WebSphere® Adapters

Version 6 Release 1

WebSphere Adapter for Flat Files User Guide Version 6 Release 1

WebSphere® Adapters

Version 6 Release 1

WebSphere Adapter for Flat Files User Guide
Version 6 Release 1

Note

Before using this information and the product it supports, read the information in "Notices" on page 153.

16 January 2007

This edition applies to version 6, release 1, 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.

Contents

Chapter 1. Overview of WebSphere Adapter for Flat Files 1
New in this release 1
Hardware and software requirements 2
Technical overview of the Adapter for Flat Files . . 3
  Outbound processing 4
  Inbound processing. 13
  Business objects 22
  The external service wizard 23
Standards compliance 24
  Accessibility 24
  Internet Protocol Version 6 (IPv6) 25

Chapter 2. Planning for adapter implementation 27
Before you begin 27
Security 27
Deployment options 27
WebSphere Adapters in clustered environments . . 30
Migrating to version 6.1.0 31
  Migration considerations 31
  Performing the migration. 33
  Updating but not migrating a version 6.0.2 project 34

Chapter 3. Samples and tutorials . . . 35

Chapter 4. Configuring the module for deployment. 37
Roadmap for configuring the module. 37
Creating the required local folders. 38
Creating the module 39
Defining business objects 41
Creating a simple service with the adapter pattern wizard 43
Creating the project. 48
Configuring the module for outbound processing . 50
  Setting deployment and runtime properties. . . 50
  Selecting the operation and data type. 53
  Configuring the data binding 56
  Configuring data handlers 57
  Setting interaction properties and generating the service 61
Configuring the module for inbound processing . . 63
  Setting deployment and runtime properties. . . 63
  Selecting the operation and data type. 66
  Configuring the data binding 68
  Configuring data handlers 69
  Setting deployment properties and generating the service 72

Chapter 5. Changing interaction specification properties using the assembly editor 75

Chapter 6. Deploying the module . . . 77
Deployment environments 77
Deploying the module for testing 77
  Generating and wiring a target component for testing inbound processing 77
  Adding the module to the server 79
  Testing the module for outbound processing using the test client. 80
Deploying the module for production 80
  Installing the RAR file (for modules using stand-alone adapters only) 80
  Exporting the module as an EAR file 82
  Installing the EAR file 84

Chapter 7. Administering the adapter module 87
Changing configuration properties for embedded adapters 87
  Setting resource adapter properties for embedded adapters 87
  Setting managed (J2C) connection factory properties for embedded adapters 89
  Setting activation specification properties for embedded adapters. 91
Changing configuration properties for stand-alone adapters 93
  Setting resource adapter properties for stand-alone adapters 93
  Setting managed (J2C) connection factory properties for stand-alone adapters 94
  Setting activation specification properties for stand-alone adapters 96
Starting the application that uses the adapter . . . 97
Stopping the application that uses the adapter. . . 97
Monitoring performance using Performance Monitoring Infrastructure. 98
  Configuring Performance Monitoring Infrastructure. 98
  Viewing performance statistics 100
Enabling tracing with the Common Event Infrastructure (CEI) 101
Troubleshooting and support 102
  Configuring logging and tracing 102
  First-failure data capture (FFDC) support . . . 105
  Business faults 105
  XAResourceNotAvailableException 109
  org.xml.sax.SAXParseException 110
  Self-help resources. 111

Chapter 8. Reference information . . . 113
Business object information. 113
  Business object structures 113
  Attribute properties 116
  Naming conventions 116
Custom file splitting 117

Outbound configuration properties 118
Connection properties for the wizard 119
Managed connection factory properties 122
Resource adapter properties 125
Interaction specification properties 126
Inbound configuration properties. 131
Connection properties for the wizard 132
Activation specification properties 135
Resource adapter properties 146
Globalization 148
Globalization and bidirectional data transformation 148
Properties enabled for bidirectional data transformation 149
Adapter messages 150
Related information 151

Notices 153
Programming interface information 155
Trademarks and service marks 155

Index 157

Chapter 1. Overview of WebSphere Adapter for Flat Files

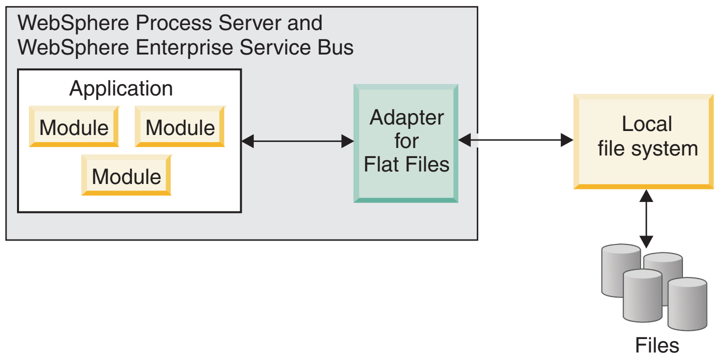
With WebSphere Adapter for Flat Files, you can create integrated processes that include the exchange of data with the local file system, without special coding.

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

New in this release

WebSphere Adapter for Flat Files, Version 6.1.0 provides enhancements to the adapter. This release also includes some deprecated features.

Updates to this information are made available at the WebSphere Adapters product support Web site. To read updated or additional information, see: http://www.ibm.com/software/integration/wbiadapters/support/.

A deprecated feature is one that is supported but no longer recommended and that might become obsolete. For a list of features from earlier versions of Adapter for Flat Files that have been deprecated in version 6.1.0, see “Deprecated features” on page 32.

New in version 6.1.0:

- New name, usability improvements, and functional enhancements in the enterprise service discovery wizard. The wizard has been renamed the external service wizard, and includes functional and usability improvements to make it easier for you to build services to use with the adapter. The wizard helps you access predefined data bindings, data handlers, and function selectors to automate the conversion between files and business objects.
- Adapter pattern wizard provides a quick and easy way of creating a simple service with the adapter.
- Expanded operating system support. For more information about the operating systems that are supported in version 6.1.0, see the hardware and software requirements for WebSphere Adapter for Flat Files on the IBM® Web site at http://www.ibm.com/support/docview.wss?uid=swg27006249.
- The adapter RAR file is available in WebSphere Integration Developer; you do not need to install it separately. The wizard automatically copies the adapter files into the project for you.
- The adapter documentation is located on the WebSphere Integration Developer Information Center, in the Configuring and using adapters section.
- Support for node-level, or stand-alone, deployment of the adapter.
- The business graph that contains each business object in version 6.0.2 is now optional. You need a business graph only for modules whose business objects were created in version 6.0.2 or for new version 6.1.0 modules that use the ApplyChanges outbound operation.
- Support for a first-failure data capture (FFDC) construct that can be contained in a WebSphere Application Server symptom database to provide information and suggested actions to assist a diagnostic module in customizing the data that is logged.
- Support for business faults. The adapter now generates business faults for business exceptions. This lets you easily assign a corrective action for those error conditions.
- Support for configuring data binding properties.
- Support for data transformation on the outbound Retrieve operation.
- Additional options for outbound processing that provide the ability to:
  - Create unique file names on Create and Append operations
  - Generate unique sequence numbers for files on Create operations
  - Delete files after they are retrieved
  - Archive retrieved files before deleting them
- Platform-independent support for Windows® and UNIX® new lines as delimiters.
- Support for IPv6 addresses.
- Support for event persistence during inbound processing using an in-memory representation of the event store.

Hardware and software requirements

The hardware and software requirements for WebSphere Adapters are documented on the IBM Web site at the location below.

Hardware and software requirements for WebSphere Adapters: 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 the link for the compatibility matrix under **Planning upgrades**: http://www.ibm.com/software/integration/wbiadapters/support/.
- Technotes for WebSphere Adapters document workarounds and additional information 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. 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 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.

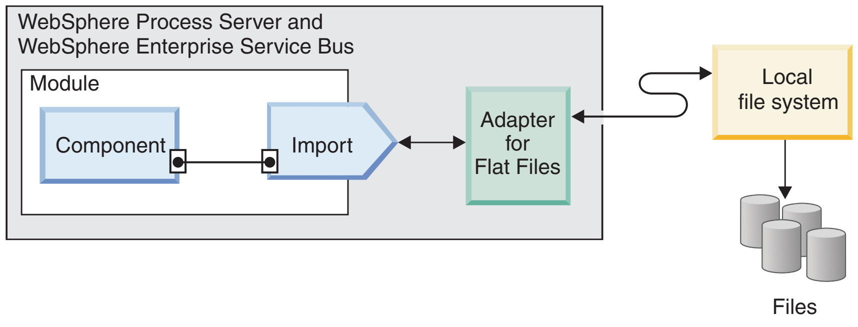


Figure 1. Outbound processing

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.

Append

If you specify "Output required" 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.

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.

Create operation:

The Create operation creates a file with the specified name.

Create

If you specify ″Output required″ 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`.

If the FileSequenceLog managed connection property is specified, the adapter appends a sequence number to the output file name specified in the request. 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, a new sequence starting with 1 is generated for Order.txt. 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 Customer*n* 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 it receives a request to create a file, the adapter will then check the file sequence log to see whether a file with that name already exists. If one does, the adapter will use 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.

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.

The sequence number will continue to increment after an adapter restart.

Delete operation:

The Delete operation deletes a specified file.

Delete

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.

Overwrite

If you specify ″Output required″ 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. 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. If the GenerateUniqueFile property is set to `true`, the adapter generates a unique file and ignores the value specified in the Filename property.

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

Retrieve operation:

The Retrieve operation retrieves the content of the specified file and returns it in the form of a business object.

Retrieve

The content of the file is retrieved and returned in the form of a generic or a content-specific business object. The file content is split according to the SplittingFunctionClassName and SplitCriteria properties defined in the interaction specification. If a data handler is configured, the adapter returns a content-specific business object; otherwise, it returns a generic business object.

To specify that the file be deleted once it has been retrieved, set the DeleteOnRetrieve property in the interaction specification. To specify that the file be archived before it is deleted, set the ArchiveDirectoryForDeleteOnRetrieve property.

If the file that is specified in the Retrieve request does not exist, the adapter generates a `RecordNotFoundException` error.

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.

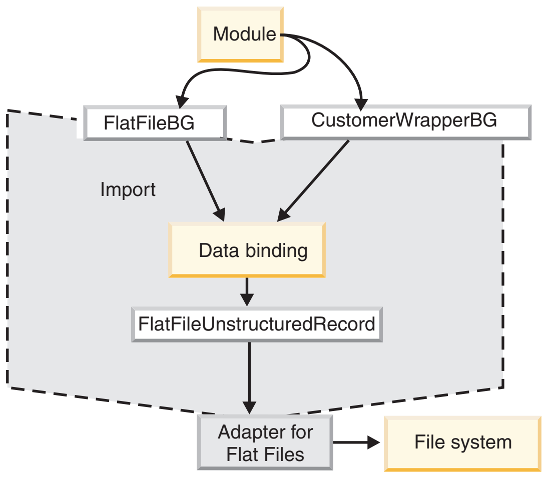


Figure 2. Data transformation during outbound processing

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 `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, no output is returned.

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 on page 9 illustrates outbound processing without data transformation.

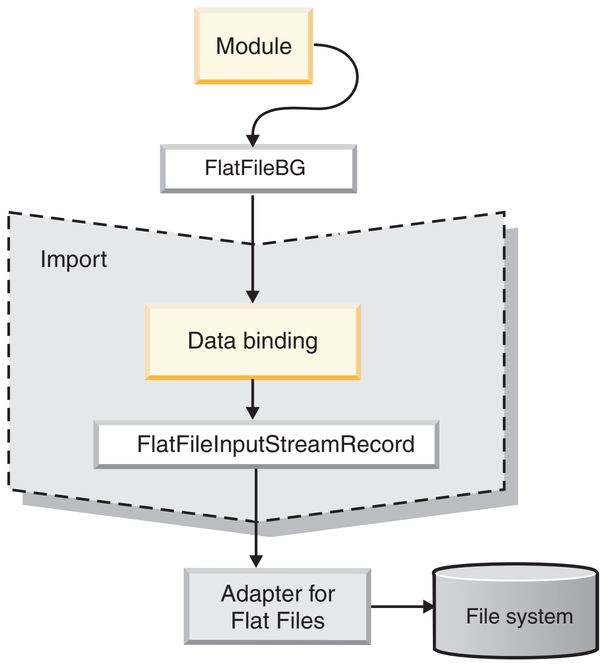


Figure 3. Outbound processing without data transformation

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 `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, no output is returned.

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.

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	`<?xml version="1.0" encoding="UTF-8"?>`
	`<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">`
	`<CustomerName>Deepa</CustomerName>`
	`<Address>IBM</Address>`
	`<City>Bangalore</City>`
	`<State>KA</State>`
	`</customer:Customer>`
		`</customer:Customer>;\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.SplitByDelimiter`, 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 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

Generate unique file names during Create operations by adding a persistent sequence number to the default file name or by using random numbers to generate file names. During Append and Overwrite operations, you must use the random number method.

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

1. By adding a persistent sequence number to the default file name. This method is recommended, especially in a clustered environment.
2. By using 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 handle a particular object differently, use the properties on 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` on the managed connection factory, the output file names that are created are `Customer.1.txt`, `Customer.2.txt`, and so on. The sequence number is maintained independently for each business object type.

