

IBM WebSphere® process integration 6.0.2 – Lab exercise

## End to end scenario

What this exercise is about .....	2
Lab requirements .....	2
What you should be able to do .....	2
Introduction .....	3
Exercise instructions .....	4
Part 1: Initialize the workspace for this lab exercise .....	5
Part 2: Understanding eXchange components .....	7
Part 3: Assemble eXchange .....	15
Part 4: Testing eXchange.....	24
Solution instructions .....	32

## What this exercise is about

The objective of this lab is to provide you with an understanding of how to use the WebSphere Integration Developer V6.0.2 tools to construct an end to end solution using existing components.

## Lab requirements

List of system and software required for the student to complete the lab.

- WebSphere Integration Developer V6.0.2 installed
- WebSphere Process Server V6 test environment installed
- Sample code in the directory C:\Labfiles602 (Windows®) or /tmp/LabFiles602 (Linux®)
- WebSphere Business Integration Toolset installed

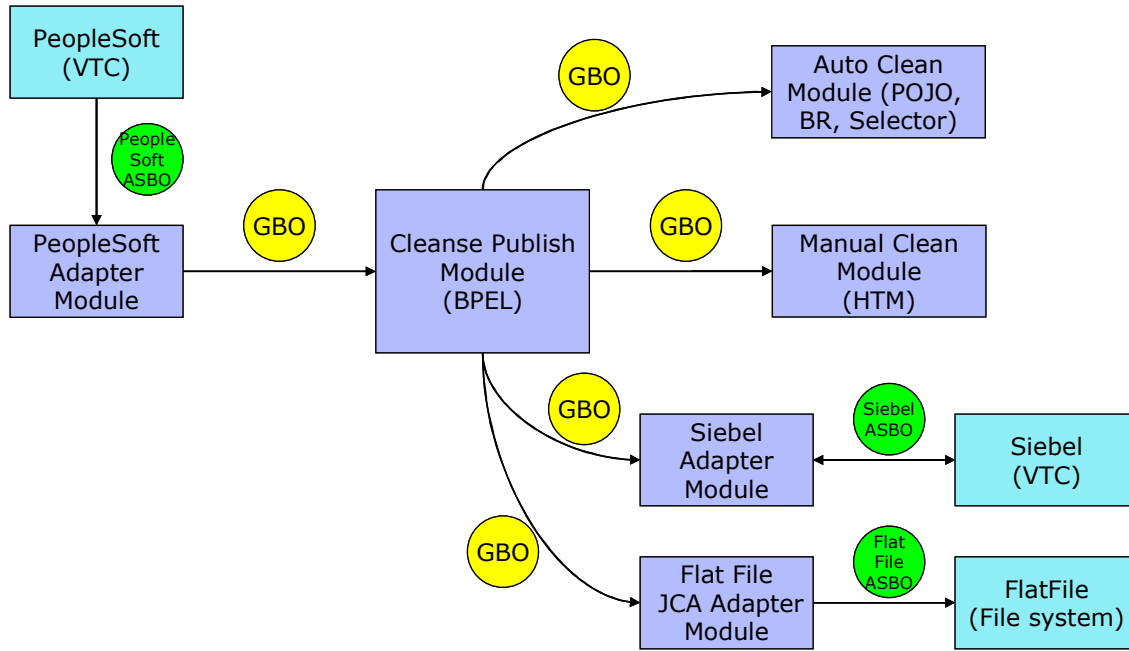
## What you should be able to do

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

- Create SCA Imports and Exports
- Define both SCA and JMS Pub/Sub bindings for the imports and exports
- Use the visual test connector to simulate adapter input and output to the process server

## Introduction

In this exercise, you will construct an end to end solution for the eXchange scenario. The overall flow of the solution is depicted in the following diagram.



All of the modules represented in the diagram are already implemented. However, the imports and exports required to connect the modules do not exist and you will be lead through creating them. After that, you will test the solution by driving input from the PeopleSoft adapter and then examine the output of the Siebel adapter.

**Note:** For this lab exercise, the flat file JCA module is not included, but could be easily added to this solution as separate module.

## Exercise instructions

Some instructions in this lab might be specific for Windows platforms. If you run the lab on a platform other than Windows, you will need to run the appropriate commands, and use appropriate files (for example .sh in place of .bat) for your operating system. The directory locations are specified in the lab instructions using symbolic references as follows:

Reference Variable	Windows Location	Linux Location
<WID_HOME>	C:\Program Files\IBM\WebSphere\ID\6.0	/opt/IBM/WebSphere/ID/6.0
<WPS_HOME>	<WID_HOME>\runtimes\bi_v6	<WID_HOME>/runtimes/bi_v6
<LAB_FILES>	C:\Labfiles602	/tmp/Labfiles602
<WORKSPACE>	C:\Labfiles602\exchange\EndToEndNoFFA\workspace	/tmp/Labfiles602/exchange/EndToEndNoFFA/workspace
<TEMP>	C:\temp	/tmp
<SOLUTION>	C:\Labfiles602\exchange\EndToEndNoFFA\Solution	/tmp/Labfiles602/exchange/EndToEndNoFFA/Solution

---

**Windows users' note:** When directory locations are passed as parameters to a Java program, such as wsadmin, you must replace the backslashes with forward slashes to follow the Java convention. For example, C:\LabFiles602\ would be replaced by C:/LabFiles602/.

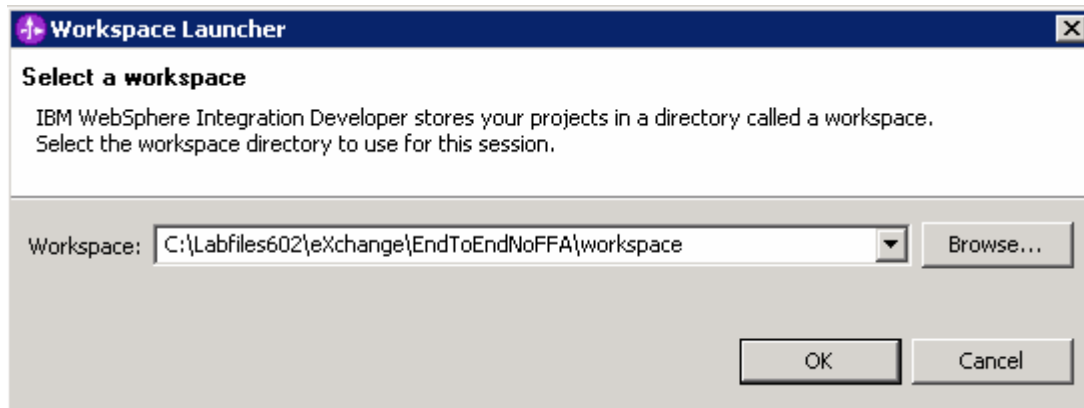
---

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

## Part 1: Initialize the workspace for this lab exercise

In this section of the lab, you will be importing a project interchange to prepare the environment.

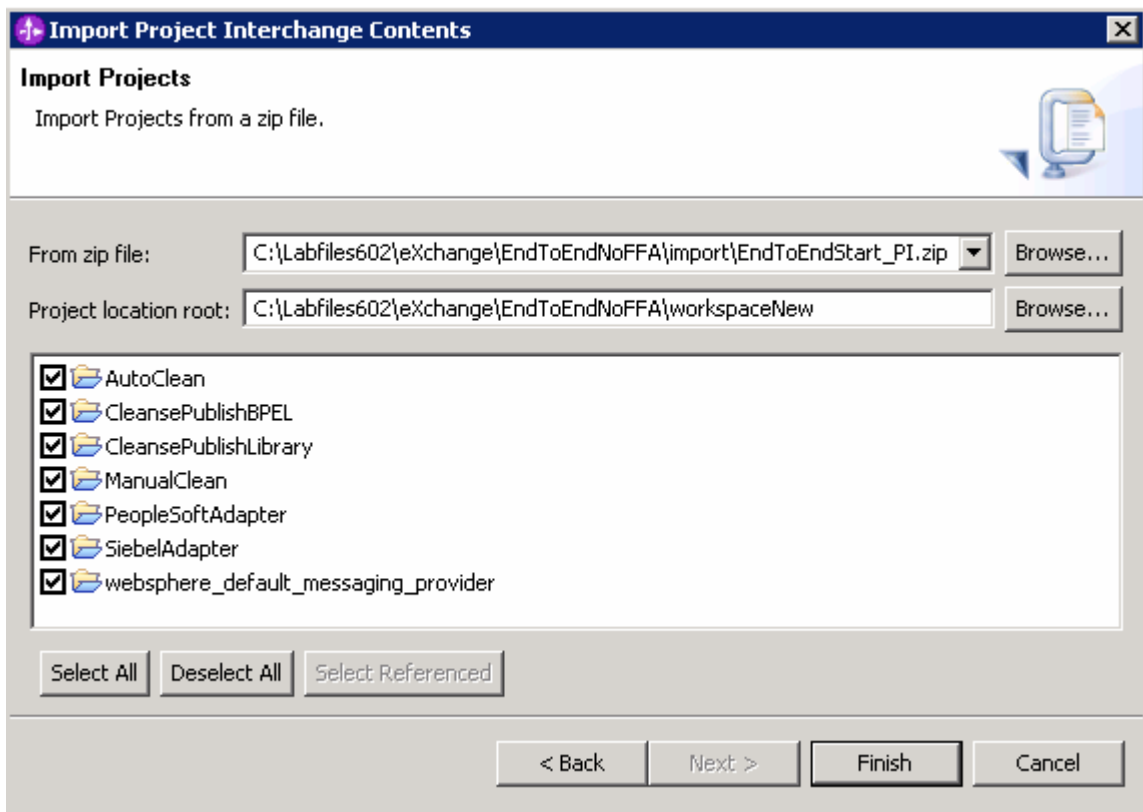
- \_\_\_ 1. Start WebSphere Integration Developer V6.0.2 with a workspace location of **<WORKSPACE>**, that is **<LAB\_FILES>\eXchange\EndToEndNoFFA\workspace**
  - \_\_\_ a. From Windows Explorer, navigate to the **<WID\_HOME>** directory and double click on wid.exe
  - \_\_\_ b. When prompted for workspace, enter the value provided by the **<WORKSPACE>** variable for this lab and click **OK**



- \_\_\_ c. When WebSphere Integration Developer V6.0.2 opens, click the curved arrow at top right to **go to Business Integration perspective**



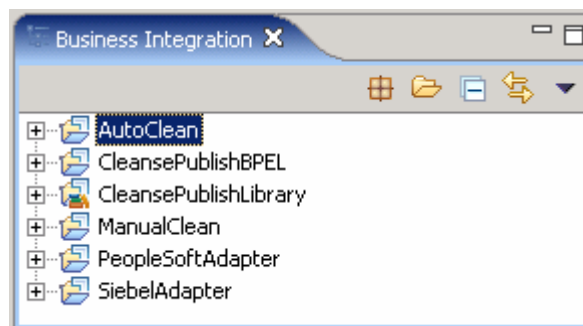
- \_\_\_ d. Ensure you are in the **Business Integration** perspective
- \_\_\_ 2. Import Project Interchange file, **EndToEndStart\_PI.zip** located at **<LAB\_FILES>\eXchange\EndToEndNoFFA\import\**
    - \_\_\_ a. Right-click inside **Business Integration view** (top left view in the Business Integration Perspective) and select **Import** from the context menu
    - \_\_\_ b. Select **Project Interchange** listed in the import dialog
    - \_\_\_ c. Click **Next**
    - \_\_\_ d. Click the **Browse** button for "From zip file" and navigate to **<LAB\_FILES>\eXchange\EndToEndNoFFA\import\EndToEndStart\_PI.zip** and hit **Open**



\_\_ e. Click the **Select All** button

\_\_ f. Click **Finish**

\_\_ g. Verify you have the following modules listed in the Business Integration view:

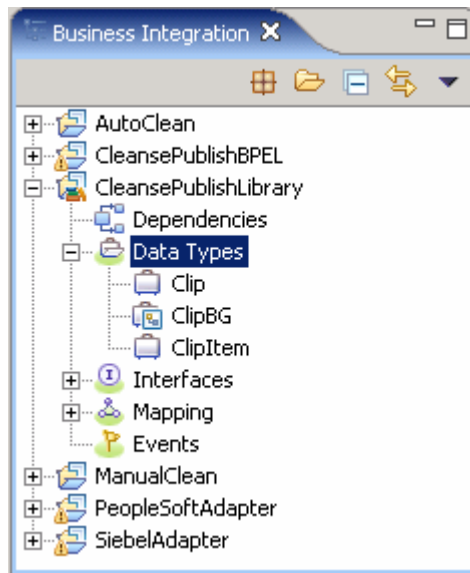


\_\_ h. Verify you have WebSphere Process Server V6.0 listed in your Servers view

