IBM WebSphere® Process Integration 6.0 – Lab Exercise

End to End Scenario

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 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 installed
- WebSphere Process Server V6 test environment installed
- Sample code in the directory C:\Labfiles60 (Windows®) or /tmp/LabFiles60 (Linux®)
- WebSphere Business Integration Toolset installed

What you should be able to do

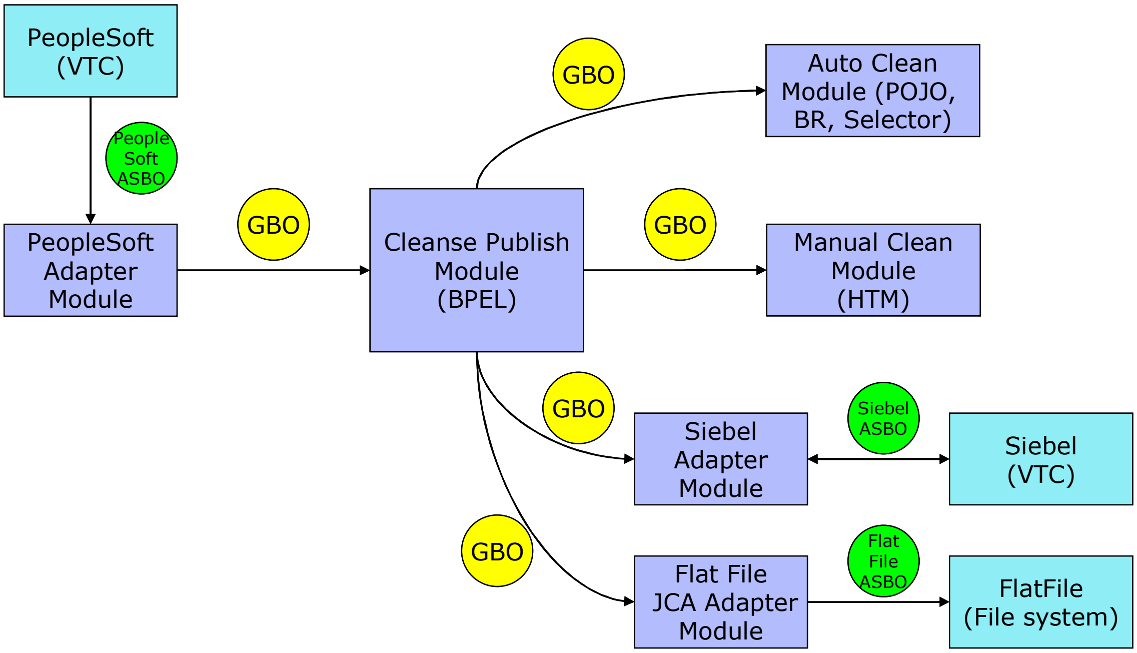
At the end of this lab you should be able to:

- 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.


PeopleSoft (VTC)
People Soft ASBO
PeopleSoft Adapter Module
GBO
Cleanse Publish Module (BPEL)
GBO
Auto Clean Module (POJO, BR, Selector)
GBO
Manual Clean Module (HTM)
GBO
Siebel Adapter Module
Siebel ASBO
Siebel (VTC)
GBO
Flat File JCA Adapter Module
Flat File ASBO
FlatFile (File system)

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:\Labfiles60	/tmp/Labfiles60
<WORKSPACE>	C:\Labfiles60\eXchange\EndToEndNoFFA\workspace	/tmp/Labfiles60/eXchange/EndToEndNoFFA/workspace
<TEMP>	C:\temp	/tmp
<SOLUTION>	C:\Labfiles60\eXchange\EndToEndNoFFA\Solution	/tmp/Labfiles60/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:\LabFiles60\ would be replaced by C:/LabFiles60/.

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 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	cl1sr01	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\AppServer	/etc/cl1cell/AppServerNode1	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<BOOTSTRAP_PORT>	2809	2809	
<TELNET_PORT>	N/A	1023	
<PROFILE_NAME>	AppSrv01	default	
<USERID>	N/A	cl1admin	
<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 WebSphere Integration Developer Test Environment**".

Part 1: Initialize the Workspace for this Lab Exercise

This task is a common description for how to initialize your workspace for doing this lab. Substitute the values listed below for the variables.

<WORKSPACE> - C:\Labfiles60\eXchange\EndToEndNoFFA\workspace

<PROJECT_INTERCHANGE> - C:\Labfiles60\eXchange\EndToEndNoFFA\import\EndToEndStart_PI.zip

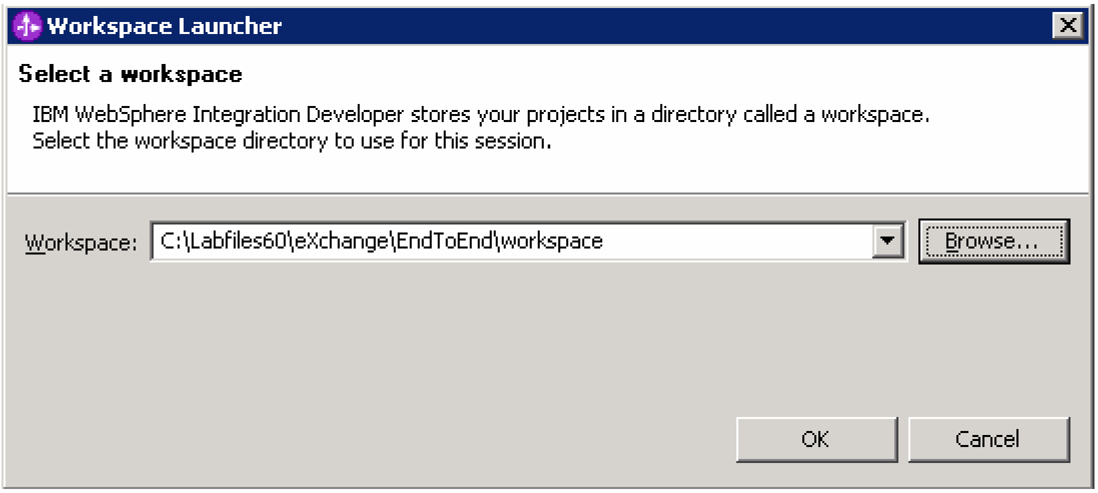
<MODULE> - n/a

<DEPENDENT_LIBRARIES> - n/a

____ 1. Start WebSphere Integration Developer V6 with a new workspace located at **<WORKSPACE>**.

__ a. From Windows Explorer, navigate to the **<WID_HOME>** directory and double click on wid.exe.

__ b. When prompted for workspace name, enter the value provided by the **<WORKSPACE>** variable for this lab and click **OK.**



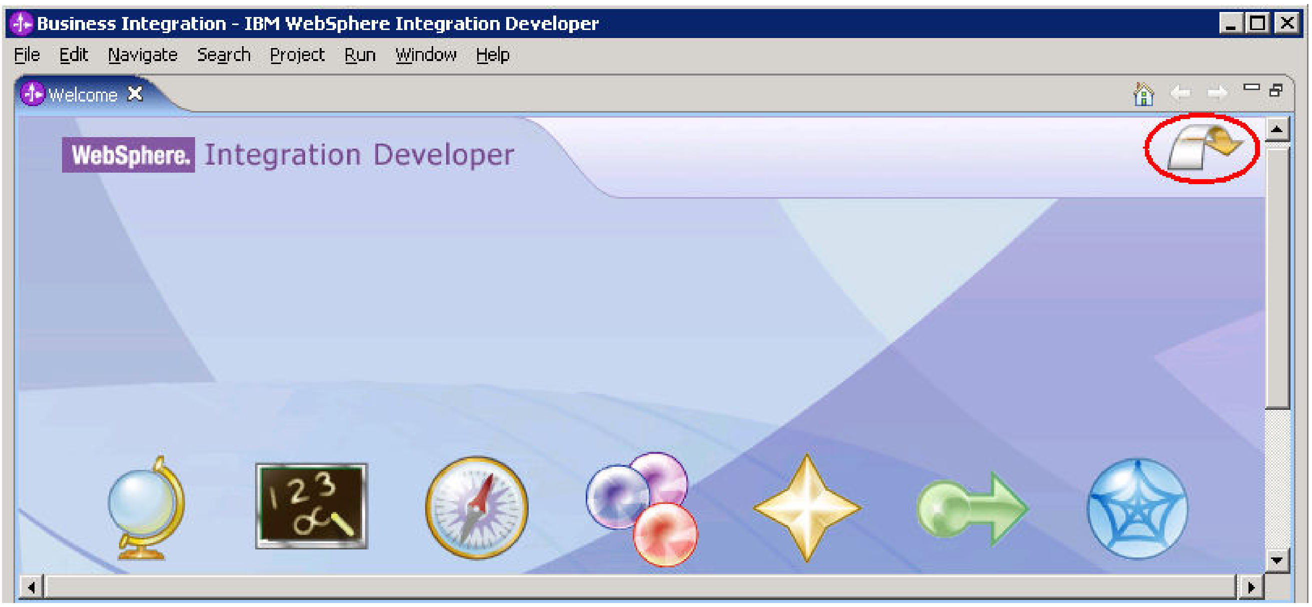
Workspace Launcher

Select a workspace

IBM WebSphere Integration Developer stores your projects in a directory called a workspace.
Select the workspace directory to use for this session.

Workspace: C:\Labfiles60\eXchange\EndToEnd\workspace

__ c. When WebSphere Integration Developer V6.0 opens, close the **Welcome page** by clicking on the Go to the workbench icon (bent over arrow at top-right).

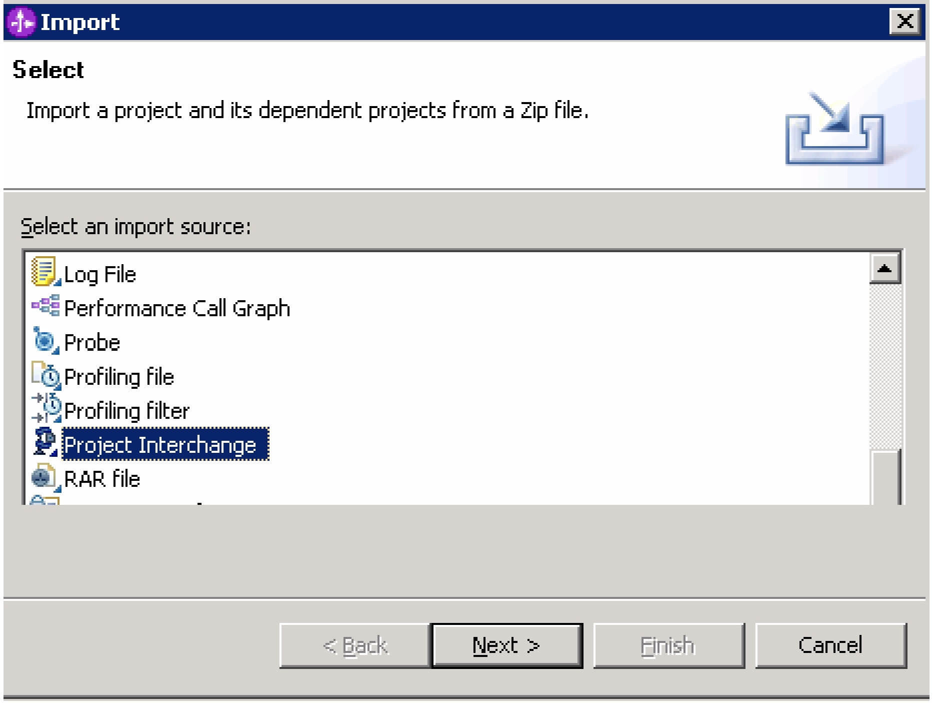

Business Integration - IBM WebSphere Integration Developer
File Edit Navigate Search Project Run Window Help
Welcome
WebSphere. Integration Developer

__ d. Ensure you are in the **Business Integration** perspective.

_____ 2. If this lab requires you to import a project interchange file, setup the required libraries and modules for this lab by importing the project interchange file **<PROJECT_INTERCHANGE>.**

__ a. From the menu bar, select **File -> Import…**

__ b. In the Import dialog, scroll down and select **Project Interchange.**


Import

Select

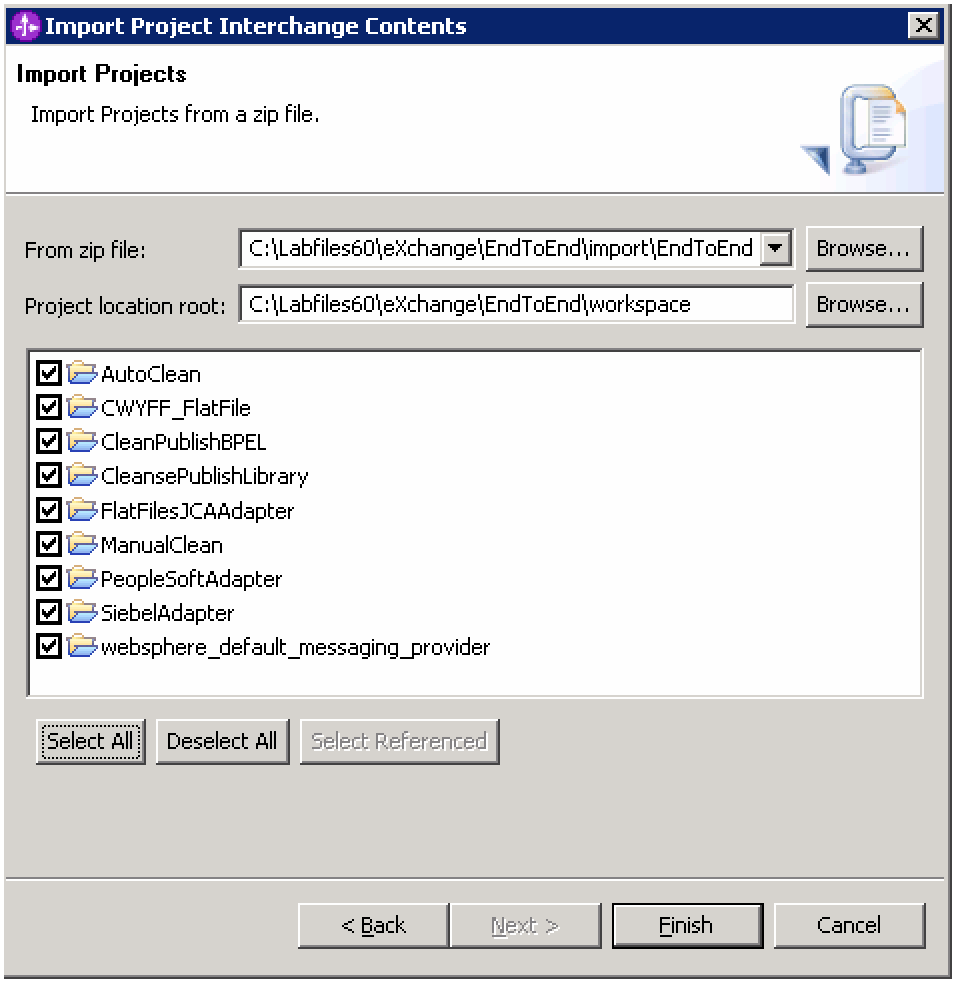
Import a project and its dependent projects from a Zip file.

Select an import source:

__ c. Click **Next.**

__ d. In the Import Projects dialog, initialize the From zip file: field to <**PROJECT_INTERCHANGE>.**

__ e. Click the **Select All** button.



From zip file: C:\Labfiles60\eXchange\EndToEnd\import\EndToEnd

Project location root: C:\Labfiles60\eXchange\EndToEnd\workspace

☑ AutoClean
☑ CWYFF_FlatFile
☑ CleanPublishBPEL
☑ CleansePublishLibrary
☑ FlatFilesJCAAdapter
☑ ManualClean
☑ PeopleSoftAdapter
☑ SiebelAdapter
☑ websphere_default_messaging_provider

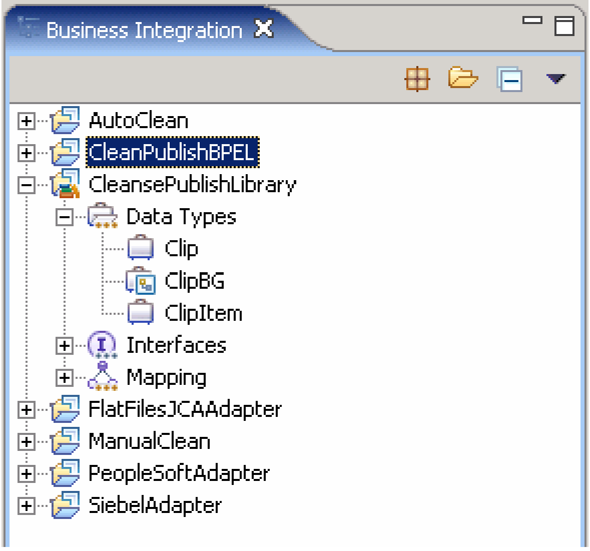
__ f. Click **Finish.**

Part 2: Understanding eXchange Components

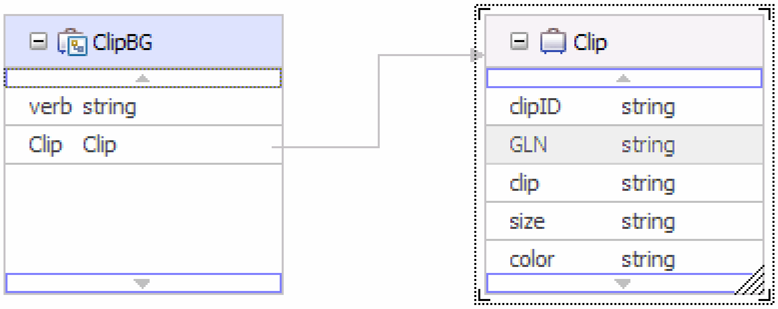
In this part you will examine the existing components that are part of the eXchange scenario. This will provide you with a basis to understand the solution you will be constructing in subsequent parts.

