© Copyright IBM Corporation 2008 All rights reserved

IBM WEBSPHERE 6.1 – LAB EXERCISE

WebSphere Enterprise Service Bus 6.1 mediation programming model

Lab One – Message augmentation

What this exercise is about

The objective of this lab is to provide you with an understanding of how to use the service invoke mediation primitive to augment the contents of a message within a mediation flow.

This lab is provided **AS-IS**, with no formal IBM support.

Lab requirements

- WebSphere Integration Developer V6.1 installed on Windows or Linux
- WebSphere Enterprise Service Bus V6.1 or WebSphere Process Server V6.1. The server can either be the test server installed by WebSphere Integration Developer or a remote server from a separate WebSphere Enterprise Service Bus V6.1 or WebSphere Process Server V6.1 installation.

© Copyright IBM Corporation 2008. All rights reserved

What you should be able to do

At the end of this lab you should be able to:

- Configure a mediation flow to use the service invoke mediation primitive to call a service from within a mediation flow

- Use the returned values from a service invoke mediation primitive to augment the contents of a message

Introduction

This lab exercise is the first of a series of four tutorials intended to illustrate the new programming model for mediations introduced by WebSphere Enterprise Service Bus V6.1. The new programming model includes message augmentation using service invoke, splitting and aggregating to handle repeating elements within a message and service call retry capabilities including the use of alternate service endpoints.

The four tutorials are described in the presentation entitled Augmentation, aggregation and retry tutorials. You should familiarize yourself with the tutorials as described in the presentation before attempting this lab.

© Copyright IBM Corporation 2008. All rights reserved

Exercise instructions

Some instructions in this lab are Windows operating system specific. If you plan on running WebSphere Integration Developer on a Linux operating system you will need to run the appropriate commands and use appropriate files for Linux. The directory locations are specified in the lab instructions using symbolic references, as follows:

Reference variable	Example Windows location	Example Linux location
<WID_HOME>	C:\Program Files\IBM\WID61	/opt/IBM/WID61
<ESB_PROFILE_HOME>	<WID_HOME>\pf\esb	<WID_HOME>/pf/esb
<LAB_FILES>	C:\Labfiles61\WESB\61ProgModel	/tmp/Labfiles61/WESB/61ProgModel

Instructions if using a remote server for testing

Note that the previous table is relative to where you are running WebSphere Integration Developer. The following table is related to where you are running the remote test environment:

Reference variable	Example: Remote Windows test server location	Example: Remote z/OS® test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	sssr011	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\AppServer	/etc/sscell/AppServer	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<SOAP_PORT>	8880	8880	
<TELNET_PORT>	N/A	1023	
<PROFILE_NAME>	AppSrv01	default	
<USERID>	N/A	ssadmin	
<PASSWORD>	N/A	fr1day	

Instructions for using a remote testing environment, such as z/OS, AIX® or Solaris, can be found at the end of this document, in the section "**Task: Adding remote server to the WebSphere Integration Developer test environment**".

© Copyright IBM Corporation 2008. All rights reserved

Understanding how to read the instructions

In this lab, the instructions are written to allow an experienced user to complete the steps easily while at the same time providing very explicit instructions needed by the new user. The format of the instructions follows this pattern:

____ 1. This is a sentence or short paragraph that describes a particular task to be completed. In some cases this is sufficient for an experienced user, but in other cases the experienced user might require some additional specific information to complete the task. In that case, there is a bulleted list that helps the experience user know specifics.

- **`Additional information for experienced user`**
- **`This information, along with the above paragraph, should allow the experienced user to complete the task`**
- **`---------------------------------------------------------------`**

__ a. First step needed by the new user

__ b. Second step needed by the new user

1) Additional details for completing this step

2) More details for completing this step

__ c. Third step needed by new user

____ 2. Next task to be completed

- **`Info for experience user`**
- **`---------------------------------------------------------------`**

__ a. First step needed by the new user

__ b. Second step needed by the new user.

© Copyright IBM Corporation 2008. All rights reserved

Part 1: Setting up the environment for the lab

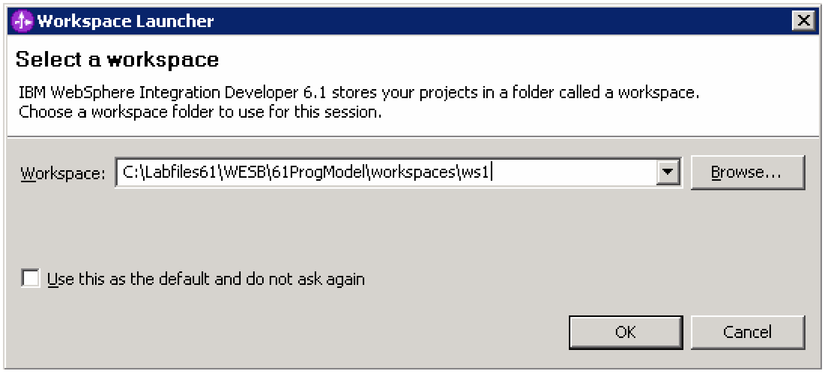
What you will do in this part: This part of the lab gets the environment ready to do the lab exercise.

____ 1. Start WebSphere Integration Developer, preferably using a new workspace

- **`Suggested location: <LAB_FILES>/workspaces/ws1`**
- **`-------------------------------------------------------------------`**

__ a. Select **Start → All Programs → IBM WebSphere Integration Developer → IBM WebSphere Integration Developer V6.1 → WebSphere Integration Developer V6.1**

__ b. From the Workspace Launcher window, enter **<LAB_FILES>/workspaces/ws1** for the 'Workspace' field



Select a workspace

IBM WebSphere Integration Developer 6.1 stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Labfiles61\WESB\61ProgModel\workspaces\ws1

☐ Use this as the default and do not ask again

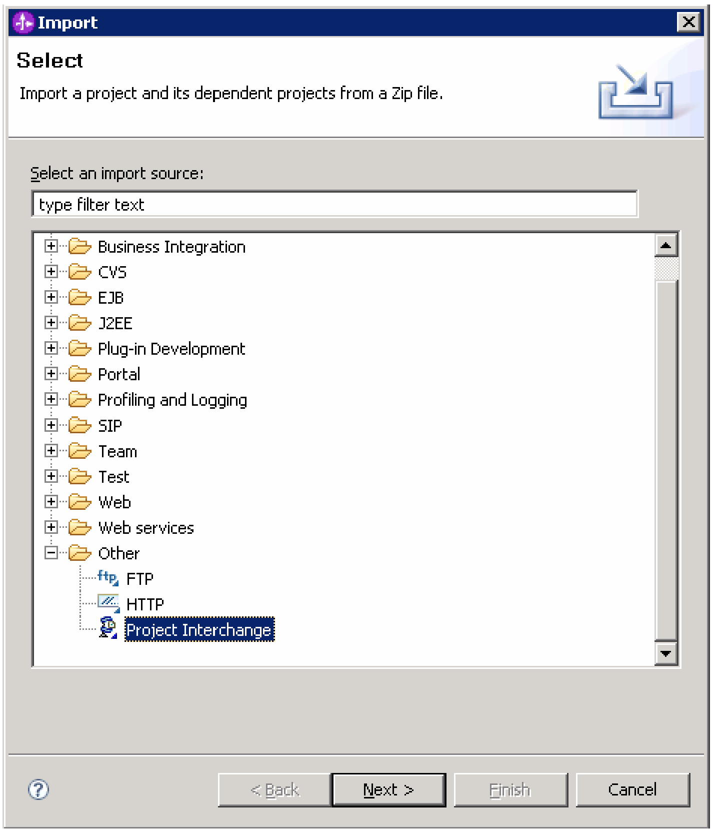
__ c. If the Welcome panel is displayed, click on **Go to the Business Integration perspective** or the arrow next to it in the upper right corner of the panel.

____ 2. Import the project interchange file containing the starting point for the lab

- **`<LAB_FILES>/PI1-AugmentStart.zip`**
- **`-------------------------------------------------------------------`**

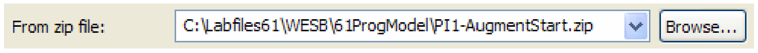
__ a. From the menu, select **File → Import…**

__ b. In the **Import** dialog, select **Other → Project Interchange**

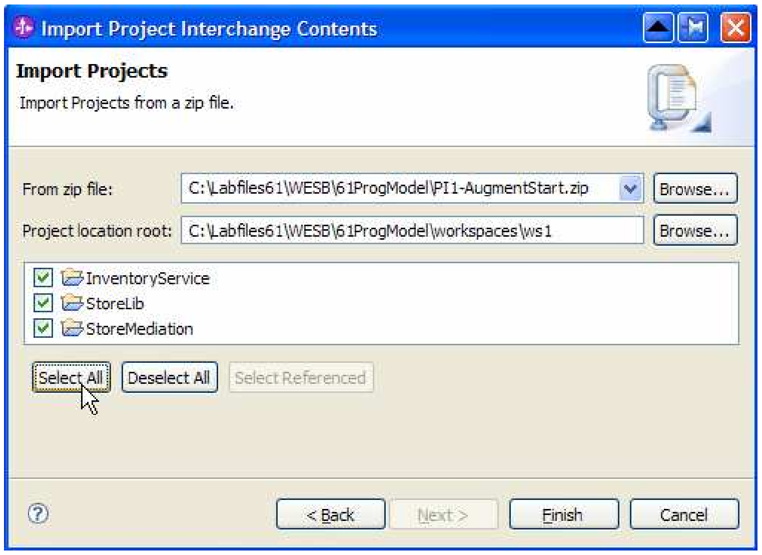


__ c. Click **Next**

__ d. In the **Import Project Interchange Contents** dialog set the **From zip file:** value to **<LAB_FILES>/PI1-AugmentStart.zip**


From zip file: C:\Labfiles61\WESB\61ProgModel\PI1-AugmentStart.zip

__ e. Click **Select All** to select all of the projects listed


Import Project Interchange Contents
Import Projects
Import Projects from a zip file.

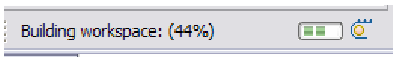
From zip file: C:\Labfiles61\WESB\61ProgModel\PI1-AugmentStart.zip
Project location root: C:\Labfiles61\WESB\61ProgModel\workspaces\ws1

[✓] InventoryService
[✓] StoreLib
[✓] StoreMediation

© Copyright IBM Corporation 2008. All rights reserved

__ f. Click **Finish**

__ g. Wait for the build process to complete for the imported projects, which is seen at the lower right corner of the WebSphere Integration Developer work bench.



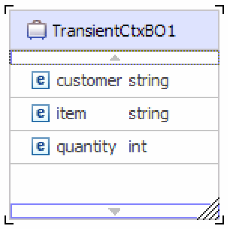
Building workspace: (44%)

Part 2: Authoring the mediation flow for message augmentation

What you will do in this part: In this part of the lab you will develop the mediation flow used to augment the message through use of a service invoke primitive. This will involve using a service invoke primitive with XSL transformation primitives before and after. See the presentation entitled Augmentation, aggregation and retry tutorials to better understand what this part is doing.

____ 1. Create a new business object that is used as the transient context for the flow. This is used to temporarily store information that otherwise would be lost during the flow.

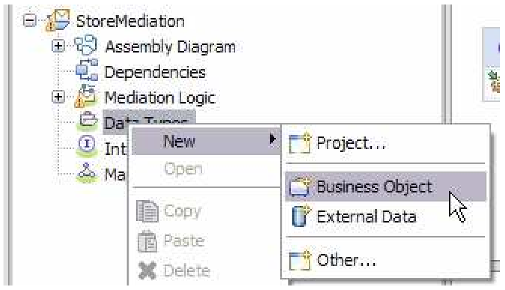
```
•  Module     : StoreMediation
•  Name       : TransientCtxBO1
•  Attributes: customer    string
•                    item        string
•                    quantity    int
```


TransientCtxBO1

customer	string
item	string
quantity	int

• ---

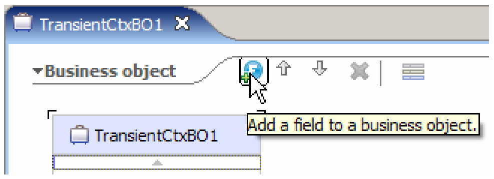
__ **a.** In the Business Integration view, expand **StoreMediation**, right click **Data Types** and then select **New → Business Object** from the pop-up menu



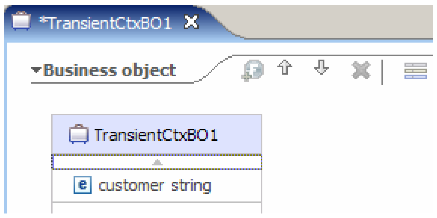
__ b. In the **New Business Object** wizard, provide the name **TransientCtxBO1**

__ c. Click **Finish**. The business object editor opens

__ d. In the editor, click the **Add a field to a business object** icon

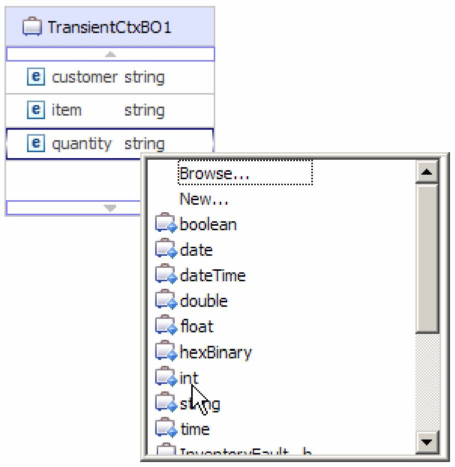


__ e. Edit the new field to have a name of **customer** and a type of **string**



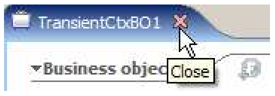
__ f. In the same way, add the field **item** of type **string**

__ g. Also add the field **quantity** of type **int**, using the drop down box to select the type

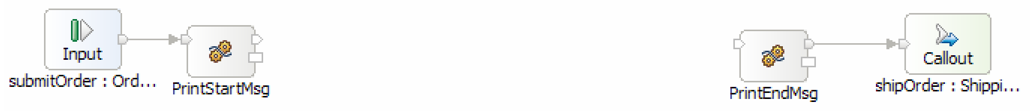


__ h. From the menu select **File → Save** (or **Ctrl + S** from your keyboard) to save your changes

__ i. Click on **X** to close TransientCtxBO1, business object editor



____ 2. Open the StoreMediation flow found in the StoreMediation module. The flow already contains two primitives that are used to write start and end messages to the log.

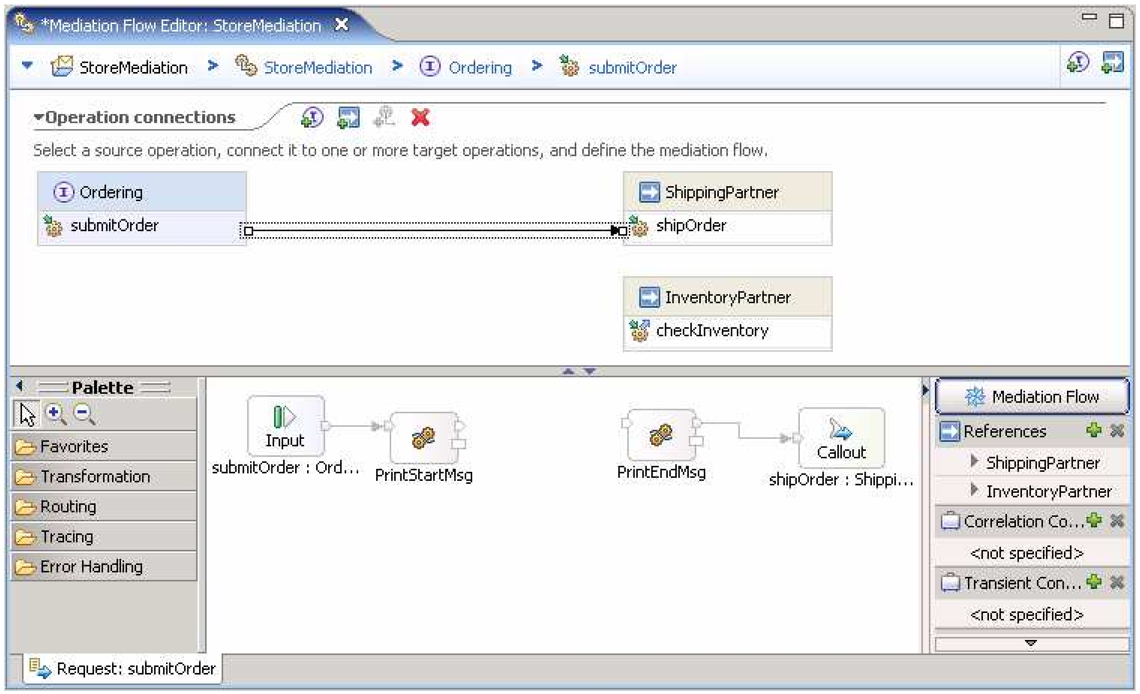


• --

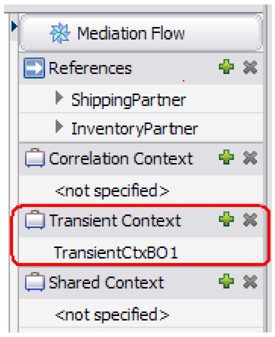
__ a. In the Business Integration view, expand **StoreMediation → Mediation Logic → Flows** and then double-click on **StoreMediation** to open it in the mediation flow editor

© Copyright IBM Corporation 2008. All rights reserved

__ b. The line connecting **Ordering → submitOrder** and **ShippingPartner → shipOrder** should be selected for you to view the flow:

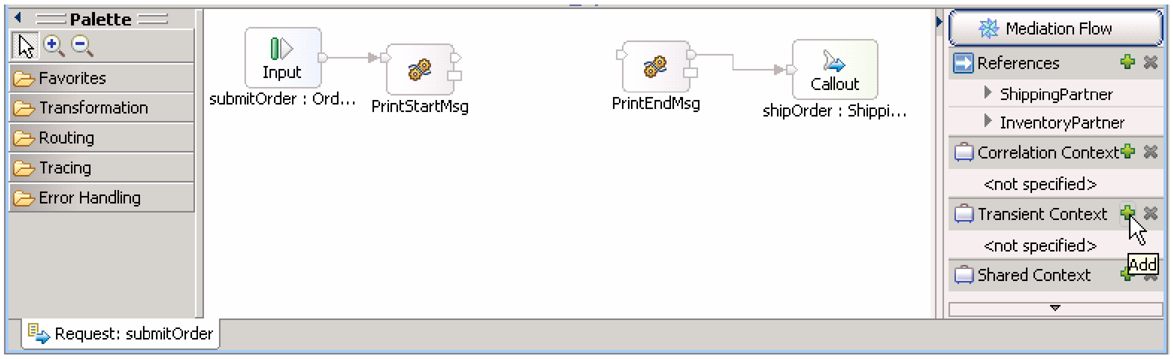


____ 3. In the right side of the mediation flow editor, configure the transient context for the flow to use the TransientCtxBO1.

