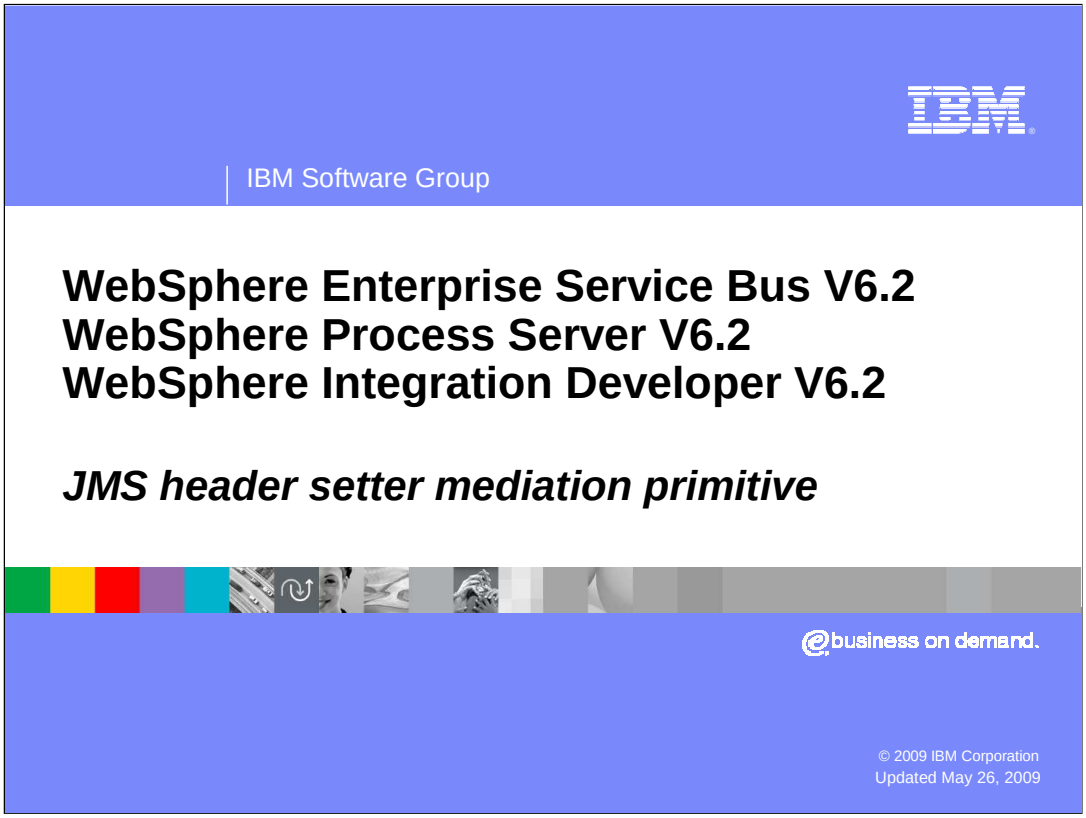


This presentation provides a detailed look at the JMS header setter mediation primitive, which is a new primitive introduced in version 6.2.