____ 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. Expand **CleansePublishLibrary** and expand **Data Types**.



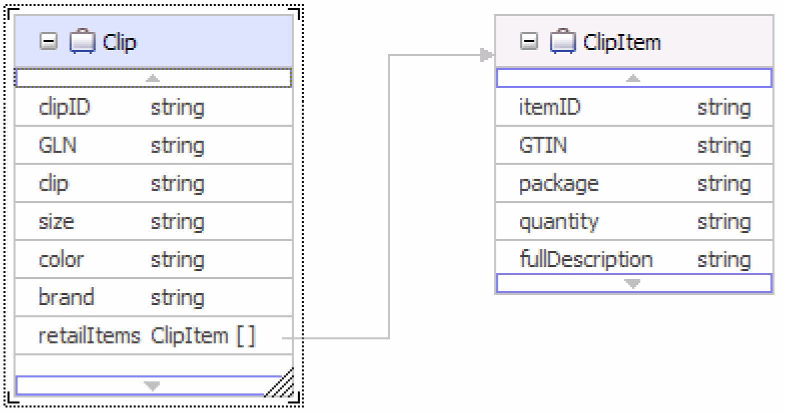
__ b. Open the ClipBG and Clip data types. Notice how ClipBG includes the Clip business object and Clip contains the ClipItem business object.


ClipBG

verb string
Clip Clip

Clip

clipID string
GLN string
clip string
size string
color string

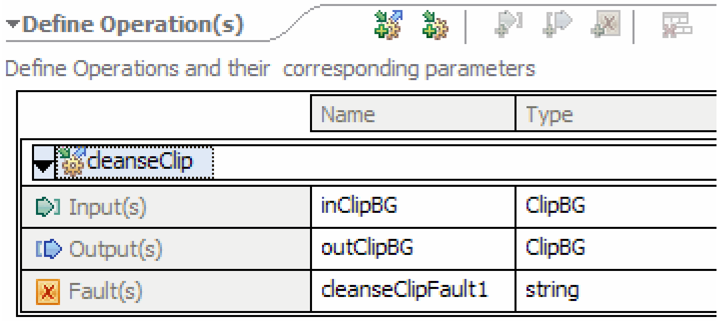


Clip	
clipID	string
GLN	string
clip	string
size	string
color	string
brand	string
retailItems	ClipItem [ ]

ClipItem	
itemID	string
GTIN	string
package	string
quantity	string
fullDescription	string

__ c. Expand **Interfaces** under CleansePublishLibrary.

__ d. Open the different interfaces and notice the operations and the messages. All messages are of the ClipBG type.

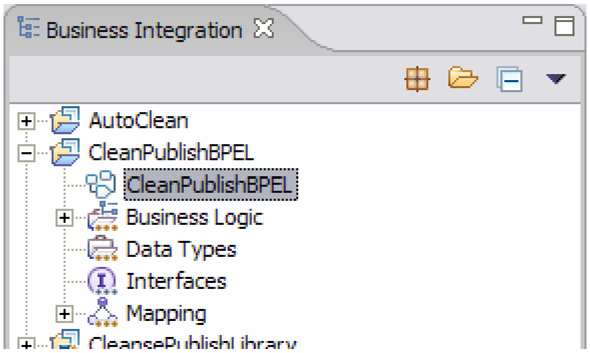


Define Operations and their corresponding parameters

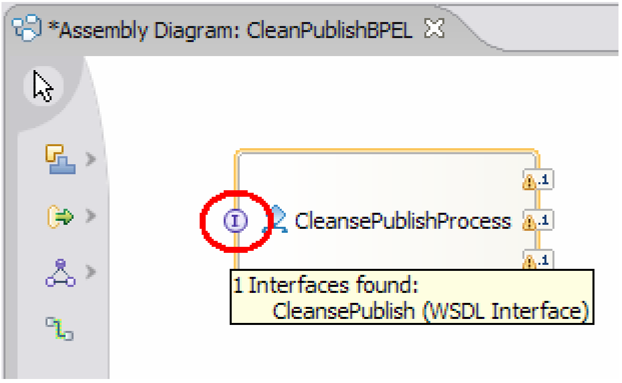
	Name	Type
cleanseClip		
Input(s)	inClipBG	ClipBG
Output(s)	outClipBG	ClipBG
Fault(s)	cleanseClipFault1	string

____ 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 **CleanPublishBPEL** and double-click on **CleanPublishBPEL** to open the assembly editor. The business process has already be created and defined as a component.

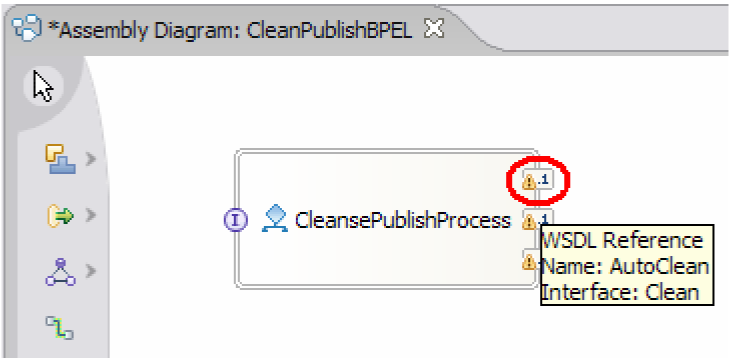


__ b. The business process has already been created and defined as a component within the module. It has an interface that can be called and 3 references to other service components. Hover the mouse over the interface (“**I**”) icon on the left to view the interface, which defines how the business process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



1 Interfaces found:
CleansePublish (WSDL Interface)

__ 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.

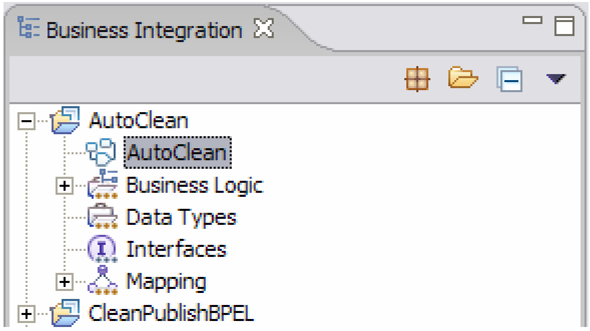


WSDL Reference
Name: AutoClean
Interface: Clean

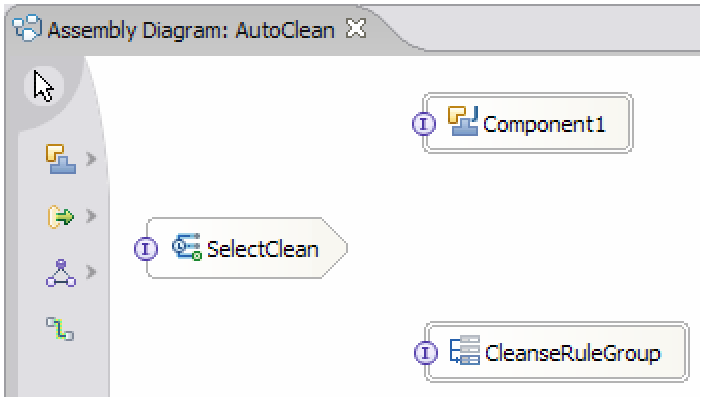
__ d. Close the assembly editor.

____ 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.

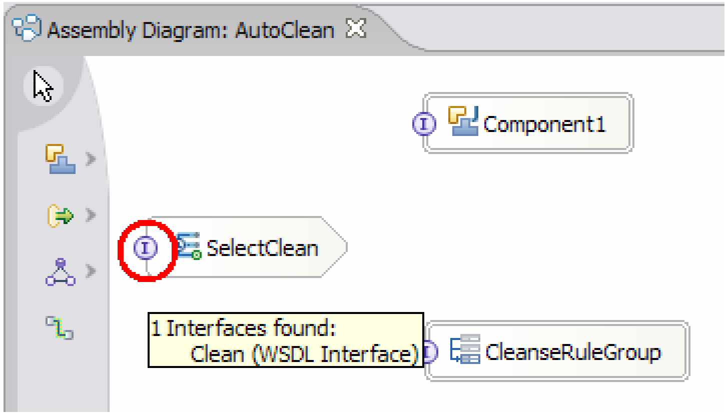
__ a. Expand **AutoClean** and double-click on **AutoClean** to open the assembly editor.



__ 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.

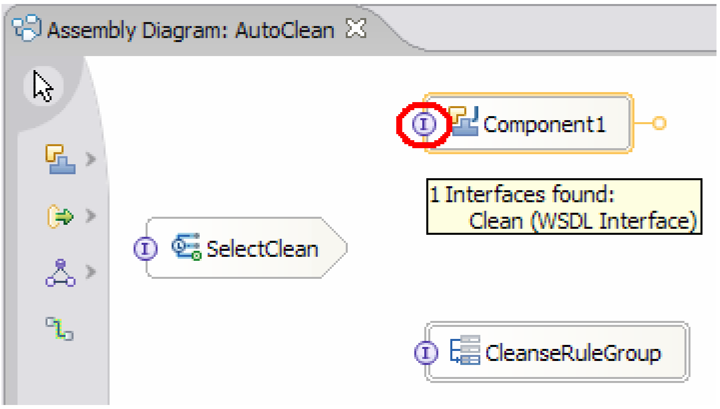


__ c. Hover the mouse over the interface icon for the Selector. Notice the Selector uses the **Clean** interface.



1 Interfaces found:
Clean (WSDL 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.

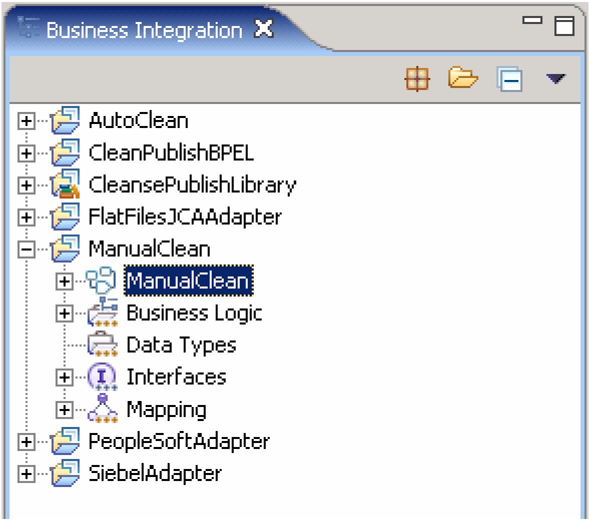


1 Interfaces found:
Clean (WSDL Interface)

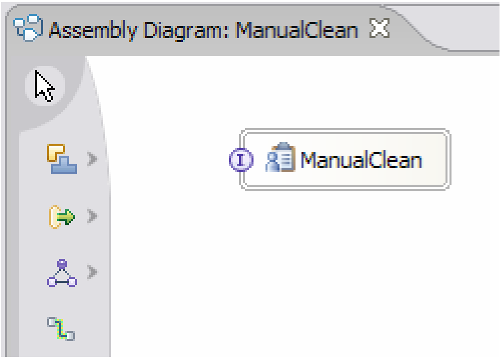
__ e. Close the assembly editor.

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

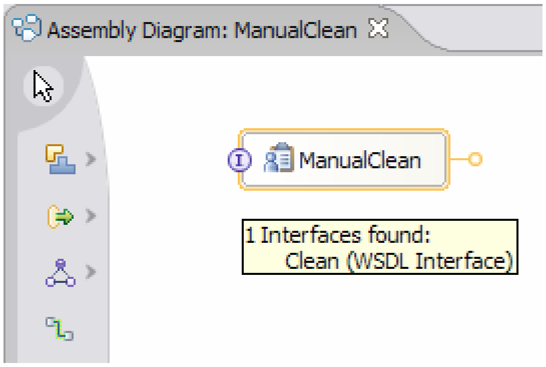
__ a. Expand **ManualClean** and double-click on **ManualClean** to open the assembly editor.



__ 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.

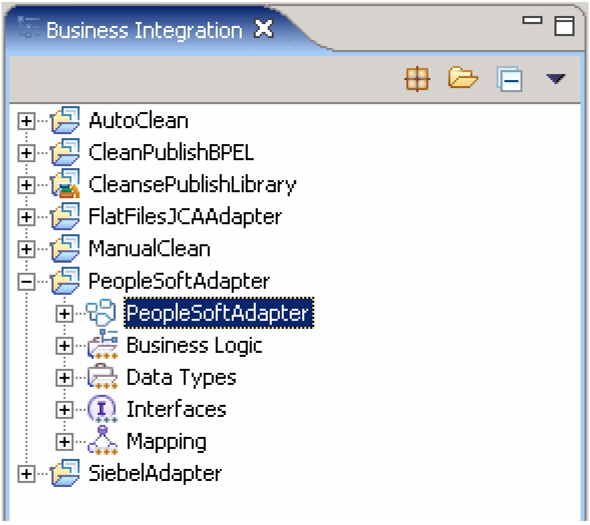


1 Interfaces found:
Clean (WSDL Interface)

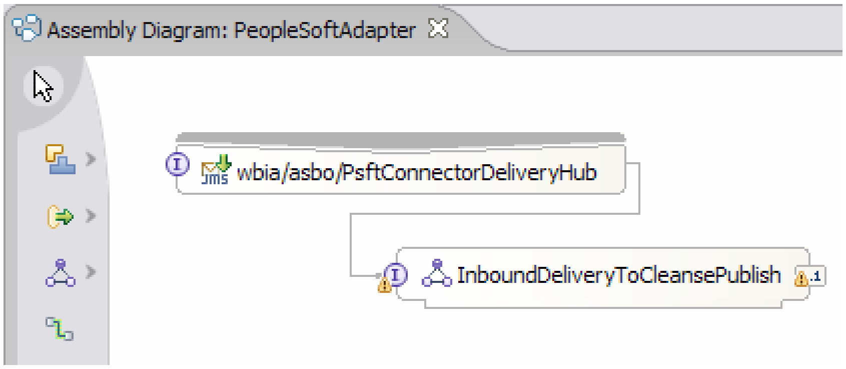
__ d. Close the assembly editor.

____ 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.

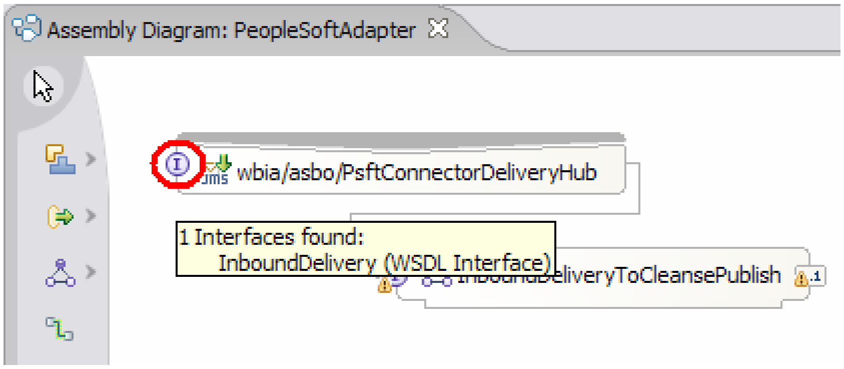
__ a. Expand **PeopleSoftAdapter** and double-click on **PeopleSoftAdapter** to open the assembly editor.



__ 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, **wbia/asbo/PsftConnectorDeliveryHub**, with a JMS binding is used on the maps. The connection between the map and Export is represented by the wire.

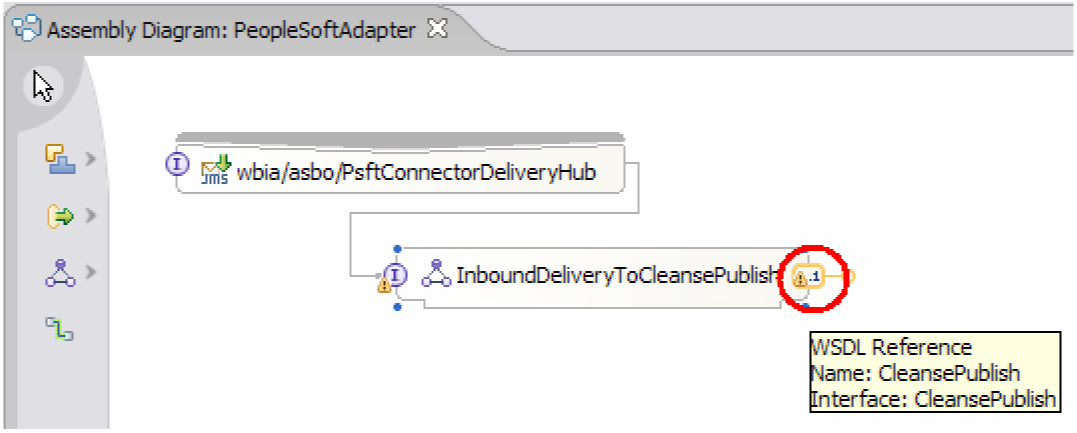

Assembly Diagram: PeopleSoftAdapter
wbia/asbo/PsftConnectorDeliveryHub
InboundDeliveryToCleansePublish

__ 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.


Assembly Diagram: PeopleSoftAdapter
wbia/asbo/PsftConnectorDeliveryHub
InboundDeliveryToCleansePublish

1 Interfaces found:
InboundDelivery (WSDL Interface)

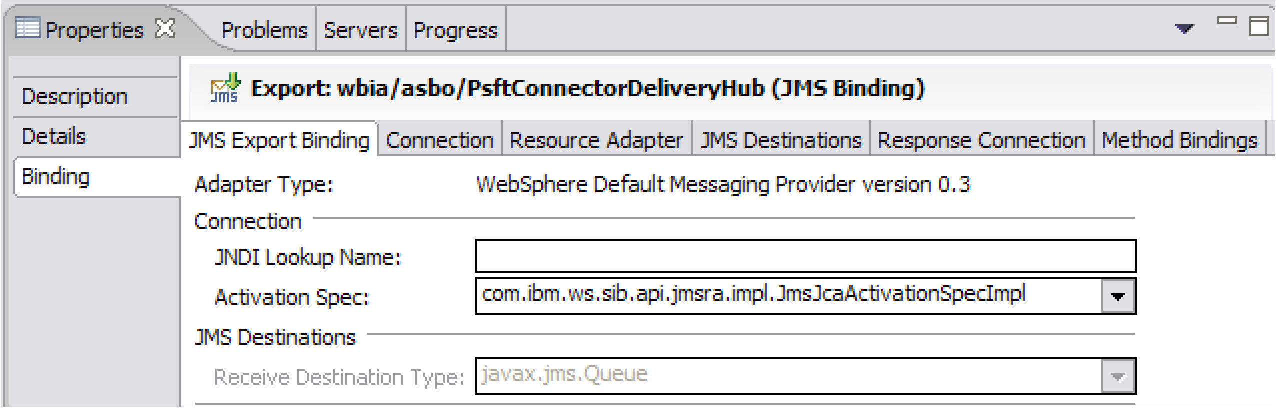
__ d. Hover the mouse over the reference on the map. Notice the reference is to the **CleansePublish** interface which is the same as used by the business process.


