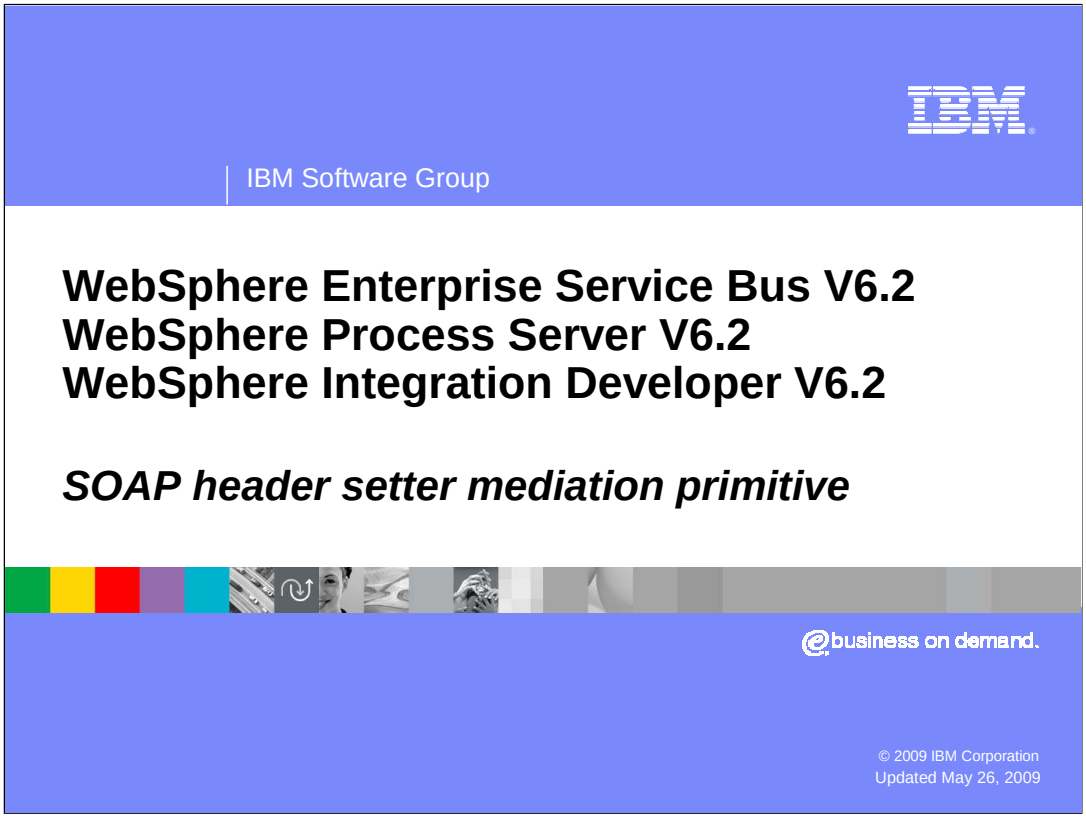


This presentation provides a detailed look at the SOAP header setter mediation primitive, which is a new primitive introduced in version 6.2.

