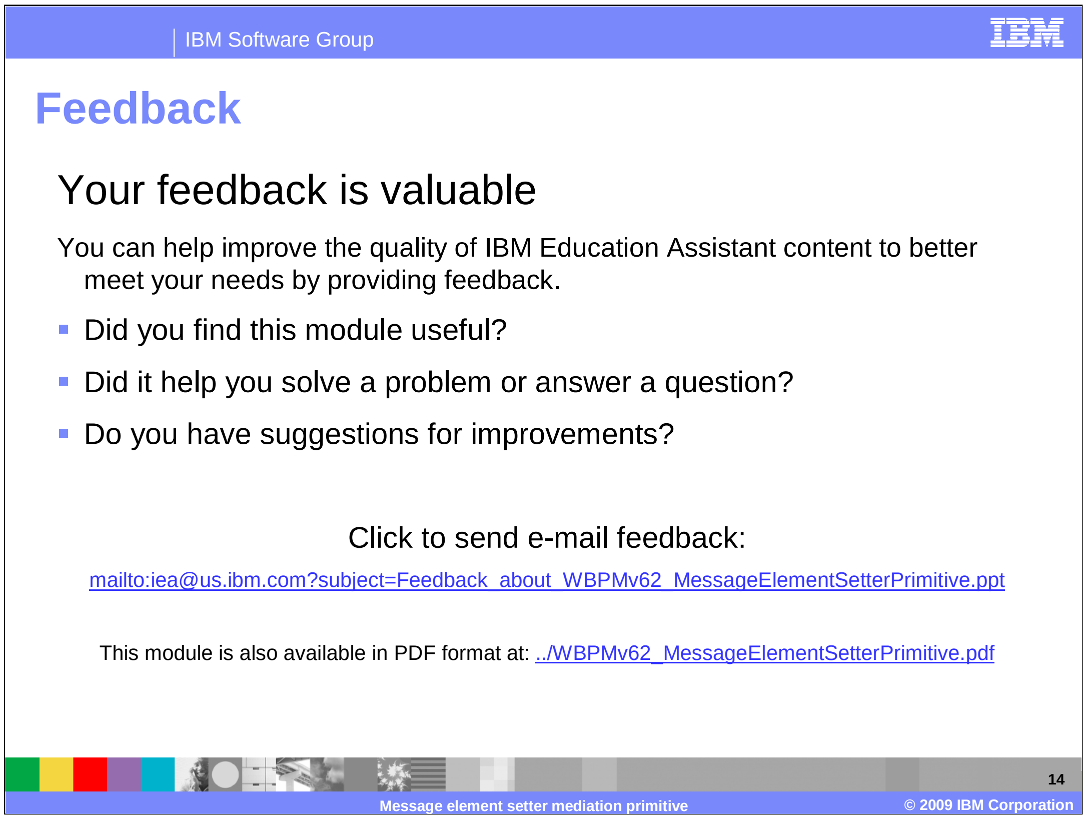
Summary

- Examined the message element setter primitive

  Message element setter

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

In summary, this presentation provided an overview of the function provided by the message element setter primitive, along with information about the primitive's use of terminals and its properties. Details of processing behavior were described followed by the error handling characteristics. Finally, a usage example of the message element setter 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.

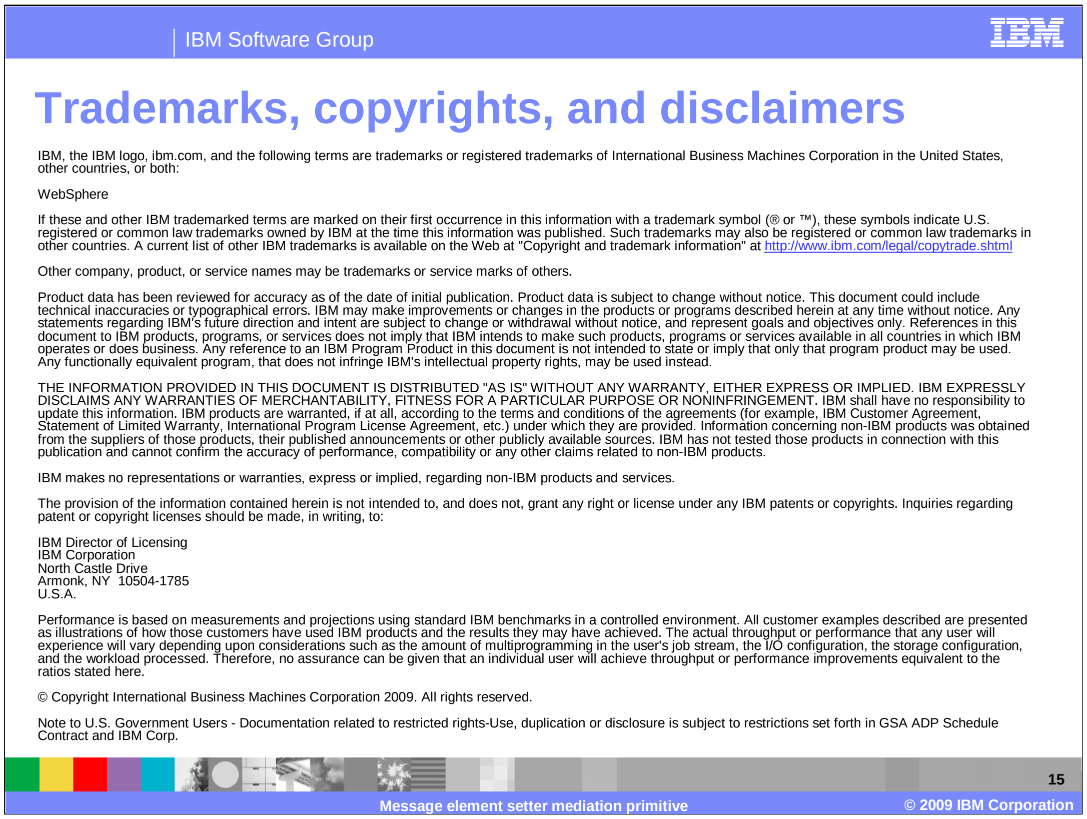
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WBPMv62_MessageElementSetterPrimitive.ppt

This module is also available in PDF format at: ../WBPMv62_MessageElementSetterPrimitive.pdf

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



Trademarks, copyrights, and disclaimers