Integration Designer
Version 7.5
Version 7 Release 0

Reading and writing files using the WebSphere Adapter for Flat Files

Note
Before using this information and the product it supports, read the information in "Notices" on page 21.

This edition applies to version 7, release 5, of IBM Integration Designer.

Contents

Chapter 1. Introduction

This sample will demonstrate how an application can read from a file on the file system and write to a file on the file system using the IBM WebSphere Adapter for Flat Files. You will also learn how to create an application with IBM® Integration Designer that processes the incoming data before writing the processed data to the file on the file system. This sample is for developers who work with adapters generally and the Flat Files adapter specifically.

Learning objectives

After completing this sample, you will understand how to use IBM Integration Designer with the WebSphere Adapter for Flat Files to develop an application that reads data from a file on the file system, processes the data with a service-oriented architecture (SCA) application and writes the processed data to a file on the file system. Specifically, you will learn how to perform the following tasks:

- Develop an inbound service using the external service pattern wizard with the Flat Files adapter to read data from a file on the file system
- Develop an outbound service with the same wizard and adapter to write data to a file on the file system
- Implement the business logic to process the data read from a file on the file system
- Test your application.

This sample can be used with the following runtime environments:

- WebSphere Enterprise Service Bus
- IBM Process Server

The time required to build and test this application yourself is about 30 minutes. You also have the option of importing the application already built and testing it, which should take you 10 minutes.

Overview

When working with this tutorial, you should know several concepts related to flat files, adapters, processing data to and from an application created in IBM Integration Designer with an adapter and have a high-level understanding of the application you will create.

The following concepts are discussed in this section:

- A flat file
- WebSphere Adapter for Flat Files
- Inbound processing
- File splitting
- Record delimiters
- Splitting by size
- Custom data bindings
- Outbound processing
- Overview of the application in this tutorial.

What is a flat file?

The term flat file refers to any file stored on a local file system, as opposed to a more complex set of files, such as those in a structured database. The data records in a flat file are typically stored sequentially and

without any metadata, such as the indices, keys, and relationships, that you would find in database storage. You can use the adapter for flat files to read and write any type of file. The two common data formats are XML or comma-separated values (CSV) records. The following example shows records in each format.

XML format

```
<?xml version="1.0" encoding="UTF-8"?>
<customer>
 <title>Mr</title>
 <name>Smith</name>
 <city>Ottawa</city>
 <state>ON</state>
</customer>
<customer>
 <title>Mrs</title>
 <name>Jones</name>
 <city>Winnipeg</city>
 <state>MB</state>
</customer>
```

CSV format

```
Mr,Smith,Ottawa,ON
Mrs,Jones,Winnipeg,MB
```

What is the WebSphere Adapter for Flat Files?

Suppose you have an external system that outputs sets of files to a directory. An example might be an order processing system that produces text files containing order information to be processed in a batch mode during off-peak hours. You would use the WebSphere Adapter for Flat Files along with IBM Integration Designer to help you create and manage these files.

Your application can use the adapter for flat files to create and manage files or to monitor a directory and read files from this directory. You have two options when creating a service associated with adapter for flat files, both of which are documented in the information center of the product:

- Simple external service wizard: With this wizard, you create a service by specifying the directories that the WebSphere Adapter for Flat Files reads from and writes to and formats of the data in those directories. Then the wizard generates the service. For many users, this quick and easy way of creating a service is sufficient. We will use this approach in this tutorial.
- Advanced service wizard: With this wizard, you create a service by a longer, more comprehensive process. Some users may find this more detailed approach more suitable. You begin by specifying the type of adapter you will require and then specify the directories that the WebSphere Adapter for Flat Files reads from and writes to, and the format of the data in those directories. Then you continue using further pages with fields in an advanced section to add more control over the generated service.

What are inbound and outbound processing?

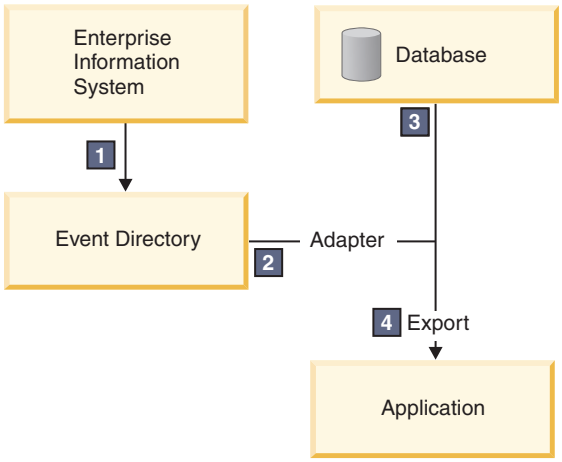
Inbound processing is the mode of operation in which the WebSphere Adapter for Flat Files monitors the file system, reads new files, and sends the data to an operation in an application.

Outbound processing is the mode of operation in which the WebSphere Adapter for Flat Files receives requests (from a component of an application) to perform a file operation and, when applicable, returns the results to the caller. Example operations include creating a file, writing to a file, or checking if a specific file exists.

Inbound processing

During inbound processing, the adapter for flat files listens for events that are produced by an event directory (for example, a file is placed in the event directory). An *event* is a record of what changes have occurred in the event directory. The directory that the adapter for flat files monitors for new files is called the *event directory*.

The following diagram shows the four steps that occur during inbound processing:



1. An external system outputs its files to the event directory.
2. The adapter for flat files polls for files from the event directory and converts the data from those files into events. The adapter can be configured to poll for only certain types of files, or for files created during a certain time period.
3. Events are temporarily placed in an event store. This is either a database or an in-memory representation of the event table. If you use a database, events will not be lost before your application can process them if the server goes down after an event is created, for example. This is referred to as *assured delivery*. Using in-memory tables, the event processing will be faster but you lose the event recovery capability.
4. The adapter for flat files retrieves the events from the event store and passes each event in the form of a business object through an exported inbound interface, which the external service wizard created.