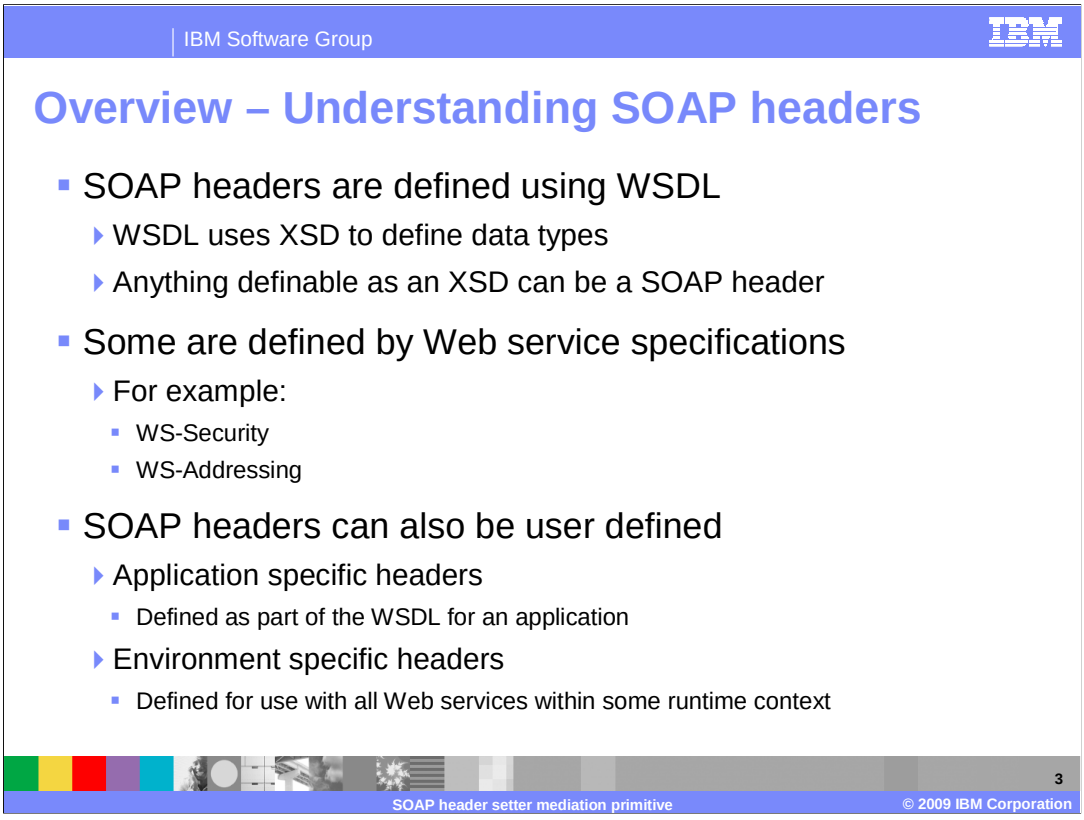
Goals

- Understand the SOAP header setter mediation primitive

SOAP 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 SOAP header setter mediation primitive. It is assumed 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 SOAP header setter primitive specific material in this presentation. This presentation contains an overview of the function provided by the SOAP 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 SOAP header setter primitive is provided.



Provided here is some background on the nature of SOAP headers, enabling you to better understand the description of the SOAP header setter primitive that follows. SOAP headers are defined using the Web Service Definition Language (WSDL), which uses XML Schema Datatypes (XSD) as the mechanism for defining a type for a SOAP header. Because of this, SOAP headers are extremely flexible, so there can be a wide range of different SOAP headers. Some SOAP headers are defined as part of a Web service specification. WS-Security and WS-Addressing are two specifications that provide examples of SOAP headers defined by a specification. Other SOAP headers are user defined. Some user defined SOAP headers are specific to an application and are defined within the WSDL for the application. However, some SOAP headers are not specific to an application, but rather are used within some environmental context, such as all the Web services run within an enterprise.

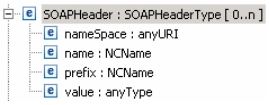
Overview

- The SOAP header setter primitive:
  - Enables easy access to SOAP headers in the SMO
    - Much easier than using other primitives to access the headers
  - Uses the XSD definition for a header
    - XSD must be available to the primitive during development and runtime
    - The XSD is used to present a header specific dialog during development
    - Weakly typed fields within a header can be downcast to more specific types
  - The primitive works on entire headers
    - Create a new header and initialize selected elements
    - Locate an existing header and update selected elements
    - Locate an existing header and copy it to a different location in the SMO
    - Locate an existing header and delete it

The SOAP header setter primitive enables access to SOAP headers in the SMO, providing a much easier mechanism to access the headers than is possible using other primitive types. The primitive makes use of the XSD that defines a SOAP header. The XSD must be available for use during both development of a mediation and at runtime. In WebSphere® Integration Developer, the XSD is used to present a header specific dialog, which you use to define the actions associated with the header. Because some headers contain weakly typed fields, the header specific dialog allows you to downcast the weakly typed field to a more specific type when needed. The primitive works by defining actions that are associated with an entire header. There are four actions. The create action adds a new SOAP header to the SMO and initializes selected elements within the header. The find and set action locates an existing header based on header type, possibly qualified by values of specific elements, and then updates other selected elements within the header. The find and copy action locates an existing header based on header type, possibly qualified by values of specific elements, 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, possibly qualified by values of specific elements, and deletes the entire header.

