

IBM Software Group | WebSphere Software

WebSphere Message Broker What's New in Toolkit 6.0.2

David Hardcastle

IBM Hursley Lab

(david_hardcastle@uk.ibm.com)

SOA on your terms and our expertise



© 2006 IBM Corporation

Trademarks

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

AIX	AS/400	DB2
IBM	iSeries	MQSeries
MVS	OS/390	SupportPac
System/390	WebSphere	xSeries
z/OS	400	

- Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc, in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Other company, product, and service names may be trademarks or service marks of others.

Agenda

- Why a 6.0.2 toolkit release?
- Key AD improvements
 - Illustrated using a Demo !
- Testing
- Other Improvements
- Looking ahead

About the 6.0.2 toolkit

- **Why**

- There is opportunity for improvement in the helping new users get up to speed quickly with the Toolkit

- **What**

- Aimed primarily at simplifying out of box experience
- Primarily targeting core ESB-like scenarios
 - But not strictly limited to that
- Improving “time to value”
 - Hopefully changes benefit all existing users too

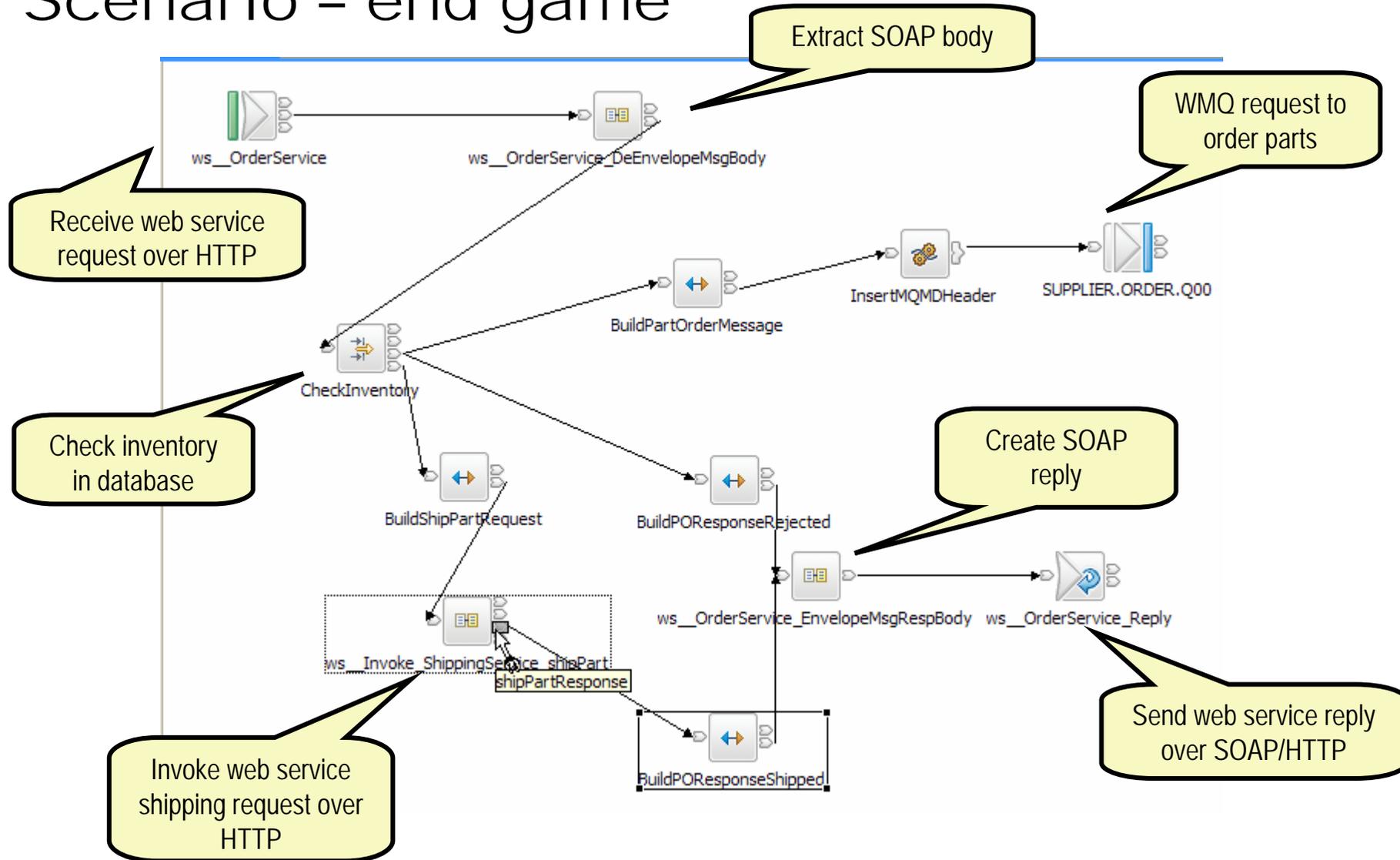
- **When**

- Some improvements in Fix Pack 2 (6.0.0.2)
- New **6.0.2** release of Toolkit in December 2006
- Via RPU Upgrade

Key AD improvements

- Illustrated using a simple scenario
- Order Processing
 - Expose a web service (SOAP/HTTP) for placing purchase order
 - Flow will check for sufficient inventory (stock)
 - Insufficient stock for order...
 - Place order from supplier via MQ message to CICS (say)
 - ‘Out of stock’ response to caller
 - Sufficient stock for order...
 - Invoke different web service to place shipping order
 - Positive response to caller

Scenario - end game

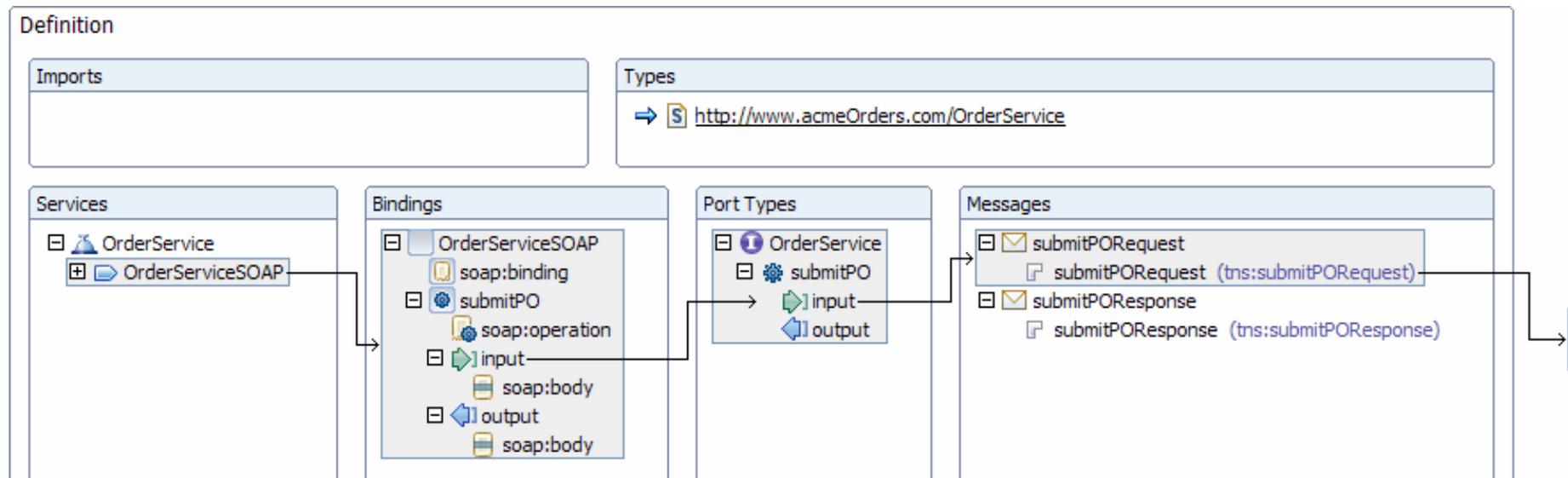


Scenario - challenges

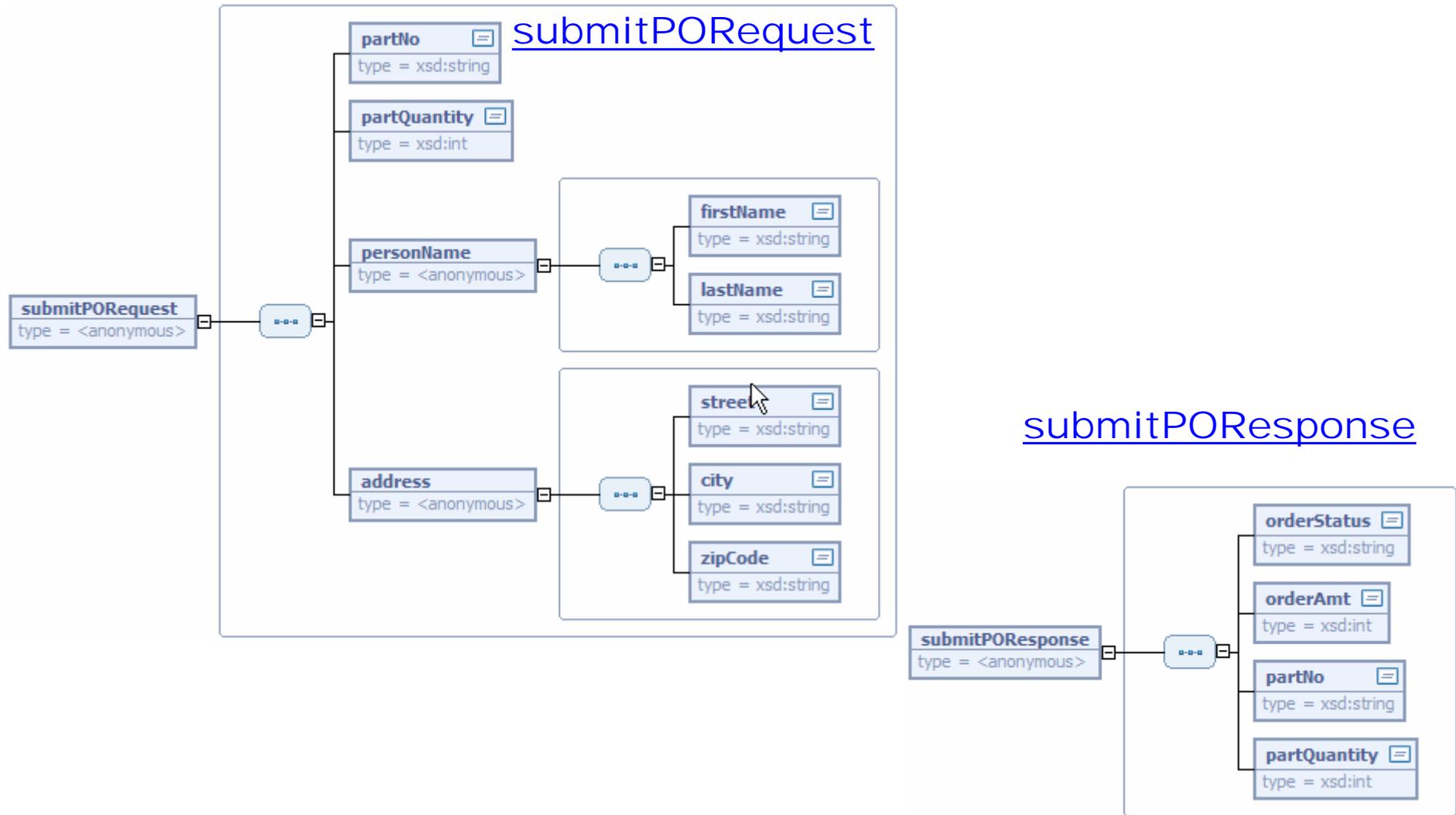
- Creating message set from WSDL and XSD files
 - WSDL for the web service purchase order operation to be exposed
 - WSDL for the web service shipping operation to be invoked
- Configuring and using HTTP nodes
 - Input and reply for the web service purchase order operation to be exposed
 - Request for the web service shipping operation to be invoked
- Dealing with the SOAP messages
 - Body has to be extracted from SOAP envelope for easy mapping
- Dealing with mixed protocols
 - Body has to be placed in a SOAP envelope to invoke web service
 - WMQ headers have to be created to use MQ output
- Choosing/finding the appropriate nodes to use for each task
- Testing ... how to send a SOAP message to HTTP Input node and see the reply from the HTTP Reply node?

Scenario: WSDL for Purchase Order

OrderService.wsdl

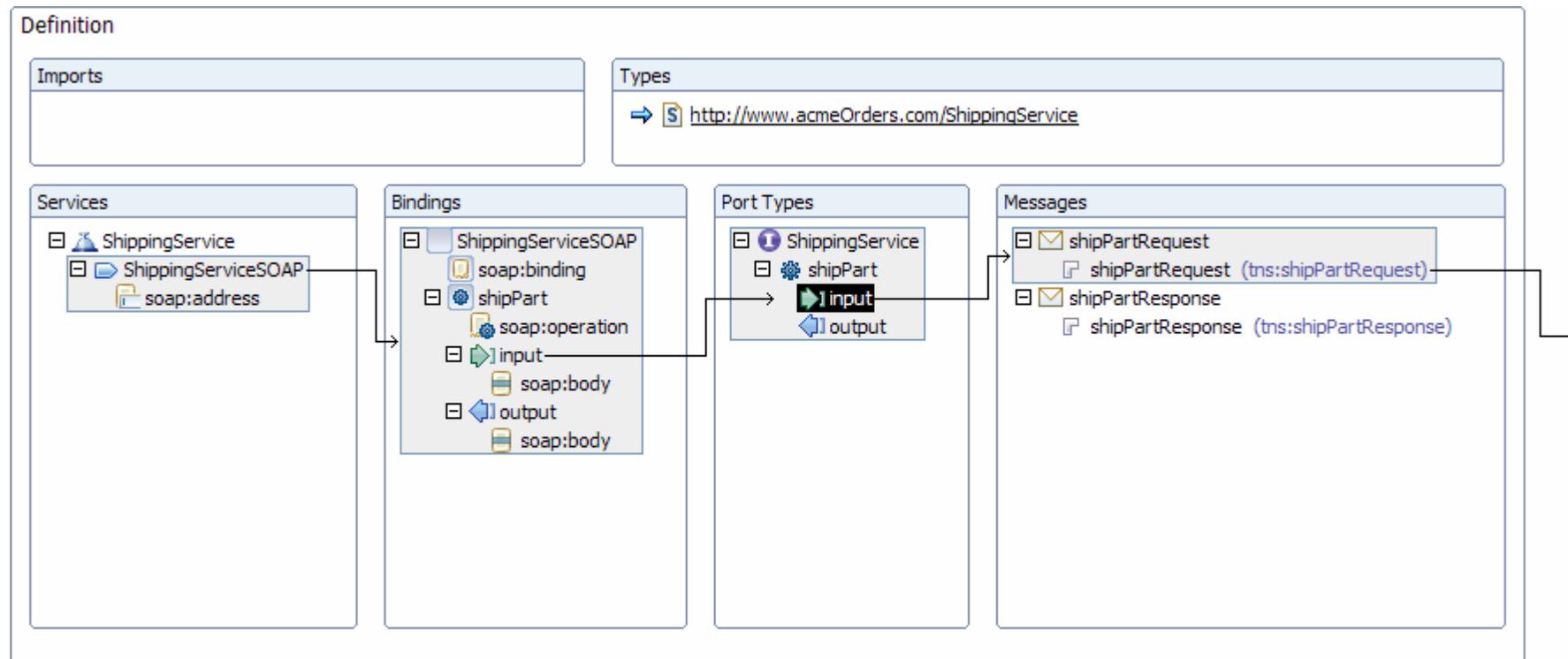


Scenario: XSDs for Purchase Order

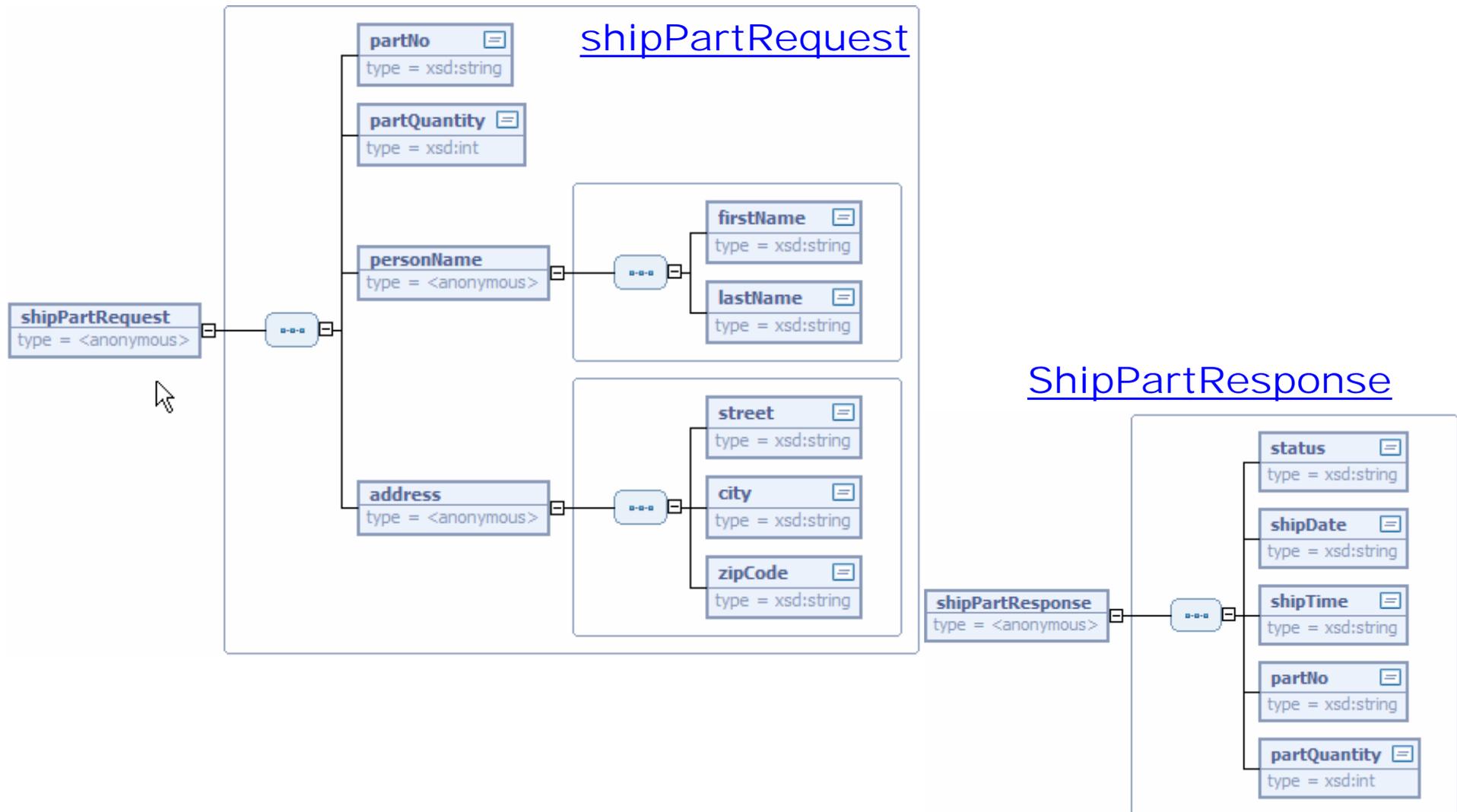


Scenario: WSDL for Shipping Service

[ShippingService.wsdl](#)

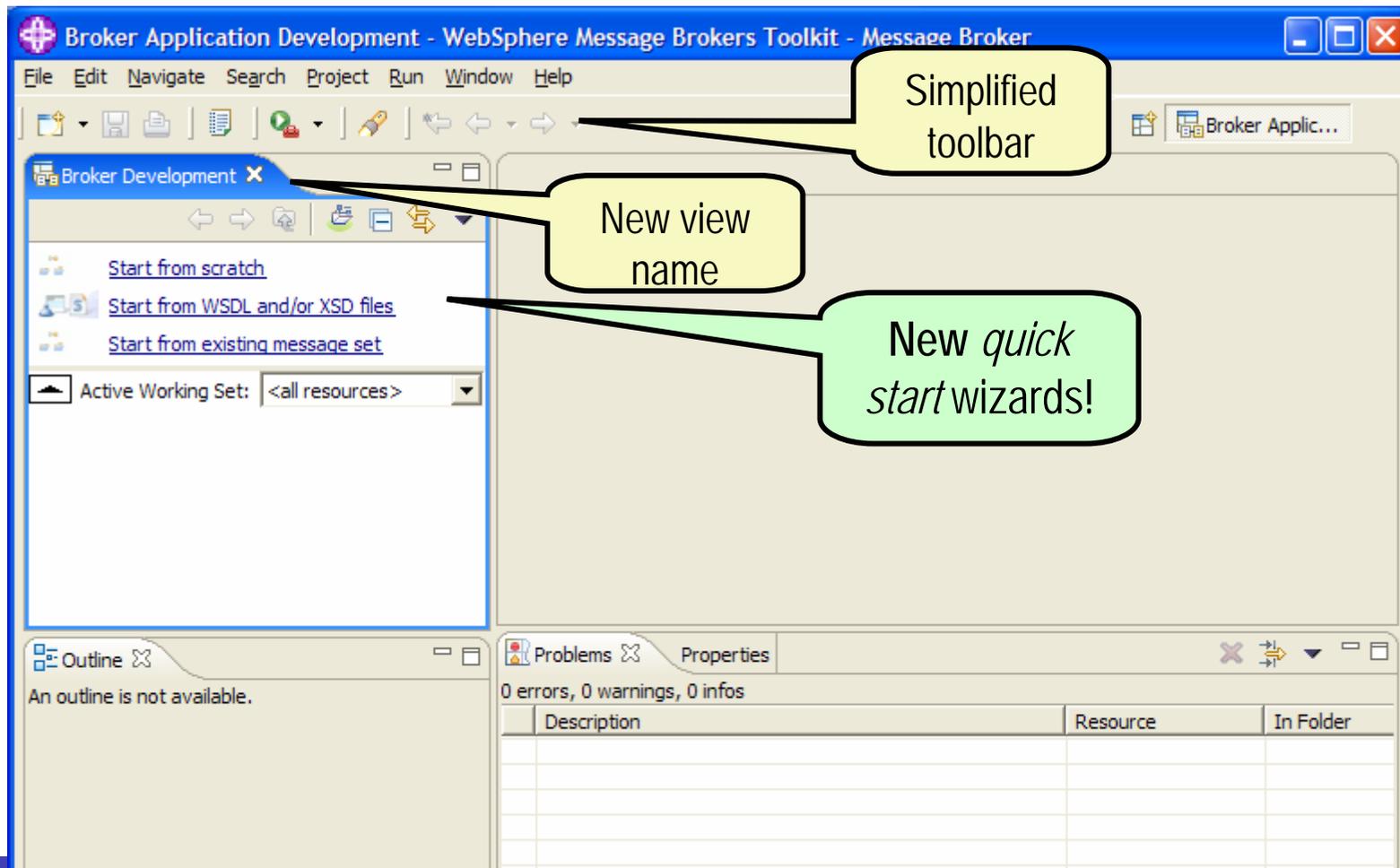


Scenario: XSDs for Shipping Service



Scenario: Step 1

- Where do I begin? Fire up Toolkit 6.0.2!



Scenario: Step 2a – page 1

- Choose root name for new “application”

Quick Start

New Message Broker Application

Set up the basic resources required to develop a Message Broker application using WSDL and XSD files as a starting point.

