# Flat Files JCA Adapter Lab



## What this exercise is about

The objective of this lab is to provide you with an understanding of the IBM WebSphere Adapter for Flat Files and how to deploy and integrate it with an SCA application for outbound processing.

## 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.
- WebSphere Adapter for Flat Files V6 installed.
- Optionally, review the Technotes located at http://www-306.ibm.com/software/integration/wbiadapters/support/ for the adapter.
- Interim Fix PK12908 for WebSphere Application Server 6.0.1.2 located at http://www-1.ibm.com/support/docview.wss?uid=swg24010665 and referred to by WebSphere Adapter Technote located at http://www-1.ibm.com/support/docview.wss?rs=695&context=SSMKUK&q1=waffv60rnote&uid=swg21217657&loc=en_US&cs=utf-8&lang=en

  Applying this interim fix resolves the exceptions described in this lab Part 3, step 3., i., in the **Note:** box. The exceptions are benign, without the interim fix; however applying the interim fix resolves the problem so the exceptions are no longer thrown.
- Sample code in the directory C:\Labfiles60 (Windows) or /tmp/LabFiles60 (Linux)

## What you should be able to do

Install and deploy the WebSphere Adapter for Flat Files integrated within an SCA application. Create applications processing outbound requests to the file system using the Flat Files adapter.

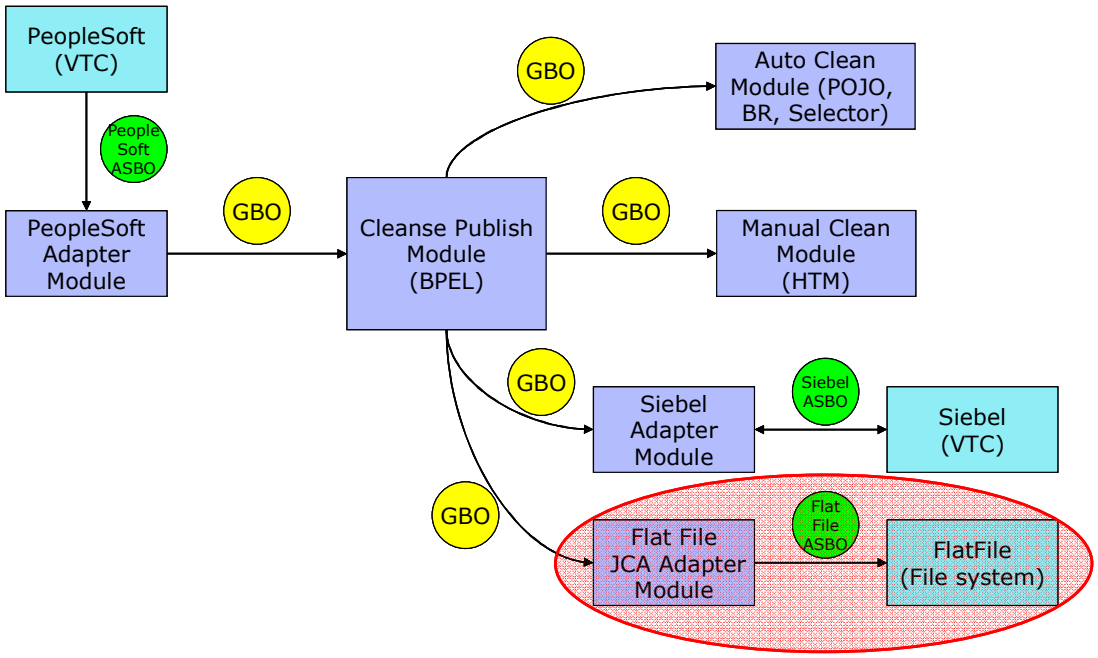
---

## Introduction

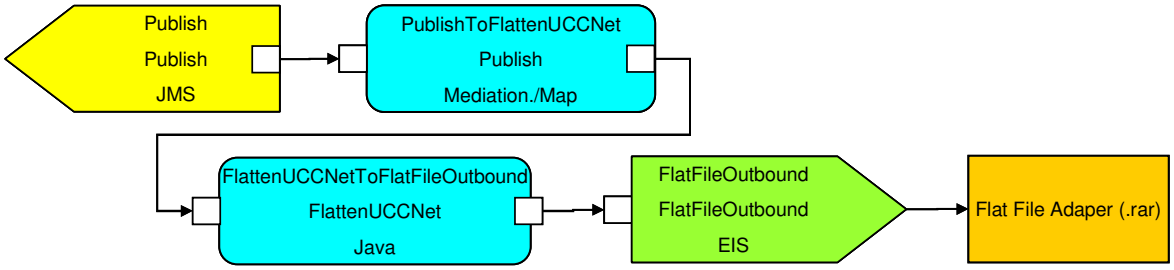
In this lab you will deploy the WebSphere Adapter for Flat Files, using WebSphere Integration Developer, and integrate it into an SCA application that processes outbound requests to the file system. As part of the application, you will create a Java component that uses the Publish Interface, and accepts the Business Objects ClipBG, Clip, and ClipItem, all from the eXchange Lab. The Java component implementation uses the BOXMLSerializer to serialize the objects into bytes which the adapter then sends out in a file to the file system. You will also create an SCA export that supports the Publish Interface, enabling this SCA application to be available to other SCA applications that may want to invoke it.

## Overview of Exercise

The following diagram highlights the parts of the overall scenario that will be addressed in this lab.



## Description of the 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 execute 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> | C:\<WID_HOME>\runtimes\bi_v6 | <WID_HOME>/runtimes/bi_v6 |
| <LAB_FILES> | C:\Labfiles60 | /tmp/Labfiles60 |
| <WORKSPACE> | C:\Labfiles60\eXchange\FlatFile\workspace | /tmp/Labfiles60/eXchange/FlatFile/workspace |
| <FFADAPTER_HOME> | C:\Program Files\IBM\ResourceAdapters\FlatFiles | /opt/IBM/ResourceAdapters/FlatFiles |
| <TEMP> | C:\temp | /tmp |

**Windows users**: When directory locations are passed as parameters to a Java program such as EJBdeploy or wsadmin, you must replace the backslashes with forward slashes to follow the Java convention. For example, C:\Labfiles60\ would be replaced by C:/Labfiles60/.

# Part 1: Initialize the Workspace for this Lab Exercise

____ 1. Follow the directions below to initialize the Workspace using the following values:

**<WORKSPACE>**

C:\Labfiles60\eXchange\FlatFile\workspace

**<PROJECT_INTERCHANGE>**

C:\Labfiles60\eXchange\CleansePublishLibrary\import\CleansePublishLibrary_PI.zip

**<MODULE>**

FlatFilesJCAAdapter

**<DEPENDENT_LIBRARIES>**

CleansePublishLibrary

---

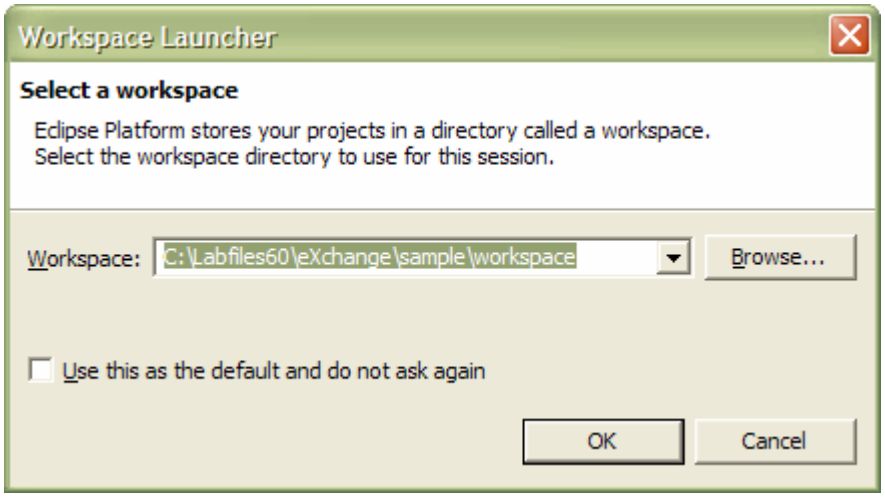
**Note:** A subdirectory named C:\Labfiles60\eXchange\FlatFile\outdir has been created for you. This is the specified subdirectory in which the application creates a file named *publish.txt* as part of the exercise. **If you have previously run any lab with the Flat File Adapter that created a file named *publish.txt* in the outdir subdirectory (for example, the "End to End Scenario" lab, delete that file(s) before continuing; otherwise, you will get an exception later in this lab when trying to create the *publish.txt,* that the file already exists.**

---

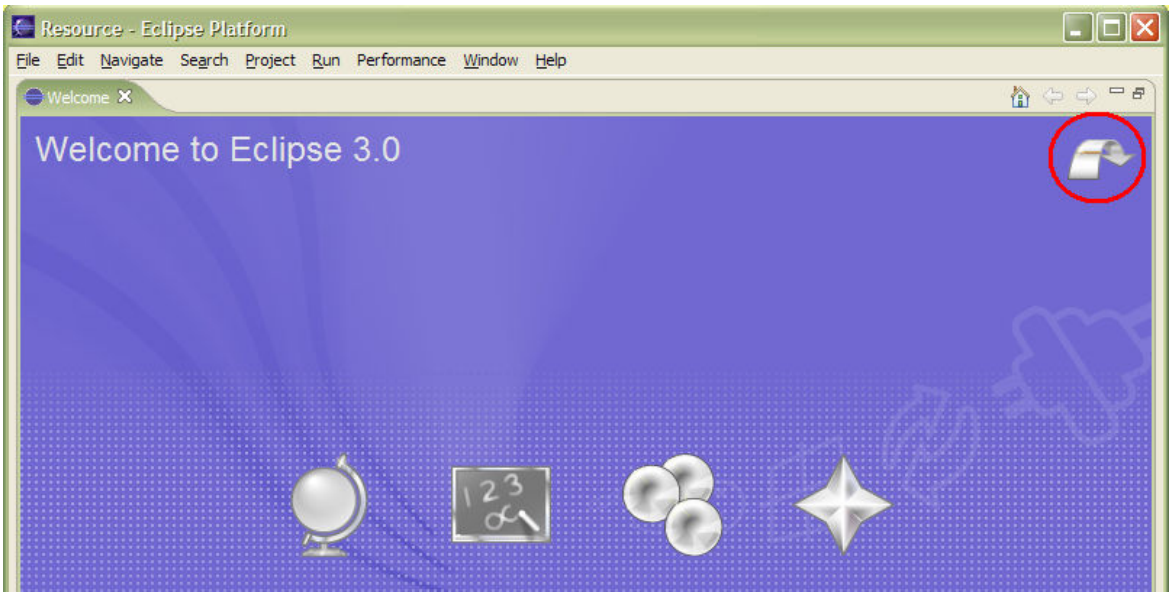
____ 2. Start WebSphere Integration Developer V6 with a new workspace located at **<WORKSPACE>**.