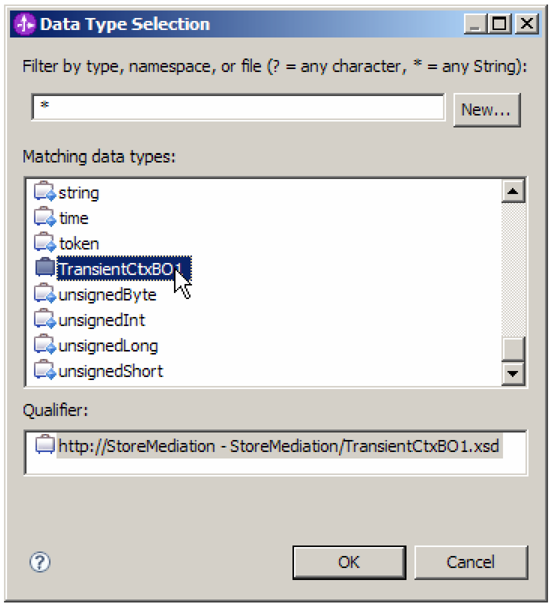


- ---

__ a. In the Mediation Flow on the right side, click the **Add** (✚) icon next to **Transient Context**



__ b. In the Data Type Selection window, scroll down to select **TransientCtxBO1** under **Matching data types:**


Data Type Selection

Filter by type, namespace, or file (? = any character, * = any String):

*

Matching data types:

string
time
token
TransientCtxBO1
unsignedByte
unsignedInt
unsignedLong
unsignedShort

Qualifier:

http://StoreMediation - StoreMediation/TransientCtxBO1.xsd

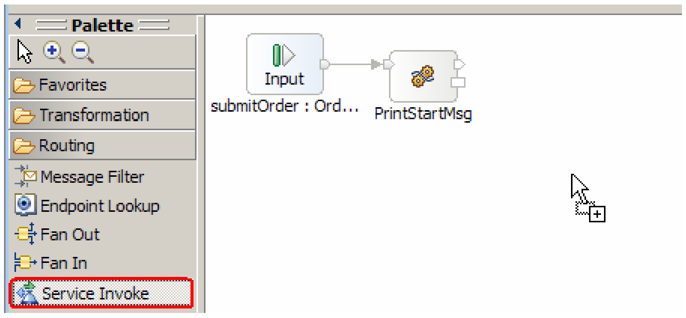
__ c. Click **OK**

__ d. From the menu, select **File → Save** (or **Ctrl + S** from your keyboard) to save your changes

____ 4. Add and configure a service invoke primitive to the canvas. This is used to call the inventory service which provides inventory status information about the order item.

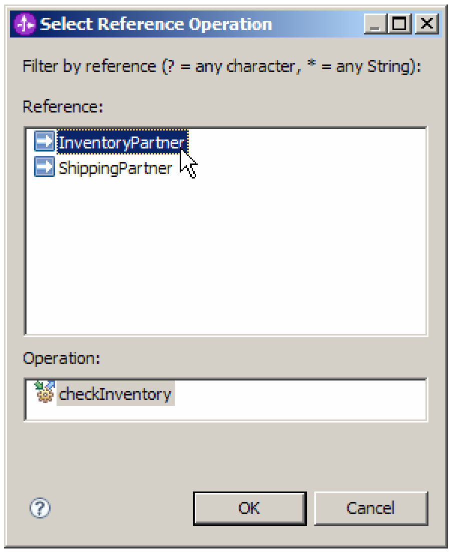
- **`Reference:  InventoryPartner`**
- **`Operation:  checkInventory`**
- **`Display name:     CheckInventory`**
- **`Dynamic endpoint: Uncheck “Use dynamic endpoint if set in message header” option on Details panel`**
- **`-----------------------------------------------------------------`**

__ a. From the **Palette**, select **Routing → Service Invoke** and then click on the canvas to drop the primitive



© Copyright IBM Corporation 2008. All rights reserved

__ b. In the Select Reference Operation window, select **InventoryPartner** under **Reference** list. Note that the selected **Operation** is **checkInventory**



Filter by reference (? = any character, * = any String):

Reference:

InventoryPartner
ShippingPartner

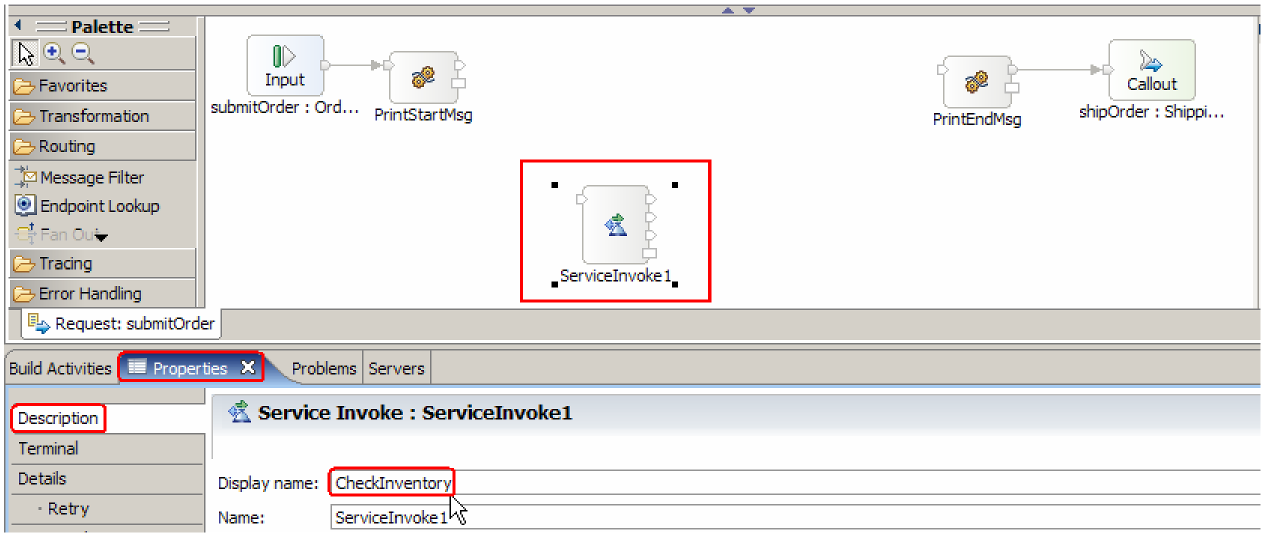
Operation:

checkInventory

__ c. Click **OK**

__ d. You will now see a new service inventory primitive added to the flow. Ensure that the service invoke primitive is selected

__ e. Select **Properties → Description** and change the **Display name** to **CheckInventory**



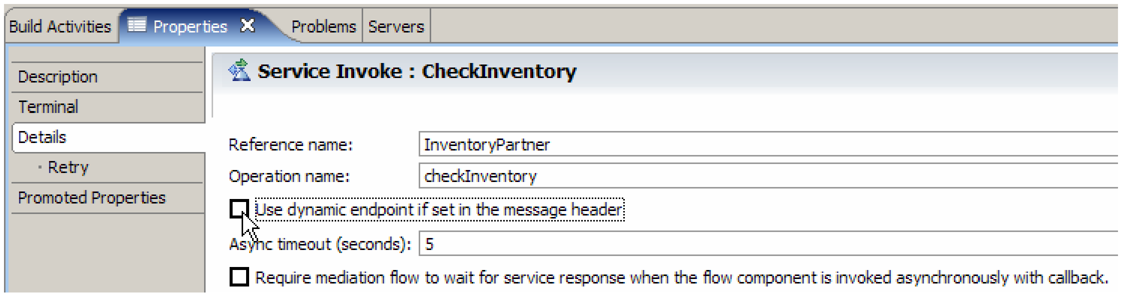
Service Invoke : ServiceInvoke1

Display name: CheckInventory

Name: ServiceInvoke1

__ f. For the same service invoke primitive, select **Properties → Details**

__ g. **Clear** the check box next to '**Use dynamic endpoint if set in the message header**'



Service Invoke : CheckInventory

Reference name: InventoryPartner

Operation name: checkInventory

☐ Use dynamic endpoint if set in the message header

Async timeout (seconds): 5

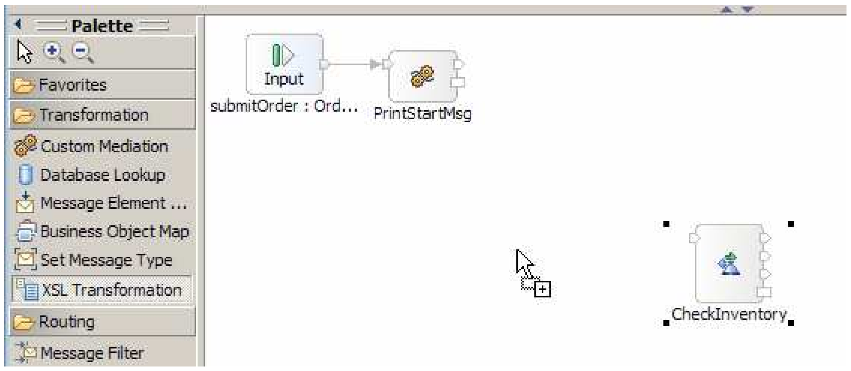
☐ Require mediation flow to wait for service response when the flow component is invoked asynchronously with callback.

__ h. From the menu, select **File → Save** (or **Ctrl + S** from your keyboard) to save your changes

____ 5. Add an XSL transformation primitive to the canvas and wire it into the flow before the CheckInventory, service invoke primitive. This primitive provides the required transformations to call the inventory service.

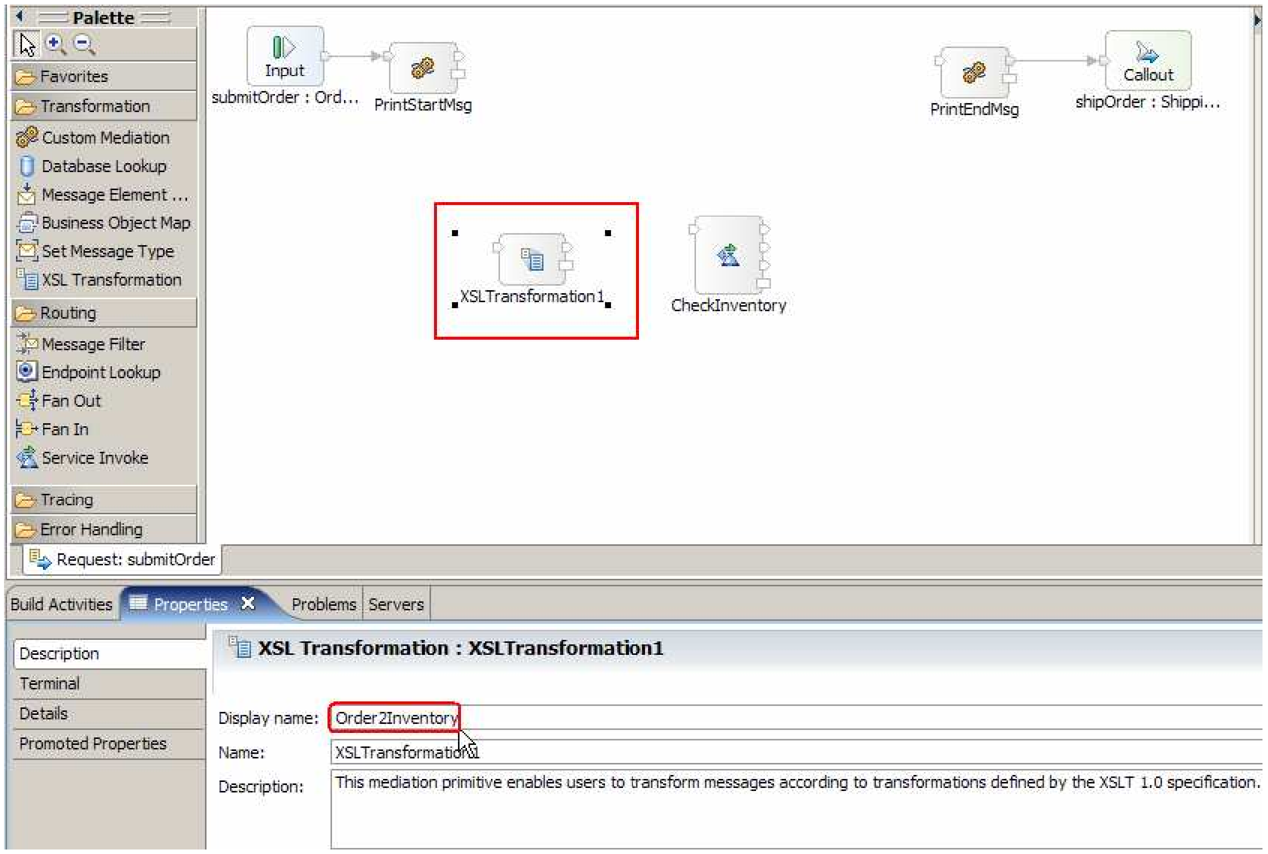
- **`Display name: Order2Inventory`**
- **`Wire         : PrintStartMsg to Order2Inventory`**
- **`Wire         : Order2Inventory to CheckInventory`**
- **`-------------------------------------------------------------------`**

__ a. From the **Palette**, select **Transformation → XSL Transformation** and then click on the canvas to drop the primitive before the CheckInventory primitive



__ b. You will now see a new XSL Transformation primitive added to the flow. Ensure that this primitive is highlighted and select **Properties → Description**

__ c. Change the **Display name** to **Order2Inventory**



XSL Transformation : XSLTransformation1

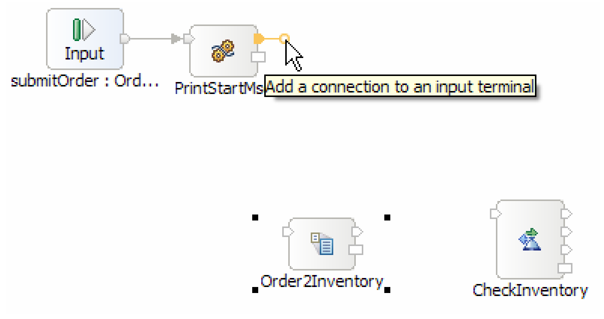
Display name: Order2Inventory

Name: XSLTransformation1

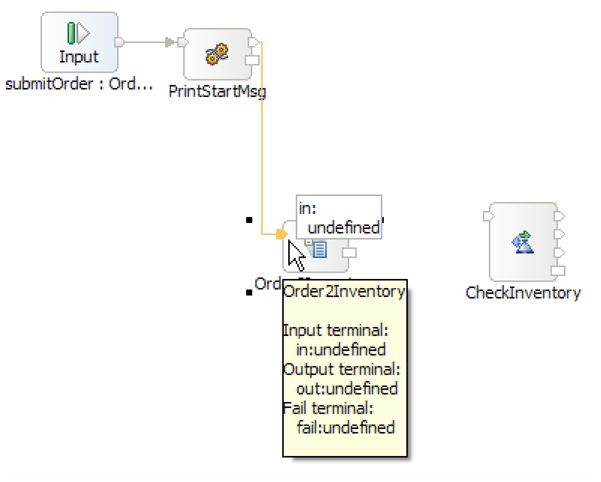
Description: This mediation primitive enables users to transform messages according to transformations defined by the XSLT 1.0 specification.