Message flow project name: PurchaseOrder

Message set project name: PurchaseOrderMessageSet

Message set name: PurchaseOrderMessageSet

Message flow creation

Create a new message flow in my flow project

Message flow name: PurchaseOrderFlow

Working set creation

Create a new working set for these resources

Working set name: PurchaseOrder

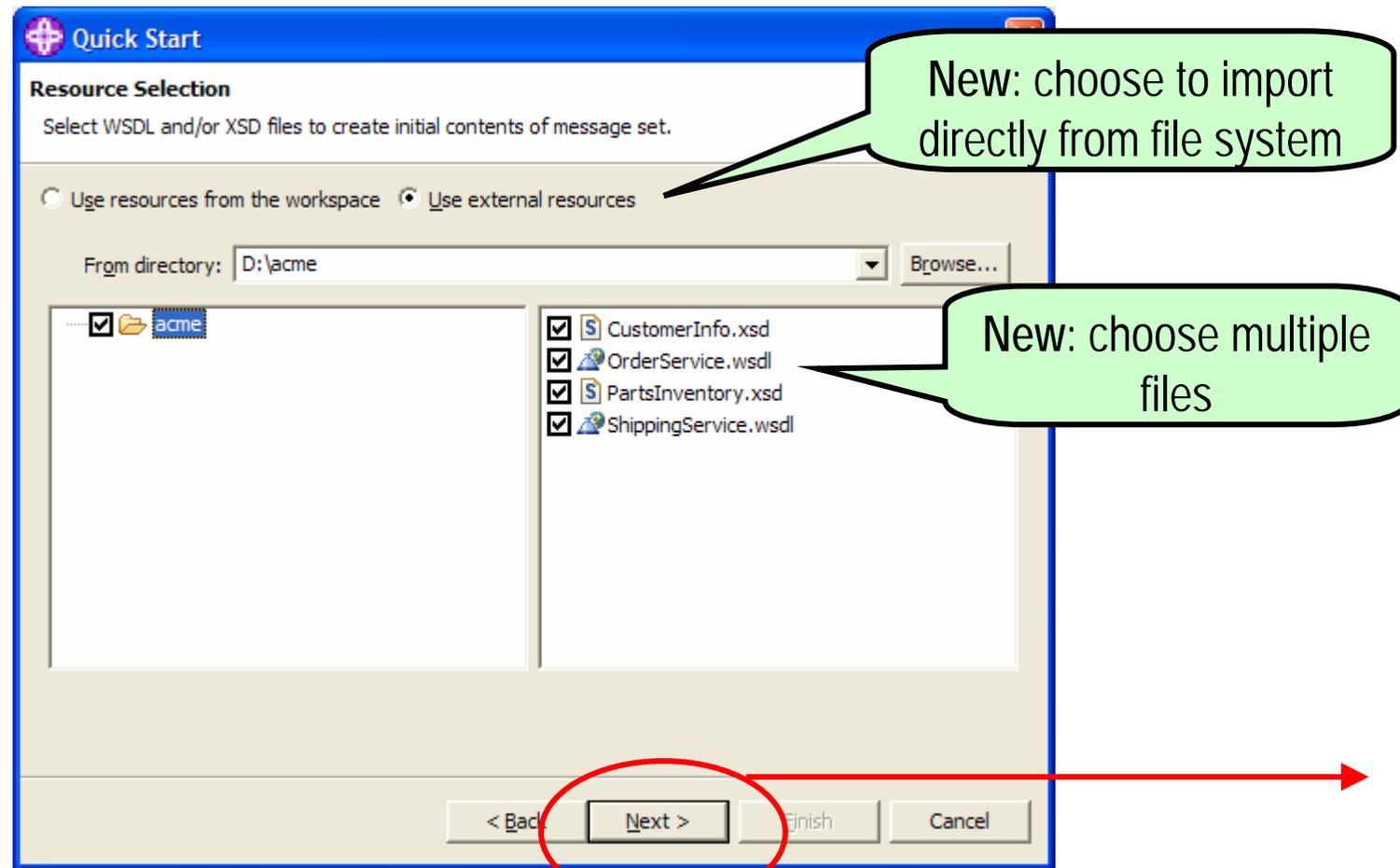
< Back **Next >** Finish Cancel

Type this

These names derive from what you type

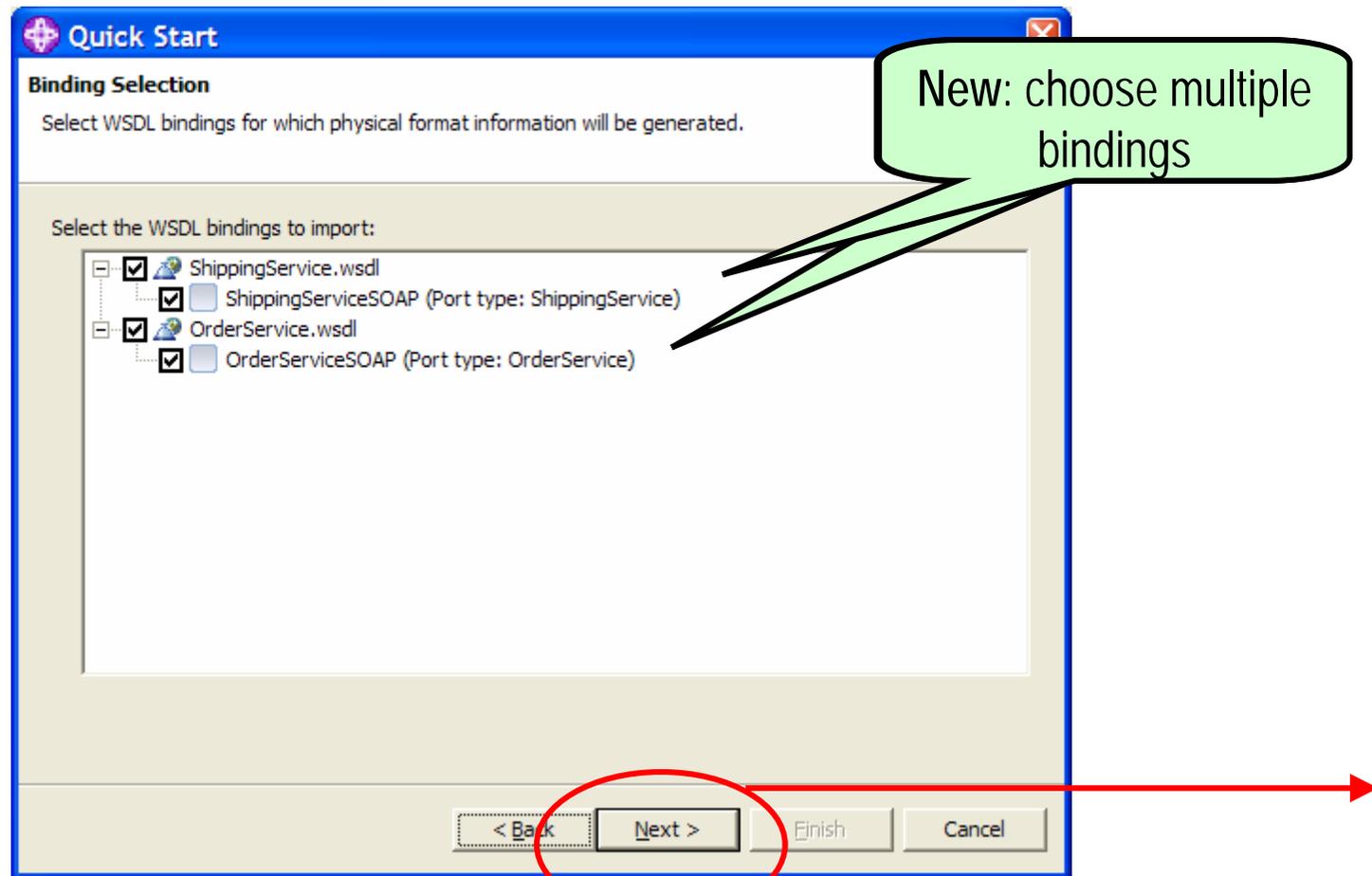
Scenario: Step 2b – page 2

- Choose WSDL and XSD files from file system



Scenario: Step 2c – page 3

- Choose bindings from which to model messages



Scenario: Step 2d – page 4

- Choose which XSD messages to model

The screenshot shows the 'Quick Start' dialog box with the 'Message Selection' tab active. The dialog prompts the user to 'Select global elements from which to create messages'. Under the 'Global elements' section, a tree view shows two XSD files: 'PartsInventory.xsd' and 'CustomerInfo.xsd'. Under 'PartsInventory.xsd', the 'orderPart' element is selected. Under 'CustomerInfo.xsd', the 'customerResponse' element is selected. All four elements have their respective checkboxes checked. A yellow callout bubble points to the 'orderPart' element with the text 'Choose global elements to create messages from'. A green callout bubble points to the 'customerResponse' element with the text 'Learn you can drag and drop imported wsdl'. A red callout bubble points to the 'Finish' button at the bottom of the dialog, which is circled in red. A 'Tip' dialog box is overlaid on the bottom right, containing the text: 'You can drag and drop an imported WSDL file into the message flow editor to easily invoke the service or expose the message flow as the service.' Below the tip text is a checkbox labeled 'Do not show this tip again' and an 'OK' button. The 'Quick Start' dialog has navigation buttons at the bottom: '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Finish' button is highlighted with a red circle and a red arrow pointing to the tip dialog.

Scenario: Step 2e - after wizard

The screenshot shows the IBM WebSphere Message Brokers Toolkit interface. The main workspace displays a project tree for 'PurchaseOrder' with the following structure:

- PurchaseOrder
 - Flows
 - (default broker schema)
 - PurchaseOrderFlow.msgflow
 - PurchaseOrderMessageSet
 - PurchaseOrderMessageSet
 - messageSet.mset
 - Message Definitions
 - com.acmeorders.www.customerinfo
 - com.acmeorders.www.orderservice
 - com.acmeorders.www.partsinfo
 - com.acmeorders.www.shippingservice
 - org.xmlsoap.schemas.soap.envelope
 - importFiles
 - CustomerInfo.xsd
 - OrderService.wsdl
 - PartsInventory.xsd
 - ShippingService.wsdl
 - log

Callouts and annotations:

- New - generated:**
 1. Message flow and project
 2. Message set and project
 3. Working set to hold projects – *and navigator is filtered to only show projects in that working set*
- Message flow editor pre-opened:** Points to the 'PurchaseOrderFlow.msgflow' file in the project tree.
- SOAP Envelope schema imported:** Points to the 'org.xmlsoap.schemas.soap.envelope' schema in the Message Definitions folder.
- Imported files:** Points to the 'importFiles' folder containing various XSD and WSDL files.

The bottom of the interface shows a table with project details:

name	path	type
PurchaseOrderFlow	/PurchaseOrder/PurchaseOrderFlow.msgflow	Message Flow

About Working Sets

- New for Broker Development perspective in 6.0.2 ... but an existing eclipse concept. A named group containing one or more projects.
 - Simply a way to filter Broker Development view to projects related to particular scope or application

The image shows a composite screenshot of the Eclipse IDE interface. At the top left, a project browser shows a tree view with 'PurchaseOrder' selected. A red arrow points from this selection to the 'New Working Set' dialog box. The 'New Working Set' dialog has a title bar 'New Working Set' and a subtitle 'Broker Working Set'. It contains a text field for 'Working set name:' and a list of 'Working set content:' with checkboxes. The 'PurchaseOrder' project is selected in the list. Below this, the 'Edit Working Set' dialog is shown, with the 'Working set name:' field filled with 'PurchaseOrder' and both 'PurchaseOrder' and 'PurchaseOrderMessageS' selected in the 'Working set content:' list. At the bottom left, the 'Preferences' dialog is open to the 'Broker Development' section. A checkbox labeled 'Hide working set and quick start links in Broker Development view' is present. A yellow callout box with a black border and a speech bubble tail points to this checkbox, containing the text 'Can hide if not used'. The 'ON DEMAND BUSINESS' logo is in the bottom right corner.

Scenario: Step 3

- Drag and drop WSDL, defining web service to expose in this flow, to flow canvas

The screenshot shows the 'Broker Development' environment. On the left, a project tree under 'PurchaseOrder' shows 'importFiles' containing 'OrderService.wsdl'. A red arrow points from this file to the 'PurchaseOrderFlow.msgflow' canvas. A 'Palette' on the right lists various components. A dialog box titled 'New web service usage pattern' is open, with the following details:

- Web service usage pattern:** Specify the details of how the selected web service will be used in the message flow. Only HTTP bindings are supported.
- Web service usage pattern:**
 - I want to expose my message flow as this web service
 - I want to invoke this web service from my message flow
- Binding:** OrderServiceSOAP
- Operations:**
 - submitPO (submitPOResult submitPORequest)

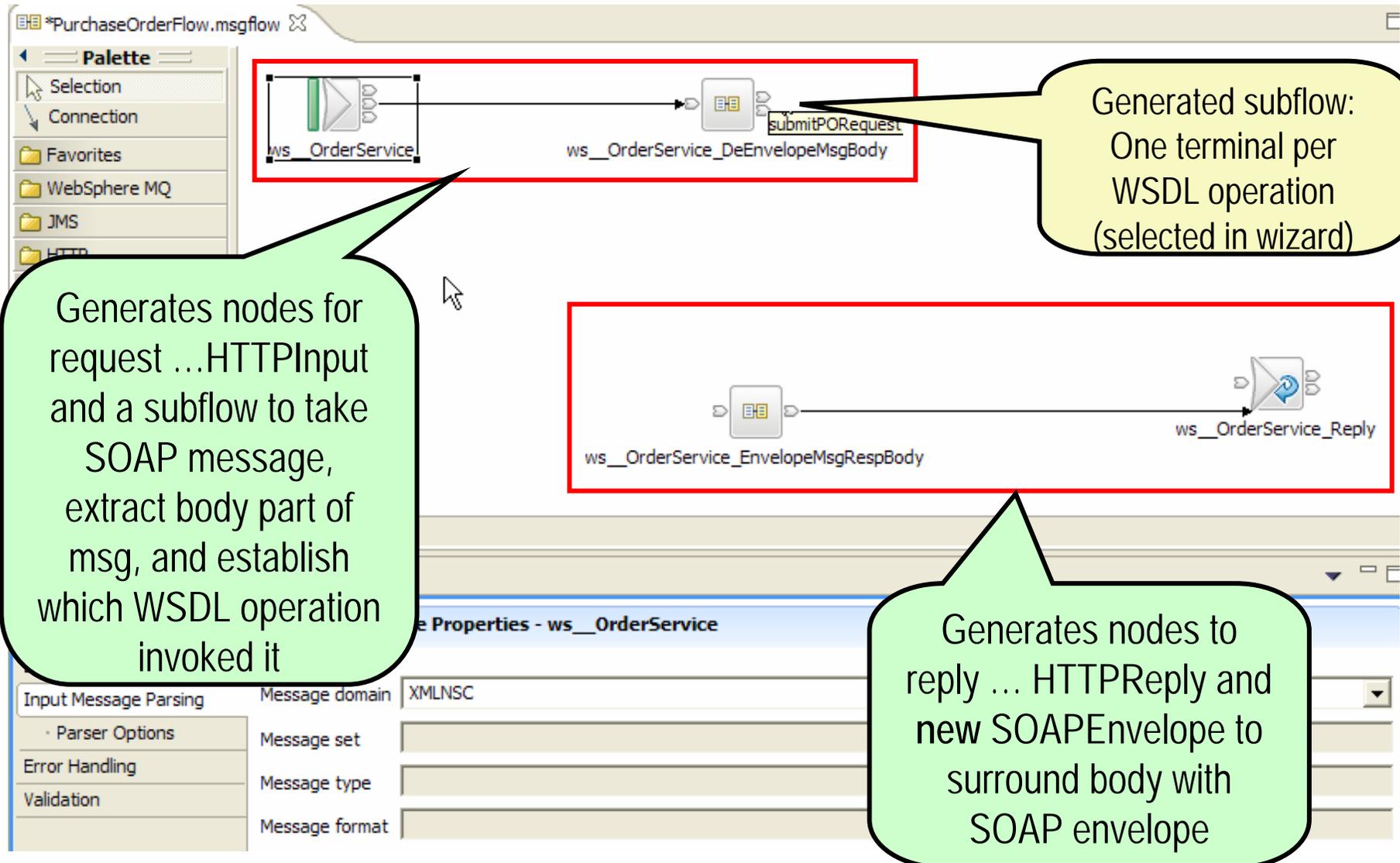
At the bottom of the dialog are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

New: can now drag and drop WSDL file from a message set project to flow editor

Select WSDL operations

Can expose flow as a web service
Can consume/call an external web service

Scenario: Step 3 – after drag and drop



Scenario: step 3 – Inside the subflow

The screenshot displays the IBM WebSphere Message Brokers Toolkit interface. The main workspace shows a message flow diagram for the subflow `OrderService_Gen_Front_Subflow.msgflow`. The flow starts with an `in` node, followed by a `ws_DeEnvelop_wsdOper` node, which then branches into a `failure` node and a `submitPORrequest` node. The `submitPORrequest` node is connected to another `submitPORrequest` node. A palette on the left lists various nodes, and a properties window at the bottom shows the configuration for the `SOAPDeEnveloper Node Properties`.

Generated subflow

New: SOAPExtract node. Extracts body from SOAP envelope, and routes to Label for each WSDL operation

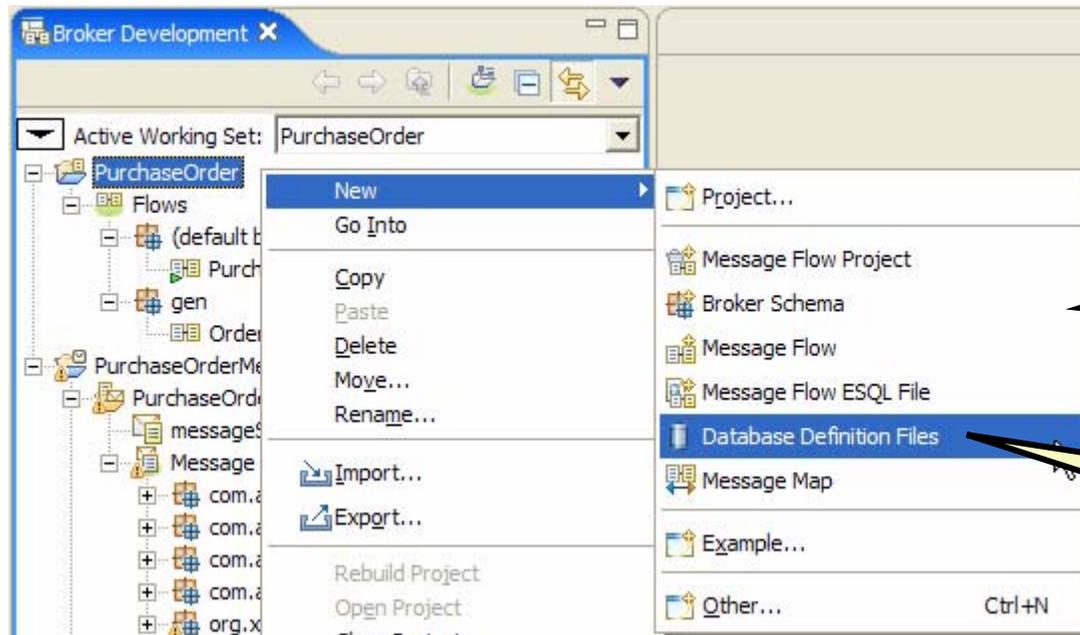
A Label and output node pair per WSDL operation (selected in wizard)

About Drag and Drop support

- We are increasing drag and drop support over time.
- As of 6.0.2, you can now drag and drop the following onto the flow editor:
 - **WSDL** files from message sets, as we have seen.
 - **XSL** files to create and/or configure XMLTransformation nodes
 - **Java** classes to create JavaCompute node and configure its class name property
 - **ESQL** modules to create Compute node and configure its ESQL module name property
 - Sub-flows, as before

Scenario: Step 4

- Create database connection for inventory table



Simplified New menus:
only show what is
applicable for current
selection

Renamed from "RDB Definition
Files"

Scenario: Step 4

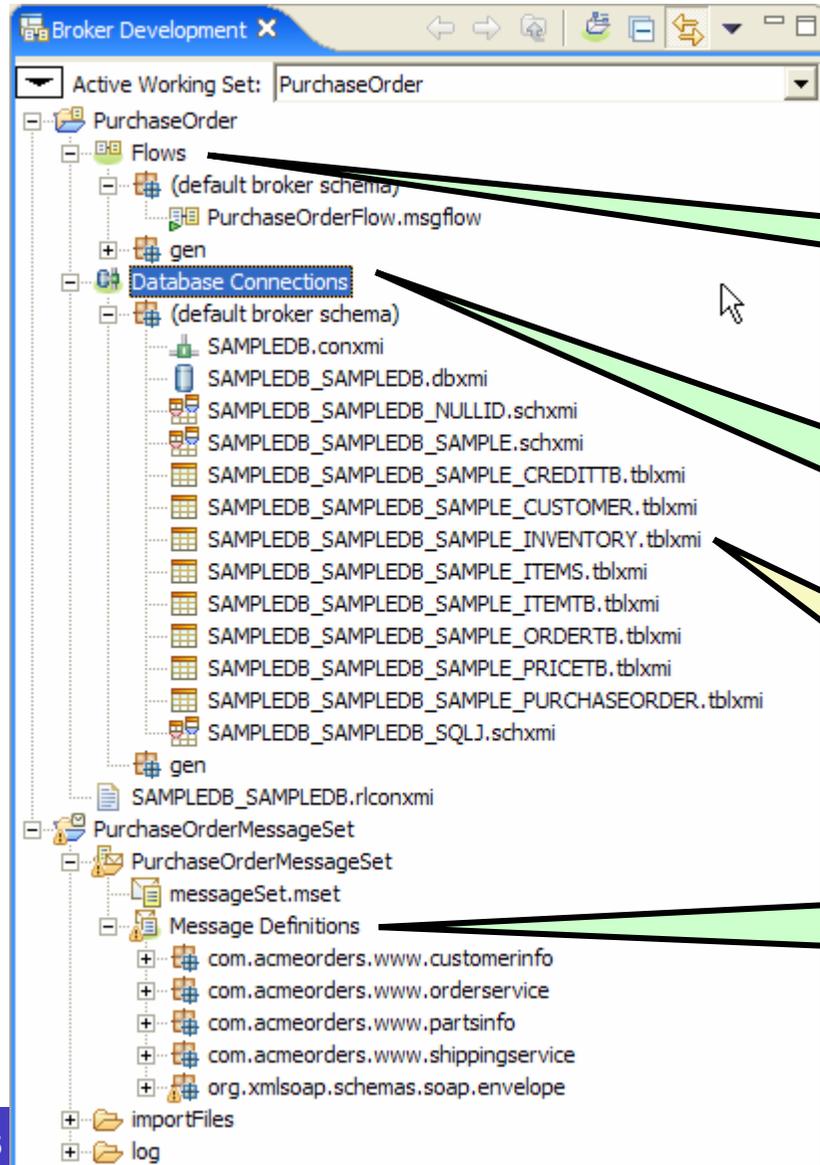
- Create database connection for inventory table

The screenshot shows two wizard windows. The top window is titled "New Database Definition Files" and contains the following fields: "Message flow project:" with a dropdown menu showing "PurchaseOrder", "Connection name:" with a text field containing "SAMPLEDB", "Flow organization:" with a checked "Use default" checkbox, and "Schema:" with a dropdown menu showing "(default)". A "New..." button is located to the right of the "Message flow project" dropdown. The bottom window is titled "Import RDB definition" and contains "Project:" and "Schema:" dropdown menus, both with "Browse..." buttons next to them. The "Project:" dropdown is currently empty, and the "Schema:" dropdown shows "(default)".

Callouts and annotations:

- New: moved this page to front of wizard** (green bubble pointing to the top window)
- New: use dropdown and "New..." vs entry field and Browse...** (yellow bubble pointing to the "New..." button)
- New: pre-filled with project of selected file** (yellow bubble pointing to the "PurchaseOrder" dropdown)
- New: changed prompt for broker schema** (yellow bubble pointing to the "Schema:" dropdown)
- New: changes made to all of toolkit's New wizards** (green bubble pointing to the "Message flow project:" dropdown)
- Old wizard: project prompt page came up separately at end** (yellow bubble pointing to the "Project:" dropdown in the bottom window)

Scenario: Step 4



■ The navigator after database connection created...

