



**WebSphere Adapter for Flat Files User Guide**  
**Version 6 Release 2**





**Note**

Before using this information and the product it supports, read the information in "Notices" on page 201.

**December 2008**

This edition applies to version 6, release 2, modification 0 of IBM WebSphere Adapter for Flat Files and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email <mailto://doc-comments@us.ibm.com>. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2006, 2008. All rights reserved. US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

© **Copyright International Business Machines Corporation 2006, 2008.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

## Chapter 1. Overview of WebSphere

### Adapter for Flat Files . . . . . 1

Hardware and software requirements . . . . .	1
Technical overview of the Adapter for Flat Files . . . . .	2
Outbound processing . . . . .	3
Inbound processing. . . . .	12
Business objects . . . . .	22
WebSphere Application Server environment variables . . . . .	24
The external service wizard . . . . .	25

### Chapter 2. Planning for adapter implementation . . . . . 27

Before you begin . . . . .	27
Security . . . . .	27
Support for protecting sensitive user data in log and trace files . . . . .	27
Deployment options . . . . .	28
WebSphere Adapters in clustered environments . . . . .	30
Migrating to version 6.2 . . . . .	32
Migration considerations . . . . .	32
Performing the migration. . . . .	34
Updating but not migrating a version 6.0.2.x or version 6.1.x project . . . . .	35
Migrating WebSphere business integration applications for use with Version 6.2 WebSphere Adapters . . . . .	36
Road map for migrating applications from WebSphere InterChange Server. . . . .	36
Migration considerations for WebSphere business integration adapters . . . . .	37
Migrating application artifacts from WebSphere InterChange Server . . . . .	38
Migrating adapter-specific artifacts . . . . .	39
Import, export, and WSDL file changes after migration . . . . .	42

### Chapter 3. Samples and tutorials . . . . . 45

### Chapter 4. Configuring the module for deployment. . . . . 47

Road map for configuring the module . . . . .	47
Creating the required local folders. . . . .	48
Creating the module . . . . .	49
Defining WebSphere Application Server environment variables . . . . .	51
Defining business objects . . . . .	54
Converting business objects to COBOL copybook files during outbound processing . . . . .	57
Converting COBOL copybook files to business objects during inbound processing. . . . .	66
Creating a simple service with the adapter pattern wizard . . . . .	75
Creating the project. . . . .	80

Configuring the module for outbound processing . . . . .	82
Setting deployment and runtime properties. . . . .	82
Selecting the operation and data type. . . . .	85
Configuring the data binding . . . . .	88
Configuring data handlers . . . . .	89
Setting interaction properties and generating the service . . . . .	94
Configuring the module for inbound processing . . . . .	97
Setting deployment and runtime properties. . . . .	97
Selecting the operation and data type . . . . .	104
Configuring the data binding . . . . .	106
Configuring data handlers . . . . .	107
Setting deployment properties and generating the service . . . . .	110

### Chapter 5. Changing interaction specification properties using the assembly editor . . . . . 113

### Chapter 6. Deploying the module . . . . . 115

Deployment environments . . . . .	115
Deploying the module for testing. . . . .	115
Generating and wiring a target component for testing inbound processing . . . . .	115
Adding the module to the server. . . . .	117
Testing the module for outbound processing using the test client . . . . .	118
Deploying the module for production . . . . .	119
Installing the RAR file (for modules using stand-alone adapters only) . . . . .	119
Exporting the module as an EAR file . . . . .	120
Installing the EAR file . . . . .	121

### Chapter 7. Administering the adapter module . . . . . 123

Changing configuration properties for embedded adapters . . . . .	123
Setting resource adapter properties for embedded adapters . . . . .	123
Setting managed (J2C) connection factory properties for embedded adapters . . . . .	125
Setting activation specification properties for embedded adapters . . . . .	127
Changing configuration properties for stand-alone adapters . . . . .	129
Setting resource adapter properties for stand-alone adapters . . . . .	129
Setting managed (J2C) connection factory properties for stand-alone adapters . . . . .	130
Setting activation specification properties for stand-alone adapters . . . . .	132
Starting the application that uses the adapter. . . . .	133
Stopping the application that uses the adapter . . . . .	134

Monitoring performance using Performance Monitoring Infrastructure . . . . .	134
Configuring Performance Monitoring Infrastructure . . . . .	135
Viewing performance statistics . . . . .	137
Enabling tracing with the Common Event Infrastructure (CEI) . . . . .	138

**Chapter 8. Troubleshooting and support . . . . . 141**

Support for the Log and Trace Analyzer . . . . .	141
Configuring logging and tracing . . . . .	142
Configuring logging properties . . . . .	142
Changing the log and trace file names . . . . .	143
First-failure data capture (FFDC) support . . . . .	144
Business faults . . . . .	145
XAResourceNotAvailableException . . . . .	146
org.xml.sax.SAXParseException . . . . .	146
Self-help resources. . . . .	147

**Chapter 9. Reference information. . . . . 149**

Business object information. . . . .	149
Business object structures . . . . .	149
Attribute properties . . . . .	152
Naming conventions . . . . .	152

Custom file splitting . . . . .	153
Outbound configuration properties . . . . .	154
Connection properties for the wizard . . . . .	156
Managed connection factory properties. . . . .	159
Resource adapter properties . . . . .	164
Interaction specification properties . . . . .	166
Inbound configuration properties. . . . .	171
Connection properties for the wizard . . . . .	173
Activation specification properties . . . . .	176
Resource adapter properties . . . . .	191
Globalization . . . . .	193
Globalization and bidirectional data transformation . . . . .	193
Bidirectional transformation in business objects . . . . .	195
Properties enabled for bidirectional data transformation . . . . .	197
Adapter messages. . . . .	198
Related information . . . . .	198

**Notices . . . . . 201**

Programming interface information . . . . .	203
Trademarks and service marks . . . . .	203

**Index . . . . . 205**

---

## Chapter 1. Overview of WebSphere Adapter for Flat Files

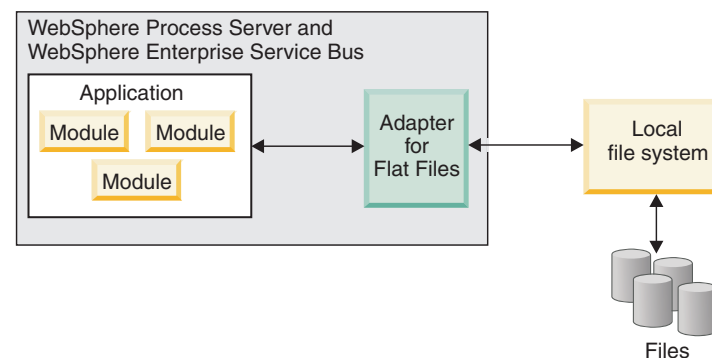
With WebSphere Adapter for Flat Files, you can create integrated processes that include the exchange of data with the local file system, without special coding.

You can use the adapter to read data from a file in the local file system, use it in an application on WebSphere Process Server or WebSphere Enterprise Service Bus, and send it back to the local file system. You can also use the adapter to poll a directory in the local file system for new files and send them to an application for processing.

The adapter can be used to read and write to any type of file stored in the local file system. It can:

- Create new files
- Append to or overwrite existing files
- Retrieve the content of a given file, retrieve a list of file names in a directory, or delete a file
- Check whether a particular file exists
- Poll a directory for new files and send them to an application for processing

The following illustration shows the adapter as part of an SOA implementation.



*Adapter overview*

---

## Hardware and software requirements

The hardware and software requirements for WebSphere Adapters are provided on the IBM® Support Web site.

To view hardware and software requirements for WebSphere Adapters, see <http://www.ibm.com/support/docview.wss?uid=swg27006249>

### Additional information

The following links provide additional information you might need to configure and deploy your adapter:

- The compatibility matrix for WebSphere Business Integration Adapters and WebSphere Adapters identifies the supported versions of required software for your adapter. To view this document, go to the WebSphere Adapters support

page and click **Compatibility Matrix** beneath the **Related** heading in the **Additional support links** section: <http://www.ibm.com/software/integration/wbiadapters/support/>.

- Technotes for WebSphere Adapters provide workarounds and additional information that are not included in the product documentation. To view the technotes for your adapter, go to the following Web page, select the name of your adapter from the **Product category** list, and click the search icon: <http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm>.

---

## Technical overview of the Adapter for Flat Files

IBM WebSphere Adapter for Flat Files makes it possible for services running on WebSphere Process Server or WebSphere Enterprise Service Bus to exchange data with the local file system.

Services can use the adapter to exchange data with the local file system in two ways:

- Through *outbound processing*, services running on WebSphere Process Server or WebSphere Enterprise Service Bus use the adapter to perform operations on files in the local file system, for example, to update an order document.
- Through *inbound processing*, services running on WebSphere Process Server or WebSphere Enterprise Service Bus use the adapter to receive events from the local file system, for example, to be notified that a customer record has been updated.

You configure the adapter to perform this processing through the external service wizard, launched through WebSphere Integration Developer. Using the external service wizard, you create a *module*, which consists of a project in WebSphere Integration Developer and a unit of deployment to WebSphere Process Server or WebSphere Enterprise Service Bus. Each module contains the components that make up a service and either an *import* or an *export*:

- An *import* is the point at which an SCA module accesses an external service (a service outside the SCA module) as if it were local. An import defines interactions between the SCA module and the service provider. An import has a binding and one or more interfaces.
- An *export*, also known as an endpoint, is an exposed interface from a Service Component Architecture (SCA) module that offers a business service to the outside world. An export has a binding that defines how the service can be accessed by service requesters, for example, as a Web service.

The module is packaged and deployed to WebSphere Process Server or WebSphere Enterprise Service Bus as an enterprise archive (EAR) file.

To represent files that are exchanged between a module and the local file system, the adapter uses business objects. A business object is a logical data container that contains the data that is processed by the adapter. You create business objects by using the external service wizard or by using the business object editor in WebSphere Integration Developer.

The adapter uses adapter-specific *data bindings* and *data handlers* to transform data from one format to another during inbound and outbound processing. *Data bindings* are essentially maps that define how a business object should be formatted. A data binding reads the fields in a business object and fills in the corresponding fields in the file. The data binding that is used depends on the



internal format of the file. Each data type has an equivalent data binding. You use the external service wizard to configure the data binding.

*Data handlers* perform the conversions between a business object and a native format. When you select a data type that contains business objects, you must specify the data handler that will perform the conversion. Data handlers are provided by WebSphere Process Server or WebSphere Enterprise Service Bus.

## Outbound processing

During outbound processing, the adapter receives a request from the module, in the form of a business object, to perform an operation on a file in the local file system. The adapter performs the requested operation and returns a business object, if applicable, that represents the result of the operation to the component.

The following illustration shows the outbound processing flow for WebSphere Adapter for Flat Files.

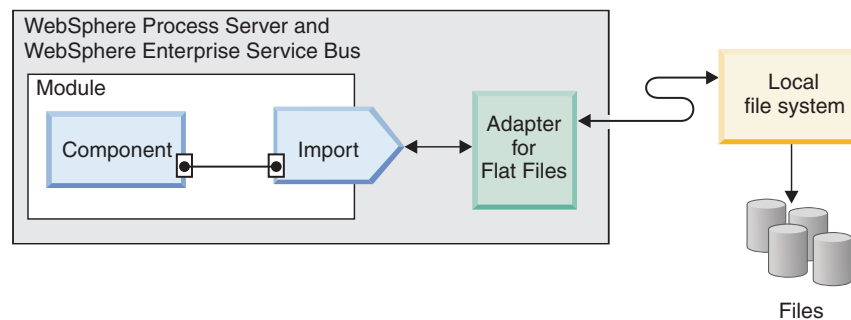


Figure 1. Outbound processing

## Outbound operations

An operation is the action that an adapter can perform on the local file system during outbound processing. The name of the operation indicates the type of action that the adapter takes.

The adapter supports the following operations during outbound processing.

### Append operation:

The Append operation appends content to a specified file. A new file is created during the append operation.

If you select the **Enable response type for the operation** check box in the external service wizard, the file name is returned to the component in a business object.

If the `CreateFileIfNotExists` property is set to `true`, the adapter creates a new file. If the `GenerateUniqueFile` property is set to `true`, the adapter generates a unique file and ignores the value in the `Filename` property.

**Note:** The `GenerateUniueFile` property has been deprecated. Although you can currently set this property, but the adapter will always treat the value for this property as `false`.

If the file that is to be appended does not exist and the `CreateFileIfNotExists` property is set to `false`, the adapter generates a `RecordNotFoundException` error.

If the `Filename` property has no value, the adapter generates a `MissingDataException` error.

**Note:** In the case of wrapper business objects, if the value is unset for the `CreateFileIfNotExists` property on the wrapper, then the value set on its interaction specification property are used.

#### **Create operation:**

The Create operation creates a file with the specified name. You can modify the created file name by specifying different properties. For example, you can attach a sequence number to the file.

If you select the **Enable response type for the operation** check box in the external service wizard, the file name is returned to the component in a business object. If a file with the specified name already exists, the adapter generates a `DuplicateRecordException` error, and no file is created.

If the `GenerateUniqueFile` property is set to `true`, the adapter generates a unique file name and ignores the value specified in the `Filename` property. The name of the unique file that is generated by the adapter is in the form of a random number prefixed by the business object name, with a file extension of `.tmp`, for example: `Customer23423.tmp`.

**Note:** In the case of a wrapper business object, if a value is unset for the `GenerateUniqueFile` property on the wrapper, the value set on the interaction specification property is used.

If the `FileSequenceLog` managed connection property is specified, the adapter appends a sequence number to the output file name specified in the request and the next request uses the sequence number in the sequence file. For example, if the output file name in the request is `Customer.txt`, a file with the name `Customer.n.txt` is created, where *n* is the sequence number for a particular request, starting with 1. If another request with an output file name of `Order.txt` is received, the sequence number is incremented with 1 and `Order.2.txt` is generated. No new sequence number is created for each individual file name. If the output file name does not have an extension, the sequence is appended at the end of the file name. For example, if the output file name in the request is `Customer`, a file with the name `Customern` is created.

To avoid having to set the output directory and file name in the business object for each request, you can generate file sequencing for a particular type of request by setting the output directory and file name at the managed connection level. When the adapter receives a request to create a file, it checks the file sequence log to see whether a file with that name already exists. If one does, the adapter uses the file sequence number to create a new file name.

**Note:** The directory path and file name specified in the business object take precedence over the managed connection property values.

In a clustered environment, an environment in which you have one instance of the adapter running on several systems, the sequence file specified by the

FileSequenceLog property must be on a mapped drive that is accessible by all the nodes in the cluster. The adapter must have write permission for the sequence log file, or an IOException error is returned.

If the FileSequenceLog property is specified and the GenerateUniqueFile property is enabled, the GenerateUniqueFile property takes precedence over the FileSequenceLog property. The sequence number will continue to increment after an adapter restart. If the sequence file is deleted manually, the sequencing starts again from 1. You can reset the sequence by changing the sequence value in the sequence file.

#### **Delete operation:**

The Delete operation deletes a specified file.

#### **Delete**

Optionally, you can select to return the output of the delete operation to a component in a business object. If you select the **Enable response type for the operation** check box in the external service wizard, the adapter returns true if the file is deleted successfully. If the file permission for delete is not present, the adapter returns false.

If the file does not exist, the adapter generates a RecordNotFoundException error.

#### **Exists operation:**

The Exists operation checks to see whether a specified file exists.

#### **Exists**

If the file that is specified exists, a successful response is returned to the component in the form of a business object. The business object has one attribute, which is set to true if the file exists or false if the file does not exist. If the file does not exist, or if the directory does not exist, the adapter returns false.

#### **List operation:**

The List operation lists the file names in the specified directory.

#### **List**

If the directory does not exist, the adapter generates a RecordNotFoundException error.

#### **Overwrite operation:**

The Overwrite operation overwrites the specified file with the content specified in the request.

If you select the **Enable response type for the operation** check box in the external service wizard, the file name is returned to the component in a business object. If a staging directory is specified in the StagingDirectory property, the file that is to be overwritten is copied from the output directory to the staging directory, and the content is overwritten for that file in the staging directory. The file is then moved

back to the output directory. If a staging directory is not specified, the content is overwritten on the file in the output directory.

**Note:** A staging directory can be configured only if the file content is to be written before the Overwrite operation returns the resultant values. A staging directory cannot be used if the Overwrite operation returns an output stream and the component writes to this stream.

When the input request is received as a FlatFileOutputStreamRecord record, the adapter returns an output stream.

If the CreateIfFileNotExists property is set to true, the adapter creates a new file. The GenerateUniqueFile property has been deprecated. Although you can currently set this property, but the adapter will always treat the value for this property as false.

If the file to be updated does not exist and the CreateFileIfNotExists property is set to false, the adapter generates a RecordNotFoundException error.

**Note:** In the case of wrapper business objects, if the value is unset for the CreateFileIfNotExists property on the wrapper, then the value set on its interaction specification property are used

#### **Retrieve operation:**

The Retrieve operation retrieves the content of the specified file and returns it in the form of a business object. During outbound processing, you can also delete and archive the file returned during the retrieve operation.

During a retrieve operation, the content of the file specified in the Retrieve request is retrieved and returned in the form of a generic or a content-specific business object. The adapter uses the file splitting feature to divide a large file into smaller chunks, which are then retrieved separately. The file content is split according to the SplittingFunctionClassName and SplitCriteria properties defined in the interaction specification. These properties contain the outbound connection properties the adapter uses to interface with the file system. If a data handler is configured, the adapter returns a content-specific business object; otherwise, it returns a generic business object.

To delete the original file after it has been retrieved, set the DeleteOnRetrieve property in the interaction specification. To archive the file before it is deleted, set the ArchiveDirectoryForDeleteOnRetrieve property.

During the retrieve operation, if the file that is specified in the Retrieve request does not exist, the adapter generates a RecordNotFoundException error.

**Note:** In the case of a wrapper business object, if a value is unset for the DeleteOnRetrieve property on the wrapper, the value set on the interaction specification property is used.

#### **Outbound data transformation**

During outbound processing, the adapter performs data transformation based on the adapter-specific data binding and data handler you select when you configure the adapter for outbound processing in the external service wizard.

## Outbound processing with data transformation

During outbound processing, the adapter transforms business objects to the data format expected by the application. The process is controlled by an adapter-specific data binding and data handler that you select when you configure the module for outbound processing.

Figure 2 illustrates the way data is transformed during outbound processing.

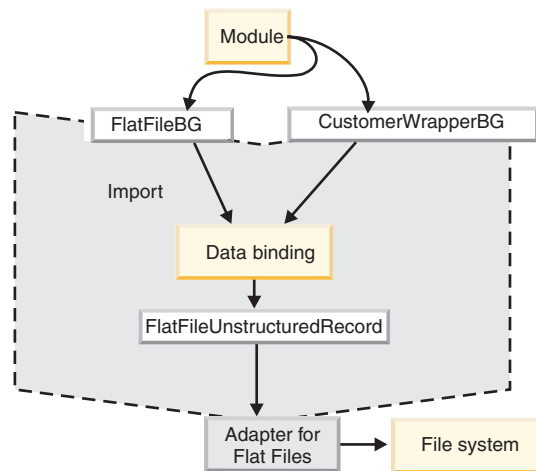


Figure 2. Data transformation during outbound processing

The following steps describe outbound processing with data transformation.

1. For all operations except Retrieve, the adapter performs data transformation based on the input data type and the configured data handler. If the input type is not a generic type (FlatFile or FlatFileBG), the adapter transforms the data. For the Retrieve operation, the adapter transforms the data only if the data handler property of the data binding is configured.
2. The configured data binding is invoked to process the business object.
3. The data binding checks the value specified for the data handler property in the data binding properties, and invokes a content-specific data handler based on the value set for the data handler property.
4. The adapter performs the requested operation on the file and may return a response business object:
  - For the Create, Append, and Overwrite operations, if output is configured, the response business object contains the file name.
  - For the List operation, the response business object contains a list of files in the specified directory.
  - For the Exists operation, the response business object contains a value of either true or false.
  - For the Retrieve operation, the content of the retrieved file is returned in the form of a generic or content-specific response business object.
  - For the Delete operation, if output is configured, the response business object contains a value of either true or false

## Outbound processing without data transformation

For all operations except Retrieve, if the input data type is a generic type (FlatFile or FlatFileBG), the adapter performs outbound processing without data transformation. For Retrieve operations, if no value is set for the data handler property of the data binding, no data transformation takes place. During this type of processing, a special data structure, UnstructuredContent, is used to hold the content.

Figure 3 illustrates outbound processing without data transformation.

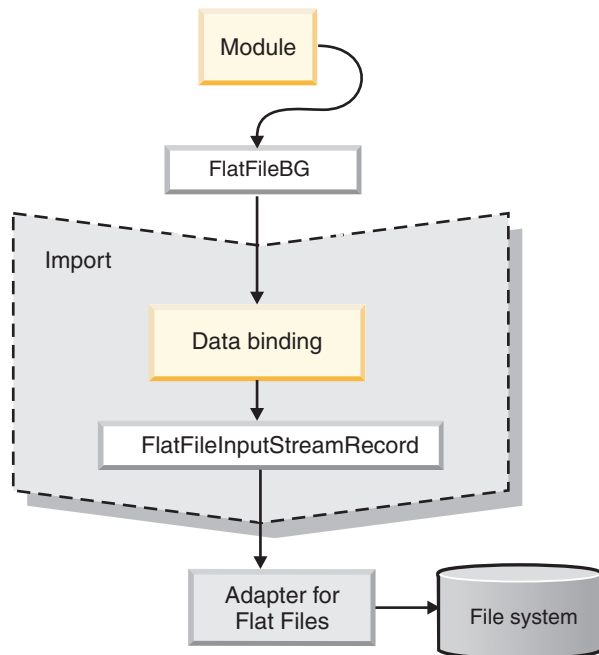


Figure 3. Outbound processing without data transformation

The following steps describe outbound processing without data transformation.

1. For all operations except Retrieve, the adapter checks the input type of the request data object. If the input type is a generic type (FlatFile or FlatFileBG), the adapter does not perform any data transformation on the incoming object. For the Retrieve operation, the adapter checks for the data handler property. If no value is specified, it does transform the data.
2. The configured data binding is invoked to process the business object.
3. For the Retrieve operation, the adapter checks the data handler property. If no value is set for the data handler, the adapter does not transform the data.
4. The adapter performs the requested operation on the file and may return a response business object as follows:
  - For the Create, Append, and Overwrite operations, if output is configured, the response business object contains the file name.
  - For the List operation, the response business object contains a list of files in the specified directory.
  - For the Exists operation, the response business object contains a value of either true or false.

- For the Retrieve operation, the content of the retrieved file is returned in the form of a generic or content-specific response business object.
- For the Delete operation, if output is configured, the response business object contains a value of either true or false.

## File splitting

To support files that contain multiple records, the adapter provides an optional file splitting feature. When this feature is used, the adapter divides large files into smaller chunks, which are then retrieved separately.

Depending upon the type of content contained in the file, the file can be split by delimiter or by size.

- When the content of the business object has a definite structure, for example, if it contains elements such as name, address and city, the file is split by delimiter.
- When the business object contains unstructured data, such as plain text or binary files, the file is split by size.

By default, the adapter splits files by size.

The value specified in the SplitCriteria property determines the method that is used. The default value for SplitCriteria property is zero, which means that no splitting is performed. You can also leave the values of the SplitCriteria and SplittingFunctionClassName properties empty if no splitting is required.

You can optionally provide a custom file splitter class. Set the SplittingFunctionClassName property to the name of the class.

## File splitting by delimiter

When one or more characters such as a comma (,), semicolon (;), quotation mark (", '), brace ({}), or slash (/ \) (delimiters) are used to separate the business objects in a file, the adapter can split the file into smaller chunks based on the delimiter. You define the delimiter that separates the business objects in the file in the SplitCriteria property.

You can enable file splitting by delimiter by specifying the value of the SplittingFunctionClassName property as `com.ibm.j2ca.utils.filesplit.SplitByDelimiter`.

The following rules apply to the use of delimiters:

- All new lines in the delimiter are represented by platform-specific newline characters. The platform-specific newline characters are shown in Table 1.

Table 1.

Platform	Newline character
Macintosh	\r
Microsoft® Windows®	\r\n
UNIX®	\n

- If there is more than one delimiter, each delimiter must be separated by a semicolon (;). The delimiters are matched in the order in which they are given. If the semicolon is part of the delimiter, it must be escaped as \;. For example, if the delimiter is ##\;##, it is processed as ##;##.

- To skip content that is part of the delimiter, specify a double semicolon (;;) in front of it so that the content between the delimiters is skipped. For example, if the event file contains a business object in the following format and the delimiter is ##;\$\$, the adapter considers ##\$\$ as the delimiter and skips content skipped by the adapter:

```
Name=Smith
Company=IBM
##content skipped by the adapter$$
```

- The delimiter can have any value, and there are no restrictions on it. The delimiter is a combination of a valid string, the newline character (for example, \n), and a semicolon separator if there is more than one delimiter. A delimiter does not have to comprise the newline character and a semicolon. The newline character is used only when a newline is to be considered when splitting the contents of the file. Examples of valid delimiters include:

- ####;\n;\n
- ####;\$\$\$\$;\n;####
- %%%;\$\$\$\$;#####
- \n;\n;\$\$\$\$
- ####\;####;\n;\$\$\$\$
- \n;\n;\n
- ####;\$\$\$\$
- \r
- \r\n
- \$\$\$;\r\n

- If the delimiter is located at the end of the file, the SplitCriteria property uses END\_OF\_FILE to determine the physical end of the file.

Example of a common scenario and the recommended delimiter format:

Table 2.

Data binding	BO content	Recommended delimiter format
XML	<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;customer:Customer xsi:type="customer:Customer" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:customer="http://www.ibm.com/xmlns/prod/websphere/ j2ca/flatfile/customer"&gt; &lt;CustomerName&gt;Deepa&lt;/CustomerName&gt; &lt;Address&gt;IBM&lt;/Address&gt; &lt;City&gt;Bangalore&lt;/City&gt; &lt;State&gt;KA&lt;/State&gt; &lt;/customer:Customer&gt;</pre>	\n

## File splitting by size

The value specified in the SplittingFunctionClassName property determines whether a file is split by size. If the SplittingFunctionClassName property is set to com.ibm.j2ca.extension.utils.filesplit.SplitBySize, the SplitCriteria property must contain a valid number that represents the maximum file size, in bytes. If the file is larger than the value specified in the SplitCriteria property, the file is split into chunks and each chunk is posted to the import separately. If the file is smaller than the SplitCriteria value, the entire file is posted to the import.



When event files are split into chunks, each chunk becomes a business object. This means that the value specified for the `PollQuantity` property and the number of business objects delivered to the import can be different. Although the adapter polls according to the `PollQuantity` value, it actually processes the number of business objects in the file one at a time. For example, if an event file is chunked into three parts, one file will be polled and three business objects will be delivered to the import (because each chunk creates a single business object).

At the import, the adapter does not reassemble the chunked data into a single file, but it provides information about the chunks to enable WebSphere Process Server or WebSphere Enterprise Service Bus to reassemble them into a single file. The chunk information is included in the `ChunkFileName` property of the `FlatFileInputStreamRecord` record, and includes the chunk size in bytes and the event ID. The event ID of a chunk uses the following form: `eventFileLocation/_timestampStr/_MofN`, where `M` is the current chunk number and `N` is the total number of chunks. An example event ID would look like this:

```
C:\flatfile\eventdir\eventfile.in/_2005_01_10_10_17_49_864/_3of5, where
timestampStr has the following format:
year_month_day_hour_minutes_seconds_milliseconds.
```

### **Generating unique file names**

To generate unique file names during Create operations, add a persistent sequence number to the default file name or use random numbers to generate file names. During Append and Overwrite operations, use the random number method.

There are two ways to generate unique file names during Create operations:

1. Add a persistent sequence number to the default file name. This method is recommended, especially in a clustered environment.
2. Use random numbers to generate unique file names without any persistence.

For Append and Overwrite operations, you must use the random number method.

### **Generating unique file names using a persistent sequence number**

To generate unique file names using a persistent sequence number, specify:

- The sequence file, which is the complete path of the file where the sequence numbers are stored
- The default target file name

The adapter generates a file name that consists of the default target file name with the sequence number appended to it.

The properties that control the generation of unique file names are present in three places:

- The managed connection factory properties (the Default target file name and Sequence file properties)
- The interaction specification properties (the Default target file name and Generate a unique file properties)
- The wrapper business object

The properties in the business object take precedence over the properties in the interaction specification, which take precedence over the managed connection

factory properties. Unless you want to specify properties for individual business objects, use the properties in the managed connection factory to control the generation of file names.

If the default file name has an extension, the sequence number is appended before the extension. For example, if the default file name is `Customer.txt` in the managed connection factory, the output file names created are `Customer.1.txt`, `Customer.2.txt`, and so on.

For each request, the adapter increments the number in the sequence file, and the input type takes the sequence number that is currently stored in the sequence file. Sequence numbers are not maintained separately for different input data types.

For compatibility with sequence files generated with previous versions of the adapter, where sequence numbers were maintained separately for different input data types, the adapter checks for all entries in the file that have the older format (`<dirPath>/Customer.txt = 2`, where `Customer.txt` is the default file name and 2 is the sequence number to be used when the adapter receives another Create request on the same file). The adapter searches for all such sequence numbers for each input type and uses the highest sequence number as the sequence number for the next input type. The adapter then overwrites the entire file with the new (incremented) sequence number.

**Important:** Unless they are part of a cluster, two adapter instances should not access the same sequence file, because this could result in delayed processing of batch requests.

### Generating unique file names using random numbers

To generate unique file names using random numbers, set the Generate a unique file (`GenerateUniqueFile`) property on the interaction specification or in the business object to `true`. The adapter generates unique file names with the following format: `ffa[RandomNumber].tmp`, where *RandomNumber* is the random number that the adapter has generated. For example, `ffa23423.tmp`.

## Inbound processing

Adapter for Flat Files supports inbound event processing. It polls the local file system at specified intervals for events, such as the creation or modification of a file. When it detects an event, it converts the event data into a business object and sends it to the module for processing.

The following illustration shows the inbound processing flow for WebSphere Adapter for Flat Files.

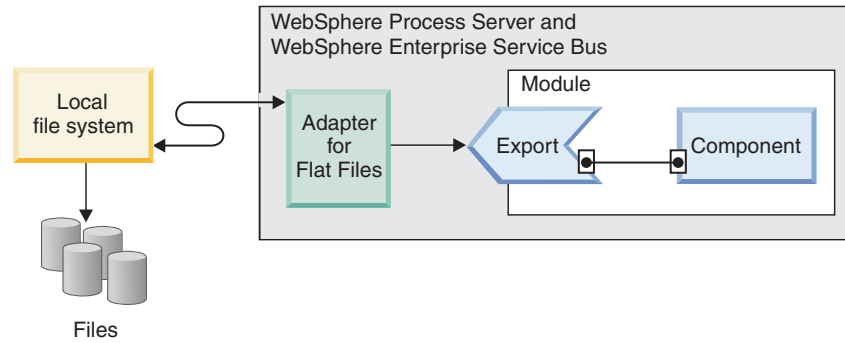


Figure 4. Inbound processing

When a change occurs in the local file system, an event file, which is a new or changed file, is created in a specific directory. You configure this directory as the event directory for the adapter. Although an event file can represent one or more events in the file system, it forms a single unit of transfer to the adapter.

The adapter polls the event directory on the file system at regular intervals, based on the value set in the PollPeriod property. When a file arrives in the event directory, the adapter sends the content of the file to the export. The file content may be sent in its entirety or split into several business objects, or chunks. The adapter sends the business objects to the export by using a function selector, which selects an operation to invoke on the component and provides the correct data binding.

The inbound processing flow is as follows:

1. Events, in the form of files, are generated in the file system.
2. The adapter polls the event directory.
3. The adapter assigns each event an event ID and stores the event ID in the event store. The event store is a persistent cache where event records are saved until a polling adapter can process them. You must create this database before you configure the adapter. The default name of the database is FFDB.
4. The adapter reads each event file as bytes. If file splitting is enabled, the adapter parses the event file based on the values set in the SplittingFunctionClassName and SplitCriteria properties:
  - If splitting is based on a delimiter, the class that performs this functionality and the split criteria are provided.
  - If splitting is based on file size, the class name that performs this functionality is provided.
5. If the configured data type is object-specific, for example, CustomerWrapper, the data handler is configured on the DataBinding, and the adapter transforms the data. If the configured data type is FlatFile or FlatFileBG, the adapter passes the content of the file as a byte array within a FlatFile business object, and no transformation is performed.

**Note:** If file splitting is enabled, the business object contains the file size and the event ID.

6. The adapter sends the business object to the export through a function selector, which selects an operation to invoke on the component and provides the correct data binding. After the business object has reached the export, the event is deleted from the event store. If archiving is enabled, the event is moved to an archiving table before it is deleted.

## Polling files in subdirectories

By default, when the adapter polls files in the event directory, it polls files from the root directory only and ignores files in the subdirectories. If you set the `PollSubDirectories` property in the activation specification to `True`, the adapter first polls the files in the root directory and then polls the files in the subdirectories. After the adapter has retrieved all the files, it sorts them according to the value set for the `SortEventFiles` property. The adapter then processes the files according to the value set for the `PollQuantity` property and sends the business objects to the downstream components.

## Event archiving

To keep track of successfully polled events, you can configure an archive directory on the file system by using the `ArchiveDirectory` activation specification property in the external service wizard. The files are copied to the archive directory with either a `success` or `fail` extensions, as specified in the activation specification.

## Event file locking

File locking behavior is operating system dependent. On Windows, if any of the files being polled by the adapter from the event directory are in use by another application and in the process of being copied to the event directory, they are not made available to the adapter for processing.

However, in UNIX environments, such as AIX®, there is no file locking mechanism that prevents applications from accessing files that are being written to. A file that is being copied to the event directory by another application is made available to the adapter for processing, causing erroneous results. There is no platform-independent way in Java™ to check whether a file is being written to.

To prevent this situation from occurring, you can first copy the event file to a staging directory and then move it to the event directory using the `move` command. Some sample UNIX scripts are provided as part of the adapter. The script file named `CheckIfFileIsOpen.sh` is available in the `Unix-script-file` folder in the adapter installer.

## Rule based filtering of events

The adapter supports the rule based filtering of events which is optional for inbound processing. You can filter the events based on multiple rules. You can define a combination of these rules, group them with Boolean logic and filter the events by following metadata:

- `FileName`
- `File Size`
- `Directory`
- `Last Modified`

For example, you can use `FileName "MatchesPattern" *.txt`, where `FileName` is the property type, `"MatchesPattern"` is the operator and `"*.txt"` is the value.

The rule will be applied over the files filtered by the event file mask criteria. By default, event file mask will have `"*.*"` as default value.

Rule based filtering do not support logical `"OR"` operator between multiple rules.

**Note:** Adapter does not support rule based filtering when the EIS is on MVS™ platform

Table 3. Metadata filtering properties

Property	Valid operators	Value	Prerequisites
FileName	MATCHES_FILE_PATTERN	For example: *.txt	Nil
	MATCHES_REGEX	Java Regular Expression	
FileSize	GT (greater than), LT (less than), GE (greater than or equal to), LE (less than or equal to), EQ (equal to), NE (not equal to).	Numeric value in Bytes. For example: FileSize LE 3628927	Nil
Directory	MATCHES_REGEX	Java regular expression	pollSubDirs = true
LastModified	GT (greater than), LT (less than), GE (greater than or equal to), LE (less than or equal to), EQ (equal to), NE (not equal to).	Day of the week or Time. The day-of-the-week value will accept only EQ (equal to) operator. For example :(LastModified EQ MONDAY), (LastModified EQ 20:41:10)	Nil
END-OF-RULE	END-OF-RULE	END-OF-RULE	Nil

## Event persistence

The adapter supports event persistence for inbound processing in case of abrupt termination. Event persistence (or assured-once delivery) is a way to make sure that events are delivered once, and only once, to the export in the case of a failure. During event processing, the adapter persists the event state in an event store located on the data source. You must set up this data source using WebSphere Process Server or WebSphere Enterprise Service Bus before you can create the event store. To use the recovery feature provided by WebSphere Process Server or WebSphere Enterprise Service Bus, you set the AssuredOnceDelivery property in the activation specification to true. This recovery feature is enabled by default.

The adapter also provides for event persistence by using an in-memory representation of the event store. When you use this feature, you do not need to create a JNDI data source or an external event store, and event processing is faster. However, with this feature there is no support for event recovery. In the case of server failure, the in-memory event stores are lost. To prevent the loss of events in the case of server failure, the recommended approach is to use the database event store.

To use the in-memory event persistence capability of the adapter, you must set the AssuredOnceDelivery property to false, or the adapter will log a warning message.

### Related reference

Chapter 3, “Samples and tutorials,” on page 45

To help you use WebSphere Adapters, samples and tutorials are available from the Business Process Management Samples and Tutorials Web site.

## Event store

The event store is a persistent cache where event records are saved until the polling adapter can process them. The adapter uses event stores to keep track of inbound events as they make their way through the system. Each time a file is created, updated, or deleted, the adapter updates the status of the event in an

event store. The status of each event is continually updated by the adapter for recovery purposes until the events are delivered to the export.

If the adapter detects that there is no event store for the inbound module in the local file system, it automatically creates one when the application is deployed to the runtime. Each event store created by the adapter is associated with a specific inbound module. The adapter does not support multiple adapter modules pointing to the same event store.

When the adapter polls the local file system, it creates an entry in the event store for each event that matches the search criteria specified in the activation specification properties. The adapter records the status of each new entry as NEW.

If an event is successfully posted, event store entries are deleted. For failed events, the entries remain in the event store. Optionally, the adapter can archive successfully polled event files in an archive directory.

**Note:** Failed events can result from incorrect data in the event file. For example, a field named `fname` could appear as `fnam`. The only way to correct the situation is to resend the event file with the correct data.

The adapter provides assured-once event delivery. This means that each event is delivered once and only once. If you set the `AssuredOnceDelivery` activation specification property to `True`, the adapter stores an XID (transaction ID) value for each event in the event store. When an event is obtained for processing:

1. The XID value for the event is updated in the event store.
2. The event is delivered to its corresponding export.
3. The event is deleted from the event store.

The following figure illustrates the event management flow for the adapter.

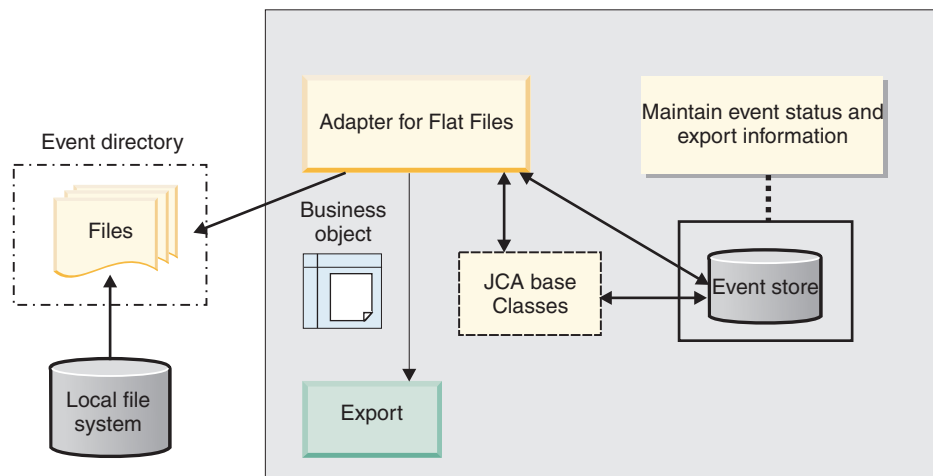


Figure 5. Event management flow

#### Event store structure:

The event store is used by the adapter to track events. The following table notes the values that are stored for each event.

Table 4. Event store structure

Column name	Type (length)	Description
EVNTID	Varchar(255)	Used to track events during inbound processing. Each event requires an event ID for tracking purposes. This must be a unique identifier in the table.
EVNTSTAT	Integer	<p>The status of the event. The adapter uses the status to determine whether an event is new or in process.</p> <p>Event status values:</p> <p><b>NEW (0)</b></p> <p>The event is ready to be processed.</p> <p><b>PROCESSED (1)</b></p> <p>The adapter successfully processed and delivered the event.</p> <p><b>FAILED (-1)</b></p> <p>The adapter was unable to process this event due to one or more problems.</p>
XID	Varchar(255)	Used by the adapter for assured event delivery and recovery.
EVNTDATA	Varchar(255)	Used to track failed events so that they will not be processed again during recoveries. Failed events are marked "ARCHIVED."

#### Event archival values:

You can configure the adapter to archive processed event files in a directory, which you can then access to obtain a list of processed events. The file extension reflects whether an archived event was successful or not.

All archived events in the specified archive directory are stored with success, failure, and original file extensions. Success is used when the event processing is successful. If the event processing fails, the file is archived with failure and original extensions. If the event file has multiple business objects and some of them are successful, there is also a file with a success extension.

The archive extensions are configurable based on the following activation specification properties: FailedArchiveExt, OriginalArchiveExt, and SuccessArchiveExt.

The following table lists the archive extensions used by the adapter.

Table 5. Event archive values

Extension	Definition	Format
SUCCESS	The event file was delivered to the export.	<filename>_<timestamp>.SUCCESS
FAIL	The event file was not delivered to the export.	<filename>_<timestamp>.FAIL
ORIGINAL	The original event file that was picked up for processing.	<filename>_<timestamp>.ORIGINAL

## Function selectors

During inbound processing, a function selector returns the appropriate operation to be called on the service. You choose a function selector when you configure the adapter for inbound processing in the external service wizard. The adapter provides two function selectors, `FilenameFunctionSelector` and `EmbeddedNameFunctionSelector`.

### FilenameFunctionSelector

`FilenameFunctionSelector` is a rule-based function selector that provides object name resolution based on regular expressions that map to file names. A regular expression is a string that is used to describe or match a set of strings according to certain syntax rules.

The following table shows examples of matching rules, where a rule consists of the `ObjectName` and `Rule` fields.

Table 6. Examples of matching rules for `FilenameFunctionSelector`

FileName	ObjectName	Rule
Customer0001.txt	Customer	CUST.*TXT
22310RZ93.z21	Order	[0-9]*OR[A-Z][0-9]{2}.*
22310RZ93.z21	Order	*OR.*

Note that the rules in the second and third rows resolve to the same name, but the rule in the second row is less “greedy” because it requires a specific sequence of numbers and letters in order for the file name to match, whereas the rule in the third row resolves anything with the characters “OR” in the file name. The character combination “.” indicates that any character may occur any number of times.

To generate the native function name, the function selector prepends `emit` to the object name that you provide. For example, if the object name is `Customer`, the function selector returns the function name `emitCustomer`. The object name should be the payload object name, for example, `Customer` or `Order`, and not the wrapper or business graph name. For pass-through scenarios, use `FlatFile` as the object name.

You can configure `FilenameFunctionSelector` with multiple rules, each containing an object name and a regular expression to match against the file name. If more than one rule matches, the function selector returns the object name based on the first matching rule. If no rule matches, the adapter generates an error. If no rules are present in the configuration, the function selector uses the function name `emitFlatFile`.



For a detailed explanation of the rules governing the use of regular expressions, see the Java Class Pattern documentation at <https://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html>.

## **EmbeddedNameFunctionSelector**

`EmbeddedNameFunctionSelector` is used for content-specific business objects, where the object name is embedded in the event file. It returns the function name based on the desired content data, not on the wrapper. For example, if the content-specific business object is `CustomerWrapperBG`, the function returned by the function selector is `emitCustomer`.

`EmbeddedNameFunctionSelector` must be configured with a data handler. The data binding must be the adapter-specific `WrapperDataBinding`, and it must be configured to use the same data handler that is configured with the function selector.

### **Related reference**

“Connection properties for the wizard” on page 156

Connection properties are used to build a service description and save the built-in artifacts. These properties are configured in the external service wizard.

## **File splitting**

To reduce memory loading during event processing, the adapter supports an optional file splitting feature. When this feature is used, the adapter divides large event files into smaller chunks, which are then posted separately to the export.

The adapter splits large event files into several business objects, also called chunks, based on the value specified in the `SplitCriteria` property, which can be either a delimiter or a chunk size. Each business object is delivered to the export separately. You split files by delimiter when the content of the business object has a definite structure; for example, if you have a `Customer` business object with elements such as name, address, and city. You split files by size when the business object contains unstructured data, such as plain text or binary files.

When event files are split into such chunks, each chunk creates a business object. This means that the value specified for the `PollQuantity` property and the number of business objects delivered to the export can be different. When file splitting based on a delimiter is enabled, the `PollQuantity` activation specification property specifies the number of such event files that are present in the event store, and the class used to split the event file is set in the `SplittingFunctionClassName` activation specification property.

The adapter does not reassemble the chunked data.

The value specified in the `SplitCriteria` property determines the method that is used. The default value for `SplitCriteria` property is zero, which means that no splitting is performed. You can also leave the values of the `SplitCriteria` and `SplittingFunctionClassName` properties empty if no splitting is required.

You can optionally provide a custom file splitter class. Set the `SplittingFunctionClassName` property to the name of the class.

### **File splitting by delimiter**

When one or more characters such as a comma (,), semicolon (;), quote (" , ' ), brace ({} ) or slash ( / \ ) (delimiters) are used to separate the business objects in a

file, the adapter can split the file into smaller chunks based on the delimiter. Each chunk is a logical unit that is used to construct a business object when forwarded to WebSphere Process Server or WebSphere Enterprise Service Bus. You define the delimiter that separates the business objects in the file in the SplitCriteria property.

To demonstrate how the PollQuantity value works with delimiter file splitting, consider two event files. The first event file contains a business object and the second event file contains two business objects. If the PollQuantity value is 2, the first business object from the first event file and the next business record from the second event file are sent in the first poll cycle. The second business object from the second file is sent in the second poll cycle.

The following rules apply to the use of delimiters:

- All new lines in the delimiter are represented by platform-specific newline characters. The platform-specific newline characters are shown in Table 7.

Table 7.

Platform	Newline character
Macintosh	\r
Microsoft Windows	\r\n
UNIX	\n

- If there is more than one delimiter, each delimiter must be separated by a semicolon (;). The delimiters are matched in the order in which they are given. If the semicolon is part of the delimiter, it must be escaped as \;. For example, if the delimiter is ##\;##, it is processed as ##;##.
- To skip content that is part of the delimiter, specify a double semicolon (;;) in front of it so that the content between the delimiters is skipped. For example, if the event file contains a business object in the following format and the delimiter is ##;\$\$, the adapter considers ##\$\$ to be the delimiter and skips content skipped by the adapter:

```
Name=Smith
Company=IBM
##content skipped by the adapter$$
```

- The delimiter can have any value, and there are no restrictions on it. The delimiter is a combination of a valid string, the newline character (for example, \n), and a semicolon separator if there is more than one delimiter. A delimiter does not have to comprise the newline character and a semicolon. The newline character is used only when a newline is to be considered when splitting the contents of the file. Examples of valid delimiters include:

- ####;\n;\n
- ####;\$\$\$;\n;####
- %%%;\$\$\$;\$###
- \n;\n;\$\$\$
- ####\;####;\n;\$\$\$
- \n;\n;\n
- ####;;\$\$\$
- \r
- \r\n
- \$\$\$;\r\n

- If the delimiter is located at the end of the file, the SplitCriteria property uses END\_OF\_FILE to determine the physical end of the file.

Example of a common scenario and the recommended delimiter format:

Table 8.

Data binding	BO content	Recommended delimiter format
XML	<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;customer:Customer xsi:type="customer:Customer" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:customer="http://www.ibm.com/xmlns/prod/websphere/ j2ca/flatfile/customer"&gt; &lt;CustomerName&gt;Deepa&lt;/CustomerName&gt; &lt;Address&gt;IBM&lt;/Address&gt; &lt;City&gt;Bangalore&lt;/City&gt; &lt;State&gt;KA&lt;/State&gt; &lt;/customer:Customer&gt;</pre>	\n

### File splitting by size

The value specified in the `SplittingFunctionClassName` property determines whether a file is split by size. If the `SplittingFunctionClassName` property is set to `com.ibm.j2ca.extension.utils.filesplit.SplitBySize`, the `SplitCriteria` property must contain a valid number that represents the maximum file size, in bytes. If the event file is larger than the value specified in the `SplitCriteria` property, the file is split into chunks and each chunk is posted to the export separately. If the event file is smaller than the `SplitCriteria` value, the entire event file is posted to the export.

When event files are split into chunks, each chunk becomes a business object. This means that the value specified for the `PollQuantity` property and the number of business objects delivered to the export can be different. Although the adapter polls according to the `PollQuantity` value, it actually processes the business objects in the file one at a time. For example, if an event file is chunked into three parts, one file will be polled and three business objects will be delivered to the export (because each chunk creates a single business object).

At the export, the adapter does not reassemble the chunked data into a single file, but it provides information about the chunks to enable WebSphere Process Server or WebSphere Enterprise Service Bus to reassemble them into a single file. The chunk information is included in the `ChunkFileName` property of the `FlatFileInputStreamRecord` record, and includes the chunk size in bytes and the event ID. The event ID of a chunk uses the following form: `eventFileLocation/_timestampStr/_MofN`, where M is the current chunk number and N is the total number of chunks. An example event ID would look like this:

`C:\flatfile\eventdir\eventfile.in/_2005_01_10_10_17_49_864/_3of5`, where `timestampStr` has the following format: `year_month_day_hour_minutes_seconds_milliseconds`.

### Inbound data transformation

During inbound processing, the adapter performs data transformation based on the adapter-specific data binding and data handler that you select when you configure the module in the external service wizard.

## Inbound processing with data transformation

The process of data transformation during inbound processing is controlled by the adapter-specific data binding and data handler that you select when you configure the module. The following steps describe inbound processing with data transformation.

1. Each individual event is retrieved from the event file based on the value set in the SplitCriteria property. The content is set on the record and sent to the data binding.
2. The adapter checks the expected data type of the inbound operation. If it is not a generic type (FlatFile or FlatFileBG), the adapter checks for the data handler property in the data binding.
3. If the data handler is set, the adapter transforms the data. The data binding invokes the data handler and returns a content-specific business object.
4. The adapter passes this content-specific business object to the endpoint by calling the method returned by the function selector.

## Inbound processing without data transformation

If no data transformation is required on the content, for example, when text\xml content must be retained as text\xml content, the event data is not converted into business objects but is passed through as unstructured content.

The following steps describe inbound processing without data transformation.

1. Each individual event is retrieved from the event file based on the value set in the SplitCriteria property. The content is set on the record and sent to the data binding.
2. The data binding checks for the expected type of the event. If it is a generic type (FlatFile or FlatFileBG), the adapter does not transform the data.
3. The data binding sets the content on the UnstructuredContent record and sends it back to the adapter.
4. The adapter passes this business object to the endpoint by calling the method returned by the function selector.

## Business objects

A business object is a logical data container that represents the data that is processed by the adapter. The data can represent either a business entity, such as an invoice or an employee record, or unstructured text, such as the body of an e-mail or a word processing document. The adapter uses business objects to send data to or obtain data from the local file system.

## How the adapter uses business objects

During outbound processing, the adapter:

1. Receives a business object from the module that represents a request to perform an operation on a file in the local file system.
2. If necessary, converts the business object into a format that can be understood by the local file system.
3. Performs the requested operation.
4. Returns a business object, if applicable, that represents the result of the operation to the module.

During inbound processing, the adapter:

1. Retrieves a file from the event directory on the local file system.
2. Constructs a business object out of the data, transforming the data, if necessary, into the required format.
3. Sends the business object to the export

## How business objects are created

You can create business objects by using either the external service wizard or the business object editor, both of which can be launched from WebSphere Integration Developer. If you use the external service wizard, the wizard examines files in the file system and generates business objects to represent them. It also generates other artifacts needed by the adapter.

If you use the business object editor, you create business objects manually. After you create your business objects, you can use the business object editor to define the hierarchy of the business objects.