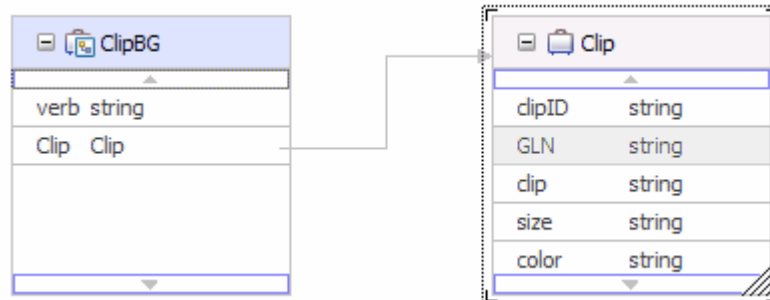
## Part 2: Understanding eXchange components

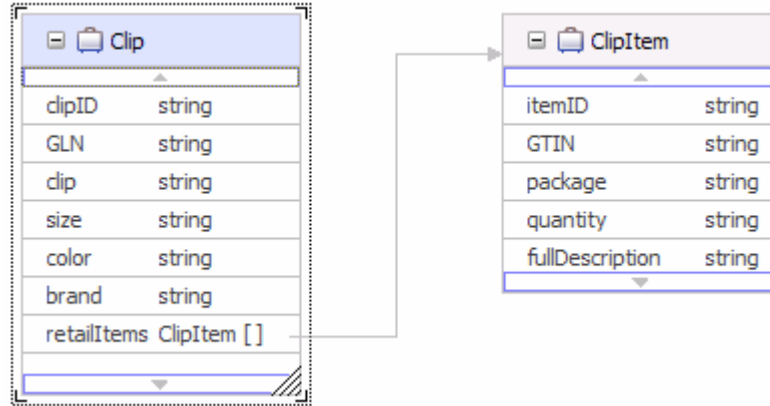
In this part you will examine the existing components that are part of the eXchange scenario. This will provide you with a basis to understand the solution you will be constructing in subsequent parts.

- \_\_\_ 1. The CleansePublishLibrary contains the common WSDL and XSD files used by the various components within the eXchange integration application. These files define the interfaces and data types used by components as they work with the requests received from the PeopleSoft adapter, cleanse the data, and publish it to the Siebel adapter. Examine the library
  - \_\_\_ a. In the Business Integration view, expand **CleansePublishLibrary** → **Data Types**



- \_\_\_ b. Open the **ClipBG** and **Clip** data types. Notice how ClipBG includes the Clip business object and Clip contains the ClipItem business object





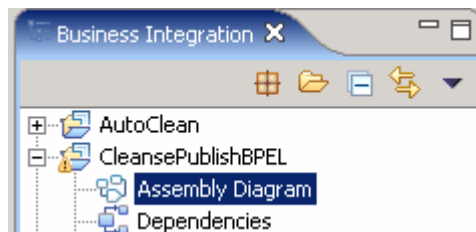
- \_\_\_ c. In the Business Integration view, expand **CleansePublishLibrary** → **Interfaces**
- \_\_\_ d. Open the interfaces listed and notice the operations and the messages. All messages are of the ClipBG type

▼ Define Operation(s)

Define Operations and their corresponding parameters

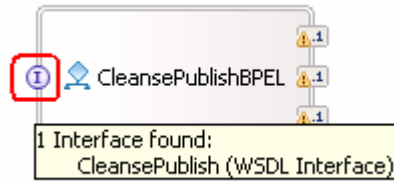
	Name	Type
Input(s)	inClipBG	ClipBG
Output(s)	outClipBG	ClipBG
Fault(s)	cleanseClipFault1	string

- \_\_\_ 2. The CleanPublishBPEL module provides the business logic for cleansing the data received from the PeopleSoft adapter before it is sent to the Siebel adapter. Examine the contents of the module
  - \_\_\_ a. Expand **CleansePublishBPEL** in the Business Integration view and double click on **Assembly Diagram** (Assembly Diagram) to open it in an Assembly Diagram editor. The business process has already been created and defined as a component

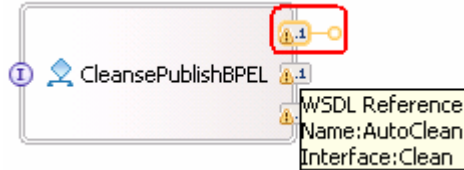


- \_\_\_ b. The business process has already been created and defined as a component within the module. It has an interface that can be called and 3 references to other service components. Hover the mouse over the interface ("I") icon on the left to view the interface, which defines how the business process is called. You should see that the **CleansePublish** interface has been specified






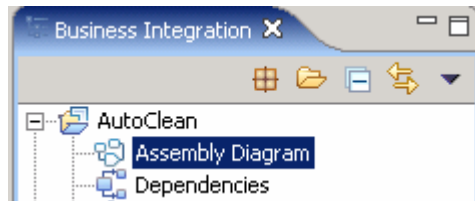
\_\_\_ c. Hover the mouse over the reference icons (“1..1”) to view the interfaces that are called. You should notice two references to the ‘Clean’ interface and one to the Publish interface



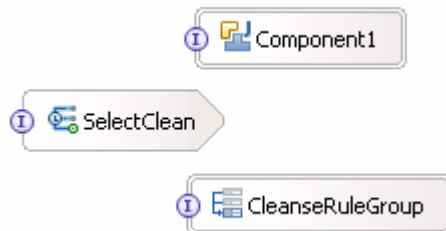
\_\_\_ d. Close the assembly editor

\_\_\_ 3. The AutoClean module provides an automated means of cleansing the data within the business logic. The cleansing is done either by a Business Rule or a Java™ component with a Selector component making the determination. Examine the contents of the module

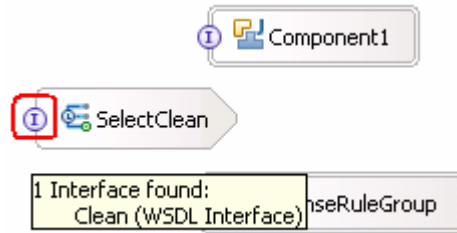
\_\_\_ a. Expand **AutoClean** in the Business Integration view and double click on **Assembly Diagram** (  Assembly Diagram ) to open it in an Assembly Diagram editor



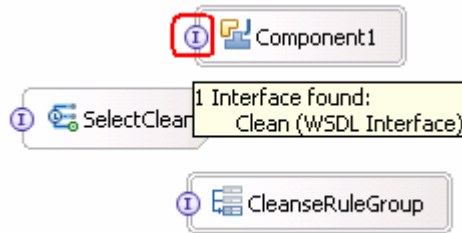
\_\_\_ b. The AutoClean module contains a Selector, Business Rule, and a Java class. All of these have been defined as components. The Selector (**SelectClean**) is the main component called to determine if the cleansing should be performed by a Business Rule (**CleanseRuleGroup**) or a Java component (**Component1**). The Selector is linked to the Business Rule and Java component through dynamic wires, which are not shown in the assembly editor. The Selector handles the incoming requests to the module and determines if the Business Rule or the Java component should be called to perform the auto-cleansing of the incoming data



\_\_\_ c. Hover the mouse over the interface icon for the Selector. Notice the Selector uses the ‘Clean’ interface.



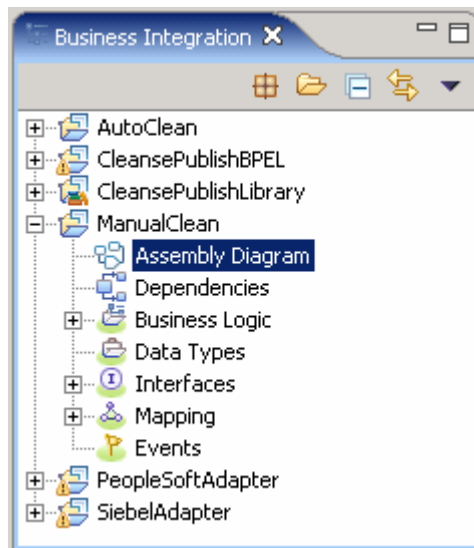
\_\_\_ d. Hover the mouse over Component1 (Java) and the CleanseRuleGroup components. Notice these also use the 'Clean' interface. With all components using the same interface, the data can be easily passed from one component to another within the module



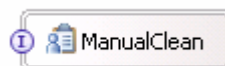
\_\_\_ e. Close the assembly editor

\_\_\_ 4. The ManualClean module provides the ability for human interaction to be part of the cleansing process. Examine the contents of the module.

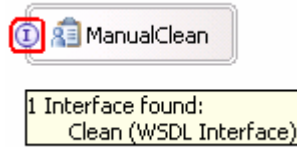
\_\_\_ a. Expand **ManualClean** in the Business Integration view and double click on **Assembly Diagram** (Assembly Diagram) to open it in an Assembly Diagram editor



\_\_\_ b. The ManualClean module contains a human task component. The human task, **ManualClean**, allows for a person to be part of the data cleansing




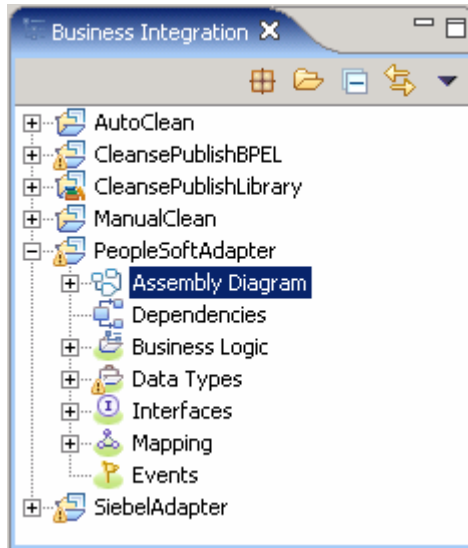
- \_\_\_ c. Hover the mouse over the interface icon. Notice the **Clean** interface is used to call the human task component



- \_\_\_ d. Close the assembly editor

- \_\_\_ 5. The PeoplesoftAdapter module provides the capability to accept requests and pass the data on to the business logic. Examine the contents of the module.

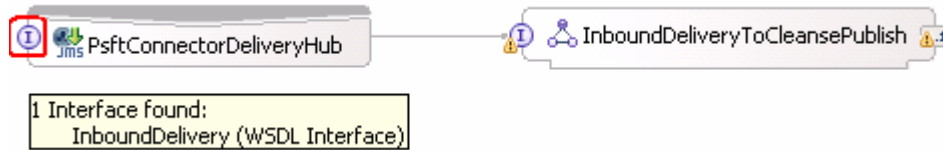
- \_\_\_ a. Expand **PeopleSoftAdapter** in the Business Integration view and double click on **Assembly Diagram** (  **Assembly Diagram** ) to open it in an Assembly Diagram editor



- \_\_\_ b. The PeopleSoftAdapter module is used to accept requests from the PeopleSoft adapter and convert an application specific business object (ASBO) to a generic business object (GBO), which is passed. The transformation is completed using interface and data type maps (**InboundDeliveryToCleansePublish**). In order to accept the request from the PeopleSoft adapter, an Export, **PsftConnectorDeliveryHub**, with a JMS binding is used on the maps. The connection between the map and Export is represented by the wire



- \_\_\_ c. Hover the mouse over the interface of the Export. The interface used is named **InboundDelivery** and is specific to interaction with the PeopleSoft adapter. The interface definition is contained within the PeopleSoftAdapter module as it is specific for this module and is not shared with other modules