__ d. Add a connection from **PrintStartMsg** to **Order2Inventory** primitive

1) Hover the mouse over **out** terminal of **PrintStartMsg** to get the 'Add a connection to an input terminal' (the orange bubble shown when hovering over the terminal)



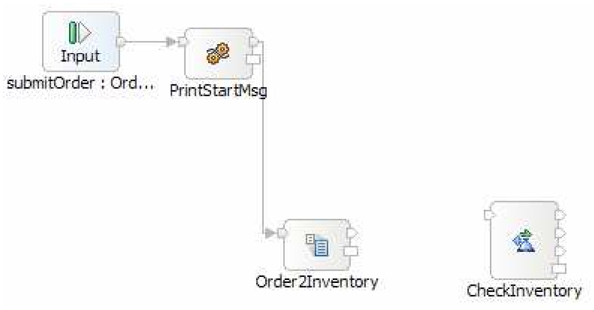
2) Click on the “Add a connection to an input terminal’ orange bubble and then click on **Order2Inventory** primitive



Order2Inventory

Input terminal:
in:undefined
Output terminal:
out:undefined
Fail terminal:
fail:undefined

3) You should now see a connection from PrintStartMsg to Order2Inventory primitive

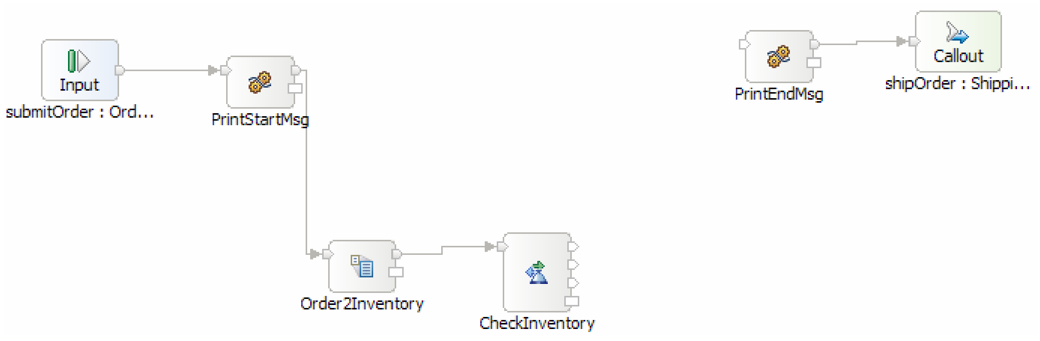


__ e. Similarly, add a connection from **Order2Inventory** primitive to **CheckInventory** primitive

1) Hover your mouse over **out** terminal of **Order2Inventory** to get the ‘Add a connection to an input terminal’

2) Click on the orange bubble and then click on **CheckInventory** primitive

__ f. Your flow now should have these connections:



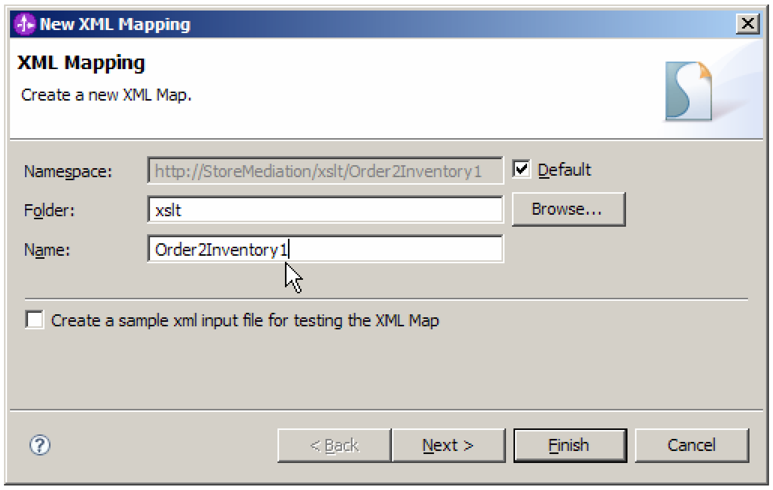
____ 6. In the Order2Inventory XML transformation primitive, create a new XML mapping file used to define the transformation.

- **`Map Name            :    Order2Inventory1`**
- **`Message Root        :    /`**
- **`Input Message Body :    submitOrderRequestMsg`**
- **`Output Message Body:    checkInventoryRequestMsg`**
- **`----------------------------------------------------------------------`**

__ a. Select **Order2Inventory** primitive from the canvas and then select **Properties → Details**

__ b. Click on **New…** next to **Mapping file**. The 'New XML Mapping' wizard opens

__ c. For **Name** enter **Order2Inventory1**



New XML Mapping

XML Mapping

Create a new XML Map.

Namespace: http://StoreMediation/xslt/Order2Inventory1 [✓ Default]

Folder: xslt

Name: Order2Inventory1

[ ] Create a sample xml input file for testing the XML Map

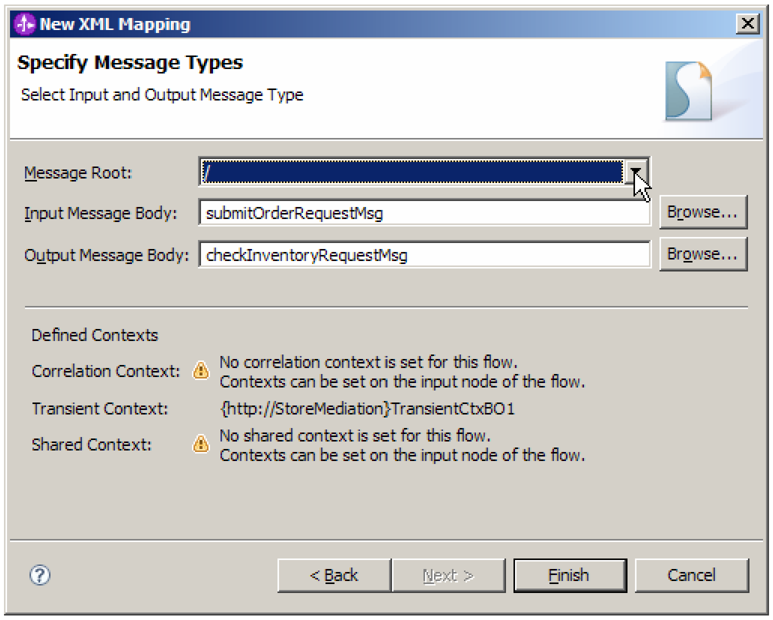
__ d. Click **Next**

__ e. From the Specify Message Types panel, enter :

1) For **Message Root**, select '**/**' from the drop down list

2) For **Input Message Body**, accept the default selection: **sumbitOrderRequestMsg**

3) For **Output Message Body**, accept the default selection: **checkInventoryRequestMsg**



Message Root: /
Input Message Body: submitOrderRequestMsg
Output Message Body: checkInventoryRequestMsg

Defined Contexts
Correlation Context: No correlation context is set for this flow. Contexts can be set on the input node of the flow.
Transient Context: {http://StoreMediation}TransientCtxBO1
Shared Context: No shared context is set for this flow. Contexts can be set on the input node of the flow.

__ f. Click **Finish**. The Order2Inventory1.map file is opened in the XML Mapping editor.

____ 7. Define the mapping for the Order2Inventory1 map. The mapping must address: (1) Moving fields between the source and target SMOs that will not change. (2) Saving information in the transient context from the source message body that needed later in the flow. (3) Setting up the target message body needed to call the inventory service.

NOTE: Since this is the first instance in this lab of using the new XML mapping editor, there are a few things that are pointed out in context below. These additional notes is not repeated in later instances of creating XML maps.

- **Use toolbar icon ( ) 'Map source to target based on name and types' which will map the context and headers using inline maps**

NOTE: This creates a nested set of inline maps for all matching elements in the source and target. In this case, the SMO bodies are different so no inline maps are created for the body, but nested inline maps are created for the context and headers which have the same elements in source and target.

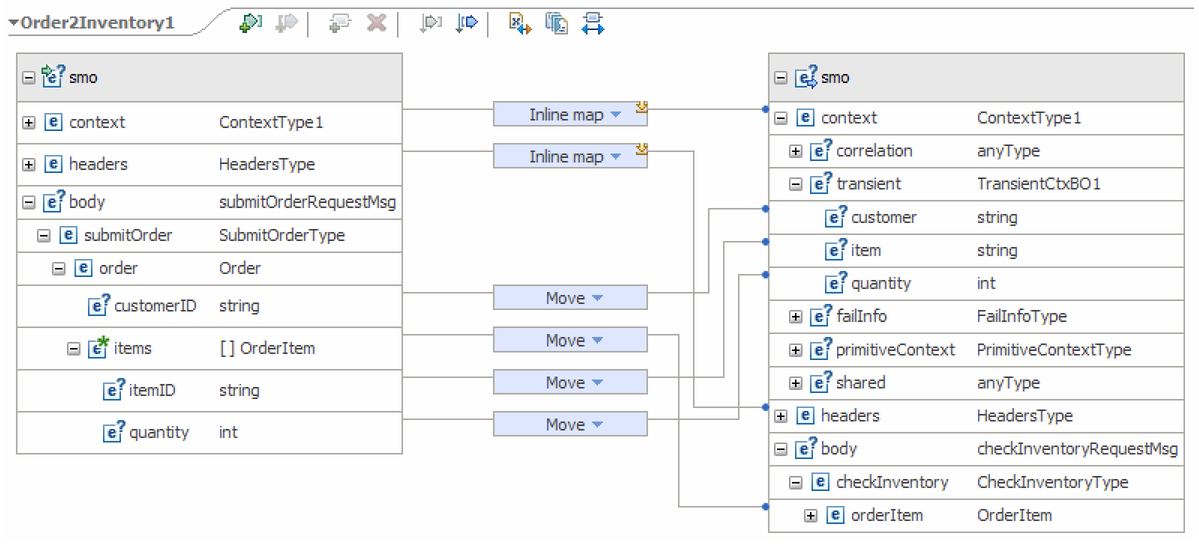
- **Navigate down into the context/transient to context/transient inline map and remove all of the move transforms**

NOTE: You will have to navigate down two levels of inline maps. First the inline map for context to context from the SMO level, and then the inline map for transient to transient from the context level. Use the **Edit** ( ) icon on an inline map transform to drop down into the inline map, and use the **Up a level** ( ) icon from an inline map to move up to its containing map. The reason you need to remove the move transforms is that you set these transient context target fields from different fields in the source.

- **At the SMO level (top level map) add these move transforms**

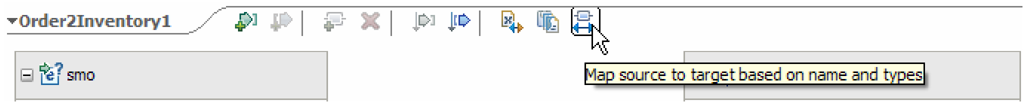
NOTE: In the XPath expressions, if there is a specific index for an element in an array (such as **items[1]**), you must use the **Properties → Cardinality** view to specify the index.

Source (left side)	Target (right side)
body/submitOrder/order/customerID	context/transient/customer
body/submitOrder/order/items[1]/itemID	context/transient/item
body/submitOrder/order/items[1]/quantity	context/transient/quantity
body/submitOrder/order/items[1]	body/checkInventory/orderItem

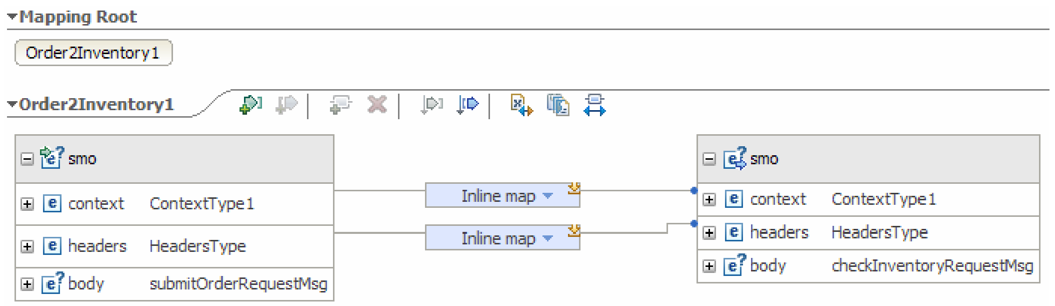


• -

__ a. Click on '**Map source to target based on name and types**' icon ( )



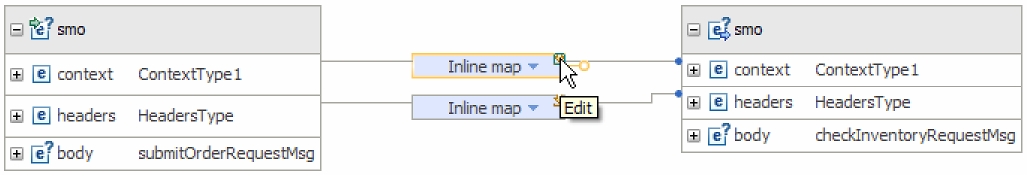
__ b. This will map context and headers using inline maps as shown below:



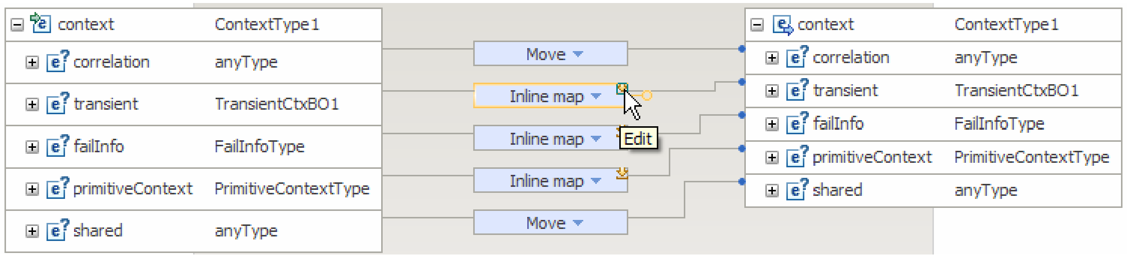
__ c. Navigate down into the context/transient to context/transient inline map and remove all of the move transformations

1) Click on **Edit** ( ) icon on the Inline map connecting **context** attributes

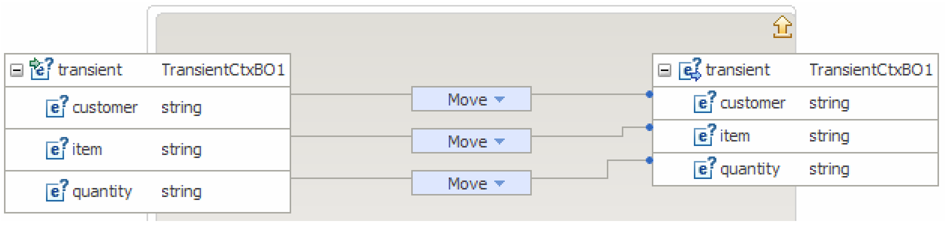
© Copyright IBM Corporation 2008. All rights reserved



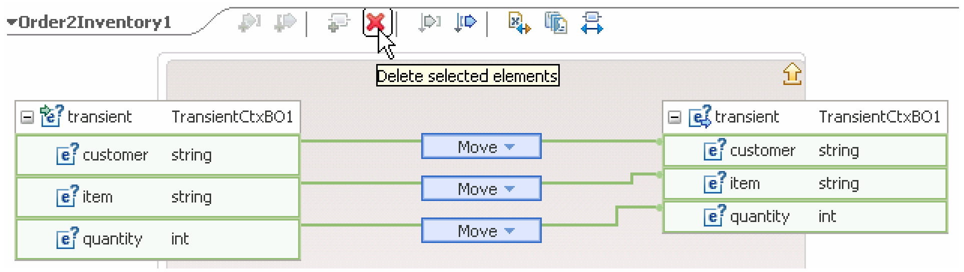
2) This will open all the transformations defined under context. Click on **Edit** ( ) icon on the Inline map connecting **transient** attributes:



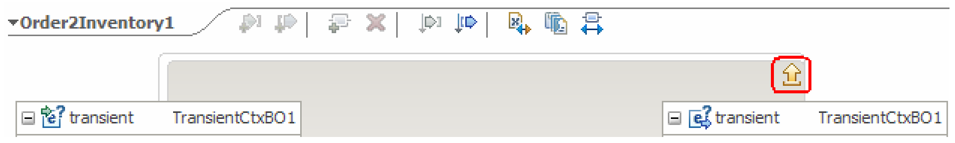
3) This will open the transformations of transient:



4) Select all three **Move** transformations (hold down Ctrl button on keyboard and click on each Move transform) and hit '**Delete selected elements**' ( ) icon from the top:



5) Click on '**Up a level**' ( ) icon which will bring you one level up to the context mapping



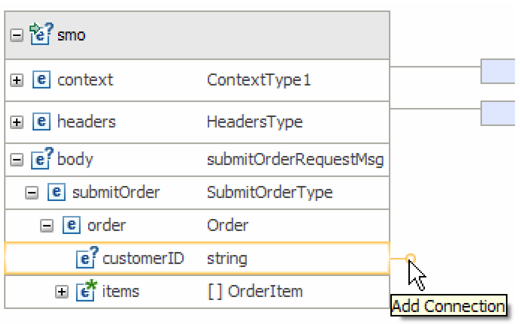
6) Click on '**Up a level**' ( ) icon one more time which will now bring you back the smo mapping level

__ d. Move body/submitOrder/order/customerID to context/transient/customer

1) In the left smo, expand **body → submitOrder → order**

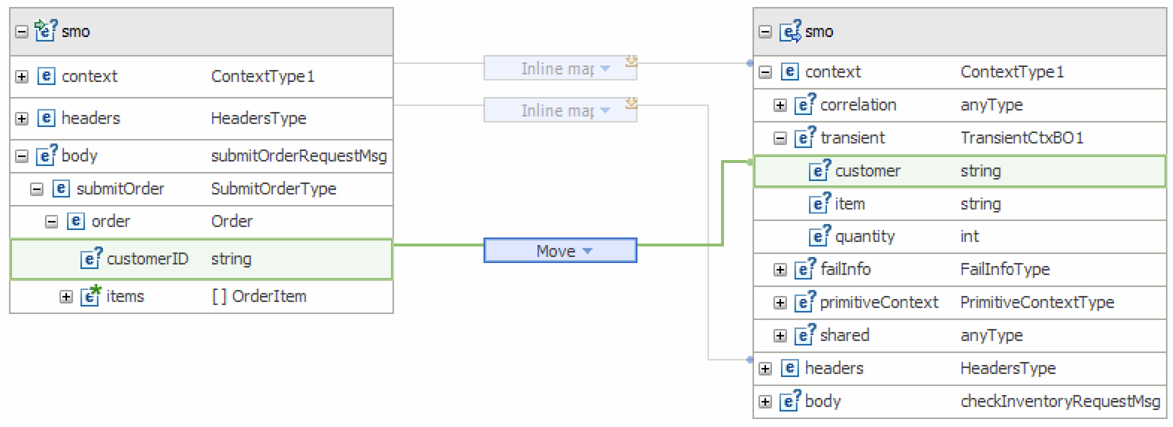
2) In the right smo, expand **context → transient**

3) In the left smo, hover your mouse over **customerID** and click on **Add Connection**:



smo	
context	ContextType1
headers	HeadersType
body	submitOrderRequestMsg
submitOrder	SubmitOrderType
order	Order
customerID	string
items	[ ] OrderItem

4) Now click on **customer** in the right smo and you should see a new connection from customerID to customer:



smo	
context	ContextType1
headers	HeadersType
body	submitOrderRequestMsg
submitOrder	SubmitOrderType
order	Order
customerID	string
items	[ ] OrderItem

smo	
context	ContextType1
correlation	anyType
transient	TransientCtxBO1
customer	string
item	string
quantity	int
failInfo	FailInfoType
primitiveContext	PrimitiveContextType
shared	anyType
headers	HeadersType
body	checkInventoryRequestMsg

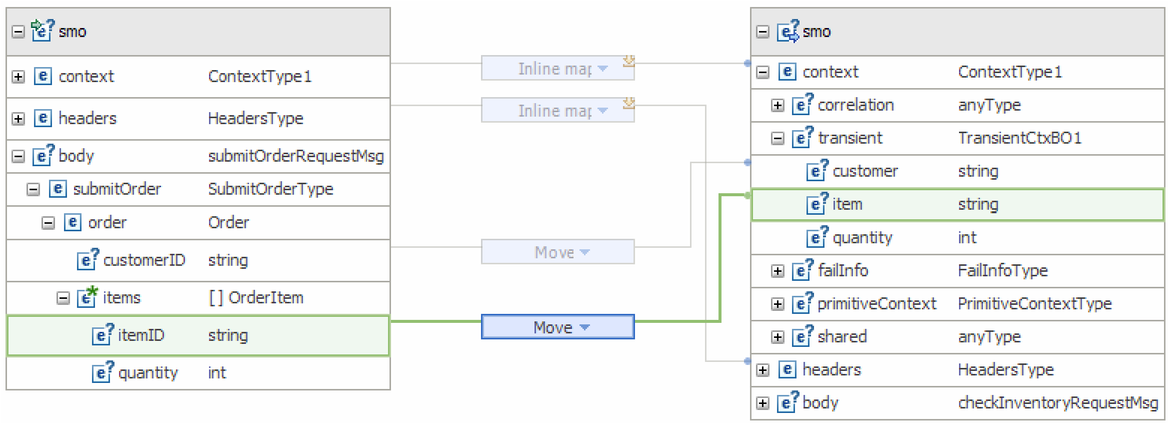
__ e. Similarly move body/submitOrder/order/items[1]/itemID to context/transient/item

1) In the left smo, expand **body → submitOrder → order → items**

2) In the right smo, expand **context → transient**

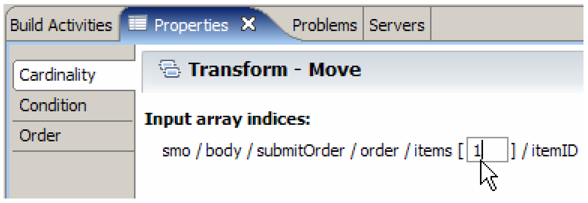
3) In the left smo, hover your mouse over **itemID** and click on **Add Connection**

4) Now click on **item** in the right smo and you should see a new connection from itemID to item



5) Ensure that the above defined Move is highlighted and then select **Properties → Cardinality**

6) Enter '**1**' in the **Input array indices**


Build Activities | Properties | Problems | Servers

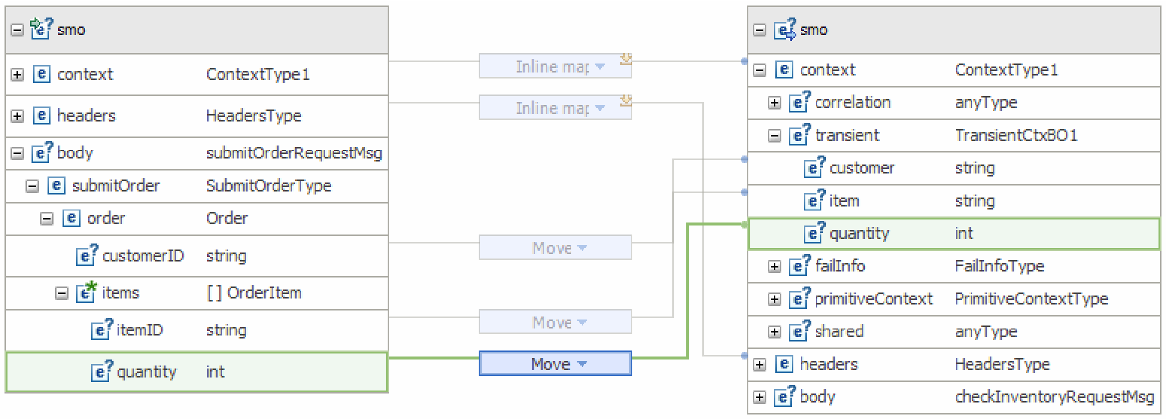
Cardinality
Condition
Order

Transform - Move

Input array indices:

smo / body / submitOrder / order / items [1] / itemID

__ f. Similarly move body/submitOrder/order/items[1]/quantity to context/transient/quantity:



1) Ensure that the above defined Move is highlighted and then select **Properties → Cardinality**

2) Enter '**1**' in the **Input array indices**

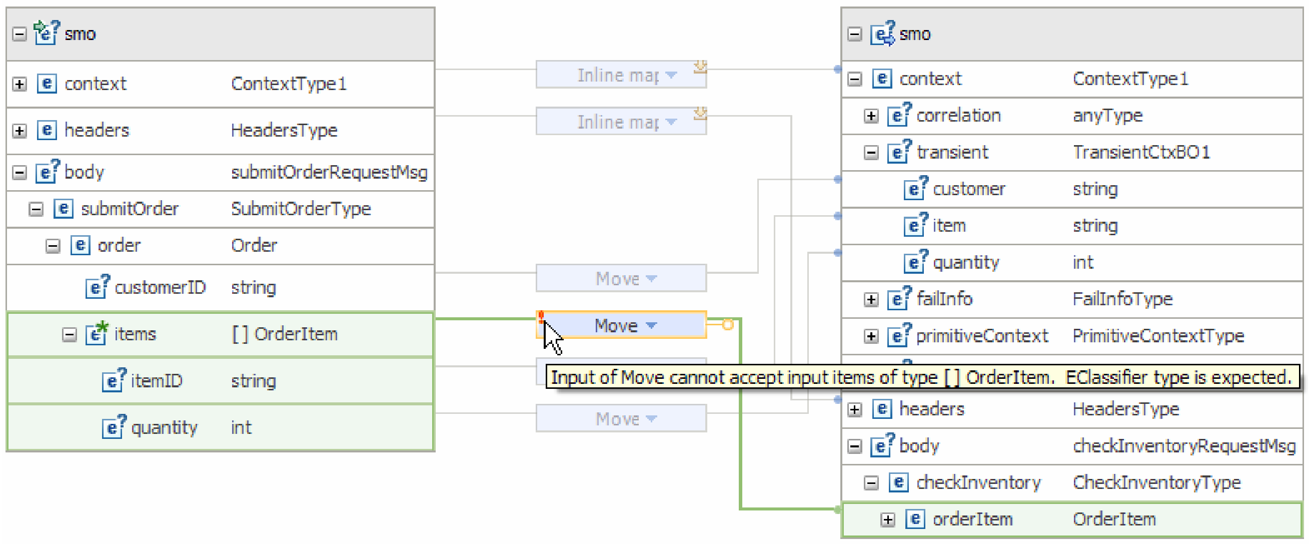
__ g. Now move body/submitOrder/order/items[1] to body/checkInventory/orderItem

1) In the left smo, expand **body → submitOrder → order → items**

2) In the right smo, expand **body → checkInventory → orderItem**

3) In the left smo, hover your mouse over **items** and click on **Add Connection**

4) Now click on **orderItem** in the right smo and you should see a new connection from items to orderItem with a **flag**:



Input of Move cannot accept input items of type [ ] OrderItem. EClassifier type is expected.

5) Ensure that the above defined Move is highlighted and then select **Properties → Cardinality**

6) Enter '**1**' in the **Input array indices**. The flag will get resolved once you enter the Input array indices

__ h. From the menu select **File → Save** (or **Ctrl + S** from your keyboard) to save your changes to the map

__ i. Click on **X** to close Order2Inventory1.map file

__ j. From the menu select **File → Save** (or **Ctrl + S** from your keyboard) to save your changes to the mediation flow

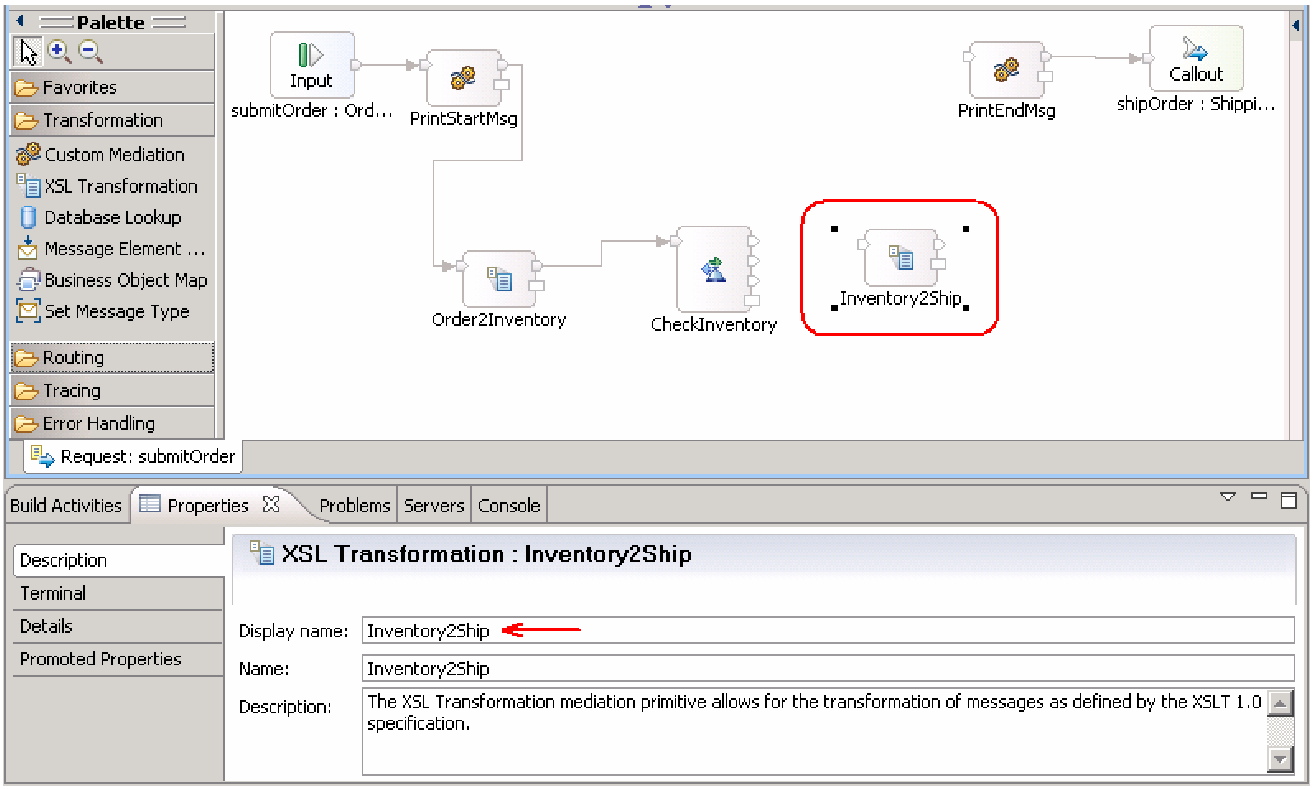
____ 8. Add an XSL transformation primitive to the canvas and wire it into the flow following the CheckInventory primitive. This primitive provides the required transformations to call the shipping service.

- **`Display Name: Inventory2Ship`**
- **`Wire         : CheckInventory(out terminal) to Inventory2Ship`**
- **`Wire         : Inventory2Ship to PrintEndMsg`**
- **`-------------------------------------------------------------------`**

__ a. From the **Palette**, select **Transformation → XSL Transformation** and then click on the canvas to drop the primitive to the right side of CheckInventory primitive

__ b. You will now see a new XSL Transformation primitive added to the flow. Ensure that this primitive is highlighted and select **Properties → Description**

__ c. Change the **Display name** to **Inventory2Ship**



XSL Transformation : Inventory2Ship

Display name: Inventory2Ship

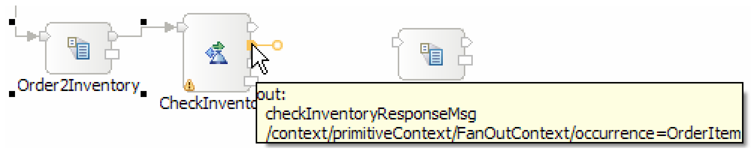
Name: Inventory2Ship

Description: The XSL Transformation mediation primitive allows for the transformation of messages as defined by the XSLT 1.0 specification.

__ d. Add a connection from **CheckInventory** to **Inventory2Ship** primitive

1) Hover your mouse over the **out** terminal of **CheckInventory** to get the 'Add a connection to an input terminal'

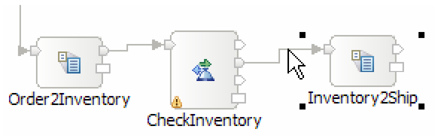
NOTE: The top two terminals, **out** and **timeout**, do not always appear in the same order on the primitive. Ensure you are adding the connection from the **out** terminal.



out:
checkInventoryResponseMsg
/context/primitiveContext/FanOutContext/occurrence=OrderItem

2) Click on the orange bubble and then click on **Inventory2Ship** primitive

3) You should now see a connection from CheckInventory to Inventory2Ship primitive



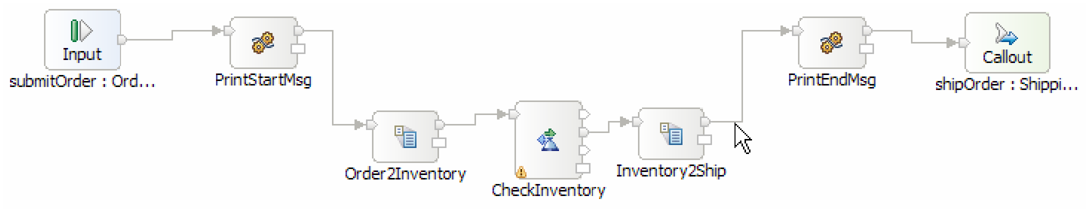
__ e. Similarly, add a connection from **Inventory2Ship** primitive to **PrintEndMsg** primitive

1) Hover your mouse over **out** terminal of **Inventory2Ship** to get the 'Add a connection to an input terminal'

© Copyright IBM Corporation 2008. All rights reserved

2) Click on the orange bubble and then click on **PrintEndMsg** primitive

__ f. Your flow now should have these connections:



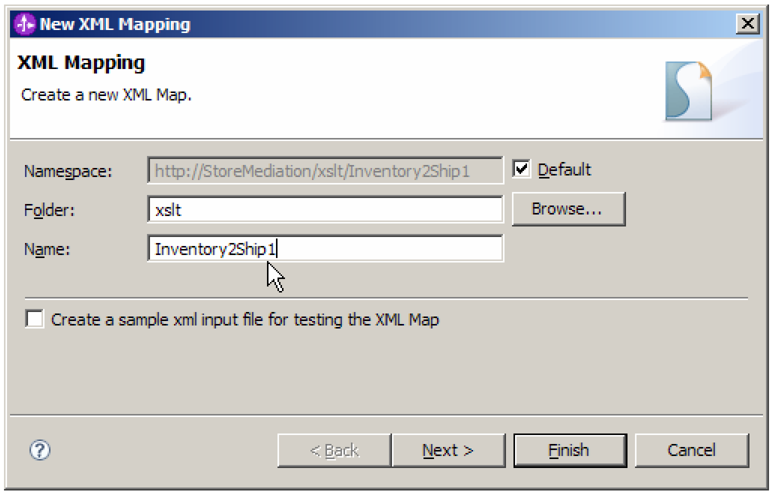
____ 9. In the Inventory2Ship XML transformation primitive, create a new XML mapping file used to define the transformation.

