Flat File adapter inbound lab

What this exercise is about

The objective of this lab is to provide you with an understanding of WebSphere Adapter for Flat Files and inbound event processing. In this lab you will deploy the WebSphere Adapter for Flat Files, using WebSphere Integration Developer, and integrate it with an SCA application that polls for inbound events and processes those inbound requests from the file system.

Lab requirements

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

- WebSphere Integration Developer V7.0 installed and updated with latest fixes
- WebSphere Process Server V7.0 Test Environment installed and updated latest fixes
- Extract Labfiles70.zip to your C:\ (your root) drive

What you should be able to do

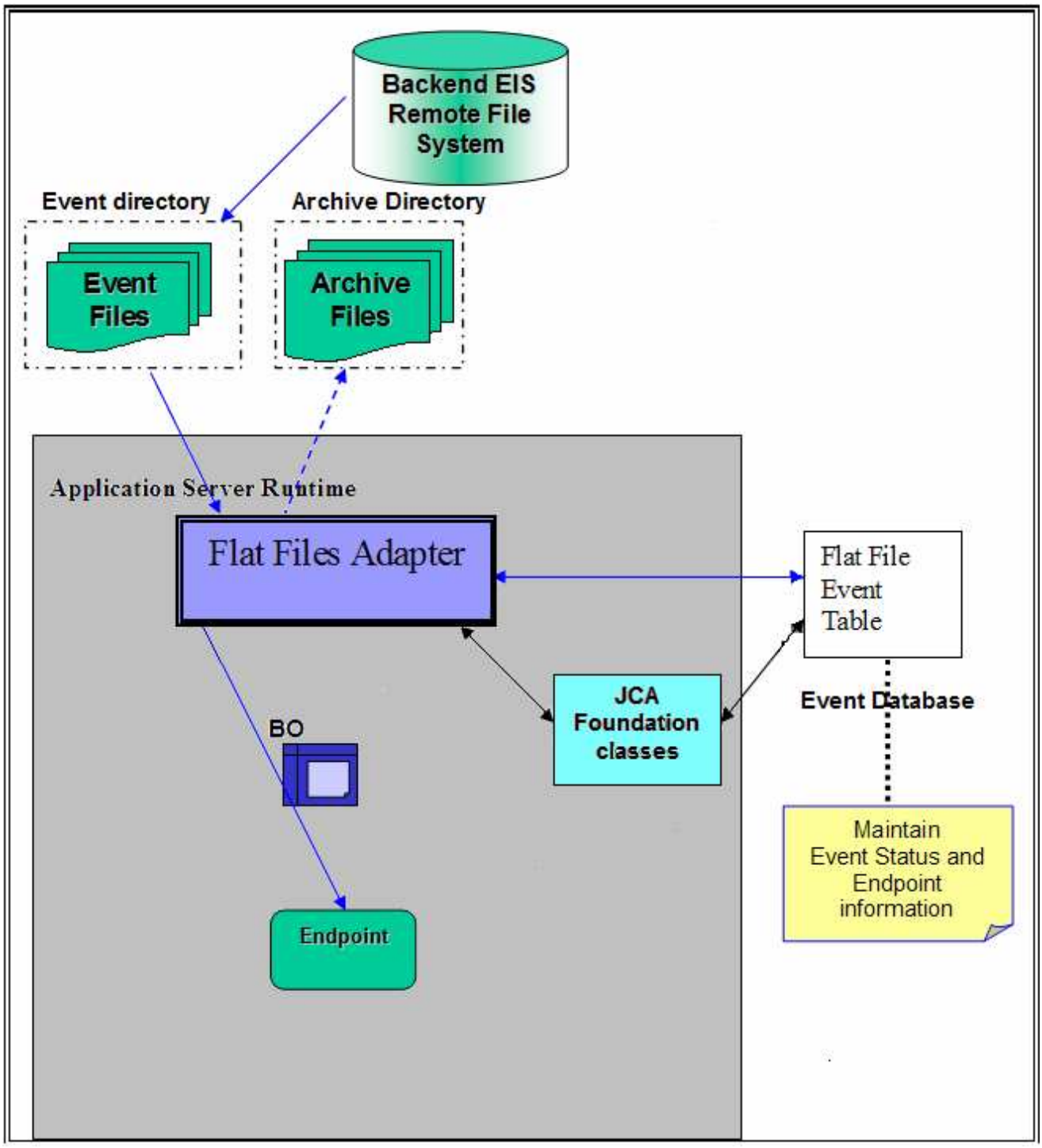
At the end of this lab you should be able to:

- Import Flat File adapter RAR file into WebSphere Integration Developer
- Use the external service wizard to configure activation spec properties, resource adapter properties to generate business objects and other artifacts and configure function selector, data binding and data handlers
- Deploy the adapter application onto WebSphere Process Server
- Test the deployed application using WebSphere Process Server test environment for both pass-through and non pass-through using different scenarios and patterns
- Restore the server configuration

Introduction

The backend EIS is the source of events. When events are generated at the EIS, files are created by the EIS in the file system at a specific directory location, which needs to be configured as the Event Directory for the adapter.

The adapter polls event files from the multiple Event Directories periodically based on user-configured Event File Mask, Poll Quantity and Poll Period. Each file will form a fundamental unit of transfer to the adapter that can internally represent more than one backend events. The adapter sends the entire file content across to the endpoint in the form of a Business Object for further processing.



The event management framework takes care of delivering the event only once to the endpoint. FF RA (Flat File Resource Adapter) internally uses an event table to log the status of the events that have been polled but not yet posted to the endpoint. Once an event is polled, the FF RA generates the Event ID and stores the event reference in the event table with a NEW status. The base class functions call the adapter methods to process the events. The FF RA changes the status for the event in the Flat Files event table to an IN_PROGRESS status. It wraps the file content in a Record and posts the same across to the configured endpoint.

Once the event business object is posted, the event entry is then deleted from the event table. After the business objects are sent out of the adapter, the events are archived optionally, in a user-configured archive directory on the file system.

Exercise instructions

Some instructions in this lab are Windows® operating-system specific. If you plan on running the lab on an operating-system other than Windows, you will need to run the appropriate commands, and use appropriate files (.sh or .bat) for your operating system. The directory locations are specified in the lab instructions using symbolic references, as follows:

Reference variable	Windows location	AIX® or UNIX® location
<WID_HOME>	C:\Program Files\IBM\WID7Beta	
<WPS_HOME>	C:\<WID_HOME>\runtimes\bi_v70	
<FFADAPTER_HOME>	<WID_HOME>\ResourceAdapters\FlatFile_7.0.0.0.0\deploy	
<LAB_FILES>	C:\Labfiles70	/tmp/Labfiles70
<WORKSPACE>	<LAB_FILES>\FlatFileInbound\workspace	
<EVENT_DIR>	<LAB_FILES>\FlatFileInbound\eventdir	
<ARCHIVE_DIR>	<LAB_FILES>\FlatFileInbound\archivedir	
<FFFILES>	<LAB_FILES>\FFFiles	
<TEMP>	C:\temp	/tmp

Windows users' note: When directory locations are passed as parameters to a Java™ program such as EJBdeploy or wsadmin, it is necessary to replace the backslashes with forward slashes to follow the Java convention. For example, replace C:\Labfiles70\ with C:/Labfiles70/

Instructions if using a remote server for testing

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

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

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

Part 1: Initialize the workspace and prepare for the lab

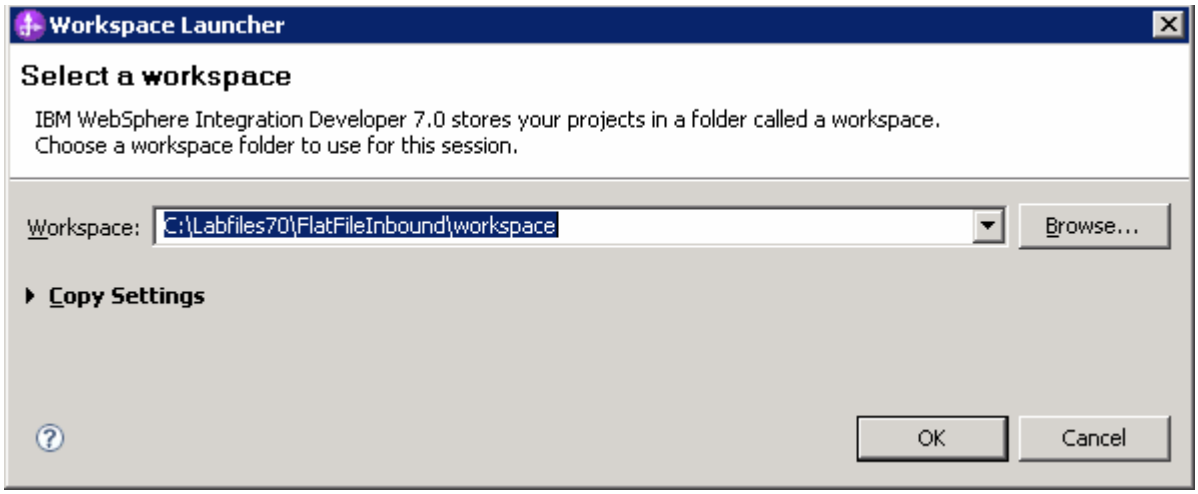
This part of the lab, you will start the WebSphere Integration Developer V7.0 with a new workspace and create required data source and database using the administrative console of WebSphere Process Server V7.0

____ 1. Extract the provided Labfiles70.zip to your C:\ (root) drive, if you have not already done so. This will create the necessary subdirectory structure to complete the lab, and provides you with sample text files

____ 2. Start the WebSphere Integration Developer V7.0 with a new workspace

__ a. Select **Start > All Programs > IBM WebSphere Integration Developer > IBM WebSphere Integration Developer V7.0 > WebSphere Integration Developer V7.0**

__ b. From the Workspace Launcher window, enter **<WORKSPACE>** for the Workspace field



____ 3. Click the  button on the right corner to close the Welcome page and proceed with the workbench

____ 4. Follow the instructions of '**WPBMv70_ConfigureDatasourceTask'** lab with these inputs and create the data source and data base required for this lab:

__ a. Data source name: **FF**

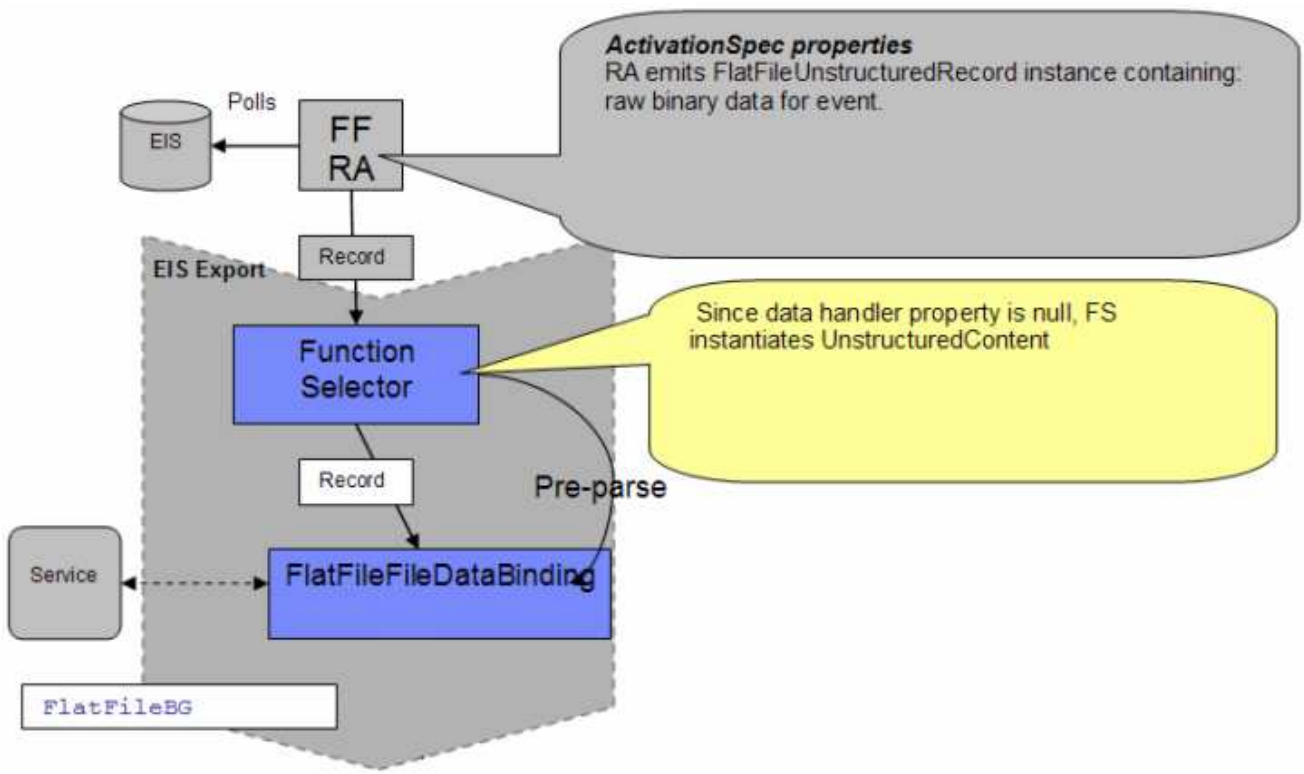
__ b. JNDI name: **jdbc/FF**

__ c. Database name: **FFDB**

Part 2: Pass through scenario

Inbound support can be broadly classified into two flows, one that involves data transformation and another without it (pass-through). In this part of the lab, you will configure the pass through scenario using the new External Service option from the WebSphere Integration Developer and then test the configuration.

Pass through flow for the inbound scenario:



- Event data is picked from the event file based on event file size and SplitCriteria, converted into a byte[] and set as a ByteArrayInputStream on FlatFileInputStreamRecord.
- Protocol specific information like event file name, directory name are also set in the FlatFileInputStreamRecord.
- The FlatFileInputStreamRecord is sent to the function selector. The function selector instantiates the wrapper. There is no value for the data handler property in the chose databinding, so no content-specific data handler is invoked but UnstructuredContent SDO is instantiated. It is set with byte content.
- The FlatFile wrapper is set with protocol specific information and UnstructuredContent is set in FlatFile
- FlatFile wrapper data object is set in FlatFileBG and sent to the endpoint.

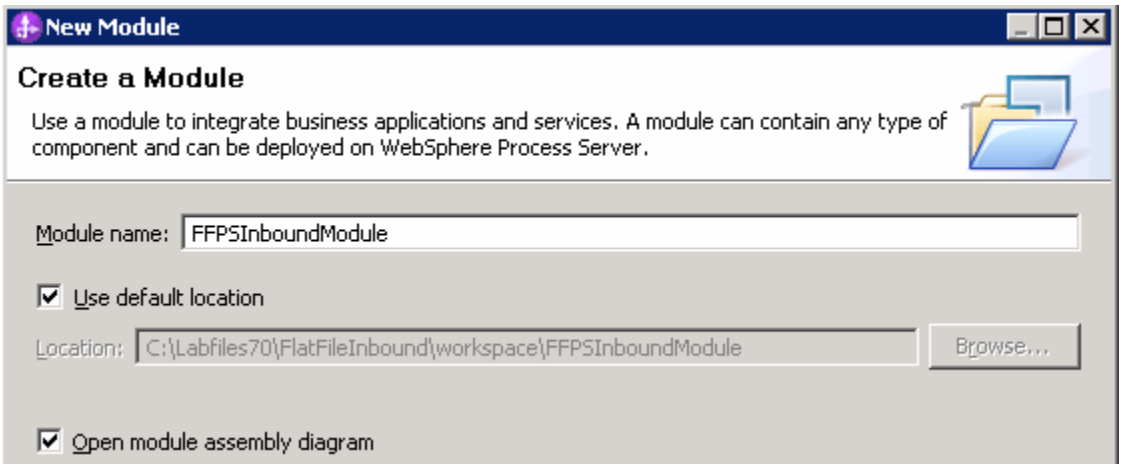
2.1. Configure pass through using the external service wizard

In this part of the lab you will use the new external service feature to create and configure the function selector, data binding and other required artifacts to test the inbound pass through scenario

____ 1. Create the module: FFPSInboundModule

__ a. From the Business Integration window, right-click and select **New > Module**

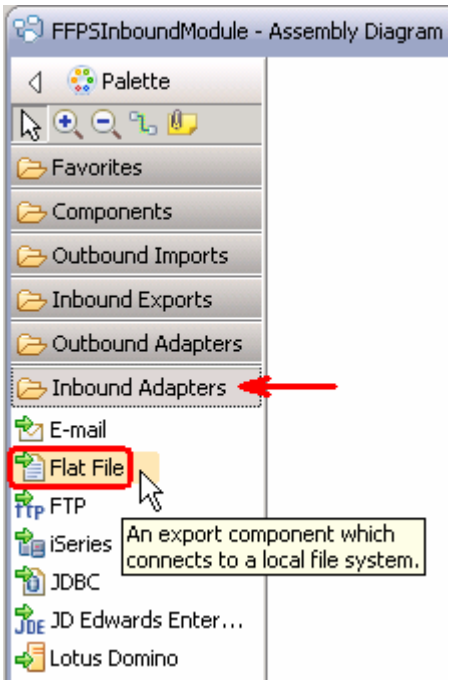
__ b. From the New Module window, enter **FFPSInboundModule** for the Module Name



__ c. Ensure that the box next to **Open module assembly diagram** is checked and then click **Finish**. You should see a new module, **FFPSInboundModule**, created from your Business Integration window

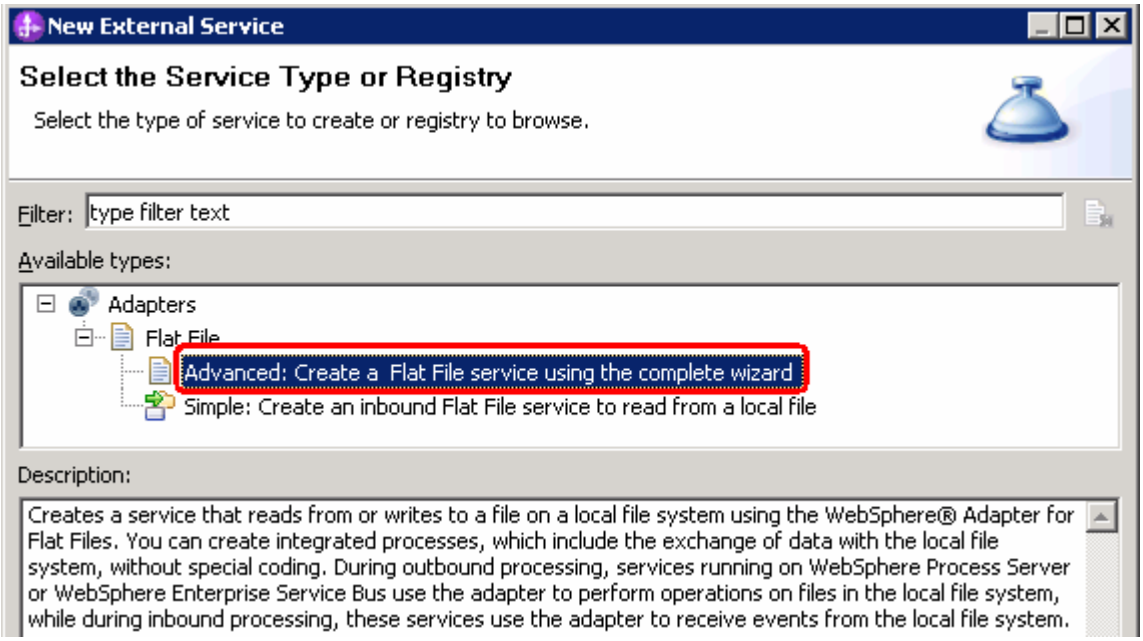
____ 2. To start external service from the palette:

__ a. From the **palette**, on the left side of assembly diagram, click **Inbound Adapters**:



____ 3. Under Inbound Adapters, click the **Flat File** and then click the empty canvas of the assembly diagram. The New Flat File Service wizard is opened

____ 4. From the New External Service window, expand **Adapters > Flat File** and select **Advanced: Create a Flat File service using the complete wizard**



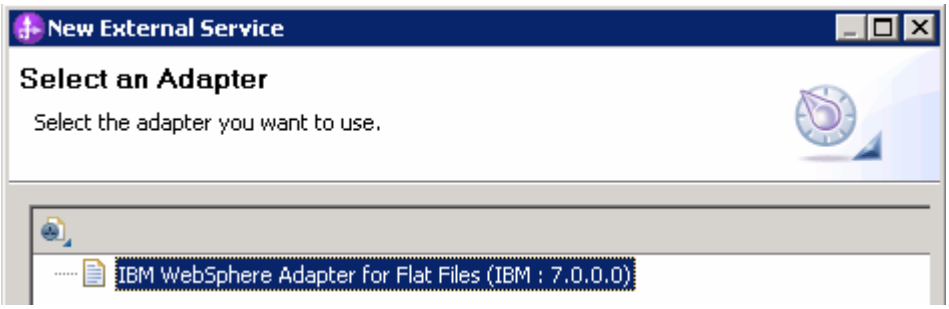
__ a. Click **Next**

Note: You can also start the external service from the **File menu** option:

From the main menu, select **File > New > External Service**. This opens an external service wizard that helps you obtain a service that establishes connectivity with other systems.

Select **Adapters > Flat File** and click **Next**

____ 5. On the Specify the Discovery Properties screen, select **IBM WebSphere Adapter for Flat Files (IBM : 7.0.0.0)** and click **Next**



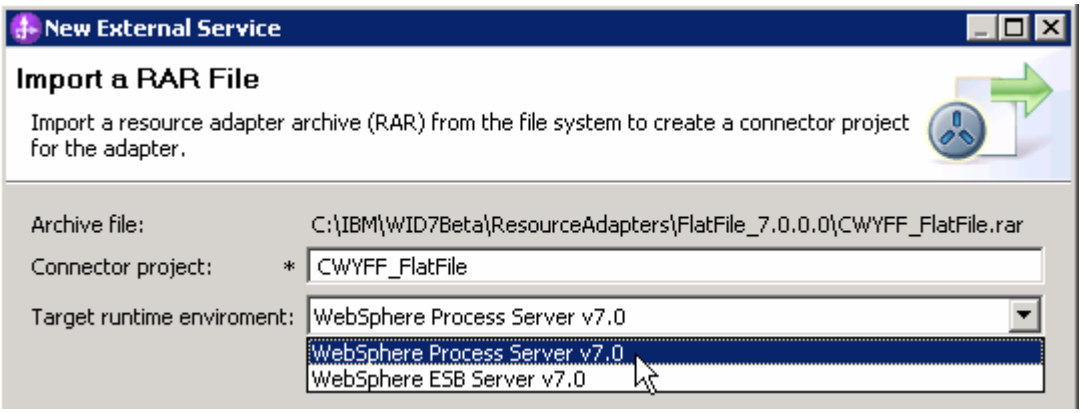
____ 6. Import a RAR File screen:

In this step, you will import a connector resource adapter archive from the file system into your WebSphere Integration Developer workspace. The adapter RAR file already exists under **<FFADAPTER_HOME**>.

__ a. The default connector file is selected. This file is shipped along with WebSphere Integration Developer

__ b. Accept the default name for connector project, **CWYFF_FlatFile**. You can change it to any other name, but for this lab, you can leave the default name.

__ c. For target server, ensure that **WebSphere Process Server v7.0** is selected



__ d. Click **Next**

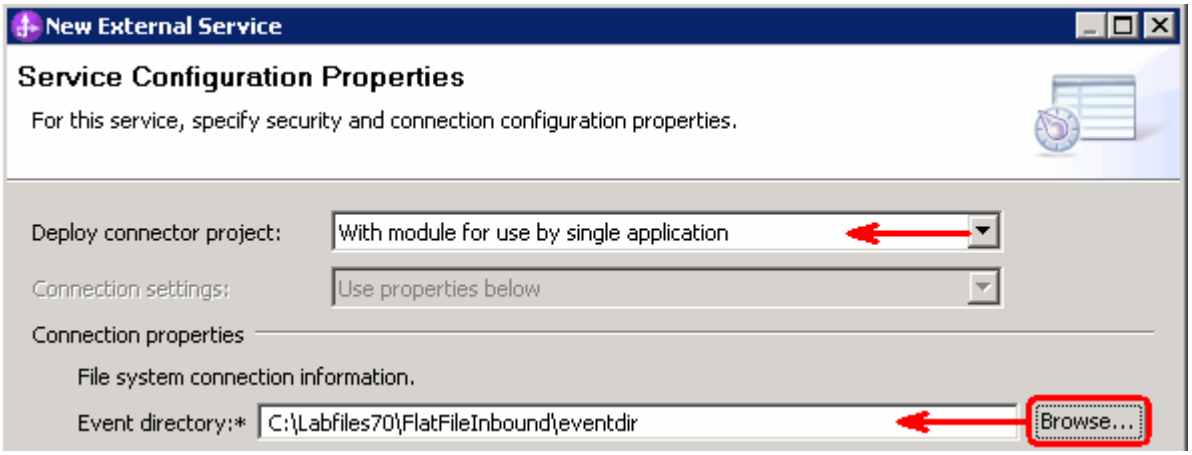
Note: The resource adapter archive file is imported and a new connector project, **CWYFF_FlatFile**, is listed under the Business Integration view.

Note: If you are using the **File menu** option to start the external service wizard, you are asked to select the **Processing Direction** at this point. Select the radio button next to **Inbound** and click **Next** to proceed to the next step.

____ 7. Service configuration properties:

__ a. Deploy connector project: ensure that the default option **With module for use by single application** is selected

__ b. Under Connection Configuration, click **Browse…** next to Event directory and select **<EVENT_DIR>** from the pop-up window:



Note: Alternatively, you can also replace the absolute directory path with WebSphere variables for the Event directory, Archive directory. Refer to '**Flat File adapter – Processing COBOL copy book files lab**' for more details on this new feature introduced in V6.2.