The sequence is stored in the following format in the sequence file:

```
<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.

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`.

For the Append and Overwrite operations, when you set the Create a new file if the file does not exist (CreateFileIfNotExists) interaction specification property to `true` and the file already exists, the adapter creates a new file. The same file name generation rules apply as for the Create operation.

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.

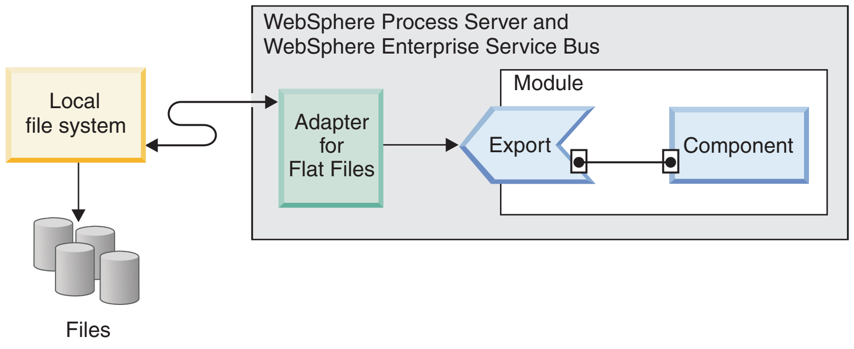


Figure 4. Inbound processing

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.
6. 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.
7. The adapter sends the business object to the export via a function selector, which selects an operation to invoke on the component and provides the correct data binding.
8. 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.

Event archiving

To keep track of successfully polled events, you can configure an archive directory on the file system using the ArchiveDirectory activation specification property in the external service wizard. The files are copied to the archive directory with `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.

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 before you can create the event store. To use the recovery feature provided by WebSphere Process Server, you set the AssuredOnceDelivery property in the activation specification to true. This recovery feature is on by default.

The adapter also provides for event persistence 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.

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.

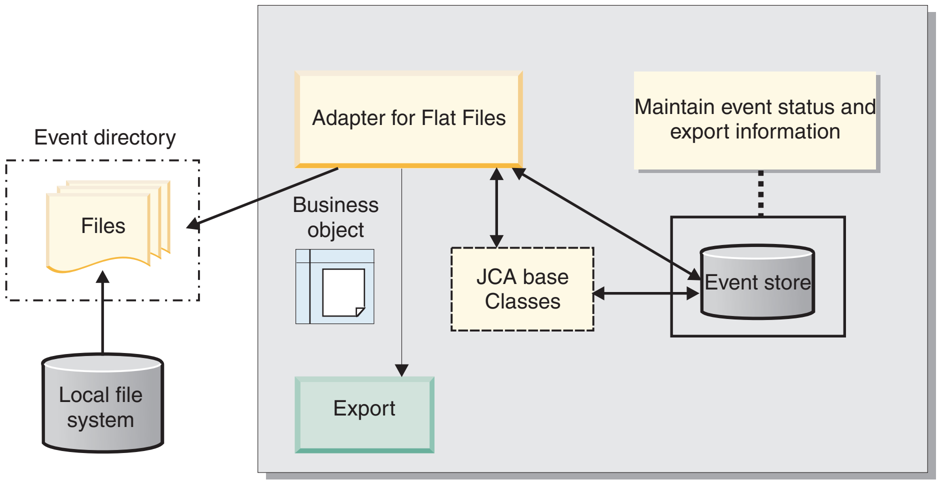


Figure 5. Event management flow

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 3. 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	The status of the event. The adapter uses the status to determine whether an event is new or in process. Event status values: **NEW (0)** The event is ready to be processed. **PROCESSED (1)** The adapter successfully processed and delivered the event. **FAILED (-1)** The adapter was unable to process this event due to one or more problems.
XID	Varchar(255)	Used by the adapter for assured event delivery and recovery.

Table 3. Event store structure (continued)

Column name	Type (length)	Description
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 4. 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

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 5. Examples of matching rules for FilenameFunctionSelector

FileName	ObjectName	Rule
`Customer0001.txt`	`Customer`	`CUST.*TXT`

Table 5. Examples of matching rules for FilenameFunctionSelector (continued)

FileName	ObjectName	Rule
2231ORZ93.z21	Order	[0-9]*OR[A-Z][0-9]{2}.*
2231ORZ93.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.

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 6.

Table 6.

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 7.

Data binding	BO content	Recommended delimiter format
XML	`<?xml version="1.0" encoding="UTF-8"?>` `<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">` `<CustomerName>Deepa</CustomerName>` `<Address>IBM</Address>` `<City>Bangalore</City>` `<State>KA</State>` `</customer:Customer>`	`</customer:Customer>;\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.SplitByDelimiter`, 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 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 FlatFileInputStreamRecord record is sent to the Foundation Class and from there to the EmbeddedNameFunctionSelector function selector, which maps between events generated by the adapter and the appropriate export function name.
5. 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 6.1.0, 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 6.1.0. If business graphs exist, they are processed, but the verb is ignored.

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.

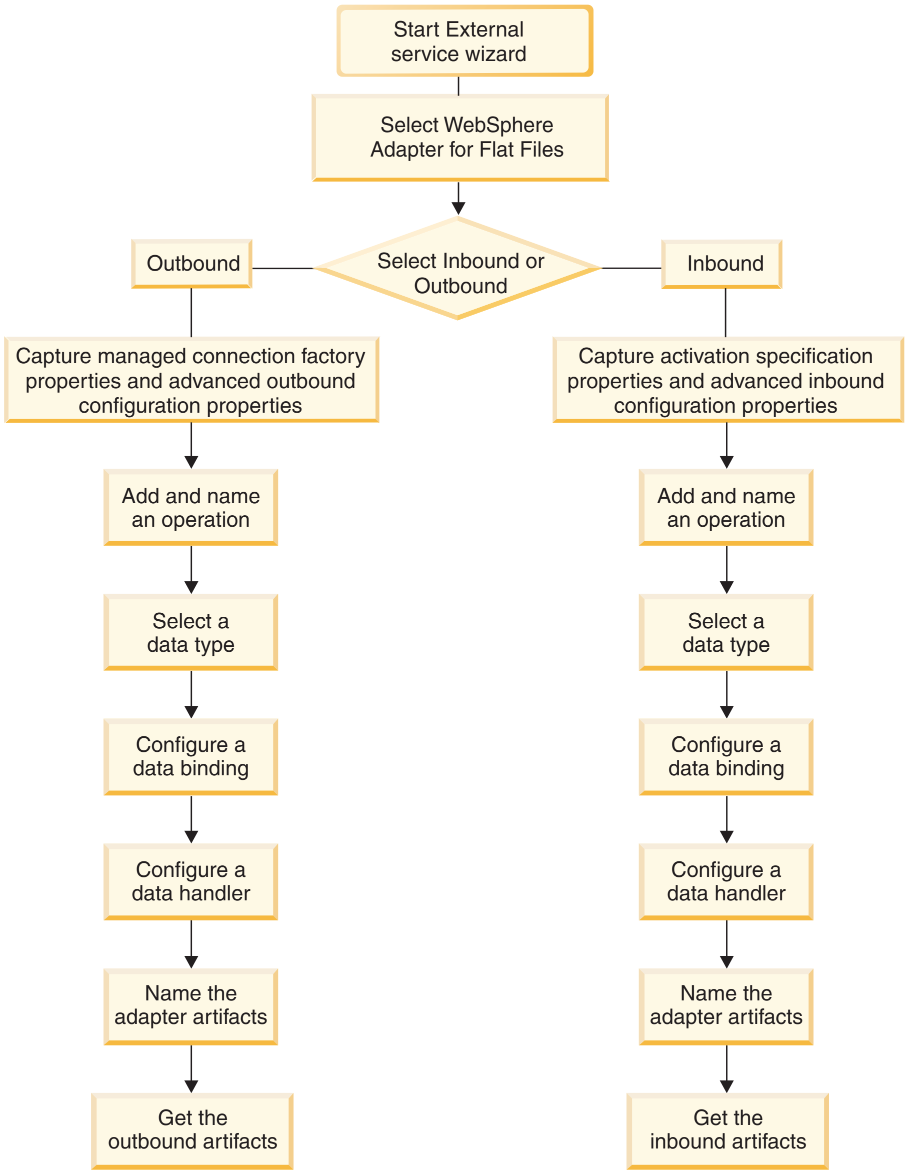

Start External service wizard
Select WebSphere Adapter for Flat Files
Select Inbound or Outbound
Outbound
Inbound
Capture managed connection factory properties and advanced outbound configuration properties
Capture activation specification properties and advanced inbound configuration properties
Add and name an operation
Select a data type
Configure a data binding
Configure a data handler
Name the adapter artifacts
Get the outbound artifacts
Get the inbound artifacts

Figure 6. Basic external service discovery wizard flow

Standards compliance

This product is compliant with several government and industry standards, including accessibility standards and Internet protocol standards.

Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability. WebSphere Adapters are fully accessible and section 508-compliant. Accessibility features enable users with physical disabilities, such as restricted mobility or limited vision, to operate software products successfully. These features are built into the installation and administration features of WebSphere Adapters.

Administration

The run time administrative console is the primary interface for deployment and administration of enterprise applications. The console is displayed within a standard Web browser. By using an accessible Web browser, such as Microsoft Internet Explorer or Netscape Browser, you are able to:

- Use screen-reader software and a digital speech synthesizer to hear what is displayed on the screen
- Use voice recognition software, such as IBM ViaVoice®, to enter data and to navigate the user interface
- Operate features by using the keyboard instead of the mouse

You can configure and use product features by utilizing standard text editors and scripted or command-line interfaces instead of the graphical interfaces that are provided.

When appropriate, the documentation for specific product features contains additional information about the accessibility of the features.

External service wizard

The external service wizard is the primary component used to create modules. This wizard, which is implemented as an Eclipse plug-in that is available through WebSphere Integration Developer, is fully accessible.

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

IBM and accessibility

See the *IBM Accessibility Center* web site http://www.ibm.com/able/ for more information about the commitment that IBM has to accessibility.

Internet Protocol Version 6 (IPv6)

WebSphere Process Server and WebSphere Enterprise Service Bus rely on WebSphere Application Server for Internet Protocol Version 6 (IPv6) compatibility.

IBM WebSphere Application Server, version 6.1.0 and later support pure Internet Protocol Version 6.0 (IPv6).

For more information about this compatibility in WebSphere Application Server, see IPv6 support in the http://www.ibm.com/software/webservers/appserv/was/library/.

For more information about IPv6, see http://www.ipv6.org.

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.

Deployment options

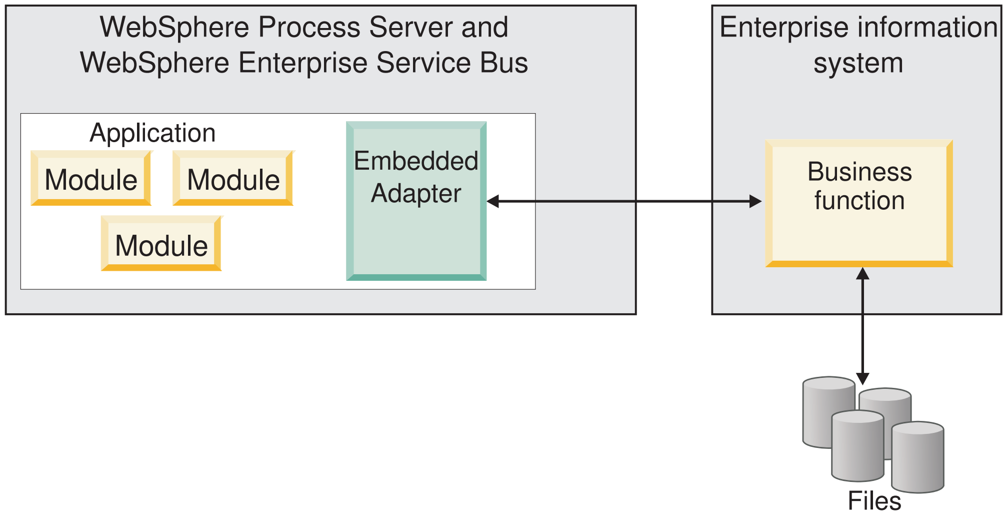
You can choose to embed the adapter to be part of the deployed application or you can choose to deploy the RAR file stand-alone.

The deployment options are described below:

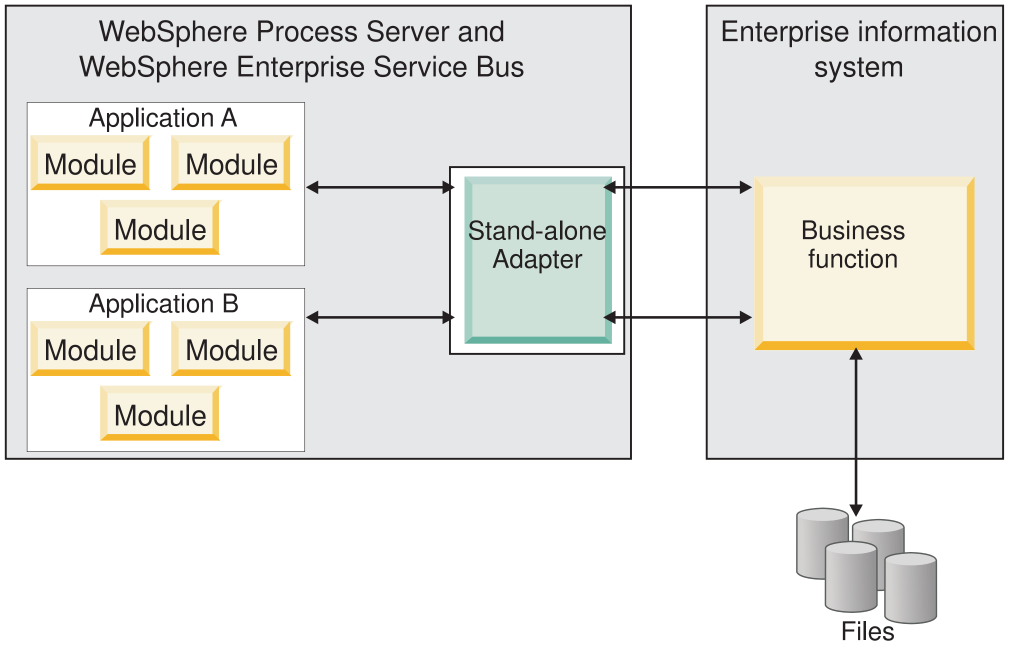
