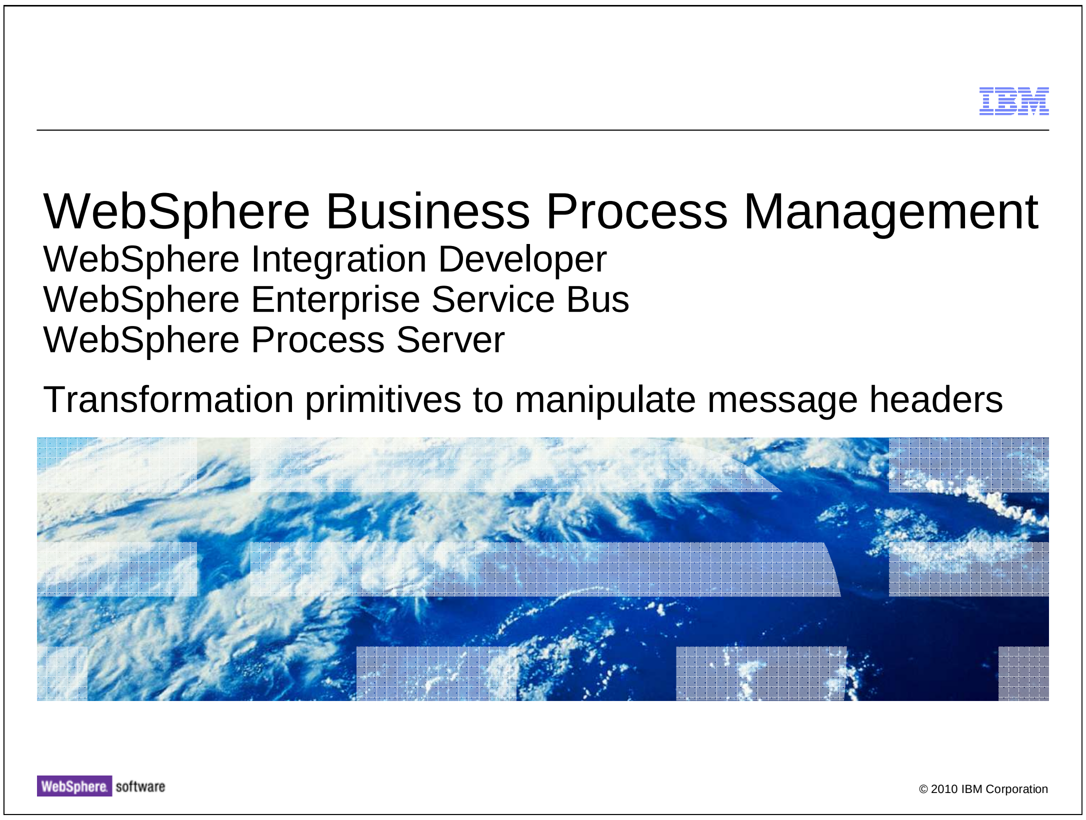


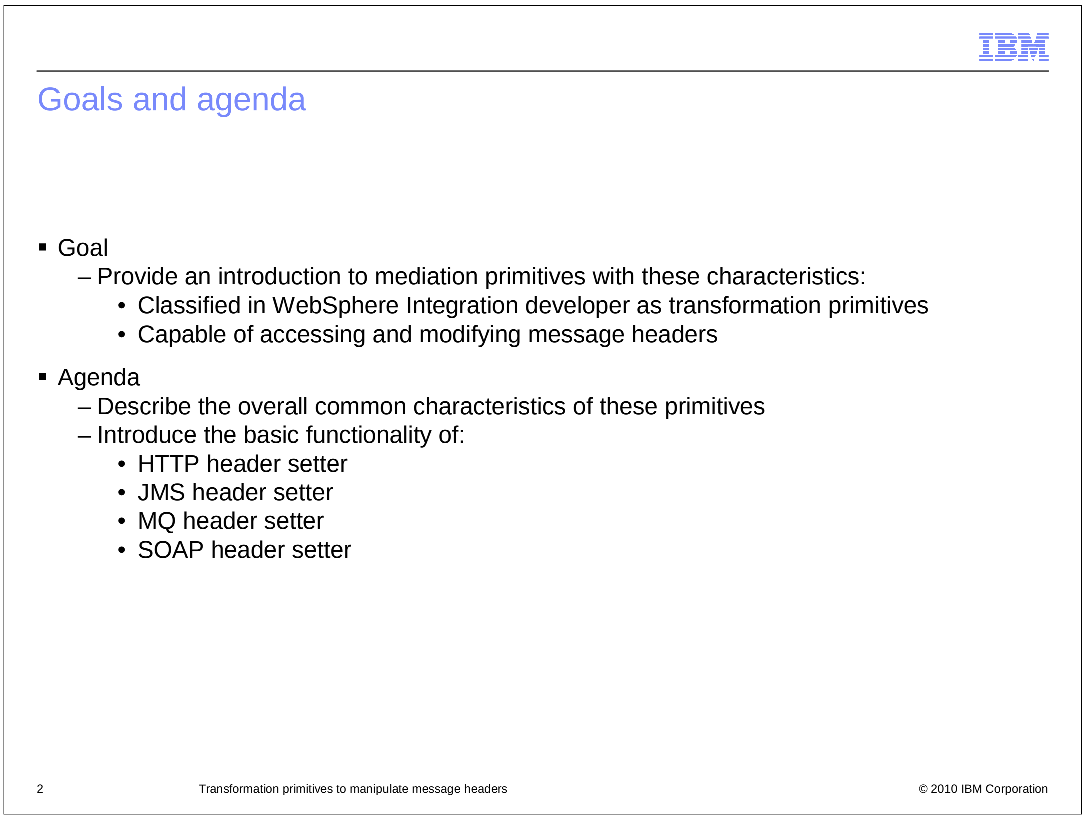
WebSphere Business Process Management

WebSphere Integration Developer
WebSphere Enterprise Service Bus
WebSphere Process Server

Transformation primitives to manipulate message headers

WebSphere. software

This presentation introduces the transformation primitives that are capable of accessing and manipulating the protocol specific headers in the message.

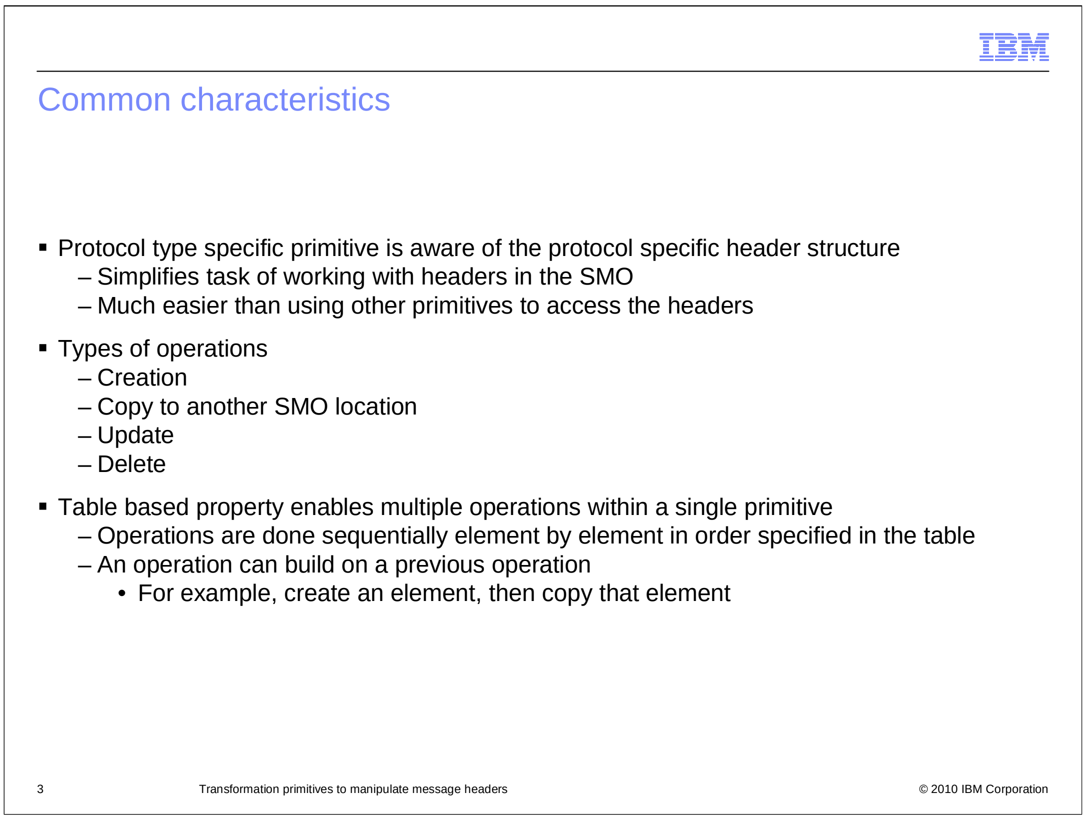

Goals and agenda

- Goal
  - Provide an introduction to mediation primitives with these characteristics:
    - Classified in WebSphere Integration developer as transformation primitives
    - Capable of accessing and modifying message headers
- Agenda
  - Describe the overall common characteristics of these primitives
  - Introduce the basic functionality of:
    - HTTP header setter
    - JMS header setter
    - MQ header setter
    - SOAP header setter

2 Transformation primitives to manipulate message headers © 2010 IBM Corporation

The goal of this presentation is to introduce a set of primitives that share some common characteristics. These primitives are classified within WebSphere Integration Developer as part of the transformation primitives group. Within that group, these are the primitives that have the capability to access and modify message headers in the service message object (SMO).

The presentation starts out by further explaining the common characteristics of these primitives, and then provides a description of the basic functionality of each. The primitives presented are the HTTP header setter, the JMS header setter, the MQ header setter and the SOAP header setter. Collectively, these are sometimes referred to as the header setter primitives.