When you run the external service wizard, the Adapter for Flat Files generates two types of business objects: content-specific and generic. The adapter generates these generic business object XSD files:

- FlatFile.xsd
- FlatFileBG.xsd
- UnstructuredContent.xsd
- FileContent.xsd

An example of a content-specific business object is Customer. If you select Customer, these content-specific XSD files are generated, in addition to the generic XSD files:

- Customer.xsd
- CustomerWrapper.xsd
- CustomerWrapperBG.xsd

**Note:** In this example, the business graph CustomerWrapperBG.xsd is generated. The generation of business graphs is optional.

You can optionally choose, during adapter configuration, to generate a business graph. In version 6.0.2, each top-level business object is contained in a business graph, which includes a verb that an application can use in version 6.0.2 to specify additional information about the operation to be performed. In Version 6.1.x, business graphs are optional; they are required only when you are adding business objects to a module created with a version of WebSphere Integration Developer earlier than Version 6.1.x. If business graphs exist, they are processed, but the verb is ignored.

### Related reference

“Business object information” on page 149

You can determine the purpose of a business object by examining both the application-specific information within the business object definition file and the name of the business object. The application-specific information dictates what operations can be performed on the local file system. The name typically reflects the operation to be performed and the structure of the business object.

## WebSphere Application Server environment variables

WebSphere Application Server environment variables can be used in the external service wizard to specify directory values.

When you configure the adapter for inbound or outbound processing using the external service wizard, you set values for various required local files and directories. You can later change these values in the deployed application from the WebSphere Process Server or WebSphere Enterprise Service Bus administrative console.

With WebSphere Process Server or WebSphere Enterprise Service Bus Version 6.1.x, instead of hard coding values for directories and files, you can declare them as WebSphere Application Server environment variables, and specify the environment variable names when you run the external service wizard. When you deploy your application, the environment variable name is replaced with the actual value and used by the adapter. If you want to change the property value, you can just change the environment variable in the WebSphere Process Server or WebSphere Enterprise Service Bus administrative console.

WebSphere Application Server environment variables can be used for all string property values (not boolean or integer variables) that are set in inbound and outbound configuration.

When you create a WebSphere Application Server environment variable, you specify:

- The name of the environment variable, for example, `EVENT_DIRECTORY`.
- The value that the symbolic name represents, for example: `C:\flatfile\event`.
- The scope for the environment variable, which determines the level at which the environment variable is visible in the administrative console. The scope level can be server, node, or cell:
  - Server scope limits visibility to the named server. The server scope is the most specific scope for defining environment variables.
  - Node scope limits visibility to all the servers on the named node. This is the default scope.
  - Cell scope limits visibility to all servers on the named cell.

To create WebSphere Application Server environment variables, use the WebSphere Process Server or WebSphere Enterprise Service Bus administrative console.

### **Related tasks**

“Defining WebSphere Application Server environment variables” on page 51  
Use the administrative console of WebSphere Process Server or WebSphere Enterprise Service Bus to define WebSphere Application Server environment variables.

### **Related reference**

“Managed connection factory properties” on page 159  
Managed connection factory properties specify information the adapter needs at run time for outbound communication with the local file system.

“Activation specification properties” on page 176  
Activation specification properties hold the inbound event processing configuration information for an export. You set activation specification properties through either the external service wizard or the administrative console.

## The external service wizard

Use the external service wizard to configure your adapter before it is deployed to WebSphere Process Server or WebSphere Enterprise Service Bus. The wizard examines files on the local file system, builds services (based on search criteria you provide), and generates business objects and interfaces.

The external service wizard provides a blueprint for business objects. It allows you to select the artifacts of interest and generates deployable service objects and descriptions. By selecting meta-object nodes from the metadata tree structure, you can generate business objects for EIS or database entities. The metadata is transformed into service data objects consisting of business graphs and business objects.

The following figure illustrates the external service wizard flow. When finished, an EAR file containing all of the information for your adapter project is created. This EAR file can then be deployed to the application server.

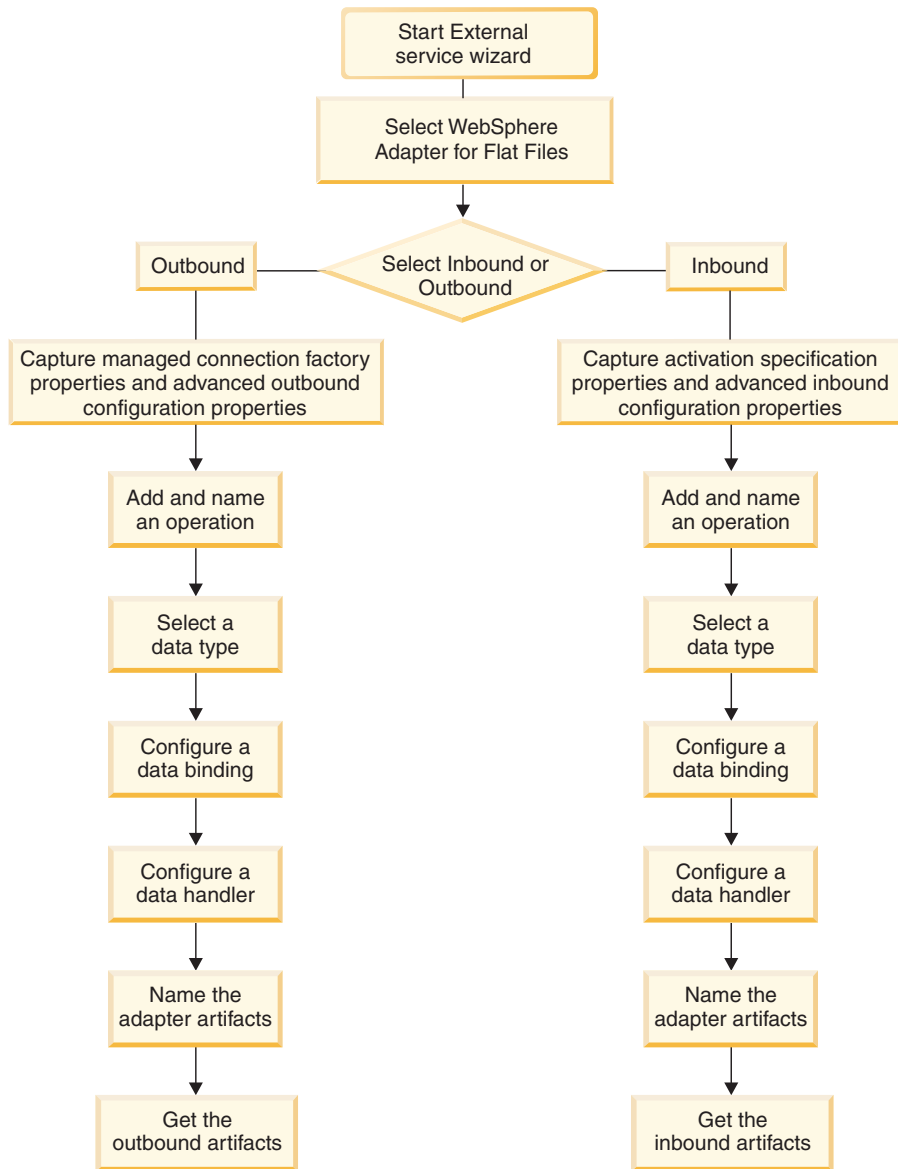


Figure 6. Basic external service wizard flow



---

## Chapter 2. Planning for adapter implementation

To implement WebSphere Adapter for Flat Files, you must plan for inbound and outbound processing and consider security and performance requirements. Also, if you are migrating from an earlier version of WebSphere Adapter for Flat Files, perform any migration tasks.

---

### Before you begin

Before you begin to set up and use the adapter, you should possess a thorough understanding of business integration concepts, the capabilities and requirements of the integration development tools and runtime environment you will use, and the environment where you will build and use the solution.

To configure and use WebSphere Adapter for Flat Files, you should understand and have experience with the following concepts, tools, and tasks:

- The business requirements of the solution you are building.
- Business integration concepts and models, including the Service Component Architecture (SCA) programming model.
- The capabilities provided by the integration development tools you will use to build the solution. You should know how to use these tools to create modules, test components, and complete other integration tasks.
- The capabilities and requirements of the runtime environment you will use for the integration solution. You should know how to configure and administer the host server and how to use the administrative console to set and modify property definitions, configure connections, and manage events.

---

### Security

Adapter for Flat Files relies on the permissions of the user that starts WebSphere Process Server.

The user of the adapter must have sufficient privileges to access the directories and files that the adapter tries to access, read, or modify.

---

### Support for protecting sensitive user data in log and trace files

The adapter provides the ability to prevent sensitive or confidential data in log and trace files from being seen by those without authorization.

Log and trace files for the adapter can contain data from your local file system, which might contain sensitive or confidential information. Sometimes these files must be seen by individuals without authorization to view sensitive data. For example, a support specialist must use the log and trace files to troubleshoot a problem.

To protect the data in situations like this, the adapter lets you specify whether or not you want to prevent confidential user data from displaying in the adapter log and trace files. You can select this option in the external service wizard or change the `HideConfidentialTrace` property. When this property is enabled, the adapter replaces the sensitive data with XXX's.

See “Managed connection factory properties” on page 159 for information about this optional property.

The following types of information are considered potentially sensitive data and are disguised:

- The contents of a business object
- The contents of the object key of the event record
- User name and password
- Business object data in an intermediate form, such as a comma-delimited version of a file

The following types of information are not considered user data and are not hidden:

- The contents of the event record that are not part of the event record object key, for example, the XID, event ID, business object name, and event status
- Business object schemas
- Transaction IDs
- Call sequences

---

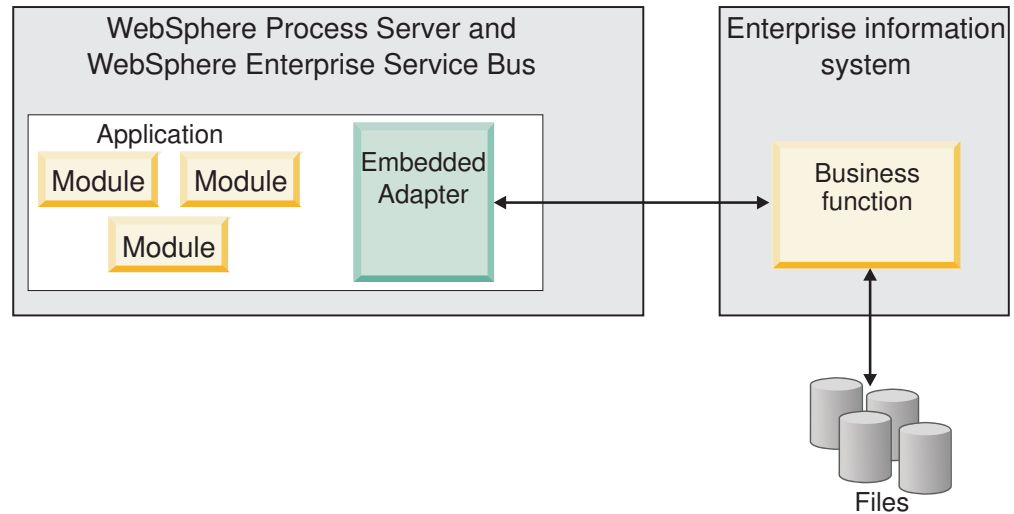
## Deployment options

There are two ways to deploy the adapter. You can either embed it as part of the deployed application, or you can deploy it as a stand-alone RAR file. The requirements of your environment will affect the type of deployment option you choose.

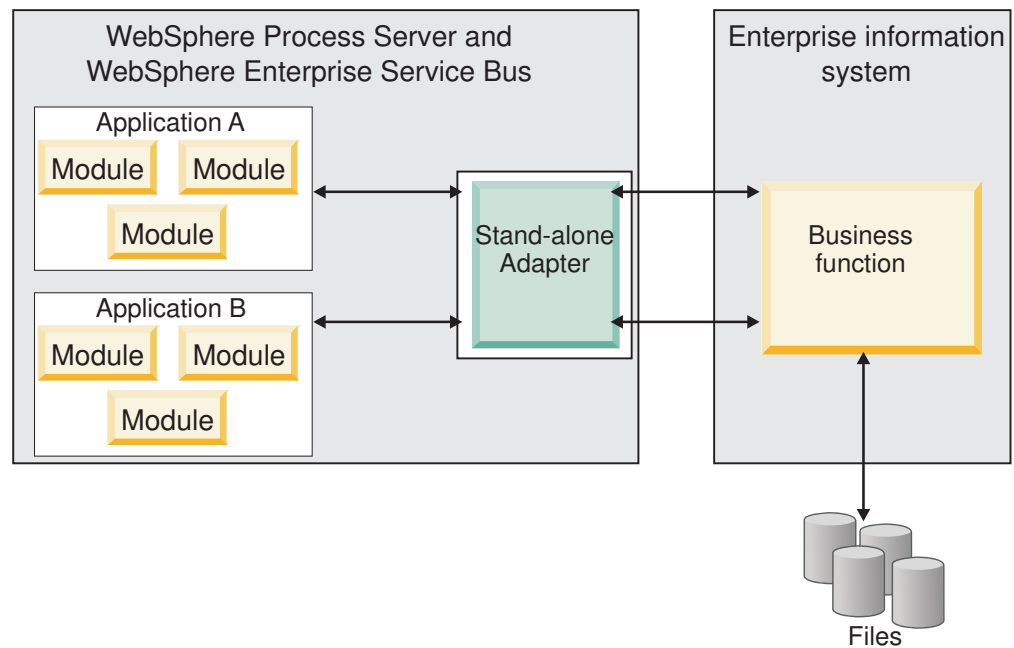
The deployment options are described below:

- **With module for use by single application.** With the adapter files embedded in the module, you can deploy the module to any application server. Use an embedded adapter when you have a single module using the adapter or if multiple modules need to run different versions of the adapter. Using an embedded adapter enables you to upgrade the adapter in a single module without the risk of destabilizing other modules by changing their adapter version.
- **On server for use by multiple applications.** If you do not include the adapter files in a module, you must install them as a stand-alone adapter on each application server where you want to run the module. Use a stand-alone adapter when multiple modules can use the same version of the adapter and you want to administer the adapter in a central location. A stand-alone adapter can also reduce the resources required by running a single adapter instance for multiple modules.

An embedded adapter is bundled within an enterprise archive (EAR) file and is available only to the application with which it is packaged and deployed.



A stand-alone adapter is represented by a stand-alone resource adapter archive (RAR) file, and when deployed, it is available to all deployed applications in the server instance.



While creating the project for your application using WebSphere Integration Developer, you can choose how to package the adapter [either bundled with the (EAR) file or as a stand-alone (RAR) file]. Your choice will affect how the adapter is used in the runtime environment, as well as how the properties for the adapter are displayed on the administrative console.

Choosing either to embed an adapter with your application or to deploy the adapter as a stand-alone module depends on how you want to administer the adapter. If you want a single copy of the adapter and do not care about disruption to multiple applications when you upgrade the adapter, then you would be more likely to deploy the adapter as a stand-alone module.

If you plan on running multiple versions, and if you care more about potential disruption when you upgrade the adapter, you would be more likely to embed the adapter with the application. Embedding the adapter with the application allows you to associate an adapter version with an application version and administer it as a single module.

## Considerations for embedding an adapter in the application

Take into consideration the following items if you plan on embedding the adapter with your application:

- An embedded adapter has class loader isolation.  
A class loader affects the packaging of applications and the behavior of packaged applications deployed on runtime environments. *Class loader isolation* means the adapter cannot load classes from another application or module. Class loader isolation prevents two similarly named classes in different applications from interfering with each other.
- Each application in which the adapter is embedded must be administered separately.

## Considerations for using a stand-alone adapter

Take into consideration the following items if you plan on using a stand-alone adapter:

- Stand-alone adapters have no class loader isolation.  
Because stand-alone adapters have no class loader isolation, only one version of any given Java artifact is run and the version and sequence of that artifact is undetermined. For example, when you use a stand-alone adapter there is only *one* resource adapter version, *one* adapter foundation class (AFC) version, or *one* third-party JAR version. All adapters deployed as stand-alone adapters share a single AFC version, and all instances of a given adapter share the same code version. All adapter instances using a given third-party library must share that library.
- If you update any of these shared artifacts, all applications using the artifacts are affected.  
For instance, if you have an adapter that is working with server version X, and you update the version of the client application to version Y, your original application might stop working.
- Adapter Foundation Classes (AFC) is compatible with previous versions, but the latest AFC version must be in every RAR file that is deployed in a stand-alone manner.  
If more than one copy of any JAR file is in the class path in a stand-alone adapter, the one that is used is random; therefore, they all must be the latest version.

---

## WebSphere Adapters in clustered environments

You can improve adapter performance and availability by deploying the module to a clustered server environment. The module is replicated across all servers in a cluster, regardless of whether you deploy the module using a stand-alone or embedded adapter.

WebSphere Process Server or WebSphere Enterprise Service Bus, WebSphere Application Server Network Deployment, and WebSphere Extended Deployment support clustered environments. Clusters are groups of servers that are managed

together to balance workloads and to provide high availability and scalability. When you set up a server cluster, you create a Deployment Manager profile. The HAManager, a subcomponent of the Deployment Manager, notifies the Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) container to activate the adapter instance. The JCA container provides a runtime environment for adapter instances. For information about creating clustered environments, see the following link: [http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun\\_wlm\\_cluster\\_v61.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_wlm_cluster_v61.html).

Using WebSphere Extended Deployment, you can optionally enhance the performance of adapter instances in your clustered environment. WebSphere Extended Deployment extends the WebSphere Application Server Network Deployment capabilities by using a dynamic workload manager instead of a static workload manager, which is used by WebSphere Application Server Network Deployment. The dynamic workload manager can optimize the performance of adapter instances in the cluster by dynamically balancing the load of the requests. This means that application server instances can be automatically stopped and started based on the load variations, allowing machines with different capacities and configurations to evenly handle load variations. For information on the benefits of WebSphere Extended Deployment, see the following link: <http://publib.boulder.ibm.com/infocenter/wxdinfo/v6r1/index.jsp>.

In clustered environments, adapter instances can handle both inbound and outbound processes.

**Restriction:** During inbound and outbound communication WebSphere Adapter for Flat Files is not able to switch polling between a WebSphere Process Server or WebSphere Enterprise Service Bus cluster backup node and the cluster's primary node when each node is installed on a different operating system. For example, if the adapter starts polling on a primary Windows node, it cannot switch to a backup UNIX node because it cannot process the Windows path used for the directory storing in progress events.

## High availability for inbound processes

Inbound processes are based on events triggered as a result of updates to data in the local file system. WebSphere Adapter for Flat Files is configured to detect updates by polling an event table. The adapter then publishes the event to its endpoint.

**Important:** In a clustered environment, the event directory should be on a shared file system and not local to any of the cluster machines.

When you deploy a module to a cluster, the JCA container checks the enableHASupport resource adapter property. If the value for the enableHASupport property is true, which is the default setting, all of the adapter instances are registered with the HAManager with a policy 1 of N. This policy means that only one of the adapter instances starts polling for events. Although other adapter instances in the cluster are started, they remain dormant with respect to the active event until the active adapter instance finishes processing the event. If the server on which the polling thread was started shuts down for some reason, an adapter instance that is running on one of the backup servers is activated.

**Important:** Do not change the setting of the enableHASupport property.

## High availability for outbound processes

In clustered environments, multiple adapter instances are available to perform outbound process requests. Accordingly, if your environment has multiple applications that interact with WebSphere Adapter for Flat Files for outbound requests, then you might improve performance by deploying the module to a clustered environment. In a clustered environment, multiple outbound requests can be processed simultaneously, as long as they are not attempting to process the same record.

If multiple outbound requests are attempting to process the same record, such as a Customer address, the workload management capability in WebSphere Application Server Network Deployment distributes the requests among the available adapter instances in the sequence they were received. As a result, these types of outbound requests in a clustered environment are processed in the same manner as those in a single server environment: one adapter instance processes only one outbound request at a time. For more information on workload management, see the following link: [http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun\\_wlm.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_wlm.html).

---

## Migrating to version 6.2

By migrating to version 6.2 of WebSphere Adapter for Flat Files, you automatically upgrade from the previous version of the adapter. Additionally, you can migrate your applications that embed an earlier version of the adapter, so that the applications can utilize features and capabilities present in version 6.2.

### Migration considerations

WebSphere Adapter for Flat Files version 6.2 includes updates that might affect your existing adapter applications. Before migrating applications that will use WebSphere Adapter for Flat Files, you must consider some factors that might affect your existing applications.

#### Compatibility with earlier versions

WebSphere Adapter for Flat Files version 6.2 is fully compatible with version 6.0.2.x and version 6.1.x of the adapter and can work with custom business objects (XSD files) and data bindings.

Because version 6.2 of WebSphere Adapter for Flat Files is fully compatible with version 6.0.2.x and version 6.1.x, any of your applications that used version 6.0.2.x or version 6.1.0.x of WebSphere Adapter for Flat Files will run unchanged when you upgrade to version 6.2. However, if you want your applications to use features and functionality present in version 6.2 of the adapter, run the migration wizard.

The migration wizard replaces (upgrades) version 6.0.2.x or version 6.1.x of the adapter with version 6.2 and enables version 6.2 features and functionality for use with your applications.

**Note:** The migration wizard does not create components or modify existing components, such as mappers and mediators to work with version 6.2 of the adapters. If you are upgrading to version 6.2 and any of your applications embed an adapter that is version 6.1.x or earlier, and you want your applications to take advantage of the features and functions in version 6.2, you might need to make changes to those applications.

If artifacts are inconsistent with regard to versioning within a single module, the entire module is marked and will not be selectable for migration. Version inconsistencies are recorded in the workspace log, because they might indicate corruptions to the project.

If the WebSphere Integration Developer version does not match that of WebSphere Adapter for Flat Files, the products cannot be run together.

The external service wizard cannot be run when the following product combinations are used:

*Table 9. Product combinations not supported*

WebSphere Integration Developer	WebSphere Adapter for Flat Files
version 6.2	version 6.1.x
version 6.1.x	version 6.0.2.x
version 6.0.2.x	version 6.2
version 6.0.2.x	version 6.1.x

## Deciding whether to upgrade or to upgrade and migrate

By default, the migration wizard upgrades the adapter and migrates the application artifacts so that the applications can use features and functions in version 6.2 of the adapter. When you choose to upgrade the adapter by selecting a project, the wizard automatically selects the associated artifacts for migration.

If you decide that you want to upgrade the adapter from version 6.0.2.x or version 6.1.x to version 6.2, but you do not want to migrate the adapter artifacts, you can do so by deselecting the adapter artifacts from the appropriate area of the migration wizard.

Running the migration wizard without any adapter artifacts selected will install and upgrade your adapter, but your artifacts will not be migrated and your applications will not be able to take advantage of the features and capabilities that exist in version 6.2 of the adapter.

## Running the migration wizard in a test environment first

Because adapter migration might require you to make changes to those applications that will use version 6.2 of WebSphere Adapter for Flat Files, you should always perform the migration in a development environment first and test your applications before deploying the application to a production environment.

The migration wizard is fully integrated with the development environment.

## Deprecated features

A deprecated feature is one that is supported but no longer recommended and that might become obsolete. The following features from earlier versions of WebSphere Adapter for Flat Files have been deprecated from version 6.1.x and might require changes to your applications:

- Activation specification:
  - ArchivingProcessed
  - EventContentType

- DefaultObjectName
- InteractionSpecification:
  - DefaultObjectName
- Wrapper properties:
  - RetrieveContentType
  - DefaultObjectName

## Performing the migration

You can migrate a project or EAR file to version 6.2, by using the adapter migration wizard. When the tool is finished, the migration is complete and you can work in the project or deploy the module.

### Before you begin

Review the information in *Migration considerations*.

### About this task

To perform the migration in WebSphere Integration Developer, complete the following steps.

**Note:** After migration is complete, the following changes will occur:

- the module will no longer be compatible with previous versions of WebSphere Process Server or WebSphere Enterprise Service Bus, or WebSphere Integration Developer.
- an XML data handler will be added to all the operations. Because this data handler is not needed for the pass-through operation, you must configure one data binding without the data handler against the pass-through operation.

**Note:** The following steps describe how to run the adapter migration wizard from the connector project context menu while in the Java EE perspective in WebSphere Integration Developer.

### Procedure

1. Import the PI (project interchange) file for an existing project or the EAR (enterprise archive) file for a deployed application into the workspace.
2. Change to the Java EE perspective.
3. Right-click the module and select **Update** → **Update Connector Project**.

**Note:** Additionally, you can launch the adapter migration wizard in the following ways:

- Right-click the project in the Java EE perspective and select **Update** → **Migrate Adapter Artifacts**.
- From the Problems view, right-click a migration-specific message and select **Quick Fix** to correct the problem.

When you launch the migration wizard from the connector project context menu while in the Java EE perspective, by default all of the dependent artifact projects are selected. If you deselect a dependent artifact project, that project is not migrated. You will be allowed to migrate any projects that you deselect at a later time. Previously migrated projects, projects with a current version, and projects that contain errors are unavailable for migration and are not selected.

4. Review the tasks and warnings presented on the welcome page, and click **Next**.



5. A warning window opens with the message, “The properties that are not supported in the version of the target adapter will be removed during the migration.” Click **OK** to continue to the next step.
6. On the Review changes window you can review the migration changes that will occur in each of the artifacts that you are migrating by clicking the + sign.
7. Click **Finish** to perform the migration.

Prior to running the migration process, the wizard performs a backup up of all projects affected by the migration. The projects are backed up to a temporary folder within the workspace. Should the migration fail for any reason, or should you decide to cancel the migration before it completes, the wizard deletes the modified projects and replaces them with the projects stored in the temporary folder.

Upon completing the migration successfully, all backed up projects are deleted.

8. Check the Problems view for messages from the migration wizard, which start with the string CWPAD.
9. If you are migrating an EAR file, optionally create a new EAR file with the migrated adapter and artifacts, and deploy it to WebSphere Process Server or WebSphere Enterprise Service Bus. For more information about exporting and deploying an EAR file, see the topics devoted to it in this documentation.

## Results

The project or EAR file is migrated to version 6.2. You do not need to run the external service wizard after exiting the adapter migration wizard.

## Updating but not migrating a version 6.0.2.x or version 6.1.x project

You can upgrade the adapter from version 6.0.2.x or version 6.1.x to version 6.2, while choosing not to migrate the adapter project artifacts.

### About this task

Because the internal name of the adapter changed in version 6.2, artifacts in a version 6.0.2.x or version 6.1.x project must be updated to use the new name before you can use the adapter wizard in WebSphere Integration Developer, version 6.2. Use the migration wizard to update a version 6.0.2.x or version 6.1.x project. Then use the Quick Fix feature of WebSphere Integration Developer to change the adapter name in project artifacts.

### Procedure

1. Import the project interchange (PI) file into the workspace.
2. In the Java EE perspective, right-click the project name and click **Update** → **Update Connector Project**. The adapter migration wizard opens.
3. On the welcome page, click **Next**.
4. On the Select Projects window, deselect the dependent artifact projects, and then click **Finish**.
5. A warning window opens with the message, “The properties that are not supported in the version of the target adapter will be removed during the migration.” Click **OK** to continue to the next step.
6. In the Quick Fix window, make sure the fix **Rename the referenced adapter** is selected, and then click **OK**.

7. If the error remains visible, click **Project** → **Clean**, select the project you just updated, and then click **OK**.

## Results

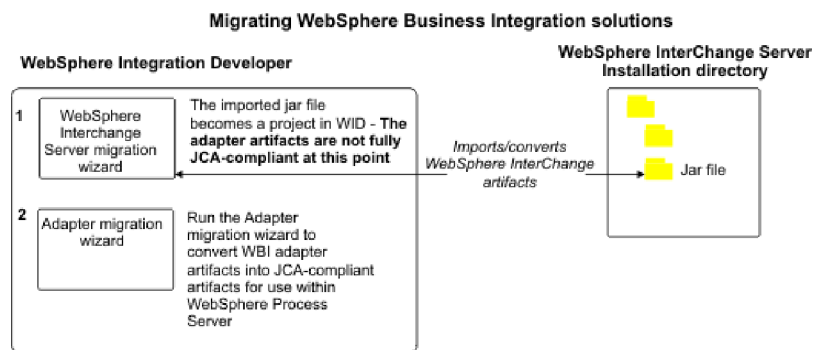
The project can now be used with WebSphere Adapter for Flat Files, version 6.2.

---

## Migrating WebSphere business integration applications for use with Version 6.2 WebSphere Adapters

Migrating WebSphere business integration applications for use with Version 6.2 of your WebSphere adapter is a multistep process. The first step migrates and converts the artifacts from the WebSphere InterChange Server and creates a project for the artifacts in WebSphere Integration Developer. The remaining steps migrate adapter-specific artifacts and complete the conversion of the artifacts into the JCA-compliant format supported by Version 6.2 the adapter.

The following diagram shows the wizards that you use to migrate WebSphere business integration solutions from WebSphere InterChange Server, so that these applications can be used with Version 6.2 of your adapter.



## Road map for migrating applications from WebSphere InterChange Server

To use Version 6.2 of WebSphere Adapter for Flat Files with applications from WebSphere InterChange Server, you need to migrate the application artifacts and convert them so that they can be deployed and run on WebSphere Process Server or WebSphere Enterprise Service Bus. Understanding this task at a high level helps you perform the steps that are needed to accomplish the task.

The following figure illustrates the flow of the migration task. The steps that follow the figure describe this task at a high level only. See the topics following this road map for the details on how to perform each of these steps.

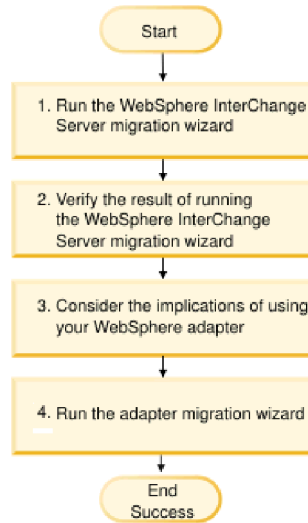


Figure 7. Road map for migrating applications from WebSphere InterChange Server

## Migrating applications from WebSphere InterChange Server

This task consists of the following steps:

1. Run the WebSphere InterChange Server migration wizard.  
The WebSphere InterChange Server migration wizard moves the application artifacts into the WebSphere Integration Developer. The migrated adapter artifacts are not fully JCA-compliant at the completion of this task.
2. Verify that the WebSphere InterChange Server migration is successful.  
Review all messages from the Migration results window and take actions if required.
3. Consider the implications of using Version 6.2 of WebSphere Adapter for Flat Files.  
In addition to considerations for migrating WebSphere InterChange Server applications, you need to consider how Version 6.2 of WebSphere Adapter for Flat Files works with the migrated applications. Some of the adapter operations supported by WebSphere InterChange Server applications might be supported and implemented differently with Version 6.2 of the adapter.
4. Run the adapter migration wizard.  
Run the adapter migration wizard to update adapter-specific artifacts such as the schemas and service definition files (.import, .export, and .wsdl files) for use with Version 6.2 of the adapter.

## Migration considerations for WebSphere business integration adapters

After migration, you will have an adapter that is compliant with the Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) and designed specifically for service-oriented architecture.

### Run the WebSphere Interchange Server migration wizard before running the migration wizard

Before running the adapter migration wizard, use the WebSphere Interchange Server migration wizard to generate the application artifacts for the WebSphere business integration adapter, including the business objects, maps, and

collaborations. Then you can run the adapter migration wizard to update the adapter-specific artifacts such as the schemas and service definition files (.import, .export, and .wsdl) so that they are suitably converted into a format that is compliant with JCA.

### **Run the migration wizard in a test environment first**

Because migrating from a WebSphere business integration adapter to WebSphere Adapter for Flat Files might require changes to those applications that use Version 6.2 of WebSphere Adapter for Flat Files, always perform the migration in a development environment first and test your applications before deploying the application to a production environment.

**Note:** After migrating the WebSphere business integration adapters to become compliant with the J2EE Connector Architecture (JCA), you will not be able to add a new application specific information (ASI) property to the business object in the business object editor. To add a new ASI property after migration, open the business object with a text editor and add the ASI property manually.

## **Migrating application artifacts from WebSphere InterChange Server**

To migrate the application artifacts into WebSphere Integration Developer, run the WebSphere InterChange Server migration wizard. The wizard imports and converts the majority of artifacts into a format that is compatible with WebSphere Process Server or WebSphere Enterprise Service Bus.

### **Before you begin**

Launch the WebSphere InterChange Server migration wizard from within WebSphere Integration Developer to migrate the application artifacts from WebSphere InterChange Server format into artifacts that are compatible with WebSphere Process Server or WebSphere Enterprise Service Bus.

### **About this task**

For information about how to prepare to migrate artifacts from WebSphere InterChange Server and for detailed instructions on performing the migration and verifying that the migration was successful, go to the IBM WebSphere Business Process Management information center and read the topic Migrating to WebSphere Process Server or WebSphere Enterprise Service Bus from WebSphere InterChange Server.

Running WebSphere InterChange Server migration wizard might not fully convert adapter-specific artifacts (such as service descriptors, service definitions, and business objects) into WebSphere Process Server or WebSphere Enterprise Service Bus compatible artifacts. To complete the migration of adapter-specific artifacts, run the adapter migration wizard after you have successfully run the WebSphere InterChange Server migration wizard.

### **Results**

The project and application artifacts are migrated and converted into WebSphere Process Server compatible artifacts.

### **What to do next**

Run the adapter migration wizard to migrate the adapter-specific artifacts.

## Migrating adapter-specific artifacts

You can migrate a project using the adapter migration wizard. The adapter migration wizard updates adapter-specific artifacts such as the schemas and service definition files (.import, .export, and .wsdl) for use with version 6.2 of the adapter. When you finish running the adapter migration wizard, the migration is complete and you can work in the project or deploy the module.

### Before you begin

Before running the adapter migration wizard you should do the following:

- Review the information in the "Migration considerations" topic.
- Run the WebSphere InterChange Server migration wizard to migrate the project and convert data objects for use with WebSphere Process Server or WebSphere Enterprise Service Bus.

### About this task

**Note:** After migration is complete, the module will work only with Version 6.2 of your adapter.

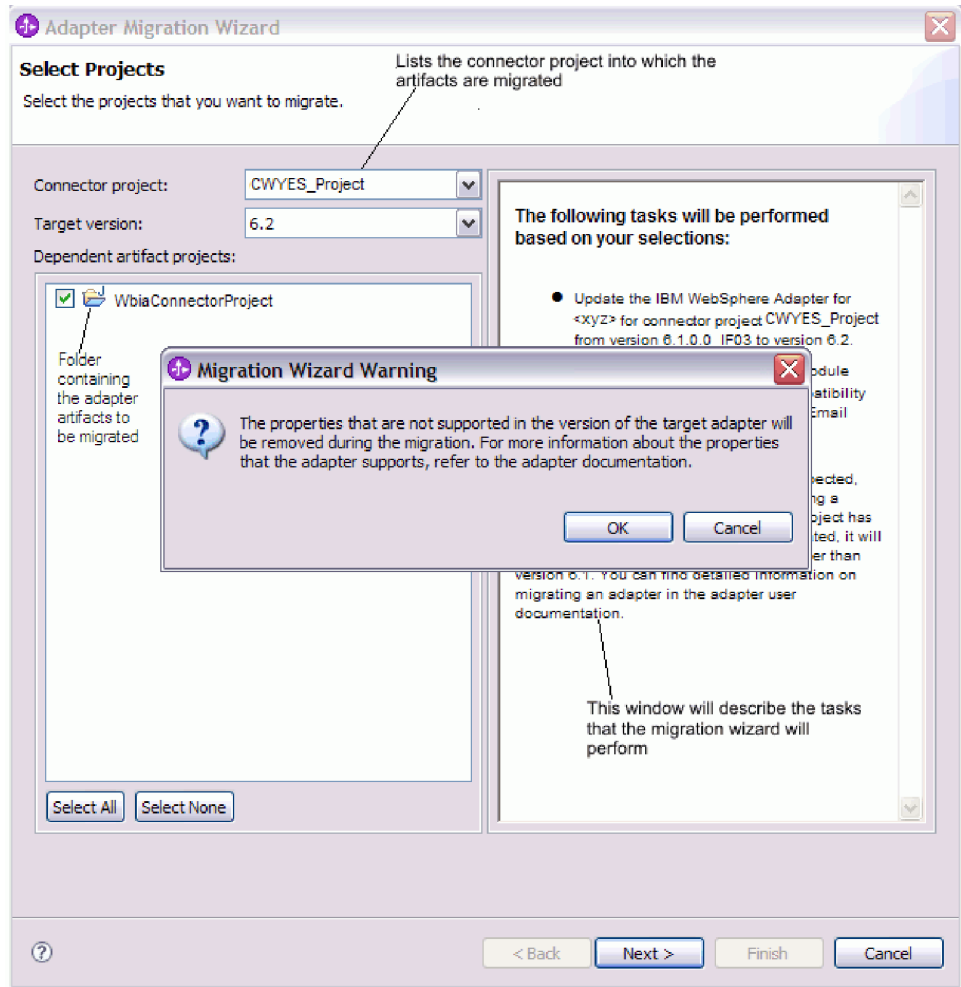
To perform the migration in WebSphere Integration Developer, complete the following steps.

### Procedure

1. In the Java EE perspective, right-click the connector project and select **Update** → **Update Connector Project**.

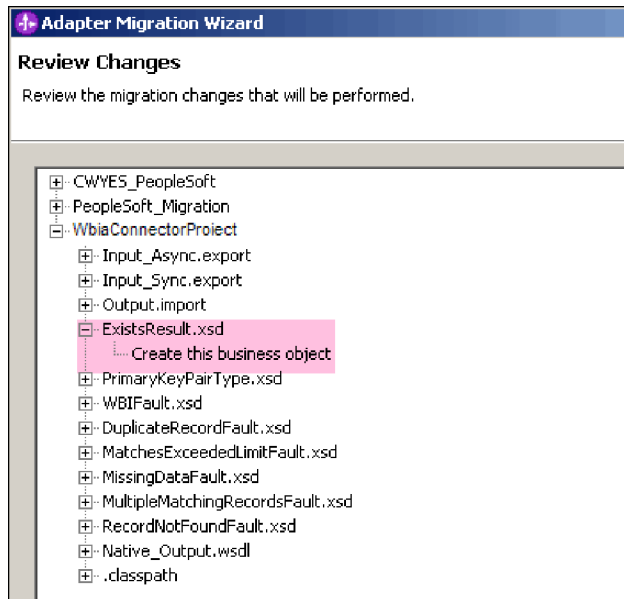
**Note:** Additionally, you can launch the adapter migration wizard by right-clicking the module project in the Java EE perspective and selecting **Update** → **Migrate Adapter Artifacts**.

The following figure describes the functional areas of the wizard.



When you launch the migration wizard from the connector project context menu while in the Java EE perspective, by default all the dependent artifact projects are selected. If you deselect a dependent artifact project, that project is not migrated.

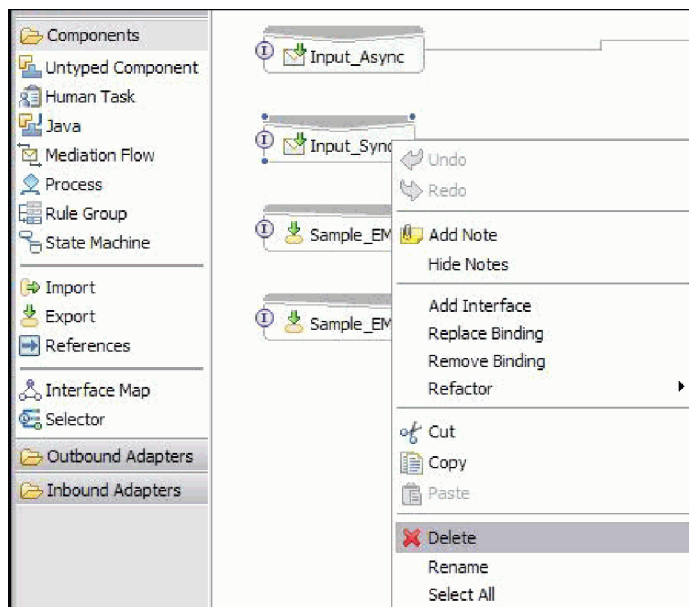
2. On the Review changes window you can review the migration changes that will occur in each of the artifacts that you are migrating by clicking the + sign.



3. Click **Finish** to perform the migration.

Before performing the migration process, the wizard backs up all projects affected by the migration. The projects are backed up to a temporary folder within the workspace. If the migration fails for any reason, or if you decide to cancel the migration before it completes, the wizard deletes the modified projects and replaces them with the projects stored in the temporary folder.

4. Select **Project > Clean**, to refresh and rebuild the workspace for the changes to take effect.
5. On successful migration, all backed up projects are deleted. Remove the Sync inbound flow manually as this flow is not used by the adapter. From the migrated project, select the Input\_Sync inbound flow, right-click and select **Delete**.



6. If you are migrating an EAR file, create a new EAR file with the migrated adapter and artifacts, and deploy it to WebSphere Process Server or WebSphere

Enterprise Service Bus. For information about exporting and deploying an EAR file, see “Deploying the module for production” on page 119.

## Results

The project is migrated to Version 6.2. You do not need to run the external service wizard after exiting the adapter migration wizard.

## Import, export, and WSDL file changes after migration

The WebSphere InterChange Server migration wizard moves the application artifacts into the WebSphere Integration Developer. The migrated adapter artifacts are not fully JCA-compliant at the completion of this task. You can complete the migration of the adapter-specific artifacts (such as service descriptors, service definitions, and business objects) to a JCA compatible format by running the adapter migration wizard. The changes by the migration wizard are reflected in the service definition files (.import, .export and, .wsdl).

### Changes to the import file after migration

During migration, the affected module artifacts are migrated to an import file. The existing JMS Binding property before the migration changes to EIS Binding property in the import file. The other property details information added in the import file include information about data binding configuration, changes to connection information in Managed Connection Factory Properties, and several new method bindings.

### Changes to the export file after migration

During migration, the affected module artifacts are migrated to an export file. The existing JMS Binding property before the migration changes to EIS Binding property in the export file. The other property details information added in the export file include information about data binding configuration, changes to connection information in Managed Connection Factory Properties, and several new method bindings.

### Changes to the WSDL file after migration

During migration, the affected module artifacts are migrated to corresponding WSDL files that include Flat Files specific service description WSDL artifacts. The service description files become JCA compatible. The WSDL files will have input and output type for each operation. Both the inbound and outbound operations work on their specific input types to produce corresponding output types after the operations execution. The outbound operations generated during migration are shown in the following table:

Operations	Input	Output
Create	CustomerWrapperBG	CreateResponse
Append	CustomerWrapperBG	AppendResponse
Overwrite	CustomerWrapperBG	OverwriteReponse

#### Note:

- When you migrate multiple inbound top level business objects in the project, only the first top level business object inbound feature works correctly. For the other top level business object inbound to work correctly, you must manually



modify the "emit + [verb name] + afterimage + [business object name]" method in the `Input_Processing.java` and `Input_Async_Processing.java` class to call the correct destination services.

- During inbound processing, a dynamic method called emit is generated. A business object name is appended to emit to create an actual method. The function selector determines the business object name to append to the emit method. For example, if the business object name is Customer, the actual method name will be emitCustomer. Emit is the only operation available during inbound processing.
- The WebSphere business integration adapter properties (such as *config*) that are not valid or not supported by WebSphere Adapter for Flat Files are removed from the migrated artifacts.



---

## Chapter 3. Samples and tutorials

To help you use WebSphere Adapters, samples and tutorials are available from the Business Process Management Samples and Tutorials Web site.

You can access the samples and tutorials in either of the following ways:

- From the welcome page that opens when you start WebSphere Integration Developer. To see samples and tutorials for WebSphere Adapter for Flat Files, click **Retrieve**. Then browse the displayed categories to make your selections.
- At this location on the Web: Business Process Management Samples and Tutorials <http://publib.boulder.ibm.com/bpcsamp/index.html>.



---

## Chapter 4. Configuring the module for deployment

To configure the adapter so that it can be deployed on WebSphere Process Server or WebSphere Enterprise Service Bus, use WebSphere Integration Developer to create a module, which is exported as an EAR file when you deploy the adapter. You then specify the business objects you want to build and the system on which you want to build them.

---

### Road map for configuring the module

Before you can use WebSphere Adapter for Flat Files in a runtime environment, you must configure the module. Understanding this task at a high level helps you perform the steps that are needed to accomplish the task.

You configure the module for WebSphere Adapter for Flat Files by using WebSphere Integration Developer. The following figure illustrates the flow of the configuration task, and the steps that follow the figure describe this task at a high level only. For the details about how to perform each of these steps, see the topics following this road map.

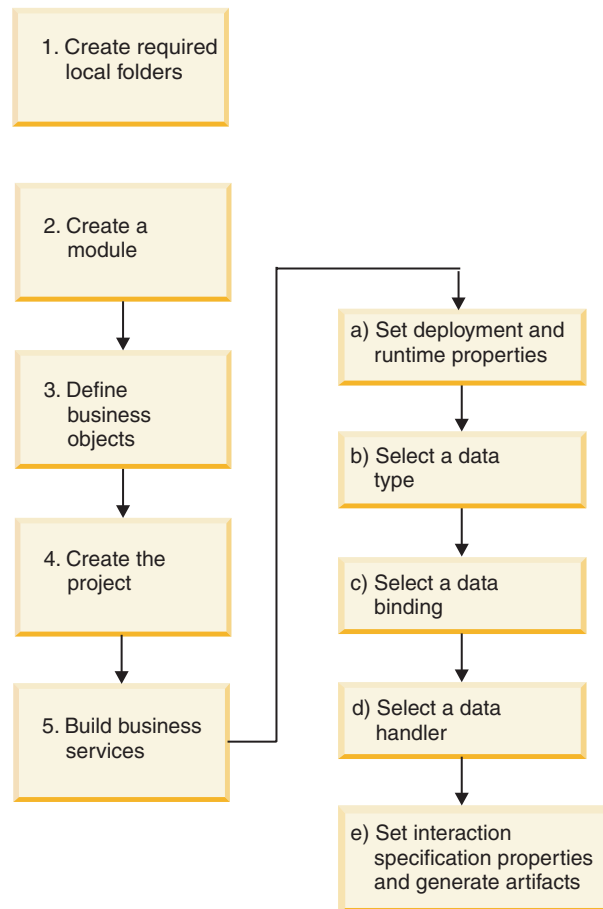


Figure 8. Road map for configuring the module

#### Configuring the module

This task consists of the following steps, which are described at a high level.

**Note:** These steps assume that you are using user-defined business objects that require data transformation. If using generic business objects, which do not require data transformation, some of the following steps will be ignored. For example, you will not need to select a data binding and a data handler.

1. Create a module in WebSphere Integration Developer. You create business objects in the module.
2. Define the business objects that will be used by the project.
3. Create a project, which is used to organize the files associated with the adapter using the external service wizard in WebSphere Integration Developer.
4. Build business services by running the external service wizard from WebSphere Integration Developer, then performing the following steps:
  - a. Specify the following deployment and runtime properties:
    - Connection properties
    - Security properties
    - Deployment options
    - Function selector - Inbound only
  - b. Select a data type and name the operation associated with this data type. For each operation, specify the following:
    - The operation kind. For example, Create, Append, Exists.
    - Specify if the operation is passthrough or user defined.
  - c. Select the data binding. Each data type has an equivalent data binding used to read the fields in a business object and fill the corresponding fields in a file.
  - d. Select the data handler that will perform the conversions between a business object and a native format.
  - e. Specify interaction specification property values and generate artifacts. The output from running the external service wizard is saved to a business integration module, which contains the business object or objects, and the import or export file.

---

## Creating the required local folders

Before you create inbound or outbound modules, you must create folders on the local file system for events and output. You can optionally create folders for staging and archiving.

Before you create inbound or outbound modules, you must specify the event directory and the output directory on the Service Configuration Properties screen of the external service wizard. You can also create a staging directory and an archive directory, but these are not required.

- The event directory stores events for inbound processing. The adapter polls this folder at regular intervals and sends any found events, in the form of business objects, to the server.
- The output directory is used by the adapter to write the final output files for Create, Append, and Overwrite operations during outbound processing.
- The staging directory is a temporary directory where the adapter writes the initial output files during Create and Overwrite operations, to avoid write conflicts. The output files are then renamed and copied to the output directory.

- The archive directory is a directory where the adapter stores processed event files.

Instead of specifying the names of these directories when you run the external service wizard, you can use WebSphere Application Server environment variables.

**Related tasks**

“Defining WebSphere Application Server environment variables” on page 51  
Use the administrative console of WebSphere Process Server or WebSphere Enterprise Service Bus to define WebSphere Application Server environment variables.

**Related reference**

“Managed connection factory properties” on page 159

Managed connection factory properties specify information the adapter needs at run time for outbound communication with the local file system.

“Activation specification properties” on page 176

Activation specification properties hold the inbound event processing configuration information for an export. You set activation specification properties through either the external service wizard or the administrative console.

## Creating the module

You create the module in WebSphere Integration Developer. The module allows you to define business objects that will be used by the project.

**About this task**

Start the external service wizard and follow this procedure to create a new module.

**Procedure**

1. If WebSphere Integration Developer is not currently running, start it now.
  - a. Click **Start** → **Programs** → **IBM WebSphere** → **Integration Developer V6.1.0** → **WebSphere Integration Developer V6.1.0**.
  - b. If you are prompted to specify a workspace, either accept the default value or select another workspace.  
The workspace is a directory where WebSphere Integration Developer stores your project.
  - c. Optional: When the WebSphere Integration Developer window is displayed, click **Go to the Business Integration perspective**.
2. Right-click inside the Business Integration section of the WebSphere Integration Developer window. Click **New** → **Module**.

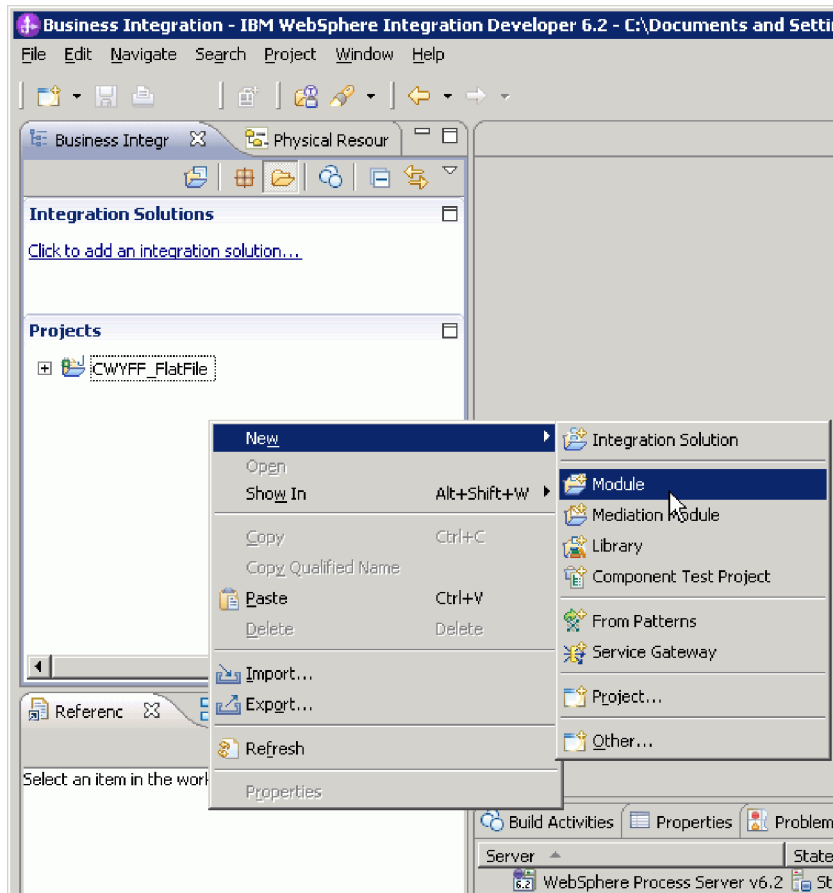


Figure 9. Business Integration section of the window

3. Type a new **Module Name** in the New Module window. Leave the other options (**Use default location** and **Open module assembly diagram**) checked.