- **With module for use by single application**. With the adapter files embedded in the module, you can deploy the module to any application server. Use an embedded adapter when you have a single module using the adapter or if multiple modules need to run different versions of the adapter. Using an embedded 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.


WebSphere Process Server and
WebSphere Enterprise Service Bus
Application
Module
Module
Module
Embedded Adapter
Enterprise information system
Business function
Files

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.
- 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 classpath 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, 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 JCA (Java EE Connector Architecture) 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.

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 pooling between a WebSphere Process Server 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 pooling on a primary Windows node, it cannot switch to a backup UNIX node 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 (Java EE Connector Architecture) 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.

Migrating to version 6.1.0

By migrating to version 6.1 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 6.1.

Migration considerations

WebSphere Adapter for Flat Files version 6.1.0 includes updates that might affect your existing applications. Before migrating applications that will utilize WebSphere Adapter for Flat Files, take into consideration the information in the sections that follow.

Compatibility with earlier versions

WebSphere Adapter for Flat Files version 6.1.0 is fully compatible with version 6.0.2 of the adapter and can work with custom business objects (XSD files), and data bindings.

Because version 6.1 of WebSphere Adapter for Flat Files is fully compatible with version 6.02, any of your applications that utilized version 6.0.2 of WebSphere Adapter for Flat Files will run unchanged when you upgrade to version 6.1. However, if you want your applications to utilize features and functionality present in version 6.1 of the adapter, run the migration wizard.

The migration wizard replaces (upgrades) version 602 of the adapter with version 6.1 *and enables version 6.1 features and functionality for use with your applications*.

Note: The migration wizard does not create new or modify existing mitigating code, such as mappers and mediators to work with version 6.1 of the adapters. If any of your applications embed a 6.0.2.x or earlier version of an adapter and you are upgrading to version 6.1.0, and you want your applications to take advantage of the features and functions in 6.1, you might need to make changes to those applications.

If artifacts are inconsistent with regard to *versioning* within a single module, this module in its entirety will be marked as such, and will not be selectable for migration. Version inconsistencies are recorded in the workspace log, as this may be a symptom of project corruption.

Deciding whether to upgrade or to upgrade and migrate

The default processing of the migration wizard is to perform an upgrade of the adapter and to migrate the application artifacts so that the applications can utilize features and functions in version 6.1 of the adapter. When you choose to upgrade the connector by selecting a connector 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 to version 6.1, but you do not want to migrate the adapter artifacts, you can do so by deselecting the adapter artifacts from the appropriate page of the migration wizard.

Running the migration wizard without any adapter artifacts selected will install and upgrade your adapter, but your artifacts are not migrated and your applications will not be able to take advantage of the features and capabilities that exist in version 6.1 of the adapter.

Run the migration wizard in a test environment first

Because adapter migration may require you to make changes to those applications that will utilize version 6.1 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

Become familiar with the deprecated features in version 6.1.0 and make any required changes to your applications.

A deprecated feature is one that is supported but no longer recommended and that might become obsolete. Features from earlier versions of WebSphere Adapter for Flat Files that have been deprecated in version 6.1.0 include:
- Activation specification:
  - ArchivingProcessed
  - EventContentType
  - DefaultObjectName
- InteractionSpecification:

  - DefaultObjectName
- Wrapper properties:
  - RetrieveContentType
  - DefaultObjectName

Performing the migration

2 You can migrate a project or EAR file using the version 6.1.0, use the adapter
2 migration wizard. When the tool is finished, the migration is complete and you can
2 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 module will no longer be compatible with previous versions of WebSphere Process Server, WebSphere Enterprise Service Bus, or WebSphere Integration Developer.

Note: The following steps describe how to run the adapter migration wizard from the connector project context menu while in the J2EE perspective in WebSphere Integration Developer.

2 **Note:** You can also migrate in one of the following ways:
- 2 Right-click the project in the J2EE perspective and select **Migrate** → **Migrate**
  2 **project**.
- 2 From the Problems view, right-click a migration-specific message and select
  2 **Quick Fix** to correct the problem.

Procedure

1. 2 Import the PI (project interchange) file for an existing project or the EAR
   2 (enterprise archive) file for an deployed application into the workspace.
2. 2 Change to the J2EE perspective.
3. 2 Right-click the module and select **Migrate** → **Update Connector Project**.
4. 2 Review the tasks and warnings presented on the welcome page, and then select
   2 **Next**.
5. On the Select Projects window, select **Next**.

   2 By default, the wizard migrates the connector project and any dependent
   2 projects. If your project has dependent projects and you do not want to migrate
   2 one or more of them at this time, clear their check boxes in the **Dependent**
   2 **adapter project** list. You can rerun the wizard to migrate the dependent project
   2 at a later time. Previously migrated projects, projects with a current version,
   2 and projects that contain errors are unavailable for migration and are not
   2 selected.
6. On the Adapter Migration window, optionally review the migration changes, but do not change any selections. Click **Finish**.

2 7. Check the Problems view for messages from the migration wizard, which start
2 with the string CWPAD.
8. 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 6.1.0. 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 project

Before you can use a version 6.0.2 project, without migrating the complete project, with WebSphere Adapter for Flat Files, version 6.1.0 in WebSphere Integration Developer, version 6.1.0, use the migration wizard to update the project, and then correct a problem.

About this task

Because the internal name of the adapter changed in version 6.1.0, artifacts in a version 6.0.2 project must be updated to use the new name before you can use the adapter wizard in WebSphere Integration Developer, version 6.1.0. Use the migration wizard to update a version 6.0.2 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 J2EE perspective, right-click the project name and click **Migrate** → **Update Connector Project**. The adapter migration wizard opens.
