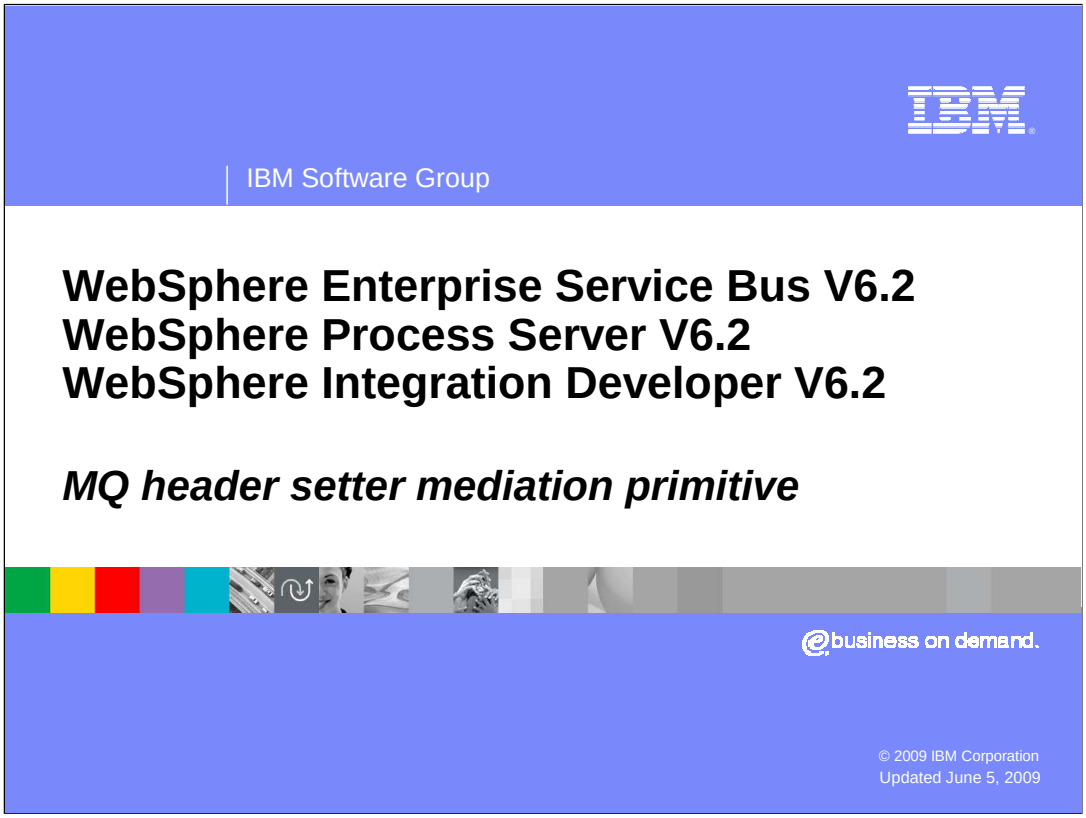


This presentation provides a detailed look at the MQ header setter mediation primitive, which is a new primitive introduced in version 6.2.

