

IBM WEBSHERE 7.0 – LAB EXERCISE

**WebSphere Enterprise Service Bus 7.0**  
**Augmentation, aggregation, and retry tutorial series**  
**Lab Four – Retry with alternate endpoints**

What this exercise is about ..... 1

Lab requirements ..... 2

What you should be able to do ..... 2

Introduction ..... 2

Exercise Instructions ..... 3

Understanding how to read the instructions ..... 4

Part 1: Setting up the environment for the lab ..... 5

Part 2: Authoring the mediation flow to retry using alternate endpoints ..... 8

Part 3: Test the retry with alternate endpoints mediation flow ..... 18

Part 4: Clean up the environment ..... 27

What you did in this exercise ..... 29

Solution instructions ..... 30

**What this exercise is about**

The objective of this lab is to provide you with an understanding of how automated service call retry capabilities can use alternate endpoints when retrying a call.

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

## Lab requirements

- WebSphere Integration Developer V7.0 installed on Windows or Linux
- WebSphere Enterprise Service Bus V7.0 or WebSphere Process Server V7.0. The server can either be the test server installed by WebSphere Integration Developer or a remote server from a separate WebSphere Enterprise Service Bus V7.0 or WebSphere Process Server V7.0 installation.

## What you should be able to do

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

- Understand the service message object structure for dynamic endpoints and alternate endpoints and how they can be initialized
- Configure a service invoke mediation primitive to use alternate endpoints when performing service call retry.

---

## Introduction

This lab exercise is the fourth and final lab in a series of labs intended to illustrate elements of the mediation programming model, addressing these capabilities:

- Using a service invoke primitive in a flow
- Augmenting message content with the results of a service invoke
- Using message splitting and aggregation for message augmentation of repeating elements
- Service call retry of failing service invocations
- Using alternate endpoints for service call retry

The four labs are described in the presentation entitled [Augmentation, aggregation and retry labs](#). You should familiarize yourself with the labs as described in the presentation before attempting this lab.

---

## 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\WID70	/opt/IBM/WID70
<ESB_PROFILE_HOME>	<WID_HOME>_WTE\runtimes\bi_v7\profiles\qesb	<WID_HOME>_WTE\runtimes\bi_v7\profiles\qesb
<LAB_FILES>	C:\Labfiles70\WESB\AugAggRetry	/tmp/Labfiles70\WESB\AugAggRetry

---

## 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 previous paragraph, should allow the experienced user to complete the task**
  - **The following line of dashes begins the step by step instructions needed by the new user to complete this task. The experience user can skip to the next 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.

## Part 1: Setting up the environment for the lab

**What you will do in this part:** In this part you are getting the environment set up to do the lab. There are three different ways you might be approaching this which will dictate what you need to do.

**(1) Proceeding from Lab Three** – You are directly continuing from Lab Three and you did not shut down the server and development environment. In this case, there is nothing that needs to be done in this part.

**(2) Restarting from Lab Three** – You are continuing from Lab Three which you did previously and therefore you did shut down the server and development environment. In this case, you will need to restart the development environment and server but will not need to import a project interchange to initialize the workspace.

**(3) Directly starting Lab Four** – You are starting this lab from scratch, irrespective of whether you had previously completed Lab Three.

\_\_\_ 1. If you are **proceeding from Lab Three** there is nothing to do, skip directly to **Part 2 Authoring the mediation flow to retry using alternate endpoints**

• -----

\_\_\_ 2. Start WebSphere Integration Developer.

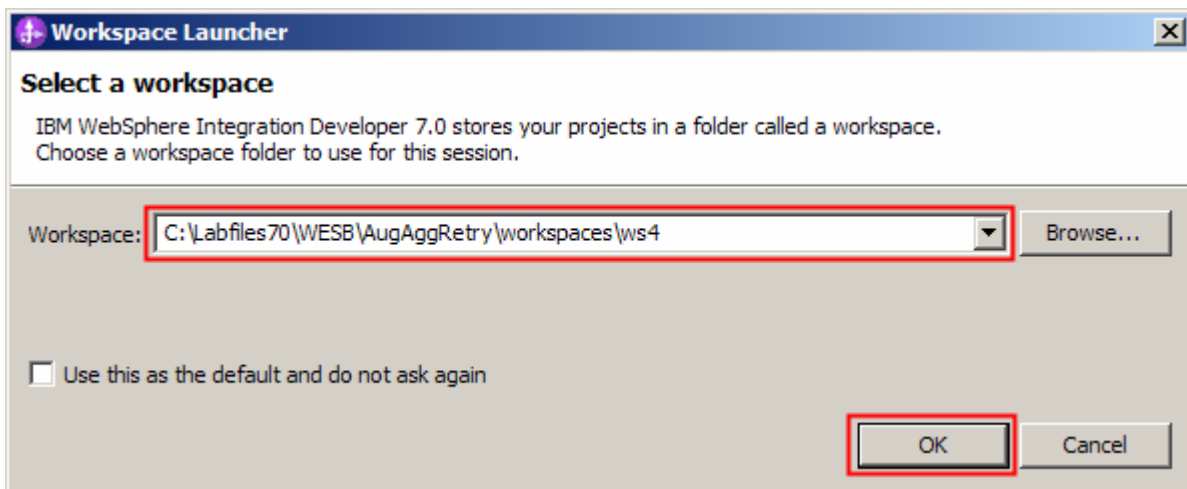
- **Restarting from Lab Three : Use the same workspace used in Lab Three**
- **Directly starting Lab Four: Suggested location:**  
`<LAB_FILES>/workspaces/ws4`
- -----

\_\_\_ a. Select **Start** → **All Programs** → **IBM WebSphere Integration Developer** → **IBM WebSphere Integration Developer V7.0** → **WebSphere Integration Developer V7.0**

\_\_\_ b. From the Workspace Launcher window, enter the name of the workspace in the Workspace field

1) Restarting from Lab Three: Use the same workspace used in Lab Three

2) Directly starting Lab Four: `<LAB_FILES>/workspaces/ws4`



\_\_\_ 3. If you are **restarting from Lab Three** there is nothing additional to do, skip directly to **Part 2 Authoring the mediation flow to retry using alternate endpoints**

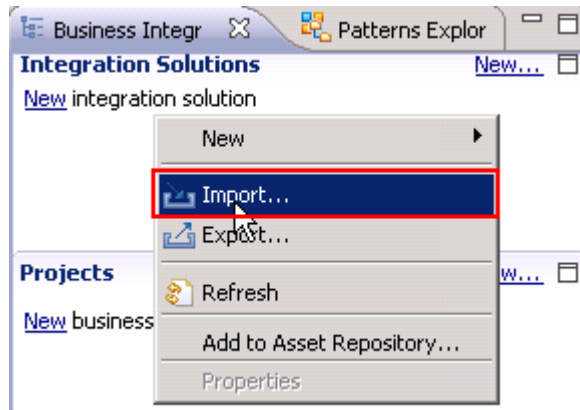
- -----

\_\_\_ 4. If you are **directly starting Lab Four** import the project interchange file containing the starting point for the lab

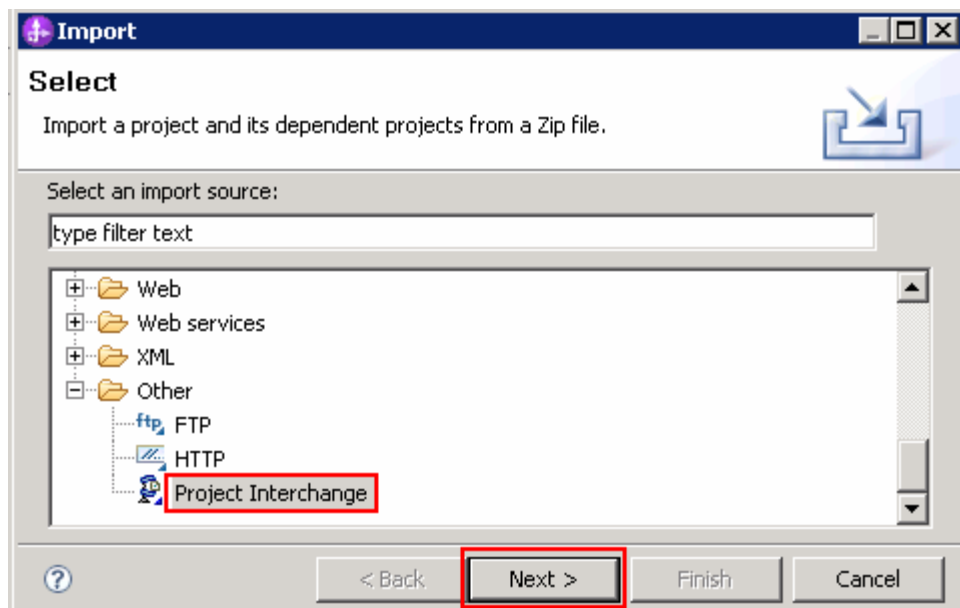
- <LAB\_FILES>/PI4-RetrySolution-AlternateEndpointsStart.zip

- -----

\_\_\_ a. Right-click inside **Business Integration View** (top left view in the Business Integration Perspective) and select **Import** from the pop-up menu

