This session discusses the enhancements made to the Message Mapping editor and Mapping node in WebSphere Message Broker V6.

Agenda

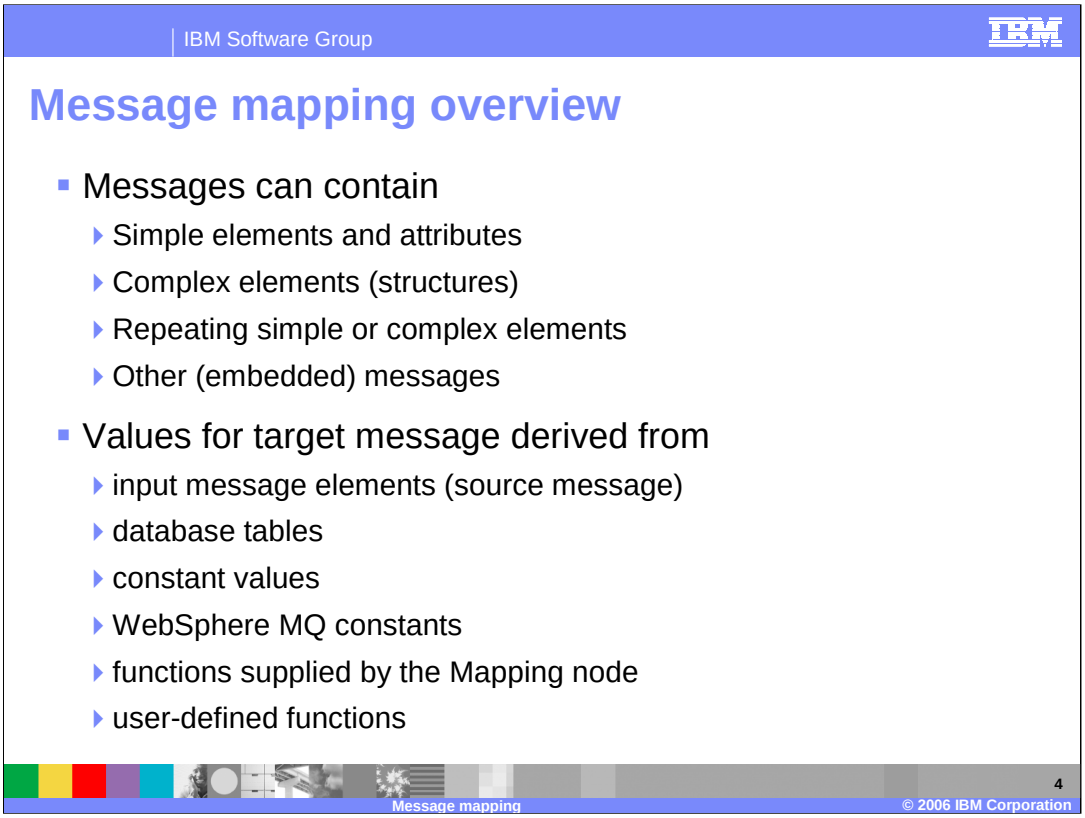
- Overview
- Message mapping
- Message Mapping editor
- Complex mapping
- Summary and references

The agenda for this presentation is to cover Message mapping and the mapping editor, and complex mapping.

Section

Overview

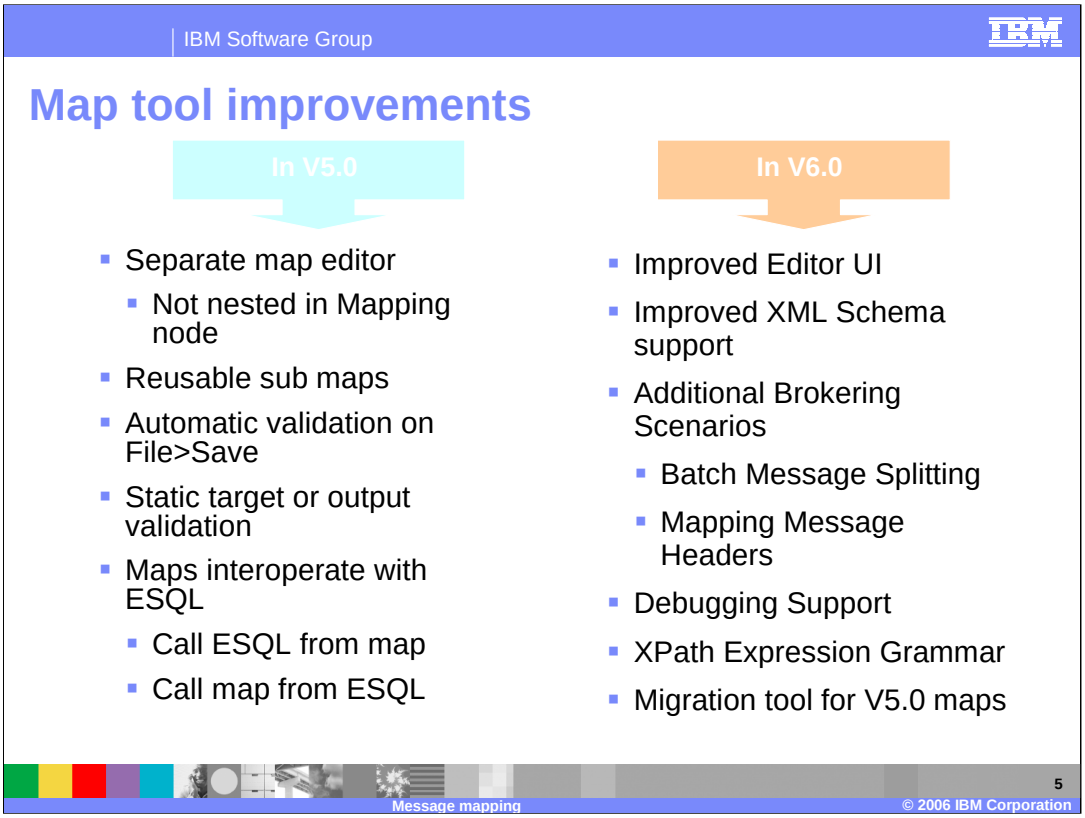
This section will provide an overview.



Message mappings define the blueprint for creating a message. Messages can contain simple and complex elements, repeating elements, and embedded messages.

Messages can contain protocol-specific headers, which might need to be manipulated by WebSphere Message Broker. Dynamic setting of a message destination (routing) within the WebSphere Message Broker might also be required. Values for target message elements can be derived from input message elements (the input message is also known as the source message), database tables, constant values, WebSphere MQ constants, functions supplied by the Mapping node or user-defined functions.

The logic to derive values can be simple or complex; conditional statements might be needed, as might loops, summations and other functions. All of the above mappings can be achieved using a Mapping node.



This slide shows the improvements in map tool in Broker versions 5 and 6.

Section

Message mapping

This section covers message mapping.

Mapping node

- Mapping node must contain:
  - Zero or one source (input) message
  - Zero or more source databases
  - One or more target (output) messages
- Source and target messages to be mapped must be defined in message definition files in a message set.

The Mapping node has one or more mappings, which are stored in message map files, which have the extension ".msgmap". These files are configured using the Message Mapping editor.

A Mapping node must contain the following inputs and outputs:

Zero or one source messages.

Zero or more source databases.