This method of retrieving data, that is, passing data in the form of a business object, is called *non-pass-through or Data Transformation Framework* (DTF), and it operates on structured data.

If the application does not know the format of the data files, you can configure the adapter for flat files to run in *pass-through* mode, operating on unstructured data. In that case, it would not transform an event into a business object.

File splitting

You use file splitting when the files you want to retrieve are large or they each contain more than one record. You can split files into smaller chunks based on a delimiter or on a fixed-size value, which allows parts of the file to be processed in parallel. Each chunk is considered a separate event and is individually sent to an operation in an application through the exported inbound interface.

Record delimiter

Typically, input is stored as a single record per file. However, when the input file contains more than one record, you often separate the records in the file with a delimiter, which can be any text string and is usually a combination of characters followed by `\r\n` (platform dependent new line character). The adapter can both read and write files that contain a delimiter.

Using our XML format and CSV format examples discussed earlier, we have the string #### followed by a new line as the delimiter in the XML file shown in the following example. We have a new line as the delimiter in the CSV file.

XML format with #### as a delimiter

```
<?xml version="1.0" encoding="UTF-8"?>
<customer>
 <title>Mr</title>
 <name>Smith</name>
 <city>Ottawa</city>
 <state>ON</state>
</customer>
####
<customer>
 <title>Mrs</title>
 <name>Jones</name>
 <city>Winnipeg</city>
 <state>MB</state>
</customer>
####
```

CSV format with a new line as a delimiter

```
Mr,Smith,Ottawa,ON
Mrs,Jones,Winnipeg,MB
```

Split by size

The split-by-size feature is similar to splitting by delimiter, because you use it to divide a file into smaller chunks and transfer them, one by one, to the operation in the application. Use this feature with unstructured data in a pass-through scenario because the unstructured data is not going to be put into business objects. You specify the split criterion in the adapter as a number of bytes. The adapter reads the file as chunks (events) of that byte size. Each chunk will be the size defined by split criteria, except the last one, which might be smaller.

Custom data binding

Data in flat files can come in many different formats, such as the previously discussed XML and CSV formats. Others formats include name and value pairs, tab-separated, and fixed-width formats. A *data handler* maps from the format in the data files to the attributes of a business object.

Using the adapter for flat files default data binding, you can convert files to and from XML format. In other formats, you have to use custom data handlers that define the mapping between the file format and a business object. To implement a custom data handler, create a Java class that converts the data for the specific format and enter this class in the external service wizard when you configure the adapter.

For more information about creating a custom data handler, refer to the IBM Integration Designer documentation.

Outbound processing

To write or modify files, the application uses the operations defined in the outbound interface, which you create using the external service wizard. You can create both unstructured data (in pass-through mode) and structured data (in non-pass-through mode) using the different operation types:

- Create – Stores data to a new file or creates an empty file.
- Append – Appends data to an existing file. A new file is created if one did not already exist.
- Overwrite – Overwrites an existing file with a new data.
- Delete – Deletes an existing file.

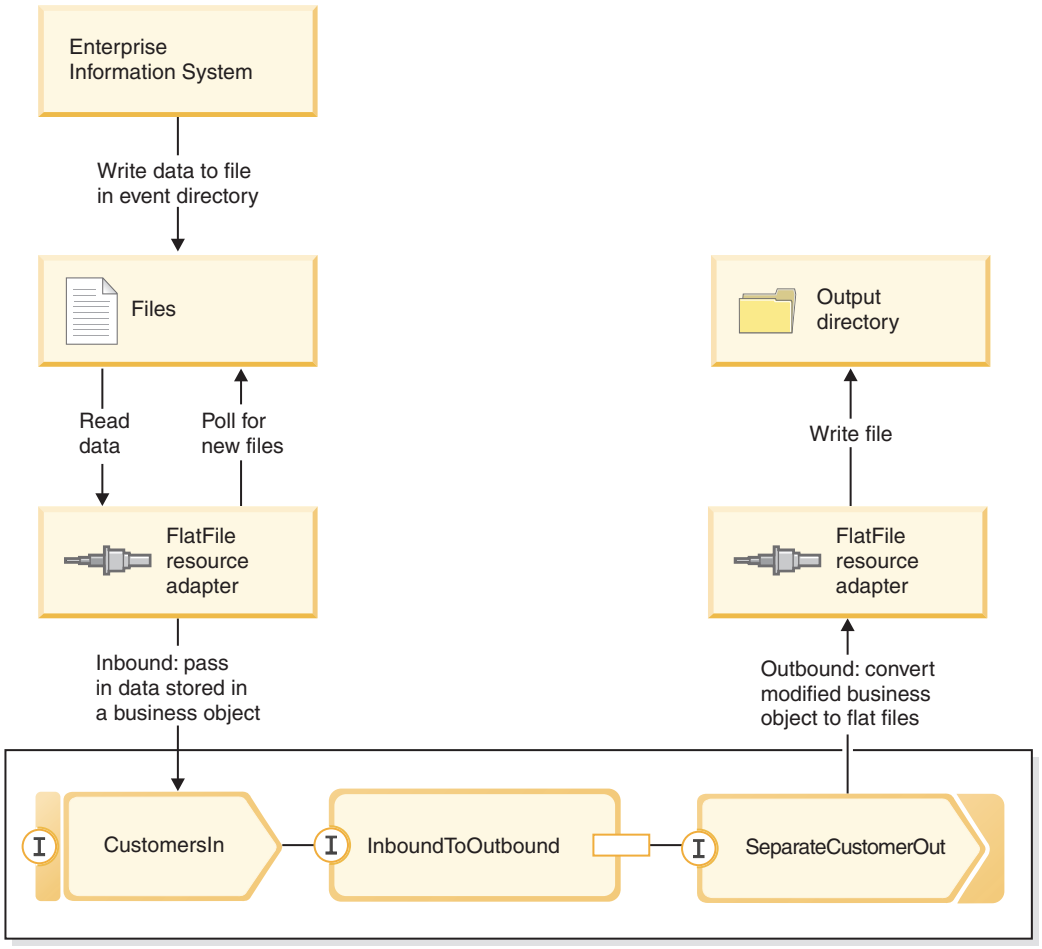
- Exists – Checks if a specific file exists.
- List – Lists files in a directory.
- Retrieve – Reads content from a file.