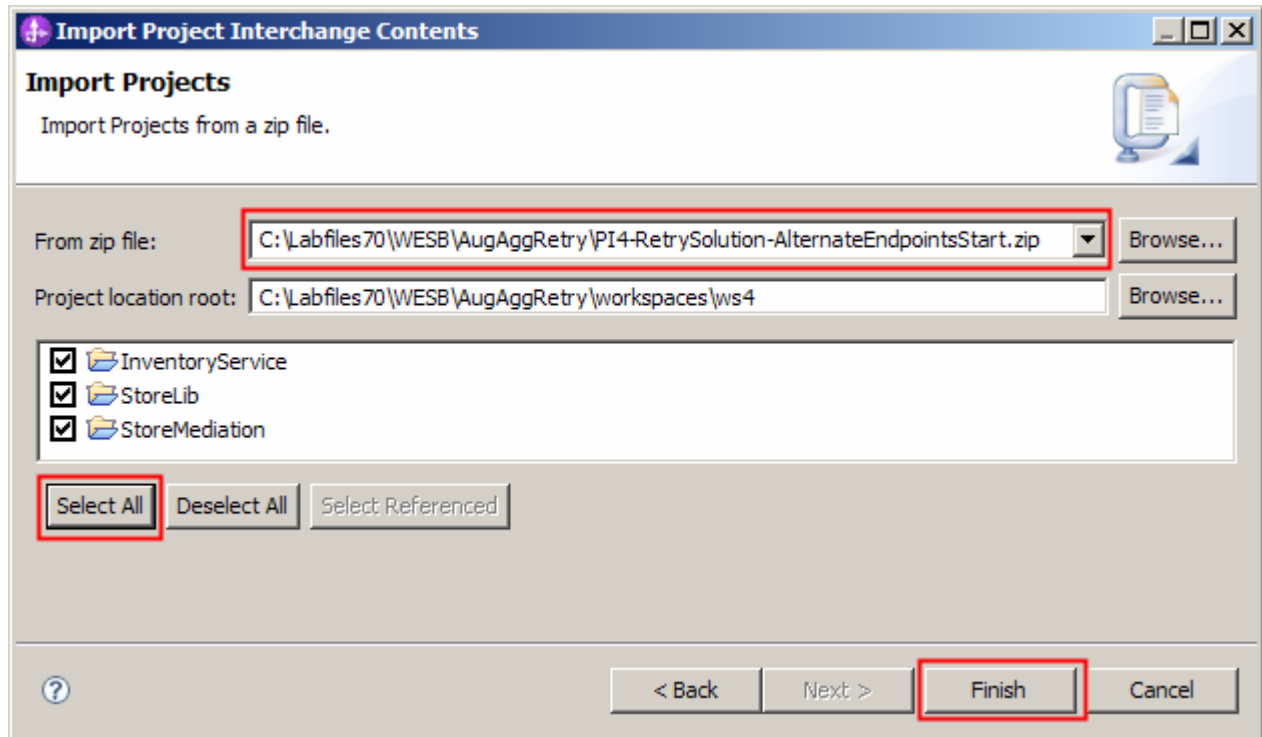


\_\_\_ b. In the **Import** dialog, expand **Other** and select **Project Interchange** from the list

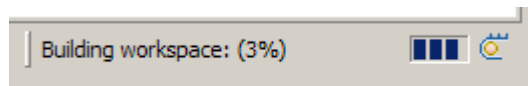


\_\_\_ c. Click **Next**

- \_\_\_ d. In the next panel, click the **Browse** button for **From zip file** and navigate to <LAB\_FILES>/PI4-RetrySolution-AlternateEndpointsStart.zip and click **Open**



- \_\_\_ e. Click **Select All** to select all the projects listed and then click **Finish**
- \_\_\_ f. Wait for the build process to complete for the imported projects, which is seen at the lower right corner of WebSphere Integration Developer.

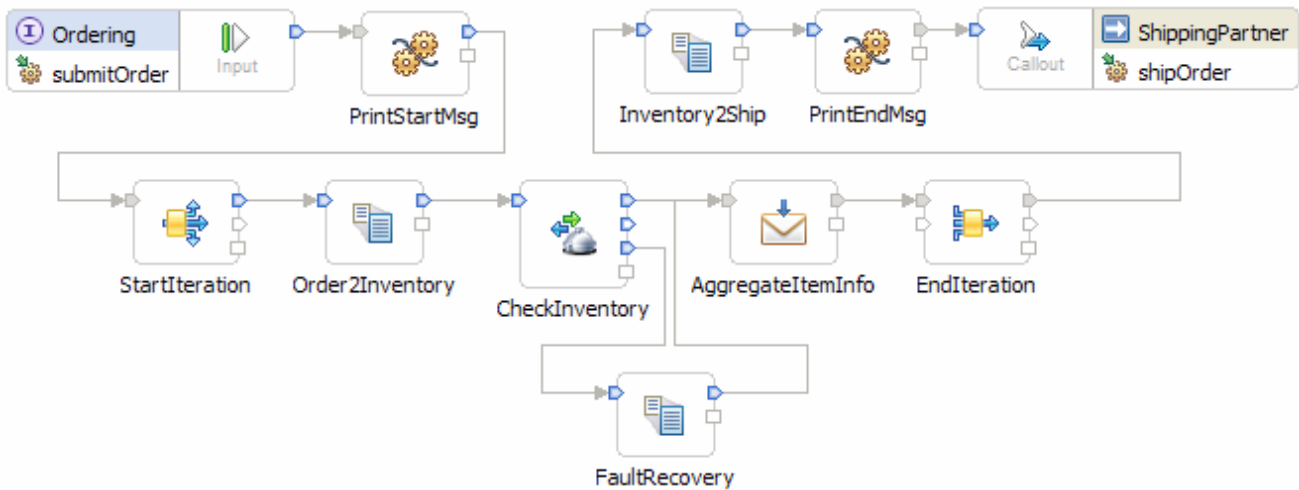


## Part 2: Authoring the mediation flow to retry using alternate endpoints

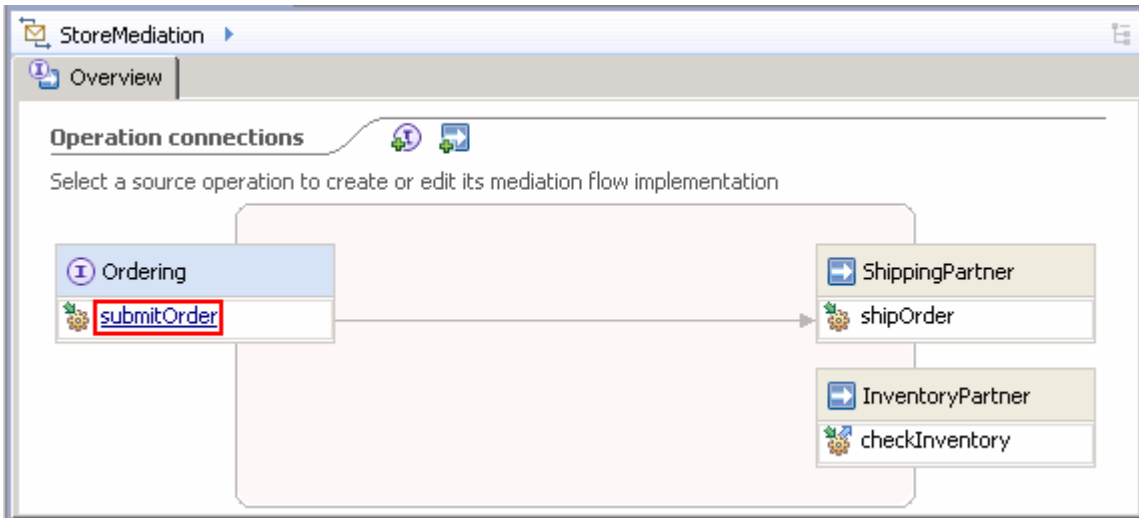
**What you will do in this part:** In this part you will modify the flow so that it initializes the SMO header fields for dynamic target address and alternate endpoints. Then you will modify the CheckInventory service invoke primitive to use the dynamic target address (rather than what is wired on the assembly diagram) and to use the alternate target addresses during service invoke retry processing.

You should see the presentation entitled [Augmentation, aggregation and retry tutorials](#) to better understand what this part is doing.

1. Open the StoreMediation flow found in the StoreMediation module and then open the flow for the submitOrder operation.

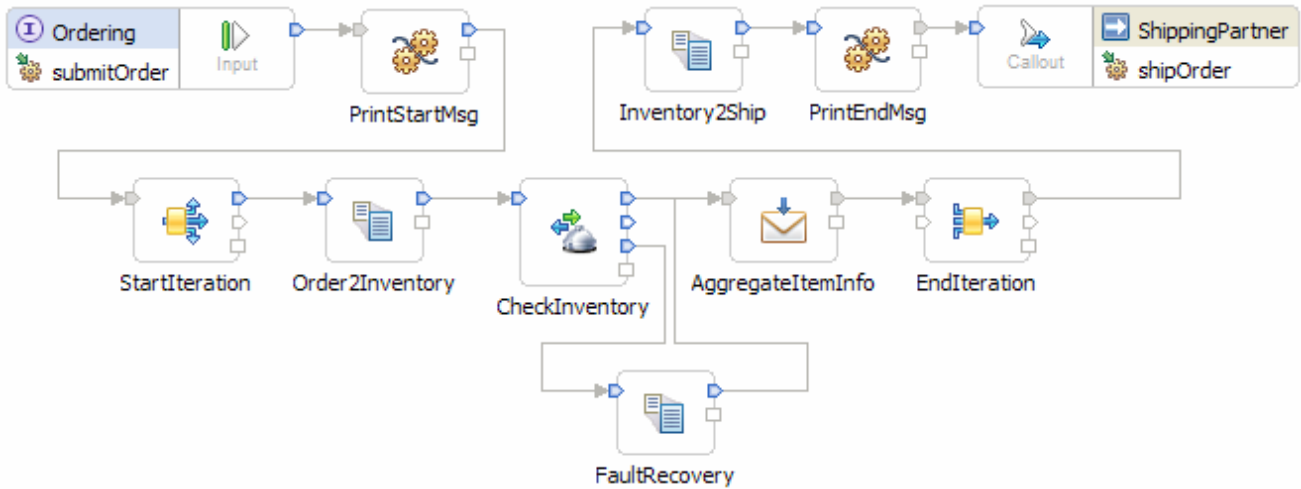


- a. In the Business Integration view, expand **StoreMediation** → **Integration Logic** → **Mediation Flows** and then double-click **StoreMediation** to open it in the mediation flow editor