__ 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.**



__ 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).

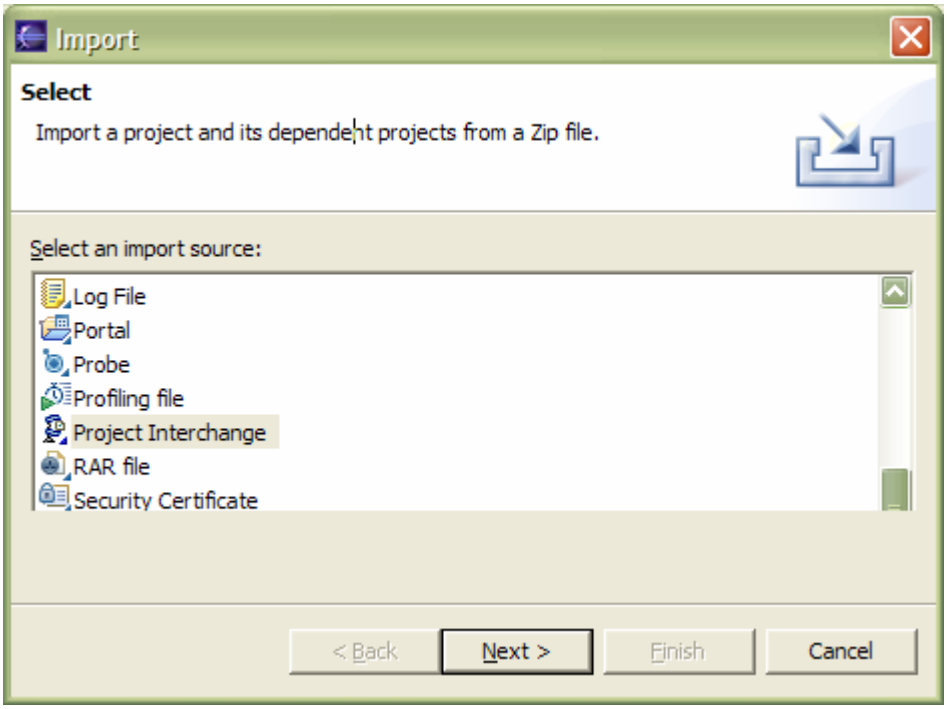



__ d. Ensure you are in the **Business Integration** perspective.

____ 3. 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. Select **File -> Import…** from the menu bar.

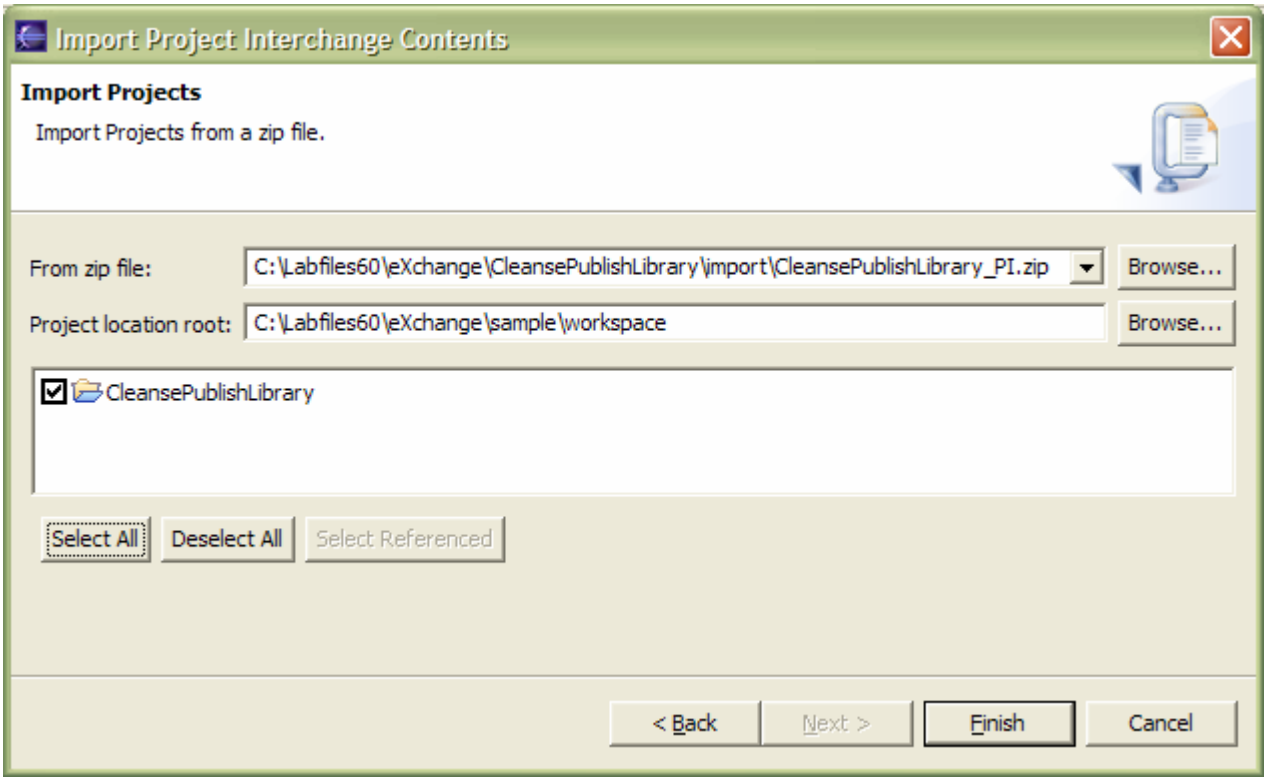
__ b. In the Import dialog, scroll down and select **Project Interchange.**




__ c. Click **Next.**

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

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

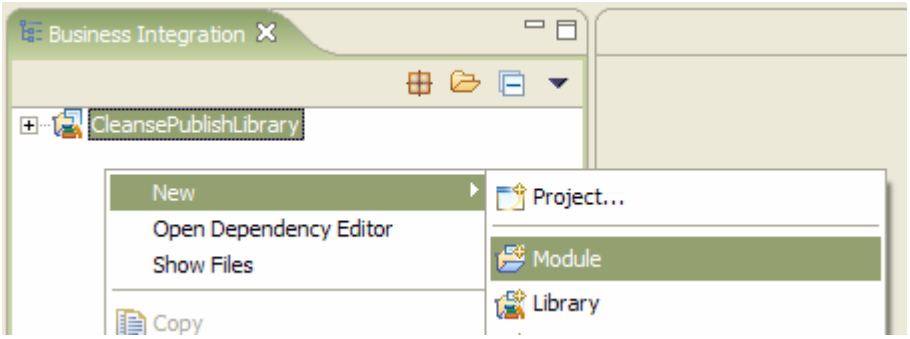



__ f. Click **Finish.**

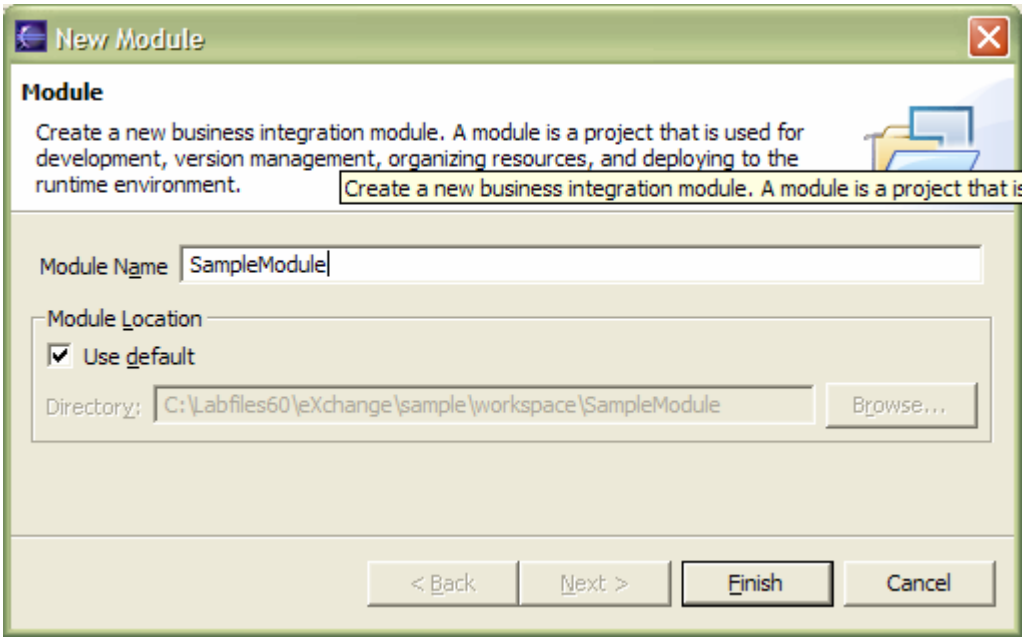
____ 4. If this lab requires that you create a Business Integration module called **<MODULE>**, complete these steps.

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

__ b. Select **New > Module.**



__ c. In the New Module dialog, enter **<MODULE>** for the Module Name.

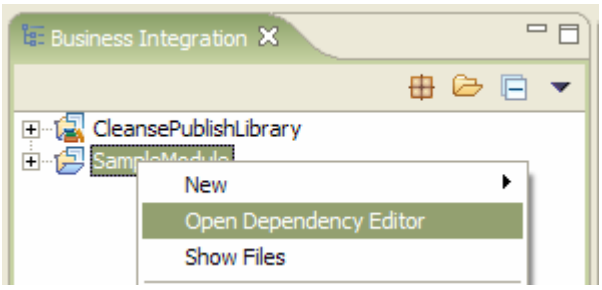



__ d. Click **Finish.**

____ 5. If this lab requires that **<MODULE>** needs any **<DEPENDENT_LIBRARIES>**, complete these steps.

__ a. In the Business Integration view, right click on the **<MODULE>** you just created to access the pop-up menu.

__ b. Select **Open Dependency Editor.**




__ c. Click the **Add…** button In the Dependency Editor.

__ d. In the Library Selection dialog, select from the list the **<DEPENDENT_LIBRARIES>.**

__ e. Click **OK.**

__ f. Press **Ctrl+S** to save the dependencies for this module.

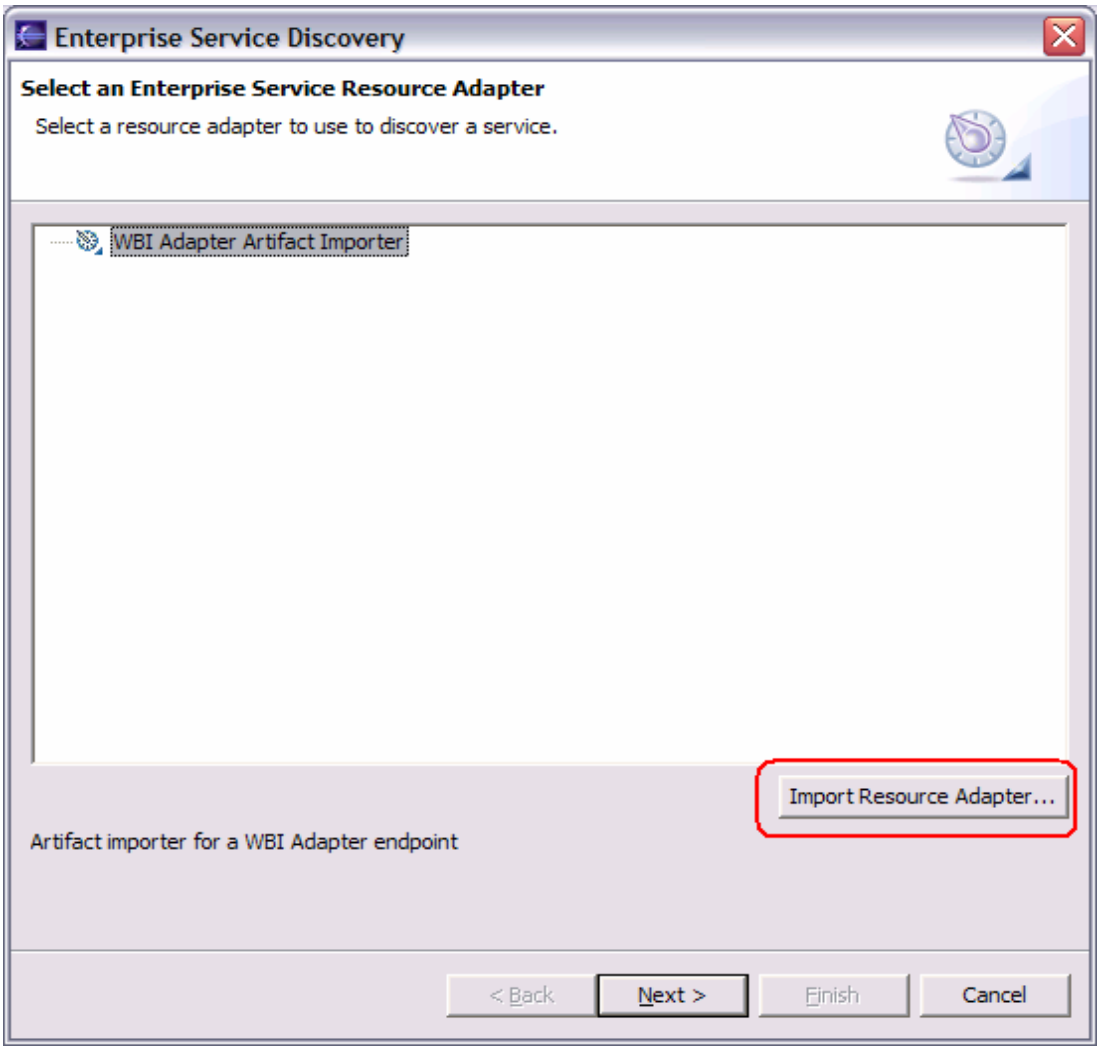
# Part 2: Create the FlatFilesJCAAdapter Application

In this part you will run Enterprise Service Discovery to import the WebSphere Adapter for Flat Files, input the property values to build the Service Description (used to create the Managed Connection Factory), create the necessary SCA artifacts, and assemble them into an SCA application. Since the Flat Files Adapter has a single, pre-defined business object structure, there is no Enterprise Service Discovery for "object discovery" of business objects.