Assembly Diagram: PeopleSoftAdapter
wbia/asbo/PsftConnectorDeliveryHub
InboundDeliveryToCleansePublish

WSDL Reference
Name: CleansePublish
Interface: CleansePublish

__ e. Select the Export **wbia/asbo/PsftConnectorDeliveryHub** and select the Properties Tab.

__ f. Select the **Bindings** tab. Notice the JMS binding information specified for the module as it receives requests off a queue.



Export: wbia/asbo/PsftConnectorDeliveryHub (JMS Binding)

Adapter Type: WebSphere Default Messaging Provider version 0.3

Connection

JNDI Lookup Name:

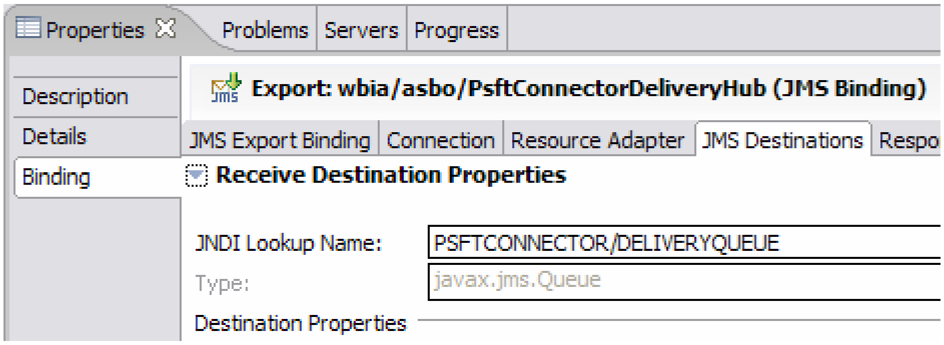
Activation Spec: com.ibm.ws.sib.api.jmsra.impl.JmsJcaActivationSpecImpl

JMS Destinations

Receive Destination Type: javax.jms.Queue

__ g. 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.



Export: wbia/asbo/PsftConnectorDeliveryHub (JMS Binding)

Receive Destination Properties

JNDI Lookup Name: PSFTCONNECTOR/DELIVERYQUEUE

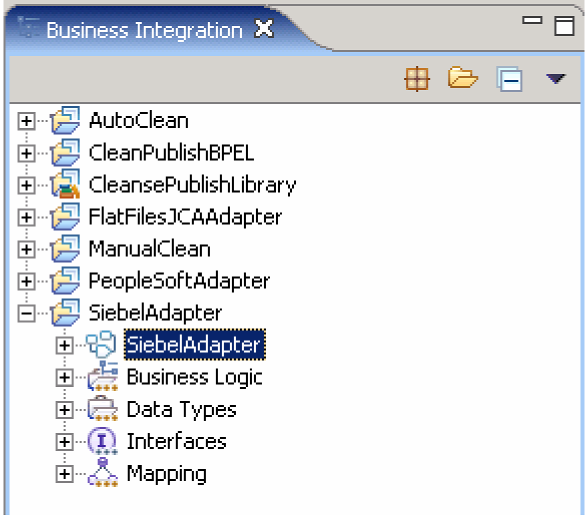
Type: javax.jms.Queue

Destination Properties

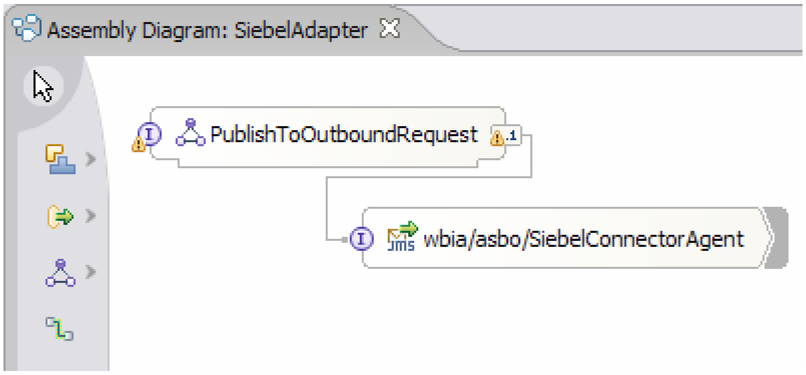
__ i. Close the assembly editor.

____ 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.

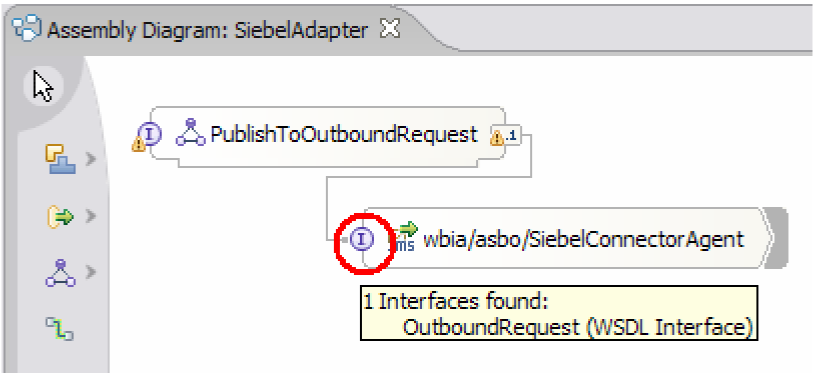
__ a. Expand **SiebelAdapter** and double-click on **SiebelAdapter** to open the assembly editor.



__ 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(**wbia/asbo/SiebelConnectorAgent)**. The connection between the map and Import is represented by the wire.

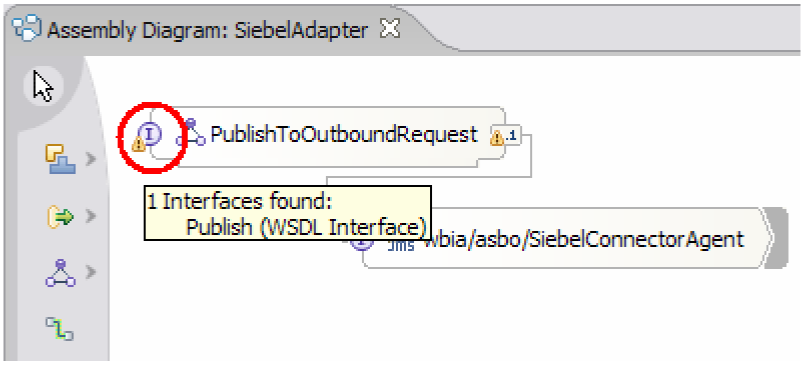

Assembly Diagram: SiebelAdapter
PublishToOutboundRequest
1
wbia/asbo/SiebelConnectorAgent

__ 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.


Assembly Diagram: SiebelAdapter
PublishToOutboundRequest
1
wbia/asbo/SiebelConnectorAgent

1 Interfaces found:
OutboundRequest (WSDL Interface)

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



Assembly Diagram: SiebelAdapter
PublishToOutboundRequest
wbia/asbo/SiebelConnectorAgent

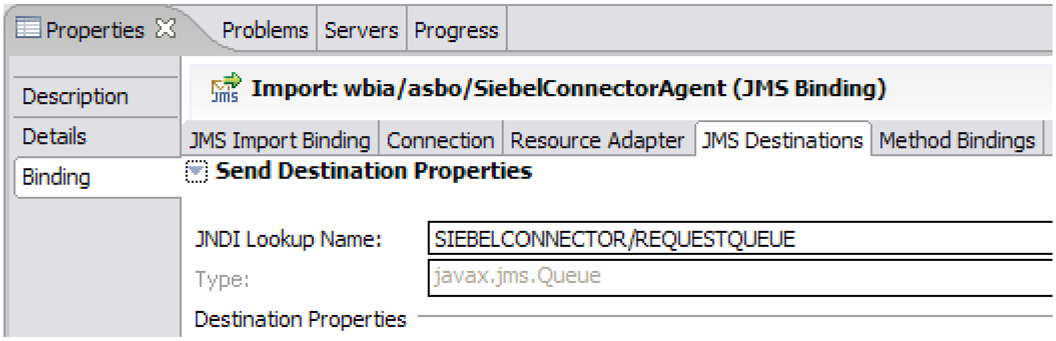
1 Interfaces found:
Publish (WSDL Interface)

__ e. Select the Import **wbia/asbo/SiebelConnectorAgent** and select the **Properties** Tab.

__ f. Select the **Bindings** 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.



Import: wbia/asbo/SiebelConnectorAgent (JMS Binding)

Send Destination Properties

JNDI Lookup Name: SIEBELCONNECTOR/REQUESTQUEUE

Type: javax.jms.Queue

Destination Properties

__ i. Close the assembly editor.

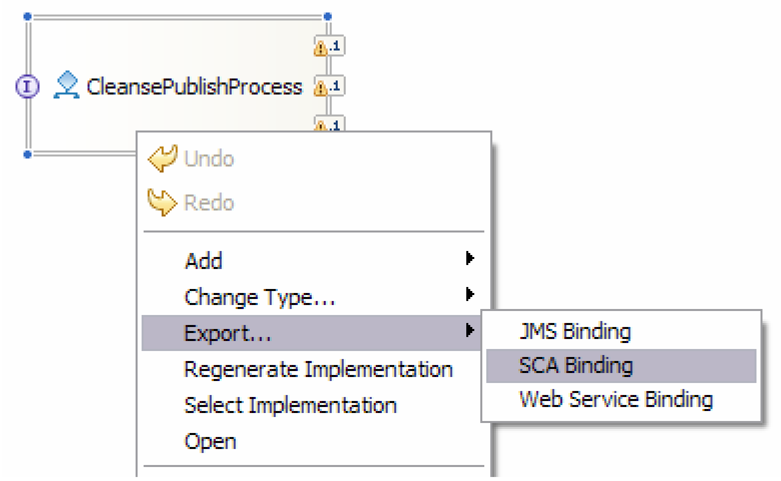
Part 3: Assemble eXchange

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.

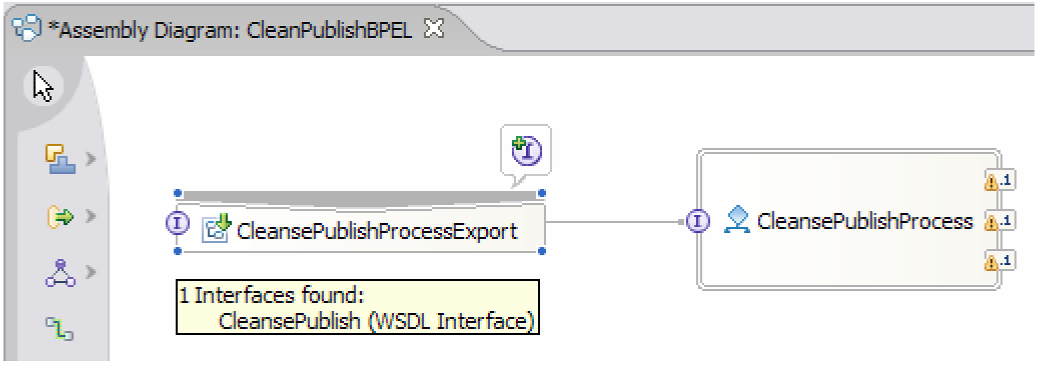
____ 1. The CleansePublishProcess has the most bindings and references that must be resolved. Start the overall assembly with defining the references and bindings for the CleanPublishBPEL module.

__ a. Open the **CleanPublishBPEL** assembly editor.

__ b. Right-click on the **CleanPublishBPEL** and select **Export…>SCA Binding**. 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.

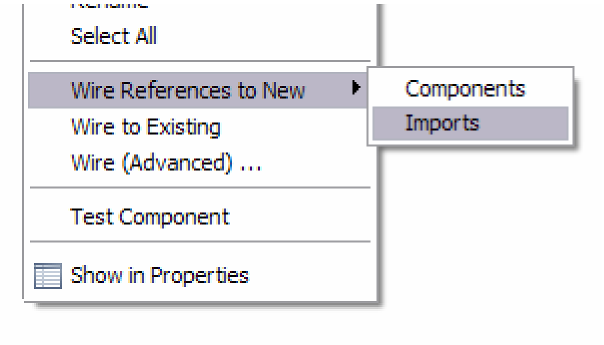


__ c. Hover the mouse over the interface on the Export. Notice how it exposes the CleansePublish interface for SCA invocation.

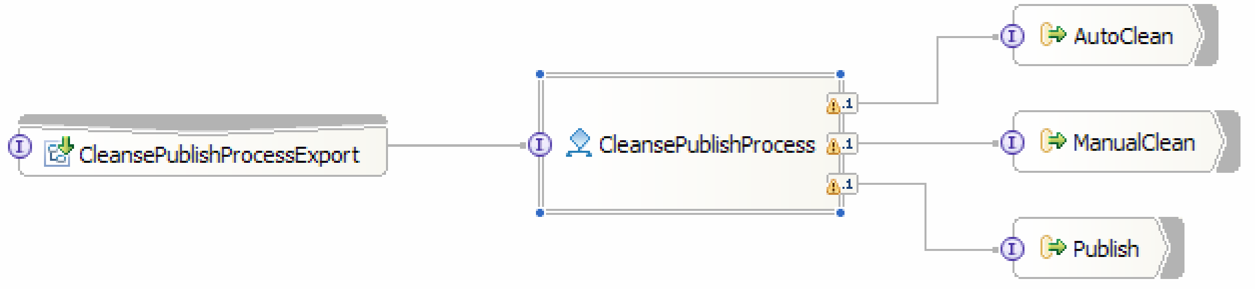

*Assembly Diagram: CleanPublishBPEL
CleansePublishProcessExport
CleansePublishProcess

1 Interfaces found:
CleansePublish (WSDL Interface)

__ 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**.



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

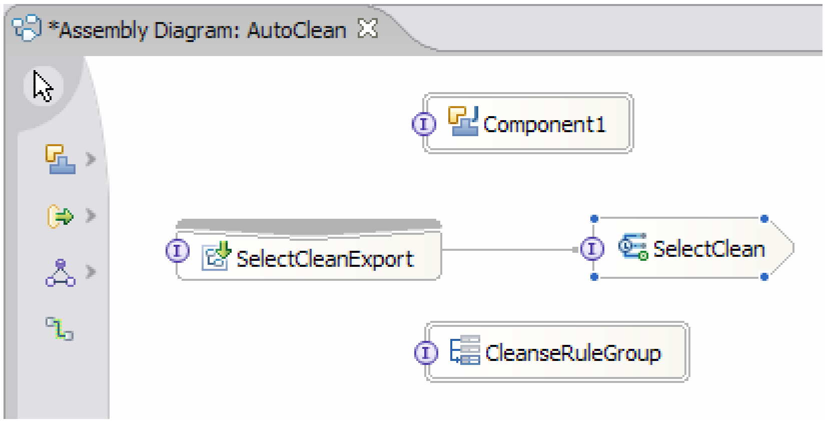


__ f. Save and close the editor for now. The SCA bindings must be set for each Import component to call the appropriate module. The SCA binding can not be completed until the AutoClean, ManualClean, and Publish modules are properly exposed as SCA components. You will return to the CleanPublishBPEL assembly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

__ b. Right-click on **SelectClean** and select **Export…>SCA Binding**. The Selector is the only component in the module that needs to be exposed as an SCA component.

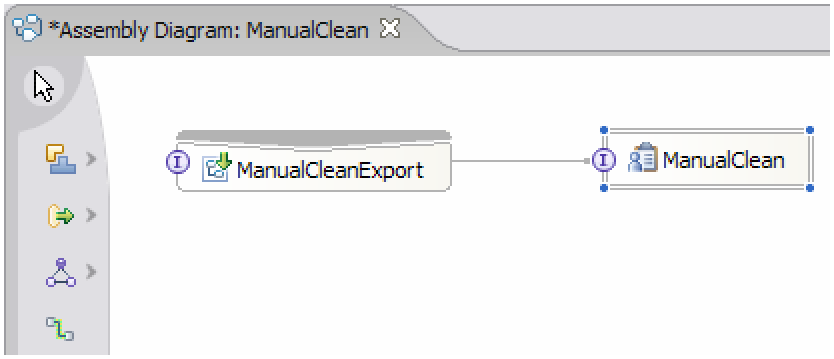


__ c. Save and close the assembly editor. The Selector, which fronts the calls to the module and determines whether to call the Java component or Business Rule, is fully available to be called as an SCA component.

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

__ a. Open the assembly editor for the ManualClean module.

__ b. Right-click on **ManualClean** and select **Export…>SCA Binding**. The human task can be easily exposed as an SCA component as well.