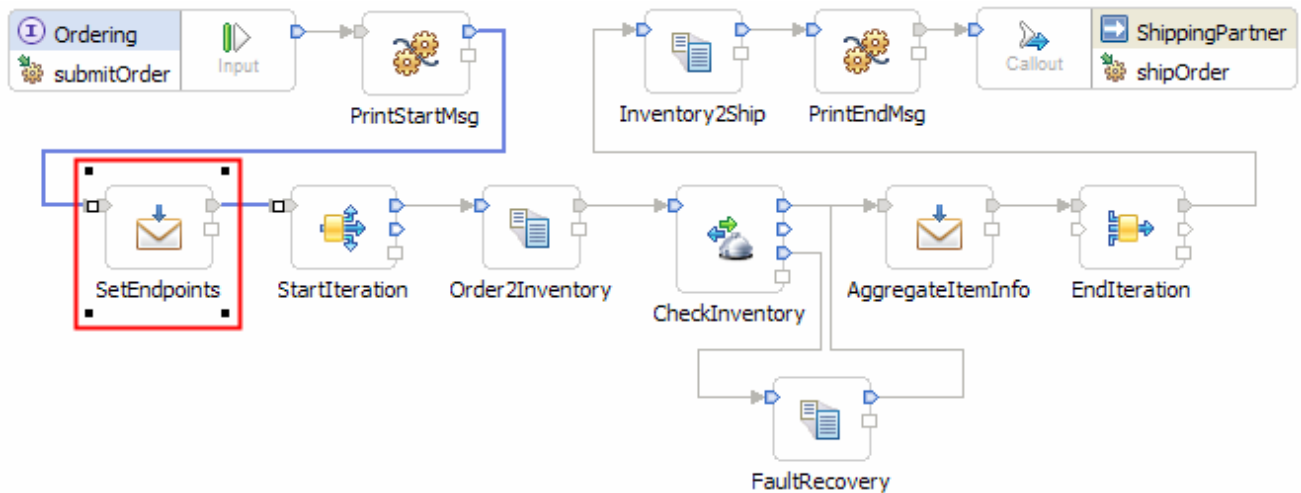


- \_\_\_ b. In the mediation flow editor, click the source operation, **submitOrder**, as shown in the picture above. This action opens the Request flow as shown here:



- \_\_\_ 2. Add a message element setter primitive and rewire the flow so that it is in between the **StartPrintMsg** primitive and the **StartIteration** primitive. This is used to initialize the target endpoint and alternate endpoints. In a typical production scenario, this is most likely done with an endpoint lookup primitive interfacing with WebSphere Service Registry and Repository.

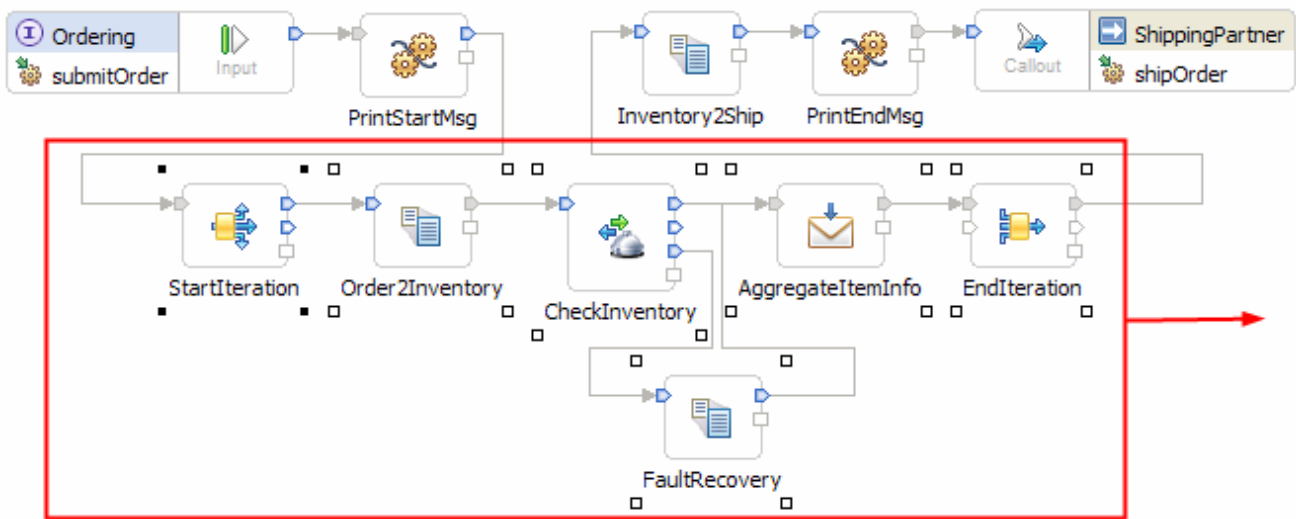
- **Display Name:** SetEndpoints
- **Wire** : PrintStartMsg to SetEndpoints
- **Wire** : SetEndpoints to StartIteration



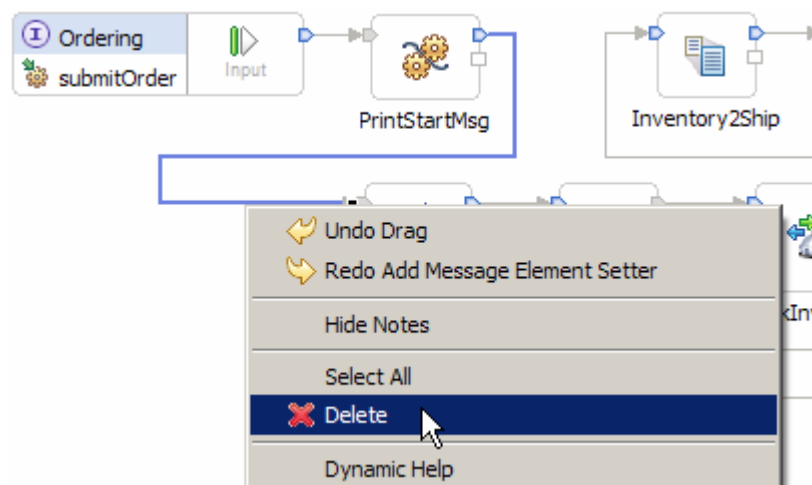
**NOTE:** In order to maintain a nicely formatted flow, the primitives from StartIteration to EndIteration were moved to the right before adding the message element setter primitive.

- -----

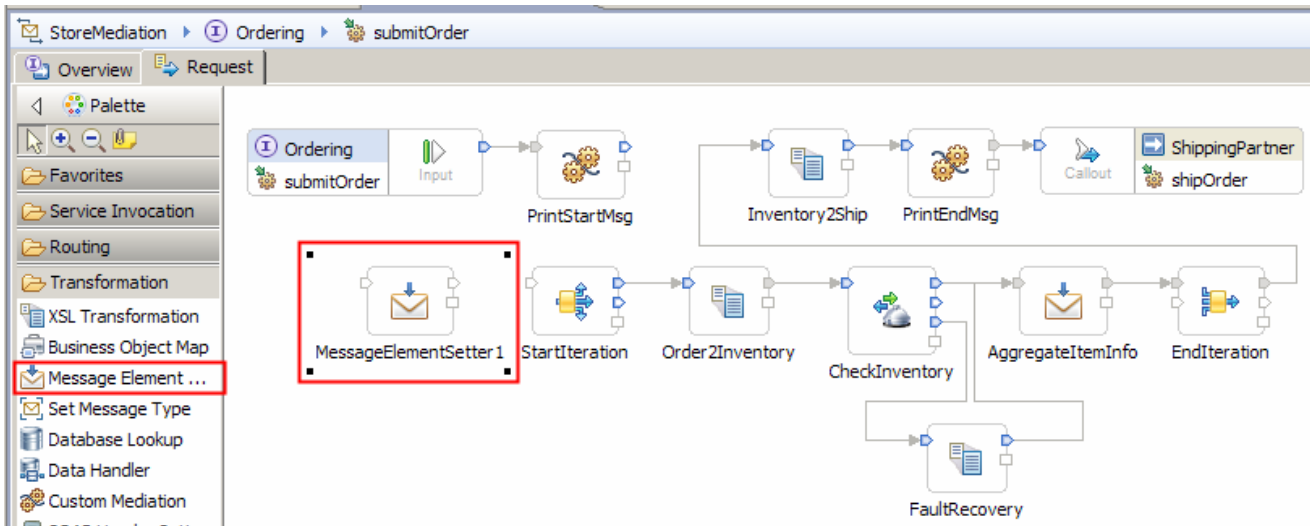
- \_\_\_ a. To make room for the new primitive, select the lower row of primitives and move them to the right.. You can select them all by rubber banding around them or selecting them one at a time while holding the Ctrl key. Then drag the entire grouping to the right.



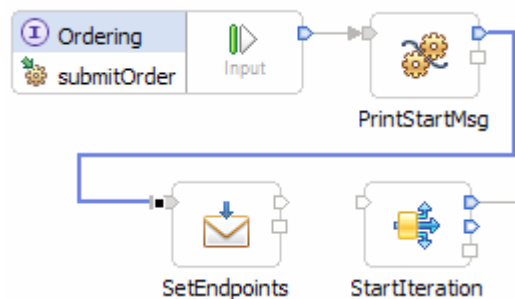
- \_\_\_ b. Delete the wire from **PrintStartMsg** to **StartIteration** by selecting it, **right clicking** to get the pop-up menu and selecting **Delete**.



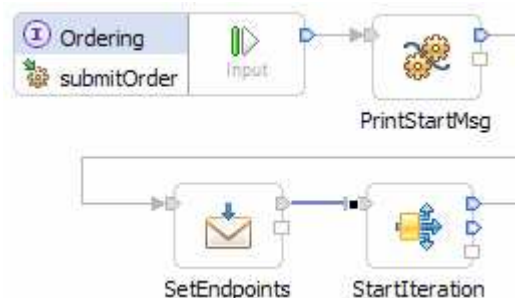
- \_\_\_ c. From the **Palette**, select **Transformation → Message Element Setter** and then click the canvas to add the primitive (to the left of the **Order2Inventory** primitive as shown below).



- \_\_\_ d. Ensure that the message element setter primitive is selected and navigate to the **Description** panel of the **Properties** view and change the **Display Name** to **SetEndpoints**.
- \_\_\_ e. Add a connection from the PrintStartMsg primitive to the SetEndpoints primitive. Click the **PrintStartMsg out** terminal and drag it to the **in** terminal of **SetEndpoints**.