____ 1. Start the Enterprise Service Discovery process and import the Flat Files Adapter Resource Archive (RAR) file. This will create a J2EE Connector Project.

__ a. From the top Menu bar, select **File > New > Enterprise Service Discovery**.

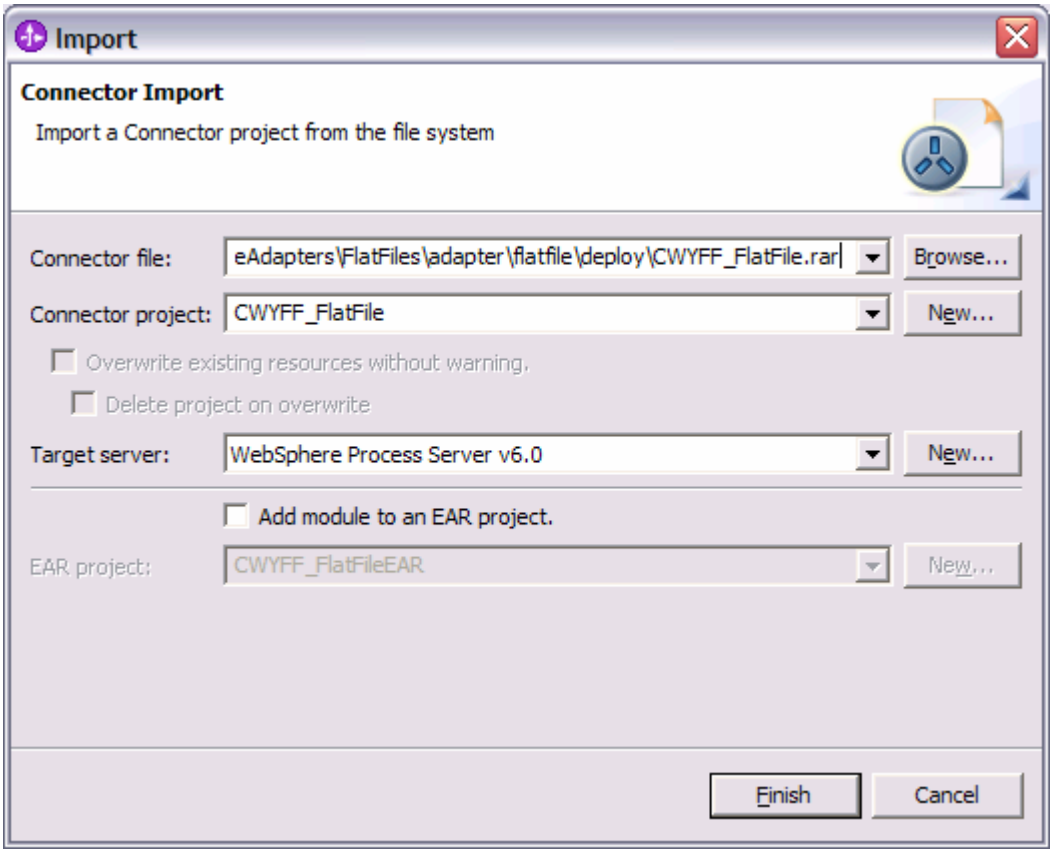
__ b. Click **Import Resource Adapter…** at the bottom right of the Select an Enterprise Service Resource Adapter panel.



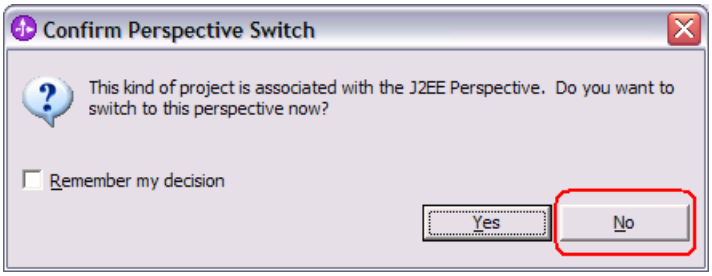

__ c. Complete the Connector Import panel.

1) Connector file: **Browse…** to the location of the CWYFF_FlatFile.rar adapter file and select it, for example: <FFADAPTER_HOME>\adapter\flatfile\deploy\CWYFF_FlatFile.rar.

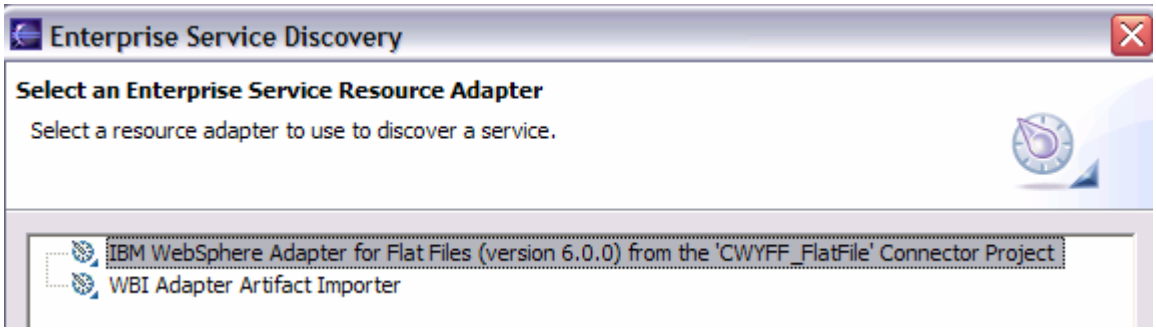
**2)** Deselect the "**Add module to an EAR project**" if it is selected, and click **Finish.**



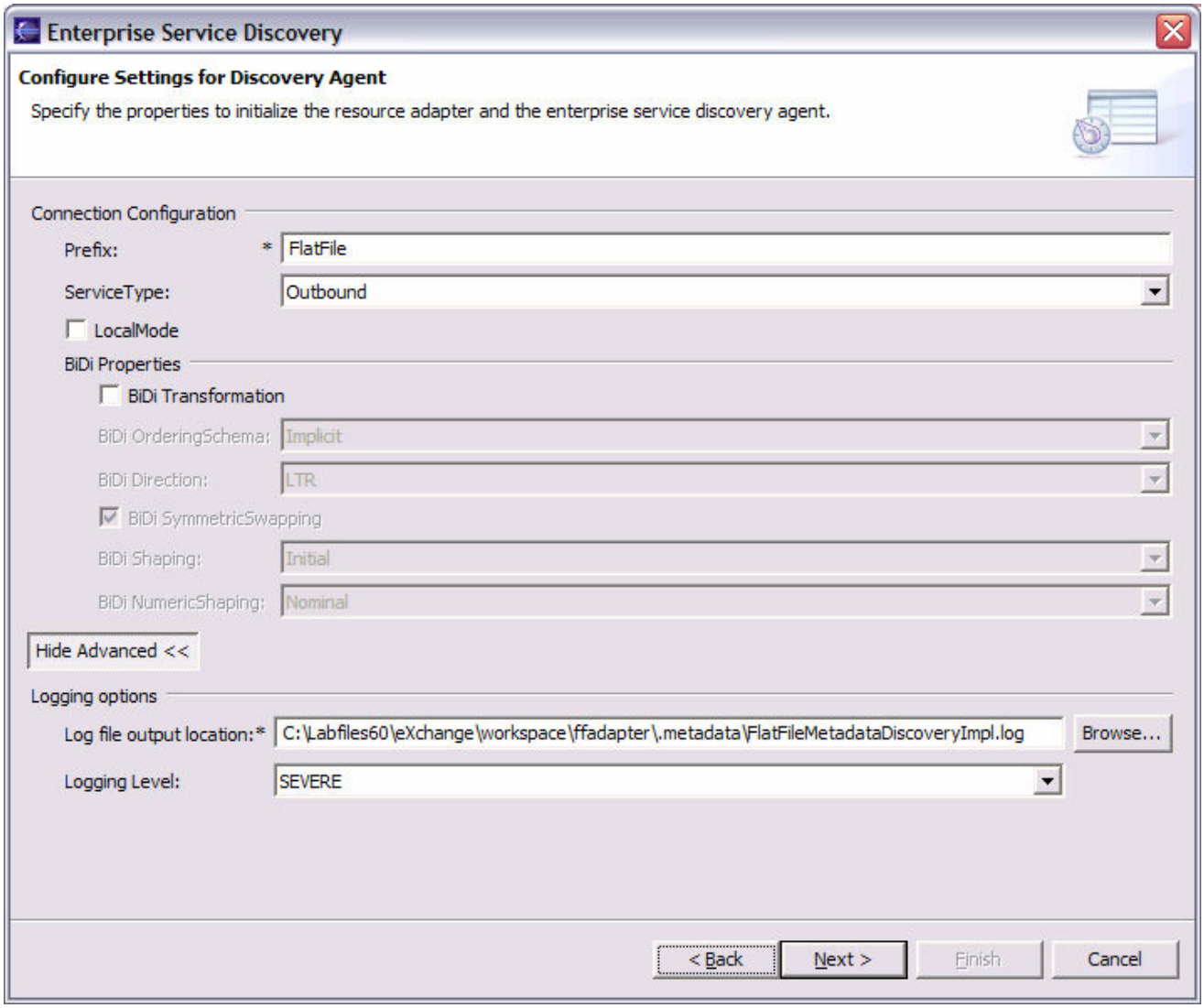

__ d. At the Confirm Perspective Switch prompt, select **No.**



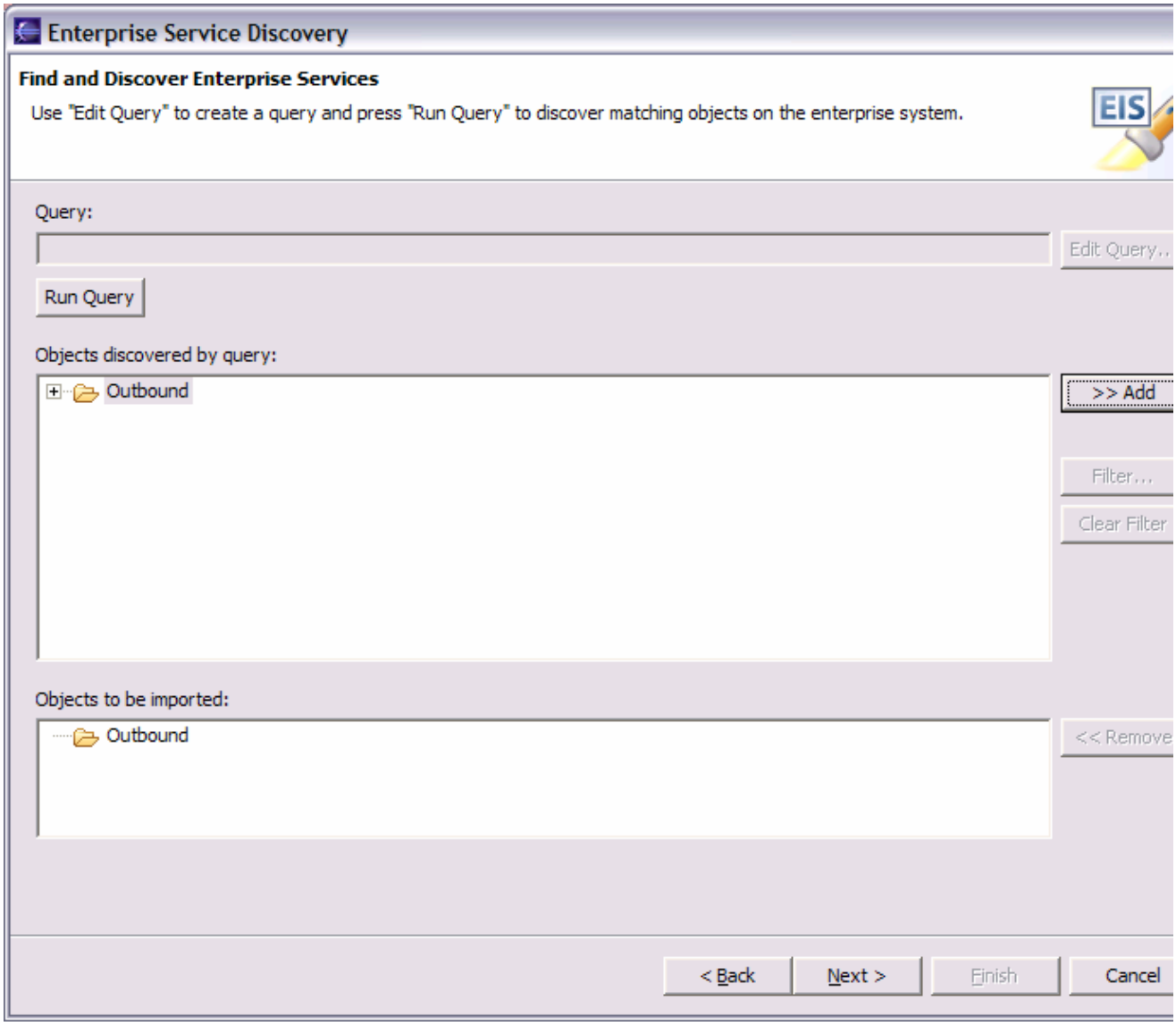

__ e. You will return to the Select an Enterprise Service Resource Adapter panel. Highlight the **IBM WebSphere Adapter for Flat Files** and click **Next.**