Only the create and append operations are used with the simple external service wizard.

Overview of the application in this tutorial

Imagine that you want to create an application that monitors a certain directory in a file system so that you can collect customer record information. When a file is created, the adapter uses the specified file splitter delimiter to split the content of the file into business objects. The export is called for each business object. The export is wired to a mediation flow which invokes the adapter to write the records into their own file in an output directory.

This scenario is shown in the following diagram:



When files that contain customer records are in the inbound events directory, the resource adapter retrieves them, creates a customer business object, and calls an export in the module, which initiates the mediation flow. The mediation flow invokes a file service which generates a new file name with its name based on the number of existing files.

Chapter 2. Build it Yourself

You can build this sample yourself.

Learning objectives

In building this hands-on sample with the wizards and editors in IBM Integration Designer, you will:

- Create the directories
- Create a module to contain the service you will develop
- Import a business object to save you development time
- Create an inbound service using the adapter pattern wizard
- Create an outbound service with the same wizard
- Create a mediation flow
- Deploy the module
- Test the module

Create the directories and input file

Several directories are used for the retrieval and storage of files. An input file is used for testing.

Create the following directories on your file system (assuming a Windows operating system):

1. Create a `c:\flatfiles` directory.
2. Create the following subdirectories in the flat files directory.
   a. `c:\flatfiles\inboundevents`
   b. `c:\flatfiles\inboundarchive`
   c. `c:\flatfiles\outputdir`

   Linux users: Create a similar file structure using the Linux file system.
3. Create the following `customer.txt` file and add it to the `c:\flatfiles` directory. When you test the application, you can copy it to the **c:\flatfiles\inboundevents** directory. The application will delete it from this directory when it processes it.

   Note that the namespace value **SeparateCustomers** must match the module name you will create in the next section.

```
<?xml version="1.0" encoding="UTF-8"?>
<p:Customer xsi:type="p:Customer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://SeparateCustomers">
  <customerName>John Doe</customerName>
  <Address>170 Baseline Ave</Address>
  <City>Ottawa</City>
  <State>ON</State>
</p:Customer>
####
<p:Customer xsi:type="p:Customer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://SeparateCustomers">
  <customerName>Susan Kidman</customerName>
  <Address>104 Candlestick Park</Address>
  <City>Ottawa</City>
  <State>ON</State>
</p:Customer>
####
<p:Customer xsi:type="p:Customer"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://SeparateCustomers">
  <customerName>Katheleen Black</customerName>
  <Address>530 Chanticler Road</Address>
  <City>Ottawa</City>
  <State>ON</State>
</p:Customer>
####
<p:Customer xsi:type="p:Customer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://SeparateCustomers">
  <customerName>Gary White</customerName>
  <Address>793 Morin Street</Address>
  <City>Ottawa</City>
  <State>ON</State>
</p:Customer>
####
```

Create a module

A module is similar to a project folder. It will contain the service you develop.

To create a module in the Business Integration perspective, follow these steps:

1. From the menu, select **File > New > Module**. The New Module window opens.
2. In the **Module Name** field, enter `SeparateCustomers`. Click **Finish**.
3. The module is created in the **Business Integration** view.

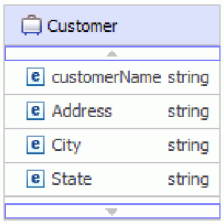
Note that the module name `SeparateCustomers` must match the same value in the namespace specified in the input file defined in the previous section.

Create customer business object

A business object is used to store your customer data, such as name, address, city and state.

To create the customer business object, follow these steps:

1. Right-click on **Data** in the **SeparateCustomers** module. in the Business Integration view. From the context menu, select **New > Business Object**. The New Business Object window opens.
2. For the **Name** enter `Customer`. Click **Finish**. The Business object editor opens.
3. Click the **Add a field to the business object** icon. A new field is added. Rename the field to `customerName`. Repeat adding fields `Address`, `City` and `State`. Leave the types as `string`, which is the default. Save your changes and close the editor.



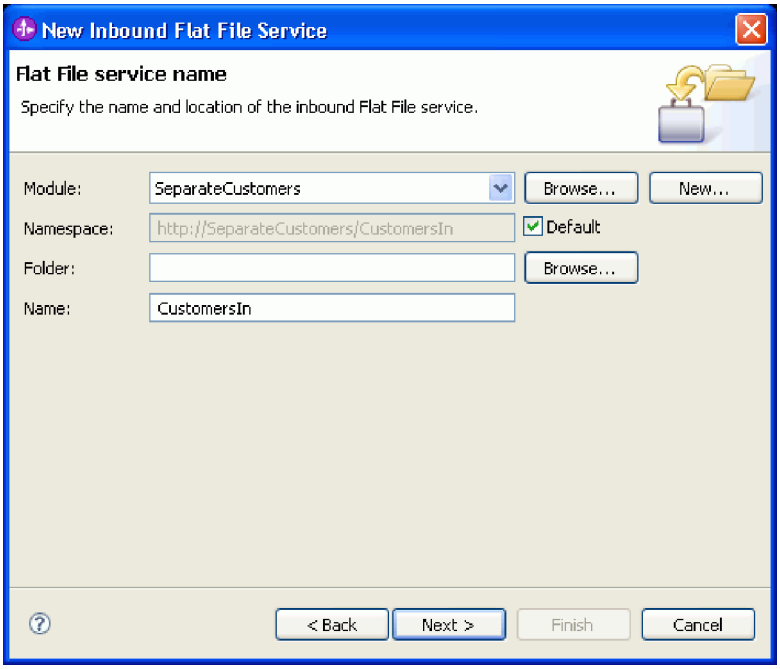
Create an inbound service

The inbound service receives the file from the file system for processing.