Overview

- The SOAP 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
  - Elements within a header can be set using:
    - A literal value
    - A value obtained from the SMO identified by an XPath expression
- SOAP headers in the SMO
  - Are located at /headers/SOAPHeader
    - SOAPHeader is an array of SOAPHeaderType



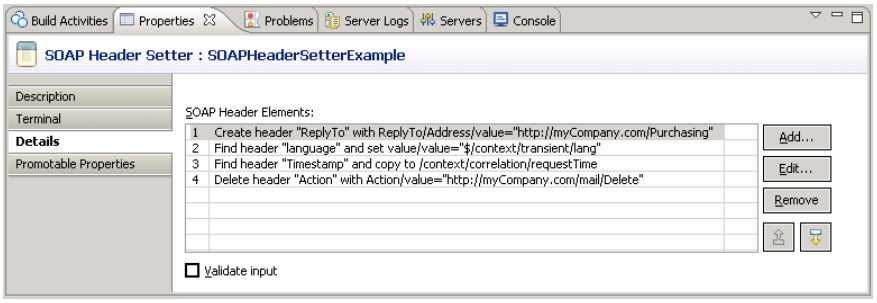
The SOAP header setter primitive contains a table with the defined actions. Each row of the table contains an action to be performed for a specific header type. Editing the action is done using a dialog that is aware of the XSD definition of the header. This allows you to define search criteria and settings 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 elements in the search dialog or setting dialog, the element value can be specified as a literal value or it can be an XPath expression to a location in the SMO containing the value. The SOAP headers are contained in the SMO at /headers/SOAPHeader. They are a sequence of SOAPHeaderType, as shown here in the screen capture.

Overview – Understanding the behavior

- Actions
  - Create
    - Creates an instance of the specified type of header
    - Initializes elements for which a value has been provided
  - Find and set
    - Finds all headers of specified type with specified element values
    - Modifies elements for which a value has been provided
    - Creates a new header if none is found
  - Find and copy
    - Finds the first header of specified type with specified element values
    - Copies the header to an SMO location specified by an XPath expression
  - Find and delete
    - Deletes all headers of the specified type with specified element values

To properly use the SOAP 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 elements within the header are set to the values provided. A new header is always created, even if there already is another header of the same type in the SMO. For the find and set action, a search is performed for all headers of the specified type that have element values matching the specified search criteria. The defined element set values are then used to update each of the headers found in the search. If no headers are found in the search, 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 that has element values matching the specified search criteria. The header is then copied to a location in the SMO defined by an XPath expression. Use the find and delete action to delete SOAP headers. A search is performed for all headers of the specified type that have element values matching the specified search criteria. All of the matching headers are deleted.

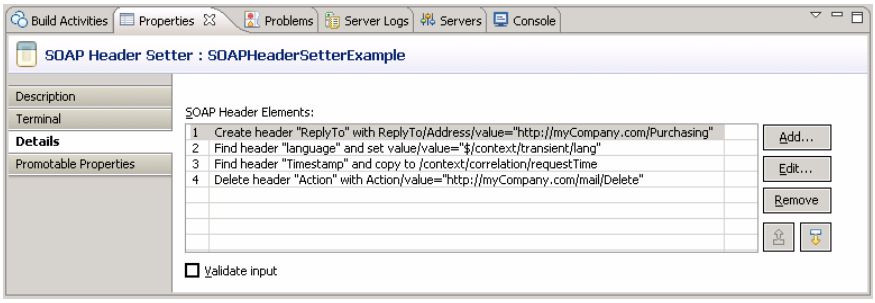
Properties



- SOAP Header Elements table
  - List of actions to perform on headers
    - Headers are identified by XSD type
    - Optionally, element values within the header are used as qualifiers
  - Each row is for a specific header or group of similar headers
    - The actions Find and Set and Find and Delete apply to all matching headers
    - The actions Create and Find and Copy apply to only one header
  - Add/Edit properties dialog used to set individual rows

Shown here is the Details panel of the Properties view of the SOAP header setter. There is a table called SOAP Header Elements, which contains the list of actions to be taken with each row defining an action. For the create, find and copy actions, the row contains an action that applies to only one header. For the find and set action and find and delete action, the row applies to all headers that match the search criteria. The search criteria identifies headers by the XSD type of the header, combined with optional qualifier values for elements within the 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 XSD type of the header.