Goals

- Understand the JMS header setter mediation primitive

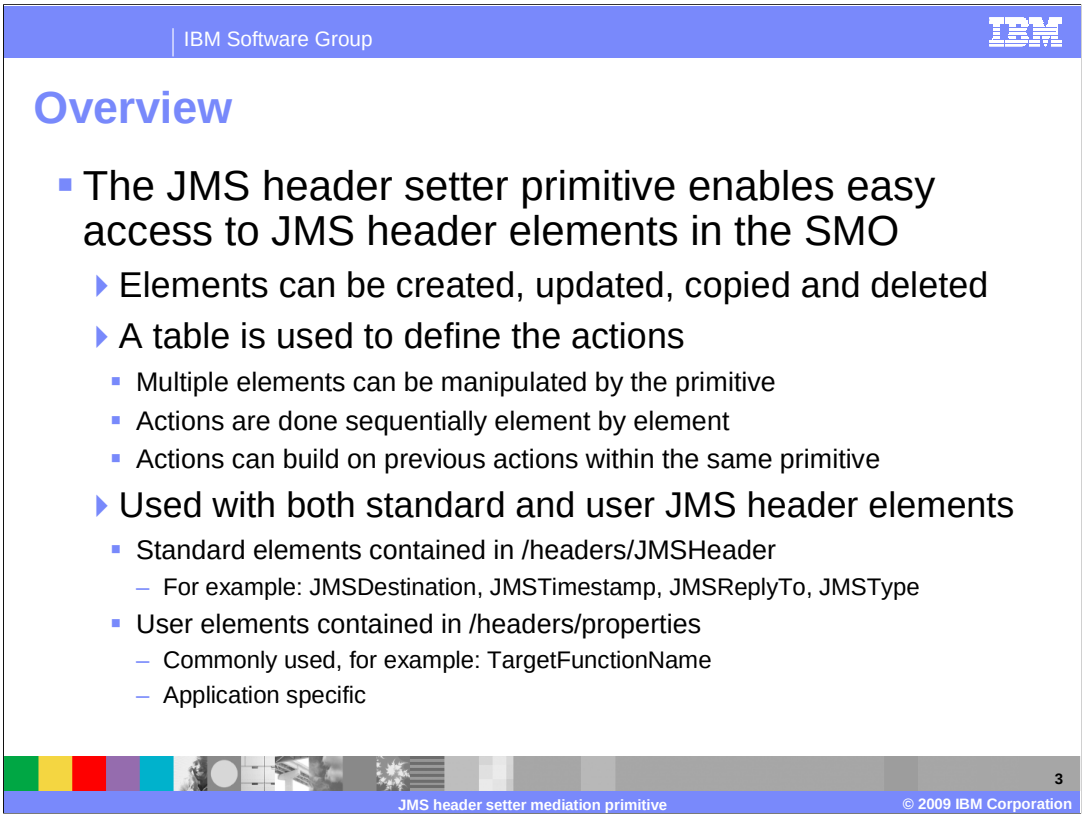
JMS header setter

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

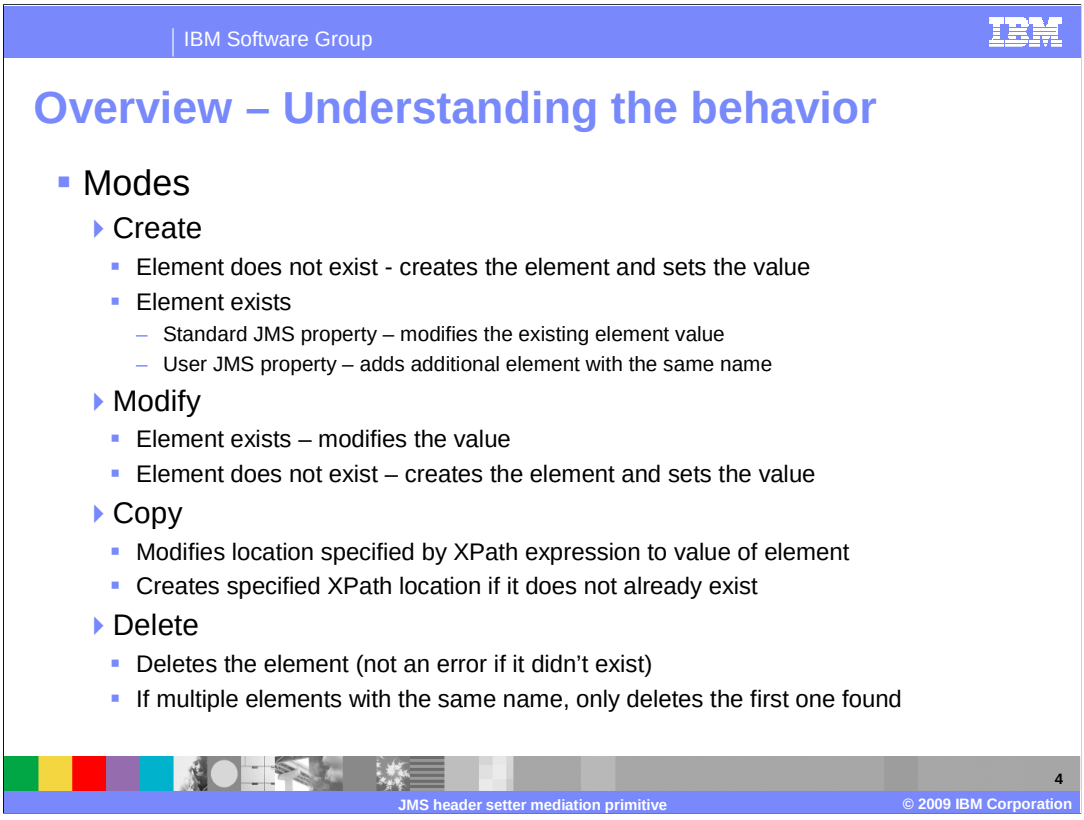
The presentation contains an overview of the function provided by the JMS header setter primitive, along with information about the primitive's use of terminals and its properties. The error handling characteristics are then covered and finally an example usage of a JMS header setter primitive is provided.