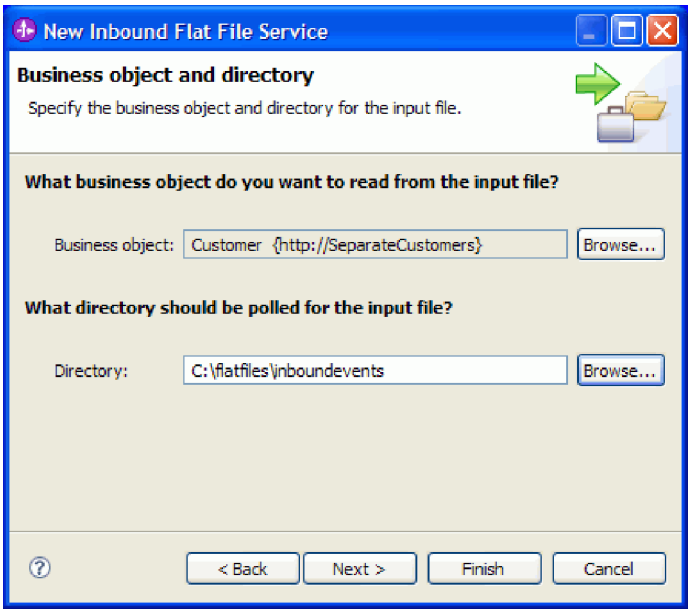
To create the inbound service using the adapter pattern wizard, follow these steps:

1. Right-click **SeparateCustomers** and from the context menu, select **New > External Service**. The External Service window opens. Expand **Adapters > Flat File**. Select **Simple: Create an inbound Flat File service to read from a local file** and click **Next**.

   Alternately, switch to **SeparateCustomers - Assembly Diagram** and from the palette in the assembly editor, expand **Inbound Adapters**. Select the **Flat File** adapter and then click on the assembly editor canvas. The External Service window opens. Expand **Adapters > Flat File**. Select **Simple: Create an inbound Flat File service to read from a local file** and click **Next**.
2. The New Inbound Flat File Service page opens. Change the name to **CustomersIn** and click **Next**.



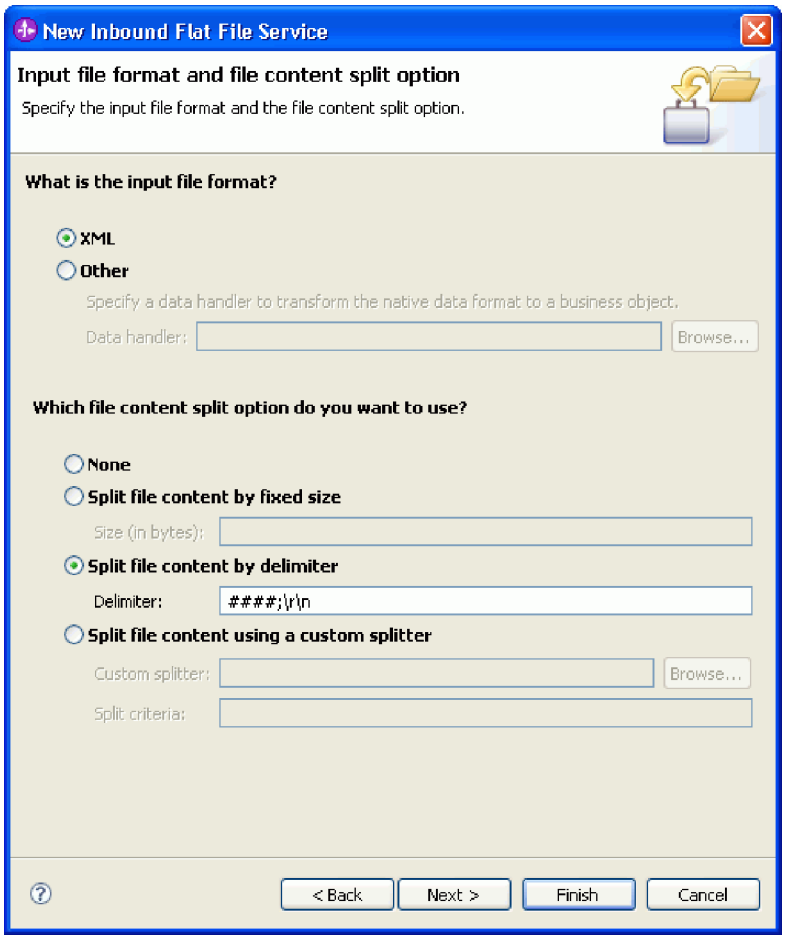
3. The Business object and directory page opens. Using **Browse**, navigate to and select the business object, **Customer**, you created earlier. For the directory entry, select the `flatfiles\inboundevents` directory where you placed the input file earlier in "Create the directories and input file" on page 7. Click **Next**.