- **`Map Name            :    Inventory2Ship1`**
- **`Message Root        :    /`**
- **`Input Message Body  :    checkInventoryResponseMsg`**
- **`Output Message Body:    shipOrderRequestMsg`**
- **`--------------------------------------------------------------------`**

__ a. Select **Inventory2Ship** primitive from the canvas and then select **Properties → Details**

__ b. Click on **New…** next to **Mapping file**. The 'New XML Mapping' wizard opens.

__ c. For **Name** enter **Inventory2Ship1**


New XML Mapping

XML Mapping

Create a new XML Map.

Namespace: http://StoreMediation/xslt/Inventory2Ship1 ☑ Default

Folder: xslt

Name: Inventory2Ship1

☐ Create a sample xml input file for testing the XML Map

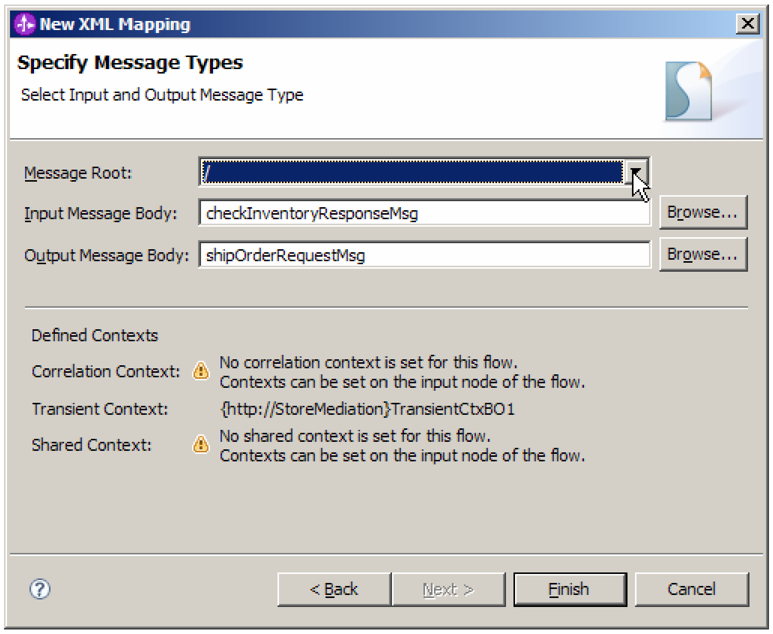
__ d. Click **Next**

__ e. From the Specify Message Types panel, enter:

1) For **Message Root**, select '**/**' from the drop down list

2) For **Input Message Body**, accept the default selection: **checkInventoryResponseMsg**

3) For **Output Message Body**, accept the default selection: **shipOrderRequestMsg**



Message Root: /

Input Message Body: checkInventoryResponseMsg

Output Message Body: shipOrderRequestMsg

Correlation Context: No correlation context is set for this flow. Contexts can be set on the input node of the flow.

Transient Context: {http://StoreMediation}TransientCtxBO1

Shared Context: No shared context is set for this flow. Contexts can be set on the input node of the flow.

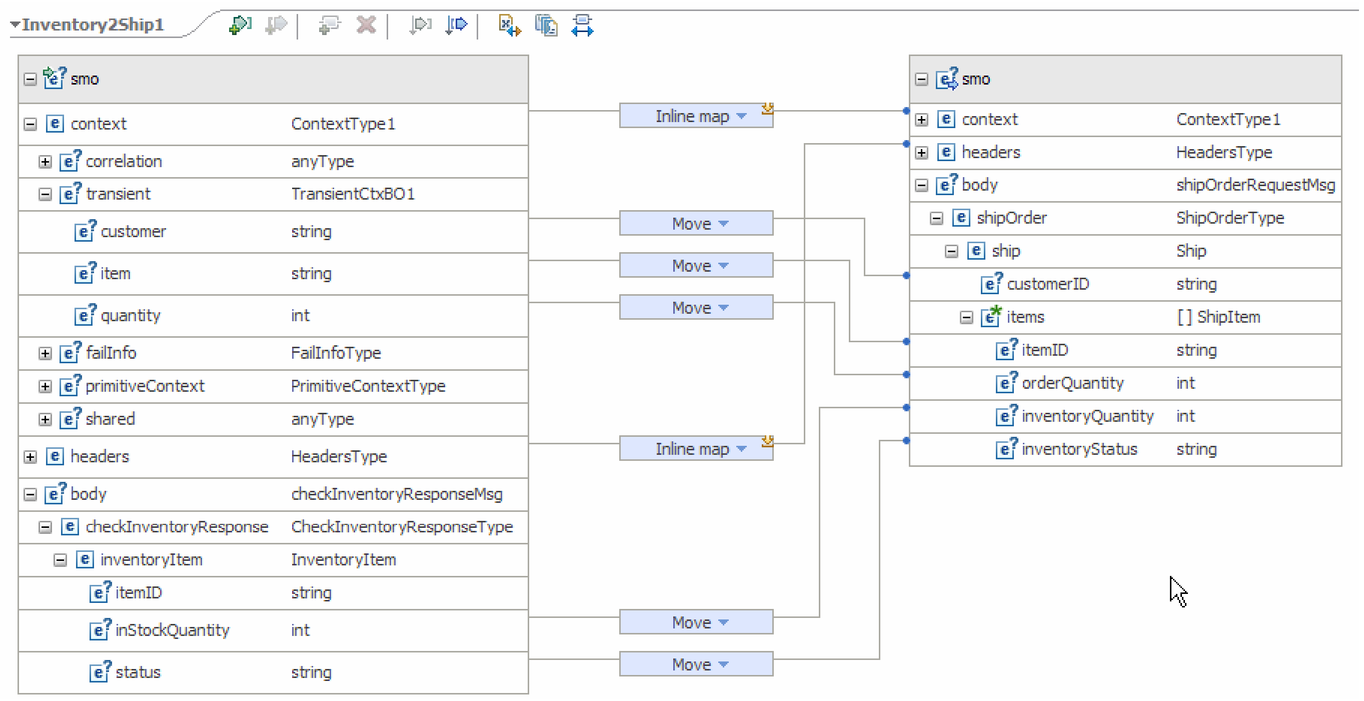
__ f. Click **Finish**. The Inventory2Ship1.map file is opened in the XML Mapping editor

____ 10. Define the mapping for the Inventory2Ship1 map. The mapping must address (1) moving fields between the source and target SMOs that will not change, (2) setting up the target message body needed to call the shipping service.

- **`Use toolbar icon ( ) 'Map source to target based on name and types' which will map the context and headers using inline maps`**
- **`At the SMO level (top level map) add these move transforms`**

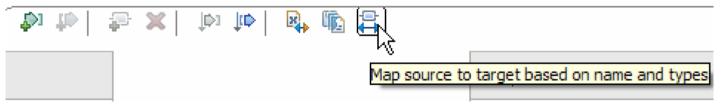
Source (left side)	**Target (right side)**
context/transient/customer	body/shipOrder/ship/customerID
context/transient/item	body/shipOrder/ship/items[1]/itemID
context/transient/quantity	body/shipOrder/ship/items[1]/orderQuantity
body/checkInventoryResponse/inventoryItem/inStockQuantity	body/shipOrder/ship/items[1]/inventoryQuantity
body/checkInventoryResponse/inventoryItem/status	body/shipOrder/ship/items[1]/inventoryStatus

-

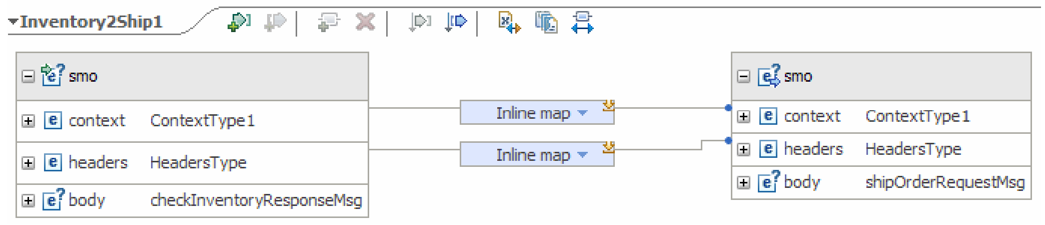


- - -

__ a. Click on '**Map source to target based on name and types**' icon



__ b. This will map context and headers using inline maps as shown below:



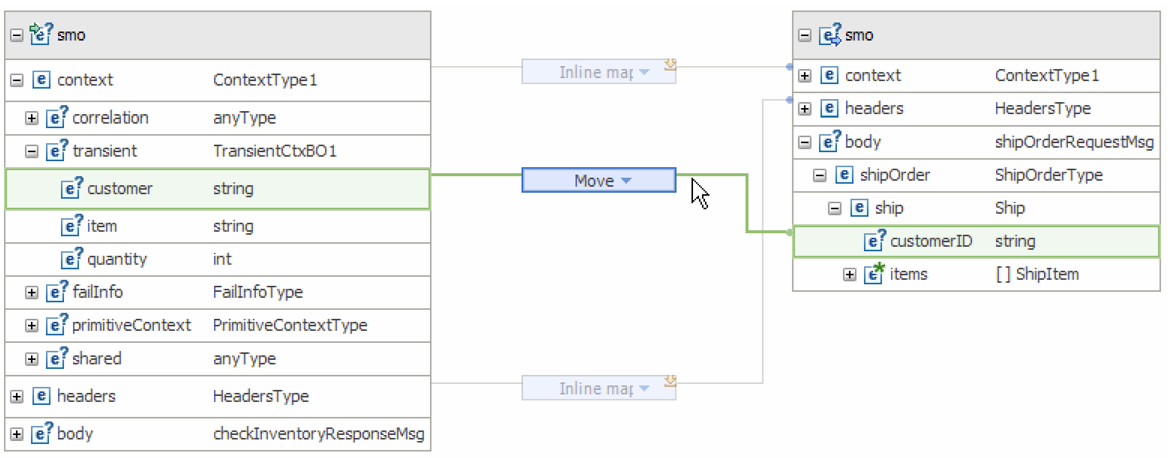
__ c. Move context/transient/customer to body/shipOrder/ship/customerID

1) In the left smo, expand **context → transient**

2) In the right smo, expand **body → shipOrder → ship → customerID**

3) In the left smo, hover your mouse over **customer** and click on **Add Connection**:

4) Now click on **customerID** in the right smo and you should see a new connection from customer to customerID:



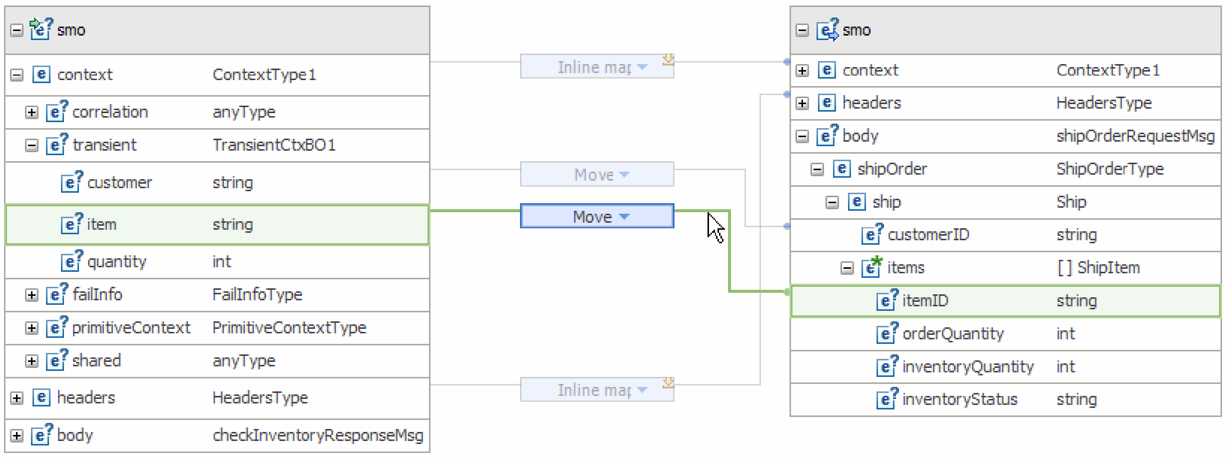
__ d. Move context/transient/item to body/shipOrder/ship/items[1]/itemID

1) In the left smo, expand **context → transient**

2) In the right smo, expand **body → shipOrder → ship → items**

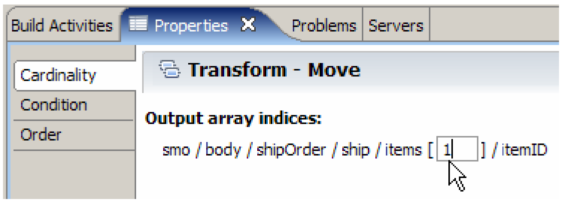
3) In the left smo, hover your mouse over **item** and click on **Add Connection**

4) Now click on **itemID** in the right smo and you should see a new connection from item to itemID



5) Ensure that the above defined Move is highlighted and then select **Properties → Cardinality**

6) Enter '**1**' in the **Output array indices**



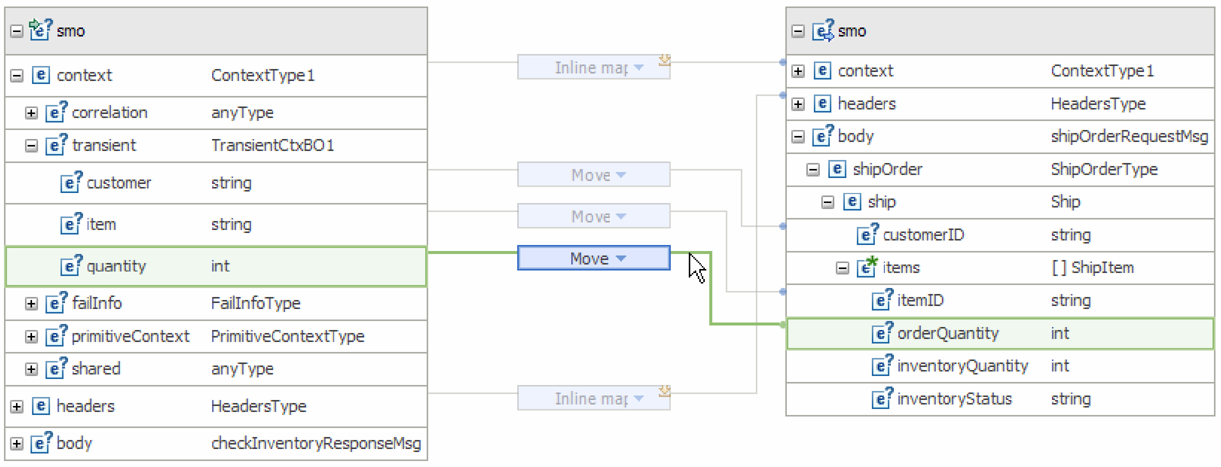
Transform - Move

Output array indices:

smo / body / shipOrder / ship / items [1] / itemID

© Copyright IBM Corporation 2008. All rights reserved

__ e. Similarly, move context/transient/quantity to body/shipOrder/ship/items[1]/orderQuantity:



1) Ensure that the above defined Move is highlighted and then select **Properties → Cardinality**