New: Flows category

New: Database Connections category

The table we are interested in

New: Message Definitions category

About Categories

- New for 6.0.2 are categories shown in the navigator, to help users understand at a glance what they are looking at:

- Message Flow projects:

- Flows
- Maps
- ESQs
- Database Connections
- Enqueues

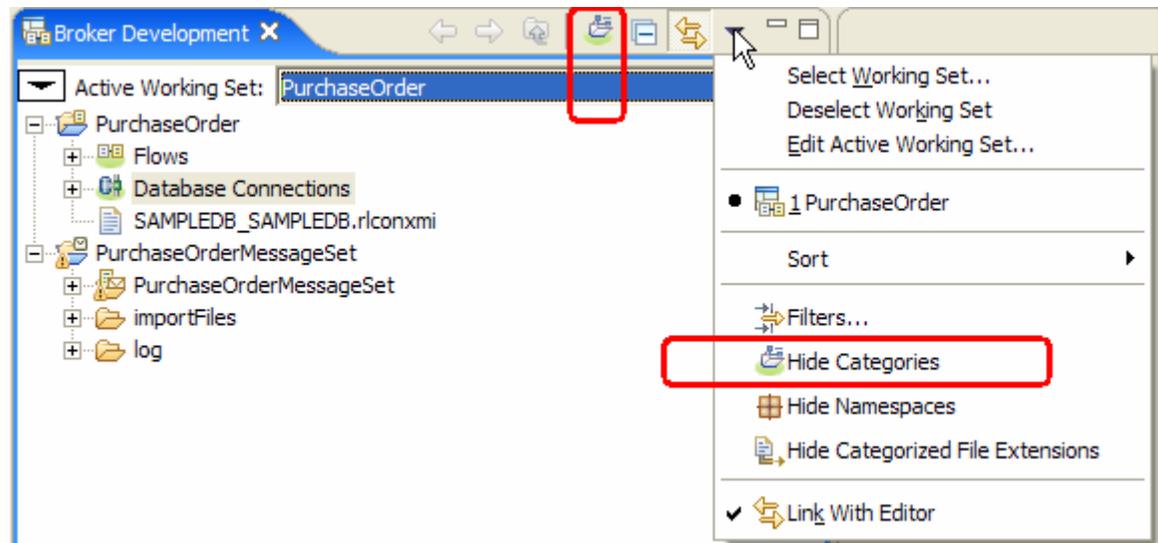
- Message Set projects:

- Message Definitions
- Message Categories

- You can turn this off

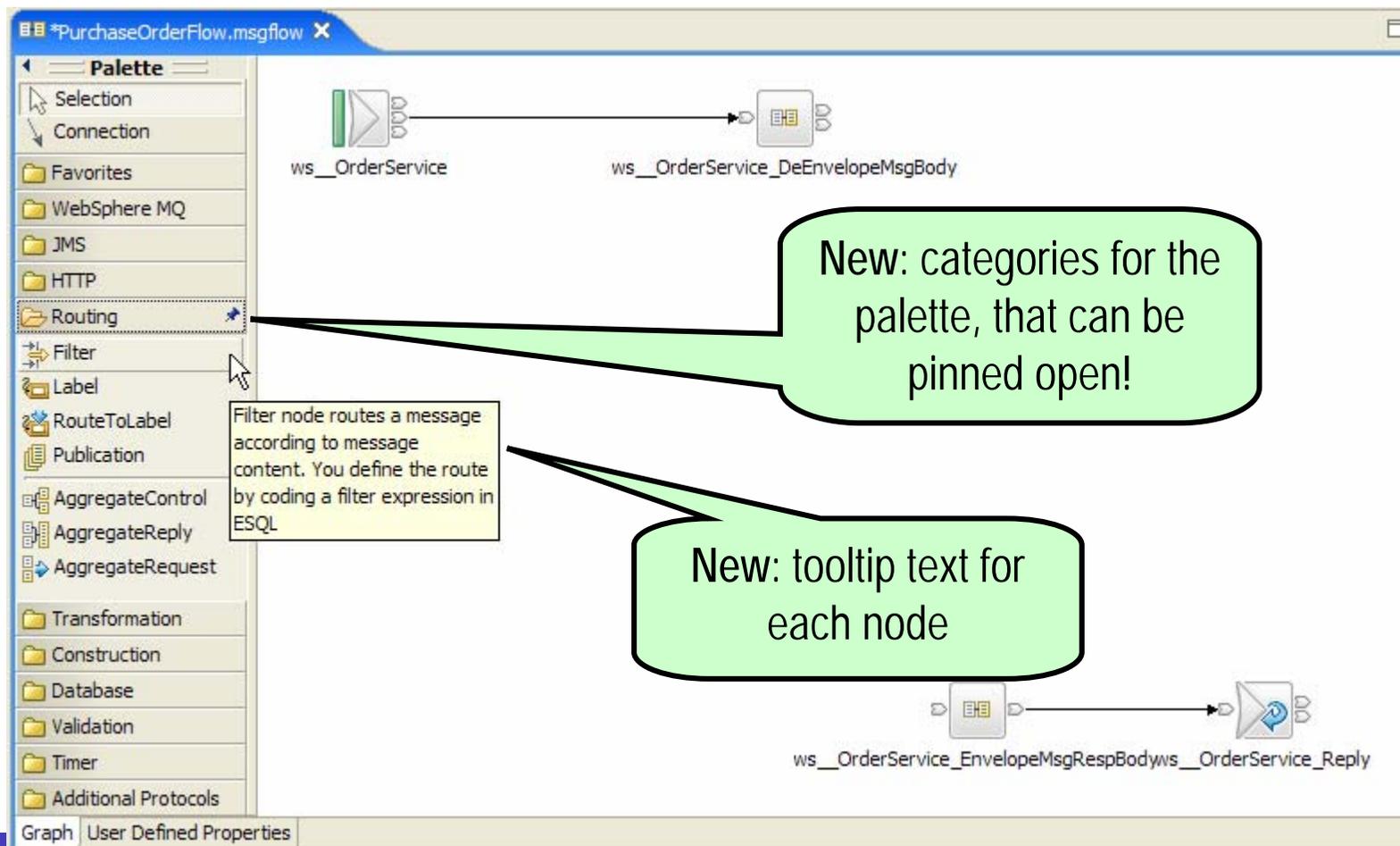
- Can also hide:

- Namespaces
- Broker schemas
- File extensions



Scenario: Step 5

- Look for a node to help us route message differently depending on inventory status



Scenario: Step 5a

- Drop a Filter node, rename it, and set expression property

Can now edit the name in place

New: can now double click to open editor

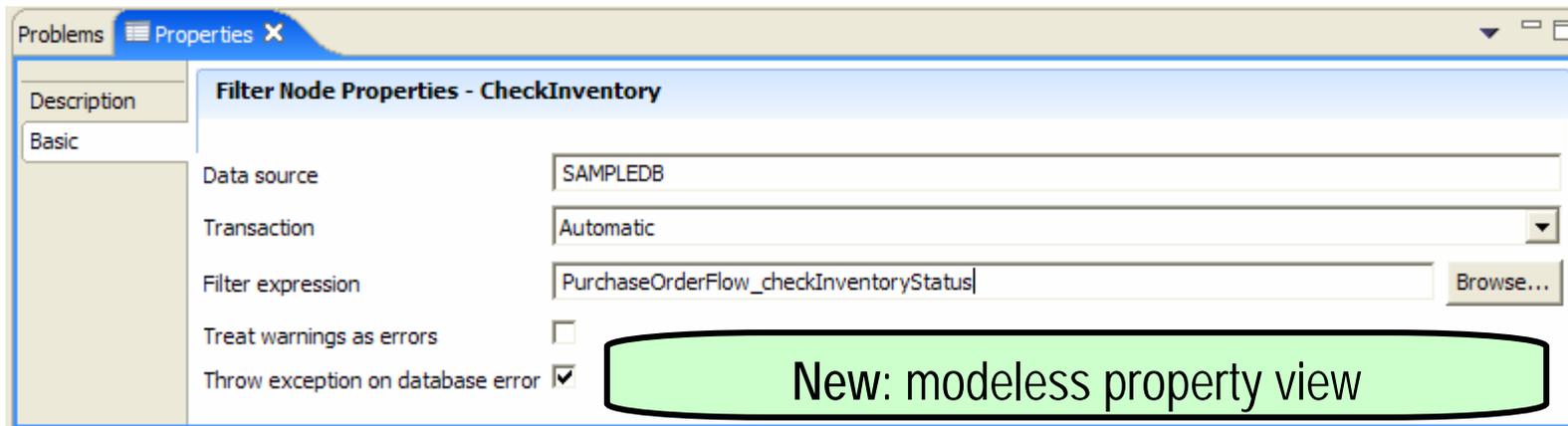
New: properties view versus modal dialogs

ws__OrderService_EnvelopeMsgRespBodyws__OrderService_Reply

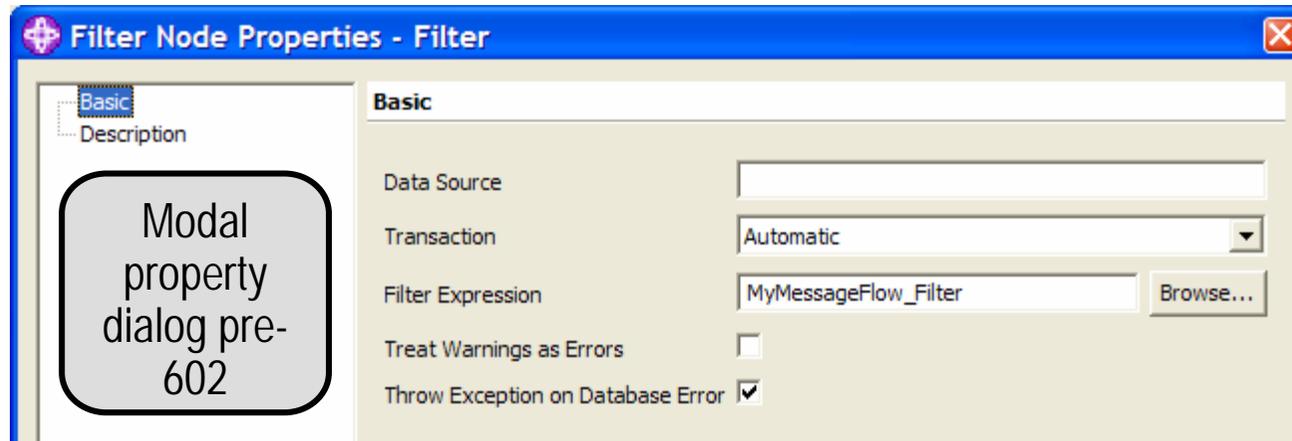
Filter Node Properties - CheckInventory

Data source	SAMPLEDB
Transaction	Automatic
Filter expression	PurchaseOrderFlow_checkInventoryStatus
Treat warnings as errors	<input type="checkbox"/>
Throw exception on database error	<input checked="" type="checkbox"/>

About properties view

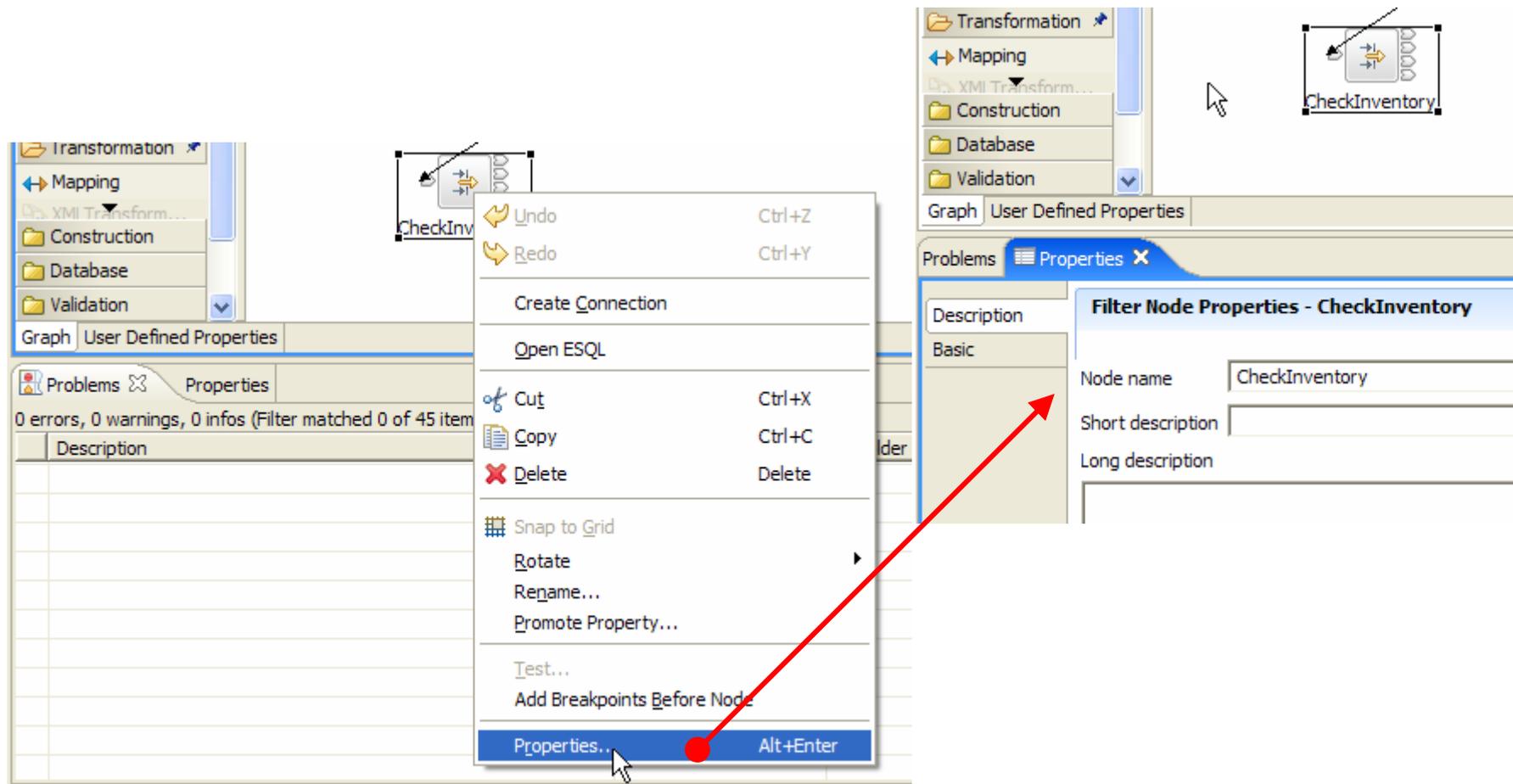


versus



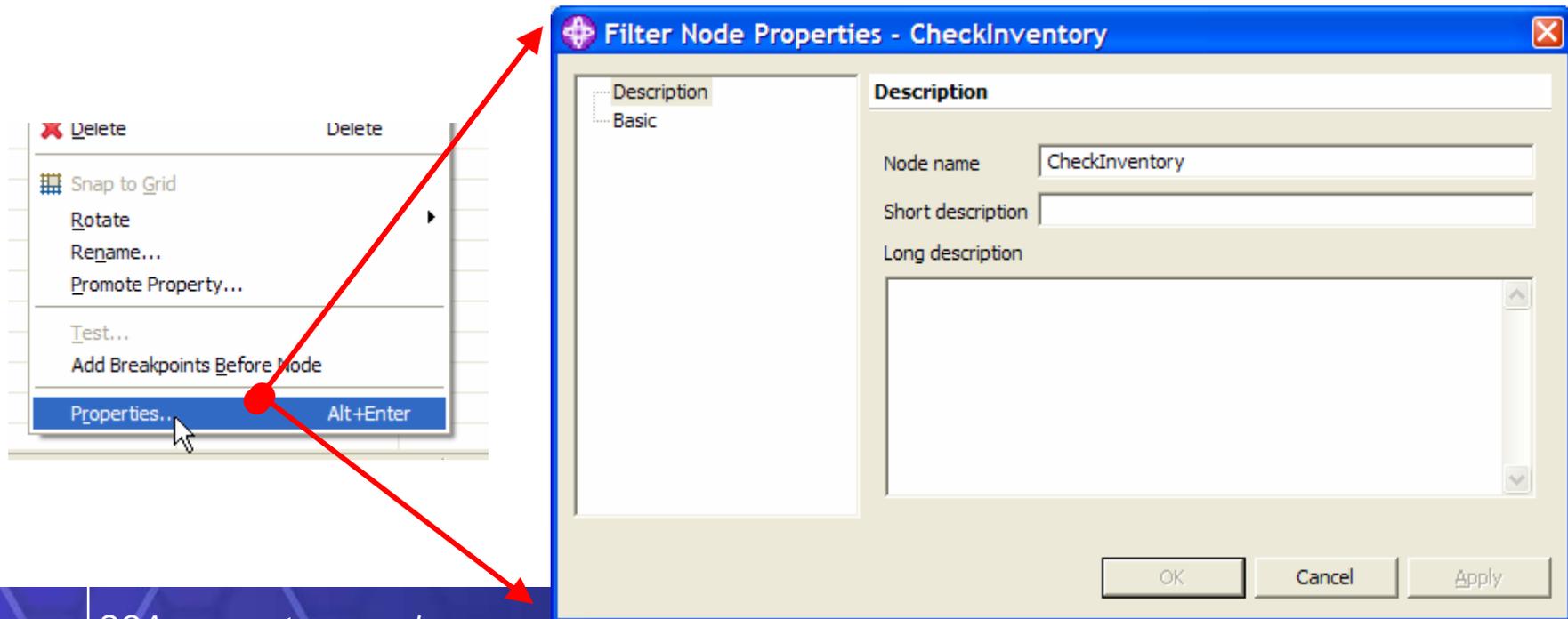
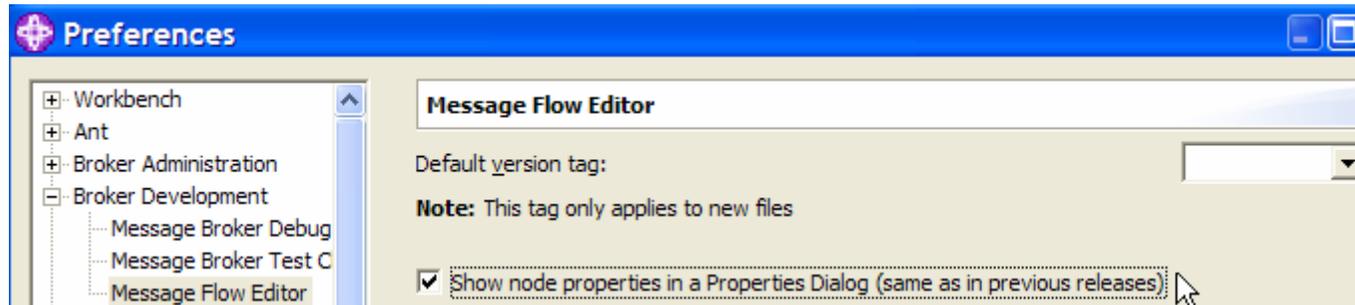
Launching properties view

- Right-click Properties action just gives Properties view focus



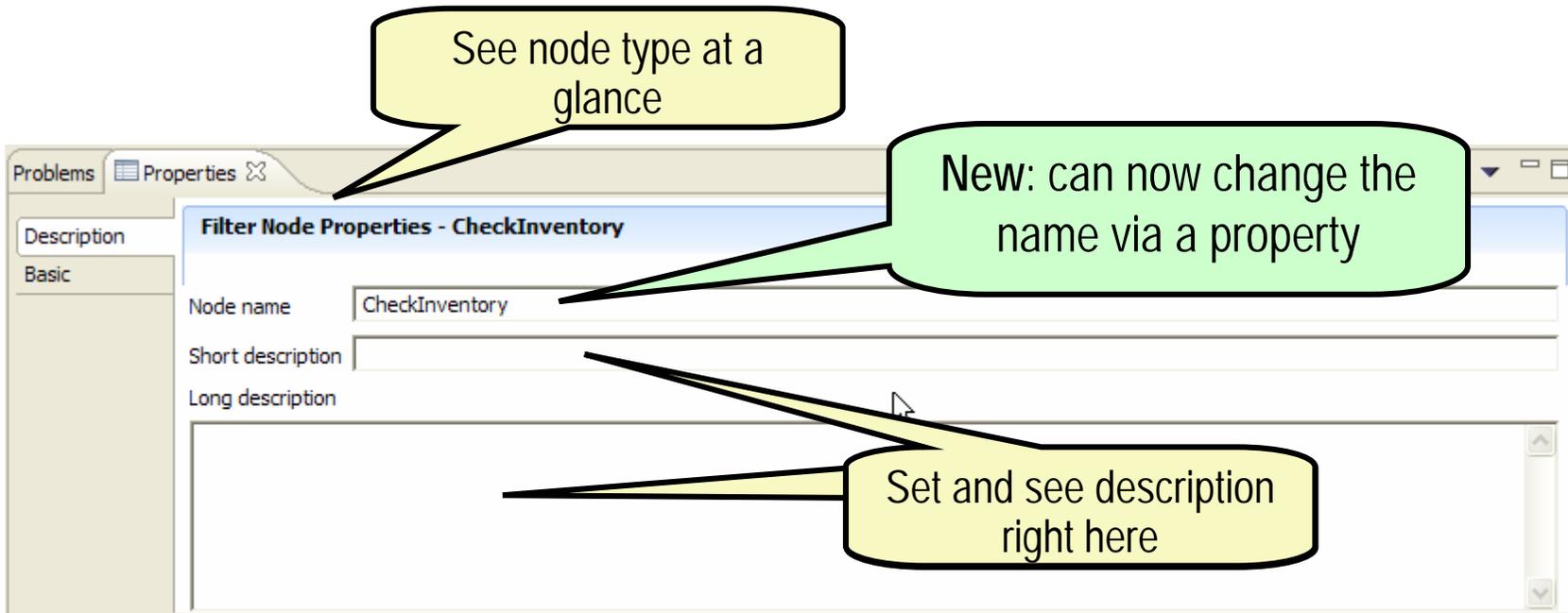
Restoring Properties Dialog

- You can change default behavior to launch Properties dialog if you prefer

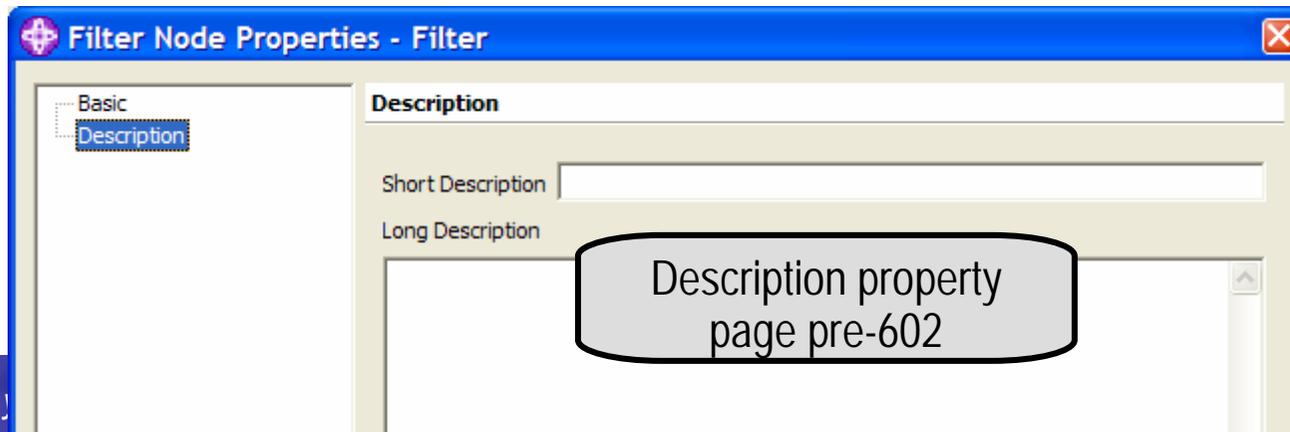


About the Description property tab

- All nodes support the same “Description” as *first* property tab

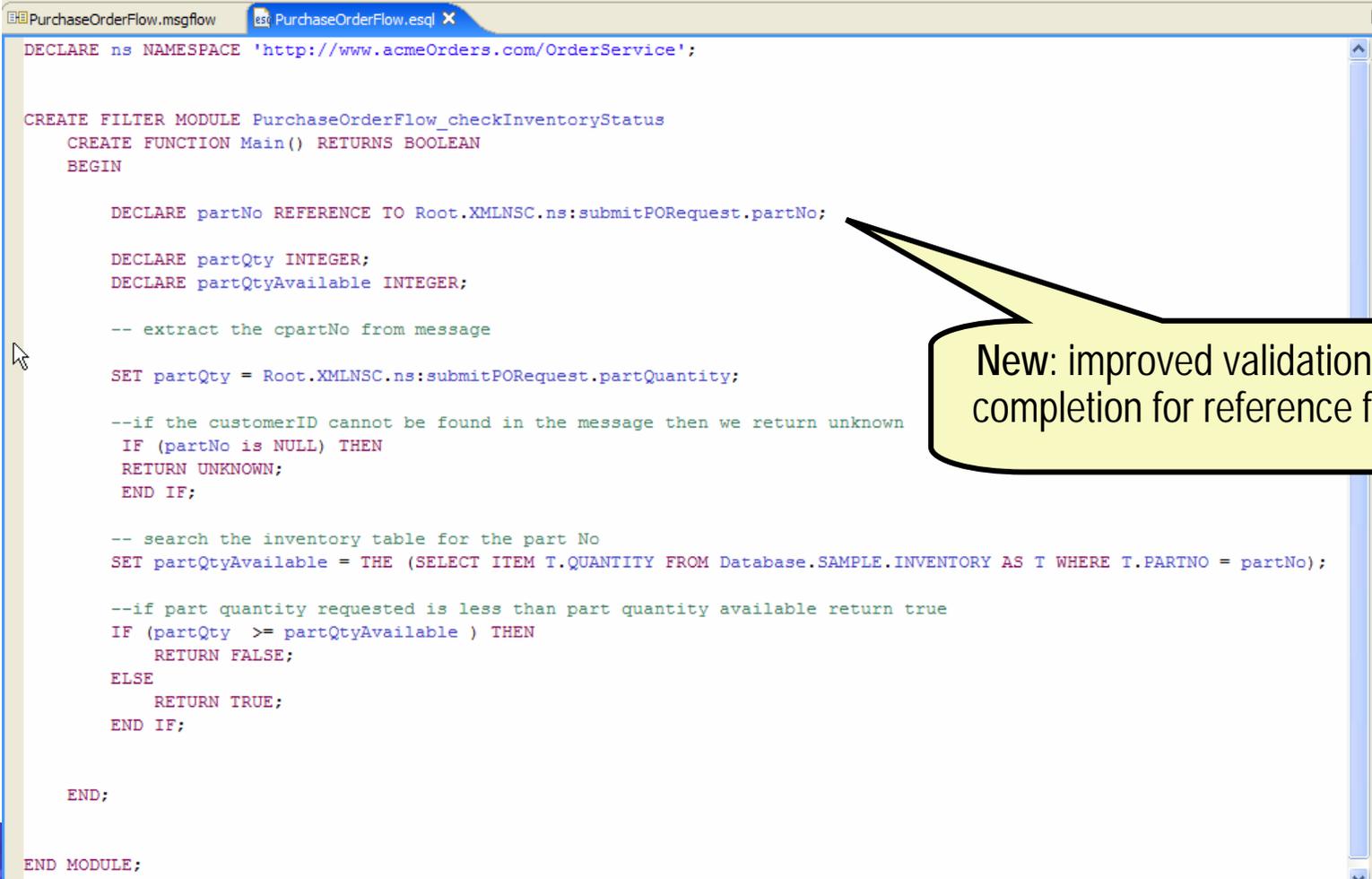


versus



Scenario: Step 5b

- Double-click on Filter node (**new!**) to open ESQL editor.
- Write ESQL logic to check inventory database (as before)



```
DECLARE ns NAMESPACE 'http://www.acmeOrders.com/OrderService';

CREATE FILTER MODULE PurchaseOrderFlow_checkInventoryStatus
  CREATE FUNCTION Main() RETURNS BOOLEAN
  BEGIN

    DECLARE partNo REFERENCE TO Root.XMLNSC.ns:submitPORequest.partNo;

    DECLARE partQty INTEGER;
    DECLARE partQtyAvailable INTEGER;

    -- extract the cpartNo from message

    SET partQty = Root.XMLNSC.ns:submitPORequest.partQuantity;

    --if the customerID cannot be found in the message then we return unknown
    IF (partNo IS NULL) THEN
      RETURN UNKNOWN;
    END IF;

    -- search the inventory table for the part No
    SET partQtyAvailable = THE (SELECT ITEM T.QUANTITY FROM Database.SAMPLE.INVENTORY AS T WHERE T.PARTNO = partNo);

    --if part quantity requested is less than part quantity available return true
    IF (partQty >= partQtyAvailable ) THEN
      RETURN FALSE;
    ELSE
      RETURN TRUE;
    END IF;

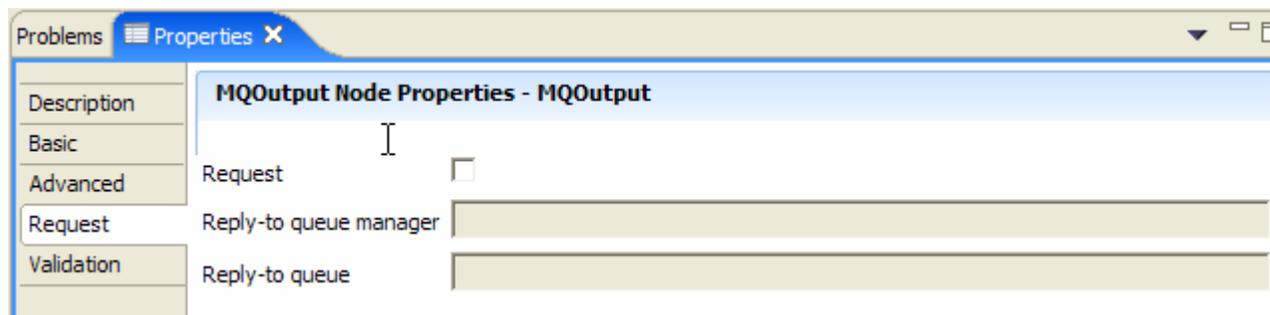
  END;

END MODULE;
```