4. The Input file format and file content split option page opens. Accept the default input file format, XML. XML file format is the standard file format used in service-oriented architecture (SOA) applications to contain a business object. However, you may also use a custom data handler, which you could have created earlier or one of the supplied data handlers. A data handler lets you transform a native input data format, for example, a stream of bytes, delimited data (such as comma separated values), fixed width data or JavaScript Object Notation (JSON) into a business object. Data handlers are discussed in "Working with data handlers, faults and registries" the information center and a sample is provided to show how to create one.

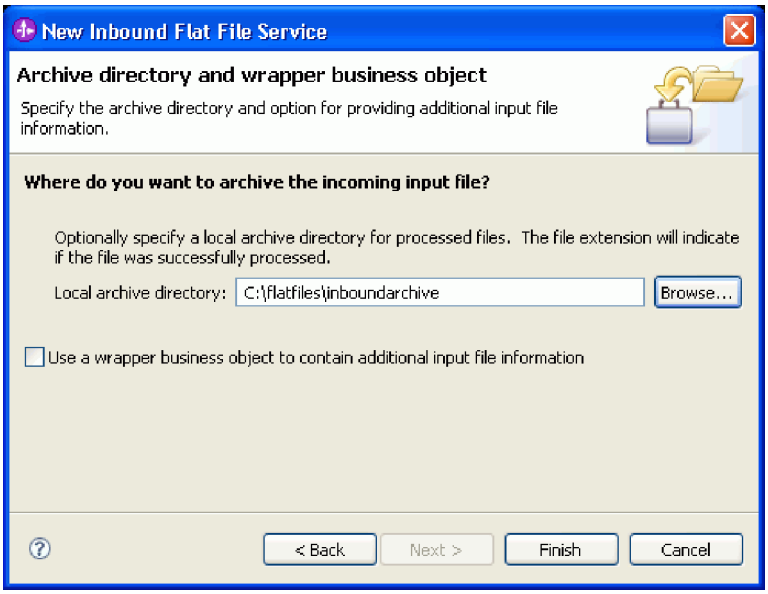
   Select **Split file content by delimiter**. In the field, enter `####;\r\n`. The delimiter is ####. The semi-colon (;) indicates there are more delimiters after the ####. The `\r\n`, in the Windows environment, indicates a new line. Linux users: use ####;\n. Click **Next**.

   Refer to the Flat Files documentation on file splitting for more information and examples.



5. The Archive directory and wrapper business object page opens. Navigate to the `flatfiles\inboundarchive` directory you created earlier. Click **Finish**. The inbound service is created.

   The optional archive directory is used if you want a record of your processed files. The collection of files will grow over time so an administrator or an application would need to regularly move them to offline storage. The wrapper business object would be used if you need to access the input file name or event directory.



Some warnings are created. They will be corrected by completing future steps in this sample.

The following artifacts are created.

Table 1. Artifacts created by the adapter pattern wizard

Artifact	Name	Description
Export	CustomersIn	The export exposes the module externally, in this case, to the WebSphere Adapter for Flat Files.
Interface	CustomersIn	This interface contains the operation that can be invoked.
Operation	emit	emit is the only operation in the interface.

Other generated artifacts allow data to be passed transparently between a service operating in a service-component architecture environment and another environment. They are the data binding (FlatFileXMLDataBinding) and the data handler (UTF8XMLDataHandler).

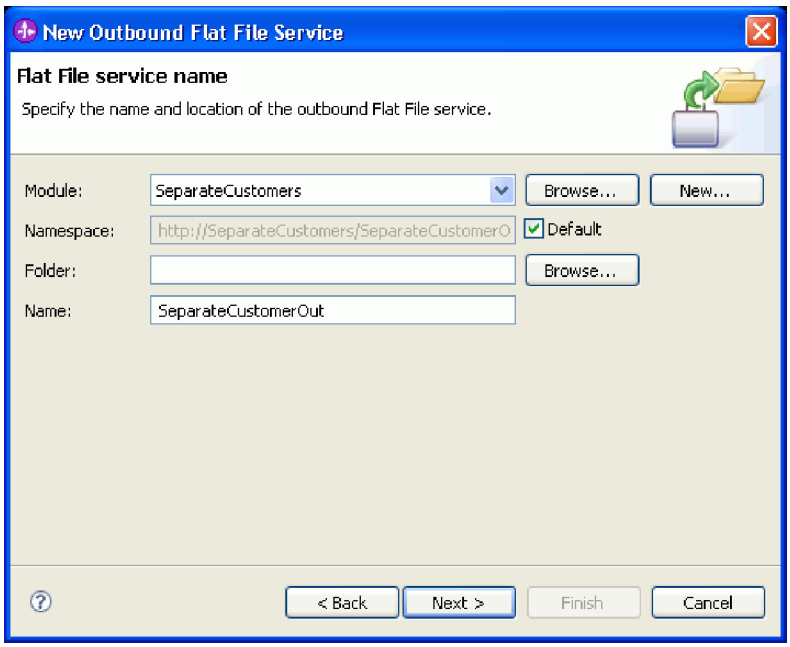
Create an outbound service

The outbound service outputs a file that has been processed to the file system.