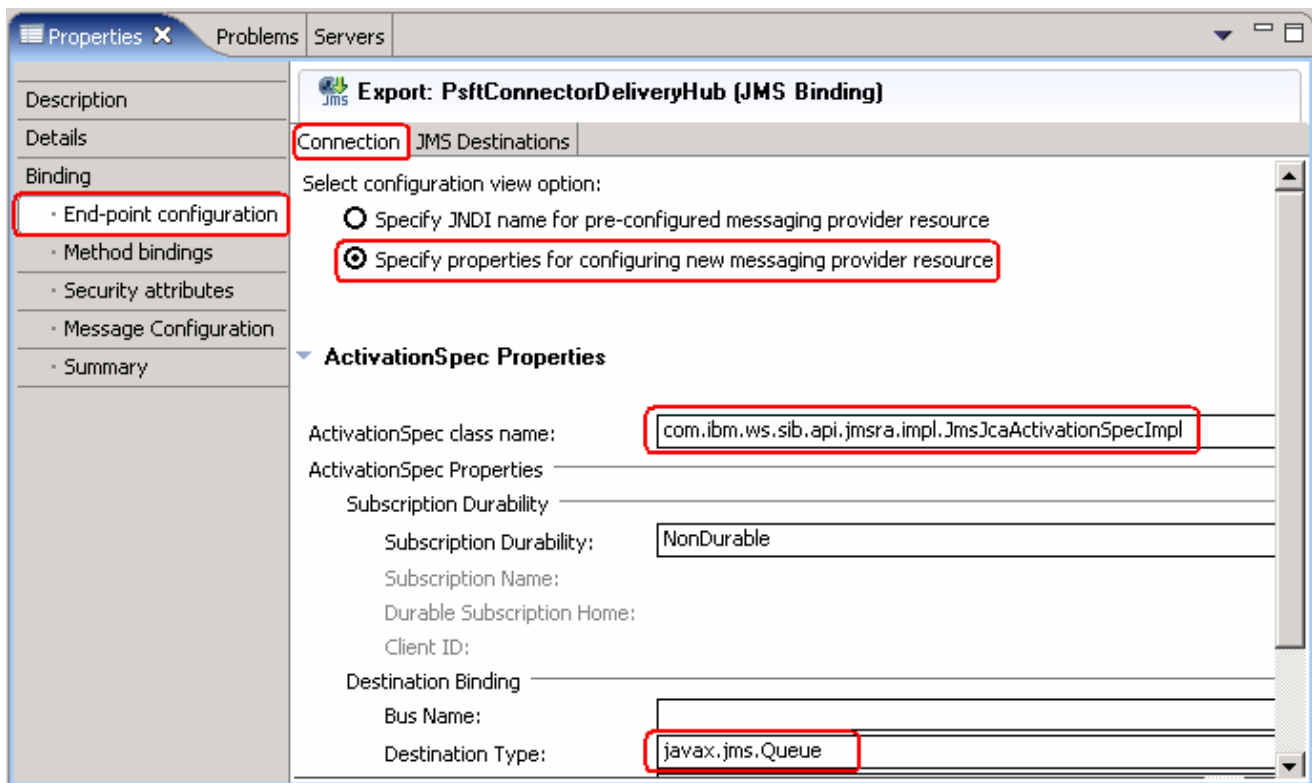


\_\_\_ d. Hover the mouse over the reference on the map. Notice the reference is to the **CleansePublish** interface which is the same as used by the business process



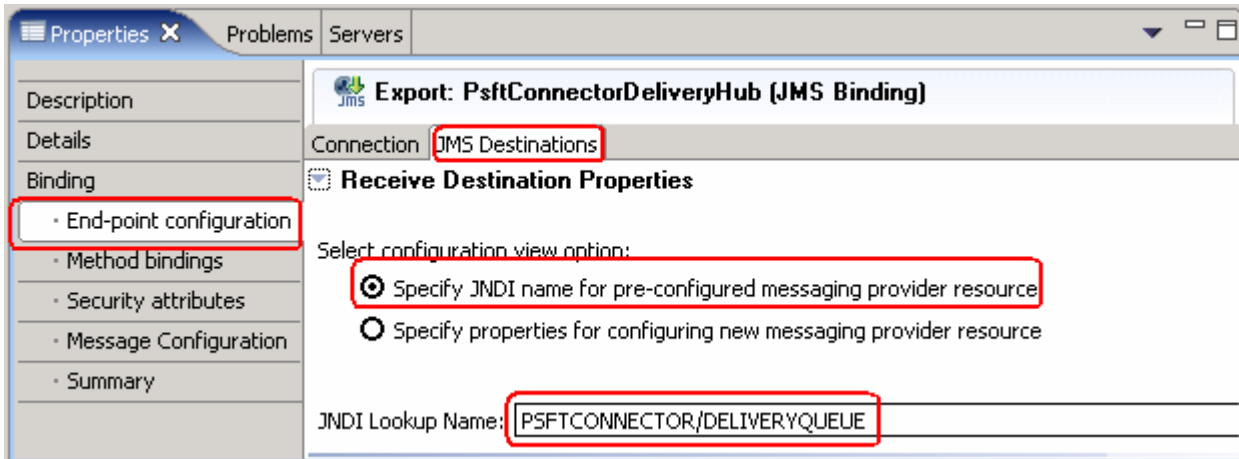
\_\_\_ e. Select the Export **PsftConnectorDeliveryHub** and select the Properties Tab

\_\_\_ f. Select the **Bindings** → **End-point configuration** tab and then select the **Connection** tab. Notice the JMS binding information specified for the module as it receives requests off a queue



\_\_\_ g. While you are here, select the **JMS Destinations** tab, and expand the **Receive Destinations Properties**

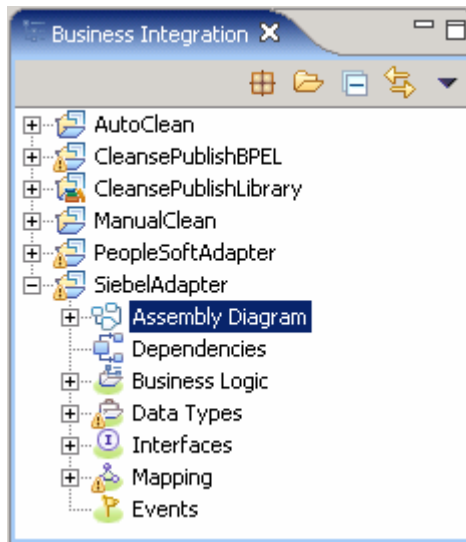
\_\_\_ h. The JNDI name of the queue, **PSFTCONNECTOR/DELIVERYQUEUE**, is set for the Export



\_\_\_ i. Close the assembly editor

\_\_\_ 6. The SiebelAdapter module provides the capability to receive cleansed data passed from the business logic and pass it onto a Siebel adapter. Examine the contents of the module

\_\_\_ a. Expand **SiebelAdapter** in the Business Integration view and double click on **Assembly Diagram** (Assembly Diagram) to open it in an Assembly Diagram editor



\_\_\_ b. The SiebelAdapter module is used to accept requests and transform the data from a generic business object (GBO) to an application specific business object (ASBO) before it is passed to the Siebel adapter. The transformation is completed using interface and data type maps (**PublishToOutboundRequest**). In order to send the request to the Siebel adapter, a JMS binding is used on an Import (**SiebelConnectorAgent**). The connection between the map and Import is represented by the wire



\_\_\_ c. Hover the mouse over the interface for the Import. The interface (**OutboundRequest**) is specific to interacting with the Siebel Adapter. The interface definition is contained within the SiebelAdapter module as it is specific for this module and is not shared with other modules



\_\_\_ d. Hover the mouse over the interface of the map. Notice the **Publish** interface is used.

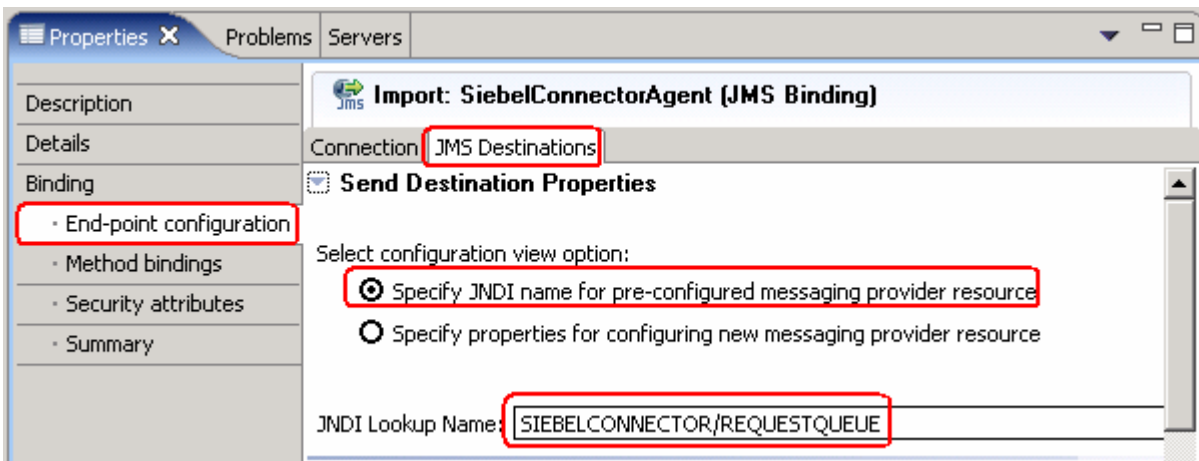


\_\_\_ e. Select the Import **SiebelConnectorAgent** and select the **Properties** Tab

\_\_\_ f. Select the **Bindings** → **End-point configuration** tab and then select the **Connection** tab. Notice the JMS binding information specified for the module, required to send messages to a queue

\_\_\_ g. Select the **JMS Destinations** tab, and expand the **Send Destinations Properties**


\_\_\_ h. The JNDI name of the queue (**SIEBELCONNECTOR/REQUESTQUEUE**) is set for the Import

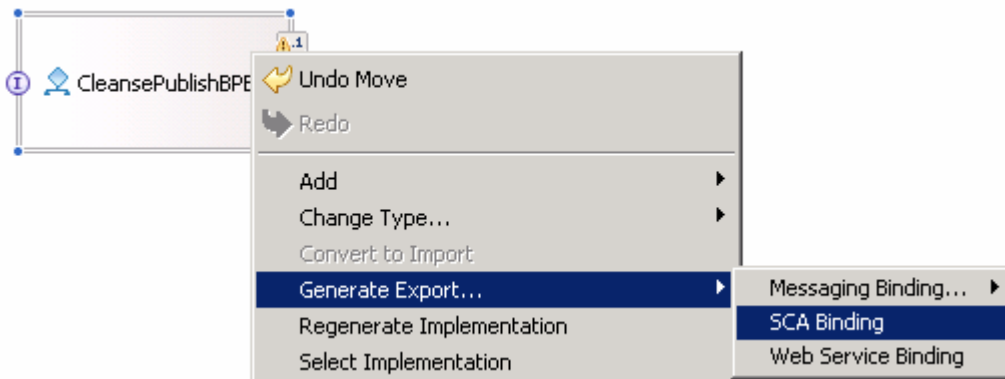


\_\_\_ i. Close the assembly editor

## Part 3: Assemble eXchange

SCA components in different modules that support the same interfaces can be associated with each other through the use of Imports, Exports, and their associated binding. In this part you will add the required Imports, Exports and bindings to build an end to end solution from the components you examined in the previous part of this lab exercise.

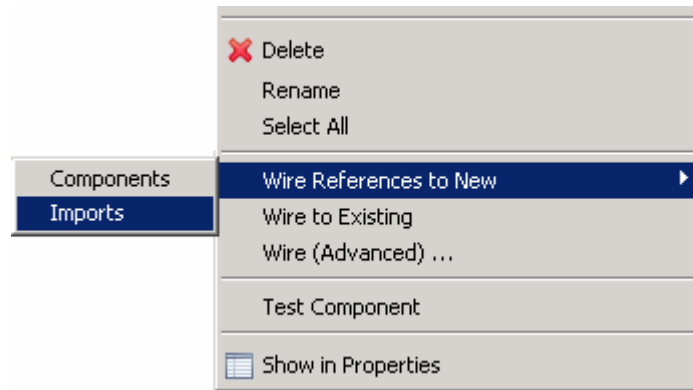
- \_\_\_ 1. The CleansePublishProcess has the most bindings and references that must be resolved. Start the overall assembly with defining the references and bindings for the CleanPublishBPEL module
  - \_\_\_ a. Expand **CleansePublishBPEL** in the Business Integration view and double click on **Assembly Diagram** (  **Assembly Diagram** ) to open it in an Assembly Diagram editor
  - \_\_\_ b. Right-click on the **CleanPublishBPEL** and select “**Generate Export... →SCA Binding**’ from the context menu. In order for the business process to be called as an SCA component from the PeopleSoft adapter an SCA binding must be created on an Export



- \_\_\_ c. Hover the mouse over the interface on the Export. Notice how it exposes the CleansePublish interface for SCA invocation



- \_\_\_ d. The references will also use SCA to call the AutoClean, ManualClean, and Publish components. These components are located in other modules and are called using Imports. An easy way to quickly generate Imports is to right-click on the CleansePublishProcess and select **Wire References to New → Imports** from the context menu



\_\_\_ e. The Imports will be created and will use the reference name for the Import component name.