3. On the welcome page, click **Next**.
4. On the Select Projects window, select none of the dependent artifact projects, and then click **Finish**.
5. In the Quick Fix window, make sure the fix **Rename the referenced adapter** is selected, and then click **OK**.
6. 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 6.1.0.

Chapter 3. Samples and tutorials

The WebSphere Integration Developer online samples/tutorials gallery includes samples and tutorials to help you use WebSphere Adapters.

You can access the online samples/tutorials gallery as follows:

- 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.

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. After completing these steps, you will have successfully created an external service.

Roadmap 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 roadmap.

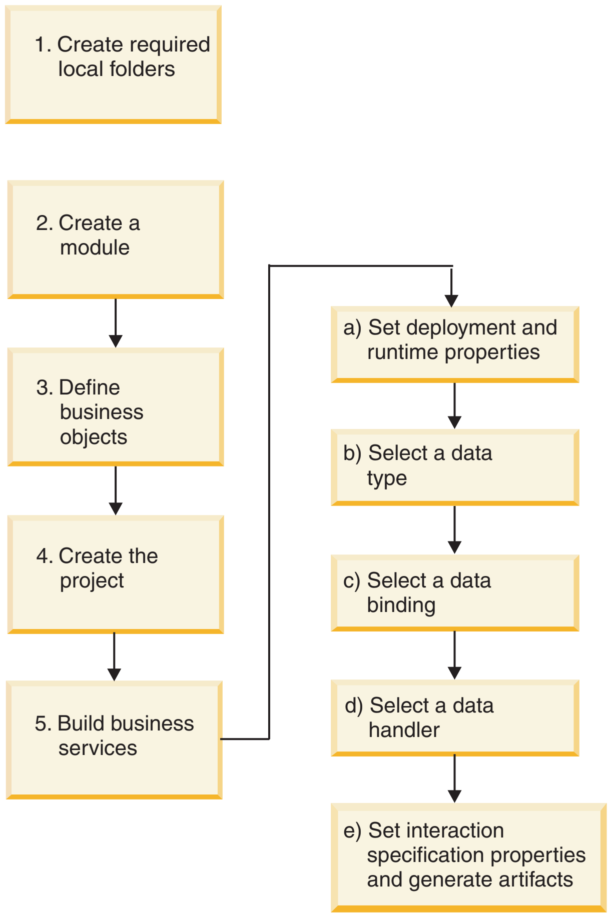

1. Create required local folders
2. Create a module
3. Define business objects
4. Create the project
5. Build business services
a) Set deployment and runtime properties
b) Select a data type
c) Select a data binding
d) Select a data handler
e) Set interaction specification properties and generate artifacts

Figure 7. Roadmap 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.

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**.

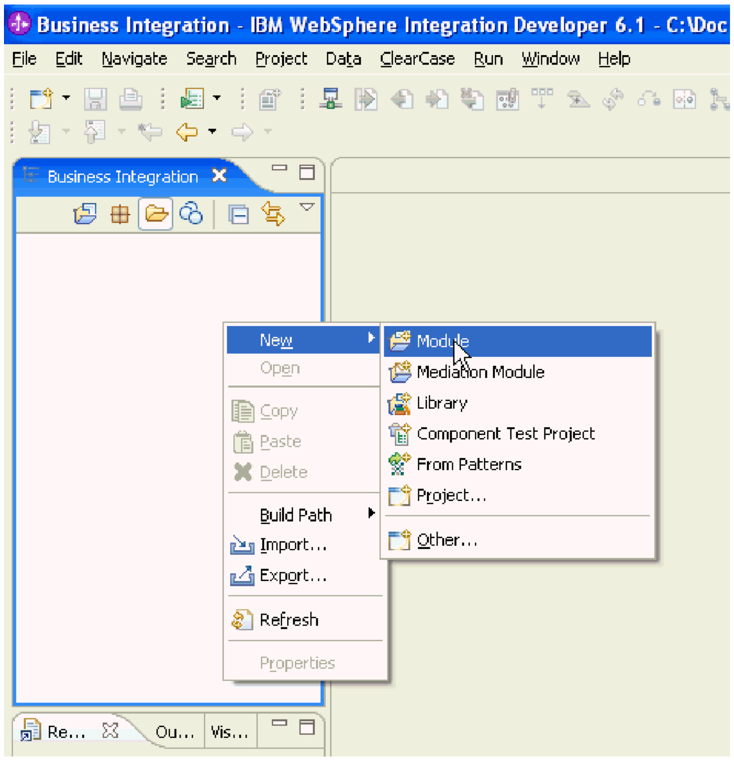
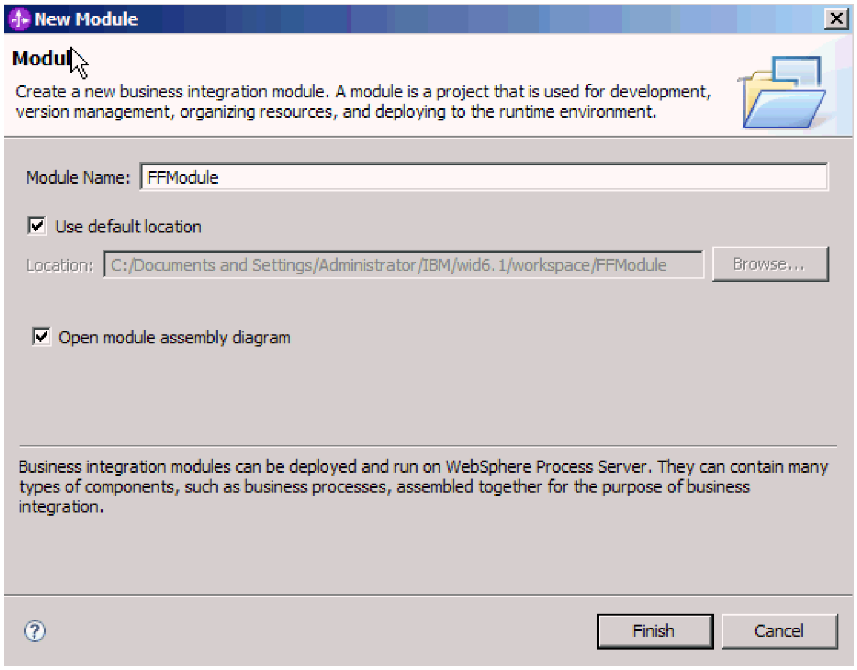


Figure 8. Business Integration section of the window

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



Module

Create a new business integration module. A module is a project that is used for development, version management, organizing resources, and deploying to the runtime environment.

Module Name: FFModule

☑ Use default location

Location: C:/Documents and Settings/Administrator/IBM/wid6.1/workspace/FFModule

☑ Open module assembly diagram

Business integration modules can be deployed and run on WebSphere Process Server. They can contain many types of components, such as business processes, assembled together for the purpose of business integration.

Figure 9. 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 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**.

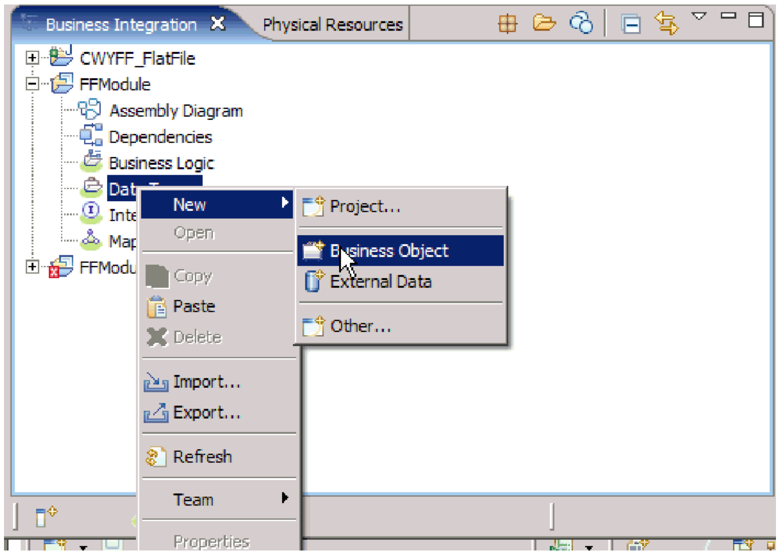
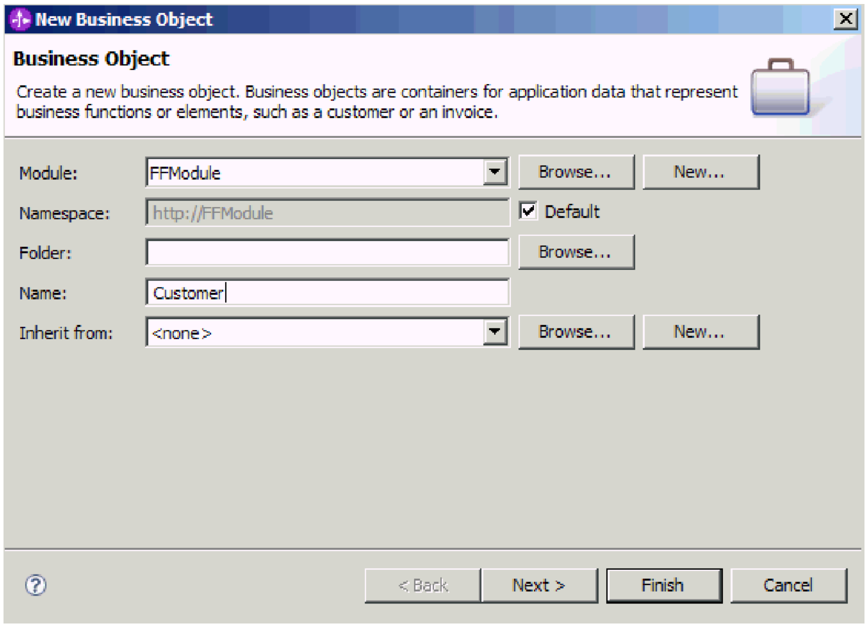


Figure 10. New Business Object selection view

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


New Business Object

Business Object

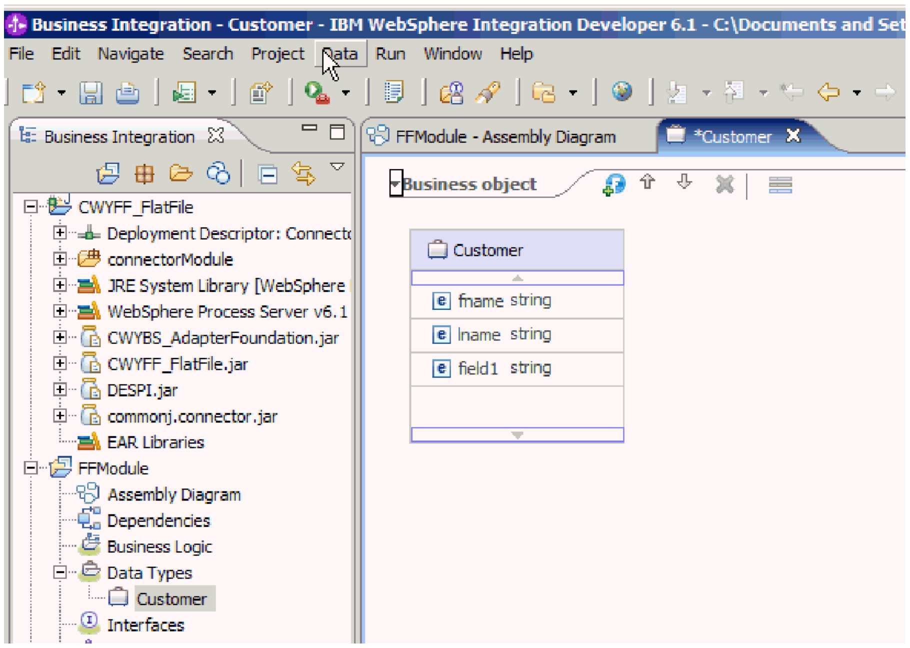
Create a new business object. Business objects are containers for application data that represent business functions or elements, such as a customer or an invoice.

Module:	FFModule	Browse...	New...
Namespace:	http://FFModule	✓ Default	
Folder:		Browse...	
Name:	Customer		
Inherit from:	<none>	Browse...	New...

< Back Next > Finish Cancel

Figure 11. 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.


Business Integration - Customer - IBM WebSphere Integration Developer 6.1 - C:\Documents and Set
File Edit Navigate Search Project Data Run Window Help
Business Integration
FFModule - Assembly Diagram
*Customer
Business object
CWYFF_FlatFile
Deployment Descriptor: Connect
connectorModule
JRE System Library [WebSphere
WebSphere Process Server v6.1
CWYBS_AdapterFoundation.jar
CWYFF_FlatFile.jar
DESPI.jar
commonj.connector.jar
EAR Libraries
FFModule
Assembly Diagram
Dependencies
Business Logic
Data Types
Customer
Interfaces

Customer	
fname	string
lname	string
field1	string

Figure 12. 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.

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.

About this task

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

Table 8.

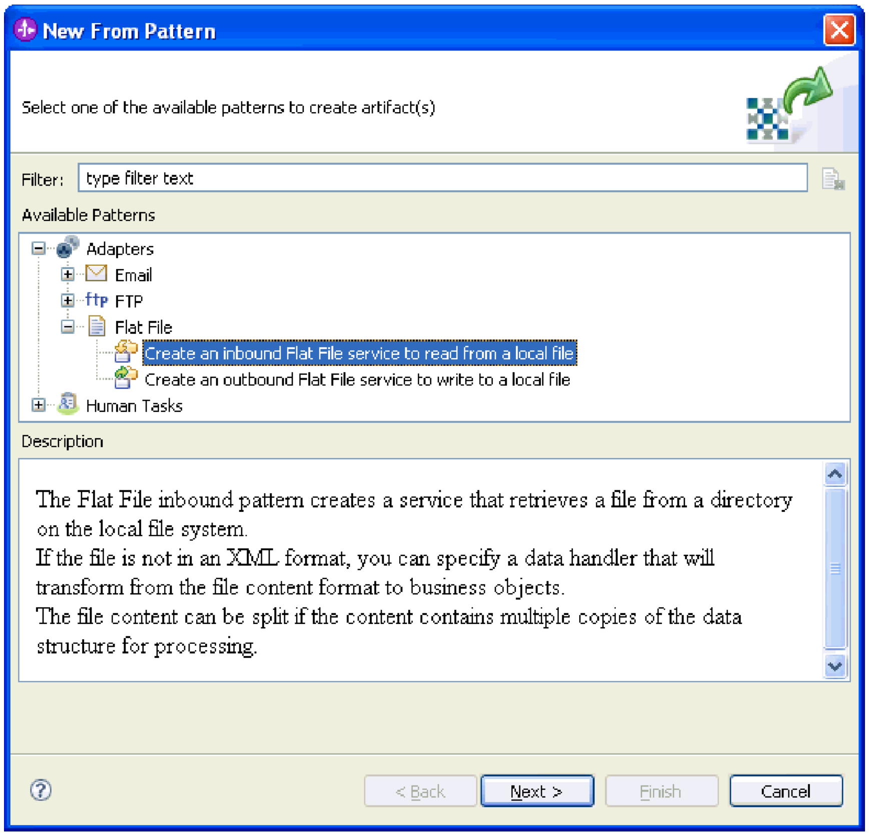
Adapter pattern	Description
Inbound Flat File pattern	The Flat File inbound pattern creates a service that retrieves a file in a specific directory on the local file system. If the file is not in an XML format, you can specify a data handler that will transform from the file content format to business 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

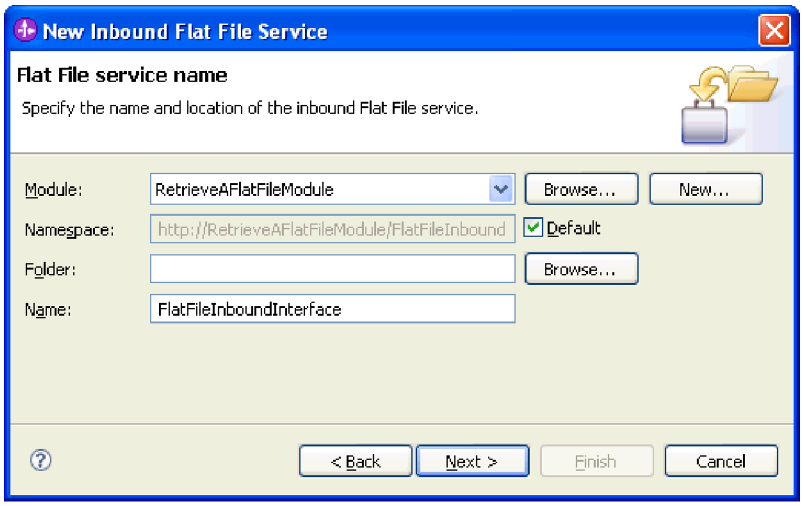
1. Right-click **RetrieveAFileModule** within the **Business Integration** section of the WebSphere Integration Developer window and select **New → From Patterns**. The New From Pattern window opens.
2. Select **Create an inbound Flat File service to read from a local file** and click **Next**.


New From Pattern
Select one of the available patterns to create artifact(s)
Filter: type filter text
Available Patterns
Adapters
Email
FTP
Flat File
Create an inbound Flat File service to read from a local file
Create an outbound Flat File service to write to a local file
Human Tasks
Description
< Back
Next >
Finish
Cancel

The Flat File inbound pattern creates a service that retrieves a file from a 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.

Figure 13. New From Pattern window

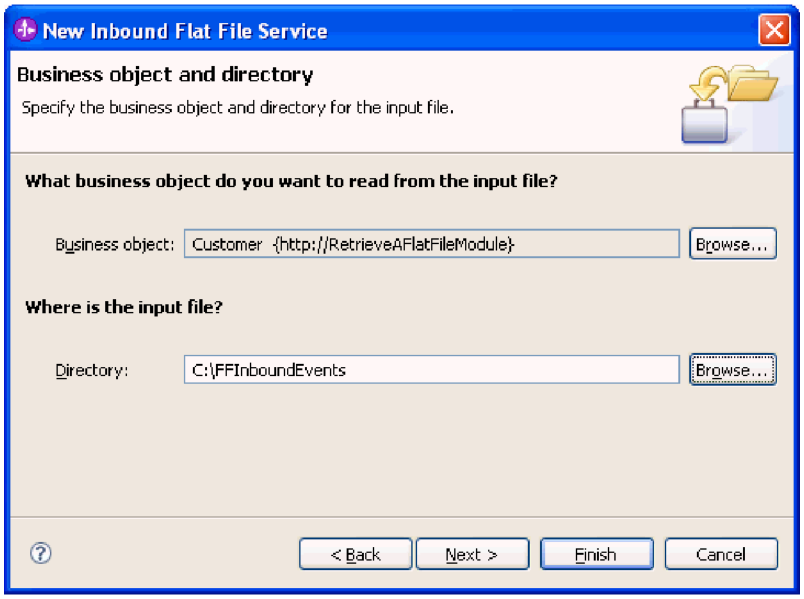
3. In the New Inbound Flat File Service window, change the name to something meaningful such as `FlatFileInboundInterface` and click **Next**.


New Inbound Flat File Service
Flat File service name
Specify the name and location of the inbound Flat File service.
Browse...
New...
Browse...
< Back
Next >
Finish
Cancel

Module: RetrieveAFlatFileModule
Namespace: http://RetrieveAFlatFileModule/FlatFileInbound ✓ Default
Folder:
Name: FlatFileInboundInterface

Figure 14. 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**.



Business object and directory

Specify the business object and directory for the input file.

What business object do you want to read from the input file?

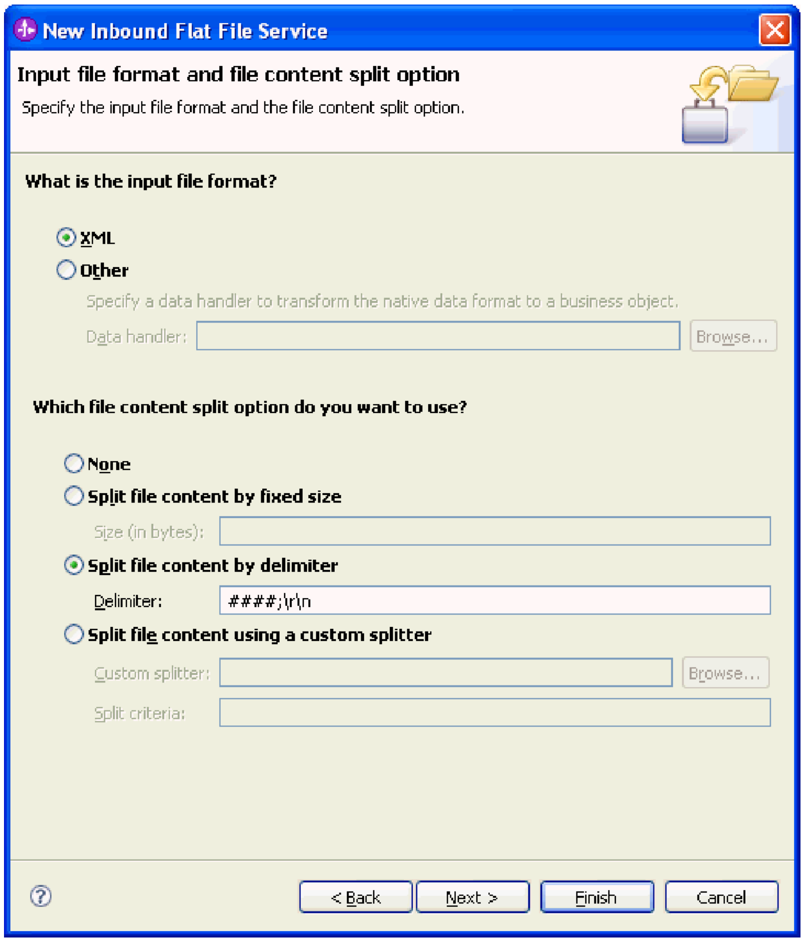
Business object: Customer {http://RetrieveAFlatFileModule}

Where is the input file?

Directory: C:\FFInboundEvents

Figure 15. 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**.


New Inbound Flat File Service

Input file format and file content split option

Specify the input file format and the file content split option.

What is the input file format?

[✓ XML]

Other

Specify a data handler to transform the native data format to a business object.

Data handler:

Which file content split option do you want to use?

None

Split file content by fixed size

Size (in bytes):

[✓ Split file content by delimiter]

Delimiter: ####;\r\n

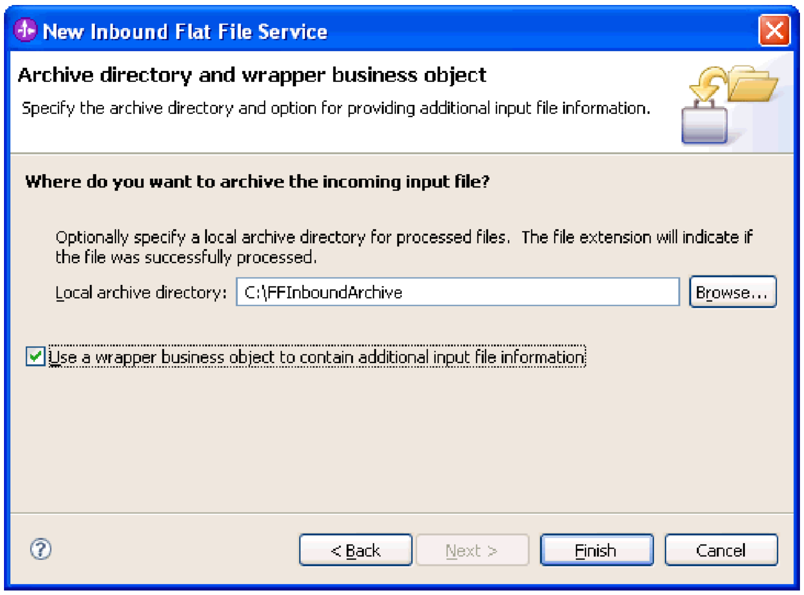
Split file content using a custom splitter

Custom splitter:

Split criteria:

Figure 16. 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. Select **Use a wrapper business object to contain additional input file information** if you want to include the adapter-specific information. Click **Finish**.


New Inbound Flat File Service

Archive directory and wrapper business object

Specify the archive directory and option for providing additional input file information.

Where do you want to archive the incoming input file?

Optionally specify a local archive directory for processed files. The file extension will indicate if the file was successfully processed.

Local archive directory: C:\FFInboundArchive

[✓] Use a wrapper business object to contain additional input file information

Figure 17. Archive directory and wrapper business object window

Results

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

Table 9.