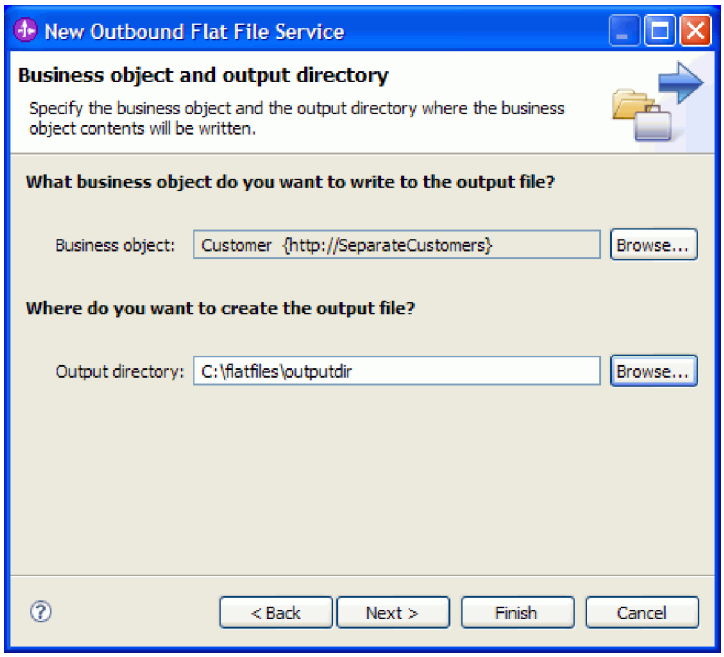
To create the outbound service using the adapter pattern wizard, follow these steps:

1. Right-click **SeparateCustomers** and from the context menu, select **New > External Service**. The External Service window opens. Expand **Adapters > Flat File**. Select **Simple: Create an outbound Flat File service to write to a local file** and click **Next**.

   Alternately, switch to **SeparateCustomers - Assembly Diagram** and from the palette in the assembly editor, expand **Outbound Adapters**. Select the **Flat File** adapter and then click on the assembly editor canvas. The External Service window opens. Expand **Adapters > Flat File**. Select **Simple: Create an outbound Flat File service to write to a local file** and click **Next**.
2. The New Outbound Flat File Service page opens. Change the name to **SeparateCustomerOut** and click **Next**.



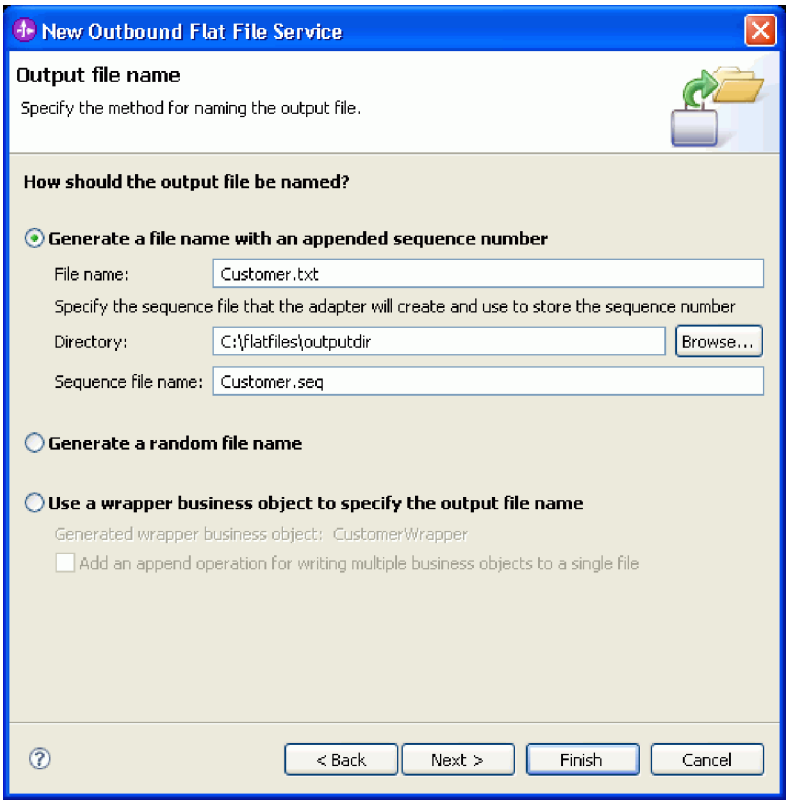
3. The Business object and output directory page opens. Using **Browse**, navigate to and select the business object, **Customer**, you created earlier. For the directory entry, select the `flatfiles\outputdir` output directory you created earlier in "Create the directories and input file" on page 7. Click **Next**.



4. The Output file name page opens. A variety of ways can be used to create an output file name. Accept the defaults, which will result in output files for each customer record in the format `Customer.<number of customer record>.txt`. The `Customer.seq` file will contain the number of customer records processed.

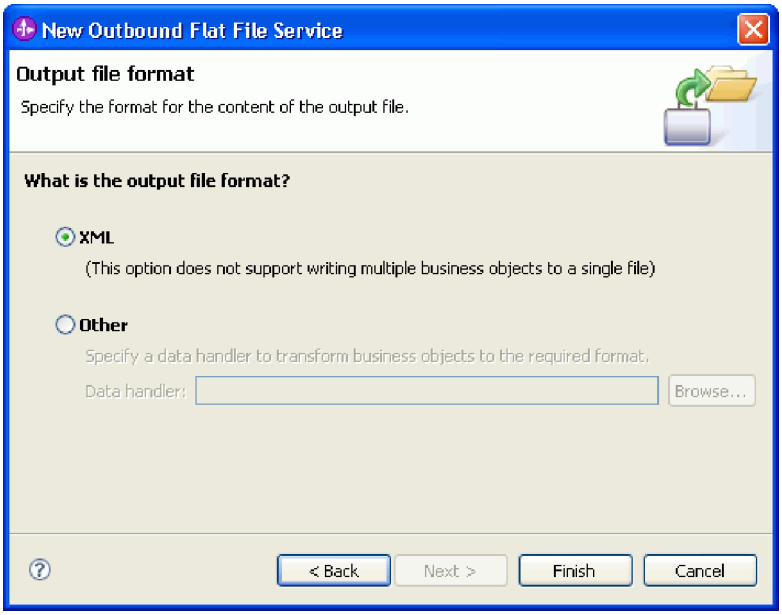
   Generating a random file name is useful if your application requires unique generated output file names. If your application will specify the business object name then select **Use a wrapper business object to specify the output file name**. In that case, you can also select an append operation for writing all business objects to a single file rather than multiple files.