2) Enter '**1**' in the **Output array indices**

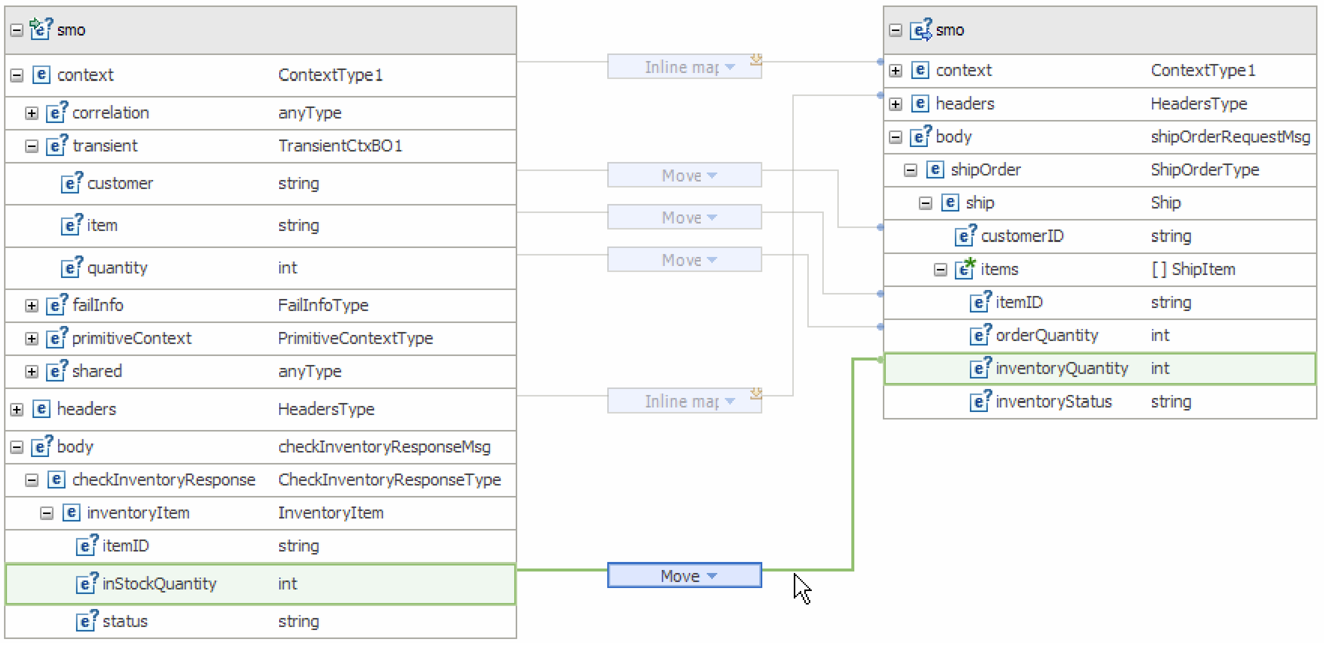
__ f. Now move body/checkInventoryResponse/inventoryItem/inStockQuantity to body/shipOrder/ship/items[1]/inventoryQuantity

1) In the left smo, expand **body → checkInventoryResponse → inventoryItem**

2) In the right smo, expand **body → shipOrder → ship → items**

3) In the left smo, hover your mouse over **inStockQuantity** and click on **Add Connection**

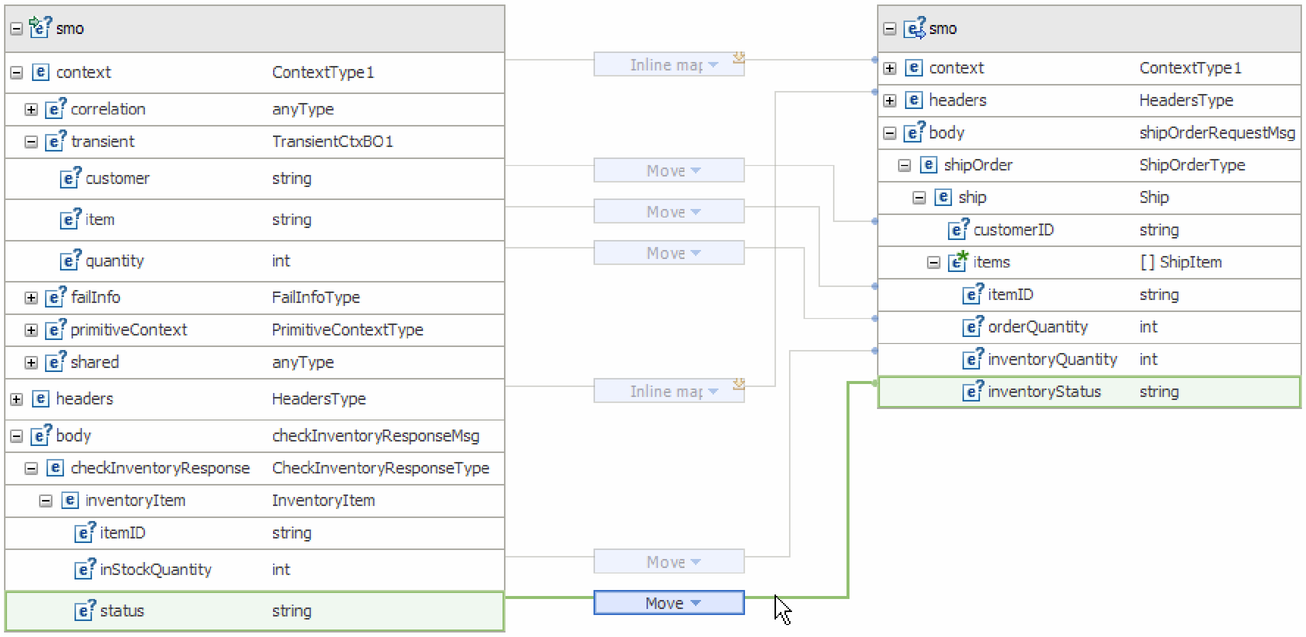
4) Now click on **inventoryQuantity** in the right smo and you should see a new connection from items to orderItem:



5) Ensure that the above defined Move is highlighted and then select **Properties → Cardinality**

6) Enter ‘**1**’ in the **Output array indices**

__ g. Similarly, move body/checkInventoryResponse/inventoryItem/status to body/shipOrder/ship/items[1]/inventoryStatus



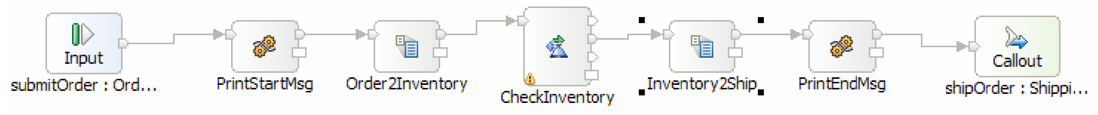
1) Ensure that the above defined Move is highlighted and then select **Properties → Cardinality**

2) Enter ‘**1**’ in the **Output array indices**

__ h. From the menu select **File → Save** (or **Ctrl + S** from your keyboard) to save your changes to the map

__ i. Click on **X** to close Inventory2Ship1.map file

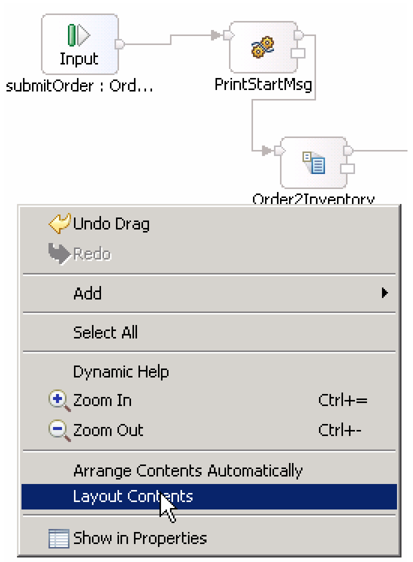
____ 11. Ensure that the resulting flow looks like this.



NOTE: Since WebSphere Integration Developer does not always render the output terminals in a specific order, the wire leaving CheckInventory might visually be coming from what looks like a different terminal.

- --

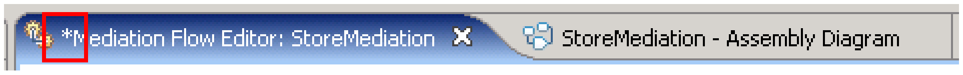
__ a. Right-click on the canvas of mediation flow and select **Layout Contents** from the pop-up menu



____ 12. Check that all the artifacts have been saved.

- --

__ a. Look at the tabs for the various artifact editors. Any tab with an asterisk (*) before the name needs to be saved:

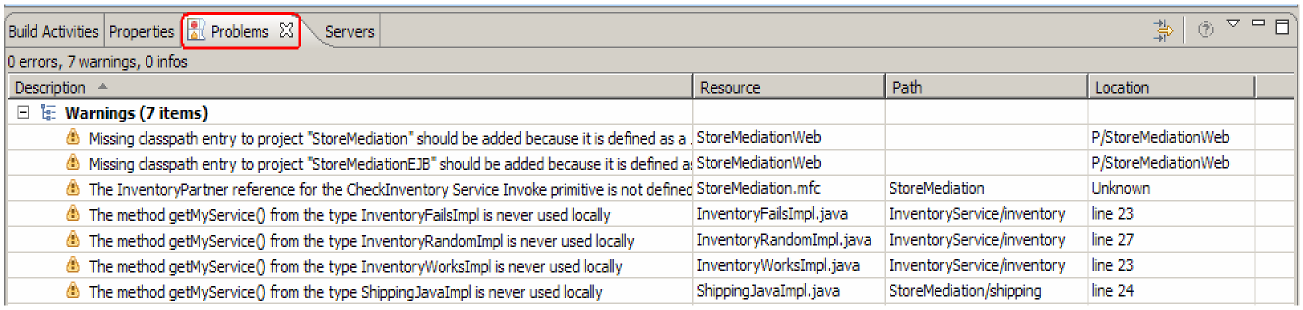

*Mediation Flow Editor: StoreMediation
StoreMediation - Assembly Diagram

__ b. For each tab with an asterisk (*), click on the tab to give it focus and from the menu select **File → Save** (or **Ctrl + S** from your keyboard) to save your changes.

____ 13. Check that there are no errors reported in the Problems view.

- --

__ a. Select **Problems** view at the bottom. You can ignore warnings, but there should not be any errors at this time:



0 errors, 7 warnings, 0 infos

Description	Resource	Path	Location
Warnings (7 items)			
Missing classpath entry to project "StoreMediation" should be added because it is defined as a	StoreMediationWeb		P/StoreMediationWeb
Missing classpath entry to project "StoreMediationEJB" should be added because it is defined a:	StoreMediationWeb		P/StoreMediationWeb
The InventoryPartner reference for the CheckInventory Service Invoke primitive is not defined	StoreMediation.mfc	StoreMediation	Unknown
The method getMyService() from the type InventoryFailsImpl is never used locally	InventoryFailsImpl.java	InventoryService/inventory	line 23
The method getMyService() from the type InventoryRandomImpl is never used locally	InventoryRandomImpl.java	InventoryService/inventory	line 27
The method getMyService() from the type InventoryWorksImpl is never used locally	InventoryWorksImpl.java	InventoryService/inventory	line 23
The method getMyService() from the type ShippingJavaImpl is never used locally	ShippingJavaImpl.java	StoreMediation/shipping	line 24

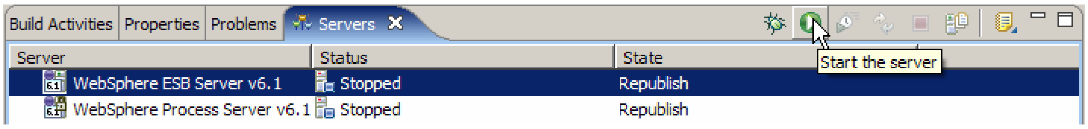
Part 3: Test the message augmentation mediation

What you will do in this part: In this part you use the component test facilities of WebSphere Integration Developer to test the mediation. The inventory service implementation has already been provided and contains some hard-coded inventory data useful for testing. The resulting output from the test is explained. See the presentation entitled Augmentation, aggregation and retry tutorials to better understand what this part is doing.

____ 1. Start the WebSphere Enterprise Service Bus (or WebSphere Process Server) test server.

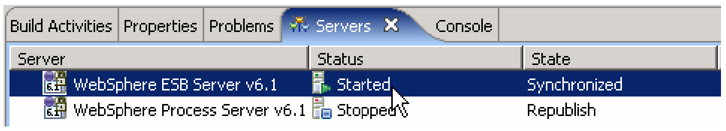
- --

__ a. From the **Servers** view of WebSphere Integration Developer, select **WebSphere ESB Server v6.1** and hit '**Start the server**' icon ( ) from the toolbar



Server	Status	State
WebSphere ESB Server v6.1	Stopped	Republish
WebSphere Process Server v6.1	Stopped	Republish

__ b. Wait until the server Status shows as **Started**



Server	Status	State
WebSphere ESB Server v6.1	Started	Synchronized
WebSphere Process Server v6.1	Stopped	Republish

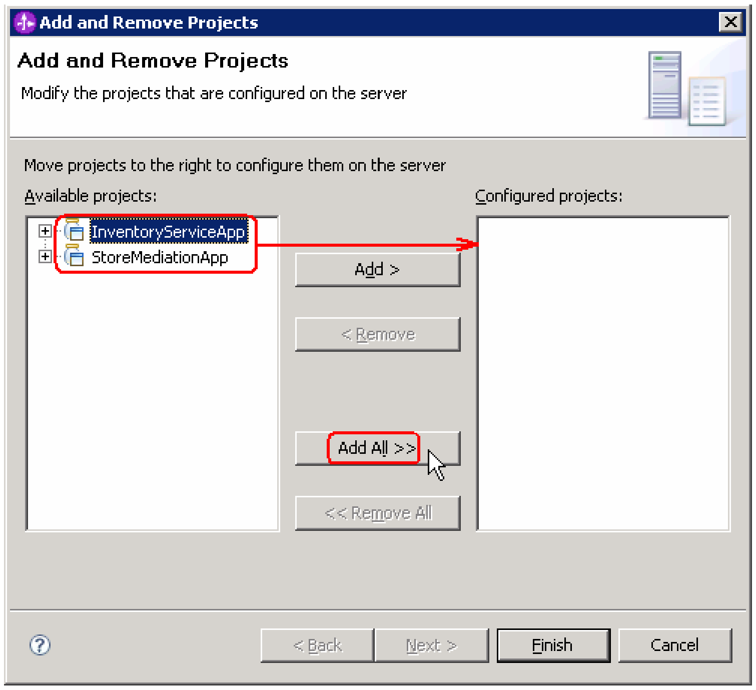
NOTE: Depending upon the preferences you have specified for the **Console** view, the **Console** view might grab the focus from the **Servers** view. If this is the case, the status in the lower right should indicate when the server startup is complete, at which time you can switch back to the **Servers** view

____ 2. Add the InventoryServiceApp and StoreMediationApp to the test server using the Add and Remove Projects dialog from the server pop-up menu.

- --

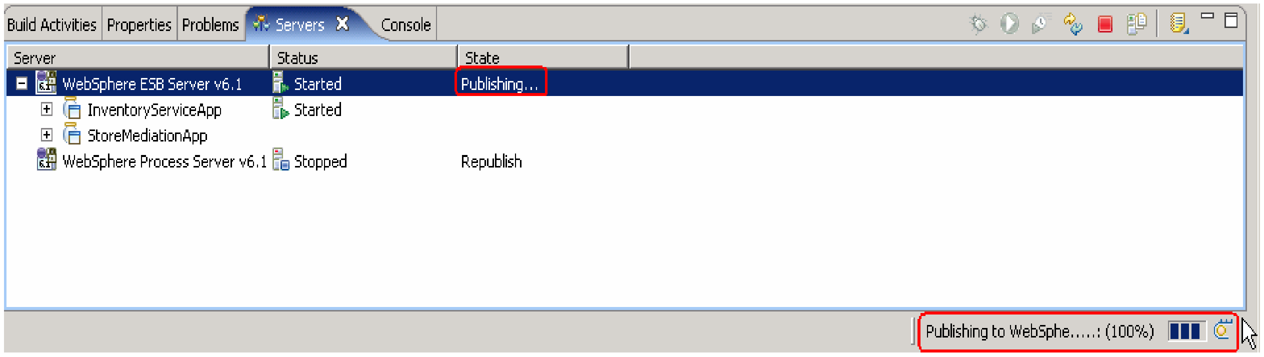
__ a. Right-click on **WebSphere ESB Server v6.1** under the Servers view and select **Add and remove projects…** from the context menu

__ b. In the Add and Remove Projects window, click **Add All >>** to add InventoryServiceApp and StoreMediationApp to the Configured projects panel



__ c. The projects will now be moved to Configured projects. Click **Finish**

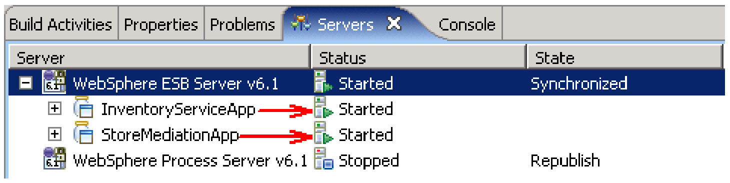
__ d. Wait while the projects are being published to the server



Server	Status	State
WebSphere ESB Server v6.1	Started	Publishing...
InventoryServiceApp	Started	
StoreMediationApp		
WebSphere Process Server v6.1	Stopped	Republish

Publishing to WebSphe.....: (100%)

__ e. Once the publishing is done, from the Servers view, expand **WebSphere ESB Server v6.1** and you should see the 2 applications started as following:



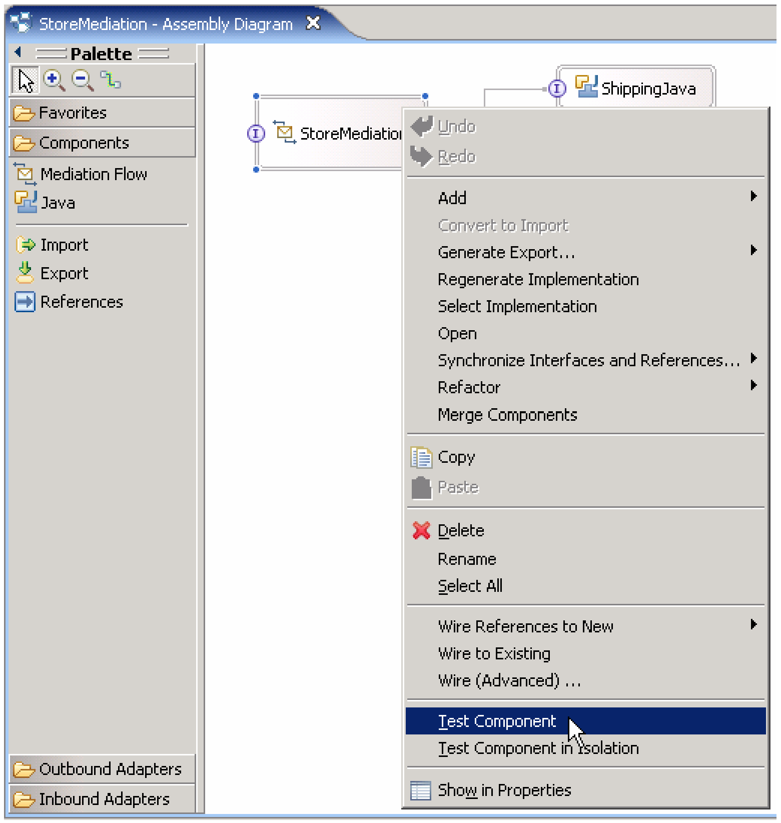
Server	Status	State
WebSphere ESB Server v6.1	Started	Synchronized
InventoryServiceApp	Started	
StoreMediationApp	Started	
WebSphere Process Server v6.1	Stopped	Republish

____ 3. From the assembly diagram for StoreMediation, start Test Component for the StoreMediation component.

- --

__ a. In the Business Integration window, expand **StoreMediation** and double-click on **Assembly Diagram** to open it in Assembly editor

__ b. From the StoreMediation-Assembly Diagram, right-click on **StoreMediation** component and select **Test Component** from the pop-up menu

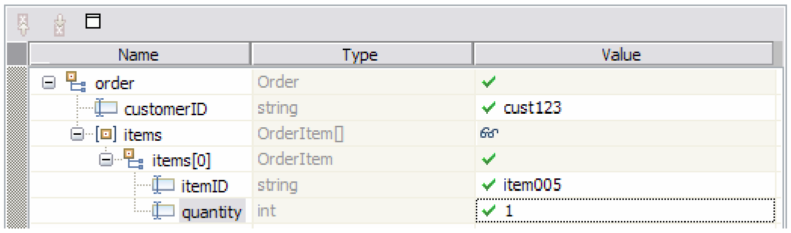


__ c. The **StoreMediation_Test** window is opened where you enter your test data

____ 4. Initialize the test data

- **`Set customerID to cust123`**
- **`Set items/items[0]/itemID to item005`**
- **`Set items/items[0]/quantity to 1`**

NOTE: To create elements in the items array, right click anywhere on the row and select **Add Elements…** from the pop-up menu, then enter **1** for the number of elements to add.



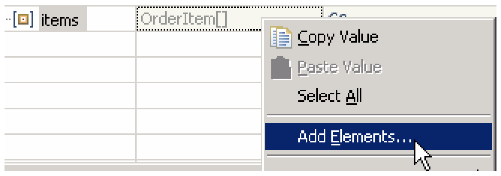
Name	Type	Value
order	Order	✓
customerID	string	✓ cust123
items	OrderItem[]	
items[0]	OrderItem	✓
itemID	string	✓ item005
quantity	int	✓ 1

- **`-----------------------------------------------------------------------`**

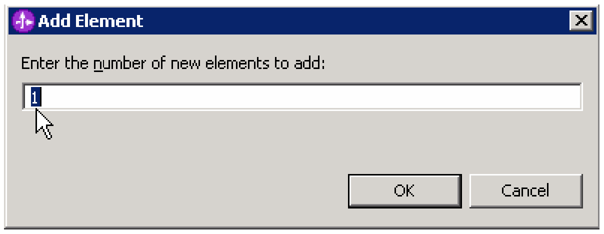
__ a. Enter these values into the Initial request parameters table:

1) For **customerID**, click under Value and enter **cust123**

2) Right-click any where on the row containing **items** and select **Add Elements…** from the pop-up menu



3) Enter '**1**' (default) in the Add Element window:


Add Element

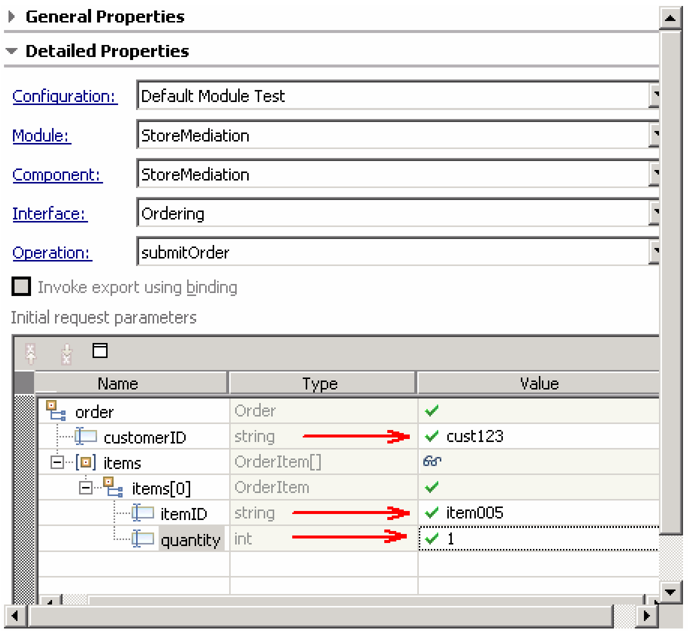
Enter the number of new elements to add:

1

4) Click **OK**

5) For **itemID**, click under Value and enter **item005**

6) For **quantity**, click under Value and enter **1**



Configuration: Default Module Test

Module: StoreMediation

Component: StoreMediation

Interface: Ordering

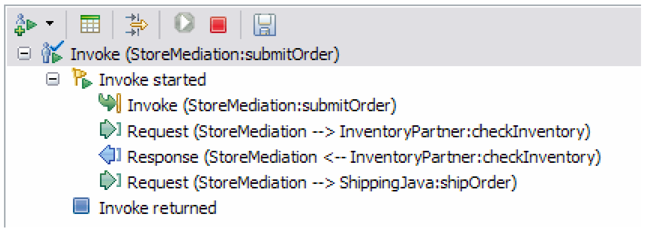
Operation: submitOrder

☐ Invoke export using binding

Initial request parameters

Name	Type	Value
order	Order	✓
customerID	string	✓ cust123
items	OrderItem[]	
items[0]	OrderItem	✓
itemID	string	✓ item005
quantity	int	✓ 1

____ 5. Run the test by hitting the Continue icon (). Results should look like this:

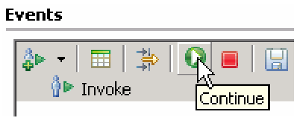


Invoke (StoreMediation:submitOrder)
Invoke started
Invoke (StoreMediation:submitOrder)
Request (StoreMediation --> InventoryPartner:checkInventory)
Response (StoreMediation <-- InventoryPartner:checkInventory)
Request (StoreMediation --> ShippingJava:shipOrder)
Invoke returned

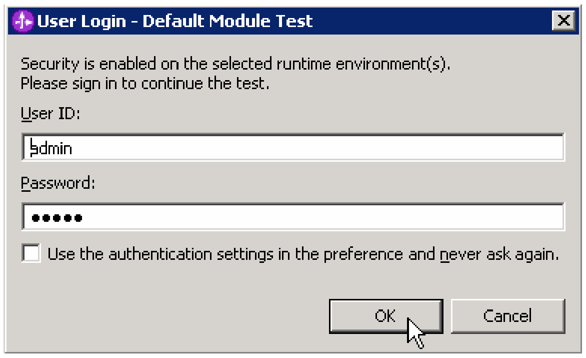
Note: If presented with the **User Login** dialog, set the **User ID** and **Password** to **admin** and **admin**. You should select the **'Use the authentication settings in the preference and never ask again'** check box to prevent this dialog from being displayed in the future.

- ---

__ a. Click the **Continue** icon ( ) under Events panel



__ b. If presented with the **User Login** dialog, enter the **User ID** and **Password**, which by default is normally set to **admin** and **admin**. Optionally, you can check the 'Use the authentication settings in the preference and never ask again' check box to prevent this dialog from being displayed in the future.



User Login - Default Module Test

Security is enabled on the selected runtime environment(s).
Please sign in to continue the test.

User ID: admin

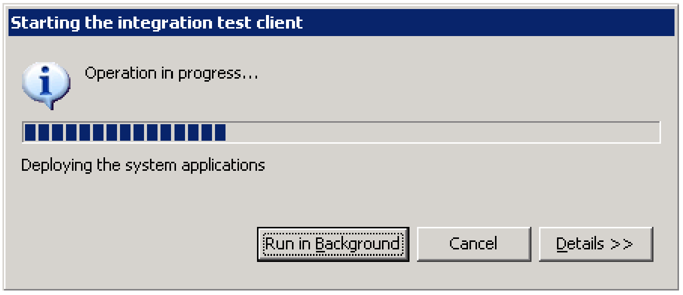
Password: ●●●●●

☐ Use the authentication settings in the preference and never ask again.

OK Cancel

__ c. click **OK**

__ d. Wait until the integration test client starts



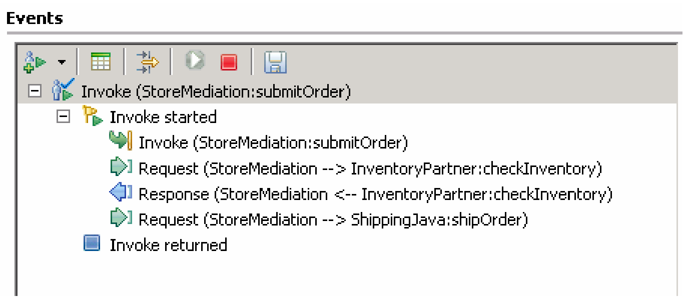
Starting the integration test client

Operation in progress...

Deploying the system applications

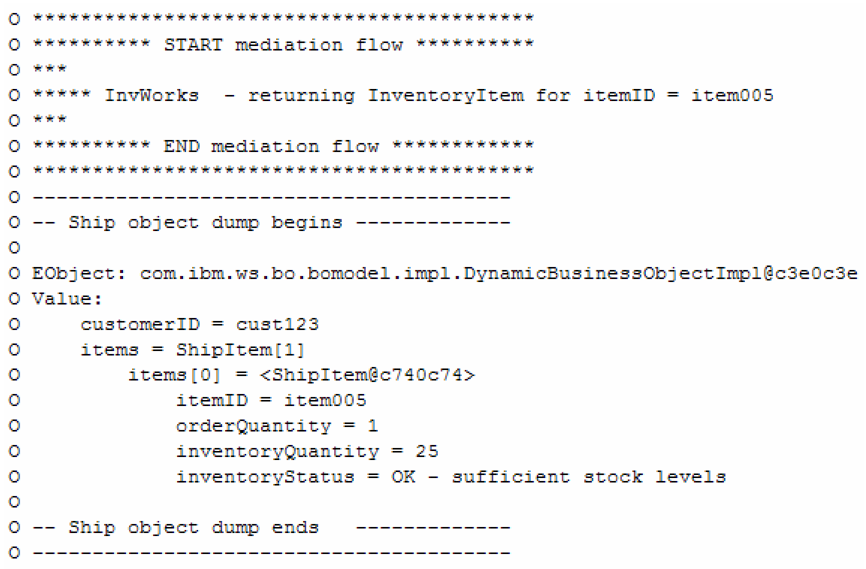
Run in Background Cancel Details >>

__ e. You should see these results:



```
Events
Invoke (StoreMediation:submitOrder)
   Invoke started
      Invoke (StoreMediation:submitOrder)
      Request (StoreMediation --> InventoryPartner:checkInventory)
      Response (StoreMediation <-- InventoryPartner:checkInventory)
      Request (StoreMediation --> ShippingJava:shipOrder)
   Invoke returned
```

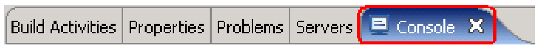
____ 6. Switch to the Console view and examine the output, which should look similar to this:



```
O *******************************************
O ********** START mediation flow **********
O ***
O ***** InvWorks  - returning InventoryItem for itemID = item005
O ***
O ********** END mediation flow ************
O *******************************************
O ----------------------------------------
O -- Ship object dump begins -------------
O
O EObject: com.ibm.ws.bo.bomodel.impl.DynamicBusinessObjectImpl@c3e0c3e
O Value:
O     customerID = cust123
O     items = ShipItem[1]
O         items[0] = <ShipItem@c740c74>
O             itemID = item005
O             orderQuantity = 1
O             inventoryQuantity = 25
O             inventoryStatus = OK - sufficient stock levels
O
O -- Ship object dump ends   -------------
O ----------------------------------------
```

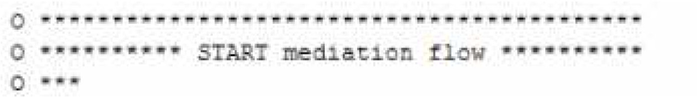
- --

__ a. Double click on **Console** view to see the above message (by double clicking, the Console view is maximized)



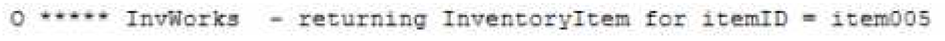
____ 7. Being able to examine and understand the Console output is important throughout this series of lab exercises. To that end, the output above is explained here:

- **Output produced by the PrintStartMsg primitive in the mediation flow:**



```
O *******************************************
O ********** START mediation flow **********
O ***
```

- **Output produced by the inventory service. The 'InvWorks' indicates which inventory service implementation was used (there are other implementations which are used in subsequent lab exercises examining service invoke retry capabilities).**



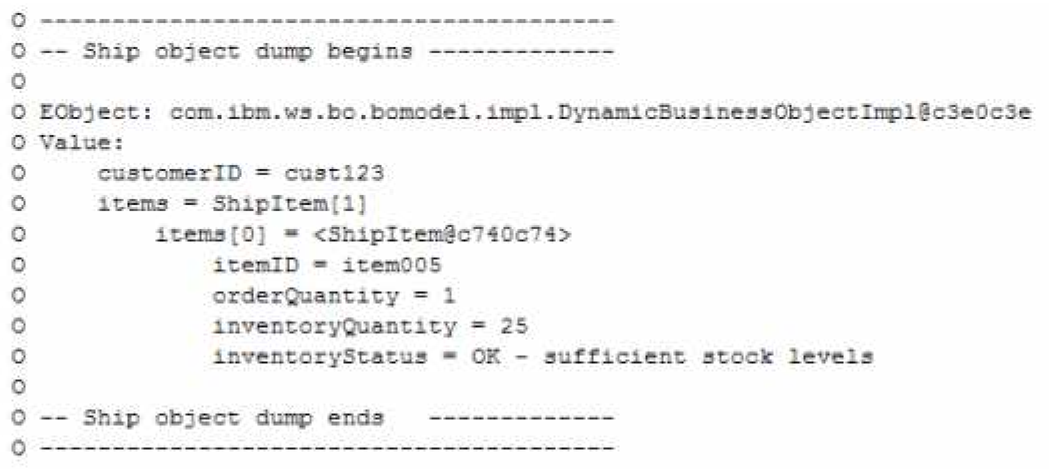
```
O ***** InvWorks  - returning InventoryItem for itemID = item005
```

- **Output produced by the PrintEndMsg primitive in the mediation flow:**

© Copyright IBM Corporation 2008. All rights reserved

```
O ***
O ********** END mediation flow ************
O ******************************************
```

- **`Output produced by the shipping service. It is a dump of the input business object Ship which is composed of a customerID and an array of ShipItem. Each ShipItem in the array is composed of information for the item from the Order and additional data for that item from the inventory service. In subsequent labs in this series, there are multiple elements in this array.`**



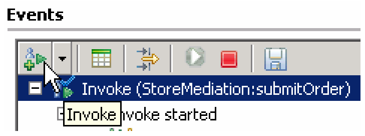
```
O ----------------------------------------
O -- Ship object dump begins -------------
O
O EObject: com.ibm.ws.bo.bomodel.impl.DynamicBusinessObjectImpl@c3e0c3e
O Value:
O     customerID = cust123
O     items = ShipItem[1]
O         items[0] = <ShipItem@c740c74>
O             itemID = item005
O             orderQuantity = 1
O             inventoryQuantity = 25
O             inventoryStatus = OK - sufficient stock levels
O
O -- Ship object dump ends   -------------
O ----------------------------------------
```