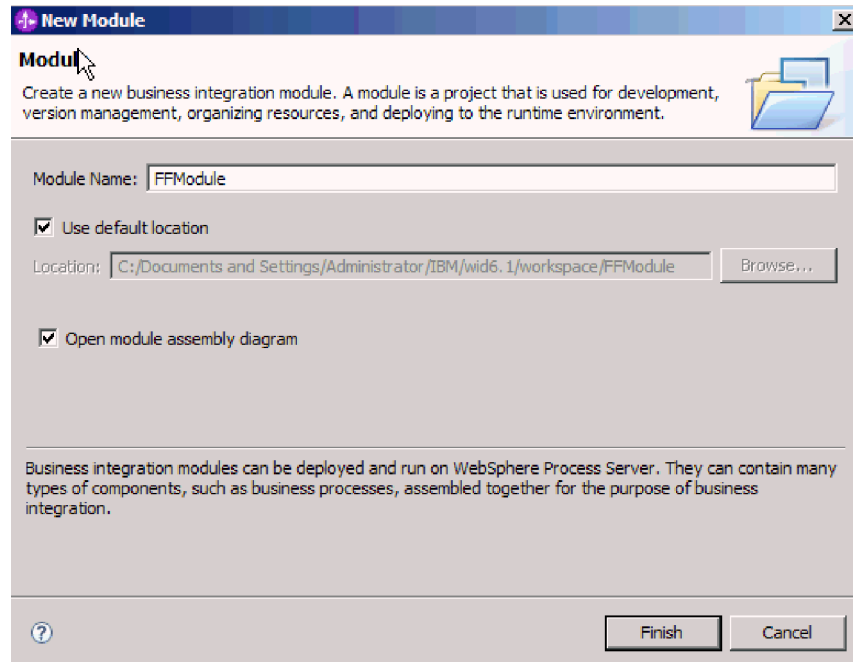


Figure 10. New Module window

4. Click **Finish**.

### Results

A new module is listed in the Business Integration window.

### What to do next

Create a project, which is used to organize the files associated with the adapter.

---

## Defining WebSphere Application Server environment variables

Use the administrative console of WebSphere Process Server or WebSphere Enterprise Service Bus to define WebSphere Application Server environment variables.

### Before you begin

### About this task

To define a WebSphere Application Server environment variable, use the following procedure.

### Procedure

1. Start the administrative console.
2. Select **Environment** → **WebSphere Variables**.
3. Select the scope for the environment variable. The scope specifies the level at which the resource definition is visible on the administrative console panel. The possible values are server, node, and cell. In this example, we're choosing Cell=widCell.

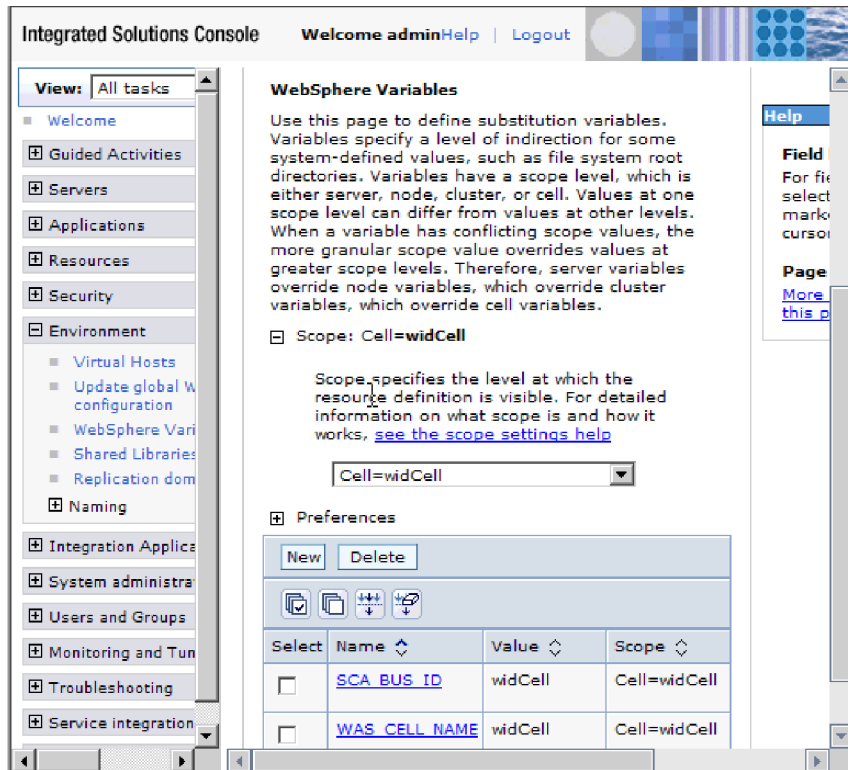


Figure 11. Setting the scope for the environment variable

4. Click **New** and provide a name and a value for the environment variable. The name is the symbolic name that represents a physical path. The value is the absolute path that the variable represents. In this example, the name is `EVENT_DIRECTORY` and the value is `C:/flatfile/event`. You can use the **Description** field, which is optional, to describe the purpose of the variable.

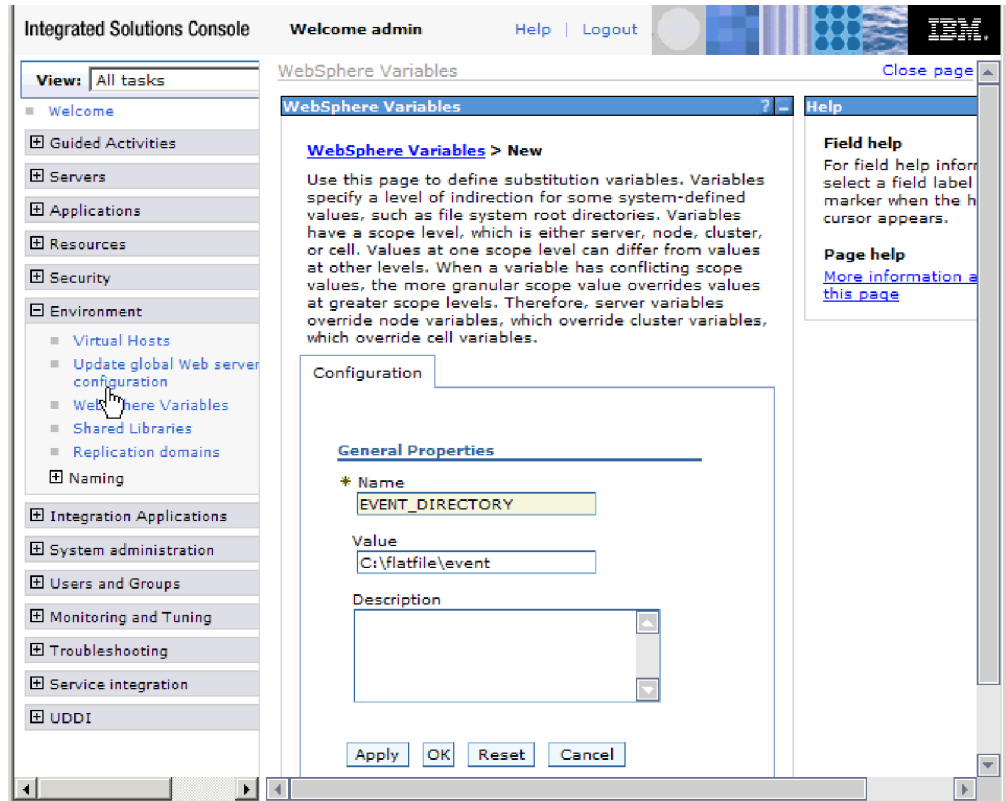


Figure 12. Providing a name and a value for the environment variable

5. Click **OK** and save the changes.

## Results

An environment variable called `EVENT_DIRECTORY` is defined, with the value `C:\flatfile\event` and a scope of `Cell=widCell`. You can use it in the external service wizard whenever you need to specify the event directory.



Figure 13. The new environment variable `EVENT_DIRECTORY` displayed in the WebSphere Variables window

## What to do next

Create a project, which is used to organize the files associated with the adapter.

### Related concepts

“WebSphere Application Server environment variables” on page 24  
 WebSphere Application Server environment variables can be used in the external service wizard to specify directory values.

“Creating the required local folders” on page 48

Before you create inbound or outbound modules, you must create folders on the local file system for events and output. You can optionally create folders for staging and archiving.

### Related reference

“Managed connection factory properties” on page 159

Managed connection factory properties specify information the adapter needs at run time for outbound communication with the local file system.

“Activation specification properties” on page 176

Activation specification properties hold the inbound event processing configuration information for an export. You set activation specification properties through either the external service wizard or the administrative console.

## Defining business objects

Predefine the business objects in WebSphere Integration Developer that will be used by the project that you will create in the next topic.

### About this task

To predefine new business objects using the business object editor, complete the following steps.

### Procedure

1. Expand the new module located inside of the Business Integration section of the WebSphere Integration Developer window.
2. Right-click the **Data Types** folder and select **New > Business Object**.

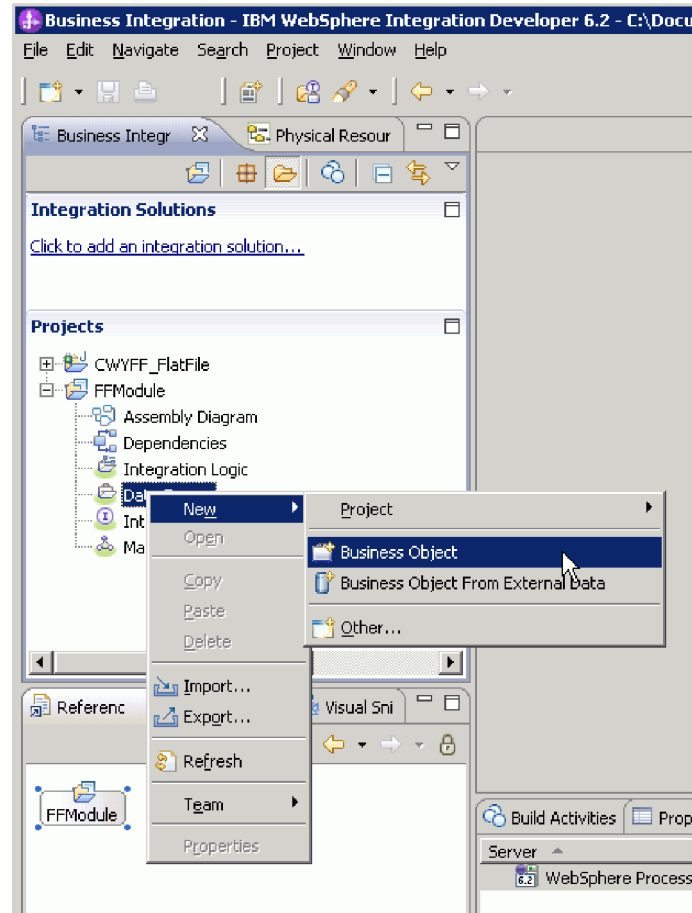


Figure 14. New Business Object selection view

3. Type in a new **Name** in the Business Object window.

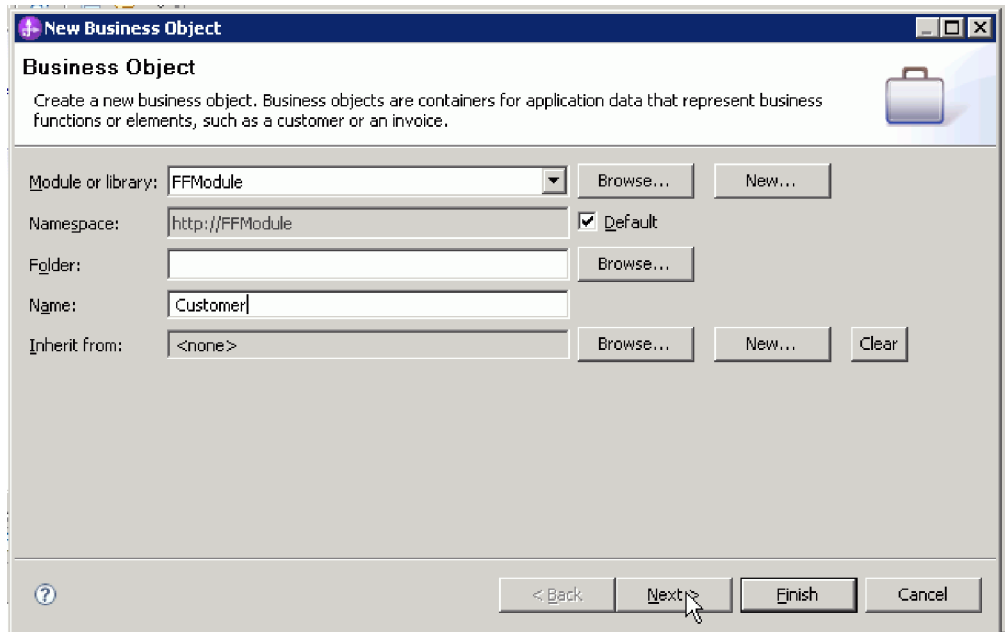


Figure 15. Business Object window

4. Click **Finish**. The new business object is added to the **Data Types** folder.
5. Click the **Add a field to a business object** icon and add the necessary fields to the business object.

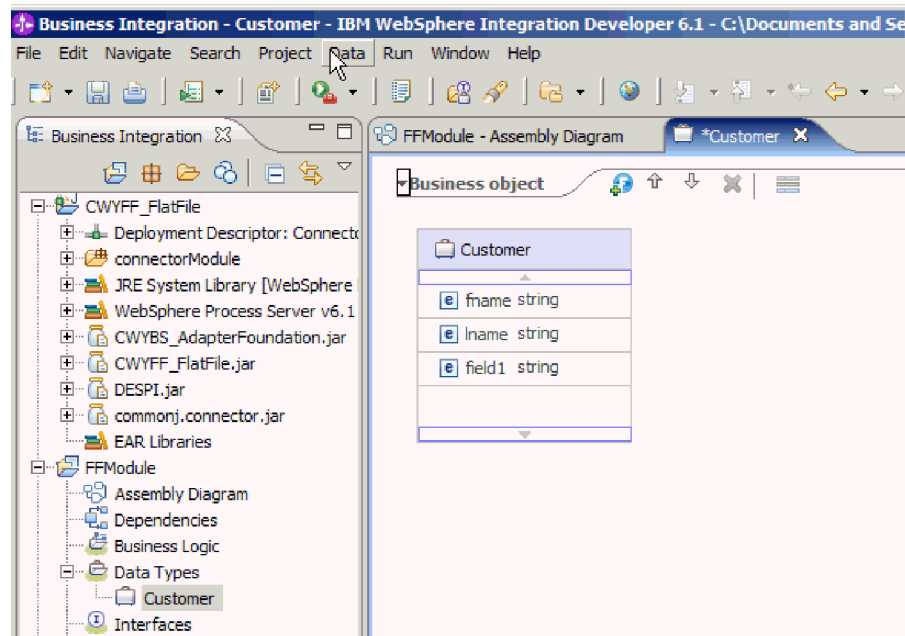


Figure 16. Add Business object fields icon

6. Click the Save icon.
7. Repeat the above steps for each business object that you want to create.

## Results

The new business objects are defined.

### What to do next

Create a project, which is used to organize the files associated with the adapter.

#### Related concepts

“Business objects” on page 22

A business object is a logical data container that represents the data that is processed by the adapter. The data can represent either a business entity, such as an invoice or an employee record, or unstructured text, such as the body of an e-mail or a word processing document. The adapter uses business objects to send data to or obtain data from the local file system.

#### Related reference

“Business object information” on page 149

You can determine the purpose of a business object by examining both the application-specific information within the business object definition file and the name of the business object. The application-specific information dictates what operations can be performed on the local file system. The name typically reflects the operation to be performed and the structure of the business object.

---

## Converting business objects to COBOL copybook files during outbound processing

Use the external data wizard in WebSphere Integration Developer to generate business object definitions from a COBOL program source file. These business object definitions are used during outbound processing.

### Before you begin

Before you perform this task, make sure that:

1. You have created a module in WebSphere Integration Developer.
2. The COBOL program source file (.ccp file) is in a local directory on your workstation.
3. If you are going to generate a wrapper business object definition, you have imported the adapter RAR file into your workspace.

### About this task

Use the external data wizard to generate a business object definition for a COBOL program source file. After you have generated the business object definition, you can optionally rerun the external data wizard to generate a wrapper business object definition from the generated business object.

### Procedure

1. Generate the business object definition for the COBOL program source file.
  - a. In the Business Integration section of the window, right-click the module and select **New** → **Business Object From External Data**.
  - b. In the Business Object window, select **Create business objects**.

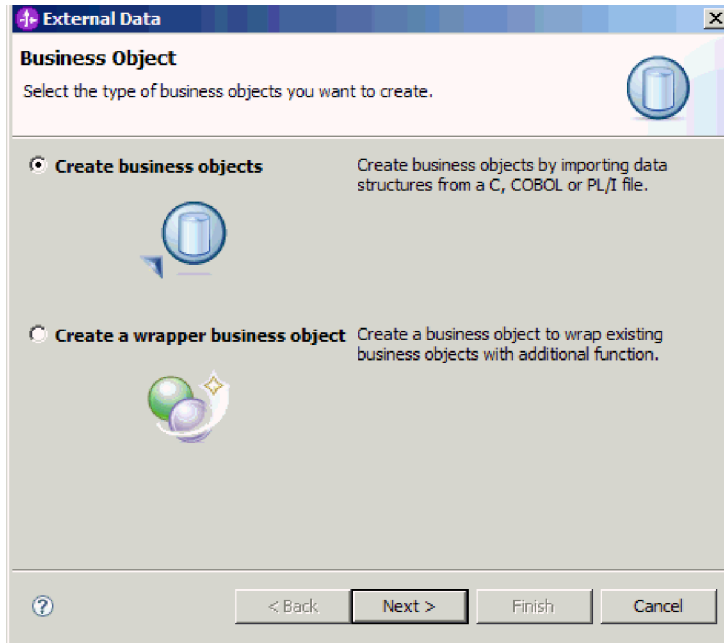


Figure 17. The Business Object window

- c. Click **Next**.
- d. In the Business Object Mapping Details window, make sure the **Choose mapping** value is **COBOL to Business Object**. Click **Browse** and select the .ccp file (in this example, taderc99.ccp).

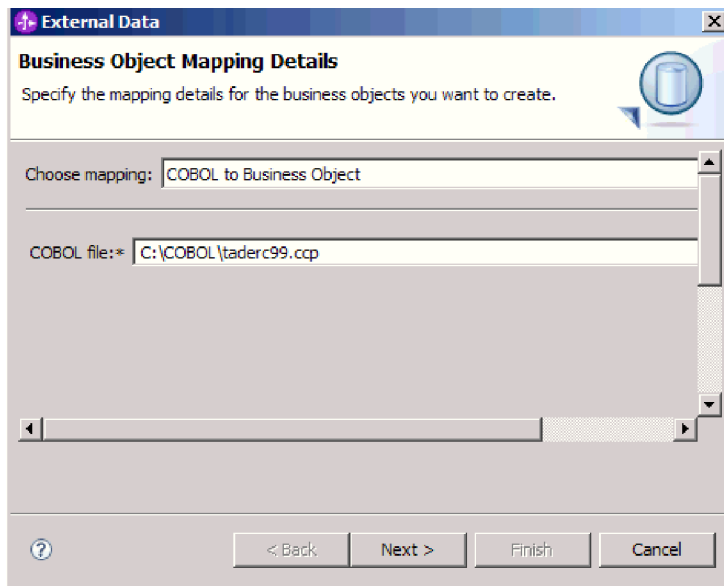


Figure 18. The Business Object Mapping Details window

- e. Click **Next**.
- f. In the Select Data Structures window, click **Find**. The new business object, called DFHCOMMAREA in the figure, is displayed.



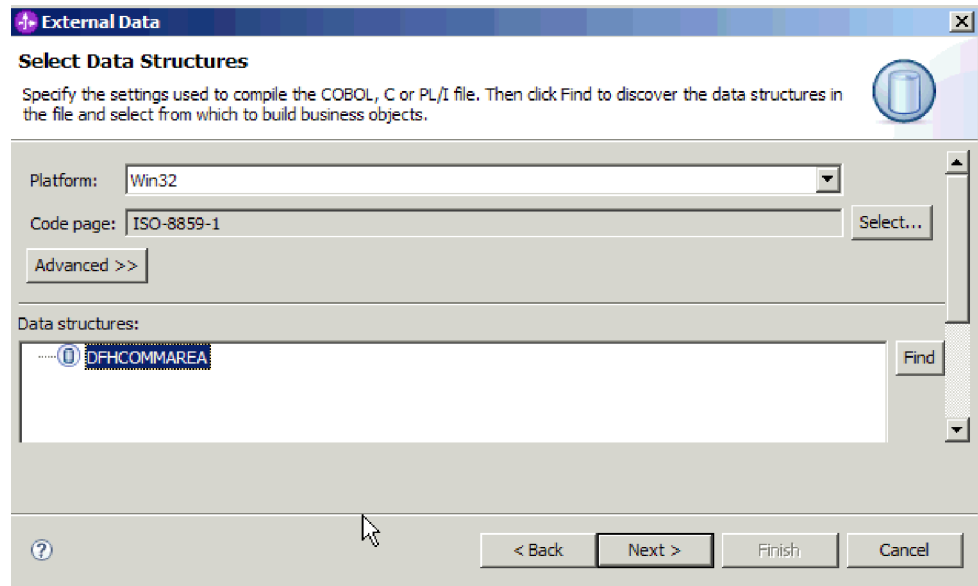


Figure 19. The Select Data Structures window

- g. Select the business object DFHCOMMAREA and click **Next**.
- h. Click **Finish**.

A business object, called DFHCOMMAREA in the figure, is created in the module.

2. Optional: Generate a wrapper business object definition. Wrapper business object definitions wrap existing business object definitions with additional function. The option to generate wrapper business object definitions is displayed only when the adapter RAR file has been imported into the workspace.
  - a. In the Business Integration section of the window, right-click on the module and select **New** → **External Data**.
  - b. In the Business Object window, select **Create a wrapper business object**.

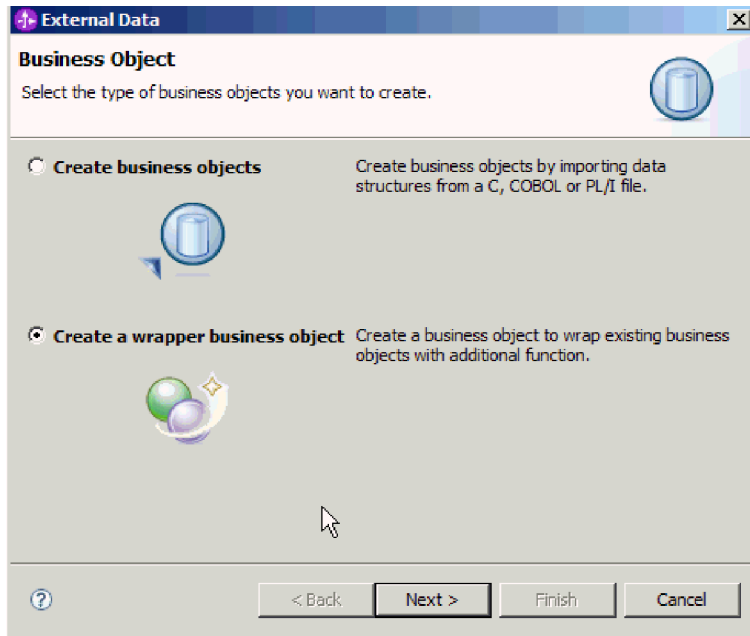


Figure 20. The Business Object window

- c. Click **Next**.
- d. In the Select an Adapter window, select the connector project for which you want to generate the wrapper business object and click **Next**.
- e. In the Business Object Properties window, click **Browse** and select the business object created in Step 1, for example, DFHCOMMAREA, for the data type.
- f. To generate a business graph, select the **Generate business graph for each business object** check box. To generate a retrieve wrapper, select the **Generate retrieve multiple business objects** check box.

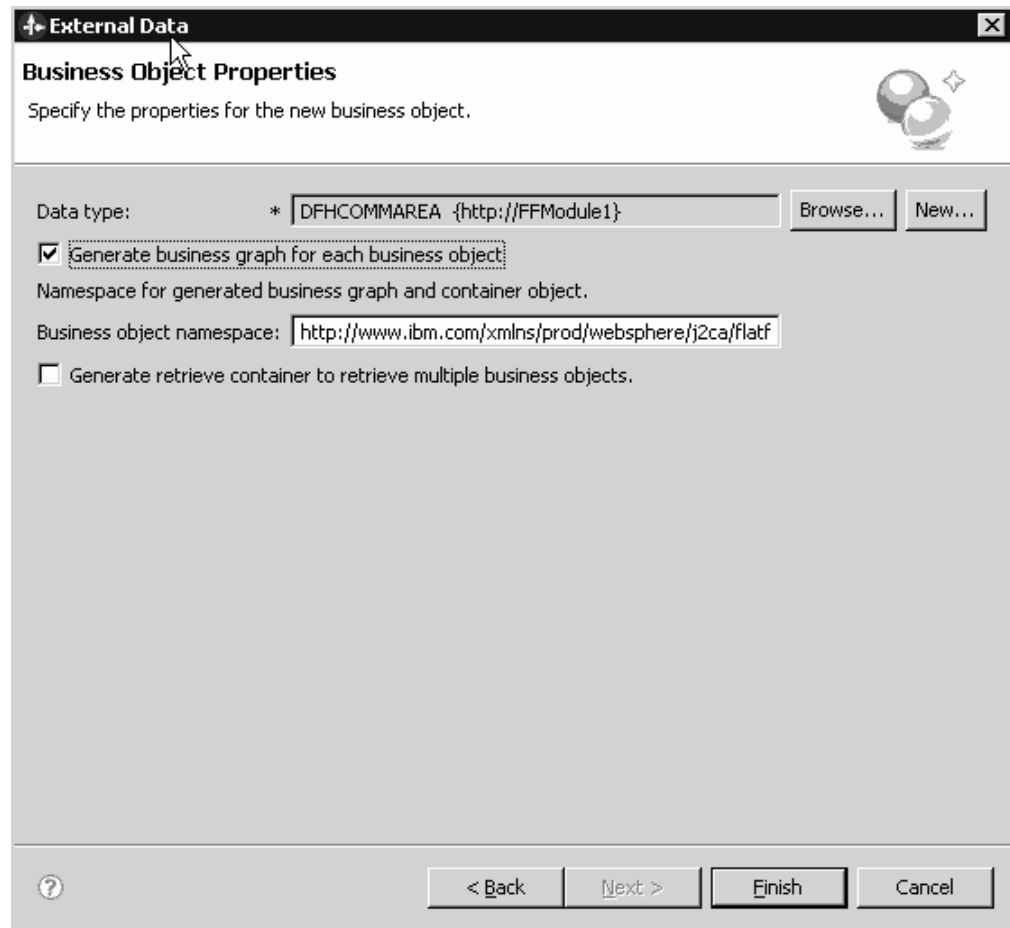


Figure 21. The Business Object Properties window

g. Click **Finish**.

A wrapper business object and a business graph, called DFHCOMMAREAWrapper and DFHCOMMAREAWrapperBG, respectively, in the figure, are listed for the current module in the Business Integration window. If wrappers for retrieve are selected, then business object called DFHCOMMAREARetrieveWrapper and a business graph called DFHCOMMAREARetrieveWrapperBG are also listed for the current module in the Business Integration window.

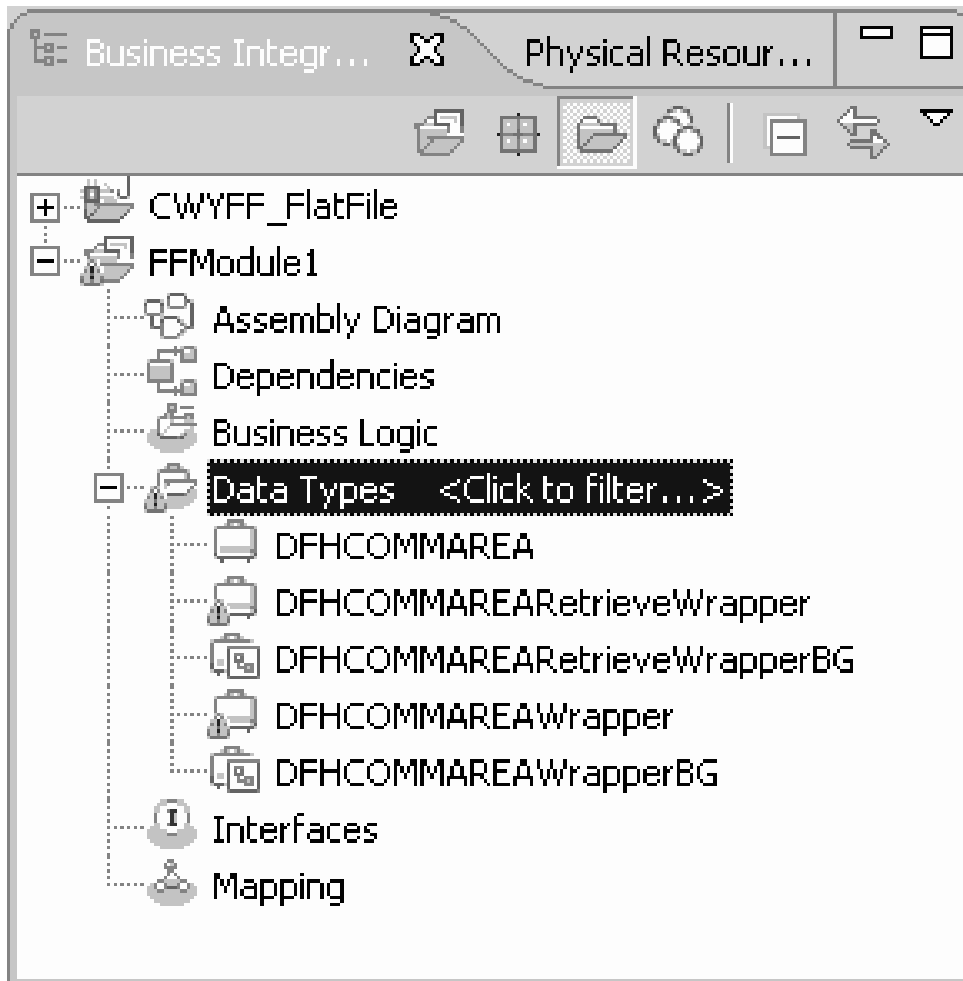


Figure 22. The wrapper business object and the business graph listed in the Business Integration window

3. Generate the required artifacts for the COBOL copybook outbound module. This example shows the configuration for a Create operation.
  - a. In the Business Integration section of the window, right-click on the module and select **New** → **External Service**.
  - b. Select **Adapters** and click **Next**.
  - c. In the Select an Adapter window, select the Flat Files connector project and click **Next**.

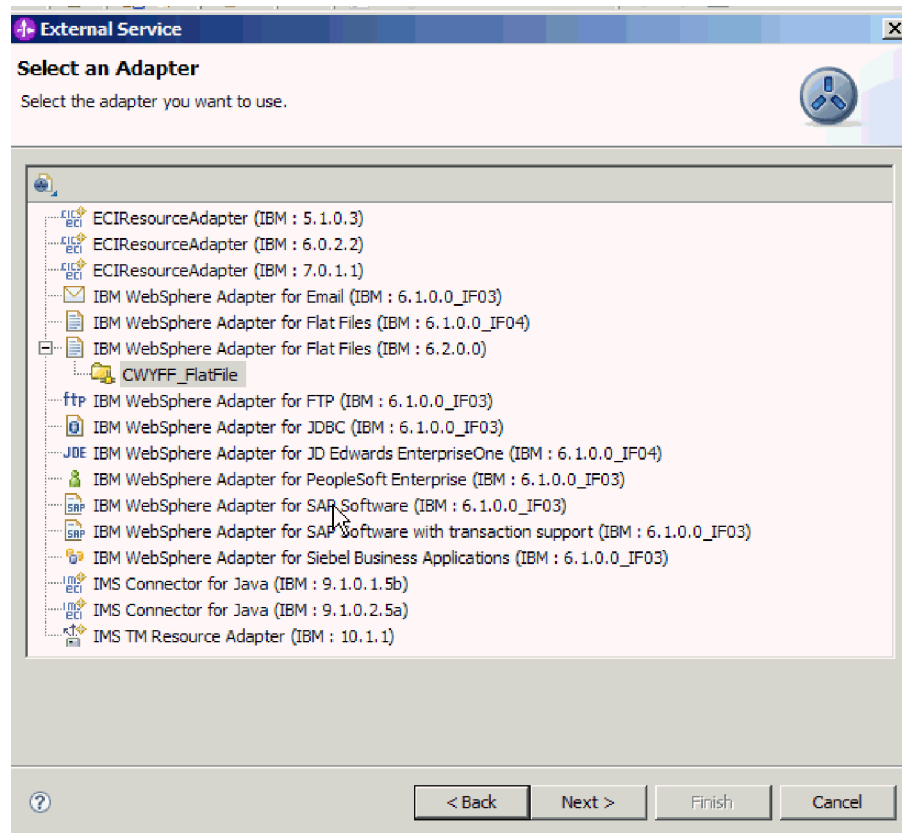


Figure 23. The Select an Adapter window

- d. In the Processing Direction window, select **Outbound**.
- e. Click **Next**.
- f. In the Service Configuration Properties window, in the **Data binding** list, select **Use COBOL, C or PL/I data binding option**.

**Note:** This is not a data binding, but a data binding generator. The tool generates the appropriate data binding code for you in the current module.

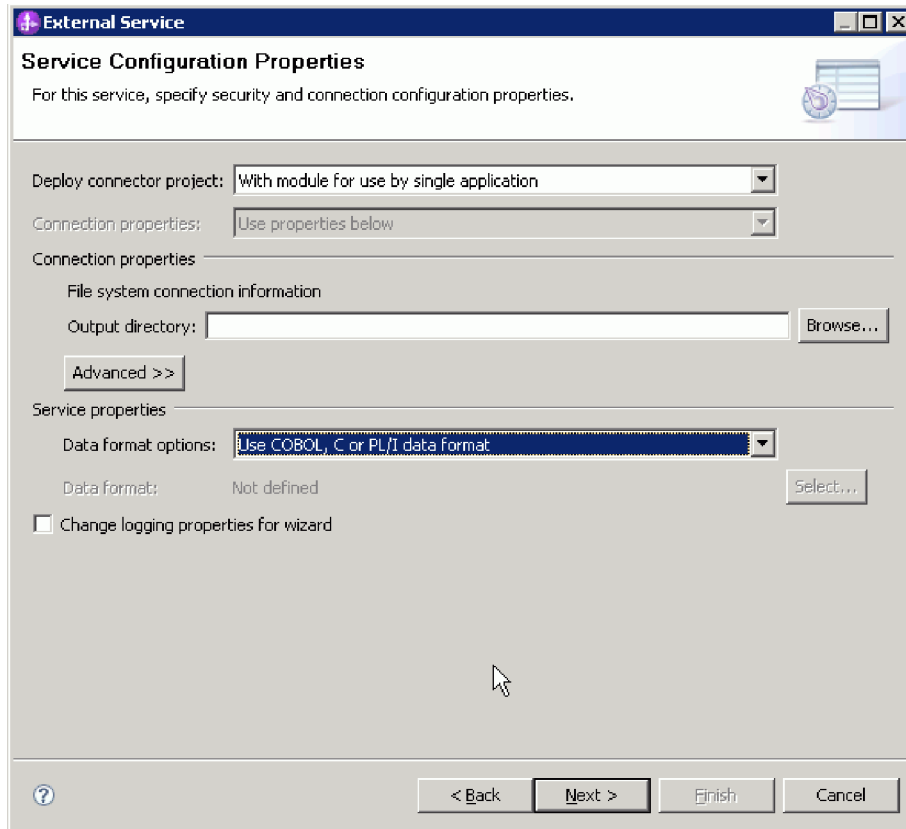


Figure 24. The Service Configuration Properties window

- g. Specify the other required properties for the outbound operation, and click **Next**.
- h. In the Operations window, click **Add** and then **Create**. For retrieve operation select **Retrieve**. Select **User defined type** from the **Data type** list, and click **Next**.
- i. Browse for the input type (DFHCOMMAREA, DFHCOMMAREAWrapper, or DFHCOMMAREAWrapperBG) and click **OK**. For **Retrieve** operation, browse for the appropriate input type (DFHCOMMAREA, DFHCOMMAREARetrieveWrapper, or DFHCOMMAREARetrieveWrapperBG)



Figure 25. The Data Type Selection window

- j. Click **Next** and complete the external service wizard.

The data bindings used by COBOL copybook, WSDL files, import files, and other artifacts are generated. See the Project Explorer window for the generated data binding classes.

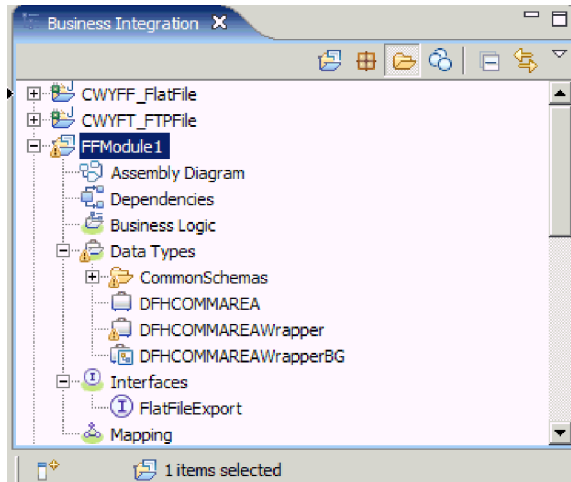


Figure 26. Data bindings used by COBOL copybook, WSDL files, import files, and other artifacts

## Results

A business object, a wrapper business object, and a business graph are created for the COBOL program source file for the outbound module. Artifacts are generated for an outbound Create operation that uses COBOL copybook data binding. This module can be deployed on WebSphere Process Server or WebSphere Enterprise Service Bus and tested for a Create operation.

**Note:** To generate artifacts for other supported operations (Append and Overwrite), follow the same steps, beginning with Step 3h.

## What to do next

Deploy the module.

---

## Converting COBOL copybook files to business objects during inbound processing

Use the external data wizard in WebSphere Integration Developer to generate business object definitions from a COBOL program source file. These business object definitions are used during inbound processing.

### Before you begin

Before you perform this task, make sure that:

1. You have created a module in WebSphere Integration Developer.
2. The COBOL program source file (.ccp file) is in a local directory on your workstation.
3. You have created a local event directory.
4. If you are going to generate a wrapper business object definition, you have imported the adapter RAR file into your workspace.

### About this task



Use the external data wizard to generate a business object definition for a COBOL program source file. After you have generated the business object definition, you can optionally rerun the external data wizard to generate a wrapper business object definition from the generated business object.

### Procedure

1. Generate the business object definition for the COBOL program source file.
  - a. In the Business Integration section of the window, right-click the module and select **New** → **External Data**.
  - b. In the Business Object window, select **Create business objects**.

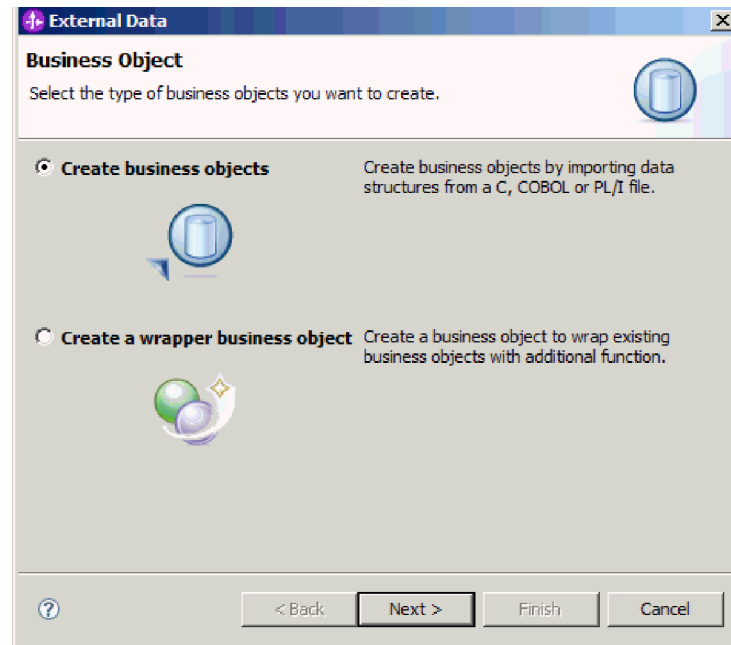


Figure 27. The Business Object window

- c. Click **Next**.
    - d. In the Business Object Mapping Details window, make sure the **Choose mapping** value is **COBOL to Business Object**. Click **Browse** and select the .ccp file (in this example, taderc99.ccp).

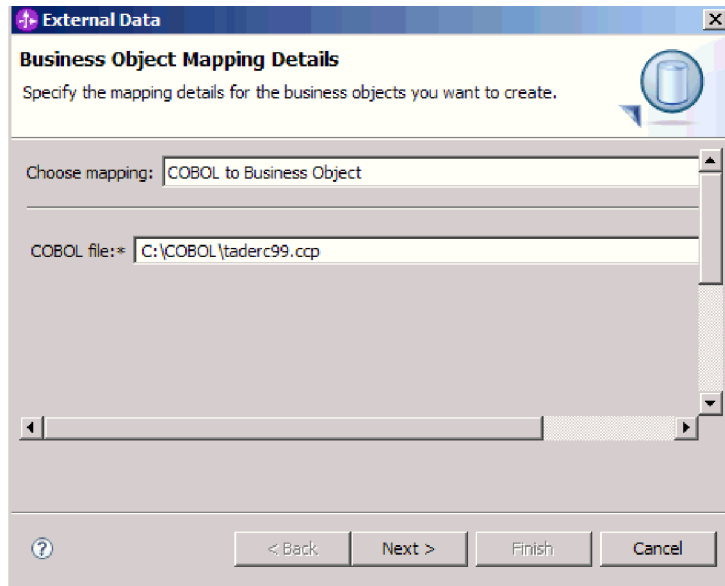


Figure 28. The Business Object Mapping Details window

- e. Click **Next**.
- f. In the Select Data Structures window, click **Find**. The new business object, called DFHCOMMAREA in the figure, is displayed.

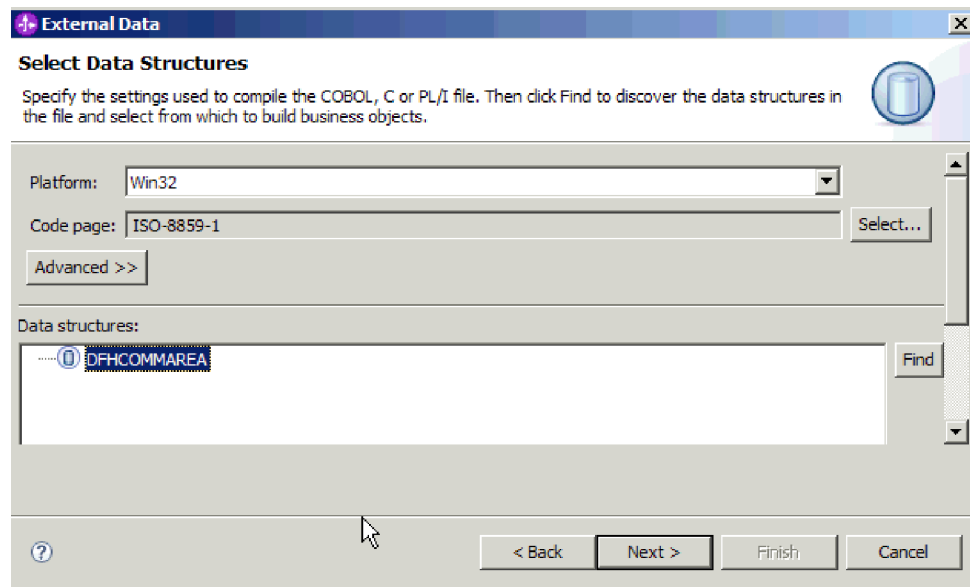


Figure 29. The Select Data Structures window

- g. Select DFHCOMMAREA and click **Next**.
- h. Click **Finish**.

A business object, called DFHCOMMAREA in the figure, is created in the module.

2. Optional: Generate a wrapper business object definition. Wrapper business object definitions wrap existing business object definitions with additional

function. The option to generate wrapper business object definitions is displayed only when the adapter RAR file has been imported into the workspace.

- a. In the Business Integration section of the window, right-click the module and select **New** → **External Data**.
- b. In the Business Object window, select **Create a wrapper business object**.

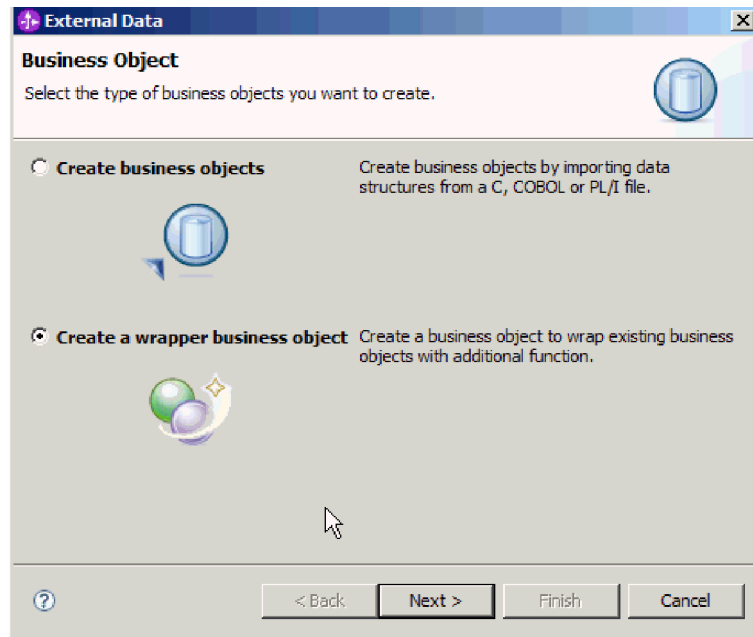


Figure 30. The Business Object window

- c. Click **Next**.
- d. In the Select an Adapter window, select the connector project in which the new business object was saved and click **Next**.
- e. In the Business Object Properties window, click **Browse** and select the business object created in Step 1, for example, DFHCOMMAREA, for the data type.
- f. To generate a business graph, select the **Generate business graph for each business object** checkbox.

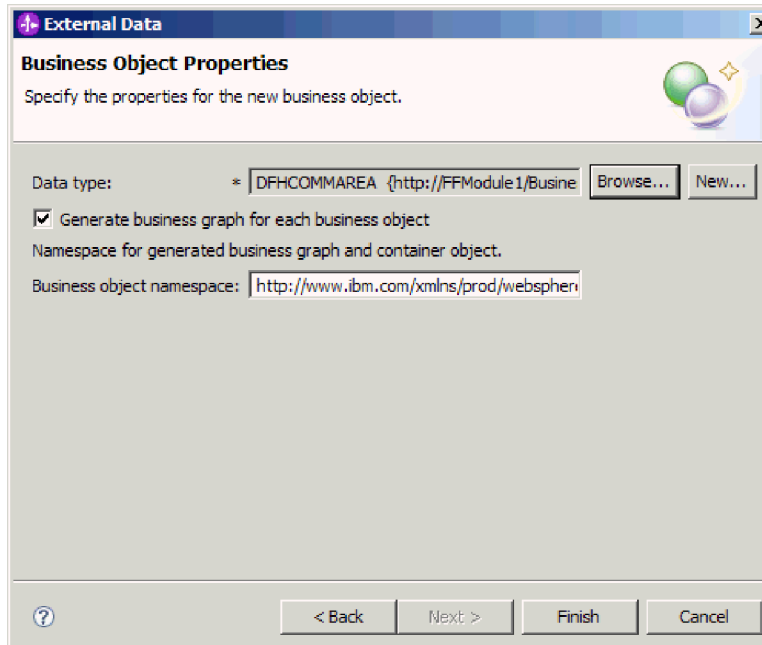


Figure 31. The Business Object Properties window

g. Click **Finish**.

A wrapper business object and a business graph, called DFHCOMMAREAWrapper and DFHCOMMAREAWrapperBG, respectively, in the figure, are listed for the current module in the Business Integration window.

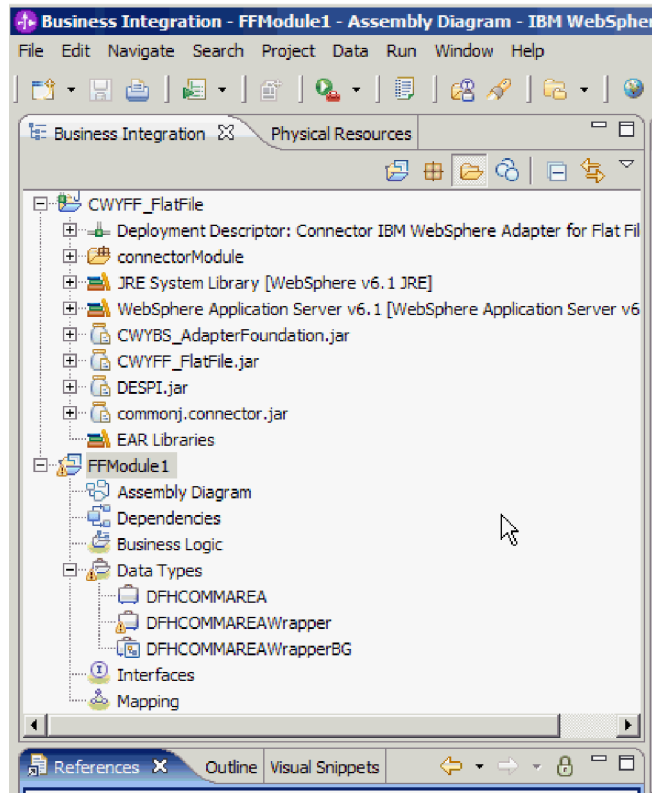


Figure 32. The wrapper business object and the business graph listed in the Business Integration window

3. Generate the required artifacts for the COBOL copybook inbound module.
  - a. In the Business Integration section of the window, right-click on the module and select **New** → **External Service**.
  - b. Select **Adapters** and click **Next**.
  - c. In the Select an Adapter window, select the connector project in which the business object that you just created was saved and click **Next**.

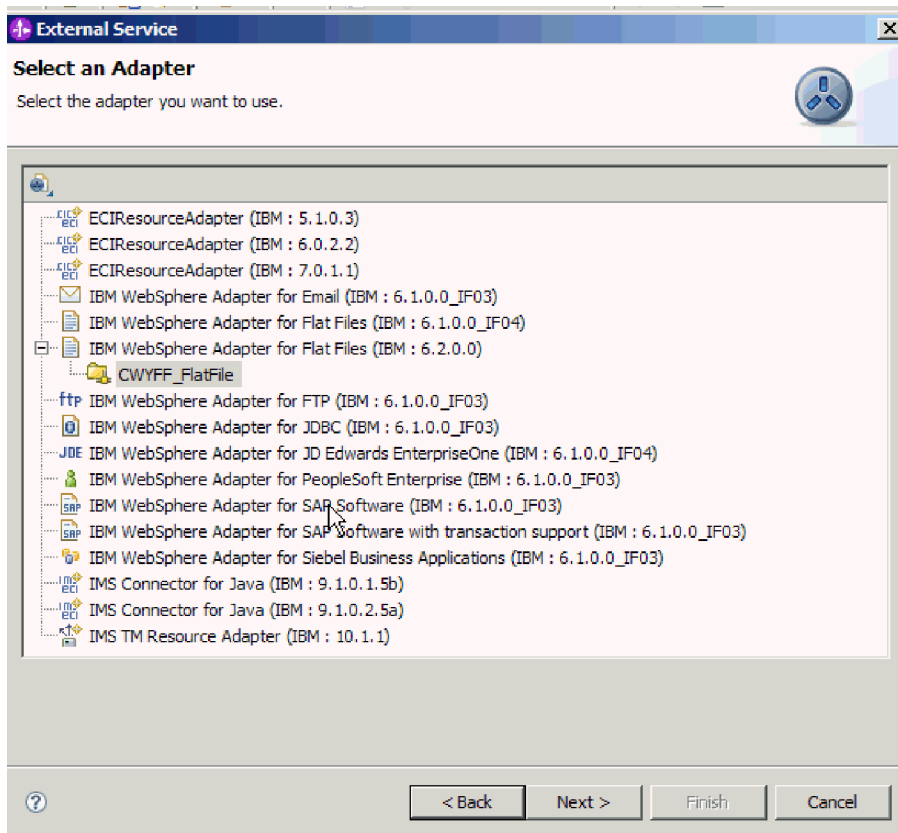


Figure 33. The Select an Adapter window

- d. In the Processing Direction window, select **Inbound** and click **Next**.
- e. Click **Browse** and select the event directory.
- f. For the **Function selector**, choose the default value.
- g. In the **Data binding** list select **Use COBOL, C or PL/I data binding** option.