Goals

- Understand the MQ header setter mediation primitive

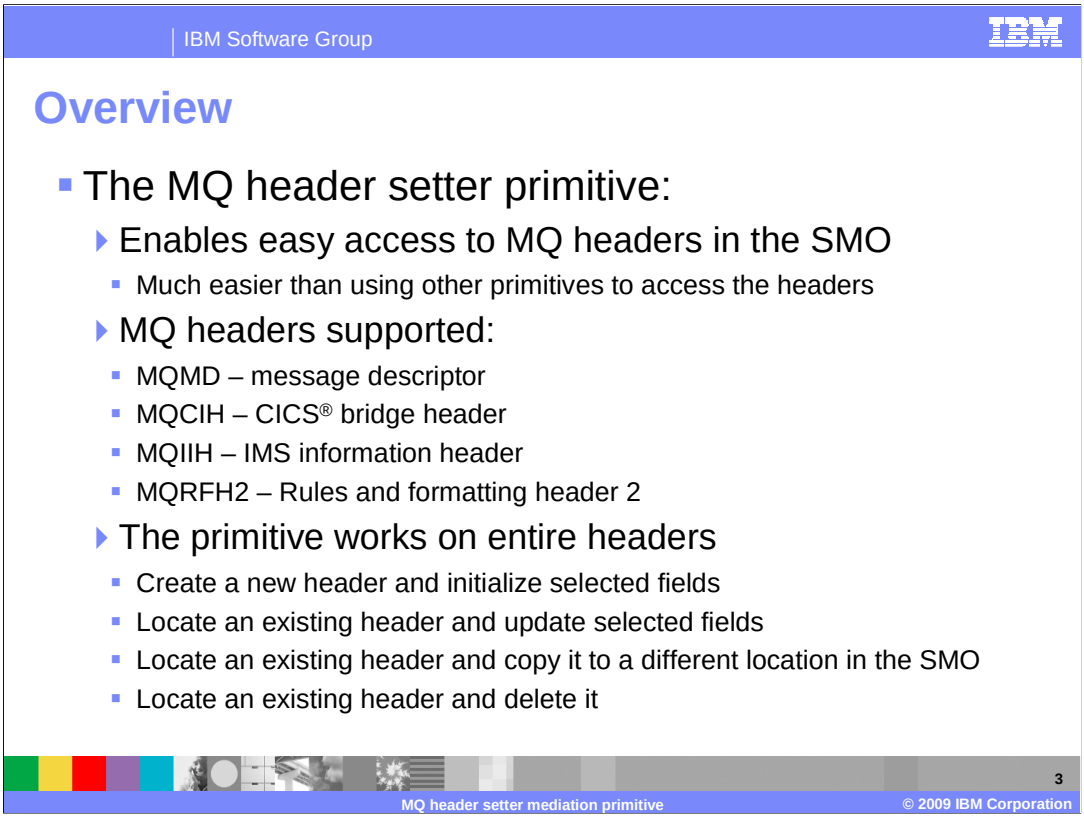
MQ header setter

- 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 MQ header 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 MQ header setter primitive specific material in this presentation.

The presentation contains an overview of the function provided by the MQ header setter 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 MQ header setter primitive is provided.



The MQ header setter primitive enables access to MQ headers in the SMO, providing a much easier mechanism to access the headers than is possible using other primitive types.

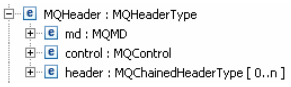
There are four MQ headers that are supported by the primitive. The message descriptor, which is used with all MQ messages, is known as the MQMD. The CICS bridge header, called MQCIH, is used for interaction with CICS. Likewise, the IMS information header, called MQIIH, is used for interaction with IMS. Finally, the rules and formatting header two, known as MQRFH2, has a variety of uses, such as when MQ is used as the transport for JMS.

The primitive makes use of schema files that define these MQ headers. You need to include these schema files in your WebSphere® Integration Developer project by selecting them in the pre-defined resources panel of the dependencies editor.

The primitive works by defining actions that are associated with an entire header. There are four actions. The create action adds a new MQ header to the SMO and initializes selected fields within the header. The find and set action locates an existing header based on header type and then updates other selected fields within the header. The find and copy action locates an existing header based on header type and copies the entire header to a designated location in the SMO. Finally, the find and delete action locates an existing header based on header type and deletes the entire header.