New: improved validation and completion for reference fields

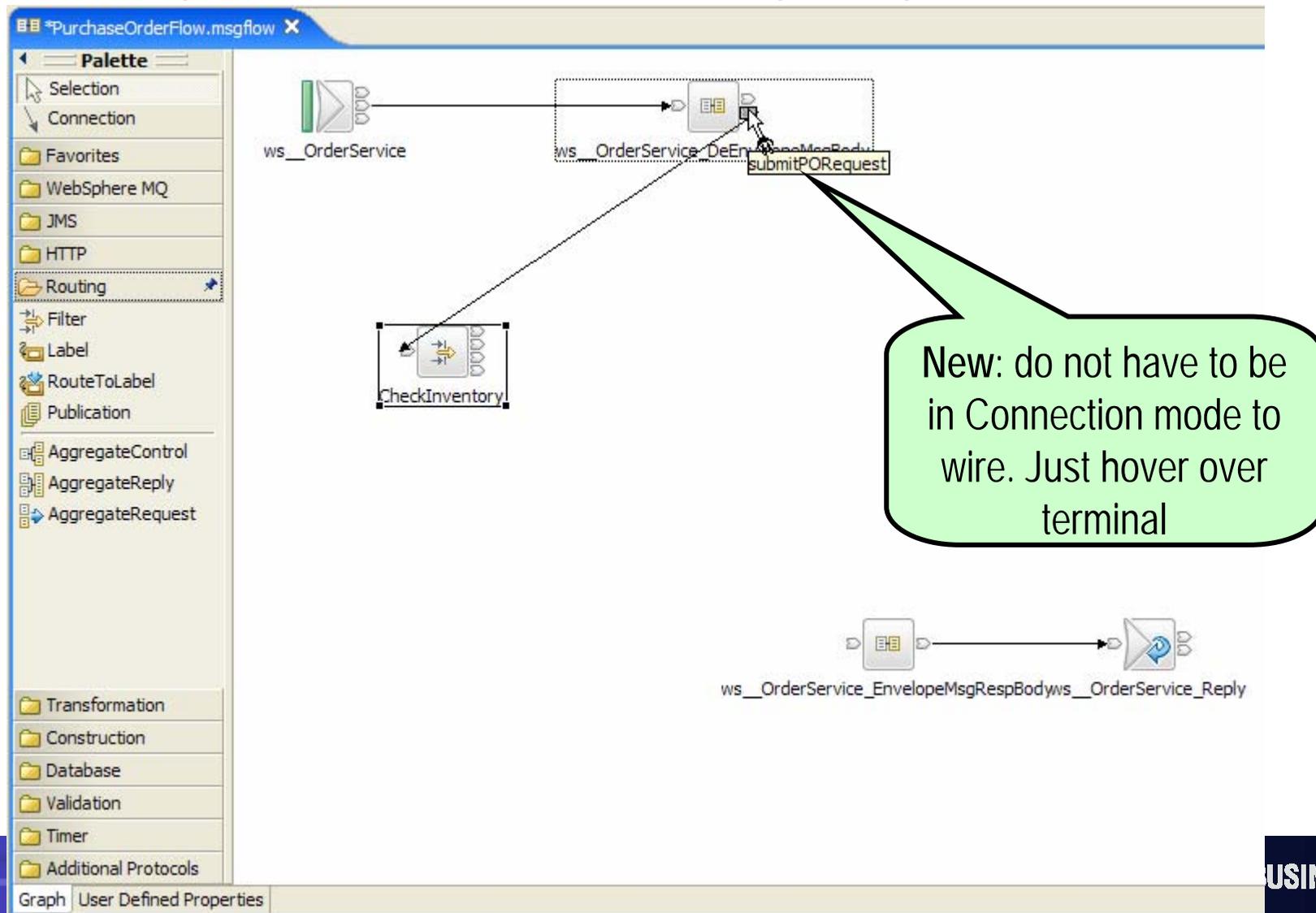
Other Properties Enhancements

- Re-organized/improved properties for following nodes:
 - HTTPInput
 - HTTPRequest
 - JMSOutput
 - XMLTransformation



Scenario: Step 5b

- Wire request subflow, for submitPORrequest operation, to filter node



Scenario: Step 6a ... Mapping Node

- Need to create a map, when out of inventory, for ordering parts. What node to use? We search Transformation category and choose first one.

The screenshot displays the IBM WebSphere Studio interface for configuring a message flow. The main window shows a message flow diagram with a 'CheckInventory' node and a 'BuildPartOrderMessage' node. A red arrow points from the 'Mapping' node in the 'Palette' to the 'BuildPartOrderMessage' node in the diagram. Another red arrow points from the 'Mapping Node Properties' dialog to the 'BuildPartOrderMessage' node in the diagram.

Palette: Shows various nodes categorized under 'Transformation' and 'Mapping'. The 'Mapping' node is selected.

Message Flow Diagram: Shows a 'CheckInventory' node with a 'False' property and a 'BuildPartOrderMessage' node. A red arrow points from the 'Mapping' node in the palette to the 'BuildPartOrderMessage' node in the diagram.

Mapping Node Properties - BuildPartOrderMessage: Shows the configuration for the 'BuildPartOrderMessage' node. The 'Mapping routine' is set to 'BuildPartOrderMessage'.

Properties Dialog:

Mapping Node Properties - BuildPartOrderMessage	
Description	
Basic	
Parser Options	
Validation	
Data source	
Transaction	Automatic
Mapping routine	BuildPartOrderMessage
Mapping mode	Message
Treat warnings as errors	<input type="checkbox"/>
Throw exception on database error	<input checked="" type="checkbox"/>

Help Text:

Use the Mapping node to construct one or more messages and populate them with new information, with modified information from the input message, or with information taken from a database

Scenario: Step 6b ... Open Map

- After setting node name and routine name property, and wiring to the Mapper node, double click on it to open mapper wizard

New: Learn how to use mapper

Incoming msg →

Outgoing msg →

By default only shows resources in currently active working set

New: this simple dialog replaces a 5-page wizard in previous releases

Tip
 The next step is to create mappings by dragging a source item and dropping it to a target. LocalEnvironment and message headers can be associated with each message.
 Do not show this tip again
 OK

OK Cancel

BUSINESS™

Scenario: Step 6c ... map the properties

- In mapping editor, select Properties on both sides, and then Map By Name

The screenshot shows the IBM WebSphere Mapping Editor interface. The left pane displays the source message set 'submitPORRequest' with its properties. The right pane displays the target message set 'orderPart' with its properties. A context menu is open over the target properties, with 'Map by Name...' selected. A 'Map By Name' dialog box is also open, showing the 'Map leaves' option selected. Two callout boxes provide instructions: one pointing to the context menu and another pointing to the dialog box.

Pre-map message set, type and format if applicable

Now an explicit "Map by name" action so more discoverable

Map By Name

Select how to map from source to target

Choose from the following options to map from selected source to selected target. Verify the generated mappings after the action completes.

Map leaves

Map immediate children

Map items of same names only

Map items of same and similar names

Finish Cancel

Undo	Ctrl+Z
Redo	Ctrl+Y
Revert	
Open Declaration	F3
Add Targets...	
Go To	
Add/Remove Message Headers...	
Change RDB Operation...	
Delete	Delete
Map from Source	Alt+C, M
Map by Name...	Alt+C, N
Enter Expression	Alt+C, T
Accumulate	Alt+C, L
Create New Submap...	Alt+C, W
Create New RDB Submap...	Alt+C, D
Call Existing Submap...	Alt+C, X
Call ESQL Routine...	Alt+C, Q
Save	Ctrl+S

Scenario: Step 6c ... mapped properties

The screenshot displays the IBM WebSphere Message Designer interface. It shows two message maps: `*PurchaseOrderFlow.msgflow` and `*BuildPartOrderMessage.msgmap`. The source message is `$source - Message "submitPORrequest" and Headers` with properties: `MessageSet (xsd:string)`, `MessageType (xsd:string)`, `MessageFormat (xsd:string)`, `Encoding (xsd:int)`, `CodedCharSetId (xsd:int)`, `Transactional (xsd:boolean)`, `Persistence (xsd:boolean)`, `CreationTime (xsd:time)`, `ExpirationTime (xsd:time)`, `Priority (xsd:int)`, `ReplyIdentifier (xsd:string)`, `ReplyProtocol (xsd:string)`, `Topic (xsd:string)`, and `ContentType (xsd:string)`. The target message is `$target - Message "orderPart" and Headers` with identical properties. The `tns1:orderPart (tns1:OrderPartType)` target is selected.

Below the message maps is the **Map Script** table, which defines the mapping for the `BuildPartOrderMessage` map:

Map Script	Value
BuildPartOrderMessage	
Parameters	
\$target	
Properties	
MessageSet	"C6UJ45C002001"
MessageType	"orderPart"
MessageFormat	<code>\$source/Properties/MessageFormat</code>
Encoding	<code>\$source/Properties/Encoding</code>
CodedCharSetId	<code>\$source/Properties/CodedCharSetId</code>
Transactional	<code>\$source/Properties/Transactional</code>
Persistence	<code>\$source/Properties/Persistence</code>
CreationTime	<code>\$source/Properties/CreationTime</code>
ExpirationTime	<code>\$source/Properties/ExpirationTime</code>
Priority	<code>\$source/Properties/Priority</code>
ReplyIdentifier	<code>\$source/Properties/ReplyIdentifier</code>
ReplyProtocol	<code>\$source/Properties/ReplyProtocol</code>
Topic	<code>\$source/Properties/Topic</code>
ContentType	<code>\$source/Properties/ContentType</code>
tns1:orderPart	

Scenario: Step 6d ... map the fields

- In mapping editor, drag and drop `partNo` and `partQuantity`

The screenshot shows the Mapping Editor interface with two message trees:

- Source Message:** `$source - Message "submitPORequest" and Headers`
 - Properties (PropertiesType_submitPORequest)
 - `tns2:submitPORequest (anonymous)`
 - `partNo (xsd:string)`
 - `partQuantity (xsd:int)`
 - `personName (anonymous)`
 - `address (anonymous)`

- Target Message:** `$target - Message "orderPart" and Headers`
- Properties (PropertiesType_orderPart)
 - `tns1:orderPart (tns1:OrderPartType)`
 - `partNo (xsd:string)`
 - `opartQuantity (xsd:string)`
 - `orderDate (xsd:string)`
 - `orderTime (xsd:string)`

The mapping table below shows the mapping of the source field `partQuantity` to the target field `opartQuantity`.

Map Script	Value
BuildPartOrderMessage	
Parameters	
\$target	
Properties	
tns1:orderPart	
partNo	<code>\$source/tns2:submitPORequest/partNo</code>
opartQuantity	<code>\$source/tns2:submitPORequest/partQuantity</code>

Scenario: Step 6e ... insert orderDate & orderTime

The screenshot shows the 'Insert Statement' dialog box in the IBM WebSphere IDE. The dialog title is 'Insert Statement' and it contains the instruction: 'Select an item valid in tns1:orderPart to be inserted after opartQuantity'. Two items are listed in the selection area: 'orderDate' and 'orderTime'. A yellow callout bubble points to this list with the text: 'Only see fields that are legal at this point and not already mapped'. Below the list, there is a warning: 'The total number of instances existing and to be added must not exceed the maximum occurrence specified for the selected element.' Two input fields are present: 'Maximum total number of instances:' with the value '1' and 'Number of instances to be added:' with the value '1'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons. In the background, the 'Map Script' window shows a project tree where 'opartQuantity' is selected. A context menu is open over 'opartQuantity', and the 'Insert After...' option is highlighted with a mouse cursor. Another yellow callout bubble at the bottom right of the IDE window contains the text: 'Repeat to insert OrderTime after OrderDate'.

Scenario: Step 6e ... set orderDate and orderTime to esql:current-date() and esql:current-time()

The screenshot shows the Message Designer interface with two message maps: `*PurchaseOrderFlow.msgflow` and `*BuildPartOrderMessage.msgmap`. The source message structure is as follows:

- `$source - Message "submitPORrequest" and Headers`
 - Properties (PropertiesType_submitPORrequest)
 - `tns2:submitPORrequest (anonymous)`
 - `partNo (xsd:string)`
 - `partQuantity (xsd:int)`
 - `personName (anonymous)`
 - `address (anonymous)`

The target message structure is:

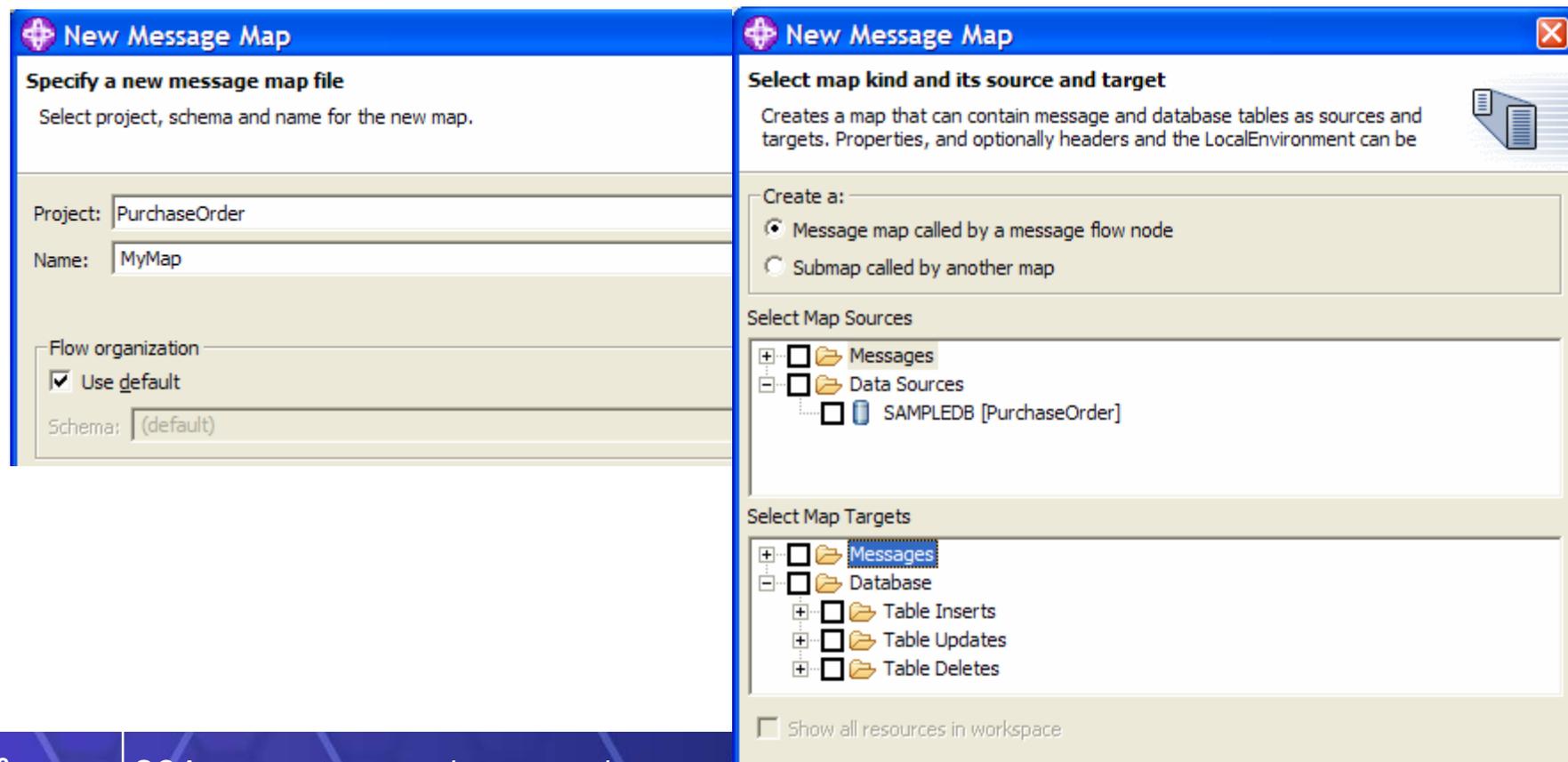
- `$target - Message "orderPart" and Headers`
 - Properties (PropertiesType_orderPart)
 - `tns1:orderPart (tns1:OrderPartType)`
 - `partNo (xsd:string)`
 - `opartQuantity (xsd:string)`
 - `orderDate (xsd:string)`
 - `orderTime (xsd:string)`

The `esql:cu` (ESQL Callout) configuration is shown below:

Map Script	Value
<code>esql:current-date</code>	
<code>esql:current-gmtdate</code>	
<code>esql:current-gmtime</code>	
<code>esql:current-gmttimestamp</code>	
<code>esql:current-time</code>	
<code>esql:current-timestamp</code>	
Properties	
<code>tns1:orderPart</code>	
<code>partNo</code>	<code>\$source/tns2:submitPORrequest/partNo</code>
<code>opartQuantity</code>	<code>\$source/tns2:submitPORrequest/partQuantity</code>
<code>orderDate</code>	<code>esql:current-date()</code>
<code>orderTime</code>	<code>esql:cu</code>

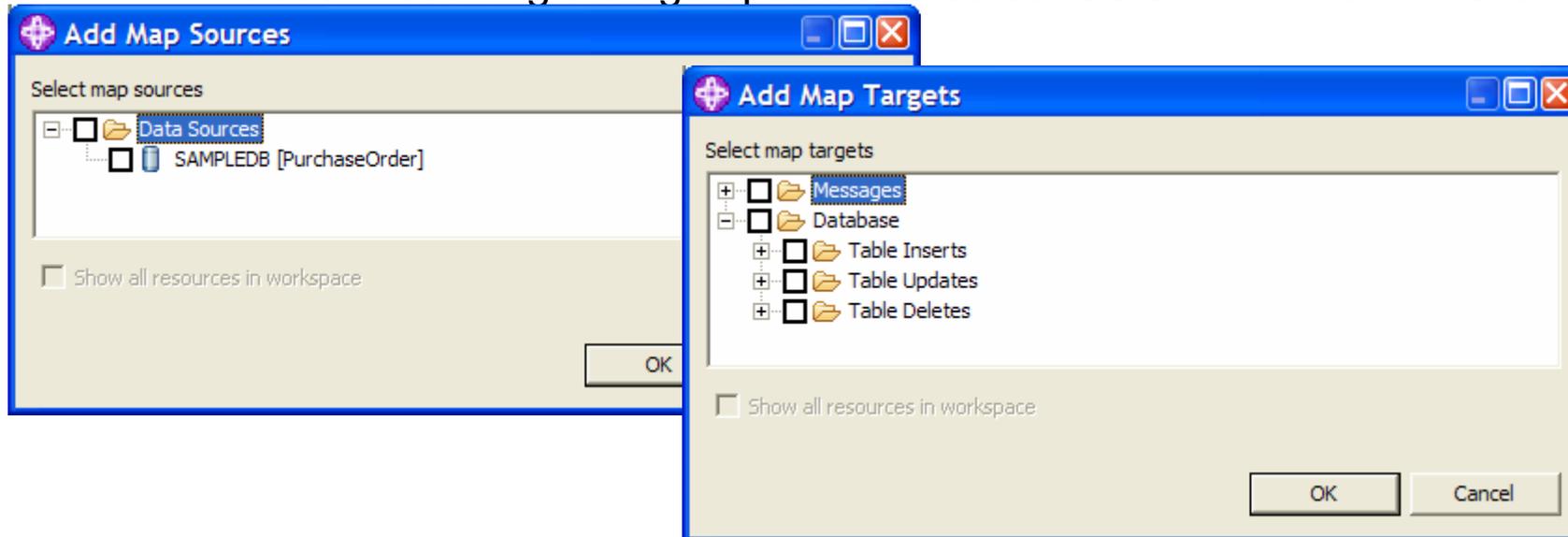
Additional Mapper Enhancements 1 of 6

- There are a number of enhancements to the mapper:
 - 1. Vastly simplified wizard, as we have seen when opened from the Flow editor. Here are the wizard pages when opened from File->New:



Additional Mapper Enhancements 2 of 6

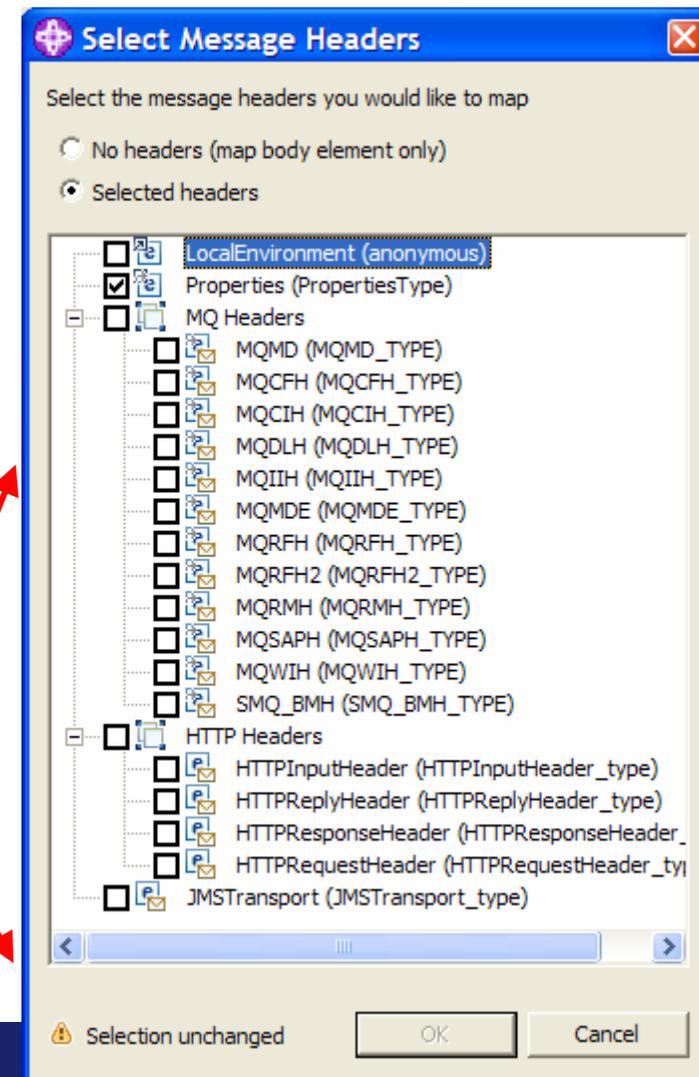
- Additional enhancements to the mapper:
 - 2. Changed “Add Sources and Targets” to become “Add Sources...” and “Add Targets...” right click actions on \$source and \$target roots in trees. Resulting dialogs updated to be consistent with new wizard:



- 3. You can also drag and drop roots from the nav to add src/tgt
 - Eg: message or database table or global element (for submaps)

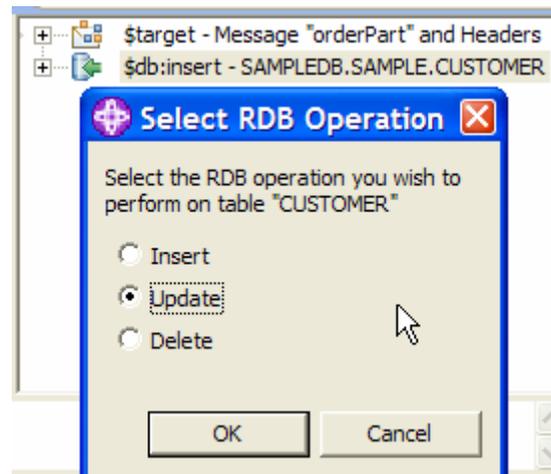
Additional Mapper Enhancements 3 of 6

- Additional enhancements to the mapper:
 - 4. In dealing with headers by default:
 - The Properties folder is shown in the mapper to enable explicit mapping
 - Any headers are automatically copied from input message by default
 - The Local Environment is not mapped by default
 - 5. But within the Map editor the user can:
 - Right click on Source or Target and select “Add/Remove Message Headers...” resulting in this:



Additional Mapper Enhancements 4 of 6

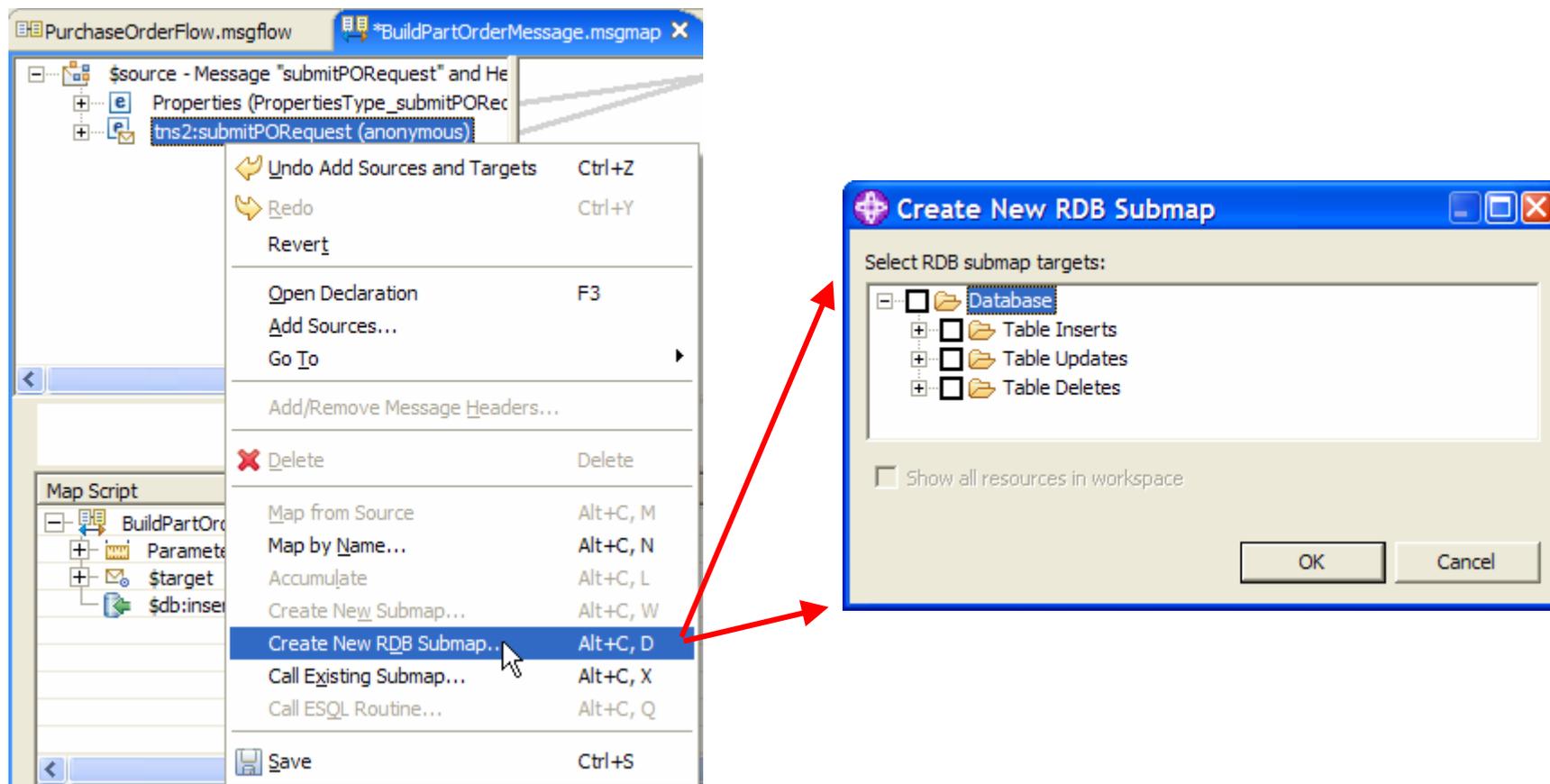
- Additional enhancements to the mapper:
 - 6. New “*Change RDB operation...*” right click action:



- Further,
 - 7. You can now map from database to database. Previously, you could only map from database to message or message to database.

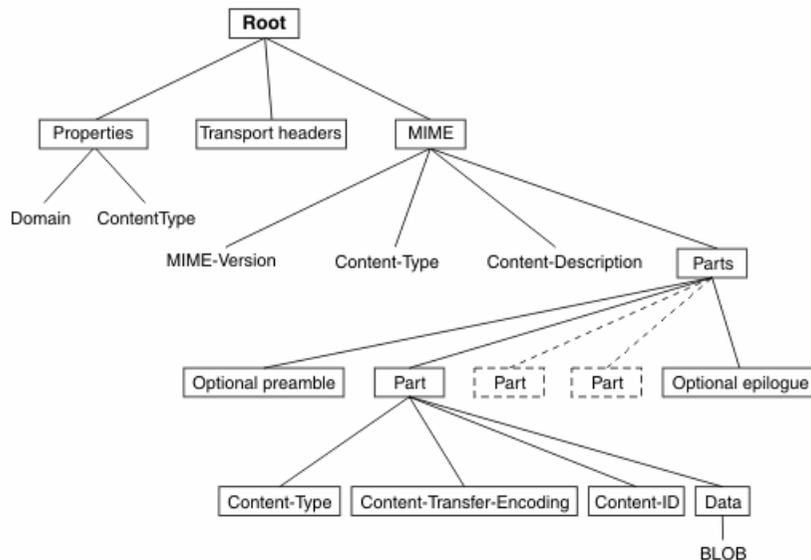
Additional Mapper Enhancements 5 of 6

- Additional enhancements to the mapper:
 - 8. New “*Create RDB Submap...*” right click action (moved from wizard):



Additional Mapper Enhancements 6 of 6

- Additional enhancements to the mapper:
 - 9. You can now use the mapper to map messages in a MIME domain
 - Domains supported: MRM, MIME, XML, XMLNS, XMLNSC, JMSMap, JMSStream



Map Script	Value
MIME_Msg	
MIME-Version	<code>\$source/MIME_Msg/MIME-Version</code>
Content-Type	<code>\$source/MIME_Msg/Content-Type</code>
Content-Transfer-Encoding	<code>\$source/MIME_Msg/Content-Transfer-Encoding</code>
Content-ID	<code>\$source/MIME_Msg/Content-ID</code>
Parts	
Part	
Content-Type	<code>'text/xml; charset=UTF-8'</code>
Content-Transfer-Encoding	<code>'8bit'</code>
Content-ID	<code>'<part1@example.com>'</code>
Data	
for	<code>\$source/MIME_Msg/Parts/Part</code>
if	
condition	<code>msgmap:occurrence(\$source/MIME_Msg/Parts/Part) = 1</code>
BLOB	<code>\$source/MIME_Msg/Parts/Part/Data/BLOB</code>
Part	
Content-Type	<code>'application/octet-stream'</code>
Content-Transfer-Encoding	<code>'binary'</code>
Content-ID	<code>'<part2@example.com>'</code>
Data	
BLOB	<code>xs:hexBinary(1234567890)</code>

Scenario: Step 7a ... Compute node

- In flow editor, drop a Compute node, wire to it, and set its name and ESQL Module property to `InsertMQMDHeader`. Press `Ctrl+S` to save.

The screenshot displays the IBM WebSphere MessageFlow Editor interface. The main workspace shows a message flow diagram with nodes: `ws__OrderService`, `ws__OrderService_DeEnvelopeMsgBody`, `CheckInventory`, `BuildPartOrderMessage`, and `InsertMQMDHeader`. The `InsertMQMDHeader` node is highlighted with a red box. A green callout bubble points to it with the text: "New: can now double click to open editor".

The bottom panel shows the **Compute Node Properties - InsertMQMDHeader** dialog. The `ESQL module` property is set to `InsertMQMDHeader` and is highlighted with a red box. Other properties include:

- `Data source`: (empty)
- `Transaction`: Automatic
- `ESQL module`: InsertMQMDHeader
- `Compute mode`: Message
- `Treat warnings as errors`:
- `Throw exception on database error`:

Scenario: Step 7b ... Compute node

- Double click (new!) the InsertMQMDHeader node to enter ESQL editor
- Change the default generated Main() function as shown below....

```
PurchaseOrderFlow.msgflow PurchaseOrderFlow.esql X
CREATE COMPUTE MODULE InsertMQMDHeader
  CREATE FUNCTION Main() RETURNS BOOLEAN
  BEGIN
    -- CALL CopyMessageHeaders();
    CALL CopyEntireMessage();

    -- Output is MQ, so remove HTTP headers
    SET OutputRoot.HTTPInputHeader = null;

    -- Add an MQMD
    CREATE NEXTSIBLING OF OutputRoot.Properties DOMAIN 'MQMD';

    RETURN TRUE;
  END;

  CREATE PROCEDURE CopyMessageHeaders() BEGIN
    DECLARE I INTEGER 1;
    DECLARE J INTEGER;
    SET J = CARDINALITY(InputRoot.*[]);
    WHILE I < J DO
      SET OutputRoot.*[I] = InputRoot.*[I];
      SET I = I + 1;
    END WHILE;
  END;

  CREATE PROCEDURE CopyEntireMessage() BEGIN
    SET OutputRoot = InputRoot;
  END;
END MODULE;
```

Scenario: Step 7c ... MQOutput node

- In flow editor, drop MQOutput node, set its name and Queue name property as shown, and wire to it from the out terminal of the compute node

The screenshot displays the IBM WebSphere flow editor interface. The main workspace shows a message flow diagram with the following nodes and connections:

- ws_OrderService** (Input) connects to **ws_OrderService_DeEnvelopeMsgBody** (Compute).
- ws_OrderService_DeEnvelopeMsgBody** connects to **BuildPartOrderMessage** (Compute).
- BuildPartOrderMessage** connects to **InsertMQMDHeader** (Compute).
- InsertMQMDHeader** connects to **SUPPLIER.ORDER.Q00** (MQOutput).
- CheckInventory** (Compute) is also connected to **BuildPartOrderMessage**.

The **MQOutput Node Properties - SUPPLIER.ORDER.Q00** dialog is open, showing the following properties:

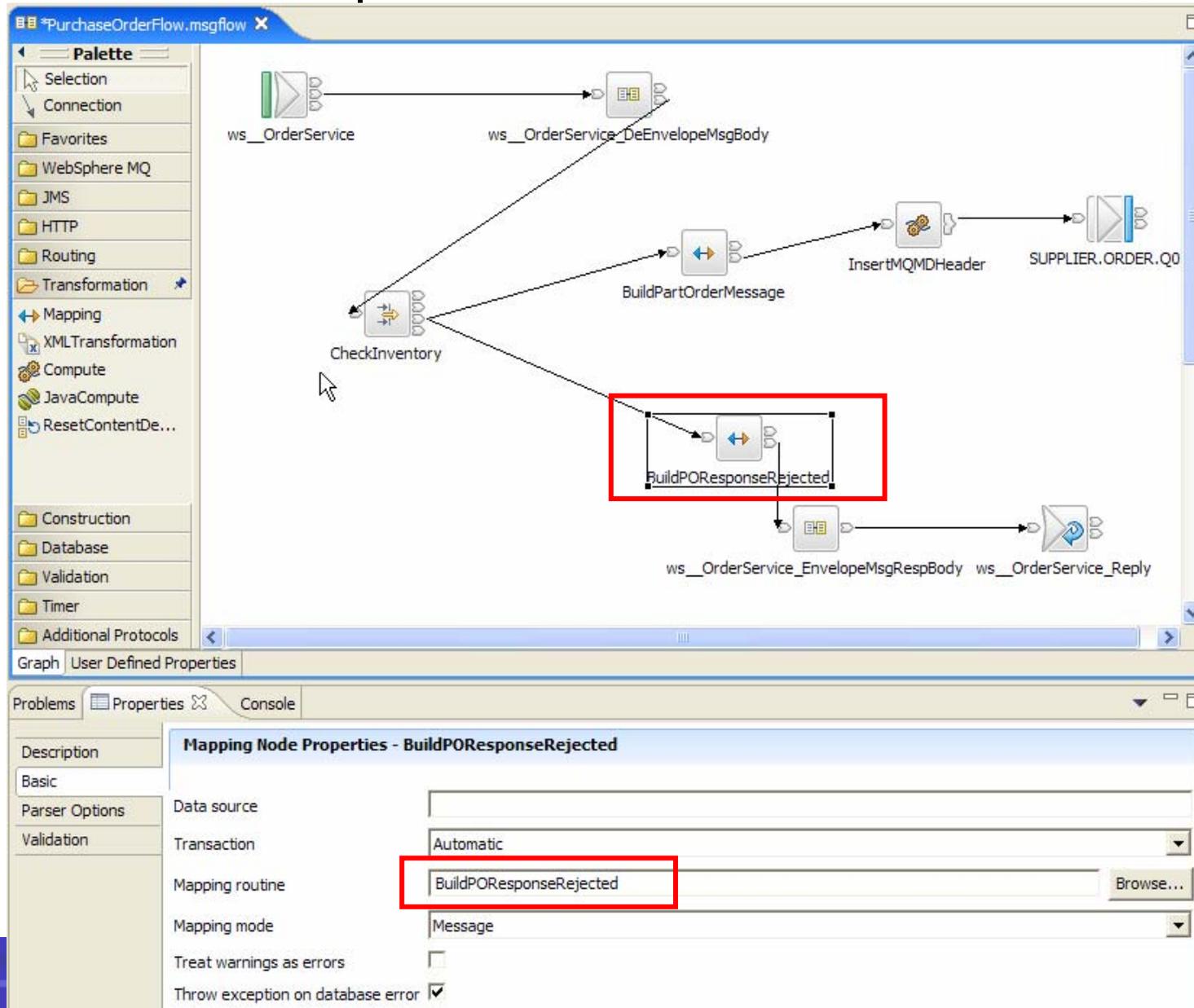
MQOutput Node Properties - SUPPLIER.ORDER.Q00	
Description	
Basic	
Advanced	Destination mode: Queue Name
Request	Transaction mode: Automatic
Validation	Persistence mode: Automatic
	New message ID: <input type="checkbox"/>
	New correlation ID: <input type="checkbox"/>
	Segmentation allowed: <input type="checkbox"/>
	Message context: Default
	Alternate user authority: <input type="checkbox"/>

The **Queue name** property in the **Advanced** section is highlighted with a red box and contains the value **NETCOMP.SUPPLIER.ORDER.Q00**.

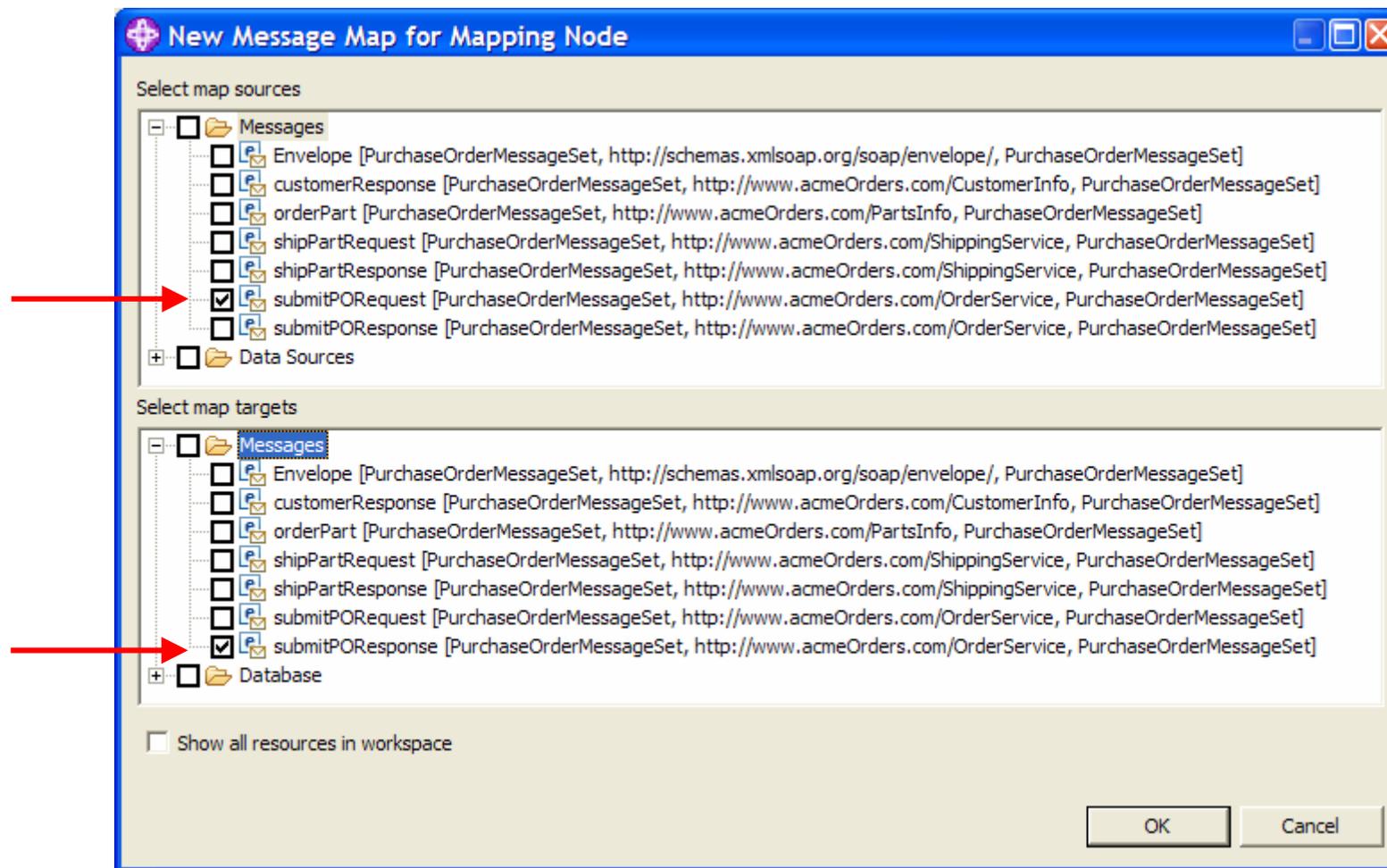
Scenario: Step 8 - Send Response When No Inventory

- Here we will again wire from the “false” terminal of the checkInventory filter node, this time to return an HTTP reply to the caller of this web service
 - To a map node to build the response message
 - From there to the generated reply nodes for this operation
- **8a:** Drag and drop a map node and wire it. Also set its node name and mapping routine property to `BuildPOResponseRejected`.

Scenario: Step 8a – Create and Wire Mapping node



Scenario: Step 8b – Open Map



Scenario: Step 8c – Do Mapping

- Map Properties by name
- Map partNo and partQuantity by drag and drop
- Set orderStatus to 'Rejected' (use right click "Enter Expression")
- Set orderAmt to 0 (use right click "Enter Expression")

Map Script	Value
BuildPORResponseRejected	
Parameters	
\$target	
Properties	
tns1:submitPORResponse	
orderStatus	'Rejected'
orderAmt	0
partNo	<code>\$source/tns1:submitPORRequest/partNo</code>
partQuantity	<code>\$source/tns1:submitPORRequest/partQuantity</code>

Scenario: Step 9 – Process “sufficient inventory” situation

- Here we will now wire from the “true” terminal of the checkInventory filter node:
 - To a map node to build the ship request message, which we later use as input to the shipping web service

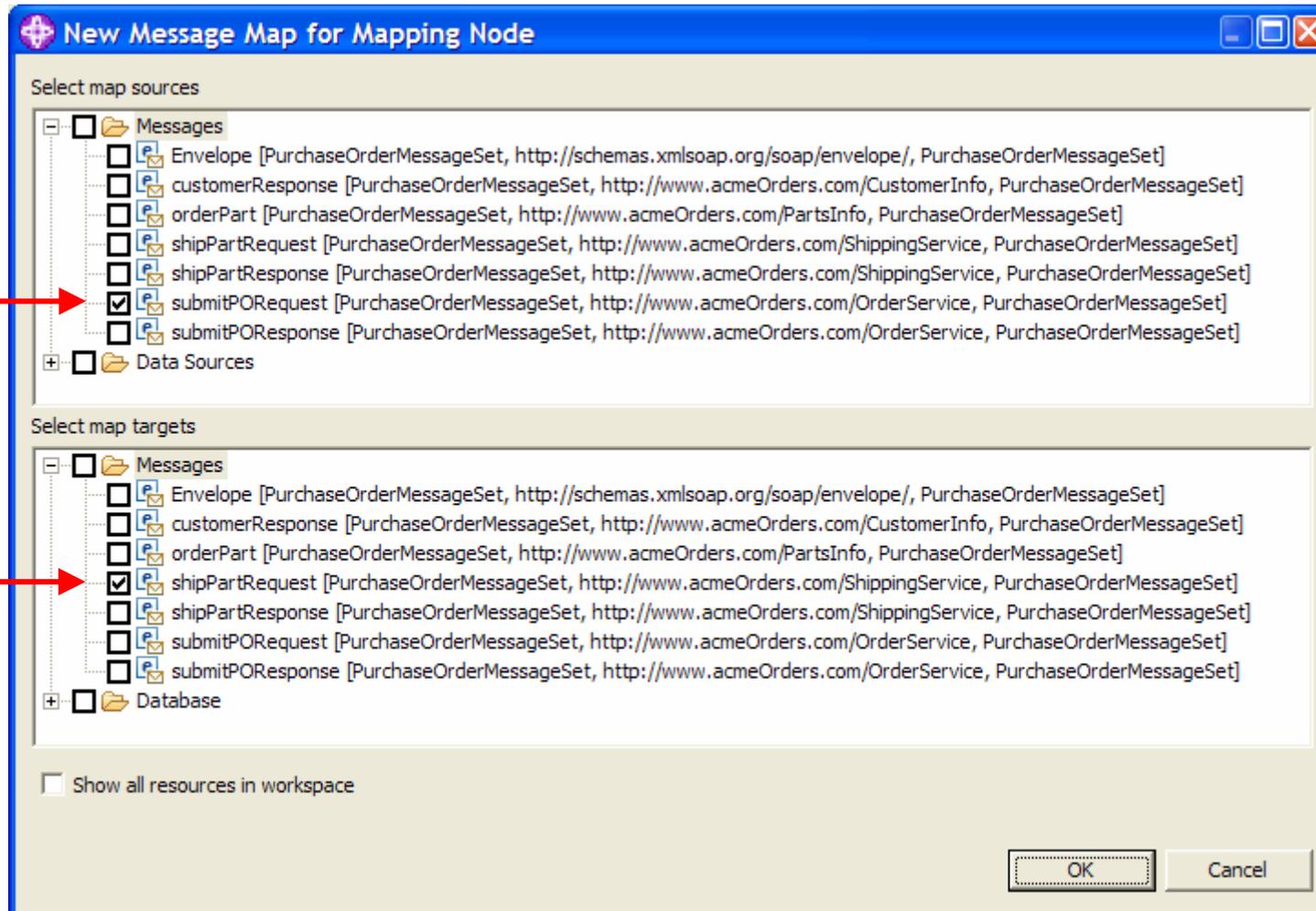
- 9a: Drag and drop a map node and wire to it. Also set its name and mapping routine property to `BuildShipPartRequest`

Scenario: Step 9a – Create, Wire BuildShipPart Map node

The screenshot displays the IBM WebSphere MessageFlow Builder interface. The main window shows a message flow diagram for a file named *PurchaseOrderFlow.msgflow. The flow starts with an input node 'ws__OrderService' connected to 'ws__OrderService_DeEnvelopeMsgBody'. This node branches into three paths: one to 'CheckInventory', one to 'BuildPartOrderMessage', and one to 'BuildPOResponseRejected'. The 'BuildPartOrderMessage' node is connected to 'InsertMQMDHeader', which then connects to 'SUPPLIER.ORDER.Q0'. The 'BuildPOResponseRejected' node connects to 'ws__OrderService_EnvelopeMsgRespBody', which finally connects to 'ws__OrderService_Reply'. A 'BuildShipPartRequest' mapping node is highlighted with a red box in the diagram. Below the diagram, the 'Properties' tab is active, showing the 'Mapping Node Properties - BuildShipPartRequest' configuration. The 'Mapping routine' field is set to 'BuildShipPartRequest' and is also highlighted with a red box. Other properties include 'Data source' (empty), 'Transaction' (Automatic), 'Mapping mode' (Message), 'Treat warnings as errors' (unchecked), and 'Throw exception on database error' (checked).