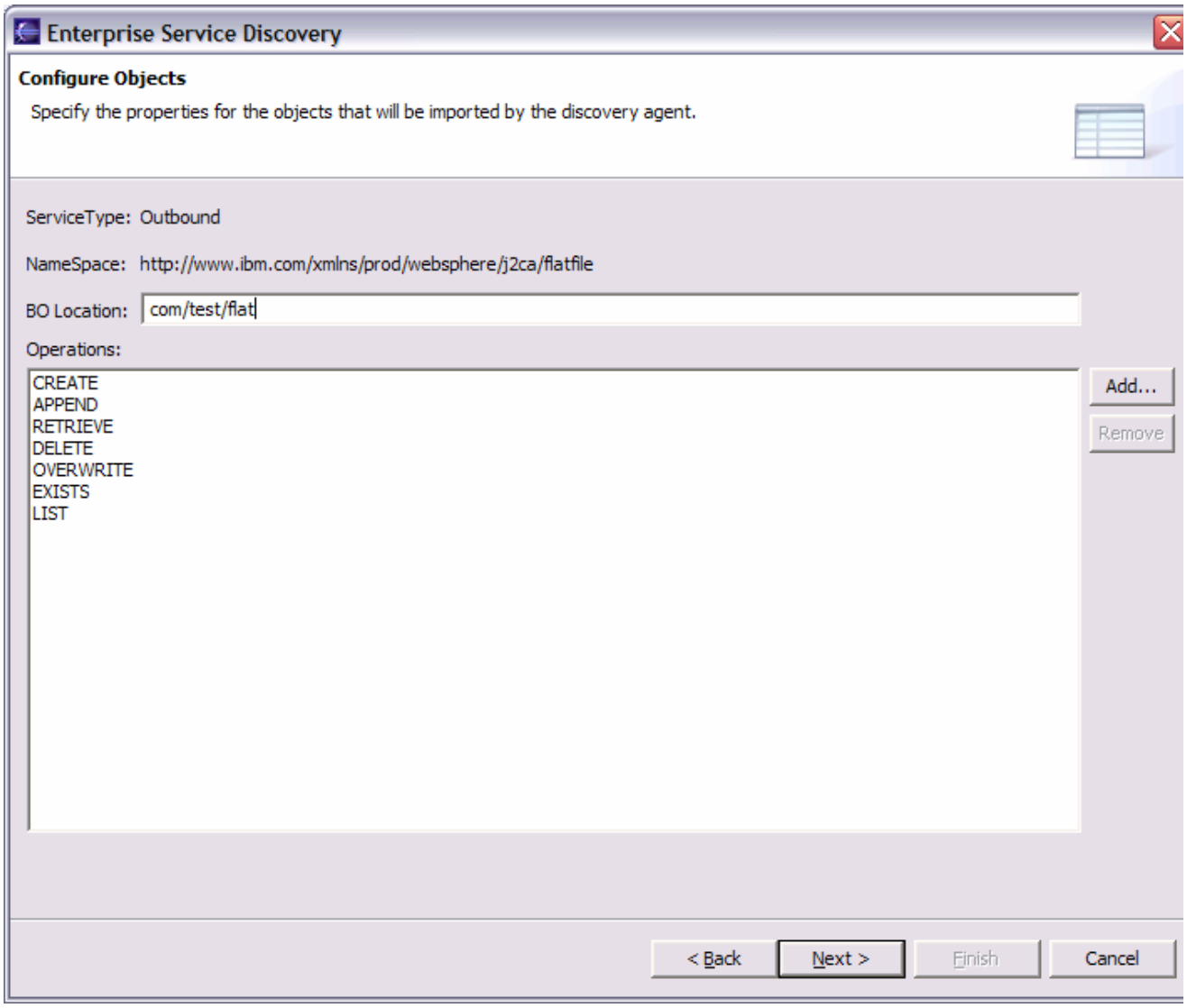
__ f. From the Configure Settings for Discovery Agent panel, ensure the **ServiceType:** is set to **Outbound**. Click the **Show Advanced >>** button to note the Log file output location and Logging Level options for discovery log, leave the default values, and then click **Next.**



__ g. From the Find and Discover Enterprise Services panel, click the **Run Query** button. An **Outbound** folder will appear in the Objects discovered by query: window. **Select** the **Outbound Folder, click** the **>> Add** button. The **Outbound** folder is now displayed in the Objects to be imported: window. Click **Next.**

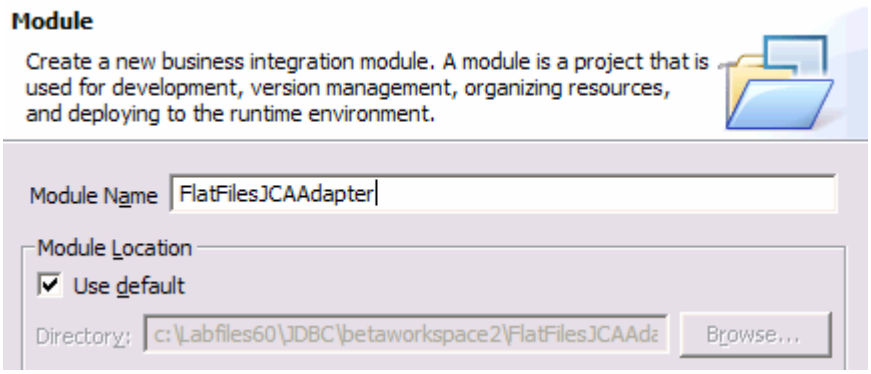


__ h. On the Configure Objects panel, **enter** a BO Location: of **com/test/flat**. Note the Operations available for the ServiceType: Outbound and click **Next.**



__ i. On the Generate Artifacts panel, specify the properties for the artifacts that will be generated in the workspace. Leave the default values except as follows:

1) Create a New Module named **FlatFilesJCAAdapter. (Note:** Be sure to type the case as shown. Later you will copy and paste sample Java code that refers to this name, and it is case sensitive.)



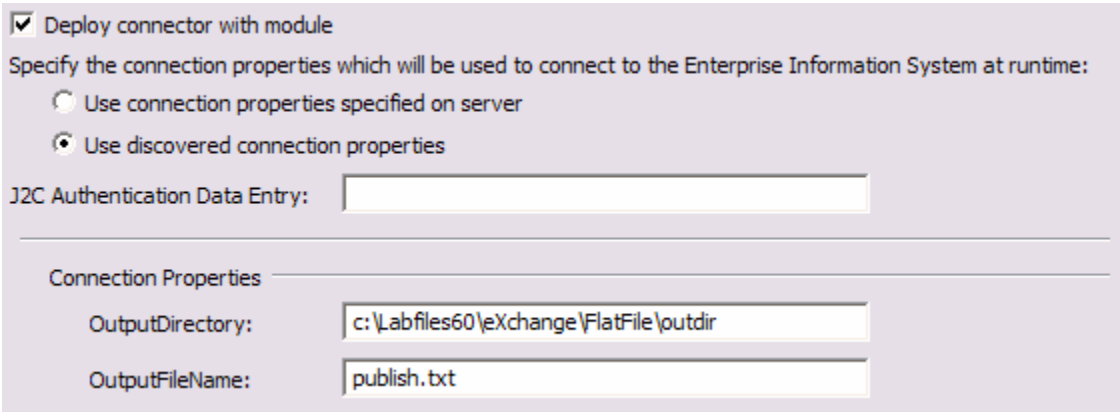

2) **Folder:** Enter **com/test/flat.**

3) **Select** the radio button to the left of **Use discovered connection properties**. Additional Connection properties and Resource Adapter properties will become visible.

4) **OutputDirectory:** Enter **C:\Labfiles60\eXchange\FlatFile\outdir.**

a) Ensure the **C:\Labfiles60\eXchange\FlatFile\outdir** exists. If the output directory does not exist, the test will fail. (For this lab, use this specific subdirectory name, however, in your own installations, this value can be whatever you want.)

5) **OutputFileName:** Enter **publish.txt.**



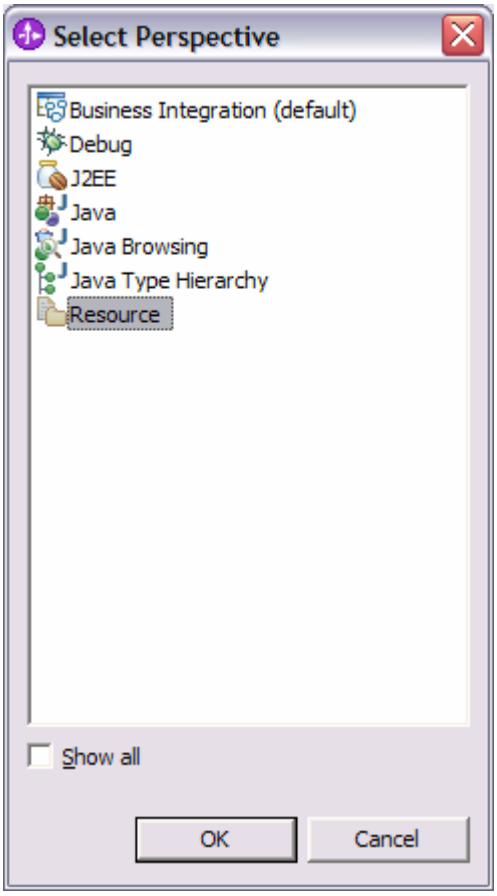

__ j. Click **Finish.**

---

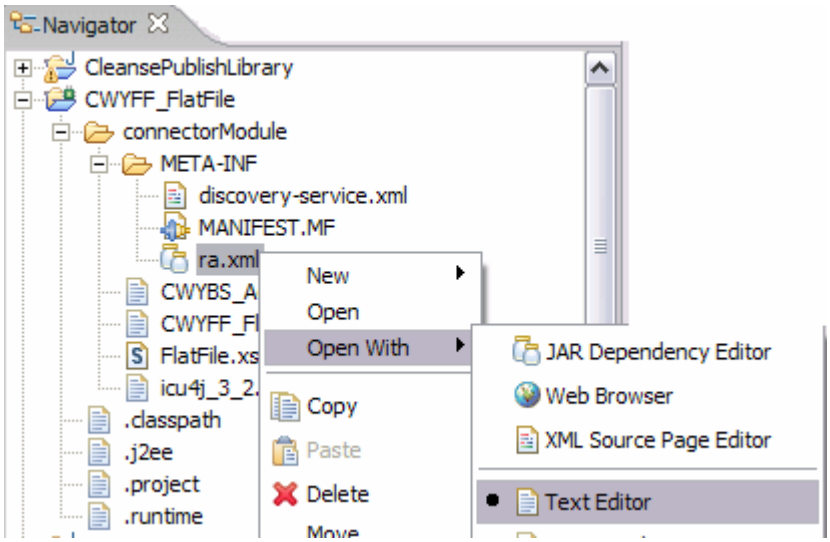
**Note:** The IBM WebSphere Adapters (this includes the IBM WebSphere Adapter for Flat Files, IBM WebSphere Adapter for JDBC, IBM WebSphere Adapter for PeopleSoft Enterprise, IBM WebSphere Adapter for Siebel Business Applications, and IBM WebSphere Adapter for SAP Applications) are supported as "Deploy connector with module" only, meaning the adapter is deployed within the module which is packaged as an Enterprise Archive file (EAR file). Therefore, the "Deploy connector with module" checkbox should always be checked and the "Use discovered connection properties" checkbox should always be checked.