Overview

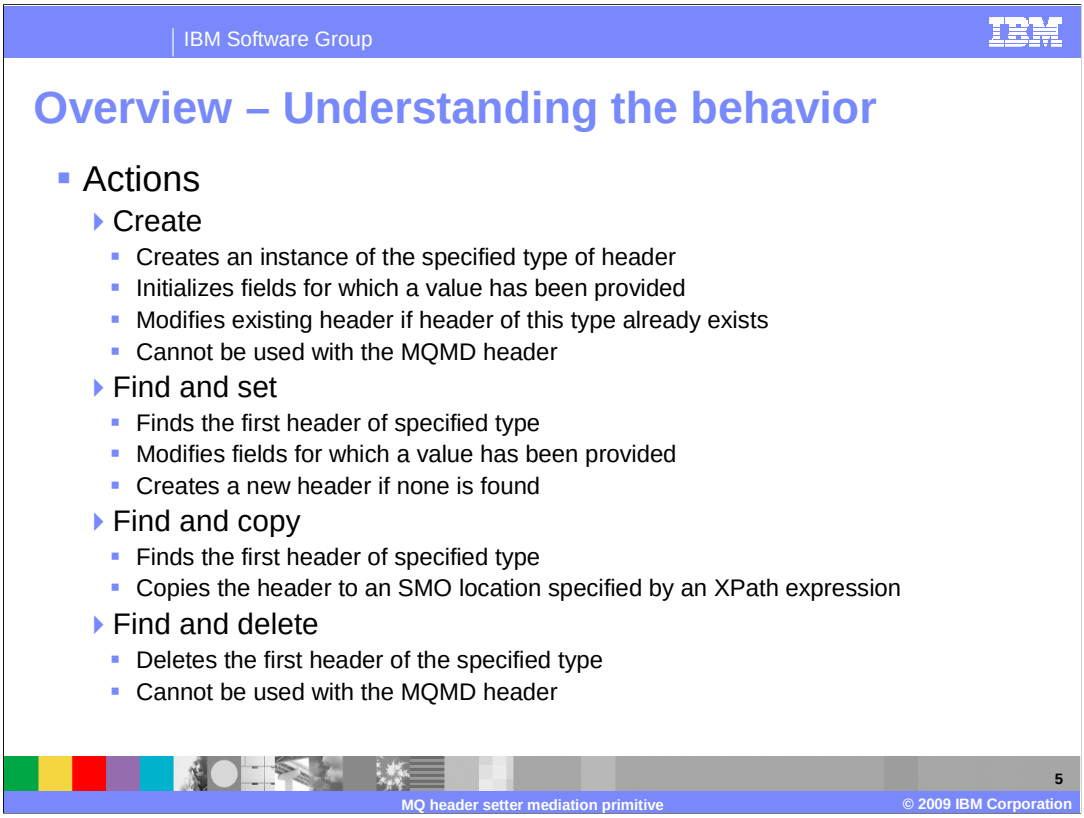
- The MQ header setter primitive:
  - A table contains the defined actions
    - Each row of the table defines an action for a specific header type
      - A dialog customized to the header type and action is presented to define the row
    - Actions are done sequentially row by row
    - Actions can build on previous actions within the primitive
  - Fields within a header can be set using:
    - A literal value
    - A value obtained from the SMO identified by an XPath expression
- MQ headers in the SMO defined by MQHeaderType
  - Are located at /headers/MQHeader
  - Contains:
    - Message descriptor
    - Control information
    - Sequence of chained headers



The MQ header setter primitive contains a table with the defined actions. Each row of the table contains an action to be performed for a specific header type. Editing the action is done using a dialog that is aware of the schema definition of the type of header being edited. This allows you to define settings that are specific to that header type. At runtime, the actions are done sequentially through the table, row by row. This allows an action in the table to build upon a previous action, such as creating a header in one action and copying it in a subsequent action.

When defining the values for fields in a setting dialog, the value can be specified as a literal value or can be an XPath expression to a location in the SMO containing the value.

The MQ headers are contained in the SMO at /headers/MQHeader. They are a sequence of MQHeaderType, as is shown in the screen capture on the slide.



To properly use the MQ header setter primitive, it is important to understand the specific behavior of the actions.

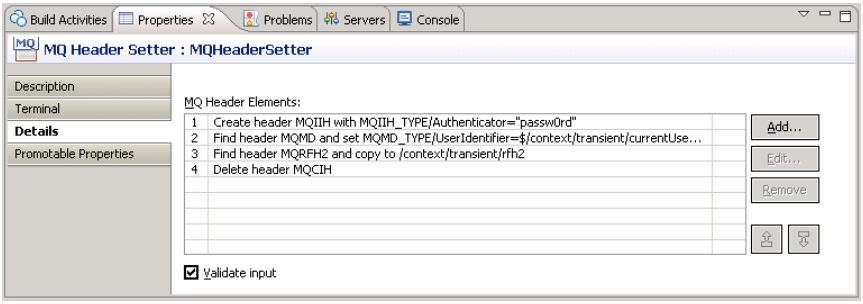
For the create action, a header of the specified type is created and fields within the header are set to the values provided. If there already is another header of the same type in the SMO, rather than creating a new header, the values are used to update the existing header. The create action is not a valid action for use with the MQMD header.