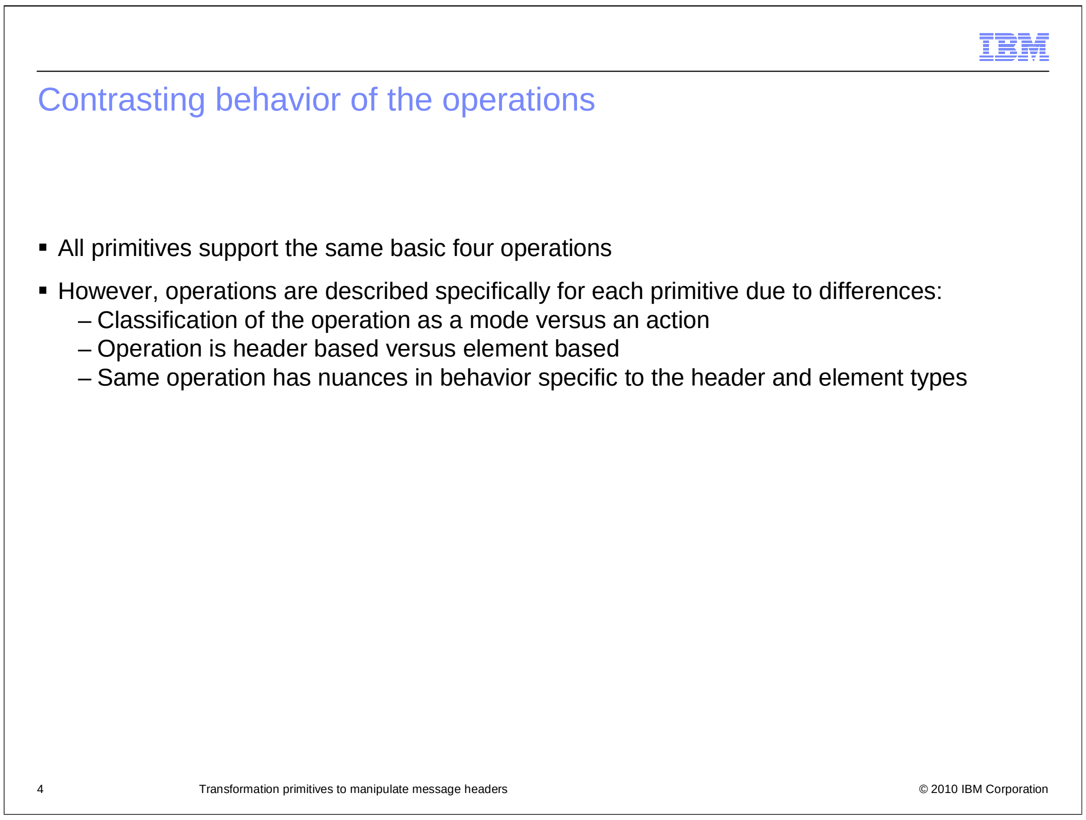
Common characteristics

- Protocol type specific primitive is aware of the protocol specific header structure
  - Simplifies task of working with headers in the SMO
  - Much easier than using other primitives to access the headers
- Types of operations
  - Creation
  - Copy to another SMO location
  - Update
  - Delete
- Table based property enables multiple operations within a single primitive
  - Operations are done sequentially element by element in order specified in the table
  - An operation can build on a previous operation
    - For example, create an element, then copy that element

The header setter primitives are each aware of the protocol specific structure of the headers, thus making it much easier to access and manipulate them. Although it can also be done, it is much more difficult to use other mediation primitive types to accomplish this.

Each of the primitives supports four basic operations, creation, coping to another SMO location, updating and deletion.

Specification of these operations is table based, thus enabling multiple operations to be specified with a single instance of the primitive. The operations are performed sequentially, row by row, in the order they are defined in the table. You can have one operation build on the result of a previous operation. For example, the operation create followed by the operation 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.



Contrasting behavior of the operations

- All primitives support the same basic four operations
- However, operations are described specifically for each primitive due to differences:
  - Classification of the operation as a mode versus an action
  - Operation is header based versus element based
  - Same operation has nuances in behavior specific to the header and element types

4 Transformation primitives to manipulate message headers © 2010 IBM Corporation

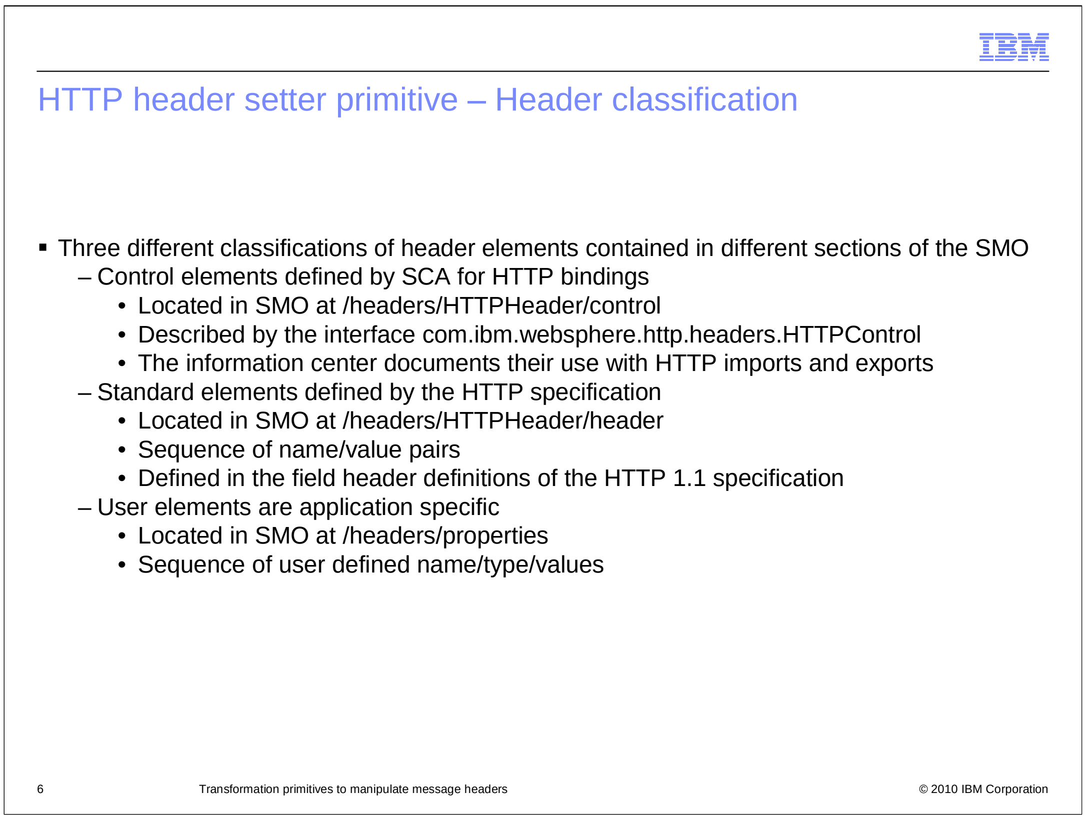
As was mentioned, all four primitives support the same four basic operations. However, the details of these are presented in the sections for the individual primitives because there are differences between them. Some of the primitives classify an operation as a mode, whereas the others classify it as an action. Also, some of the primitives apply an operation to an entire header, while the others apply an operation to an individual element. Finally, there are nuances in the behavior of an operation that is specific to the individual header type or element type being operated upon.

Section

HTTP header setter

5 Transformation primitives to manipulate message headers © 2010 IBM Corporation

The first primitive described is the HTTP header setter.



HTTP header setter primitive – Header classification

- Three different classifications of header elements contained in different sections of the SMO
  - 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
    - The information center documents their use with HTTP imports and exports
  - Standard elements defined by the HTTP specification
    - Located in SMO at /headers/HTTPHeader/header
    - Sequence of name/value pairs
    - Defined in the field header definitions of the HTTP 1.1 specification
  - User elements are application specific
    - Located in SMO at /headers/properties
    - Sequence of user defined name/type/values

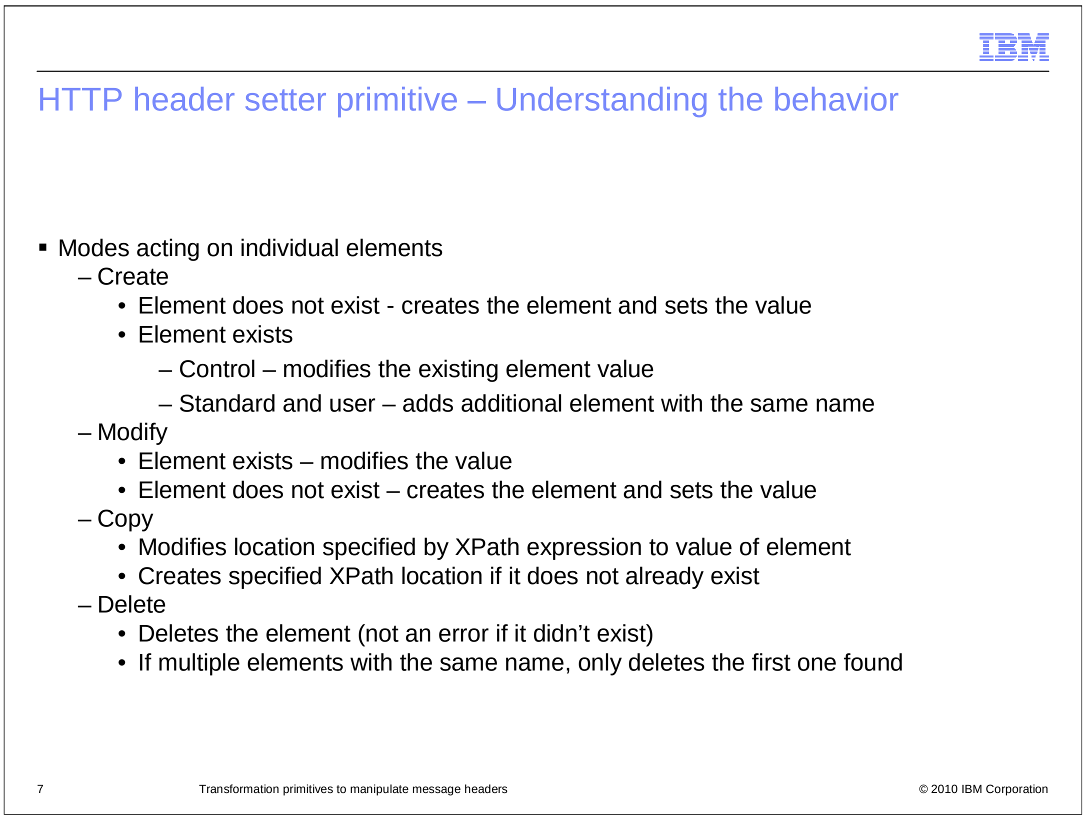
6 Transformation primitives to manipulate message headers © 2010 IBM Corporation

The HTTP headers are grouped into three different classifications of header elements, which are located at different places in the SMO.