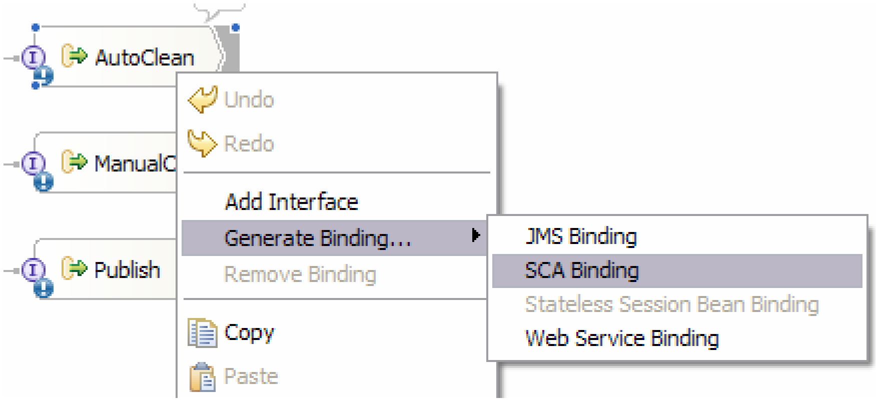


__ c. Save and close the editor. The human task is complete as an SCA component.

____ 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.

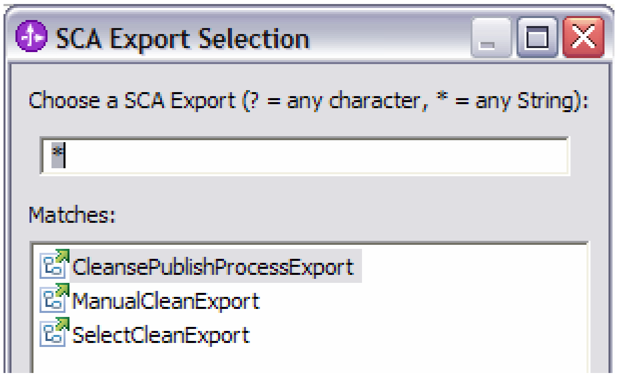
__ a. Open the assembly editor for the CleanPublishBPEL module again.

__ b. Right-click the **AutoClean** Import component and select **Generate Binding… > SCA Binding**.

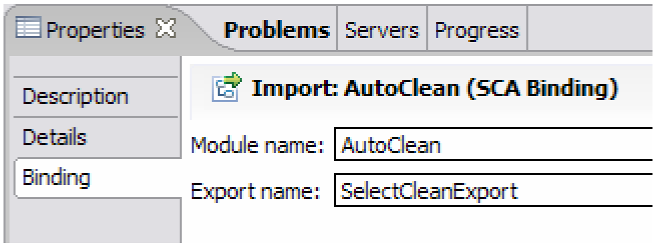


__ c. With the AutoClean Import component selected, select the **Bindings** tab in the **Properties** view.

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


SCA Export Selection
Choose a SCA Export (? = any character, * = any String):
*
Matches:
CleansePublishProcessExport
ManualCleanExport
SelectCleanExport

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



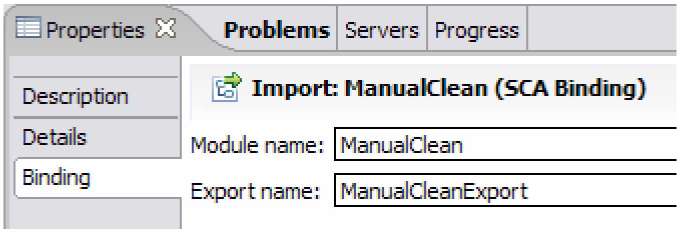
Import: AutoClean (SCA Binding)

Module name: AutoClean

Export name: SelectCleanExport

__ 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.

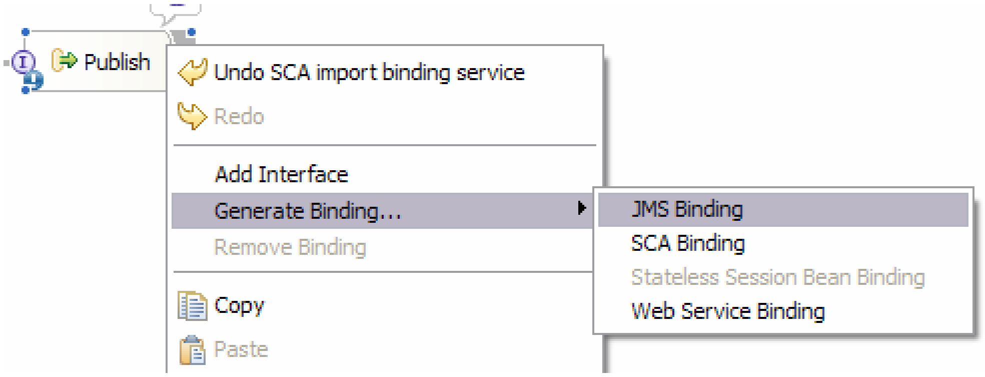


Import: ManualClean (SCA Binding)

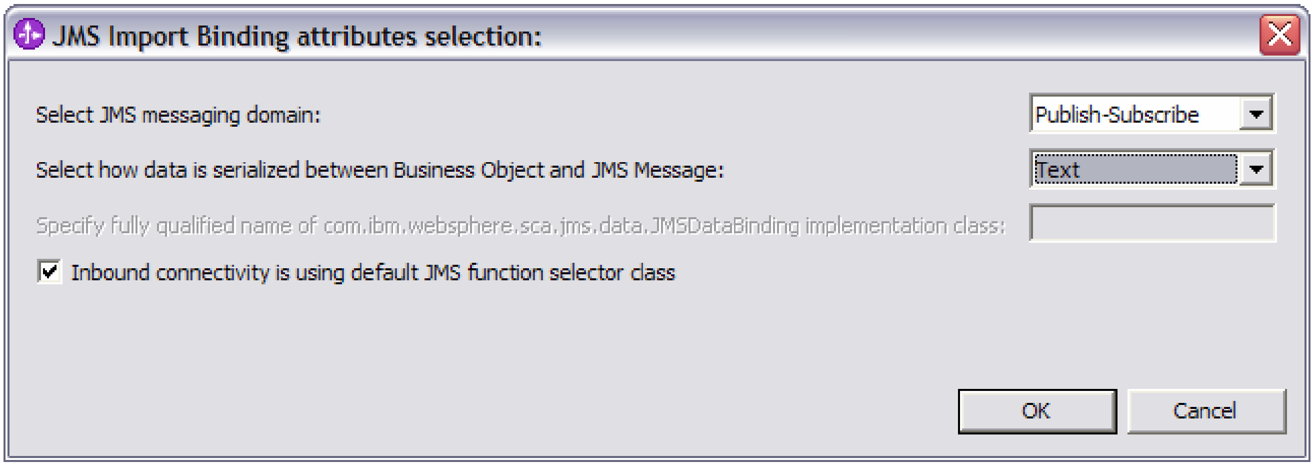
Module name: ManualClean

Export name: ManualCleanExport

__ 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… > JMS Binding**.



__ i. Select **Publish-Subscribe** for the Select JMS message domain. Select **Text** for the data is serialized between Business Object and JMS Message in the **JMS Import Binding attributes selection** dialog.



JMS Import Binding attributes selection:

Select JMS messaging domain: Publish-Subscribe

Select how data is serialized between Business Object and JMS Message: Text

Specify fully qualified name of com.ibm.websphere.sca.jms.data.JMSDataBinding implementation class:

☑ Inbound connectivity is using default JMS function selector class

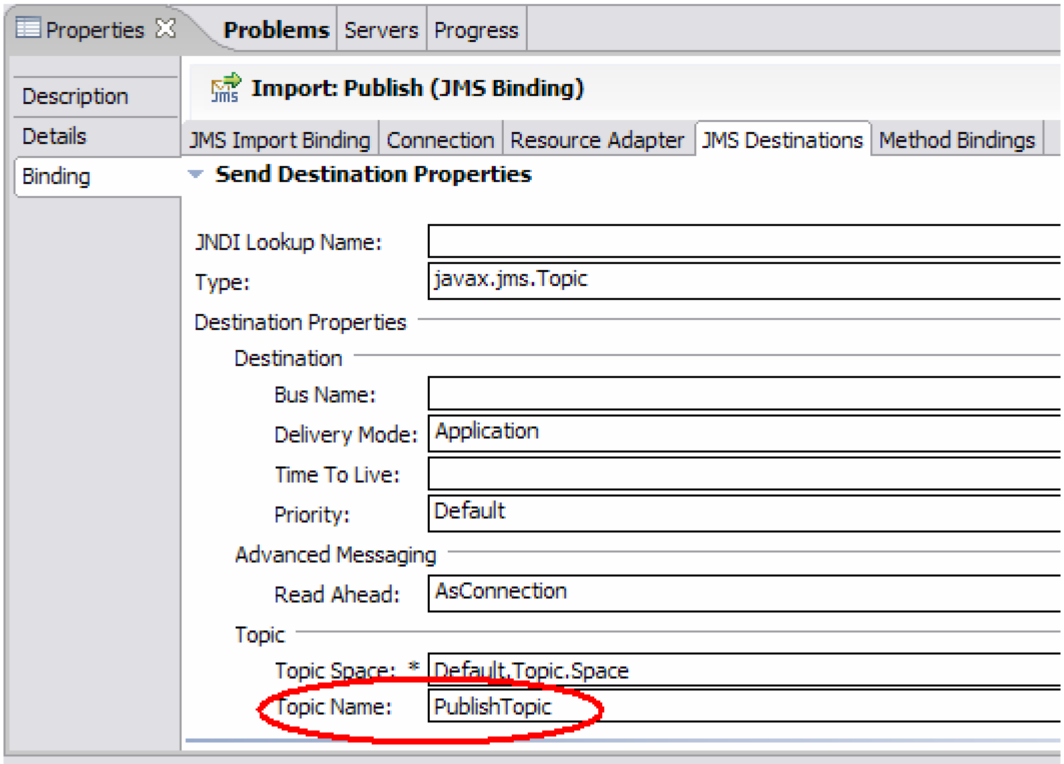
__ j. Click **OK**.

__ k. Select the **Binding** tab in the **Properties** view. The specific topic where the message will be published must be specified to complete the binding.

__ l. Select the **JMS Destinations** tab.

__ m. Click on **Send Destination Properties.**

__ n. 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.



Import: Publish (JMS Binding)

Send Destination Properties

JNDI Lookup Name:
Type: javax.jms.Topic

Destination Properties

Destination

Bus Name:
Delivery Mode: Application
Time To Live:
Priority: Default

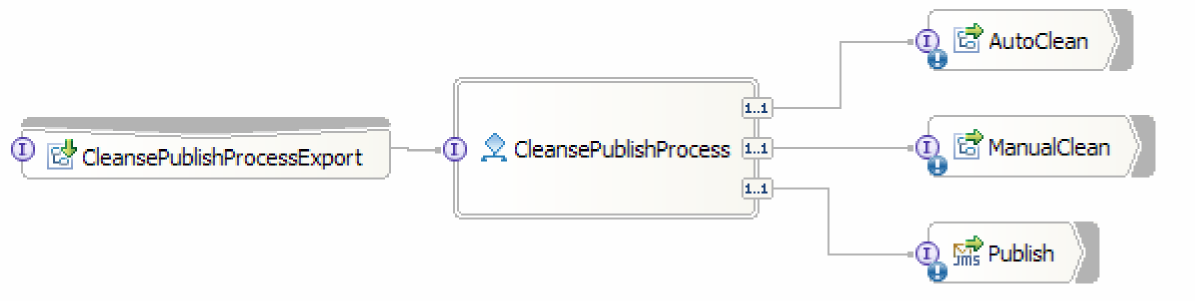
Advanced Messaging

Read Ahead: AsConnection

Topic

Topic Space: * Default.Topic.Space
Topic Name: PublishTopic

__ o. 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.

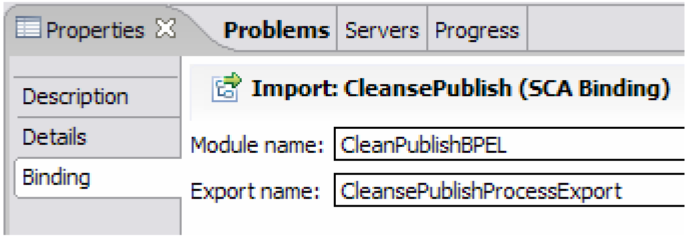


____ 5. The only binding needed for the PeopleSoftAdapter module is to the CleansePublishProcesss.

__ a. Open the assembly editor for the PeopleSoft module.

__ b. Right-click on **InboundDeliveryToCleansePublish** and select **Wire References to New > Imports**. 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 Bindings…>SCA Binding**.

__ c. Browse to and select **CleansePublishProcessExport** on the Bindings tab in the Properties view. This will complete the SCA binding for PeopleSoftAdapter module.



Import: CleansePublish (SCA Binding)

Module name: CleanPublishBPEL

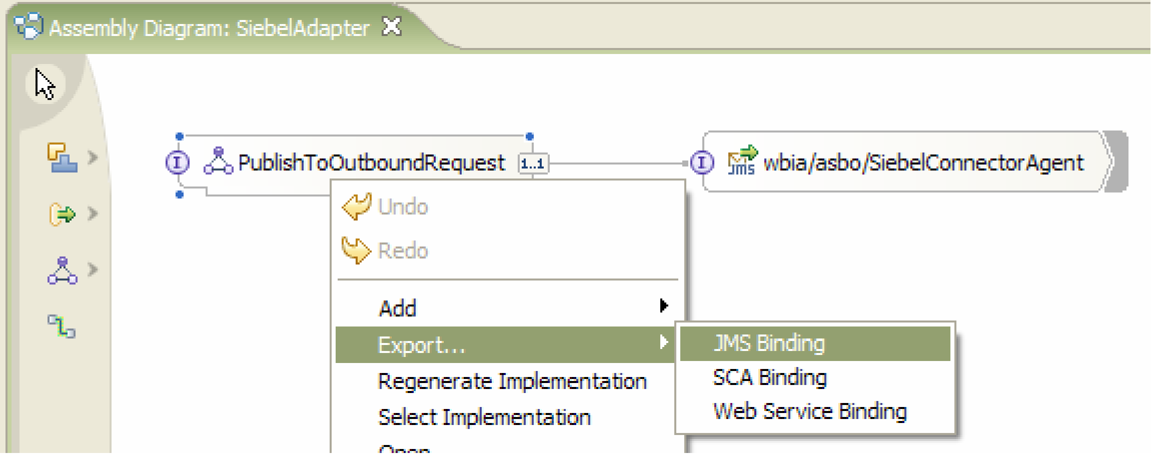
Export name: CleansePublishProcessExport

__ d. Save and close the assembly editor.

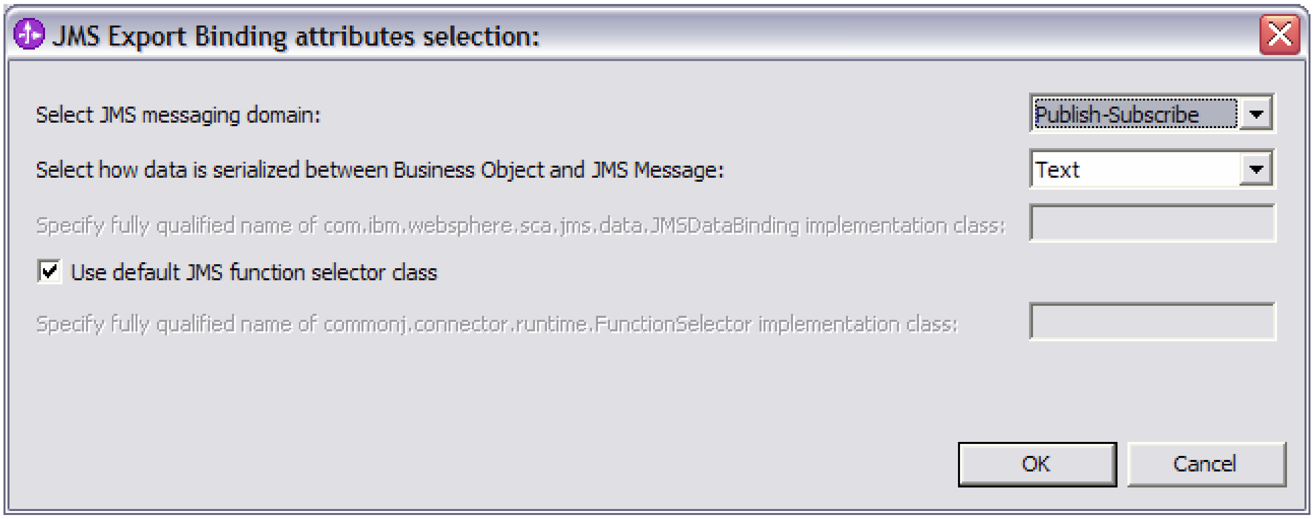
____ 6. The Siebel Adapter must be able to accept the data (ClipBG) published from the CleansePublishProcess to the topic. A JMS binding to the topic must be set. Assemble the SiebelAdapter module to utilize the topic to receive the updated data.

__ a. Open the assembly editor for the SiebelAdapter module.

__ b. Right-click on PublishToOutboundRequest and select **Export…>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 **Text** for the data is serialized between Business Object and JMS Message, in the **JMS Import Binding attributes selection** dialog.