____ 8. Rule editor: Configure rules for event filtering.

Note: You will configure three rules to match three different file name patterns. Adapter polls only those files that match the rule that is configured here.

New in V7.0: The rule table in the Properties view is now displayed in the table format instead of a string.

__ a. Define a rule to filter only .txt files:

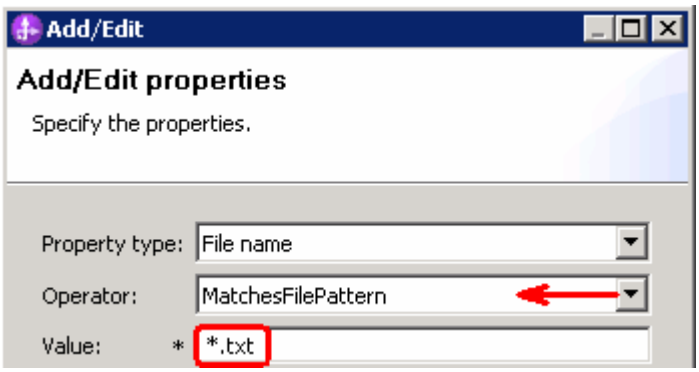
1) Click **Add…** next to the rule editor table

2) From the Add/Edit window, enter these:

a) Property Type: select **File name** from the drop down list

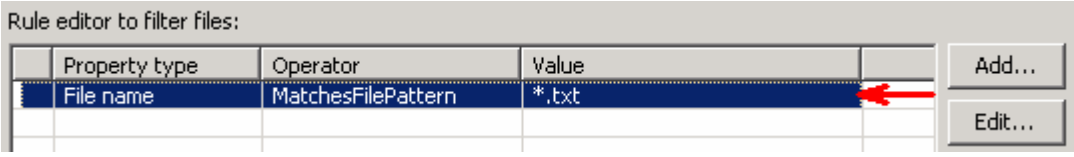
b) Operator: select **MatchesFilePattern** from the drop down list

c) Value: ***.txt**

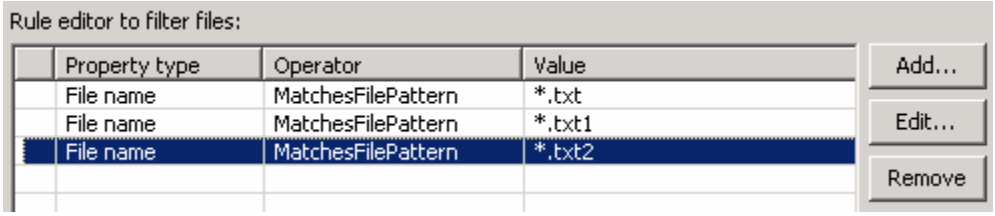


d) Click **Finish**

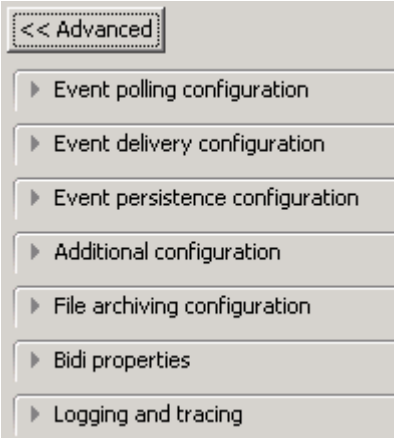
3) You should see a new rule entry in the table as shown below:



__ b. Repeat the instructions listed in step 8.a and create two more rules as shown below:



____ 9. Click **Advanced >>** to see the hidden advanced properties that can be configured:



You can click each of the configurations and review the options available under it. For this lab, you will need only some of these properties.

__ a. Event polling configuration: This has all the polling configuration details and for this lab, you can accept the defaults.

__ b. Event delivery configuration:

1) **Ensure once-only event delivery**: You should check this box only if you are using data source and table name in the Event persistence configuration (below). If this property is set to true, while using in-memory capability (explained below), the adapter will log a warning message. By default this is selected and you can accept the default selection.

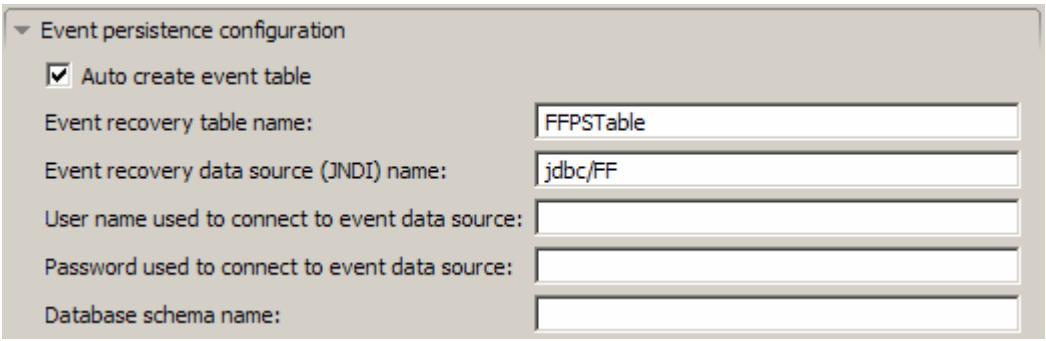
__ c. Event persistence configuration:

1) Ensure that the **Auto create event table** is **checked**

2) Event recovery table name: **FFPSTABLE**

3) Event recovery data source (JNDI) name: **jdbc/FF**

Note: This represents the JNDI name of the data source used by event persistence to get the JDBC database connection. The data source must be created in the WebSphere Process Server. You should enter the data source JNDI name that you created in Step 3 of Part 1.



Note: The event recovery data source (JNDI) name is **not mandatory** from V6.1. Now, the adapter can use **in-memory representation** of event table to store all the necessary information. The adapter uses this feature when event database information is not configured during inbound event polling. This feature will not support the capability of handling “Ensure once-only event delivery”.

__ d. Additional Properties: Accept all the **default** values

1) **Include business object delimiter in the file content (IncludeEndBODelimiter)**: When this property is set to true (selected) the delimiter (given in Split criteria) is appended to the business object content before further processing is done. This property is valid only if you are splitting the event files based on a delimiter, that is, only if the Split function class name (below) is com.ibm.j2ca.utils.filesplit.SplitByDelimiter. This is not selected by default.

2) **Split function class name**: This value takes a fully qualified class name of the class to be used in order to split the event file. It takes two values as of now:

- **com.ibm.j2ca.utils.filesplit.SplitBySize**: a class that splits the event file based on event file size. This is the **default** value for the split function class name.
- **com.ibm.j2ca.utils.filesplit.SplitByDelimiter**: a class that splits the event file based on delimiter (used to separate business objects in event file)

The delimiter or file size is specified in split criteria attribute.

3) **Specify the criteria to split file content**: This attribute takes different values based on value set in Split function class name.

- If split function class name is set to **com.ibm.j2ca.utils.filesplit.SplitBySize**, then split criteria must contain a valid number that represents the size in bytes. If event file size is greater than this value, it is split into chunks of this value and so many chunks are posted. If event file size is less than this value the entire event file is posted in one shot. When split criteria=0 (**default**), chunking is disabled.

- If split function class name is set to **com.ibm.j2ca.utils.filesplit.SplitByDelimiter**, then split criteria must contain the delimiter that separates the business objects in the event file.

Note: You will use the SplitByDelimiter class later in the content specific (non-pass through) scenario

---

4) **Poll subdirectories in event directory**:

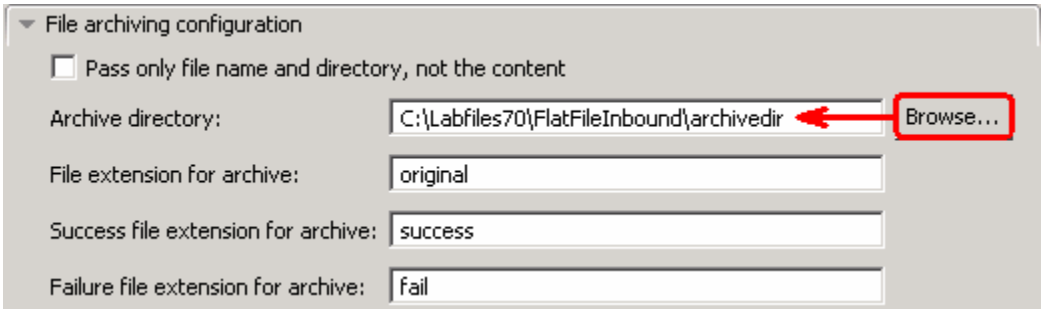
Note: **'Poll subdirectories in event directory'** is the new Boolean property added in V6.2. This property is used for recursive polling. Flat File adapter polls the given 'Event directory' and all the sub-directories present in the 'Event directory'. Refer to '**Log and confidential trace lab**' for more details on this new feature

---

__ e. File archiving configuration:

1) **Pass only file name and directory, not the content (FilePassByReference)**: This property determines if the adapter needs to load the contents of the file or just provide information on the directoryName and file name. If the value is true, the adapter just provides the directory name and file name. This is not selected by default.

2) Archive directory: click **Browse…** and select **<ARCHIVE_DIR>**



Note: Alternatively, you can also replace the absolute directory path with WebSphere variables for the Event directory, Archive directory. Refer to '**Flat File adapter – Processing COBOL copy book files lab**' for more details on this new feature introduced in V6.2.

__ f. **Logging and tracing**: Refer to '**Log and confidential trace**' lab for more details on this new feature

Function selector configuration: Function selectors are required in order to map between events generated by resource adapters and the appropriate SCA export function name.

There are two function selectors provided by the adapter that are supported by the FlatFile adapter - **FilenameFunctionSelector** and **EmbeddedNameFunctionSelector.**

FilenameFunctionSelector is used for generic FlatFile business object, where the object name cannot be determined from the event file The FilenameFunctionSelector is a rule-based function selector that can match a regular expression on a file name to an object name. This is represented in properties as a 2-column table, with N rows. So, you can have a mapping as follows: For any event file with a .txt extension, the corresponding object name is FFBG. The endpoint method name generated by the function selector in this case is emitFFBG. You need to set this same name for the Native method property after you add the operation.

You can also have same rules multiple times. If more than one rule matches, the function selector will return the object name based on the first matching rule.

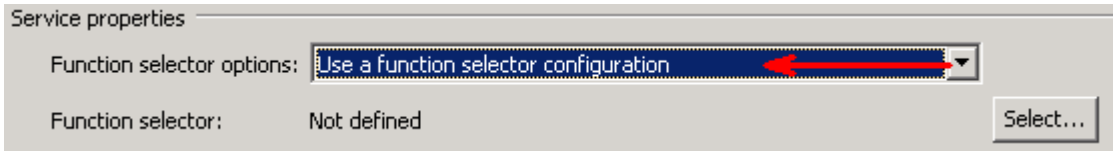
The function selector can be configured with multiple "rules", each of which contains an object name, and a regular expression to match against the file name. In the next part of this lab, you will create three such rules.

____ 10. Under Service properties, for function selector options, select **Use a function selector configuration** from the drop down menu

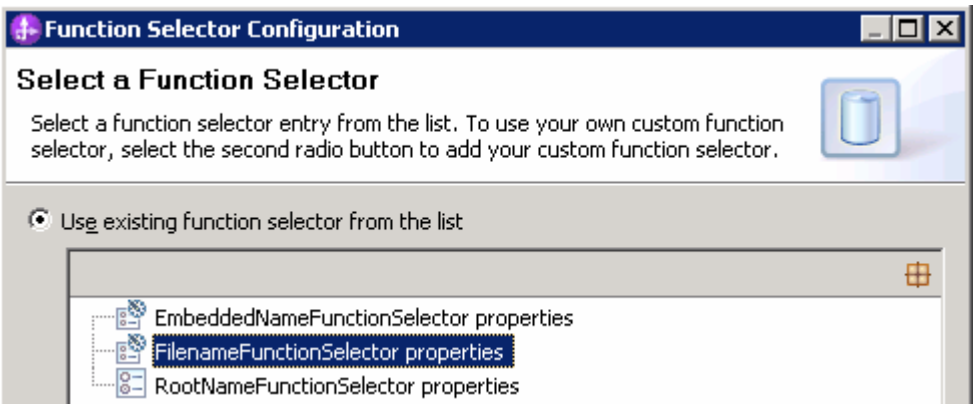
---

Note: If you select **Use default function selector 'FilenameFunctionSelector'** option for the function selector, the adapter automatically creates a function selector with FilenameFunctionSelector as the class name. But, for this lab, you need to define three rules at the function selector and so you are going to using the **'Use a function selector configuration'** option.

__ a. Click **Select** next to Function selector. The Binding Resource Configuration window is opened



__ b. Under '**Use existing function selector from the list**', select **FilenameFunctionSelector properties**



__ c. Click **Next**

____ 11. Define Function selector rules: In this step, you will define three different rules and object names associated with those rules.

When the adapter gets an event file, the function selector looks at the file type (ex: .txt, .tmp) and if the type matches with any of the rules defined in this table, it will associate the corresponding Object name of that rule to that event file.

Object Name	Rule	Operation Name	Native method name = emit + Object Name
FFBG	.*txt	emitFlatFileBG	**emitFFBG**
FFType1	.*txt1	emitFlatFile1	**emitFFype1**
FFType2	.*txt2	emitFlatFile2	**emitFFype2**

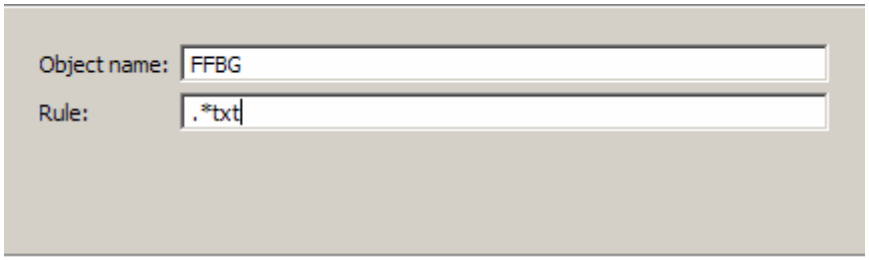
Note: **Make a note of the Object name you specify here as you are going to use this in the later part of the lab while specifying the Native method name**.

__ a. Click **Add…** next to the Function selector rules table

__ b. From the Add/Edit pop-up window, enter these:

1) Object name: **FFBG**

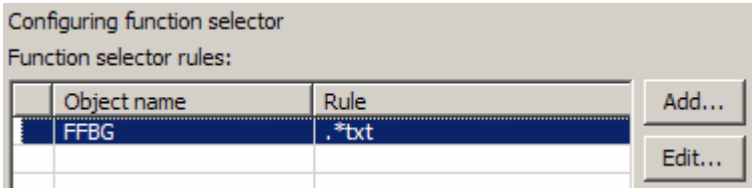
2) Rule: ***.txt**



Note: You have now created a rule for the files of type **.txt** and an Object name corresponding to those file type

3) Click **Finish** from Add/Edit window

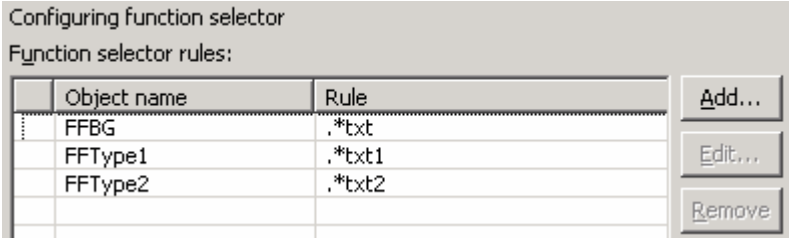
__ c. The created rule is populated under Function selector rules:



__ d. Repeat Steps 9.a and 9.b to add these function selector rules:

Object name	Rule
FFType1	*.txt1
FFType2	*.txt2

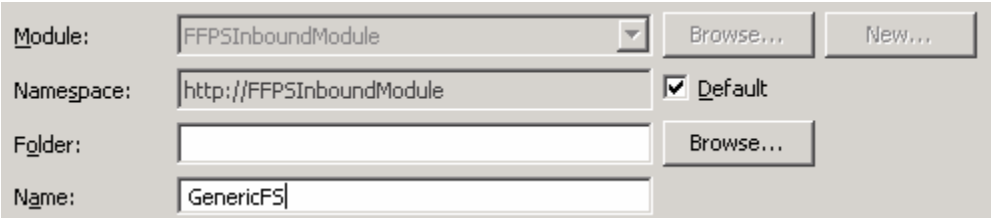
__ e. You should see these three rules:



__ f. Click **Next**

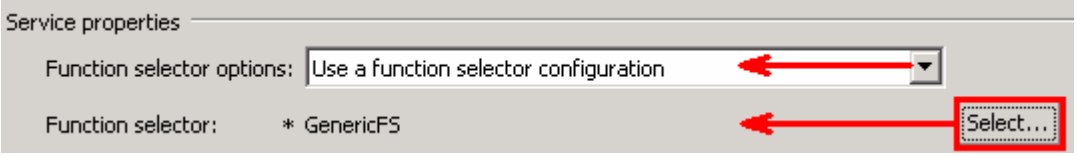
__ g. Ensure that the selected module is **FFPSInboundModule**

__ h. For **Name**, enter any string. For Ex: **GenericFS**



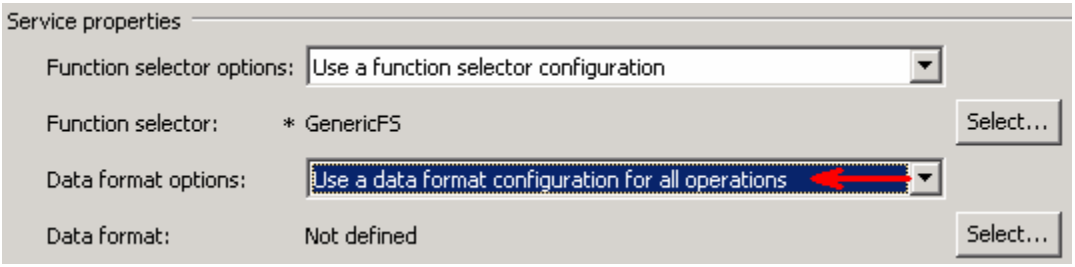
__ i. Click **Finish**

____ 12. You will now be back to External Service window and the function selector created in the previous steps is populated:



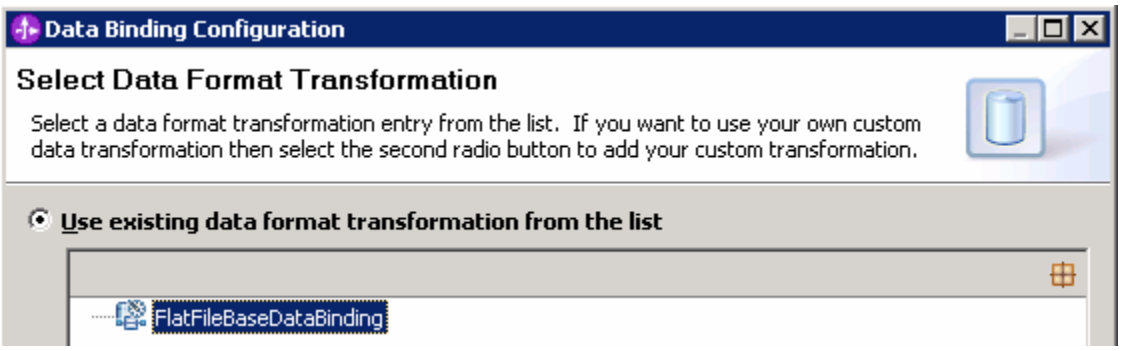
____ 13. You can define data binding in two places - service level (current screen of external service wizard) or later at the method level (Operations screen of the external service wizard). In this lab, you will define data binding at the service level (from this screen)

__ a. From the dropdown menu next to Data format options, select '**Use a data binding configuration for all operations**'



__ b. Click **Select…** next to **Data format**. A Binding Resource Configuration window is opened

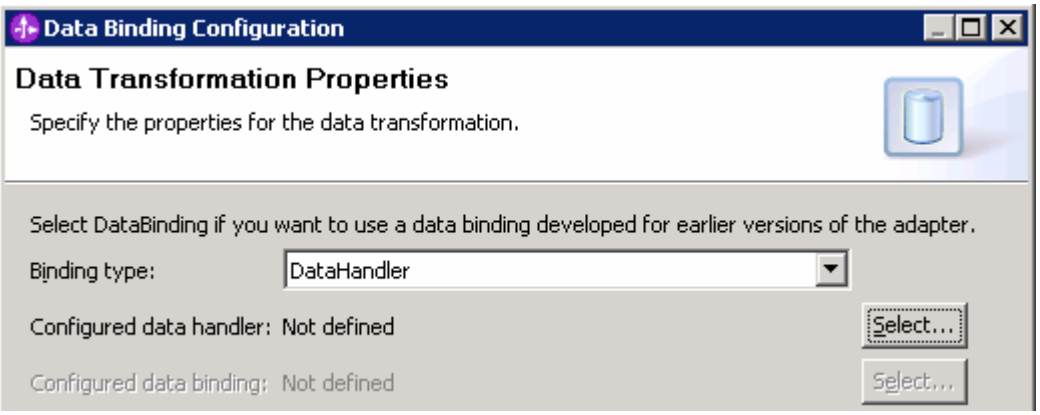
__ c. Select the radio button for **'Use existing data format transformation from the list'** and then select **FlatFileBaseDataBinding**



__ d. Click **Next**

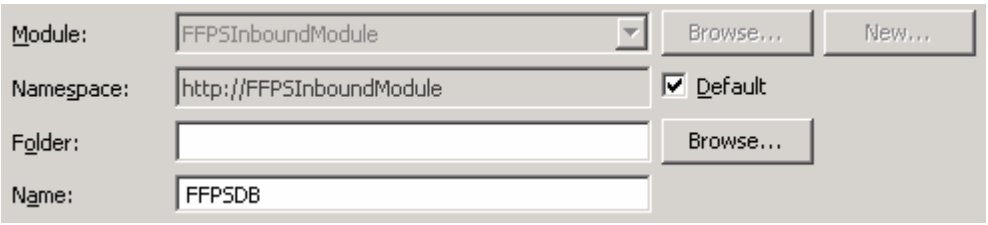
Note: Data handler configuration: Since you are doing the pass through scenario, you do not need to configure any data handler.

__ e. Click **Next** from the Data Transformation Properties screen



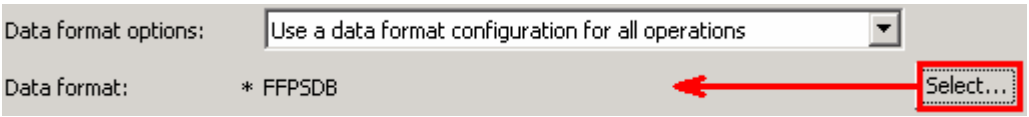
__ f. Note that the selected module is **FFPSInboundModule**

1) For the **Name**, enter **FFPSDB**



2) Click **Finish**

__ g. Now the **FFPSDB** should be displayed for Data format:



____ 14. Check the box next to **Change logging properties for wizard** to view the output location of the log file and the logging level. You can change the logging level using the drop down menu and click **Next**

Following screen is the Operations screen where you can define all your operations.

From V6.1, you can select from three different Data types for any operation:

User defined type

Generic FlatFile business object

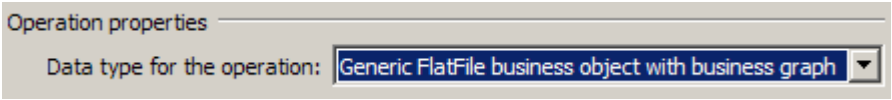
Generic FlatFile business object with business graph

The pass through scenario of this lab demonstrates the creation of three operations (for the three rules you have defined in the previous steps) using the Generic FlatFile business object and Generic FlatFile business object with business graph data types.

Create emitFlatFileBG Operation:

____ 15. From the Operations screen, click **Add…**

__ a. Add Operation window is opened. Select **Generic FlatFile business object with business graph** for the Data type and click **Next**



You are back to Operation window and because you have chosen the data type with business graph, the Input type is populated as **FlatFileBG**.