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.

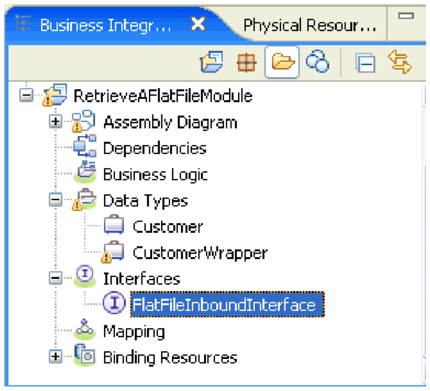


Figure 18. The **Business Integration** *section of the WebSphere Integration Developer window with the new artifacts*

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

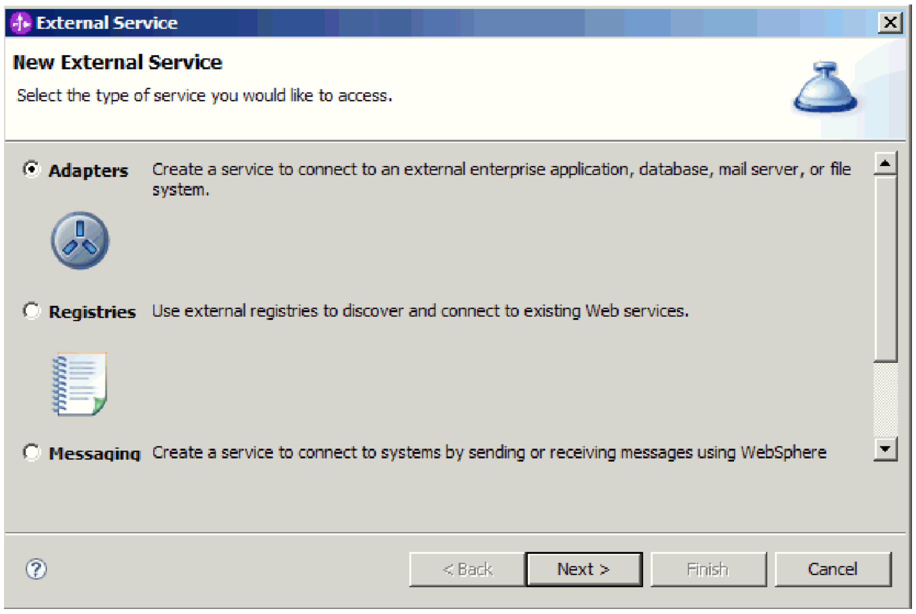
Start the external service wizard to create a project for the adapter in WebSphere Integration Developer. If you have an existing project, you can select it instead of having the wizard create one.

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

Procedure

1. If WebSphere Integration Developer is not currently running, start it now.
   a. Click **Start** → **Programs** → **IBM Software Development Platform** → **IBM WebSphere Integration Developer 6.1** → **WebSphere Integration Developer 6.1**.
   b. If you are prompted to specify a workspace, accept the default value or select another workspace.

      The workspace is a directory where WebSphere Integration Developer stores your project.
   c. When the WebSphere Integration Developer window is displayed, click **Go to the Business Integration perspective**.
2. To start the external service wizard, click **File** → **New** → **External Service**.
3. In the New external service window, make sure **Adapters** is selected, and click **Next**.


External Service

New External Service

Select the type of service you would like to access.

(●) Adapters Create a service to connect to an external enterprise application, database, mail server, or file system.

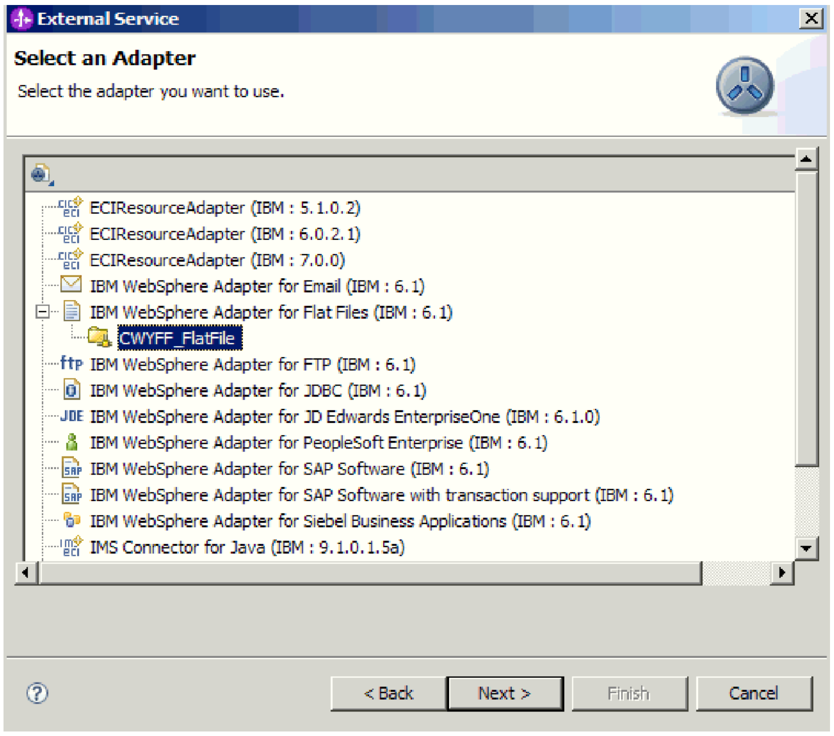
( ) Registries Use external registries to discover and connect to existing Web services.

( ) Messaging Create a service to connect to systems by sending or receiving messages using WebSphere

Figure 19. The New external service window

4. 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.



ECIResourceAdapter (IBM : 5.1.0.2)
ECIResourceAdapter (IBM : 6.0.2.1)
ECIResourceAdapter (IBM : 7.0.0)
IBM WebSphere Adapter for Email (IBM : 6.1)
IBM WebSphere Adapter for Flat Files (IBM : 6.1)
CWYFF_FlatFile
IBM WebSphere Adapter for FTP (IBM : 6.1)
IBM WebSphere Adapter for JDBC (IBM : 6.1)
IBM WebSphere Adapter for JD Edwards EnterpriseOne (IBM : 6.1.0)
IBM WebSphere Adapter for PeopleSoft Enterprise (IBM : 6.1)
IBM WebSphere Adapter for SAP Software (IBM : 6.1)
IBM WebSphere Adapter for SAP Software with transaction support (IBM : 6.1)
IBM WebSphere Adapter for Siebel Business Applications (IBM : 6.1)
IMS Connector for Java (IBM : 9.1.0.1.5a)

Figure 20. 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.

Setting deployment and runtime properties

Using the external service wizard in WebSphere Integration Developer, choose whether your module will be used for outbound or inbound communication with the local file system. Then configure the managed connection factory properties. Managed connection factory properties are stored in the business object and contain the information the adapter will need to make the connection between the module and the local file system.

Before you begin

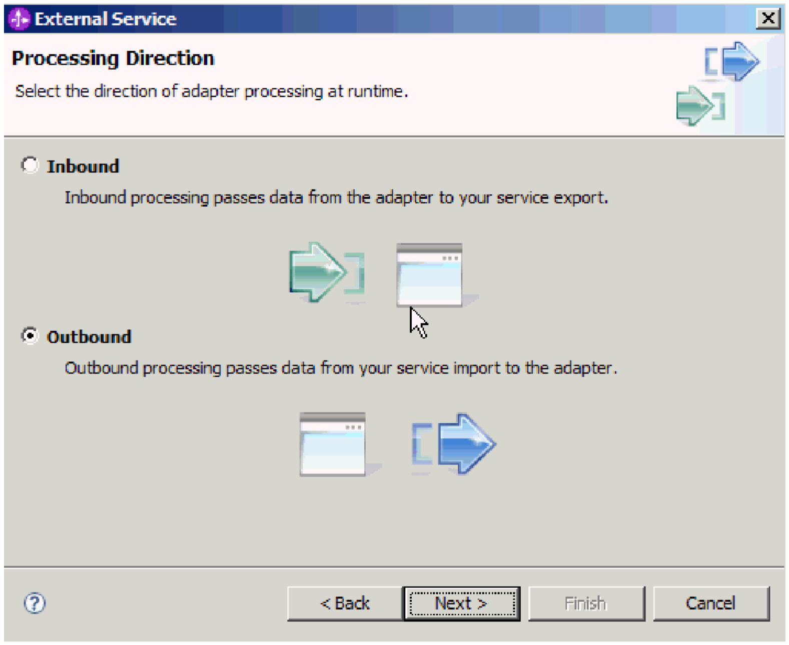
Before you can set the properties in this section, you must have created your adapter module. It should be displayed in WebSphere Integration Developer below the adapter project. For more information on creating the adapter project, see the topic devoted to it in this documentation.

About this task

To set connection properties, follow this procedure. For more information on the properties in this topic, see the reference topic devoted to managed connection factory properties in this documentation.

Procedure

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



Processing Direction

Select the direction of adapter processing at runtime.

( ) **Inbound**

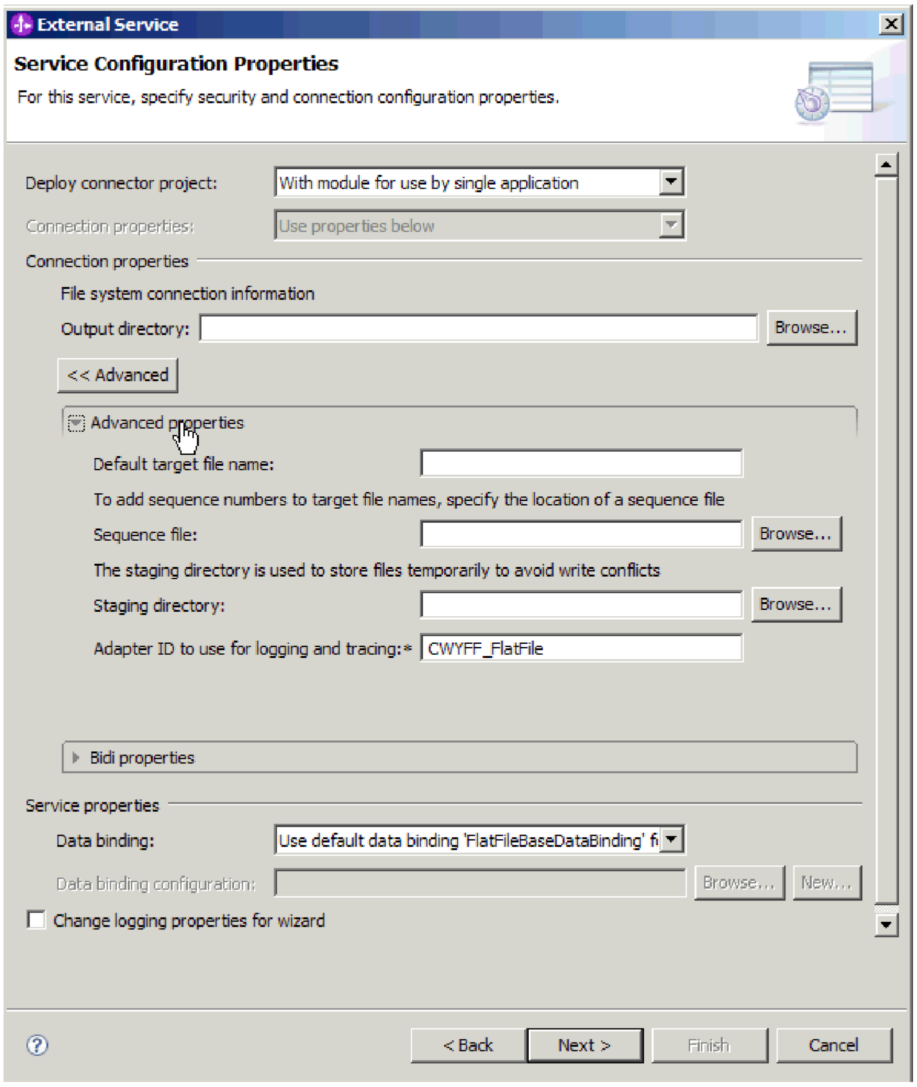
Inbound processing passes data from the adapter to your service export.

(•) **Outbound**

Outbound processing passes data from your service import to the adapter.

Figure 21. 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 reference topic.


External Service

Service Configuration Properties

For this service, specify security and connection configuration properties.

Deploy connector project: With module for use by single application

Connection properties: Use properties below

Connection properties

File system connection information

Output directory:

<< Advanced

Advanced properties

Default target file name:

To add sequence numbers to target file names, specify the location of a sequence file

Sequence file:

The staging directory is used to store files temporarily to avoid write conflicts

Staging directory:

Adapter ID to use for logging and tracing:* CWYFF_FlatFile

Bidi properties

Service properties

Data binding: Use default data binding 'FlatFileBaseDataBinding' f

Data binding configuration:

Change logging properties for wizard

Figure 22. Setting the connection properties

4. Optional: To change the **Adapter ID to use for logging and tracing**, enter a new value. For more information about this property, see the Resource adapter properties reference topic.
5. Optional: Select the **Change logging properties for wizard** checkbox if you want to specify the log file output location or define the level of logging for this module. For information about logging levels, see the section on configuring logging properties in the Troubleshooting and support topic.
6. 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.

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

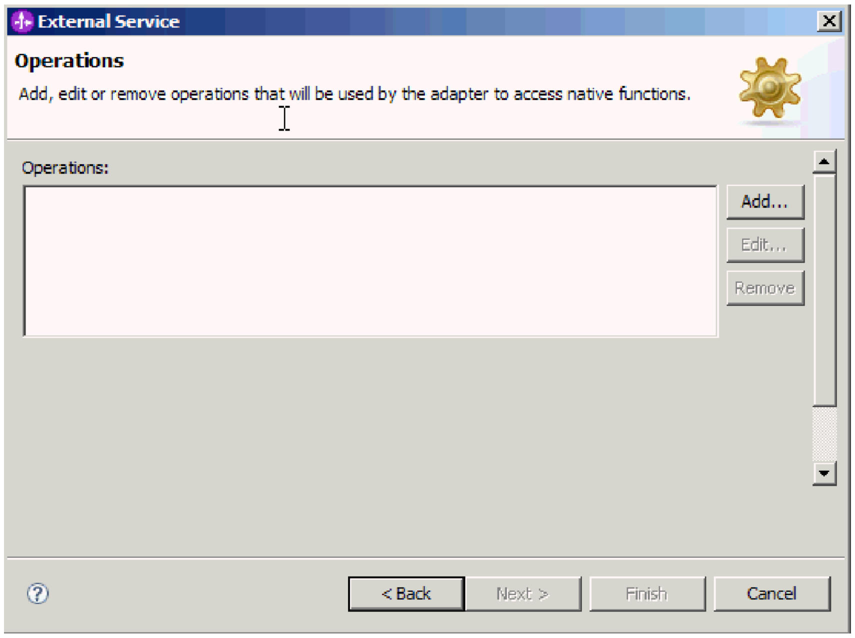
You must have specified the connection properties for the adapter to connect to the local file system before you can complete the steps below.

About this task

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

Procedure

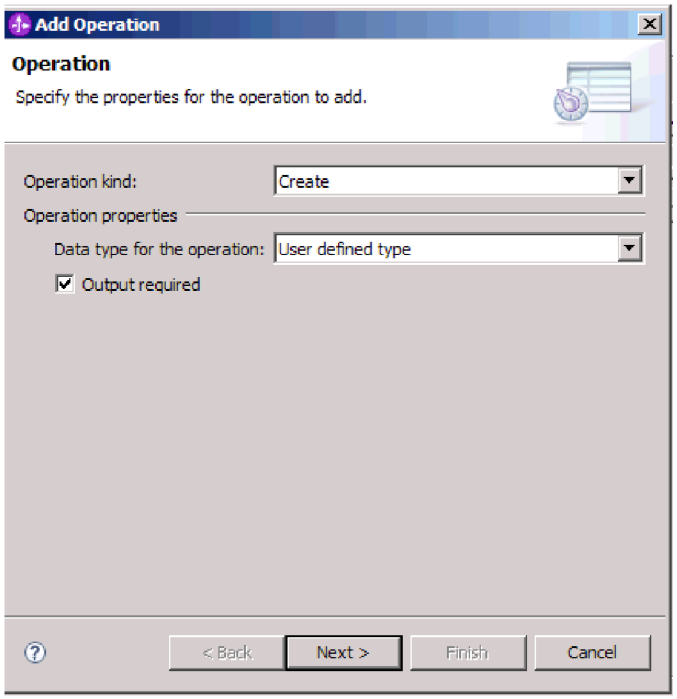
1. On the Operations window, click **Add**.



Add, edit or remove operations that will be used by the adapter to access native functions.

Figure 23. Adding an operation

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



Operation kind: Create

Operation properties

Data type for the operation: User defined type

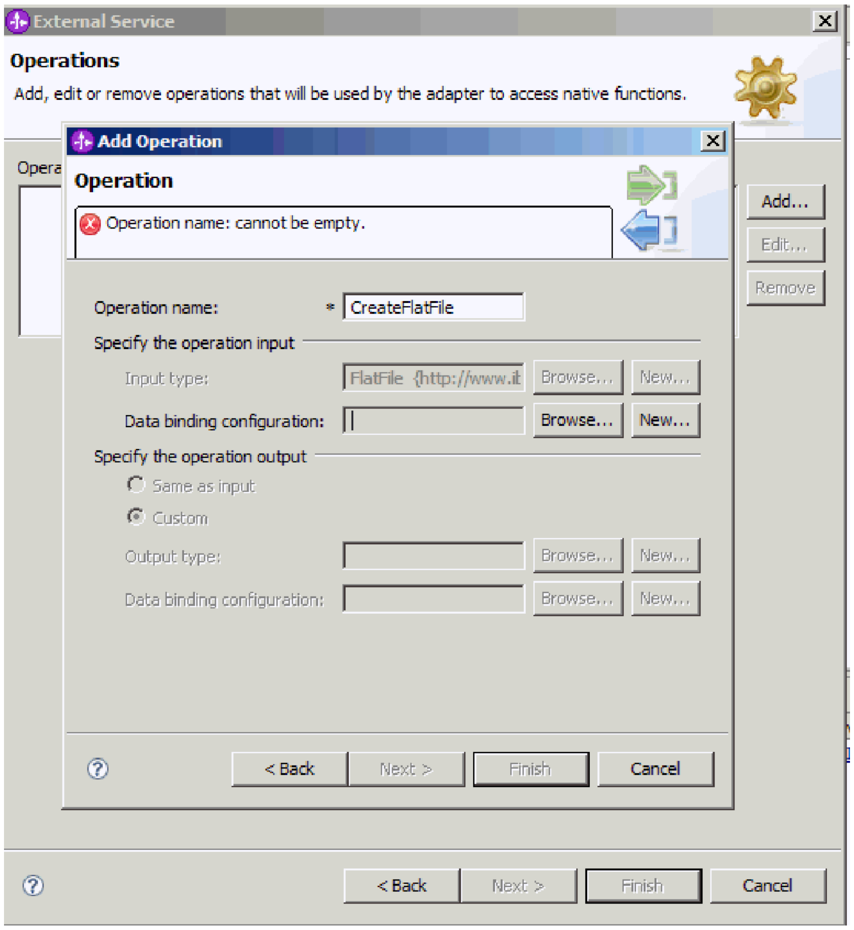
☑ Output required

Figure 24. 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, you can select the **Output required** checkbox to have the file name returned. Select this if you are generating a unique file name or have enabled file sequencing. See the descriptions of the GenerateUniqueFile and FileSequenceLog interaction specification properties for more information. For the Exists, List, and Retrieve operations, output is required, and the **Output required** checkbox is checked and disabled. For the Delete operation, no output is returned, and the **Output required** checkbox is unchecked and disabled.


External Service

Operations

Add, edit or remove operations that will be used by the adapter to access native functions.

Add Operation

Operation

Operation name: cannot be empty.

Operation name: * CreateFlatFile

Specify the operation input

Input type: FlatFile {http://www.it

Data binding configuration:

Specify the operation output

Same as input

Custom

Output type:

Data binding configuration:

Figure 25. Naming the operation and specifying the input data type

5. Click **Next**.
6. On 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.

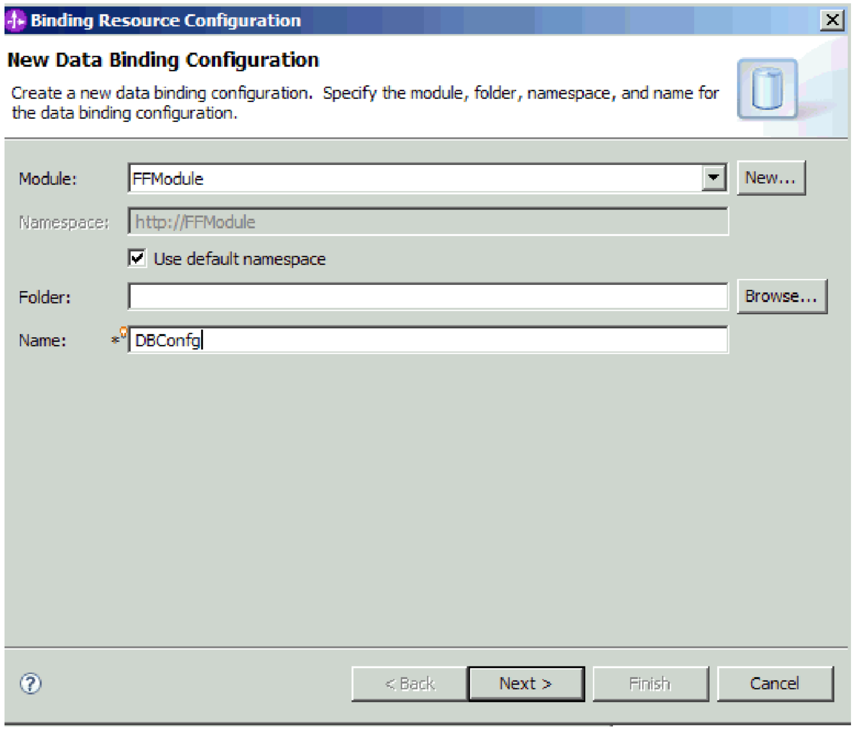