JMS Export Binding attributes selection:

Select JMS messaging domain: Publish-Subscribe

Select how data is serialized between Business Object and JMS Message: Text

Specify fully qualified name of com.ibm.websphere.sca.jms.data.JMSDataBinding implementation class:

☑ Use default JMS function selector class

Specify fully qualified name of commonj.connector.runtime.FunctionSelector implementation class:

__ d. Click **OK**.

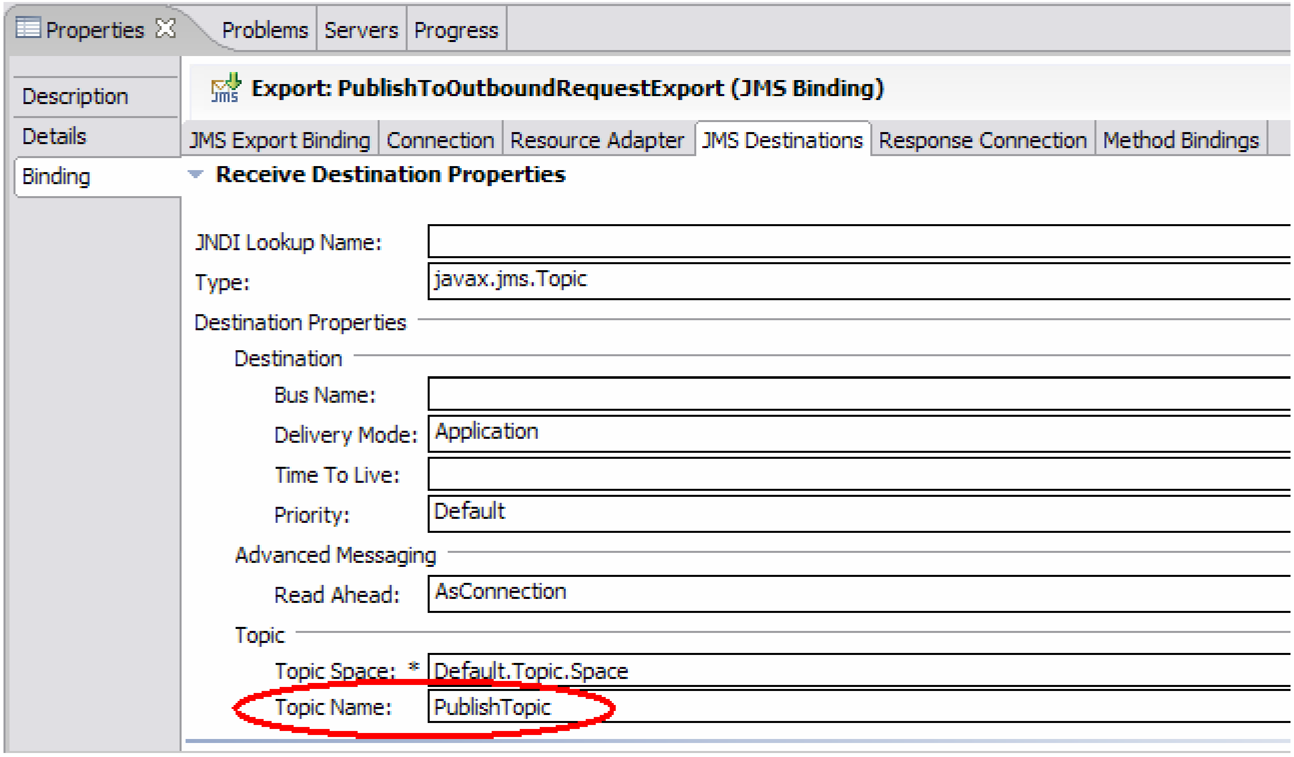
__ e. Select the PublishToOutboundRequestExport you just created and select the **Properties** view.

__ f. Click on the **Binding** tab.

__ g. Select the **JMS Destinations** tab. You must set the Export to use the specific topic where the data (ClipBG) will be published.

__ h. Expand **Receive Destinations Properties**.

__ i. Specify the **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.



Export: PublishToOutboundRequestExport (JMS Binding)

Receive Destination Properties

JNDI Lookup Name:

Type: javax.jms.Topic

Destination Properties

Destination

Bus Name:

Delivery Mode: Application

Time To Live:

Priority: Default

Advanced Messaging

Read Ahead: AsConnection

Topic

Topic Space: * Default.Topic.Space

Topic Name: PublishTopic

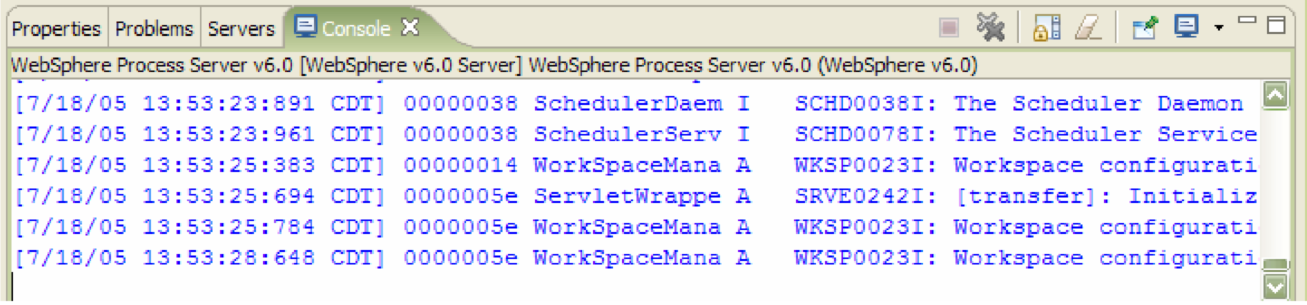
__ j. Save and close the assembly editor. The SiebelAdapter assembly is complete.

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.

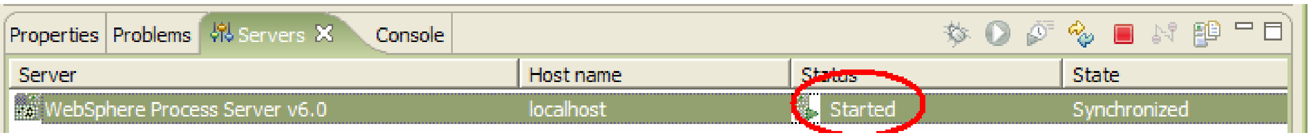
____ 1. If using a remote testing environment, follow directions provided in **Task: Adding Remote Server to WebSphere Integration Developer Test Environment** at the end of this document to add a server to the WebSphere Integration Developer test environment and start it. This is especially true for z/OS, AIX, Solaris remote test environment, where the WebSphere Integration Developer will be remote to the test environment.

__ a. Your Console view will look something like this once the server has started.



```
WebSphere Process Server v6.0 [WebSphere v6.0 Server] WebSphere Process Server v6.0 (WebSphere v6.0)
[7/18/05 13:53:23:891 CDT] 00000038 SchedulerDaem I   SCHD0038I: The Scheduler Daemon
[7/18/05 13:53:23:961 CDT] 00000038 SchedulerServ I   SCHD0078I: The Scheduler Service
[7/18/05 13:53:25:383 CDT] 00000014 WorkSpaceMana A   WKSP0023I: Workspace configurati
[7/18/05 13:53:25:694 CDT] 0000005e ServletWrappe A   SRVE0242I: [transfer]: Initializ
[7/18/05 13:53:25:784 CDT] 0000005e WorkSpaceMana A   WKSP0023I: Workspace configurati
[7/18/05 13:53:28:648 CDT] 0000005e WorkSpaceMana A   WKSP0023I: Workspace configurati
```

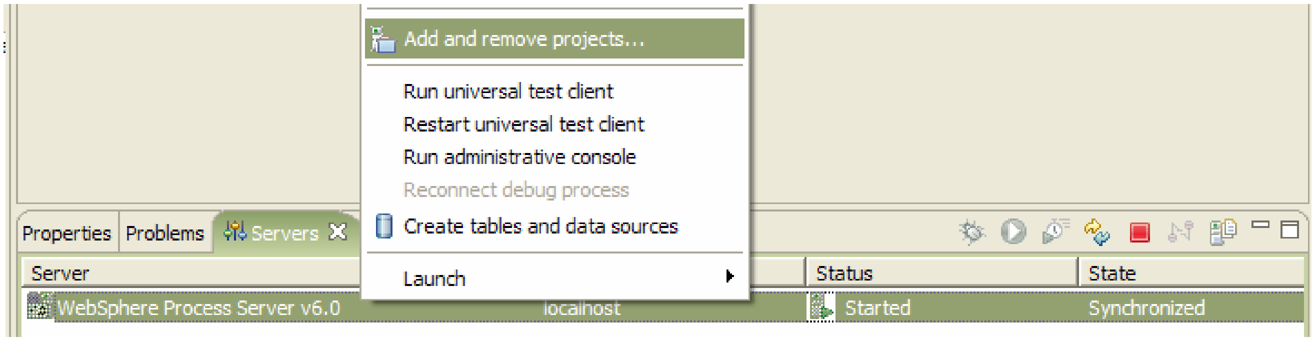
__ b. Click on the **Servers** tab and verify that the server status shows as **Started**



Server	Host name	Status	State
WebSphere Process Server v6.0	localhost	Started	Synchronized

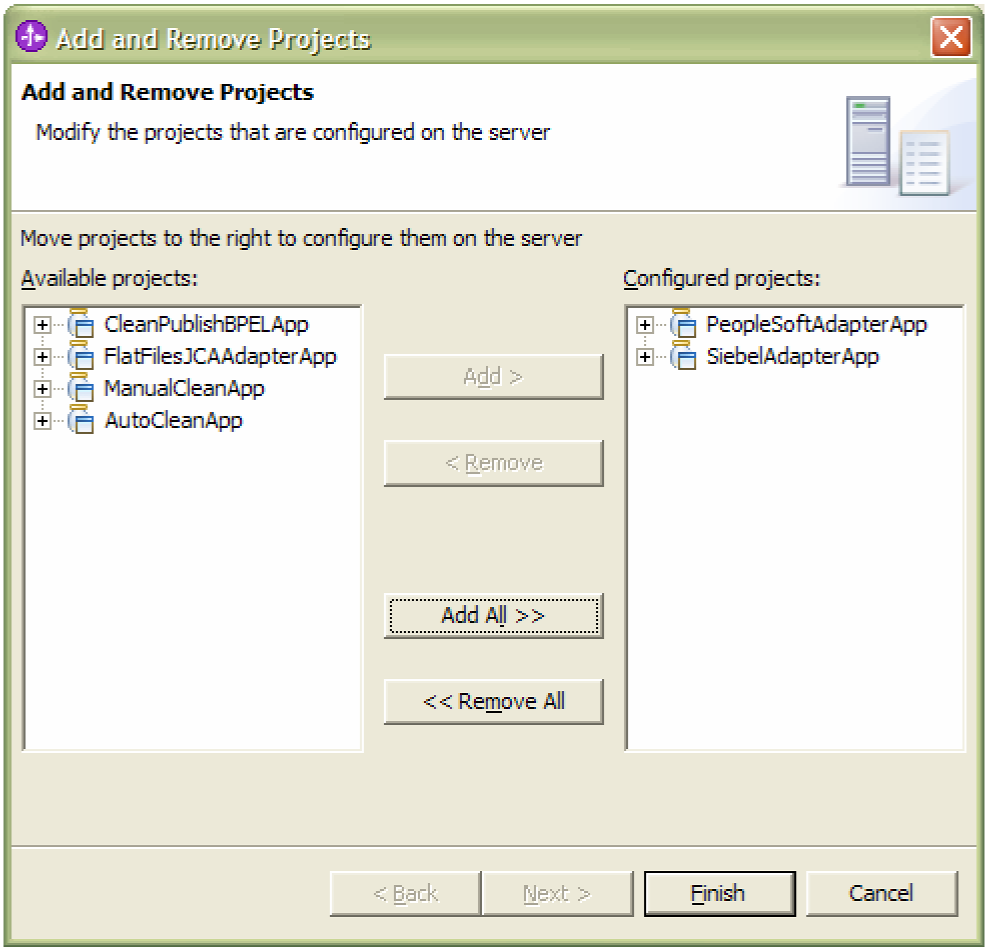
____ 2. Publish the PeopleSoftAdapter and SiebelAdapter modules so that the required messaging queues can be configured.

__ a. Right click on the server and select **Add and remove projects…** from the pop-up menu.


Add and remove projects…
Run universal test client
Restart universal test client
Run administrative console
Reconnect debug process
Create tables and data sources
Launch
Properties
Problems
Servers

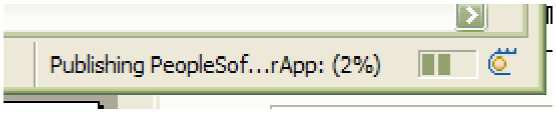
Server	Host name	Status	State
WebSphere Process Server v6.0	localhost	Started	Synchronized

__ b. Select the **PeopleSoftAdapterApp** in the Add and Remove Projects dialog and use the **Add>** button to add it to the Configured projects list. Repeat this for the **SiebelAdapterApp**. The dialog should now look like this.



__ c. Click **Finish.**

__ d. In the lower right of the window you will see messages while the publishing is proceeding. Wait until there are no longer any messages.


Publishing PeopleSof...rApp: (2%)

____ 3. Now that the PeopleSoft and Siebel projects have been started in the server for the first time, the messaging configuration required by them has been defined to the server. The server must now be restarted in order for that messaging configuration to be activated.

NOTE: As part of publishing the SiebelAdapterApp you may see some exceptions. Ignore the exceptions. There is a workaround that is specified at a later stage in this lab

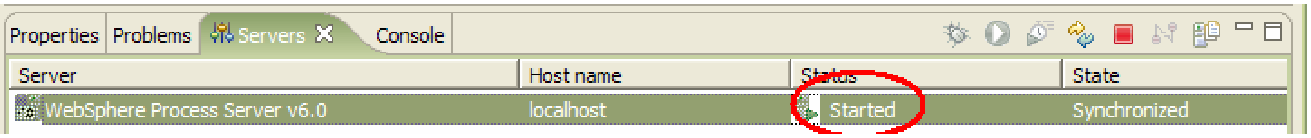
NOTE: The requirement for a server restart after the installation of an application containing a WBI Adapter Import or Export is a one time requirement. After this has been done the first time, the required messaging configuration is defined to the server. Additional applications using WBI Adapters can be installed without requiring a server restart.

__ a. Restart the server. Right click on WebSphere Process Server v6.0 server from the Servers view and select **Restart** from the context menu, or **Stop** if you are using a remote test environment.

__ b. Once the server is stopped, follow directions provided in **Task: Adding Remote Server to WebSphere Integration Developer Test Environment** to restart the server. This is especially true for z/OS, AIX, Solaris remote test environment, where the WebSphere Integration Developer will be remote to the test environment.

__ c. Wait for the server to fully restart.

NOTE: When the server restarts, there might be messages in the Console view. Wait until you see the server return to a Started status.

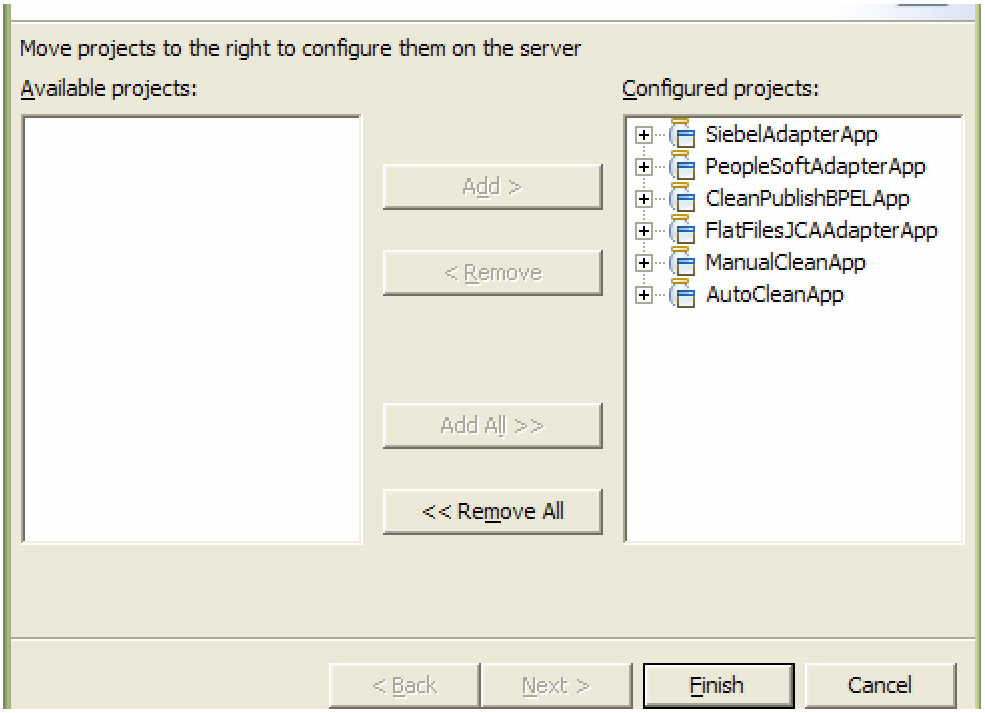


Server	Host name	Status	State
WebSphere Process Server v6.0	localhost	Started	Synchronized

____ 4. Publish all of the projects to the server.

__ a. Right click on the server and select **Add and remove projects…** from the pop-up menu.

__ b. Click the **Add All>>** button.


Move projects to the right to configure them on the server

Available projects:

Configured projects:

- SiebelAdapterApp
- PeopleSoftAdapterApp
- CleanPublishBPELApp
- FlatFilesJCAAdapterApp
- ManualCleanApp
- AutoCleanApp

__ 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.

NOTE: If the SiebelAdapterApp publishing results in exceptions, follow the steps below as a workaround

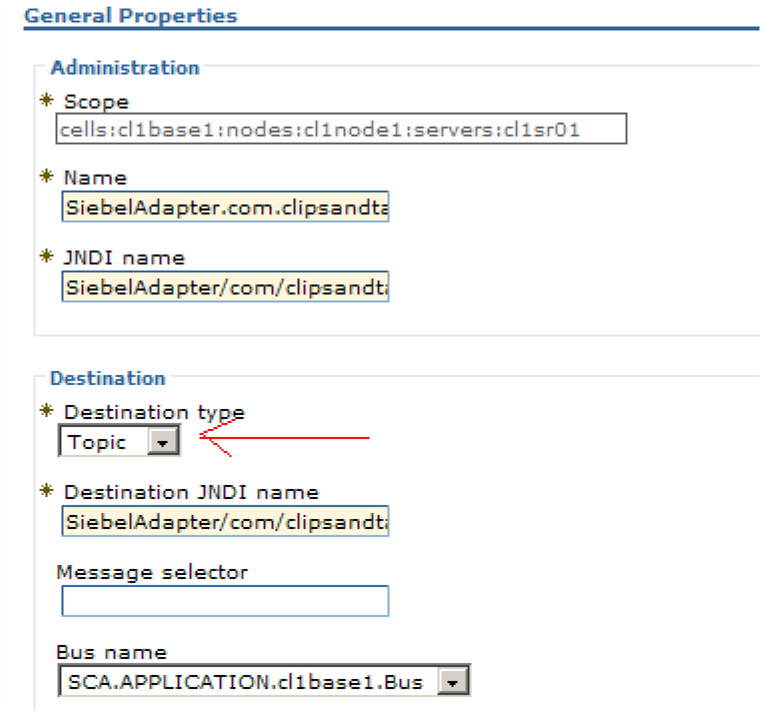
In the Administrative Console, navigate to Resources → JMS Providers → Default Messaging

For the scope, select the Server scope

Click on **the "JMS activation specification"** link under **"Activation Specifications"**

Click on the SiebelAdapter.com.clipsandtacks.scm.PublishToOutboundRequestExport_AS link

Change the Destination Type from Queue to Topic



General Properties

Administration

* Scope: cells:cl1base1:nodes:cl1node1:servers:cl1sr01

* Name: SiebelAdapter.com.clipsandta

* JNDI name: SiebelAdapter/com/clipsandt:

Destination

* Destination type: Topic

* Destination JNDI name: SiebelAdapter/com/clipsandt:

Message selector:

Bus name: SCA.APPLICATION.cl1base1.Bus

Click Ok and save the changes.