---

____ 2. You will return to the Business Integration view. While not necessary, if you would like to view the Connector Project, examine the RAR and the deployment descriptor ra.xml file, switch to the Resource view. From the top level menu select Window **> Open Perspective > Other**, then select **Resource.** Click **OK.**



__ a. Expand the **CWYFF_FlatFile** folders. Under the META-INF folder, **Right click** the **ra.xml** file, and **Open with > Text Editor.**

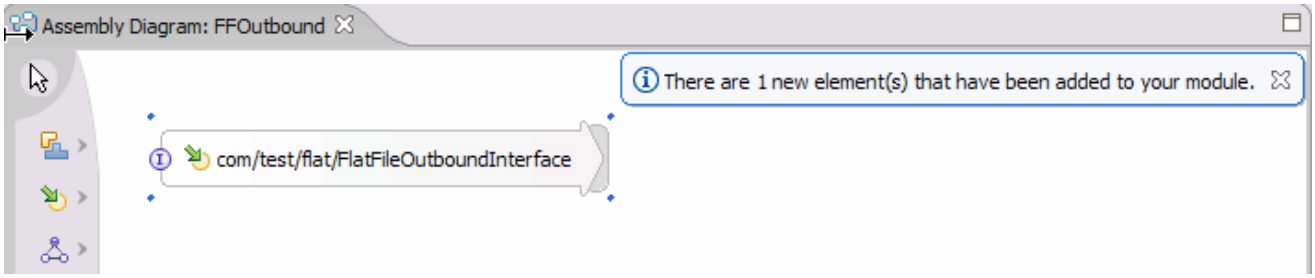



__ b. Note the various properties and their values. You will see that the inbound properties are not set, they are not needed as this is a ServiceType of Outbound. Be careful to not change anything. Close the ra.xml file, and return to the Business Integration view. Select **Window > Open Perspective > Other > Business Integration** and click **OK.**

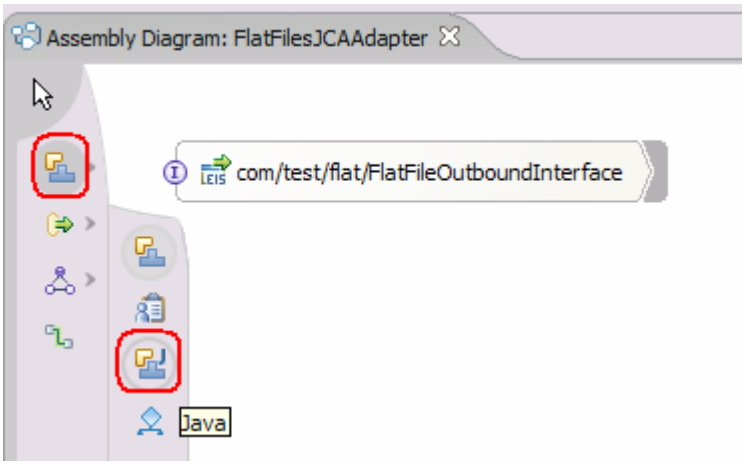
____ 3. Use the Assembly Diagram to wire a Java Component to the **FlatFileOutboundInterface.**

__ a. Expand the **FlatFilesJCAAdapter** folder from the **Business Integration** perspective.

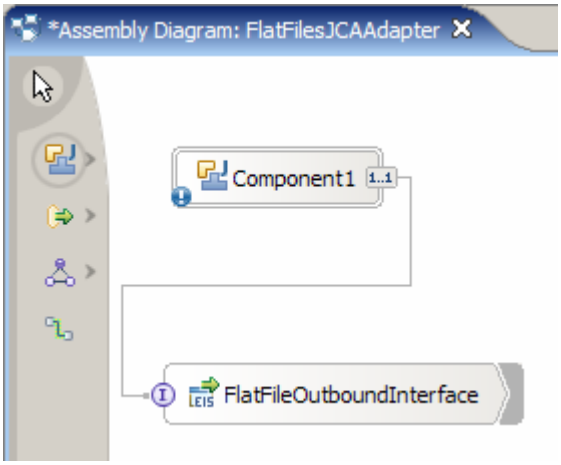
__ b. Double-click the **FlatFilesJCAAdapter** module. This will open the module in the Assembly Diagram. You will see a message that there is one new element added to the module.



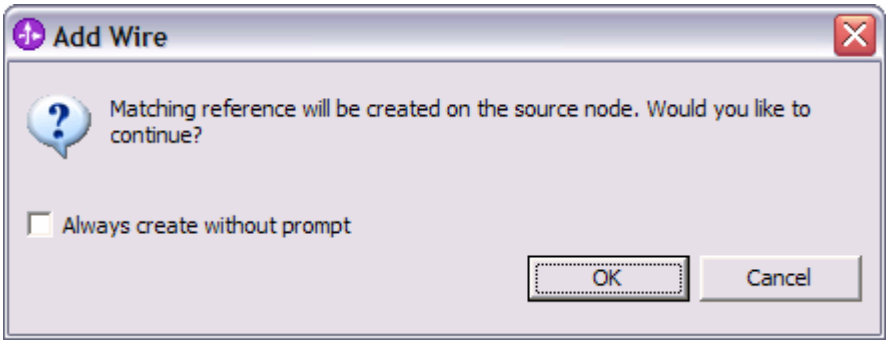

__ c. From the palette, select the **Component (with no implementation type)** icon, then select the **Java Component** and click in the Assembly Diagram to drop the component on to the Assembly Diagram editor.



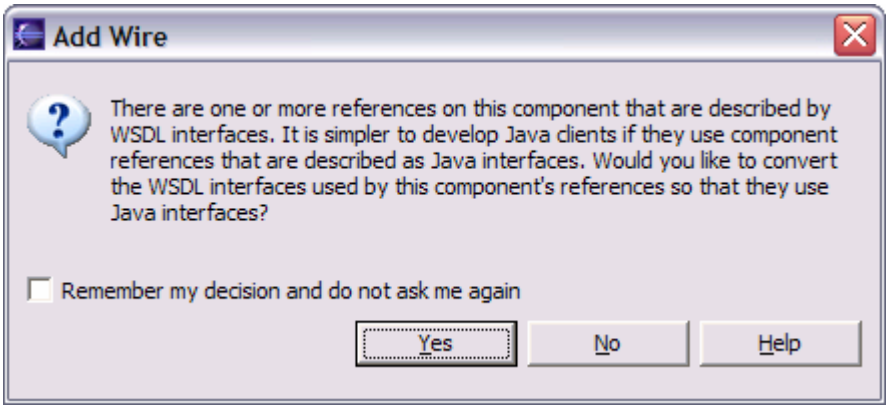

__ d. Wire the **Java Component** to the **com/test/flat/FlatFileOutboundInterface.**



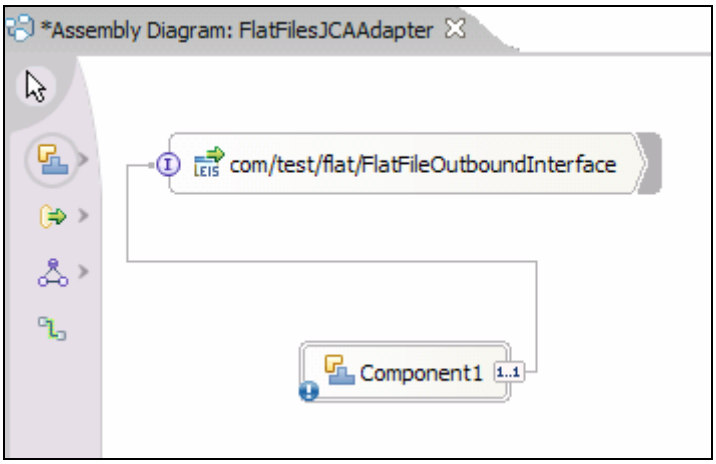

__ e. Click **OK** on the Add Wire popup window.



__ f. If another Add Wire popup window appears as the following, Select **No.**

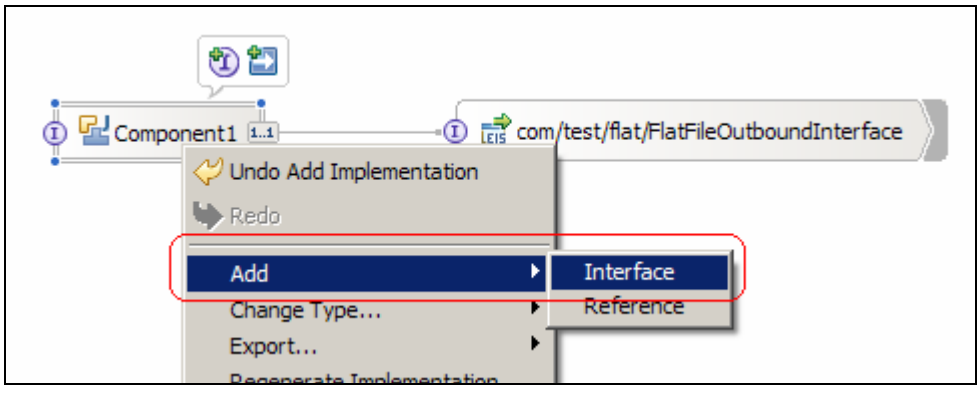



__ g. The Assembly Diagram should now look as follows.

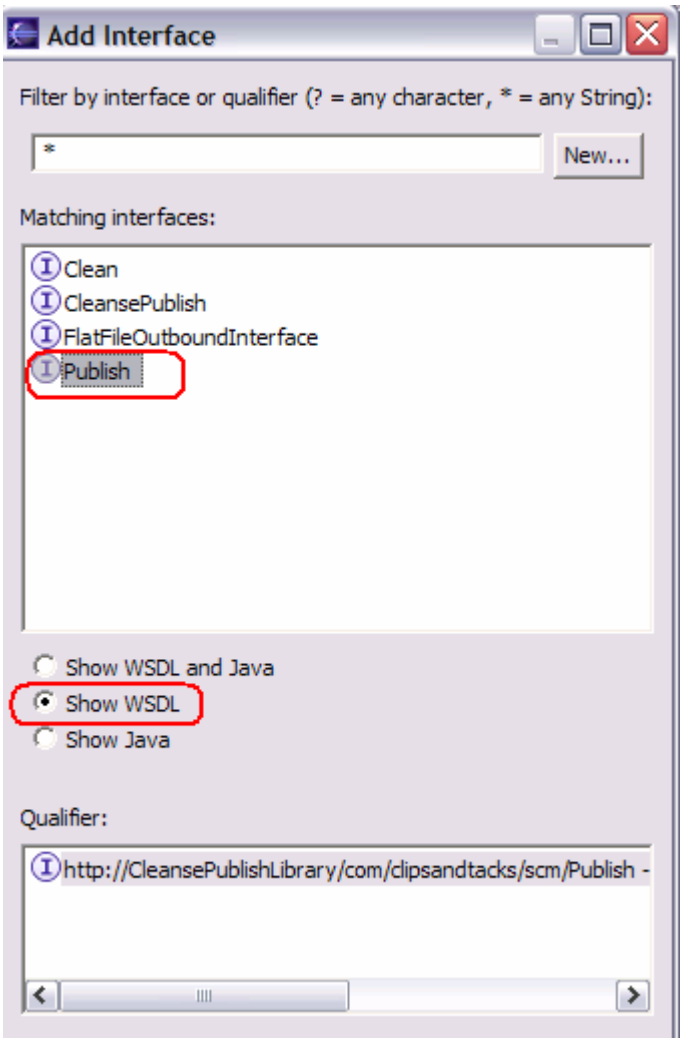



____ 4. Add an Interface to the Java Component1.

__ a. Right click the **Java Component1**, select **Add Interface** from the context menu.

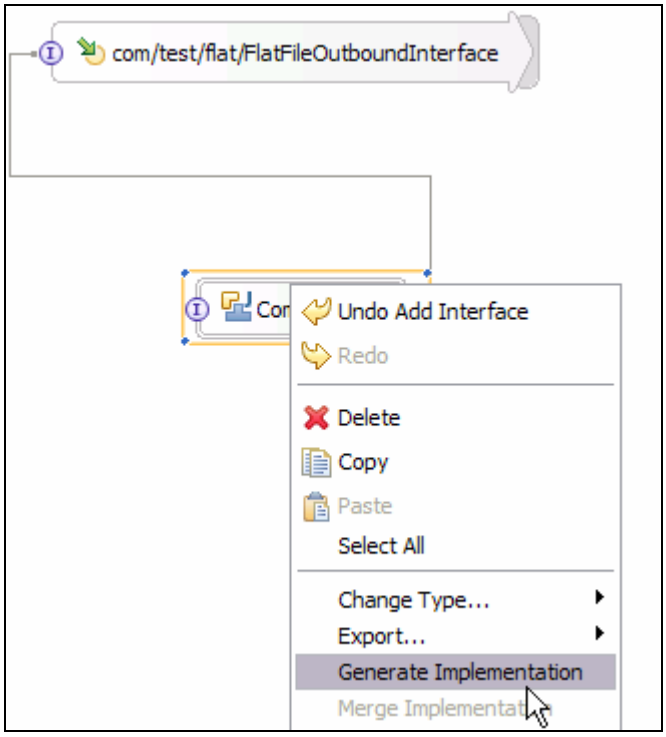


__ b. In the **Add Interface** panel, select the radio button **Show WSDL** in the middle of the panel, to limit the list, then **double click** the **Publish** interface.