- \_\_\_ f. Similarly add a connection from the SetEndpoints to the StartIteration primitive. Click the **out** terminal of **SetEndpoints** and drag to the **in** terminal of **StartIteration**



3. Configure the SetEndpoints message element setter primitive to set up the target address and alternate target addresses. To illustrate retry with alternate endpoints, the target address is set to the endpoint that always fails and thus a retry will always occur. Then the alternate addresses are set to the endpoints that randomly fail and always work, in that order. So in some cases, the first alternate endpoint will work and in other cases it will fail, but the second alternate endpoint will always work.

- Set properties of the message element setter to assign string values as follows

Target location in SMO	String value
/headers/SMOHeader/Target/address	sca://InventoryService/InvFails
/headers/SMOHeader/AlternateTarget[1]/address	sca://InventoryService/InvRandom
/headers/SMOHeader/AlternateTarget[2]/address	sca://InventoryService/InvWorks

Message Elements:

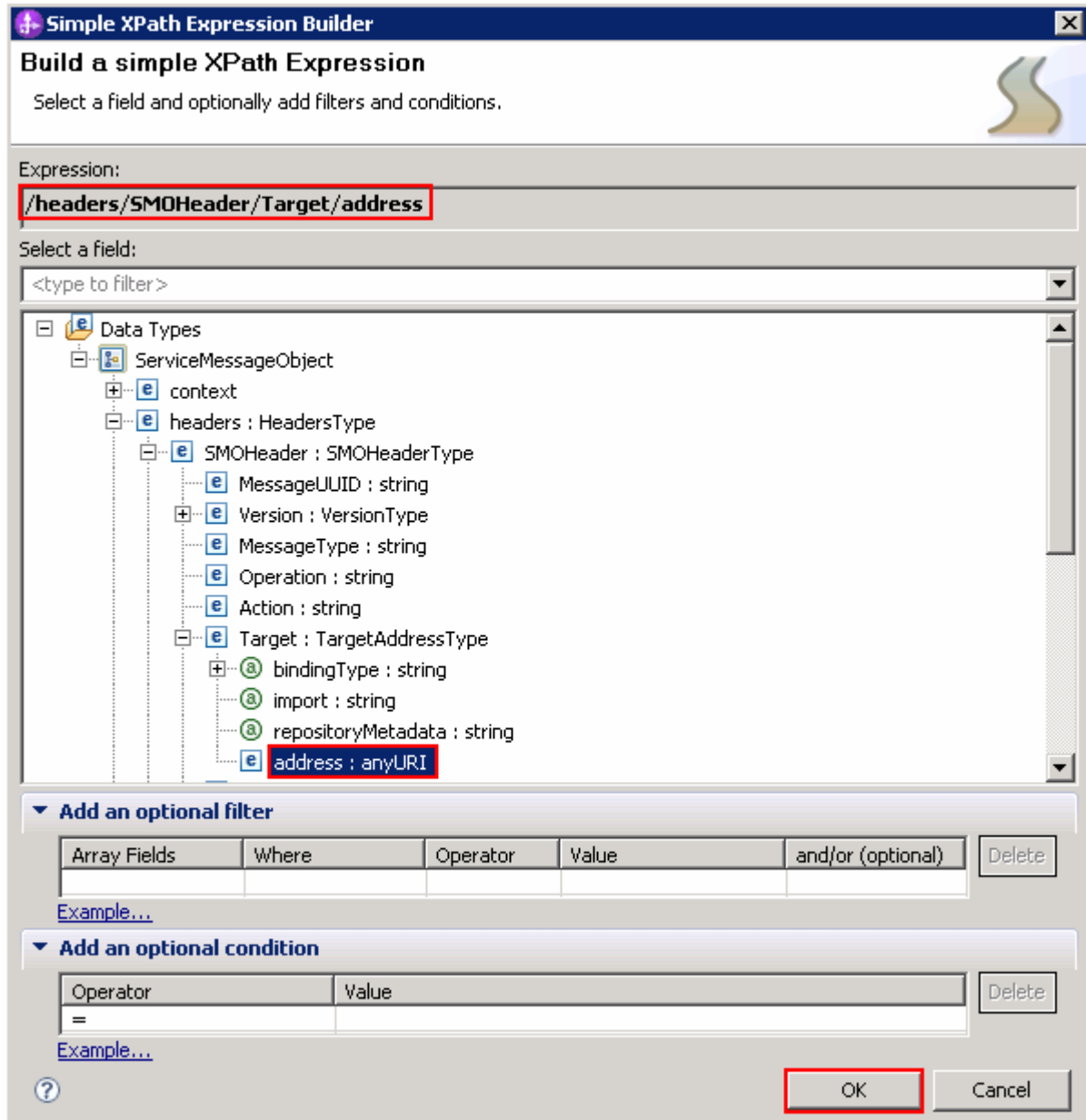
1	Set element /headers/SMOHeader/Target/address to "sca://InventoryService/InvFails"
2	Set element /headers/SMOHeader/AlternateTarget[1]/address to "sca://InventoryService/InvRandom"
3	Set element /headers/SMOHeader/AlternateTarget[2]/address to "sca://InventoryService/InvWorks"

**Note:** Ensure that the setting of alternate target address[1] comes before the setting of alternate target address[2] in the table. If not, use the up or down icons to rearrange the table rows so that they match the screen capture.

- -----
- \_\_\_ a. Ensure that the **SetEndpoints** primitive is selected and navigate to the **Details** panel of the **Properties** view.
- \_\_\_ b. Set /headers/SMOHeader/Target/address to the string URL value sca://InventoryService/InvFails
  - 1) Under Message Elements table, click the **Add...** button. The Add/Edit window is opened
  - 2) For **Action**, select **Set** from the drop down box.

3) Define **Target**:

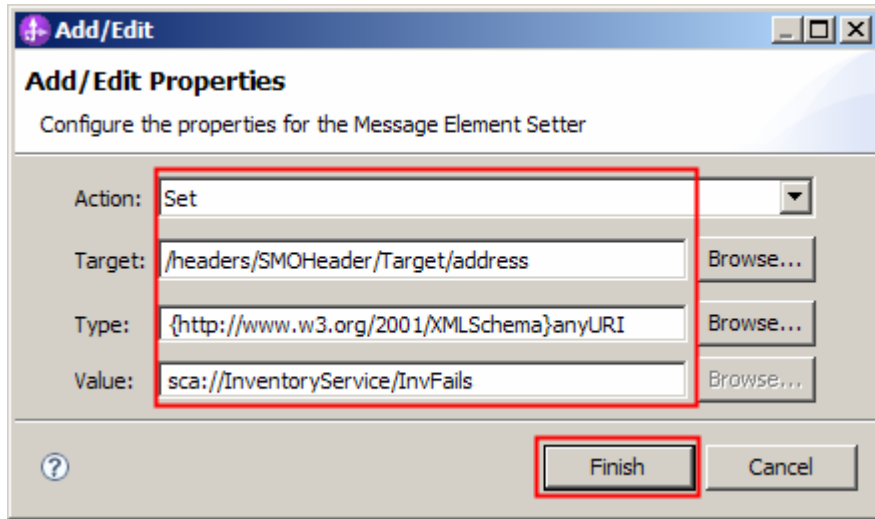
- a) Click **Browse** for **Target**. The Simple XPath Expression Builder wizard is opened
- b) Expand **ServiceMessageObject** → **headers** → **SMOHeader** → **Target**
- c) Select **address**. You should see the XPath expression populated in the **Expression** text area as shown below:



- d) Click **OK** button. You are now back to Add/Edit window

- 4) For **Type**, accept the default value `{http://www.w3.org/2001/XMLSchema}anyURI`
- 5) For **Value**, enter `sca://InventoryService/InvFails`

6) Your Add/Edit window should look like the picture shown below:



7) Click **Finish**

8) You will see this entry in the Message Elements table:

Message Elements:

1	Set element /headers/SMOHeader/Target/address to "sca://InventoryService/InvFails"
---	--

\_\_\_ c. Now set /headers/SMOHeader/AlternateTarget[1]/address to the string URL value sca://InventoryService/InvRandom

1) Under Message Elements table, click the **Add...** button. The Add/Edit window is opened

2) For **Action**, select **Set** from the drop down box.

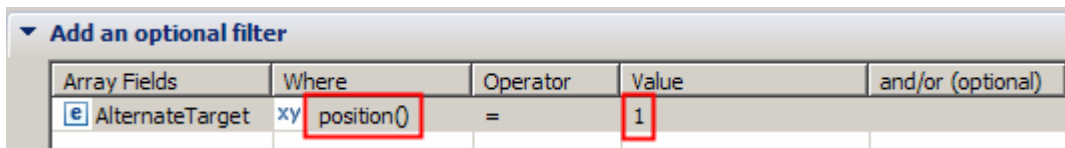
3) Define **Target**:

a) Click **Browse** for **Target**. The Simple XPath Expression Builder wizard is opened