Got to Applications → Enterprise Applications. select the check box next to SiebelAdapterApp and click the start button. This will start the SiebelAdpaterApp application

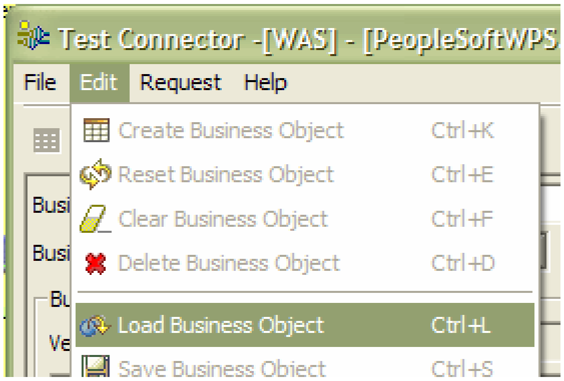
____ 5. Once the server is running with the entire solution deployed, you are ready to test it. To do that, start the Visual Test Connectors that will be used to simulate the PeopleSoft and Siebel applications.

__ 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\PsftConnectorWPS.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.**

____ 6. Load a predefined business object into the Visual Test Connector for the PeopleSoft and examine its values.

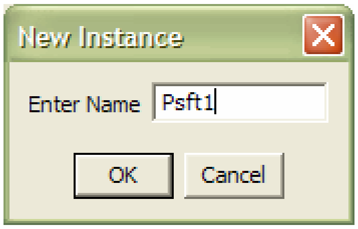
__ a. In the VTC window for PeopleSoft, select **Edit > Load Business Object**.


Test Connector -[WAS] - [PeopleSoftWPS
File
Edit
Request
Help
Create Business Object Ctrl+K
Reset Business Object Ctrl+E
Clear Business Object Ctrl+F
Delete Business Object Ctrl+D
Load Business Object Ctrl+L
Save Business Object Ctrl+S
Busi
Busi
Bu
Ve

__ b. Navigate to **c:\Labfiles60\eXchange\PeopleSoft\ASBOInstances** in the Open dialog, and select **PsftASBO1.bo**.

__ c. Click **Open.**

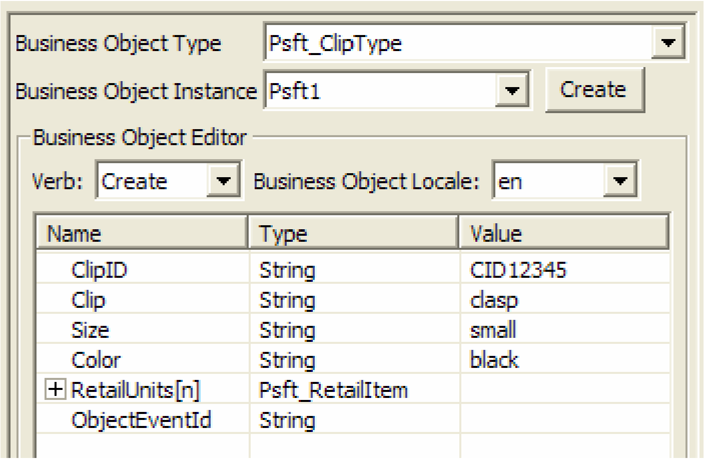
__ d. Enter a name to be assigned to this instance, such as **Psft1**.



Enter Name Psft1

__ e. Click **OK**.

__ f. Look at the values assigned to the Psft_ClipType fields.



Business Object Type: Psft_ClipType

Business Object Instance: Psft1

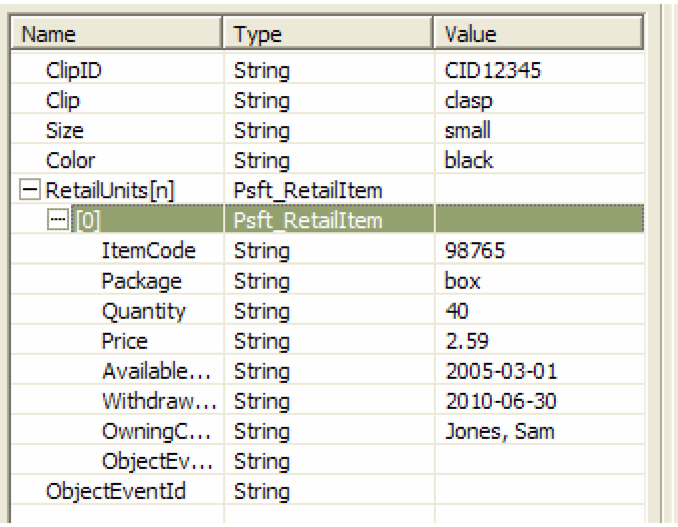
Verb: Create

Business Object Locale: en

Name	Type	Value
ClipID	String	CID12345
Clip	String	clasp
Size	String	small
Color	String	black
+ RetailUnits[n]	Psft_RetailItem	
ObjectEventId	String	

__ g. Click the **+** in front of **RetailUnits[n]** to expand the Psft_RetailItem array.

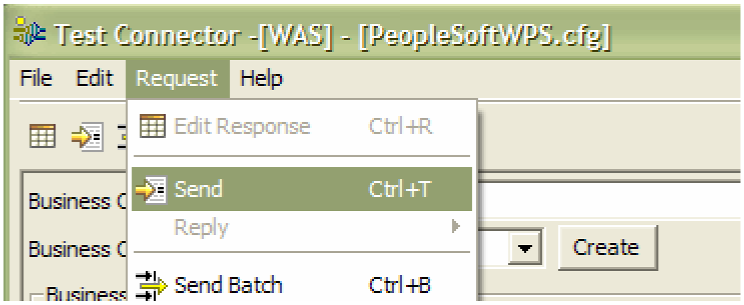
__ h. Click the **+** in front of **[0]** to see an instance of Psft_RetailItem.



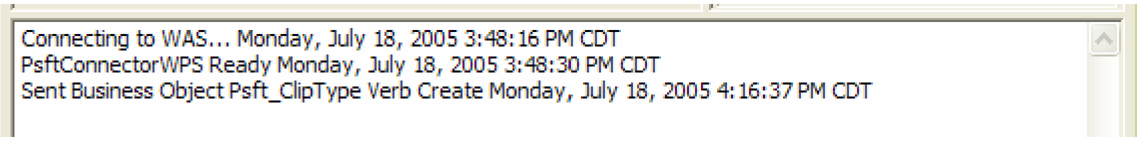
Name	Type	Value
ClipID	String	CID12345
Clip	String	clasp
Size	String	small
Color	String	black
⊟ RetailUnits[n]	Psft_RetailItem	
⊟ [0]	Psft_RetailItem	
ItemCode	String	98765
Package	String	box
Quantity	String	40
Price	String	2.59
Available...	String	2005-03-01
Withdraw...	String	2010-06-30
OwningC...	String	Jones, Sam
ObjectEv...	String	
ObjectEventId	String	

____ 7. Initiate the test by sending the business object from the PeopleSoft VTC to the WebSphere Process Server.

__ a. Select **Request > Send.**



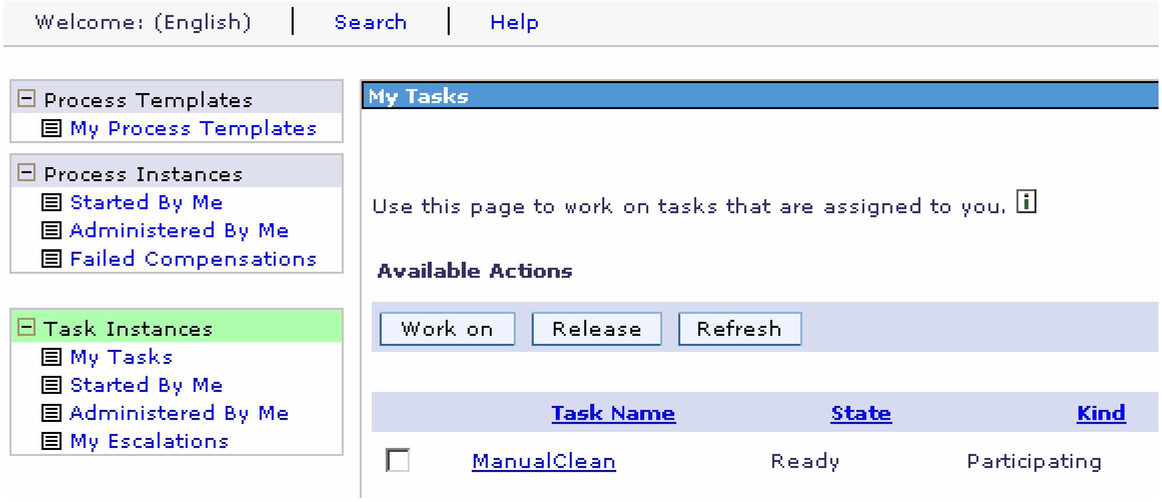
__ b. You should see a message in the VTC indicating that the message was sent.



```
Connecting to WAS... Monday, July 18, 2005 3:48:16 PM CDT
PsftConnectorWPS Ready Monday, July 18, 2005 3:48:30 PM CDT
Sent Business Object Psft_ClipType Verb Create Monday, July 18, 2005 4:16:37 PM CDT
```

____ 8. 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 WPS Server v6.0 in the Servers view and select **Launch > BPC Explorer**. 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.

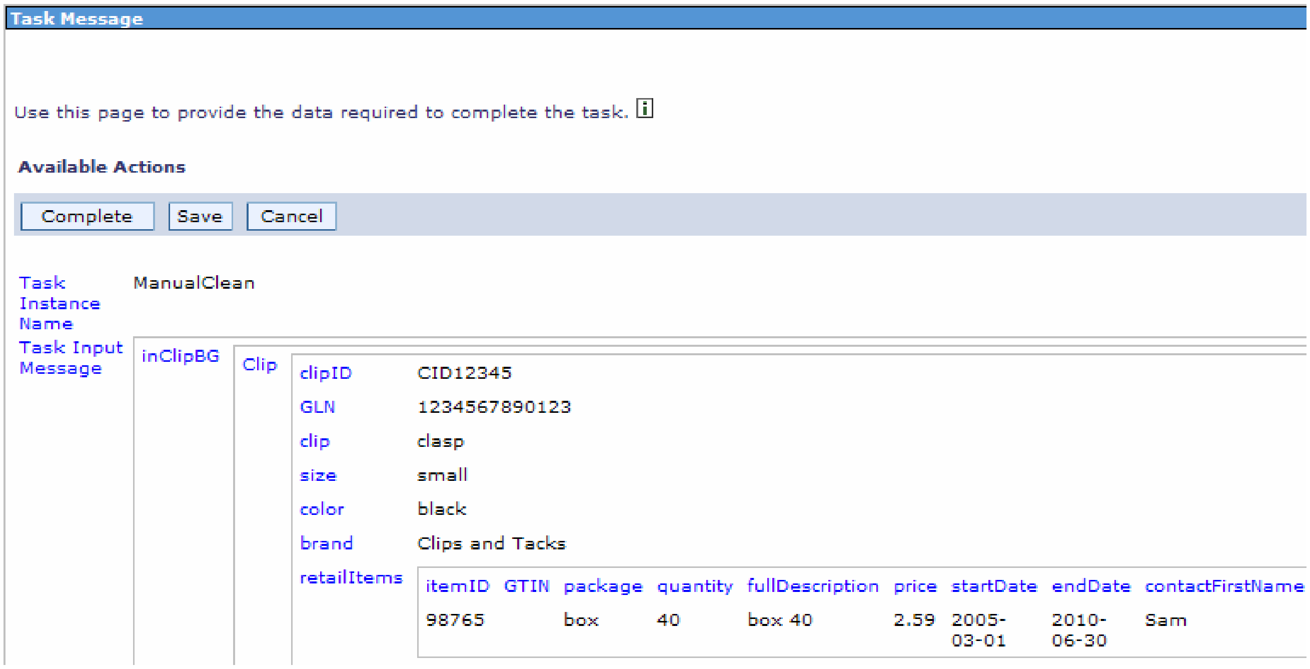


Use this page to work on tasks that are assigned to you.

	Task Name	State	Kind
□	ManualClean	Ready	Participating

_____ 9. Complete the task so that the flow of the scenario can continue.

__ a. Click on the check box for instance to select it and then click "**Work on**". This will claim the task on behalf of the current user.



Use this page to provide the data required to complete the task.

Task Instance Name: ManualClean

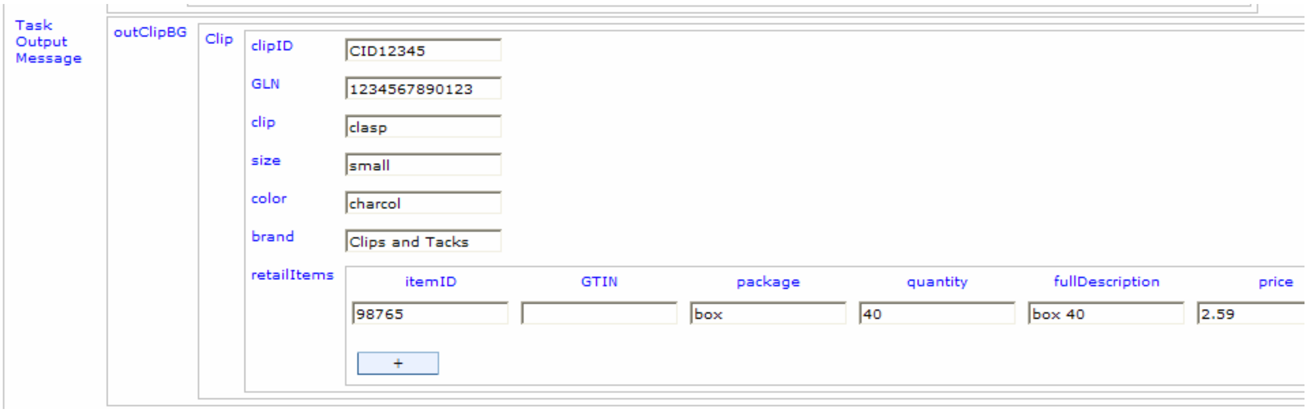
Task Input Message: inClipBG → Clip

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 screen shot. You will need to click the **+** in **retailItems** to fill in the data values for an instance.



Task Output Message

outClipBG

Clip

clipID	CID12345
GLN	1234567890123
clip	clasp
size	small
color	charcol
brand	Clips and Tacks

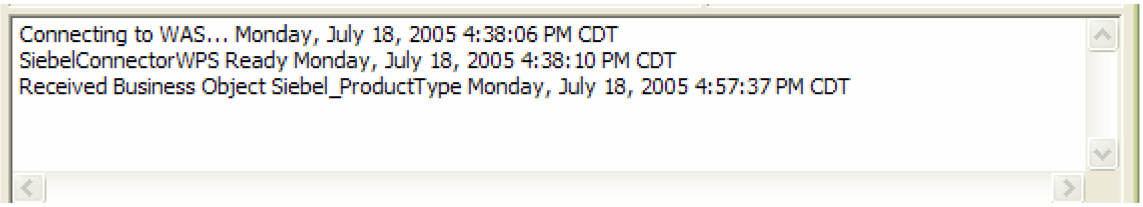
retailItems

itemID	GTIN	package	quantity	fullDescription	price
98765		box	40	box 40	2.59

__ c. Click the **Complete** button.

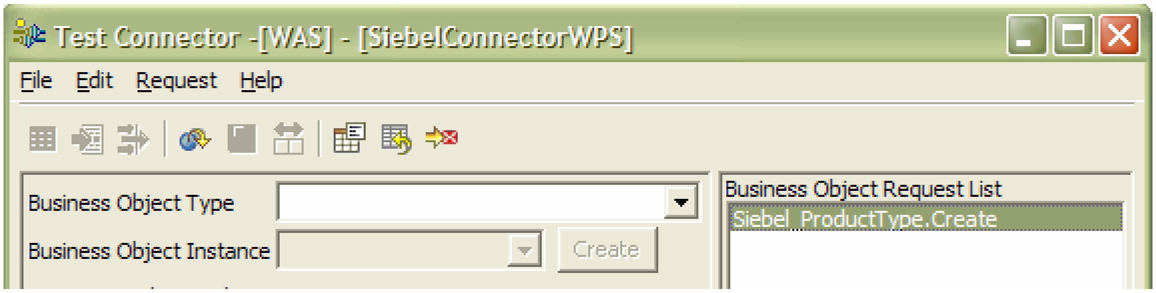
_____ 10. Verify the data that has been sent to Siebel.