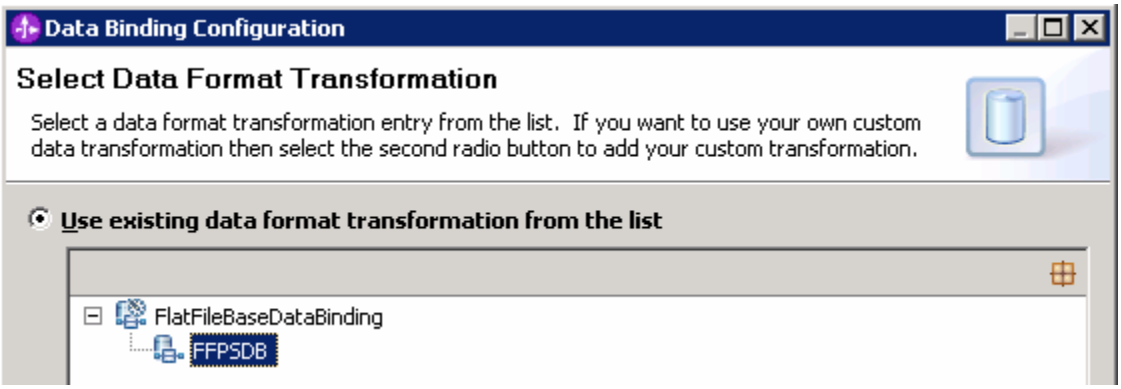
__ b. For **Operation name**, enter any name, for Ex: **emitFlatFileBG**

__ c. For **Data format options**, select **Use a data binding configuration** from the dropdown list

Define Data format:

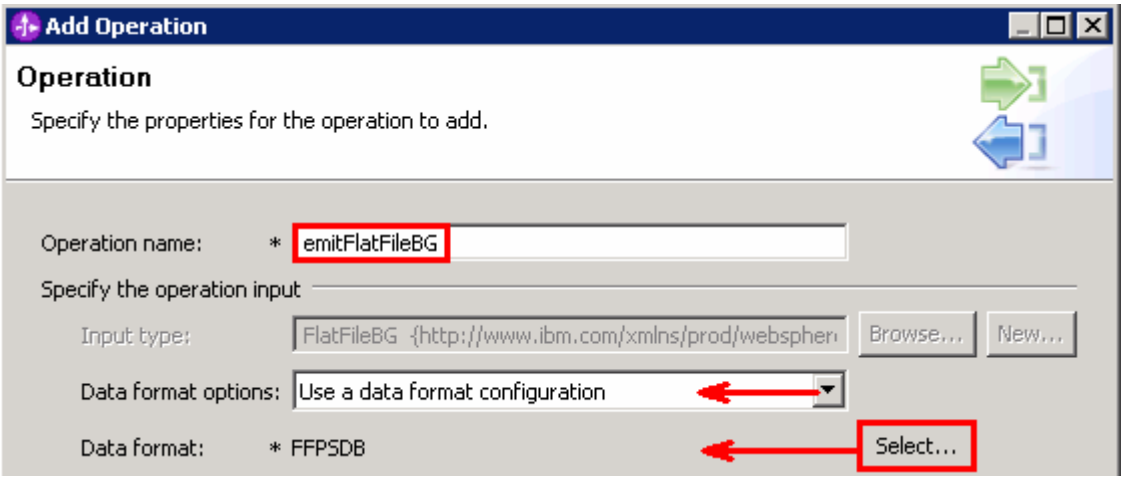
__ d. Click **Select…** next to **Data format**. A Binding Resource Configuration window is opened

__ e. Ensure that the radio button for 'Use existing data format transformation from the list' and then select **FlatFileBaseDataBinding > FFPSDB**

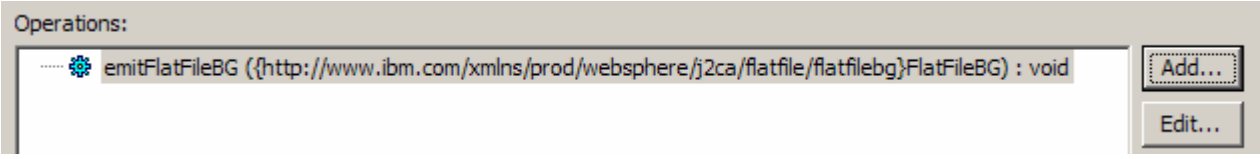


__ f. Click **Finish**

__ g. Now the **FFPSDB** is displayed for **Data format** in the Add Operation window:



__ h. Click **Finish**. The defined operation, **emitFlatFileBG**, is populated under Operations list



Create emitFlatFile1 operation:

____ 16. From the Operations screen, click **Add…**

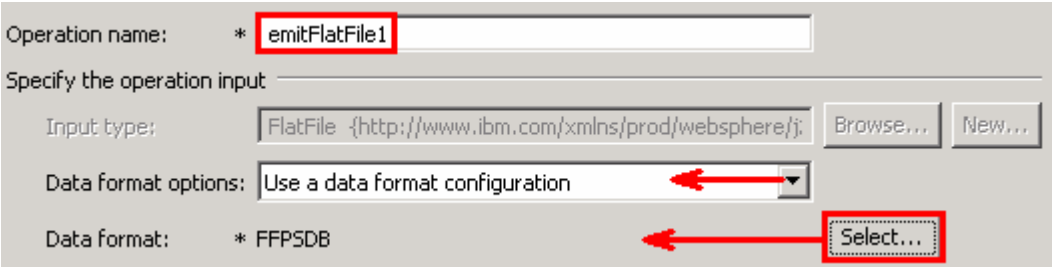
__ a. Select **Generic FlatFile business object** for the Data type and click **Next**

You are back to the Operation window and because you have chosen the data type with no business graph, the Input type is populated as **FlatFile**.

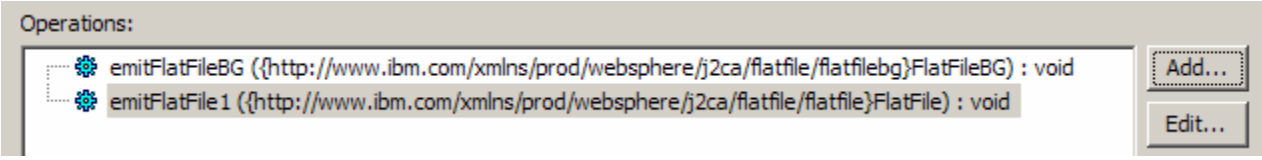
__ b. For the operation name, enter **emitFlatFile1**

__ c. Follow the instructions used to define data binding for emitFlatFileBG in the previous step, and define the same **FFPSDB** for Data format

__ d. Now the **FFPSDB** is displayed for **Data format** in the Add Operation window:



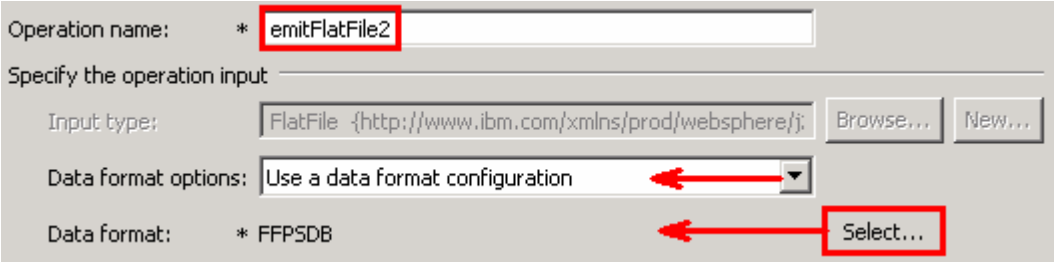
__ e. Click **Finish**. The defined operations, **emitFlatFile1**, is populated under Operations list



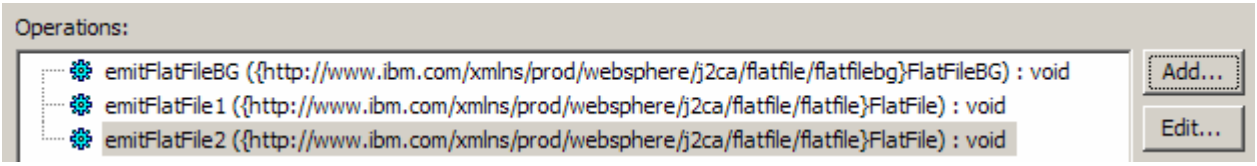
Create emitFlatFile2 Operation

____ 17. Repeat steps under 'Create emitFlatFile1 operation' (Step15) to add one more operation, emitFlatFile2, with these changes:

__ a. Operation Name: **emitFlatFile2**



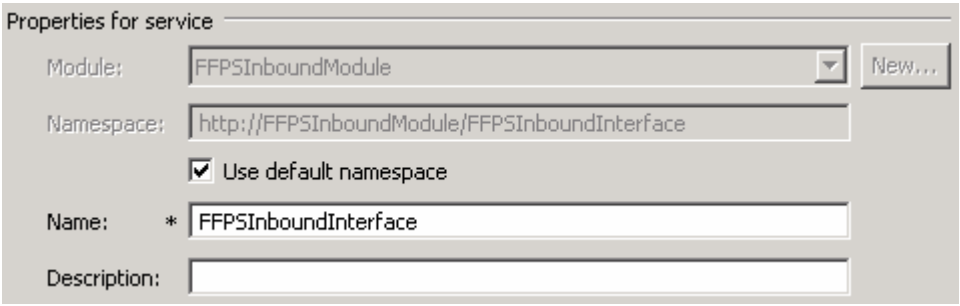
____ 18. You should now see these three operations:



__ a. Click **Next** from Operations screen

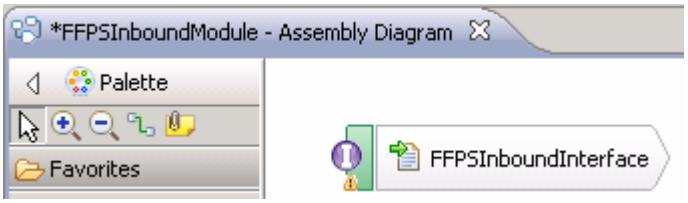
____ 19. From Generate Artifacts screen:

__ a. For Name, enter **FFPSInboundInterface**



__ b. Click **Finish**

____ 20. The Assembly diagram for FFPSInboundModule is opened with an Export component, **FFPSInboundInterface**:



__ a. Save (**Ctrl + S**) changes to your assembly diagram

Note: You need to manually change the Native method names.

The following table summarizes the rules, operations you have defined so far in this lab. Note the last column, which shows the Native method name for each of the rule and operation defined:

Object Name	Rule	Operation Name	Native method name = emit + Object Name
FFBG	.*txt	emitFlatFileBG	**emitFFBG**
FFType1	.*txt1	emitFlatFile1	**emitFFype1**
FFType2	.*txt2	emitFlatFile2	**emitFFype2**

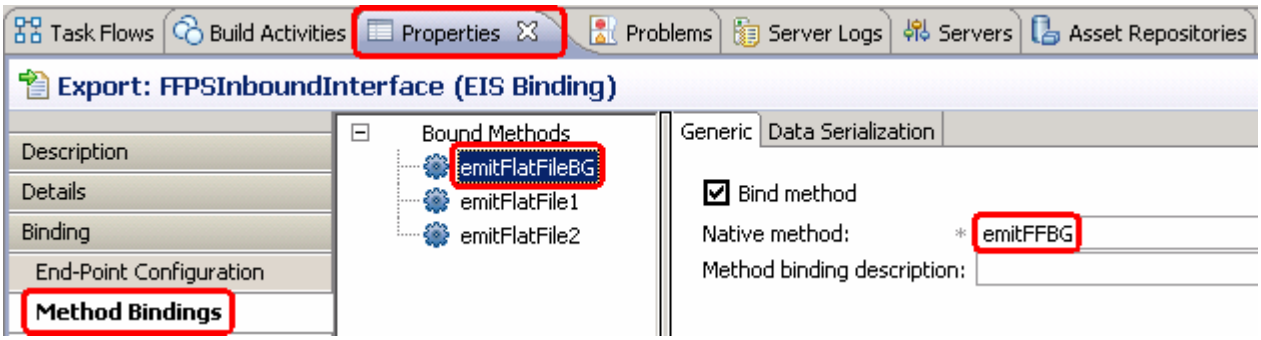
____ 21. Change the Native method binding name

__ a. Ensure that the **FFPSInboundInterface** is selected from the Assembly diagram

__ b. From the bottom panel, select **Properties > Binding > Method bindings**

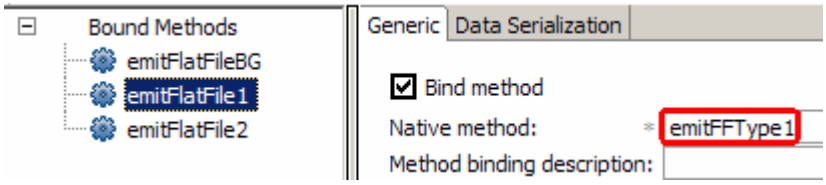
__ c. From the Bound Methods list, click **emitFlatFileBG**

__ d. From the right side, ensure that the Generic tab is selected, and then change the **Native method** to **emitFFBG**

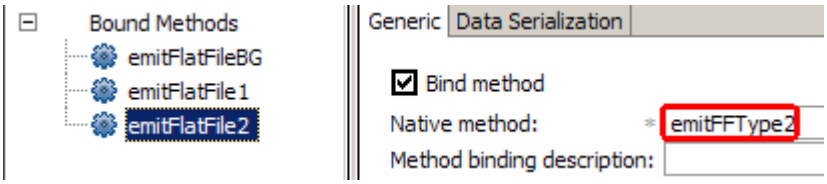


__ e. From the Bound Methods list, click **emitFlatFile1**

__ f. Similarly, for **emitFlatFile1**, change the **Native method** to **emitFFType1**

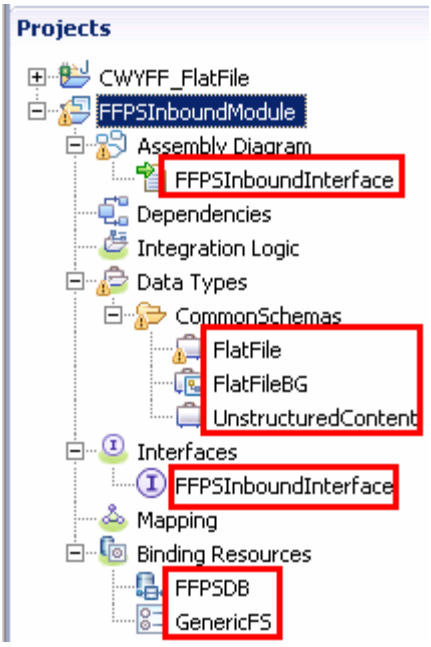


__ g. Similarly, for **emitFlatFile2**, change the **Native method** to **emitFFType2**



__ h. Save (**Ctrl + S**) your changes

____ 22. Review the FFPSInboundModule: The generated **Data Types**, **Interface**, and Function selector (**GenericFS**) and Data binding (**FFPSDB**) under Configured Resources can be found inside FFPSInboundModule

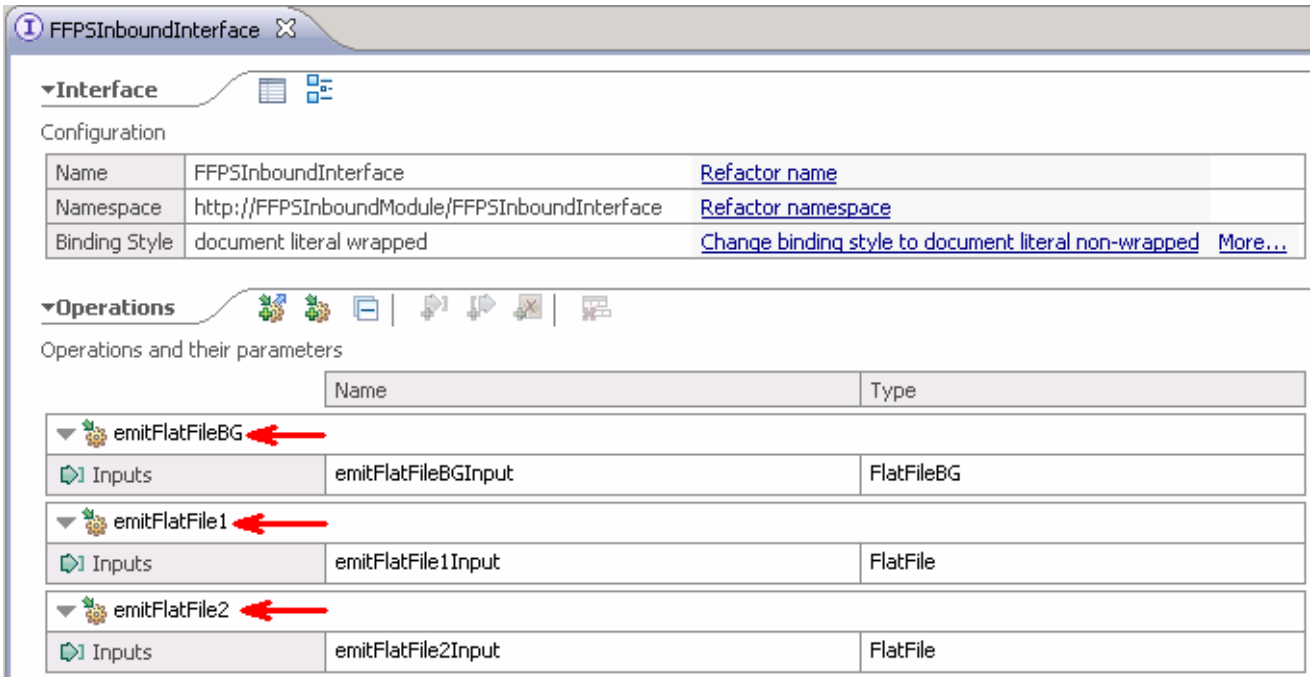


You can open each of these generated artifacts and business objects and review the properties inside.

Review the created methods inside the interface:

__ a. From the Business Integration view, expand FFPSInboundModule > Interfaces and then double-click **FFPSInboundInterface** to open it

__ b. You should see these three operations:



__ c. Close the interface, FFPSInboundInterface

2.2. Add Java component

In this part of the lab, you will add a Java component and then wire the component to the existing Export interface. The Java component is your endpoint.

____ 1. Open the assembly diagram for FFPSInboundModule (if it is already not open)

__ **a.** From the business integration view, expand **FFPSInboundModule** and double click **Assembly diagram**

____ 2. Drop a Java component to onto the assembly diagram

__ a. From the **Palette**, click **Components** to expand it

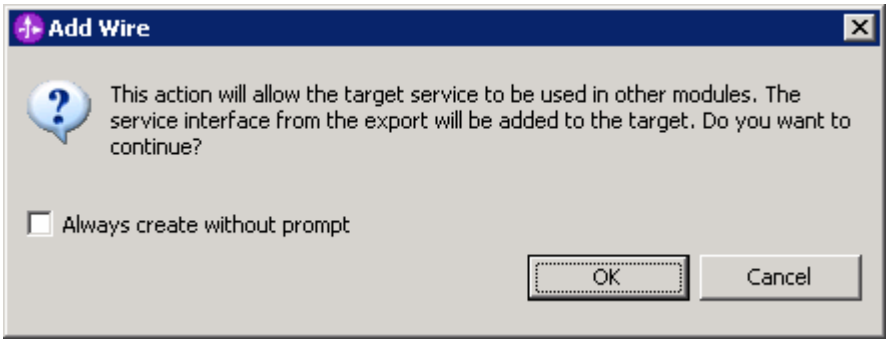
__ b. Click **Java** and then click the empty space of FFPSInboundModule assembly diagram. This will place a new component, **Component1** on the assembly diagram.

____ 3. Wire the FFPSInboundInterface to the Component1

__ a. Select the **wire** ( ) icon from the Palette

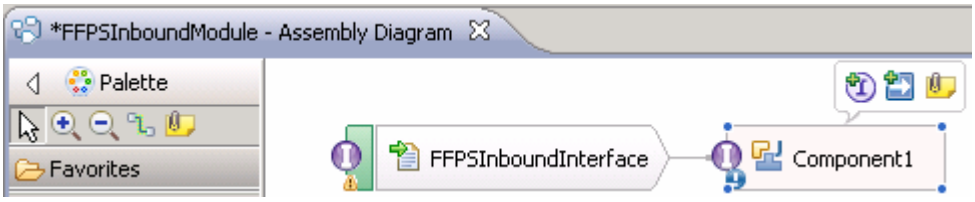
__ b. Click **FFPSInboundInterface** and then click **Component1** to wire them together

__ c. Select **OK** for the Add Wire pop-up window:



__ d. From the top of the Palette, click the **Selection Tool** icon ( ) to get back to the normal cursor mode

Your assembly diagram for FFPSInboundModule will look like this:



__ e. Right-click **Component1** and select **Generate Implementation** from the pop-up menu

__ f. On the **Generate Implementation** panel, select **default package**, and click **OK**

__ g. **Component1Impl.java** is opened in Assembly editor. Scroll down to the method **emitFlatFileBG** that needs to be implemented and add this code under that method:

```
System.out.println("****Inside emitFlatFileBG:type .txt files****");
```

__ h. Similarly add the print statement to **emitFFType1** method:

```
System.out.println("****Inside emitFlatFileBG:type .txt1 files****");
```

__ i. And finally, add the print statement to **emitFFType2** method:

```
System.out.println("****Inside emitFlatFileBG:type .txt2 files****");
```

__ j. Save (**Ctrl + S**) and close Component1Impl.java

__ k. Save (**Ctrl + S**) and close Assembly diagram: FFPSInboundModule

2.3. Test pass through scenario

In this part of the lab, you will use the WebSphere Process Server Test Environment to test the SCA application Inbound processing for the pass through scenario.

____ 1. Add the project to the WebSphere Test Environment server

__ a. Right-click **WebSphere Process Server v7.0** under the Servers view and select **Add and remove projects…** from the pop-up menu

__ b. From the Add and Remove Projects window, select **FFPSInboundModuleApp** under Available projects panel and click **Add >**

__ c. You will now see the **FFPSInboundModuleApp** added to the **Configured projects**

__ d. Click **Finish** and wait until the project is being published onto the server. The server will start in Debug mode if it is not already started before

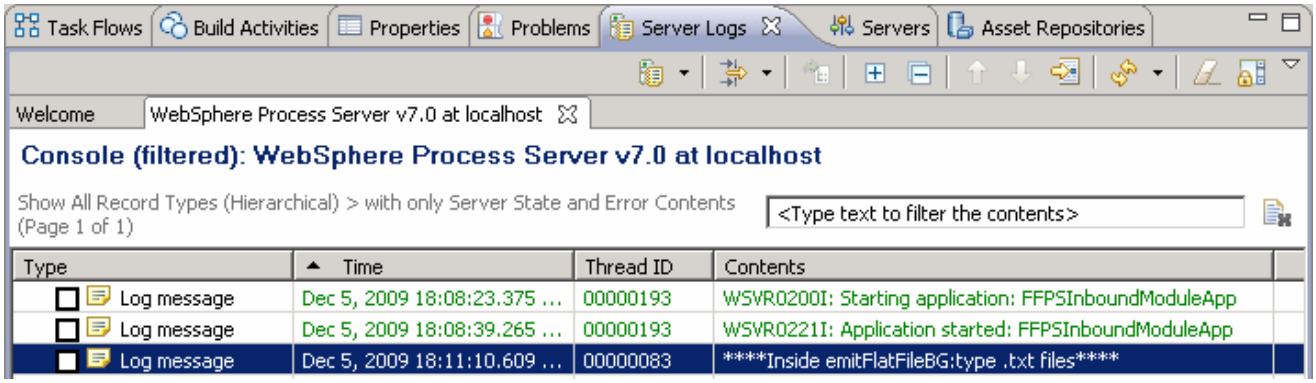
____ 2. Test the application by providing three different input files:

Note: For your convenience, three test files, **sample.txt, sample.txt1, sample.txt2** are placed in **<FFFILES>.**

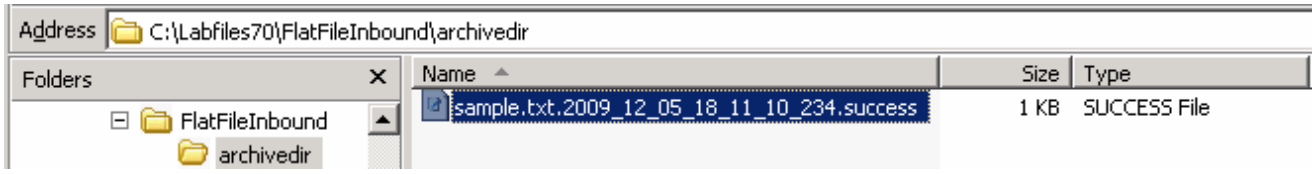
Test scenario 1: processing of .txt files

__ a. Copy the **sample.txt** file from <FFFILES> to <EVENT_DIR>. The adapter will poll the copied file from the event directory and will transfer it to the archive directory

__ b. Because you have placed a **.txt** file, it will pass through the **emitFlatFileBG** method and you should see this message in your **Server Logs** view (or SystemOut.log):



__ c. To verify your test results, check the **<ARCHIVE_DIR>** subdirectory, which should contain an archive of the event file with the same file name appended with year, month, date, system time, and success



Test scenario 2: processing of .txt1 files:

__ d. Copy the **sample.txt1** file from <FFFILES> to <EVENT_DIR>. The adapter will poll the copied file from the event directory and will transfer it to the archive directory

__ e. Because you have placed a **.txt1** file, it will pass through the **emitFlatFile1** method and you should see this message in your **Server Logs** view (or SystemOut.log):

Type	Time	Thread ID	Contents
Log message	Dec 5, 2009 18:08:23.375 ...	00000193	WSVR0200I: Starting application: FFPSInboundModuleApp
Log message	Dec 5, 2009 18:08:39.265 ...	00000193	WSVR0221I: Application started: FFPSInboundModuleApp
Log message	Dec 5, 2009 18:11:10.609 ...	00000083	****Inside emitFlatFileBG:type .txt files****
Log message	Dec 5, 2009 18:12:38.531 ...	000001ce	****Inside emitFlatFileBG:type .txt1 files****

__ f. To verify your test results, check the **<ARCHIVE_DIR>** subdirectory, which should contain an archive of the event file with the same file name appended with year, month, date, system time, and success

Address C:\Labfiles70\FlatFileInbound\archivedir

Folders: FlatFileInbound, archivedir

Name	Size	Type
sample.txt1.2009_12_05_18_12_38_500.success	1 KB	SUCCESS File
sample.txt.2009_12_05_18_11_10_234.success	1 KB	SUCCESS File

Test scenario 3: processing of .txt2 files:

__ g. Copy the **sample.txt2** file from <FFFILES> to <EVENT_DIR>. The adapter will poll the copied file from the event directory and will transfer it to the archive directory

__ h. Because you have placed a **.txt2** file, it will pass through the **emitFFType2** method and you should see this message in your **Server Logs** view (or SystemOut.log):

Type	Time	Thread ID	Contents
Log message	Dec 5, 2009 18:08:23.375 ...	00000193	WSVR0200I: Starting application: FFPSInboundModuleApp
Log message	Dec 5, 2009 18:08:39.265 ...	00000193	WSVR0221I: Application started: FFPSInboundModuleApp
Log message	Dec 5, 2009 18:11:10.609 ...	00000083	****Inside emitFlatFileBG:type .txt files****
Log message	Dec 5, 2009 18:12:38.531 ...	000001ce	****Inside emitFlatFileBG:type .txt1 files****
Log message	Dec 5, 2009 18:13:34.562 ...	000001ce	****Inside emitFlatFileBG:type .txt2 files****

__ i. To verify your test results, check the **<ARCHIVE_DIR>** subdirectory, which should contain an archive of the event file with the same file name appended with year, month, date, system time, and success

Address C:\Labfiles70\FlatFileInbound\archivedir

Folders: FlatFileInbound, archivedir, eventdir

Name	Size	Type
sample.txt2.2009_12_05_18_13_34_515.success	1 KB	SUCCESS File
sample.txt1.2009_12_05_18_12_38_500.success	1 KB	SUCCESS File
sample.txt.2009_12_05_18_11_10_234.success	1 KB	SUCCESS File

____ 3. Restore the Sever Configuration

__ a. Right-click **WebSphere Process Server v7.0** under the Servers view and select **Add and remove projects…** from the pop-up menu

__ b. Select **FFPSInboundModuleApp** under Configured projects and click **< Remove**

__ c. Click **Finish** after you see the application moved to Available projects. Wait until the application is being unpublished

2.4. Test pass through scenario with SplitBySize

In this last part of the lab, you will use the WebSphere Process Server Test Environment to test the SCA application Inbound processing for the pass through scenario with split by size.