The purpose of the JMS header setter primitive is to enable access to the JMS 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. This is due to the optional presence of these elements and the difference in structure for standard JMS properties and JMS user properties. Using the JMS 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 JMS header elements. The primitive is configured with a table that defines the sequence of actions, Each row of the table references a single JMS 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.

The primitive is used to 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. These can be commonly used properties, such as TargetFunctionName which is used with SCA applications. They can also be properties specific to an individual application.



To properly use the JMS 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 type of JMS property it is. For a standard JMS property, the create mode will update the existing value to the 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.

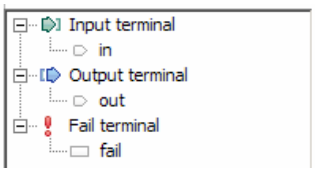
For the modify mode, if the element exists it is updated to the new value. If the element does not exist, it is created and the value set. When dealing with user properties, the use of modify might be preferred over create in that there is no possibility of creating duplicate entries for the same property.

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.

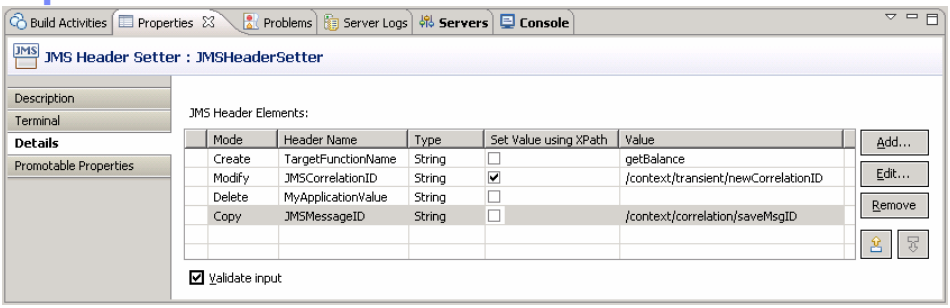
Terminals

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



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

Properties

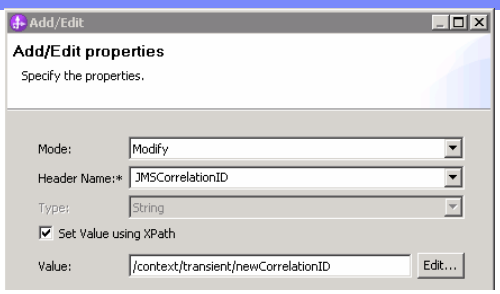


- JMS Header Elements table
  - List of the defined actions and settings
  - Each row is for an individual property
  - Add/Edit properties dialog used to set individual rows
  - Values can be static or obtained from the SMO using XPath
- Validate input
  - Validates if incoming message is of the expected 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 JMS header setter. There is a table called JMS 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 JMS 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 standard JMS 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)

This slide describes the Add/Edit properties dialog used to define rows of the JMS header elements table shown on the previous slide.

The mode field is a drop down box allowing you to set what action is to be performed. The choices are Create, Modify, Copy and Delete.