\_\_\_ f. Save and close the editor for now. The SCA bindings must be set for each Import component to call the appropriate module. The SCA binding can not be completed until the AutoClean, ManualClean, and Publish modules are properly exposed as SCA components. You will return to the CleanPublishBPEL assembly editor and set these SCA bindings later

\_\_\_ 2. The AutoClean module only needs a single binding to be generated for calls from the CleansePublishProcess. Assemble the AutoClean module with an SCA binding.

\_\_\_ a. Expand **AutoClean** in the Business Integration view and double click on **Assembly Diagram** (Assembly Diagram) to open it in an Assembly Diagram editor


\_\_\_ b. Right-click on **SelectClean** and select '**Generate Export... →SCA Binding**' from the context menu. The Selector is the only component in the module that needs to be exposed as an SCA component



\_\_\_ c. Save and close the assembly editor. The Selector, which fronts the calls to the module and determines whether to call the Java component or Business Rule, is fully available to be called as an SCA component

\_\_\_ 3. The ManualClean module also only needs a single binding to be generated for calls from the CleansePublishProcess. Assemble the ManualClean module




\_\_\_ a. Expand **ManualClean** in the Business Integration view and double click on **Assembly Diagram** (  **Assembly Diagram** ) to open it in an Assembly Diagram editor

\_\_\_ b. Right-click on **ManualClean** and select '**Generate Export...→SCA Binding**' from the context menu. The human task can be easily exposed as an SCA component as well

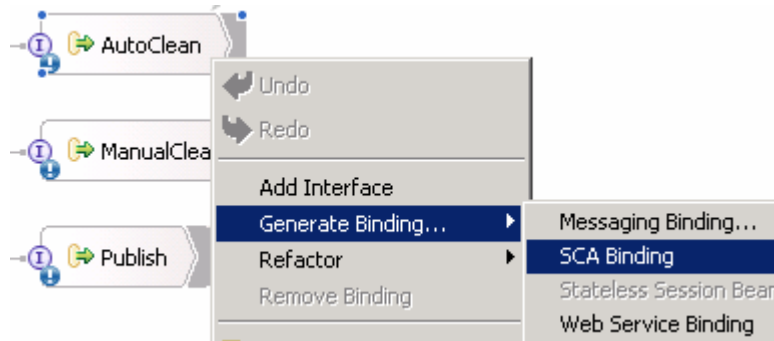


\_\_\_ c. Save and close the editor. The human task is complete as an SCA component

\_\_\_ 4. Set the SCA bindings in the business process. With the AutoClean and ManualClean SCA components defined, the SCA bindings can be defined for the references in the business process.

\_\_\_ a. Expand **CleansePublishBPEL** in the Business Integration view and double click on **Assembly Diagram** (  **Assembly Diagram** ) to open it in an Assembly Diagram editor

\_\_\_ b. Right-click the **AutoClean** Import component and select **Generate Binding... → SCA Binding**

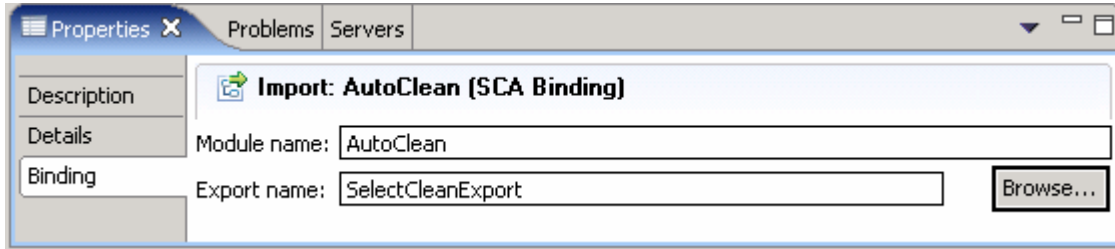


\_\_\_ c. With the AutoClean Import component selected, select the **Bindings** tab in the **Properties** view

\_\_\_ d. Click the **Browse** button. The different SCA components which are available will be listed. Notice that it is only of those modules where the SCA Export has been defined

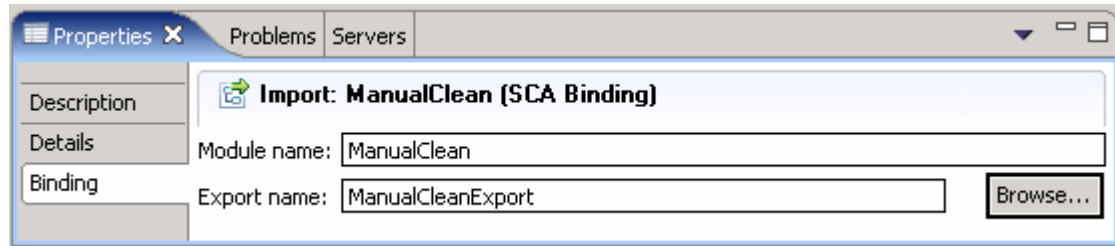


- \_\_\_ e. Select **SelectCleanExport** and click **OK**. The binding is complete for the business process to call AutoClean

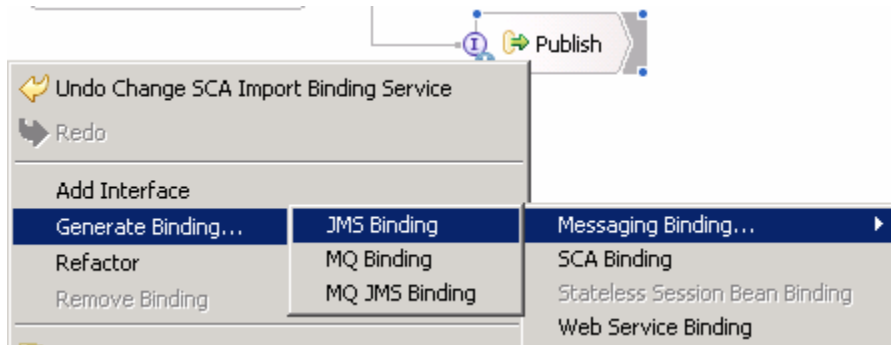


- \_\_\_ f. Right-click on **ManualClean** and select **Generate Binding... → SCA Binding**. A similar set of steps can be used for setting the SCA binding on the ManualClean Import component.

- \_\_\_ g. Browse to and select **ManualCleanExport** on the Bindings tab in the Properties view. This will complete the SCA binding for ManualClean.



- \_\_\_ h. For the binding of the Publish Import Component, JMS will be used instead of SCA. In order to support the publish of data to the Siebel adapter; a JMS Topic will be used. Set the JMS binding by right-clicking on **Publish** and selecting '**Generate Binding... → Messaging Binding → JMS Binding**' from the context menu



- \_\_\_ i. Select **Publish-Subscribe** for the Select JMS message domain. Select **Business Object XML using JMSTextMessage** for the data is serialized between Business Object and JMS Message in the **JMS Import Binding attributes selection** dialog

**Configure JMS Import Service**

**JMS Import Binding**

The configuration properties represent the minimum set required to deploy and execute JMS Import Service

**Messaging model**

JMS messaging domain: Publish-Subscribe

**End-point configuration**

Configure new messaging provider resources

Use pre-configured messaging provider resources

JNDI name for connection factory: CleansePublishBPEL/Publish\_CF

JNDI name for send destination: CleansePublishBPEL/Publish\_SEND\_D

**Security configuration**

J2C Authentication Data Entry:

**Data format**

Serialization type: Business Object XML using JMSMessage

Data binding class: com.ibm.websphere.sca.jms.data.impl.JMSDataBindingImplXML Browse...

**Function selector**

Generate "TargetFunctionName" message header property for default JMS Function Selector

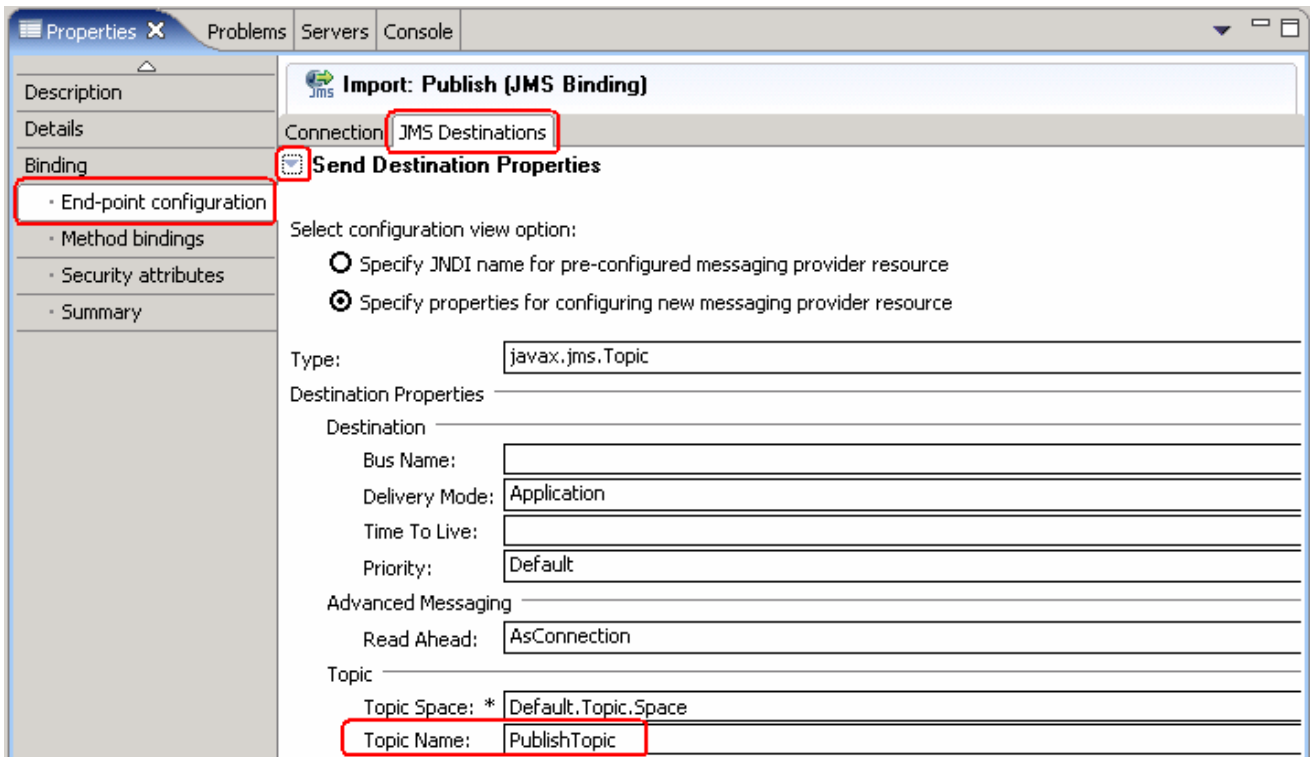
OK Cancel

\_\_\_ j. Click **OK**

\_\_\_ k. Select the **Bindings** → **End-point configuration** tab and then select the **JMS Destinations** tab in the **Properties** view to access the properties for the Publish Import. The specific topic where the message will be published must be specified to complete the binding

\_\_\_ l. Click on **Send Destination Properties**

\_\_\_ m. At the bottom enter **PublishTopic** for the Topic name. This will cause a topic to be dynamically created using the Service Integration Technologies. If a predefined topic exists or is located on a different server, the JNDI name can be specified for the JNDI lookup name



\_\_\_ n. Save and close the assembly editor. The binding for the Publish Import component is complete and the CleanPublishBPEL component now has bindings for all of its imports and exports

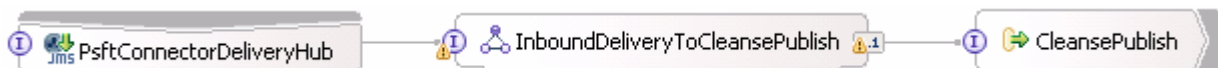


\_\_\_ o. Save and close the Assembly Diagram

\_\_\_ 5. The only binding needed for the PeopleSoftAdapter module is to the CleansePublishProcess

\_\_\_ a. Expand **PeopleSoftAdapter** in the Business Integration view and double click on **Assembly Diagram** (Assembly Diagram) to open it in an Assembly Diagram editor

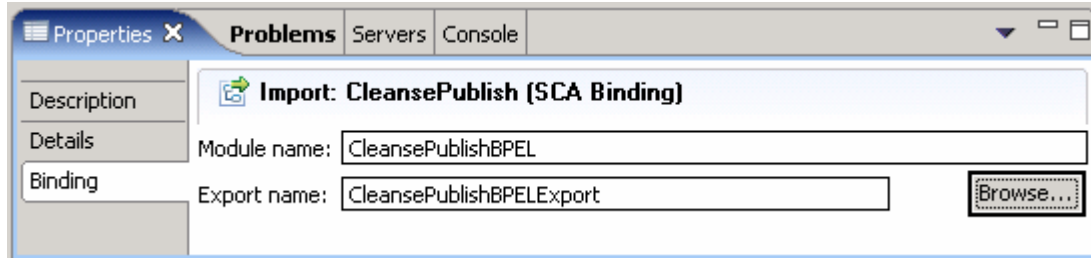
\_\_\_ b. Right-click on **InboundDeliveryToCleansePublish** and select **Wire References to New → Imports**.