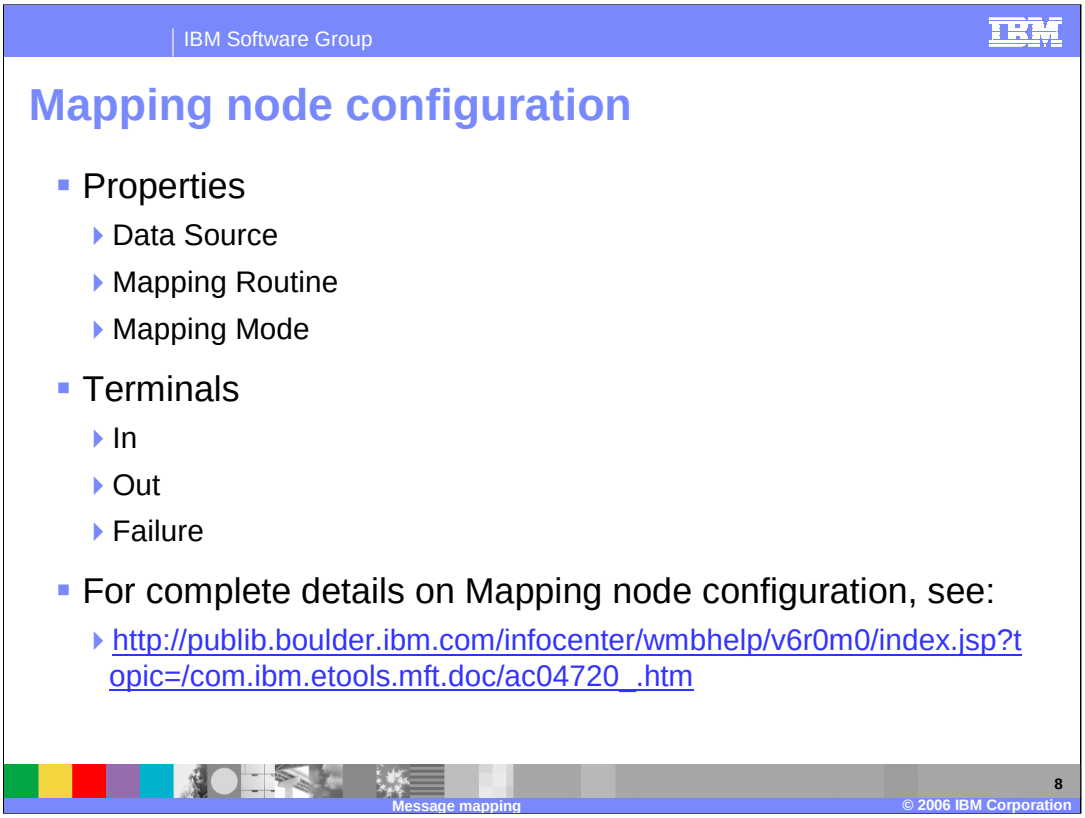
One or more target messages.

The source and target messages to be mapped must be defined in message definition files in a message set. The parser of the source message can be specified at run time (for example, in an MQRFH2 header), but the target message is built using the runtime parser that is specified in the message set.

If a message mapping is between elements of different types, you might need to include casts in your mapping definitions, depending on which runtime parser is specified in your message set.

The Mapping node uses a language to manipulate messages that are based on XPath.

To develop message mappings for a Mapping node, use the Message Mapping editor, which provides separate panes for working with sources, targets and expressions.



Some of the properties of the Mapping node are the *Data Source:* the name by which the appropriate database is known on the system on which this message flow is to run, and *Mapping Routine*, which contains the statements to run against the database or the message tree. The routine is unique to this type of node. A mapping routine is specific to the type of node with which it is associated; you cannot use a mapping routine that you have developed for a Mapping node with any other node that uses mappings (for example, a DataInsert node).

Another property is the *Mapping Mode*, which is the mode that you want to use to process information being passed through the Mapping node. You can choose any combination of Message (the message generated or passed through by the Mapping node as modified within the node), LocalEnvironment, and Exception components to be generated and modified by the Mapping node.

The Mapping node has In, Out and Failure terminals.

See the Information Center for complete details on Mapping node configuration.

Creating a message map file

- To create a message map (.msgmap) file:
  - From File>New>Message Map

  or
  - Open Map from Mapping node (or other node that supports mapping)
- In the New Message Map wizard specify:
  - Whether mapping headers
  - Whether map is a submap (called from another map)
  - Source message or database
  - Target message

To create a message map file:

From the Broker Application Development perspective, click File > New > Message Map. Alternatively, to create a message map from a message flow, open the message flow, right-click a node that supports mapping (such as a Mapping node or DataDelete node), and click Open Map.

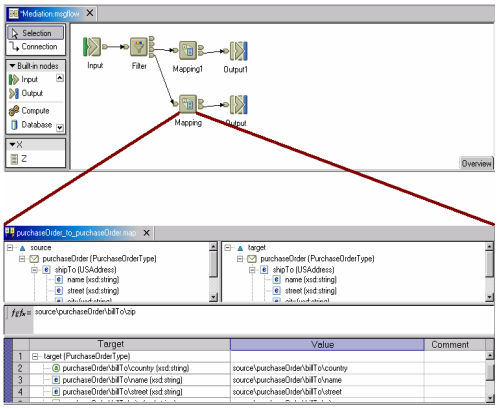
In the New Message Map wizard, specify the following four criteria:

1) Whether you are going to map headers (after you have specified this, you cannot change it)
2) Whether the map is intended to be called from another map, which is known as a submap
3) The source message or database
4) The target message

Mapping capabilities are provided in Mapping, DataInsert, DataUpdate, DataDelete, Extract, and Warehouse nodes.

Role of mapping in brokers

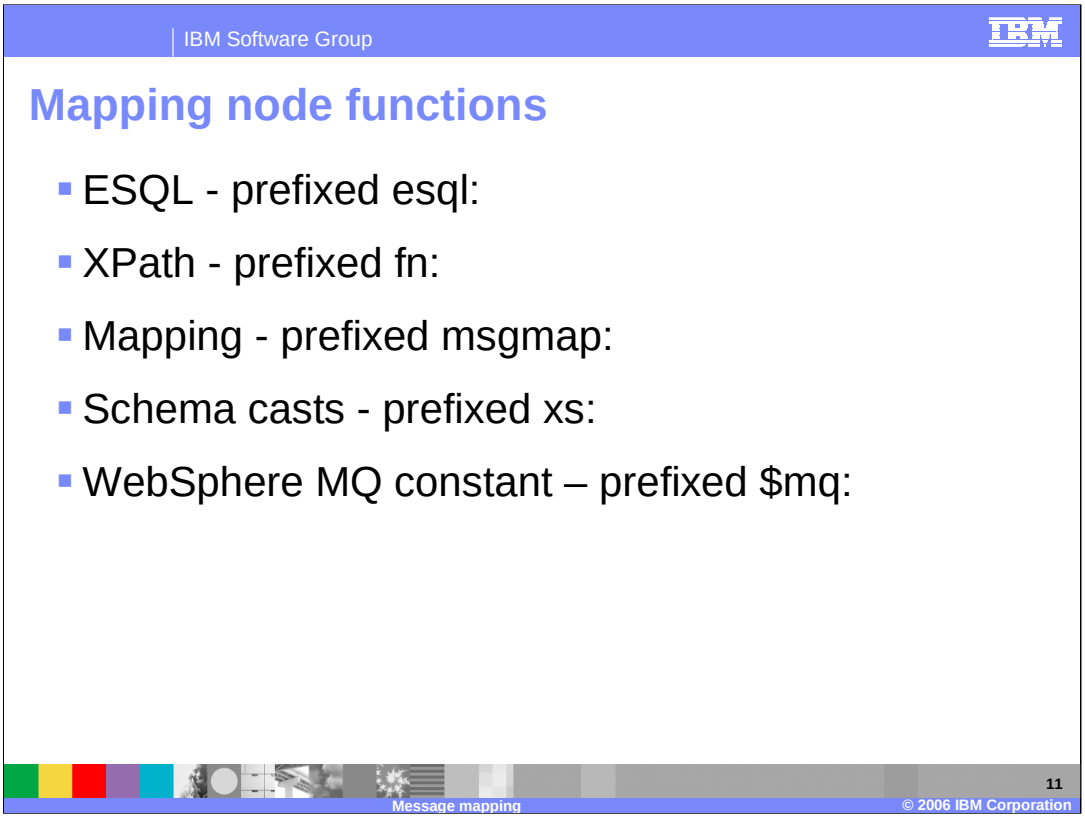
- Message flows are a graphical language for sequencing and decision making in brokers.
- Maps are transformation steps contained by a message flow.
  - (contained by reference)
- Maps concentrate on structural transformations of the message, not on sequencing steps



The message flows are a graphical paradigm for sequencing steps and decisions on complete messages.