__ a. Look at the Visual Test Connector window for Siebel and verify that it has received a business object.



```
Connecting to WAS... Monday, July 18, 2005 4:38:06 PM CDT
SiebelConnectorWPS Ready Monday, July 18, 2005 4:38:10 PM CDT
Received Business Object Siebel_ProductType Monday, July 18, 2005 4:57:37 PM CDT
```

__ b. Double click on **Siebel_ProductType.create** to open a window with the received BO.



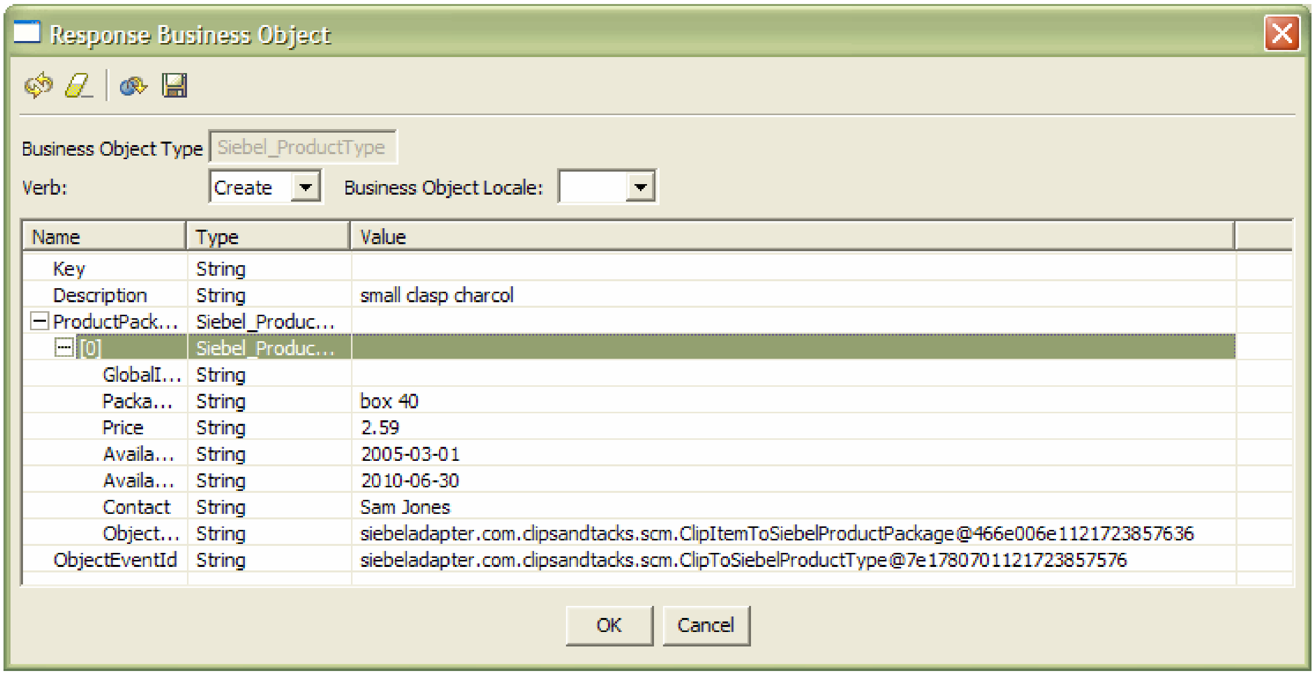
Business Object Type

Business Object Instance

Business Object Request List

Siebel_ProductType.Create

__ c. Examine the data in the Response Business Object window.



Response Business Object

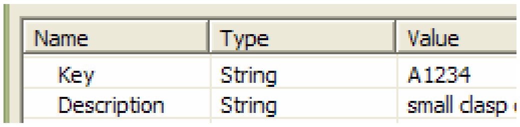
Business Object Type: Siebel_ProductType

Verb: Create

Business Object Locale:

Name	Type	Value
Key	String	
Description	String	small clasp charcol
⊟ProductPack...	Siebel_Produc...	
⊟[0]	Siebel_Produc...	
GlobalI...	String	
Packa...	String	box 40
Price	String	2.59
Availa...	String	2005-03-01
Availa...	String	2010-06-30
Contact	String	Sam Jones
Object...	String	siebeladapter.com.clipsandtacks.scm.ClipItemToSiebelProductPackage@466e006e1121723857636
ObjectEventId	String	siebeladapter.com.clipsandtacks.scm.ClipToSiebelProductType@7e1780701121723857576

__ d. Type in a value for the Key field, such as **A1234.**



Name	Type	Value
Key	String	A1234
Description	String	small clasp

__ e. Click **OK.**

____ 11. Send a response back to the server indicating the request was received and providing the Key value.

__ a. Ensure that S**iebel_ProductType.create** is selected.

__ b. Select **Request -> Reply -> Success** from the menu,

____ 12. Now that testing is complete, shut down your environment.

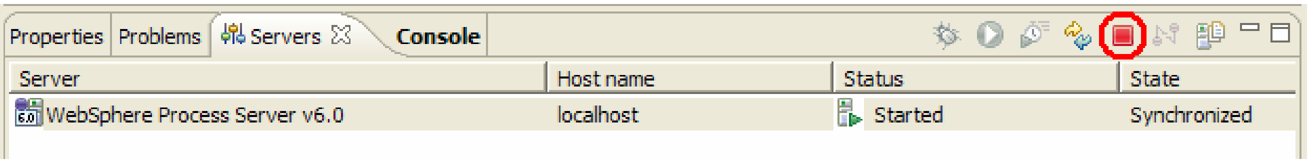
__ a. Right click on the server in the Servers view and select **Add and remove projects…** from the pop-up menu.

__ b. Use the **<<Remove All** button to remove the projects from the server. The dialog should now look like this:

__ c. Click **Finish.**

__ d. Wait until the projects complete unpublishing and there are no more messages in the lower right corner.

__ e. Click on the **Servers** tab and then the **Stop the server** icon.



Server	Host name	Status	State
WebSphere Process Server v6.0	localhost	Started	Synchronized

__ f. 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:\Labfiles60\eXchange\EndToEndNoFFA\workspace

<PROJECT_INTERCHANGE>

C:\Labfiles60\eXchange\EndToEndNoFFA\Solution\EndToEnd_PI.zip

<MODULE>

n/a

<DEPENDENT_LIBRARIES>

n/a

____ **2.** Continue with **Part 4: Testing eXchange.**

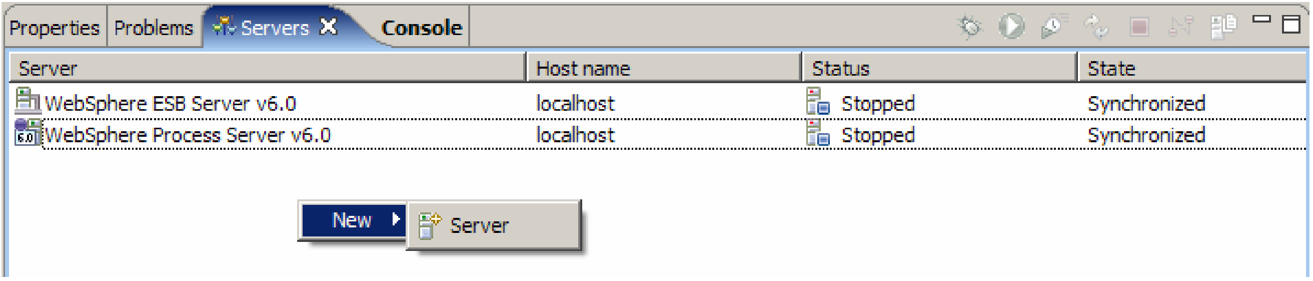
Task: Adding Remote Server to WebSphere Integration Developer Test Environment

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

____ 1. Create a new remote server.

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

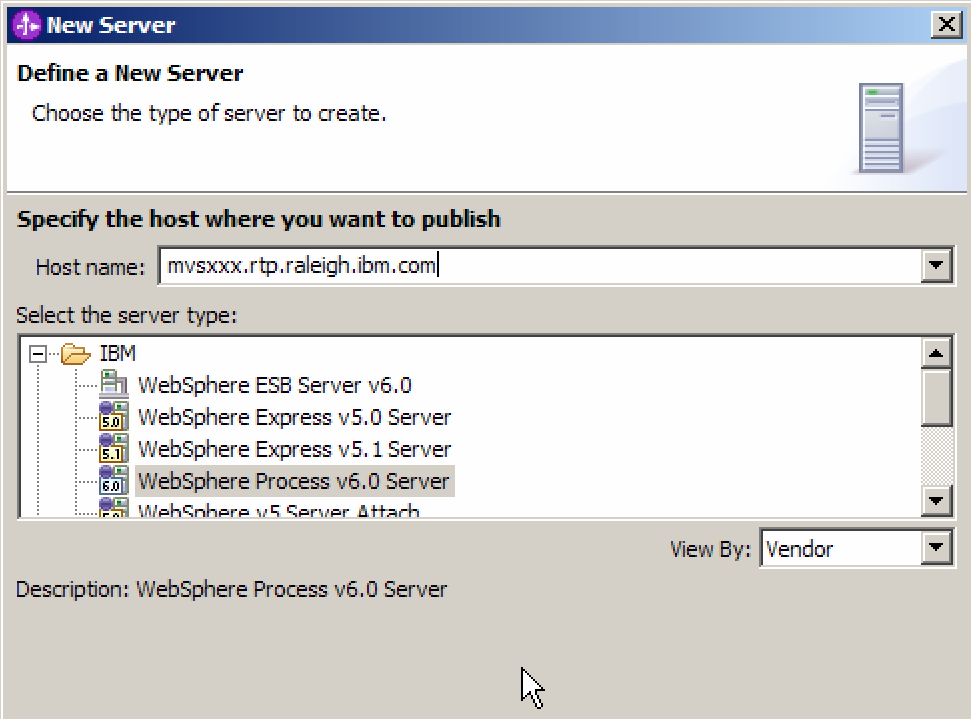
b. Select **New > Server.**



Server	Host name	Status	State
WebSphere ESB Server v6.0	localhost	Stopped	Synchronized
WebSphere Process Server v6.0	localhost	Stopped	Synchronized

c. Specify hostname to the remote server, **<HOSTNAME>**.

d. Ensure that '**WebSphere Process v6.0 Server**' is highlighted in the server type list.



Define a New Server

Choose the type of server to create.

Specify the host where you want to publish

Host name: mvsxxx.rtp.raleigh.ibm.com

Select the server type:

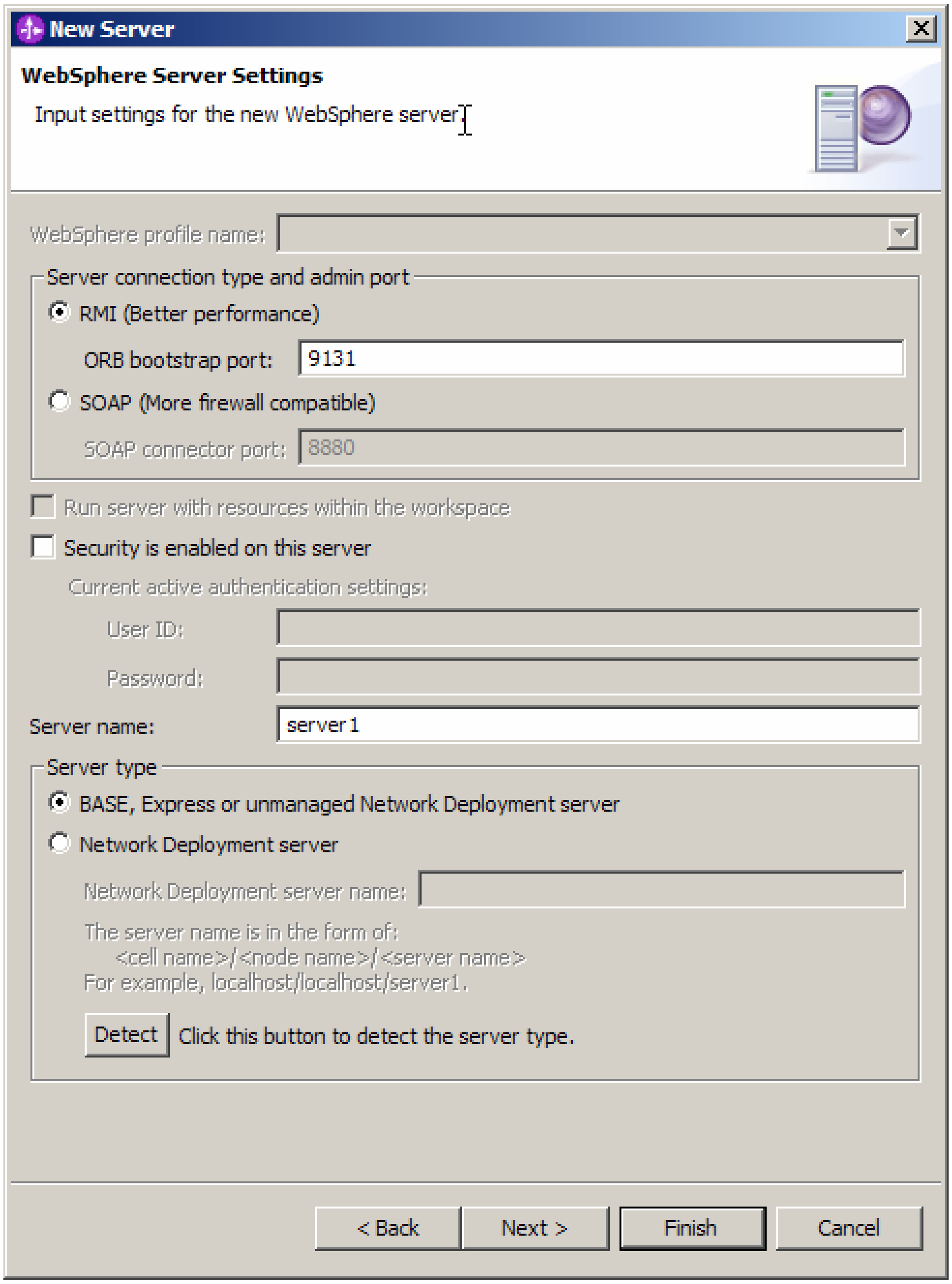
- IBM
  - WebSphere ESB Server v6.0
  - WebSphere Express v5.0 Server
  - WebSphere Express v5.1 Server
  - WebSphere Process v6.0 Server
  - WebSphere v5 Server Attach

View By: Vendor

Description: WebSphere Process v6.0 Server

e. Click **Next.**

f. On the WebSphere Server Settings page, select the radio button for **RMI** and change the ORB bootstrap port to the correct setting (**<BOOTSTRAP_PORT>**).



New Server

WebSphere Server Settings

Input settings for the new WebSphere server.

WebSphere profile name:

Server connection type and admin port

● RMI (Better performance)

ORB bootstrap port: 9131

○ SOAP (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:

Password:

Server name: server1

Server type

● BASE, Express or unmanaged Network Deployment server

○ Network Deployment server

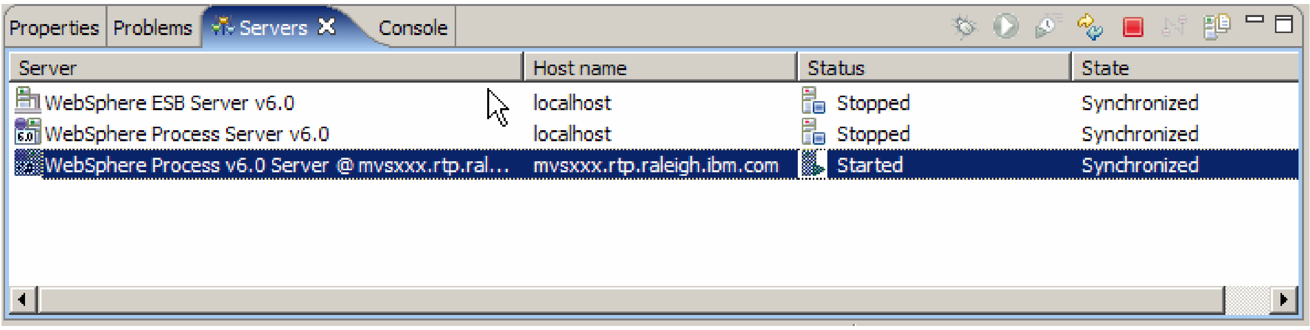
Network Deployment server name:

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

Detect Click this button to detect the server type.

g. Click **Finish**.

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



Server	Host name	Status	State
WebSphere ESB Server v6.0	localhost	Stopped	Synchronized
WebSphere Process Server v6.0	localhost	Stopped	Synchronized
WebSphere Process v6.0 Server @ mvsxxx.rtp.ral...	mvsxxx.rtp.raleigh.ibm.com	Started	Synchronized

____ 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:

‘**telnet <HOSTNAME> <TELNET_PORT>**’

userid : **<USERID>**

pw : **<PASSWORD>**

If the remote server runs on a System i® server, sign on to the system and start a Qshell session.

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 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).