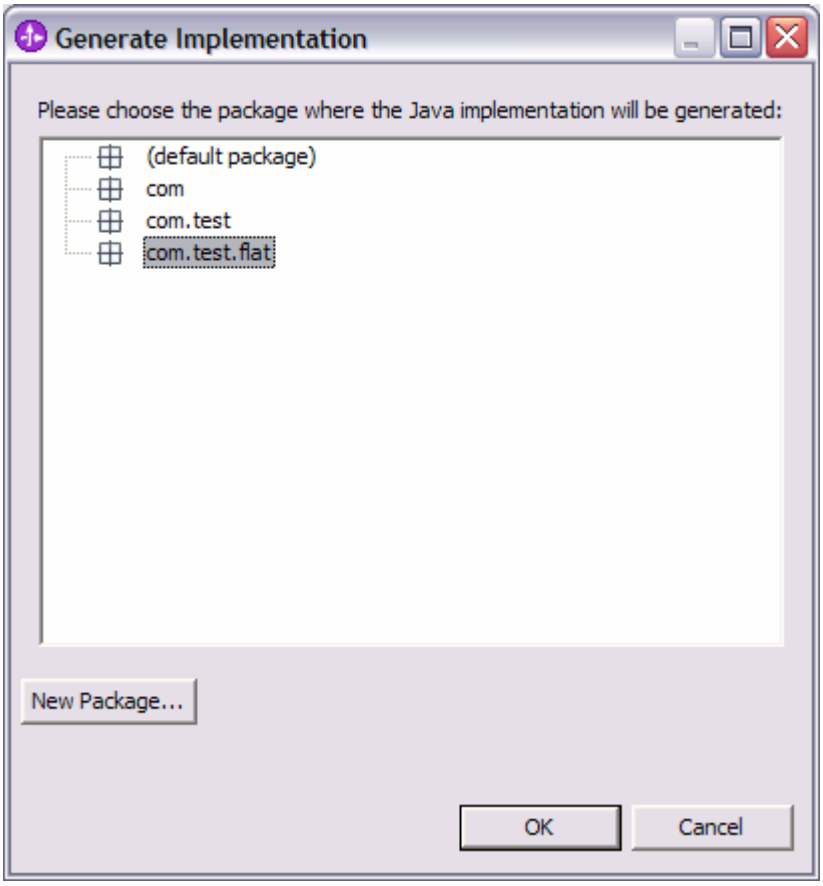


____ 5. Generate a Java Implementation for the **Component1Impl.java.**

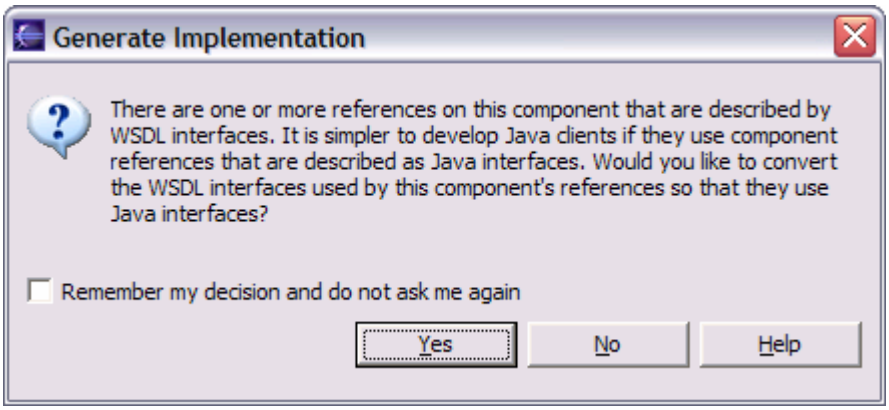
__ a. Right click the **Component1** component, select **Generate Implementation.**



__ b. Highlight the **com.test.flat** package. Click **OK.**




____ 6. Click **No** on the Generate Implementation popup window.




__ a. The Java Editor will open with the **Component1Impl.java** file.

---

**NOTE:** For your convenience, the following code snippets can be found in **<LAB_FILES>\eXchange\FlatFile\snippets\component1impl.txt.**

---

__ b. Add the following lines of imports directly below the existing set of import statements at the top of the file.

```
import java.io.ByteArrayOutputStream;
import java.io.ObjectOutputStream;
import com.ibm.websphere.bo.BOFactory;
import com.ibm.websphere.bo.BOXMLSerializer;
```

__ c. Scroll down to the method **public void publishClip(Dataobject inClipBG)** that needs to be implemented and enter the following code. (The code continues on the following page.)

---

**Note** these two specific set methods in the following code. These values will overwrite any previous values entered during Enterprise Service Discovery. These dataObject.set methods are not required, as the values entered during Enterprise Service Discovery would suffice. This is just an additional example showing how to set these values in code.

dataObject.set("directoryPath", "C:\\Labfiles60\\eXchange\\FlatFile\\outdir");
dataObject.set("fileName", "publish.txt");

---

```
try{
        ByteArrayOutputStream bos=new ByteArrayOutputStream();
        //ObjectOutputStream oos=new ObjectOutputStream(bos);

        BOXMLSerializer
xmlser=(BOXMLSerializer)ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/B
OXMLSerializer");


  xmlser.writeDataObject(inClipBG.getDataObject("Clip"),"http://CleansePublishLibrary/com/clip
sandtacks/scm","Clip",bos);

        //invoke the FlatFilesJCAAdapter outbound test
        //
        byte [] bytearray=bos.toByteArray();


    Service service =
(Service)ServiceManager.INSTANCE.locateService("FlatFileOutboundInterfacePartner");
    BOFactory boFactory
=(BOFactory)ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOFactory");

//  Create the wrapper element
    DataObject element=boFactory.createByElement
          ("http://FlatFilesJCAAdapter/com/test/flat/FlatFileOutboundInterface" ,"create");
    DataObject dataObjectx=element.createDataObject("createInput");
    DataObject dataObject=dataObjectx.createDataObject("FlatFile");

        dataObject.set("directoryPath", "C:\\Labfiles60\\eXchange\\FlatFile\\outdir");
        dataObject.set("fileName", "publish.txt");
        dataObject.set("inputBytes", bytearray);


        DataObject do2=(DataObject)service.invoke("create",element);

    DataObject outputDataObject = do2.getDataObject("createOutput");//BG
```

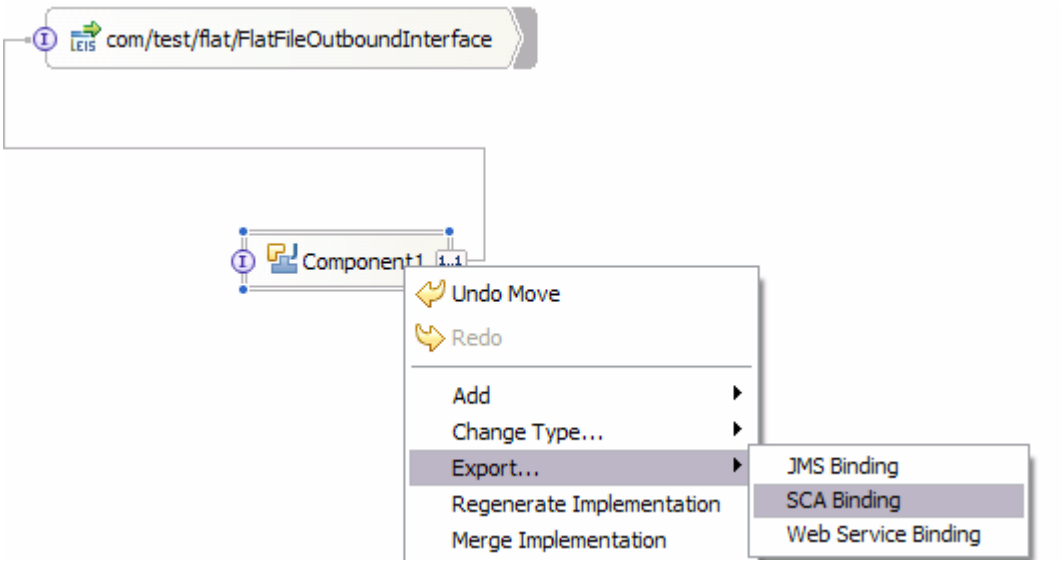
```
System.out.println(outputDataObject.getDataObject("FlatFile").getString("directoryPath"));
System.out.println(outputDataObject.getDataObject("FlatFile").getString("fileName"));


}
catch(Throwable t){
t.printStackTrace();
System.out.println("IT FAILED WITH:  "+t.toString());
}
```

__ d. Make sure there are no “red x” compile errors.  Close the Java file and save the changes.

____ 7. Create an SCA export that supports the Publish interface to make this module available to be invoked by other SCA modules.

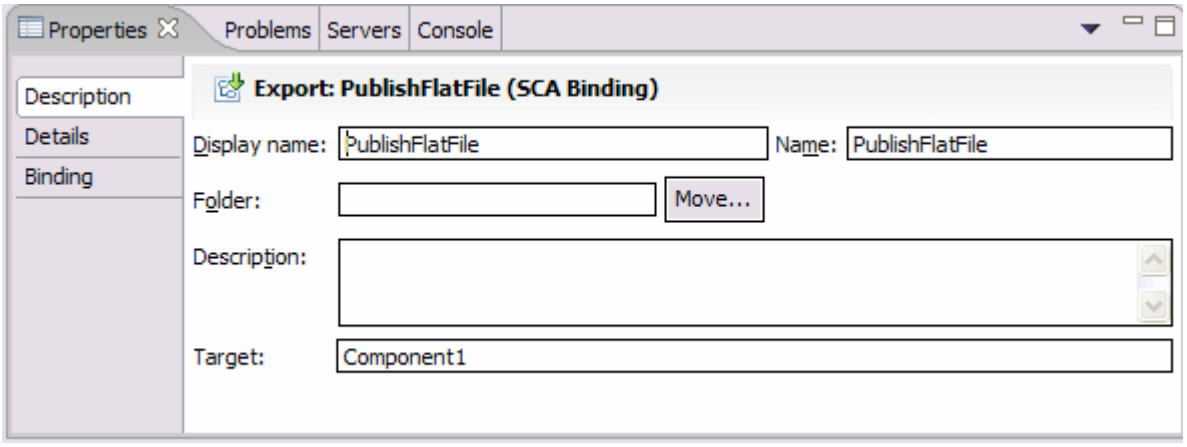
__ a. Right click the Component1 Java component, select **Export** > **SCA Binding.**



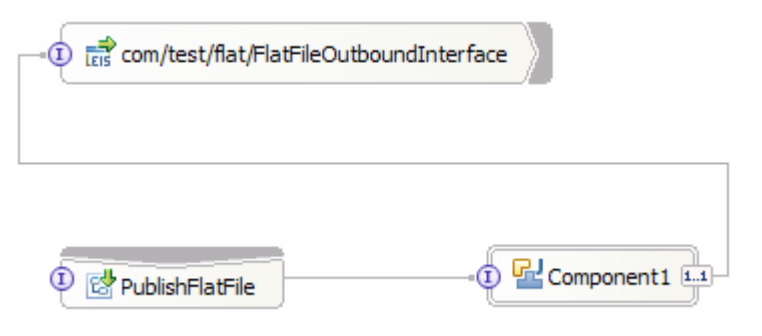

____ 8. Change the name and the display name of the newly created **Component1Export** to **PublishFlatFile**.

__ a. Single click the Component1Export to select it.

__ b. From the Properties pane below the Assembly Diagram, on the Description tab, change the **Display name**: and **Name**: values to **PublishFlatFile.**

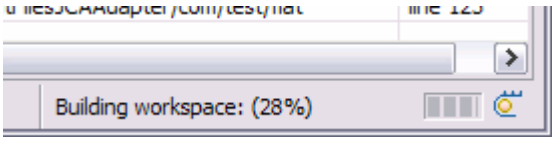



__ c. Your module should now look as follows:




____ 9. Save and close the Assembly Diagram.

__ a. Close the Assembly Diagram and click Yes at the **Save** Resource popup window to save the changes. In the bottom right corner of the main window, you will see **Building workspace:** and some percentage. Wait for the workspace to complete building.