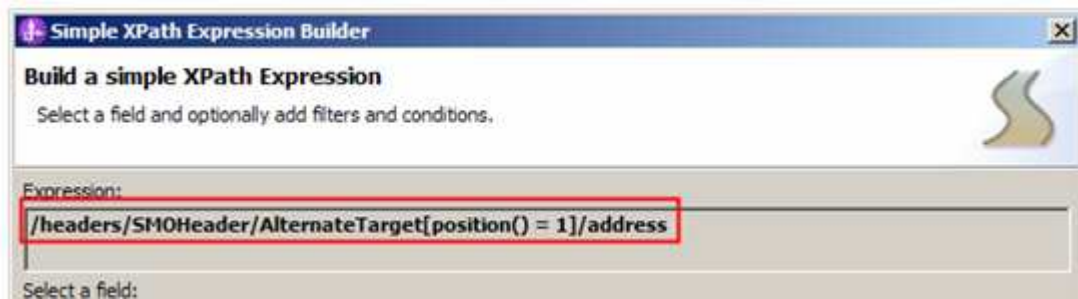
b) Expand **ServiceMessageObject** → **headers** → **SMOHeader** → **AlternateTarget**

c) Select **address**. You should see the XPath expression populated in the **Expression** text area

d) In the **Add an optional filter** section, under **Where** select **position()** from the drop down and under **Value** type the value **1**



e) You should now see this expression under **Expression**



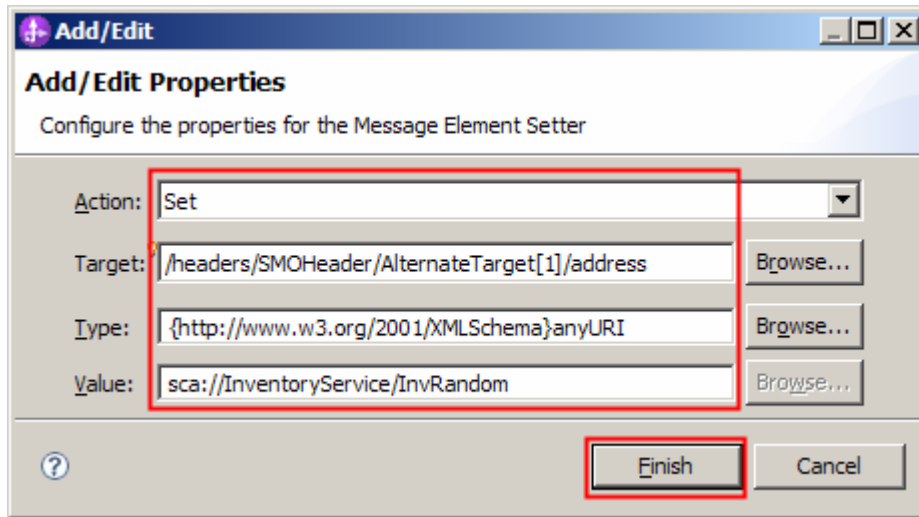
f) Click **OK** button. You are now back to Add/Edit window

g) Unfortunately, this expression needs to be modified in order to work for a Set operation in a message element setter primitive, and the Simple XPath Expression Builder does not enable you to build the appropriate expression. Rather than the expression: `/headers/SMOHeader/AlternateTarget[position() = 1]/address` the expression required is: `/headers/SMOHeader/AlternateTarget[1]/address`. Edit the Target field in the Add/Edit dialog to remove “`position() =`”

4) For **Type**, accept the default value `{http://www.w3.org/2001/XMLSchema}anyURI`

5) For **Value**, enter `sca://InventoryService/InvRandom`

6) Your Add/Edit window should look like this:



7) Click **Finish**

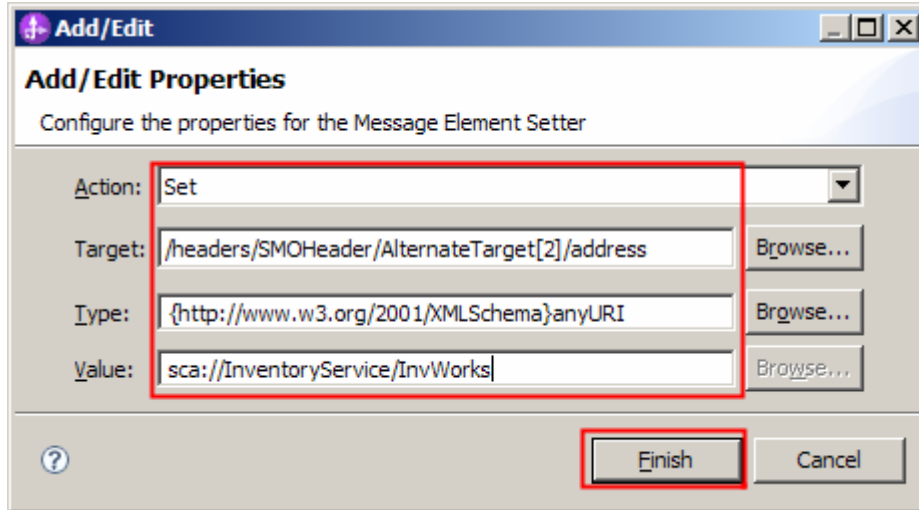
8) You will see the second entry in the Message Elements table:

Message Elements:

1	Set element <code>/headers/SMOHeader/Target/address</code> to <code>"sca://InventoryService/InvFails"</code>
2	Set element <code>/headers/SMOHeader/AlternateTarget[1]/address</code> to <code>"sca://InventoryService/InvRandom"</code>

\_\_\_ d. Finally, set /headers/SMOHeader/AlternateTarget[2]/address to the string URL value sca://InventoryService/InvWorks

- 1) Under Message Elements table, click the **Add...** button.
- 2) Using the same steps as previously, you should be able to initialize the Add/Edit dialog with these values:



- 3) Click **Finish**
- 4) Examine the Message Element table to ensure you have correctly defined the each of the elements to be set.

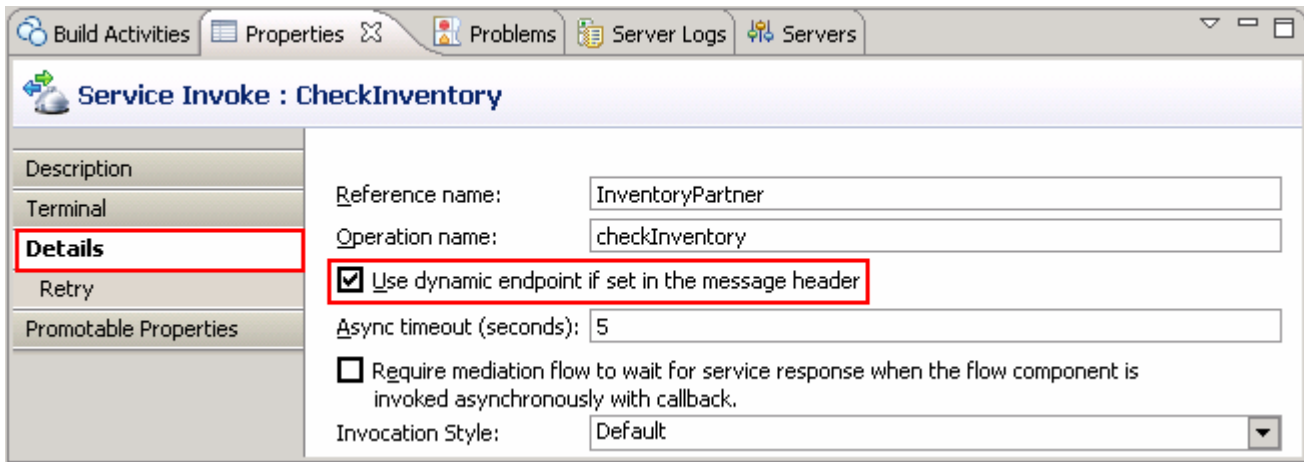
Message Elements:

1	Set element /headers/SMOHeader/Target/address to "sca://InventoryService/InvFails"
2	Set element /headers/SMOHeader/AlternateTarget[1]/address to "sca://InventoryService/InvRandom"
3	Set element /headers/SMOHeader/AlternateTarget[2]/address to "sca://InventoryService/InvWorks"

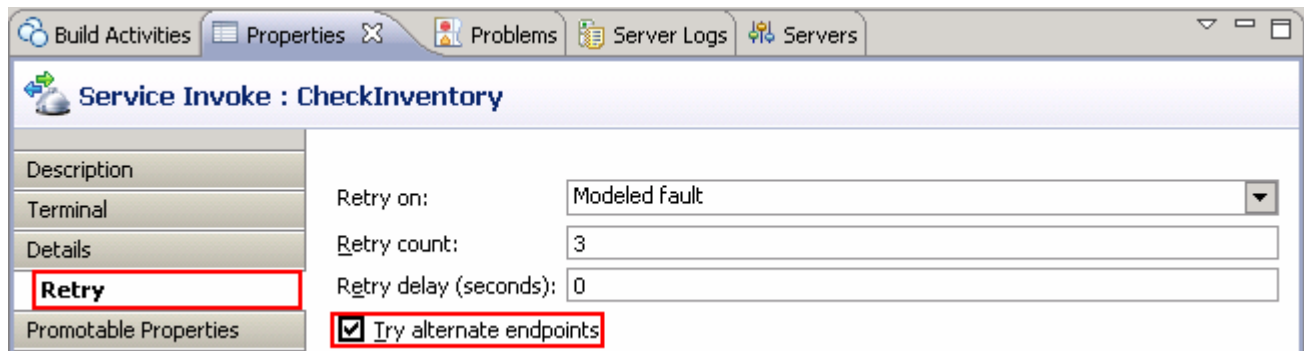
- \_\_\_ 4. Modify the CheckInventory service invoke primitive to take the endpoint URL from the target address field and to perform retries using the endpoint URLs from the alternate target address array.
  - **Details panel:** Check the "Use dynamic endpoint if set in message header" check box
  - **Retry panel:** Check the "Try alternate endpoints" check box
  - -----



- \_\_\_ a. From the mediation flow, select the **CheckInventory** service invoke primitive and then navigate to the **Details** panel of the **Properties** view
- \_\_\_ b. Check the box for 'Use dynamic endpoint if set in the message header'



- \_\_\_ c. Now navigate to the **Retry** panel
- \_\_\_ d. Check the box for **Try alternate endpoints**




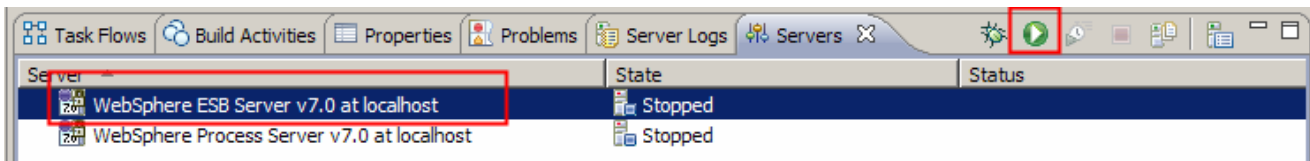
- \_\_\_ 5. Check that all artifacts have been saved.
  - -----
  - \_\_\_ a. From the menu select **File** → **Save All** to save your changes
  
- \_\_\_ 6. Check that there are no problems reported in the Problems view
  - -----
  - \_\_\_ a. Select **Problems** view at the bottom. You can ignore warnings, but there should not be any errors at this time:

## Part 3: Test the retry with alternate endpoints mediation flow

**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 resulting output is explained.

You should see the presentation entitled [Augmentation, aggregation and retry tutorials](#) to better understand what this part is doing.