The first grouping are the control elements, which are defined by the service component architecture (SCA) for use with HTTP bindings. They are located in the SMO at /headers/HTTPHeader/control. These are described by the interface com.ibm.websphere.http.headers.HTTPControl. The information center documents 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. These are located in the SMO at /headers/HTTPHeader/header and are represented as a sequence of name value pairs. The names of these elements along with their defined usage is documented by the World Wide Web Consortium (W3C) in the HTTP 1.1 specification.

The final grouping are user defined HTTP header elements, found in the SMO at /headers/properties. These are a sequence of name, type, and value triplets. The meaning of these are application specific.



HTTP header setter primitive – Understanding the behavior

- Modes acting on individual elements
  - Create
    - Element does not exist - creates the element and sets the value
    - Element exists
      - Control – modifies the existing element value
      - Standard and user – adds additional element with the same name
  - Modify
    - Element exists – modifies the value
    - Element does not exist – creates the element and sets the value
  - Copy
    - Modifies location specified by XPath expression to value of element
    - Creates specified XPath location if it does not already exist
  - Delete
    - Deletes the element (not an error if it didn't exist)
    - If multiple elements with the same name, only deletes the first one found

7 Transformation primitives to manipulate message headers © 2010 IBM Corporation

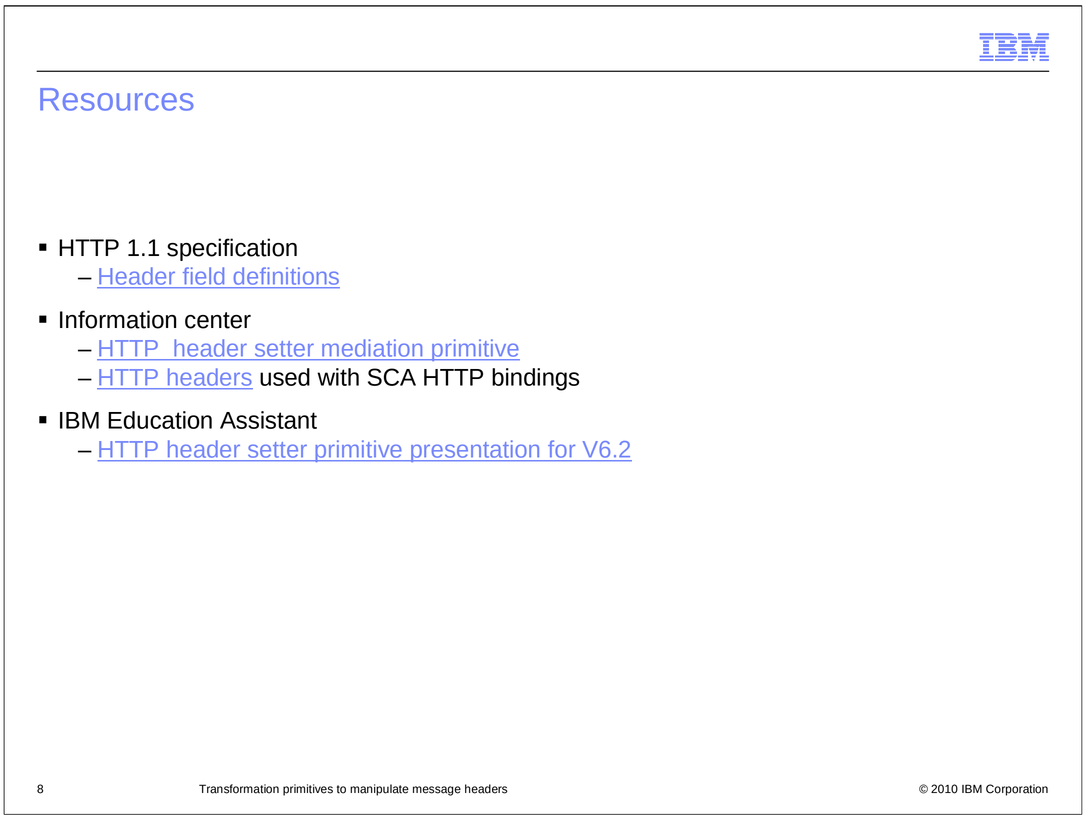
To properly use the HTTP header setter primitive, it is important to understand the specific behavior of the operations, which are referred to as modes. A mode acts on an individual header element.

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.



Resources

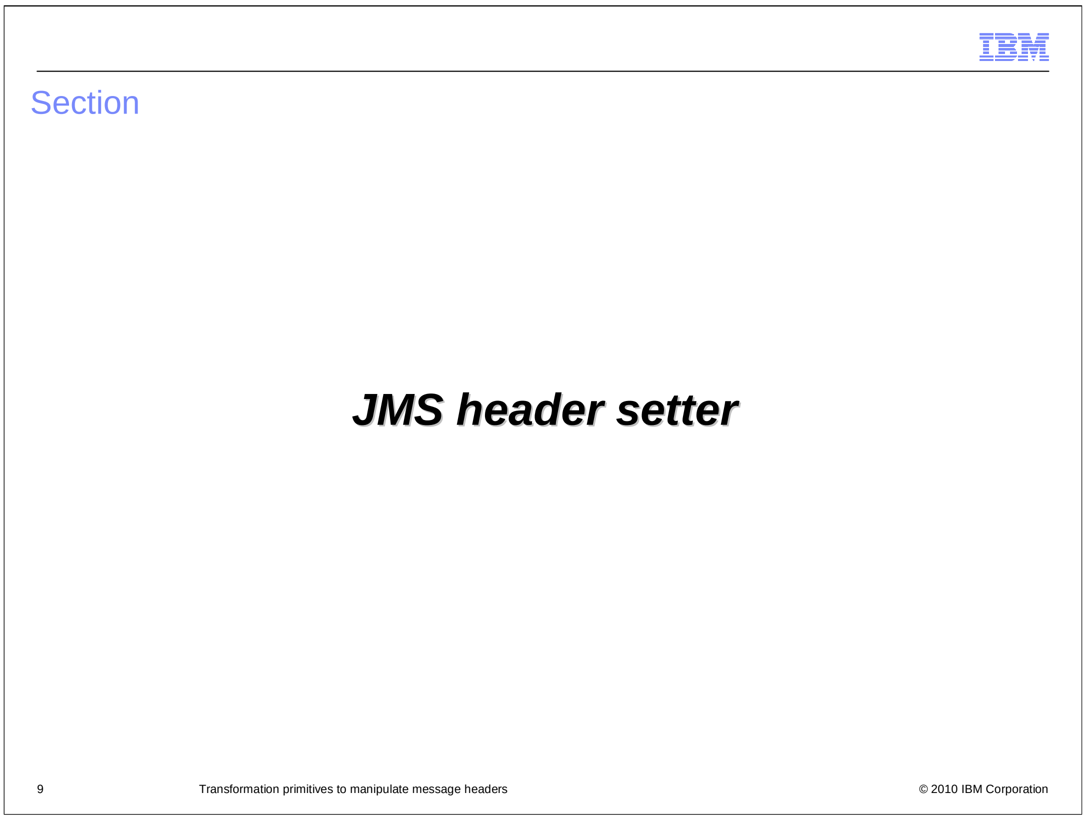
- HTTP 1.1 specification
  - Header field definitions
- Information center
  - HTTP header setter mediation primitive
  - HTTP headers used with SCA HTTP bindings
- IBM Education Assistant
  - HTTP header setter primitive presentation for V6.2

8 Transformation primitives to manipulate message headers © 2010 IBM Corporation

The first resource listed is to the HTTP 1.1 specification, specifically chapter 14 that provides the definition of the header fields that are part of the standard.

The information center, of course, has a section on the HTTP header setter primitive. This describes the usage of the primitive, defines the properties and provides some considerations to keep in mind when using it. The next link is to a section that defines the HTTP header properties that are used with SCA HTTP import and export bindings.

The IBM Education Assistant for V6.2 contains a more detailed presentation on the HTTP header setter primitive. Its content is also applicable to V7.

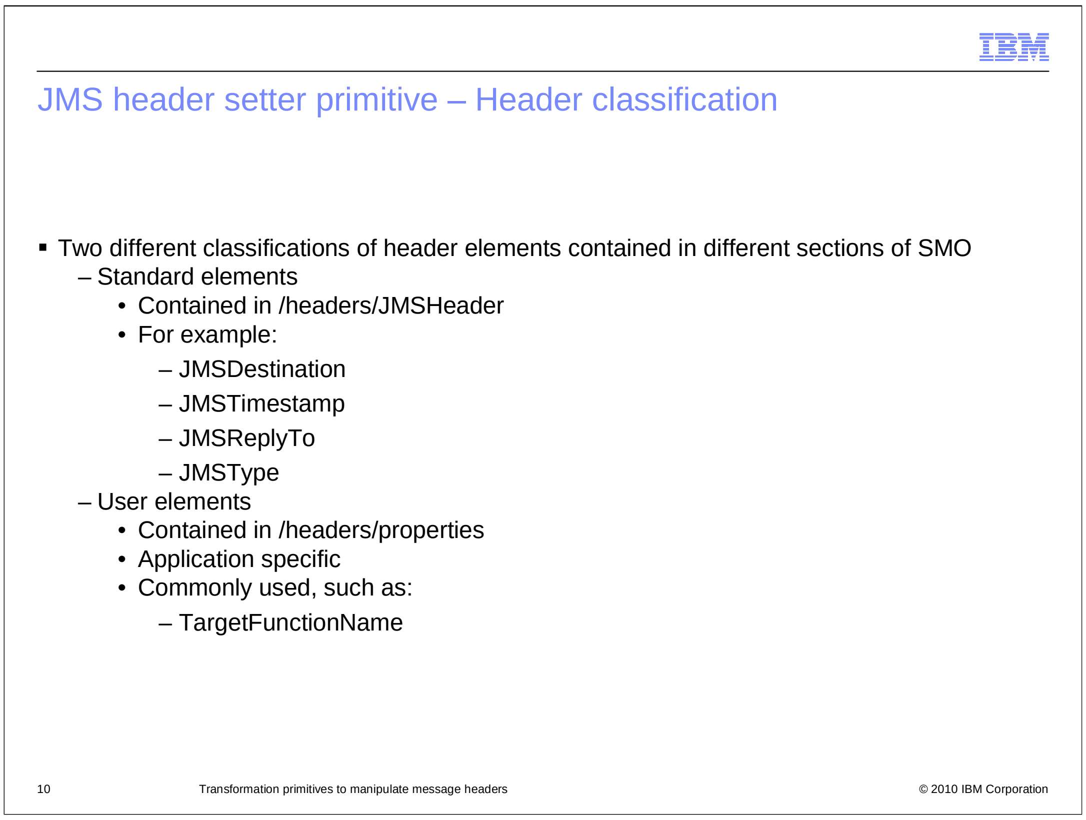


Section

JMS header setter

9 Transformation primitives to manipulate message headers © 2010 IBM Corporation

The JMS header setter primitive is described in this section.