**Note:** This is not a data binding, but a data binding generator. The tool generates the appropriate data binding code for you in the current module.

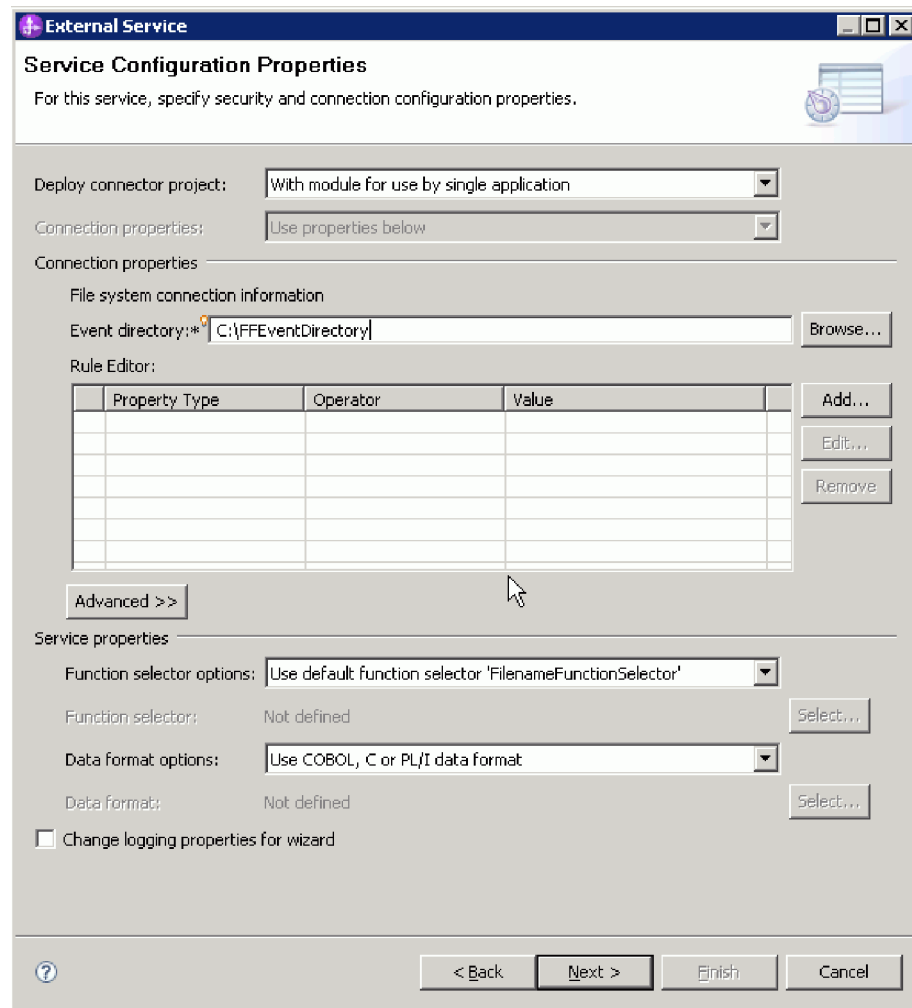


Figure 34. The Service Configuration Properties window

- h. Optional: If the input file contains multiple COBOL program source files, you can enable file splitting by size or by delimiter. To enable file splitting, click **Advanced** and then click **Advanced properties**. To enable file splitting by size, you must provide the correct length of each COBOL program source file. You can either open the business object in a text editor and add up the maximum lengths, or look for the content size of DFHCOMMAREA at the top of the file. See “Specify criteria to split file content” on page 188.
- i. Click **Next**.
- j. In the Operations window, click **Add**.
- k. In the Operation window, select **User defined type** for the date type. Click **Next**.
- l. For the input type, click **Browse** and select the generated business object (DFHCOMMAREA). Click **OK**.

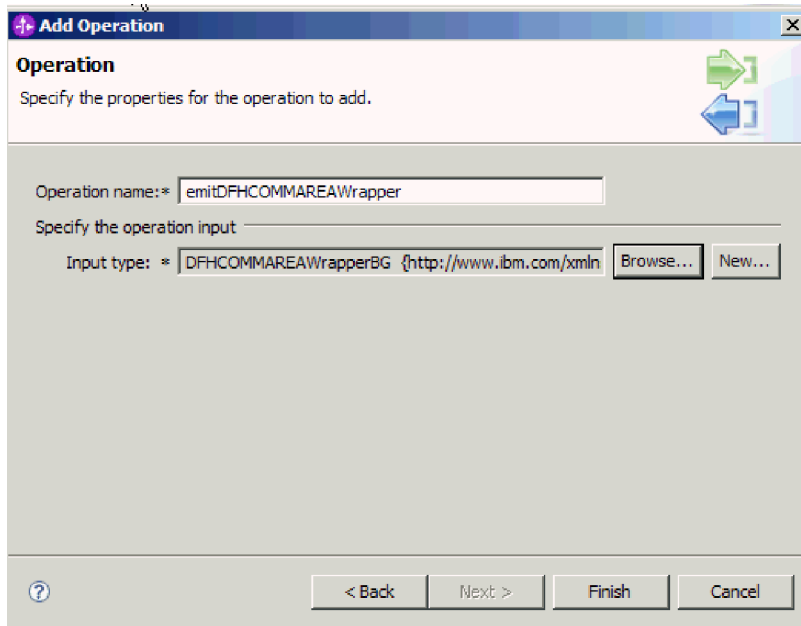


Figure 35. Selecting the input type in the Data Type Selection window

- m. Click **Finish**.
- n. Click **Next** and then **Finish**.

The data bindings used by COBOL copybook, WSDL files, export files, and other artifacts are generated. See the Project Explorer window for the generated data binding classes.

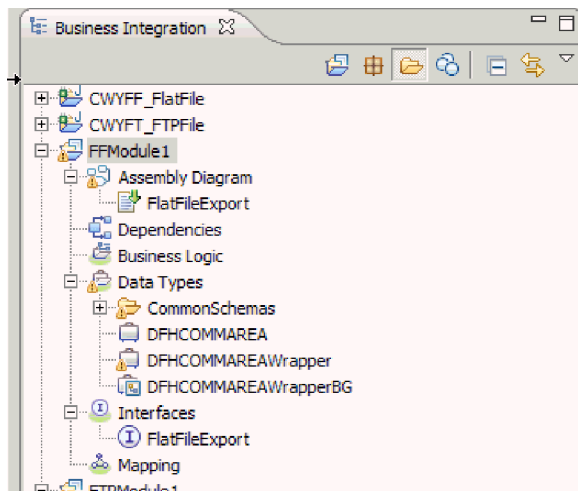


Figure 36. Data bindings used by COBOL copybook, WSDL files, export files, and other artifacts

## Results

A business object, a wrapper business object, and a business graph are created for the COBOL program source file for the inbound module. Artifacts are generated for an inbound operation that uses COBOL copybook data binding. This module can be deployed on WebSphere Process Server or WebSphere Enterprise Service Bus and tested for an inbound operation.



## What to do next

Deploy the module.

### Related reference

“Activation specification properties” on page 176

Activation specification properties hold the inbound event processing configuration information for an export. You set activation specification properties through either the external service wizard or the administrative console.

---

## Creating a simple service with the adapter pattern wizard

Adapter patterns provide a quick and easy way of creating a simple service with an adapter.

### Before you begin

A module has already been created called RetrieveAFileModule and a business object called Customer has already been created. If you are using WebSphere Application Server environment variables to specify local files and directories, you have defined them using the WebSphere Process Server administrative console.

### About this task

The following adapter patterns are available for the adapter for Flat Files:

Table 10.

Adapter pattern	Description
Inbound Flat File pattern	The Flat File inbound pattern creates a service that retrieves a file in a specific directory on the local file system. If the file is not in an XML format, you can specify a data handler that will transform from the file content format to business objects. The file content can be split if the content contains multiple copies of the data structure for processing.
Outbound Flat File pattern	The Flat File outbound pattern creates a service that stores data in a file in a specific directory on the local file system. If the required output format is not an XML format, you can specify a data handler that will transform the business object to the file content format.

In this example, we create a Flat File inbound service that receives a file from the file system for processing. The completed service in this example will read in a file and split the contents into separate files based on a delimiter.

Complete the following steps to create a service with the adapter pattern wizard:

### Procedure

1. Right-click **RetrieveAFileModule** within the **Business Integration** section of the WebSphere Integration Developer window and select **New** → **From Patterns**. The New From Pattern window opens.
2. Select **Create an inbound Flat File service to read from a local file** and click **Next**.

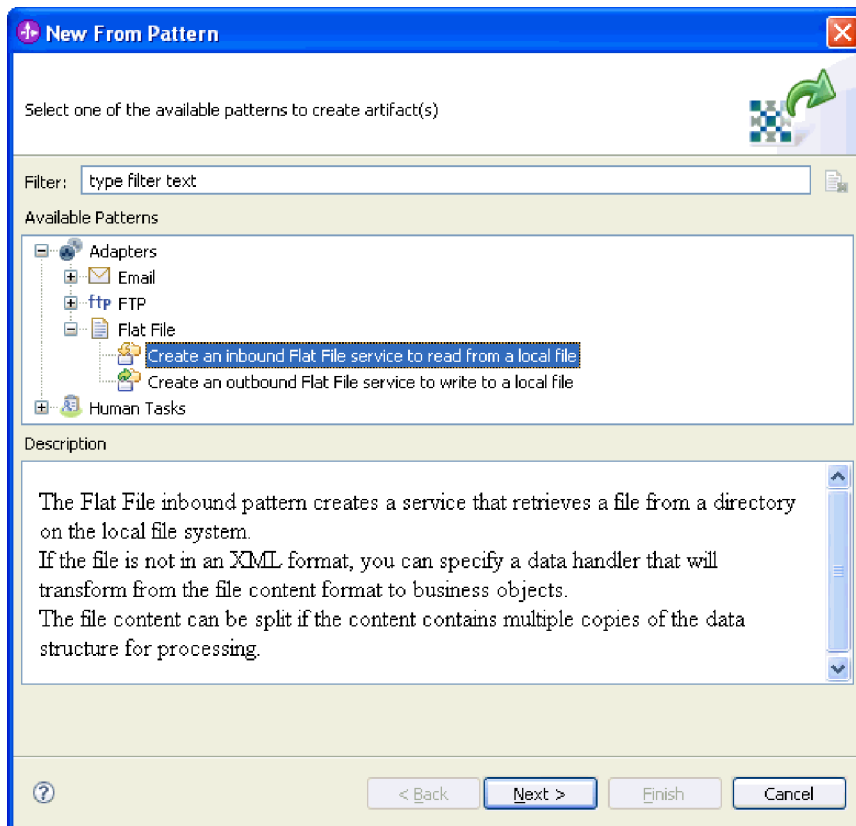


Figure 37. New From Pattern window

3. In the New Inbound Flat File Service window, change the name to something meaningful such as FlatFileInboundInterface and click Next.

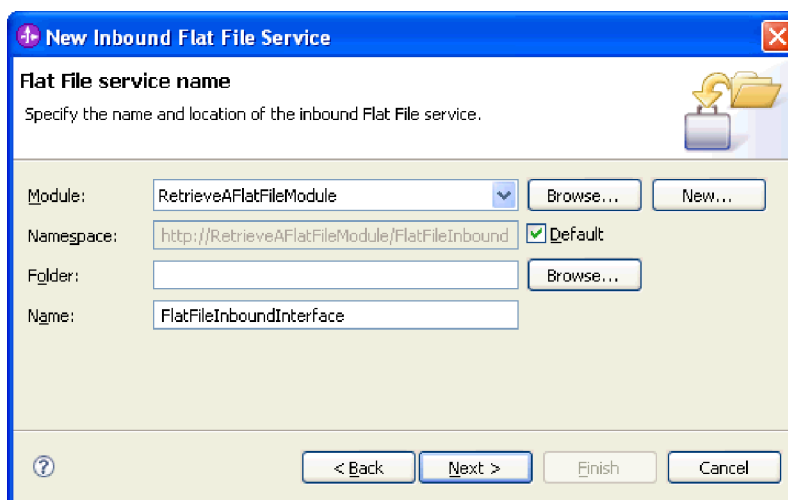


Figure 38. Flat File service name window

4. In the Business object and directory window, click **Browse** and navigate to the **Customer** business object.
5. Specify the directory where you placed the input file, in this case the FFInboundEvents directory, and click **Next**. To use a WebSphere Application Server environment variable for this value, specify the name of the variable in

braces, preceded by a \$ symbol. For example: `${FFINBOUNDEVENTS}`.

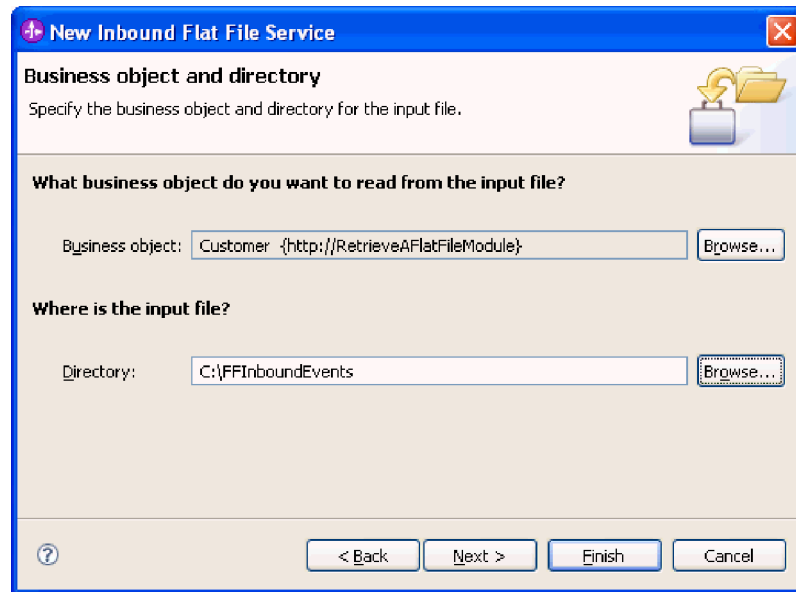


Figure 39. Business object and directory window

6. In the Input file format and file content split option window, accept the default XML input file format or select **Other** and specify a data handler to transform the data from your native format to the business object format.
7. Select **Split file content by delimiter** and enter your delimiter, which is `####;\r\n` in this example. Click **Next**.

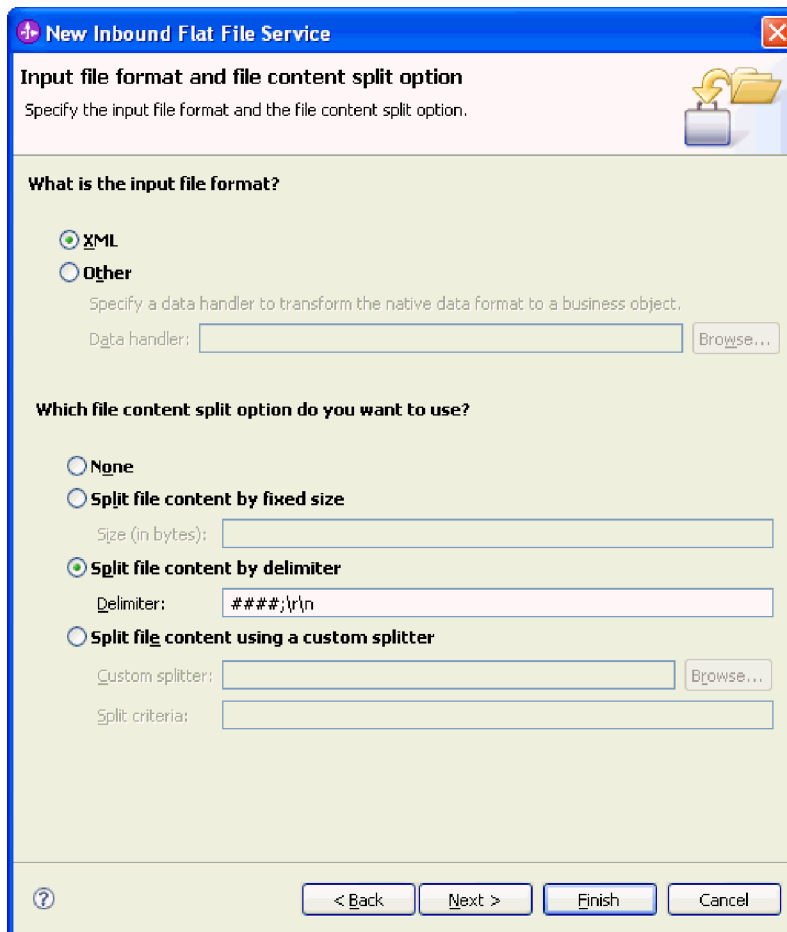


Figure 40. Input file format and file content split option window

8. In the Archive directory and wrapper business object window, specify the **Local archive directory**, which is FFInboundArchive in this example. To use a WebSphere Application Server environment variable for this value, specify the name of the variable in braces, preceded by a \$ symbol. For example: `${FFINBOUNDARCHIVE}`. Select **Use a wrapper business object to contain additional input file information** if you want to include the adapter-specific information. Click **Finish**.

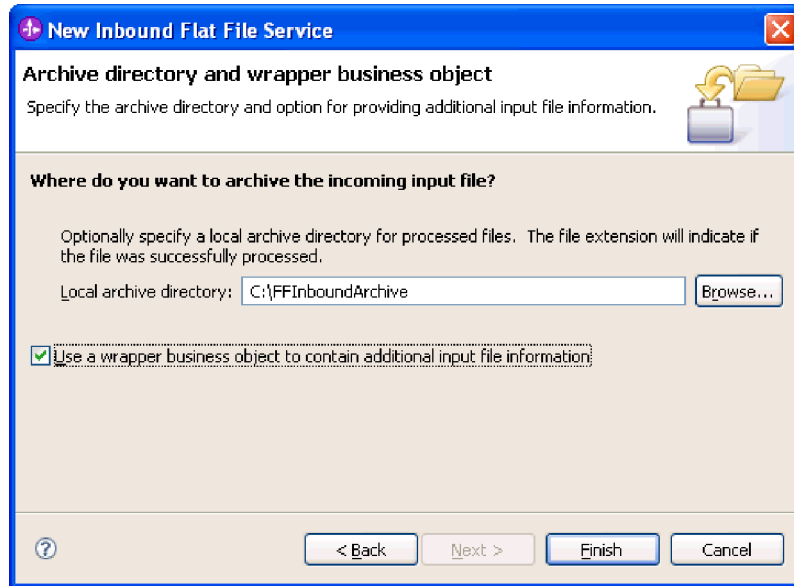


Figure 41. Archive directory and wrapper business object window

## Results

The inbound service is created, which includes the following artifacts:

Table 11.

Artifact	Name	Description
Export	FlatFileInboundInterface	The export exposes the module externally, in this case, to the WebSphere Adapter for Flat Files.
Business objects	Customer, CustomerWrapper	The Customer business object contains the fields for customer data such as name, address, city, and state. The CustomerWrapper business object contains additional fields for adapter-specific information.
Interface	FlatFileInboundInterface	This interface contains the operation that can be invoked.
Operation	emitCustomerInput	emitCustomerInput is the only operation in the interface.

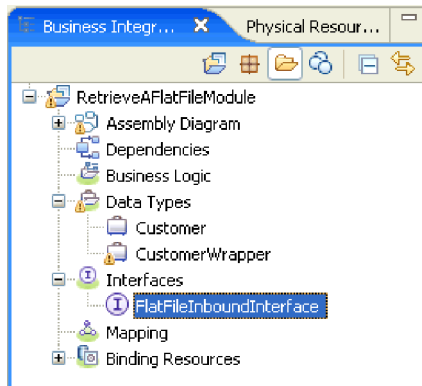


Figure 42. The **Business Integration** section of the *WebSphere Integration Developer* window with the new artifacts

---

## Creating the project

To create and deploy a module, you start the external service wizard in WebSphere Integration Developer. The wizard creates a project that is used to organize the files associated with the module.

### About this task

Start the external service wizard to create a project for the adapter in WebSphere Integration Developer. If you have an existing project, you can select it instead of having the wizard create one.

To start the external service wizard and create a project, use the following procedure.

### Procedure

1. To start the external service wizard, go to the Business Integration perspective of WebSphere Integration Developer, and then click **File** → **New** → **External Service**.
2. In the New external service window, make sure **Adapters** is selected, and click **Next**.

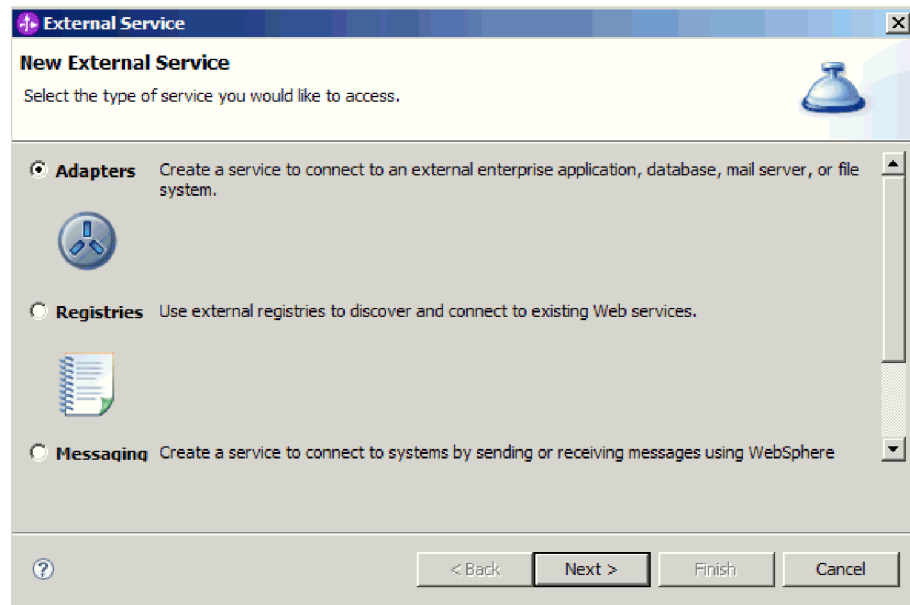


Figure 43. The New external service window

3. From the Select an Enterprise Service Resource Adapter window, create a project or select an existing project.
  - To create a project, perform the following steps:
    - a. Select **IBM WebSphere Adapter for Flat Files** and click **Next**.
    - b. In the Connector Import window, provide another name for the project (to use a name other than **CWYFF\_FlatFile**), select the server (for example, **WebSphere Process Server v6.1**) and click **Next**.
  - To select an existing project, perform the following steps:
    - a. Expand **IBM WebSphere Adapter for Flat Files** .
    - b. Select a project.  
 For example, if you have an existing project named **CWYFF\_FlatFiles**, you can expand **IBM WebSphere Adapter for Flat Files** and select **CWYFF\_FlatFile**, as shown in the following figure.

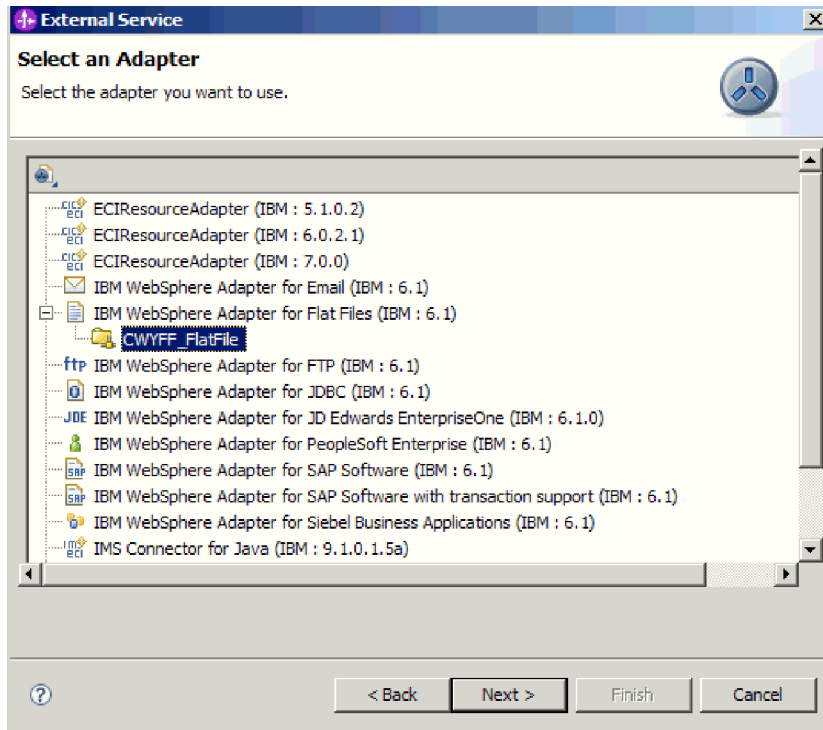


Figure 44. The Select an Enterprise Service Resource Adapter window

- c. Click Next.

### Results

A new project is created and is listed in the Business Integration window.

### What to do next

---

## Configuring the module for outbound processing

To configure a module to use the adapter for outbound processing, use the external service wizard in WebSphere Integration Developer to build business services, specify data transformation processing, and generate business object definitions and related artifacts.

### Related concepts

“Outbound processing” on page 3

During outbound processing, the adapter receives a request from the module, in the form of a business object, to perform an operation on a file in the local file system. The adapter performs the requested operation and returns a business object, if applicable, that represents the result of the operation to the component.

## Setting deployment and runtime properties

After you have decided whether your module will be used for outbound or inbound communication with the local file system, you must configure the managed connection factory properties so that the adapter can make the connection between the module and the local file system.

### Before you begin



Before you can set the properties in this section, you must have created your adapter module. It should be displayed in WebSphere Integration Developer below the adapter project. For more information on creating the adapter project, see “Creating the project” on page 80.

### About this task

To set deployment and runtime properties, follow this procedure. For more information on the properties in this topic, see “Managed connection factory properties” on page 159.

### Procedure

1. On the Processing Direction window, select **Outbound** and click **Next**.

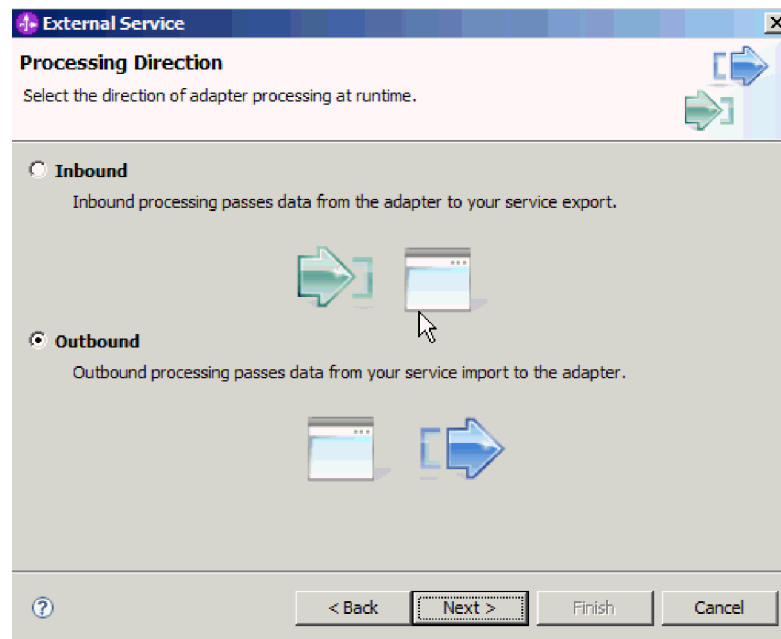


Figure 45. Choosing inbound or outbound processing in the external service wizard

2. On the Service Configuration Properties window, in the **Deploy connector project** field, select **With module for use by single application**.
3. Define the Connection properties for your module. For more details on the properties found on this window, see the Managed connection factory properties topic.

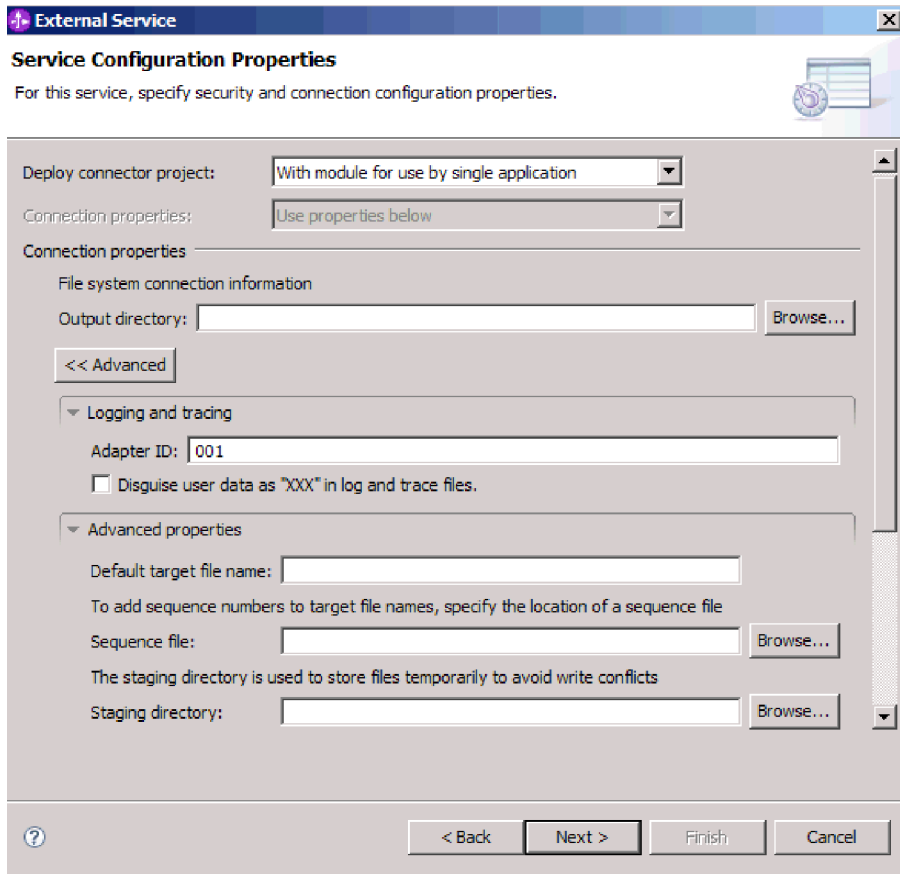


Figure 46. Setting the connection properties

4. Optional: If you have multiple instances of the adapter, expand **Logging and tracing** and set Adapter ID to a value that is unique for this instance. For more information about this property, see the Managed connection factory properties reference topic.
5. \*\*\* Confidential trace info goes here \*\*\*
6. Optional: If you want to specify the log file output location or define the level of logging for this module, select the **Change logging properties for wizard** check box. For information about setting logging levels, see the section about configuring logging properties in the Troubleshooting and support topic.
7. Click **Next**.

## Results

The adapter saves the connection properties.

## What to do next

Select a data type for the module and name the operation associated with the chosen data type.

### Related reference

“Connection properties for the wizard” on page 156

Connection properties are used to build a service description and save the built-in artifacts. These properties are configured in the external service wizard.

## Selecting the operation and data type

Use the external service wizard to select the outbound operation that will be used to access functions on the local file system and the data type to be used with it. The operations supported are Create, Append, Overwrite, Delete, Exists, List, and Retrieve. The external service wizard gives you a choice of three data types: generic FlatFile business object, generic FlatFile business object with business graph, and user defined type. Each data type corresponds to a business object structure.

### Before you begin

You must have specified the connection properties for the adapter to connect to the local file system before you can complete the steps below.

### About this task

To select an outbound operation and the data type to be used with it, follow this procedure.

### Procedure

1. In the Operations window, click **Add**.

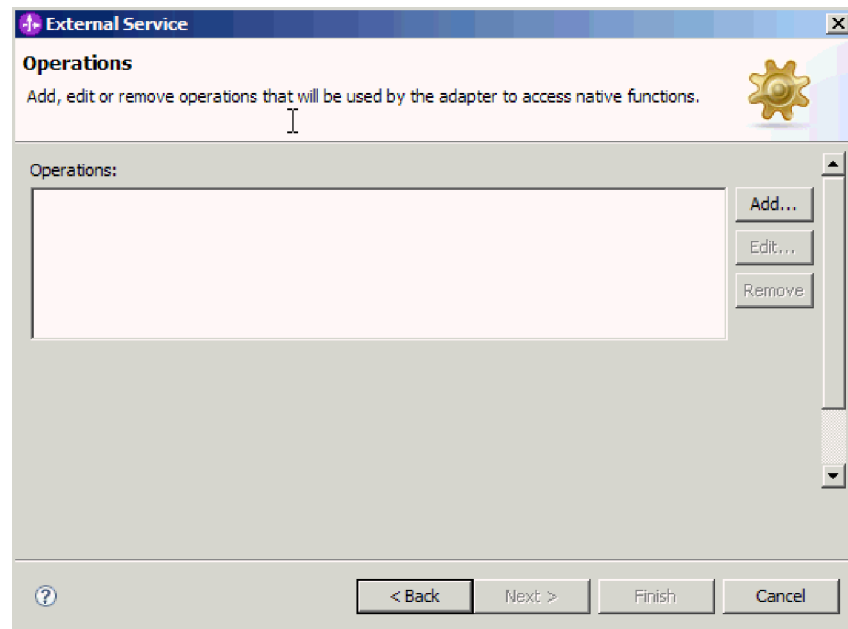


Figure 47. Adding an operation

2. In the Add Operation window, open the **Operation kind** list and select an operation. In this example, the **Create** operation is selected.
3. In the Add Operations window, select a data type and click **Next**. In this example, the **User defined** type is selected.

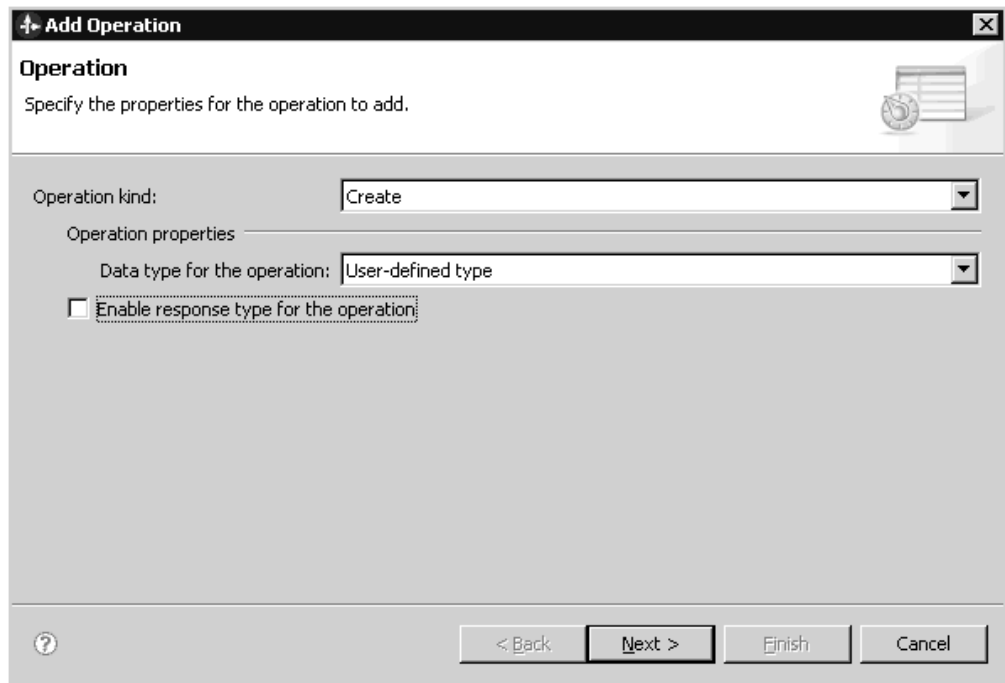


Figure 48. Selecting a data type for the operation

For Delete, Retrieve, Exists, and List operations, only the generic data type (generic FlatFile business object or generic FlatFile business object with business graph) is supported as input. If you choose user defined type with one of these operations, you must provide a user defined data binding to support it.

For Create, Append, and Overwrite operations, the choices are user defined type, generic FlatFile business object, and generic FlatFile business object with business graph. For more information about data types, see the topic that describes business object structures in this documentation.

4. Optional: For Create, Append, and Overwrite operations, if you want the file name returned or if you are generating a unique file name or have enabled file sequencing, select the **Enable response type for the operation** check box. For the Exists, List and Retrieve operations, output is required, and by default the **Enable response type for the operation** check box is selected. For the Delete operation, select the **Enable response type for the operation** check box if you want a value of true returned when the operation is successful.

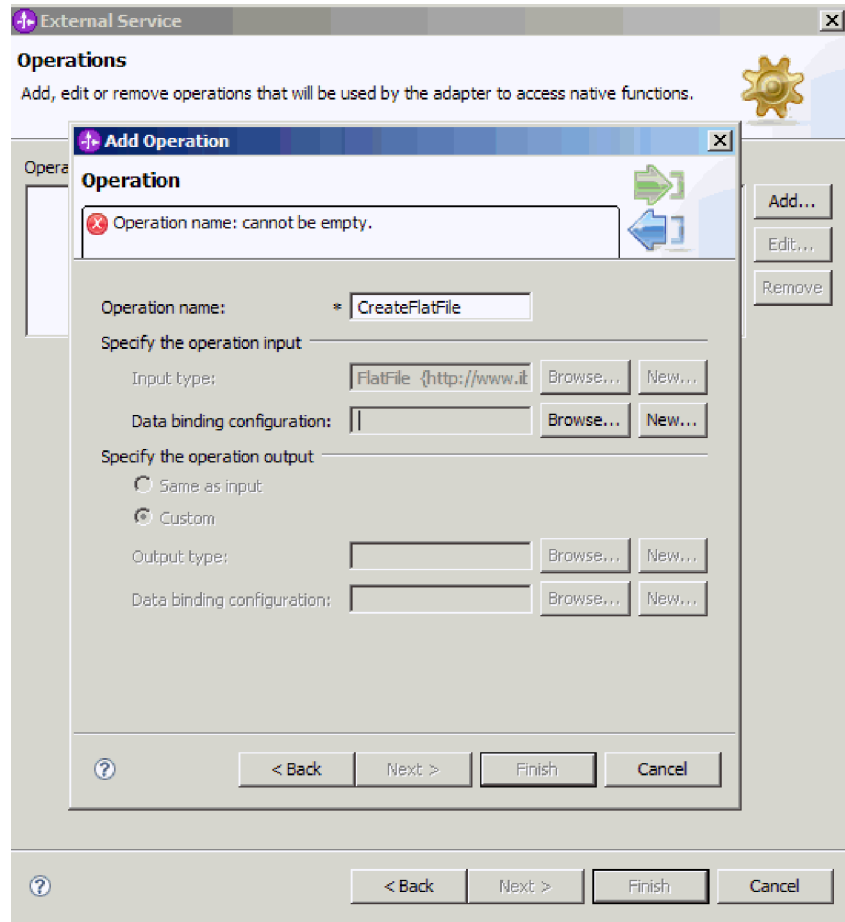


Figure 49. Naming the operation and specifying the input data type

5. Click **Next**.
6. In the Add Operation screen, type an **Operation name**. Name the operation something meaningful. For information about the types of operations the adapter can perform, see the topic on Supported operations in this documentation.

**Note:** Names cannot contain spaces.

By default, the data type for the output is set to CreateResponse or CreateResponseBG.

7. Select the Input type. Click on **Browse** and choose the business object you created earlier. If you specified a generic data type (generic FlatFile business object or generic FlatFile business object with business graph), the Input type is set by default to FlatFile or FlatFileBG.

## Results

A data type is defined for the module and the operation associated with this data type is named.

## What to do next

Add and configure a data binding to be used with the module.

## Configuring the data binding

Each data type has an equivalent data binding that is used to read the fields in a business object and fill the corresponding fields in the file. In the external service wizard, you add a data binding to your module and configure it to correspond with your data type. This way, the adapter knows how to populate the fields in the file with the information it receives in the business object.

### Before you begin

You must have selected an operation and the data type to be used with it.

### About this task

To add and configure a data binding for the module, follow this procedure.

**Note:** Data bindings can be configured prior to running the external service wizard using WebSphere Integration Developer. To do this, select **New** → **Resource configuration** in WebSphere Integration Developer and complete the data binding screens described in this documentation.

### Procedure

1. On the Add Operation window, select **New** for the operation input Data binding configuration field. You do this the first time you set the data binding. To use the same data binding configuration later, click on **Browse** and select it.
2. Type a **Name** for the data binding (this example uses DBConfig), and click **Next**.

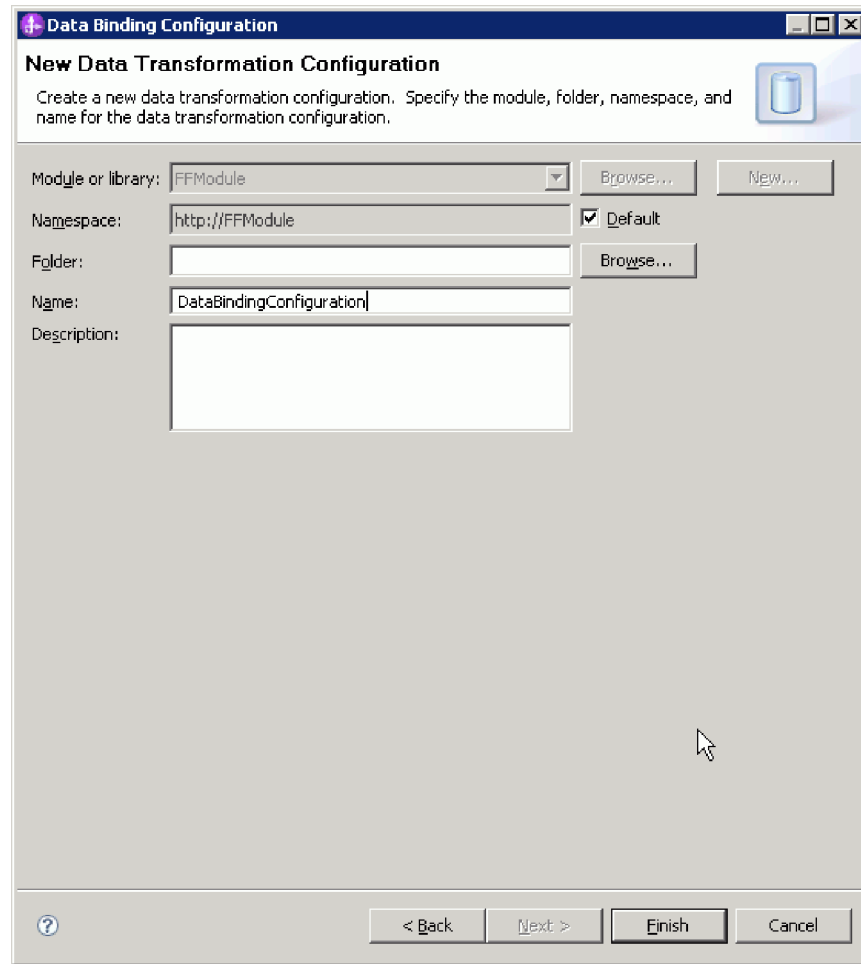


Figure 50. Naming the data binding

3. Click **Next**.

### Results

A data binding is configured for use with the module.

### What to do next

Select the data handler configuration.

## Configuring data handlers

Data handlers perform the conversions between a business object and a native format.

### Before you begin

You must have created a data binding before you specify data handlers for the module. Also, you must have predefined business objects using WebSphere Integration Developer Business Object Editor. If you stop the wizard here to create business objects, you will need to start the wizard steps from the beginning.

**Note:** Data handlers can be configured prior to running the external service wizard using WebSphere Integration Developer. To do this, select **New** → **Resource configuration** in WebSphere Integration Developer and complete the data handler screens described in this documentation.

### About this task

To specify data handlers, follow this procedure.

### Procedure

1. In the Add Operation window, click **New** and provide a name for the data handler configuration (this example uses `DataBindingConfiguration`). You do this the first time you set the data handler. To use the same data handler later, click on **Browse** and select it.

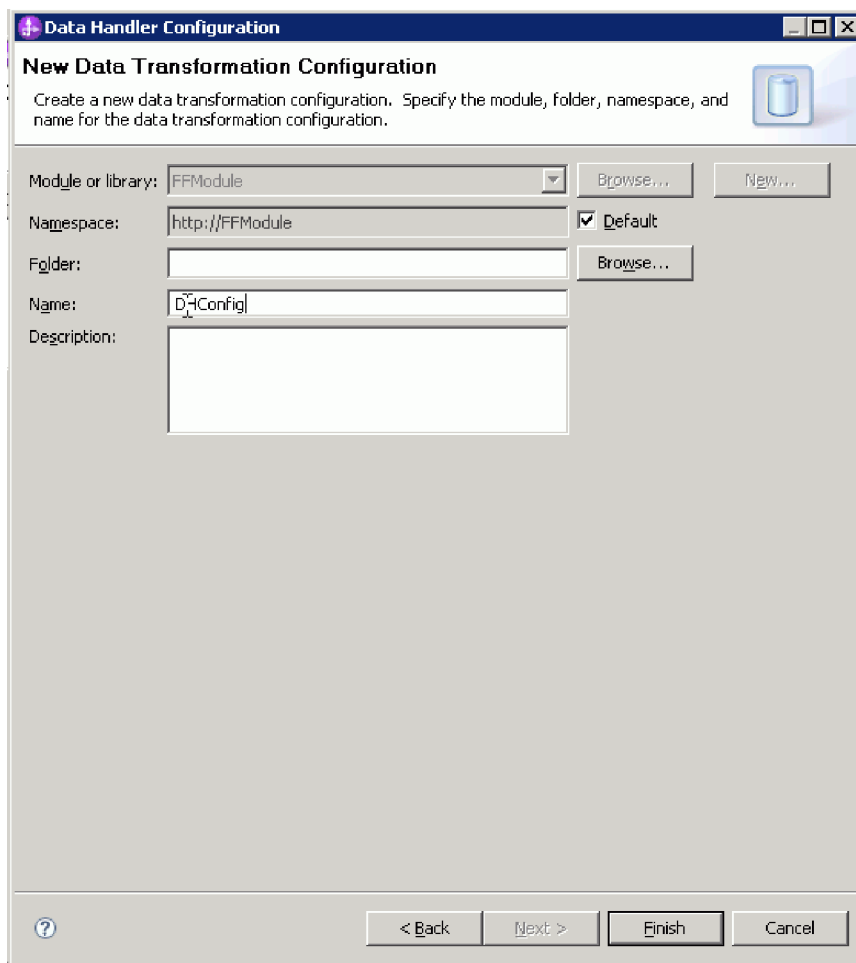
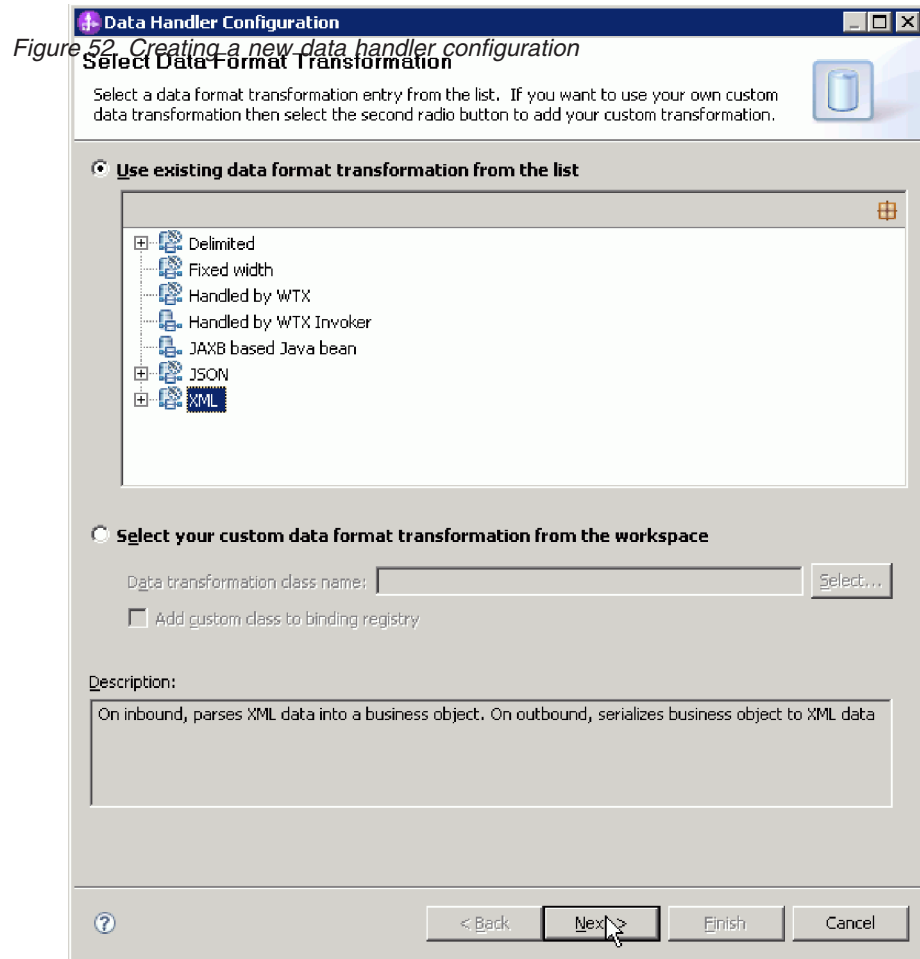


Figure 51. Specifying a name for the data handler configuration

2. Click **Next**.
3. In the Data Binding Properties window, click the dropdown list next to the Binding type property. Two choices are provided: `DataBinding` and `DataHandler`. To use a data binding developed for an earlier version of the adapter, select `DataBinding`. To configure a new data handler, select `DataHandler`. Click **New** to create a new data handler configuration.
4. Click **New**.



5. On the New Data Handler Configuration window, specify the Module, Namespace, Folder, and Name for the data handler configuration.
6. Click **Next**.
7. Choose the class name for the data handler. In the Select Data Format Transformation window, click **Use existing data format transformation from the list** option. A list of available data handler classes is displayed. Select the data handler class (this example uses XML data handler). Click **Next**.