# Part 3: Test the application using the WebSphere Test Environment (WTE)

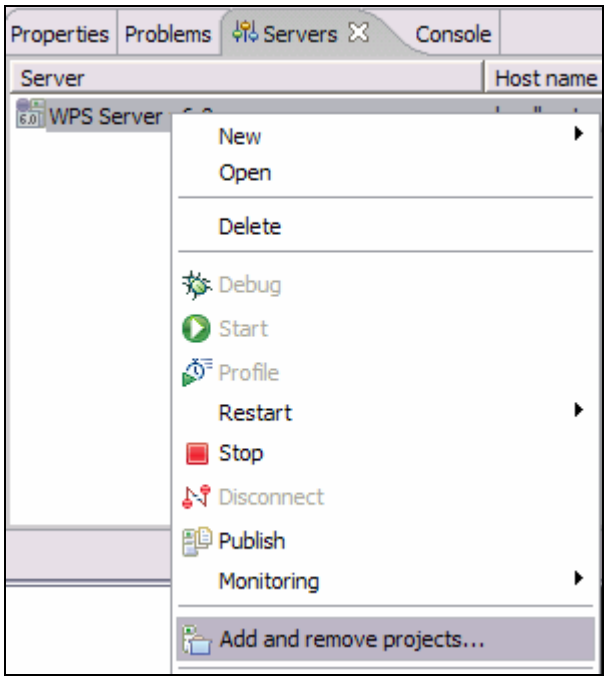
In this part you will use the WebSphere Test Environment to test the SCA application outbound processing.

____ 1. Start the WebSphere Process Server Test environment.

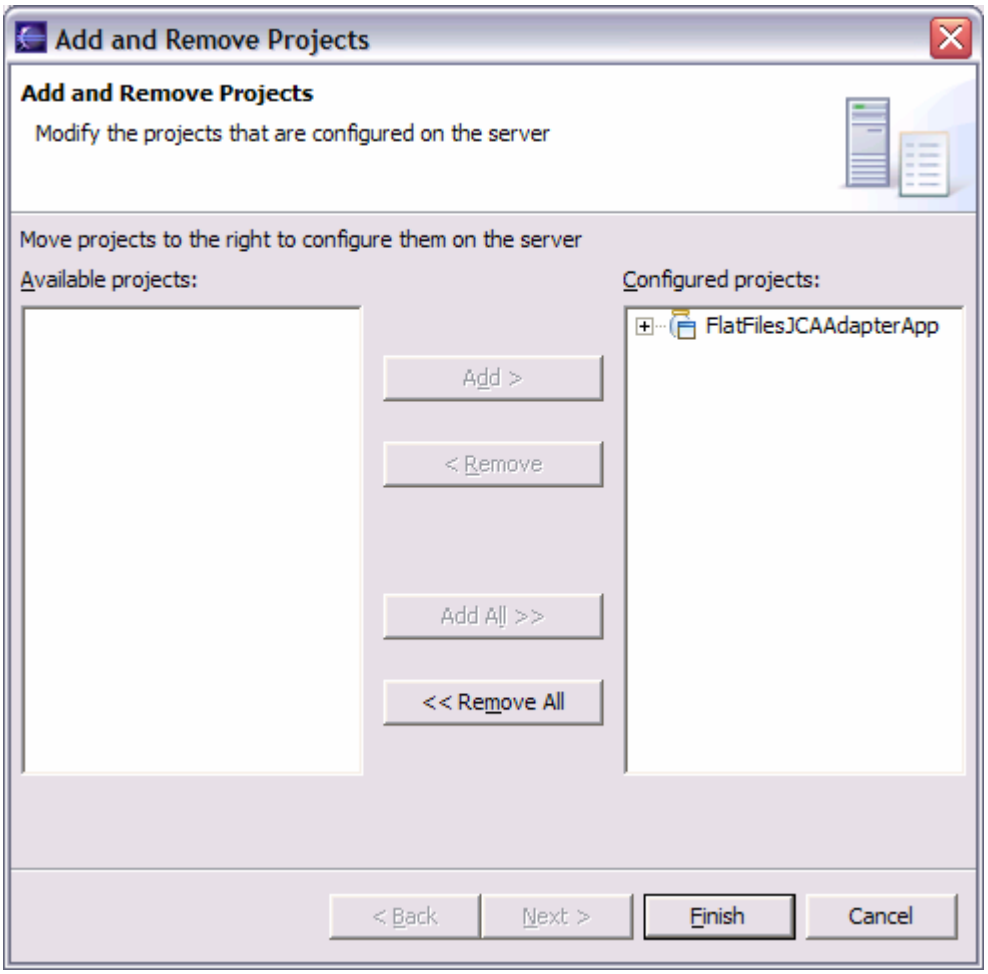
__ a. Right click on the server in the **Servers view** and select **Start.** Wait for the server to be started successfully, as indicated by the Server status of “Started”. This could take a few minutes.

____ 2. Add the project to the server for the WebSphere Test Environment.

__ a. Right click on the server in the server view and select **Add and remove projects** …



__ b. In the Add and Remove Projects dialog, select the **FlatFilesJCAAdapterApp** project from the Available projects panel. Click **Add >** to add it to the Configured projects panel. Click **Finish.**

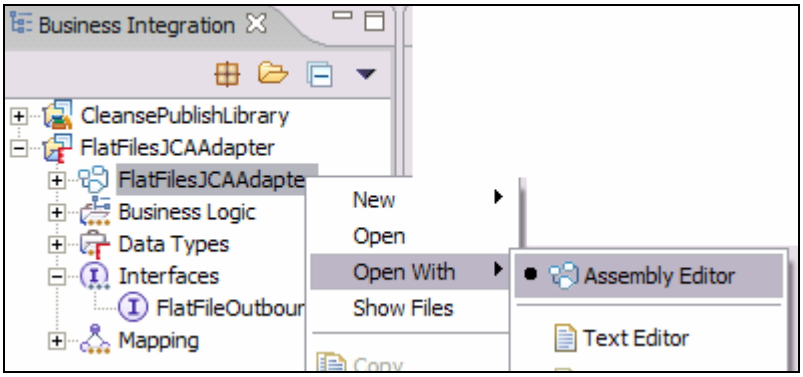



__ c. Wait for the project to be added to the server and the application to start in the server. View the console log from the console view for the application start logs.

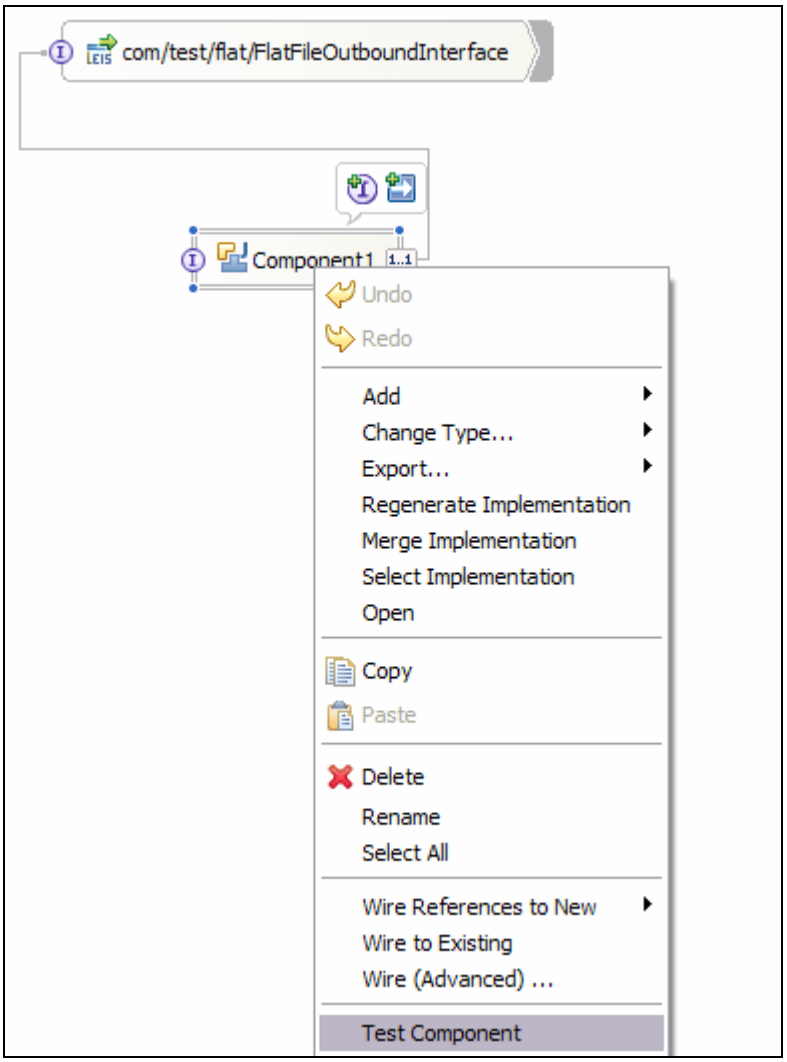
```
WSVR0221I: Application started: FlatFilesJCAAdapterApp
```

____ 3. Use Component Test to test the application.

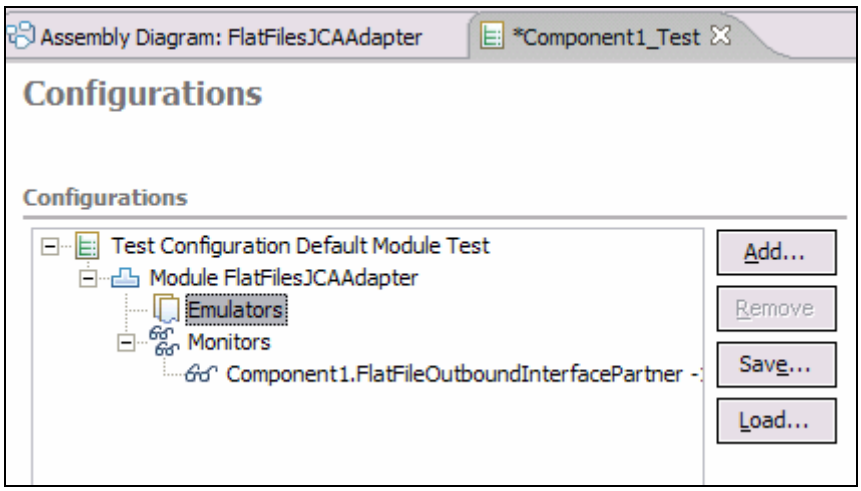
__ a. Right click the **FlatFilesJCAAdapter** module and open in the Assembly Editor.



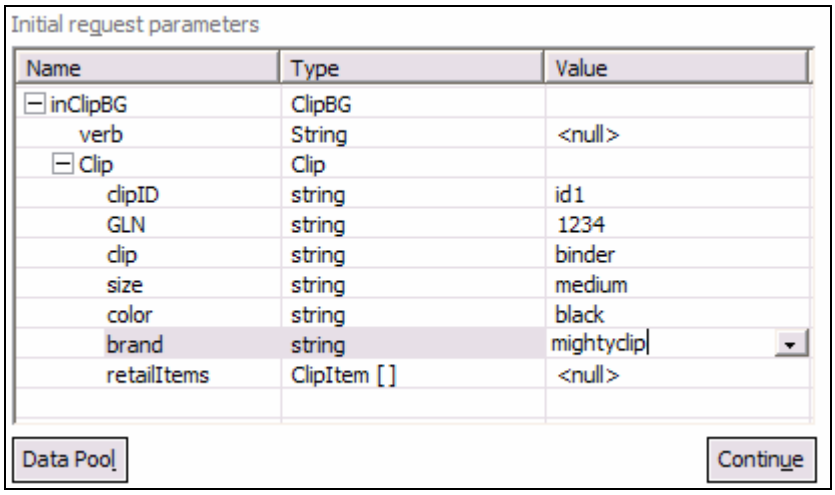
__ b. Right click the **Component1** component and select **Test Component.**



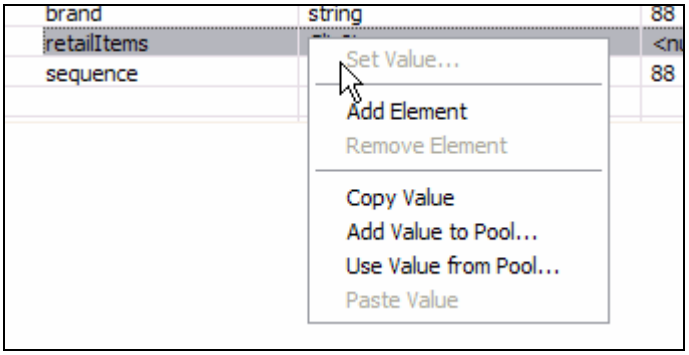
__ c. Switch from the Events tab to the **Configurations** tab, right click **FlatFileOutboundInterface** under Emulators and click **Remove**. Rather than having Test Component "emulate", we want to use the Java Component implementation code previously entered. There should now be no Emulators, as shown.