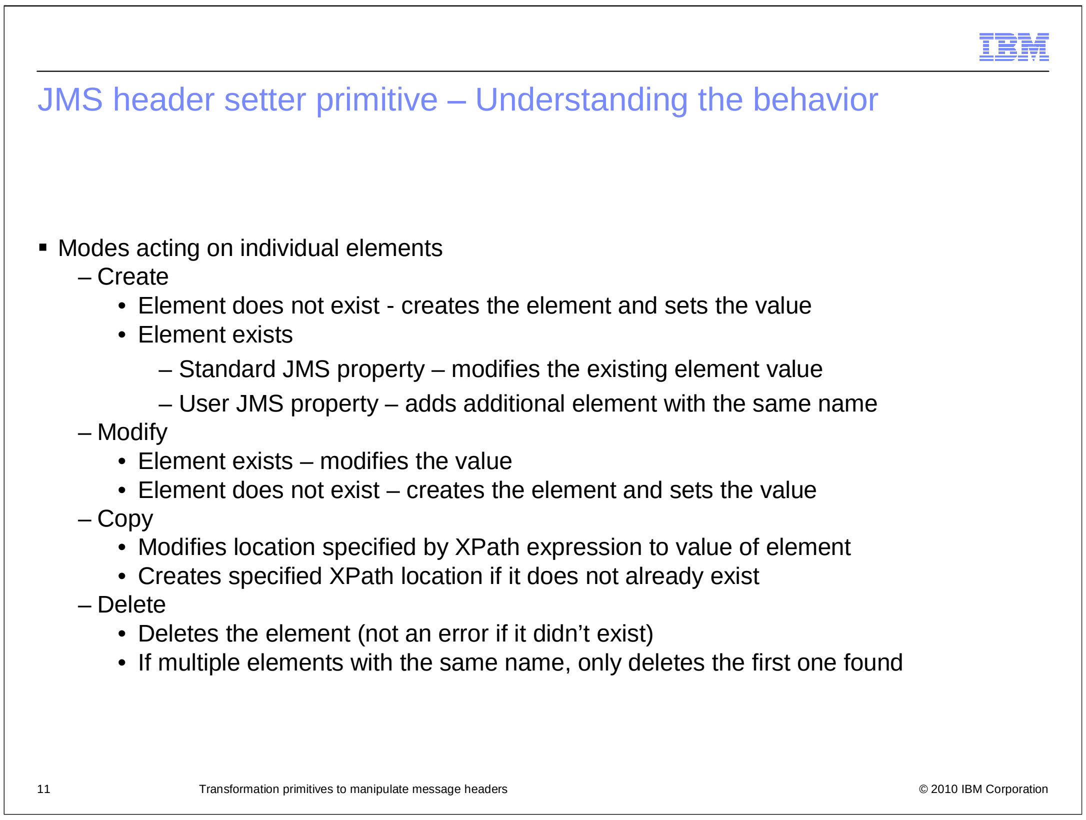


JMS header setter primitive – Header classification

- Two different classifications of header elements contained in different sections of SMO
  - Standard elements
    - Contained in /headers/JMSHeader
    - For example:
      - JMSDestination
      - JMSTimestamp
      - JMSReplyTo
      - JMSType
  - User elements
    - Contained in /headers/properties
    - Application specific
    - Commonly used, such as:
      - TargetFunctionName

10 Transformation primitives to manipulate message headers © 2010 IBM Corporation

The JMS header setter primitive is used to access and manipulate both standard and user JMS properties. These are represented differently within the SMO. The standard JMS properties are located in the SMO at /headers/JMSHeader. The standard properties are those defined within the JMS specification, for example JMSDestination, JMSTimestamp, JMSReplyTo and JMSType. The JMS user properties are located in the SMO at /headers/properties and are a sequence of name value pairs. They are generally properties that are specific to a particular application. They can also be commonly used properties, such as TargetFunctionName which is used with SCA applications.



JMS header setter primitive – Understanding the behavior

- Modes acting on individual elements
  - Create
    - Element does not exist - creates the element and sets the value
    - Element exists
      - Standard JMS property – modifies the existing element value
      - User JMS property – adds additional element with the same name
  - Modify
    - Element exists – modifies the value
    - Element does not exist – creates the element and sets the value
  - Copy
    - Modifies location specified by XPath expression to value of element
    - Creates specified XPath location if it does not already exist
  - Delete
    - Deletes the element (not an error if it didn't exist)
    - If multiple elements with the same name, only deletes the first one found

11 Transformation primitives to manipulate message headers © 2010 IBM Corporation

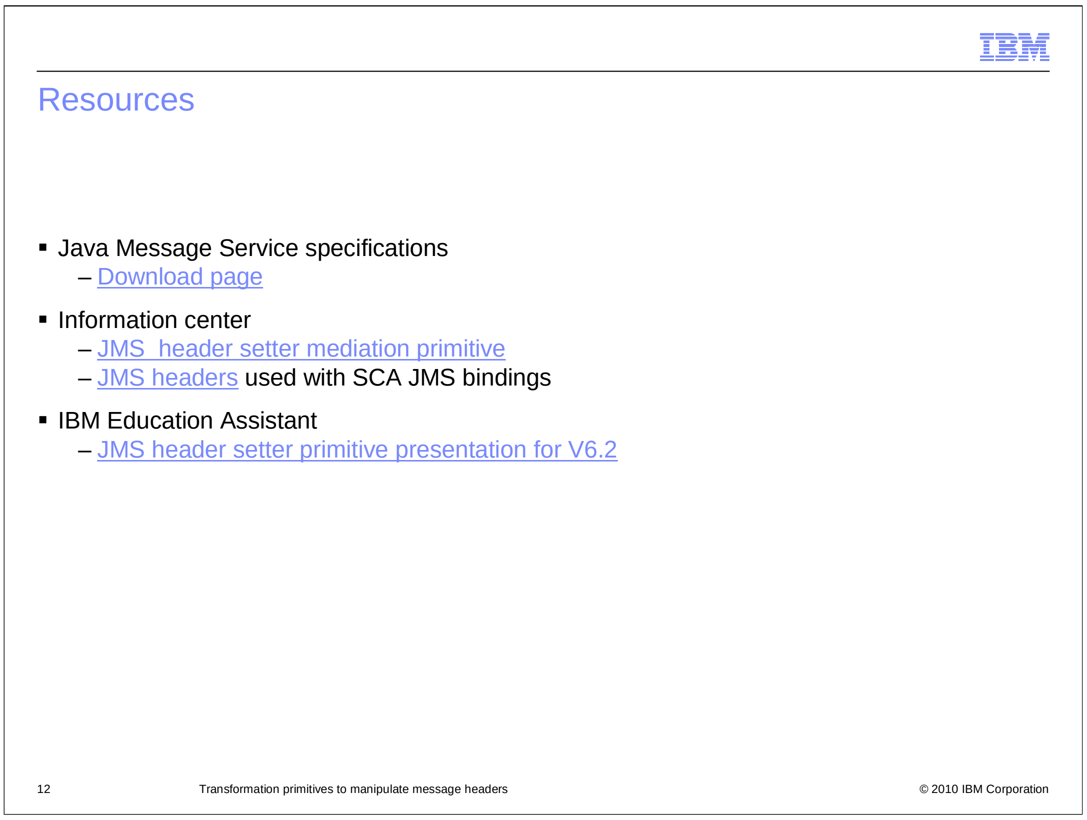
To properly use the JMS header setter primitive, it is important to understand the specific behavior of the operations, which are referred to as modes. A mode acts upon an individual header element.

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 type of JMS property it is. For a standard JMS property, the create mode will update the existing value to a new value. For a JMS user property, the create mode will add an additional element, resulting in multiple elements with the same name. The newly created one is placed at the end of the /headers/properties 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 user properties, 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 property 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 a JMS property. If an element for the property does not exist, it is not considered an error. If there are multiple elements for the same property, only the first one is deleted.