\_\_\_ c. To create the binding to the CleansePublishProcess SCA component, an Import component must be used from the map. The SCA binding must be used. Right-click on the CleansePublish Import component and select **Generate Binding... → SCA Binding**



- \_\_\_ d. Browse to and select **CleansePublishBPELExport** on the Bindings tab in the Properties view. This will complete the SCA binding for PeopleSoftAdapter module

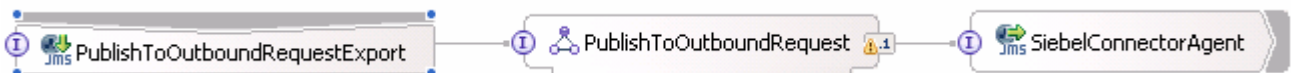


- \_\_\_ e. Save and close the assembly editor

- \_\_\_ 6. The Siebel Adapter must be able to accept the data (ClipBG) published from the CleansePublishProcess to the topic. A JMS binding to the topic must be set. Assemble the SiebelAdapter module to utilize the topic to receive the updated data.

- \_\_\_ a. Expand **SiebelAdapter** in the Business Integration view and double click on **Assembly Diagram** (Assembly Diagram) to open it in an Assembly Diagram editor
- \_\_\_ b. Right-click on PublishToOutboundRequest and select **Generate Export... → Message Binding → JMS Binding**. The JMS binding is set on the Export component for the **PublishToOutboundRequest** map component as the generic business object (GBO) will be converted to the application specific business object (ASBO) for the Siebel adapter
- \_\_\_ c. Select **Publish-Subscribe** for the Select JMS message domain. Select **Business Object XML using JMSTextMessage** for the data is serialized between Business Object and JMS Message, in the **JMS Import Binding attributes selection** dialog

\_\_ d. Click **OK**

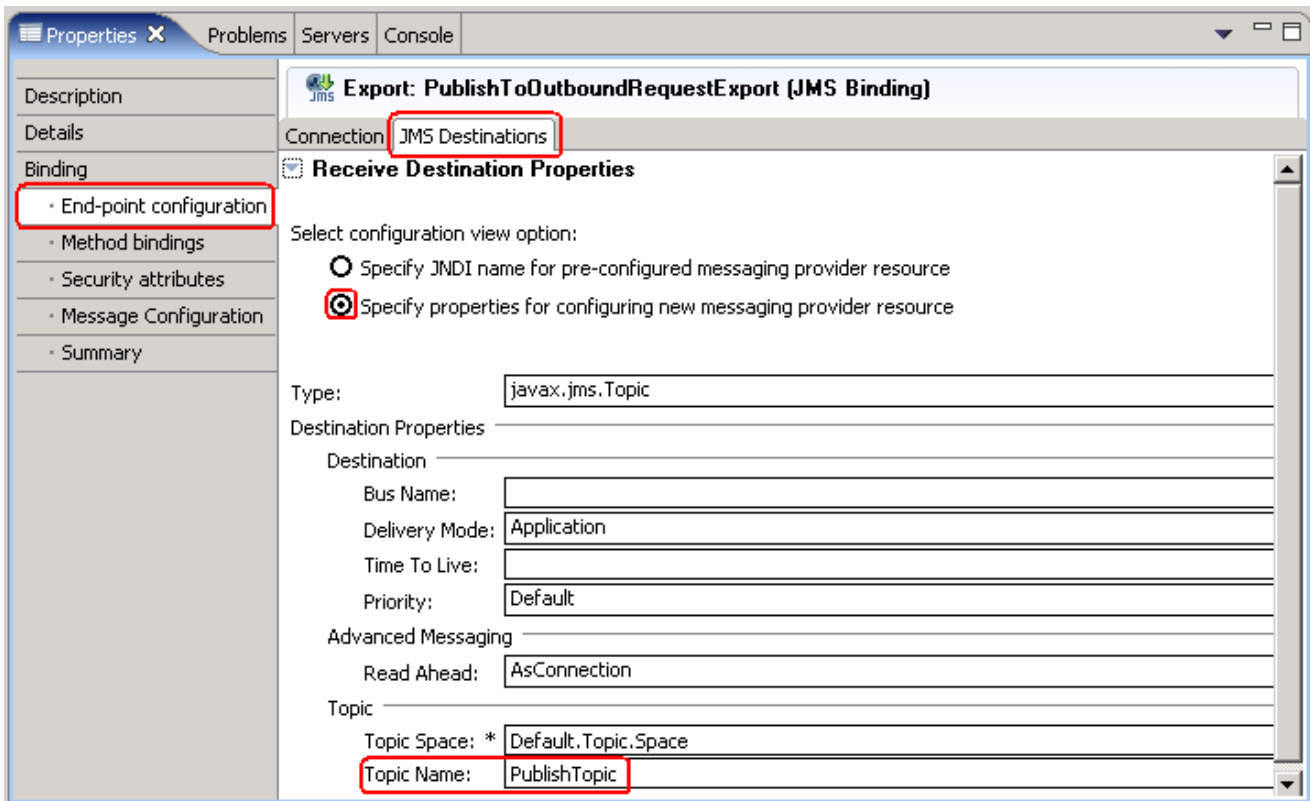


\_\_ e. Select the PublishToOutboundRequestExport you just created

\_\_ f. Select the **Bindings** → **End-point configuration** tab and then select the **JMS Destinations** tab in the **Properties** view. You must set the Export to use the specific topic where the data (ClipBG) will be published

\_\_ g. Expand **Receive Destinations Properties**

\_\_ h. Enter **PublishTopic** for the Topic Name at the bottom. This is the same name used by the Import Component of the CleansePublishProcess to publish the data

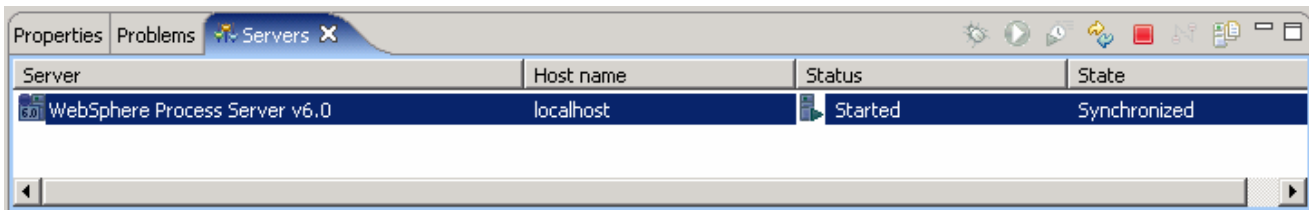


\_\_\_ i. Save and close the assembly editor. The SiebelAdapter assembly is complete.

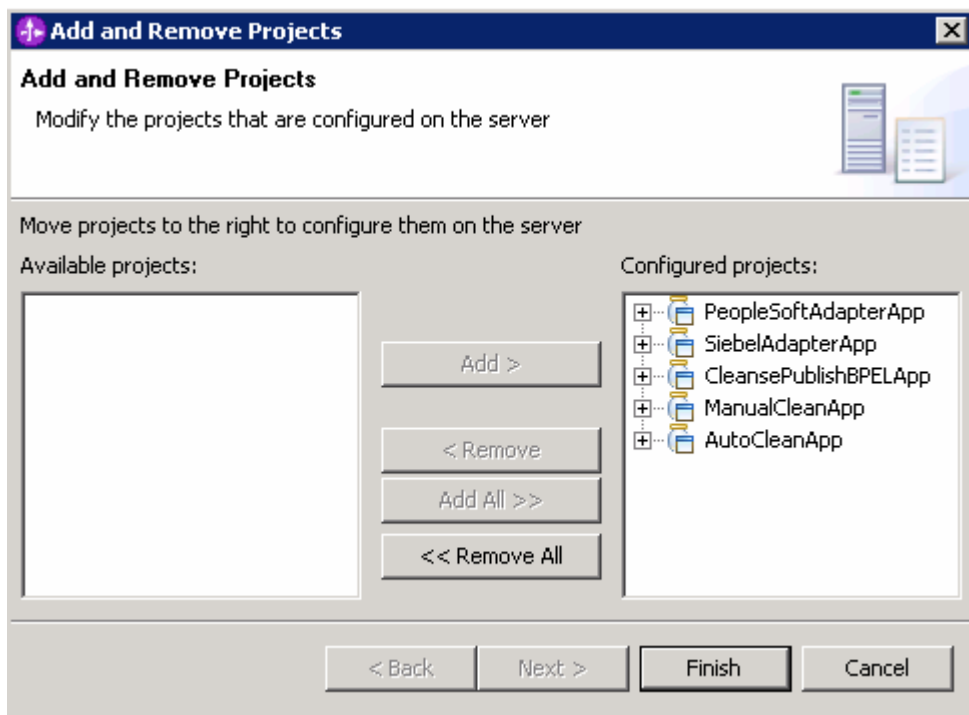
## Part 4: Testing eXchange

In this part you will test the end to end solution. This will be done by using two Visual Test Connectors, one to simulate the PeopleSoft application, which will drive the input and the other to simulate the Siebel application, which will receive the output. Once the flow is initiated from the Visual Test Connector for PeopleSoft, you will need to interact with the Human Task Manager. Once that is done, you should see the output in the Visual Test Connector for Siebel.