About this task

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 (in this example we used DBConfg) and click **Next**.



New Data Binding Configuration

Create a new data binding configuration. Specify the module, folder, namespace, and name for the data binding configuration.

Module: FFModule

Namespace: http://FFModule

☑ Use default namespace

Folder:

Name: * DBConfg

Figure 26. 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.

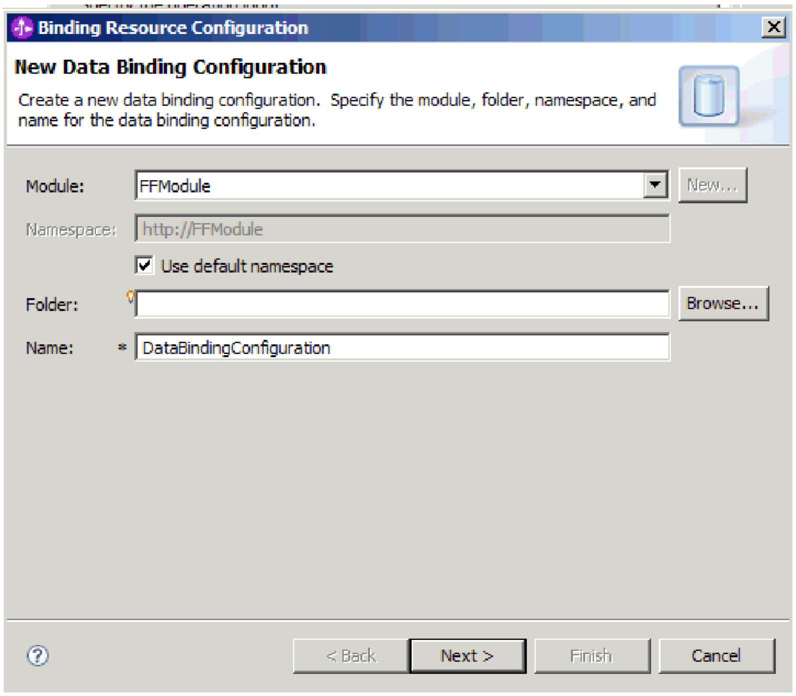
Note: Data handlers can be configured prior to running the external service wizard using WebSphere Integration Developer. To do this, select **New** → **Resource configuration** in WebSphere Integration Developer and complete the data handler screens described in this documentation.

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 (in this example we used 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.


Binding Resource Configuration

New Data Binding Configuration

Create a new data binding configuration. Specify the module, folder, namespace, and name for the data binding configuration.

Module: FFModule

Namespace: http://FFModule

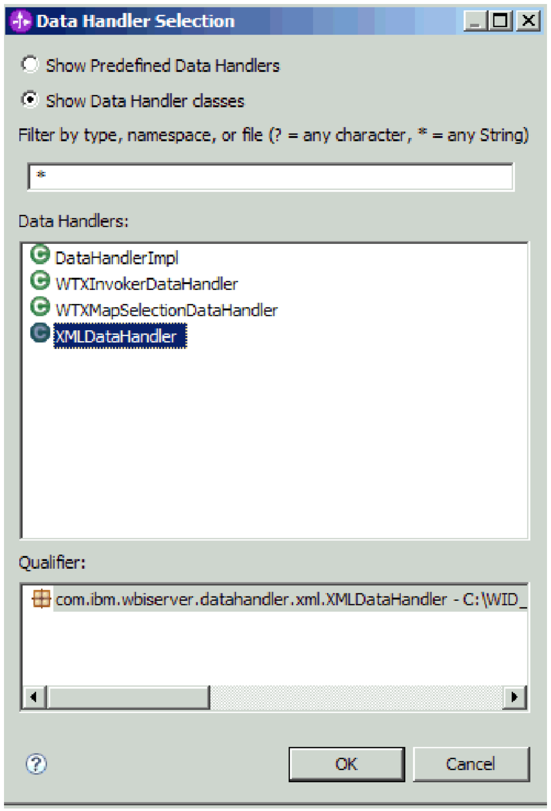
☑ Use default namespace

Folder:

Name: * DataBindingConfiguration

Figure 27. 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.


Data Handler Selection

Show Predefined Data Handlers [✓ Show Data Handler classes]

Filter by type, namespace, or file (? = any character, * = any String)

*

Data Handlers:

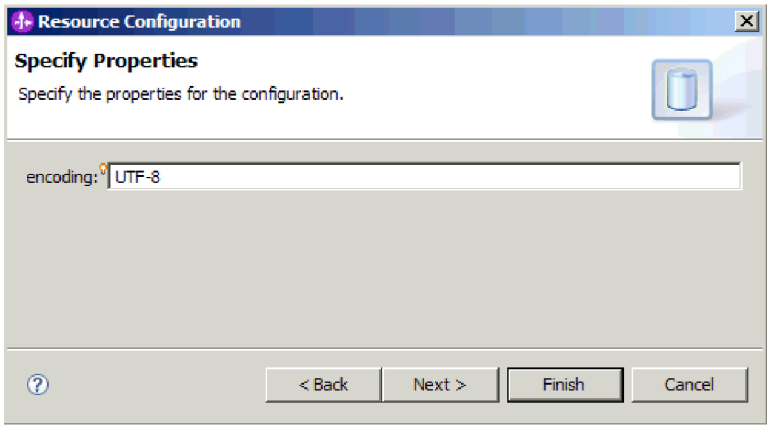
- DataHandlerImpl
- WTXInvokerDataHandler
- WTXMapSelectionDataHandler
- XMLDataHandler

Qualifier:

com.ibm.wbiserver.datahandler.xml.XMLDataHandler - C:\WID_

Figure 28. Creating a new data handler configuration

6. Click **Next**.
7. 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 (in this example, we used XMLDataHandler). Click **OK**
8. Click **Next**.
9. In the Specify properties window, specify the encoding. The default is UTF-8.


Resource Configuration

Specify Properties

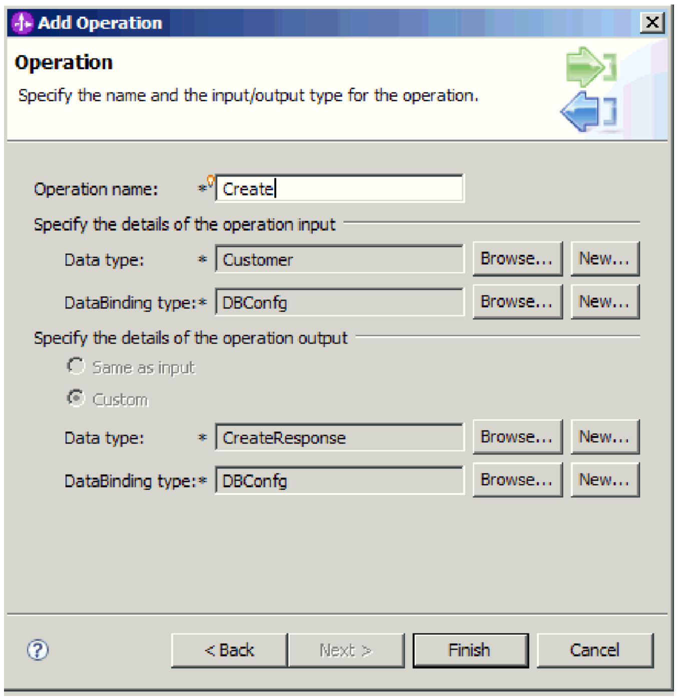
Specify the properties for the configuration.

encoding: UTF-8

Figure 29. Specifying the encoding for the data handler configuration

10. Click **Finish**.

11. 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 we set the operation input DataBinding type, we select the same data binding type (DBConfg) for the operation output DataBinding type.


Add Operation

Operation

Specify the name and the input/output type for the operation.

Operation name: * Create

Specify the details of the operation input

Data type: * Customer

DataBinding type:* DBConfg

Specify the details of the operation output

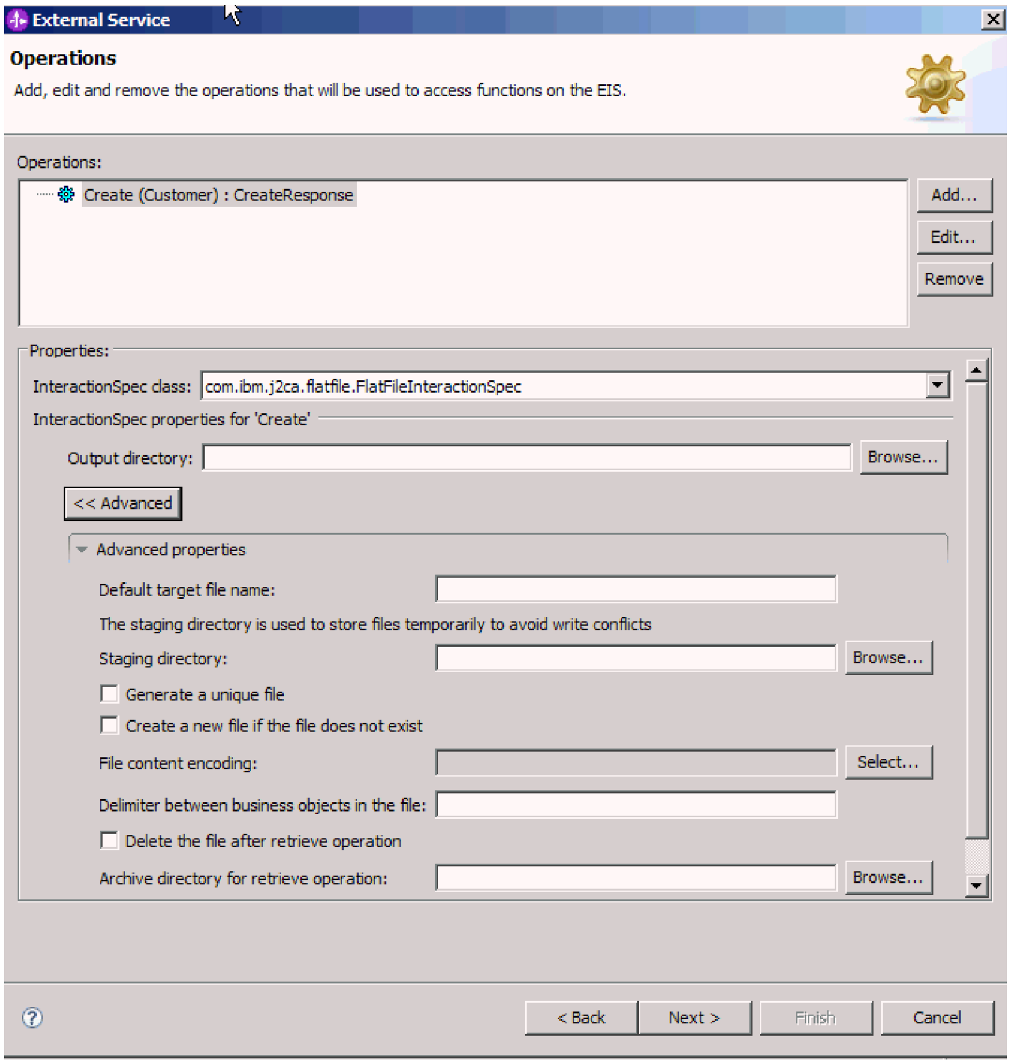
Same as input [✓ Custom]

Data type: * CreateResponse

DataBinding type:* DBConfg

Figure 30. Choosing the data binding configuration for the operation output

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


External Service

Operations

Add, edit and remove the operations that will be used to access functions on the EIS.

Operations:

Create (Customer) : CreateResponse

Properties:

InteractionSpec class: com.ibm.j2ca.flatfile.FlatFileInteractionSpec

InteractionSpec properties for 'Create'

Output directory:

<< Advanced

Advanced properties

Default target file name:

The staging directory is used to store files temporarily to avoid write conflicts

Staging directory:

☐ Generate a unique file

☐ Create a new file if the file does not exist

File content encoding:

Delimiter between business objects in the file:

☐ Delete the file after retrieve operation

Archive directory for retrieve operation:

Figure 31. The Create operation with InteractionSpec properties

13. 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

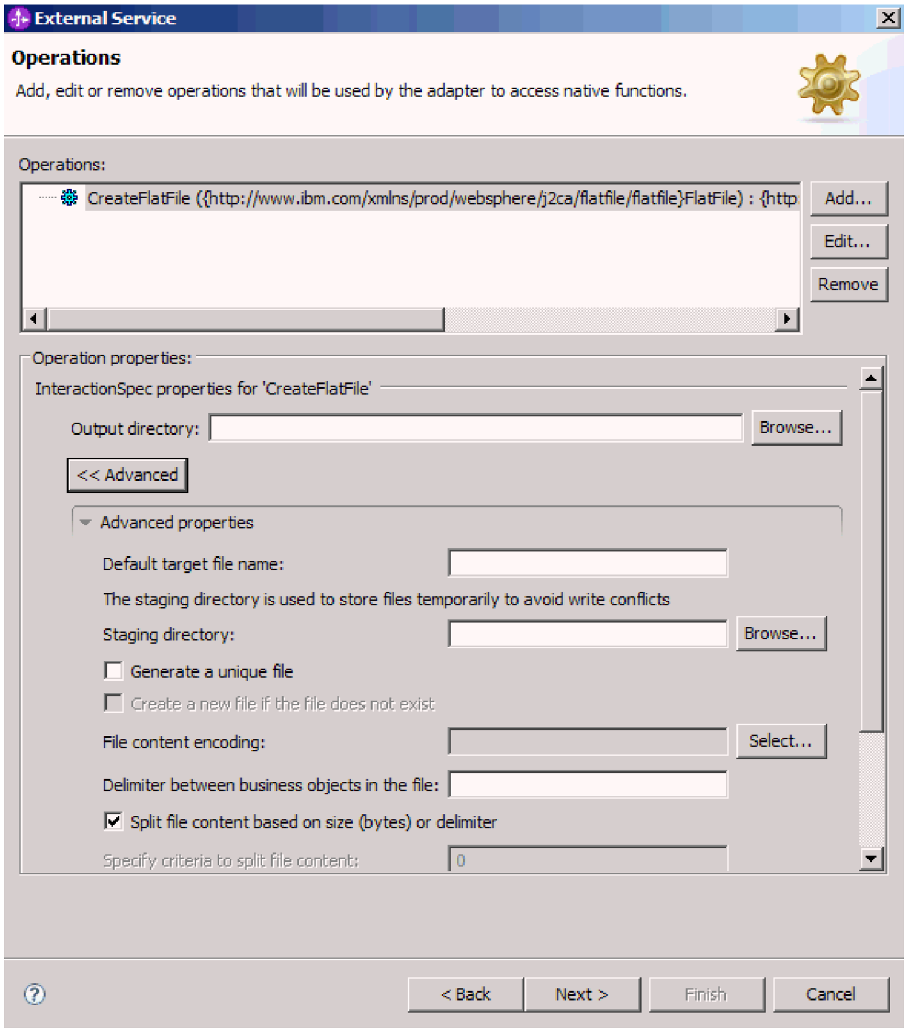
To set interaction specification properties and generate artifacts for your module, you must have already configured data bindings and selected business objects.

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**.


External Service

Operations

Add, edit or remove operations that will be used by the adapter to access native functions.

Operations:

CreateFlatFile ({http://www.ibm.com/xmlns/prod/websphere/j2ca/flatfile/flatfile}FlatFile) : {http

Operation properties:

InteractionSpec properties for 'CreateFlatFile'

Output directory:

Advanced properties

Default target file name:

The staging directory is used to store files temporarily to avoid write conflicts

Staging directory:

☐ Generate a unique file

☐ Create a new file if the file does not exist

File content encoding:

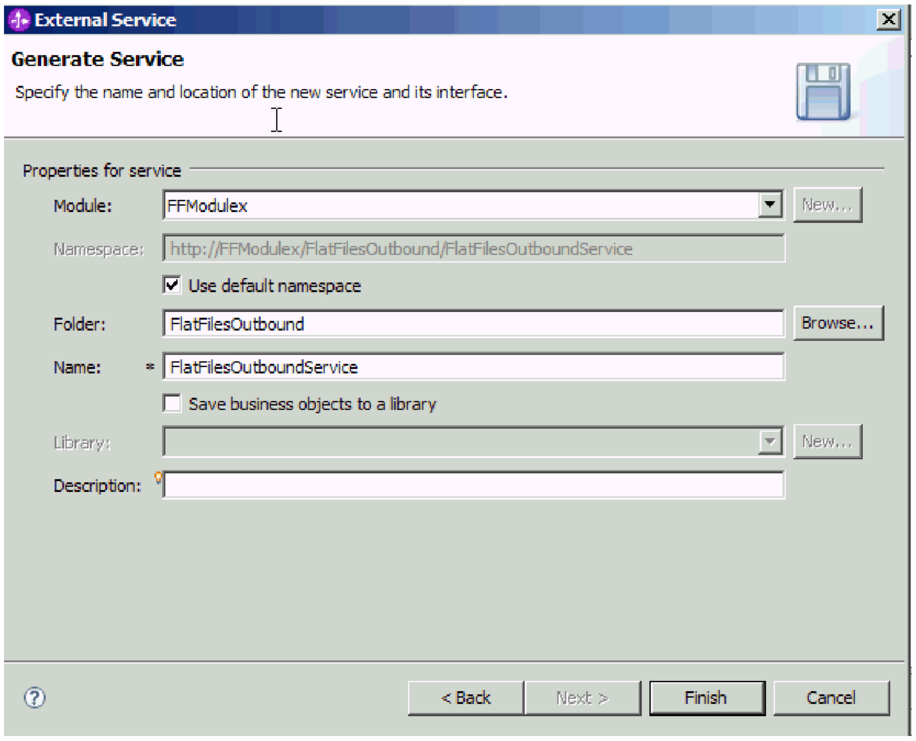
Delimiter between business objects in the file:

☑ Split file content based on size (bytes) or delimiter

Specify criteria to split file content: 0

Figure 32. 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.


External Service

Generate Service

Specify the name and location of the new service and its interface.

Properties for service

Module: FFModulex

Namespace: http://FFModulex/FlatFilesOutbound/FlatFilesOutboundService

☑ Use default namespace

Folder: FlatFilesOutbound

Name: * FlatFilesOutboundService

☐ Save business objects to a library

Library:

Description:

Figure 33. 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.

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.

Setting deployment and runtime properties

Using the external service wizard in WebSphere Integration Developer, choose whether your module will be used for outbound or inbound communication with the local file system. Then configure the activation specification properties. Activation specification properties hold the inbound event processing configuration information for the export.

Before you begin

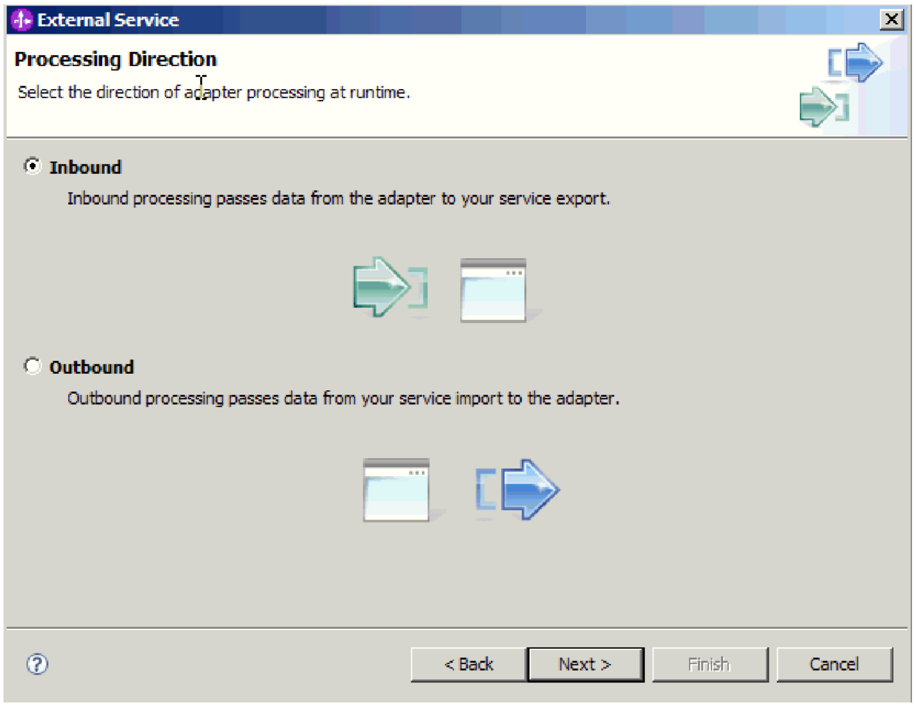
Before you can set the properties in this section, you must have created your adapter module. It should be displayed in WebSphere Integration Developer below the adapter project. For more information on creating the adapter project, see the topic devoted to it in this documentation.

About this task

To set the activation specification properties, follow this procedure. For more information on the properties in this topic, see the reference topic devoted to activation specification properties in this documentation.

Procedure

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


External Service

Processing Direction

Select the direction of adapter processing at runtime.

[✓ Inbound]

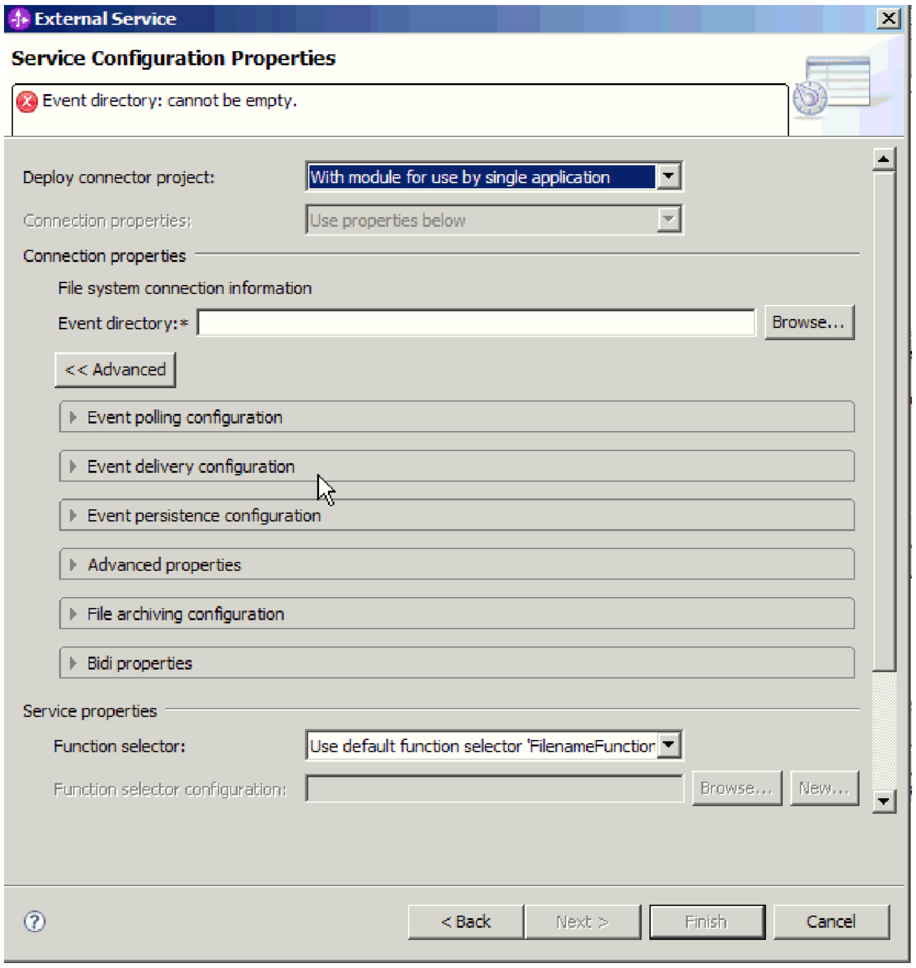
Inbound processing passes data from the adapter to your service export.

Outbound

Outbound processing passes data from your service import to the adapter.

Figure 34. Choosing inbound or outbound 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 activation specification properties for your module. For more details on the properties found on this window, see the activation specification properties reference topic.


External Service

Service Configuration Properties

Event directory: cannot be empty.

Deploy connector project: With module for use by single application

Connection properties: Use properties below

Connection properties

File system connection information

Event directory:*

Browse...
<< Advanced
Event polling configuration
Event delivery configuration
Event persistence configuration
Advanced properties
File archiving configuration