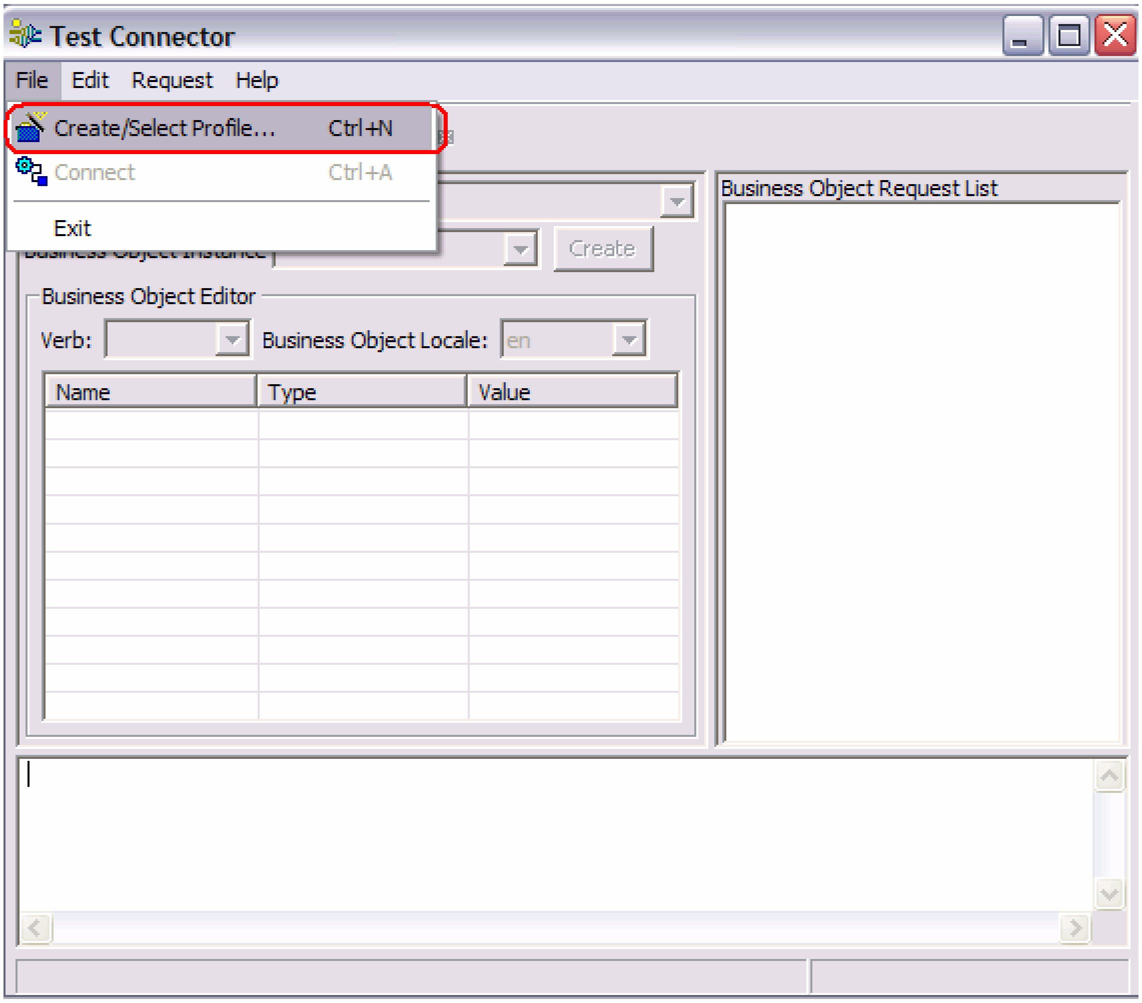
____ 1. Ensure the WebSphere Process Server is active (the server is the JMS Broker). The VTC requires an active JMS Broker to start it's connection.

____ 2. Start the Visual Test Connector

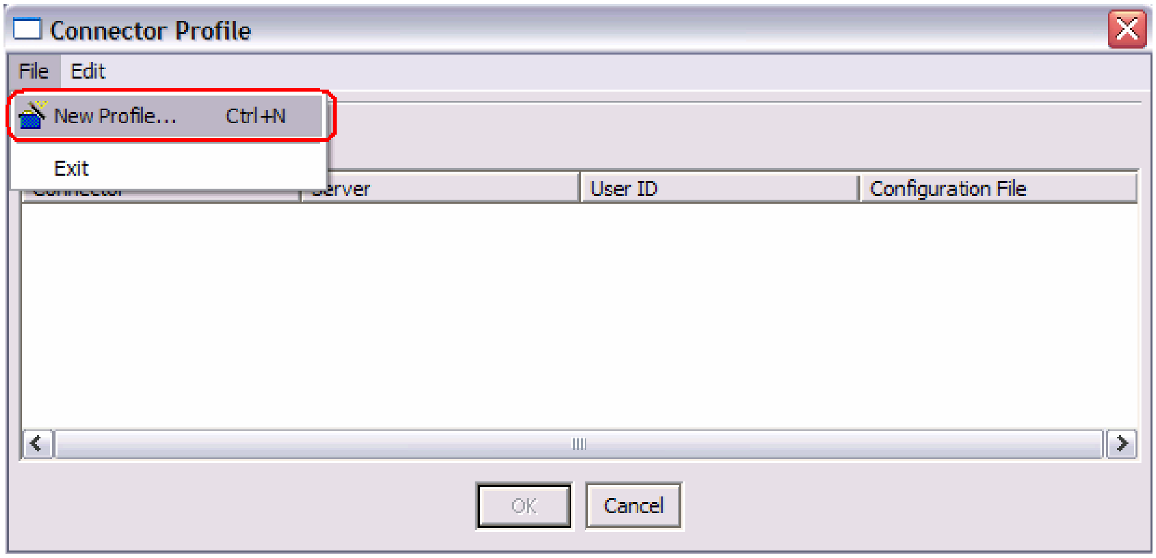
__ a. From the **start** menu, select **Programs > IBM WebSphere Business Integration Adapters > IBM WebSphere Business Integration Toolset > Visual Test Connector** (VTC). This opens a command window in the background where the process runs and the Visual Test Connector Interface in the foreground. Do not close the command window as this will cause the VTC to exit.

____ 3. Create/Select a Profile

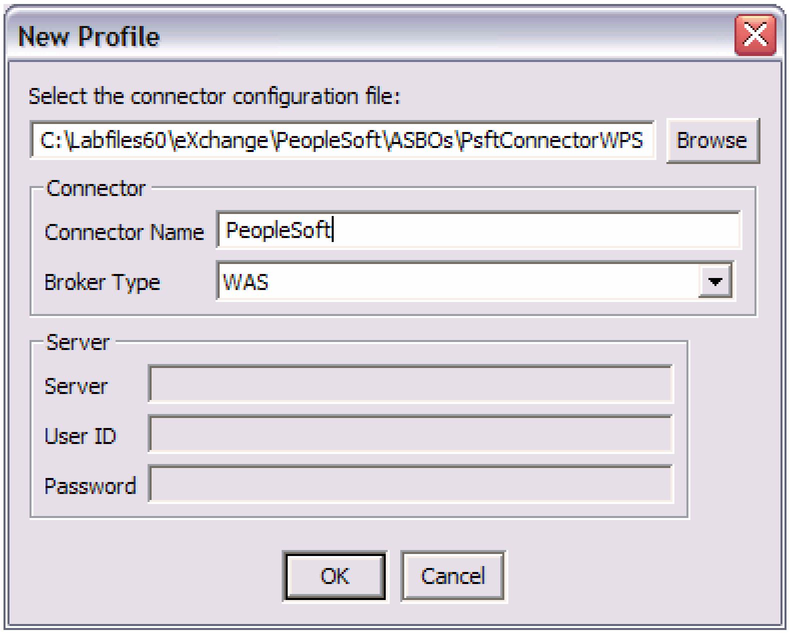
__ a. From the VTC interface, select **File > Create/Select Profile**



__ b. If a profile does not already exist, create one by selecting **File > New Profile …** If a profile does already exist skip to step d.



__ c. Browse to and select the connector configuration file, type in a Connector Name for the Profile, and select WAS for the Broker Type. Click OK.


New Profile

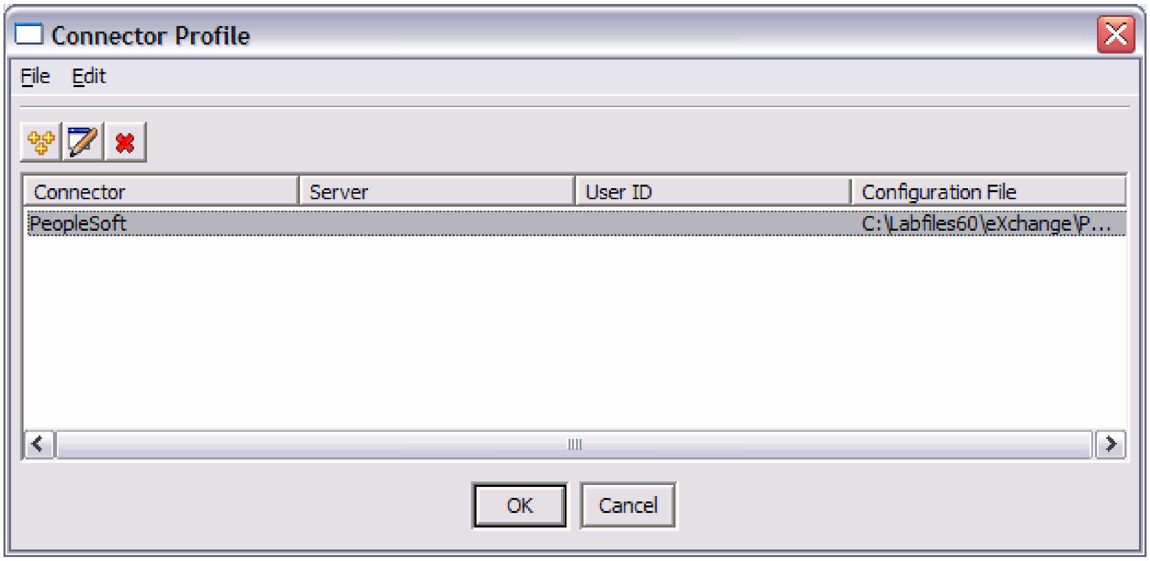
Select the connector configuration file: C:\Labfiles60\eXchange\PeopleSoft\ASBOs\PsftConnectorWPS

Connector

Connector Name: PeopleSoft

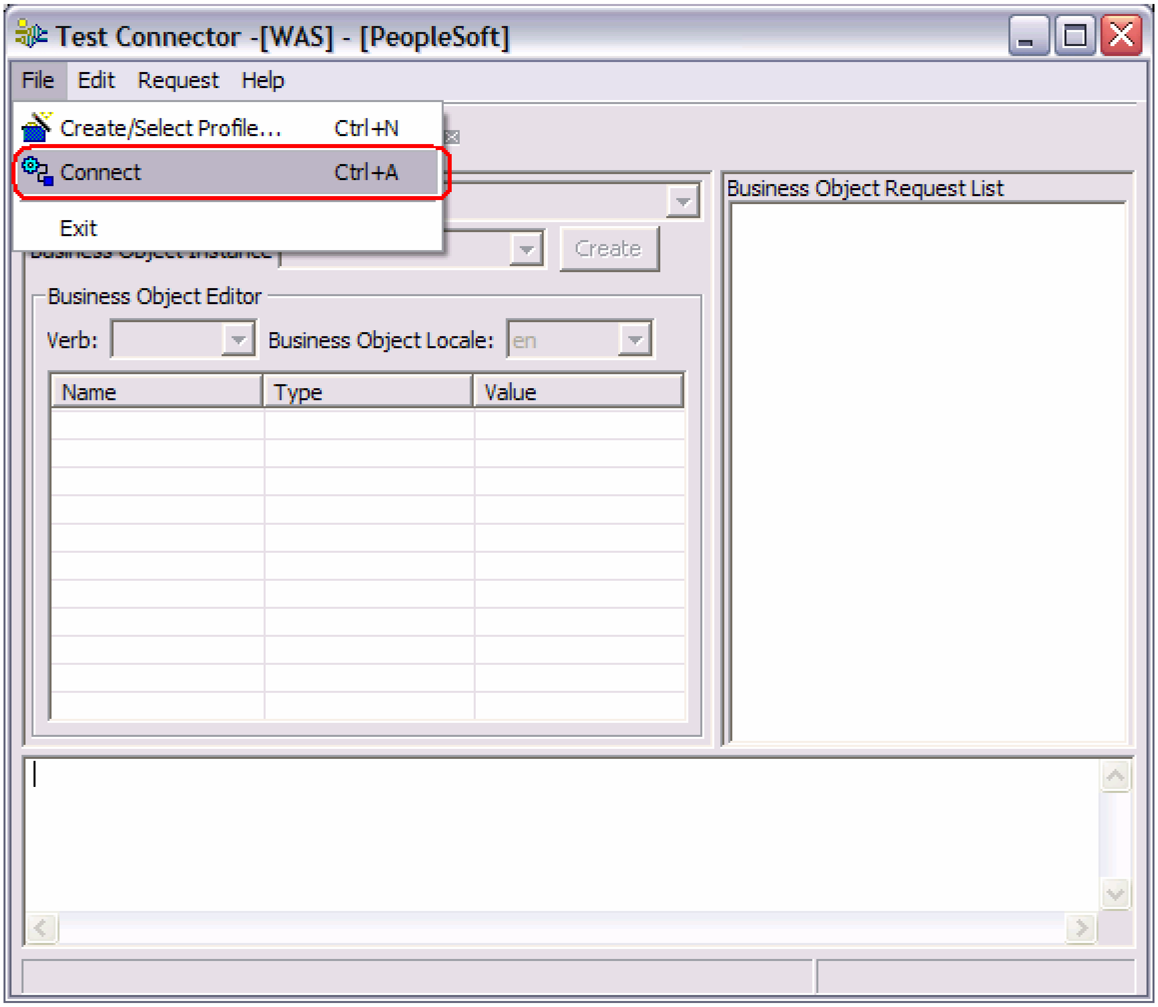
Broker Type: WAS

__ d. Select and highlight the profile and click **OK**

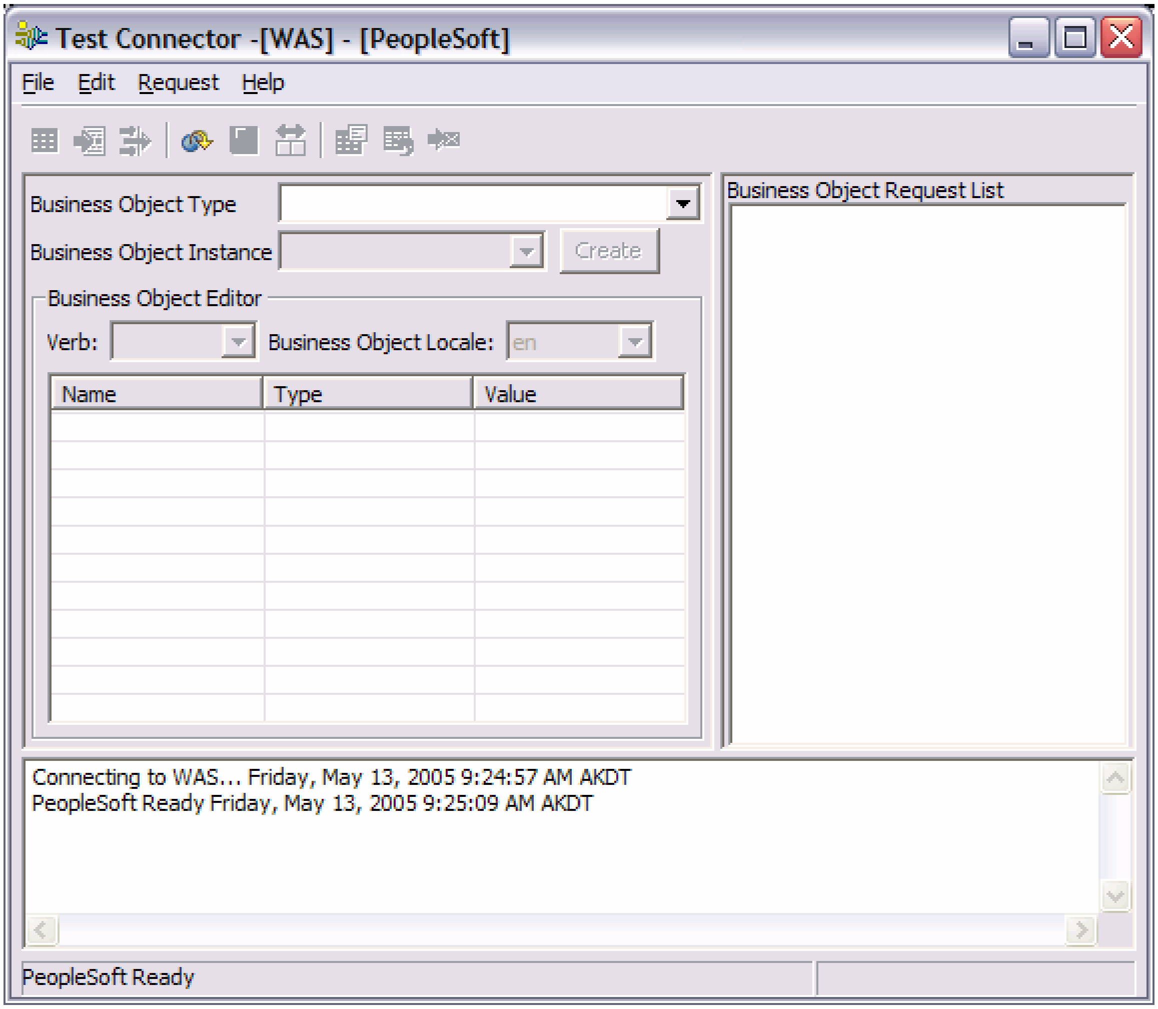


Connector	Server	User ID	Configuration File
PeopleSoft			C:\Labfiles60\eXchange\P...

__ e. Select **File > Connect**



__ f. View messages in the lower window.



Connecting to WAS... Friday, May 13, 2005 9:24:57 AM AKDT
PeopleSoft Ready Friday, May 13, 2005 9:25:09 AM AKDT

PeopleSoft Ready