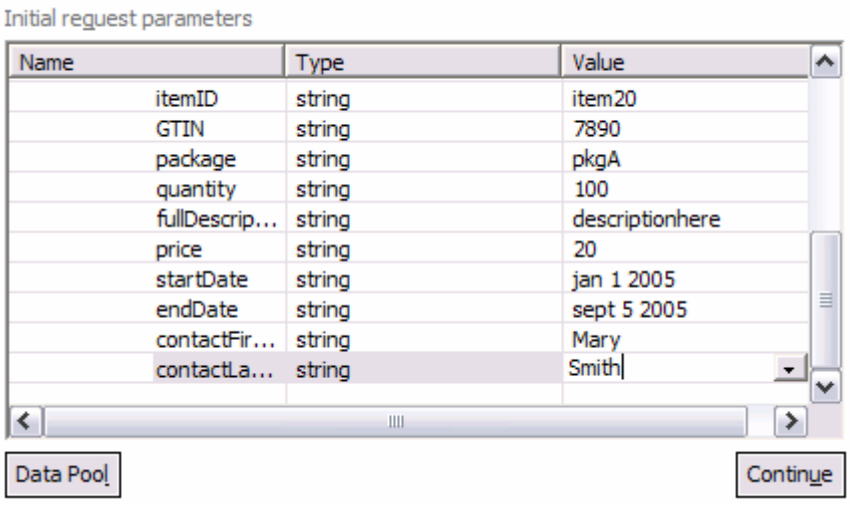

__ d. Switch back to the **Events** tab, enter whatever values you'd like for the fields of type string, underneath the Clip business object (clipID, GLIN, clip, size, color, brand). For example:




__ e. Right click **retailItems** and select **Add Element.**

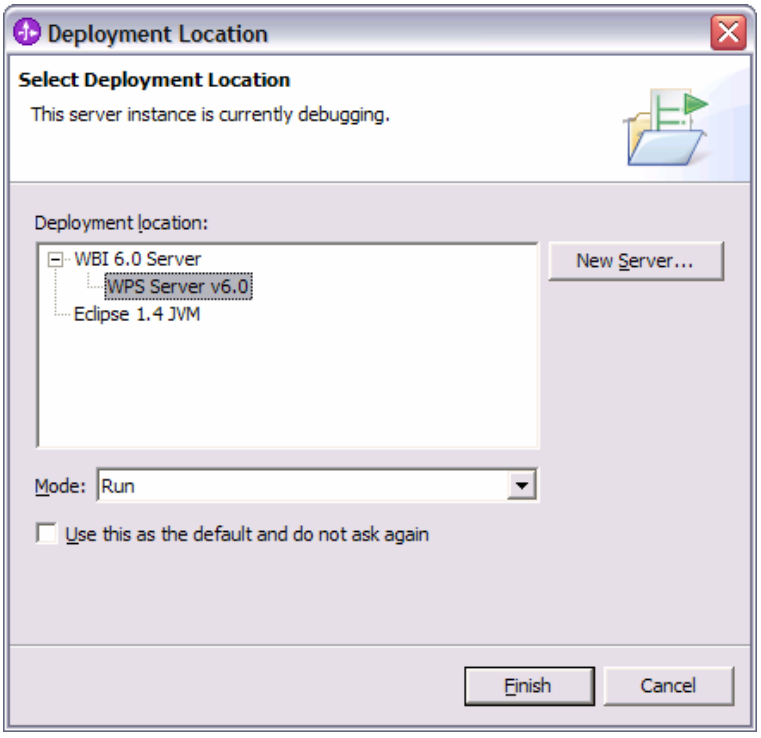


__ f. Expand the + next to the element to expand twice, enter whatever values you'd like for the string fields, for example:




__ g. Select **Continue.**

__ h. Expand the WBI 6.0 Server on the **Select Deployment Location** panel and select **WPS Server v6.0.**



__ i. Select **Finish.**

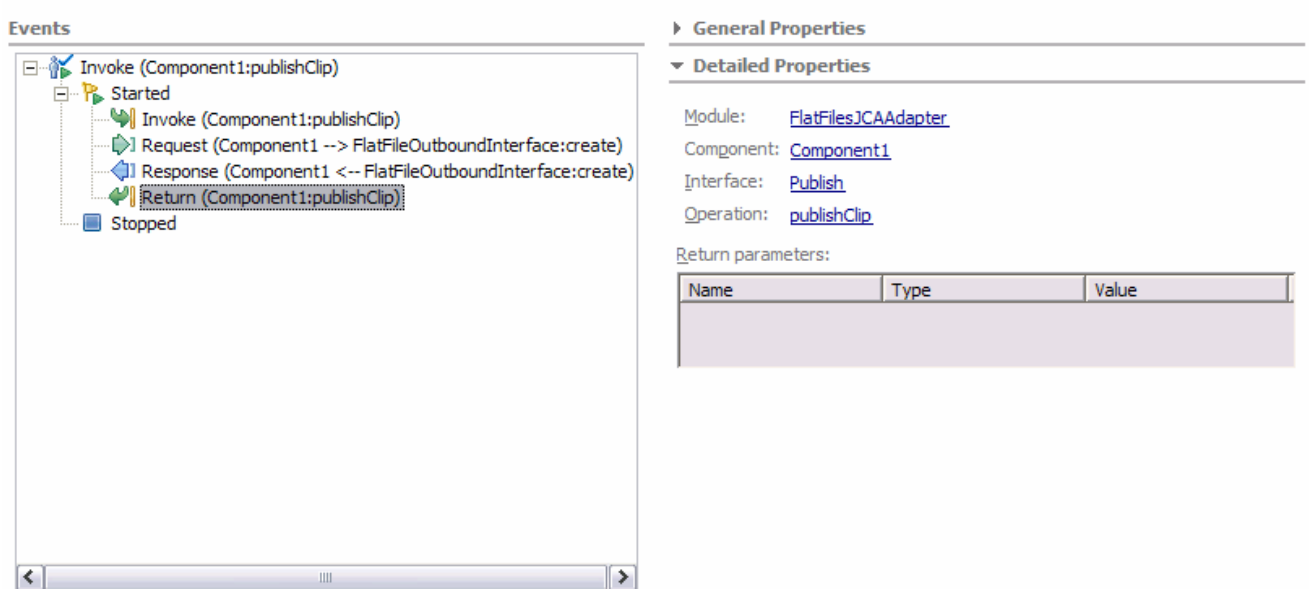
---

**Note:** You may see several exceptions repeated in the Console and SystemOut.log similar to the following. Ignore these exceptions. J2CA0009E: An exception occurred while trying to instantiate the ManagedConnectionFactory class com.ibm.j2ca.flatfile.FlatFileManagedConnectionFactory used by resource FlatFilesJCAAdapter/com/test/flat/FlatFileOutboundInterface_CF : java.lang.ClassNotFoundException: com.ibm.j2ca.flatfile.FlatFileManagedConnectionFactory

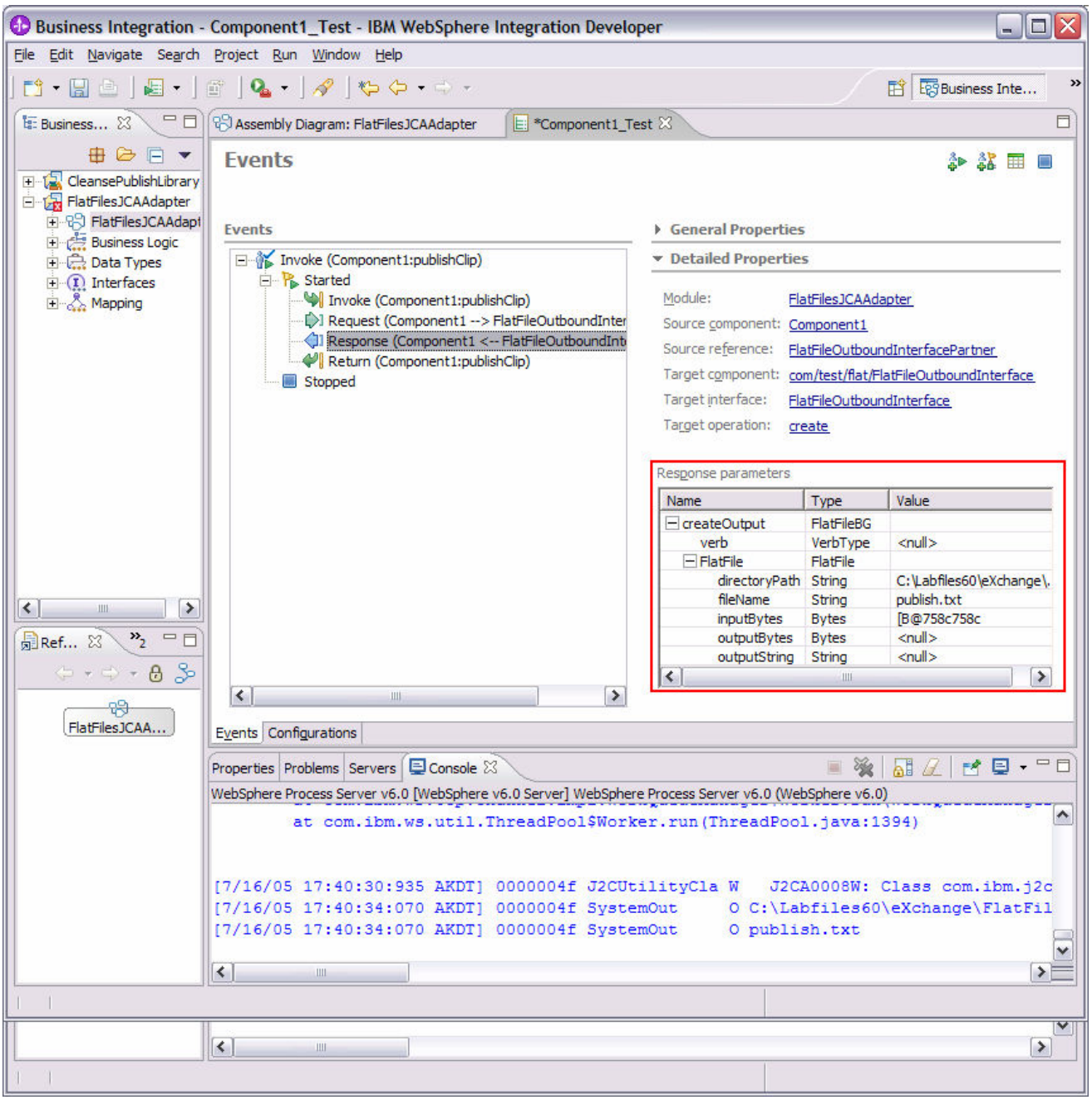
---

____ 4. Verifying Results.

__ a. From the **Events** tab, the panels will look as follows:



__ b. Click on the **Response Monitor Component 1** and the panel should look as follows:




__ c. On the file system, located in the specified output directoryPath and fileName specified, there should be a file:

**C:\Labfiles60\eXchange\FlatFile\outdir** with a file named **publish.txt** with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>

<scm:Clip xsi:type="scm:Clip"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:scm="http://CleansePublishLibrary/com/clipsandtacks/scm">
```

```
        <clipID>id1</clipID>
        <GLN>1234</GLN>
        <clip>binder</clip>
        <size>medium</size>
        <color>black</color>
        <brand>mightyclip</brand>
        <retailItems>
          <itemID>item20</itemID>
          <GTIN>7890</GTIN>
          <package>pkgA</package>
          <quantity>100</quantity>
          <fullDescription>descriptionhere</fullDescription>
          <price>20</price>
          <startDate>jan 1 2005</startDate>
          <endDate>sept 5 2005</endDate>
          <contactFirstName>Mary</contactFirstName>
          <contactLastName>Smith</contactLastName>
        </retailItems>
      </scm:Clip>
```

____ 5. Clean up test environment.

__ a. Close the Component1_Test panel, click **No** if prompted to save changes. Close the Assembly Diagram panel.

__ b. Right-click on WPS Server v6.0 in the Servers view and select **Add and remove projects…**

__ c. Select **FlatFilesJCAAdapter** and click **< Remove**. Click **Finish**.

__ d. Right-click on **WPS Server v6.0** and select **Stop.**

## What you did in this exercise

In this exercise, you imported the WebSphere Adapter for Flat Files and integrated it into an SCA application that creates a file to the file system. You created and implemented a Java component containing the business logic of the application. Component Test was used to input values and test the application.

An SCA export was also created in this exercise, enabling this SCA application to be available to other SCA applications that may want to invoke it.