Property	Value
Description	Basic
Parser Options	
Validation	
Data source	
Transaction	Automatic
Mapping routine	BuildShipPartRequest
Mapping mode	Message
Treat warnings as errors	<input type="checkbox"/>
Throw exception on database error	<input checked="" type="checkbox"/>

Scenario: Step 9b – Open Map



Scenario: Step 9c – Do Mapping

- Select \$source and \$target roots
- Right click on \$target and select Map By Name. Press Enter. Done!

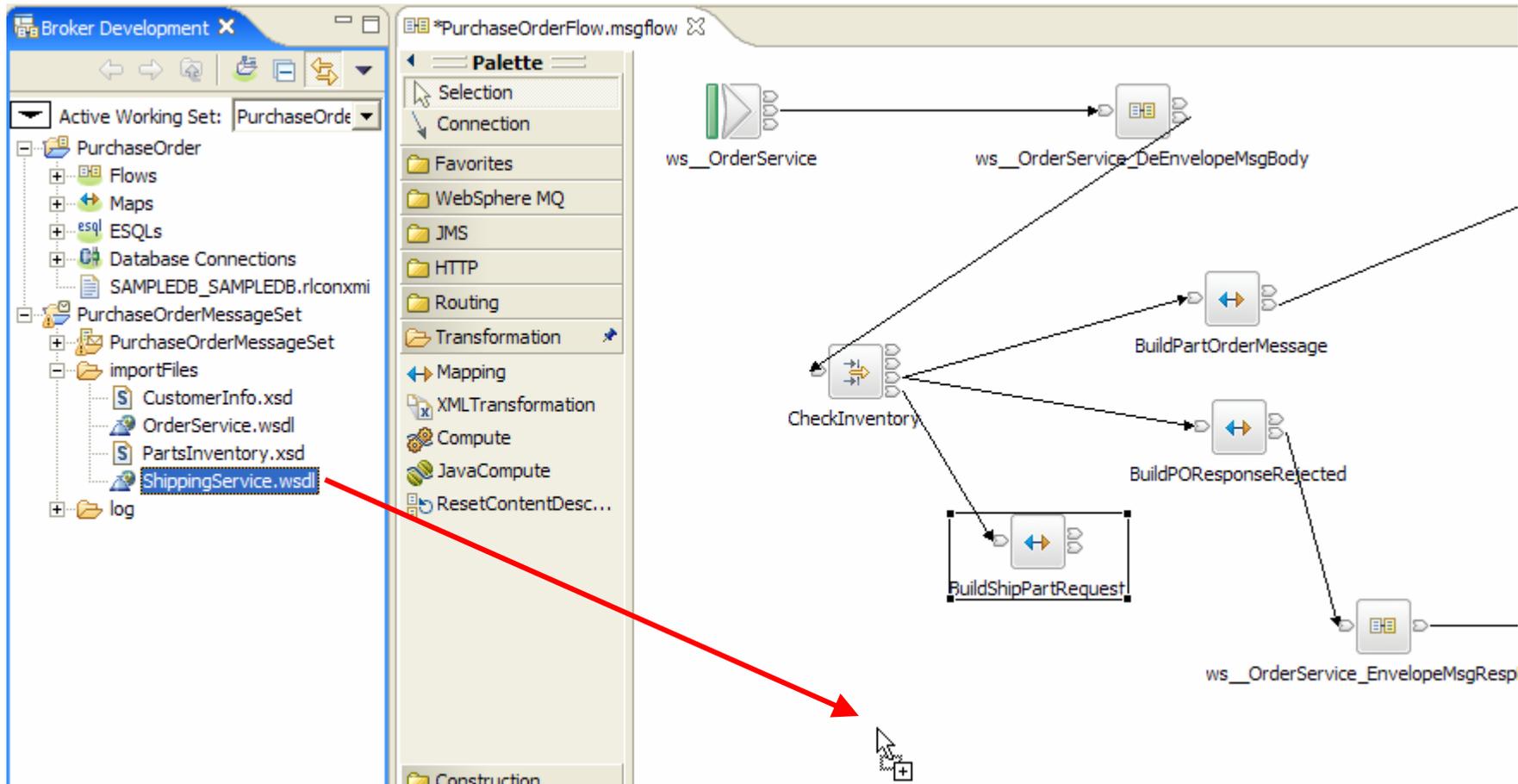
Map Script	Value
BuildShipPartRequest	
Parameters	
\$target	
Properties	
tns1:shipPartRequest	
partNo	<code>\$source/tns2:submitPORrequest/partNo</code>
partQuantity	<code>\$source/tns2:submitPORrequest/partQuantity</code>
personName	
firstName	<code>\$source/tns2:submitPORrequest/personName/firstName</code>
lastName	<code>\$source/tns2:submitPORrequest/personName/lastName</code>
address	
street	<code>\$source/tns2:submitPORrequest/address/street</code>
city	<code>\$source/tns2:submitPORrequest/address/city</code>
zipCode	<code>\$source/tns2:submitPORrequest/address/zipCode</code>

Scenario: Step 10 – Call Shipping Web Service

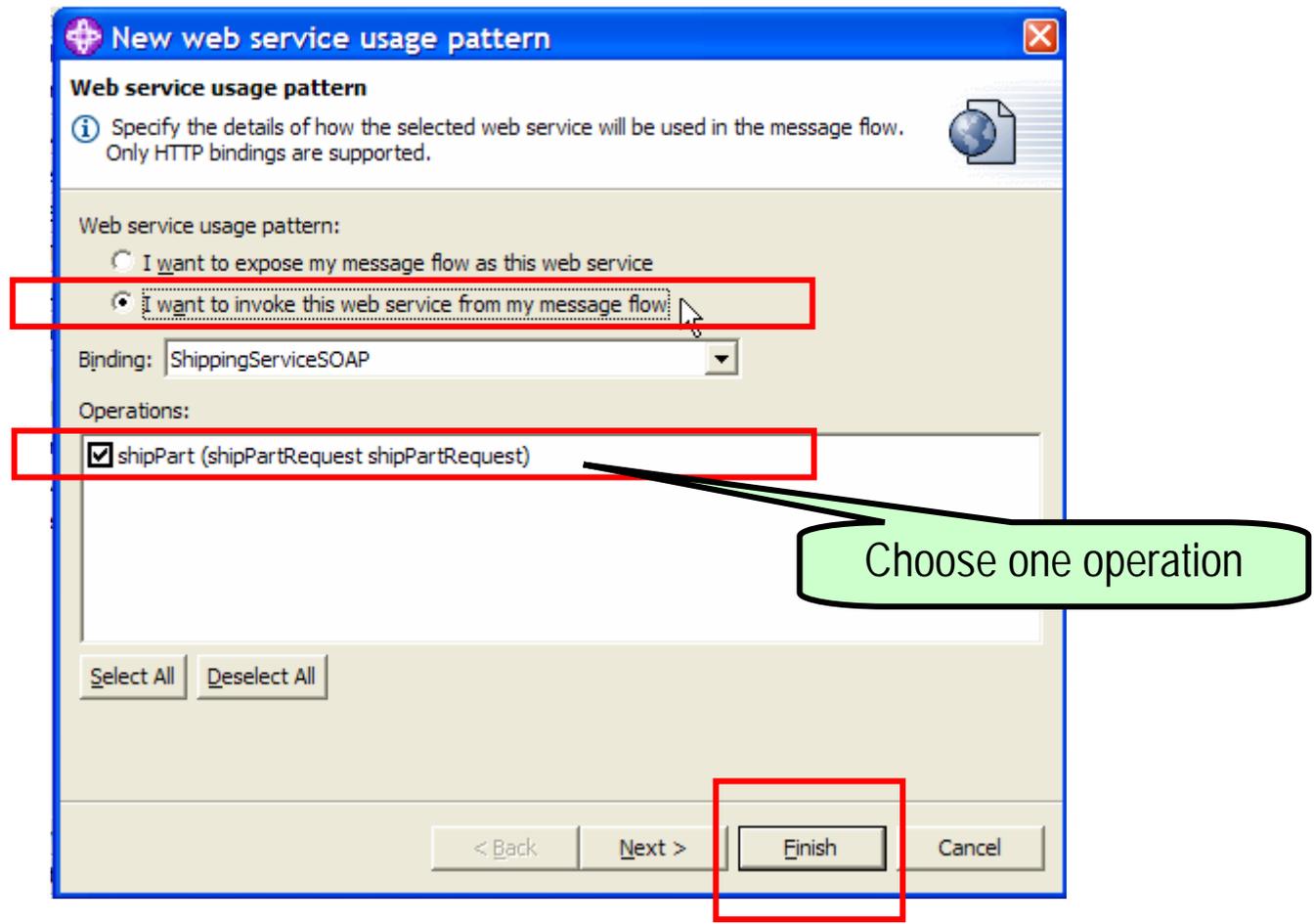
- Now that we have built up the message for invoking the web service, its time to actually do that:
 - Drag and drop the target web service’s WSDL to canvas
 - Choose to invoke the web service
 - Choose the operation to invoke
 - Wire to generated nodes

- **10a** – drag and drop `ShippingService.wsdl` from message set’s “importFiles” folder ...

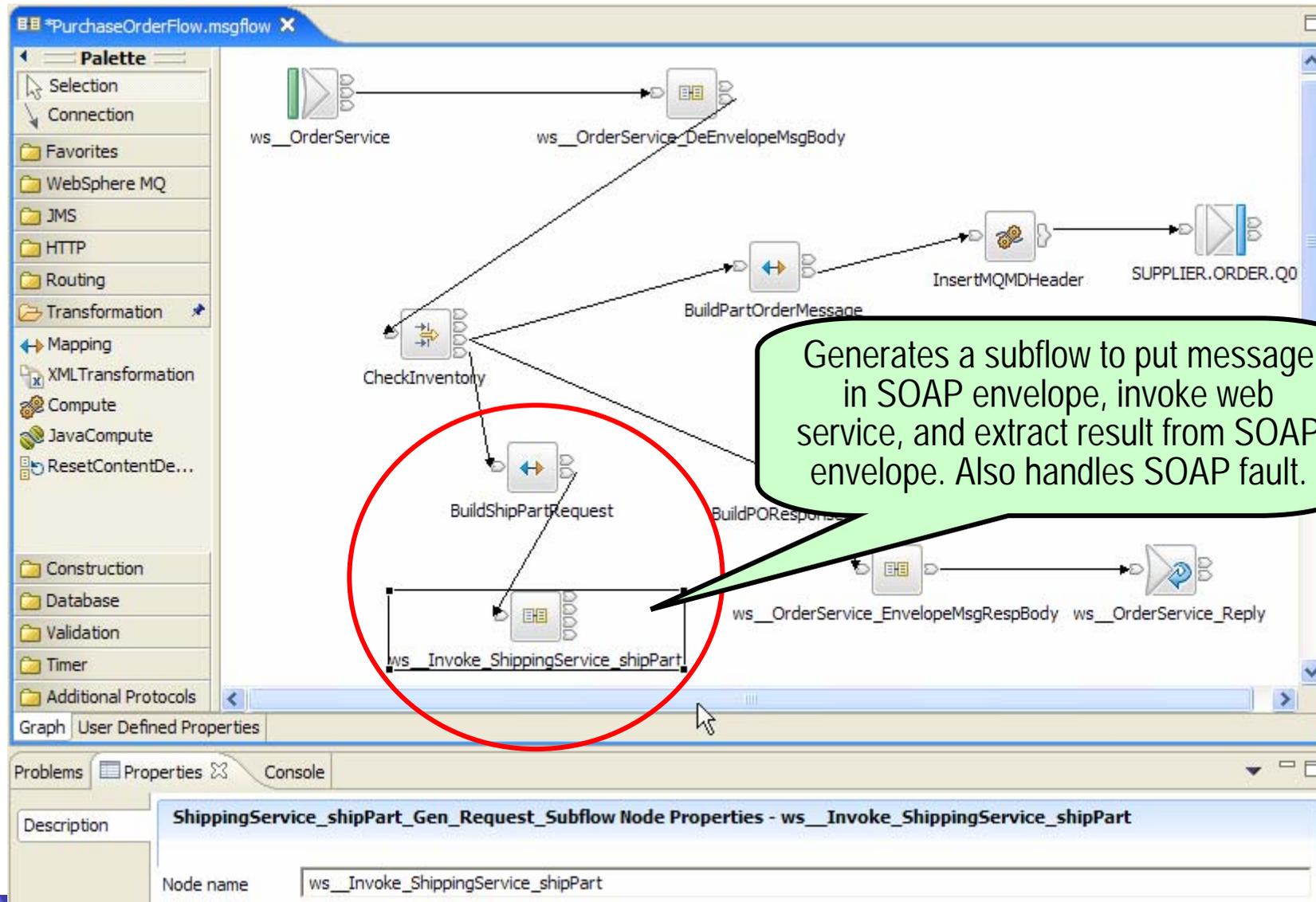
Scenario: Step 10a – Drag and Drop WSDL



Scenario: Step 10b – WSDL Drag and Drop Wizard



Scenario: Step 10c – Generated Subflow



Scenario: Step 10c – Inside the Subflow

The screenshot displays the IBM WebSphere Message Brokers ToolKit interface. The main workspace shows a message flow diagram for a subflow named 'ShippingService_shipPart_Gen_Request_Subflow.msgflow'. The flow starts with an 'in' node, followed by a 'ws_Envelope_Message' node, then a 'ws_ShippingService_Invoke' node, and finally a 'ws_De-envelope_Message' node. The flow then branches into two paths: one leading to 'ws_shipPartResponse' and 'shipPartResponse' nodes, and another leading to 'ws_Fault' and 'Fault' nodes. A 'failure' node is also present. The left sidebar shows a project tree with 'PurchaseOrder' and 'ShippingService_shipPart_Gen_Request_Subflow' highlighted. The bottom of the screen shows the 'Properties' panel for the 'HTTP Request' node, with fields for 'Web service URL' and 'Request timeout (sec)'. Several callout boxes provide additional context:

- Generated subflow:** Points to the project tree on the left.
- HTTPRequest node to invoke web service:** Points to the 'ws_ShippingService_Invoke' node.
- New SOAPEnvelope node. Surrounds body with SOAP envelope:** Points to the 'ws_Envelope_Message' node.
- New SOAPExtract node. Extracts response message body from SOAP envelope:** Points to the 'ws_De-envelope_Message' node.
- Output node for the WSDL operation (selected in wizard) and output node for SOAP fault:** Points to the 'shipPartResponse' and 'Fault' nodes.

Scenario: Step 11 – Prepare and Send Reply

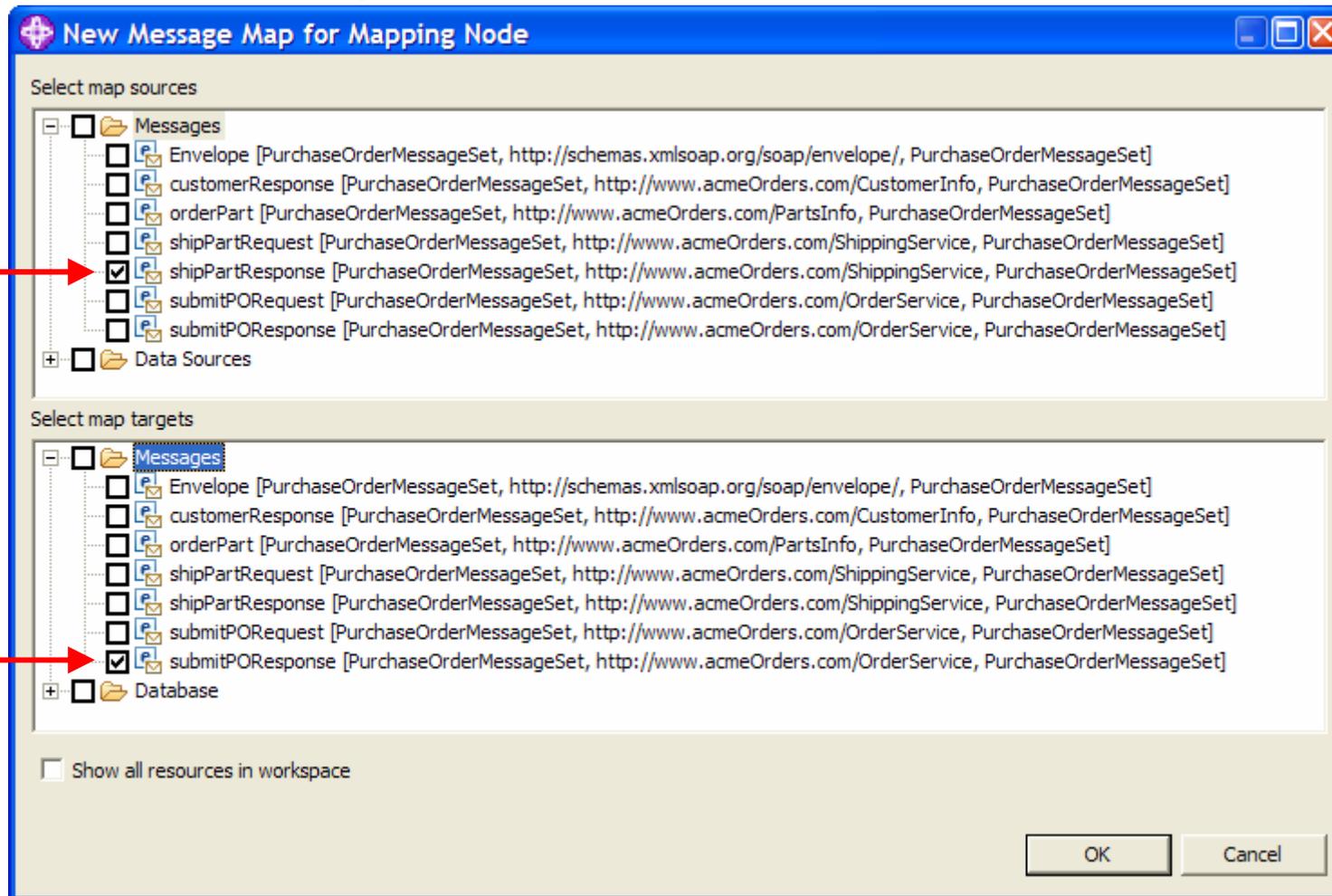
- Finally, after successfully invoking the web service to ship the parts to the customer, we can reply to the caller who invoked our web service:
 - Use a map node to create the response message
 - Wire from generated subflow to the map
 - Wire from the map to generated reply nodes

- **11a** – drag and drop a map node and set its name and mapping routine property to `BuildPOResponseShipped`

Scenario: Step 11a – Create Mapping Node

The screenshot displays the IBM WebSphere Business Modeler interface. At the top, a diagram shows a flow from 'BuildShipPartRequest' to 'ws__Invoke__ShippingService__shipPart', which then connects to 'ws__OrderService__EnvelopeMsgRespBodyws__OrderService__Reply'. A 'BuildPOResponseShipped' node is highlighted with a red box. Below the diagram, the 'Mapping Node Properties - BuildPOResponseShipped' dialog is open, showing the 'Mapping routine' field set to 'BuildPOResponseShipped', also highlighted with a red box. Other fields include 'Data source', 'Transaction' (Automatic), 'Mapping mode' (Message), 'Treat warnings as errors' (unchecked), and 'Throw exception on database error' (checked).

Scenario: Step 11b – Open Map



Scenario: Step 11c – Do Mapping

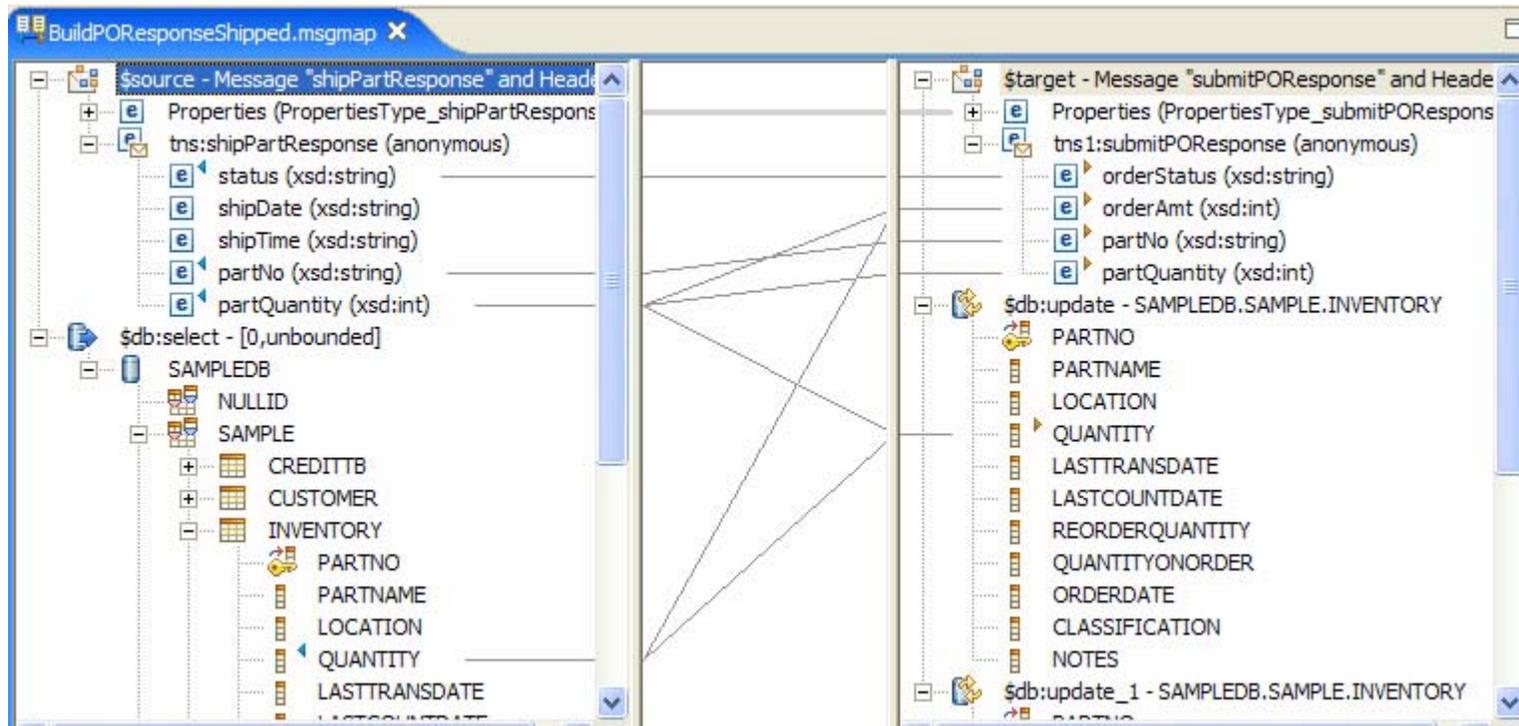
- Select \$source and \$target roots
- Right click on \$target and select Map By Name. Press Enter.
- Drag and drop status to OrderStatus.