Properties



- Validate input
  - Validates if incoming message is of the expected type
  - Ensures it meets any constraints defined
    - For example, minOccurs, maxValue, and so on

The Validate input property is a check box used to indicate if incoming messages to the SOAP 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 introduction

- Introduction to the Add/Edit properties dialog
  - Used to configure a row in the table
  - Presents a series of panels
    - The first panel is always the “Choose an Action” panel
    - The header type is also selected on this panel
  - The additional panels presented depend upon the action selected
    - Create – “Set the Values”
    - Find and Set – “Define the Search”, “Set the Values”
    - Find and Copy – “Define the Search”, “Choose a Destination”
    - Find and Delete – “Define the Search”

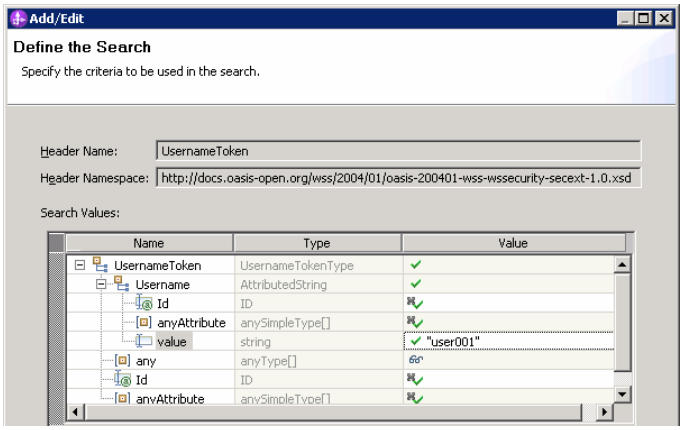
The Add/Edit properties dialog used to define rows of the SOAP header elements table is customized according to the XSD type of the header. The next several slides describe the Add/Edit dialog used with this primitive. The dialog presents a series of panels. The first panel is used to choose the action to take and defines the XSD type of the header. The subsequent panels that are displayed depend upon the action that was chosen. For the create action, a panel is presented to allow you to set the element values for the new header. For the find and set action there are two panels presented. The first is a search panel that allows you to specify qualifier values for header elements used to perform the search. The next is a panel that allows you to set the element values for elements to be updated. For the find and copy action, the same search panel is used, followed by a panel that allows you to specify the destination where the header is copied. And finally, the find and delete action just makes use of the search panel. The following slides show screen captures of these panels.

Properties – Add/Edit, Choose an Action

- Action – drop down box to select the action
- Header Element
  - Defines the header name, namespace and type
  - The Browse… button opens a selection dialog
    - The dialog contains XSDs that define a type and header element of that type
    - Standard header types can be included using the project dependencies
      - For example, WS-Addressing, WS-Security

Shown here is the Choose an Action panel of the Add/Edit properties dialog. The action is specified using a drop down box from which you select the appropriate action. The header element is defined by the name of the header, its namespace and its type. These are set by using the Browse… button which displays a list of all XSDs which define a type and a header element of that type. For the standard SOAP headers, such as WS-Addressing and WS-Security, the type definitions are included in your project through use of the project dependencies editor.

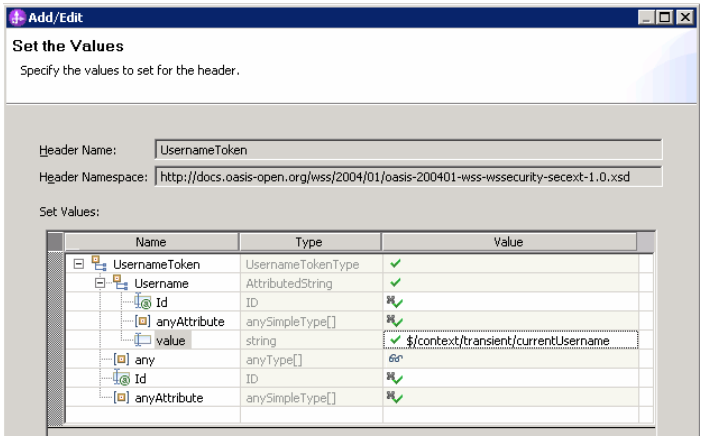
Properties – Add/Edit, Define the Search