Resources

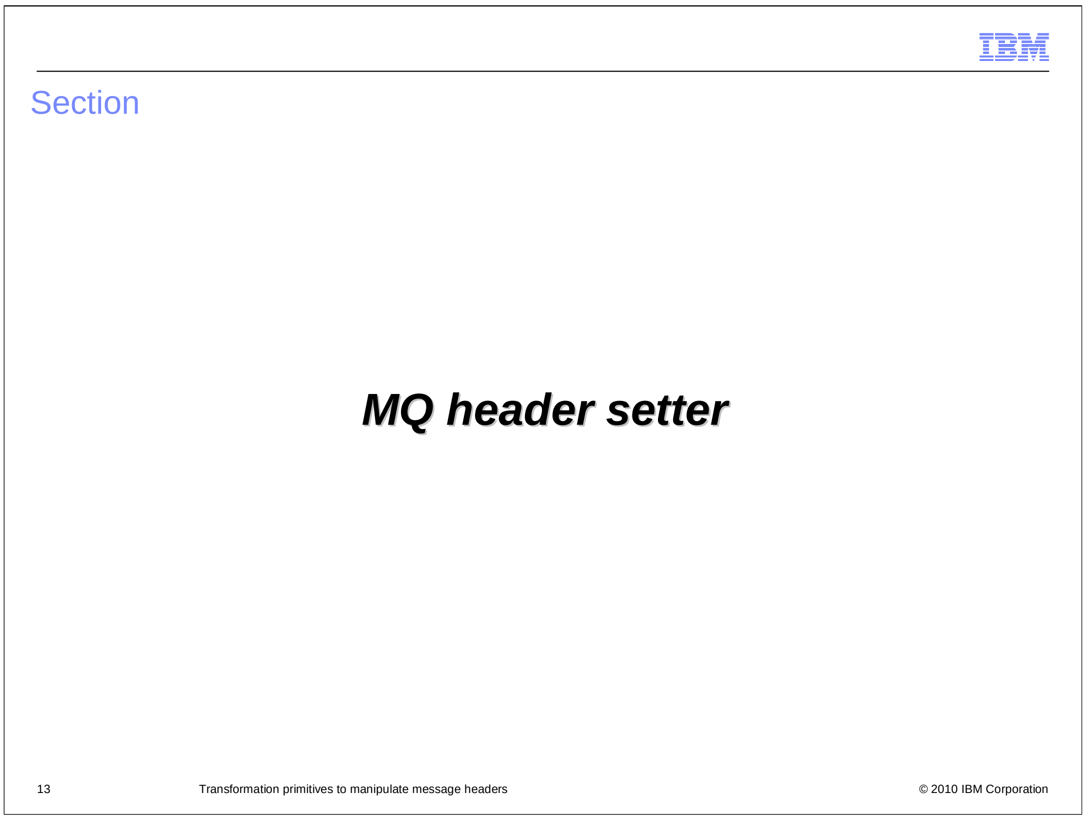
- Java Message Service specifications
  – Download page
- Information center
  – JMS header setter mediation primitive
  – JMS headers used with SCA JMS bindings
- IBM Education Assistant
  – JMS header setter primitive presentation for V6.2

12 Transformation primitives to manipulate message headers © 2010 IBM Corporation

The first resource listed is to the download page for the JMS specifications. You can download a copy of the specification which documents the header properties defined by the standard.

The information center provides a section on the JMS header setter primitive, describing its usage, definition of the properties and provides some considerations. The next link is to a section that defines the JMS header properties that are used with SCA JMS import and export bindings.

The IBM Education Assistant for V6.2 contains a more detailed presentation on the JMS header setter primitive. Its content is also applicable to V7.

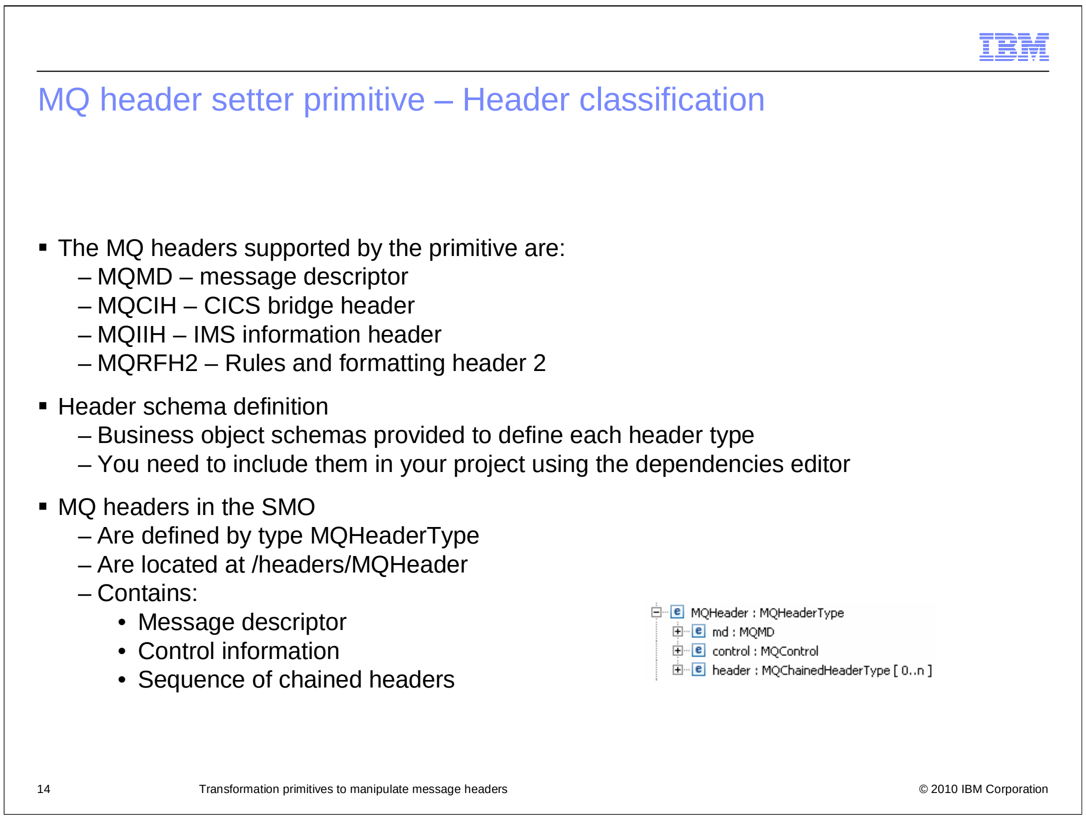


Section

MQ header setter

13 Transformation primitives to manipulate message headers © 2010 IBM Corporation

This section examines the MQ header setter primitive.



MQ header setter primitive – Header classification

- The MQ headers supported by the primitive are:
  - MQMD – message descriptor
  - MQCIH – CICS bridge header
  - MQIIH – IMS information header
  - MQRFH2 – Rules and formatting header 2
- Header schema definition
  - Business object schemas provided to define each header type
  - You need to include them in your project using the dependencies editor
- MQ headers in the SMO
  - Are defined by type MQHeaderType
  - Are located at /headers/MQHeader
  - Contains:
    - Message descriptor
    - Control information
    - Sequence of chained headers

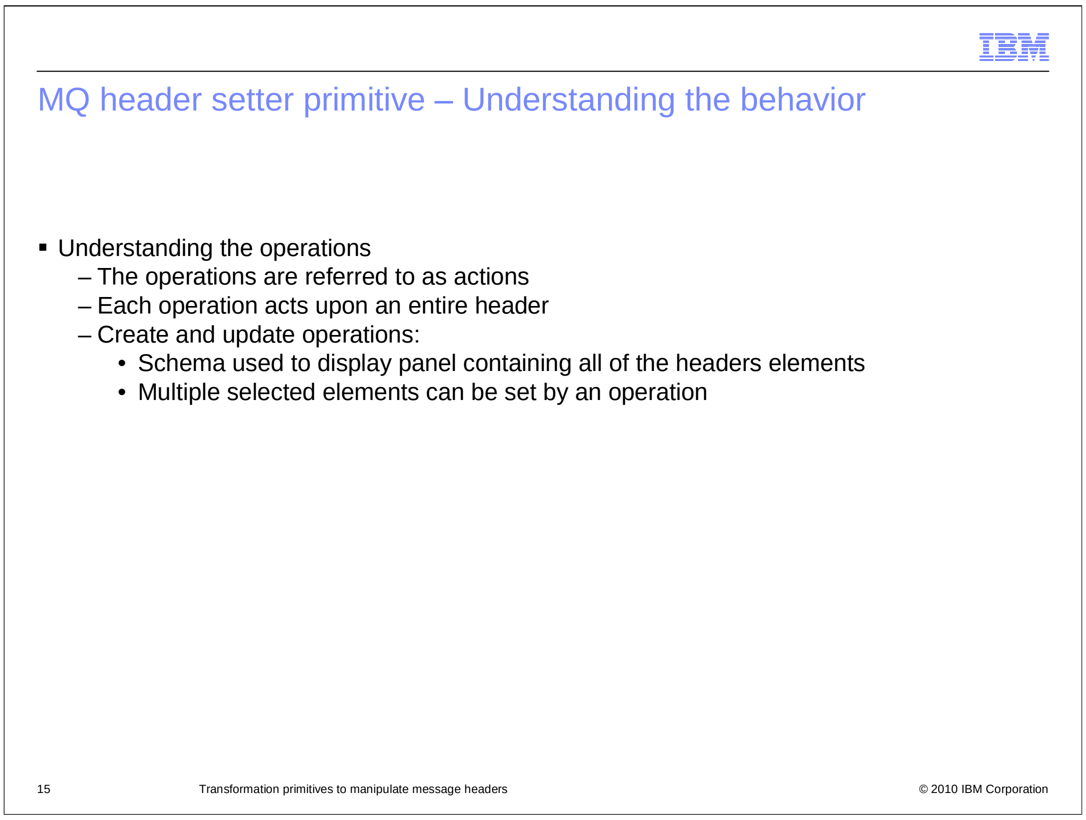
```
MQHeader : MQHeaderType
  md : MQMD
  control : MQControl
  header : MQChainedHeaderType [ 0..n ]
```

14 Transformation primitives to manipulate message headers © 2010 IBM Corporation

There are four MQ headers that are supported by the MQ header setter 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 MQ headers are contained in the SMO at /headers/MQHeader and are defined by MQHeaderType. It contains the MQMD which represents the contents of the WebSphere MQ message description, except for information determining the structure and encoding of the body. The MQControl contains information determining the structure and encoding of a message body. All the other headers are contained in a sequence of chained MQHeader objects.