- \_\_\_ 1. If using a remote testing environment, follow directions provided in [Task: Adding remote server to WebSphere Integration Developer test environment](#) at the end of this document to add a server to the WebSphere Integration Developer test environment and start it. This is especially true for z/OS, AIX, Solaris remote test environment, where the WebSphere Integration Developer will be remote to the test environment
  - \_\_\_ a. Click on the **Servers** tab and verify that the server status shows as **Started**



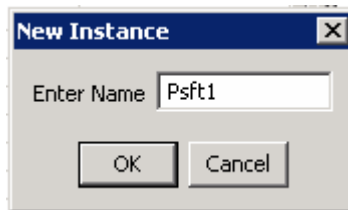
- \_\_\_ 2. Publish all of the projects to the se
  - \_\_\_ a. Right click on WebSphere Process Server V6 in the Servers view and select **Add and remove projects...** from the context menu
  - \_\_\_ b. Click the **Add All>>** button



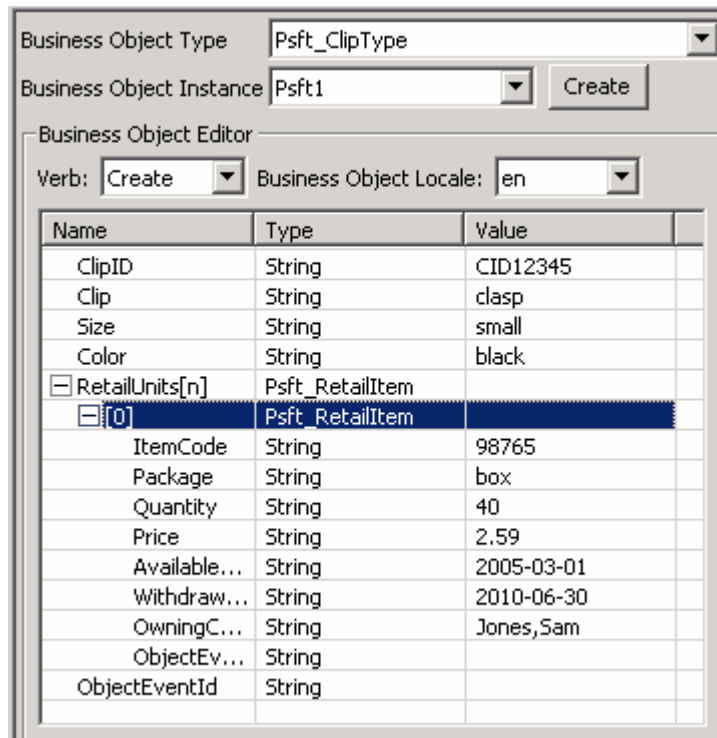
- \_\_\_ c. Click **Finish**



- \_\_\_ d. Wait for publishing of the projects to complete (no more messages in the lower right corner). This can take several minutes to complete
- \_\_\_ 3. Once the server is running with the entire solution deployed, you are ready to test it. To do that, start the Visual Test Connectors that will be used to simulate the PeopleSoft and Siebel applications
  - \_\_\_ a. Follow the instructions found in the [Task: Connecting with the Visual Test Connector](#), using the configuration data found at **<LAB\_FILES>\eXchange\PeopleSoft\ASBOs\PsfConnectorWPS.cfg**
  - \_\_\_ b. Open a second visual test connector by following the instructions found in the [Task: Connecting with the Visual Test Connector](#) at the end of this document, using the configuration data found at **<LAB\_FILES>\eXchange\Siebel\ASBOs\SiebelConnectorWPS.cfg**
- \_\_\_ 4. Load a predefined business object into the Visual Test Connector for the PeopleSoft and examine its values
  - \_\_\_ a. In the VTC window for PeopleSoft, select **Edit → Load Business Object**
  - \_\_\_ b. Navigate to **<LAB\_FILES>\eXchange\PeopleSoft\ASBOInstances** in the Open dialog, and select **PsfASBO1.bo**
  - \_\_\_ c. Click **Open**
  - \_\_\_ d. Enter a name to be assigned to this instance, such as **Psft1**



- \_\_\_ e. Click **OK**
- \_\_\_ f. Look at the values assigned to the Psft\_ClipType fields



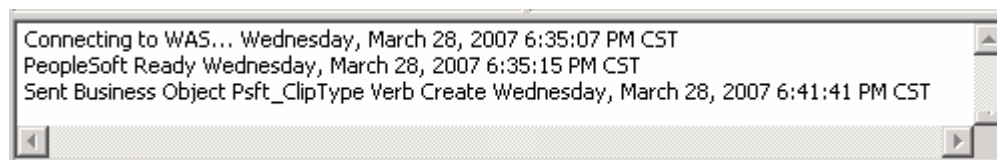
\_\_\_ g. Click the + in front of **RetailUnits[n]** to expand the Psft\_RetailItem array

\_\_\_ h. Click the + in front of **[0]** to see an instance of Psft\_RetailItem

\_\_\_ 5. Initiate the test by sending the business object from the PeopleSoft VTC to the WebSphere Process Server

\_\_\_ a. From the main menu, select **Request → Send**

\_\_\_ b. You should see a message in the VTC indicating that the message was sent



\_\_\_ 6. Part of the application scenario includes a Human Task. This is the opportunity for a human to review the contents of the business object and make any necessary changes to it. Start by opening the BPC Explorer to handle the Human Task activity

\_\_\_ a. Right-click on WebSphere Process Server V6.0 in the Servers view and select **Launch → Business Process Choreographer Explorer** from the context menu. If you are using a remote server, you can instead open a Web browser and navigate to <http://<HOSTNAME>:9080/bpc>. The My Tasks view will be shown with the task ready to be completed

The screenshot shows the Business Process Choreographer Explorer interface. The left sidebar contains three main sections: 'Process Templates' with a sub-link 'My Process Templates'; 'Process Instances' with sub-links 'Started By Me', 'Administered By Me', 'Critical Processes', 'Terminated Processes', and 'Failed Compensations'; and 'Task Templates' with a sub-link 'My Task Templates'. The main content area is titled 'My Tasks' and includes a welcome message, a list of 'Available Actions' (Start, Work on, Release, Transfer, Refresh), and a table of tasks. The 'Work on' button and the 'ManualClean' task name are highlighted with red boxes. The table shows one task: 'ManualClean' with state 'Ready' and kind 'Participating'. The footer of the table indicates 'Items found: 1' and 'Items selected: 1'.

<input type="checkbox"/>	Task Name	State	Kind	Owner	Originator
<input checked="" type="checkbox"/>	ManualClean	Ready	Participating		UNAUTHENTICATED

- \_\_\_ 7. Complete the task so that the flow of the scenario can continue
  - \_\_\_ a. Select the check box next to **ManualClean** and then click the “**Work on**” button. This will claim the task on behalf of the current user

**Task Message**

Use this page to provide the data required to complete the task. [i](#)

**Available Actions**

Task Instance Name: ManualClean

Task Input Message: inClipBG

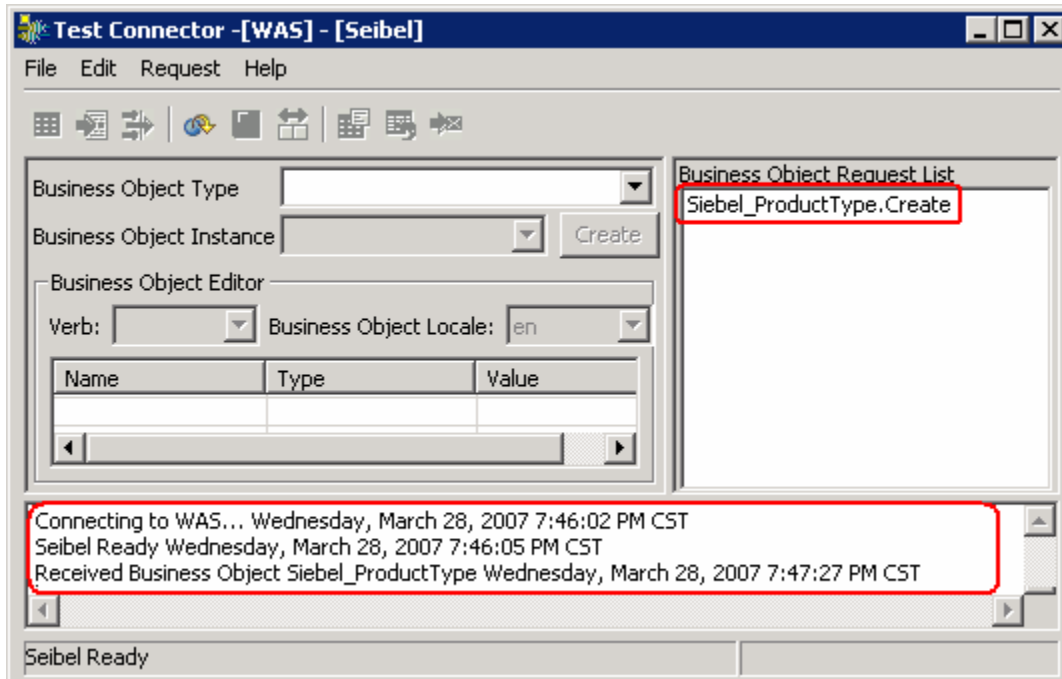
Clip	clipID	CID12345							
	GLN	1234567890123							
	clip	clasp							
	size	small							
	color	black							
	brand	Clips and Tacks							
retailItems	itemID	GTIN	package	quantity	fullDescription	price	startDate	endDate	contactFirstName
	98765		box	40	box 40	2.59	2005-03-01	2010-06-30	Sam

- \_\_\_ b. Fill in the values for fields in the task output message based on the values in the input message. Since this step in the business process allows a human to review and make updates to the business object, you will make a slight modification. Fill in all the output message fields with the same values from the input message, except change the color from black to charcoal as shown in the image. You will need to click the **+** in **retailItems** to fill in the data values for an instance

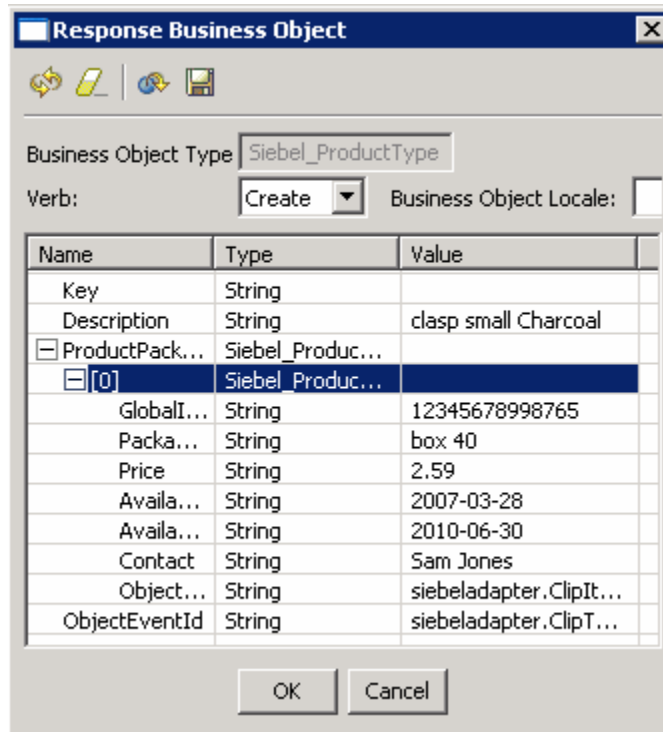
Task Output Message: outClipBG

Clip	clipID	<input type="text" value="CID12345"/>				
	GLN	<input type="text" value="1234567890123"/>				
	clip	<input type="text" value="clasp"/>				
	size	<input type="text" value="small"/>				
	color	<input type="text" value="charcol"/>				
	brand	<input type="text" value="Clips and Tacks"/>				
retailItems	itemID	GTIN	package	quantity	fullDescription	price
	<input type="text" value="98765"/>	<input type="text"/>	<input type="text" value="box"/>	<input type="text" value="40"/>	<input type="text" value="box 40"/>	<input type="text" value="2.59"/>
<input type="button" value="+"/>						

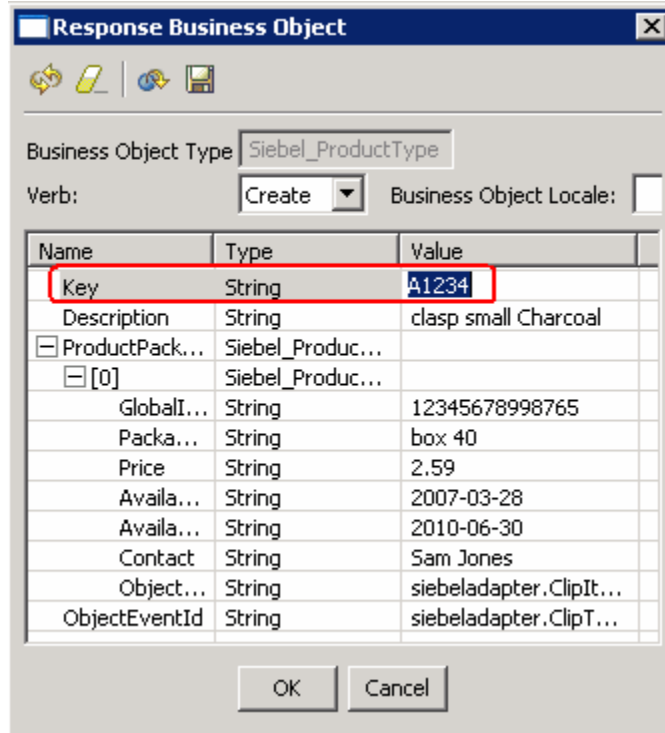
- \_\_\_ c. Click the **Complete** button
- \_\_\_ 8. Verify the data that has been sent to Siebel
  - \_\_\_ a. Look at the Visual Test Connector window for Siebel and verify that it has received a business object