- Search Values
  - Content is specific to the header type selected
  - Define values you want considered in the search
    - Values can be literals or obtained from an SMO location defined by XPATH

Shown here is the Define the Search panel of the Add/Edit properties dialog. The content of this panel is specific to the header type chosen, as defined by the XSD for the header. Any values that you specify in this panel are used as qualifiers in the search for headers of this type. The values can be specified as literals or as XPath expressions to locations in the SMO containing the value to be used.

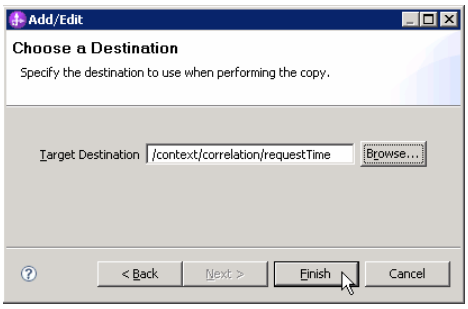
Properties – Add/Edit, Set the Values



- Set Values
  - Display is specific to the header type selected
  - Define values you want set in the header
    - Values can be literals or obtained from an SMO location defined by XPATH

Shown here is the Set the Values panel of the Add/Edit properties dialog. The content of this panel is specific to the header type chosen, as defined by the XSD for the header. 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.

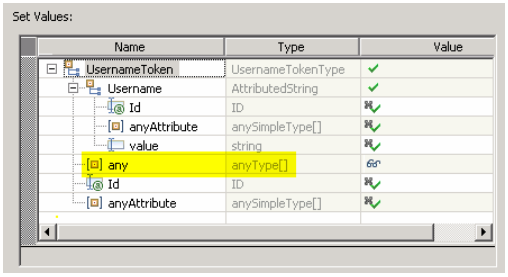
Properties – Add/Edit, Choose a Destination



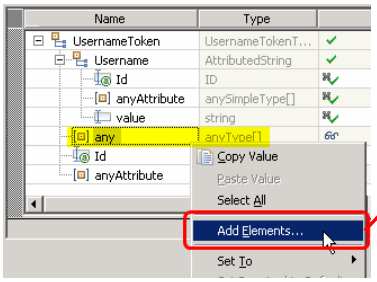
- Target Destination
  - XPath defining SMO location where header is copied
  - Browse… button opens the XPath expression builder dialog
  - Target location type must be compatible with a SOAP header type

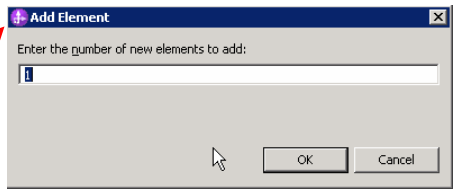
Shown here is the Choose a Destination panel of the Add/Edit properties dialog. The destination is specified as an XPath expression to the SMO location that is the target of the copy. The Browse… button opens the XPath expression builder dialog which helps you to build the expression. The location specified must be compatible with holding a value of a SOAP header type.