   Click **Next**.



5. The Output file format page opens. Accept the default, XML, and click **Finish**.

   As discussed in “Create an inbound service” on page 8, XML is a standard file format in SOA applications. However, there can be cases where the native data is in another format such as a stream of bytes, delimited data (such as comma separated values), fixed width data or JavaScript Object Notation (JSON). In these cases, you can browse to select the data handler. Data handlers are discussed in "Working with data handlers, faults and registries" in the information center and a sample is provided to show how to create one.



The following artifacts are created.

Table 2. Artifacts created by the adapter pattern wizard

Artifact	Name	Description
Import	SeparateCustomerOut	The import lets your service use the WebSphere Adapter for Flat Files to output files.
Business objects	CreateResponse, UnstructuredContent	CreateResponse, which will be used in creating the output of the application, uses the Customer business object as its input. The UnstructuredContent business object can be used for generic input types.
Business objects for fault handling	DuplicateRecordFault, MissingDataFault, PrimaryKeyPairType, RecordNotFoundFault, WBIFault	These business objects can be used to handle exceptions. When a business exception is thrown, the adapter throws a fault exception and passes the fault information to the calling component. If you wish to use this information, you will need to create a fault selector configuration file.
Interface	SeparateCustomerOut	This interface contains the operation that can be invoked.
Operation	create	This operation creates the files in the output directory.

For our particular application, we will only be using a one-way inbound request and do not require fault handling. In the following steps, we will remove the faults from the import binding and the interface.

To remove the faults from the import binding, follow these steps:

1. Select `SeparateCustomerOut` in the assembly editor and then select the **Faults Configuration** tab in the **Properties** view.
2. Select **Default Fault Configuration**. In the panel to the right, click **Clear**.
3. Expand **Default Fault Configuration** and the **create** operation beneath it. Select `MISSING_DATA`. In the panel to the right, click **Clear**.
4. Repeat the previous step for `DUPLICATE_RECORD` and `RECORD_NOT_FOUND`.
5. Save your changes. From the menu, select **File > Save**. Close the assembly editor.

To remove the faults from the interface, follow these steps:

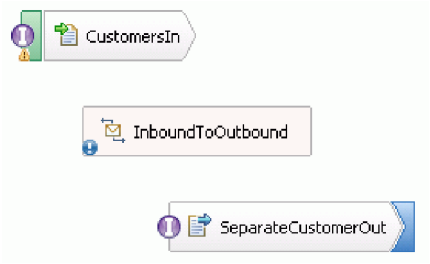
1. Expand **Interfaces** in the **Business Integration** view. Double-click **SeparateCustomerOut** to open the interface editor.
2. Click `MISSING_DATA` fault. Then select the **Delete** icon. It is on the top of the editor on the right.
3. Similarly, delete `DUPLICATE_RECORD` and `RECORD_NOT_FOUND` faults.
4. Save your changes. From the menu, select **File > Save**.Close the interface editor.

Create the mediation flow

The import and export must be linked together through a mediation flow to allow data to be passed between them because the interfaces are different.

To create a mediation flow, follow these steps:

1. Open the assembly editor, if it is not open. Double-click **Assembly Diagram** under **SeparateCustomers**.
2. From the **Palette**, select **Mediation Flow**, release the mouse button and then click with the mouse button again when you want the mediation flow to be placed onto the canvas. Place it between **CustomersIn** and **SeparateCustomerOut**. Rename the mediation flow to `InboundToOutbound`. Save your work. From the menu, click **File > Save**.

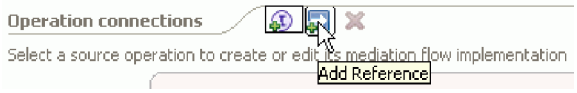


The entries you will see in the **Problems** view will disappear at the end of this section.

3. Double-click **InboundToOutbound**. A message asks if you would like to implement the component now. Click **Yes**. A Generate Implementation window opens. Accept the defaults and click **OK**. The mediation flow editor opens.
4. Near the top of the editor, select **Add Interface**.

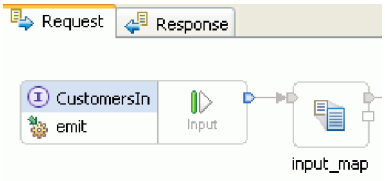
In the **Add Interface** dialog box, select **CustomersIn** and click **OK**.

5. Near the top of the editor, select **Add Reference**.



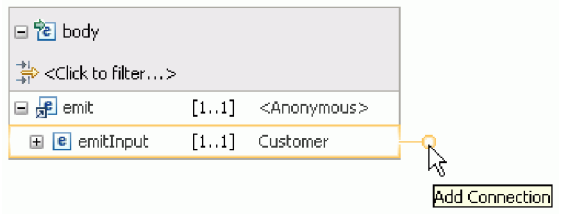
In the **Add Reference** dialog box, select **SeparateCustomerOut** and click **OK**.

6. In the mediation flow editor, click the **emit** link and from the **Select a Mediation Flow Template** dialog box click on the **Operation Map** link. In the **Select Reference Operation** dialog box, select **SeparateCustomerOutPartner** and click **OK**. Save your work. From the menu, click **File > Save**.
7. In the **Request** tab of the mediation flow editor, double-click on the **input_map** node to implement the input map.



The **New XML Map** window opens. Accept the defaults and click **Finish**. The map editor opens.

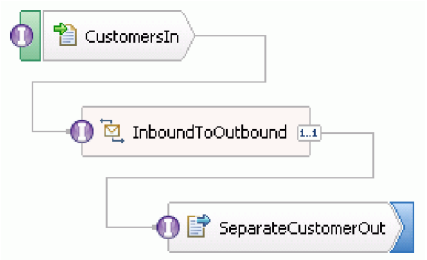
8. In the map editor, expand the **emit** element to reveal the **emitInput** element. Expand the **create** element to reveal the **createInput** element. Click the **Add Connection** handle beside the **emitInput** element. Drag the handle to the **createInput** element to create a **Move** transformation.