- \_\_ b. Double click on **Siebel\_ProductType.create** to open a window with the received BO
- \_\_ c. Examine the data in the Response Business Object window as shown below:



- \_\_ d. Enter a value for the Key field, such as **A1234**, as shown below:



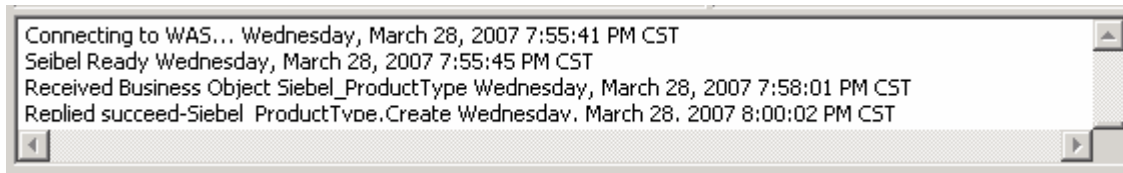
\_\_\_ e. Click **OK**

\_\_\_ 9. Send a response back to the server indicating the request was received and providing the Key value

\_\_\_ a. Ensure that **Siebel\_ProductType.create** is selected

\_\_\_ b. Select **Request → Reply → Success** from the menu

\_\_\_ c. Review the message window for a successful reply as shown below:



\_\_\_ 10. Now that testing is complete, clean the environment

\_\_\_ a. Right click on the server in the Servers view and select **Add and remove projects...** from the context menu

\_\_\_ b. Use the **<<Remove All** button to remove the projects from the server

\_\_\_ c. Click **Finish**

\_\_\_ d. Stop the Server

\_\_\_ 11. Close both Visual Test Connector windows

## What you did in this exercise

In this chapter you examined existing components that can be used in the eXchange scenario. You then defined the Imports, Exports and bindings needed to create an end to end solution. Finally, you tested that solution by driving input from a Visual Test Connector and examining the resulting output.

## Solution instructions

- \_\_\_ 1. Follow the directions in the task [Initialize the Workspace for a Lab Exercise](#), using the following values:

**<WORKSPACE>**

C:\Labfiles602\exchange\EndToEndNoFFA\workspace

**<PROJECT\_INTERCHANGE>**

C:\Labfiles602\exchange\EndToEndNoFFA\Solution\EndToEnd\_Pi.zip

**<MODULE>**

n/a

**<DEPENDENT\_LIBRARIES>**

n/a

- \_\_\_ 2. Continue with **Part 4: Testing eXchange**.

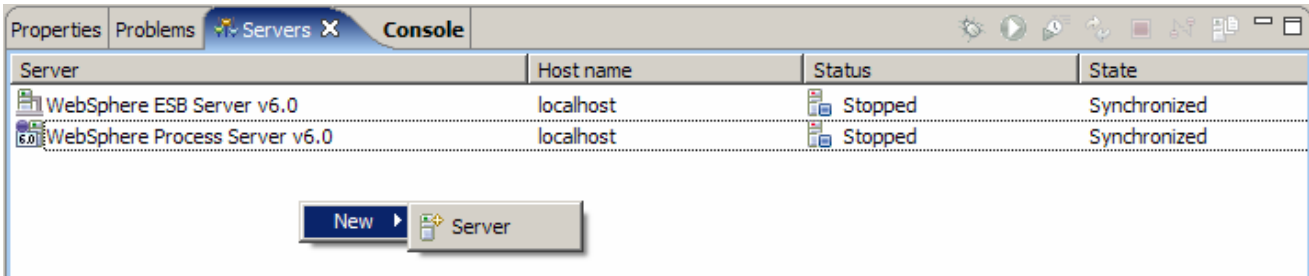


# Task: Adding remote server to WebSphere Integration Developer test environment

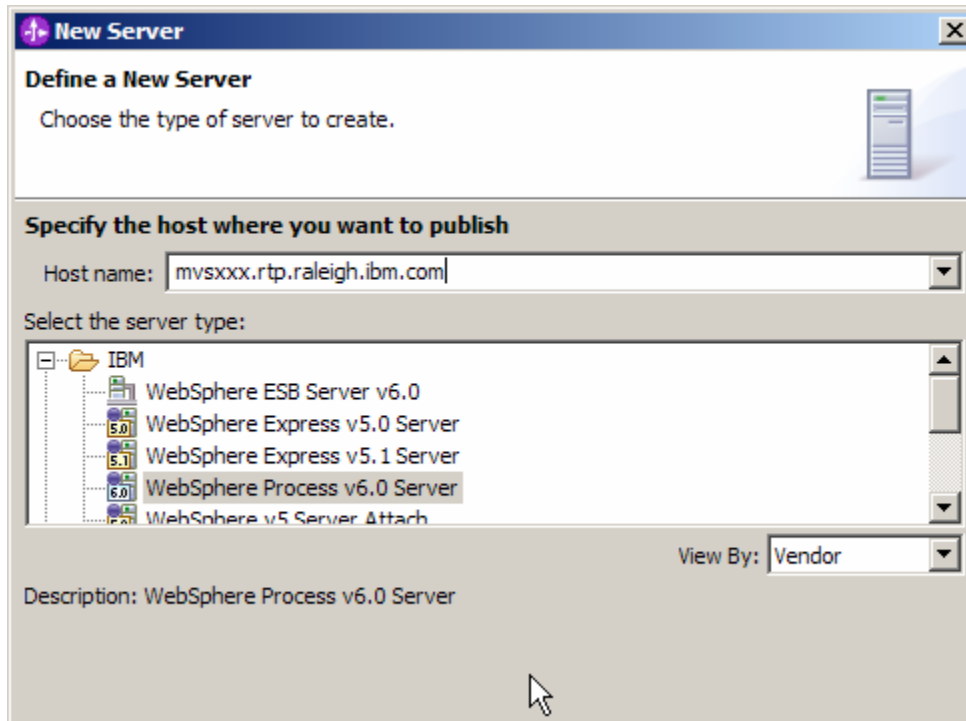
This task describes how to add a remote server to the WebSphere Integration Developer test environment. The sample will use a z/OS machine.

Create a new remote server.

- \_\_\_ 1. Right click on the background of the Servers view to access the pop-up menu.
- \_\_\_ 2. Select New → Server.

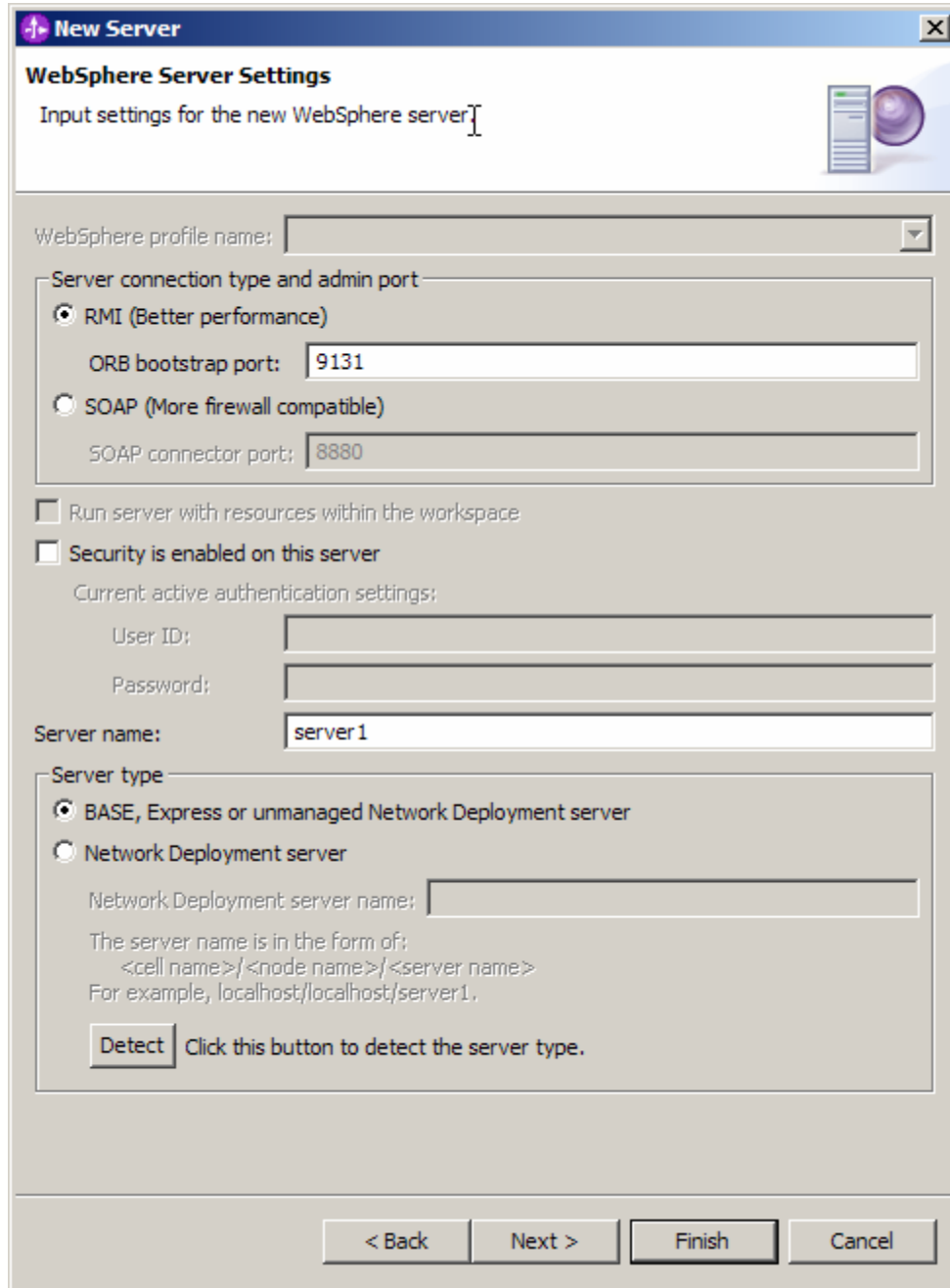


- \_\_\_ 3. Specify host name to the remote server, <HOSTNAME>.
- \_\_\_ 4. Ensure that 'WebSphere Process V6.0 Server' is highlighted in the server type list.

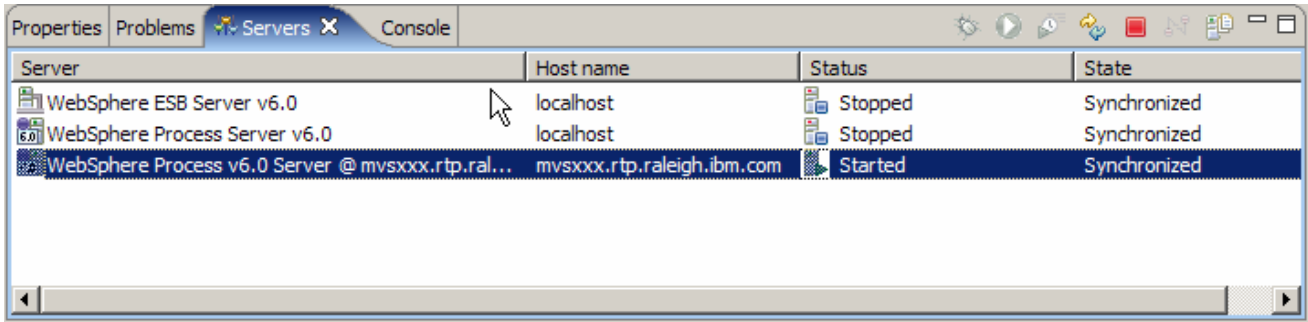


- \_\_\_ 5. Click **Next**.

- 6. On the WebSphere Server Settings page, select the radio button for **RMI** and change the ORB bootstrap port to the correct setting (<BOOTSTRAP\_PORT>).



- 7. Click **Finish**.
- 8. The new server should be seen in the Server view.