8. In the Data Transformation Properties window, specify the encoding. The default is UTF-8.

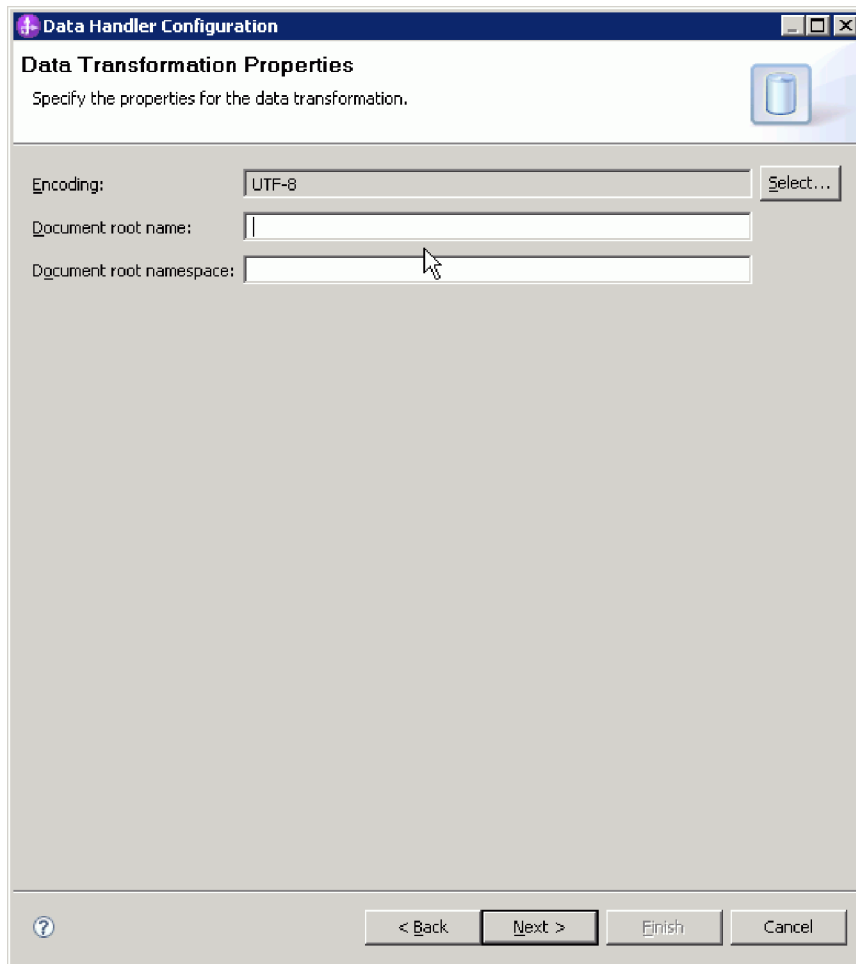


Figure 53. Specifying the encoding for the data handler configuration

9. Click **Finish**.
10. Choose the data binding configuration for the operation output. In the Add operation window, click on **Browse** for the output Data binding configuration field. Because the adapter provides only one data binding and this was configured when the operation input DataBinding type was set, you select the same data binding type (DBConfig) for the operation output DataBinding type.

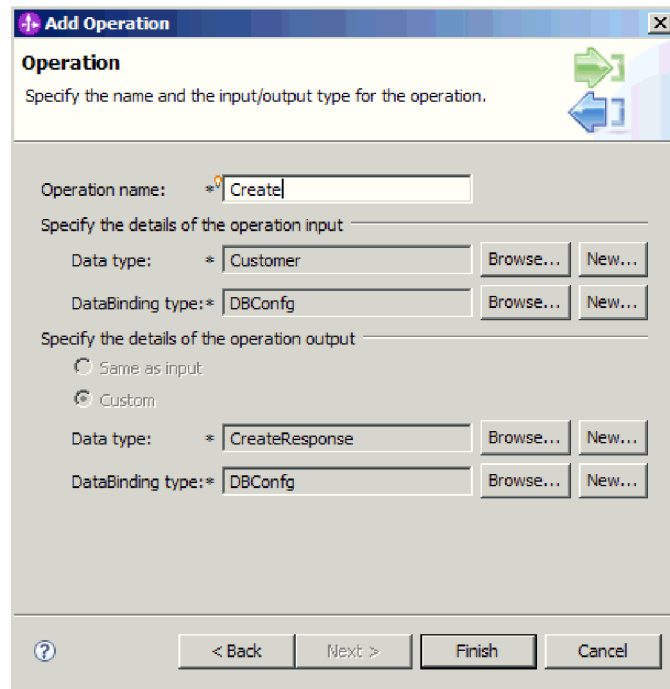


Figure 54. Choosing the data binding configuration for the operation output

11. Click **Finish**. The next screen shows the Create operation that has been added, with the interaction specification properties.

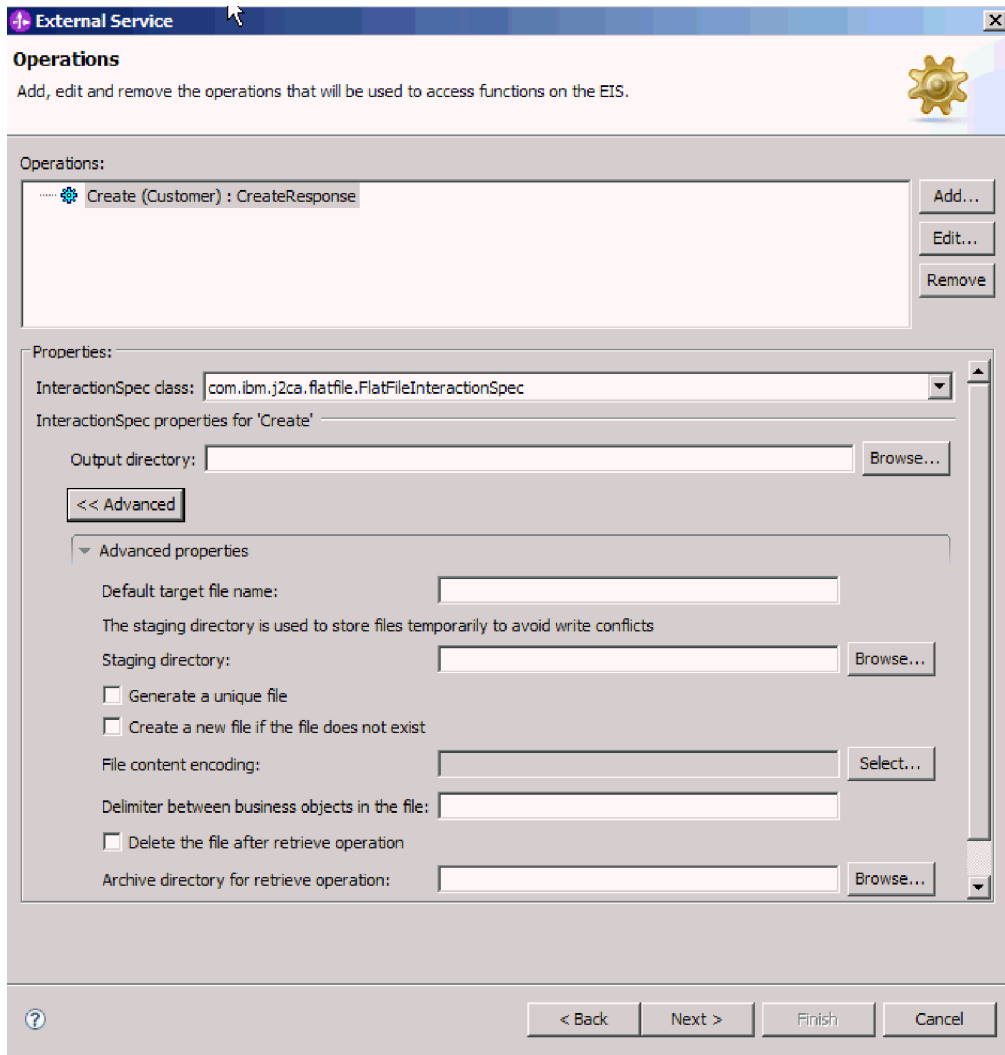


Figure 55. The Create operation with InteractionSpec properties

12. Click **Finish**.

### Results

Data handlers are created.

### What to do next

Specify interaction specification properties and generate artifacts for the module.

## Setting interaction properties and generating the service

Interaction properties are optional. If you choose to set them, the values you specify will appear as defaults in all parent business objects generated by the external service wizard. While creating artifacts for the module, the adapter generates an import file. The import file contains the operation for the top-level business object.

### Before you begin

To set interaction specification properties and generate artifacts for your module, you must have already configured data bindings and selected business objects.

### About this task

To set interaction specification properties and generate artifacts, follow this procedure. For more information about interaction specification properties, see the reference topic devoted to it in this documentation.

### Procedure

1. Optional: To set interaction specification properties, complete these steps:
  - a. In the Operations window, click **Advanced**.
  - b. Type values for any fields you wish to set as defaults.
  - c. Click **Next**.

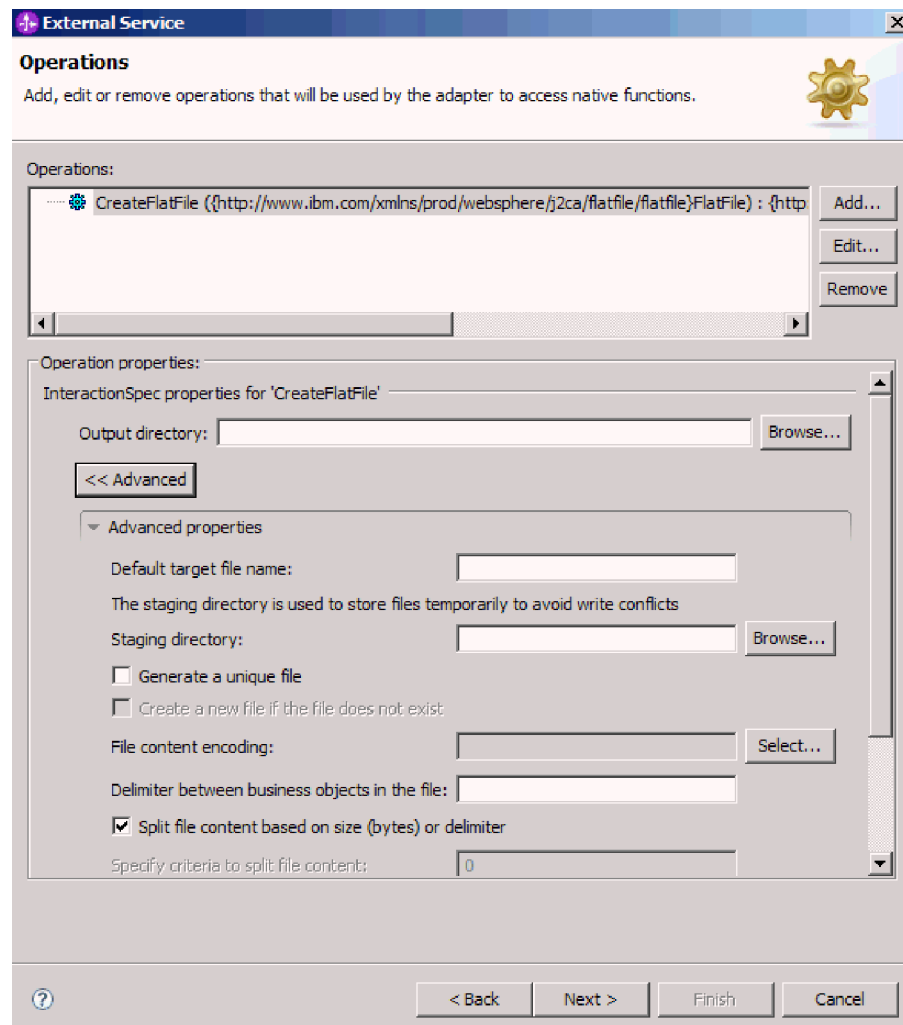


Figure 56. Setting interaction specification properties

2. On the Operations window, click **Next**. On the Generate Service screen, supply a name for the interface. This is the name that will display in the WebSphere Integration Developer assembly diagram.

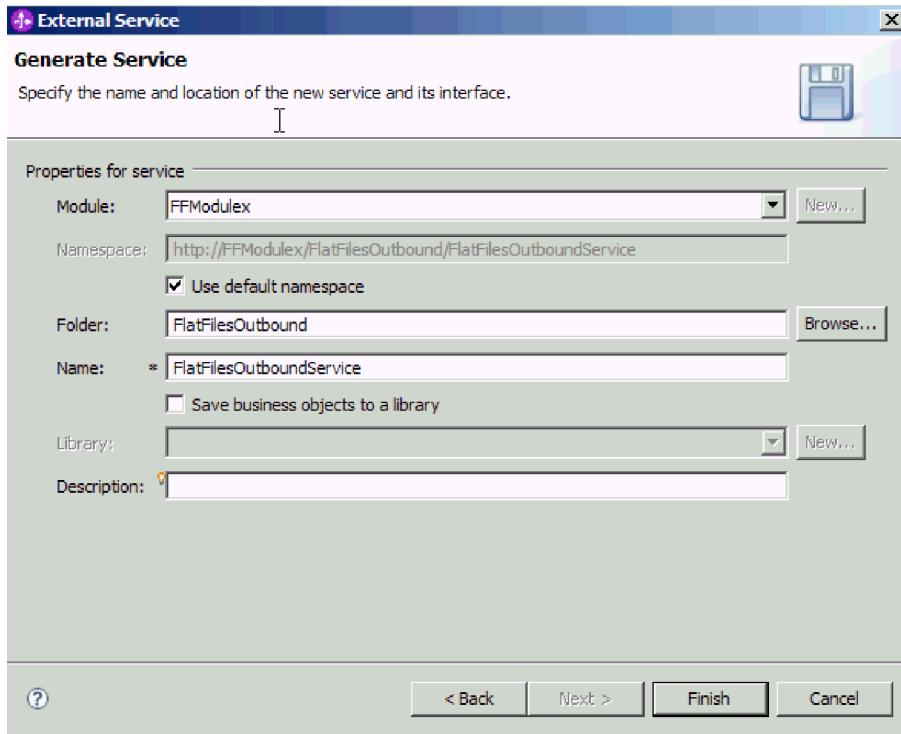


Figure 57. Naming the service

3. Click **Finish**.

### Results

The WebSphere Integration Developer generates the service and an import. The outbound artifacts that are created are visible in the WebSphere Integration Developer Project Explorer under your module.

### What to do next

Deploy the module.

#### Related reference

“Outbound configuration properties” on page 154

WebSphere Adapter for Flat Files has several categories of outbound connection configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter and managed connection factory properties after you deploy the module to WebSphere Process Server or WebSphere Enterprise Service Bus using WebSphere Integration Developer or the administrative console, but connection properties for the external service wizard cannot be changed after deployment.

“Inbound configuration properties” on page 171

WebSphere Adapter for Flat Files has several categories of inbound connection configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter and activation specification properties after you deploy the module using WebSphere Integration Developer or the administrative console, but connection properties for the external service wizard cannot be changed after deployment.

“Globalization” on page 193

WebSphere Adapter for Flat Files is a globalized application that can be used in

multiple linguistic and cultural environments. Based on character set support and the locale of the host server, the adapter delivers message text in the appropriate language. The adapter supports bidirectional script data transformation between integration components.

---

## Configuring the module for inbound processing

To configure a module to use the adapter for inbound processing, use the external service wizard in WebSphere Integration Developer to build business services, specify data transformation processing, and generate business object definitions and related artifacts.

### Related concepts

“Inbound processing” on page 12

Adapter for Flat Files supports inbound event processing. It polls the local file system at specified intervals for events, such as the creation or modification of a file. When it detects an event, it converts the event data into a business object and sends it to the module for processing.

### Related reference

“Custom file splitting” on page 153

You can implement a custom class containing the splitting logic. The adapter provides a Java interface for the class. The details of the interface are shown below.

## Setting deployment and runtime properties

After you have decided whether your module will be used for outbound or inbound communication with the enterprise information system (local file system), you must configure the activation specification properties, which hold the inbound event processing configuration information for the export.

### Before you begin

Before you can set the properties in this section, you must have created your adapter module. It should be displayed in WebSphere Integration Developer below the adapter project. For more information on creating the adapter project, see “Creating the project” on page 80.

### About this task

To set the activation specification properties, follow this procedure. For more information about the properties in this topic, see “Activation specification properties” on page 176.

### Procedure

1. In the Processing Direction window, select **Inbound** and click **Next**.

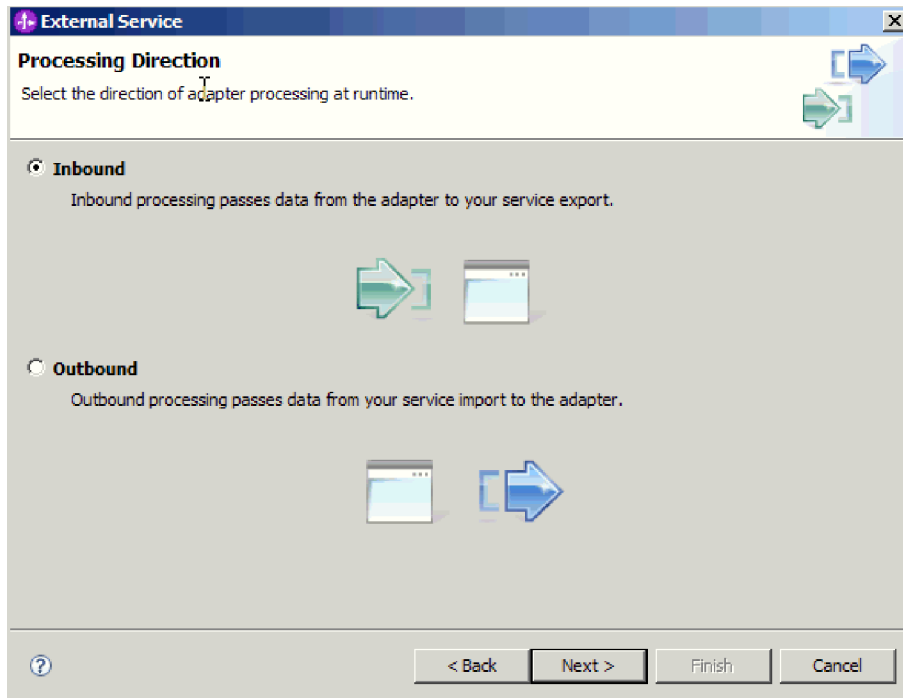


Figure 58. Choosing inbound or outbound in the external service wizard

2. In the Service Configuration Properties window, in the **Deploy connector project** field, select **With module for use by single application**.
3. In the Service Configuration Properties window, define the activation specification properties for your module. For more details on the properties found on this window, see "Activation specification properties" on page 176.



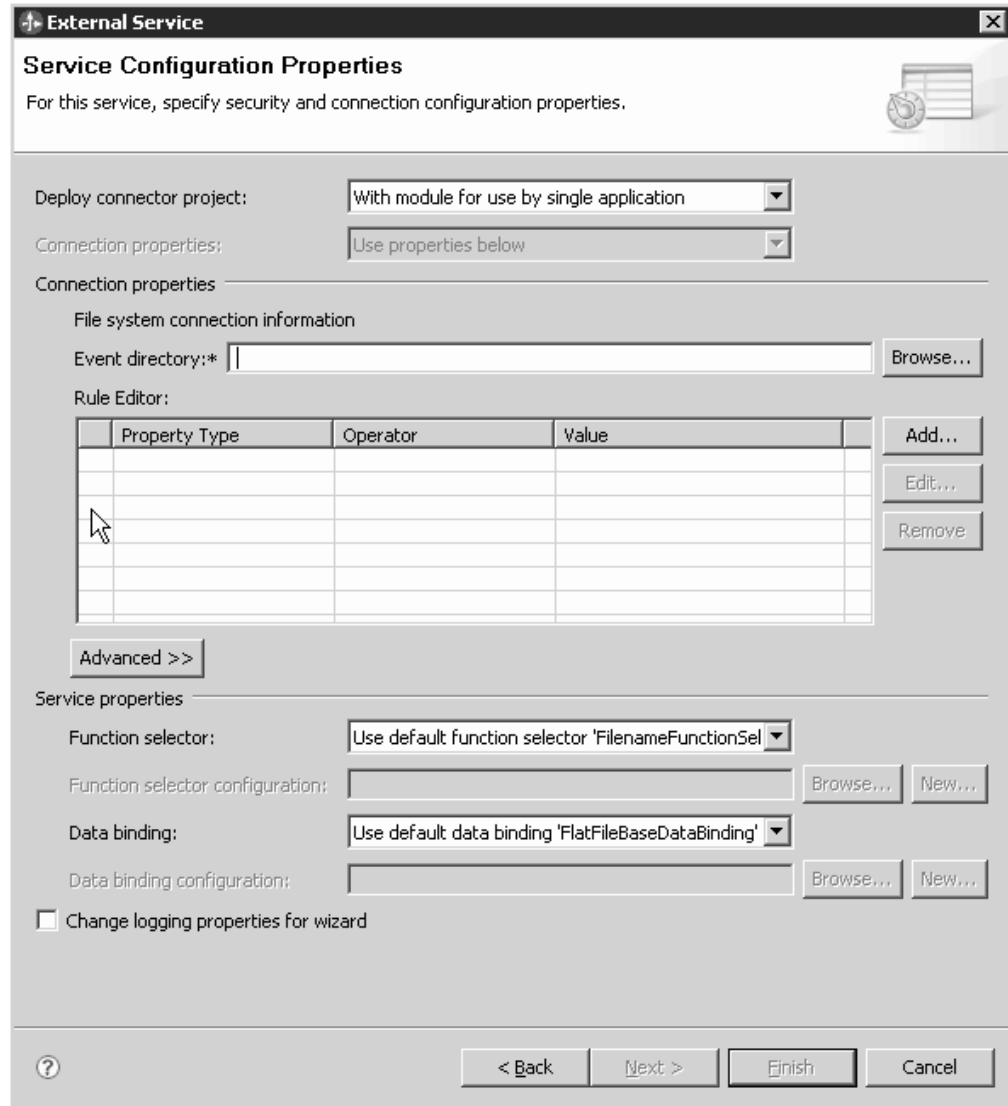


Figure 59. Setting the connection properties

4. For the **Event directory** property, specify the directory in the local file system where the event files are stored.
5. Click **Advanced** and expand the Event polling configuration, Event delivery configuration, Event persistence configuration, Advanced Properties, File archiving configuration, Bidi properties, and Logging and tracing sections to specify additional properties.
  - a. Optional: In the Event polling configuration section, select **Retry EIS connection on startup**. If you select this property, the adapter continues trying to connect to a system to which it could not connect when starting. For more information, see “Retry EIS connection on startup (RetryConnectionOnStartup)” on page 187.

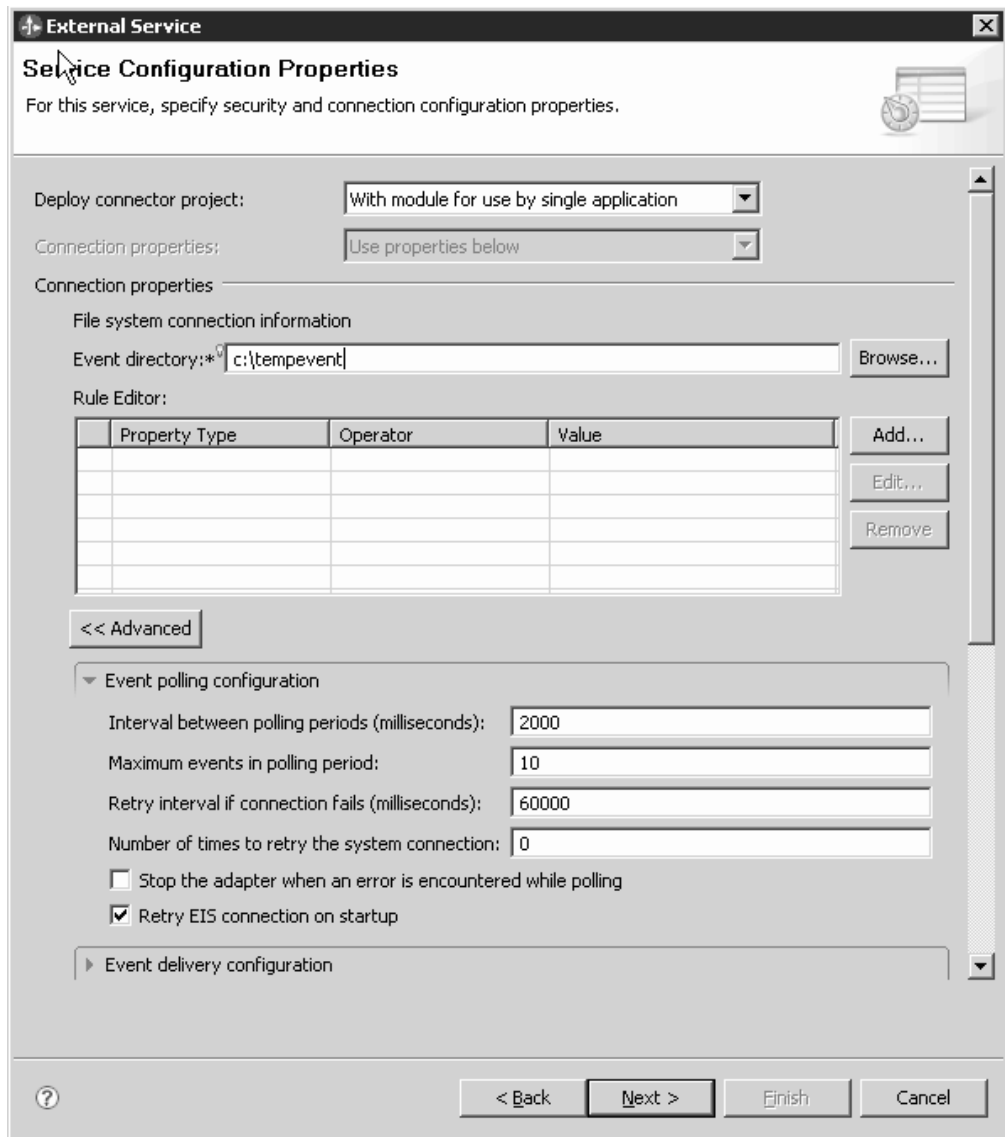


Figure 60. Selecting the Check property if you want to retry EIS connection on startup check box

- b. Optional: In the Advanced Properties section, select a value for the **File content encoding** field. If you are working with binary event data, select BINARY. If you are working with non-binary event data, such as text or XML, select a valid file encoding value, such as UTF-8 (the default value).

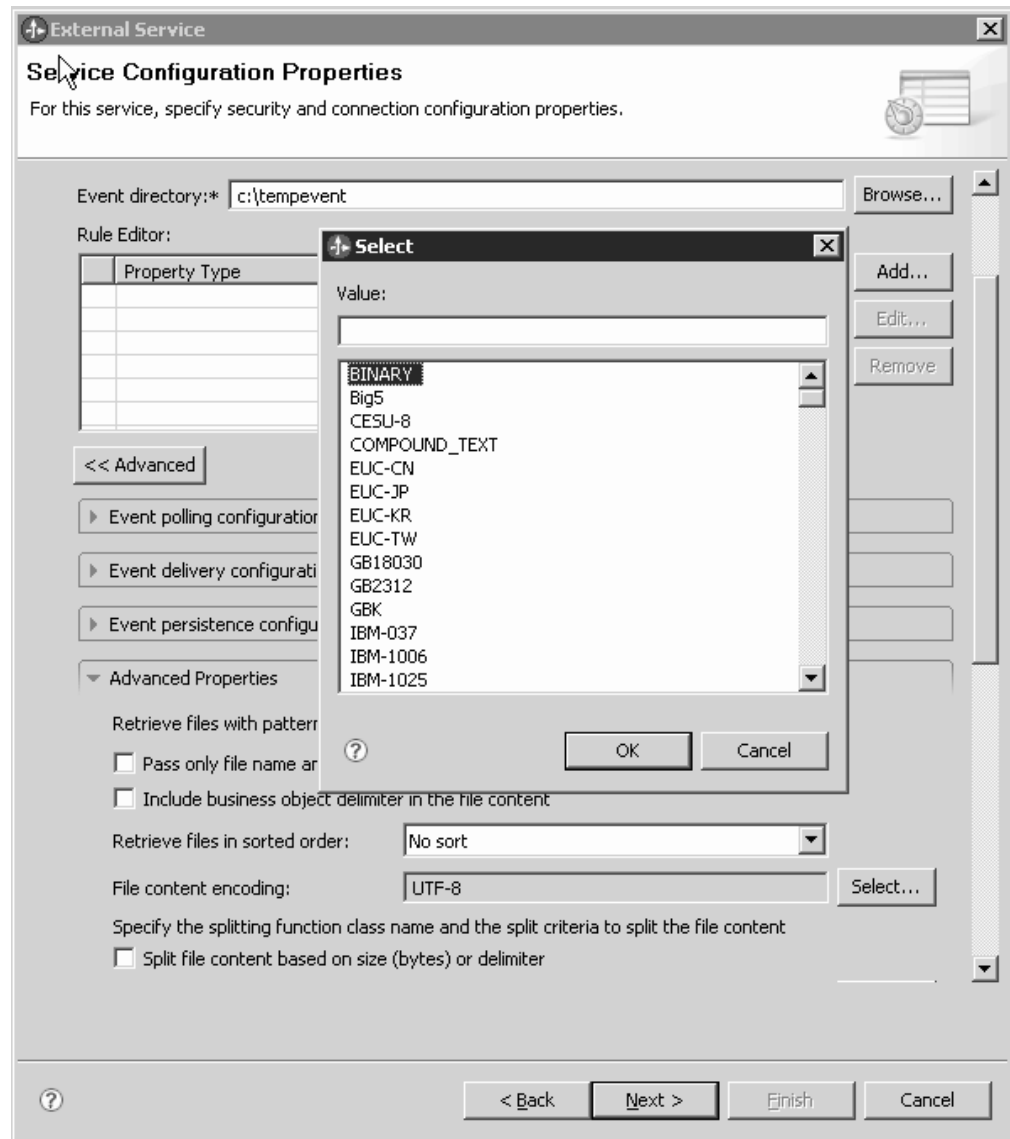


Figure 61. Selecting BINARY in the File content encoding field

- c. Optional: If you have multiple instances of the adapter, expand **Logging and tracing** and set the value for the **Adapter ID** to a value that is unique for this instance. For more information about this property, see “Resource adapter properties” on page 164.
  - d. **\*\*\* Confidential trace info goes here \*\*\***
  - e. Optional: To specify the log file output location or define the level of logging for this module, select the **Change logging properties for wizard** check box. For information about setting logging levels, see “Configuring logging properties” on page 142.
6. For the **Function selector** field, choose whether to use an existing function selector configuration or create a new one. A function selector assigns incoming messages or requests to the correct operation on the service.
    - To use an existing function selector configuration, click **Browse** to see a list of function selectors. See the topic Connection properties for the external service wizard for a description of the available function selectors.

- To use an existing function selector configuration, click **Browse** to see a list of function selectors. See the topic Connection properties for the external service wizard for a description of the available function selectors.

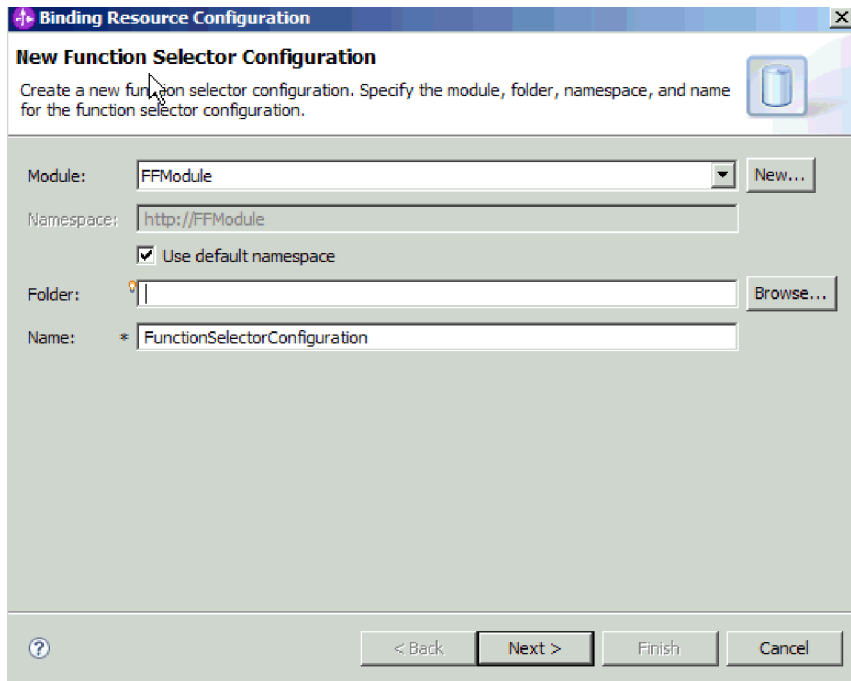


Figure 62. Creating a new function selector configuration

**Note:** The enterprise information system (EIS) function name is not available in the external service wizard. If you want to specify a value other than the default that is generated by the adapter (base classes), you can edit it using the assembly editor.

7. To filter the inbound event file by configuring rules, click **Add** or **Edit** in the Rule Editor table. The rule constitutes three parameters: Property Type, Operator and Value.

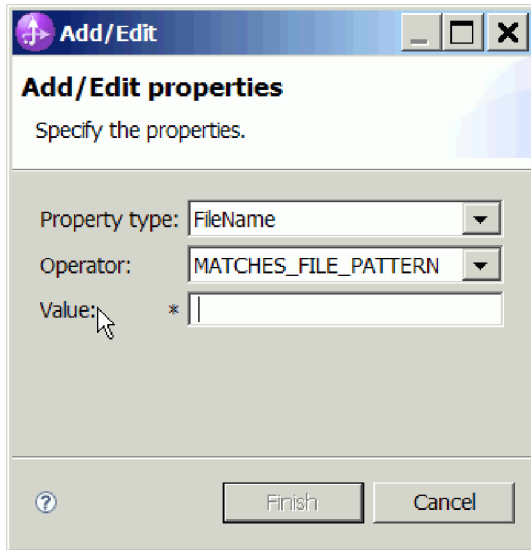


Figure 63. Adding or editing a rule

- a. Select any of the following metadata filtering property types from **Property Type** list.
  - FileName
  - FileSize
  - Directory
  - LastModified
- b. Select the operator for the property type from the **Operator** list. Each of the property type metadata has its own operators. FileName contains MATCHES\_FILE\_PATTERN (matches pattern) and MATCHES\_REGEXP (matches regular expression) as its operators. FileSize contains GT (greater than), LT (less than), GE (greater than or equal to), LE (less than or equal to), EQ (equal to), NE (not equal to) in Bytes as its operators. Directory contains MATCHES\_REGEXP as its operators. LastModified contains GT (greater than), LT (less than), GE (greater than or equal to), LE (less than or equal to), EQ (equal to), NE (not equal to) (in Day of the week or Time) as its operators. The day-of-the-week value will accept only EQ operator.
- c. Type the value for filtering the event file in the **Value** column. You must enter a valid Java regular expression in value for MATCHES\_REGEXP operator.

To configure multiple rules, select **END-OF-RULE** option for each rule from the **Property Type** list.

**Note:** Each expression is grouped by the logical **OR** operator, unless **END-OF-RULE** is selected in the property field. Because you need to select **END-OF-RULE** in a property field to end a rule, all rules will be grouped by the logical **AND** operator, and all expressions within a rule will be grouped by the logical **OR** operator.

**Important:** Do not select **END-OF-RULE** if you want to configure only one rule, otherwise the rule becomes invalid during run time. If you select **END-OF-RULE** from the **Property Type** list and do not select the second rule, the adapter throws an error.

For more information see, “Rule based event filtering (ruleTable)” on page 190.  
8. Click **Finish**.

### **Results**

The adapter saves the activation specification properties.

### **What to do next**

Select a data type for the module and name the operation associated with the chosen data type.

#### **Related reference**

“Connection properties for the wizard” on page 156

Connection properties are used to build a service description and save the built-in artifacts. These properties are configured in the external service wizard.

“Activation specification properties” on page 176

Activation specification properties hold the inbound event processing configuration information for an export. You set activation specification properties through either the external service wizard or the administrative console.

## **Selecting the operation and data type**

Use the external service wizard to select a data type and name the operation associated with this data type. The external service wizard gives you a choice of three data types: generic FlatFile business object, generic FlatFile business object with business graph, and user-defined type. Each data type corresponds to a business object structure.

### **Before you begin**

You must have specified the connection properties for the adapter to connect to the local file system before you can complete the steps below.

### **About this task**

To select a data type and name the operation associated with it, follow this procedure.

### **Procedure**

1. On the Operations window, click **Add**.

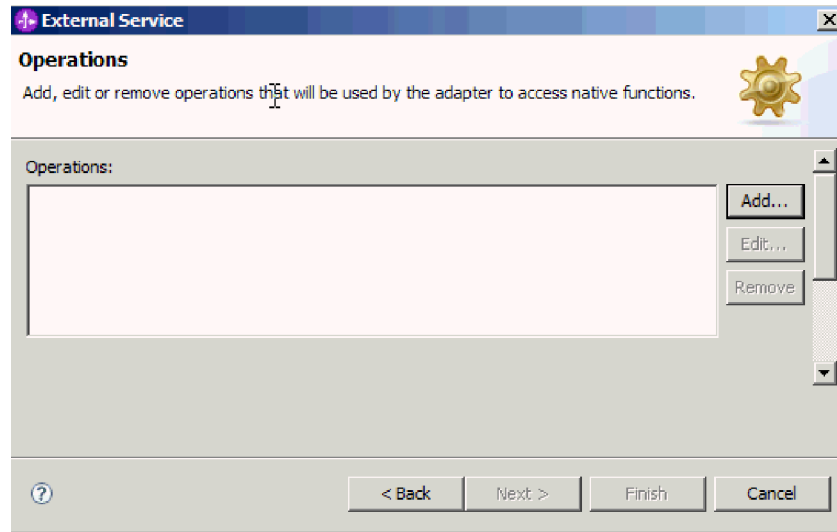


Figure 64. Adding an operation

2. On the Add Operations window, select a data type. Three data types are available: Generic FlatFile business object, Generic FlatFile business object with business graph, and User-defined type. For more information on data types and the types of business objects they are used to produce, see the section devoted to business object structures in this documentation. In this example, **Generic FlatFile business object** is selected.
3. Click **Next**. The Operation window displays the operation name, which is **emitFlatFile**. The emit operation is the only operation available during inbound processing.

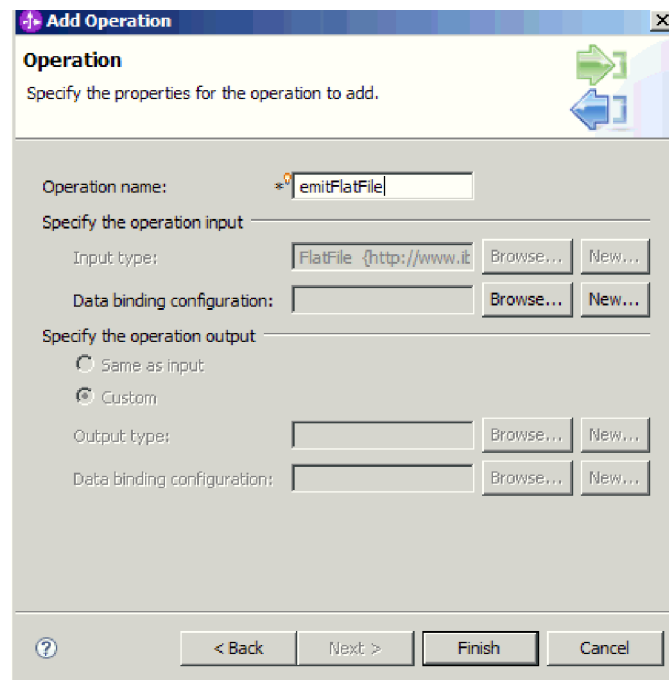


Figure 65. Adding an operation

## Results

A data type is defined for the module and the operation associated with this data type is named.

#### What to do next

Add and configure a data binding to be used with the module.

## Configuring the data binding

Each data type has an equivalent data binding used to read the fields in a business object and fill the corresponding fields in the file. In the external service wizard, you add a data binding to your module and configure it to correspond with your data type. This way, the adapter knows how to populate the fields in the file with information it receives in the business object.

#### Before you begin

You must have selected a data type and chosen an operation name to be associated with the data type.

#### About this task

To add and configure a data binding for the module, follow this procedure.

**Note:** Data bindings can be configured prior to running the external service wizard using WebSphere Integration Developer. To do this, select **New** → **Resource configuration** in WebSphere Integration Developer and complete the data binding screens described in this documentation.

#### Procedure

1. On the Operation window, select **New** for the operation input Data binding configuration field. You do this the first time you set the data binding. To use the same data binding configuration later, click on **Browse** and select it.
2. Optional: On the New Data Binding Configuration screen, the **Module** defaults to the module name you typed earlier in the wizard. If this is not the module that you want to create a data binding for, choose **New** to create a new module.
3. Optional: If you want to choose a new folder for the artifact, click **Browse** and select a new folder location. If you do not browse for a new folder location, the artifacts will be created in the root directory for the module.
4. Type a **Name** for the data binding configuration (this example uses DataBindingConfiguration). Click **Next**.



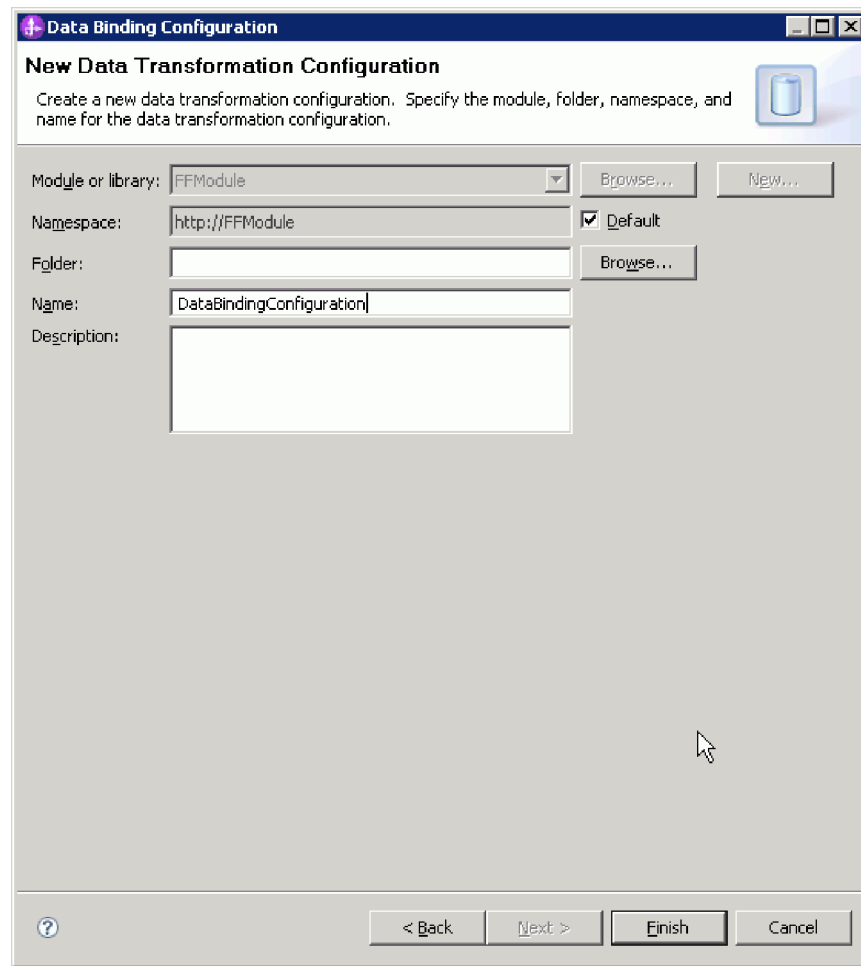


Figure 66. Naming the data binding configuration

5. Click **Next**.

### Results

A data binding is configured for use with the module.

### What to do next

Select the data handler configuration.

## Configuring data handlers

Data handlers perform the conversions between a business object and a native format.

### Before you begin

You must have created a data binding before you specify data handlers for the module. Also, you must have predefined business objects using WebSphere Integration Developer Business Object Editor. If you stop the wizard here to create business objects, you will need to start the wizard steps from the beginning.

**Note:** Data handlers can be configured prior to running the external service wizard using WebSphere Integration Developer. To do this, select **New** → **Resource configuration** in WebSphere Integration Developer and complete the data handler screens described in this documentation.

### About this task

To specify data handlers, follow this procedure.

### Procedure

1. In the Data Binding Properties window, click **New** and provide a name for the data handler configuration (example uses DHConfig). You click **New** the first time you set the data handler. To use this data handler later, click **Browse**.

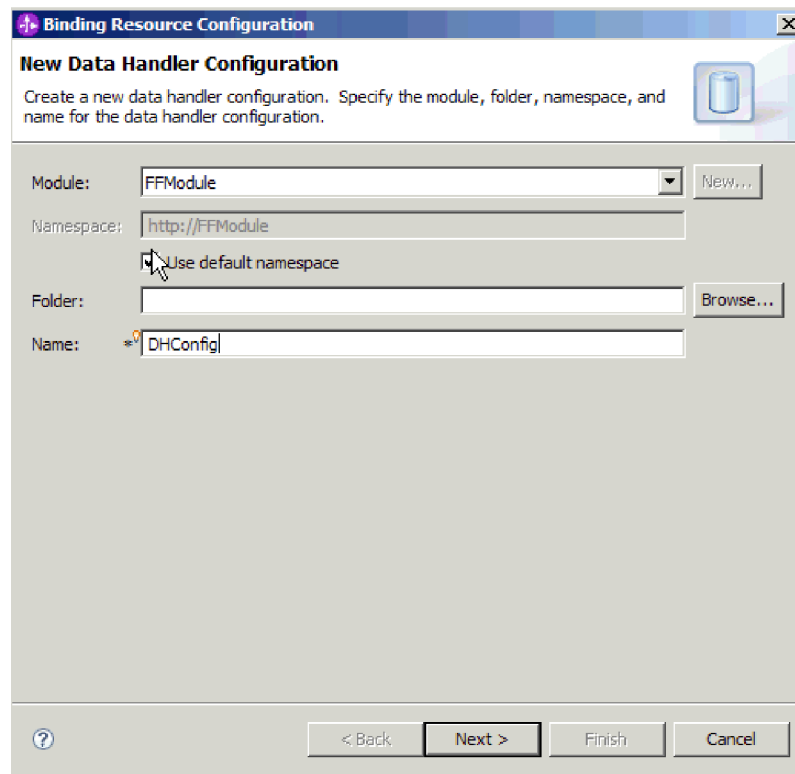


Figure 67. Specifying a name for the data handler configuration

2. Click **Next**.
3. Choose the class name for the data handler. In the Select a Configuration Type window, click **Browse** for Data handler class name. Select the **Show data handler classes** radio button. A list of available data handler classes is displayed. Select the data handler class (this example uses XMLDataHandler). Click **OK**.

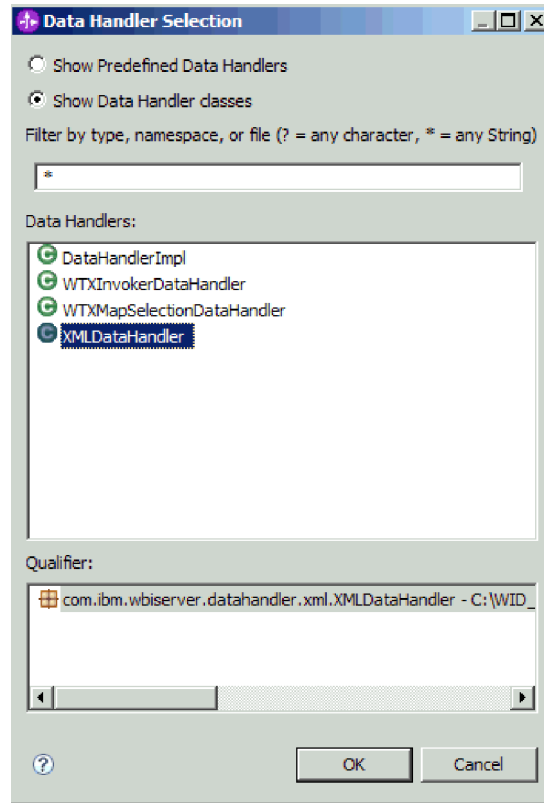


Figure 68. Selecting the data handler class

4. Click **Next**.
5. In the Specify properties window, specify the encoding (this example uses UTF-8).

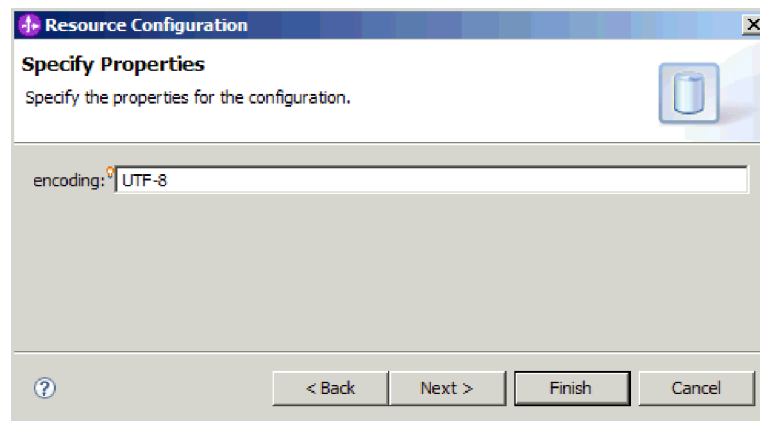


Figure 69. Specifying the encoding for the data handler configuration

6. Click **Finish**. The next screen shows the inbound operation that has been added, with the interaction specification properties.

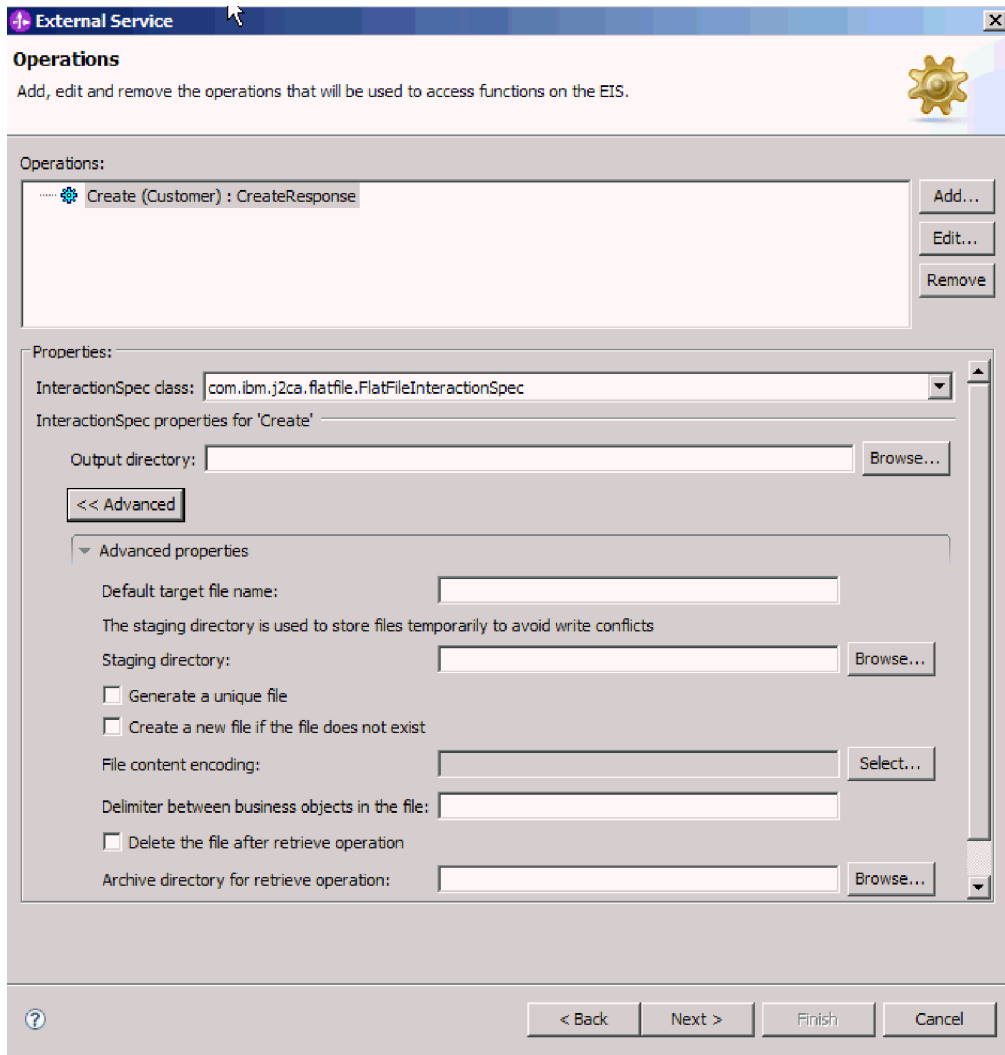


Figure 70. The inbound operation with InteractionSpec properties

7. Click **Finish**.

### Results

Data handlers are created.