____ 1. Configure/change the adapter properties from the Properties view:

__ a. Change to Business Integration perspective if you are in a different perspective

1) Select **Window > Open Perspective > Other…**

2) From the Select Perspective window, select **Business Integration (default)** and click **OK**

__ b. Expand **FFPSInboundModule** and double-click **FFPSInboundModule** to open it in Assembly Editor

__ c. Click **FFPSInboundInterface** from the Assembly Editor and select **Properties** tab from the bottom

__ d. Select **Binding** under Properties and select **End-point configuration** under Binding itself and then select the **Connection** tab

__ e. Now under **ActivationSpec Properties**, you can review the properties that were given during the external service wizard

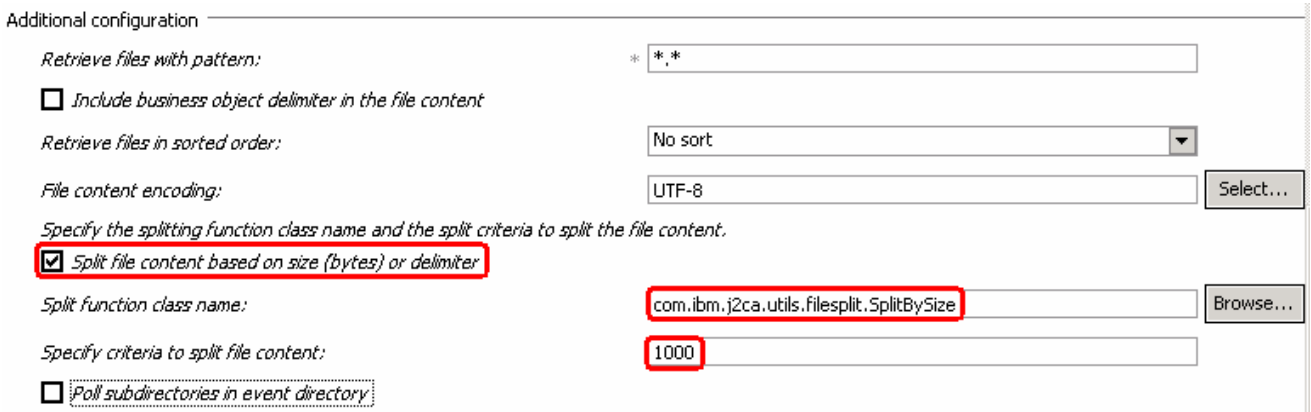
__ f. Scroll down to **Additional configuration**

1) Select the box for '**Split file content based on size (bytes) or delimiter**'

2) Split function class name: **com.ibm.j2ca.utils.filesplit.SplitBySize** (default)

3) Specify criteria to split file content: **1000**

Note: The value, 1000, you entered for the field, Specify criteria to split the content, is the size of the input file in Bytes.



__ g. Click Assembly diagram (or any where else so that the save button is enabled) and then save **(Ctrl +S)** your changes

____ 2. Modify Java code:

__ a. From the assembly diagram of **FFPSInboundModule**, double-click **Component1**

__ b. **Component1Impl.java** is opened in Assembly editor. Scroll down to the method **emitFlatFileBG** and add this code under that method:

```
System.out.println("***********Inside emitFlatFileBG: type .txt
files************");
DataObject FlatFile = emitFlatFileBGInput.getDataObject("FlatFile");
DataObject Unstructured = FlatFile.getDataObject("Content");
String astext = Unstructured.getString("AsText");

System.out.println("File content-----> "+astext);
```

Note: You can also copy the Java code from **<FFFILES>\SplitBySizeJavaCode.txt**.

__ c. Save (**Ctrl + S**) and close Component1Impl.java

__ d. Save (**Ctrl + S**) and close Assembly diagram: FFPSInboundModule

____ 3. Repeat Step 1 of part 2.3 to add the saved project **FFPSInboundModuleApp** to the server

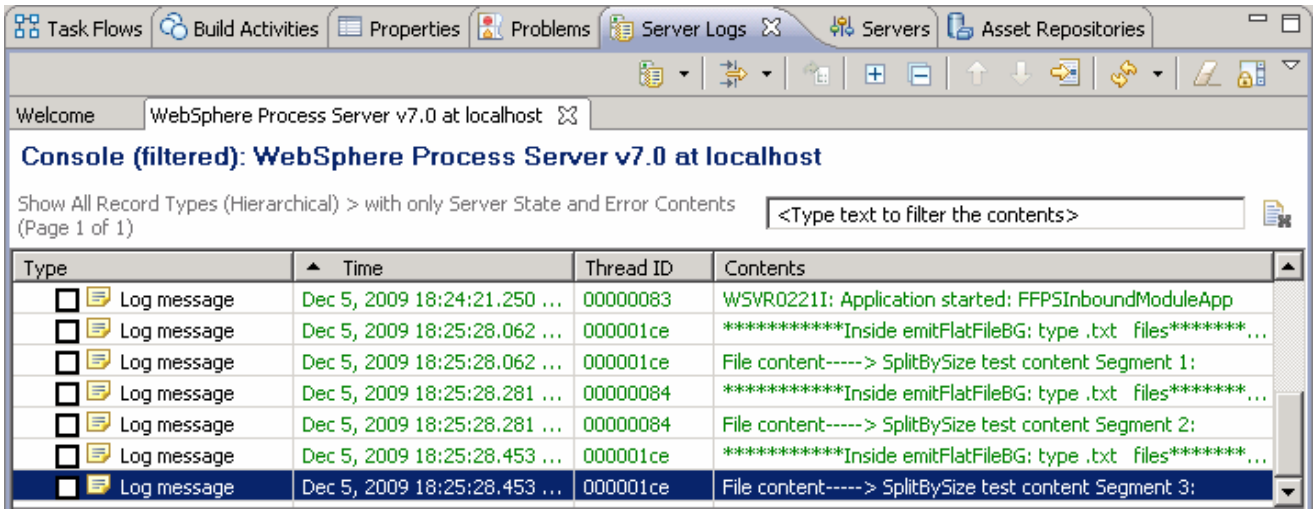
____ 4. Test the application with and input file **larger than 1000 Bytes**

Note: For your convenience, a test file **SplitBySize.txt** is placed in **<FFFILES>**. The size of the file is around 3KB.

__ a. Copy the **SplitBySize.txt** file from <FFFILES> to <EVENT_DIR>. The adapter will poll the copied file from the event directory and will transfer it to the archive directory

____ 5. Verify your results

__ a. Because you have placed a **.txt** file, it will pass through the **emitFlatFileBG** method and you should see this message in your Server Logs view



__ b. For full stack of message, open your SystemOut.log located at <WID_HOME>\pf\wps\logs\server1 and verify this message:

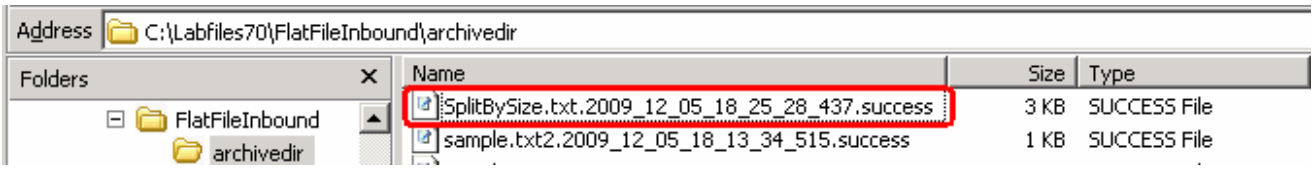
```
Console
WebSphere Process Server v6.1 [WebSphere Process Server] WebSphere Process Server v6.1 (WebSphere v6.1)
[1/9/08 13:10:05:250 CST] 0000008d SystemOut     O ************Inside emitFlatFileBG: type .txt files*************
[1/9/08 13:10:05:250 CST] 0000008d SystemOut     O File content-----> SplitBySize test content Segment 1:
**********************************************************************************************************************
**********************************************************************************************************************
**********************************************************************************************************************
**********************************************************************************************************************
**********************************************************************************************************************
**********************************************************************************************************************

[1/9/08 13:10:05:281 CST] 00000091 SystemOut     O ************Inside emitFlatFileBG: type .txt files*************
[1/9/08 13:10:05:281 CST] 00000091 SystemOut     O File content-----> SplitBySize test content Segment 2:
######################################################################################################################
######################################################################################################################
######################################################################################################################
######################################################################################################################
######################################################################################################################
######################################################################################################################

[1/9/08 13:10:05:312 CST] 00000092 SystemOut     O ************Inside emitFlatFileBG: type .txt files*************
[1/9/08 13:10:05:312 CST] 00000092 SystemOut     O File content-----> SplitBySize test content Segment 3:
3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
```

Note: The same SplitBySize.txt file, if delivered without specifying Split criteria to 1000 (about 1KB), will reach the end point only once. Here, in this part, you saw the successful event delivery message for three times for the same file (with the file size being 3KB) because of the split criteria that has been given as 1KB, which means it splits the file into three parts and delivers to the endpoint.

1) Check your **<ARCHIVE_DIR>,** which should contain an archive of the event file with the same file name appended with year, month, date, system time, and success

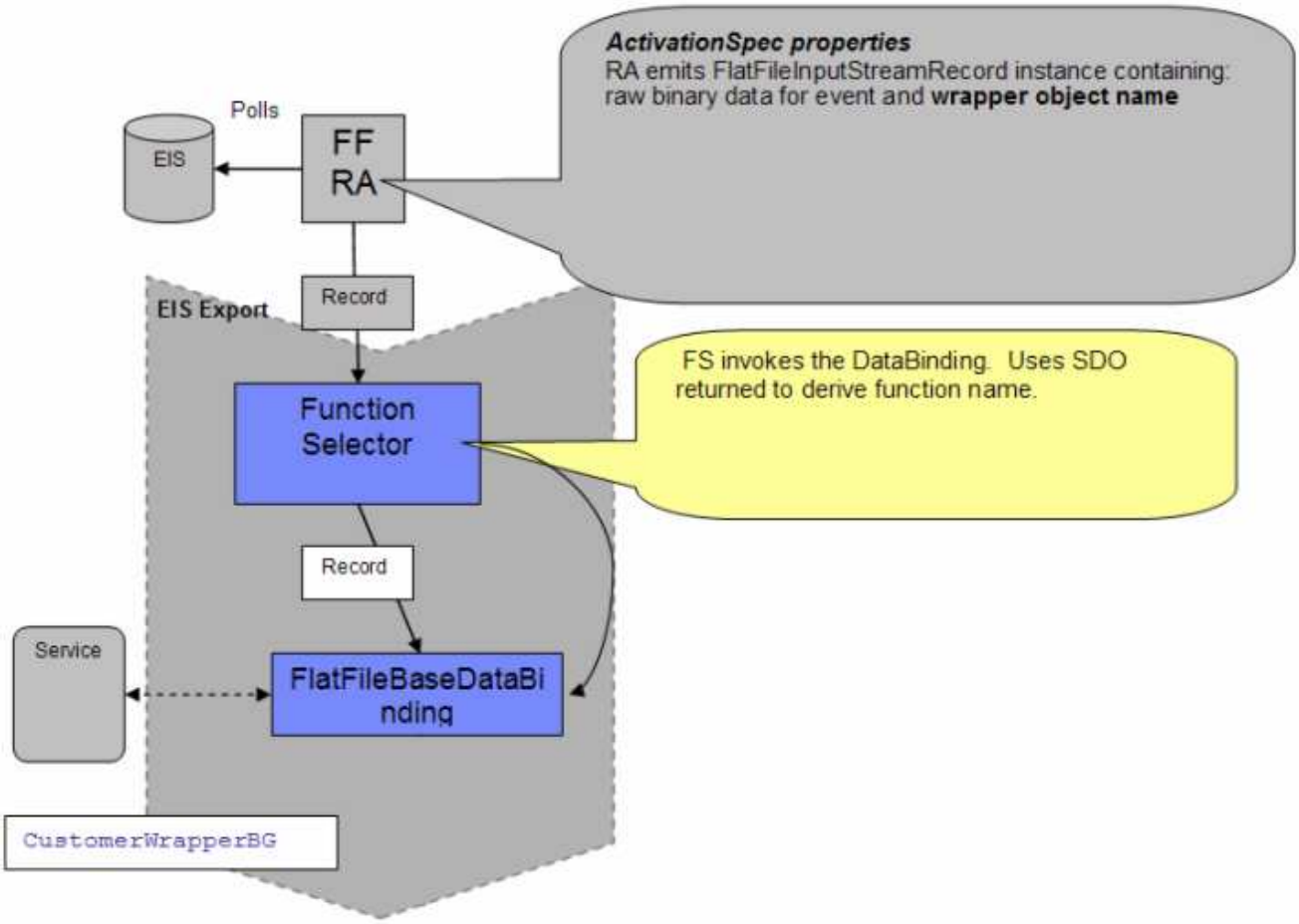


____ 6. Restore the Sever Configuration: follow the instructions under **Step 3** of **Part 2.3**

Part 3: Content specific (non-pass through) scenario

Of the two flows, you have just tested the pass through scenario that does not involve data transformation. In this part of the lab, you will configure the non-pass through scenario using the new External Service option from the WebSphere Integration Developer and then test the configuration with different cases.

Content specific flow for the inbound scenario:



- Event data is picked from the event file based on SplitCriteria, converted into a byte[], and set on FlatFileInputStreamRecord.
- Protocol specific information like event file name, directory name are also set in the FlatFileInputStreamRecord.
- The FlatFileInputStreamRecord is sent to the foundation class and from there to the function selector. The function selector instantiates the wrapper and then based on the data handler property value in the chosen databinding, the function selector will invoke that data binding and get back a content specific data object (Customer SDO).
- This content specific data object is set on the wrapper data object. Protocol specific information is also set the wrapper. The wrapper is set in a BG and sent to the end point.

3.1. Configure content specific (non-pass through) scenario using external service wizard

In this part of the lab you will use this new external service feature to create and configure the function selector, **data handler**, data binding, and other required artifacts to test the inbound pass through scenario

____ 1. Create FFCustomInboundModule

__ a. From the Business Integration window, right-click and select **New > Module**

__ b. From the New Module window, enter **FFCustomInboundModule** for the Module Name

__ c. Ensure that the box next to **Open module assembly diagram** is checked and then click **Finish**

You will now see a new module, FFCustomInboundModule, created from your Business Integration window

____ 2. Import required business objects

New in V7.0: Wrapper business objects for the business objects containing global elements are supported in this version. So, you can now pass the protocol specific information as part of each request.

__ a. Expand FFCustomInboundModule (if not already expanded), right-click **Data Types** and select **Import…** from the pop-up menu

__ b. From the Import window, expand **General** and select **File System** and then click **Next**

__ c. Enter From directory

1) Click **Browse…** next to **From directory**

2) From the Import from directory window, select **<FFFILES >** and click **OK**

Now, you will see FFFiles folder added on the left side, and all the xsds and files under that folder on the right side.

__ d. Select the box next to **Customer.xsd** and **Order.xsd**

__ e. Ensure that the **FFCustomInboundModule** is selected for Into folder

__ f. Click **Finish** from the Import window

The Business Integration window is updated with the imported business objects.

____ 3. Review imported business objects:

__ a. Expand **FFCustomInboundModule > Data Types** and you will now see two new data types **Customer** and **Order** under it.

__ b. Double-click **Customer** review the fields inside the object:

____ 4. Now, double-click Order and review the fields inside the object:

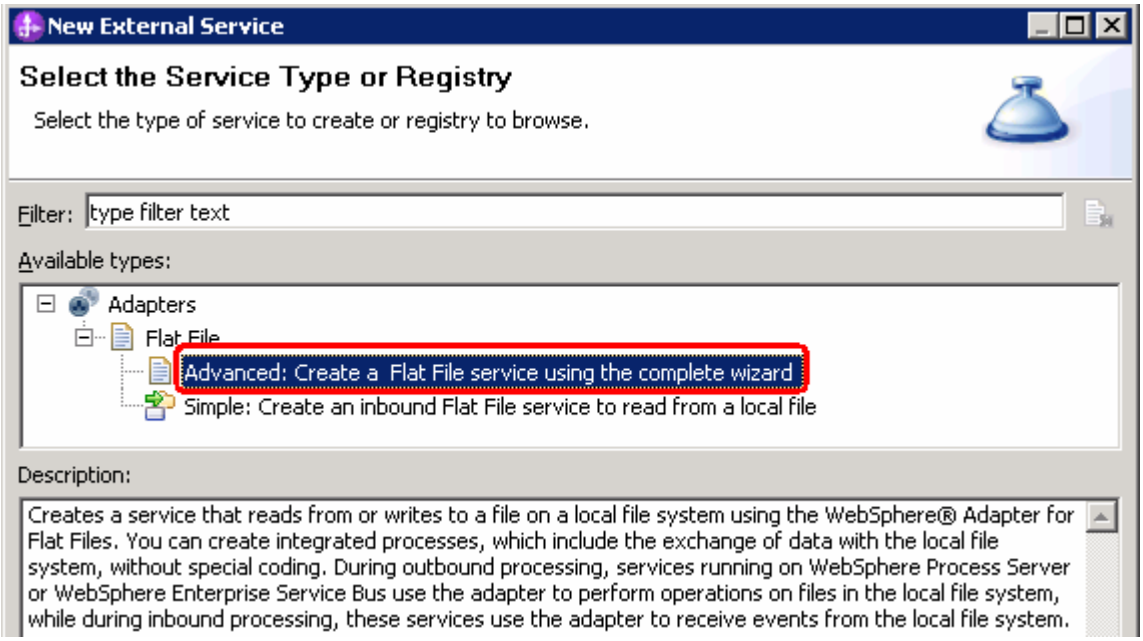
____ 5. After reviewing, close the Customer business object from the Assembly editor

____ 6. To start External Service from the Palette:

__ a. From the **Palette** on the left side of Assembly Diagram, click **Inbound Adapters**:

____ 7. Under Inbound Adapters, click the **Flat File** and then click the empty canvas of the assembly diagram. The New Flat File Service wizard is opened

____ 8. From the New External Service window, expand **Adapters > Flat File** and select **Advanced: Create a Flat File service using the complete wizard**

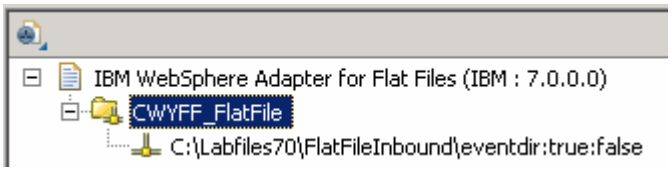


__ a. Click **Next**

Note: You can also start the External Service from the **File menu** option:

From the main menu, select **File > New > External Service**. This opens an external service wizard that helps you obtain a service that establishes connectivity with other systems. Select **Adapters > Flat File** and click **Next**

____ 9. On the Select an Adapter screen, expand **IBM WebSphere Adapter for Flat Files (IBM : 7.0.0.0)** and select **CWYFF_FlatFile**



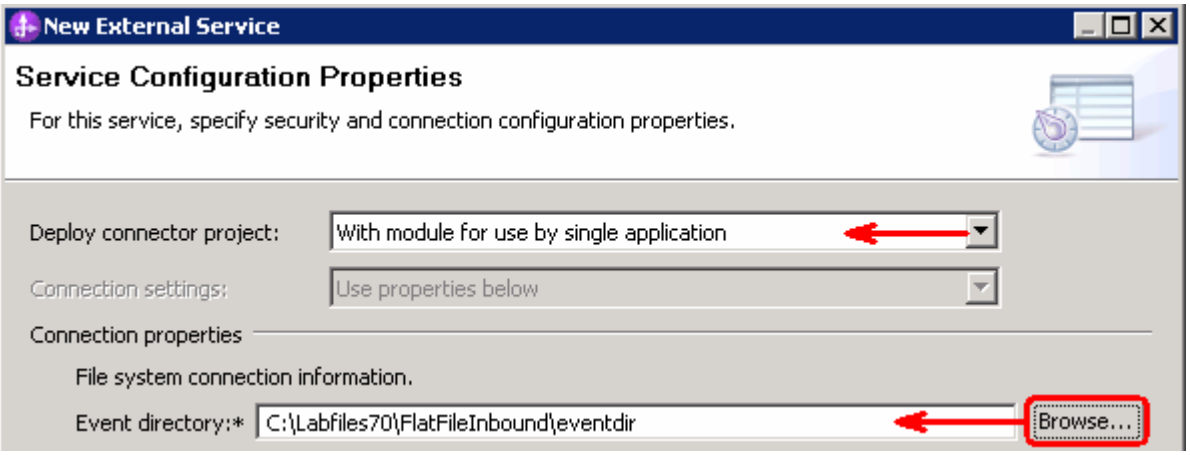
__ a. Click **Next**

Note: If you are using the **File menu** option to start the external service wizard, you are asked to select the **processing direction** at this point. Select the radio button next to **Inbound** and click **Next** to proceed to the next step.

____ 10. Service configuration properties:

__ a. Deploy connector project: ensure that the default option With module for use by single application is selected

__ b. Under Connection Configuration, click **Browse…** next to Event directory and select **<EVENT_DIR>** from the pop-up window:



__ c. Click **Advanced >>** to see the hidden advanced properties that can be configured:

You can click each of the configurations and review the options available under it. This lab guides you through some of the important and required configurations.

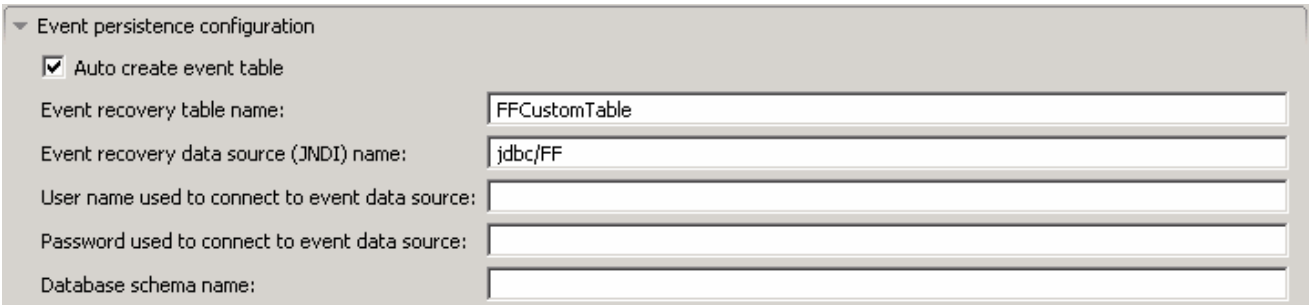
__ d. Event persistence configuration:

1) Ensure that the Auto create event table is checked

2) Event recovery table name: **FFCustomTable**

3) Event recovery data source (JNDI) name: **jdbc/FF**

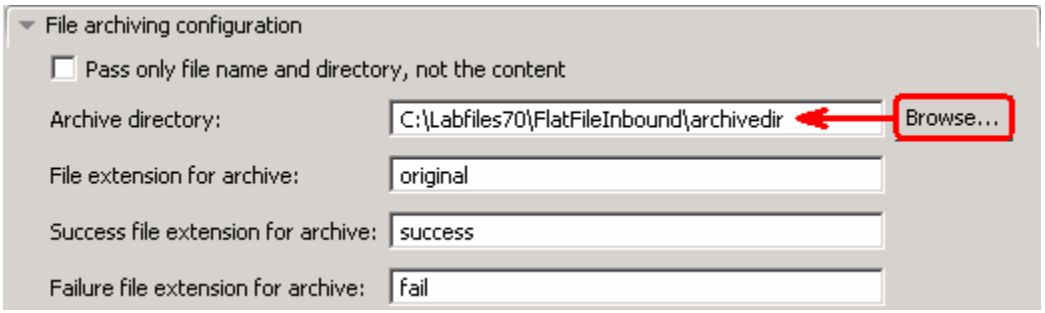
Note: This represents the JNDI name of the Data Source used by Event Persistence to get the JDBC database connection. The Data Source must be created in the WebSphere Process Server. You can use the same data source JNDI name that you created in Step 3 of Part 1



Note: The Event recovery data source (JNDI) name is **not mandatory** from V6.1. Now, the adapter can use **in-memory representation** of event table to store all the necessary information. Adapter uses this feature when event database information is not configured during inbound event polling. This feature will not support the capability of handling “Ensure once-only event delivery”.