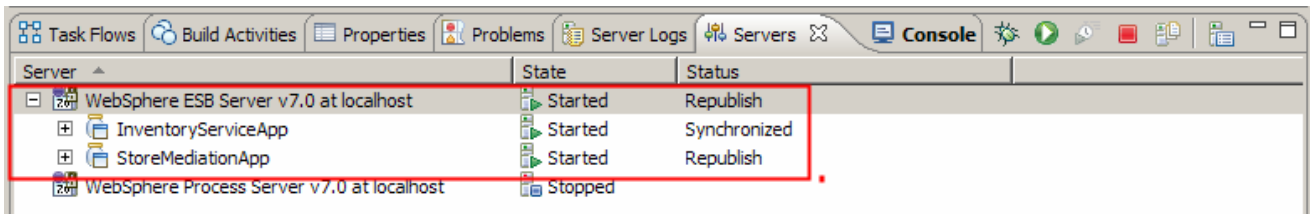
- \_\_\_ 1. If not already running, start the WebSphere Enterprise Service Bus (or WebSphere Process Server) test server.
  - -----
  - \_\_\_ a. From the **Servers** view of WebSphere Integration Developer, select **WebSphere ESB Server v7.0** and hit **Start the server** icon (  ) from the toolbar



- \_\_\_ b. Wait until the server Status shows as **Started**

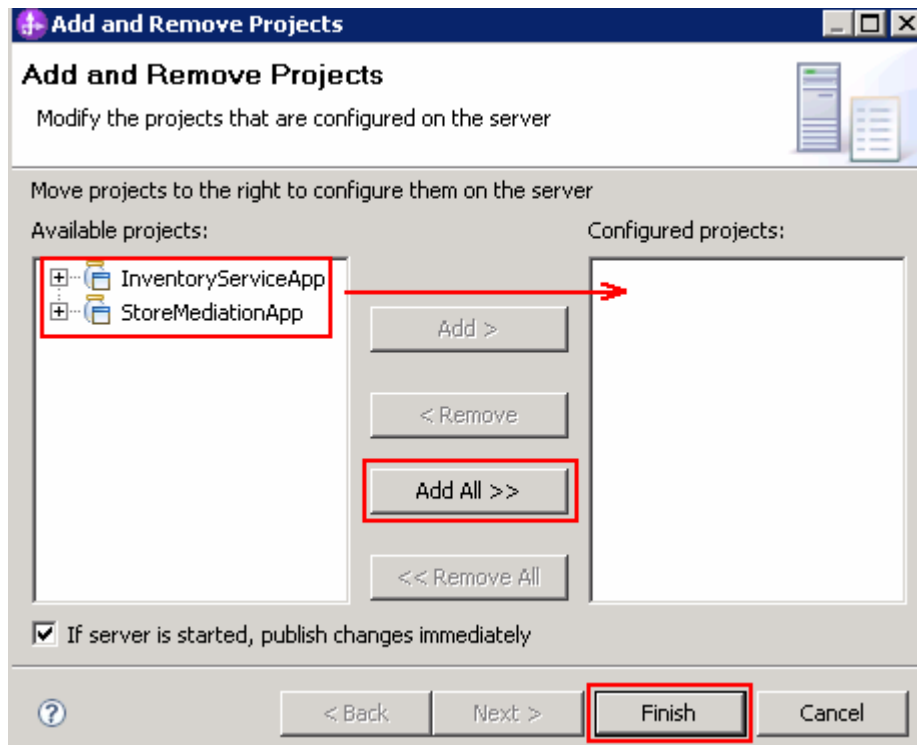
**NOTE:** Depending upon the preferences you have specified for the **Console** view, the **Console** view might grab the focus from the **Servers** view. If this is the case, the status in the lower right should indicate when the server start is complete, at which time you can switch back to the **Servers**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

- \_\_\_ 2. Check to see if the InventoryServiceApp and StoreMediationApp are deployed on the server.
  - **No, not deployed yet:** Add both projects to the server
  - **Yes, already deployed:** Republish to ensure latest versions are loaded
  - -----
  - \_\_\_ a. Check the Servers view to see if the InventoryServiceApp and StoreMediationApp are already deployed. If they are, they will appear below the server as shown here. **Follow the appropriate instructions in step b or step c**



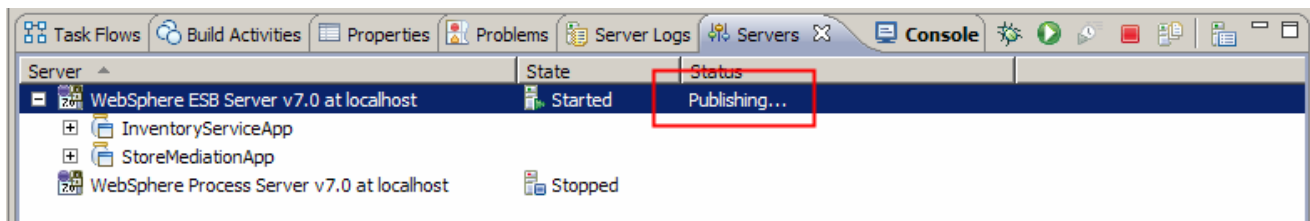
- \_\_\_ b. If the applications are **not deployed yet**, add both projects to the server following these steps
  - 1) Right-click **WebSphere ESB Server v7.0** under the Servers view and select **Add and remove projects...** from the context menu

- 2) In the Add and Remove Projects window, click **Add All >>** to add InventoryServiceApp and StoreMediationApp to the Configured projects panel

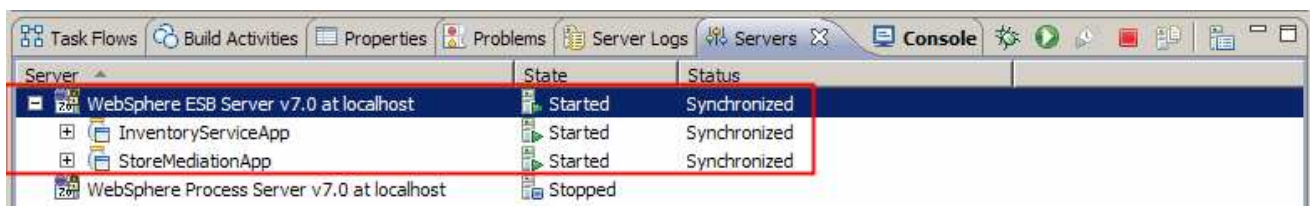


- 3) The projects will now be moved to Configured projects. Click **Finish**

- 4) Wait while the projects are being published to the server

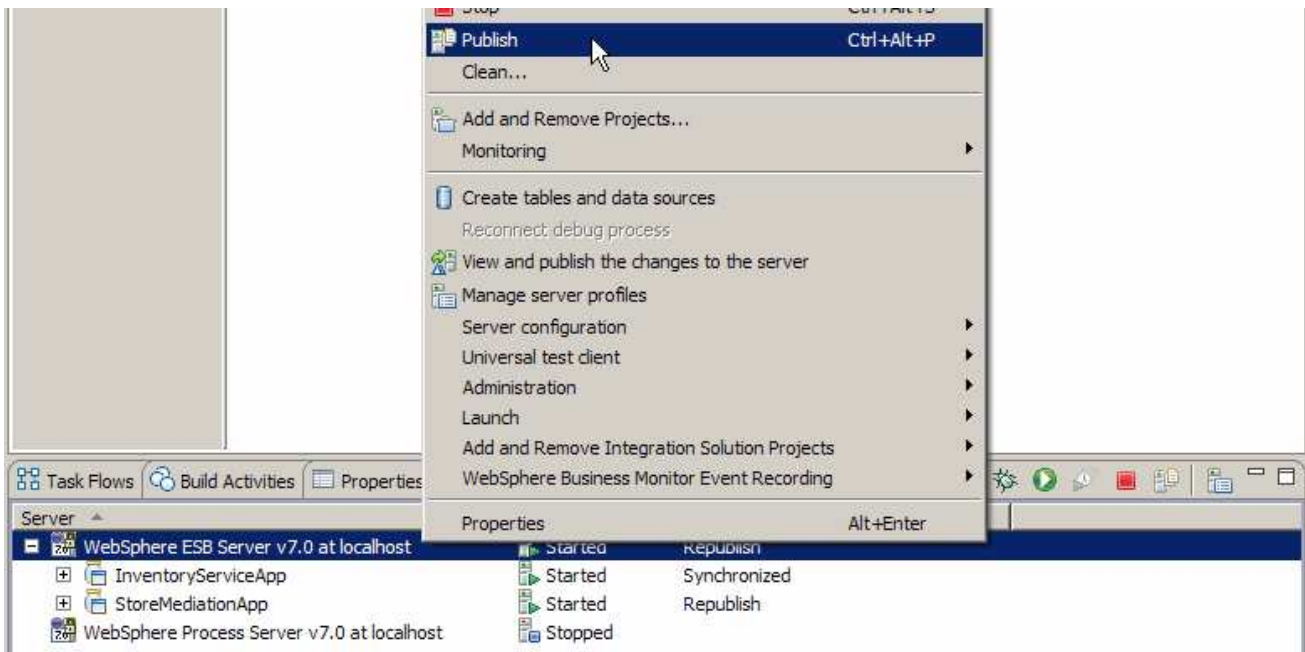


- 5) Once the publishing is done you should see the two applications started as shown here:

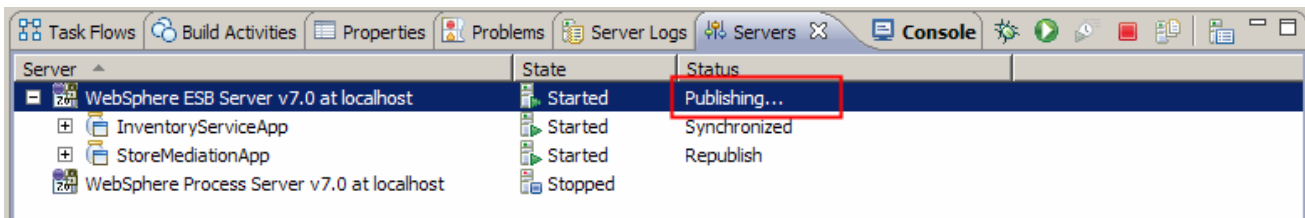


\_\_\_ c. If the applications are **already deployed**, a new publish needs to be done to load them into the server with the latest changes.