A map is a single transformation step in brokers. A natural starting point for creating a map is to create a mapping node in a message flow.

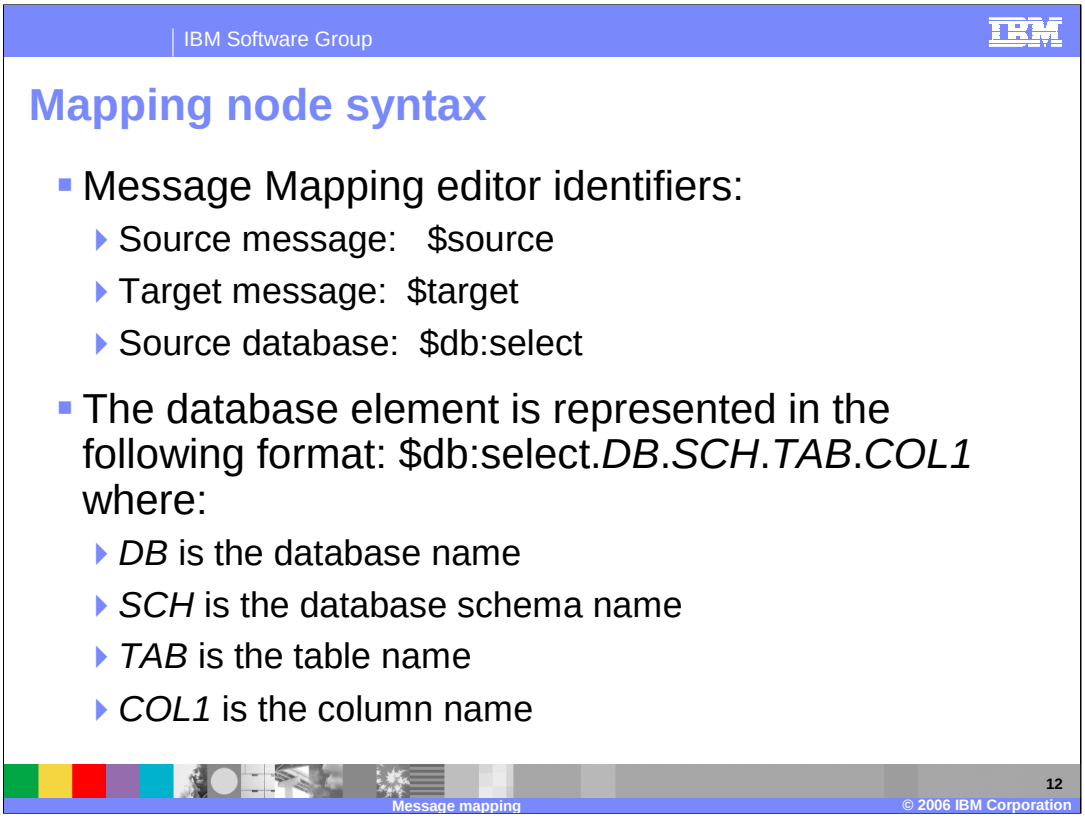
In the Mapping node properties, you can give a name for the map to use. You can also reuse an existing map, and there is a browse button to show the names of all the maps already defined.



Various functions are available in a Mapping node. These are either provided within the Mapping node or user-defined. The functions that are provided with the Mapping node are of the ESQL, XPath, Mapping, and Schema casts, identified by their prefixes.

Not all ESQL functions can be used in a Mapping node. For information about which functions are supported, and for a description of how to achieve equivalent processing for ESQL functions that are not supported, see the ESQL topics.

An additional capability provided by the mapping node is the ability to set the value of a target to a WebSphere MQ constant. The expression to set the value looks similar to a function with $mq: used as the prefix.



In a Mapping node, the source message, if present, is identified in the Message Mapping editor by “$source”.

The message tree is represented in XPath format. For example, if you have an element called Body within a source message called Envelope, this is represented in the Mapping node as: $source/soap11:Envelope/soap11:Body

Where *soap11* is a namespace prefix.

The first target message is identified by

$target

Additional target messages are identified by

$target_1, $target_2, and so on.

The first source database is identified by $db:select; additional source databases are identified by $db:select_1, $db:select_2, and so on.

Mapping node syntax (cont.)

- Use the Mapping node to:
  - make comparisons
  - perform arithmetic
  - create complex conditions

You can also use the Mapping node to make comparisons, perform arithmetic and create complex conditions.

Mapping node syntax (cont.)

- Objects that can be mapped
  - Local Environment
    - MQDestination
    - HTTP
    - Router
  - Message headers (optional)
    - MQ headers
    - HTTP headers
    - JMSTransport
  - Message elements
  - Database columns

The objects that can be mapped include Local Environment, Message headers, Message elements, and database columns.

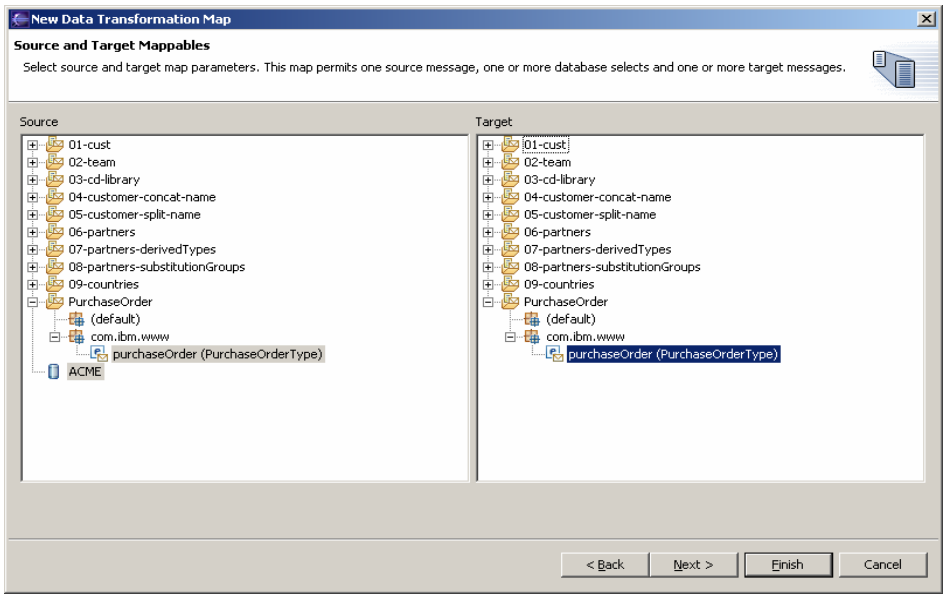
Mapping node casts

- Source and target elements can be different types
- When automatic casting cannot be done use:
  - xs:boolean
  - xs:date
  - xs:dateTime
  - xs:dayTimeDuration
  - xs:decimal
  - xs:duration
  - xs:double
  - xs:hexBinary
  - xs:int
  - xs:integer
  - xs:string
  - xs:long
  - xs:time
  - xs:yearMonthDuration

Source and target elements can be of different types in a Mapping node.

Depending on which runtime parsers are used, automatic casting cannot be done. In these cases, use one of the cast functions listed here.