The screenshot displays the IBM WebSphere Message Designer interface for configuring a message mapping. The source message is "shipPartResponse" and the target is "submitPOResponse". The mapping table below shows the following configuration:

Map Script	Value
BuildPOResponseShipped	
Parameters	
\$target	
Properties	
tns2:submitPOResponse	
orderStatus	\$source/tns:shipPartResponse/status
orderAmt	50.0
partNo	\$source/tns:shipPartResponse/partNo
partQuantity	\$source/tns:shipPartResponse/partQuantity

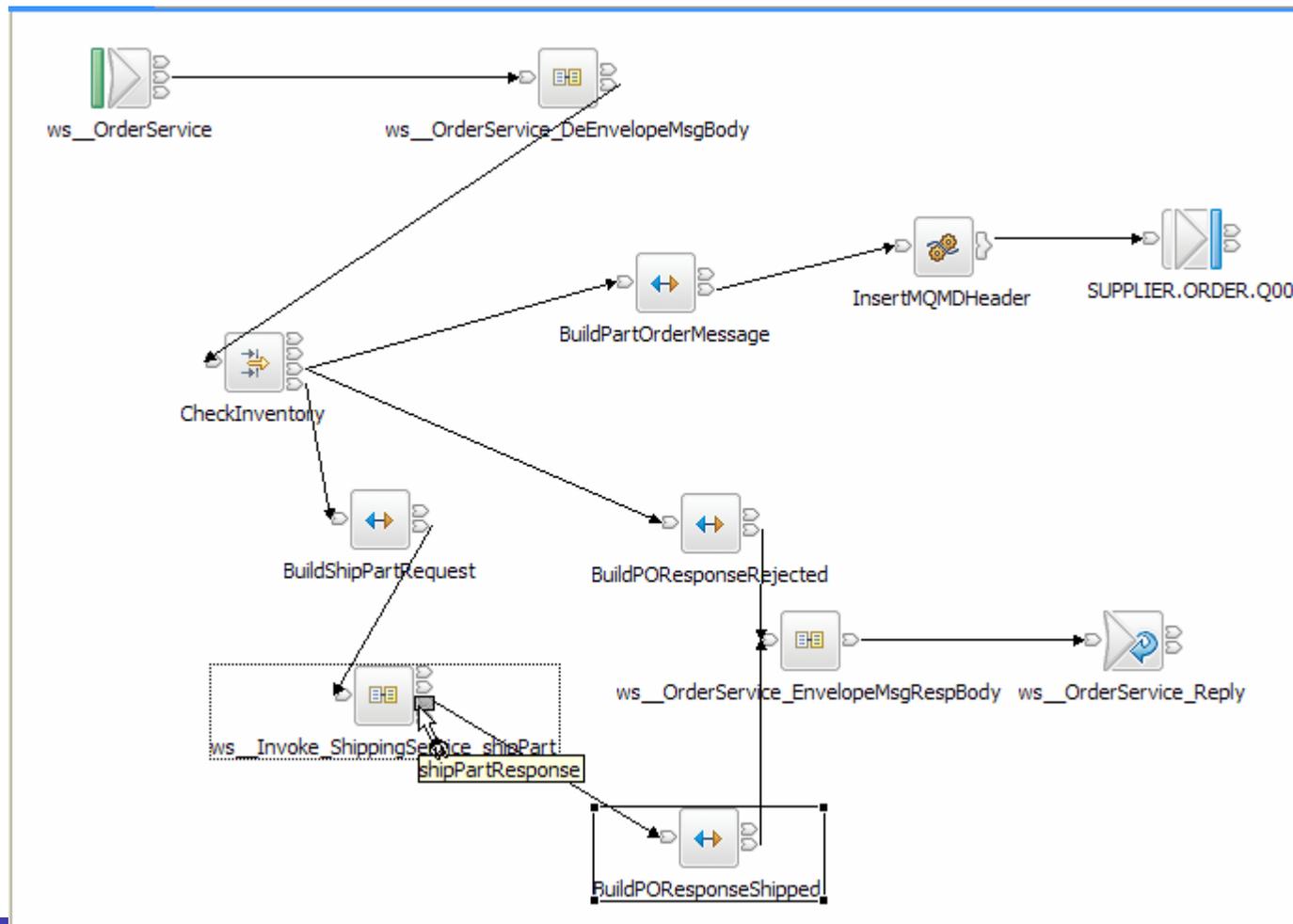
About that "orderAmt" value

- The orderAmt value should be set to the quantity times the part price, but in interest of time aren't doing that here.
- But it is doable. In fact, you could in one map, check the inventory, limit the orderAmt to what is actually available, update the inventory, and compute the orderAmt! Here it is:

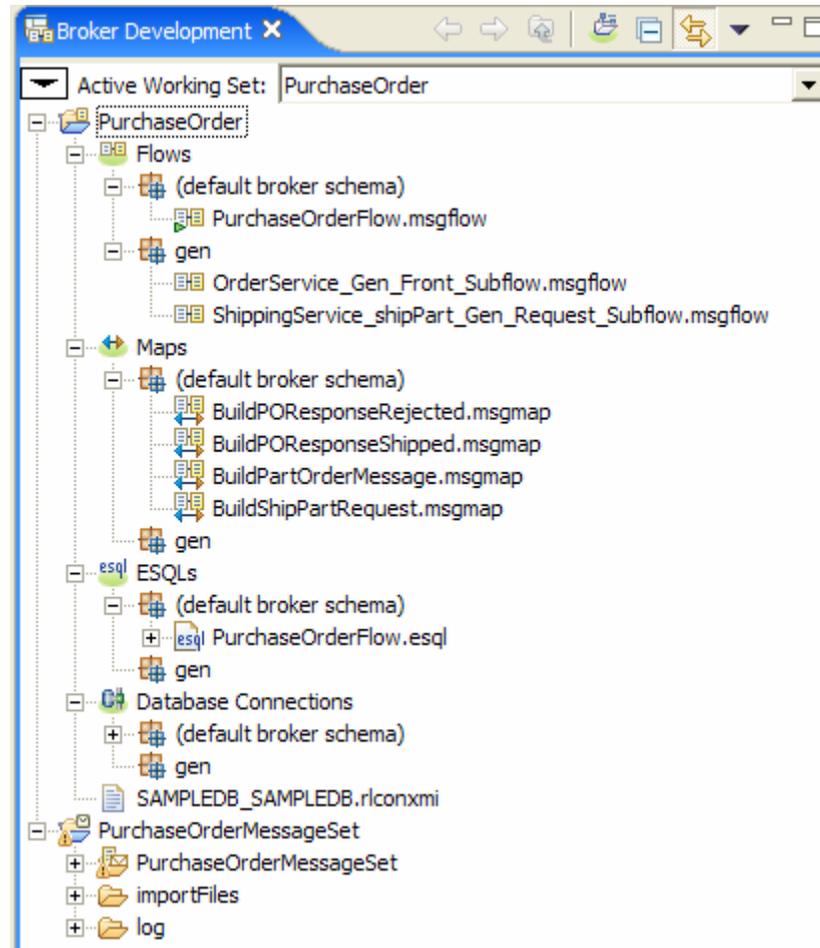


Scenario: Step 11d – Wire Map Node

- Wire as shown below: from `shipPartResponse` terminal to map, and from map's out terminal to generated reply `SOAPEnvelope` node



AUTHORING STEPS ARE DONE!



Time to Test!

- In previous releases, there was no built-in way to test a flow that starts with HTTPInput.
- New in 6.0.2 is support for a new “Test client” for testing flows that are started via HTTPInput or MQInput nodes.

Scenario: Step 12a – Invoke Test Client

- Assuming you have created all necessary configurations (config mgr, queue mgr listener, broker, queues, etc), you can test your flow by right clicking on the HTTPInput node:

The screenshot displays the IBM WebSphere Studio interface. On the left, a message flow diagram shows a node named 'ws__OrderService'. A context menu is open over this node, with the 'Test...' option highlighted. A yellow callout bubble points to the 'Test...' option with the text 'Event viewer'. On the right, the 'Events' test client window is open, showing the 'PurchaseOrderFlow.msgflow' and '*PurchaseOrderFlow.mbtest' tabs. The 'Message Flow Test Events' section has 'Invoke Message Flow' selected. A yellow callout bubble points to this section with the text 'Instance data editor for XML data.'. The 'General Properties' section shows 'Message flow: /PurchaseOrder/PurchaseOrderFlow.msgflow' and 'Input node: ws__OrderService'. The 'Detailed Properties' section shows 'Message content: XML Structure' and a table with columns 'Name', 'Type', and 'Value'. The table contains the following data:

Name	Type	Value
[-] Envelope	Envelope	
Header	Header	<unset>
[-] Body	Body	
Fault	Fault	<unset>
encodingStyle	anyURI	<unset>

Buttons for 'Saved Messages', 'Show Generated Source...', and 'Send Message' are visible at the bottom of the 'Events' window.

Scenario: Step 12b – Choose Message Body

- Right click on “Body” in message content, and select “**Add SOAP Message Body**”

The screenshot displays the IBM WebSphere Message Designer interface. The main window shows a message flow named "PurchaseOrderFlow.msgflow" with a test event "Invoke Message Flow". A context menu is open over the "Body" element in the message content, with "Add SOAP Message Body" selected. A "Type Selection" dialog box is open, showing a list of matching types including "Body", "customerResponse", "Envelope", "Fault", "Header", "orderPart", "shipPartRequest", "shipPartResponse", "submitPORrequest", and "submitPORresponse". The "shipPartRequest" type is selected. The "Qualifier" field contains the URL "http://www.acmeOrders.com/ShippingService".

Initial "event"

Initially shows generic SOAP envelope. Right click to pick the message describing the specific body expected for this input node

74

Scenario: Step 12c – Set Input Data

- Set `partNo` to 100-abc and `partQuantity` to 5. Press **Send Message**. Then press **Show Generated Source...**