Properties – Add/Edit – Set and Search Values



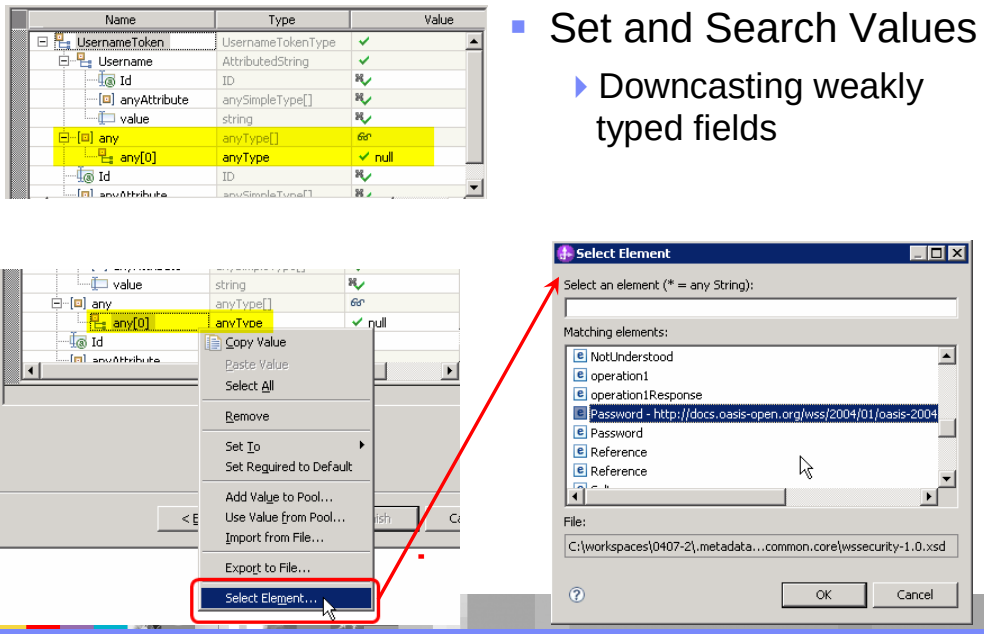
- Set and Search Values
  - Defining cardinality of sequences





This slide describes how both the Set and Search panels are edited when defining the number of elements in a sequence. For an element that is defined as a sequence, the pop-up menu contains an Add Elements… selection which opens the Add Element dialog, which allows you to add additional elements to the sequence.

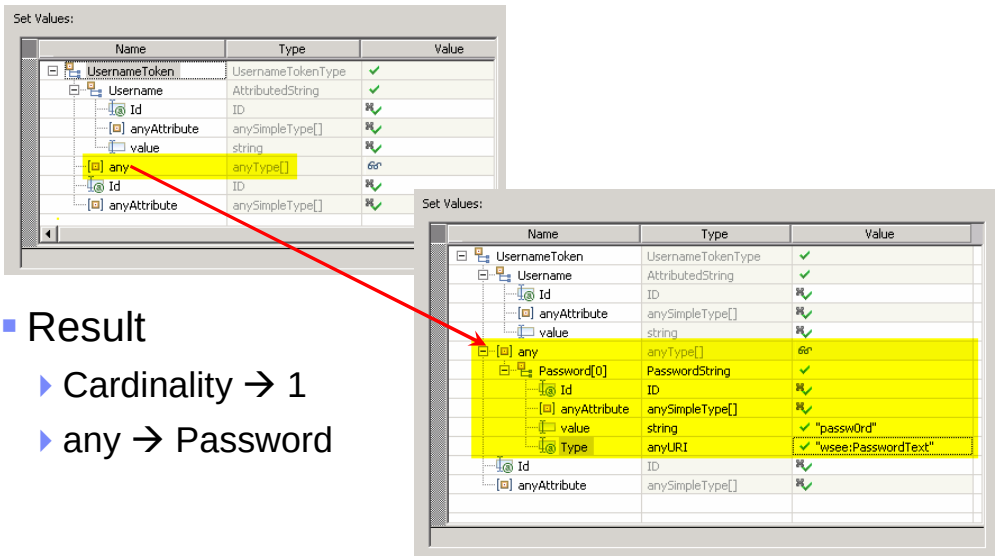
Properties – Add/Edit – Set and Search Values



- Set and Search Values
  - Downcasting weakly typed fields

This slide shows how both the Set and Search panels are edited when downcasting a weakly typed field to a stronger type. For a weakly typed field, such as an anyType, the pop-up menu contains a Select Element… choice, which opens a Select Element dialog that displays the possible types from which you can select.

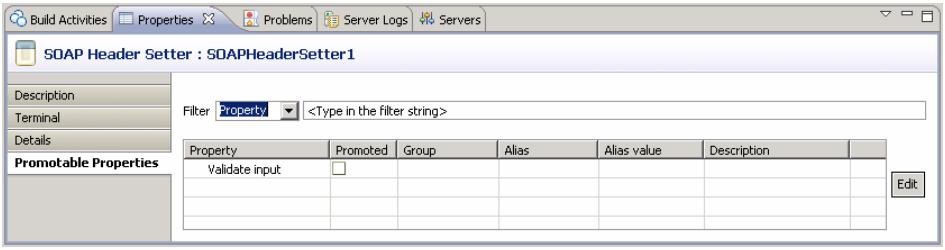
Properties – Add/Edit – Set and Search Values



- Result
  - Cardinality → 1
  - any → Password

This slide shows the result of the previous two slides. In the upper left is a screen capture showing a sequence called any of type anyType which has no elements. The screen capture on the lower right shows the result after adding one element and then downcasting the element to be of type Password.