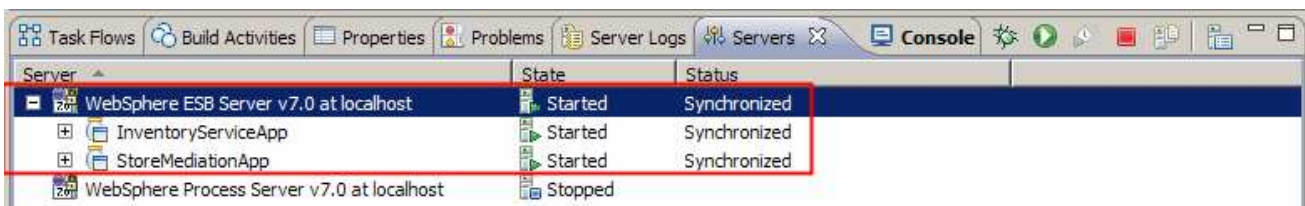
1) Right click the server and select **Publish** from the pop-up menu




2) Wait while the applications are publishing to the server



3) Once the publishing is done you should see the two applications started as shown here:



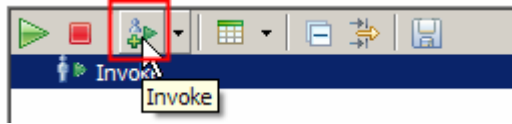
\_\_\_ 3. Check if the StoreMediation\_Test panel is still present from a previous lab.

- **No:** From the assembly diagram for StoreMediation, start Test Component for the StoreMediation component.
- **Yes:** Hit the Invoke icon (  ) in the Events panel
- -----

- \_\_\_ a. Look at the tabs to see if StoreMediation\_Test is still open as shown in this screen capture.  
**Follow the appropriate instructions in step b or step c**



- \_\_\_ b. **Yes, it is open.** Click the Invoke icon (  ) in the Events panel



- \_\_\_ c. **No, it is not open.** From the assembly diagram for StoreMediation, start Test Component for the StoreMediation component

- 1) In the Business Integration window, expand **StoreMediation** and double-click **Assembly Diagram** to open it in Assembly editor
- 2) From the StoreMediation-Assembly Diagram, right-click **StoreMediation** component and select **Test Component** from the pop-up menu
- 3) The **StoreMediation\_Test** window is opened where you will enter your test data

\_\_\_ 4. Initialize the test data

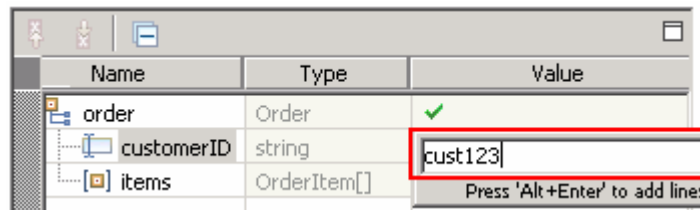
- Set `customerID` to `cust123`
- Set `items/items[0]/itemID` to `item001`
- Set `items/items[0]/quantity` to `3`
- Set `items/items[1]/itemID` to `item009`
- Set `items/items[1]/quantity` to `5`
- Set `items/items[2]/itemID` to `item002`
- Set `items/items[2]/quantity` to `15`

Name	Type	Value
order	Order	✓
customerID	string	✓ cust123
items	OrderItem[]	60
items[0]	OrderItem	✓
itemID	string	✓ item001
quantity	int	✓ 3
items[1]	OrderItem	✓
itemID	string	✓ item009
quantity	int	✓ 5
items[2]	OrderItem	✓
itemID	string	✓ item002
quantity	int	✓ 15

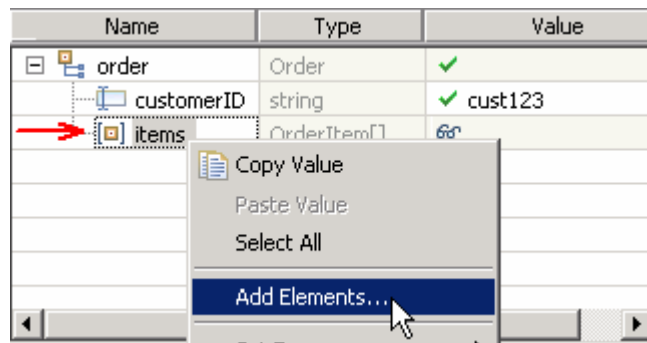
• -----

\_\_\_ a. Enter this under 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 **3** in the Add Element window:

4) Click **OK**

5) Enter values for items[0]:

a) For **itemID**, click under Value and enter **item001**

b) For **quantity**, click under Value and enter **3**

6) Enter values for items[1]:

a) For **itemID**, click under Value and enter **item009**

b) For **quantity**, click under Value and enter **5**

7) Enter values for items[2]:


a) For **itemID**, click under Value and enter **item002**

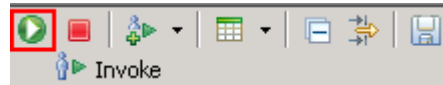
b) For **quantity**, click under Value and enter **15**

8) Your input data should now look like the table shown above.

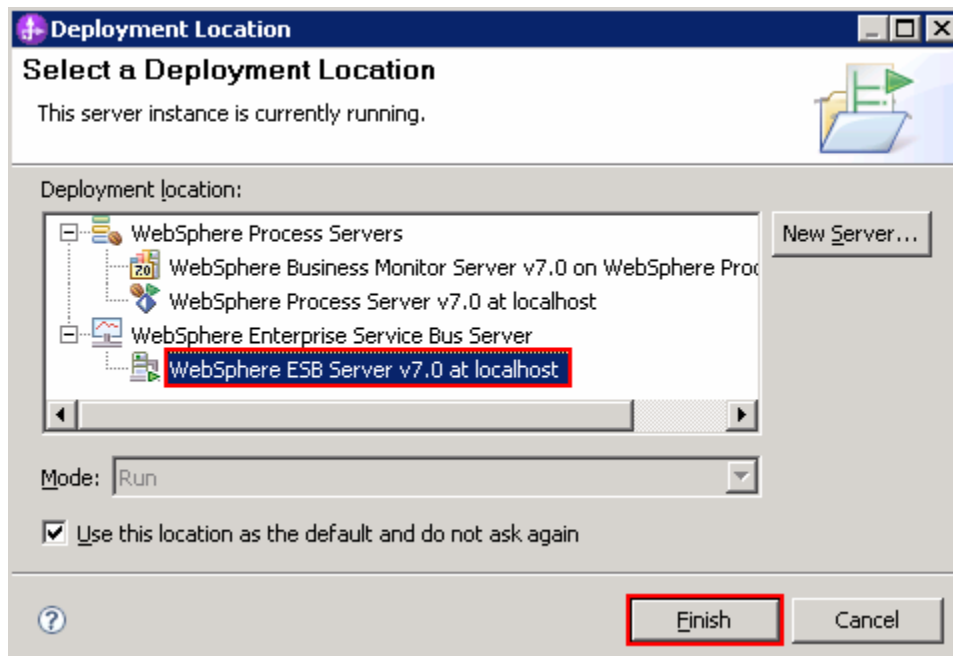
\_\_\_ 5. Run the test by hitting the Continue icon (▶)

• -----

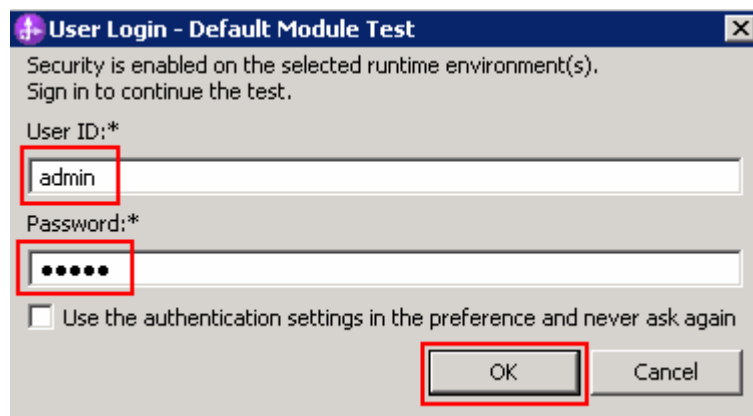
- \_\_\_ a. Click **Continue** icon (  ) under Events panel



- \_\_\_ b. If test client is run for the first time, you should see a **Deployment Location** dialog prompting to select a run time sever where the applications are deployed. Select **WebSphere ESB Server v7.0** from the list

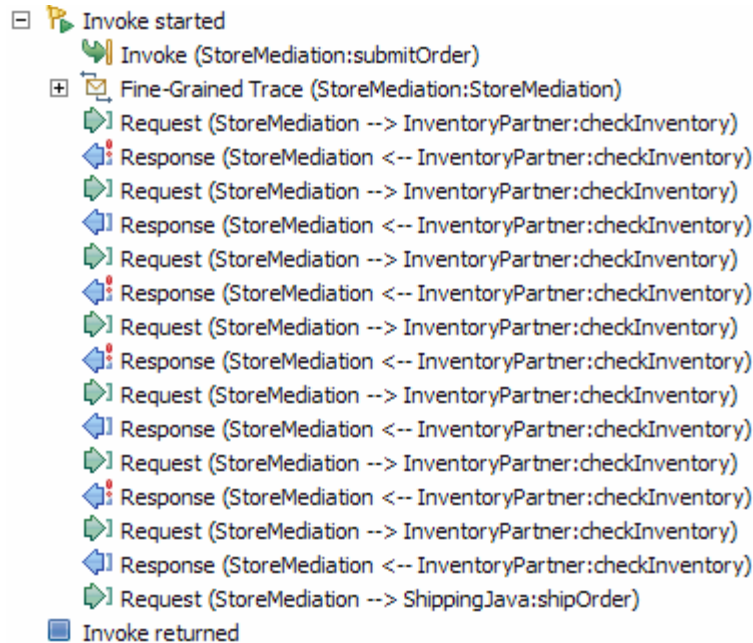


- \_\_\_ c. Click **Finish**
- \_\_\_ d. If presented with the **User Login** dialog, enter the **User ID** and **Password**, which by default is normally set to **admin** and **admin**. Optionally, you can select the 'Use the authentication settings in the preference and never ask again' check box to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



- \_\_\_ e. Click **OK**
- \_\_\_ f. Wait until the integration test client starts

6. The result of running should look similar to this. Notice that there are a total of seven calls made to the inventory service. Your result may be different, as the algorithm used in the lab can result in anywhere from six to nine calls being made. This is based on there being three items, the first call for each item will always fail, the second call might work, and if not, the third call will work. Therefore, each item results in two or three calls, yielding a total of six to nine calls for three items.