__ e. File archiving configuration:

1) Archive directory: click **Browse…** and select **<ARCHIVE_DIR>**



Function selector configuration: Of the two function selectors that are supported by the FlatFile adapter, FilenameFunctionSelector was used for pass through scenario. The **EmbeddedNameFunctionSelector** is used in case of content-specific business objects, where the object name is embedded in the event file. For example, if the content-specific business object is CustomerWrapperBG, the function returned by the function selector is emitCustomer. This function selector should be configured with a data handler. The data binding should be the adapter-specific WrapperDataBinding. The data binding should be configured to use the same data handler configured with the function selector.

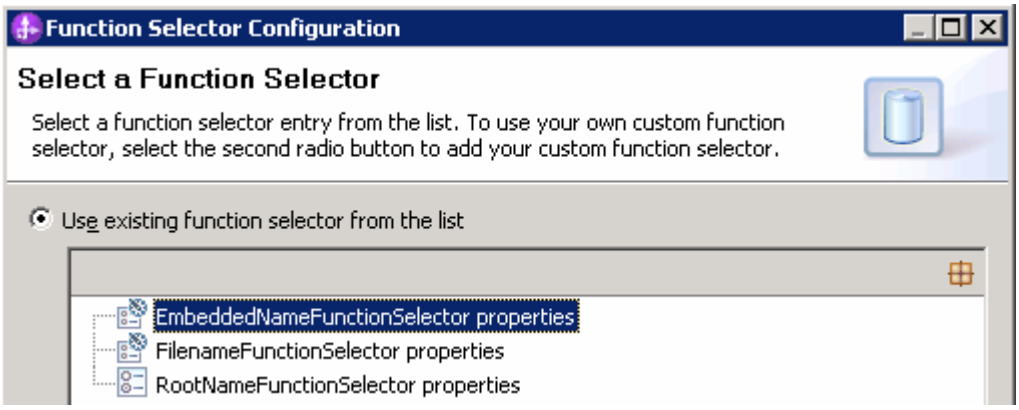
____ 11. Under Service properties, for Function selector options, select **Use a function selector configuration** from the drop down menu

---

Note: If you select **Use default function selector 'FilenameFunctionSelector'** option for the Function selector, the adapter automatically creates a function selector with FilenameFunctionSelector as the class name. For non-pass through scenario, you need to define a function selector that uses **EmbeddedNameFunctionSelector**.

__ a. Click **Select** next to Function selector. The Binding Resource Configuration window is opened

__ b. Under '**Use existing function selector from the list**', select **EmbeddedNameFunctionSelector properties**



__ c. Click **Next**

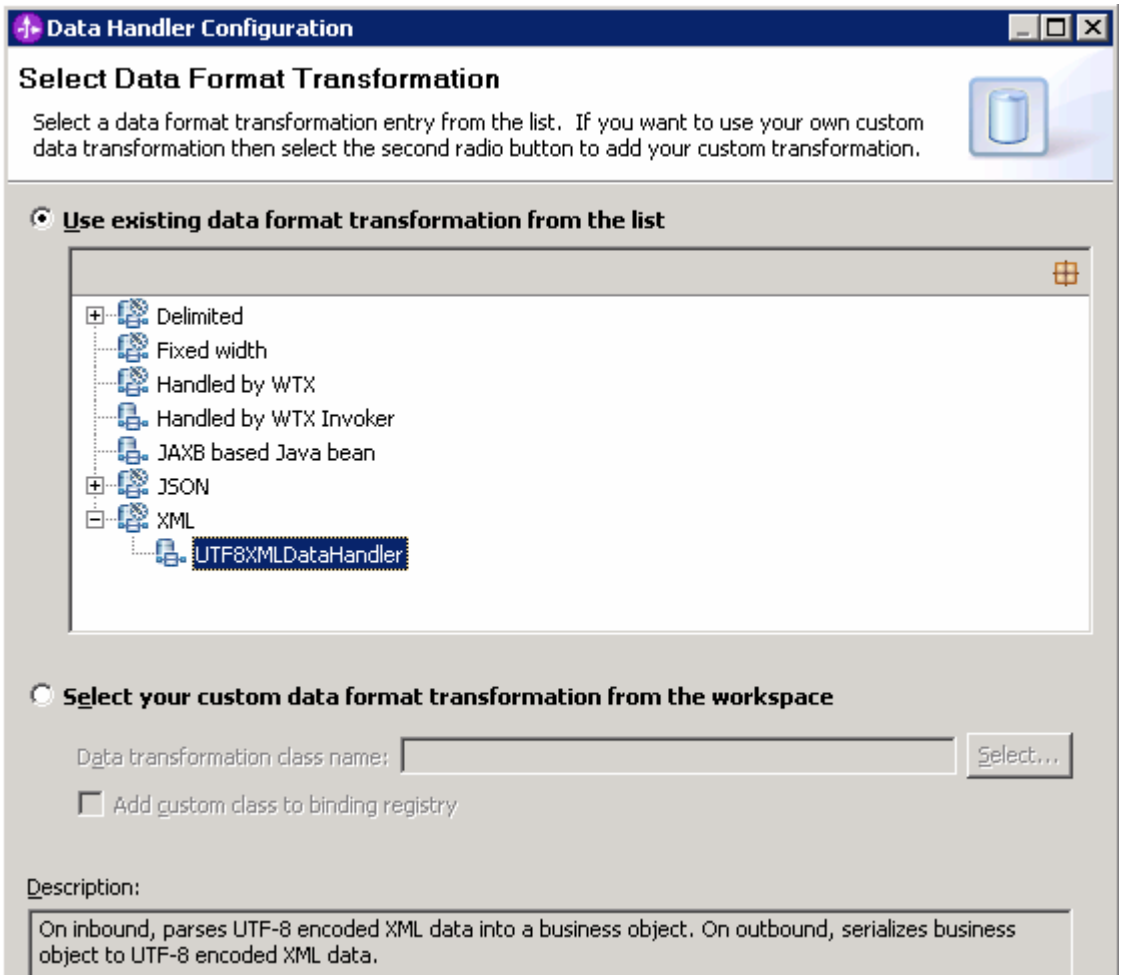
The following screen is Function Selector Properties screen where you will define the data handler and encoding used in inbound processing.

Data handler configuration:

__ d. Click **Select…** next to Data handler configuration name. A Binding Resource Configuration window is opened for you to define the data handler

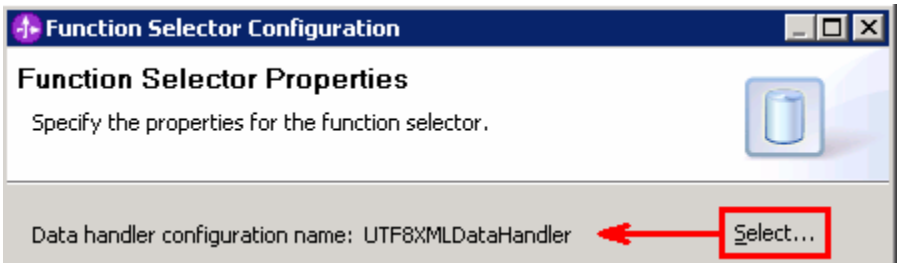
__ e. Under 'Use existing data format transformation from the list', select **XML > UTF8XMLDataHandler**

Note: UTF8XMLDataHandler listed under XML is the predefined data handler with UTF-8 as the encoding. You can also select XML and then select the encoding of your choice in the next screen to define a data handler of your choice.



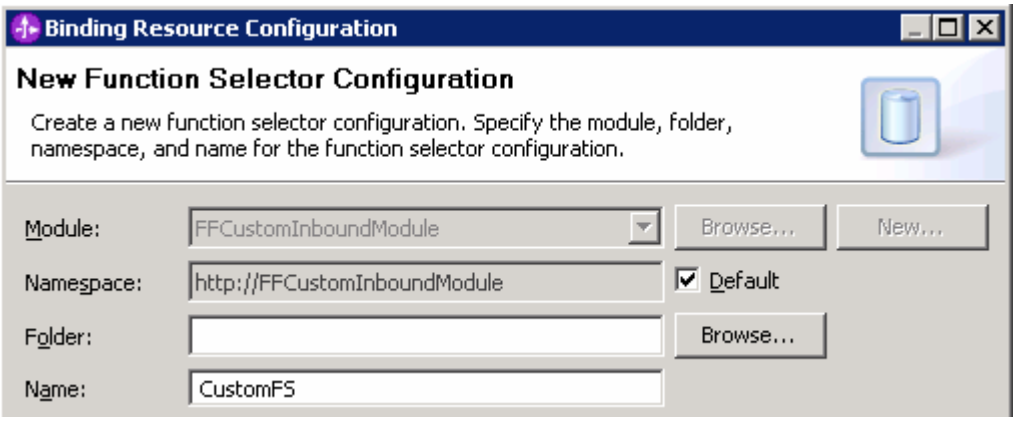
__ f. Click **Finish**

__ g. From the next screen, the Data handler configuration name that was selected in the previous steps - **UTF8XMLDataHandler** - is displayed



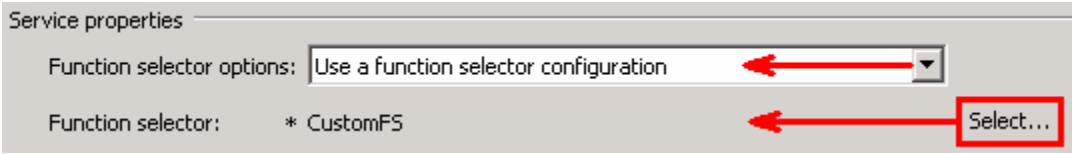
__ h. Click **Next**

__ i. From the New Function Selector Configuration screen, enter **CustomFS** for **Name**



__ j. Click **Finish**

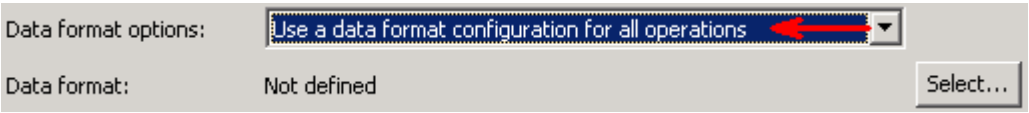
You are now done with configuring the function selector, CustomFS and, in that process, configured a data handler, CustomDH. You are back to the external service wizard and the configured function selector, CustomFS is displayed for function selector configuration field:



Data binding configuration:

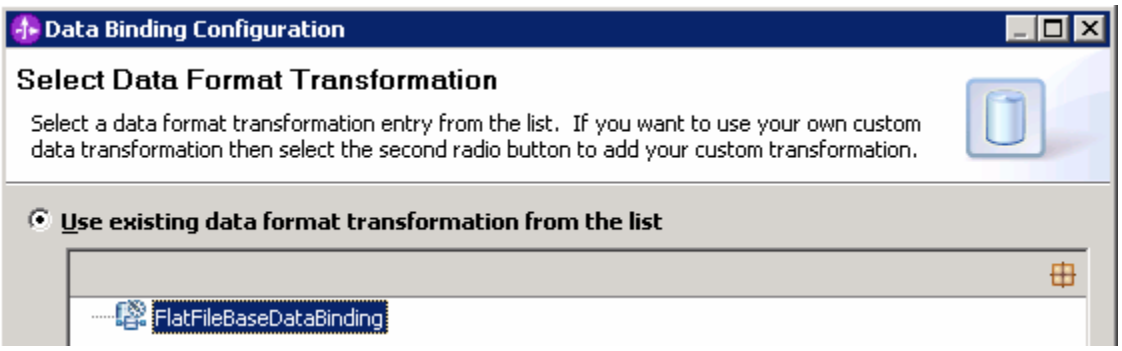
____ 12. You can define data binding in two places - service level (current screen of external service wizard) or later at the method level (Operations screen of the external service wizard). In this lab, you will define data binding at the service level (from this screen)

__ a. From the dropdown menu next to Data format options, select '**Use a data binding configuration for all operations**'



__ b. Click **Select…** next to **Data format**. A Binding Resource Configuration window is opened

__ c. Select the radio button for **'Use existing data format transformation from the list'** and then select **FlatFileBaseDataBinding**



__ d. Click **Next**

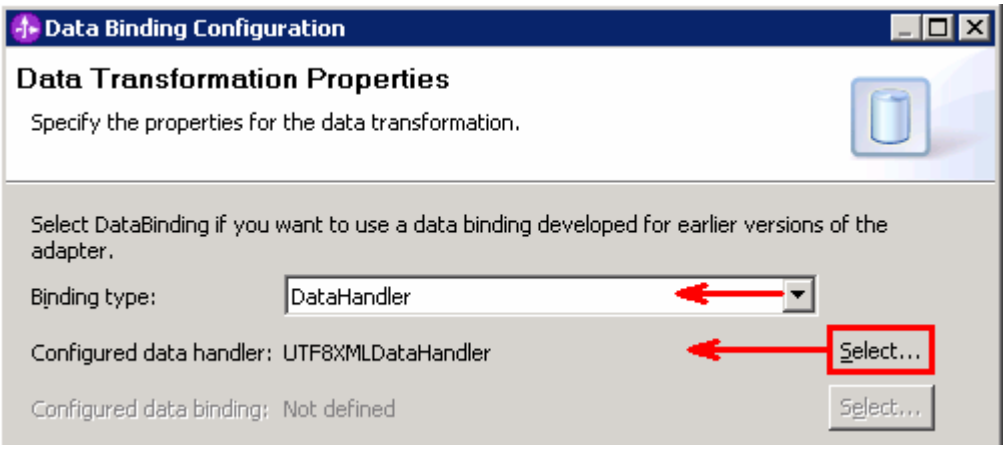
__ e. Select the **UTF8XMLDataHandler** for data handler:

1) Click **Select** next to 'Configured data handler'

2) From the Binding Resource Configuration window, select **XML > UTF8XMLDataHandler** listed under '**Use existing data format transformation from the list**'

3) Click **Finish**

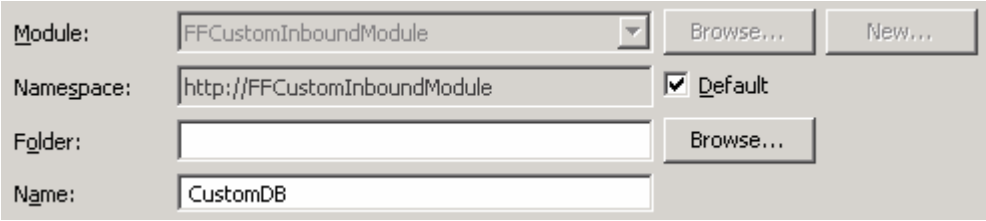
__ f. Back to Data Transformation Properties and the selected data handler '**UTF8XMLDataHandler**' is displayed here:



__ g. Click **Next**

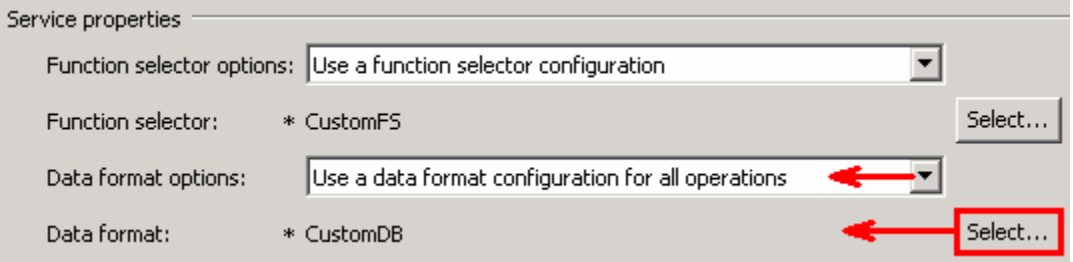
__ h. Note that the selected module is **FFCustomInboundModule**

1) For the **Name**, enter **CustomDB**



2) Click **Finish**

__ i. Now the **CustomDB** should be displayed for Data format



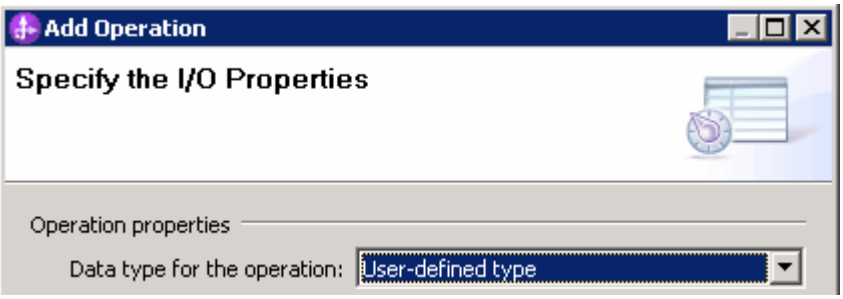
____ 13. Check the box next to **Change logging properties for wizard** to view the output location of the log file and the logging level and click **Next**

Define emitCustomerFile operation:

____ 14. From the Operations screen, click **Add…**

Add Operation window is opened

__ a. Select **User-defined type** for the Data type and click **Next**



You are now back to Operation window and because you chose the User defined data type, the Input type is **blank** and because you have selected Output required box, the Output type is **CreateResponse**

__ b. For Operation name, enter **emitCustomerFile**

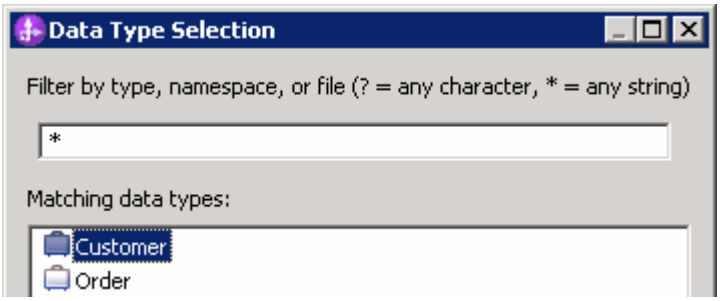
__ c. Define Input type:

1) Click **New…** next to **Input type** to open a New Business Object window

2) From this window, ensure that the Module selected is **FFCustomInboundModule** and click **Next**

3) Click **Browse…** next to Data type

4) From the Data Type Selection window, select **Customer** under Matching data types:

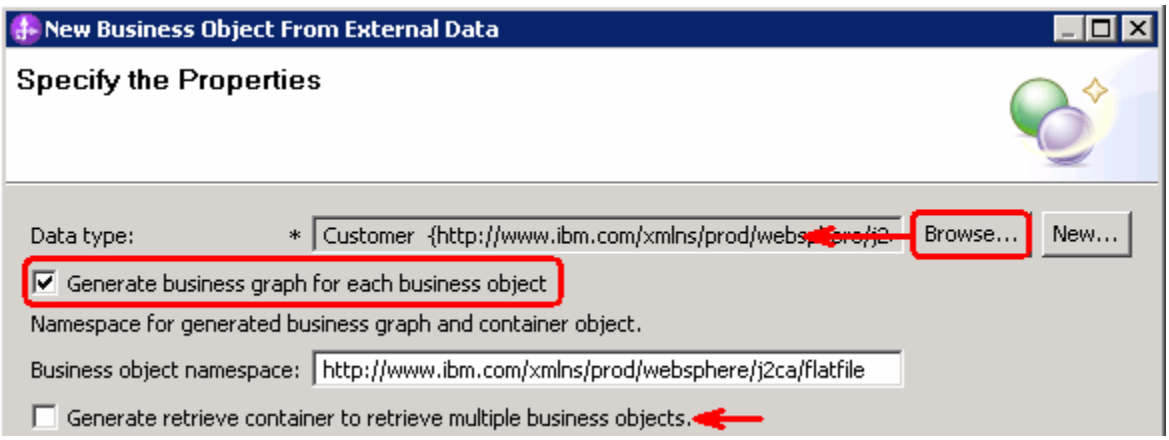


5) Click **OK**

6) From the wizard, **check** the box next to **Generate business graph for each business object**

7) **Do not** check the box for **'Generate retrieve container to retrieve multiple business objects'**

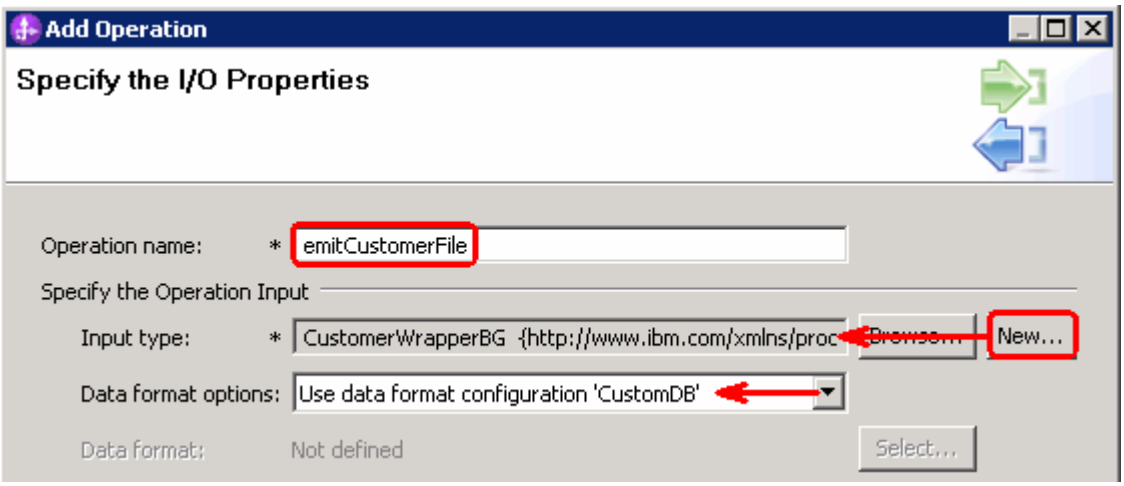
Note: The 'Generate retrieve container to retrieve multiple business objects' is used only during outbound retrieve operation.



8) Click **Finish**

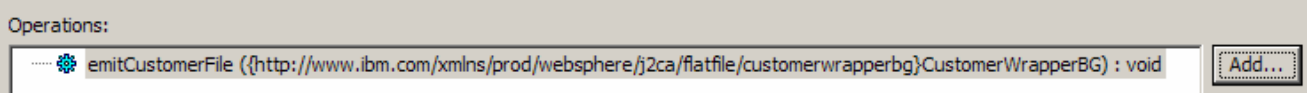
__ d. The Input type in Add Operation window is populated with CustomerWrapperBG, because you have selected to have business graph (BG) generated

__ e. For **Data format options**, accept the default selection **Use data format configuration 'CustomDB'** from the dropdown list



__ f. Click **Finish**

__ g. The defined operation, emitCustomerFile, is populated in the Operations list



Define emitOrderFile operation:

____ 15. From the Operations screen, click **Add…**

Add Operation window is opened

__ a. Select **User defined type** for the Data type and click **Next**

You are back to Operation window and because you chose the User defined data type, the Data type is blank.

__ b. For Operation name, enter **emitOrderFile**

__ c. Define Data type:

1) Click **New…** next to **Input type** to open a new window

2) From the next screen, ensure that **FFCustomInboundModule** is selected and click **Next**

3) Click **Browse…** next to Data type

4) From the Data Type Selection window, select **Order** under Matching data types:

5) Click **OK**

6) **Do not** check the box next to **Generate business graph for each business object**

7) **Do not** check the box for **'Generate retrieve container to retrieve multiple business objects'**

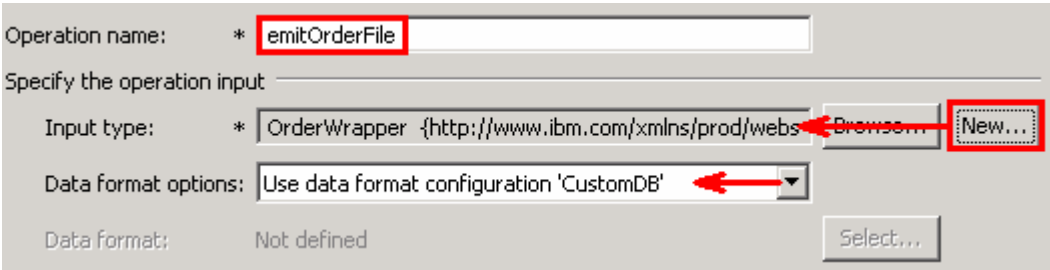
Note: From V6.1, the generation of business graph is optional and you can leave this option unchecked. As a result, Adapter will not generate the BG for Order business object and you can confirm this bye reviewing the generated data types from the business integration view of your WebSphere Integration Developer.

Note: The 'Generate retrieve container to retrieve multiple business objects' is used only during outbound retrieve operation.