Bidi properties

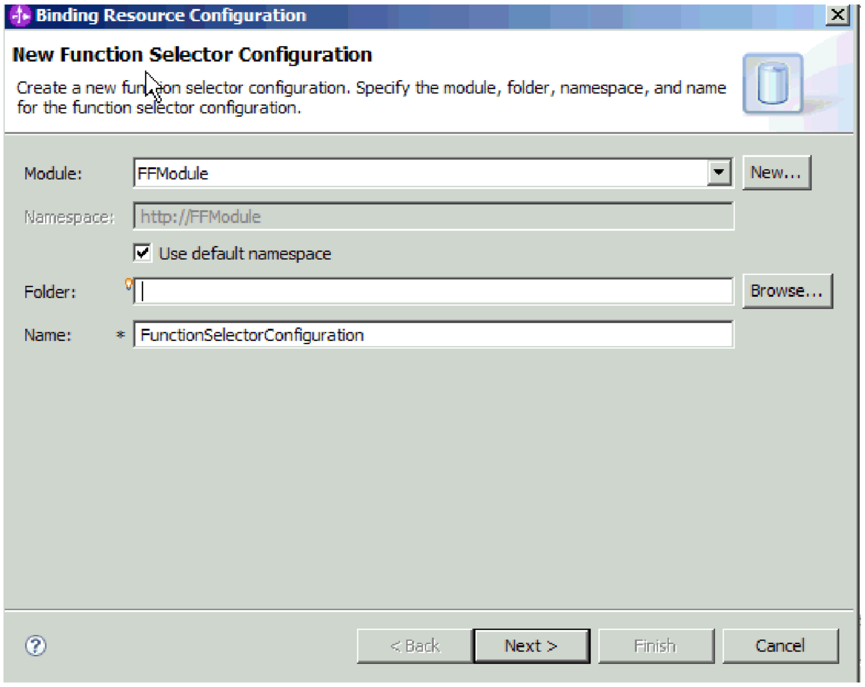
Service properties

Function selector: Use default function selector 'FilenameFunctior

Function selector configuration:

Figure 35. 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. Optional: To change the **Adapter ID to use for logging and tracing**, enter a new value. For more information about this property, see the Resource adapter properties reference topic.
6. Optional: Select the **Change logging properties for wizard** checkbox if you want to specify the log file output location or define the level of logging for this module. For information about logging levels, see the section on configuring logging properties in the Troubleshooting and support topic.
7. 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.
   a. 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.
   b. To create a new function selector configuration, click **New**. In the **New Function Selector Configuration** window, specify the **module**, **folder**, and **name** for the function selector configuration. Click **Next**.


Binding Resource Configuration

New Function Selector Configuration

Create a new function selector configuration. Specify the module, folder, namespace, and name for the function selector configuration.

Module: FFModule

Namespace: http://FFModule

☑ Use default namespace

Folder:

Name: * FunctionSelectorConfiguration

Figure 36. Creating a new function selector configuration

Note: The 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.

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.

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**.

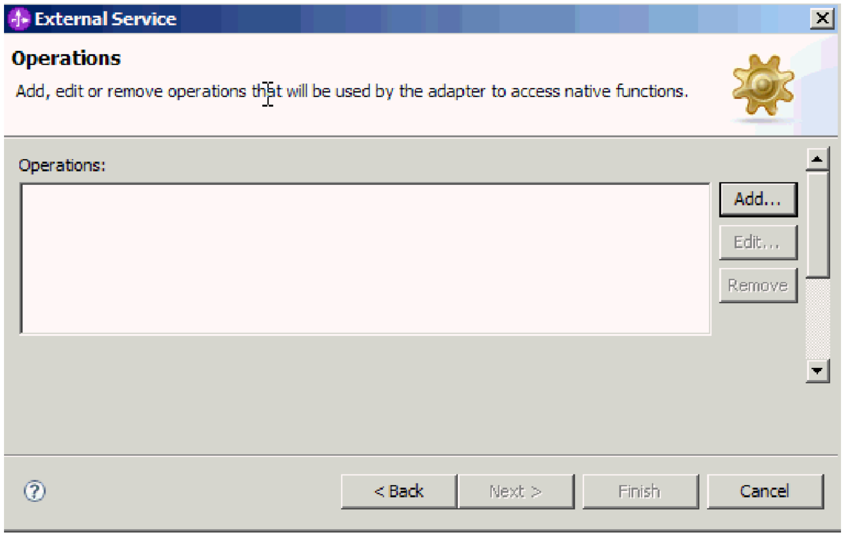
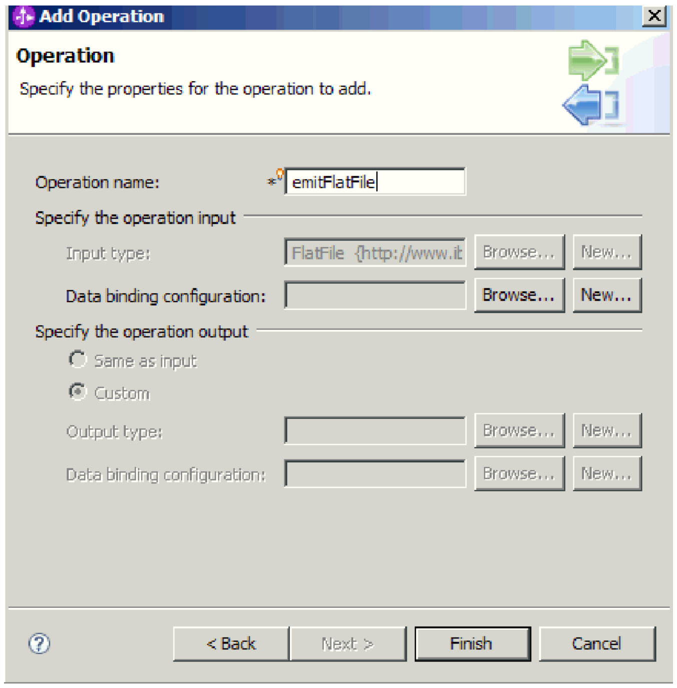

External Service
Operations
Add, edit or remove operations that will be used by the adapter to access native functions.
Operations:
Add...
Edit...
Remove
< Back
Next >
Finish
Cancel

Figure 37. 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, we selected Generic FlatFile business object.
3. Click **Next**. The Operation window displays the operation name, which is **emitFlatFile**. The emit operation is the only operation available during inbound processing.


Add Operation

Operation

Specify the properties for the operation to add.

Operation name: emitFlatFile

Specify the operation input

Input type: FlatFile {http://www.ib

Data binding configuration:

Specify the operation output

Same as input

Custom

Output type:

Data binding configuration:

Figure 38. 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.

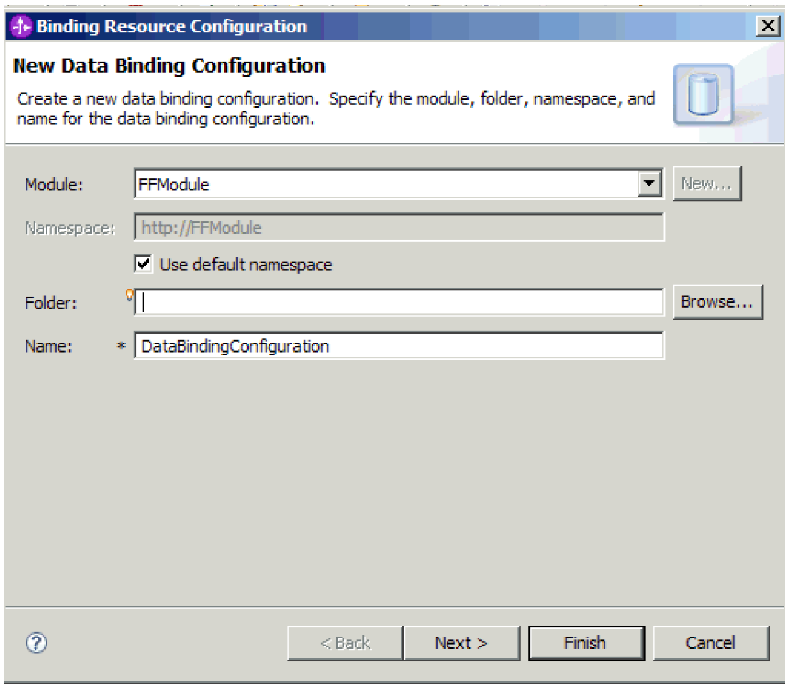
About this task

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 (in this example we used DataBindingConfiguration). Click **Next**.


Binding Resource Configuration

New Data Binding Configuration

Create a new data binding configuration. Specify the module, folder, namespace, and name for the data binding configuration.

Module: FFModule

Namespace: http://FFModule

☑ Use default namespace

Folder:

Name: * DataBindingConfiguration

Figure 39. 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.

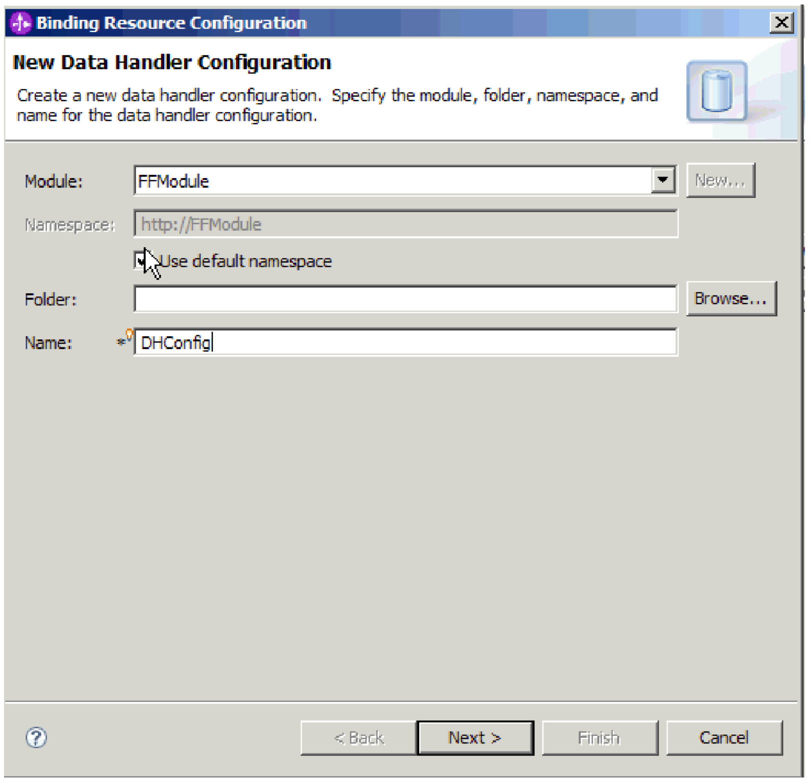
Note: Data handlers can be configured prior to running the external service wizard using WebSphere Integration Developer. To do this, select **New** → **Resource configuration** in WebSphere Integration Developer and complete the data handler screens described in this documentation.

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 (in this example we used DHConfig). You click **New** the first time you set the data handler. To use this data handler later, click **Browse**.


Binding Resource Configuration

New Data Handler Configuration

Create a new data handler configuration. Specify the module, folder, namespace, and name for the data handler configuration.

Module: FFModule

Namespace: http://FFModule

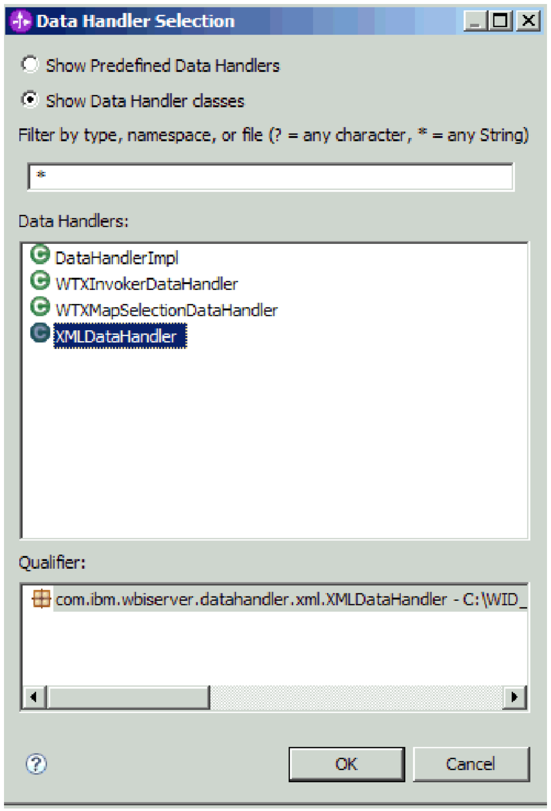
Use default namespace

Folder:

Name: DHConfig

Figure 40. 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 (in this example, we used XMLDataHandler). Click **OK**.


Data Handler Selection

Show Predefined Data Handlers [✓ Show Data Handler classes]

Filter by type, namespace, or file (? = any character, * = any String)

*

Data Handlers:

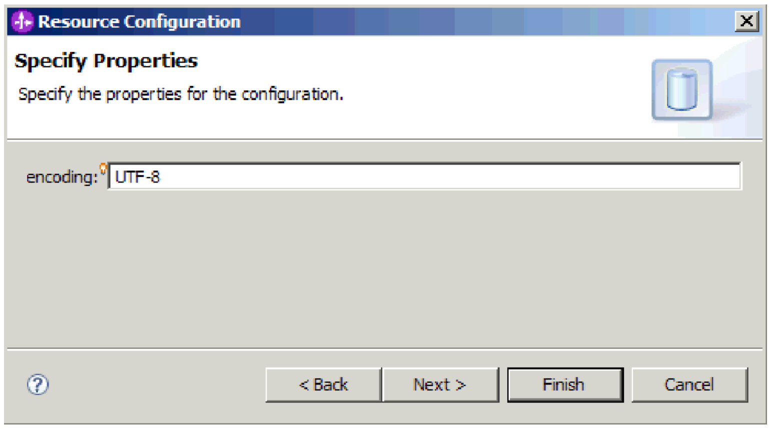
DataHandlerImpl
WTXInvokerDataHandler
WTXMapSelectionDataHandler
XMLDataHandler

Qualifier:

com.ibm.wbiserver.datahandler.xml.XMLDataHandler - C:\WID_

Figure 41. Selecting the data handler class

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


Resource Configuration

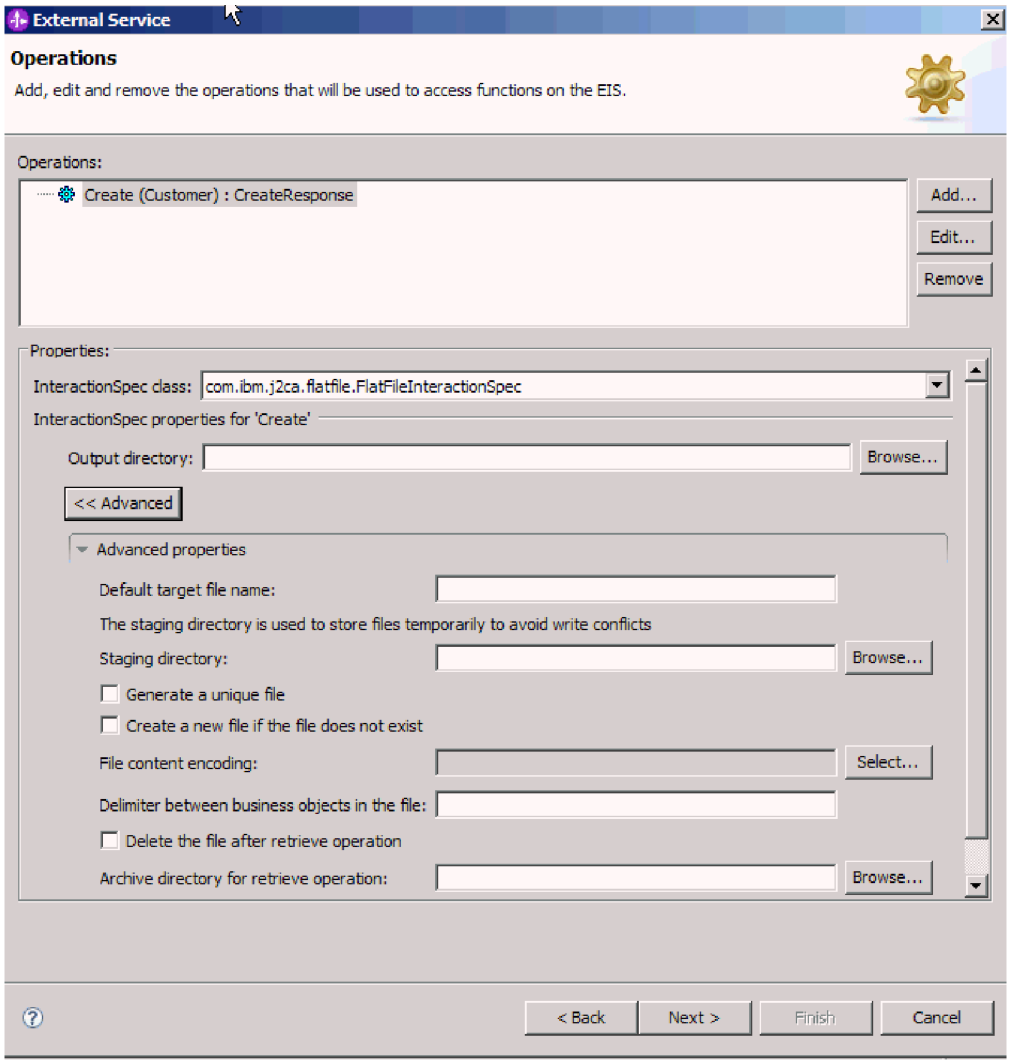
Specify Properties

Specify the properties for the configuration.

encoding: UTF-8

Figure 42. 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.


External Service

Operations

Add, edit and remove the operations that will be used to access functions on the EIS.

Operations:

Create (Customer) : CreateResponse

Properties:

InteractionSpec class: com.ibm.j2ca.flatfile.FlatFileInteractionSpec

InteractionSpec properties for 'Create'

Output directory:

Advanced properties

Default target file name:

The staging directory is used to store files temporarily to avoid write conflicts

Staging directory:

☐ Generate a unique file

☐ Create a new file if the file does not exist

File content encoding:

Delimiter between business objects in the file:

☐ Delete the file after retrieve operation

Archive directory for retrieve operation:

Figure 43. 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

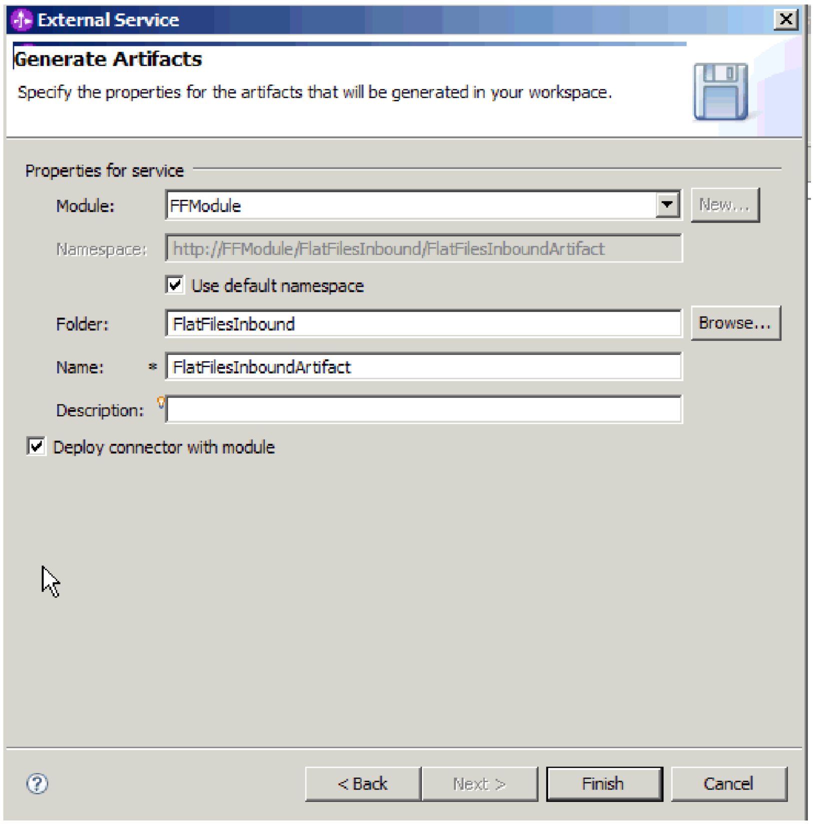
To set activation specification properties and generate artifacts for your module, you must have already configured data bindings and selected business objects.

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.



Specify the properties for the artifacts that will be generated in your workspace.

Module: FFModule

Namespace: http://FFModule/FlatFilesInbound/FlatFilesInboundArtifact

☑ Use default namespace

Folder: FlatFilesInbound

Name: * FlatFilesInboundArtifact

Description:

☑ Deploy connector with module

Figure 44. 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.

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.

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.

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.

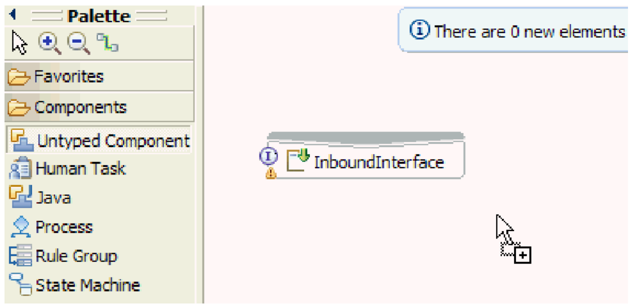


Figure 45. Adding a component to the Assembly Diagram

      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:

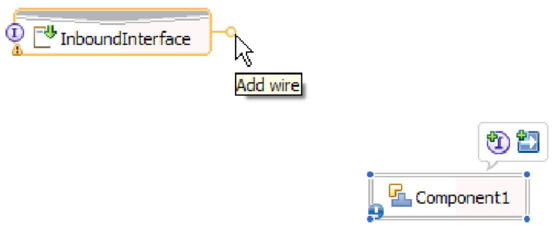


Figure 46. Selecting the wire icon

   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** → **Java**.

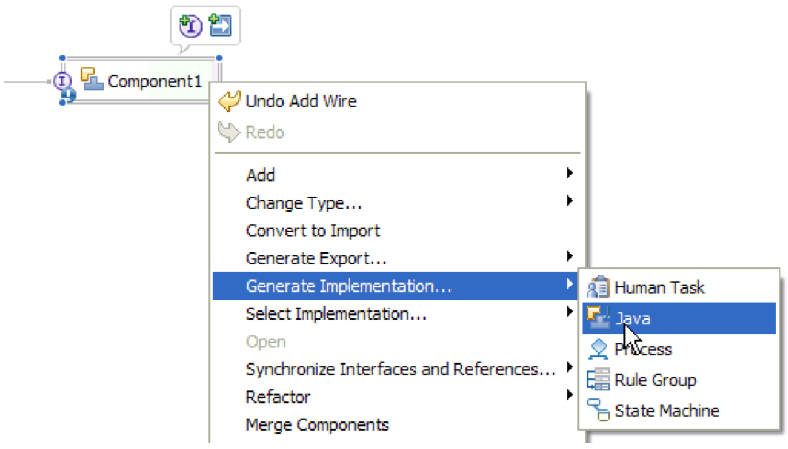


Figure 47. Generating a Java implementation

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 administrative console.

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

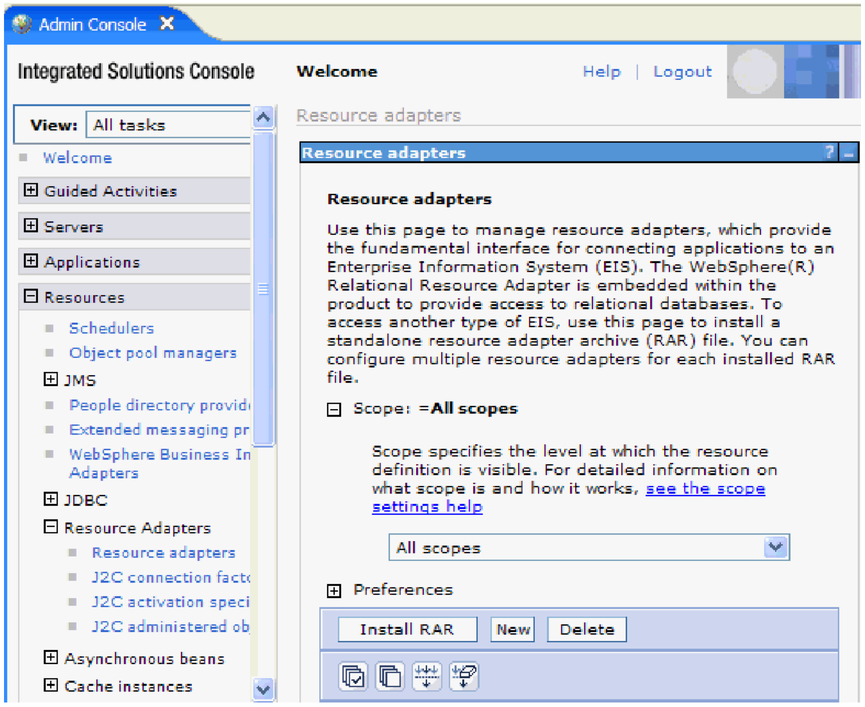
You must have set **Deploy connector project** to **On server for use by multiple adapters** in the Service Generation and Deployment Configuration window of the external service wizard.

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. From the Resource adapters page, click **Install RAR**.