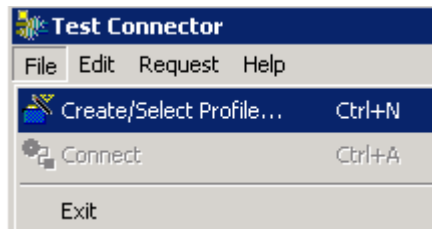
- \_\_\_ 9. Start the remote server if it is not already started. WebSphere Integration Developer does not support starting remote servers from the Server View.
- \_\_\_ 10. From a command prompt, telnet to the remote system if needed:
- ```
'telnet <HOSTNAME> <TELNET_PORT>'
      userid : <USERID>
      pw : <PASSWORD>
```
- \_\_\_ 11. Navigate to the bin directory for the profile being used:
- ```
cd <WAS_HOME>/profiles/<PROFILE_NAME>/bin
```
- \_\_\_ 12. Run the command file to start the server: `./startServer.sh <SERVER_NAME>`
- \_\_\_ 13. Wait for status message indicating server has started:

```
ADMU3200I: Server launched. Waiting for initialization status
ADMU3000I: Server cl1sr01 open for e-business; process id is 0000012000000002
```

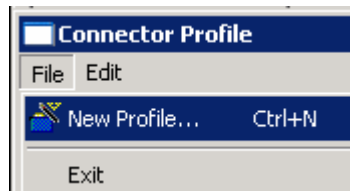
## Task: Connecting with the Visual Test Connector

This task describes how to create a Profile and start a connection with the Visual Test Connector (VTC).

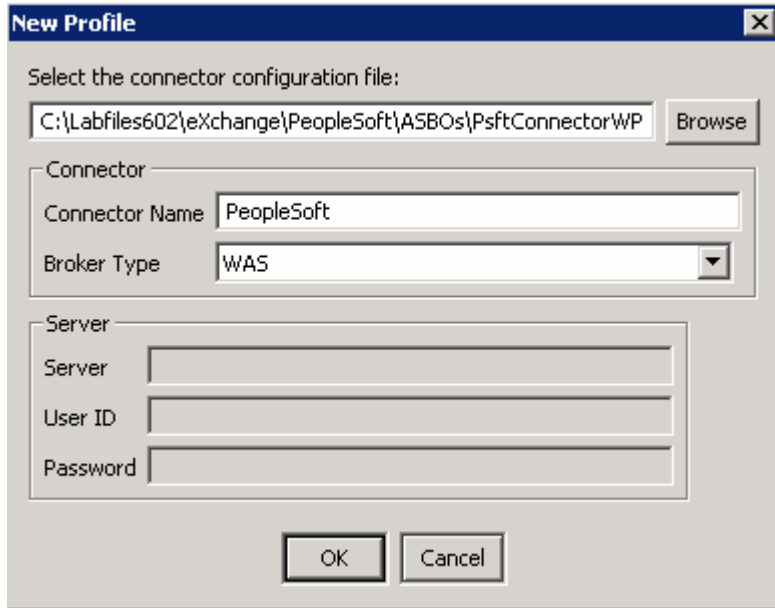
- \_\_\_ 1. Ensure the WebSphere Process Server is active (the server is the JMS Broker). The VTC requires an active JMS Broker for a successful connection
- \_\_\_ 2. Start the Visual Test Connector
  - \_\_\_ a. From the **start** menu, select **Programs > IBM WebSphere Business Integration Adapters > IBM WebSphere Business Integration Toolset > Visual Test Connector (VTC)**. This opens a command window in the background where the process runs and the Visual Test Connector Interface in the foreground. Do not close the command window as this will cause the VTC to exit.
- \_\_\_ 3. Create/Select a Profile
  - \_\_\_ a. From the main menu of the VTC interface, select **Create/Select a Profile**. The Connector Profile panel opens



- \_\_\_ b. If a profile doesn't exist, create a new PeopleSoft profile by selecting **File → New Profile...** from the main menu of the Connector Profile panel.

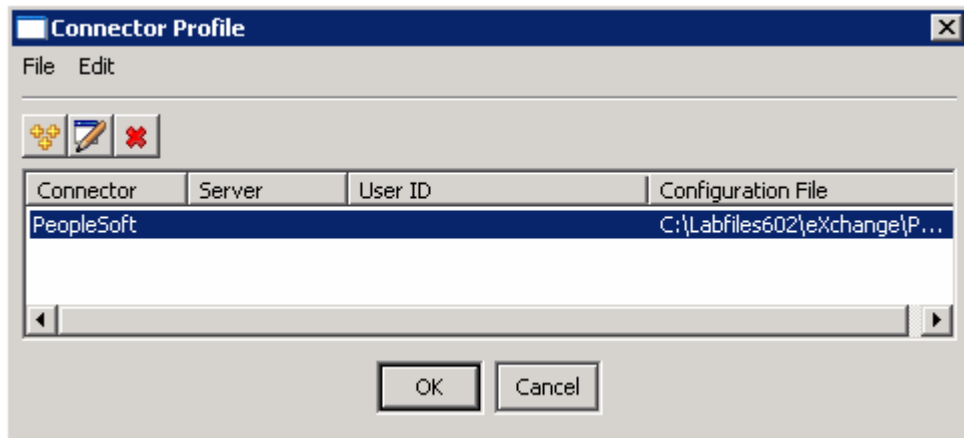


- \_\_\_ c. Enter the following information in the New Profile panel:
  - 1) Click the **Browse** button for "Select the connector configuration file:" filed and select **<LAB\_FILES>\eXchange\PeopleSoft\ASBOs\PsfConnectorWPS.cfg** file
  - 2) Connector Name : **PeopleSoft**
  - 3) Broker Type : **WAS**



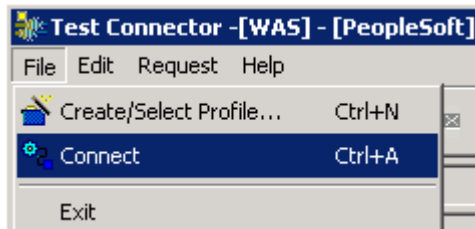
\_\_ d. Click **OK**

\_\_ e. Select the PeopleSoft profile you had created as shown below:

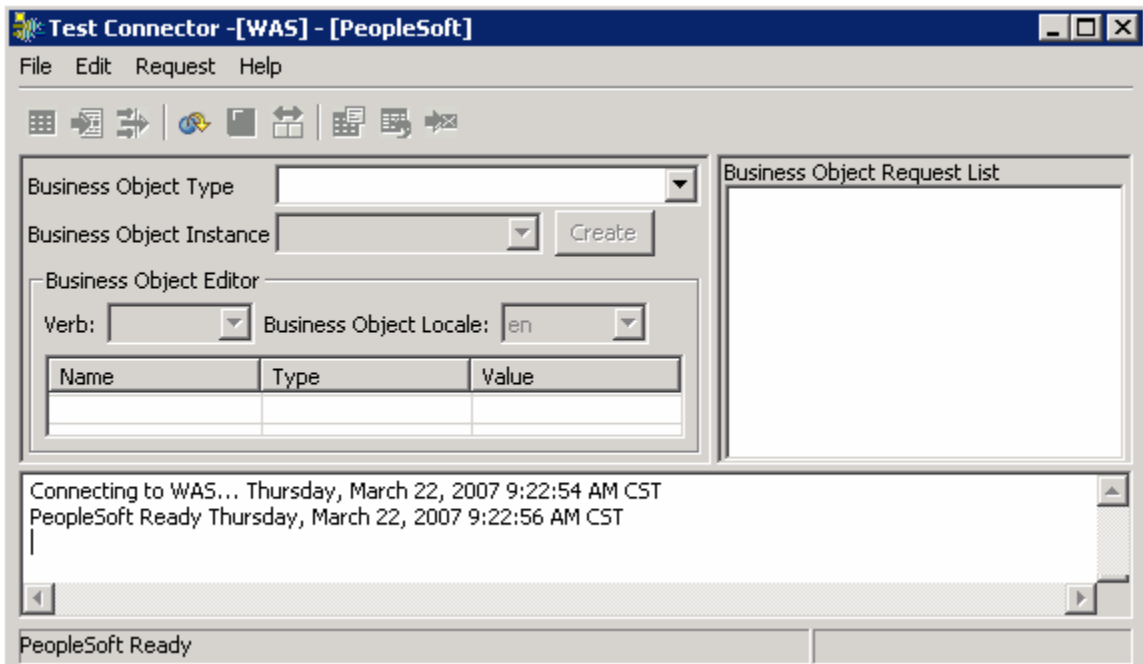


\_\_ f. Click **OK**

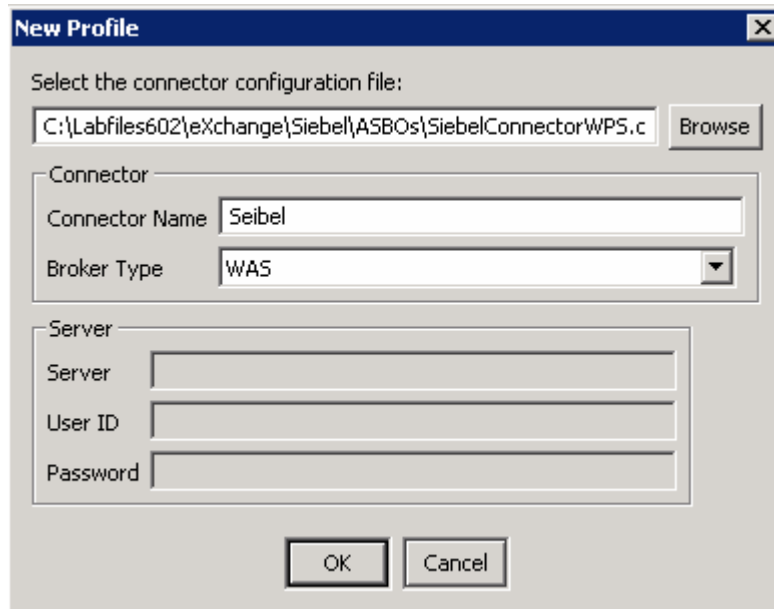
\_\_ g. Now in the Test Connector interface, select **File** → **Connect** from the main menu



\_\_ h. Ensure that a successful connection is made. You can confirm this by watching the trace messages displayed at the bottom frame of the Test Connector interface as shown below:

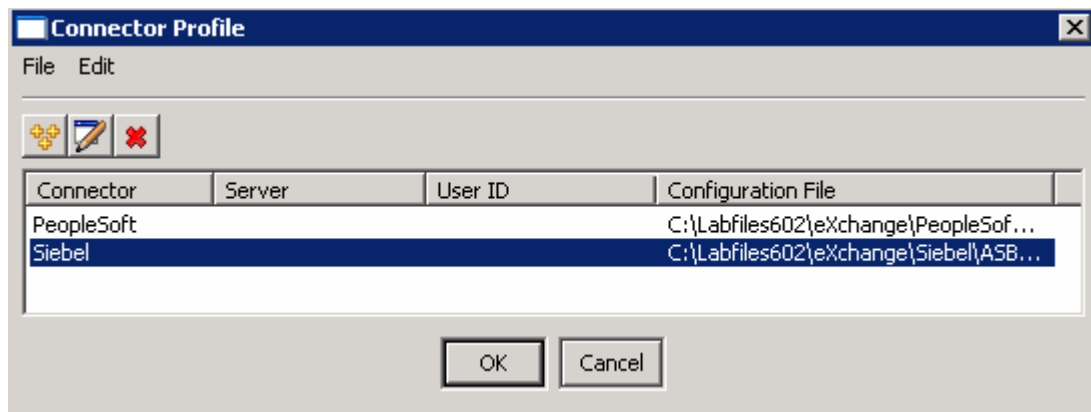


- \_\_\_ 4. Follow the above steps to create a new Siebel profile as follows:
- \_\_\_ a. From the main menu of the VTC interface, select **Create/Select a Profile**. The Connector Profile panel opens
  - \_\_\_ b. Create a new Seibel profile by selecting **File → New Profile...** from the main menu of the Connector Profile panel
  - \_\_\_ c. Enter the following information in the New Profile panel:
    - 1) Click the **Browse** button for "Select the connector configuration file:" filed and select **<LAB\_FILES>\eXchange\Siebel\ASBOs\SiebelConnectorWPS.cfg** file
    - 2) Connector Name : **Siebel**
    - 3) Broker Type : **WAS**



\_\_\_ d. Click **OK**

\_\_\_ e. Select the Siebel profile you had created as shown below:



\_\_\_ f. Click **OK**

\_\_\_ g. Now in the Test Connector interface, select **File** → **Connect** from the main menu

\_\_\_ h. Ensure that a successful connection is made. You can confirm this by watching the trace messages displayed at the bottom frame of the Test Connector interface as shown below:

