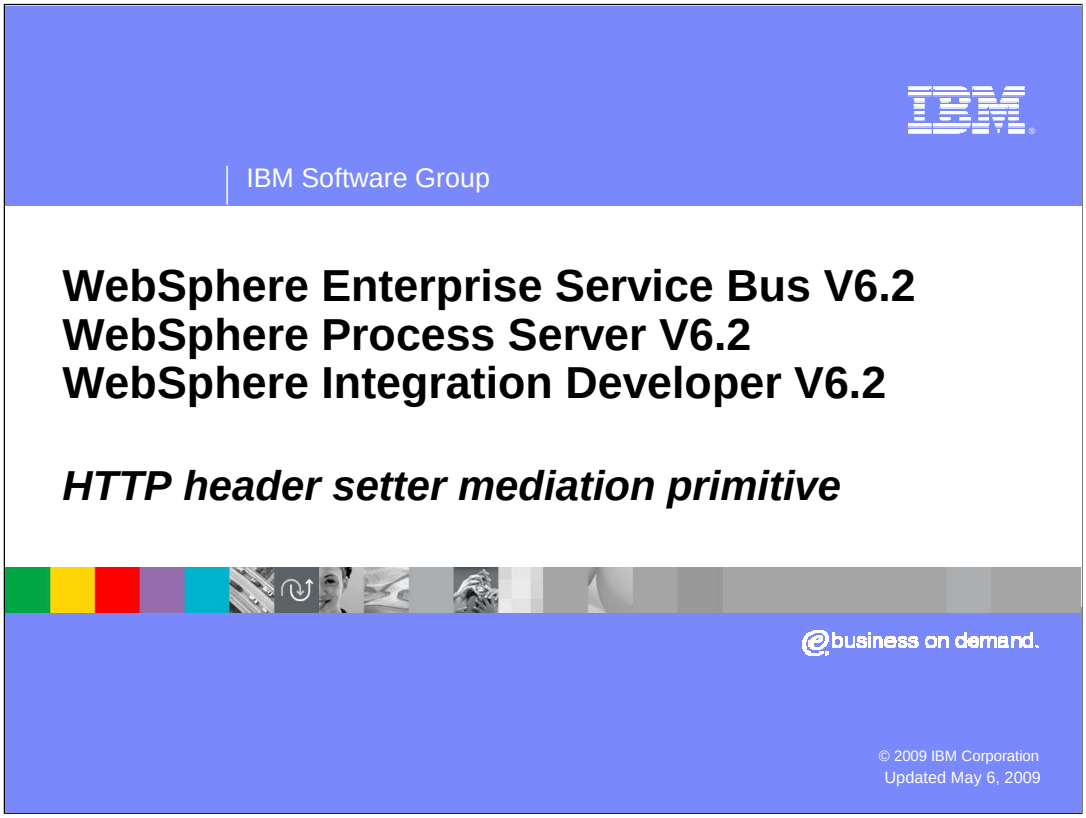


This presentation provides a detailed look at the HTTP header setter mediation primitive, which is a new primitive introduced in version 6.2.

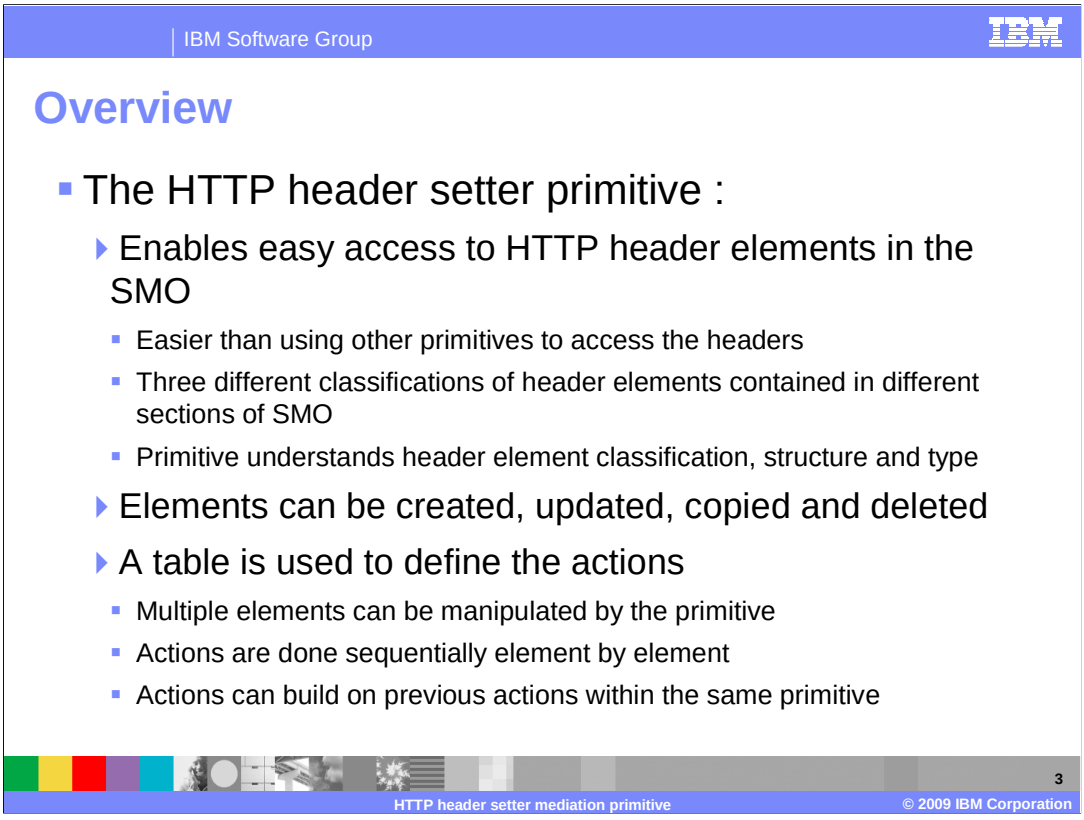
Goals

- Understand the HTTP header setter mediation primitive

HTTP header setter

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

The goal is to provide you with a full understanding of the HTTP 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 HTTP header setter primitive specific material in this presentation. The presentation contains an overview of the function provided by the HTTP 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 an HTTP header setter primitive is provided.

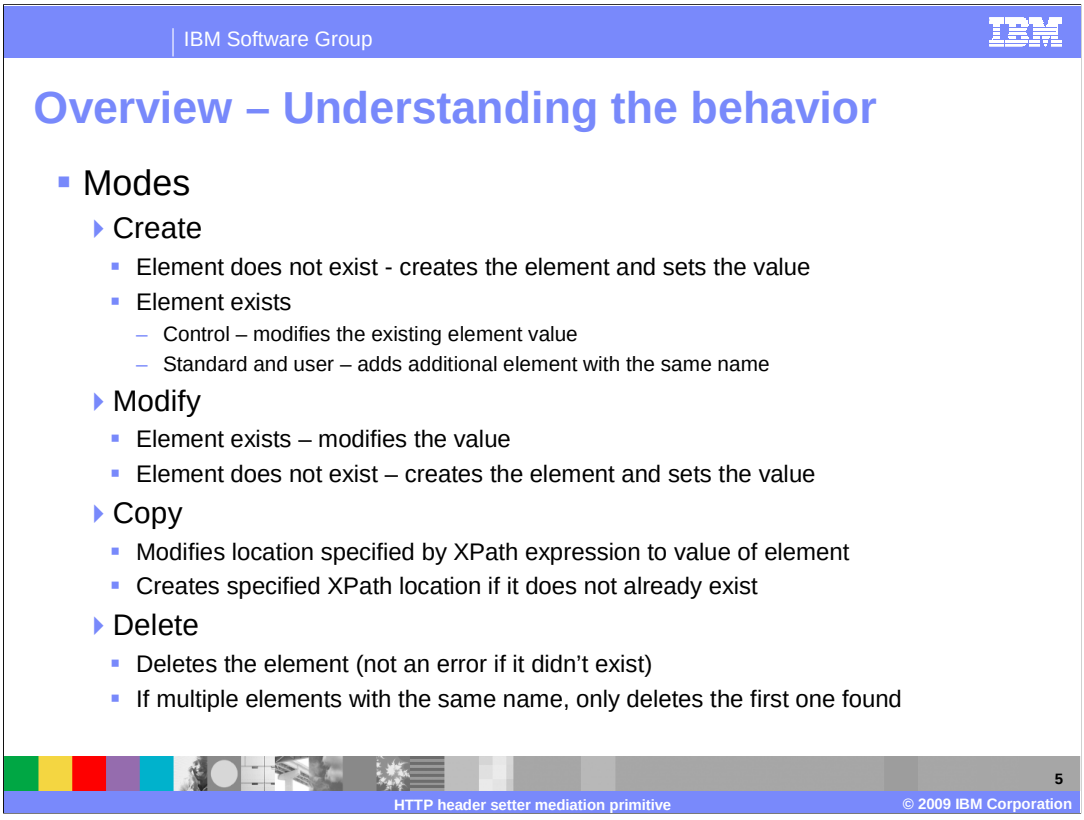


The purpose of the HTTP header setter primitive to enable access to the HTTP header properties, which are elements within the headers section of the service message object. Although these elements can be manipulated with other primitives, it is not always easy to do so due to the optional presence of these elements and the difference in structure of the different classifications of HTTP properties. Using the HTTP header setter makes access to these elements much easier as the primitive is aware of how these are represented in the SMO. Using this primitive, you can create, update, copy or delete HTTP header elements. The primitive is configured with a table that defines the sequence of actions. Each row of the table references a single HTTP property. The table is processed in order and operations that affect the same property can build on one another. For example, the action create followed by the action copy enables you to create the element for a particular property, set its value and then copy the value to another location in the SMO.

Overview – Header classification

- Classification of HTTP header elements
  - Control elements defined by SCA for HTTP bindings
    - Located in SMO at /headers/HTTPHeader/control
      - Described by the interface com.ibm.websphere.http.headers.HTTPControl
    - Documentation located in the Information Center page “HTTP Headers”
  - Standard elements defined by the HTTP specification
    - Located in SMO at /headers/HTTPHeader/header
      - Sequence of name/value pairs
    - Documentation:
      - Field header definitions in HTTP 1.1 specification
      - http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
  - User elements are application specific
    - Located in SMO at /headers/properties
      - Sequence of user defined name/type/values

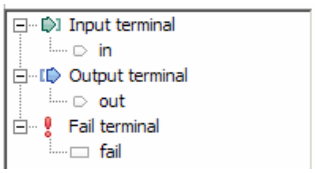
The three different classifications of HTTP header elements are described here. The first grouping are the control elements, which are defined by service component architecture for use with HTTP bindings and are located in the SMO at /headers/HTTPHeader/control and are described by the interface com.ibm.websphere.http.headers.HTTPControl. The documentation for these is found in the Information Center on the page entitled “HTTP headers”. The documentation describes the use of each of these elements as they apply to HTTP exports and imports for both inbound and outbound message flows. The next grouping are the standard elements defined by the HTTP specification, located in the SMO at /headers/HTTPHeader/header, which is a sequence of name value pairs. The names of these elements along with their defined usage is documented by the World Wide Web Consortium in the HTTP 1.1 specification, which can be found at the URL provided on the slide. The final grouping are user defined HTTP header elements, found in the SMO at /headers/properties, which is a sequence of name type value triplets. The meaning of these are application specific.



To properly use the HTTP header setter primitive, it is important to understand the specific behavior of the actions, which are referred to as modes. For the create mode, if the element does not exist, it is created and the value set. If the element exists, the behavior depends upon what classification of HTTP header element it is. For a control HTTP element, the create mode will update the existing value to the new value. For a standard or user HTTP header element, the create mode will add an additional element, resulting in multiple elements with the same name, with the newly created one placed at the end of the sequence. For the modify mode, if the element exists it is updated to the new value. If the element does not exist, it is created and the value set. When dealing with standard and user header elements, the use of modify might be preferred over create in that there is no possibility of creating duplicate entries for the same property. The copy mode copies the value of the header element to a location in the SMO, which is identified using an XPath expression. If the SMO location of the target does not yet exist, it is created. Use the delete mode to delete an HTTP header element. If an element for the property does not exist, it is not considered an error. If there are multiple instances of the same header element, only the first instance is deleted.

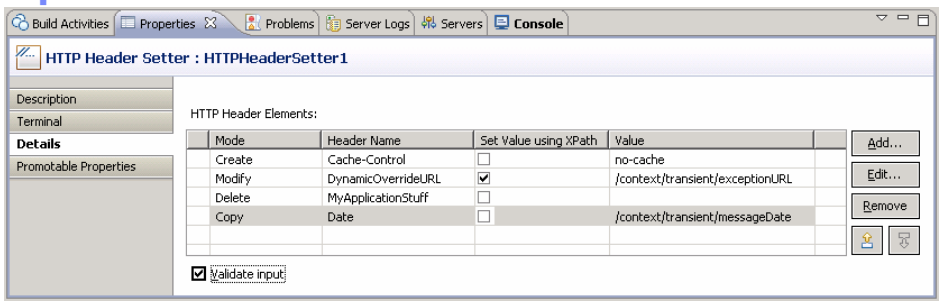
Terminals

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



The HTTP header 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, because the HTTP header setter primitive does not modify the message body. Shown here is an HTTP header setter primitive with its terminals and the terminals as seen in the properties view.

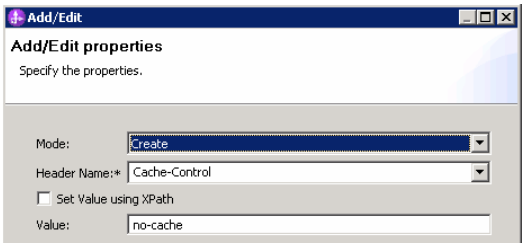
Properties



- HTTP Header Elements table
  - List of the defined actions and settings
  - Each row is for an individual property
  - Add/Edit properties dialog used to set individual rows
  - Values can be static or obtained from the SMO using XPath
- 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 HTTP header setter. There is a table called HTTP Header Elements which contains the list of actions to be taken along with configuration settings for each action. Each row defines an action to be taken for a single property. The Add… and Edit… buttons open the Add/Edit properties dialog which is used to configure individual rows of the table. The value when using the create or modify modes can be specified directly in the table, or alternatively can be an XPath expression to a source location in the SMO. The Validate input property is a check box used to indicate if incoming messages to the HTTP 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.

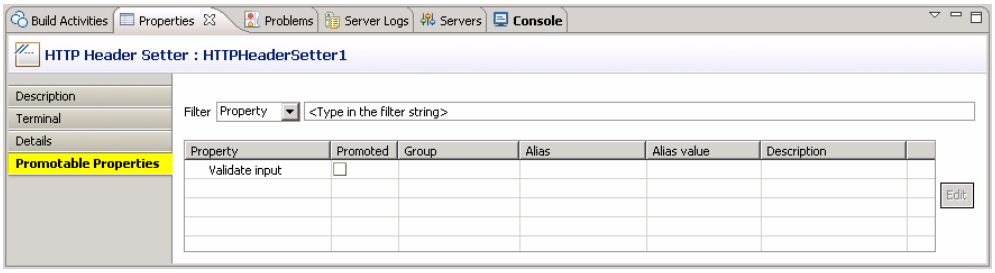
Properties



- Add/Edit properties dialog
  - Mode: – select from drop down (Create, Modify, Copy, Delete)
  - Header Name:*
    - Select from drop down for control and standard HTTP header property
    - Type in property name for user defined property
  - Type: – the property type, settable if a user defined property
  - Set Value using XPath – select to obtain value from element in the SMO
  - Value
    - Value to set or XPath to source value (Create or Modify)
    - XPath to target value (Copy)
    - Leave blank (Delete)

The Add/Edit properties dialog used to define rows of the HTTP header elements table shown on the previous slide is described here. The mode field is a drop down box allowing you to set the action to be performed. The choices are Create, Modify, Copy and Delete. The Header Name field is where you specify the name of the HTTP header element to be acted upon and is a combination of a drop down box and text entry field. If the property is a control or standard HTTP header element, it can be selected from the drop down. However, if you are specifying a user property, it must be typed directly into the field. The Type field is only needed when the property being set is an HTTP user property. It defines the type of the property, with valid types being the Java™ primitive types and String. The selection is made using a drop down box. The Set Value using XPath selection box is only needed when the mode is either create or modify. When selected, it indicates that the value is actually an XPath expression defining the source location for the value within the SMO. The Value field contains the value associated with the action. If the mode is Create or Modify and the Set Value using XPath is not set, it contains the value for the property. If the mode is Create or Modify and Set Value using XPath is set, or if the mode is copy, the value field contains an XPath expression identifying a location in the SMO. When this is the case, the Edit… button opens the XPath Expression Builder dialog to help you create the appropriate XPath expression. Finally, for the Delete mode, this field is not set.

Promotable properties



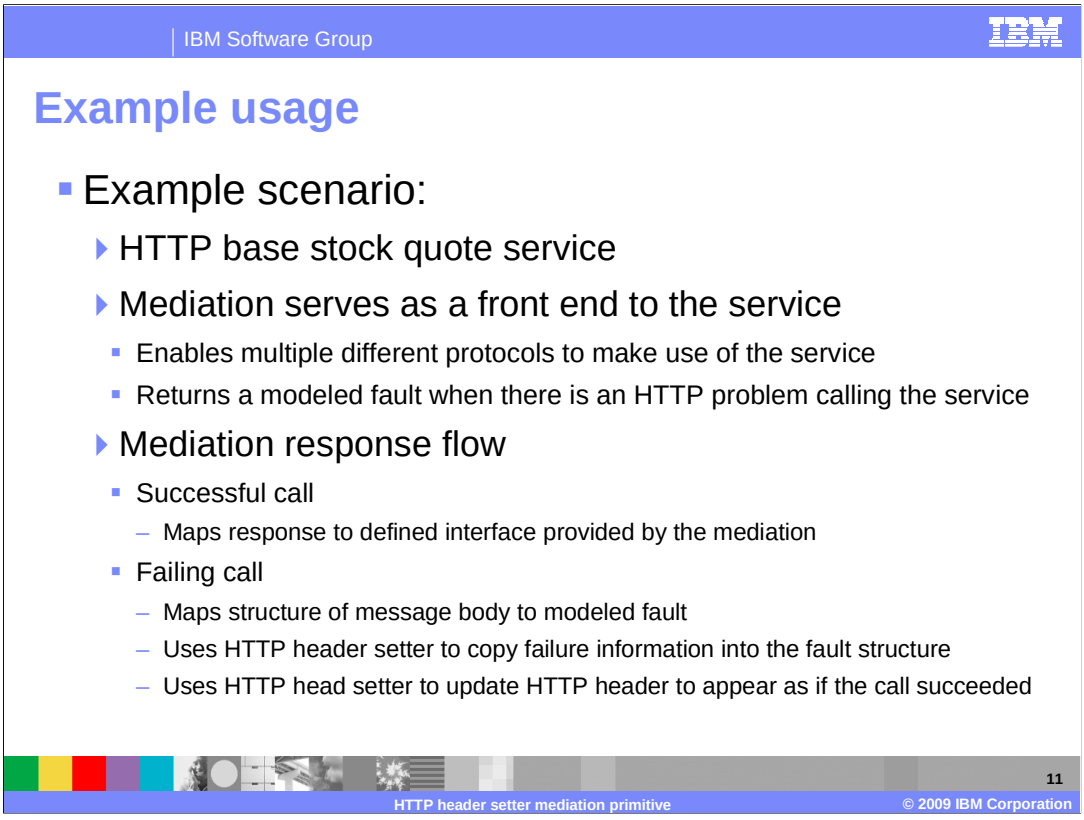
- Promotable
  - Validate input
- Not promotable
  - HTTP Header Elements

Shown here is the Promotable Properties panel. None of the columns in the HTTP Header Elements table are promotable. The Validate input property is promotable. This enables the ability to turn validation checking on and off administratively, allowing production environments to run with better performance while enabling validation to be turned on for debugging when needed.