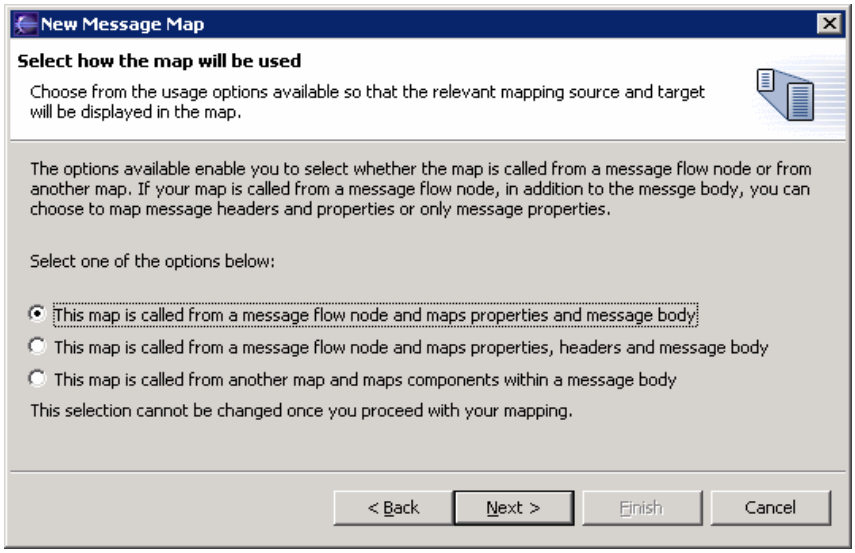
Create new map



When creating a new mapping, the Message Map wizard guides you through the process of configuring the map. This screen capture shows just one page in the wizard, where a database, an input message and an output message are selected.

The last page of the wizard provides a summary of the actions that will be taken, so you can make sure everything is exactly like you want it.

Choosing properties or headers

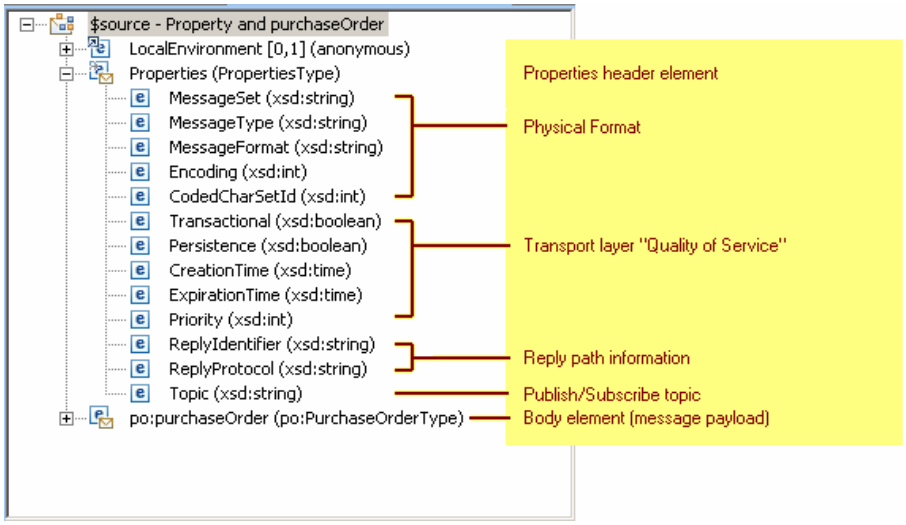


Mapping nodes always process a complete message assembly. There is a ‘simple’ version of a message assembly and a ‘full’ version. The simple version supports mapping just the Properties virtual folder; the “full” version supports the full complement of IBM headers.

The ‘Properties’ folder is a simplified view of the message transport headers. It is a virtual header that lets users specify a selection of common, important transport header values without having to deal with the entire header hierarchy. At runtime, reads and writes to the Properties folder are translated to reads and writes to the “real” headers.

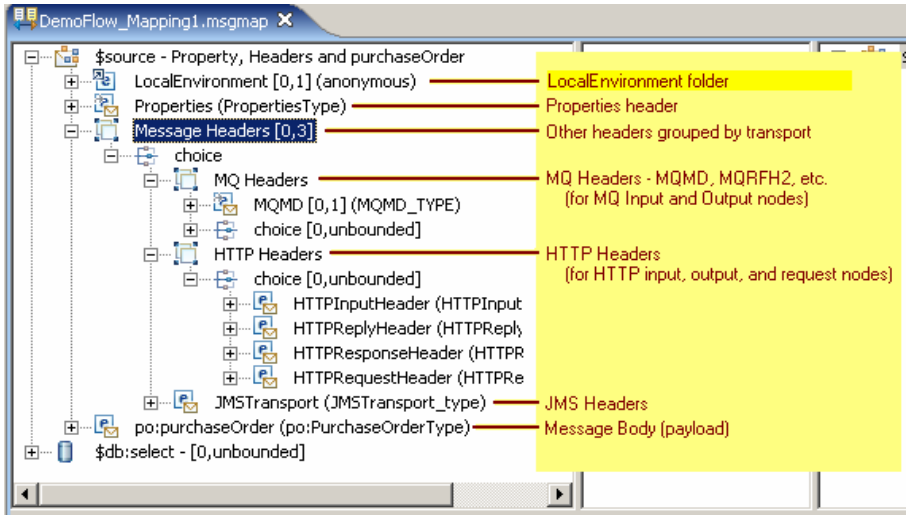
In the map tool, you can use either the properties header or the complete headers when working with a file. This is set in the New Map wizard as shown in the screen print here.

Properties mapping



This is what the simplified message assembly with properties folder looks like in the map editor, with some comments describing the content of the virtual header.

Headers mapping



Here is what the full message assembly, with the complete headers, looks like.

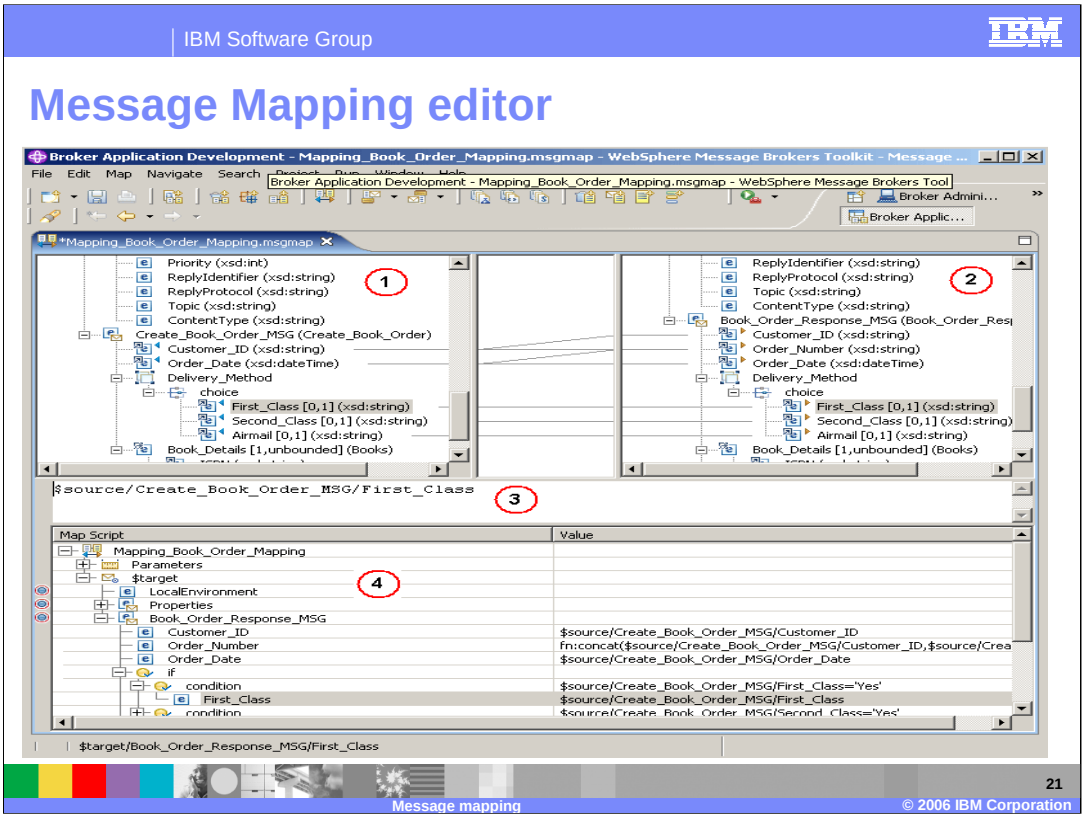
The structure of a brokers message assembly is modeled using XML schema, but provides human readable labels for many of the items to make their purpose clear.