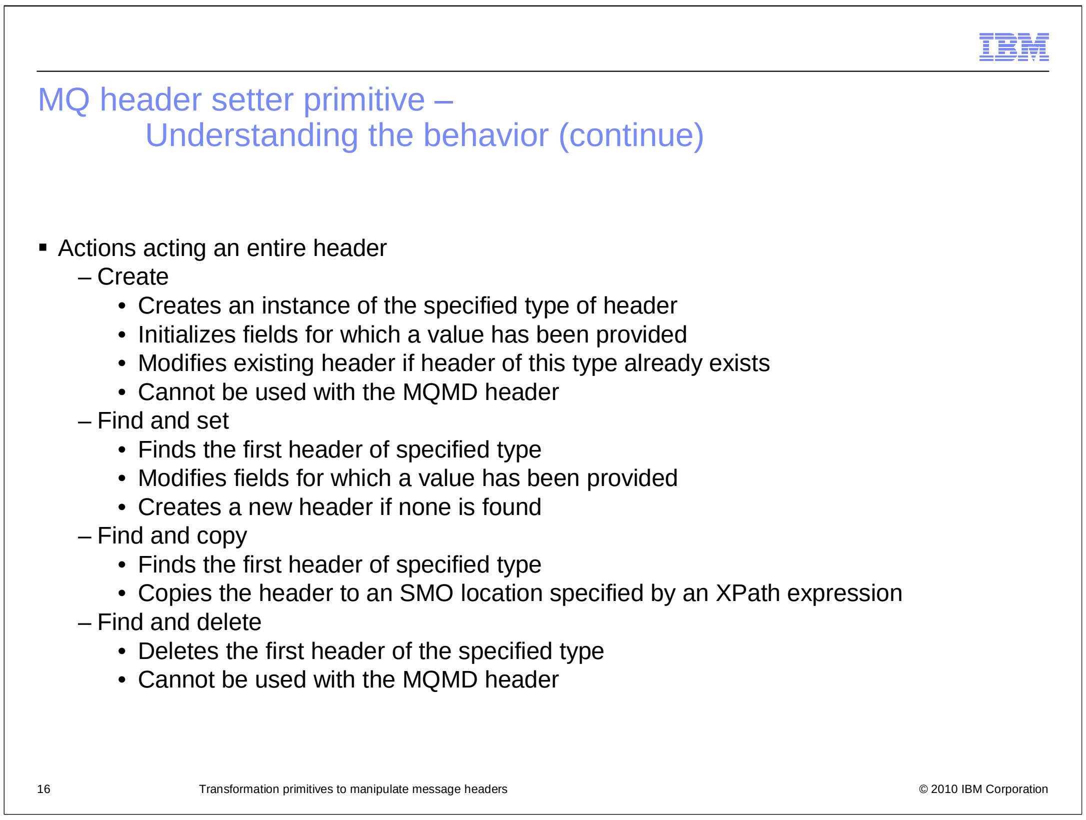


MQ header setter primitive – Understanding the behavior

- Understanding the operations
  - The operations are referred to as actions
  - Each operation acts upon an entire header
  - Create and update operations:
    - Schema used to display panel containing all of the headers elements
    - Multiple selected elements can be set by an operation

15 Transformation primitives to manipulate message headers © 2010 IBM Corporation

The operations are called actions. Each operation is associated with an entire header. For the operations that create and update a header, the schema for the header is used to display a customized panel showing all the elements of the header. This allows you to specify values for multiple elements within the same operation.



MQ header setter primitive – Understanding the behavior (continue)

- Actions acting an entire header
  - Create
    - Creates an instance of the specified type of header
    - Initializes fields for which a value has been provided
    - Modifies existing header if header of this type already exists
    - Cannot be used with the MQMD header
  - Find and set
    - Finds the first header of specified type
    - Modifies fields 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
    - Copies the header to an SMO location specified by an XPath expression
  - Find and delete
    - Deletes the first header of the specified type
    - Cannot be used with the MQMD header

16 Transformation primitives to manipulate message headers © 2010 IBM Corporation

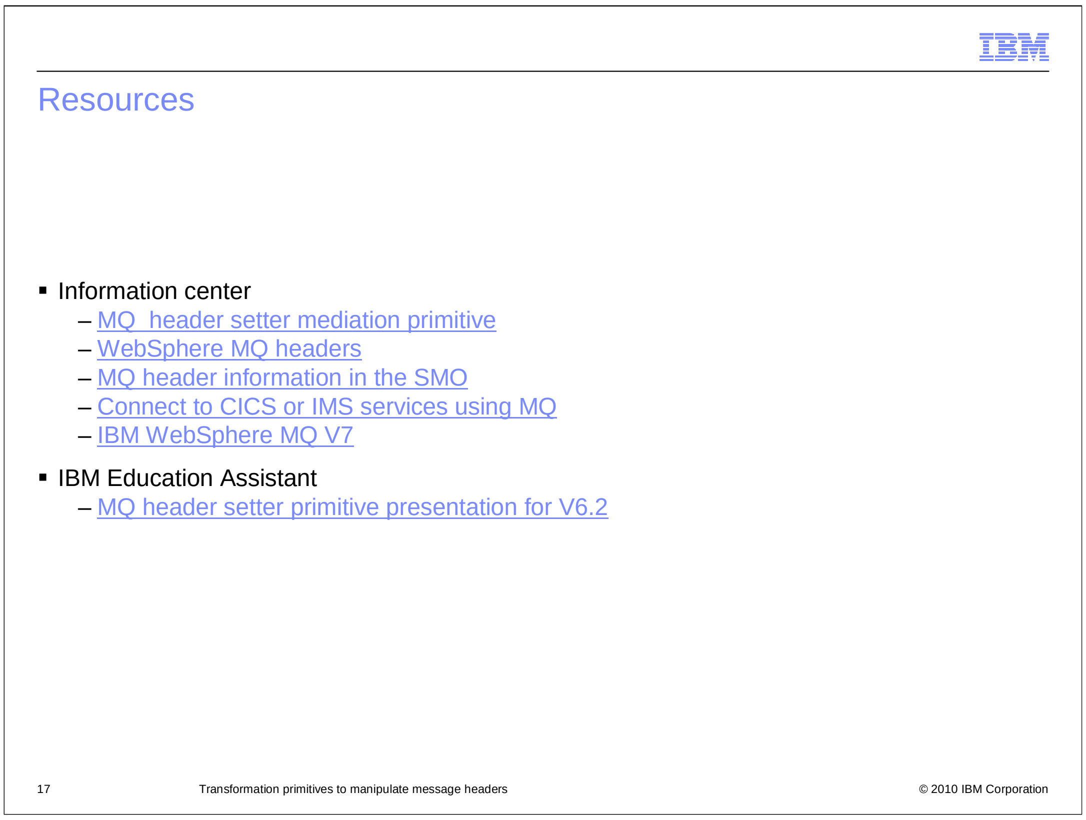
To properly use the MQ header setter primitive, it is important to understand the specific behavior of the actions, which act upon an entire header.

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 values to be set 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. 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.


Resources

- Information center
  - MQ header setter mediation primitive
  - WebSphere MQ headers
  - MQ header information in the SMO
  - Connect to CICS or IMS services using MQ
  - IBM WebSphere MQ V7
- IBM Education Assistant
  - MQ header setter primitive presentation for V6.2

17 Transformation primitives to manipulate message headers © 2010 IBM Corporation

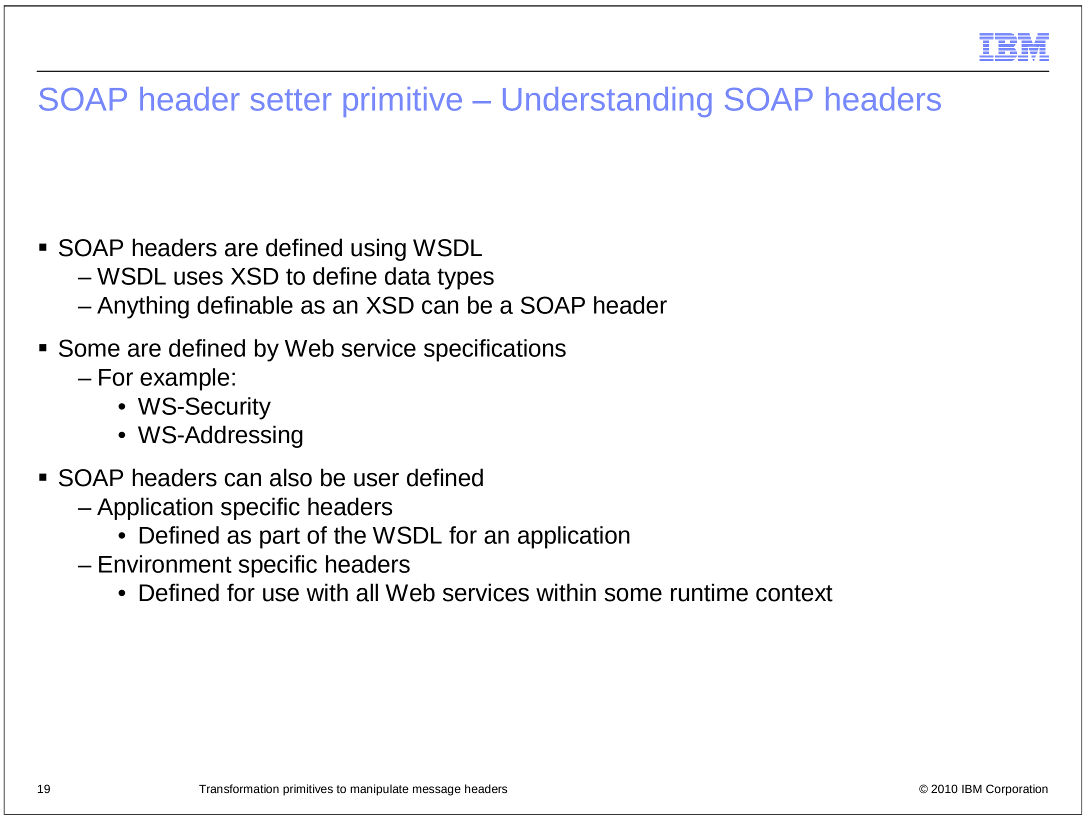
Some of the useful information center links are provided on this slide. The first is to the section for the MQ header setter primitive, describing the usage of the primitive, definition of the properties and some additional considerations. The next link, WebSphere MQ headers, provides information about how MQ headers are handled, such as how the server parses the headers. The link MQ header information in the SMO describes how the MQ headers are represented in the SMO. The next link describes using MQ to connect to CICS and IMS. There you will find links describing how to set up the CICS bridge header and the IMS information header. The final information center link is for the IBM WebSphere MQ V7 information center, where you can find much more detailed information on the use of headers in MQ.

The IBM Education Assistant for V6.2 contains a more detailed presentation on the MQ header setter primitive. Its content is also applicable to V7.

Section

Soap header setter

The SOAP header setter primitive is described in this section.