Promotable properties



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

This slide shows the Promotable Properties panel. The SOAP 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 SOAP header elements table
  - This is not an error, SMO is propagated unchanged

The error processing details and considerations are examined here. 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 an element value using a source XPath expression to a location that does not exist in the SMO. Another possible cause is for create or find and set, and is when the source XPath expression for an element specifies a location whose value is on 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 SOAP 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 SOAP header elements table is empty is not considered an error .The SMO is propagated unchanged through the out terminal.

Example usage

- Example scenario:
  - Target Web service requires SOAP UsernameToken header to contain the username and password

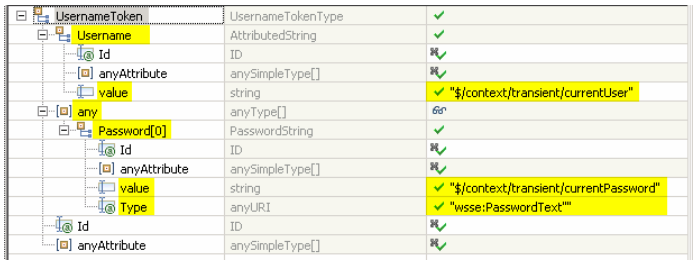
```
<soapenv:Header>
    <_0:UsernameToken
                xmlns:_0="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <_0:Password Type="wsse:PasswordText">passw0rd</_0:Password>
        <_0:Username>JoeUser</_0:Username>
    </_0:UsernameToken>
</soapenv:Header>
```

  - Steps needed to define the header
    - Add schema files to project
      - Dependencies →Predefined Resources → WS-Security 1.1 schema files
    - Add SOAP header setter to the mediation flow

This slide introduces an example usage of a SOAP header setter primitive. In the example, there is a Web service which requires the SOAP UsernameToken header containing both the username and the password. This is a SOAP header defined by the WS-Security specification. In the slide you can see an example of what the header looks like within a SOAP message. To add this header to a message, you must follow these steps. First, the schema files for the WS-Security 1.1 specification must be added to your project, which can be done using the Dependencies editor. Next, a SOAP header setter primitive must be added to your mediation flow.

Example usage (continued)

- Steps needed to define the header (continued)
  - Define a Create action with UsernameToken as the header element
    - Set Username/value to an XPath value used to obtain the username
    - Add an element to the any sequence (see cardinality of sequences slide)
    - Downcast the anyType element to Password (see downcasting weakly typed fields slide)
    - Set any/Password[0]/value to an XPath value used to obtain the password
    - Set any/Password[0]/Type to literal “wsse:PasswordText”



Continuing from the previous slide, the next step is to open the Add/Edit dialog and define a create action, selecting UsernameToken as the header element. Having done this, you are then presented with the Set The Values panel in which you do the remaining steps.

The UsernameToken XSD contains an element for the user name. You define how to set the Username/value field by adding an XPath to a location in the SMO containing the user name. The setting of the password is more difficult, in that the UsernameToken does not explicitly define a password field. To add the password, you must add an element to the sequence any as was previously described in the slide on cardinality. Then downcast the element in the sequence to type Password, as described in the slide on downcasting weakly typed fields. You can then set the field any/Password[0]/value by adding an XPath to a location in the SMO containing the password. Finally, to indicate the header contains a clear text password, set the field any/Password[0]/Type to the literal value wsse:PasswordText. The screen capture at the bottom of the slide shows the Set the Values panel once all this has been done.

Summary

- Examined the SOAP header setter mediation primitive

SOAP header setter

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

In summary, this presentation provided details regarding the SOAP 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 SOAP 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_SOAPHeaderSetterPrimitive.ppt

This module is also available in PDF format at:
../WBPMv62_SOAPHeaderSetterPrimitive.pdf

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

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

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.