### What to do next

Specify interaction specification properties and generate artifacts for the module.

## Setting deployment properties and generating the service

Use the external service wizard to set activation specification properties and generate artifacts for use with your module. Artifacts are the business objects, WSDL files, and import and export files that are created as part of the external service. While creating artifacts for the module, the adapter generates an export file. The export file contains the operation for the top-level business object.

### Before you begin

To set activation specification properties and generate artifacts for your module, you must have already configured data bindings and selected business objects.

### About this task

To set activation specification properties and generate artifacts, follow this procedure. For more information about activation specification properties, see the reference topic devoted to it in this documentation.

### Procedure

1. To set activation specification properties and generate artifacts, complete these steps:
  - a. In the Service Configuration Properties window, Click **Advanced**.
  - b. Type values for any fields you wish to set as defaults.
  - c. Click **Next**.
2. In the Operations window, click **Next**. On the Generate Service screen, supply a name for the interface. This is the name that will display in the WebSphere Integration Developer assembly diagram.

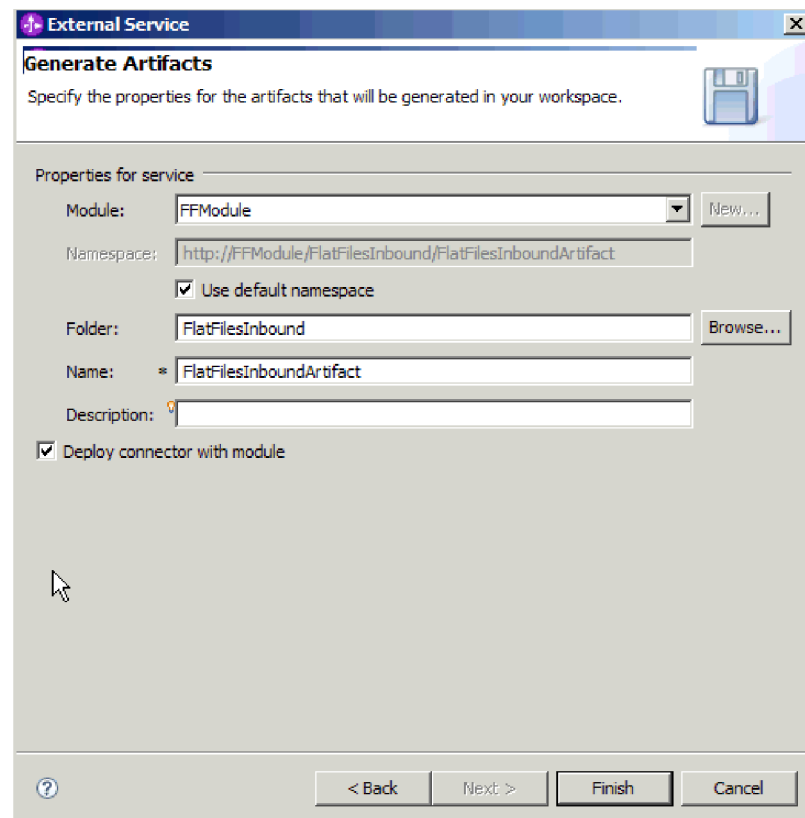


Figure 71. Naming the artifact

3. Click **Finish**.

### Results

The WebSphere Integration Developer generates the artifacts and an import. The inbound artifacts that are created are visible in the WebSphere Integration Developer Project Explorer under your module.

## What to do next

Deploy the module.

### **Related reference**

“Outbound configuration properties” on page 154

WebSphere Adapter for Flat Files has several categories of outbound connection configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter and managed connection factory properties after you deploy the module to WebSphere Process Server or WebSphere Enterprise Service Bus using WebSphere Integration Developer or the administrative console, but connection properties for the external service wizard cannot be changed after deployment.

“Inbound configuration properties” on page 171

WebSphere Adapter for Flat Files has several categories of inbound connection configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter and activation specification properties after you deploy the module using WebSphere Integration Developer or the administrative console, but connection properties for the external service wizard cannot be changed after deployment.

“Globalization” on page 193

WebSphere Adapter for Flat Files is a globalized application that can be used in multiple linguistic and cultural environments. Based on character set support and the locale of the host server, the adapter delivers message text in the appropriate language. The adapter supports bidirectional script data transformation between integration components.

---

## Chapter 5. Changing interaction specification properties using the assembly editor

To change interaction specification properties for your adapter module after generating the service, use the assembly editor in WebSphere Integration Developer.

### Before you begin

You must have used the external service wizard to generate a service for the adapter.

### About this task

You might want to change interaction specification properties after you have generated a service for the adapter. Interaction specification properties, which are optional, are set at the method level, for a specific operation on a specific business object. The values you specify will appear as defaults in all parent business objects generated by the external service wizard. You can change these properties before you export the EAR file. You cannot change these properties after you deploy the application.

To change the interaction specification properties, use the following procedure.

### Procedure

1. From the Business Integration perspective of WebSphere Integration Developer, expand the module name.
2. Expand **Assembly Diagram** and double-click the interface.
3. Click the interface in the assembly editor. (It shows the module properties if you don't do the extra click.)
4. Click the **Properties** tab. (You can also right-click the interface in the diagram and click **Show in Properties**.)
5. Under **Binding**, click **Method bindings**. The methods for the interface are displayed, one for each combination of business object and operation.
6. Select the method whose interaction specification property you want to change.
7. Click **Advanced** and change the property in the **Generic** tab. Repeat this step for each method whose interaction specification property you want to change.

### Results

The interaction specification properties associated with your adapter module are changed.

### What to do next

Deploy the module.

#### Related reference

"Interaction specification properties" on page 166

Interaction specification properties contain the outbound connection properties the adapter uses to interface with the file system. You configure these properties using the external service wizard. To change the interaction

specification properties after the application has been deployed, use the assembly editor in WebSphere Integration Developer.



---

## Chapter 6. Deploying the module

Deploy a module to place the files that make up your module and adapter into an operational environment for production or testing. In WebSphere Integration Developer, the integrated test environment features runtime support for WebSphere Process Server or WebSphere Enterprise Service Bus, or both, depending on the test environment profiles that you selected during installation.

---

### Deployment environments

There are test and production environments into which you can deploy modules and adapters.

In WebSphere Integration Developer, you can deploy your modules to one or more servers in the test environment. This is typically the most common practice for running and testing business integration modules. However, you can also export modules for server deployment on WebSphere Process Server or WebSphere Enterprise Service Bus as EAR files using the administrative console or command-line tools.

---

### Deploying the module for testing

In WebSphere Integration Developer, you can deploy a module that includes an embedded adapter to the test environment and work with server tools that enable you to perform such tasks as editing server configurations, starting and stopping servers and testing the module code for errors. The testing is generally performed on the interface operations of your components, which enables you to determine whether the components are correctly implemented and the references are correctly wired.

#### About this task

### Generating and wiring a target component for testing inbound processing

Before deploying to the test environment a module that includes an adapter for inbound processing, you must first generate and wire a target component. This target component serves as the *destination* to which the adapter sends events.

#### Before you begin

You must have generated an export module, using the external service wizard.

#### About this task

Generating and wiring a target component for inbound processing is required in a testing environment only. It is not necessary when deploying the adapter in a production environment.

The target component receives events. You *wire* the export to the target component (connecting the two components) using the assembly editor in WebSphere Integration Developer. The adapter uses the wire to pass event data (from the export to the target component).

## Procedure

1. Create the target component
  - a. From the Business Integration perspective of WebSphere Integration Developer, expand **Assembly Diagram** and double-click the export component. If you did not change the default value, the name of the export component is the name of your adapter + **InboundInterface**.

An interface specifies the operations that can be called and the data that is passed, such as input arguments, returned values, and exceptions. The **InboundInterface** contains the operations required by the adapter to support inbound processing and is created when you run the external service wizard.
  - b. Create a new component by expanding **Components**, selecting **Untyped Component**, and dragging the component to the Assembly Diagram. The cursor changes to the placement icon.
  - c. Click the component to have it displayed in the Assembly Diagram.
2. Wire the components.
  - a. Click and drag the export component to the new component. This draws a wire from the export component to the new component, as shown in the following figure:
  - b. Save the assembly diagram. Click **File** → **Save**
3. Generate an implementation for the new component.
  - a. Right-click on the new component and select **Generate implementation**.

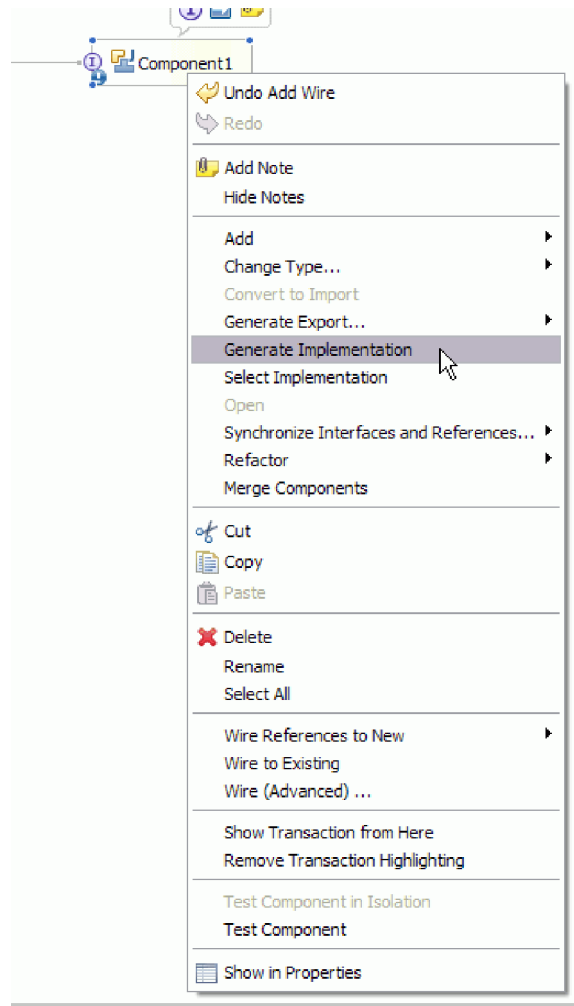


Figure 72. Generating a Java implementation

- b. Select **(default package)** and click **OK**. This creates an endpoint for the inbound module.  
The Java implementation is displayed in a separate tab.
- c. **Optional:** Add print statements to print the data object received at the endpoint for each of the endpoint methods.
- d. Click **File** → **Save** to save the changes.

#### What to do next

Continue deploying the module for testing.

## Adding the module to the server

In WebSphere Integration Developer, you can add modules to one or more servers in the test environment.

#### Before you begin

If the module you are testing uses an adapter to perform inbound processing, you need to generate and wire a *target component* to which the adapter will send events.

### About this task

In order to test your module and its use of the adapter, you need to add the module to the server.

### Procedure

1. *Conditional:* If there are no servers in the **Servers view**, add and define a new server by performing the following steps:
  - a. Place your cursor in the **Servers view**, right click and select **New** → **server**
  - b. From the Define a New Server window, select the server type.
  - c. Configure server's settings.
  - d. Click **Finish** to publish the server.
2. Add the module to the server
  - a. Switch to the servers view. In WebSphere Integration Developer, select **Windows** → **Show View** → **Servers**
  - a. Start the server. In the Servers tab in the lower-right pane of the WebSphere Integration Developer screen, right-click on the server, and then select **Start**.
3. When the server status is *Started*, right-click on the server, and select **Add and remove projects**.
4. In the Add and Remove Projects screen, select your project and click **Add**. The project moves from the **Available projects** list to the **Configured projects** list.
5. Click **Finish**. This deploys the module on the server.

The Console tab in the lower-right pane displays a log while the module is being added to the server.

### What to do next

Test the functionality of your module and the adapter.

## Testing the module for outbound processing using the test client

Test the assembled module and adapter for outbound processing using the WebSphere Integration Developer integration test client.

### Before you begin

You need to add the module to the server first.

### About this task

Testing a module is generally performed on the interface operations of your components, which enables you to determine whether the components are correctly implemented and the references are correctly wired.

### Procedure

1. Select the module you want to test, right-click on it, and select **Test** → **Test Module**.
2. For information on testing a module using the test client, see the *Testing modules and components* topic in the WebSphere Integration Developer information center.

### What to do next

If you are satisfied with the results of testing your module and adapter, you can deploy the module and adapter to the production environment.

---

## Deploying the module for production

Deploying a module created with the external service wizard to WebSphere Process Server or WebSphere Enterprise Service Bus in a production environment is a two-step process. First, you export the module in WebSphere Integration Developer as an enterprise archive (EAR) file. Second, you deploy the EAR file using the WebSphere Process Server or WebSphere Enterprise Service Bus administrative console.

### About this task

## Installing the RAR file (for modules using stand-alone adapters only)

If you chose not to embed the adapter with your module, but instead choose to make the adapter available to all deployed applications in the server instance, you will need to install the adapter in the form of a RAR file to the application server. A RAR file is a Java archive (JAR) file that is used to package a resource adapter for the Java 2 Connector (J2C) architecture.

### Before you begin

You must have set **Deploy connector project** to **On server for use by multiple adapters** in the Service Generation and Deployment Configuration window of the external service wizard.

### About this task

Installing the adapter in the form of a RAR file results in the adapter being available to all J2EE application components running in the server runtime.

### Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. In the Resource adapters page, click **Install RAR**.

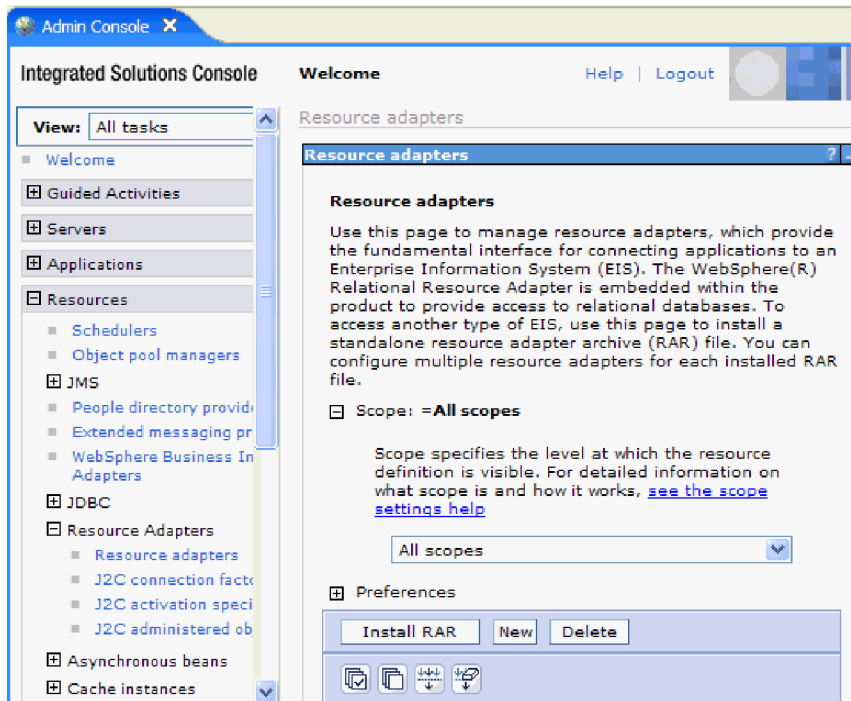


Figure 73. The Install RAR button on the Resource adapters page

4. In the Install RAR file page, click **Browse** and navigate to the RAR file for your adapter.

The RAR files are typically installed in the following path:  
`WID_installation_directory/ResourceAdapters/adapter_name/deploy/adapter.rar`

5. Click **Next**.
6. In the Resource adapters page, optionally change the name of the adapter and add a description.
7. Click **OK**.
8. Click **Save** in the **Messages** box at the top of the page.

### What to do next

The next step is to export the module as an EAR file that you can deploy on the server.

## Exporting the module as an EAR file

Using WebSphere Integration Developer, export your module as an EAR file. By creating an EAR file, you capture all of the contents of your module in a format that can be easily deployed to WebSphere Process Server or WebSphere Enterprise Service Bus.

### Before you begin

Before you can export a module as an EAR file, you must have created a module to communicate with your service. The module should be displayed in the WebSphere Integration Developer Business Integration perspective.

### About this task

To export the module as an EAR file, perform the following procedure.

#### Procedure

1. Right-click the module and select **Export**.
2. In the Select window, expand **Java EE**.
3. Select **EAR file** and click **Next**.
4. Optional: Select the correct EAR application. The EAR application is named after your module, but with “App” added to the end of the name.
5. **Browse** for the folder on the local file system where the EAR file will be placed.
6. Optionally, if you want to export the source files, select **Export source files**. This option is provided in case you want to export the source files in addition to the EAR file. Source files include files associated with Java components, data maps, and so on.
7. To overwrite an existing file, click **Overwrite an existing file**.
8. Click **Finish**.

#### Results

The contents of the module are exported as an EAR file.

Install the module in the administrative console. This deploys the module to WebSphere Process Server or WebSphere Enterprise Service Bus.

## Installing the EAR file

Installing the EAR file is the last step of the deployment process. When you install the EAR file on the server and run it, the adapter, which is embedded as part of the EAR file, runs as part of the installed application.

#### Before you begin

You must have exported your module as an EAR file before you can install it on WebSphere Process Server or WebSphere Enterprise Service Bus.

#### About this task

To install the EAR file, perform the following procedure. For more information on clustering adapter module applications, see the <http://www.ibm.com/software/webservers/appserv/was/library/>.

#### Procedure

1. Open the WebSphere Process Server or WebSphere Enterprise Service Bus administrative console by right-clicking your server instance and selecting **Run administrative console**.
2. In the administrative console window, click **Applications** → **Install New Applications**.



Figure 74. Preparing for the application installation window

3. Click **Browse** to locate your EAR file and click **Next**. The EAR file name is the name of the module followed by "App."
4. Optional: If you are deploying to a clustered environment, complete the following steps.
  - a. On the **Step 2: Mapping modules to servers** window, select the module.
  - b. Select the name of the server cluster.
  - c. Click **Apply**.
5. Click **Next** to open the Summary. Verify that all settings are correct and click **Finish**.
6. Optional: If you are using an authentication alias, complete the following steps:
  - a. Expand **Security** and select **Business Integration Authentication Aliases**.
  - b. Select the authentication alias that you want to configure. You must have administrator or operator authority to make changes to authentication alias configurations.
  - c. Optional: If it is not already filled in, type the **User name**.
  - d. If it is not already filled in, type the **Password**.
  - e. If it is not already filled in, type the password again in the **Confirm Password** field.
  - f. Click **OK**.

## Results

The project is now deployed and the Enterprise Applications window is displayed.

## What to do next

If you want to set or reset any properties or you would like to cluster adapter project applications, make those changes using the administrative console before configuring troubleshooting tools.



---

## Chapter 7. Administering the adapter module

When you are running the adapter in a stand-alone deployment, use the administrative console of the server to start, stop, monitor, and troubleshoot the adapter module. In an application that uses an embedded adapter, the adapter module starts or stops when the application is started or stopped.

---

### Changing configuration properties for embedded adapters

To change configuration properties after you deploy the adapter as part of a module, you use the administrative console of the runtime environment. You can update resource adapter properties (used for general adapter operation), managed connection factory properties (used for outbound processing), and activation specification properties (used for inbound processing).

#### Related reference

“Inbound configuration properties” on page 171

WebSphere Adapter for Flat Files has several categories of inbound connection configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter and activation specification properties after you deploy the module using WebSphere Integration Developer or the administrative console, but connection properties for the external service wizard cannot be changed after deployment.

“Outbound configuration properties” on page 154

WebSphere Adapter for Flat Files has several categories of outbound connection configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter and managed connection factory properties after you deploy the module to WebSphere Process Server or WebSphere Enterprise Service Bus using WebSphere Integration Developer or the administrative console, but connection properties for the external service wizard cannot be changed after deployment.

### Setting resource adapter properties for embedded adapters

To set resource adapter properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the property you want to configure and then change or set the value.

#### Before you begin

Your adapter module must be deployed on WebSphere Process Server or WebSphere Enterprise Service Bus.

#### About this task

Custom properties are default configuration properties shared by all WebSphere adapters.

To configure properties using the administrative console, use the following procedure.

#### Procedure

1. Start the administrative console.

2. Under **Applications**, select **Enterprise Applications**.
3. From the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change. The Configuration page is displayed.

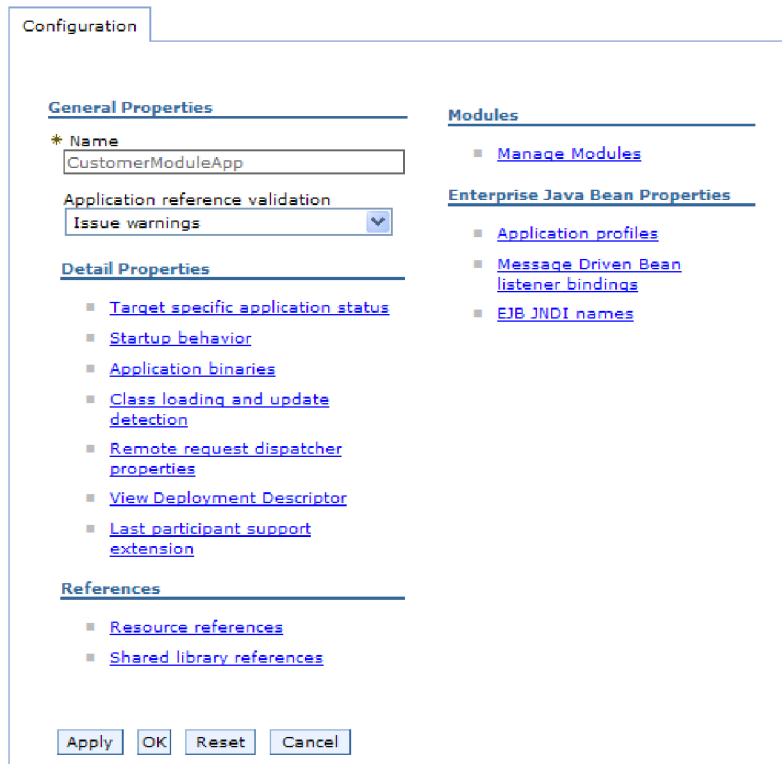


Figure 75. The Manage Modules selection in the Configuration tab

4. Under **Modules**, click **Manage Modules**.
5. Click **IBM WebSphere Adapter for Flat Files**.
6. From the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **Custom properties**.
8. For each property you want to change, perform the following steps.

**Note:** See “Resource adapter properties” on page 164 for more information about these properties.

- a. Click the name of the property. The **Configuration** page for the selected property is displayed. For example, if you click the **logNumberOfFiles** property, you see the page shown in the figure below.

The image shows a configuration dialog box titled "Configuration" with a tab labeled "Configuration". Under the "General Properties" section, there are several fields:
 

- \* Scope:** A text box containing "widNode".
- Required:** An unchecked checkbox.
- Name:** A text box containing "logNumberOfFiles".
- Value:** A text box containing "1".
- Description:** An empty text area with a vertical scrollbar.
- Type:** A dropdown menu showing "java.lang.String".

 At the bottom of the dialog are four buttons: "Apply", "OK", "Reset", and "Cancel".

Figure 76. The Configuration tab for the logNumberOfFiles property

- b. Change the contents of the **Value** field or type a value, if the field is empty. You can change the number in the **Value** field and add a description of the property.
  - c. Click **OK**.
9. Click the **Save** link in the **Messages** box at the top of the window.

### Results

The resource adapter properties associated with your adapter module are changed.

#### Related reference

“Resource adapter properties” on page 164

The resource adapter properties control the general operation of the adapter, such as specifying the namespace for business objects. You set the resource adapter properties using the external service wizard when you configure the adapter. After deploying the adapter, use the administrative console to change these properties.

## Setting managed (J2C) connection factory properties for embedded adapters

To set managed connection factory properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the property you want to configure and then change or set the value.

### Before you begin

Your adapter module must be deployed on WebSphere Process Server or WebSphere Enterprise Service Bus.

### About this task

You use managed connection factory properties to configure the target local file system instance.

**Note:** In the administrative console, the properties are referred to as "J2C connection factory properties."

To configure properties using the administrative console, use the following procedure.

### Procedure

1. Start the administrative console.
2. Under **Applications**, select **Enterprise Applications**.
3. In the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change.
4. Under **Modules**, click **Manage Modules**.

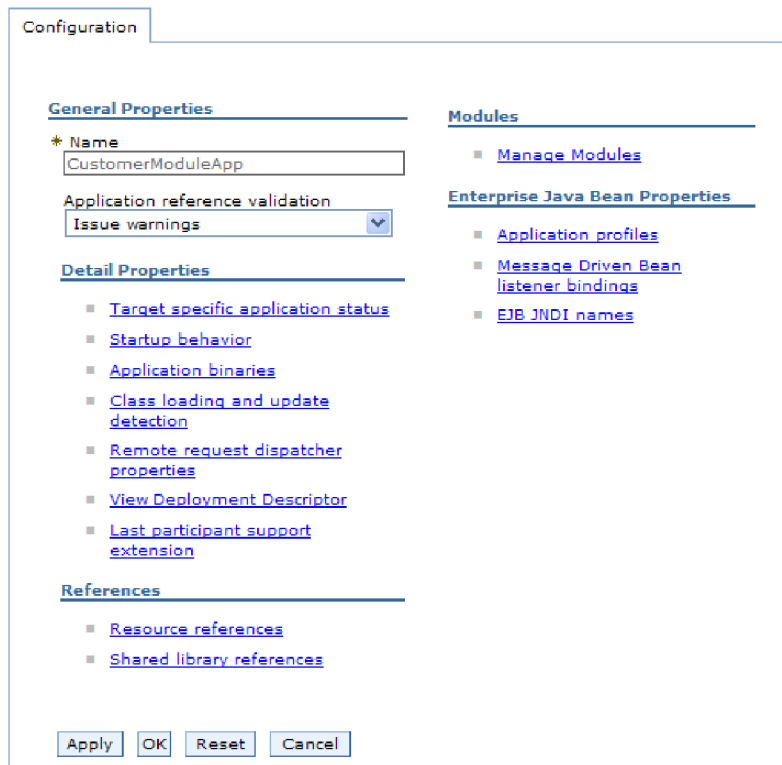


Figure 77. The Manage Modules selection in the Configuration tab

5. Click **IBM WebSphere Adapter for Flat Files**.
6. In the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **J2C connection factories**.
8. Click the name of the connection factory associated with your adapter module.
9. In the **Additional Properties** list, click **Custom properties**.

Custom properties are those J2C connection factory properties that are unique to Adapter for Flat Files. Connection pool and advanced connection factory properties are properties you configure if you are developing your own adapter.

10. For each property you want to change, perform the following steps.

**Note:** See “Managed connection factory properties” on page 159 for more information about these properties.

- a. Click the name of the property.
  - b. Change the contents of the **Value** field or type a value, if the field is empty.
  - c. Click **OK**.
11. Click the **Save** link in the **Messages** box at the top of the window.

## Results

The managed connection factory properties associated with your adapter module are changed.

### Related reference

“Managed connection factory properties” on page 159

Managed connection factory properties specify information the adapter needs at run time for outbound communication with the local file system.

## Setting activation specification properties for embedded adapters

To set activation specification properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the message endpoint property you want to configure, and then change or set the value.

### Before you begin

Your adapter module must be deployed on WebSphere Process Server or WebSphere Enterprise Service Bus.

### About this task

You use activation specification properties to configure the endpoint for inbound processing.

To configure properties using the administrative console, use the following procedure.

### Procedure

1. Start the administrative console.
2. Under **Applications**, select **Enterprise Applications**.
3. From the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change.
4. Under **Modules**, click **Manage Modules**.

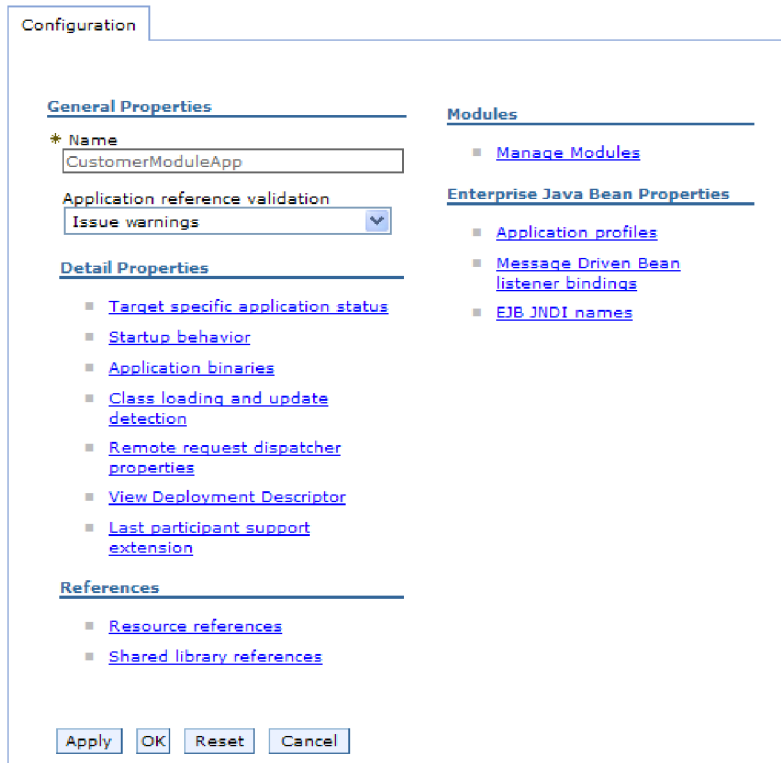


Figure 78. The Manage Modules selection in the Configuration tab

5. Click **IBM WebSphere Adapter for Flat Files**.
6. From the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **J2C activation specifications**.
8. Click the name of the activation specification associated with the adapter module.
9. From the **Additional Properties** list, click **J2C activation specification custom properties**.
10. For each property you want to change, perform the following steps.

**Note:** See “Activation specification properties” on page 176 for more information about these properties.

- a. Click the name of the property.
  - b. Change the contents of the **Value** field or type a value, if the field is empty.
  - c. Click **OK**.
11. Click the **Save** link in the **Messages** box at the top of the window.

## Results

The activation specification properties associated with your adapter module are changed.

### Related reference

“Activation specification properties” on page 176

Activation specification properties hold the inbound event processing

configuration information for an export. You set activation specification properties through either the external service wizard or the administrative console.

---

## Changing configuration properties for stand-alone adapters

To set configuration properties after you install a stand-alone adapter, you use the administrative console of the runtime environment. You provide general information about the adapter and then set resource adapter properties (which are used for general adapter operation). If the adapter will be used for outbound operations, you create a connection factory and then set properties for it. If the adapter will be used for inbound operations, you create an activation specification and then set properties for it.

### Setting resource adapter properties for stand-alone adapters

To set resource adapter properties for your stand-alone adapter after it has been installed on WebSphere Process Server or WebSphere Enterprise Service Bus, use the administrative console. You select the name of the property you want to configure and then change or set the value.

#### Before you begin

Your adapter must be installed on WebSphere Process Server or WebSphere Enterprise Service Bus.

#### About this task

Custom properties are default configuration properties shared by all WebSphere adapters.

To configure properties using the administrative console, use the following procedure.

#### Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. In the Resource adapters page, click **IBM WebSphere Adapter for Flat Files**.
4. In the **Additional Properties** list, click **Custom properties**.
5. For each property you want to change, perform the following steps.

**Note:** See “Resource adapter properties” on page 164 for more information about these properties.

- a. Click the name of the property.
- b. Change the contents of the **Value** field or type a value, if the field is empty.  
For example, if you click **logNumberOfFiles**, you see the following page:

The image shows a configuration dialog box titled "Configuration" with a tab labeled "Configuration". Under the "General Properties" section, there are several fields:
 

- \* Scope:** A text box containing "widNode".
- Required:** An unchecked checkbox.
- Name:** A text box containing "logNumberOfFiles".
- Value:** A text box containing "1".
- Description:** A large empty text area with vertical scroll bars.
- Type:** A dropdown menu showing "java.lang.String".

 At the bottom of the dialog are four buttons: "Apply", "OK", "Reset", and "Cancel".

Figure 79. The Configuration tab for the logNumberOfFiles property

You can change the number in the **Value** field and add a description of the property.

- c. Click **OK**.
6. Click **Save** in the **Messages** box at the top of the page.

### Results

The resource adapter properties associated with your adapter are changed.

#### Related reference

“Resource adapter properties” on page 164

The resource adapter properties control the general operation of the adapter, such as specifying the namespace for business objects. You set the resource adapter properties using the external service wizard when you configure the adapter. After deploying the adapter, use the administrative console to change these properties.

## Setting managed (J2C) connection factory properties for stand-alone adapters

To set managed connection factory properties for your stand-alone adapter after it has been installed on WebSphere Process Server or WebSphere Enterprise Service Bus, use the administrative console. You select the name of the property you want to configure and then change or set the value.

### Before you begin

Your adapter must be installed on WebSphere Process Server or WebSphere Enterprise Service Bus.

### About this task



You use managed connection factory properties to configure the target local file system instance.

**Note:** In the administrative console, the properties are referred to as "J2C connection factory properties."

To configure properties using the administrative console, use the following procedure.

### Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. In the Resource adapters page, click **IBM WebSphere Adapter for Flat Files**.
4. In the **Additional Properties** list, click **J2C connection factories**.
5. If you are going to use an existing connection factory, skip ahead to select from the list of existing connection factories.

**Note:** If you selected **Use predefined connection properties** when you used the external service wizard to configure the adapter module, you do not need to create a connection factory.

If you are creating a connection factory, perform the following steps:

- a. Click **New**.
- b. In the **General Properties** section of the **Configuration** tab, type a name for the connection factory. For example, you could type AdapterCF.
- c. Type a value for **JNDI name**. For example, you could type com/eis/AdapterCF.
- d. Select an authentication alias from the **Component-managed authentication alias** list.
- e. Click **OK**.
- f. Click **Save** in the **Messages** box at the top of the page.

The newly created connection factory is displayed.

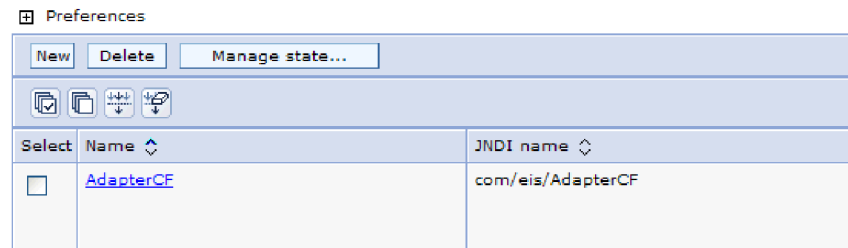


Figure 80. User defined connection factories for use with the resource adapter

6. In the list of connection factories, click the one you want to use.
7. In the **Additional Properties** list, click **Custom properties**.  
Custom properties are those J2C connection factory properties that are unique to Adapter for Flat Files. Connection pool and advanced connection factory properties are properties you configure if you are developing your own adapter.
8. For each property you want to change, perform the following steps.

**Note:** See “Managed connection factory properties” on page 159 for more information about these properties.

- a. Click the name of the property.
  - b. Change the contents of the **Value** field or type a value, if the field is empty.
  - c. Click **OK**.
9. After you have finished setting properties, click **Apply**.
  10. Click **Save** in the **Messages** box at the top of the window.

## Results

The managed connection factory properties associated with your adapter are set.

### Related reference

“Managed connection factory properties” on page 159

Managed connection factory properties specify information the adapter needs at run time for outbound communication with the local file system.

## Setting activation specification properties for stand-alone adapters

To set activation specification properties for your stand-alone adapter after it has been installed on WebSphere Process Server or WebSphere Enterprise Service Bus, use the administrative console. You select the name of the message endpoint property you want to configure, and then change or set the value.

### Before you begin

Your adapter must be installed on WebSphere Process Server or WebSphere Enterprise Service Bus.

### About this task

You use activation specification properties to configure the endpoint for inbound processing.

To configure properties using the administrative console, use the following procedure.

### Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. In the Resource adapters page, click **IBM WebSphere Adapter for Flat Files**.
4. In the **Additional Properties** list, click **J2C activation specifications**.
5. If you are going to use an existing activation specification, skip ahead to select from an existing list of activation specifications.

**Note:** If you selected **Use predefined connection properties** when you used the external service wizard to configure the adapter module, you do not need to create an activation specification.

If you are creating an activation specification, perform the following steps:

- a. Click **New**.

- b. In the **General Properties** section of the **Configuration** tab, type a name for the activation specification. For example, you could type AdapterAS.
- c. Type a value for **JNDI name**. For example, you could type com/eis/AdapterAS.
- d. Select an authentication alias from the **Authentication alias** list.
- e. Select a message listener type.
- f. Click **OK**.
- g. Click **Save** in the **Messages** box at the top of the page.  
The newly created activation specification is displayed.
6. In the list of activation specifications, click the one you want to use.
7. In the Additional Properties list, click **J2C activation specification custom properties**.
8. For each property you want to set, perform the following steps.

**Note:** See “Activation specification properties” on page 176 for more information about these properties.

- a. Click the name of the property.
- b. Change the contents of the **Value** field or type a value, if the field is empty.
- c. Click **OK**.
9. After you have finished setting properties, click **Apply**.
10. Click **Save** in the **Messages** box at the top of the page.

## Results

The activation specification properties associated with your adapter are set.

### Related reference

“Activation specification properties” on page 176

Activation specification properties hold the inbound event processing configuration information for an export. You set activation specification properties through either the external service wizard or the administrative console.

---

## Starting the application that uses the adapter

Use the administrative console of the server to start an application that uses the adapter. By default, the application starts automatically when the server starts.

### About this task

Use this procedure to start the application, whether it is using an embedded or a stand-alone adapter. For an application that uses an embedded adapter, the adapter starts when the application starts. For an application that uses a stand-alone adapter, the adapter starts when the application server starts.

### Procedure

1. On the administrative console, click **Applications** → **Enterprise Applications**.

**Note:** The administrative console is labeled “Integrated Solutions Console”.

2. Select the check box of the application that you want to start. The application name is the name of the EAR file you installed, without the .EAR file extension.

3. Click **Start**.

### Results

The status of the application changes to Started, and a message stating that the application has started displays at the top of the administrative console.

---

## Stopping the application that uses the adapter

Use the administrative console of the server to stop an application that uses the adapter. By default, the application stops automatically when the server stops.

### About this task

Use this procedure to stop the application, whether it is using an embedded or a stand-alone adapter. For an application with an embedded adapter, the adapter stops when the application stops. For an application that uses a stand-alone adapter, the adapter stops when the application server stops.

### Procedure

1. On the administrative console, click **Applications** → **Enterprise Applications**.

**Note:** The administrative console is labeled “Integrated Solutions Console”.

2. Select the check box of the application that you want to stop. The application name is the name of the EAR file you installed, without the .EAR file extension.
3. Click **Stop**.

### Results

The status of the application changes to Stopped, and a message stating that the application has stopped displays at the top of the administrative console.

---

## Monitoring performance using Performance Monitoring Infrastructure

Performance Monitoring Infrastructure (PMI) is a feature of the administrative console that allows you to dynamically monitor the performance of components in the production environment, including the adapter for Flat Files. PMI collects adapter performance data, such as average response time and total number of requests, from various components in the server and organizes the data into a tree structure. You can view the data through the Tivoli® Performance Viewer, a graphical monitoring tool that is integrated with the administrative console in WebSphere Process Server.

### About this task

You can monitor the performance of your adapter by having PMI collect data at the following points:

- At outbound processing to monitor outbound requests
- At inbound event retrieval to monitor the retrieval of an event from the event table
- At inbound event delivery to monitor the delivery of an event to the endpoint or endpoints

Before you can enable and configure PMI for your adapter, you must first set the level of tracing detail and run some events from which to gather performance data.

To learn more about how PMI can help you monitor and improve the overall performance of your adapter environment, search for PMI on the WebSphere Application Server web site: <http://www.ibm.com/software/webservers/appserv/was/library/>.

## Configuring Performance Monitoring Infrastructure

You can configure Performance Monitoring Infrastructure (PMI) to gather adapter performance data, such as average response time and total number of requests. After you configure PMI for your adapter, you can monitor the adapter performance using Tivoli Performance viewer.

### Before you begin

Before you can configure PMI for your adapter, you must first set the level of tracing detail and run some events from which to gather performance data.

1. To enable tracing and to receive event data, the trace level must be set to either fine, finer, finest, or all. After \*=info, add a colon and a string, for example:

```
*=info: WBILocationMonitor.CEI.ResourceAdapter.  
*=finest: WBILocationMonitor.LOG.ResourceAdapter.*=finest:
```

For instructions on setting the trace level, refer to “Enabling tracing with the Common Event Infrastructure (CEI)” on page 138.

2. Generate at least one outbound request or inbound event to produce performance data that you can configure.

### Procedure

1. Enable PMI for your adapter.
  - a. In the administrative console, expand **Monitoring and Tuning**, and then select **Performance Monitoring Infrastructure (PMI)**.
  - b. From the list of servers, click the name of your server.
  - c. Select the Configuration tab, and then select the **Enable Performance Monitoring (PMI)** check box.
  - d. Select **Custom** to selectively enable or disable statistics.

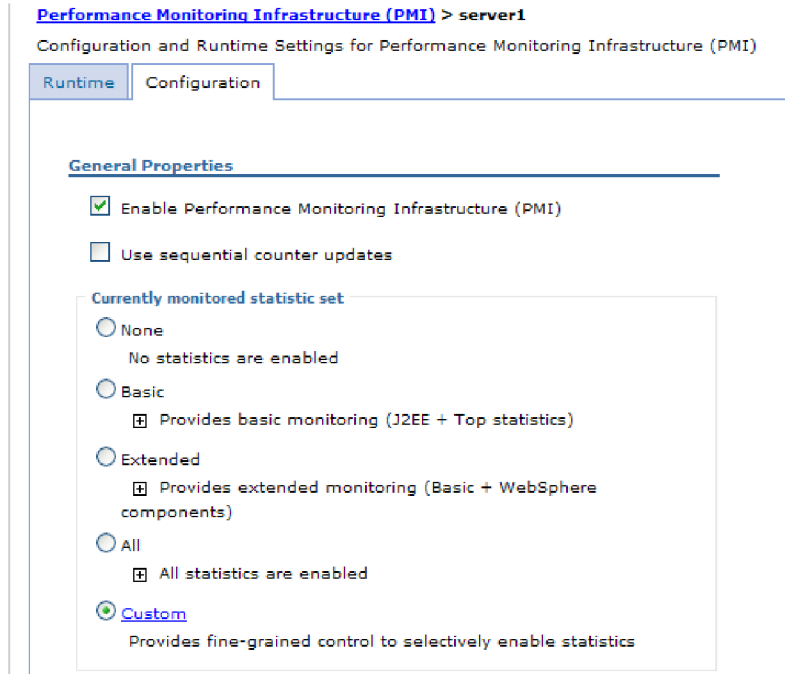


Figure 81. Enabling Performance Monitoring Infrastructure

- e. Click **Apply** or **OK**.
  - f. Click **Save**. PMI is now enabled.
2. Configure PMI for your adapter.
    - a. In the administrative console, expand **Monitoring and Tuning**, and then select **Performance Monitoring Infrastructure (PMI)**.
    - b. From the list of servers, click the name of your server.
    - c. Select **Custom**.
    - d. Select the **Runtime** tab. The following figure shows the Runtime tab.

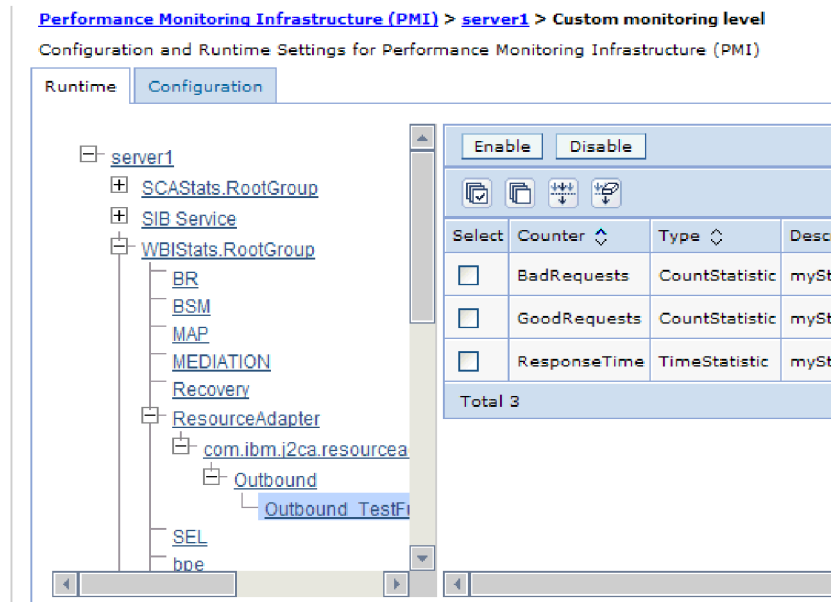


Figure 82. Runtime tab used for configuring PMI

- e. Click **WBISStats.RootGroup**. This is a PMI sub module for data collected in the root group. This example uses the name WBISStats for the root group.
- f. Click **ResourceAdapter**. This is a sub module for the data collected for the JCA adapters.
- g. Click the name of your adapter, and select the processes you want to monitor.
- h. In the right pane, select the check boxes for the statistics you want to gather, and then click **Enable**.

### Results

PMI is configured for your adapter.

### What to do next

Now you can view the performance statistics for your adapter.

## Viewing performance statistics

You can view adapter performance data through the graphical monitoring tool, Tivoli Performance Viewer. Tivoli Performance Viewer is integrated with the administrative console in WebSphere Process Server.

### Before you begin

Configure Performance Monitoring Infrastructure for your adapter.

### Procedure

1. In the administrative console, expand **Monitoring and Tuning**, expand **Performance Viewer**, then select **Current Activity**.
2. In the list of servers, click the name of your server.
3. Under your server name, expand **Performance Modules**.

4. Click **WBIStatsRootGroup**.
5. Click **ResourceAdapter** and the name of your adapter module.
6. If there is more than one process, select the check boxes for the processes whose statistics you want to view.

## Results

The statistics are displayed in the right panel. You can click **View Graph** to view a graph of the data, or **View Table** to see the statistics in a table format. The following figure shows adapter performance statistics as a graph.

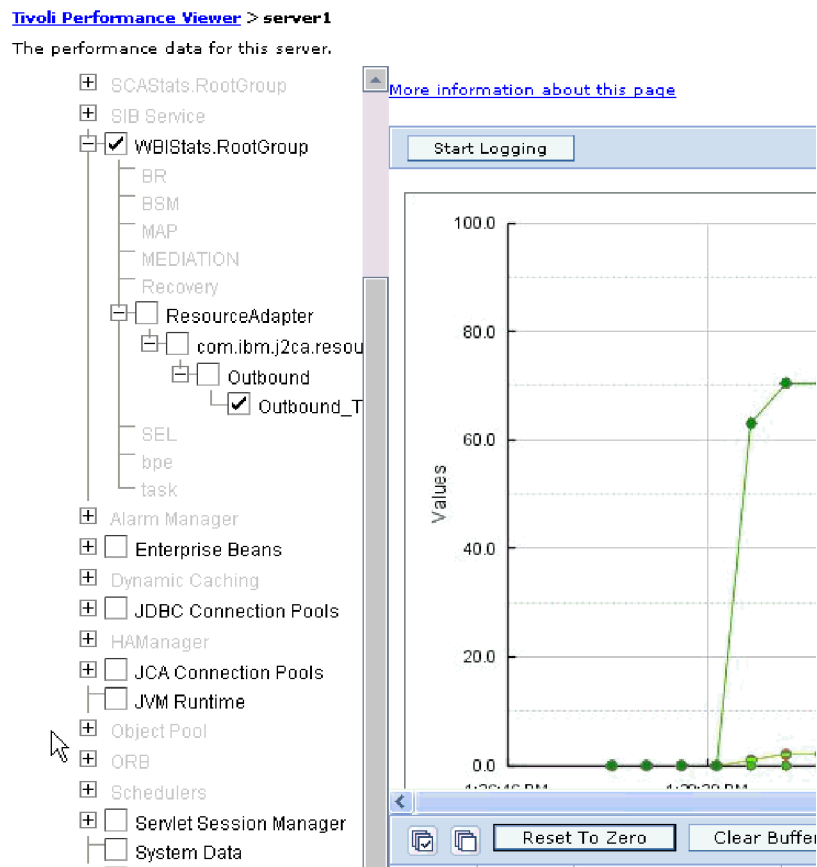


Figure 83. Adapter performance statistics, using graph view

## Enabling tracing with the Common Event Infrastructure (CEI)

The adapter can use the Common Event Infrastructure, a component embedded in the server, to report data about critical business events such as the starting or stopping of a poll cycle. Event data can be written to a database or a trace log file depending on configuration settings.

### About this task

#### Procedure

1. In the administrative console, click **Troubleshooting**.
2. Click **Logs and Trace**.
3. In the list of servers, click the name of your server.



4. In the **Change Log Detail Levels** box, click the name of the CEI database (for example, `WBIEventMonitor.CEI.ResourceAdapter.*`) or the trace log file (for example, `WBIEventMonitor.LOG.ResourceAdapter.*`) to which you want the adapter to write event data.
5. Select the level of detail about business events that you want the adapter to write to the database or trace log file, and (optionally) adjust the granularity of detail associated with messages and traces.
  - **No Logging.** Turns off event logging.
  - **Messages Only.** The adapter reports an event.
  - **All Messages and Traces.** The adapter reports details about an event.
  - **Message and Trace Levels.** Settings for controlling the degree of detail the adapter reports about the business object payload associated with an event. If you want to adjust the detail level, choose one of the following:
    - Fine.** The adapter reports the event but none of the business object payload.
    - Finer.** The adapter reports the event and the business object payload description.
    - Finest.** The adapter reports the event and the entire business object payload.
6. Click **OK**.

### **Results**

Event logging is enabled. You can view CEI entries in the trace log file or by using the Common Base Event Browser within the administrative console.



---

## Chapter 8. Troubleshooting and support

Common troubleshooting techniques and self-help information help you identify and solve problems quickly.

### Related reference

“Adapter messages” on page 198

View the messages issued by WebSphere Adapter for Flat Files at the following location.

---

## Support for the Log and Trace Analyzer

The adapter creates log and trace files that can be viewed with the Log and Trace Analyzer.

The Log and Trace Analyzer can filter log and trace files to isolate the messages and trace information for the adapter. It can also highlight the adapter’s messages and trace information in the log viewer.

The adapter’s component ID for filtering and highlighting is a string composed of the characters FFRA plus the value of the adapter ID property. For example, if the adapter ID property is set to 001, the component ID is FFRA001.

If you run multiple instances of the same adapter, make sure that the first nine characters of the adapter ID property are unique for each instance so that you can correlate log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate log and trace information to a particular instance of an adapter. To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for Flat Files to 001 and 002. The component IDs for those instances, FFRA001 and FFRA002, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. For example, suppose you set the adapter ID properties of two instances to Instance01 and Instance02. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to FFRAInstance0.

For outbound processing, the adapter ID property is located in both the resource adapter and managed connection factory property groups. If you update the adapter ID property after using the external service wizard to configure the adapter for outbound processing, be sure to set the resource adapter and managed connection factory properties consistently, to prevent inconsistent marking of the log and trace entries. For inbound processing, the adapter ID property is located only in the resource adapter properties, so this consideration does not apply.

For more information about the adapter ID property, see “Adapter ID (AdapterID)” on page 165. For more information about the Log and Trace Analyzer, see <http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/org.eclipse.hyades.log.ui.doc.user/concepts/cltaviews.htm>.

---

## Configuring logging and tracing

Configure logging and tracing to suit your requirements. Enable logging for the adapter to control the status of event processing. Change the adapter log and trace file names to separate them from other log and trace files.

### About this task

## Configuring logging properties

Use the administrative console to enable logging and to set the output properties for a log, including the location, level of detail, and output format of the log.