SOAP header setter primitive – Understanding SOAP headers

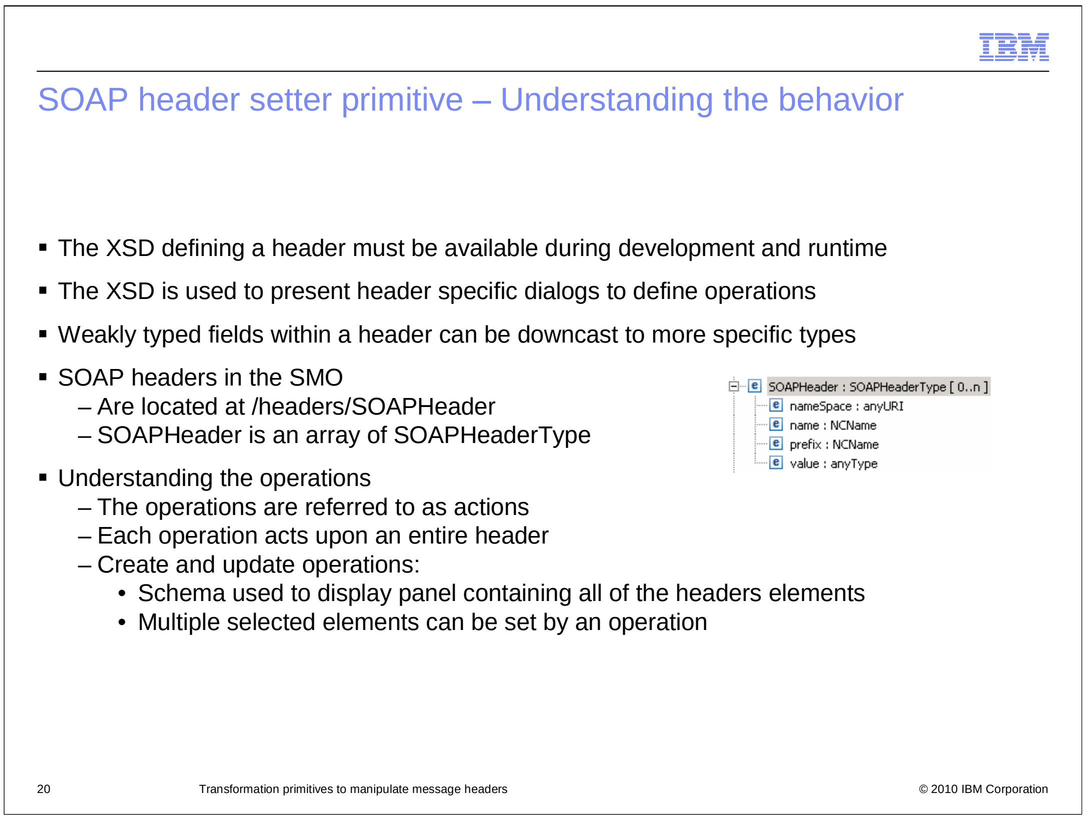
- SOAP headers are defined using WSDL
  - WSDL uses XSD to define data types
  - Anything definable as an XSD can be a SOAP header
- Some are defined by Web service specifications
  - For example:
    - WS-Security
    - WS-Addressing
- SOAP headers can also be user defined
  - Application specific headers
    - Defined as part of the WSDL for an application
  - Environment specific headers
    - Defined for use with all Web services within some runtime context

19 Transformation primitives to manipulate message headers © 2010 IBM Corporation

Provided here is some background on the nature of SOAP headers, enabling you to better understand the description of the SOAP header setter primitive. 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.



SOAP header setter primitive – Understanding the behavior

- The XSD defining a header must be available during development and runtime
- The XSD is used to present header specific dialogs to define operations
- Weakly typed fields within a header can be downcast to more specific types
- SOAP headers in the SMO
  – Are located at /headers/SOAPHeader
  – SOAPHeader is an array of SOAPHeaderType
- Understanding the operations
  – The operations are referred to as actions
  – Each operation acts upon an entire header
  – Create and update operations:
    • Schema used to display panel containing all of the headers elements
    • Multiple selected elements can be set by an operation

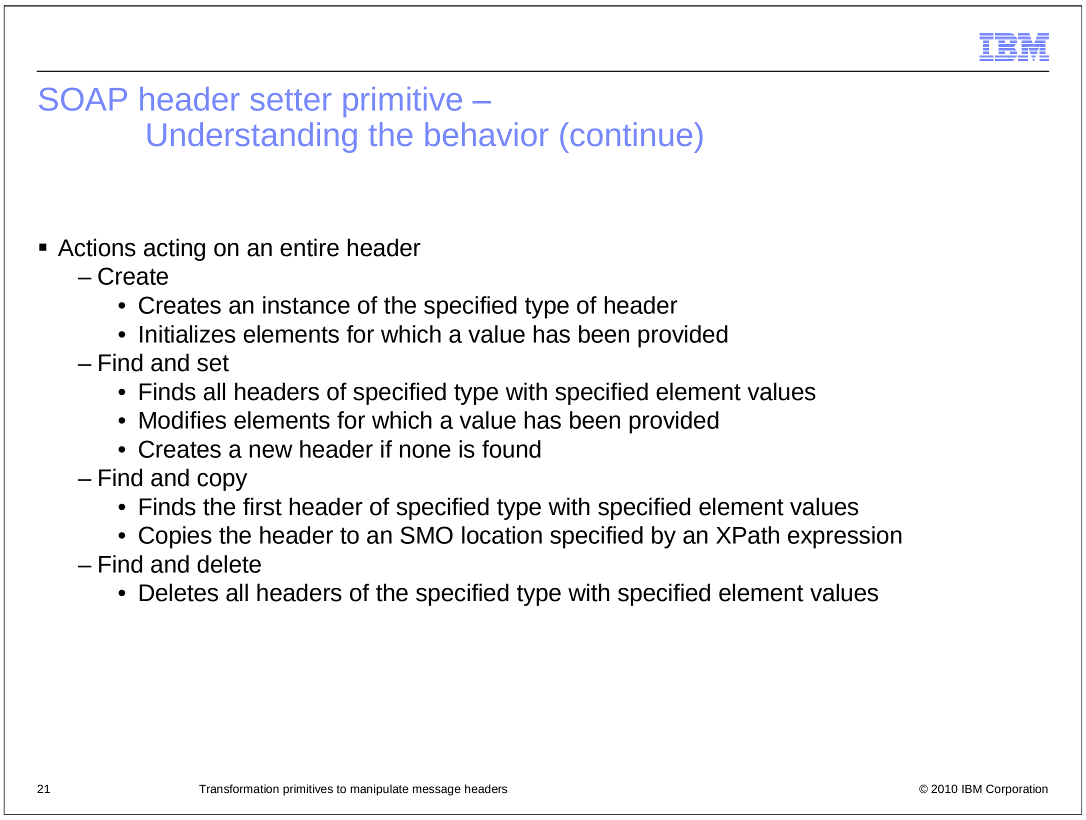
```
SOAPHeader : SOAPHeaderType [ 0..n ]
    nameSpace : anyURI
    name : NCName
    prefix : NCName
    value : anyType
```

20 Transformation primitives to manipulate message headers © 2010 IBM Corporation

The SOAP header setter primitive makes use of an XSD which 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 customized 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 SOAP headers are contained in the SMO at /headers/SOAPHeader. They are a sequence of SOAPHeaderType,

The operations are called actions. Each action is associated with an entire header. For the operations that create and update a header, the schema for the header is used to display a customized panel showing all the elements of the header. This allows you to specify values for multiple elements within the same operation.

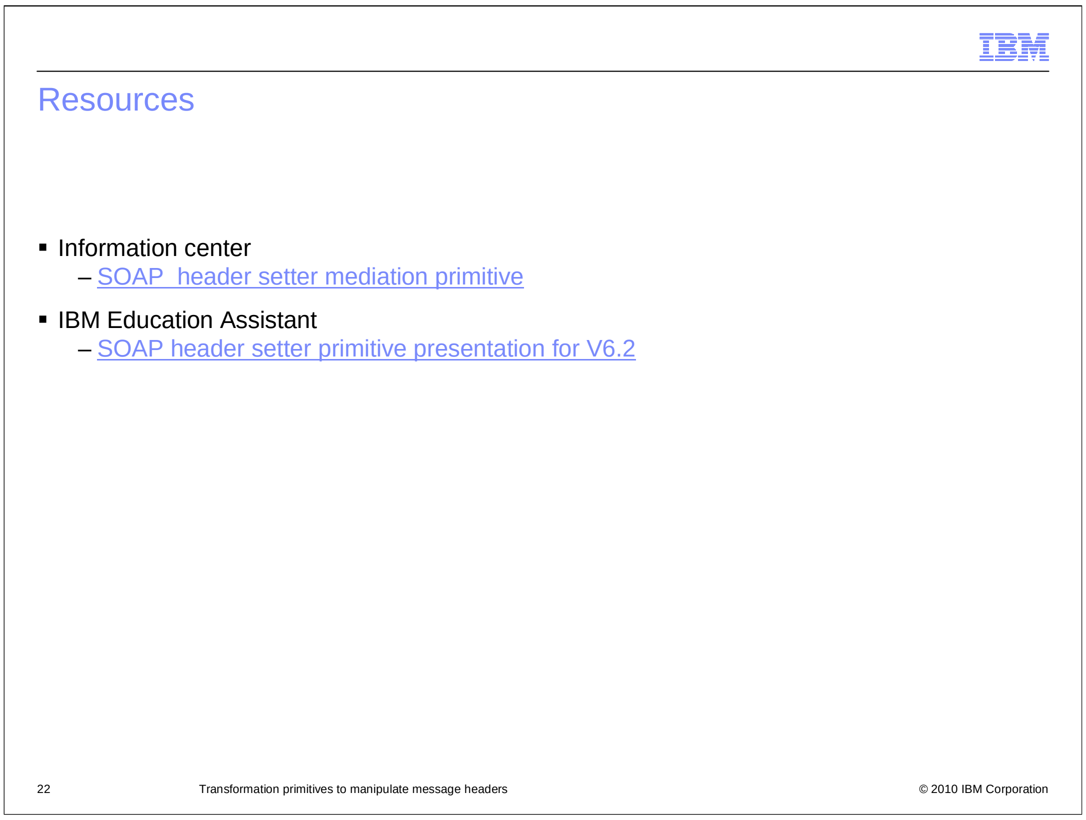


SOAP header setter primitive – Understanding the behavior (continue)

- Actions acting on an entire header
  - 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

21 Transformation primitives to manipulate message headers © 2010 IBM Corporation

To properly use the SOAP header setter primitive, it is important to understand the specific behavior of the actions which act upon an entire header. 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.