For the find and set action, a search is performed for the first header of the specified type. The defined set values are then used to update the header. If no headers are found of that type, a new one is created and initialized with the values provided.

The find and copy action searches for the first header of the specified type and the header is then copied to a location in the SMO defined by an XPath expression.

Use the find and delete action to delete an MQ header. The first header found of the specified type is deleted. The delete action is not a valid action for use with the MQMD header.

Properties



- MQ Header Elements table
  - List of actions to perform on headers
    - Headers are identified by MQ header type
  - Each row is for a specific header
  - Add/Edit properties dialog used to set individual rows
- Validate input
  - Validates if incoming message is of the expected type
  - Ensures it meets any constraints defined
    - For example, minOccurs, maxValue, and so on

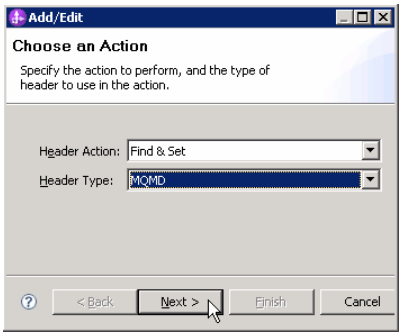
This slide looks at the Details panel of the Properties view of the MQ header setter. There is a table called MQ Header Elements which contains the list of actions to be taken. Each row defines an action which is associated with a specific type of MQ header.

The Add… and Edit… buttons open the Add/Edit properties dialog which is used to configure individual rows of the table. The dialog is specific to the type of MQ header.

The Validate input property is a check box used to indicate if incoming messages to the MQ header setter 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.

Properties – Add/Edit dialog

- The first panel
  - Always the “Choose an Action” panel
  - Contains two drop down boxes to select:
    - Action
    - Type of header
- Additional panels
  - Depends on the action
    - Delete – no further panels
    - Find and Copy – “Choose a Destination”
    - Create, Find and Set - “Set the Values”
  - Depends on the type
    - Set the values panel unique to header type



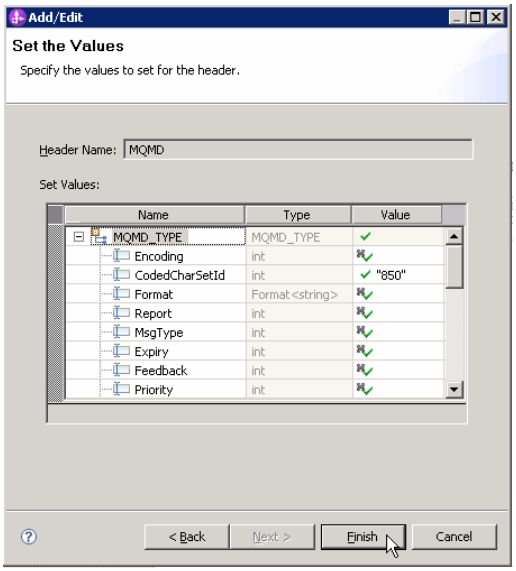
The Add/Edit properties dialog is used to define rows of the MQ header elements table and is customized according to the type of MQ header.

The dialog presents a couple of panels. The first panel is used to choose the action to take and defines the type of the MQ header. This panel is shown in the screen capture on the slide.

The subsequent panels that are displayed depend upon the action that was chosen. For delete, no further panels are displayed. For find and copy, the choose a destination panel is displayed that enables you to specify the destination where the header is copied. For create, and find and set actions, the set the values panel is displayed which allows you to set the values for fields in the header. The panel is specific to the type of header.

Properties – Add/Edit, Set the Values

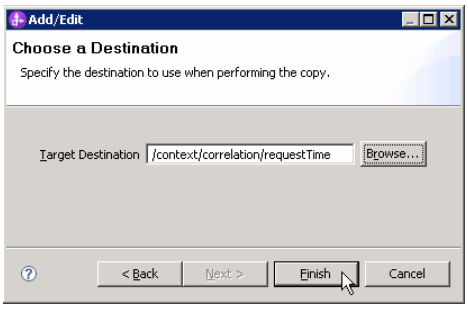
- Set Values
  - Specific to the header type selected
    - Example here is for MQMD
    - MQCIH and MQIIH are similar
    - MQRFH2 described later
  - Define values you want set in the header
    - Values can be literals
    - Values can be obtained from an SMO location defined by XPATH



This slide shows the Set the Values panel of the Add/Edit properties dialog. The screen capture on this slide shows the panel displayed when editing an MQMD header. The panels for the MQCIH and MQIIH headers are similar, showing a list of specific fields contained in each of those headers. The MQRFH2 header has a somewhat more flexible structure which is described later in the presentation.