### About this task

Before the adapters can log monitored events, you must specify the service component event points that you want to monitor, what level of detail you require for each event, and format of the output used to publish the events to the logs. Use the administrative console to perform the following tasks:

- Enable or disable a particular event log
- Specify the level of detail in a log
- Specify where log files are stored and how many log files are kept
- Specify the format for log output

If you set the output for log analyzer format, you can open trace output using the Log Analyzer tool, which is an application included with your process server. This is useful if you are trying to correlate traces from two different server processes, because it allows you to use the merge capability of the Log Analyzer.

For more information about monitoring on a process server, including service components and event points, see the documentation for your process server.

You can change the log configuration statically or dynamically. Static configuration takes effect when you start or restart the application server. Dynamic, or runtime, configuration changes apply immediately.

When a log is created, the detail level for that log is set from the configuration data. If no configuration data is available for a particular log name, the level for that log is obtained from the parent of the log. If no configuration data exists for the parent log, the parent of that log is checked, and so on up the tree, until a log with a non-null level value is found. When you change the level of a log, the change is propagated to the children of the log, which recursively propagate the change to their children, as necessary.

To enable logging and set the output properties for a log, use the following procedure.

### Procedure

1. In the navigation pane of the administrative console, click **Servers** → **Application Servers**.
2. Click the name of the server that you want to work with.
3. Under **Troubleshooting**, click **Logs and trace**.
4. Click **Change Log Detail Levels**.
5. Specify when you want the change to take effect:

- For a static change to the configuration, click the **Configuration** tab.
  - For a dynamic change to the configuration, click the **Runtime** tab.
6. Click the names of the packages whose logging level you want to modify. The package names for WebSphere Adapters start with **com.ibm.j2ca.\***:
    - For the adapter base component, select **com.ibm.j2ca.base.\***.
    - For the adapter base component and all deployed adapters, select **com.ibm.j2ca.\***.
    - For the Adapter for Flat Files only, select the **com.ibm.j2ca.flatfile.\*** package.
  7. Select the logging level.

Logging Level	Description
Fatal	The task cannot continue or the component cannot function.
Severe	The task cannot continue, but the component can still function. This logging level also includes conditions that indicate an impending fatal error, that is, situations that strongly suggest that resources are on the verge of being depleted.
Warning	A potential error has occurred or a severe error is impending. This logging level also includes conditions that indicate a progressive failure, for example, the potential leaking of resources.
Audit	A significant event has occurred that affects the server state or resources.
Info	The task is running. This logging level includes general information outlining the overall progress of a task.
Config	The status of a configuration is reported or a configuration change has occurred.
Detail	The subtask is running. This logging level includes general information detailing the progress of a subtask.

8. Click **Apply**.
9. Click **OK**.
10. To have static configuration changes take effect, stop and then restart the process server.

### Results

Log entries from this point forward contain the specified level of information for the selected adapter components.

## Changing the log and trace file names

To keep the adapter log and trace information separate from other processes, use the administrative console to change the file names. By default, log and trace information for all processes and applications on a process server is written to the SystemOut.log and trace.log files, respectively.

### Before you begin

You can change the log and trace file names at any time after the adapter module has been deployed to an application server.

### About this task

You can change the log and trace file names statically or dynamically. Static changes take effect when you start or restart the application server. Dynamic or run time changes apply immediately.

Log and trace files are in the *install\_root/profiles/profile\_name/logs/server\_name* folder.

To set or change the log and trace file names, use the following procedure.

#### Procedure

1. In the navigation pane of the administrative console, select **Applications > Enterprise Applications**.
2. In the Enterprise Applications list, click the name of the adapter application. This is the name of the EAR file for the adapter, but without the .ear file extension. For example, if the EAR file is named Accounting\_OutboundApp.ear, then click **Accounting\_OutboundApp**.
3. In the Configuration tab, in the Modules list, click **Manage Modules**.
4. In the list of modules, click IBM WebSphere Adapter for Flat Files.
5. In the Configuration tab, under Additional Properties, click **Resource Adapter**.
6. In the Configuration tab, under Additional Properties, click **Custom properties**.
7. In the Custom Properties table, change the file names.
  - a. Click either **logFilename** to change the name of the log file or **traceFilename** to change the name of the trace file.
  - b. In the Configuration tab, type the new name in the **Value** field. By default, the log file is called SystemOut.log and the trace file is called trace.log.
  - c. Click **Apply** or **OK**. Your changes are saved on your local machine.
  - d. To save your changes to the master configuration on the server, use one of the following procedures:
    - **Static change:** Stop and restart the server. This method allows you to make changes, but those changes do not take effect until you stop and start the server.
    - **Dynamic change:** Click the **Save** link in the Messages box above the Custom properties table. Click **Save** again when prompted. This method allows you to make changes that take effect right away.

---

## First-failure data capture (FFDC) support

The adapter supports first-failure data capture (FFDC), which provides persistent records of failures and significant software incidents that occur during run time in WebSphere Process Server or WebSphere Enterprise Service Bus.

The FFDC feature runs in the background and collects events and errors that occur at run time. The feature provides a means for associating failures to one another, allowing software to link the effects of a failure to their causes, and thereby facilitate the quick location of the root cause of a failure. The data that is captured can be used to identify exception processing that occurred during the adapter run time.

When a problem occurs, the adapter writes exception messages and context data to a log file, which is located in the *install\_root/profiles/profile/logs/ffdc* directory.

For more information about first-failure data capture (FFDC), see the WebSphere Process Server or WebSphere Enterprise Service Bus documentation.

---

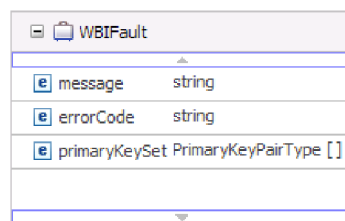
## Business faults

The adapter supports business faults, which are exceptions that are anticipated and declared in the outbound service description, or import. Business faults occur at predictable points in a business process, and are caused by a business rule violation or a constraint violation.

Although WebSphere Process Server and WebSphere Enterprise Service Bus support other types of faults, the adapter generates only business faults, which are called simply *faults* in this documentation. Not all exceptions become faults. Faults are generated for errors that are actionable, that is, errors that can have a recovery action that does not require the termination of the application. For example, the adapter generates a fault when it receives a business object for outbound processing that does not contain the required data or when the adapter encounters certain errors during outbound processing.

### Fault business objects

The external service wizard creates a business object for each fault that the adapter can generate. In addition, the wizard creates a WBIFault superset business object, which has information common to all faults, such as the message, errorCode, and primaryKeySet attributes as shown in Figure 84.



WBIFault	
message	string
errorCode	string
primaryKeySet	PrimaryKeyPairType []

Figure 84. The structure of the WBIFault business object

Some faults contain the matchCount attribute, to provide additional information about the error. For others, WBIFault contains all the information needed to handle the fault.

The WebSphere Adapter for Flat Files enables faults for you. Manual configuration of faults is not required. The adapter provides the following fault business objects that the wizard creates:

- DuplicateRecordFault

This fault is generated during the outbound Create operation when the file already exists in the specified directory.

- RecordNotFoundFault

This fault is generated during Append, Delete, Overwrite, and Retrieve operations when the file does not exist in the specified directory.

- MissingDataFault

If the business object that is passed to the outbound operation does not have all the required attributes, the adapter throws this fault. This fault can occur for the Create, Delete, Update, Retrieve, ApplyChanges and Exists operations.

For example, the adapter throws this fault if the content of the specified file is null, or the file name or directory path is empty.

---

## XAResourceNotAvailableException

When the process server log contains repeated reports of the `com.ibm.ws.Transaction.XAResourceNotAvailableException` exception, remove transaction logs to correct the problem.

### Symptom:

When the adapter starts, the following exception is repeatedly logged in the process server log file:

```
com.ibm.ws.Transaction.XAResourceNotAvailableException
```

### Problem:

A resource was removed while the process server was committing or rolling back a transaction for that resource. When the adapter starts, it tries to recover the transaction but cannot because the resource was removed.

### Solution:

To correct this problem, use the following procedure:

1. Stop the process server.
2. Delete the transaction log file that contains the transaction. Use the information in the exception trace to identify the transaction. This prevents the server from trying to recover those transactions.

**Note:** In a test or development environment, you can generally delete all of the transaction logs. In WebSphere Integration Developer, delete the files and subdirectories of the transaction log directory, `server_install_directory\profiles\profile_name\tranlog`.

In a production environment, delete only the transactions that represent events that you do not need to process. One way to do this is to reinstall the adapter, pointing it to the original event database used, and deleting only the transactions you do not need. Another approach is to delete the transactions from either the log1 or log2 file in the following directory:

```
server_install_directory\profiles\profile_name\tranlog\node_name\wps\  
server_name\transaction\tranlog
```

3. Start the process server.

---

## org.xml.sax.SAXParseException

When the adapter is configured with the XML data handler, an `org.xml.sax.SAXParseException` exception is generated if the content is not in the specified business object format. To correct the problem, make sure the file content matches the business object structure. If the file contains multiple business objects, make sure the delimiter is specified correctly.

### Symptom:

When the adapter is configured with the XML data handler, the following exception is thrown:

```
org.xml.sax.SAXParseException: Content is not allowed in trailing section
```



**Problem:**

The content of the file is not in the specified business object format.

**Solution:**

To correct this problem, use the following procedure:

1. Make sure the file content matches the business object structure.
2. If the content file contains multiple business objects, make sure the delimiter is specified correctly.

---

## Self-help resources

Use the resources of IBM software support to get the most current support information, obtain technical documentation, download support tools and fixes, and avoid problems with WebSphere Adapters. The self-help resources also help you diagnose problems with the adapter and provide information about how to contact IBM software support.

### Support Web site

The WebSphere Adapters software support Web site at <http://www.ibm.com/software/integration/wbiadapters/support/> provides links to many resources to help you learn about, use, and troubleshoot WebSphere Adapters, including the following types of

- Flashes (alerts about the product)
- Technical information including the product information center, manuals, IBM Redbooks®, and white papers
- Educational offerings
- Technotes

### Recommended fixes

A list of recommended fixes you should apply is available at the following location: <http://www.ibm.com/support/docview.wss?fdoc=aimadp&rs=695&uid=swg27010397>

### Technotes

Technotes provide the most current documentation about the Adapter for Flat Files, including the following topics:

- Problems and their currently available solutions
- Answers to frequently asked questions
- How-to information about installing, configuring, using, and troubleshooting the adapter
- *IBM Software Support Handbook*

For a list of technotes for WebSphere Adapters, visit this address:

<http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm>

## **Plug-in for IBM Support Assistant**

Adapter for Flat Files provides a plug-in for IBM Support Assistant, which is a free, local software serviceability workbench. The plug-in supports the dynamic trace feature. For information about installing or using IBM Support Assistant, visit this address:

<http://www.ibm.com/software/support/isa/>

---

## Chapter 9. Reference information

To support you in your tasks, reference information includes details about business objects that are generated by the external service wizard and information about adapter properties, including those that support bidirectional transformation. It also includes pointers to adapter messages and related product information.

---

### Business object information

You can determine the purpose of a business object by examining both the application-specific information within the business object definition file and the name of the business object. The application-specific information dictates what operations can be performed on the local file system. The name typically reflects the operation to be performed and the structure of the business object.

### Business object structures

The Adapter for Flat Files defines and generates business objects during external service. The business object structure is based on the generic WebSphere Business Integration business object structure, which is modeled as a base XML schema.

#### Generic FlatFileBG object

Two types of business objects are generated during enterprise metadata discovery: content-specific and generic.

The generic FlatFileBG business object is used for generic XSD files (for example, UnstructuredContent). The FlatFileBG business object is a wrapper business object that contains the FlatFile business object as a child. The following graphic illustrates this relationship:

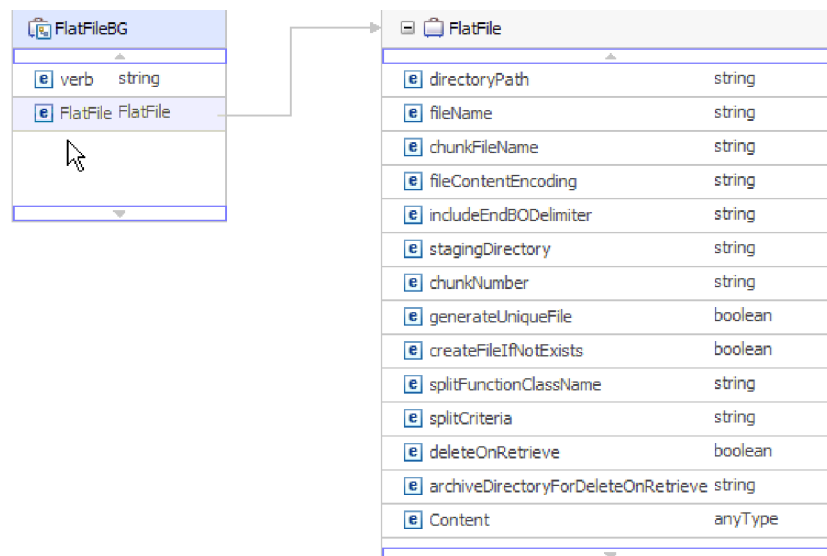


Figure 85. The generic FlatFileBG business object structure

## CustomerWrapperBG object

In this example, CustomerWrapperBG represents a content-specific XSD file. The CustomerWrapperBG is a wrapper business object that contains the CustomerWrapper business object as a child. The following graphic illustrates this relationship:

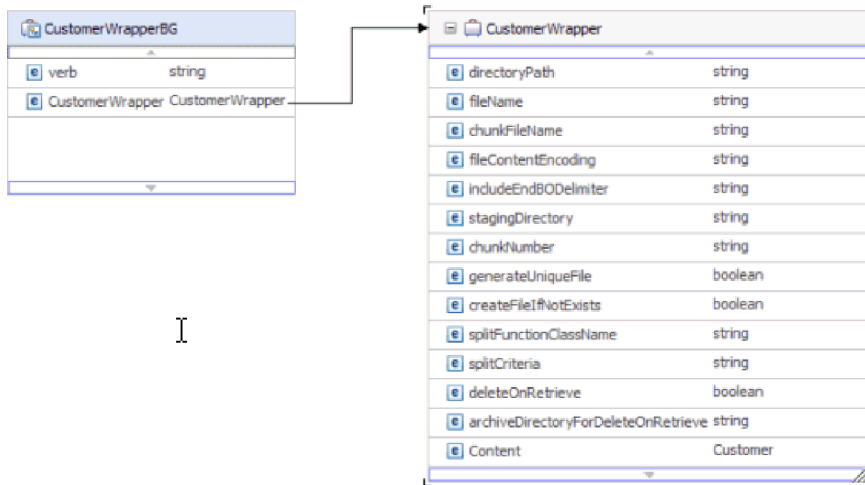


Figure 86. The CustomerWrapperBG business object structure

## Append operation response business object

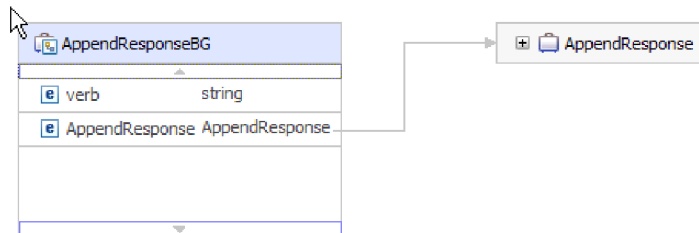


Figure 87. Structure of the Append operation response business object

## Create operation response business object

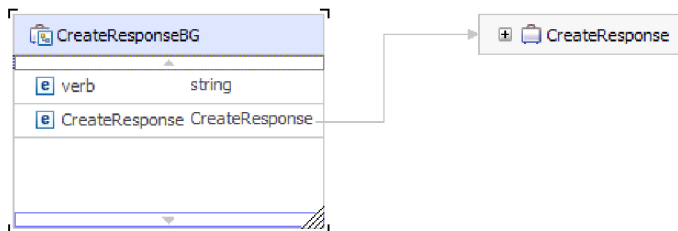


Figure 88. Structure of the Create operation response business object

## Exists operation response business object

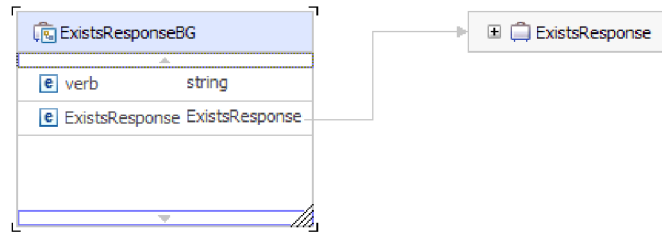


Figure 89. Structure of the Exists operation response business object

## List operation response business object

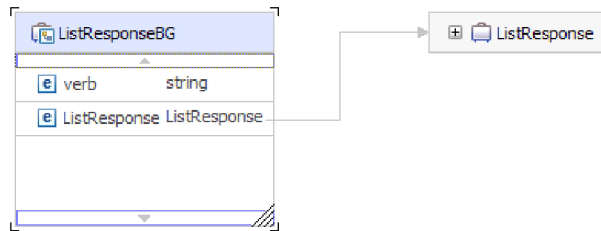


Figure 90. Structure of the List operation response business object

## Overwrite operation response business object

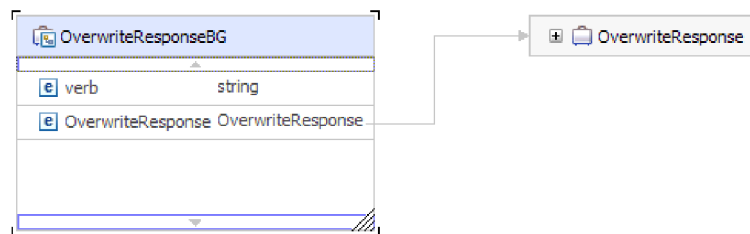


Figure 91. Structure of the Overwrite operation response business object

## Retrieve operation response business object

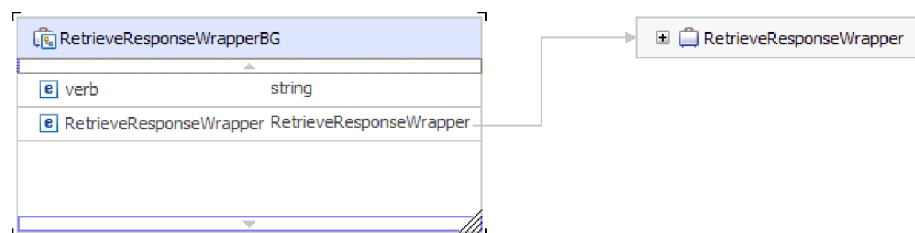


Figure 92. Structure of the Retrieve operation response business object

## Attribute properties

Business object architecture defines various properties that apply to attributes. This section describes how the adapter interprets these properties.

The following table describes these properties.

Table 12. Attribute properties

Attribute property	Description
Cardinality	Each business object attribute that represents a child or an array of child business objects has the value of single (1) or multiple (n) cardinality. Only single cardinality flat business objects are supported.
Key and foreign key	These attributes are not used by the adapter.
Name	Represents the unique name of the attribute.
Required	This attribute is not used by the adapter.
Type	The attribute type can be either simple or complex. Simple types are: Boolean, String, LongText, Integer, Float, Double and Byte[ ]. A typical complex type is another business object type.

## Naming conventions

When the external service wizard generates a business object, it provides a name for the business object based on the name of the object in the local file system that it uses to build the business object.

When the external service wizard provides a name for the business object, it converts the name of the object to mixed case, which means that it removes any separators, such as spaces or underscores, and then capitalizes the first letter of each word. For example, if the external service wizard uses a local file system object called CUSTOMER\_ADDRESS to generate a business object, it generates a business object called CustomerAddress.

The generated business object name can indicate the structure of the business object. However, business objects names have no semantic value to the adapter. This means that if you change the business object name, the behavior of the business object remains the same.

**Important:** If you choose to rename a business object, use the refactoring functionality in WebSphere Integration Developer to ensure that you update all of the business object dependencies. For instructions on using refactoring to rename business objects, refer to the following link: <http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.wbit.help.refactor.doc/topics/trenameboatt.html>.

The following table describes the naming conventions that the external service wizard uses when it generates business objects for the adapter for Flat Files.

Table 13. Naming conventions

Element	Naming convention	Example
Name of the business graph	The business graph that contains the parent business object is named for the contained business object, followed by the string BG. There can be a business graph only if there is a wrapper. CustomerWrapperBG is a wrapper business object that contains the CustomerWrapper business object as a child.	CustomerWrapperBG

**Note:** Business graph generation is optional and is supported for WebSphere Process Server or WebSphere Enterprise Service Bus only.

## Custom file splitting

You can implement a custom class containing the splitting logic. The adapter provides a Java interface for the class. The details of the interface are shown below.

```
public interface SplittingFunctionalityInterface extends Iterator{
    public int getTotalBOs(String filename) throws SplittingException;
    public void setBODetails(String filename, int currentPosition, int totalBOs,
        boolean includeEndBODElimiter) throws SplittingException;
    public void setSplitCriteria(String splitCriteria);
    public void setEncoding(String encoding);
    public void setLogUtils(LogUtils logUtils);
    public boolean isSplitBySize()
}
```

- `public int getTotalBOs(String filename) throws SplittingException`  
This method returns the total number of business object's present in the event file given by filename.
- `public void setSplitCriteria(String splitCriteria)`  
This method takes the `splitCriteria`, which is based on the number of business object's in the event file. Each business object is returned during the `next()` call.
- `public void setLogUtils(LogUtils logUtils)`  
This method is used to set the `LogUtils` object, which is the class that the user can use to write trace and log messages to the files.
- `public void setEncoding(String encoding)`  
This method is used to set the encoding of the event file content. This encoding is used while reading the file content. This encoding is also used for the `SplitCriteria`.
- `public void setBODetails(String filename, int currentPosition, int totalBOs, boolean includeEndBODElimiter) throws SplittingException`  
This method is used to set the current business object number so that whenever a `next()` call is made, the business object number set in the `currentPosition` is returned. It also takes an `includeEndBODElimiter` parameter, which when set to `true`, includes the `SplitCriteria` at the end of the business object content. This method must be called before every `next()` call so that the `next()` method returns the business object content for the business object set in this method.
- The iterator has 3 methods: `hasNext()`, `next` and `remove()`, which also need to be implemented. The `next()` method returns the business object content (as a `byte[]`) for the business object position set in `setBODetails()`. If the business object position is not set, it fails. The `hasNext()` method indicates whether the business

object position set in the `setBODetails()` exists or not. Before a `hasNext()` call, the `setBODetails()` method must be called. The `remove()` method is called for each of the business object entries being deleted from the Event persistence table. Do not delete the event file in this method. Only clean up resources that are being used.

- `public boolean isSplitBySize()`

This method indicates whether the event file is parsed based on size or based on delimiter.

#### **Related concepts**

“Inbound processing” on page 12

Adapter for Flat Files supports inbound event processing. It polls the local file system at specified intervals for events, such as the creation or modification of a file. When it detects an event, it converts the event data into a business object and sends it to the module for processing.

#### **Related tasks**

“Configuring the module for inbound processing” on page 97

To configure a module to use the adapter for inbound processing, use the external service wizard in WebSphere Integration Developer to build business services, specify data transformation processing, and generate business object definitions and related artifacts.

---

## **Outbound configuration properties**

WebSphere Adapter for Flat Files has several categories of outbound connection configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter and managed connection factory properties after you deploy the module to WebSphere Process Server or WebSphere Enterprise Service Bus using WebSphere Integration Developer or the administrative console, but connection properties for the external service wizard cannot be changed after deployment.

### **Guide to information about properties**

The properties used to configure WebSphere Adapter for Flat Files are described in detail in tables included in each of the configuration properties topics, such as Resource adapter properties, Managed connection factory properties, and so on. To help you use these tables, information about each row you might see is explained here.

The following table explains the meaning of each row that might be displayed in the table for a configuration property.



Row	Explanation
Required	<p>A required field (property) must have a value in order for the adapter to work. Sometimes the external service wizard provides a default value for required properties.</p> <p>Removing a default value from a required field on the external service wizard <i>will not change that default value</i>. When a required field contains no value at all, the external service wizard will process the field using its assigned default value, and that default value will also be displayed on the administrative console.</p> <p>Possible values are <b>Yes</b> and <b>No</b>.</p> <p>Sometimes a property is required only when another property has a specific value. When this is the case, the table will note this dependency. For example,</p> <ul style="list-style-type: none"> <li>• Yes, when the EventQueryType property is set to Dynamic</li> <li>• Yes, for Oracle databases</li> </ul>
Possible values	Lists and describes the possible values that you can select for the property.
Default	<p>The predefined value that is set by the external service wizard. When the property is required, you must either accept the default value or specify one yourself. If a property has no default value, the table will state No default value.</p> <p>The word None is an acceptable default value, and does not mean that there is no default value.</p>
Unit of measure	Specifies how the property is measured, for example in kilobytes or seconds.
Property type	<p>Describes the property type. Valid property types include the following:</p> <ul style="list-style-type: none"> <li>• Boolean</li> <li>• String</li> <li>• Integer</li> </ul>
Usage	<p>Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented:</p> <p>For Rational® Application Developer for WebSphere Software version 6.40 or earlier, the password:</p> <ul style="list-style-type: none"> <li>• Must be uppercase</li> <li>• Must be 8 characters in length</li> </ul> <p>For versions of Rational Application Developer for WebSphere Software later than 6.40, the password:</p> <ul style="list-style-type: none"> <li>• Is not case sensitive</li> <li>• Can be up to 40 characters in length.</li> </ul> <p>This section lists other properties that affect this property or the properties that are affected by this property and describes the nature of the conditional relationship.</p>
Example	<p>Provides sample property values, for example:</p> <p>"If Language is set to JA (Japanese), Codepage number is set to 8000".</p>
Globalized	<p>If a property is globalized, it has national language support, meaning that you can set the value in your national language.</p> <p>Valid values are <b>Yes</b> and <b>No</b>.</p>
Bidi supported	<p>Indicates whether the property is supported in bidirectional (bidi) processing. Bidirectional processing pertains to the task of processing data that contains both left-to-right (Hebrew or Arabic, for example) and right-to-left (a URL or file path, for example) semantic content within the same file.</p> <p>Valid values are <b>Yes</b> and <b>No</b>.</p>

## Connection properties for the wizard

Connection properties are used to build a service description and save the built-in artifacts. These properties are configured in the external service wizard.

The following table lists the connection properties for the external service wizard. These can only be configured using the external service wizard and cannot be changed after deployment. A complete description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see “Guide to information about properties” on page 154.

Table 14. Connection properties for the external service wizard

Property name in the wizard	Description
“Bidi format string”	The bidi format string of the content data
“Data binding”	Specifies the data binding that is to be used for all operations or specifies that a data binding is to be selected for each operation.
“Function selector” on page 157	During inbound processing, the name of the function selector configuration to be used.
“Log file output location” on page 157	The full path name of the log file generated by the external service wizard
“Logging level” on page 158	The level of logging to be used by the adapter
“NameSpace” on page 158	The namespace of the business object that is generated
“Operation name” on page 158	The operation defined in the external service wizard
“Processing Direction” on page 158	The processing direction, Inbound or Outbound

### Bidi format string

The bidi format string of the content data.

Table 15. Bidi format string

Required	No
Default	None
Property type	String

### Data binding

Specifies the data binding that is to be used for all operations or specifies that a data binding is to be selected for each operation.

Table 16. Data binding details

Required	No
Default	Use default data binding ‘FlatFileBaseDataBinding’ for all operations
Usage	<p>The value of this property can be:</p> <ul style="list-style-type: none"> <li>• Use default data binding ‘FlatFileBaseDataBinding’ for all operations</li> <li>• Use a data binding configuration for all operations</li> <li>• Specify a data binding for each operation</li> </ul>

Table 16. Data binding details (continued)

Globalized	No
Bidi supported	No

## Function selector

During inbound processing, the name of the function selector configuration to be used.

Table 17. Function selector details

Required	Yes
Default	FilenameFunctionSelector
Property type	String
Usage	<p>The function selector returns the appropriate operation to be called on the service. The adapter provides two function selectors, <code>FilenameFunctionSelector</code> and <code>EmbeddedNameFunctionSelector</code>.</p> <ul style="list-style-type: none"> <li><code>FilenameFunctionSelector</code> is a rule-based function selector that matches a regular expression on a file name to an object name. Use <code>FilenameFunctionSelector</code> for generic <code>FlatFile</code> business objects, where the object name cannot be determined from the event file. <code>FilenameFunctionSelector</code> is represented in properties as a two-column table with <math>N</math> rows. For any event file with a <code>.txt</code> extension, the corresponding object name is <code>FlatFile</code>, and the endpoint method name generated by the function selector is <code>emitFlatFile</code>. You must set this same name in the <code>EISFunctionName</code> property after you add the operation. You can configure <code>FilenameFunctionSelector</code> with multiple rules, each containing an object name and a regular expression to match against the file name. If more than one rule matches, the function selector returns the object name based on the first matching rule.</li> <li>Use <code>EmbeddedNameFunctionSelector</code> for content-specific business objects, where the object name is embedded in the event file. <code>EmbeddedNameFunctionSelector</code> returns the function name based on the desired content data, not the wrapper. For example, if the content-specific business object is <code>CustomerWrapperBG</code>, the function returned by the function selector is <code>emitCustomer</code>. You must configure <code>EmbeddedNameFunctionSelector</code> with a data handler. The data binding must be the adapter-specific <code>WrapperDataBinding</code>, and it must be configured to use the same data handler that is configured with the function selector.</li> </ul>
Globalized	Yes
Bidi supported	No

## Log file output location

The full path name of the log file generated by the external service wizard.

Table 18. Log file output location details

Required	No
Default	<code>\.metadata \FlatFileMetadataDiscoveryImpl.log</code>
Property type	String
Usage	
Globalized	No
Bidi supported	No

## Logging level

The level of logging to be used by the adapter.

Table 19. Logging level details

Required	No
Possible values	Severe Warning Audit Info Config Detail
Default	Severe
Property type	List of values
Globalized	No
Bidi supported	No

## NameSpace

The namespace of the business object that is generated.

Table 20. NameSpace details

Required	Yes
Default	<a href="http://www.ibm.com/xmlns/prod/websphere/j2ca/flatfile">http://www.ibm.com/xmlns/prod/websphere/j2ca/flatfile</a>
Property type	String
Globalized	Yes
Bidi supported	No

## Operation name

The name you give to the operation defined for this module.

Table 21. Operation name details

Required	No
Default	When the ServiceType property is set to Outbound, the operations listed are Create, Append, Retrieve, Delete, List, Overwrite, and Exists.
Property type	String
Globalized	No
Bidi supported	No

## Processing Direction

The processing direction, inbound or outbound.

Table 22. Processing Direction details

Required	Yes
Possible values	Outbound Inbound
Default	Outbound

Table 22. Processing Direction details (continued)

Property type	String
Globalized	No
Bidi supported	No

### Related concepts

“Globalization and bidirectional data transformation” on page 193  
 The adapter is globalized to support single- and multi-byte character sets and deliver message text in the specified language. The adapter also performs bidirectional script data transformation, which refers to the task of processing data that contains both right-to-left (Hebrew or Arabic, for example) and left-to-right (a URL or file path, for example) semantic content within the same file.

“Function selectors” on page 18  
 During inbound processing, a function selector returns the appropriate operation to be called on the service. You choose a function selector when you configure the adapter for inbound processing in the external service wizard. The adapter provides two function selectors, `FilenameFunctionSelector` and `EmbeddedNameFunctionSelector`.

### Related tasks

“Configuring logging properties” on page 142  
 Use the administrative console to enable logging and to set the output properties for a log, including the location, level of detail, and output format of the log.

## Managed connection factory properties

Managed connection factory properties specify information the adapter needs at run time for outbound communication with the local file system.

The following table lists the managed connection factory properties for outbound communication. You set the managed connection factory properties using the external service wizard and can change them using the WebSphere Integration Developer Assembly Editor, or after deployment through the WebSphere Process Server or WebSphere Enterprise Service Bus administrative console.

A more detailed description of each property is provided in the sections that follow the table. For information about how to read the property details tables in the sections that follow, see “Guide to information about properties” on page 154.

**Note:** The external service wizard refers to these properties as managed connection factory properties and the WebSphere Process Server or WebSphere Enterprise Service Bus administrative console refers to them as (J2C) connection factory properties.

Table 23. Managed connection factory properties

Property name		Description
In the wizard	In the administrative console	
Adapter ID	AdapterID	Identifies the adapter instance for PMI events and for logging and tracing

Table 23. Managed connection factory properties (continued)

Property name		Description
In the wizard	In the administrative console	
"Default target file name" on page 161	OutputFileName	The name of the file that is created in the output directory, or a WebSphere Application Server environment variable that represents this file
Disguise user data as "XXX" in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing strings of X's instead of user data in log and trace files
"Output directory" on page 162	OutputDirectory	The full path name of the directory where the adapter creates files during outbound operations, or a WebSphere Application Server environment variable that represents this directory
"Sequence file" on page 162	FileSequenceLog	The full path name of the file where sequences are stored during outbound Create operations, or a WebSphere Application Server environment variable that represents this file
"Staging directory" on page 163	StagingDirectory	The full path name of the temporary directory where the adapter writes the initial output files for Create and Overwrite operations during outbound processing, or a WebSphere Application Server environment variable that represents this directory

### Adapter ID (AdapterID)

This property identifies a specific deployment, or instance, of the adapter.

Table 24. Adapter ID details

Required	Yes
Default	001
Property type	String

Table 24. Adapter ID details (continued)

Usage	<p>This property identifies the adapter instance in log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, FFRA, to form the component name used by the Log and Trace Analyzer tool. For example, if the adapter ID property is set to 001, the component ID is FFRA001.</p> <p>If you run multiple instances of the same adapter, make sure that the first nine characters of the adapter ID property are unique for each instance so that you can correlate log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate log and trace information to a particular instance of an adapter.</p> <p>To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for Flat Files to 001 and 002. The component IDs for those instances, FFRA001 and FFRA002, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. For example, suppose you set the adapter ID properties of two instances to Instance01 and Instance02. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to FFRAInstance0.</p> <p>For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, make sure you set them consistently, to prevent inconsistent marking of the log and trace entries.</p>
Globalized	Yes
Bidi supported	No

### Default target file name

The name of the file that is created in the output directory, or a WebSphere Application Server environment variable that represents this file.

Table 25. Default target file name details

Required	No
Default	None
Property type	String
Usage	<p>If a value for OutputFileName is specified in the record object, this value is overridden. You can use a WebSphere Application Server environment variable to represent the default target file name. Specify the name of the environment variable in braces, preceded by a \$ symbol. For example: \${OUTPUT_FILENAME}. See the topic on creating an environment variable in this documentation.</p>
Globalized	Yes
Bidi supported	Yes

### Disguise user data as "XXX" in log and trace files (HideConfidentialTrace) property

This property specifies whether to replace user data in log and trace files with a string of X's to prevent unauthorized disclosure of potentially sensitive data.

Table 26. Disguise user data as "XXX" in log and trace files details

Required	No
Possible values	True False
Default	False
Property type	Boolean
Usage	<p>If you set this property to True, the adapter replaces user data with a string of X's when writing to log and trace files.</p> <p>For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, make sure you set them consistently, to prevent inconsistent marking of the log and trace entries.</p>
Globalized	No
Bidi supported	No

## Output directory

The full path name of the directory where the adapter creates files during outbound operations, or a WebSphere Application Server environment variable that represents this directory.

Table 27. Output directory details

Required	No
Default	None
Property type	String
Usage	<p>The output directory is used by the adapter to write the final output files. You can use a WebSphere Application Server environment variable to represent the output directory. Specify the name of the environment variable in braces, preceded by a \$ symbol. For example: <code>\${OUTPUT_DIRECTORY}</code>. See the topic on creating an environment variable in this documentation.</p>
Globalized	Yes
Bidi supported	Yes

## Sequence file

This property specifies the full path name of the file where sequences are stored during outbound Create operations, or a WebSphere Application Server environment variable that represents this file.

Table 28. Sequence file details

Required	No
Default	None
Property type	String



Table 28. Sequence file details (continued)

Usage	<p>When the adapter receives a Create request, it checks the file sequence log to see whether a file with the specified name already exists. If there is a file with that name, the adapter uses the file sequence number to create a new file name. For example, if the output file name in the request is Customer.txt, the adapter creates a file with the name Customer.n.txt, where <i>n</i> is the sequence number. If the output file name does not have an extension, the sequence is appended at the end of the file name; for example, Customer<i>n</i>. All sequences begin with 1.</p> <p>If this property is not specified, and the adapter receives a request to create a file with a name that already exists, the adapter generates a <code>DuplicateRecordException</code> error.</p> <p>The sequence number continues to increment after an adapter restart. You can reset the file sequence by changing the sequence value in the sequence file.</p> <p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>1. To generate file sequencing for a particular request type, set the output directory and the file name at the managed connection level.</li> <li>2. When the adapter is working in a clustered environment, make sure that the sequence file is on a mapped drive that is accessible by all the clusters. The adapter must have write permission for the sequence log file, or an <code>IOException</code> error is returned.</li> <li>3. If the <code>FileSequenceLog</code> property is specified and the <code>GenerateUniqueFile</code> property is enabled, the <code>GenerateUniqueFile</code> property takes precedence over the <code>FileSequenceLog</code> property.</li> <li>4. The directory path and file name, if specified in the business object, take precedence over the values specified at the managed connection level.</li> </ol> <p>You can use a WebSphere Application Server environment variable to represent the sequence file. Specify the name of the environment variable in braces, preceded by a \$ symbol. For example: <code>\${SEQUENCE_FILE}</code>. See the topic on creating an environment variable in this documentation.</p> <p><b>Important:</b> Unless they are part of a cluster, two adapter instances should not use the same sequence file, because this could result in delayed processing of batch requests.</p>
Globalized	Yes
Bidi supported	Yes

## Staging directory

The full path name of the temporary directory where the adapter writes the initial output files for Create and Overwrite operations during outbound processing, or a WebSphere Application Server environment variable that represents this directory.

Table 29. Staging directory details

Required	No
Default	None
Property type	String
Usage	<p>If this property is specified, the output file is first written to a staging directory and then renamed to the output directory. The adapter temporarily stores the initial output files for Create and Overwrite operations in the staging directory to avoid write conflicts during outbound processing.</p> <p>You can use a WebSphere Application Server environment variable to represent the staging directory. Specify the name of the environment variable in braces, preceded by a \$ symbol. For example: <code>\${STAGING_DIRECTORY}</code>. See the topic on creating an environment variable in this documentation.</p>
Globalized	Yes

Table 29. Staging directory details (continued)

Bidi supported	Yes
----------------	-----

**Related concepts**

“WebSphere Application Server environment variables” on page 24  
 WebSphere Application Server environment variables can be used in the external service wizard to specify directory values.

“Creating the required local folders” on page 48

Before you create inbound or outbound modules, you must create folders on the local file system for events and output. You can optionally create folders for staging and archiving.

**Related tasks**

“Defining WebSphere Application Server environment variables” on page 51  
 Use the administrative console of WebSphere Process Server or WebSphere Enterprise Service Bus to define WebSphere Application Server environment variables.

**Resource adapter properties**

The resource adapter properties control the general operation of the adapter, such as specifying the namespace for business objects. You set the resource adapter properties using the external service wizard when you configure the adapter. After deploying the adapter, use the administrative console to change these properties.

The following properties for logging and tracing are no longer required in version 6.1.0. They are visible from the administrative console for compatibility with previous versions.

- LogFileMaxSize
- LogFileName
- LogNumberOfFiles
- TraceFileMaxSize
- TraceFileName
- TraceNumberOfFiles

The following table lists the resource adapter properties and their purpose. A complete description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see “Guide to information about properties” on page 154.

Table 30. Resource adapter properties for the Adapter for Flat Files

Name		Description
In the wizard	In the administrative console	
Adapter ID	AdapterID	Identifies the adapter instance for PMI events and for logging and tracing
Disguise user data as "XXX" in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing strings of X's instead of user data in log and trace files
(Not available)	Enable HA support	Do not change this property.
(Not available)	LogFileMaxSize	Deprecated
(Not available)	LogFilename	Deprecated

Table 30. Resource adapter properties for the Adapter for Flat Files (continued)

Name		Description
In the wizard	In the administrative console	
(Not available)	LogNumberOfFiles	Deprecated
(Not available)	TraceFileMaxSize	Deprecated
(Not available)	TraceFileName	Deprecated
(Not available)	TraceNumberOfFiles	Deprecated

## Adapter ID (AdapterID)

This property identifies a specific deployment, or instance, of the adapter.

Table 31. Adapter ID details

Required	Yes
Default	001
Property type	String
Usage	<p>This property identifies the adapter instance in log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, FFRA, to form the component name used by the Log and Trace Analyzer tool. For example, if the adapter ID property is set to 001, the component ID is FFRA001.</p> <p>If you run multiple instances of the same adapter, make sure that the first nine characters of the adapter ID property are unique for each instance so that you can correlate log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate log and trace information to a particular instance of an adapter.</p> <p>To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for Flat Files to 001 and 002. The component IDs for those instances, FFRA001 and FFRA002, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. For example, suppose you set the adapter ID properties of two instances to Instance01 and Instance02. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to FFRAInstance0.</p> <p>For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, make sure you set them consistently, to prevent inconsistent marking of the log and trace entries.</p>
Globalized	Yes
Bidi supported	No

## Disguise user data as "XXX" in log and trace files (HideConfidentialTrace) property

This property specifies whether to replace user data in log and trace files with a string of X's to prevent unauthorized disclosure of potentially sensitive data.

Table 32. Disguise user data as "XXX" in log and trace files details

Required	No
Possible values	True False
Default	False
Property type	Boolean
Usage	<p>If you set this property to True, the adapter replaces user data with a string of X's when writing to log and trace files.</p> <p>For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, make sure you set them consistently, to prevent inconsistent marking of the log and trace entries.</p>
Globalized	No
Bidi supported	No

## Enable high availability support (enableHASupport)

Do not change this property. It must be set to true.

## Interaction specification properties

Interaction specification properties contain the outbound connection properties the adapter uses to interface with the file system. You configure these properties using the external service wizard. To change the interaction specification properties after the application has been deployed, use the assembly editor in WebSphere Integration Developer.

Interaction specification properties control the interaction for an operation. The external service wizard sets the interaction specification properties when you configure the adapter. Typically, you do not need to change these properties. However, some properties for outbound operations can be changed by the user. To change these properties after the application is deployed, use the assembly editor in WebSphere Integration Developer. The properties reside in the method binding of the import.

The following table lists the interaction specification properties. A complete description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 154.

Table 33. Interaction specification properties

Property name		Description
In the wizard	In the administrative console	
"Archive directory for retrieve operation" on page 167	ArchiveDirectoryForDeleteOnRetrieve	The directory where retrieved files are stored before they are deleted, if the DeleteOnRetrieve property is set to true

Table 33. Interaction specification properties (continued)

Property name		Description
In the wizard	In the administrative console	
“Create a new file if the file does not exist”	CreateFileIfNotExists	When this property is set to true, the adapter creates a new file during Append and Overwrite operations if the file does not exist
“Default target file name” on page 168	OutputFileName	The name of the output file that is created or modified
“Delete the file after retrieve operation” on page 168	DeleteOnRetrieve	During Retrieve operations, when this property is set to true, the file is deleted from the file system after the file content has been retrieved
“Delimiter between business objects in the file” on page 168	IncludeEndBODelimiter	The file content is appended with this value.
“File content encoding” on page 169	FileContentEncoding	Specifies the encoding set used in writing to or reading from the event file.
“Generate a unique file” on page 169	GenerateUniqueFile	Specifies that the adapter will create a unique file during Create, Append, and Overwrite operations
“Output directory” on page 170	OutputDirectory	The full path name of the directory on the local file system where the adapter writes the output files
“Specify criteria to split file content” on page 170	SplitCriteria	Specifies either the delimiter that separates the business objects in the retrieved file or the size of the chunks into which the retrieved file is split
“Split function class name” on page 171	SplittingFunctionClassName	Specifies how the retrieved file is to be split, by delimiter or by size, during an outbound Retrieve operation
“Staging directory” on page 171	StagingDirectory	A temporary directory where the adapter stores the initial output files during Create and Overwrite operations

### Archive directory for retrieve operation

The directory where retrieved files are stored before they are deleted, if the DeleteOnRetrieve property is set to true.

Table 34. Archive directory for retrieve operation details

Required	No
Default	None
Property type	String
Globalized	Yes
Bidi supported	Yes

### Create a new file if the file does not exist

When this property is set to true, the adapter creates a new file during Append and Overwrite operations if the file does not exist.

Table 35. Create a new file if the file does not exist details

Required	No
----------	----

Table 35. Create a new file if the file does not exist details (continued)

Possible values	True False
Default	False
Property type	Boolean
Usage	When this property is set to false and the file does not exist, the adapter generates the RecordNotFoundException error. <b>Note:</b> If a value is unset for this property on the wrapper, then the value set here is used.
Globalized	No
Bidi supported	No

### Default target file name

The name of the output file that is created or modified.

Table 36. Default target file name details

Required	Required for all outbound operations except List
Default	None
Property type	String
Globalized	Yes
Bidi supported	Yes

### Delete the file after retrieve operation

During Retrieve operations, if this property is set to true, the file is deleted from the file system after the file content has been retrieved.

Table 37. Delete the file after retrieve operation details

Required	No
Possible values	True False
Default	False
Property type	Boolean
Usage	To archive the file before it is deleted, specify a directory in the ArchiveDirectoryForDeleteOnRetrieve property. <b>Note:</b> If a value is unset for this property on the wrapper, then the value set here is used.
Globalized	No
Bidi supported	No

### Delimiter between business objects in the file

The file content is appended with this value.

Table 38. Delimiter between business objects in the file details

Required	No
Default	None
Property type	String

Table 38. Delimiter between business objects in the file details (continued)

Usage	This property is used during outbound Create, Append, and Overwrite operations. Any value specified in this property is appended to the file. If the value specified has escape sequence characters and Unicode escape characters, they are parsed and the corresponding control characters are inserted in the file. The escape sequence characters include the following: carriage return (\r), new line (\n), carriage return and new line (\r\n), tab space (\t), backspace (\b), form feed (\f), and so on. An example of a Unicode escape character that represents the character '?' is \u2297."
Globalized	Yes
Bidi supported	Yes

## File content encoding

The encoding set used when writing to or reading from the event file.

**Note:** During the Create operation, the adapter creates the file with the specified encoding.

Table 39. File content encoding details

Required	No
Possible values	Any Java supported encoded character sets.
Default	UTF-8
Property type	String
Usage	You can specify any Java supported encoding set, such as UTF-8. If the adapter is working with binary event data, set this property to BINARY. If the adapter is working with non-binary event data, such as text or XML, set this property to a valid file encoding value, such as UTF-8 or UTF-16. <b>Note:</b> The value set in the interaction specification property is used only if no value is set on the wrapper.
Globalized	No
Bidi supported	No

## Generate a unique file

Specifies that the adapter will create a unique file during Create, Append, and Overwrite operations.

Table 40. Generate a unique file details

Required	No
Possible values	True False
Default	False
Property type	Boolean
Usage	During Create operations, if this property is set to True, the adapter creates a unique file and ignores any value set for the Filename property. <b>Note:</b> If a value is unset for this property on the wrapper, then the value set here is used.
Globalized	Yes
Bidi supported	No

## Output directory

The full path name of the directory on the local file system where the adapter writes the output files.

Table 41. Output directory details

Required	No
Default	None
Property type	String
Usage	If this property is not specified, the adapter writes the output files to the directory specified by the OutputFileName property on the request.
Globalized	Yes
Bidi supported	Yes

## Specify criteria to split file content

This property specifies either the delimiter that separates the business objects in the retrieved file, or the size of the chunks into which the retrieved file is split.

Table 42. Specify criteria to split file content details

Required	No
Possible values	A delimiter or a valid number
Default	0
Property type	String
Usage	<p>This property specifies either the delimiter that separates the business objects in the retrieved file, or the size of the chunks into which the retrieved file is split. The value of this property is determined by the value that is set in the SplittingFunctionClassName property:</p> <ul style="list-style-type: none"><li>• If the SplittingFunctionClassName property is set to <code>com.ibm.j2ca.utils.filesplit.SplitByDelimiter</code>, the SplitCriteria property must contain the delimiter that separates the business objects in the retrieved file.</li><li>• If the SplittingFunctionClassName property is set to <code>com.ibm.j2ca.utils.filesplit.SplitBySize</code>, the SplitCriteria property must contain a valid number that represents the size in bytes. If the retrieved file size is greater than this value, it is split into chunks of this value and that number of chunks are posted. If the file size is less than this value, the entire event file is posted.</li></ul> <p>If the SplitCriteria property is set to 0, chunking is disabled.</p> <p>The SplitCriteria property must contain the same value for the newline character as the event file. For example, if the event file was created on a Macintosh system, the newline character is <code>\r</code>, and the SplitCriteria property must contain <code>\r</code>. The platform-specific newline characters are as follows:</p> <ul style="list-style-type: none"><li>Macintosh - <code>\r</code></li><li>Microsoft Windows - <code>\r\n</code></li><li>UNIX - <code>\n</code></li></ul> <p>If there is more than one delimiter in the SplitCriteria property, each delimiter must be separated by a semicolon (;). If a semicolon (;) itself is part of the delimiter, the semicolon (;) should be escaped, as in <code>\;</code>. For example, if the delimiter given is <code>##\;##</code>, it is evaluated to <code>##;##</code>.</p>
Globalized	Yes
Bidi supported	Yes



## Split function class name

This property specifies how the retrieved file is to be split, by delimiter or by size, during an outbound Retrieve operation.

Table 43. Split function class name details

Required	No
Possible values	com.ibm.j2ca.utils.filesplit.SplitByDelimiter – Files are split based on a delimiter that separates business objects in the event file com.ibm.j2ca.utils.filesplit.SplitBySize – Files are split based on the size of the event file
Default	com.ibm.j2ca.utils.filesplit.SplitBySize
Property type	String
Usage	The delimiter or file size is set in the SplitCriteria property.
Globalized	No
Bidi supported	No

## Staging directory

A temporary directory where the adapter stores the initial output files during Create and Overwrite operations to avoid write conflicts.

Table 44. Staging directory details

Required	No
Default	None
Property type	String
Usage	If a staging directory is specified, the file that is to be operated on is copied from the output directory to the staging directory. The operation is performed on the file in the staging directory, and the file is then renamed and copied to the output directory.
Globalized	Yes
Bidi supported	Yes

### Related tasks

Chapter 5, “Changing interaction specification properties using the assembly editor,” on page 113

To change interaction specification properties for your adapter module after generating the service, use the assembly editor in WebSphere Integration Developer.

---

## Inbound configuration properties

WebSphere Adapter for Flat Files has several categories of inbound connection configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter and activation specification properties after you deploy the module using WebSphere Integration Developer or the administrative console, but connection properties for the external service wizard cannot be changed after deployment.

## Guide to information about properties

The properties used to configure WebSphere Adapter for Flat Files are described in detail in tables included in each of the configuration properties topics, such as

Resource adapter properties, Managed connection factory properties, and so on. To help you use these tables, information about each row you might see is explained here.

The following table explains the meaning of each row that might be displayed in the table for a configuration property.