The headers are grouped into 4 categories, based mostly on the transport used: Properties (the same virtual folder), MQ, HTTP, and JMS.

Section

Message Mapping editor

This section details the message mapping editor.



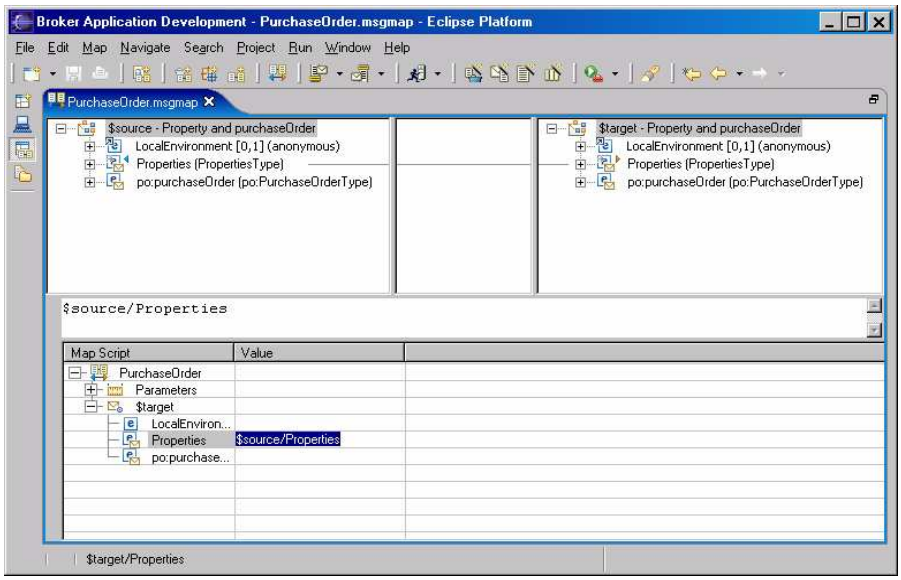
You configure a message mapping using the Message Mapping editor, which you use to set values for the message destination, message headers, and message content.

Here is an example of the Message Mapping editor. There are separate panes for working with sources, targets and expressions, and a spreadsheet view.

1. **Source pane:** displays a source message or database table
2. **Target pane:** displays a target message or database
3. **Edit pane:** displays the expression to be used to derive the target element value
4. **Spreadsheet pane:** displays a summary of the mappings in spreadsheet columns (each target field and its value)

Undo-redo support is complete. The drag and drop method can be used to associate source and target. Drag the appropriate source element or elements onto the target element or elements; you can Ctrl-click to select multiple source or target elements.

Copy message headers



To copy the headers unchanged from source to target, drag and drop the entire header from source to target.

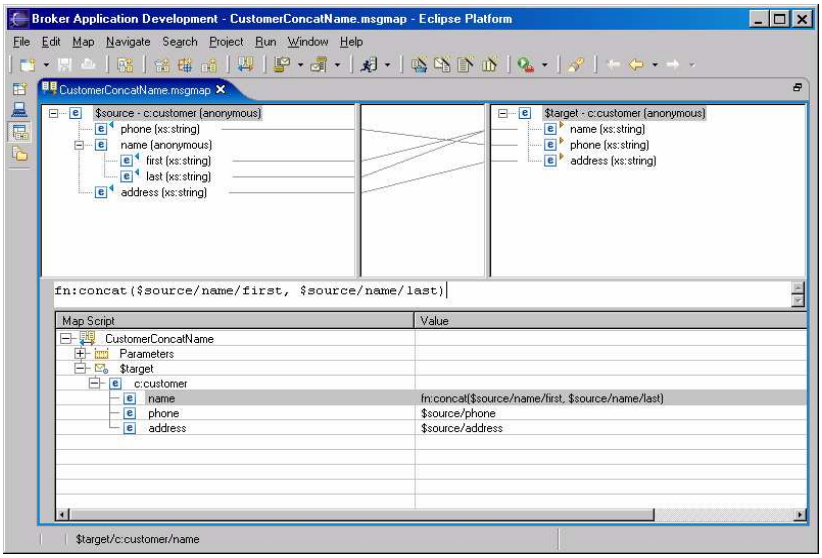
To copy some Properties values, and override others, create a complete set of mappings for Properties' children.

Submaps

- Submaps are maps that don't have Properties or Headers.
- Submaps permit multiple input elements
- Submaps can map databases
- Submaps can be called from ESQL or from other maps

Submaps are maps that do not have properties or headers; submaps are called from ESQL or from other maps. The simplicity of drag and drop mapping generation is combined with a script-generation approach to enable seamless support from simple transformations to complex transformations.

Submap scenarios

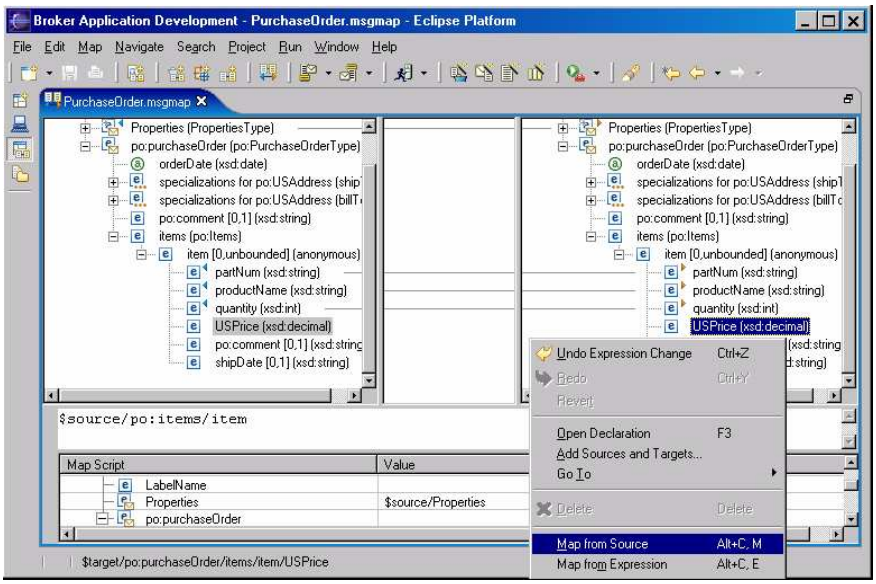


Creating a map is a matter of associating source and target elements through drag and drop.

When you perform a drag and drop, little “mapped” triangles appear in the source and target. New in version 6, lines are drawn connecting the sources and targets as well. This is a visual guide to where the mappings are, and when the source and target trees collapse the lines are replaced with a ‘compound’ line. This helps you navigate around the source and target trees, to locate which fields are mapped and which are not.

When the source or target is collapsed, a ‘compound’ mapping line indicates that there are mapped items inside. This helps you find a mapping even if the mappings are fairly sparse. A single mapping line then indicates where the source contributes to the target.