- **`----------------------------------------------------------------------`**

____ 8. You can run additional tests using different input data to see how the output varies. Key things to note about the data you use:

- **`customerID can be any string and should not have any particular affect on the results`**
- **`items should always be a 1 element array`**
- **`itemID values of “item001” through “item010” are recognized by the inventory service, all other values are not recognized`**
- **`inventory status will change according to the relationship between the order and inventory quantities`**
- **`----------------------------------------------------------------------`**

__ a. Click the **Invoke** icon () under Events panel

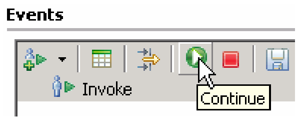


Invoke (StoreMediation:submitOrder)

Invoke started

__ b. Enter values for **customerID**, **itemID**, and **quantity** as per the above instructions

__ c. Click the **Continue** icon () under Events panel



Part 4: Clean up the environment if you will not proceed to the next lab

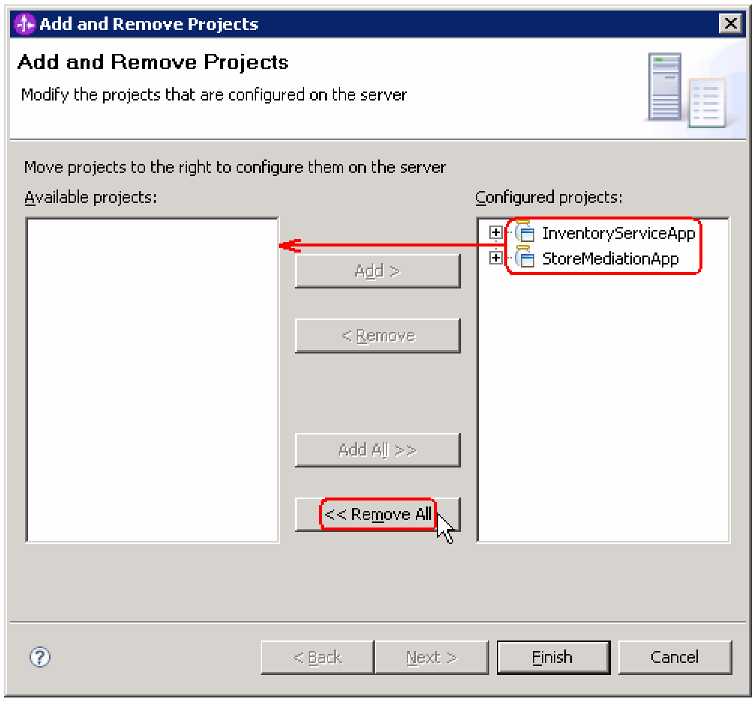
Perform this part only if you are not continuing to the message splitting and aggregation lab (the second lab in this series of labs).

____ 1. Remove the InventoryServiceApp and StoreMediationApp from the test server.

- ---

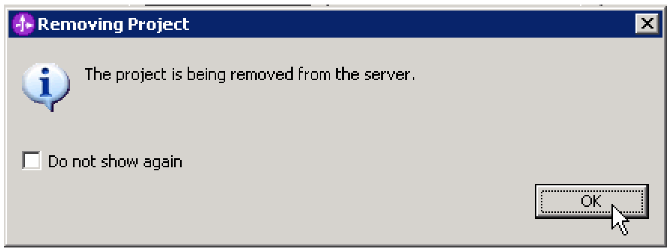
__ a. Right-click on **WebSphere ESB Server v6.1** under the Servers view and select **Add and remove projects…** from the context menu

__ b. From the Add and Remove Projects window, click **<< Remove All**



__ c. Click **Finish** after you see the applications moved to Available projects.

__ d. If displayed, click **OK** in 'Removing Project' window. Optionally, you can select the check box for 'Do not show again' not to be asked again when you remove projects later

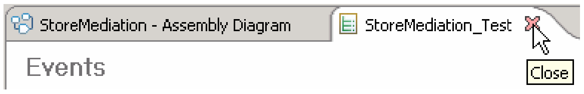


__ e. Wait until the application is removed from the server

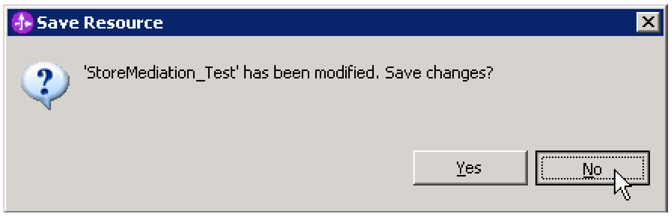
____ 2. Close the StoreMediation_Test panel without saving

- ---

__ a. Click **X** on the StoreMediation_Test tab

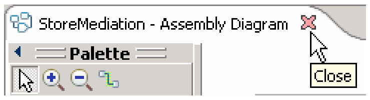


__ b. Click **No** from Save Resource window


Save Resource

'StoreMediation_Test' has been modified. Save changes?

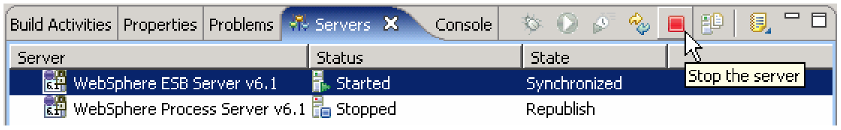
__ c. Click **X** on the StoreMediation – Assembly Diagram tab



____ 3. Stop the test server

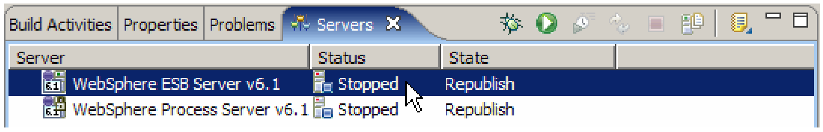
- --

__ a. From the **Servers** view of WebSphere Integration Developer, select **WebSphere ESB Server v6.1** and hit '**Stop the server**' icon (🟥) from the toolbar



Server	Status	State
WebSphere ESB Server v6.1	Started	Synchronized
WebSphere Process Server v6.1	Stopped	Republish

__ b. Wait until the server Status shows as **Stopped**



Server	Status	State
WebSphere ESB Server v6.1	Stopped	Republish
WebSphere Process Server v6.1	Stopped	Republish

____ 4. Exit from WebSphere Integration Developer.

- --

__ a. From menu, select **File → Exit** or click '**X**' at the right top corner of your WebSphere Integration Developer window

© Copyright IBM Corporation 2008. All rights reserved

What you did in this exercise

In this exercise, you used a service invoke primitive to obtain data used to augment a message. In order to use the service invoke primitive, you needed to use XSL transformation primitives before and after to modify the message type and correctly set the data into the message.

Reviewing the presentation entitled Augmentation, aggregation and retry tutorials will help you better understand what was done in the lab.

© Copyright IBM Corporation 2008. All rights reserved

Solution instructions

If you want to run this lab with a completed solution rather than authoring the flow yourself, follow these instructions:

____ 1. Follow **Part 1: Setting up the environment for the lab**, however use the project interchange file that contains the solution.

- **`<LAB_FILES>/PI2-AugmentSolution-AggregateStart.zip`**
- **`-------------------------------------------------------------`**

__ a. Start WebSphere Integration Developer, preferably using a new workspace

1) Select **Start → All Programs → IBM WebSphere Integration Developer → IBM WebSphere Integration Developer V6.1 → WebSphere Integration Developer V6.1**

2) From the Workspace Launcher window, enter **<LAB_FILES>/workspaces/ws1solution** for the Workspace field

__ b. Import the project interchange file containing the starting point for the lab

1) From the menu, select **File → Import…**

2) In the **Import** dialog, select **Other → Project Interchange**

3) Click **Next**

4) In the **Import Project Interchange Contents** dialog set the **From zip file:** value to **<LAB_FILES>/PI2-AugmentSolution-AggregateStart.zip**

5) Click **Select All** to select all of the projects

6) Click **Finish**

7) Wait for the build process to complete for the imported projects, which is seen at the lower right corner of WebSphere Integration Developer

____ 2. Skip to **Part 3: Test the message augmentation mediation** and proceed through the rest of the lab.

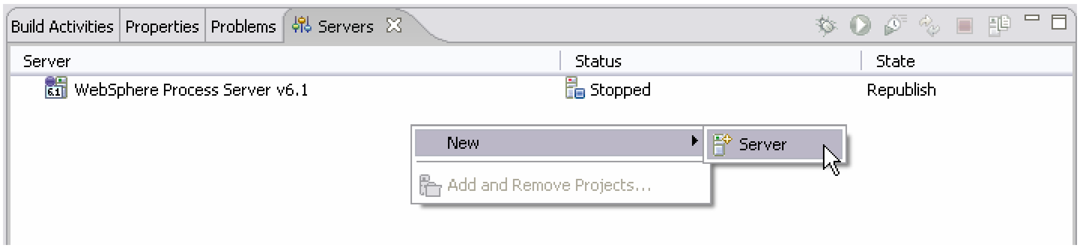
Task: Adding remote server to the WebSphere Integration Developer test environment

This task describes how to add a remote server to the WebSphere Integration Developer test environment. This sample uses a z/OS machine.

____ 1. Define a new remote server to WebSphere Integration Developer.

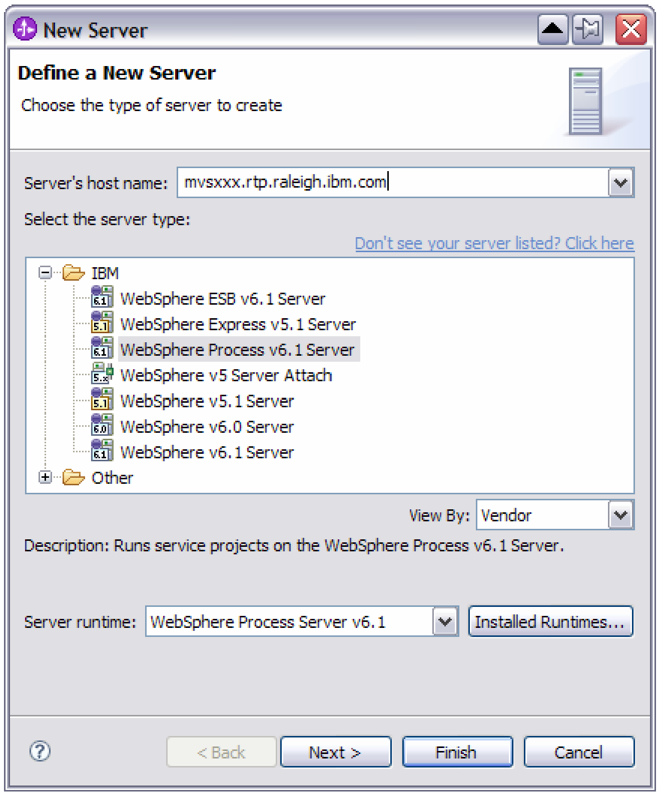
__ a. Right click on the background of the **Servers** view to access the pop-up menu.

__ b. Select **New → Server**.



__ c. In the New Server dialog, specify the remote server's host name, **<HOSTNAME>**.

__ d. Ensure that the appropriate server type, '**WebSphere Process v6.1 Server**' or '**WebSphere ESB v6.1 Server**', is highlighted in the server type list


New Server

Define a New Server

Choose the type of server to create

Server's host name: mvsxxx.rtp.raleigh.ibm.com

Select the server type:

Don't see your server listed? Click here
IBM
WebSphere ESB v6.1 Server
WebSphere Express v5.1 Server
WebSphere Process v6.1 Server
WebSphere v5 Server Attach
WebSphere v5.1 Server
WebSphere v6.0 Server
WebSphere v6.1 Server
Other

View By: Vendor

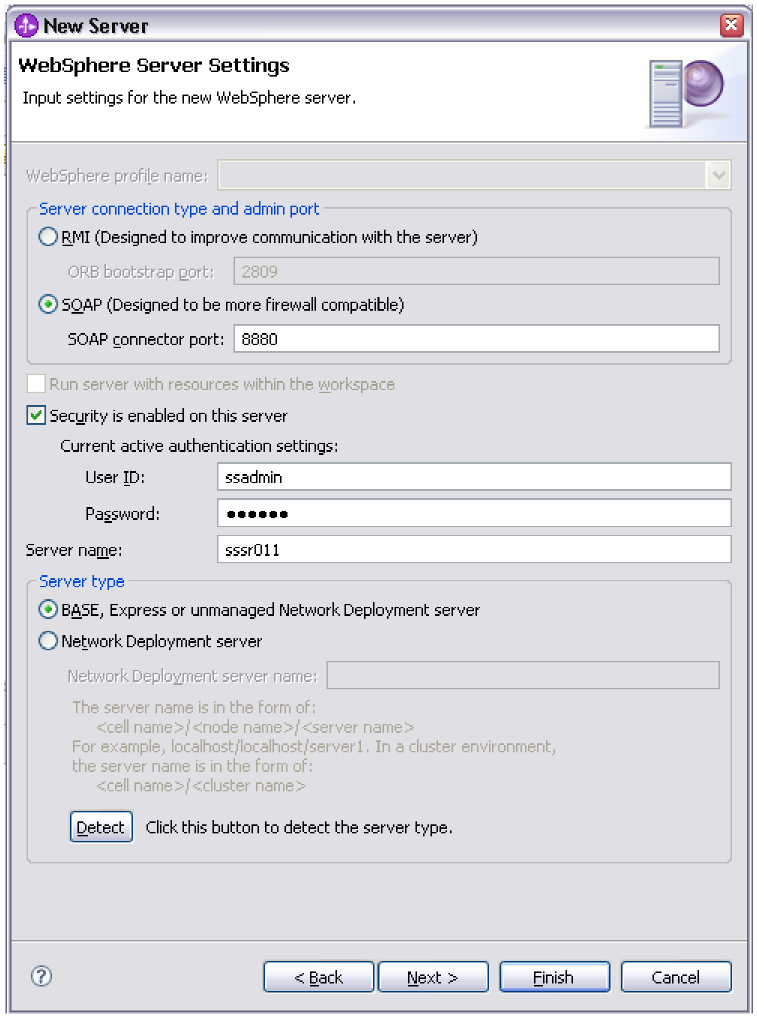
Description: Runs service projects on the WebSphere Process v6.1 Server.

Server runtime: WebSphere Process Server v6.1

Installed Runtimes...
< Back
Next >
Finish
Cancel

__ e. Click **Next.**

__ f. On the WebSphere Server Settings page, leave the radio button for **SOAP** selected, changing the **SOAP connector port** to the correct setting (**<SOAP_PORT>**). If security is on in your server, check the box for **'Security is enabled on this server'** and input **<USERID>** for the user ID and **<PASSWORD>** for the password.


New Server
WebSphere Server Settings
Input settings for the new WebSphere server.

WebSphere profile name:

Server connection type and admin port
- RMI (Designed to improve communication with the server)
  - ORB bootstrap port: 2809
- [✓ SOAP (Designed to be more firewall compatible)]
  - SOAP connector port: 8880

Run server with resources within the workspace
[✓] Security is enabled on this server
Current active authentication settings:
User ID: ssadmin
Password: ●●●●●●
Server name: sssr011

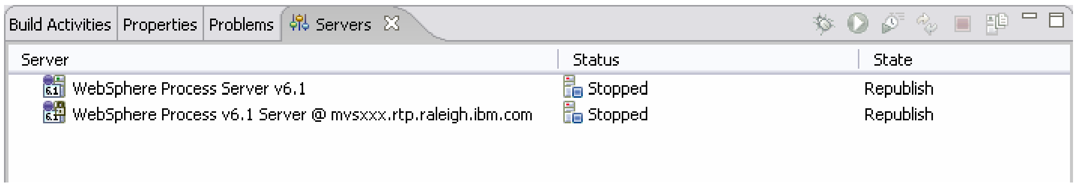
Server type
- [✓ BASE, Express or unmanaged Network Deployment server]
- Network Deployment server
  - Network Deployment server name:

The server name is in the form of:
<cell name>/<node name>/<server name>
For example, localhost/localhost/server1. In a cluster environment, the server name is in the form of:
<cell name>/<cluster name>

Detect — Click this button to detect the server type.

__ g. Click **Finish**.

__ h. The new server should be seen in the Server view.



Server	Status	State
WebSphere Process Server v6.1	Stopped	Republish
WebSphere Process v6.1 Server @ mvsxxx.rtp.raleigh.ibm.com	Stopped	Republish

____ 2. Start the remote server if it is not already started. WebSphere Integration Developer does not support starting remote servers from the Server View.

__ a. From a command prompt, telnet to the remote system if needed:

© Copyright IBM Corporation 2008. All rights reserved

‘**telnet <HOSTNAME> <TELNET_PORT>**’

User ID: **<USERID>**

Password: **<PASSWORD>**

__ b. Navigate to the bin directory for the profile being used:

cd <WAS_HOME>/profiles/<PROFILE_NAME>/bin

__ c. Run the command file to start the server: **./startServer.sh <SERVER_NAME>**

__ d. Wait for status message indicating server has started:

```
ADMU3200I: Server launched. Waiting for initialization status
ADMU3000I: Server sssr01 open for e-business; process id is 0000012000000002
```