Row	Explanation
Required	<p>A required field (property) must have a value in order for the adapter to work. Sometimes the external service wizard provides a default value for required properties.</p> <p>Removing a default value from a required field on the external service wizard <i>will not change that default value</i>. When a required field contains no value at all, the external service wizard will process the field using its assigned default value, and that default value will also be displayed on the administrative console.</p> <p>Possible values are <b>Yes</b> and <b>No</b>.</p> <p>Sometimes a property is required only when another property has a specific value. When this is the case, the table will note this dependency. For example,</p> <ul style="list-style-type: none"> <li>• Yes, when the EventQueryType property is set to Dynamic</li> <li>• Yes, for Oracle databases</li> </ul>
Possible values	Lists and describes the possible values that you can select for the property.
Default	<p>The predefined value that is set by the external service wizard. When the property is required, you must either accept the default value or specify one yourself. If a property has no default value, the table will state No default value.</p> <p>The word None is an acceptable default value, and does not mean that there is no default value.</p>
Unit of measure	Specifies how the property is measured, for example in kilobytes or seconds.
Property type	<p>Describes the property type. Valid property types include the following:</p> <ul style="list-style-type: none"> <li>• Boolean</li> <li>• String</li> <li>• Integer</li> </ul>
Usage	<p>Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented:</p> <p>For Rational Application Developer for WebSphere Software version 6.40 or earlier, the password:</p> <ul style="list-style-type: none"> <li>• Must be uppercase</li> <li>• Must be 8 characters in length</li> </ul> <p>For versions of Rational Application Developer for WebSphere Software later than 6.40, the password:</p> <ul style="list-style-type: none"> <li>• Is not case sensitive</li> <li>• Can be up to 40 characters in length.</li> </ul> <p>This section lists other properties that affect this property or the properties that are affected by this property and describes the nature of the conditional relationship.</p>
Example	<p>Provides sample property values, for example:</p> <p>"If Language is set to JA (Japanese), Codepage number is set to 8000".</p>

Row	Explanation
Globalized	If a property is globalized, it has national language support, meaning that you can set the value in your national language.  Valid values are <b>Yes</b> and <b>No</b> .
Bidi supported	Indicates whether the property is supported in bidirectional (bidi) processing. Bidirectional processing pertains to the task of processing data that contains both left-to-right (Hebrew or Arabic, for example) and right-to-left (a URL or file path, for example) semantic content within the same file.  Valid values are <b>Yes</b> and <b>No</b> .

## Connection properties for the wizard

Connection properties are used to build a service description and save the built-in artifacts. These properties are configured in the external service wizard.

The following table lists the connection properties for the external service wizard. These can only be configured using the external service wizard and cannot be changed after deployment. A complete description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see “Guide to information about properties” on page 154.

*Table 45. Connection properties for the external service wizard*

Property name in the wizard	Description
“Bidi format string”	The bidi format string of the content data
“Data binding” on page 174	Specifies the data binding that is to be used for all operations or specifies that a data binding is to be selected for each operation.
“Function selector” on page 174	During inbound processing, the name of the function selector configuration to be used.
“Log file output location” on page 174	The full path name of the log file generated by the external service wizard
“Logging level” on page 175	The level of logging to be used by the adapter
“NameSpace” on page 175	The namespace of the business object that is generated
“Operation name” on page 175	The operation defined in the external service wizard
“Processing Direction” on page 176	The processing direction, Inbound or Outbound

### Bidi format string

The bidi format string of the content data.

*Table 46. Bidi format string*

Required	No
Default	None
Property type	String

## Data binding

Specifies the data binding that is to be used for all operations or specifies that a data binding is to be selected for each operation.

Table 47. Data binding details

Required	No
Default	Use default data binding 'FlatFileBaseDataBinding' for all operations
Usage	The value of this property can be: <ul style="list-style-type: none"><li>• Use default data binding 'FlatFileBaseDataBinding' for all operations</li><li>• Use a data binding configuration for all operations</li><li>• Specify a data binding for each operation</li></ul>
Globalized	No
Bidi supported	No

## Function selector

During inbound processing, the name of the function selector configuration to be used.

Table 48. Function selector details

Required	Yes
Default	FilenameFunctionSelector
Property type	String
Usage	<p>The function selector returns the appropriate operation to be called on the service. The adapter provides two function selectors, <code>FilenameFunctionSelector</code> and <code>EmbeddedNameFunctionSelector</code>.</p> <ul style="list-style-type: none"><li>• <code>FilenameFunctionSelector</code> is a rule-based function selector that matches a regular expression on a file name to an object name. Use <code>FilenameFunctionSelector</code> for generic <code>FlatFile</code> business objects, where the object name cannot be determined from the event file. <code>FilenameFunctionSelector</code> is represented in properties as a two-column table with <math>N</math> rows. For any event file with a <code>.txt</code> extension, the corresponding object name is <code>FlatFile</code>, and the endpoint method name generated by the function selector is <code>emitFlatFile</code>. You must set this same name in the <code>EISFunctionName</code> property after you add the operation. You can configure <code>FilenameFunctionSelector</code> with multiple rules, each containing an object name and a regular expression to match against the file name. If more than one rule matches, the function selector returns the object name based on the first matching rule.</li><li>• Use <code>EmbeddedNameFunctionSelector</code> for content-specific business objects, where the object name is embedded in the event file. <code>EmbeddedNameFunctionSelector</code> returns the function name based on the desired content data, not the wrapper. For example, if the content-specific business object is <code>CustomerWrapperBG</code>, the function returned by the function selector is <code>emitCustomer</code>. You must configure <code>EmbeddedNameFunctionSelector</code> with a data handler. The data binding must be the adapter-specific <code>WrapperDataBinding</code>, and it must be configured to use the same data handler that is configured with the function selector.</li></ul>
Globalized	Yes
Bidi supported	No

## Log file output location

The full path name of the log file generated by the external service wizard.

Table 49. Log file output location details

Required	No
Default	\\.metadata \\FlatFileMetadataDiscoveryImpl.log
Property type	String
Usage	
Globalized	No
Bidi supported	No

## Logging level

The level of logging to be used by the adapter.

Table 50. Logging level details

Required	No
Possible values	Severe Warning Audit Info Config Detail
Default	Severe
Property type	List of values
Globalized	No
Bidi supported	No

## NameSpace

The namespace of the business object that is generated.

Table 51. NameSpace details

Required	Yes
Default	<a href="http://www.ibm.com/xmlns/prod/websphere/j2ca/flatfile">http://www.ibm.com/xmlns/prod/websphere/j2ca/flatfile</a>
Property type	String
Globalized	Yes
Bidi supported	No

## Operation name

The name you give to the operation defined for this module.

Table 52. Operation name details

Required	No
Default	When the ServiceType property is set to Outbound, the operations listed are Create, Append, Retrieve, Delete, List, Overwrite, and Exists.
Property type	String
Globalized	No
Bidi supported	No

## Processing Direction

The processing direction, inbound or outbound.

Table 53. Processing Direction details

Required	Yes
Possible values	Outbound Inbound
Default	Outbound
Property type	String
Globalized	No
Bidi supported	No

## Activation specification properties

Activation specification properties hold the inbound event processing configuration information for an export. You set activation specification properties through either the external service wizard or the administrative console.

The following activation specification properties are no longer required in version 6.1.0, but are supported for compatibility with previous versions.

- ArchivingProcessed
- DefaultObjectName
- EventContentType

The following table lists the activation specification properties for inbound communication. You set the activation specification properties using the external service wizard and can change them before deployment by using the WebSphere Integration Developer Assembly Editor or after deployment through the WebSphere Process Server administrative console.

A detailed description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see “Guide to information about properties” on page 154.

Table 54. Activation specification properties

Property name		
In the wizard	In the administrative console	Description
“Archive directory” on page 179	ArchiveDirectory	The directory where the adapter archives processed event files.
(Not available)	ArchivingProcessed	Deprecated
“Auto create event table” on page 179	EP_Create Table	Determines whether the event persistence table is created automatically or manually.
“Bidirectional transformation of event persistence properties” on page 179	EP_BiDiFormat	Determines whether the adapter transforms any of the event persistence properties
(Not available)	DefaultObjectName	Deprecated

Table 54. Activation specification properties (continued)

Property name		
In the wizard	In the administrative console	Description
Delivery type	DeliveryType	Determines the order in which events are delivered by the adapter to the export
Ensure once only event delivery	AssuredOnceDelivery	Specifies whether the adapter provides assured once delivery of events
“Database schema name” on page 180	EP_SchemaName	The schema name of the database used by event persistence processing.
Do not process events that have a timestamp in the future	FilterFutureEvents	Specifies whether the adapter filters out future events by comparing the timestamp on each event with the system time
(Not available)	EventContentType	Deprecated
“Event directory” on page 181	EventDirectory	The directory where the event files are stored.
“Event recovery data source (JNDI) name” on page 182	EP_DataSource_JNDIName	The JNDI name of the data source used by event persistence processing to obtain the JDBC database connection. The data source must be created in WebSphere Process Server.
“Event recovery table name” on page 182	EP_TableName	The name of the table used by the adapter for event persistence processing.
Event types to process	EventTypeFilter	A delimited list of event types that indicates to the adapter which events it should deliver
Retry limit for failed events (FailedEventRetryLimit)	FailedEventRetryLimit	The number of times the adapter attempts to redeliver an event before marking the event as failed
“Failure file extension for archive” on page 183	FailedArchiveExtension	The file extension used to archive unsuccessfully processed business objects in the input event file.
“File content encoding” on page 183	FileContentEncoding	The encoding of the files that are read by the adapter.
“File extension for archive” on page 184	OriginalArchiveExtension	The file extension used to archive the original event file.
“Include business object delimiter in the file content” on page 184	IncludeEndBO Delimiter	Specifies whether the delimiter value specified in the SplitCriteria property is sent with the business object content for further processing.
Interval between polling periods	PollPeriod	The length of time that the adapter waits between polling periods

Table 54. Activation specification properties (continued)

Property name		
In the wizard	In the administrative console	Description
Number of times to retry the system connection	RetryLimit	The number of times the adapter tries to reestablish an inbound connection after an error
“Pass only file name and directory, not the content” on page 186	FilePassByReference	Specifies whether the adapter delivers the file content to the export.
“Password used to connect to event data source” on page 186	EP_Password	The password used by event persistence processing to obtain the JDBC database connection from the data source.
Poll quantity	PollQuantity	The number of events that the adapter delivers to the export during each poll period
“Poll subdirectories in event directory” on page 186	PollSubDirectories	Specifies whether the adapter polls the subdirectories within the event directory.
“Retrieve files in sorted order” on page 187	SortEventFiles	The sorting order of polled event files.
“Retrieve files with pattern” on page 187	EventFileMask	The file filter for the event files.
Retry connection on startup	RetryConnectionOnStartup	Controls whether the adapter retries the connection to the local file system if it cannot connect at startup
Retry interval if connection fails	RetryInterval	The length of time that the adapter waits between attempts to establish a new connection after an error during inbound operations
“Specify criteria to split file content” on page 188	SplitCriteria	The delimiter that separates the business objects in the event file or the maximum size of the event file, depending on the value that is set in the Splitting Function Class Name.
“Split function class name” on page 189	SplittingFunctionClassName	Specifies how the event file is to be split, by delimiter or by size.
“Stop the adapter when an error is encountered while polling (StopPollingOnError)” on page 190	StopPollingOnError	Specifies whether the adapter stops polling for events when it encounters an error during polling
“Success file extension for archive” on page 190	SuccessArchiveExtension	The file extension used to archive successfully processed business objects.
“User name used to connect to event data source” on page 190	EP_UserName	The user name used by event persistence processing to obtain the JDBC database connection from the data source.
Rule Editor	ruleTable	The collection of rules used to filter the events



## Archive directory

This property specifies the directory where the adapter archives processed event files.

*Table 55. Archive directory details*

Required	No
Default	None
Property type	String
Usage	You can use a WebSphere Application Server environment variable to represent the archive directory. Specify the name of the environment variable in braces, preceded by a \$ symbol. For example: \${ARCHIVE_DIRECTORY}. See the topic on creating an environment variable in this documentation.
Globalized	Yes
Bidi supported	Yes

## Auto create event table

This property determines whether the event persistence table is created automatically or manually.

*Table 56. Auto create event table details*

Required	No
Possible values	True False
Default	True
Property type	Boolean
Usage	If this value is set to True, the adapter creates the event persistence table. If the value is set to False, the adapter does not create the table and you must manually create it. The recommended setting is True.
Globalized	No

## Bidirectional transformation of event persistence properties

This property determines whether the adapter transforms any of the event persistence properties.

*Table 57. Bidirectional transformation of event persistence properties*

Required	No
Possible values	You can specify a string value, such as VRYNN.
Default	None
Property type	String

Table 57. Bidirectional transformation of event persistence properties (continued)

Usage	The value set on the event persistence bidirectional format property (EP_BiDiFormat) determines the bidirectional transformation. You can specify a string value, such as VRYNN to enable bidirectional transformation of event persistence properties. If the EP_BiDiFormat property is not specified, the adapter will display a null value. <b>Note:</b> You can do bidirectional transformation of only those event properties whose values are set on the bidirectional context enterprise information system (EIS) property.
Globalized	No
Bidi supported	Yes

## Database schema name

This property specifies the schema name of the database used by event persistence processing.

Table 58. Database schema name details

Required	No
Default	None
Property type	String
Globalized	Yes
Bidi supported	Yes

## Type of delivery (DeliveryType)

This property specifies the order in which events are delivered by the adapter to the export.

Table 59. Type of delivery details

Required	No
Possible values	ORDERED UNORDERED
Default	ORDERED
Property type	String
Usage	The following values are supported: <ul style="list-style-type: none"> <li>• ORDERED: The adapter delivers events to the export one at a time.</li> <li>• UNORDERED: The adapter delivers all events to the export at once.</li> </ul>
Globalized	No
Bidi supported	No

## Do not process events that have a timestamp in the future (FilterFutureEvents)

This property specifies whether the adapter filters out future events by comparing the timestamp on each event with the system time.

Table 60. Do not process events that have a timestamp in the future details

Required	Yes
----------	-----

Table 60. Do not process events that have a timestamp in the future details (continued)

Possible values	True False
Default	False
Property type	Boolean
Usage	If set to True, the adapter compares the time of each event to the system time. If the event time is later than the system time, the event is not delivered.  If set to False, the adapter delivers all events.
Globalized	No
Bidi supported	No

### Ensure assured-only event delivery (AssuredOnceDelivery)

This property specifies whether to provide ensure once-only event delivery for inbound events.

Table 61. Ensure assured-only event delivery details

Required	Yes
Possible values	True False
Default	True
Property type	Boolean
Usage	When this property is set to True, the adapter provides assured once event delivery. This means that each event will be delivered once and only once. A value of False does not provide assured once event delivery, but provides better performance.  When this property is set to True, the adapter attempts to store transaction (XID) information in the event store. If it is set to False, the adapter does not attempt to store the information.  This property is used only if the export component is transactional. If it is not, no transaction can be used, regardless of the value of this property.
Globalized	No
Bidi supported	No

### Event directory

This property specifies the directory in the local file system where the event files are stored.

Table 62. Event directory details

Required	Yes
Default	None
Property type	String
Usage	You can use a WebSphere Application Server environment variable to represent the event directory. Specify the name of the environment variable in braces, preceded by a \$ symbol. For example: \${EVENT_DIRECTORY}. See the topic on creating an environment variable in this documentation.
Globalized	Yes
Bidi supported	Yes

## Event recovery data source (JNDI) name

This property specifies the JNDI name of the data source used by event persistence processing to obtain the JDBC database connection.

Table 63. Event recovery data source (JNDI) name details

Required	No
Default	None
Property type	String
Usage	The data source must be created in WebSphere Process Server. Leave this value empty to enable event polling without using the database.
Globalized	Yes
Bidi supported	Yes

## Event recovery table name

This property specifies the name of the table to be used by the adapter for event persistence processing.

Table 64. Event recovery table name details

Required	No
Default	None
Property type	String
Usage	When multiple activation specification instances are used, this value must be unique for each activation specification instance.
Globalized	Yes
Bidi supported	Yes

## Event types to process (EventTypeFilter)

This property contains a delimited list of event types that indicates to the adapter which events it should deliver.

Table 65. Event types to process details

Required	No
Possible values	A comma-delimited (,) list of business object types
Default	null
Property type	String
Usage	Events are filtered by business object type. If the property is set, the adapter delivers only those events that are in the list. A value of null indicates that no filter will be applied and that all events will be delivered to the export.
Example	To receive only events relating to the Customer and Order business objects, specify this value: Customer,Order
Globalized	No
Bidi supported	No

## Retry limit for failed events (FailedEventRetryLimit)

This property specifies the number of times that the adapter attempts to redeliver an event before marking the event as failed.

Table 66. Retry limit for failed events details

Required	No
Possible values	Integers
Default	5
Property type	Integer
Usage	<p>Use this property to control how many times the adapter tries to send an event before marking it as failed. It accepts the following values:</p> <p><b>Default</b></p> <p>If this property is not set, the adapter tries five additional times before marking the event as failed.</p> <p><b>0</b></p> <p>The adapter tries to deliver the event an infinite number of times. When the property is set to 0, the event remains in the event store and the event is never marked as failed.</p> <p><b>&gt; 0</b></p> <p>For integers greater than zero, the adapter retries the specified number of times before marking the event as failed.</p> <p><b>&lt; 0</b></p> <p>For negative integers, the adapter does not retry failed events.</p>
Globalized	No
Bidi supported	No

## Failure file extension for archive

This property specifies the file extension used to archive unsuccessfully processed business objects in the input event file.

Table 67. Failure file extension for archive details

Required	No
Default	fail
Property type	String
Globalized	Yes
Bidi supported	Yes

## File content encoding

This property specifies the encoding of the files read by the adapter.

Table 68. File content encoding details

Required	No
Default	UTF-8
Property type	String

Table 68. File content encoding details (continued)

Usage	You can specify any Java-supported encoding set, such as UTF-8. If the FileContentEncoding property is not specified, the adapter uses the default system encoding.  If the adapter is working with binary event data, set this property to BINARY. If the adapter is working with non-binary event data, such as text or XML, set this property to a valid file encoding value, such as UTF-8.
Globalized	No
Bidi supported	No

### File extension for archive

This property specifies the file extension used to archive the original event file.

Table 69. File extension for archive details

Required	No
Default	original
Property type	String
Usage	This property preserves the entire event file for reference in the event that any of the business objects fail processing.
Globalized	Yes
Bidi supported	Yes

### Include business object delimiter in the file content

This property specifies whether the delimiter value specified in the SplitCriteria property is sent with the business object content for further processing.

Table 70. Include business object delimiter in the file content details

Required	No
Possible values	True False
Default	False
Property type	Boolean
Usage	When this property is set to true, the delimiter value specified in the SplitCriteria property is sent with the business object content for further processing. This property is valid only if event file splitting is based on a delimiter; that is, if the SplittingFunctionClassName property is set to <code>com.ibm.j2ca.extension.utils.filesplit.SplitByDelimiter</code> . <b>Note:</b> This property must be used with a custom data binding that is capable of handling end BO delimiter in the contents. Using it with XMLDataHandler results in failure at the data binding level.
Globalized	No
Bidi supported	No

### Interval between polling periods (PollPeriod)

This property specifies the length of time that the adapter waits between polling periods.

Table 71. Interval between polling periods details

Required	Yes
Possible values	Integers greater than or equal to 0.
Default	2000
Unit of measure	Milliseconds
Property type	Integer
Usage	The poll period is established at a fixed rate, which means that if running the poll cycle is delayed for any reason (for example, if a prior poll cycle takes longer than expected to complete) the next poll cycle will occur immediately to make up for the lost time caused by the delay.
Globalized	No
Bidi supported	No

### Maximum events in polling period (PollQuantity)

This property specifies the number of events that the adapter delivers to the export during each poll period.

Table 72. Maximum events in polling period details

Required	Yes
Default	10
Property type	Integer
Usage	The value must be greater than 0. If this value is increased, more events are processed per polling period and the adapter may perform less efficiently. If this value is decreased, fewer events are processed per polling period and the adapter's performance may improve slightly.
Globalized	No
Bidi supported	No

### Number of times to retry the system connection (RetryLimit)

This property specifies the number of times the adapter tries to reestablish an inbound connection.

Table 73. Number of times to retry the system connection details

Required	No
Possible values	0 and positive integers
Default	0
Property type	Integer
Usage	This property controls how many times the adapter retries the connection if the adapter cannot connect to the local file system to perform inbound processing. A value of 0 indicates an infinite number of retries.  To control whether the adapter retries if it cannot connect to the local file system when it is first started, use the RetryConnectionOnStartup property.
Globalized	Yes
Bidi supported	No

## Pass only file name and directory, not the content

Table 74. Pass only file name and directory, not the content details

Required	No
Possible values	True False
Default	False
Property type	Boolean
Usage	<p>If this property is set to True, the adapter sends the directory name and file name, but does not load the content of the file. The event file is appended with a time stamp and archived to the archive directory. For example, if a.txt is the event file, it is archived as a.txt.yyyy_MM_dd_HH_mm_ss_SSS in the archive directory.</p> <p><b>Note:</b> This property can be used with a custom data binding that does not fail at run time if no content is set; or it can be used in a pass-through scenario. Using this property with XMLDataHandler results in failure at the data binding level, because XMLDataHandler expects content in addition to the file name and directory path.</p>
Globalized	No

## Password used to connect to event data source

This property specifies the password used by event persistence processing to obtain the JDBC database connection from the data source.

Table 75. Password used to connect to event data source details

Required	No
Default	None
Property type	String
Globalized	Yes
Bidi supported	Yes

## Poll subdirectories in event directory

This property specifies whether the adapter polls the subdirectories within the event directory.

Table 76. Poll subdirectories in event directory details

Required	No
Default	False
Property type	Boolean
Usage	<p>When this property is set to True, the adapter polls the files in the event directory and also the files in its subdirectories. When this property is set to False, the adapter polls only the files in the root directory and ignores any subdirectories.</p> <p>During a poll cycle, the adapter first polls the files in the root directory and then polls the files in the subdirectories. It sorts them according to the value set for the SortEventFiles property and processes them according to the value set for the PollQuantity property. It then sends the business objects to the downstream components.</p> <p>When the PollSubDirectories property is set to True and archiving is enabled, all of the polled files, including the files that are polled from the subdirectories, are archived to the archive directory.</p>
Globalized	No



Table 76. Poll subdirectories in event directory details (continued)

Bidi supported	No
----------------	----

### Retrieve files in sorted order

This property specifies the sorting order of polled event files.

Table 77. Retrieve files in sorted order details

Required	No
Possible values	File name - sort in ascending order on file name Time stamp- sort in ascending order on last modified time stamp No sort- not sorted
Default	No sort
Property type	String
Usage	To support globalization, the sorting of file names is provided according to the system locale. The ICU4J package is used to track the locales and the rules corresponding to the locales.
Globalized	No
Bidi supported	No

### Retrieve files with pattern

This property specifies the file filter for the event files.

Table 78. Retrieve files with pattern details

Required	Yes
Default	*.*
Property type	String
Usage	The file filter is a well-qualified valid regular expression that can consist of alphanumeric characters and the wildcard character "*". *. For example, if you specify event*, only file names beginning with event are processed.
Globalized	Yes
Bidi supported	Yes

### Retry EIS connection on startup (RetryConnectionOnStartup)

This property controls whether the adapter attempts to connect again to the local file system if it cannot connect at startup.

Table 79. Retry EIS connection on startup details

Required	No
Possible values	True False
Default	False
Property type	Boolean

Table 79. Retry EIS connection on startup details (continued)

Usage	<p>This property indicates whether the adapter should retry the connection to the local file system if the connection cannot be made when the adapter is started:</p> <ul style="list-style-type: none"> <li>• Set the property to <code>False</code> when you want immediate feedback about whether the adapter can establish a connection to the local file system, for example, when you are building and testing the application that receives events from the adapter. If the adapter cannot connect, the adapter writes log and trace information and stops. The administrative console shows the application status as <code>Stopped</code>. After you resolve the connection problem, start the adapter manually.</li> <li>• Set the property to <code>True</code> if you do not need immediate feedback about the connection. If the adapter cannot connect during startup, it writes log and trace information, and then attempts to reconnect, using the <code>RetryInterval</code> property to determine how frequently to retry and the value of the <code>RetryLimit</code> property to retry multiple times until that value is reached. The administrative console shows the application status as <code>Started</code>.</li> </ul>
Globalized	No
Bidi supported	No

### Retry interval if connection fails (RetryInterval)

When the adapter encounters an error related to the inbound connection, this property specifies the length of time the adapter waits before trying to establish a new connection.

Table 80. Retry interval details

Required	Yes
Default	2000
Unit of measure	Milliseconds
Property type	Integer
Usage	Only positive values are valid. When the adapter encounters an error related to the inbound connection, this property specifies the length of time the adapter waits before trying to establish a new connection.
Globalized	Yes
Bidi supported	No

### Specify criteria to split file content

This property specifies either the delimiter that separates the business objects in the event file or the maximum size of the event file.

Table 81. Specify criteria to split file content details

Required	No
Default	0
Property type	String

Table 81. Specify criteria to split file content details (continued)

Usage	<p>This property specifies the delimiter that separates the business objects in the event file or the maximum size of the event file. The value of this property is determined by the value that is set in the SplittingFunctionClassName property:</p> <ul style="list-style-type: none"> <li>• If the SplittingFunctionClassName property is set to <code>com.ibm.j2ca.extension.utils.filesplit.SplitByDelimiter</code>, the SplitCriteria property must contain the delimiter that separates the business objects in the event file.</li> <li>• If the SplittingFunctionClassName property is set to <code>com.ibm.j2ca.utils.filesplit.SplitBySize</code>, the SplitCriteria property must contain a valid number that represents the maximum file size in bytes. If the event file size is greater than this value, it is split into chunks of this value and that number of chunks are posted. If the event file size is less than this value, the entire event file is posted.</li> </ul> <p>If the SplitCriteria property value is set to 0, file splitting is disabled.  <b>Note:</b> During an inbound pass-through scenario, if file splitting is based on size and the FilePassByReference property is enabled, the event files are not split into chunks.  <b>Note:</b> For input files that contain multiple COBOL copybook records, in order to enable file splitting by size you must provide the correct length of each record. To determine the size of each record, use the following method:</p> <ol style="list-style-type: none"> <li>1. Open the Business Object in a text editor.</li> <li>2. Look for the complex type tag with the business object name value in the name attribute. In the example that follows, the business object name is DFHCOMMAREA.</li> <li>3. Locate a namespace-appended tag called aggregateInstanceTD and use the value for the attribute contentSize. In this example, the value is 117. This is the size of each record of type DFHCOMMAREA.</li> </ol> <pre>&lt;complexType name="DFHCOMMAREA"&gt; &lt;annotation&gt; &lt;appinfo source="http://www.ibm.com/cam/2005/typedescriptor"&gt; &lt;td:typeDescriptorCT&gt; &lt;td:aggregateInstanceTD accessor="readWrite" attributeInBit="false" contentSize="117" offset="0" size="117"&gt;</pre>
Globalized	Yes
Bidi supported	Yes

## Split function class name

This property determines how the event file is to be split.

Table 82. Split function class name details

Required	No
Possible values	<code>com.ibm.j2ca.extension.utils.filesplit.SplitByDelimiter</code> – Files are split based on a delimiter that separates business objects in the event file <code>com.ibm.j2ca.utils.filesplit.SplitBySize</code> – Files are split based on the size of the event file
Default	<code>com.ibm.j2ca.utils.filesplit.SplitBySize</code>
Property type	String
Usage	<p>The delimiter or file size is set in the SplitCriteria property.  <b>Note:</b> If the EventContentType property is null, the SplittingFunctionClassName property is automatically set to <code>com.ibm.j2ca.utils.filesplit.SplitBySize</code>.</p>
Globalized	No
Bidi supported	No

## Stop the adapter when an error is encountered while polling (StopPollingOnError)

This property specifies whether the adapter will stop polling for events when it encounters an error during polling.

Table 83. Stop the adapter when an error is encountered while polling details

Required	No
Possible values	True False
Default	False
Property type	Boolean
Usage	If this property is set to True, the adapter stops polling when it encounters an error.  If this property is set to False, the adapter logs an exception when it encounters an error during polling and continues polling.
Globalized	No
Bidi supported	No

## Success file extension for archive

This property specifies the file extension used to archive successfully processed business objects.

Table 84. Success file extension for archive details

Required	No
Default	success
Property type	String
Globalized	Yes
Bidi supported	Yes

## User name used to connect to event data source

This property specifies the user name used by event persistence processing to obtain the JDBC database connection from the data source.

Table 85. User name used to connect to event data source details

Required	No
Default	None
Property type	String
Globalized	Yes
Bidi supported	Yes

## Rule based event filtering (ruleTable)

This property is used to filter event files based on a set of rules

Table 86. Rules based event filtering

Required	Optional
----------	----------

Table 86. Rules based event filtering (continued)

Default	None
Property type	String
Usage	During an inbound processing, if the value in the rule table is specified, then the event files are fetched after filtering, based on the specified rules before polling those event files.
Globalized	Yes
Bidi supported	No

### Related concepts

“WebSphere Application Server environment variables” on page 24  
 WebSphere Application Server environment variables can be used in the external service wizard to specify directory values.

“Creating the required local folders” on page 48

Before you create inbound or outbound modules, you must create folders on the local file system for events and output. You can optionally create folders for staging and archiving.

### Related tasks

“Defining WebSphere Application Server environment variables” on page 51  
 Use the administrative console of WebSphere Process Server or WebSphere Enterprise Service Bus to define WebSphere Application Server environment variables.

“Setting deployment and runtime properties” on page 97

After you have decided whether your module will be used for outbound or inbound communication with the enterprise information system (local file system), you must configure the activation specification properties, which hold the inbound event processing configuration information for the export.

## Resource adapter properties

The resource adapter properties control the general operation of the adapter, such as specifying the namespace for business objects. You set the resource adapter properties using the external service wizard when you configure the adapter. After deploying the adapter, use the administrative console to change these properties.

The following properties for logging and tracing are no longer required in version 6.1.0. They are visible from the administrative console for compatibility with previous versions.

- LogFileMaxSize
- LogFileName
- LogNumberOfFiles
- TraceFileMaxSize
- TraceFileName
- TraceNumberOfFiles

The following table lists the resource adapter properties and their purpose. A complete description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see “Guide to information about properties” on page 154.

Table 87. Resource adapter properties for the Adapter for Flat Files

Name		Description
In the wizard	In the administrative console	
Adapter ID	AdapterID	Identifies the adapter instance for PMI events and for logging and tracing
Disguise user data as "XXX" in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing strings of X's instead of user data in log and trace files
(Not available)	Enable HA support	Do not change this property.
(Not available)	LogFileMaxSize	Deprecated
(Not available)	LogFilename	Deprecated
(Not available)	LogNumberOfFiles	Deprecated
(Not available)	TraceFileMaxSize	Deprecated
(Not available)	TraceFileName	Deprecated
(Not available)	TraceNumberOfFiles	Deprecated

## Adapter ID (AdapterID)

This property identifies a specific deployment, or instance, of the adapter.

Table 88. Adapter ID details

Required	Yes
Default	001
Property type	String
Usage	<p>This property identifies the adapter instance in log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, FFRA, to form the component name used by the Log and Trace Analyzer tool. For example, if the adapter ID property is set to 001, the component ID is FFRA001.</p> <p>If you run multiple instances of the same adapter, make sure that the first nine characters of the adapter ID property are unique for each instance so that you can correlate log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate log and trace information to a particular instance of an adapter.</p> <p>To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for Flat Files to 001 and 002. The component IDs for those instances, FFRA001 and FFRA002, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. For example, suppose you set the adapter ID properties of two instances to Instance01 and Instance02. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to FFRAInstance0.</p> <p>For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, make sure you set them consistently, to prevent inconsistent marking of the log and trace entries.</p>
Globalized	Yes

Table 88. Adapter ID details (continued)

Bidi supported	No
----------------	----

### Disguise user data as "XXX" in log and trace files (HideConfidentialTrace) property

This property specifies whether to replace user data in log and trace files with a string of X's to prevent unauthorized disclosure of potentially sensitive data.

Table 89. Disguise user data as "XXX" in log and trace files details

Required	No
Possible values	True False
Default	False
Property type	Boolean
Usage	If you set this property to True, the adapter replaces user data with a string of X's when writing to log and trace files.  For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, make sure you set them consistently, to prevent inconsistent marking of the log and trace entries.
Globalized	No
Bidi supported	No

### Enable high availability support (enableHASupport)

Do not change this property. It must be set to true.

---

## Globalization

WebSphere Adapter for Flat Files is a globalized application that can be used in multiple linguistic and cultural environments. Based on character set support and the locale of the host server, the adapter delivers message text in the appropriate language. The adapter supports bidirectional script data transformation between integration components.

### Globalization and bidirectional data transformation

The adapter is globalized to support single- and multi-byte character sets and deliver message text in the specified language. The adapter also performs bidirectional script data transformation, which refers to the task of processing data that contains both right-to-left (Hebrew or Arabic, for example) and left-to-right (a URL or file path, for example) semantic content within the same file.

#### Globalization

Globalized software applications are designed and developed for use within multiple linguistic and cultural environments rather than a single environment. WebSphere Adapters, WebSphere Integration Developer, WebSphere Process Server

or WebSphere Enterprise Service Bus are written in Java. The Java runtime environment within the Java virtual machine (JVM) represents data in the Unicode character code set. Unicode contains encodings for characters in most known character code sets (both single- and multi-byte). Therefore, when data is transferred between these integration system components, there is no need for character conversion.

To log error and informational messages in the appropriate language and for the appropriate country or region, the adapter uses the locale of the system on which it is running.

## Bidirectional script data transformation

Languages such as Arabic and Hebrew are written from right to left, yet they contain embedded segments of text that are written left to right, resulting in bidirectional script. When software applications handle bidirectional script data, standards are used to display and process it. Bidirectional script data transformation applies only to string type data. WebSphere Process Server or WebSphere Enterprise Service Bus uses the Windows standard format, but applications or file systems that exchange data with the server might use a different format. The adapter transforms bidirectional script data passed between the two systems so that it is accurately processed and displayed on both sides of a transaction. It transforms the script data by using a set of properties that defines the format of script data, as well as properties that identify content or metadata to which transformation applies.

### Bidirectional script data formats

WebSphere Process Server or WebSphere Enterprise Service Bus use the bidirectional format of ILYNN (implicit, left-to-right, on, off, nominal). This is the format used by Windows. If an enterprise information system uses a different format, the adapter converts the format before introducing the data to WebSphere Process Server or WebSphere Enterprise Service Bus.

Five attributes comprise bidirectional format. When you set bidirectional properties, you assign values for each of these attributes. The attributes and settings are listed in the following table.

Table 90. Bidirectional format attributes

Letter position	Purpose	Values	Description	Default setting
1	Order schema	I	Implicit (Logical)	I
		V	Visual	
2	Direction	L	Left-to-Right	L
		R	Right-to-Left	
		C	Contextual Left-to-Right	
		D	Contextual Right-to-Left	
3	Symmetric Swapping	Y	Symmetric swapping is on	Y
		N	Symmetric swapping is off	



Table 90. Bidirectional format attributes (continued)

Letter position	Purpose	Values	Description	Default setting
4	Text Shaping	S	Text is shaped	N
		N	Text is not shaped (Nominal)	
		I	Initial shaping	
		M	Middle shaping	
		F	Final shaping	
		B	Isolated shaping	
5	Numeric Shaping	H	National (Hindi)	N
		C	Contextual shaping	
		N	Numbers are not shaped (Nominal)	

### Bidirectional properties that identify data for transformation

To identify business data for transformation, set the BiDiContextEIS property. Do this by specifying values for each of the five bidirectional format attributes (listed in the preceding table) for the property. The BiDiContextEIS property can be set for the managed connection factory and for the activation specification.

To identify event persistence data subject for transformation, set the EP\_BiDiFormat property. The value for the EP\_BiDiFormat property is set with the value you specified in the BiDiContextEIS property. The EP\_BiDiFormat property can be set for the activation specification.

To identify application-specific data for transformation, annotate the BiDiContextEIS property and the BiDiMetadata property within a business object. Do this by using the business object editor within WebSphere Integration Developer to add the properties as application-specific elements of a business object.

#### Related reference

“Activation specification properties” on page 176

Activation specification properties hold the inbound event processing configuration information for an export. You set activation specification properties through either the external service wizard or the administrative console.

“Managed connection factory properties” on page 159

Managed connection factory properties specify information the adapter needs at run time for outbound communication with the local file system.

## Bidirectional transformation in business objects

For outbound processing, you can modify the business objects to enable the bidirectional transformation of the wrapper properties in the WebSphere Adapter for Flat Files business object and the data in content-specific or generic business objects.

You must add an annotation to the complex type of the business object to specify the bidirectional formatting attributes in the files for the following business objects:

- For the generic business object, change the FlatFile.xsd file.
- For the user-defined business object, change the custom wrapper (for example, the CustomWrapper.xsd file and Customer.xsd).
- For the UnstructuredContent business object, change the UnstructuredContent.xsd.

The following sections include annotations that can serve as examples.

## Bidirectional formatting attributes of the business object

The following annotation, which contains the bidirectional context information, applies to all the attributes in the Flat Files business objects. The FlatFileBaseDataBinding uses the bidirectional information in the element BiDiContext to transform all the attributes.

```
<xsd:complexType name="Customer">
<xsd:annotation>
  <xsd:appinfo
    source="http://www.ibm.com/xmlns/prod/websphere/j2ca/datatrans
formation/databindingm
apping">
    <dtm:DataBindingMapping
      xsi:type="dtm:DataBindingMapping"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dtm="http://www.ibm.com/xmlns/prod/websphere/j2ca/da
tatransformation/databindingmapping">
        <BiDiContext>
          <orientation>rtl</orientation>
          <textShape>nominal</textShape>
          <orderingScheme>visual</orderingScheme>
          <symmetricSwapping>true</symmetricSwapping>
          <numeralShapes>nominal</numeralShapes>
        </BiDiContext>
      </dtm:DataBindingMapping>
    </xsd:appinfo>
  </xsd:annotation>
```

## Bidirectional formatting attributes of the wrapper

You can add an annotation to the wrapper of a user-defined type business object. The annotation in the wrapper business objects such as the generic (FlatFile) and the user-defined type (CustomerWrapper) is used to do bidirectional transformation of wrapper attributes. The content-specific business objects that are used inside the wrapper business objects are not transformed using annotation in the wrapper business objects. (To transform content-specific business objects, you must edit the respective business object definition to add the annotation shown in the previous example for bidirectional formatting of attributes of the business object.

The following annotation is an example for the wrapper:

```
<complexType name="CustomerWrapper">
<annotation>
  <appinfo
    source="http://www.ibm.com/xmlns/prod/websphere/j2ca/
datatransformation/databindingmapping">
    <dtm:DataBindingMapping
      xsi:type="dtm:DataBindingMapping"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dtm="http://www.ibm.com/xmlns/prod/websphere/j2ca/
datatransformation/databindingmapping">
      <BiDiContext>
```

```

        <orientation>rtl</orientation>
        <textShape>nominal</textShape>
        <orderingScheme>visual</orderingScheme>
        <symmetricSwapping>true</symmetricSwapping>
        <numeralShapes>nominal</numeralShapes>
    </BiDiContext>
</dtm:DataBindingMapping>
</appinfo>
</annotation>

```

## Properties enabled for bidirectional data transformation

Bidirectional data transformation properties enforce the correct format of bidirectional script data exchanged between an application or file system and integration tools and runtime environments. Once these properties are set, bidirectional script data is correctly processed and displayed in WebSphere Integration Developer and WebSphere Process Server or WebSphere Enterprise Service Bus.

### Managed connection factory properties

The following managed connection factory properties control bidirectional script data transformation:

- FileSequenceLog
- OutputDirectory
- OutputFilename
- StagingDirectory

### Activation specification properties

The following activation specification properties control bidirectional script data transformation:

- ArchiveDirectory
- EventDirectory
- EventFileMask
- FailedArchiveExtension
- OriginalArchiveExtension
- SplitCriteria
- SuccessArchiveExtension

### Deployment Descriptor configuration properties

The following deployment descriptor configuration properties control bidirectional script data transformation:

- EPDatabasePassword
- EPDatabaseSchemaName
- EPDatabaseUsername
- EPDataSourceJNDIName
- EPEventTableName

### Business object wrapper properties

The following business object wrapper properties control bidirectional script data transformation:

- DirectoryPath
- FileName
- IncludeEndBODelimiter
- StagingDirectory
- ArchiveDirectoryForDeleteOnRetrieve
- ChunkFileName

---

## Adapter messages

View the messages issued by WebSphere Adapter for Flat Files at the following location.

Link to messages: <http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/topic/com.ibm.wbit.620.help.messages.doc/messages.html>

The displayed Web page shows a list of message prefixes. Click a message prefix to see all the messages with that prefix:

- Messages with the prefix CWYFF are issued by WebSphere Adapter for Flat Files
- Messages with the prefix CWYBS are issued by the adapter foundation classes, which are used by all the adapters

---

## Related information

The following information centers, IBM Redbooks, and Web pages contain related information for the WebSphere Adapter for Flat Files.

### Samples and tutorials

To help you use WebSphere Adapters, samples and tutorials are available from the Business Process Management Samples and Tutorials Web site. You can access the samples and tutorials in either of the following ways:

- From the welcome page that opens when you start WebSphere Integration Developer. To see samples and tutorials for WebSphere Adapter for Flat Files, click **Retrieve**. Then browse the displayed categories to make your selections.
- At this location on the Web: <http://publib.boulder.ibm.com/bpcsamp/index.html>.

### Information resources

- The WebSphere Business Process Management information resources Web page includes links to articles, Redbooks, documentation, and educational offerings to help you learn about WebSphere Adapters: <http://www14.software.ibm.com/webapp/wsbroker/redirect?version=pix&product=wps-dist&topic=bpmroadmaps>
- The WebSphere Adapters library page includes links to all versions of the documentation: <http://www.ibm.com/software/integration/wbiadapters/library/infocenter/>

### Information about related products

- WebSphere Business Process Management, version 6.2, information center, which includes WebSphere Process Server, WebSphere Enterprise Service Bus, and WebSphere Integration Developer information: <http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp>

- WebSphere Adapters, version 6.1.x, information center: <http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.wbit.612.help.adapter.emd.ui.doc/topics/tcreatecmps.html>
- WebSphere Business Integration Adapters information center: [http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/index.jsp?topic=/com.ibm.wbi\\_adapters.doc/welcome\\_adapters.htm](http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/index.jsp?topic=/com.ibm.wbi_adapters.doc/welcome_adapters.htm)

### **developerWorks® resources**

- WebSphere Adapter Toolkit
- WebSphere business integration zone

### **Support and assistance**

- WebSphere Adapters technical support: <http://www.ibm.com/software/integration/wbiadapters/support/>
- WebSphere Adapters technotes: <http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm>. In the **Product category** list, select the name of the adapter and click **Go**.



---

## 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 document 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 (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:

IBM Corporation  
Department 2Z4A/SOM1  
294 Route 100  
Somers, NY 10589-0100  
U.S.A.

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.

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: (c) (your company name) (year). Portions of



this code are derived from IBM Corp. Sample Programs. (c) 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](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (<sup>®</sup> or <sup>™</sup>), indicating US 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 complete and current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

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

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

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.

This product includes software developed by the Eclipse Project (<http://www.eclipse.org>).



---

# Index

## A

- activation specification properties
  - Archive directory 176
  - Auto create event table 176
  - Database schema name 176
  - Delivery type 176
  - Do not process events that have a timestamp in the future 176
  - Ensure once only event delivery 176
  - Event directory 176
  - Event recovery data source (JNDI) name 176
  - Event recovery table name 176
  - Event types to process 176
  - Failure file extension for archive 176
  - File content encoding 176
  - File extension for archive 176
  - Include business object delimiter in the file content 176
  - Interval between polling periods 176
  - Number of times to retry the system connection 176
  - Pass only file name and directory, not the content 176
  - Password used to connect to event data source 176
  - Poll quantity 176
  - Poll subdirectories in event directory 176
  - Retrieve files in sorted order 176
  - Retrieve files with pattern 176
  - Retry connection on startup 176
  - Retry interval if connection fails 176
  - setting in administrative console 127, 132
  - Specify criteria to split file content 176
  - Split function class name 176
  - Stop the adapter when an error is encountered while polling 176
  - Success file extension for archive 176
  - User name used to connect to event data source 176
- adapter application
  - starting 133
  - stopping 134
- Adapter for Flat Files
  - administering 123
- Adapter for Flat Files module
  - exporting as EAR file 120
  - installing EAR file on server 121
  - starting 133
  - stopping 134
- adapter messages 198
- adapter patterns wizard 75
- adapter performance 134
- adapter technotes 199
- append 3
- Append 3
- artifacts, generating 94

## B

- backward compatibility
  - project interchange files 35
  - projects 35
- business faults 145
- business integration adapters to JCA-compliant adapters 36
- business object information 149

- business object, predefining 49, 54
- business objects 3, 22
  - attribute properties 152
  - naming conventions 152
  - structure 149
- business objects, converting COBOL copybook files into 66
- business objects, converting into COBOL copybook files 57

## C

- CEI (Common Event Infrastructure) 138
- clustered environment
  - deploying in 30
  - description 30
  - inbound processes 31
  - outbound processes 32
- COBOL copybook files, converting from business objects 57
- COBOL copybook files, converting into business objects 66
- Common Event Infrastructure (CEI) 138
- compatibility matrix 1
- confidential data, disguising 27
- confidential tracing 27
- configuring
  - logging 142
  - Performance Monitoring Infrastructure (PMI) 135
  - tracing 142
- configuring the data binding, inbound 106
- configuring the data binding, outbound 88
- connection properties, inbound 97
- connection properties, outbound 82
- create 3
- Create 4
- custom properties
  - activation specification 127, 132
  - managed connection factory 125, 130
  - resource adapter 123, 129

## D

- data transformation (inbound) 22
- data transformation (outbound) 7
- debugging
  - org.xml.sax.SAXParseException exception 146
  - self-help resources 147
  - XAResourceNotAvailableException exception 146
- delete 3
- Delete 5
- deployment
  - environments 115
  - options 28
  - to production environment 119
  - to test environment 115
- developerWorks 199
- developerWorks resources, WebSphere Adapters 198

## E

- EAR file
  - exporting 120
  - installing on server 121

- education, WebSphere Adapters 198
- embedded adapter
  - activation specification properties, setting 127
  - considerations for using 30
  - description 28
  - managed connection factory properties, setting 125
  - resource adapter properties, setting 123
- EmbeddedNameFunctionSelector 18
- enableHASupport property 31
- Event archival values 17
- event store
  - overview 16
  - structure 17
- exceptions
  - org.xml.sax.SAXParseException 146
  - XAResourceNotAvailableException 146
- exists 3
- Exists 5
- exporting module as EAR file 120
- external service
  - generating inbound artifacts 110
  - overview 25
- external service connection properties
  - Bidi format string 156, 173
  - Data binding 156, 173
  - Function selector 156, 173
  - Log file output location 156, 173
  - Logging level 156, 173
  - NameSpace 156, 173
  - Operation name 156, 173
  - Processing Direction 156, 173
- external service discovery, connection properties 82, 97
- external service wizard
  - starting 80

## F

- faults
  - description 145
- FFDC (first-failure data capture) 144
- file splitting
  - delimiter-based 9, 19
  - size-based 9, 19
- FilenameFunctionSelector 18
- files
  - SystemOut.log log file 143
  - trace.log trace file 143
- first-failure data capture (FFDC) 144
- function selector 18

## G

- generating artifacts 94

## H

- hardware and software requirements 1
- hardware requirements 1
- high-availability environment
  - deploying in 30
  - description 30
  - inbound processes 31
  - outbound processes 32

## I

- IBM WebSphere Adapter Toolkit 199
- implementation, Java 116
- inbound configuration properties 171
- installing EAR file 121
- interaction specification properties
  - Archive directory for retrieve operation 166
  - changing 113
  - Create a new file if the file does not exist 166
  - Default target file name 166
  - Delete the file after retrieve operation 166
  - Delimiter between business objects in the file 166
  - File content encoding 166
  - Generate a unique file 166
  - Output directory 166
  - Specify criteria to split file content 166
  - Split function class name 166
  - Staging directory 166

## J

- Java implementation 116

## L

- list 3
- List 5
- Log Analyzer 142
- Log and Trace Analyzer, support for 141
- log and trace files 141
- log files
  - changing file name 143
  - disabling 142
  - enabling 142
  - level of detail 142
  - location 144
- logging
  - configuring properties with administrative console 142

## M

- managed (J2C) connection factory properties
  - setting in administrative console 125, 130
- managed connection properties
  - Default target file name 159
  - Output directory 159
  - Sequence file 159
  - Staging directory 159
- matrix, compatibility 1
- messages, adapter 198
- migration 36
  - WebSphere InterChange Server migration wizard 38
- migration considerations 32
- migration overview
  - WebSphere InterChange Server applications 37
- module, creating 49
- monitoring performance 134

## N

- naming conventions for business objects 152

## O

- operations 3, 4, 5, 6
- org.xml.sax.SAXParseException 146
- outbound 3, 4, 5, 6
  - processing 3
  - supported operations 3
- outbound configuration properties 154
- outbound operations
  - append 3
  - create 3
  - delete 3
  - exists 3
  - list 3
  - overwrite 3
  - retrieve 3
- overwrite 3
- Overwrite 5

## P

- package files for adapters 143
- patterns 75
- Performance Monitoring Infrastructure (PMI)
  - configuring 135
  - description 134
  - viewing performance statistics 137
- performance statistics 137
- PMI (Performance Monitoring Infrastructure)
  - configuring 135
  - description 134
  - viewing performance statistics 137
- problem determination
  - org.xml.sax.SAXParseException exception 146
  - self-help resources 147
  - XAResourceNotAvailableException exception 146
- project interchange (PI) file
  - updating without migrating 35
- project, creating 80
- properties
  - activation specification 127, 132
  - configuration properties
    - inbound 171
    - outbound 154
  - inbound configuration 171
  - managed (J2C) connection factory 125, 130
  - outbound configuration 154
  - resource adapter 123, 129

## R

- RAR (resource adapter archive) file
  - description 119
  - installing on server 119
- Redbooks, WebSphere Adapters 198
- related information 198
- related products, information 198
- required local folders 48
- requirements, hardware and software 1
- resource adapter archive (RAR) file
  - description 119
  - installing on server 119
- resource adapter properties
  - Adapter ID 164, 191
  - details 164, 191
  - Enable HA support 164, 191
  - setting in administrative console 123, 129

- retrieve 3
- Retrieve 6
- Retry limit property 185
- road map for configuring the module 47
- road map for migrating
  - WebSphere InterChange Server applications 36
- runtime environment
  - deploying EAR file to 119

## S

- samples 45
- security 27
  - disguising sensitive data 27
- self-help resources 147
- sensitive data, disguising 27
- software requirements 1
- stand-alone adapter
  - activation specification properties, setting 132
  - considerations for using 30
  - description 28
  - managed connection factory properties, setting 130
  - resource adapter properties, setting 129
- starting adapter applications 133
- stopping adapter applications 134
- support
  - overview 141
  - self-help resources 147
  - technical 199
- supported operations 3, 4, 5, 6
- SystemOut.log file 143

## T

- target component 115
- technical overview 2
- technical support 199
- technotes 1, 147, 199
- technotes, WebSphere Adapters 198
- test environment
  - adding module to 117
  - deploying to 115, 117
  - testing modules 118
- trace files
  - changing file name 143
  - disabling 142
  - enabling 142
  - level of detail 142
  - location 144
- trace.log file 143
- tracing
  - configuring properties with administrative console 142
- troubleshooting
  - org.xml.sax.SAXParseException exception 146
  - overview 141
  - self-help resources 147
  - XAResourceNotAvailableException exception 146
- tutorials 45

## U

- unique file names, generating 11

## W

- WebSphere Adapter for Flat Files 159, 164, 191
  - inbound processing 12
  - introduction 1
  - outbound processing 3
  - planning adapter implementation 27
  - security 27
  - technical overview 2
- WebSphere Adapters, version 6.0, information 198
- WebSphere Adapters, version 6.0.2.x, information 198
- WebSphere Application Server environment variables 24, 48, 159, 176
- WebSphere Application Server environment variables, defining 51
- WebSphere Application Server information 198
- WebSphere business integration adapters 36
- WebSphere Business Integration Adapters information 198
- WebSphere Business Process Management, version 6.1.x, information 198
- WebSphere Enterprise Service Bus
  - information 198
- WebSphere Extended Deployment 31
- WebSphere Integration Developer
  - information 198
  - starting 49, 54, 80
  - test environment 115
- WebSphere Process Server
  - information 198
- WebSphere Process Server or WebSphere Enterprise Service Bus
  - deploying to 119
- wiring components 115

## X

- XAResourceNotAvailableException 146





Printed in USA