Any values that you specify in this panel are set in the header. The values can be specified as literals or as XPath expressions to locations in the SMO containing the value to be used.

Properties – Add/Edit, Choose a Destination



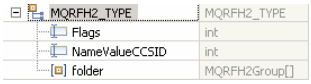
- Target Destination
  - XPath defining SMO location where header is copied
  - Browse… button opens the XPath expression builder dialog
  - Target location type must be compatible with a MQ header type

This slide shows the Choose a Destination panel of the Add/Edit properties dialog used when the find and copy action is chosen.

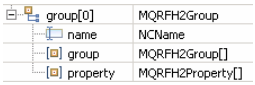
The destination is specified as an XPath expression to the SMO location that is the target of the copy. The Browse… button opens the XPath expression builder dialog which helps you to build the expression. The location specified must be compatible with the MQ header to be copied there.

Properties – Add/Edit – MQRFH2 Set the Values

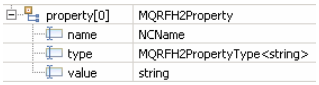
- MQRFH2 headers have a generic structure which is reflected in the Set the Values panel
  - Header contains flags, character set ID and folders



  - Folders are groups which contain a name, other groups and properties



  - Properties contain a name, type and value



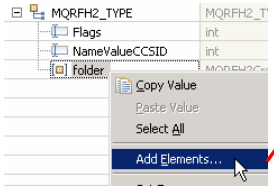
The next few slides describe the panel used to set the values for an MQRFH2 header. This header has a generic structure, and the panel reflects this. You must expand the structure in the panel based on what you need to do.

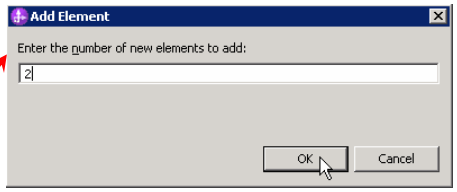
The basic MQRFH2 header contains a flag field, a character set ID field and a sequence of folders. This is shown in the first screen capture. An instance of a folder is a group. Each group contains a name and a sequence of contained groups and a sequence of properties. A group is shown in the second screen capture. A property contains a name, type and value as you can see in the last screen capture.

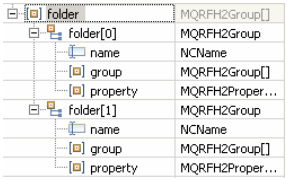
Properties – Add/Edit – MQRFH2 Set the Values

- Editing the MQRFH2 – the starting point

- Adding folders





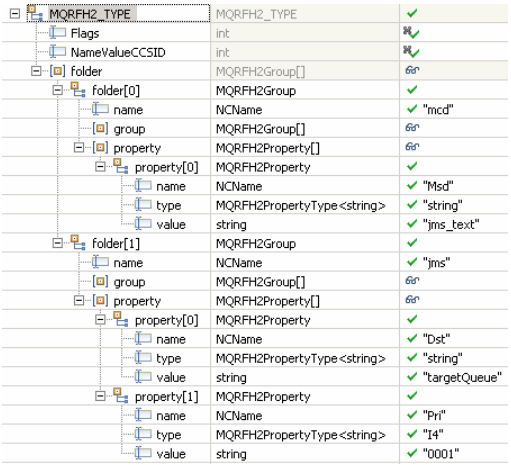


- Groups and properties added in the same way

This slide shows how you expand the MQRFH2 structure to enable you to set field values. When you first open the Add/Edit dialog, the MQRFH2 structure looks like the screen capture at the top of the slide. Other than the flags and character set ID, you need to expand the structure to provide values in the header. That is done by getting a pop-up menu for folder, selecting Add Elements… and specifying the number of elements to add in a dialog box. In the example shown on the slide, there were two folders added as you can see in the lower screen capture. Each folder is actually a group. Within each group, you add other groups and properties in exactly the same manner.