---

**NOTE:** The Fine-Grained Trace section has been collapsed. When expanded, it shows all of the nodes and primitives invoked by the flow. Although not discussed or explained in this lab, clicking on any of the entries within the fine grained trace will show the contents of the SMO at that point in the flow (in the Mediation Message panel to the right). This is a good way for you to see the changes in the SMO made by each of the primitives.

---

7. Switch to the Console view and examine the output, which should look similar to the screen capture on the following page. You will see several calls to the inventory service. Notice that there is a mixture of calls to the different inventory service implementations (InvFails, InvRandom, and InvWorks) because of the use of alternate endpoints. Your screen capture might be slightly different because of the use of the InvRandom implementation of the inventory service. The target address was set to InvFails so the first attempt to access the inventory service for each item should show an exception. The first alternate address was set to the InvRandom implementation, so the second call for each item will vary, with some working and others not. In the cases where the second call failed, the third call for the item is OK because the second alternate address was set to the InvWorks implementation. The final ship object sent to the shipping service should contain the three items, each initialized with its appropriate inventory information as the call to the inventory service eventually was successful for each item.



```

O *****
O ***** START mediation flow *****
O ***
I   processMessage
O ***** InvFails - EXCEPTION: InventoryFault for itemID = item001 item001
I   processMessage
E   ServiceMessageObject: ServiceMessageObject@d100d1 (context=ContextType@11
O ***** InvRandom - returning InventoryItem for itemID = item001
I   processMessage
O ***** InvFails - EXCEPTION: InventoryFault for itemID = item009 item009
I   processMessage
E   ServiceMessageObject: ServiceMessageObject@3aae3aae (context=ContextType@
O ***** InvRandom - EXCEPTION: InventoryFault for itemID = item009
I   processMessage
E   ServiceMessageObject: ServiceMessageObject@40d040d0 (context=ContextType@
O ***** InvWorks - returning InventoryItem for itemID = item009
I   processMessage
O ***** InvFails - EXCEPTION: InventoryFault for itemID = item002 item002
I   processMessage
E   ServiceMessageObject: ServiceMessageObject@5a255a25 (context=ContextType@
O ***** InvRandom - returning InventoryItem for itemID = item002
I   processMessage
O ***
O ***** END mediation flow *****
O *****
O -----
O -- Ship object dump begins -----
O
O EObject: com.ibm.ws.bo.bomodel.impl.DynamicBusinessObjectImpl@6c636c63
O Value:
O   customerID = cust123
O   items = ShipItem[3]
O     items[0] = <ShipItem@6c806c80>
O       itemID = item001
O       orderQuantity = 3
O       inventoryQuantity = 5
O       inventoryStatus = OK - but stock is running low
O     items[1] = <ShipItem@6c8d6c8d>
O       itemID = item009
O       orderQuantity = 5
O       inventoryQuantity = 45
O       inventoryStatus = OK - sufficient stock levels
O     items[2] = <ShipItem@6c9a6c9a>
O       itemID = item002
O       orderQuantity = 15
O       inventoryQuantity = 10
O       inventoryStatus = Backorder - insufficient stock to fill order
O
O -- Ship object dump ends -----
O -----

```

- -----
- \_\_\_ a. Double click **Console** view to maximize (lower quadrant of the work bench) for a better view of the server trace. Alternatively to launch the console view, from the main menu, select **Window** → **Show view** → **Console**



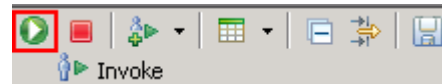
- \_\_\_ 8. If you like, you can run additional tests using different input data to see how the output varies. Key things to note about the data you use:
- **customerID** can be any string and should not have any particular affect on the results
  - The **items** array can have any number of elements
  - **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 **Invoke** icon (  ) under Events panel



- \_\_\_ b. Enter values for **customerID**, **itemID**, and **quantity** as per the previous instructions

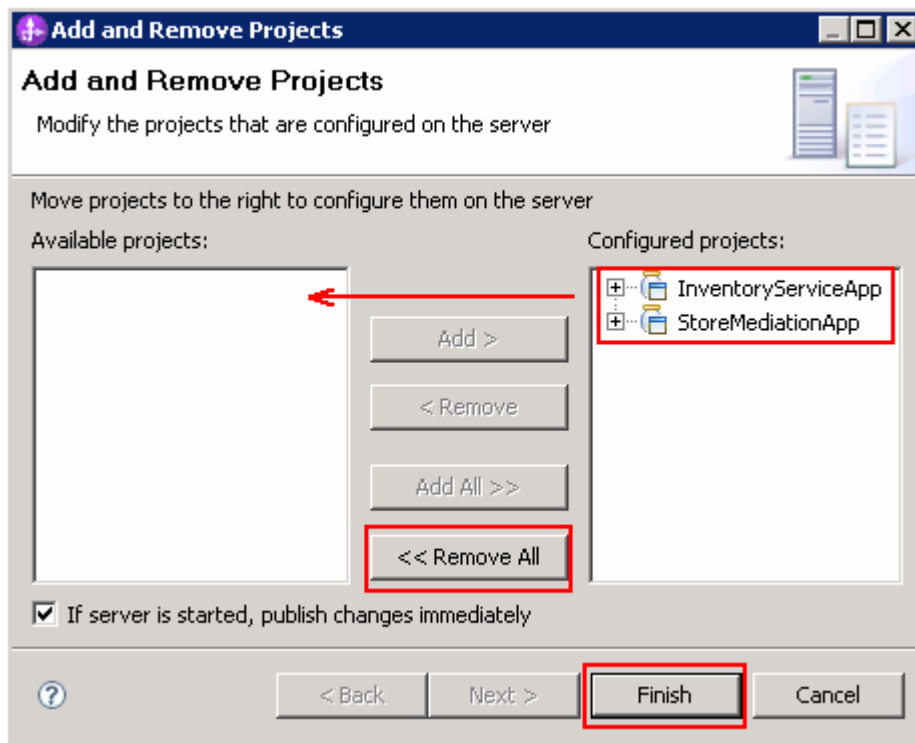
- \_\_\_ c. Click **Continue** icon (  ) under Events panel



## Part 4: Clean up the environment

**What you will do in this part:** Since this is the last lab in this sequence of labs, you should perform this part to clean up your development environment. This will leave your test server in a known state without any of the projects deployed on it.


- \_\_\_ 1. Remove the InventoryServiceApp and StoreMediationApp from the test server.
  - -----
  - \_\_\_ a. Right-click **WebSphere ESB Server v7.0** 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**

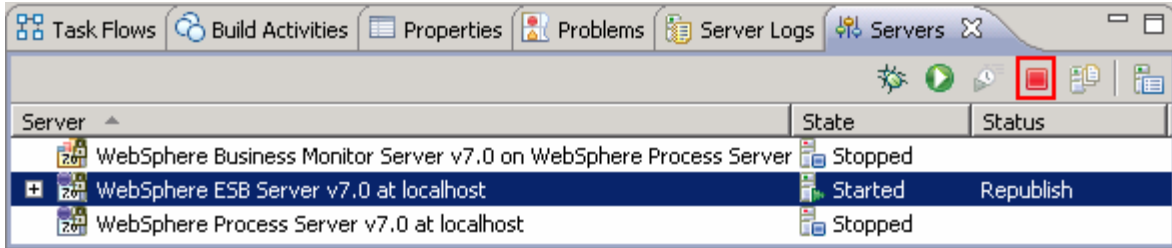


- \_\_\_ c. Click **Finish** after you see the applications moved to Available projects.
- \_\_\_ d. Wait until the application is removed from the server

- \_\_\_ 2. Close the StoreMediation\_test panel without saving
  - -----
  - \_\_\_ a. Close the StoreMediation\_Test client
  - \_\_\_ b. Click **No** from Save Test Trace window
- \_\_\_ 3. Close the **StoreMediation** – Assembly Diagram

\_\_\_ 4. Stop the test server

- -----
- \_\_\_ a. From the **Servers** view of WebSphere Integration Developer, select **WebSphere ESB Server v7.0** and hit **Stop the server** icon (  ) from the toolbar



- \_\_\_ b. Wait until the server Status shows as **Stopped**

\_\_\_ 5. Exit from WebSphere Integration Developer.

- -----
- \_\_\_ a. From the main menu, select **File → Exit**

## What you did in this exercise

In this exercise, you modified an augmentation and aggregation message flow so that the 'service invoke' primitive used alternate endpoints when performing a service call retry. This involved setting up the endpoints in the SMO and appropriately configuring the service invoke.

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

## 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>/PI5-AlternateEndpointsSolution.zip`
  
- \_\_\_\_ 2. Skip to **Part 3: Test the retry with alternate endpoints mediation flow** and proceed through the rest of the lab.