The screenshot displays the IBM WebSphere IDE interface. A 'Generated Source' dialog box is open in the foreground, showing the following XML code:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/so
<soapenv:Body>
  <tns:shipPartRequest>
    <partNo>100-abc</partNo>
    <partQuantity>5</partQuantity>
    <personName>
      <firstName>firstName</firstName>
      <lastName>lastName</lastName>
    </personName>
    <address>
      <street>street</street>
      <city>city</city>
      <zipCode>zipCode</zipCode>
    </address>
  </tns:shipPartRequest>
</soapenv:Body>
</soapenv:Envelope>
```

The background window shows the 'Send Messages' configuration. The 'Message flow' is set to `/PurchaseOrder/PurchaseOrderFlow.msgflow` and the 'Node' is `ws__OrderService`. The 'Message content' is set to 'XML Structure'. A table below shows the input data:

Name	Type	Value
shipPartRequest	shipPartRequest	
partNo	string	100-abc
partQuantity	int	5
personName	personName	
firstName	string	firstName
lastName	string	lastName
address	address	
street	string	street

The 'Show Generated Source...' button is circled in red, and a red arrow points from it to the 'Generated Source' dialog. The 'Send Message' button is also visible.

Scenario: Step 12d – Save Input Data

- Since we don't want to have to enter the data in each time, right click and select **Add Value to Pool**.

The screenshot shows the IBM WebSphere Studio interface. A right-click context menu is open over the 'Invoke Message Flow' event in the 'Message Flow Test Events' pane. The menu options include 'Add SOAP Message Body', 'Remove Message Part', 'Change Message Part', 'Set Value...', 'Unset', 'Set to Null', 'Reset to Default', 'Add Element', 'Remove Element', 'Delete', 'Copy Value', 'Add Value to Pool...', 'Use Value from Pool...', 'Paste Value', and 'Select All'. The 'Add Value to Pool...' option is highlighted, and a red arrow points to it. A 'Value Name' dialog box is also open, with the text 'OrderProcess_InStock' entered in the input field. The dialog has 'OK', 'Cancel', and 'Show Value' buttons. The background shows the 'PurchaseOrderFlow.msgflow' and '*PurchaseOrderFlow.mbttest' tabs, and a message flow diagram with various nodes like 'Header', 'Body', 'Fault', 'shipPartRequest', 'partNo', 'partQuantity', and 'personName'.

Scenario: Step 12e – Run Test

- Ok, time to launch! Press Send Message....

The screenshot shows the 'Events' window in the IBM WebSphere Message Tester. The window title is 'PurchaseOrderFlow.msgflow' and the active tab is '*PurchaseOrderFlow.mbtest'. The 'Events' section contains a single event: 'Invoke Message Flow'. The 'General Properties' section is expanded, showing 'Message flow: /PurchaseOrder/PurchaseOrderFlow.msgflow' and 'Input node: ws__OrderService'. The 'Detailed Properties' section is also expanded, showing 'Message content: XML Structure' and a table of message properties.

Name	Type	Value
[-] shipPartRequest	shipPartRequest	
partNo	string	100-abc
partQuantity	int	5
[-] personName	personName	
firstName	string	firstName
lastName	string	lastName
[-] address	address	
street	string	street

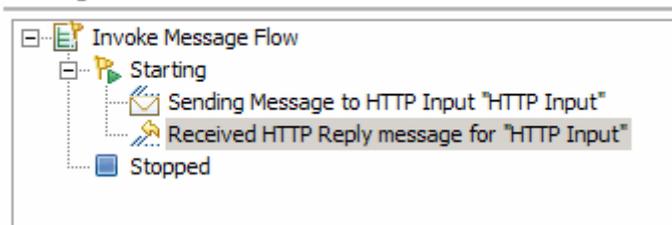
At the bottom right of the window, the 'Send Message' button is circled in red, with a red arrow pointing to it from the right.

Scenario: Step 12e - Events

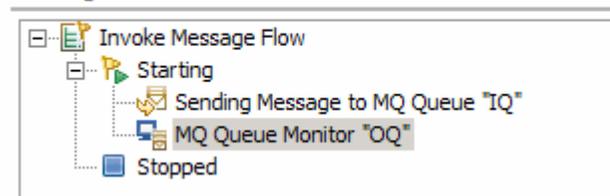
- The Message Flow Test Events tree will show events:

Event Type	What you can see
Starting	All MQ output queue monitors
Sending message to HTTP Input node	Request message in a readonly source editor
Sending message to MQ Input node queue	The queue, queue manager, host, port of the MQ queue that the message came from The message in a readonly source editor.
Receiving message from HTTP Reply node	Response message in a readonly source editor.
Receiving message at MQ Output node queue	The queue, queue manager, host, port of the mq queue that is being listened to. The reply message in a readonly source editor
Stopping	

Message Flow Test Events



Message Flow Test Events



Test Client after two runs

The screenshot displays the IBM WebSphere Test Client interface for a message flow named 'purchaseOrderFlow'. The 'Events' pane on the left shows a tree view of 'Message Flow Test Events' under 'Invoke Message Flow'. The tree includes two cycles of 'Starting', 'Sending Message to HTTP Input "ws__Ord...', 'Received HTTP Reply message for "ws__C...', and 'Stopped'.

The right pane shows 'General Properties' and 'Detailed Properties'. The 'Message flow' is set to '/purchaseOrder/purchaseOrderFlow.msgflow' and the 'Input node' is 'ws__OrderService'. The 'Message content' is set to 'Source'. Below this, the XML message content is displayed in a scrollable text area:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  <soapenv:Body>
    <tns:submitPORrequest>
      <partNo>ABC-100</partNo>
      <partQuantity>555</partQuantity>
      <personName>
        <firstName>firstName</firstName>
        <lastName>lastName</lastName>
      </personName>
      <address>
        <street>street</street>
        <city>city</city>
        <zipCode>zipCode</zipCode>
      </address>
    </tns:submitPORrequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Buttons for 'Import Source...' and 'Send Message' are visible at the bottom of the interface.

More about Test Client

- The Configuration page is for configuring your test session:

The screenshot shows the 'Configuration' page in the IBM Test Client. The page is divided into several sections:

- Test Client Configuration:** A tree view on the left shows a hierarchy of configurations. A callout points to this section, stating: "Lists the message flows part of this test. Can add and remove flows".
- General Properties:** A section for general settings.
- MQMD Settings for MQ Test Messages:** A section for MQMD-related settings.
- Detailed Properties:** A section for detailed settings, including:
 - Broker Archive Settings:** A field for "Override broker archive file:" with a "Browse..." button. A callout points to this field, stating: "Can specify your own BAR file versus letting client generate one".
 - Monitor Settings:** A section with three options:
 - Stop testing when message reaches first output monitor
 - Purge message from MQ output queue
 - Browse message from MQ output queueA callout points to this section, stating: "What to do when you get an event that a msg has arrived on a monitored queue".

At the bottom of the window, there are two tabs: "Events" and "Configuration". The "Configuration" tab is circled in red.

More about Test Client

- The Configuration page is for configuring your test session:

Test Client Configuration

General Properties

Name: Configuration

Description:

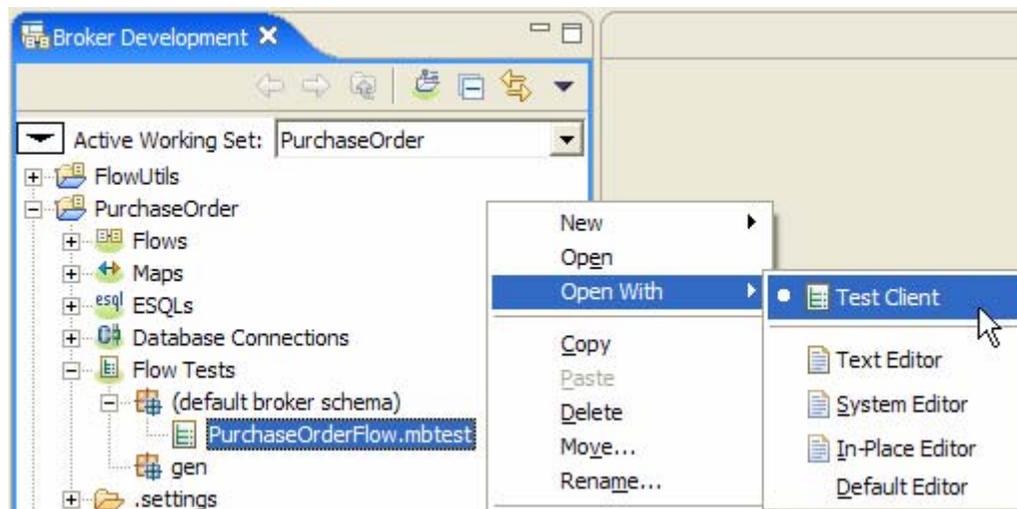
MQMD Settings for MQ Test Messages

Application origin data	<input type="text"/>	Offset	<input type="text" value="0"/>
Application id data	<input type="text"/>	Original length	<input type="text" value="-1"/>
Backout count	<input type="text" value="0"/>	Persistence	<input type="text" value="2"/>
Character set	<input type="text" value="0"/>	Priority	<input type="text" value="-1"/>
Encoding	<input type="text" value="273"/>	Put application name	<input type="text"/>
Expiry	<input type="text" value="-1"/>	Put application type	<input type="text" value="0"/>
Feedback	<input type="text" value="0"/>	Put date/time	<input type="text"/>
Format	<input type="text"/>	Report	<input type="text" value="0"/>
Message flags	<input type="text" value="0"/>	Reply to queue manager name	<input type="text"/>
Message type	<input type="text" value="8"/>	Reply to queue name	<input type="text"/>
Message sequence number	<input type="text" value="1"/>	User id	<input type="text"/>

Detailed Properties

More about Test Client

- You can save your test session configuration. This allows you to reload it from the navigator:



Looking Ahead for the Test Client

- This is only the first release for this exciting new technology. It is technology we are sharing with WebSphere Integration Developer (WID), where it has proved very popular.
- Over time we have many ideas for evolving it beyond this first release, including:
 - Replacing Enqueue/Dequeue support for MQ.
 - Offer support for other protocols beyond just MQ and HTTP. Eg: JMS
 - Support events for entering and exiting each node in the flow
 - Support emulating nodes, so you don't have to actually invoke payroll. Instead, you just supply a result value as though it had been invoked
 - Support testcase creation, management and execution

Other Enhancements

- There are plenty of other enhancements that we couldn't show in the scenario...

BAR Editor

Content
Add and remove deployable files from this archive.
Deployable files are message flows, message sets, style sheets, XML and JAR files.

Buttons: Add, Remove, Refresh

New Refresh button!

Name	Type	Modified	Version	Comm...	Size	Path
PurchaseOrderFlow.cmf	Compiled message flow	Oct 10, 2006 9:45:...			457...	
PurchaseOrderMessageSet.dictionary	Dictionary file	Oct 10, 2006 9:45:...			23333	

Show source files

Content | Configure | User Log | Service Log

New User Log tab

New Service Log tab

New Properties view

Problems | Properties

Deployable

PurchaseOrderFlow.cmf

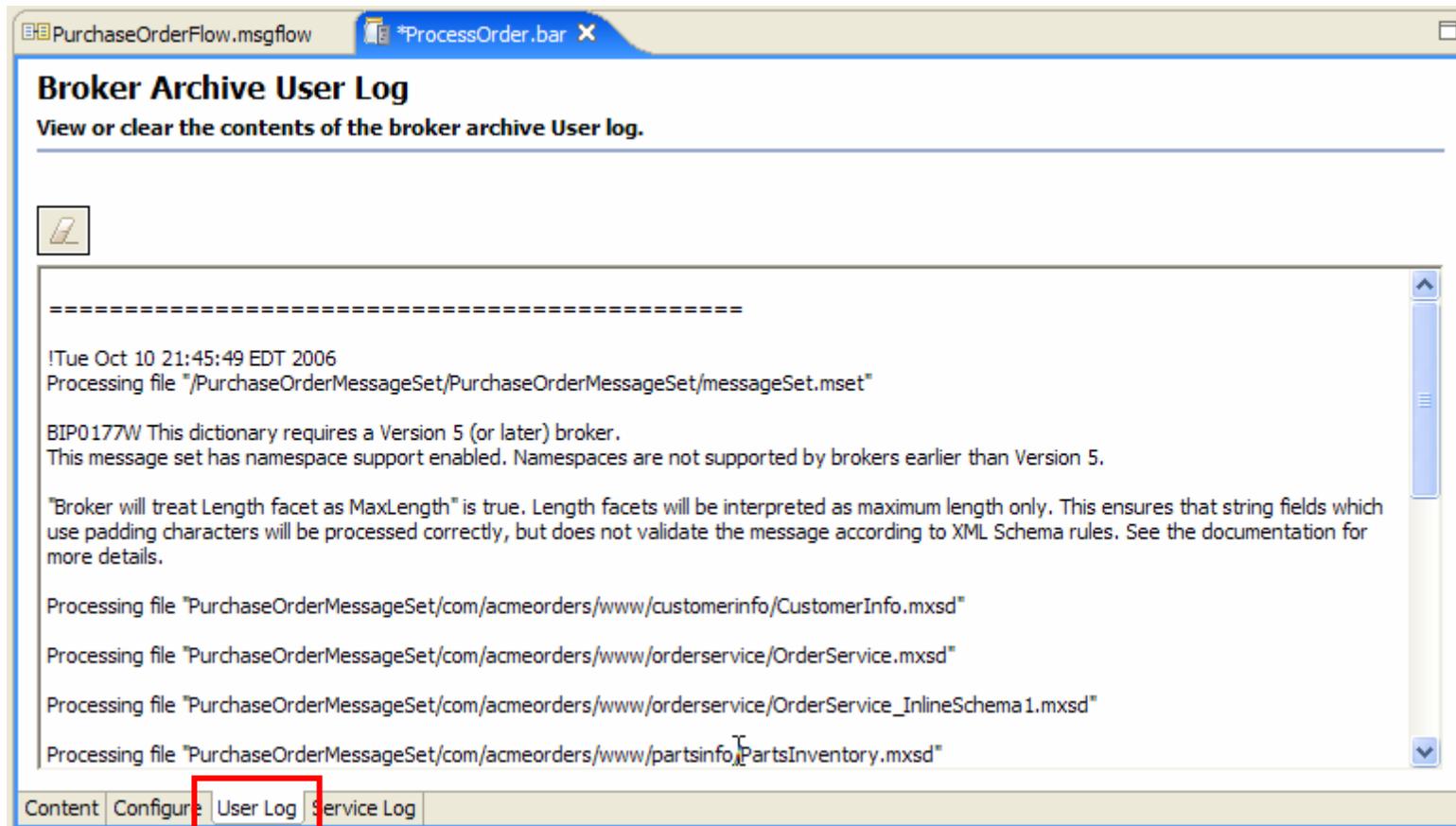
Workspace resource: /PurchaseOrder/PurchaseOrderFlow.msgflow [Change...]

Last compile status:

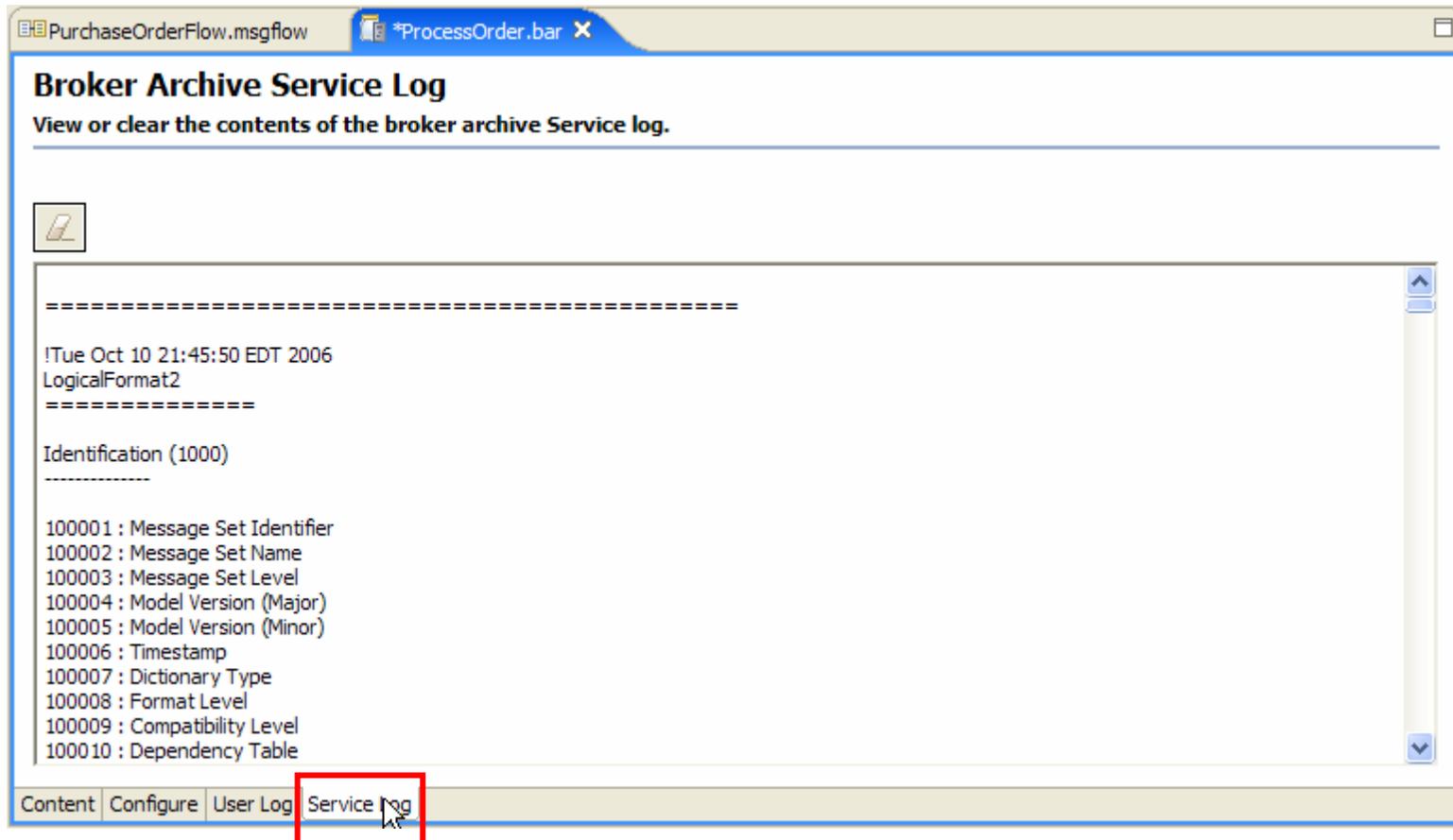
```
Processing file PurchaseOrderFlow.msgflow
Successfully added file PurchaseOrderFlow.msgflow to archive file
Elapsed time: 0.562 second(s).
The ESQL level for PurchaseOrderFlow.msgflow is 5.0
```

Show icons

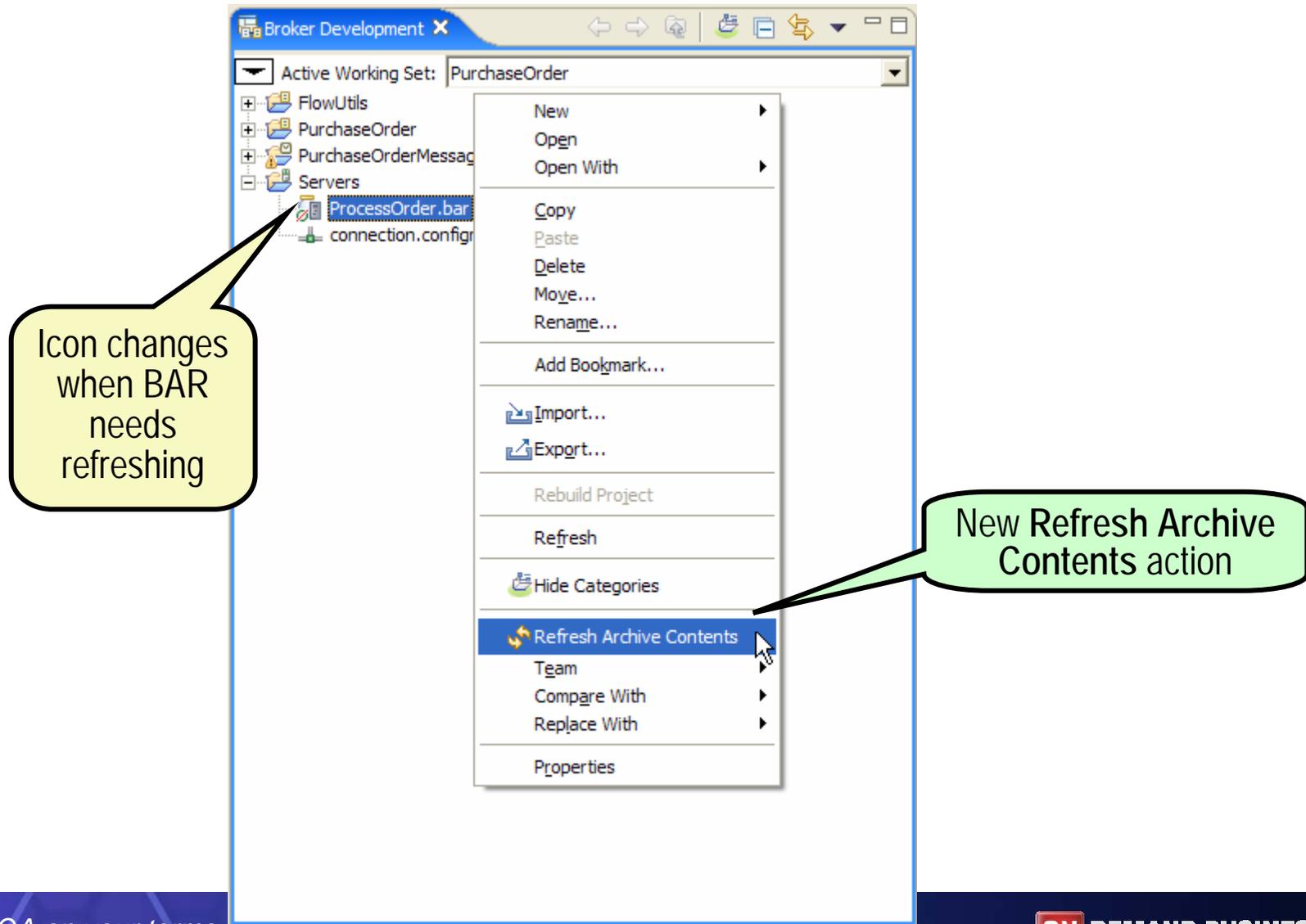
More BAR Editor



More BAR Editor



More BAR Editor - Refresh



Command : mqsiapplybaroverride

- There is also a new command for updating the override values in a BAR file:

- `>>-mqsiapplybaroverride-- -b --barFile -p --propertiesFile`

- Parameters

- `-b barFile`

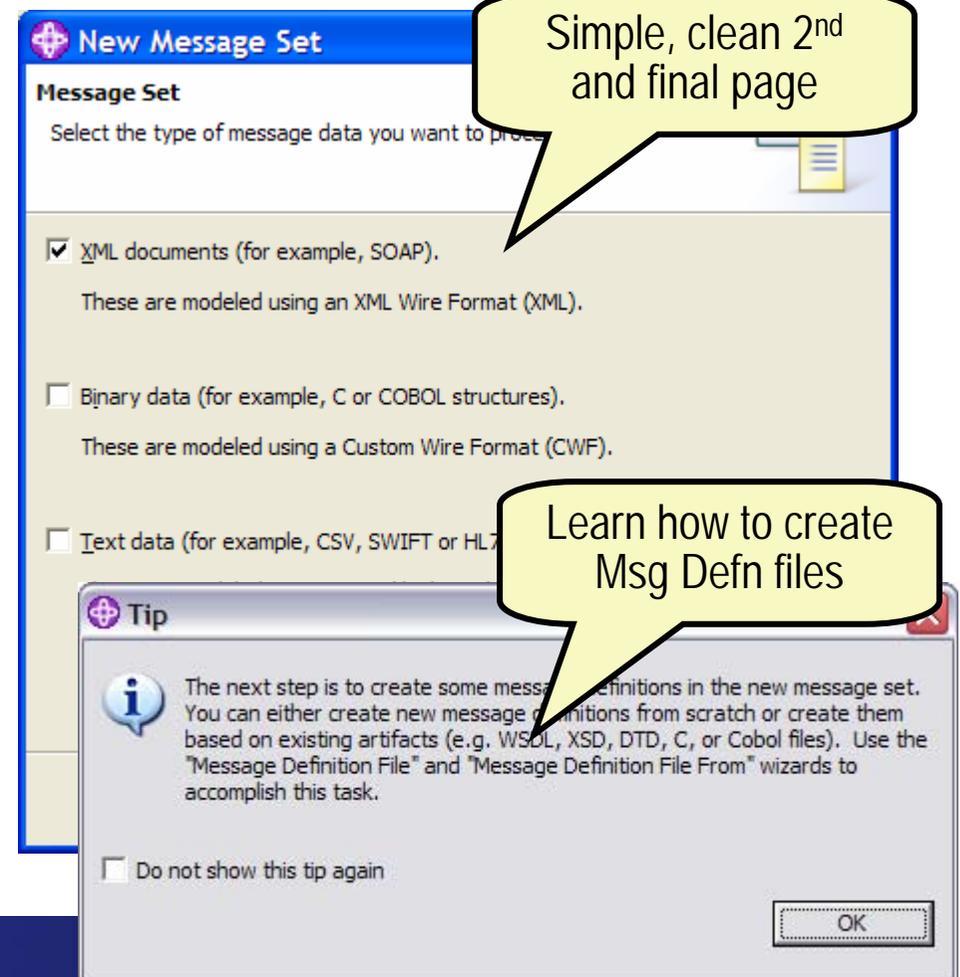
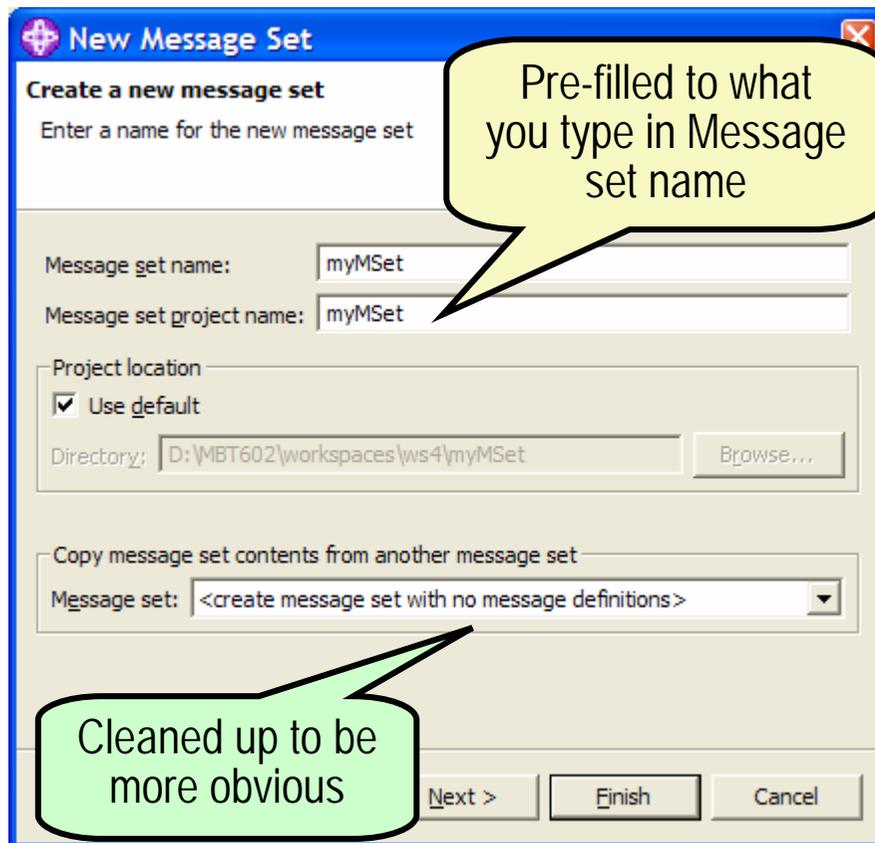
- (Required) The path to the broker archive file to apply override values to.
- i.e. relative from the executable or a full path to the broker archive file

- `-p propertiesFile`

- (Required) The path to the properties file containing the override values
- i.e. relative from the executable or a full path to the properties file

Simpler New Message Set wizard

- We have removed the New Message Set Project wizard, and instead enhanced the New Message Set wizard to allow creating the project at the same time:



Message Set Editor

The screenshot shows the Message Set Editor for a Message Set named 'PurchaseOrderMessageSet'. The Properties Hierarchy on the left lists categories: Message Set, XML Wire Formats, Custom Wire Formats, and Tagged/Delimited String Formats. The Details pane on the right shows the following configuration:

Property	Value
Message Domain	XMLNSC : For XML messages (namespace aware, low memory use)
Default Wire Format	<no default specified>
Message Set ID	C6UJ45C002001
Message Set Alias	<Optionally set a value if you identify embedded messages using a Message Identity. Press F1 for guidance.>
Message Type Prefix	<Optionally set a value if you identify embedded messages using a Message Path. Press F1 for guidance.>
Use namespaces	<input checked="" type="checkbox"/>
Broker will treat Length facet as MaxLength	<input checked="" type="checkbox"/>
Version	
Documentation	

Message Domain Descriptions:

- MRM : For binary, text or XML messages (namespace aware, validation, low memory use)
- XMLNSC : For XML messages (namespace aware, low memory use)
- XMLNS : For XML messages (namespace aware)
- IDOC : For SAP IDocs from the WebSphere MQ Link for R3
- MIME : For MIME wrapped data including multipart
- JMSMap : For JMS MapMessage messages (XML)
- JMSStream : For JMS StreamMessage messages (XML)
- XML : For XML messages (for Version 2 compatibility)

New: Message Set Editor Enhancements:

1. Changed "Runtime parser" to "Message Domain"
2. Show descriptions for each domain (fix pack 2)
3. Re-ordered fields and domains according to importance
4. Show text versus blank entry fields for Prefix and Alias
Show wire formats as categories in tree
5. Move Version and Description to each page

Message Set Editor for Wire Formats

Properties Hierarchy

- Message Set
 - XML Wire Formats
 - XML 1
 - Custom Wire Formats
 - Tagged/Delimited String Formats

XML Wire Format Details - XML1

Namespace settings

Namespace declarations

Namespace URI	Prefix	
http://schemas.xmlsoap.org/soap/envelope/	soapenv	Add
http://www.acmeOrders.com/CustomerInfo	tns	Delete
http://www.acmeOrders.com/OrderService	tns	
http://www.acmeOrders.com/PartsInfo	tns	
http://www.acmeOrders.com/ShippingService	tns	

Namespace schema locations

Namespace URI	Schema location	
<no target namespace>		Add
		Delete

Output Namespace Declaration

At start of document

- ▶ XML declaration
- ▶ XML document type settings
- ▶ XML representation of boolean values
- ▶ XML representation of null values
- ▶ Datetime settings
- ▶ xsi:type settings
- ▼ **Deprecated**

These properties are used to control legacy behaviour in the MRM parser and should not be changed from their default settings. They will be withdrawn in a future release.

Root Tag Name

Suppress Timestamp Comment

Enable Versioning Support

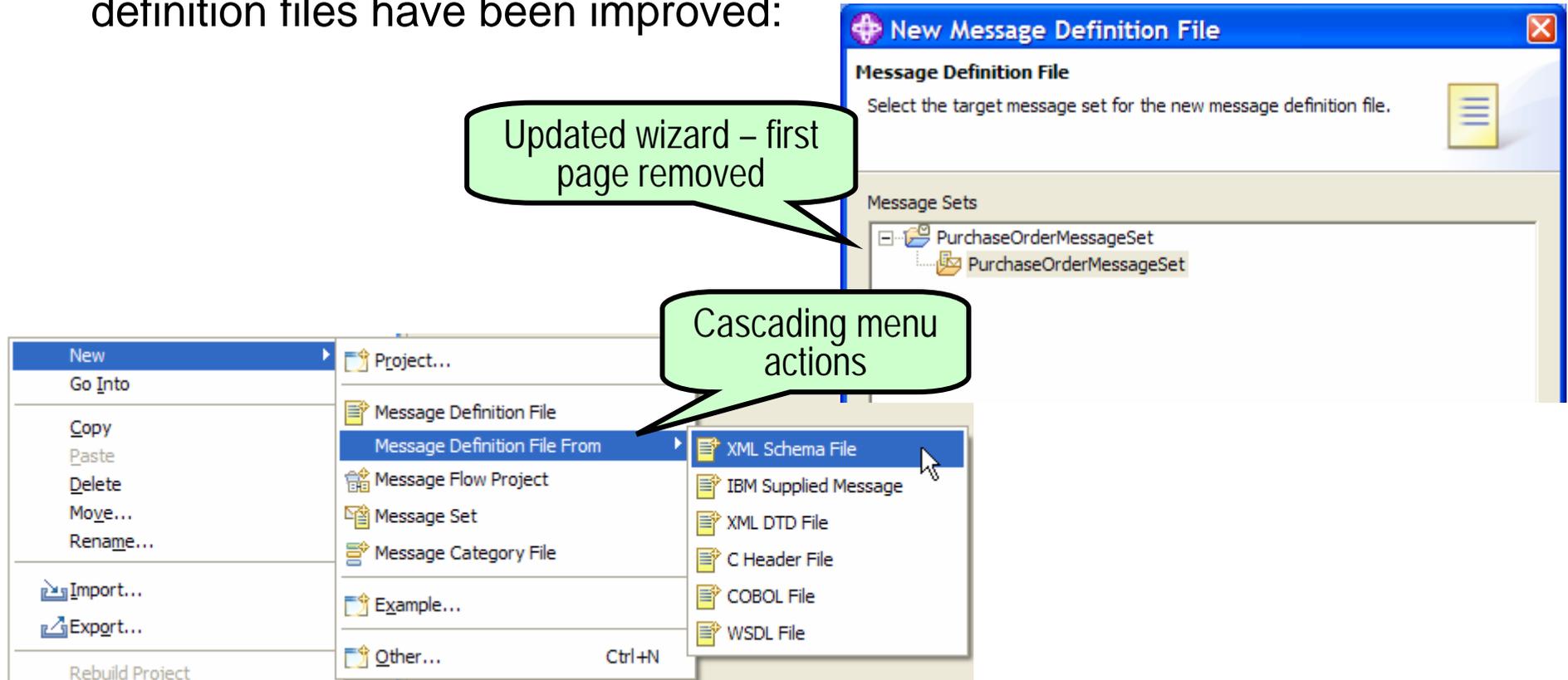
New: now partition wire format properties into collapsible sections

Slight tweaks to categorization of namespace-related properties

Collected older properties into new Deprecated section

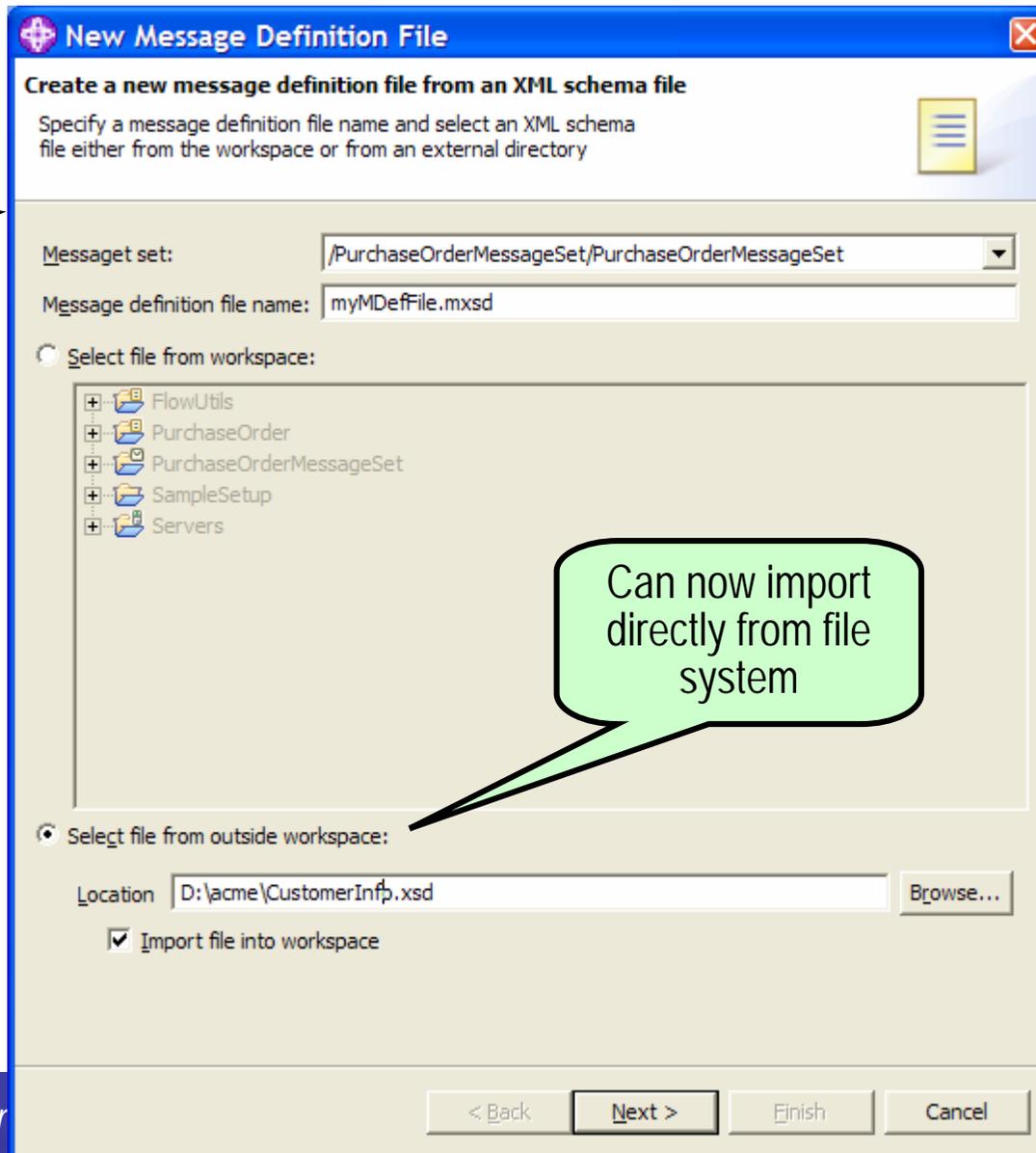
Simpler New Msg Defn File wizards

- The menu actions and most of the wizards for creating message definition files have been improved:



XSD, DTD, COBOL & C wizards

Simplified wizard pages



Can now import directly from file system

WSDL wizard

New Message Definition File
Create a new message definition file from a WSDL definition
Copied source files will overwrite existing files in the 'importFiles' folder.

Message set: /PurchaseOrderMessageSet/PurchaseOrderMessageSet

Select file from workspace:

- FlowUtils
- PurchaseOrder
- PurchaseOrderMessageSet
- SampleSetup
- Servers

Select file from outside workspace:

Location: D:\acme\OrderService.wsdl [Browse...]

New Message Definition File
Create a new message definition file from a WSDL definition
Select WSDL bindings for which physical format information will be generated.

Select the WSDL bindings to import:

- OrderService.wsdl
 - OrderServiceSOAP (Port type: OrderService)

< Back Next > Finish Cancel

Simplified wizard pages

Can now import directly from file system

Can now select multiple bindings

Looking Ahead

- Other high priority consumability features we are hoping to do **:
 - New non-programming Filter nodes
 - Specifying XPath expressions into the message.
First/all non-null expression fires associated terminal.
 - Specifying database query expressions using message content as input.
First/all non-null database query fires associated terminal
 - New transport header nodes to create MQMD, HTTP, JMS headers
 - MQOutput nodes auto-creates an MQMD when none present
 - Saves having to use Compute node prior to output nodes
 - New JMSReply node
 - So JMSOutput isn't overloaded

**** These statements represent IBM's current intent for future product delivery.
IBM directions and plans are subject to change without notice**

What are you waiting for!

- eGA was Dec 22nd 2006 for existing customers
- Go to WebSphere Message Broker Recommended Fixes site
<http://www-1.ibm.com/support/docview.wss?rs=849&uid=swg27006041>
- SOAPExtract & SOAPEnvelope nodes available via SupportPac IA90 **
<http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg24014070>
- Installation using RPU

- eGA for Passport Advantage customers in Feb 2007
- pGA (CDs) in March 2007

**** Requires WMB V6.0 FixPack 3 runtime**