8) Click **Finish**

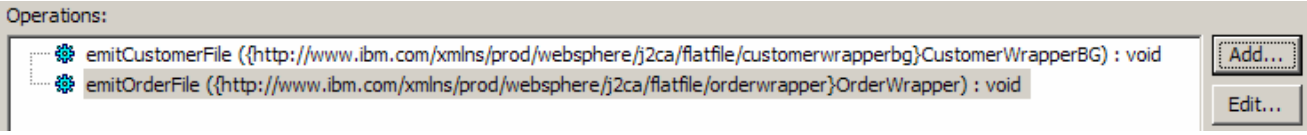
__ d. The Input Type in Add Operation window is populated with **OrderWrapper**, because you have **not** selected to have business graph (BG) generated

__ e. For **Data format options**, accept the default selection **Use data format configuration 'CustomDB'** from the dropdown list



__ f. Click **Finish**

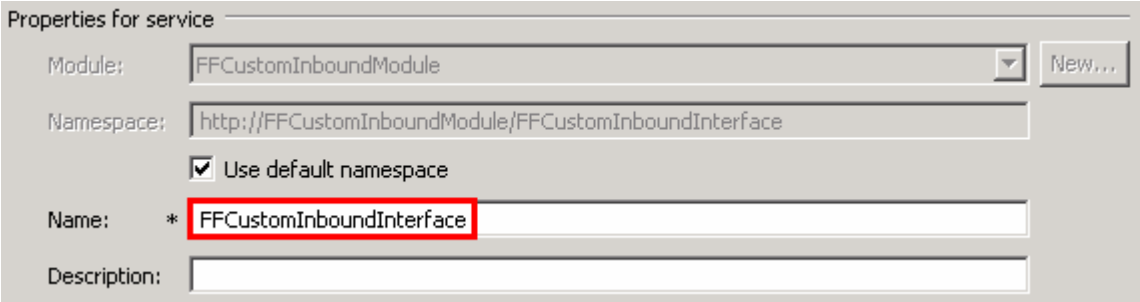
__ g. The defined operation, emitOrderFile, is populated in the Operations list



__ h. Click **Next**

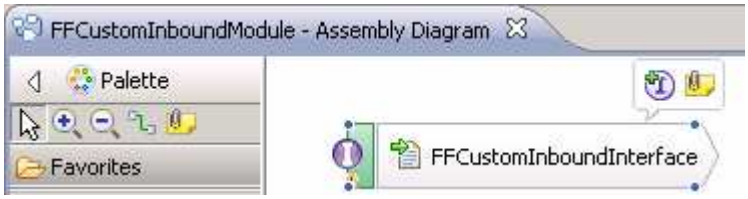
____ 16. From the Generate Artifacts screen, enter these:

__ a. For Name, enter **FFCustomInboundInterface**



__ b. Click **Finish**

____ 17. You will now see a new export component, **FFCustomInboundInterface** in the assembly diagram of FFCustomInboundModule



__ a. Save (**Ctrl+S)** your changes to the assembly diagram

____ 18. Review the generated Native method bindings for the defined operations:

__ a. Ensure that the FFCustomInboundInterface is selected from the Assembly diagram

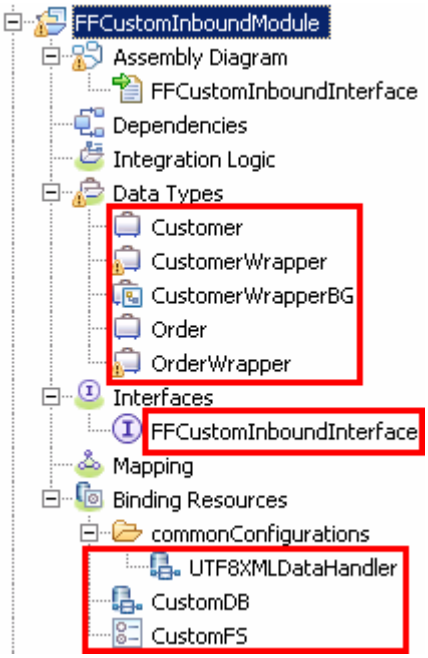
__ b. From the bottom panel, select **Properties > Binding > Method bindings**

Note: The Native method name should be 'emit' added as a prefix to the business object name, that is., **Native method = emit + Business Object name**. In this case, for the processed business object is **Customer**, the Native method is **emitCustomer** and for **Order** it is **emitOrder**.

__ c. From the Bound Methods list, click **emitCustomerFile** and you should see **emitCustomer** as the **Native method**

__ d. Now, click **emitOrderFile** from Bound Methods and you should see **emitOrder** as the **Native method**

____ 19. Review FFCustomInboundModule: The generated **Data Types**, **Interface**, and the Function selector (**CustomFS**) Data binding (**CustomDB**), and Data handler (**UTF8XMLDataHandler**) under Configured Resources can be found inside FFCustomInboundModule

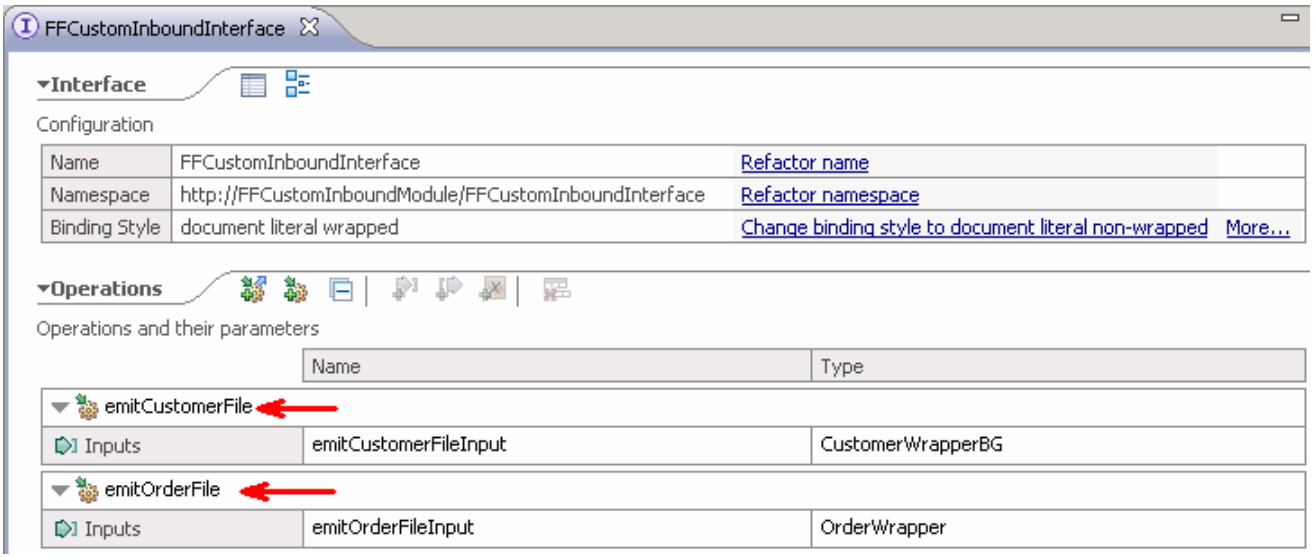


You can open each of these generated artifacts and business objects and review the properties inside.

Review the created methods inside the interface:

__ a. From the Business Integration view, expand FFCustomInboundModule > Interfaces and then double-click **FFCustomInboundInterface** to open it

__ b. You should see these two operations:



__ c. Close the interface, FFCustomInboundInterface

3.2. Add Java component

In this part of the lab, you will add a Java component to the module and then wire the export to the added component. Then you will add the custom Java code to the added module.

____ 1. Open the assembly diagram for FFCustomInboundModule, if it is already not open

__ **a.** From the business integration view, expand **FFCustomInboundModule** and double click **Assembly diagram**

____ 2. Drop a Java component to onto the assembly diagram

__ a. From the Palette, click Components to expand it

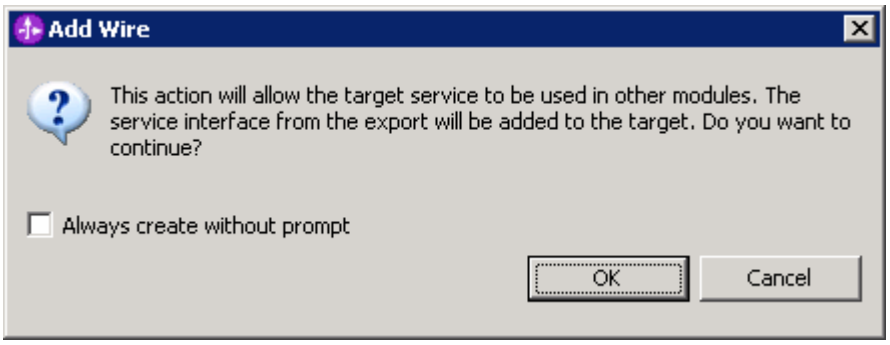
__ b. Click **Java** and then click the empty space of FFCustomInboundModule assembly diagram. This will place a new component, Component1 on the assembly diagram.

____ 3. Wire the FFCustomInboundInterface to the Component1

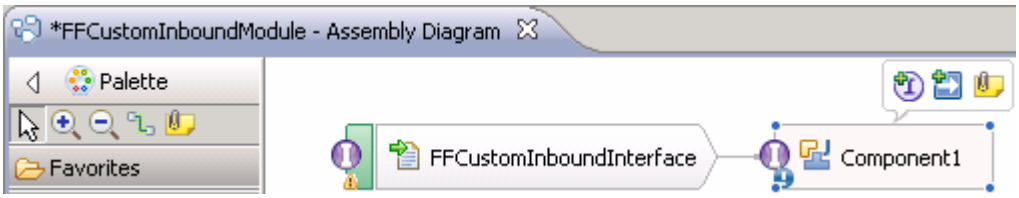
__ a. Select the **wire** ( ) icon from the Palette

__ b. Click **FFCustomInboundInterface** and then click **Component1** to wire them together

__ c. Select **OK** for the Add Wire pop-up window:



Your assembly diagram for FFCustomInboundModule will look like this:



__ d. From the top of the Palette, click the **Selection Tool** icon ( ) to get back to the normal cursor mode

____ 4. Generate Java Implementation

__ a. Right-click **Component1** and select **Generate Implementation** from the pop-up menu

__ b. On the **Generate Implementation** panel, select default package, and click **OK**

__ c. **Component1Impl.java** is opened in Assembly editor. Scroll down to the method **emitCustomerFile(DataObject emitCustomerFileInput)** that needs to be implemented and add this code under that method:

```
System.out.println("*************ENDPOINT emitCustomer**************");
DataObject wrapper =
emitCustomerFileInput.getDataObject("CustomerWrapper");
String filename = wrapper.getString("fileName");
System.out.println("FILENAME : "+filename);
DataObject customer = wrapper.getDataObject("Content");
String name = customer.getString("customerName");
System.out.println("NAME-----> "+name);
String address = customer.getString("Address");
System.out.println("ADDRESS--> "+address);
String city = customer.getString("City");
System.out.println("CITY-----> "+city);
String state = customer.getString("State");
System.out.println("STATE----> "+state);
```

Note: The code is also available at **<FFFILES>\CustomerJavaCode.txt** for your convenience

---

__ d. Scroll down to the method **emitOrderFile(DataObject emitOrderFileInput)** and add this code:

```
System.out.println("*************ENDPOINT emitOrder**************");
System.out.println("FILENAME :
"+emitOrderFileInput.getString("fileName"));
DataObject order = emitOrderFileInput.getDataObject("Content");
String OrderNumber = order.getString("OrderNumber");
System.out.println("ORDER NUMBER-----> "+OrderNumber);
String OrderType = order.getString("OrderType");
System.out.println("ORDER TYPE--> "+OrderType);
String Quantity = order.getString("Quantity");
System.out.println("QUANTITY-----> "+Quantity);
String Price = order.getString("Price");
System.out.println("PRICE----> "+Price);
```

Note: The code is also available at **<FFFILES>\OrderJavaCode.txt** for your convenience

---

__ e. Save (**Ctrl + S**) and close Component1Impl.java

____ 5. Save (**Ctrl + S**) and close Assembly diagram: FFCustomInboundModule

3.3. Test non pass through scenario

In this part of the lab, you will use the WebSphere Process Server Test Environment to test the SCA application Inbound processing for the non pass through Scenario with input file having a single business object.

____ 1. Add the project to the WebSphere Test Environment server

__ a. Right-click **WebSphere Process Server v7.0** under the Servers view and select **Add and remove projects…** from the pop-up menu

__ b. From the Add and Remove Projects window, select **FFCustomInboundModuleApp** under Available projects panel and click **Add >**

__ c. You will now see the **FFCustomInboundModuleApp** added to the **Configured projects**

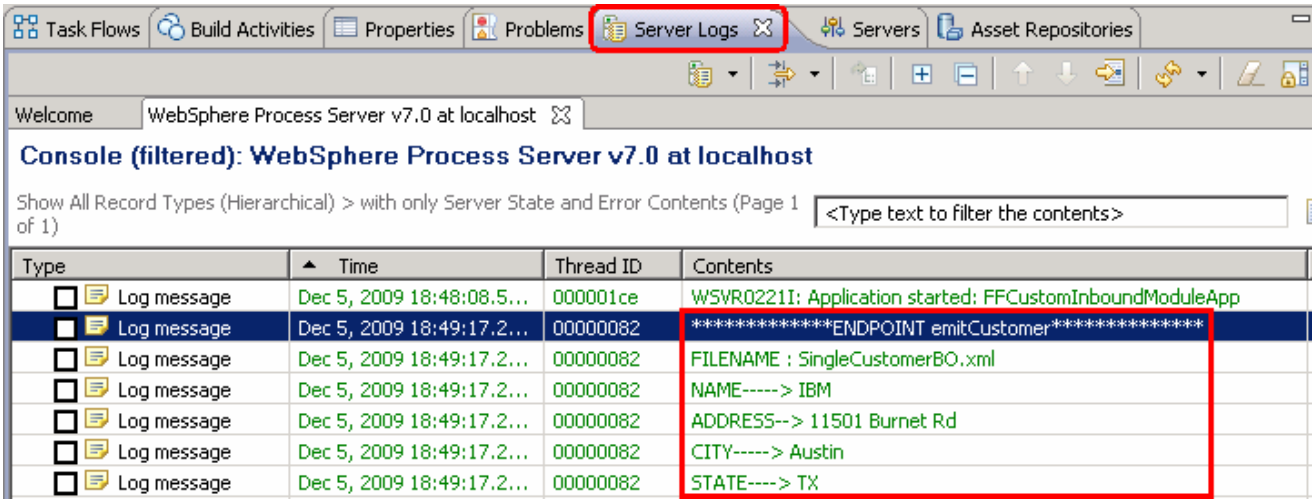
__ d. Click **Finish** and wait until the project is being published onto the server. The server will start in Debug mode if it is not already started before

____ 2. Put the input files in the event directory

Note: For your convenience, the test files **SingleCustomerBO.xml, SingleOrderBO.xml** are placed in **<FFFILES>.**

__ a. Copy the **SingleCustomerBO.xml** file from <FFFILES> to <EVENT_DIR>. The adapter will poll the copied xml file from the event directory and will transform it to the archive directory

__ b. Check your **Server Logs** view (or Systemout.log file) for this successful message:



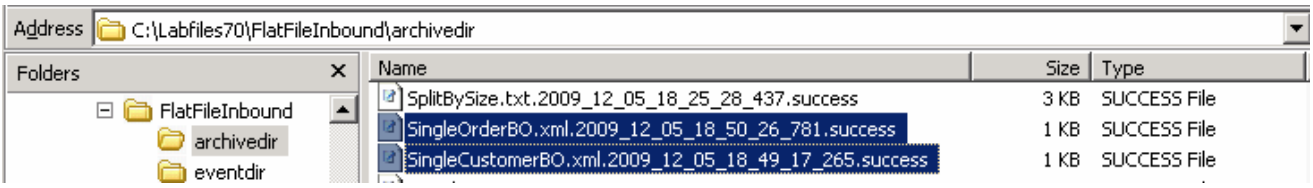
__ c. Now copy the **SingleOrderBO.xml** file from <FFFILES> to <EVENT_DIR>. The adapter will poll the copied xml file from the event directory and will transform it to the archive directory

__ d. Check your **Server Logs** view (or SystemOut.log) for this message:

Type	Time	Thread ID	Contents
Log message	Dec 5, 2009 18:49:17.2...	00000082	**********ENDPOINT emitCustomer**********
Log message	Dec 5, 2009 18:49:17.2...	00000082	FILENAME : SingleCustomerBO.xml
Log message	Dec 5, 2009 18:49:17.2...	00000082	NAME-----> IBM
Log message	Dec 5, 2009 18:49:17.2...	00000082	ADDRESS--> 11501 Burnet Rd
Log message	Dec 5, 2009 18:49:17.2...	00000082	CITY-----> Austin
Log message	Dec 5, 2009 18:49:17.2...	00000082	STATE----> TX
Log message	Dec 5, 2009 18:50:26.8...	00000083	**********ENDPOINT emitOrder**********
Log message	Dec 5, 2009 18:50:26.8...	00000083	FILENAME : SingleOrderBO.xml
Log message	Dec 5, 2009 18:50:26.8...	00000083	ORDER NUMBER-----> ABC12345
Log message	Dec 5, 2009 18:50:26.8...	00000083	ORDER TYPE--> BULK
Log message	Dec 5, 2009 18:50:26.8...	00000083	QUANTITY-----> 500
Log message	Dec 5, 2009 18:50:26.8...	00000083	PRICE----> 26.59

____ 3. You can also verify the results by reviewing the archive directory

__ a. Check the **<ARCHIVE_DIR>** subdirectory, which should contain two archives of the event files with the same file name appended with year, month, date, system time, and success



____ 4. Restore the Sever Configuration

__ a. Right-click **WebSphere Process Server v7.0** under the Servers view and select **Add and remove projects…** from the pop-up menu

__ b. Select **FFCustomInboundModuleApp** under Configured projects and click **< Remove**

__ c. Click **Finish** after you see the application moved to Available projects. Wait until the application is being unpublished

3.4. Test non-pass through scenario with SplitByDelimiter

In this last part of the lab, you will use the WebSphere Process Server Test Environment to test the SCA application Inbound processing for the non pass through Scenario with input file having multiple business objects separated by a delimiter.

____ 1. Configure/change the adapter properties from the Properties view:

__ a. Change to Business Integration perspective if you are in a different perspective

1) Select **Window > Open Perspective > Other…**

2) From the Select Perspective window, select **Business Integration (default)** and click **OK**

__ b. Expand **FFCustomInboundModule** and double-click **FFCustomInboundModule** to open it in Assembly Editor

__ c. Click **FFCustomInboundInterface** from the Assembly Editor and select **Properties** tab from the bottom

__ d. Select **Binding** under Properties and select **End-point configuration** under Binding itself and then select the **Connection**

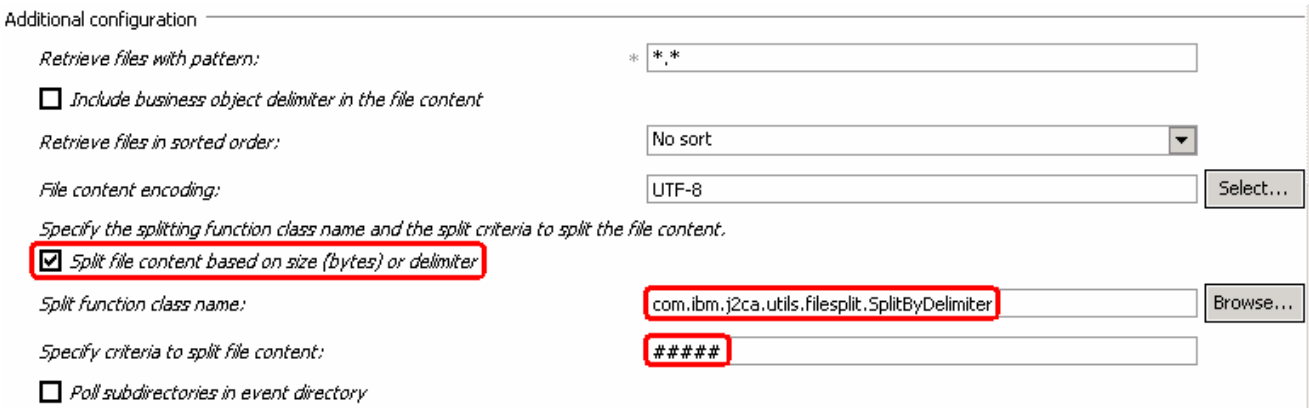
__ e. Now click **Advanced>>** button at the bottom. You can see all the properties that were given during the external service wizard

__ f. Scroll down to Additional configurations part and enter these:

1) Select the box for 'Split file content based on size (bytes) or delimiter

2) Split function class name: **com.ibm.j2ca.utils.filesplit.SplitByDelimiter** (you can Browse… and select)

3) Specify criteria to split file content: #####



__ g. Click Assembly diagram (or any where else so that the save button is enabled) and then save (**Ctrl +S**) your changes

____ 2. Repeat Step 1 of Part 3.3 to add the saved project FFCustomInboundModuleApp to the server

____ 3. Test the input file **CustomerSplitByDelimiter.xml**

Note: For your convenience, a test file **CustomerSplitByDelimiter.xml** is placed in **<FFFILES>**. The file contains two Customer Business Objects separated by the delimiter #####.

__ a. Copy the **CustomerSplitByDelimiter.xml** file from <FFFILES> to <EVENT_DIR>. The adapter will poll the copied xml file from the event directory and will transform it to the archive directory

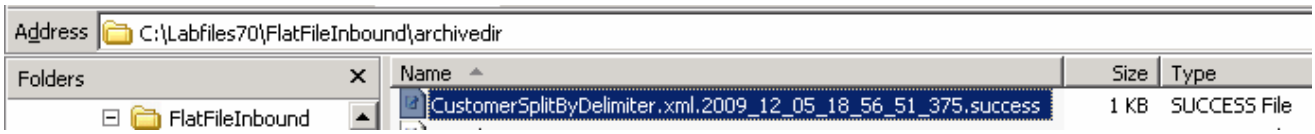
__ b. Verify your results

1) Check your **Server Logs** view (or Systemout.log file) for this successful message:

Note: You will see the successful event delivery message twice as there were two Business Objects present in the event file separated by the delimiter #####.

Type	Time	Thread ID	Contents
Log message	Dec 5, 2009 18:56:30.8...	00000083	WSVR0221I: Application started: FFCustomInboundModuleApp
Log message	Dec 5, 2009 18:56:51.2...	000001c8	**********ENDPOINT emitCustomer**********
Log message	Dec 5, 2009 18:56:51.2...	000001c8	FILENAME : CustomerSplitByDelimiter.xml
Log message	Dec 5, 2009 18:56:51.2...	000001c8	NAME-----> IBM
Log message	Dec 5, 2009 18:56:51.2...	000001c8	ADDRESS--> 11501 Burnet Rd
Log message	Dec 5, 2009 18:56:51.2...	000001c8	CITY-----> Austin
Log message	Dec 5, 2009 18:56:51.2...	000001c8	STATE----> TX
Log message	Dec 5, 2009 18:56:51.3...	00000193	**********ENDPOINT emitCustomer**********
Log message	Dec 5, 2009 18:56:51.3...	00000193	FILENAME : CustomerSplitByDelimiter.xml
Log message	Dec 5, 2009 18:56:51.3...	00000193	NAME-----> MBI
Log message	Dec 5, 2009 18:56:51.3...	00000193	ADDRESS--> RTP
Log message	Dec 5, 2009 18:56:51.3...	00000193	CITY-----> Raleigh
Log message	Dec 5, 2009 18:56:51.3...	00000193	STATE----> NC

2) Check your **<ARCHIVE_DIR>,** which should contain an archive of the event file with the same file name appended with year, month, date, system time, and success



____ 4. Test the input file **OrderSplitByDelimiter.xml**:

Note: For your convenience, the test file **OrderSplitByDelimiter.xml** is placed in **<FFFILES>.** The file contains two Order Business Objects separated by the delimiter #####.

__ a. Copy the **OrderSplitByDelimiter.xml** file from <FFFILES> to <EVENT_DIR>. The adapter will poll the copied xml file from the event directory and will transform it to the archive directory

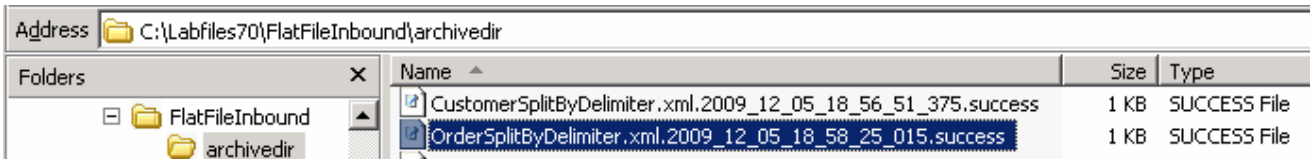
__ b. Verify the results for OrderSplitByDelimiter.xml file:

1) Check your **Server Logs** view (or Systemout.log file) for this successful message:

Note: You will see the successful event delivery message twice as there were two Business Objects present in the event file separated by the delimiter #####.

Type	Time	Thread ID	Contents
Log message	Dec 5, 2009 18:58:24.9...	00000083	**********ENDPOINT emitOrder**********
Log message	Dec 5, 2009 18:58:24.9...	00000083	FILENAME : OrderSplitByDelimiter.xml
Log message	Dec 5, 2009 18:58:24.9...	00000083	ORDER NUMBER-----> ABC12345
Log message	Dec 5, 2009 18:58:24.9...	00000083	ORDER TYPE--> BULK
Log message	Dec 5, 2009 18:58:24.9...	00000083	QUANTITY-----> 500
Log message	Dec 5, 2009 18:58:24.9...	00000083	PRICE----> 26.59
Log message	Dec 5, 2009 18:58:25.0...	00000193	**********ENDPOINT emitOrder**********
Log message	Dec 5, 2009 18:58:25.0...	00000193	FILENAME : OrderSplitByDelimiter.xml
Log message	Dec 5, 2009 18:58:25.0...	00000193	ORDER NUMBER-----> XYZ987
Log message	Dec 5, 2009 18:58:25.0...	00000193	ORDER TYPE--> SINGLE
Log message	Dec 5, 2009 18:58:25.0...	00000193	QUANTITY-----> 1000
Log message	Dec 5, 2009 18:58:25.0...	00000193	PRICE----> 56.67

2) Check your **<ARCHIVE_DIR>,** which should contain an archive of the event file with the same file name appended with year, month, date, system time, and success



____ 5. Restore the Sever Configuration

__ a. Right-click **WebSphere Process Server v7.0** under the Servers view and select **Add and remove projects…** from the pop-up menu

__ b. Select **FFCustomInboundModuleApp** under Configured projects and click **< Remove**

__ c. Click **Finish** after you see the application moved to Available projects. Wait until the application is being unpublished

Part 4: Use default function selector and data binding

This part of the lab will show you how to use the default use the default function selector and data binding options from the external service wizard and generate other required artifacts.

When you use the default function selector, you cannot define the rules as you did in Part 2 and hence there will only be one method that handles all types of files.

When you use the default data binding, you cannot have multiple data types as in Part 3 and each data type is handled by different method. Instead, there will only be one method and one data type.

After running the external service wizard, you will add the required Java component with implementation and then will continue to test the adapter.

4.1. Configure inbound using default function selector and data binding

In this part of the lab you will use the default function selector and data binding options from the external service wizard and generate other required artifacts to test the inbound scenario.