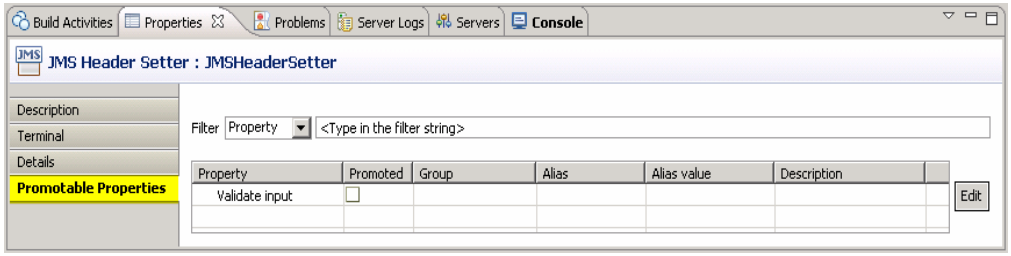
The Header Name field is where you specify the name of the JMS property to be acted upon. This is a combination of a drop down box and text entry field. If the property is a standard JMS header property, it can be selected from the drop down. However, if you are specifying a user property, it needs to be typed directly into the field.

The Type field is only needed when the property being set is a JMS 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
  - JMS Header Elements

This slide shows the Promotable Properties panel. None of the columns in the JMS Header Elements table are promotable. However, 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

- 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 JMS 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 the validation processing.

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

Example usage

- Example using the dynamic service gateway pattern
  - Backend banking system
    - SCA based back end system using JMS bindings
    - Input messages are JMS text message containing XML
    - Requires TargetFunctionName be set in JMS user properties
  - Input to gateway
    - JMS text message with banking transaction information
    - Messages have varying formats based on source of message
    - All messages have transaction type encoded in the message
  - Function of gateway
    - Receive input JMS text message
    - Custom mediation
      - Converts input message to XML
      - Sets target function name in transient context
    - JMS header setter
      - Uses modify action to add/update the TargetFunctionName user property
    - Forwards JMS text message containing XML to backend system

This slide introduces an example usage of a JMS header setter primitive in the context of a flow that implements a dynamic service gateway pattern. In this pattern, both the export and import support the service gateway interface. For this example, both the export and import are configured with JMS bindings. The example is explained on this slide and screen captures illustrating the JMS header setter primitive usage are shown on the next slide.

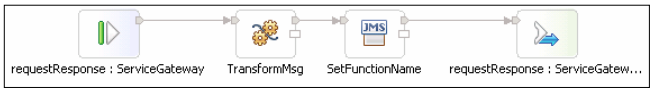
The backend system is an SCA based implementation of banking transactions which accepts JMS text messages containing serialized XML as input. The TargetFunctionName JMS user property is used to indicate which operation should be invoked.

The dynamic gateway implementation receives messages from different sources which are JMS text messages for banking transactions that have varying formats. Within each of the formats is an indication of the type of transaction the message represents. The function of the gateway is to receive a message and use a custom mediation to examine the content. The custom mediation converts it to serialized XML, determines the type of transaction and sets the appropriate target function name into a field in the transient context. The flow then uses a JMS header setter to add or update the TargetFunctionName user property. The message is then forwarded to the backend system.

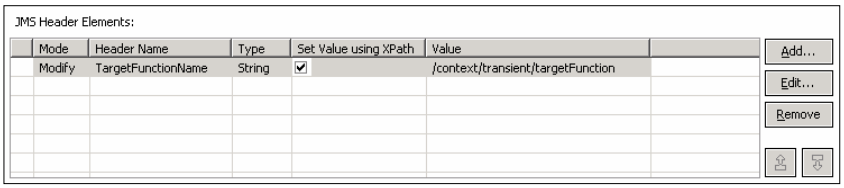
Example usage (continued)

Assembly diagram

Mediation request flow



JMS header setter properties



This slide contains screen captures illustrating the example described on the previous slide. The top portion shows the assembly diagram containing the export, mediation flow component and import. The center of the slide shows the mediation flow. The primitive called SetFunctionName is the JMS header setter. The configuration of the primitive is shown in the bottom screen capture. It uses a modify rather than a create to set the TargetFunctionName. This prevents a duplicate being created for this property in the case where it already existed on input to the gateway. It also has an XPath value identifying the source location in the transient context that contains the value to be set into the TargetFunctionName.

Summary

- Examined the JMS header setter mediation primitive

JMS header setter

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

In summary, this presentation provided details regarding the JMS 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 JMS 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_JMSHeaderSetterPrimitive.ppt

This module is also available in PDF format at: ../WBPMv62_JMSHeaderSetterPrimitive.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.