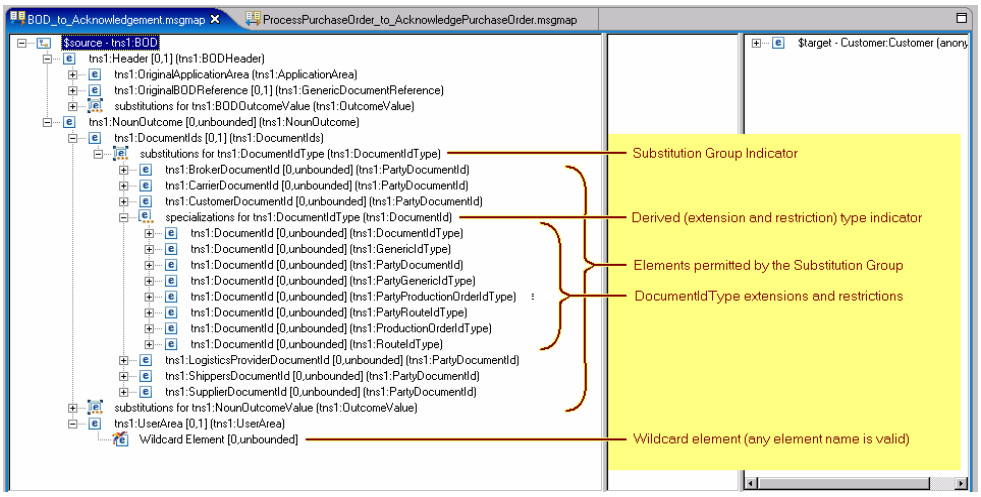
Nested mappings of repeating elements



In this scenario, target field **items** contains repeating child **item**.

You can map **item**'s children inline in a single map. In V5, this required the use of a submap.

Maps for type and substitution group



The Business Object Document (BOD) is a model defined in the OAGIS 8.0 (Open Application Group Integration Specifications) standard. It showcases the type, substitution group, and choice capabilities of V6 mapping.

The basic principle visible is that the V6 map editor provides an ‘element’ or ‘instance’ flavour to the presentation of XML schema models. Instead of showing the raw schema model, it is converted into a more user-friendly form that looks like a document hierarchy.

In the tree hierarchy, there are ‘folder’ nodes that capture the notion of a ‘set-of-values’; this screen capture shows examples of a Substitution group folder and a Specializations folder.

XML schema ‘choice’, ‘all’, and nested ‘sequence’ content models are also captured this way.

Section

Complex mappings

This section deals with complex mappings.

Map batch messages

- Message mapping that sorts, orders and splits the components into a series of batch messages
- RouteToLabel node receives multipart messages
- Message Mapping editor can build maps to transform and propagate batch messages in a single node
- Multipart messages can contain repeating embedded messages

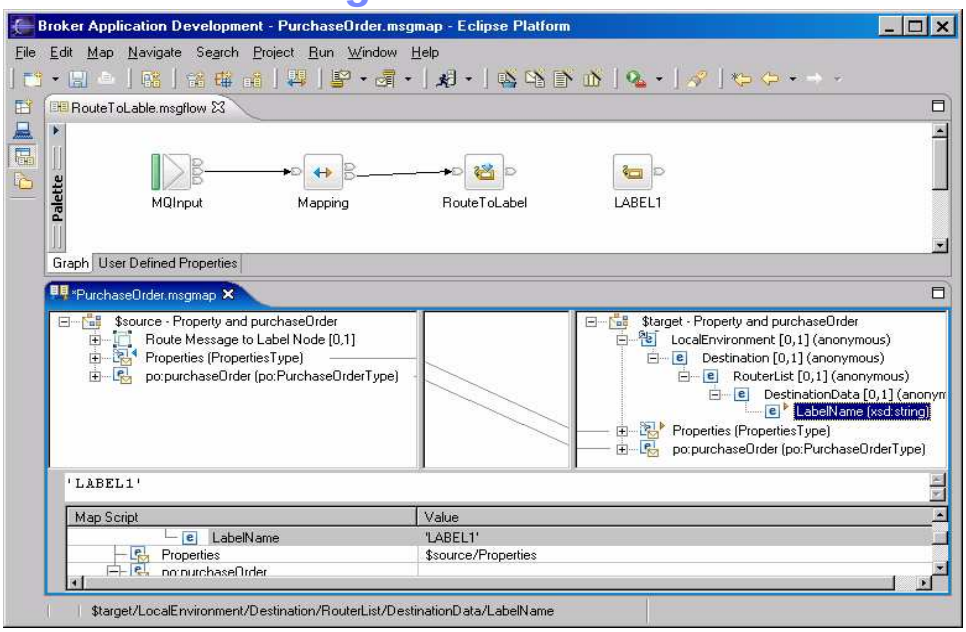
You can configure a message mapping that sorts, orders and splits the components in a multipart message into a series of batch messages. These components can be messages or objects and they can have different formats. If this is the case, each component is converted and the message is reassembled before being forwarded.

Use a RouteToLabel node in the message flow to receive multipart messages as input. The RouteToLabel node is the next node in sequence after the Mapping node, and causes the flow to jump automatically to the specified label.

Use the Message Mapping editor to build maps that transform and propagate batch messages in a single node, without having to define an intermediate data structure.

Multipart messages can also contain repeating embedded messages, where each repeated instance of a message is propagated separately. Embedded messages must be from the same message set as the parent message.

RouteToLabel integration



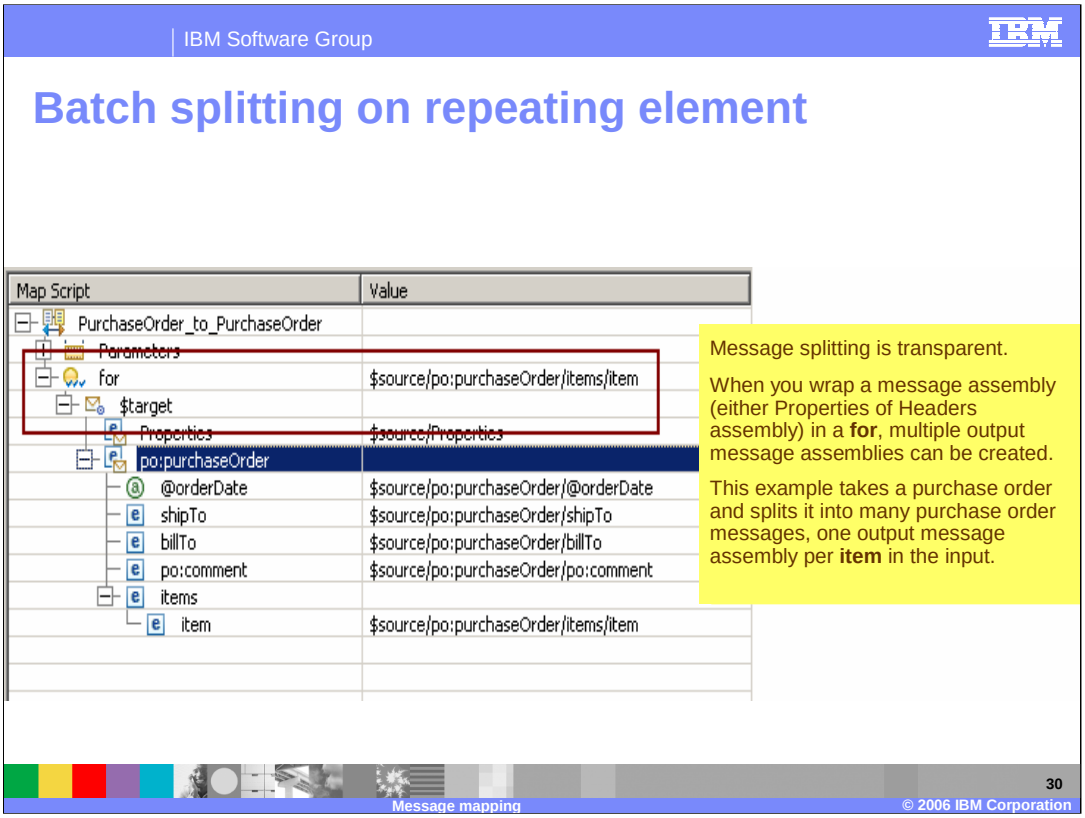
Use a RouteToLabel node in the message flow to receive multipart messages as input. The RouteToLabel node is the next node in sequence after the Mapping node, and causes the flow to jump automatically to the specified label. You can specify a single RouteToLabel value in a splitting map, for all maps that output a message assembly. You can also use conditions to set the RouteToLabel value depending on the values in the source message.