____ 1. Create FFDefaultsInboundModule

__ a. From the Business Integration window, right-click and select **New > Module**

__ b. From the New Module window, enter **FFDefaultsInboundModule** for the Module Name

__ c. Ensure that the box next to **Open module assembly diagram** is checked and then click **Finish**

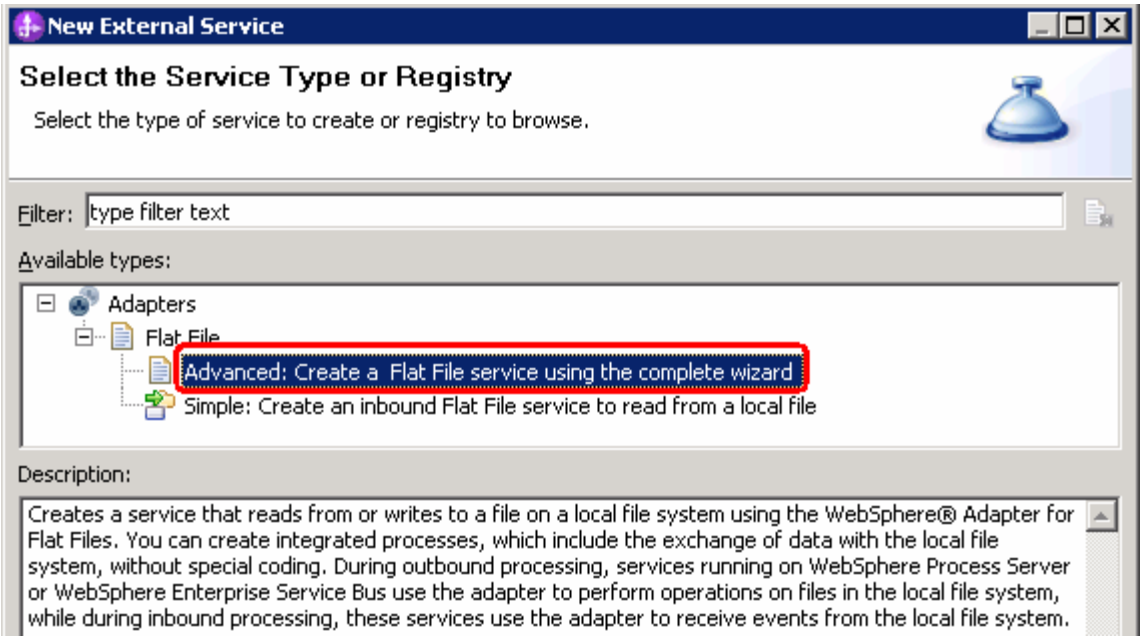
You will now see a new module, FFDefaultsInboundModule, created from your Business Integration window

____ 2. To start an external service from the Palette:

__ a. From the **Palette** on the left side of Assembly Diagram, click **Inbound Adapters**:

__ b. Under Inbound Adapters, click the **Flat File** and then click the empty canvas of the assembly diagram. The New Flat File Service wizard is opened

____ 3. From the New External Service window, expand **Adapters > Flat File** and select **Advanced: Create a Flat File service using the complete wizard**



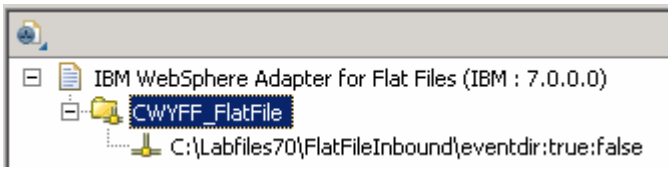
__ a. Click **Next**

Note: You can also start the external service from the **File menu** option:

From the main menu, select **File > New > External Service**. This opens an external service wizard that helps you obtain a service that establishes connectivity with other systems.

Select **Adapters > Flat File** and click **Next**.

____ 4. On the Select an Adapter screen, expand **IBM WebSphere Adapter for Flat Files (IBM : 7.0.0.0)** and select **CWYFF_FlatFile**

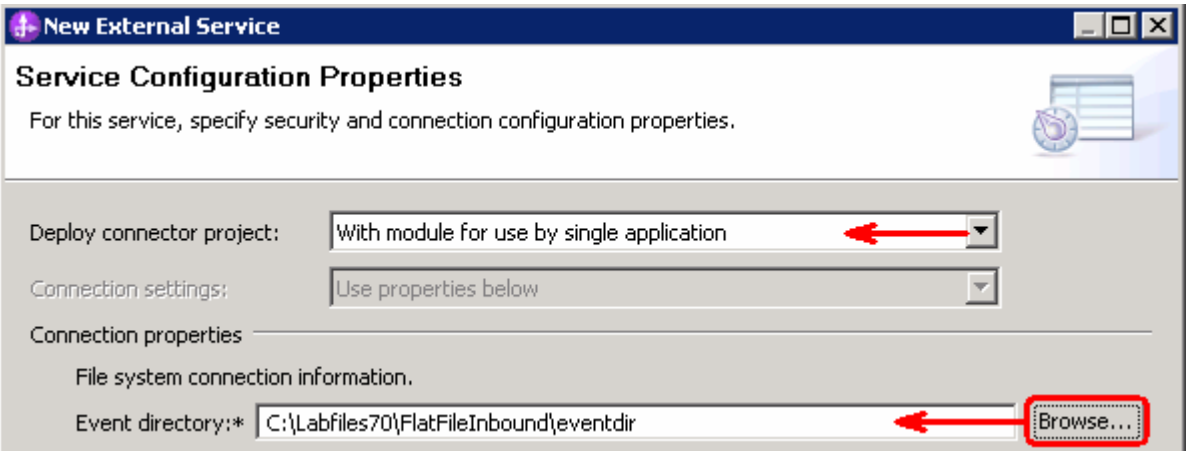


__ a. Click **Next**

____ 5. Service Configuration Properties:

__ a. Deploy connector project: ensure that the default option With module for use by single application is selected

__ b. Under Connection Configuration, click **Browse…** next to Event directory and select **<EVENT_DIR>** from the pop-up window:



__ c. Click **Advanced >>** to see the hidden advanced properties that can be configured:

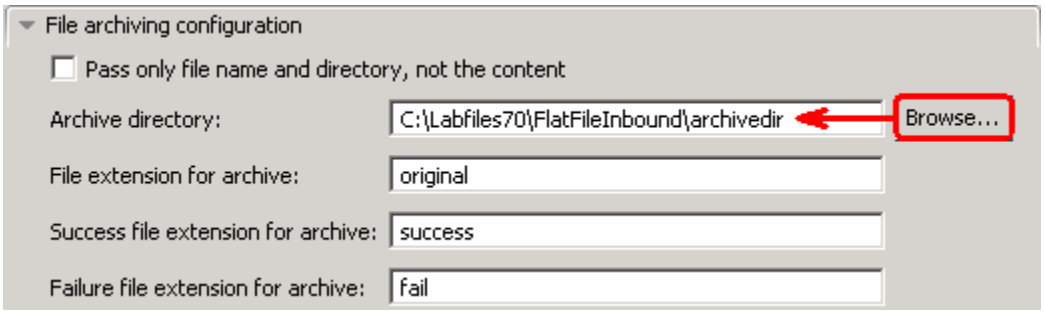
You can click each of the configurations and review the options available under it. This lab guides you through some of the important and required configurations.

__ d. Event persistence configuration: In this part of the lab, you will not use any JNDI instead use adapter's in-memory representation of event table to store all the necessary information

Note: The Event recovery data source (JNDI) name is **not mandatory** from V6.1. Now, the adapter can use **in-memory representation** of event table to store all the necessary information. Adapter uses this feature when event database information is not configured during inbound event polling. This feature will not support the capability of handling "Ensure once-only event delivery".

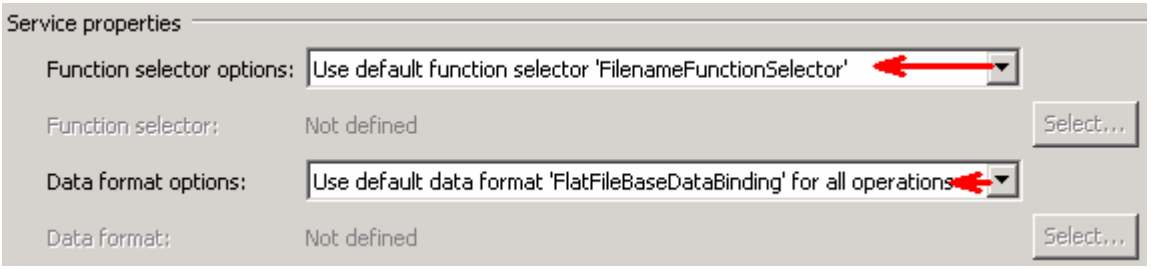
__ e. File archiving configuration:

1) Archive directory: click **Browse…** and select **<ARCHIVE_DIR>**



____ 6. Under Service properties, for **Function selector options**, select **Use default function selector 'FilenameFunctionSelector'** from the drop down list

____ 7. For **Data format options**, select **Use default data binding 'FlatFileBaseDataBinding' for all operations** from the drop down list

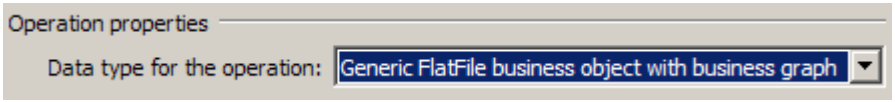


____ 8. Check the box next to **Change logging properties for wizard** to view the output location of the log file and the logging level and click **Next**

Define emitFlatFileBG operation:

____ 9. From the Operations screen, click **Add…**

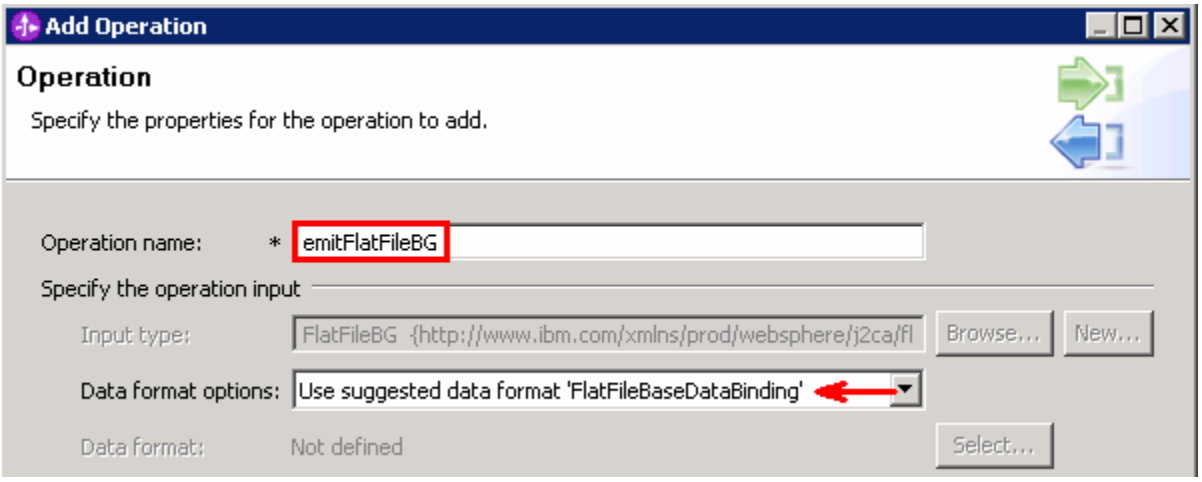
__ a. Add Operation window is opened. Select **Generic FlatFile business object with business graph** for the Data type and click **Next**



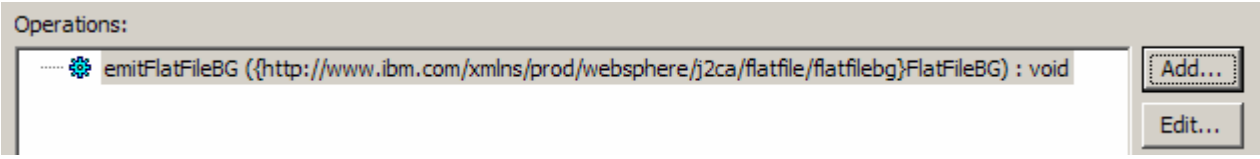
You are back to Operation window and because you have chosen the data type with business graph, the Input type is populated as **FlatFileBG**.

____ 10. For **Operation name**, enter any name, for Ex: **emitFlatFileBG**

____ 11. Accept the default selection, **Use suggested data format 'FlatFileBaseDataBinding'**, for **Data format options**

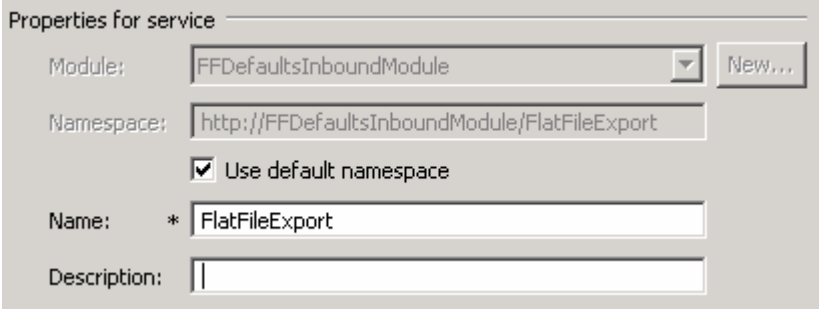


__ a. Click **Finish**. The defined operation, **emitFlatFileBG**, is populated under Operations list



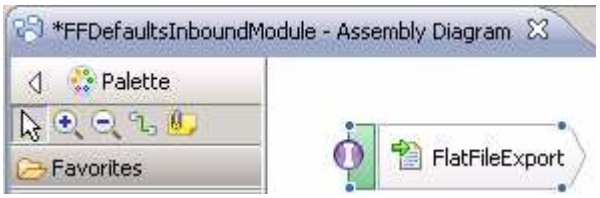
__ b. Click **Next** from Operations screen

____ 12. From Generate Service screen, accept the default value, **FlatFileExport**, for **Name**



__ a. Click **Finish**

____ 13. The Assembly diagram for FFDefaultsInboundModule is opened with an Export component, **FlatFileExport**:



____ 14. Save (**Ctrl + S**) changes to your assembly diagram

4.2. Add Java component

In this part of the lab, you will add a Java component and then wire the component to the existing Export interface. The Java component is your endpoint.

____ 1. Open the assembly diagram for FFDefaultsInboundModule (if it is already not open)

__ **a.** From the business integration view, expand **FFDefaultsInboundModule** and double click **Assembly diagram**

____ 2. Drop a Java component to onto the assembly diagram

__ a. From the **Palette**, click **Components** to expand it

__ b. Click **Java** and then click the empty space of FFDefaultsInboundModule assembly diagram. This will place a new component, **Component1** on the assembly diagram.

____ 3. Wire the FFDefaultsInboundInterface to the Component1

__ a. Select the **wire** ( ) icon from the Palette

__ b. Click **FlatFileExport** and then click **Component1** to wire them together

__ c. Select **OK** for the Add Wire pop-up window:

__ d. From the top of the Palette, click the **Selection Tool** icon ( ) to get back to the normal cursor mode

__ e. Right-click the empty space of the Assembly diagram and select **Arrange Contents Automatically** from the pop-up menu

Your assembly diagram for FFDefaultsInboundModule will look like this:

__ f. Right-click **Component1** and select **Generate Implementation** from the pop-up menu

__ g. On the **Generate Implementation** panel, select **default package**, and click **OK**

__ h. **Component1Impl.java** is opened in Assembly editor. Scroll down to the method **emitFlatFileBG** that needs to be implemented and add this code under that method:

```
System.out.println("***********Reached Endpoint************");
```

__ i. Save (**Ctrl + S**) and close Component1Impl.java

__ j. Save (**Ctrl + S**) and close Assembly diagram: FFDefaultsInboundModule

4.3. Test all defaults scenario

In this part of the lab, you will use the WebSphere Process Server Test Environment to test the SCA application Inbound processing for the pass through scenario.

____ 1. Add the project to the WebSphere Test Environment server

__ a. Right-click **WebSphere Process Server v7.0** under the Servers view and select **Add and remove projects…** from the pop-up menu

__ b. From the Add and Remove Projects window, select **FFDefaultsInboundModuleApp** under Available projects panel and click **Add >**

__ c. You will now see the **FFDefaultsInboundModuleApp** added to the **Configured projects**

__ d. Click **Finish** and wait until the project is being published onto the server. The server will start in Debug mode if it is not already started before

____ 2. Test the application by providing three different input files:

Note: For your convenience, three test files, **sample.txt, sample.txt1, sample.txt2** are placed in **<FFFILES>.**

__ a. Copy the any of the three test files (for Ex: **sample.txt** file) from <FFFILES> to <EVENT_DIR>. The adapter will poll the copied file from the event directory and will transfer it to the archive directory

__ b. No matter what type of file you put in Event directory, it will pass through the only existing method, FlatFileBG, and you should see this message your **Server Logs** view (or SystemOut.log):

Type	Time	Thread ID	Contents
Log message	Dec 5, 2009 19:08:48.2...	000001ce	WSVR0200I: Starting application: FFDefaultsInboundModuleApp
Log message	Dec 5, 2009 19:08:51.0...	000001ce	WSVR0221I: Application started: FFDefaultsInboundModuleApp
Log message	Dec 5, 2009 19:08:59.2...	00000082	**********Reached Endpoint**********

__ c. To verify your test results, check the **<ARCHIVE_DIR>** subdirectory, which should contain an archive of the event file with the same file name appended with year, month, date, system time, and success

____ 3. Restore the Sever Configuration

__ a. Right-click **WebSphere Process Server v7.0** under the Servers view and select **Add and remove projects…** from the pop-up menu

__ b. Select **FFDefaultsInboundModuleApp** under Configured projects and click **< Remove**

__ c. Click **Finish** after you see the application moved to Available projects. Wait until the application is being unpublished

Part 5: Use 'Create a service from a typical pattern'

In this part of the lab you will use the **typical pattern** option from the external service wizard to create and configure the function selector, data binding, and other required artifacts to test the inbound scenario.

Based on your selection, the binding resources (data binding and function selector) are created. You will review these later in this part.

After running the external service wizard, you will continue to add the Java component with implementation and then test the adapter.

5.1. Configure inbound using 'Create a service from a pattern (typical)' option

In this part of the lab you will use the **typical pattern** from the external service feature to create and configure the function selector

____ 1. Create the module: FFTypicalInboundModule

__ a. From the Business Integration window, right-click and select **New > Module**

__ b. From the New Module window, enter **FFTypicalInboundModule** for the Module Name

__ c. Ensure that the box next to **Open module assembly diagram** is checked and then click **Finish**

You will now see a new module, **FFTypicalInboundModule**, created from your Business Integration window and the Assembly diagram for the same module is opened in the Assembly Editor.

____ 2. Import required business objects

New in V7.0: Wrapper business objects for the business objects containing global elements are supported in this version. So, you can now pass the protocol specific information as part of each request.

__ a. Expand FFTypicalInboundModule (if not already expanded), right-click **Data Types** and select **Import…** from the pop-up menu

__ b. From the Import window, expand **General** and select **File System** and then click **Next**

__ c. Enter From directory

1) Click **Browse…** next to **From directory**

2) From the Import from directory window, select **<FFFILES >** and click **OK**

Now, you will see FFFiles folder added on the left side, and all the xsds and files under that folder on the right side.

__ d. Select the box next to **Customer.xsd**

__ e. Ensure that the **FFTypicalInboundModule** is selected for Into folder

__ f. Click **Finish** from the Import window

The Business Integration window is updated with the imported business objects.

____ 3. Review imported business object:

__ a. Expand **FFTypicalInboundModule > Data Types** and you will now see a new data type **Customer** and **Order** under it.

__ b. Double-click **Customer** review the fields inside the object:

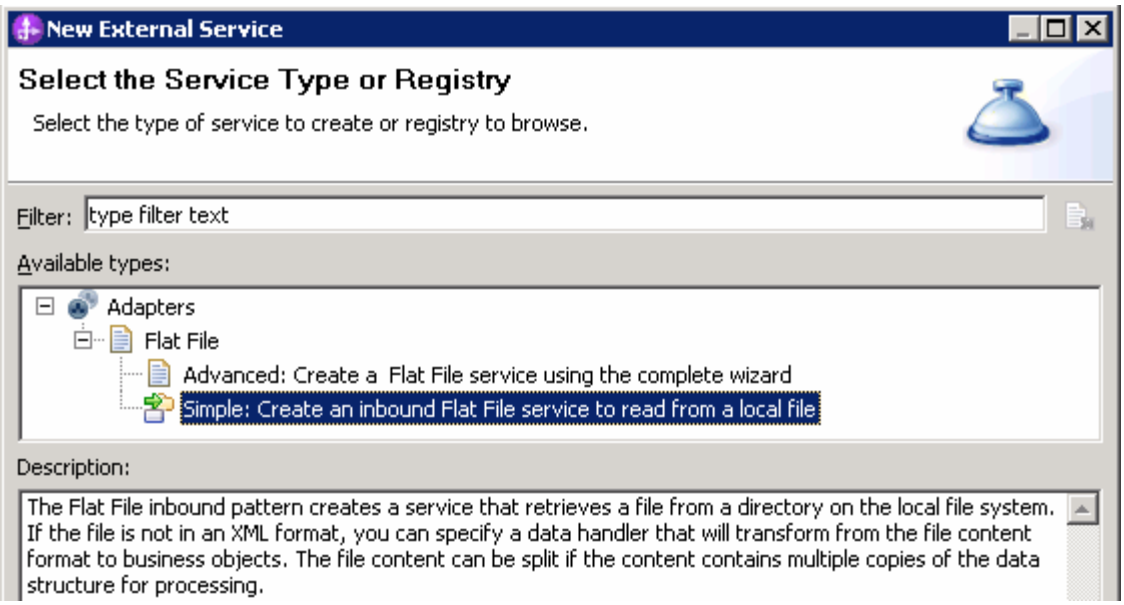
__ c. After reviewing, close the Customer business object from the Assembly editor

____ 4. To start an external service from the Palette:

__ a. From the **Palette** on the left side of Assembly Diagram, click **Inbound Adapters**:

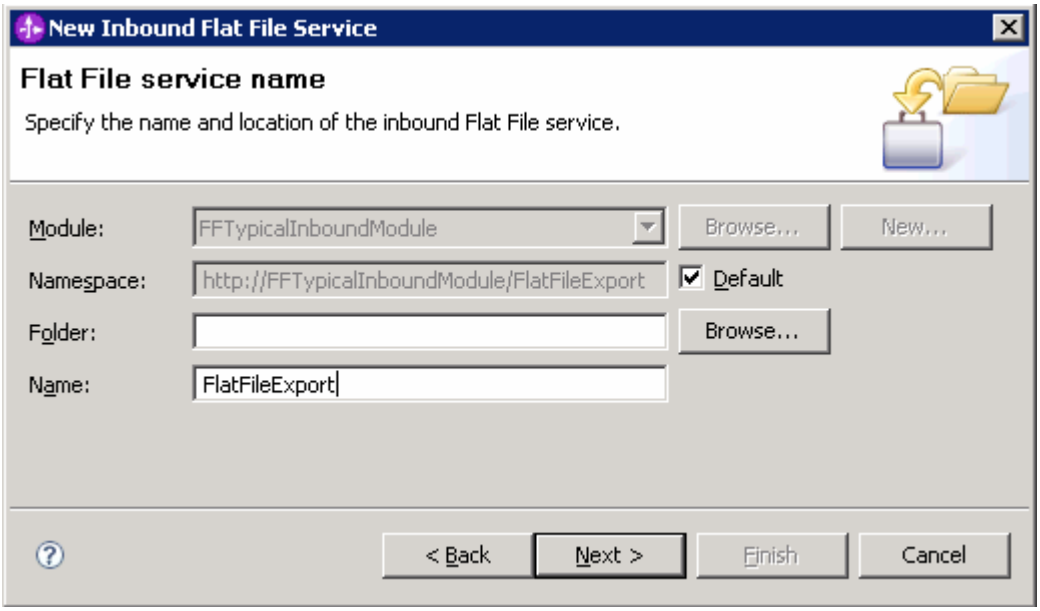
____ 5. Under Inbound Adapters, click the **Flat File** and then click the empty canvas of the assembly diagram. The New Flat File Service wizard is opened

____ 6. From the New External Service window, expand **Adapters > Flat File** and select **Simple: Create an inbound Flat File service to read from a local file**



__ a. Click **Next**

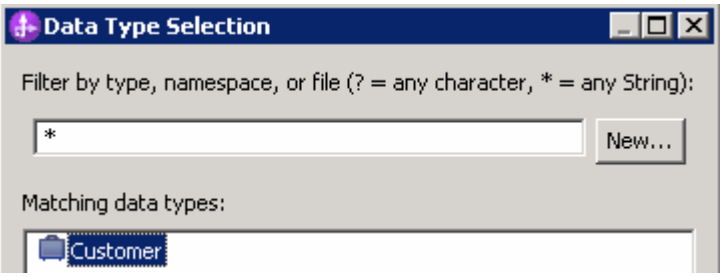
____ 7. From this **Flat File service name** screen, for **Name**, accept the default value '**FlatFileExport**' and click **Next**