Save your work. Close the map editor. In the mediation flow editor, save your work. Close the mediation flow editor. In the assembly editor, save your work.

Since **emit** is a one-way operation and the application is splitting the content into separate files and creating output files from the split content, the application is complete at that point. Therefore, there is no need to map the response.

9. Connect the components. In the assembly editor, select the handle on the right end of **CustomersIn** and drag it to the left end of **InboundToOutbound**. A message tells you that the service interface from the export will be added to the target. Click **OK**. Select the handle on the right end of **InboundToOutbound** and drag it to the left end of **SeparateCustomerOut**. A message tells you that a matching reference will be created on the source. Click **OK**. Save your work.



Chapter 3. Run the sample

You can run the sample once you have finished building it or importing it.

If you imported the ready-made sample, you need to create the directories and add an input file as shown in "Create the directories and input file" on page 7.

To run the sample, follow these steps:

- If you imported the sample rather than built it yourself, import the WebSphere for Flat Files adapter.
- Deploy the module to the server
- Run and test the module

Import the adapter

Two versions of the sample are provided. One version includes the Flat File adapter in case you do not have it in your workspace yet. One version does not include the Flat File adapter. If previously you worked on the sample to create services that respond to Flat File events or you have worked with files that use the Flat File adapter, then the Flat File adapter is imported into the workspace for you. But if you need to import the adapter, the following steps are provided.

Once the adapter is imported, you need to use the clean function, which is also explained in these steps.

To import the WebSphere Adapter for Flat Files, follow these steps:

1. From the menu, select **File > Import**. The Import window opens. Expand **Java EE** and select **RAR file**. Click **Next**. The Connector Import page opens.
2. Beside the **Connector file** field, click **Browse**. Navigate to the following directory and select the RAR file:

   `<installdir>\ResourceAdapters\FlatFile<version number>`

   The remaining fields are completed for you.

   Deselect **Add Project to an EAR**. Click **Finish**. A message asks if you wish to switch to the Java EE perspective. Click **No**.
3. The adapter is imported and is shown in the Business Integration view.
4. Use the clean function to correctly set some values. From the menu, select **Project > Clean**.
5. In the Clean dialogue box, accept the defaults and click **OK**.

Deploy the module

Deploy the module to the server so that you can test your application on the runtime.

Testing begins as soon as the module is deployed because the application has been started implicitly as it is in inbound processing mode.

To deploy the module to the server:

1. Click **Servers** to open the **Server** view.
2. Right-click **IBM Process Server** and select **Start** from the context menu.
3. Once the server is in the **Started** state, right-click **IBM Process Server**. From the context menu, select **Add and Remove**. Select **SeparateCustomersApp** from **Available projects**. Click **Add**. Click **Finish**.
4. Testing the module at runtime begins at this point since, as discussed previously, the application is in inbound processing mode. See "Test the module" on page 18 for the output expected.

Test the module

Looking at the output in the console and in the output directory will tell you the success of the service you created.

To test the module, follow these steps:

1. Copy the `customer.txt` file into the `c:\flatfiles\inboundevents` directory.
2. Check the contents of the `c:\flatfiles\inboundevents` directory. The input file, `customer.txt`, should be removed from the folder once it has been processed by your application.
3. Check the `c:\flatfiles\outputdir` directory. You should see a list of files with a file name matching this pattern: `Customer.<sequence number>.txt`. The content of each file is a single customer data record. You will also see a sequencing file indicating the customer records processed.
4. In the `c:\flatfiles\inboundarchive` directory, you should see your original file with a timestamp suffix. For example, `customer.txt.<timestamp>.success`.
5. If you want to test again, put the `customer.txt` file into the `c:\flatfiles\inboundevents` directory again. This file and the directory structure are discussed in "Create the directories and input file" on page 7.

When you have finished testing, remove the projects from the server. Right-click **IBM Process Server** and from the context menu select **Add and Remove Projects**. In the Add and Remove Projects window, click **Remove All** to move the projects from the **Configured Projects** to the **Available Projects** pane. Click **Finish**.

Stop the server, which otherwise will continue to run. Right-click **IBM Process Server** again and from the context menu click **Stop**.

Chapter 4. Import

A complete ready-made version of this sample is available for you to import.

To import the ready-made sample:

1. In IBM Integration Designer, select **Help > Samples and Tutorials > IBM Integration Designer 7.5**. The **Samples and Tutorials** page opens.
2. Under the **Reading and Writing Files** section, click the **Import** link. The Sample Import page opens that allows you to import just the sample if you already have the Flat File connector project CWYFF_FlatFile in your workspace or the sample with the Flat Files adapter. Choose the option that applies to you and select **OK**.
3. The ready-made sample is imported into the workbench.

Instructions for running the sample are found in the topic "Run the Sample". The instructions include another import to add the adapter to your imported application. See “Import the adapter” on page 17.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for WebSphere Software
IBM Corporation
3600 Steeles Ave. East
Markham, Ontario
Canada L3R 9Z7

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Terms of use

Permissions for the use of publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, non commercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Readers' Comments — We'd Like to Hear from You

Integration Designer
Version 7.5
Reading and writing files using the WebSphere Adapter for Flat Files
Version 7 Release 0

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Send your comments to the address on the reverse side of this form.

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	Email address

Readers' Comments — We'd Like to Hear from You

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Canada Ltd. Laboratory
Information Development for
IBM Integration Designer
8200 Warden Avenue
Markham, Ontario
Canada L6G 1C7

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line