The Mapping node integrates support for this message flow node using the message assembly **$target/labelName** element. This is a virtual element that code generation recognizes and handles.

To take advantage of this feature, the user creates a message flow like **RouteToLabel.msgflow**, with a **Mapping** node in sequence with a **RouteToLabel** node.

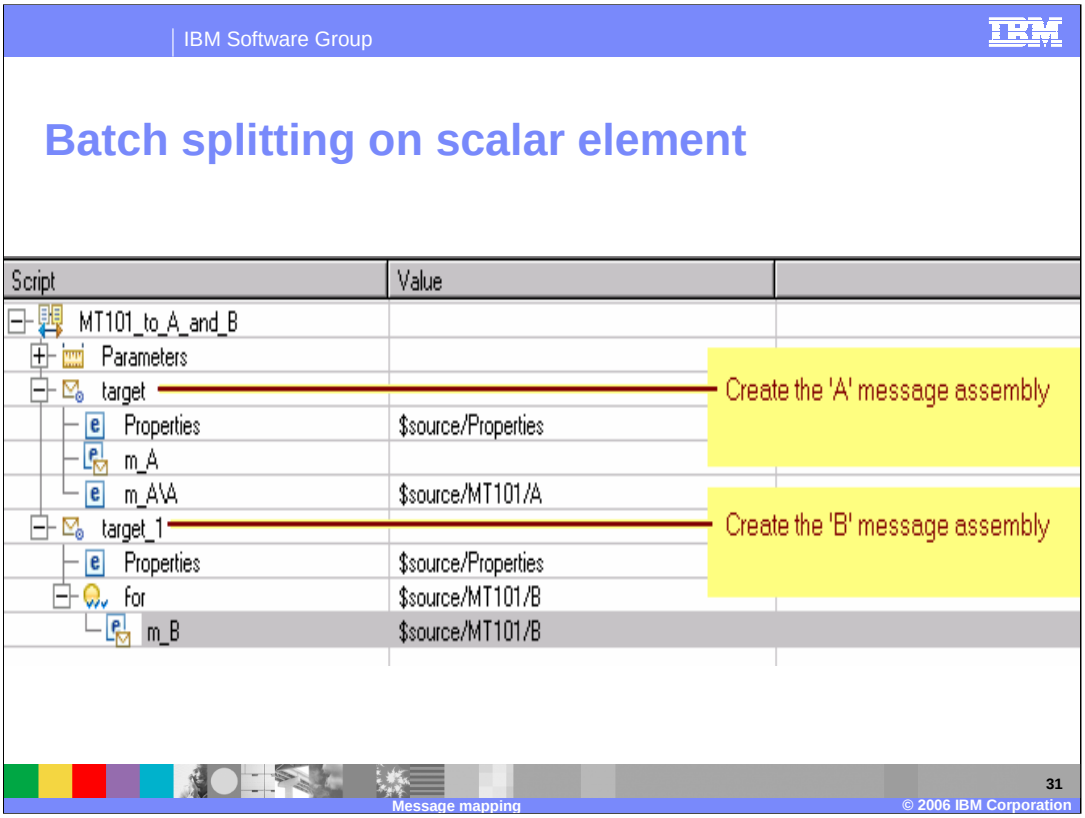
In this example the value was hard-coded to the string constant ***LABEL1***. The **labelName** value can be computed, for example by a combination of:

1. Configuring an expression, data from the source can be combined with other data including constants to compute the target value;
2. Using if/default mappings, the value can be hard-coded and selected based on source values;
3. Using a Select mapping, the **labelName** value could be extracted from a database;



This is an example of batch splitting on repeating fields or repeating nested messages.

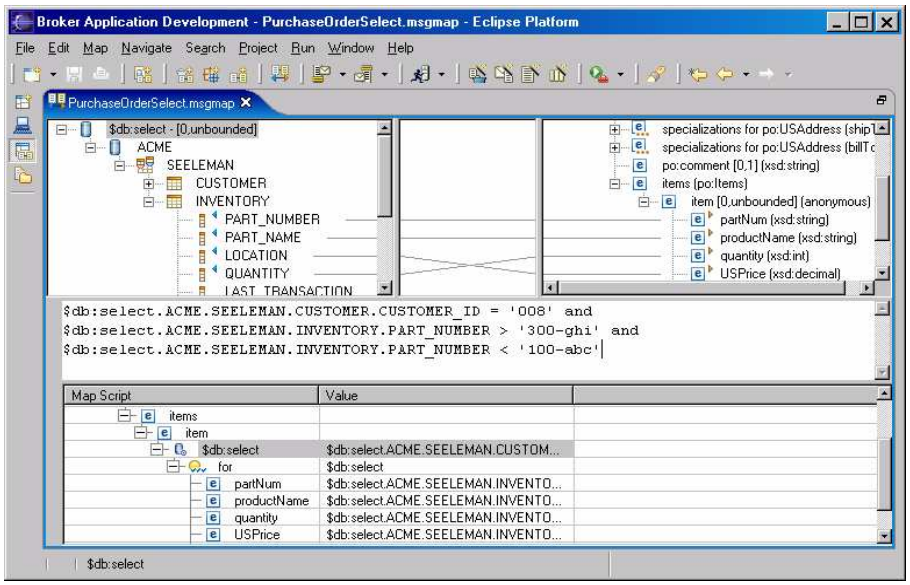
This example uses a **for** step to indicate that multiple targets will be created.



A message assembly is automatically propagated for every target assembly in a map.

In this example, there are two different target assemblies created from a single input assembly.

Using select in maps



The Spreadsheet pane can be used to invoke a number of actions, a list of which is displayed when you right-click within the Spreadsheet pane. Select is one of those functions.

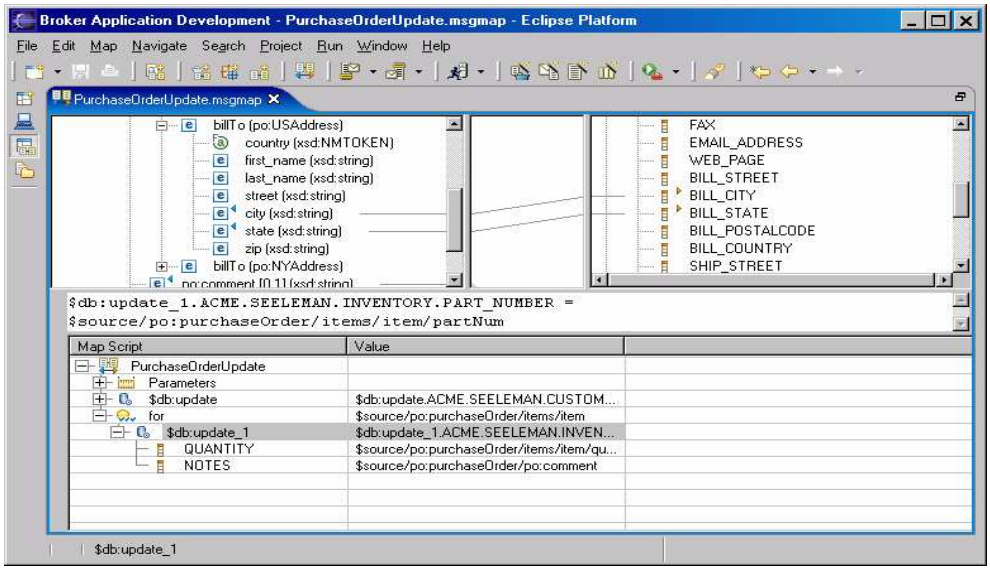
Selects are embedded in the script, so a select can be repeatedly evaluated.