____ 8. From the **Business object and directory screen**, enter these:

__ a. Click **Browse…** next to **Business object** and a Data Type Selection window is opened

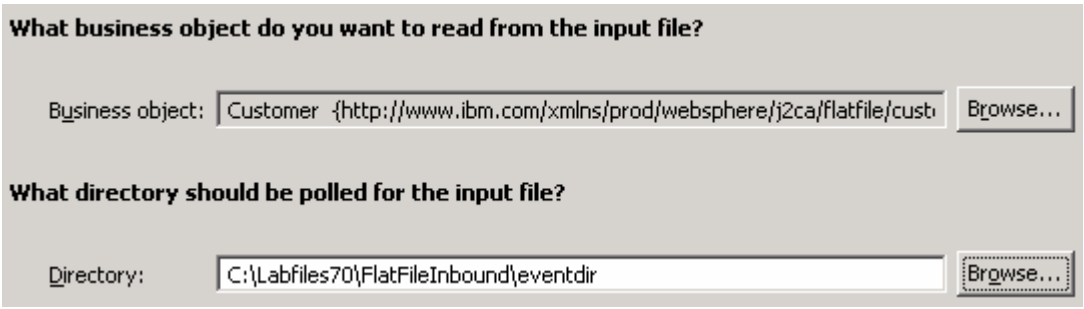
__ b. Select **Customer** under Matching data types and click **OK**



__ c. Now, click **Browse…** next to **Directory** and a Browse For Folder window is opened

__ d. From this window, navigate to select **<EVENT_DIR**> and click **OK**

__ e. Your Business object and directory screen should look like this:

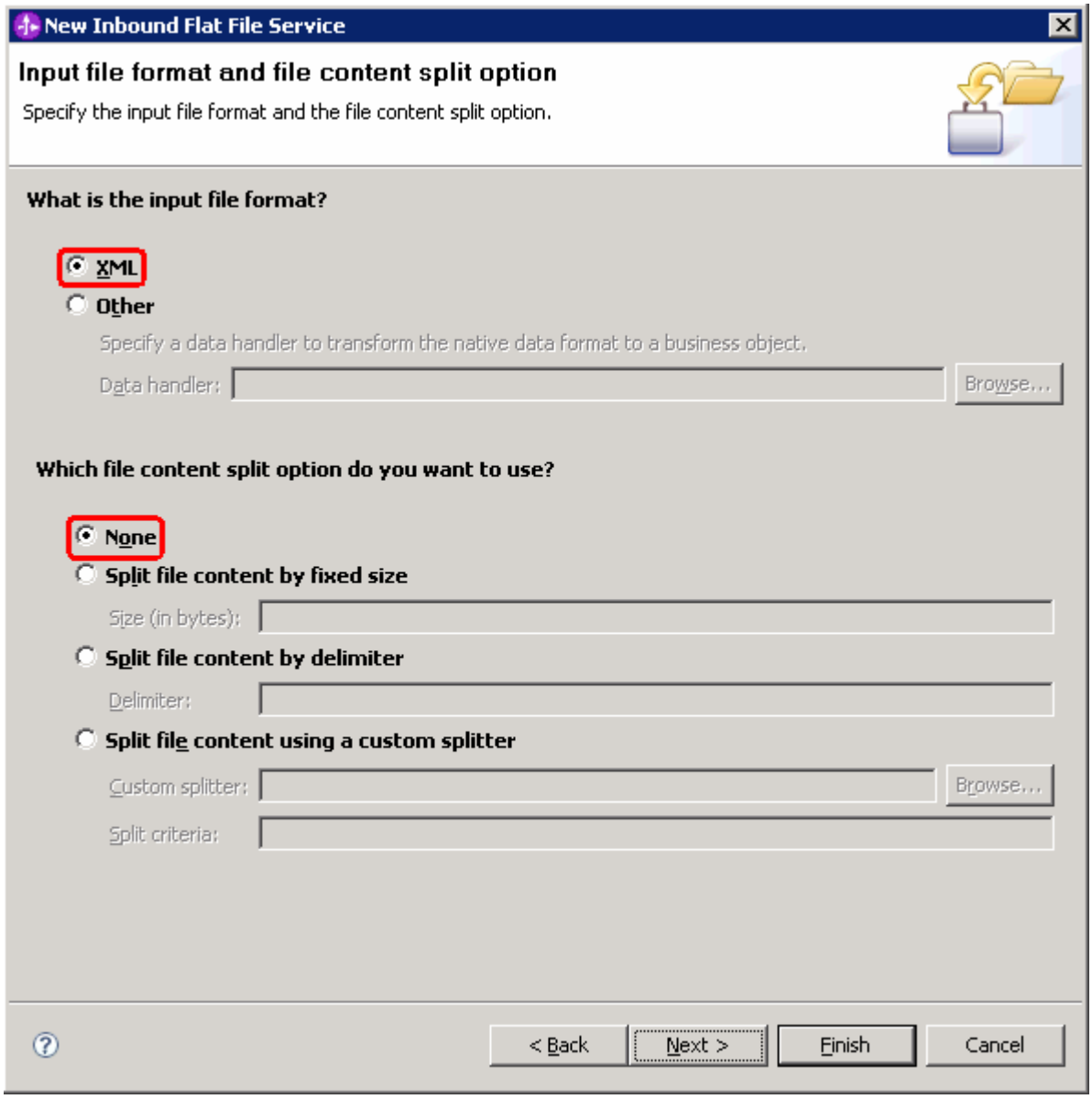


__ f. Click **Next**

____ 9. From the **Input file format and file content split option** screen, enter these:

__ a. For input file format, accept the default **XML** selection

__ b. For file content split option, accept the default selection, **None**

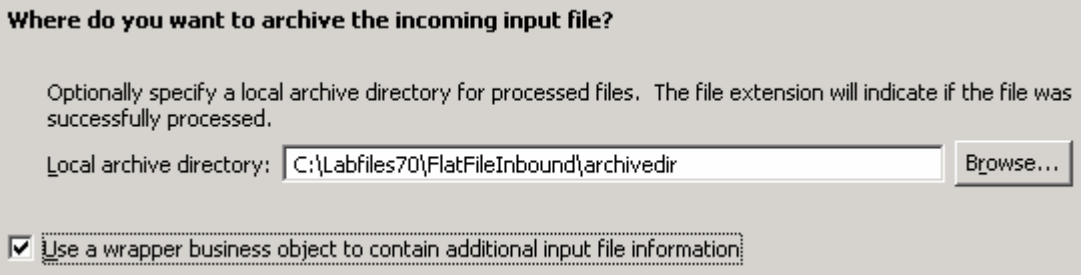


__ c. Click **Next**

____ 10. From the Archive directory and wrapper business object screen, enter these:

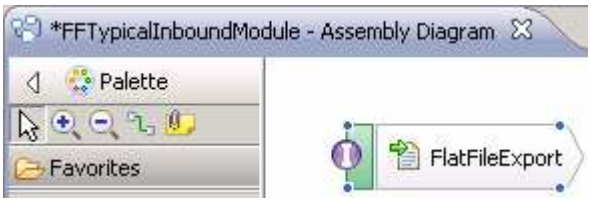
__ a. Click **Browse…** next to **Local archive directory** and select **<ARCHIVE_DIR>**

__ b. Check the box next to **Use a wrapper business object to contain additional input file information**. This will generate a Customer Wrapper under the Data Types of your Module

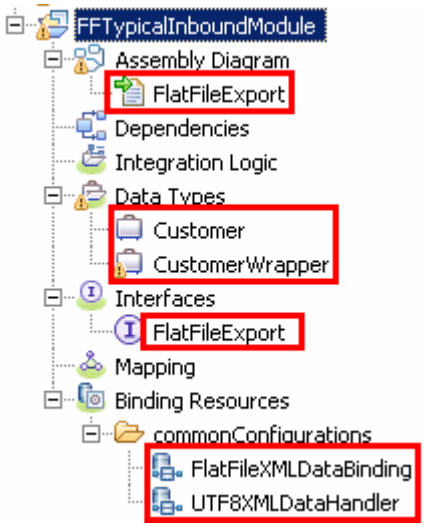


__ c. Click **Finish**

____ 11. Save (**Ctrl + S**) changes to your assembly diagram



____ 12. Review the FFTypicalInboundModule and the generated artifacts: The generated **Data Types**, **Interface**, Data handler (**UTF8XMLDataHandler**) and Data binding (**FlatFileXMLDataBinding**) under Configured Resources can be found under FFTypicalInboundModule. You can open each of these generated artifacts, business objects and review the properties inside.



5.2. Add Java component

In this part of the lab, you will add a Java component and then wire the component to the existing Export interface. The Java component is your endpoint.

____ 1. Open the assembly diagram for FFTypicalInboundModule (if it is already not open)

__ **a.** From the business integration view, expand **FFTypicalInboundModule** and double click **Assembly diagram**

____ 2. Drop a Java component to onto the assembly diagram

__ a. From the **Palette**, click **Components** to expand it

__ b. Click **Java** and then click the empty space of FFTypicalInboundModule assembly diagram. This will place a new component, **Component1** on the assembly diagram.

____ 3. Wire the FlatFileExport to the Component1

__ a. Select the **wire** ( ) icon from the Palette

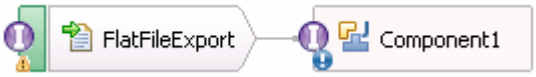
__ b. Click **FlatFileExport** and then click **Component1** to wire them together

__ c. Select **OK** for the Add Wire pop-up window:

__ d. From the top of the Palette, click the **Selection Tool** icon ( ) to get back to the normal cursor mode

__ e. Right-click the empty space of the Assembly diagram and select **Arrange Contents Automatically** from the pop-up menu

Your assembly diagram for FFTypicalInboundModule will look like this:



__ f. Right-click **Component1** and select **Generate Implementation** from the pop-up menu

__ g. On the **Generate Implementation** panel, select **default package**, and click **OK**

__ h. **Component1Impl.java** is opened in Assembly editor. Scroll down to the method **emitCustomer** that needs to be implemented and add this code under that method:

```
System.out.println("*************ENDPOINT emitCustomer**************");
  System.out.println("FILENAME : "+emitInput.getString("fileName"));
  DataObject customer = emitInput.getDataObject("Content");
  String name = customer.getString("customerName");
  System.out.println("NAME-----> "+name);
  String address = customer.getString("Address");
  System.out.println("ADDRESS--> "+address);
  String city = customer.getString("City");
  System.out.println("CITY-----> "+city);
  String state = customer.getString("State");
  System.out.println("STATE----> "+state);
```

Note: The code is also available at **<FFFILES>\TypicalCustomerJavaCode.txt** for your convenience

__ i. Save (**Ctrl + S**) and close Component1Impl.java

__ j. Save (**Ctrl + S**) and close Assembly diagram: FFTypicalInboundModule

5.3. Test typical pattern scenario

In this part of the lab, you will use the WebSphere Process Server Test Environment to test the SCA application Inbound processing for the typical pattern with input file having single business object.

____ 1. Add the project to the WebSphere Test Environment server

__ a. Right-click **WebSphere Process Server v7.0** under the Servers view and select **Add and remove projects…** from the pop-up menu

__ b. From the Add and Remove Projects window, select **FFTypicalInboundModuleApp** under Available projects panel and click **Add >**

__ c. You will now see the **FFTypicalInboundModuleApp** added to the **Configured projects**

__ d. Click **Finish** and wait until the project is being published onto the server. The server will start in Debug mode if it is not already started before

____ 2. Put the input files in the event directory

Note: For your convenience, the test file **SingleCustomerBO.xml** is placed in **<FFFILES>.**

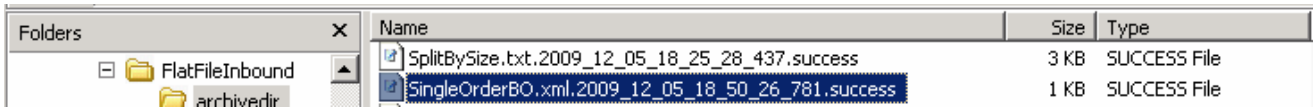
__ a. Copy the **SingleCustomerBO.xml** file from <FFFILES> to <EVENT_DIR>. The adapter will poll the copied xml file from the event directory and will transform it to the archive directory

__ b. Check your **Server Logs** view (or Systemout.log file) for this successful message:

Type	Time	Thread ID	Contents
Log message	Dec 5, 2009 19:14:16.6...	000001c8	WSVR0200I: Starting application: FFTypicalInboundModuleApp
Log message	Dec 5, 2009 19:14:19.4...	000001c8	WSVR0221I: Application started: FFTypicalInboundModuleApp
Log message	Dec 5, 2009 19:15:35.6...	000001ce	**********ENDPOINT emitCustomer**********
Log message	Dec 5, 2009 19:15:35.6...	000001ce	FILENAME : SingleCustomerBO.xml
Log message	Dec 5, 2009 19:15:35.6...	000001ce	NAME-----> IBM
Log message	Dec 5, 2009 19:15:35.6...	000001ce	ADDRESS--> 11501 Burnet Rd
Log message	Dec 5, 2009 19:15:35.6...	000001ce	CITY-----> Austin
Log message	Dec 5, 2009 19:15:35.6...	000001ce	STATE----> TX

____ 3. You can also verify the results by reviewing the archive directory

__ a. Check the **<ARCHIVE_DIR>** subdirectory, which should contain one archive of the event file with the same file name appended with year, month, date, system time, and success



____ 4. Restore the Sever Configuration

__ a. Right-click **WebSphere Process Server v7.0** under the Servers view and select **Add and remove projects…** from the pop-up menu

__ b. Select **FFTypicalInboundModuleApp** under Configured projects and click **< Remove**

__ c. Click **Finish** after you see the application moved to Available projects. Wait until the application is being unpublished

Solution instructions

____ 1. Start WebSphere Integration Developer V7.0 with a new workspace

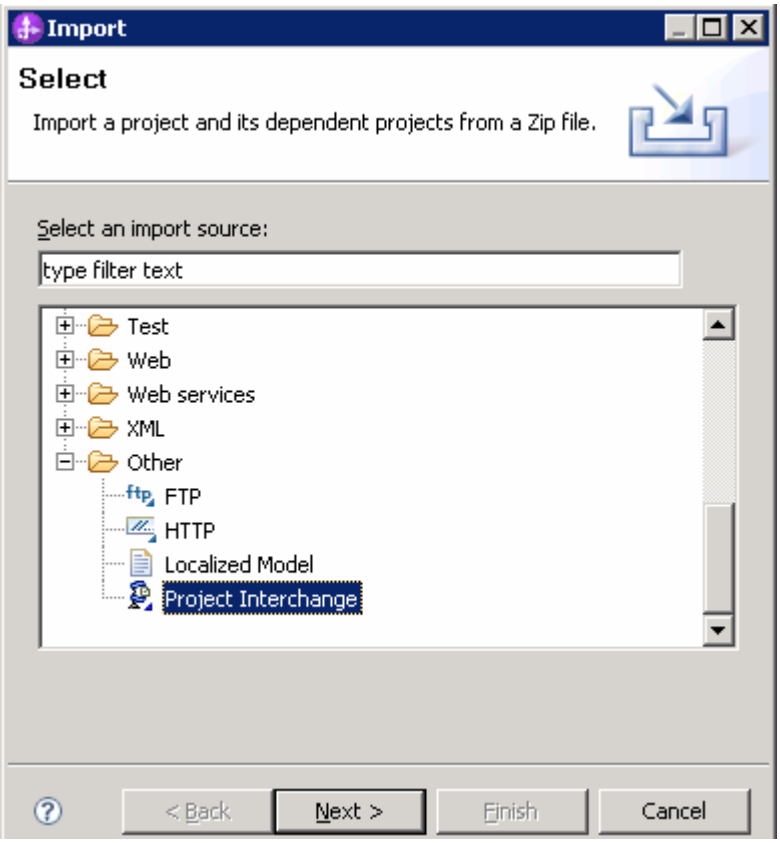
__ a. Follow the instructions outlined in **Part 1** of this exercise

____ 2. Import the solution Project Interchange

__ a. Import the project interchange file **FFInbound_PI.zip** from **<LAB_FILES>\FlatFileInbound\solution** directory

__ b. Select **File → Import** from the menu

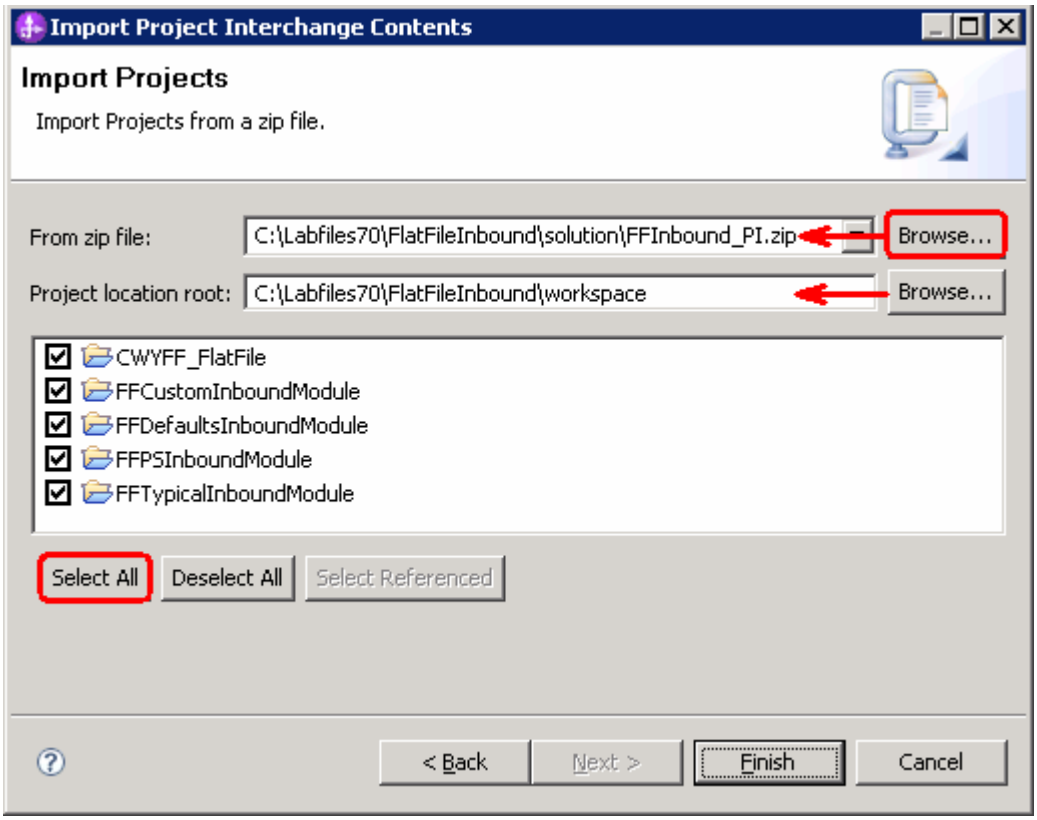
__ c. Select **Other → Project Interchange** in the **Import** dialog and click **Next**



__ d. For the **From zip file**, click on the **Browse** button and select the **FFInbound_PI.zip** in the **<LAB_FILES>>\ FlatFileInbound\solution** directory

__ e. Enter **<LAB_FILES>\FlatFileInbound\workspace** for the **Project location root**

__ f. Click the **Select All** button. This will select all modules: **CWYFF_FlatFile, FFCustomInboundModule, FFDefaultsInboundModule, FFPSInboundModule,** and **FFTypicalInboundModule**



__ g. Click **Finish**

____ 3. Test inbound pass through scenario:

__ a. Continue with **Part 2.3** and **Part 2.4** of this lab to test the inbound pass through scenario with and without SplitBySize

____ 4. Test inbound non pass through scenario:

__ a. Continue with **Part 3.3** and **Part 3.4** of this lab to test the inbound non pass through scenario with and without SplitByDelimiter

____ 5. Test inbound scenario with default data binding and data handler: Continue with **Part 4.3** of this lab

____ 6. Test inbound scenario using typical pattern: Continue with **Part 5.3** of this lab

What you did in this exercise

In this lab, you started with importing the Flat File Adapter RAR file into your WebSphere Integration Developer new workspace. Next, you made use of the external service wizard available in WebSphere Integration Developer to specify activation spec properties, define data binding, data handler, and function selectors which, after deploying onto the server, will generate Business Objects and other artifacts.

At the end of each part, you deployed and then tested the adapter application for these scenarios - pass-through (with and without SplitBySize) test scenario, two content specific or non pass through (with and without SplitByDelimiter) test scenarios, using all defaults (default data binding, function selector) scenario, and then finally using the typical pattern.

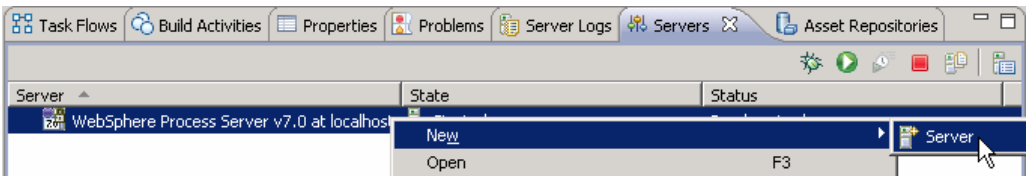
Task: Adding remote server to WebSphere Integration Developer test environment

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

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

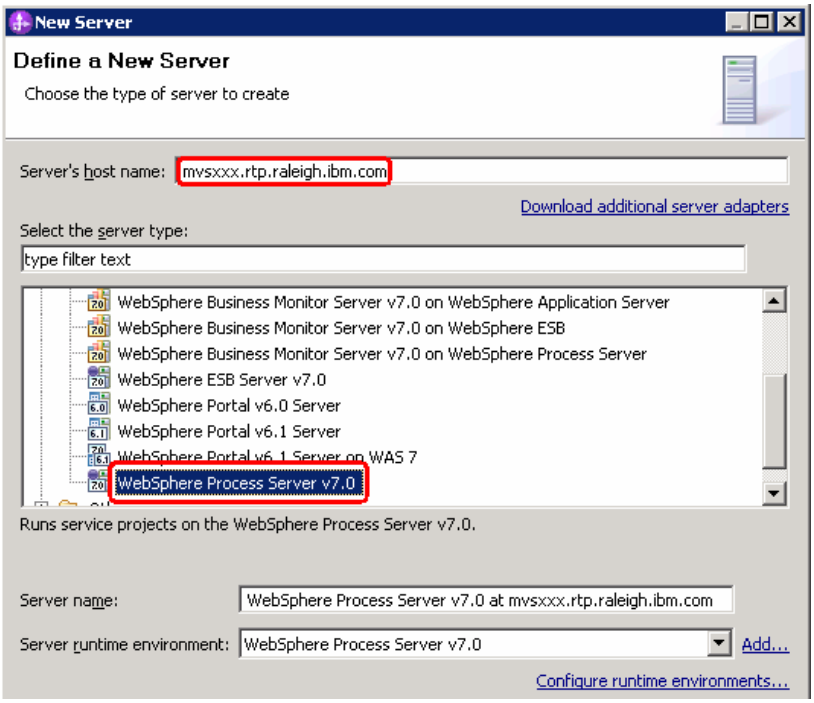
__ a. Right click the background of the Servers view to access the pop-up menu.

__ b. Select **New → Server**.



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

__ d. Ensure that the appropriate server type, '**WebSphere Process Server v7.0**' or '**WebSphere ESB Server v7.0**', is highlighted in the server type list

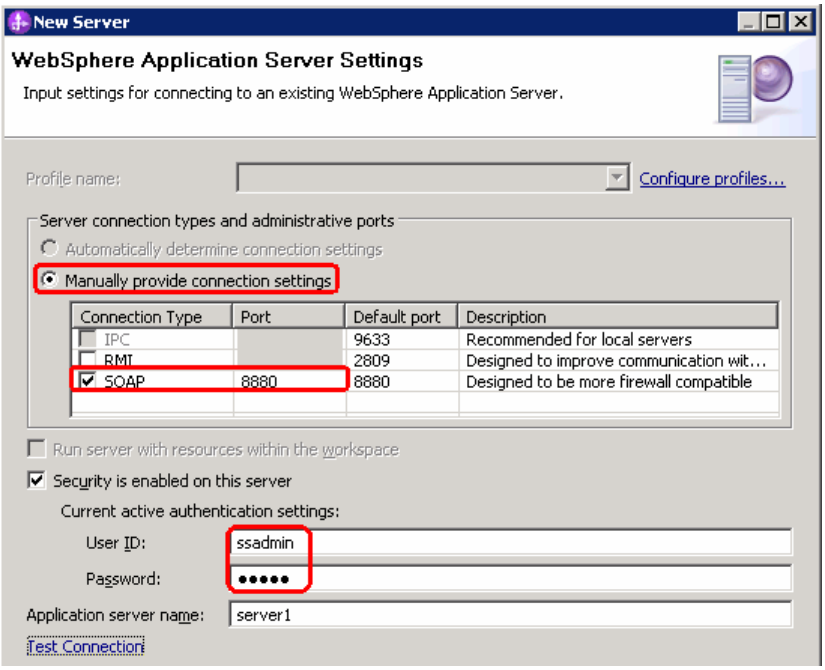


__ e. Click **Next**

__ f. On the WebSphere Server Settings page, leave the radio button for **Manually provide connection settings** selected, and select the box for SOAP

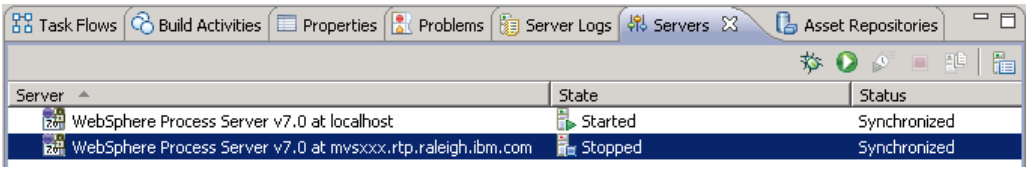
__ g. Enter the correct setting (**<SOAP_PORT>**) for **Port** column

__ h. If security is enabled on your server, select the box for **'Security is enabled on this server'** and enter **<USERID>** for the user ID and **<PASSWORD>** for the password.



__ i. Click **Finish**.

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



____ 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>**'

User ID : **<USERID>**

Password : **<PASSWORD>**

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

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

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

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

```
ADMU3200I: Server launched. Waiting for initialization status

ADMU3000I: Server sssr01 open for e-business; process id is 0000012000000002
```