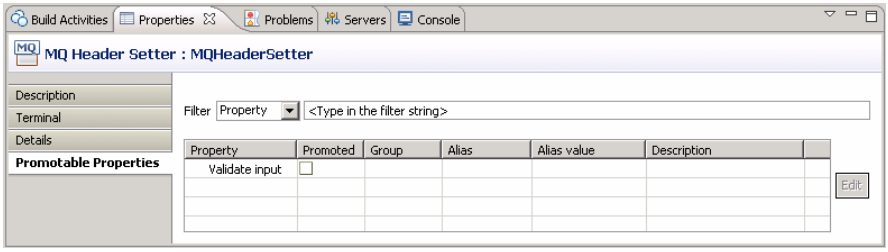
Properties – Add/Edit – MQRFH2 Set the Values

- After adding groups and properties, fill in the appropriate values
- Example result



This slide provides an example of what the panel looks like after adding a couple of folders, adding some properties within the folders and specifying values for the fields.

Promotable properties



- Promotable
  - Validate input
- Not promotable
  - MQ Header Elements

This slide shows the Promotable Properties panel. The MQ Header Elements table is not promotable. The Validate input property is promotable. This enables the ability to turn validation checking on and off administratively, which allows production environments to run with better performance while enabling validation to be turned on for debugging when needed.

Error processing

- MediationBusinessException (fail terminal flow)
  - Create, Find and Set – source XPath specifies value that does not exist
  - Create, Find and Set – source XPath specifies value of incompatible type
  - Find and Copy – target XPath specifies location of incompatible type
  - Validate input specified and message fails validation testing
- No entry in MQ header elements table
  - This is not an error, SMO is propagated unchanged

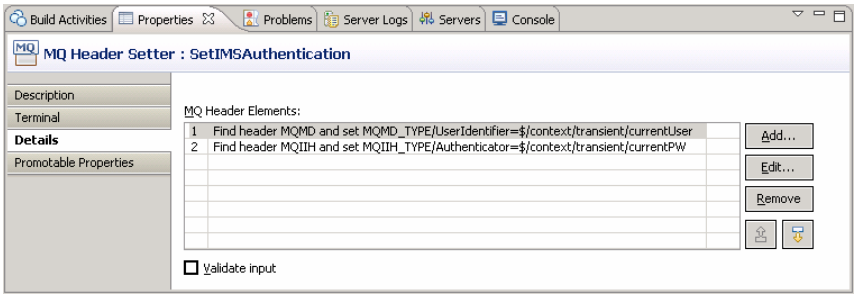
The error processing details and considerations are examined in this slide.

The MediationBusinessException causes the fail terminal to be fired. One of the cases that cause this exception to occur is a create, or find and set action that specifies a value using a source XPath expression to a location that does not exist in the SMO. Another possible cause is also for create, or find and set, and is when the source XPath expression specifies a location whose value is of an incompatible type with that of the element being set. A third reason for this exception is when a find and copy specifies an XPath location whose type is incompatible with the MQ header being copied there. Another cause of the MediationBusinessException is when the validate input property has been specified and the message fails the validation processing.

The case where the MQ header elements table is empty is not considered an error. The SMO is propagated unchanged through the out terminal.

Example usage

- Example - calling IMS transaction requiring authentication
  - Flow logic places the user ID and password in the transient context
  - In the MQ header setter primitive:
    - MQMD must have UserIdentifier set to the user ID
    - MQIIH must have Authenticator set to the password



This slide introduces an example usage of a MQ header setter primitive.

In the example, the mediation flow makes a call to an IMS system using MQ. The IMS transaction being called requires a user ID and password to authenticate the caller. The user ID needs to be passed in the MQMD header and the password passed in the MQIIH header. In the mediation flow, the user ID and password for the current user are placed into the transient context. The MQ header setter configuration shown in the screen capture does two find and set actions to set these values in the MQMD and MQIIH headers, copying the values from the transient context.

Summary

- Examined the MQ header setter mediation primitive

MQ header setter

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

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

This module is also available in PDF format at: ../WBPMv62_MQHeaderSetterPrimitive.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:

CICS 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.