Error processing

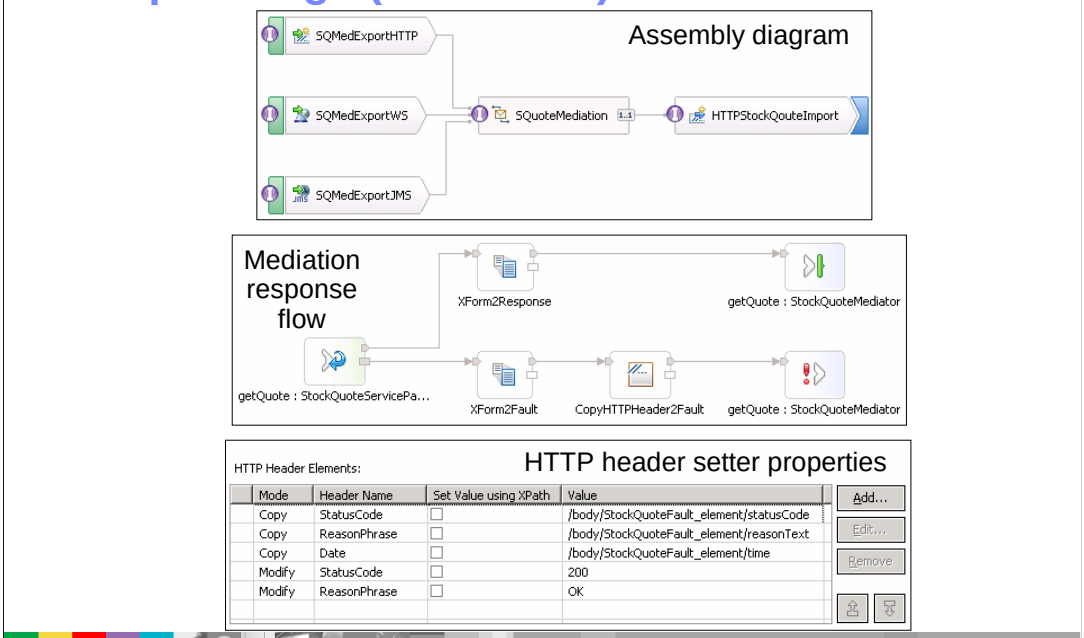
- MediationRuntimeException thrown for:
  - Create, modify or copy - value column is blank
- MediationBusinessException (fail terminal flow)
  - Create or modify – XPath specifies value that does not exist
  - Copy – target XPath specifies value of incompatible type
  - Create, modify or copy – XPath expression syntax error
  - Validate input specified and message fails validation testing
- No entry in HTTP header elements table
  - This is not an error, SMO is propagated unchanged

The error processing details and considerations are examined in this slide. A MediationRuntimeException is thrown for create, modify or copy modes if the value column has been left blank. The MediationBusinessException causes the fail terminal to be fired. This occurs for a create or modify when the source value identified by an XPath expression does not exist in the SMO. It also occurs when the XPath for the target specifies an SMO element whose type is incompatible with the type of the property to be copied. This exception can also occur for a create, modify or copy containing an XPath expression that has a syntax error. Another cause of the MediationBusinessException is when the validate input property has been specified and the message fails validation processing. The case where the HTTP header elements table is empty is not considered an error . The SMO is propagated unchanged through the out terminal.



Introduced here is an example usage of an HTTP header setter primitive. The scenario is for an HTTP based stock quote service on the back end and a mediation through which it is called. The mediation enables multiple protocols, such as HTTP, Web service or SCA, to be used to call the HTTP service. In the event that the HTTP service cannot be called successfully, the mediation converts the HTTP error into a modeled fault to be returned to the caller. The interface presented by the mediation is different than the interface used to call the HTTP stock quote service. Therefore, in the case of a successful call, the mediation response flow needs to map the response to the appropriate return value to send back to the caller. In the case of a failing call to the HTTP stock quote service, the response flow must change the body of the message to that for the modeled fault. Then the HTTP header setter is used to copy information about the failure from the control and standard HTTP headers in the SMO to the body of the message. The HTTP header setter is used to update the HTTP control header to appear as if the call succeeded. This is necessary so that the failure is not propagated back to the original caller as if there was not a modeled fault.

Example usage (continued)



This slide contains screen captures illustrating the example described on the previous slide. The top portion shows the assembly diagram containing the exports for the various protocols supported by the mediation, the mediation flow component and the HTTP import. The center of the slide shows the mediation response flow. For a normal response, the XForm2Response XSL transformation maps to the appropriate response format. In the event of a failure in the HTTP call to the service, the fail flow uses the XForm2Fault XSL transformation to map the message body to the appropriate format for the fault. The CopyHTTPHeader2Fault HTTP header setter then initializes the values in the fault message body. This can be seen in the bottom screen capture. The StatusCode and ReasonPhrase are copied from the control HTTP header to the message body. The time and date of the failure is copied from the standard HTTP header to the message body. Finally, the StatusCode and ReasonPhrase are modified to look as if the HTTP call was successful, so that the HTTP error is not propagated directly back to the caller.

Summary

- Examined the HTTP header setter mediation primitive

HTTP header setter

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

In summary, this presentation provided details regarding the HTTP 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 an HTTP 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_HTTPHeaderSetterPrimitive.ppt

This module is also available in PDF format at:
../WBPMv62_HTTPHeaderSetterPrimitive.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

Java, and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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.