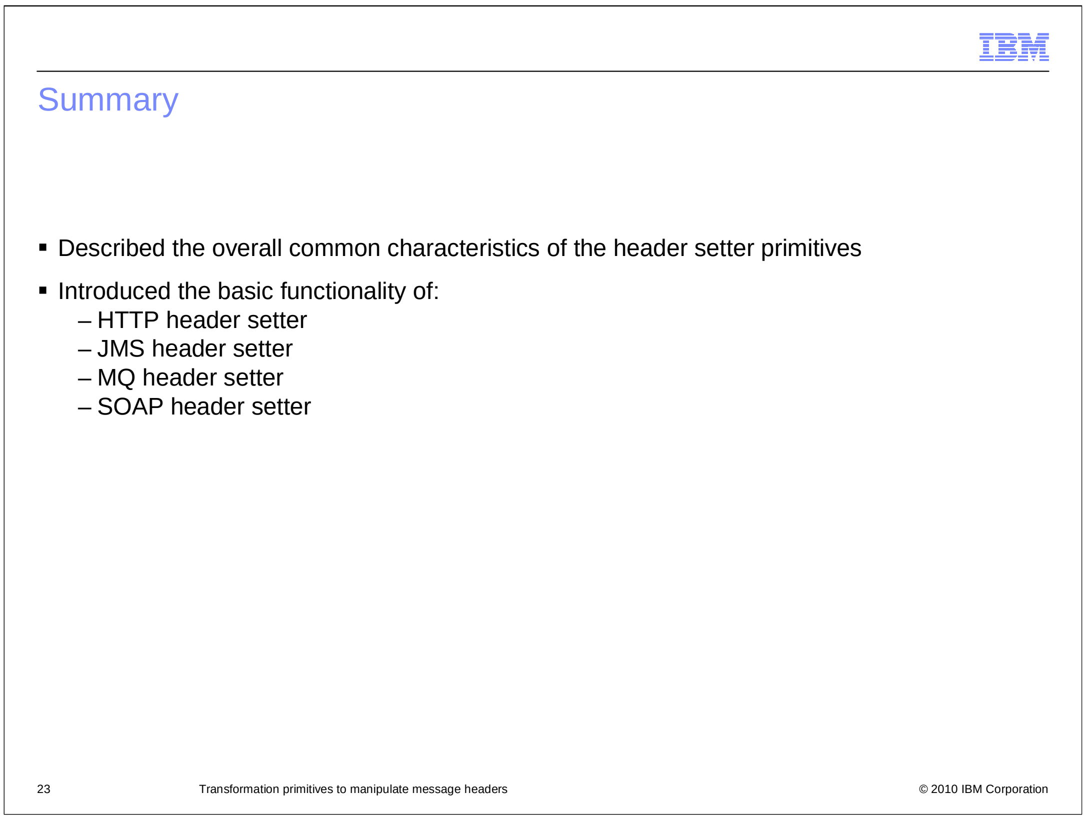
Resources

- Information center
  - SOAP header setter mediation primitive
- IBM Education Assistant
  - SOAP header setter primitive presentation for V6.2

22 Transformation primitives to manipulate message headers © 2010 IBM Corporation

The information center provides a section on the SOAP header setter primitive, describing the usage of the primitive, definition of the properties and provides some considerations for using it effectively.

The IBM Education Assistant for V6.2 contains a more detailed presentation on the SOAP header setter primitive. Its content is also applicable to V7.

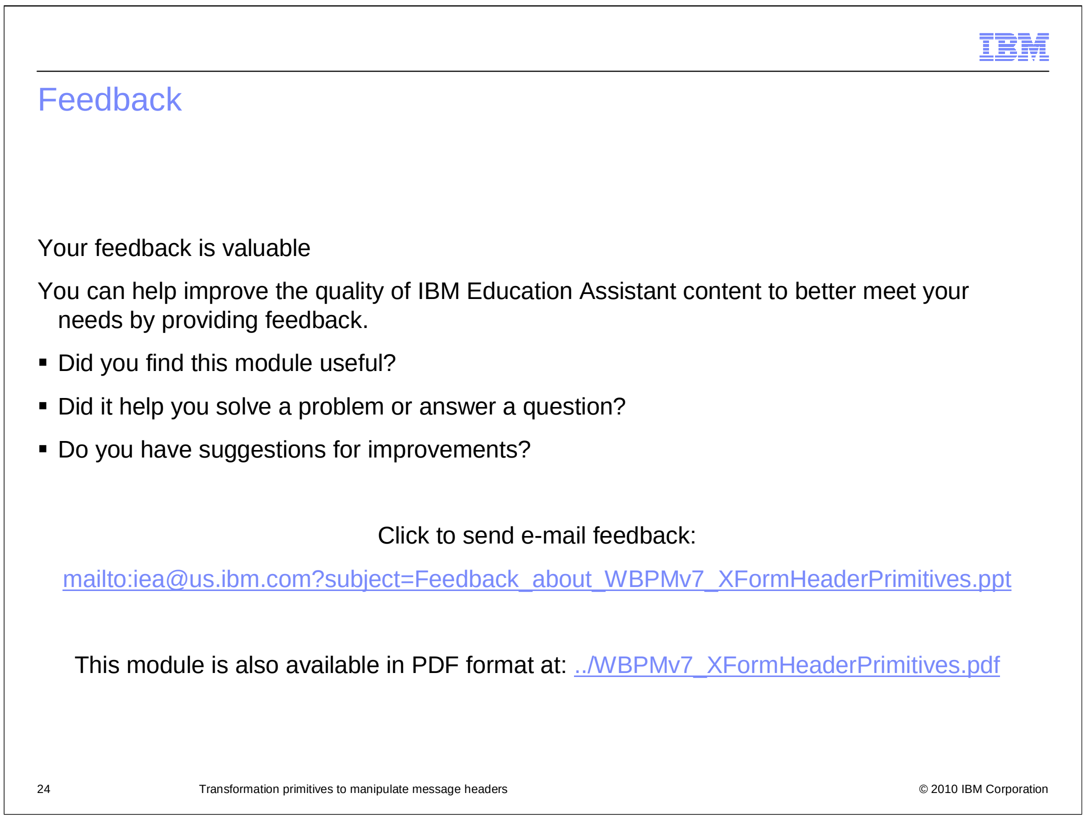


Summary

- Described the overall common characteristics of the header setter primitives
- Introduced the basic functionality of:
  - HTTP header setter
  - JMS header setter
  - MQ header setter
  - SOAP header setter

23 Transformation primitives to manipulate message headers © 2010 IBM Corporation

In summary, the presentation started out by explaining the common characteristics of the header setter primitives, and then provided a description of the basic functionality of each. The primitives presented were the HTTP header setter, the JMS header setter, the MQ header setter and the SOAP header setter.



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?

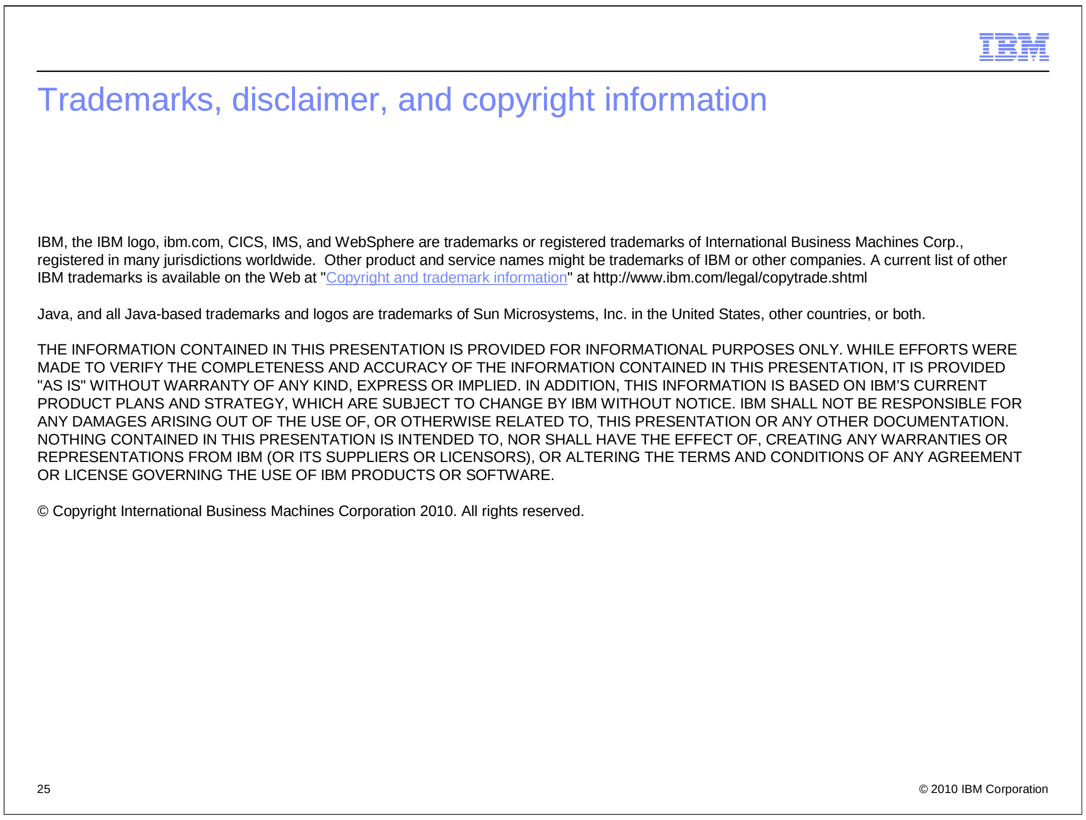
Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WBPMv7_XFormHeaderPrimitives.ppt

This module is also available in PDF format at: ../WBPMv7_XFormHeaderPrimitives.pdf

24 Transformation primitives to manipulate message headers © 2010 IBM Corporation

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



Trademarks, disclaimer, and copyright information