Select uses a where clause, composed of database paths, XPath constants, and XPath source path expressions. The where clause controls which data gets selected from the database.

The select source shows the columns in the field. Only columns that contribute to a mapping are extracted from the database.

The result of the select expression is a repeating value accessed using the $db:select variable.

Using insert, update, and delete in maps

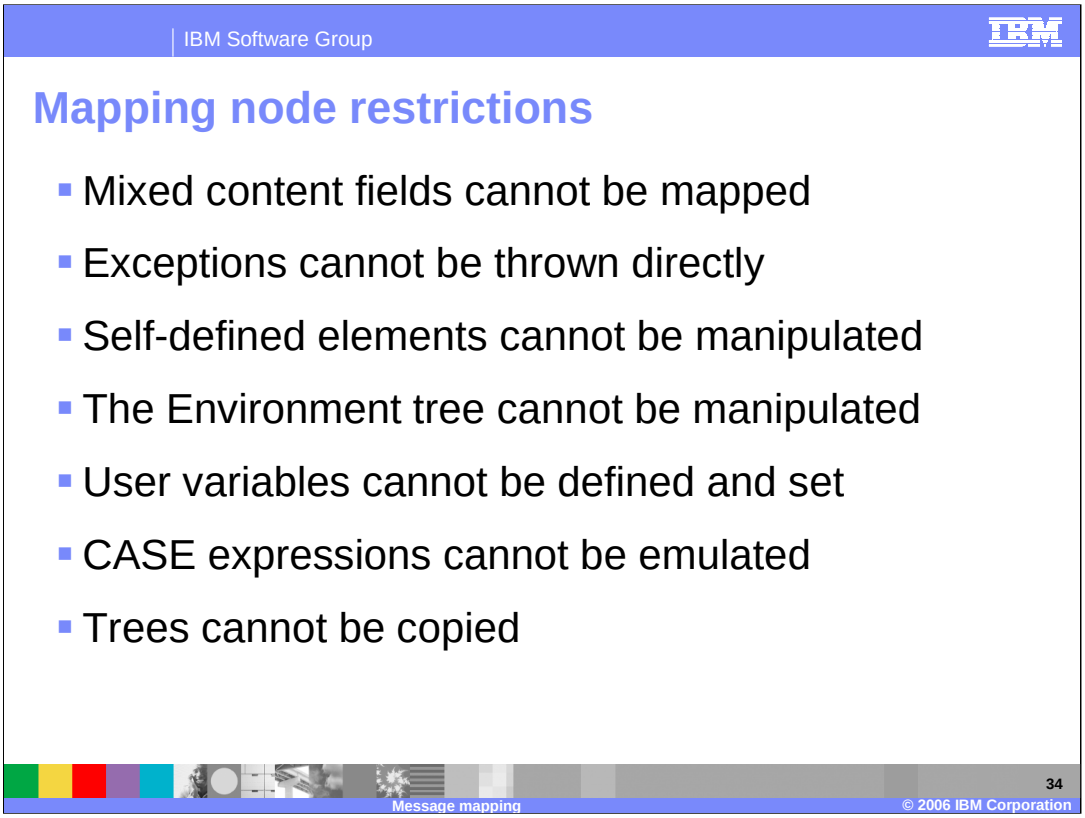


Like the select function, insert and delete are functions available in the Spreadsheet pane.

They are embedded in the script, so a select can be repeatedly evaluated.

Insert and delete use a where clause, composed of database paths, XPath constants, and XPath source path expressions. The where clause controls which data gets selected from the database.

This screen capture shows target trees, the columns in the field. The values of the target column expressions, as a batch, are processed as a single row or record.



Unless stated explicitly, you can achieve required functionality by calling an ESQL function or procedure. Here are some restrictions:

•Mixed content fields cannot be mapped.

•Exceptions cannot be thrown directly in mapping nodes.

•Self-defined elements cannot be manipulated in mapping nodes (there is limited support for wildcards when the wildcards represent embedded messages).

•The Environment tree cannot be manipulated in the Mapping node.

•User variables cannot be defined and set.

•CASE expressions cannot be emulated; you must use if/else.

•Trees cannot be copied from input to output in order to modify elements within the copied tree. For example, the following ESQL cannot be modeled in a Mapping node: SET OutputRoot.MQMD = InputRoot.MQMD; SET OutputRoot.MQMD.ReplyToQ = 'NEW.QUEUE'; You must set each field in the structure individually if you intend to modify one or more sibling fields.

V6 mapping scalability

- Maps support in-line expansion of repeating elements
  - minimizes the number of different map files a user needs to manage
- Submaps support modularization of large transformations
  - large message definitions, like ACORD, can be easily handled as collection of pieces

In V6, maps support in-line expansion of repeating elements which minimizes the number of map file needed. Submaps support modularization of large transformations.

The Association for Cooperative Operations Research and Development (ACORD) definitions provide a good scalability sample because they:

1. Show nested repeating use
2. Show submap creation and use
3. Save the map, control returns to you in seconds. Even large builds complete in minutes instead of tens of minutes.

Section

Summary and references

And, in summary…

Summary

- Map creation
- Mapping node
- Message Mapping editor
- Complex mapping

…this module provided information on using mapping in the broker including:

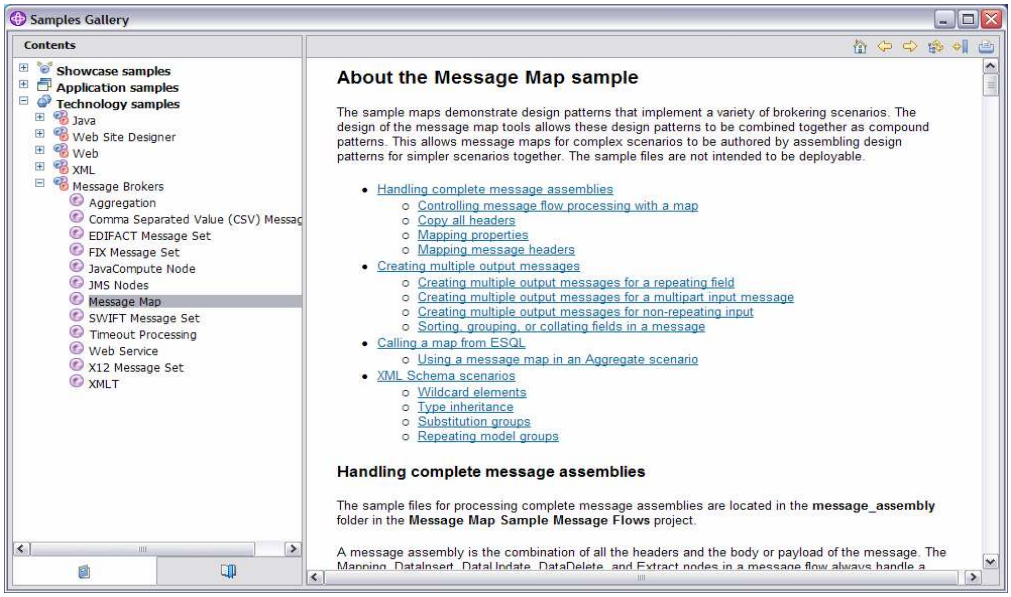
•Creating a map

•Configuring the Mapping node

•Using the Message Mapping editor

•Complex mapping techniques

Samples



To see how this function is used in a real example, see the Message Map sample in the Samples Gallery.

References

- WebSphere Message Broker library:

http://www-306.ibm.com/software/integration/wbimessagebroker/library/

- WebSphere Message Broker Information Center:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

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.

Information is provided "AS IS" without warranty of any kind. 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.