Use this page to manage resource adapters, which provide the fundamental interface for connecting applications to an Enterprise Information System (EIS). The WebSphere(R) Relational Resource Adapter is embedded within the product to provide access to relational databases. To access another type of EIS, use this page to install a standalone resource adapter archive (RAR) file. You can configure multiple resource adapters for each installed RAR file.

Scope: =All scopes

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help

All scopes

Figure 48. The Install RAR button on the Resource adapters page

4. From 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. From 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 **J2EE**.
3. Select **EAR file** and click **Next**.

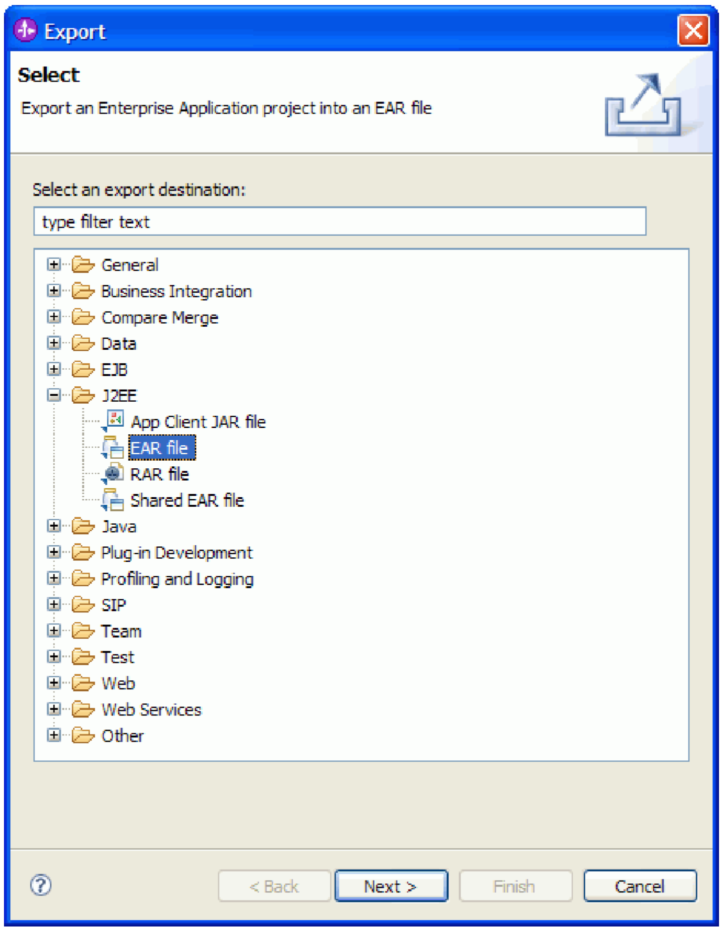


Figure 49. Selecting **EAR file** *from the Select window*

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.

What to do next

Install the module in the administrative console. This deploys the module to WebSphere Process Server.

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.

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 administrative console by right-clicking your server instance and selecting **Run administrative console**.
2. In the administrative console window, click **Applications** → **Install New Applications**.

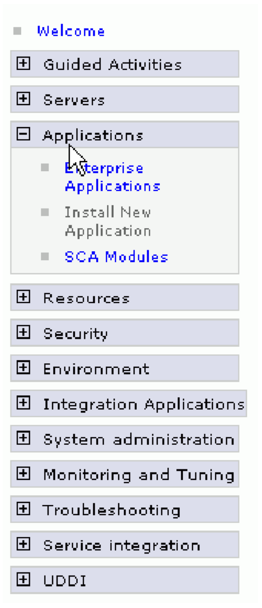


Figure 50. Preparing for the application installation window

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).

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.

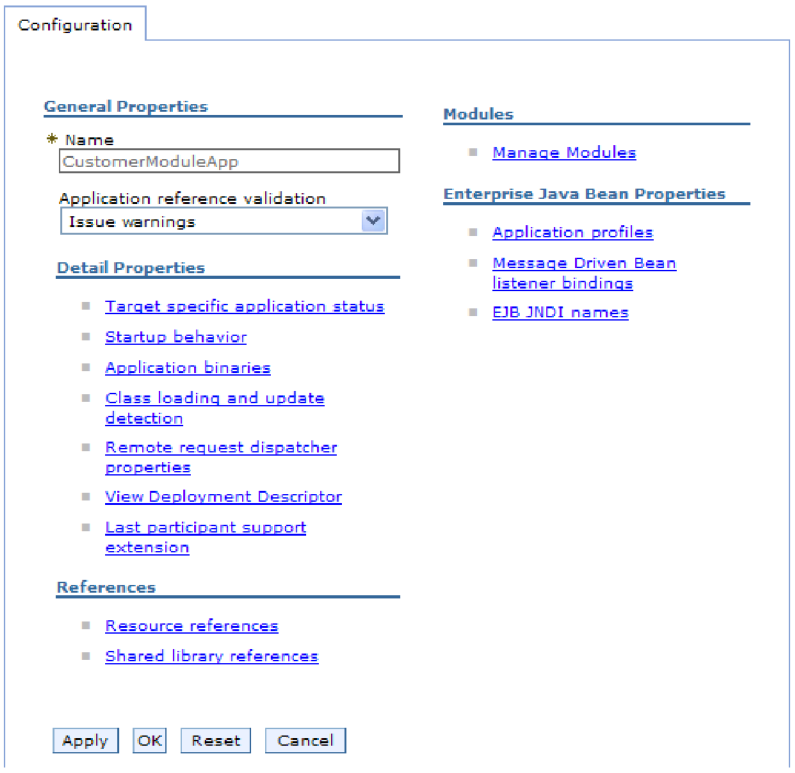
About this task

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.
4. Under **Modules**, click **Manage Modules**.


Configuration

General Properties

* Name

CustomerModuleApp

Application reference validation

Issue warnings

Detail Properties

- Target specific application status
- Startup behavior
- Application binaries
- Class loading and update detection
- Remote request dispatcher properties
- View Deployment Descriptor
- Last participant support extension

References

- Resource references
- Shared library references

Modules

- Manage Modules

Enterprise Java Bean Properties

- Application profiles
- Message Driven Bean listener bindings
- EJB JNDI names

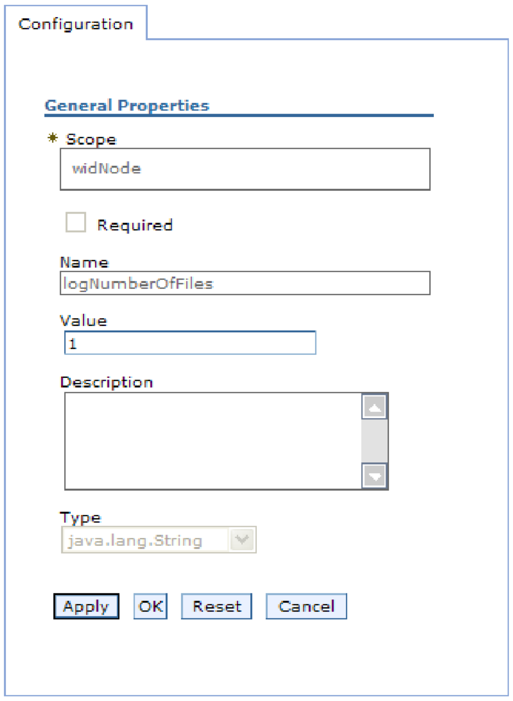
Figure 51. The Manage Modules selection in the Configuration tab

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 **Custom properties**.
8. For each property you want to change, perform the following steps.

   Note: See "Resource adapter properties" on page 125 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:


Configuration

General Properties

* Scope
widNode

Required

Name
logNumberOfFiles

Value
1

Description

Type
java.lang.String

Figure 52. 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**.

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.

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. From the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change.
4. Under **Modules**, click **Manage Modules**.

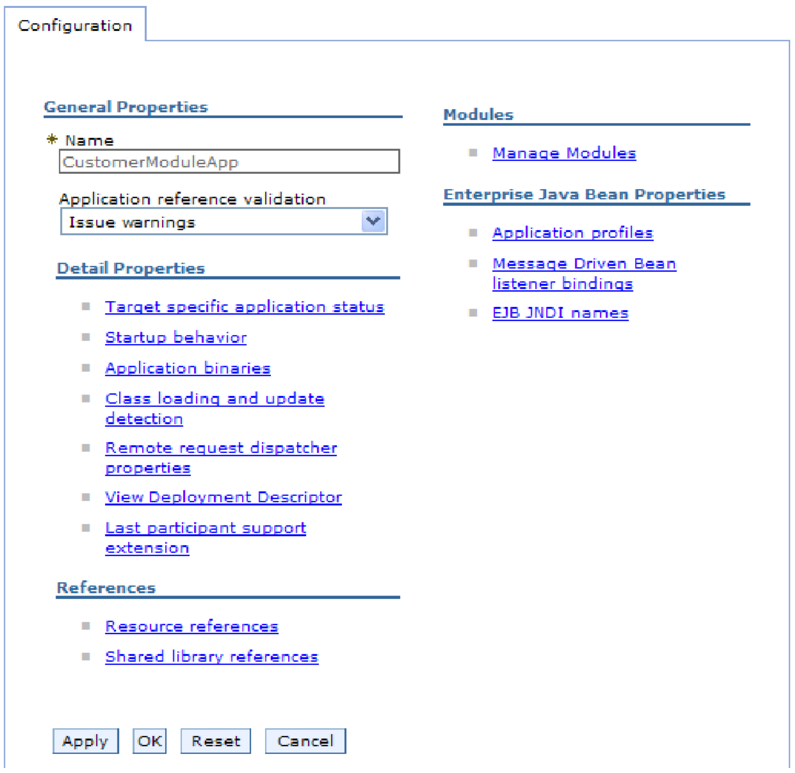

Configuration
General Properties
* Name
CustomerModuleApp
Application reference validation
Issue warnings
Detail Properties
Target specific application status
Startup behavior
Application binaries
Class loading and update detection
Remote request dispatcher properties
View Deployment Descriptor
Last participant support extension
References
Resource references
Shared library references
Modules
Manage Modules
Enterprise Java Bean Properties
Application profiles
Message Driven Bean listener bindings
EJB JNDI names
Apply
OK
Reset
Cancel

Figure 53. 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 connection factories**.
8. Click the name of the connection factory associated with your adapter module.
9. From 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 122 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.

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.

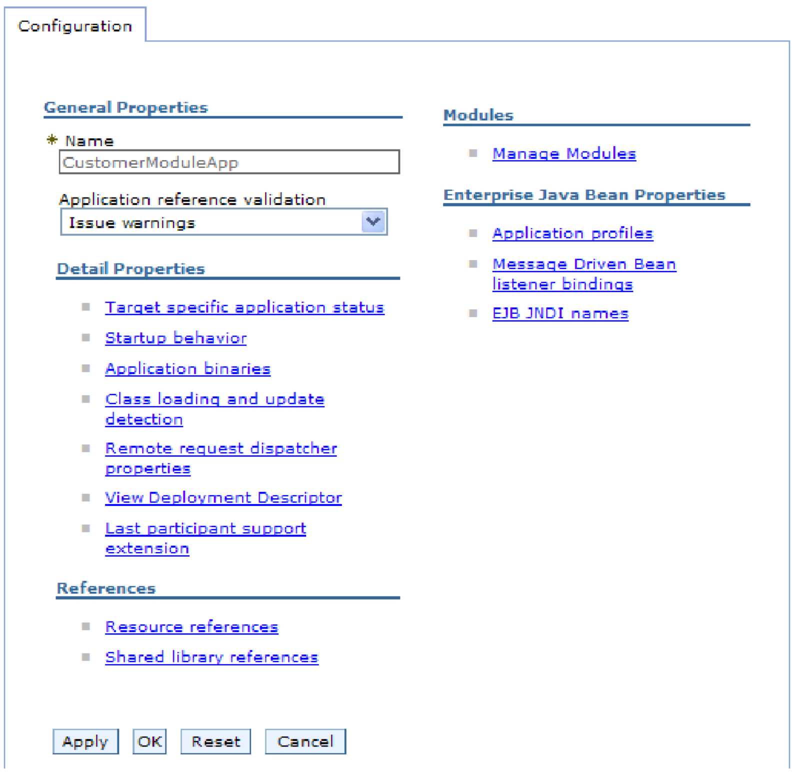
About this task

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**.


Configuration

General Properties

* Name
CustomerModuleApp

Application reference validation
Issue warnings

Detail Properties

- Target specific application status
- Startup behavior
- Application binaries
- Class loading and update detection
- Remote request dispatcher properties
- View Deployment Descriptor
- Last participant support extension

References

- Resource references
- Shared library references

Modules

- Manage Modules

Enterprise Java Bean Properties

- Application profiles
- Message Driven Bean listener bindings
- EJB JNDI names

Figure 54. 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 135 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.

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.

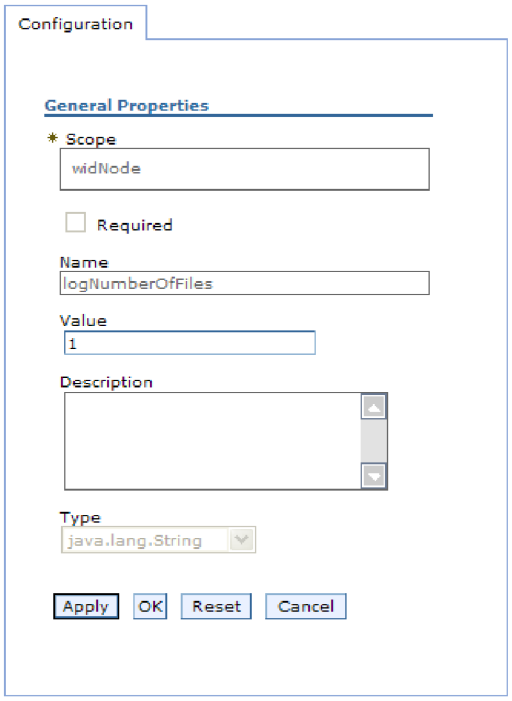
Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. From the Resource adapters page, click **IBM WebSphere Adapter for Flat Files**.
4. From 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 125 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:


Configuration

General Properties

* Scope: widNode

Required: ☐

Name: logNumberOfFiles

Value: 1

Description:

Type: java.lang.String

Figure 55. 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.

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

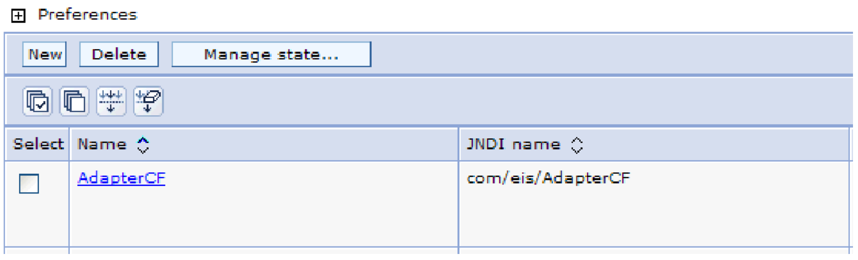
1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. From the Resource adapters page, click **IBM WebSphere Adapter for Flat Files**.
4. From the **Additional Properties** list, click **J2C connection factories**.
5. If you are going to use an existing connection factory, skip ahead to step 6.

   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.



Select	Name	JNDI name
☐	AdapterCF	com/eis/AdapterCF

Figure 56. The list of connection factories

6. From the list of connection factories, click the one you want to use.
7. From 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 122 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.

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. From the Resource adapters page, click **IBM WebSphere Adapter for Flat Files**.
4. From the **Additional Properties** list, click **J2C activation specifications.**.
5. If you are going to use an existing activation specification, skip ahead to step 6.

   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. From the list of activation specifications, click the one you want to use.

7. From 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 135 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.

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:

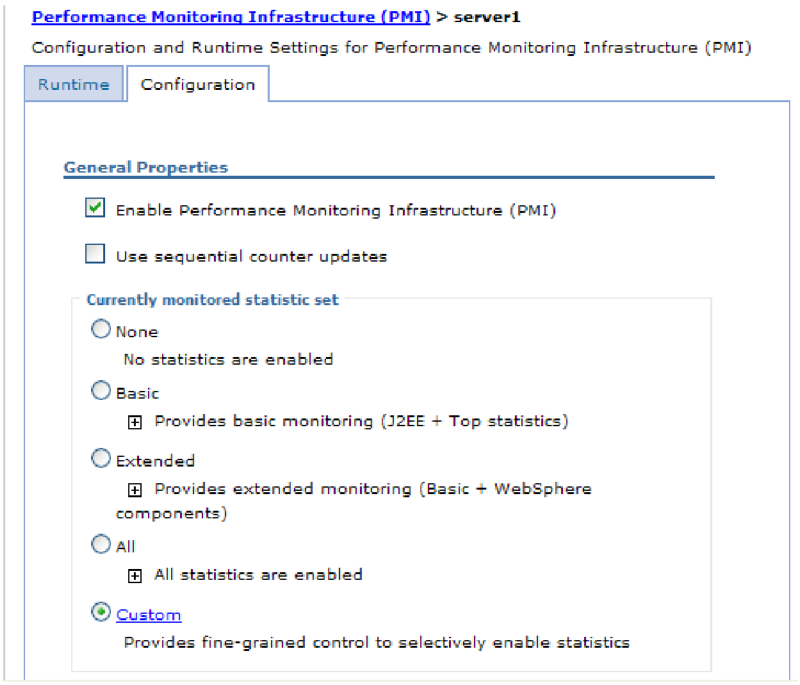
```
*=info: WBILocationMonitor.CEI.ResourceAdapter.
*=finest: WBILocationMonitor.LOG.ResourceAdapter.*=finest:
```

For instructions on setting the trace level, refer to "Enabling tracing with the Common Event Infrastructure (CEI)" on page 101.

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, then select the **Enable Performance Monitoring (PMI)** check box.
   d. Select **Custom** to selectively enable or disable statistics.


Performance Monitoring Infrastructure (PMI) > server1

Configuration and Runtime Settings for Performance Monitoring Infrastructure (PMI)

Runtime | Configuration

General Properties

☑ Enable Performance Monitoring Infrastructure (PMI)

☐ Use sequential counter updates

Currently monitored statistic set

○ None
No statistics are enabled

○ Basic
⊞ Provides basic monitoring (J2EE + Top statistics)

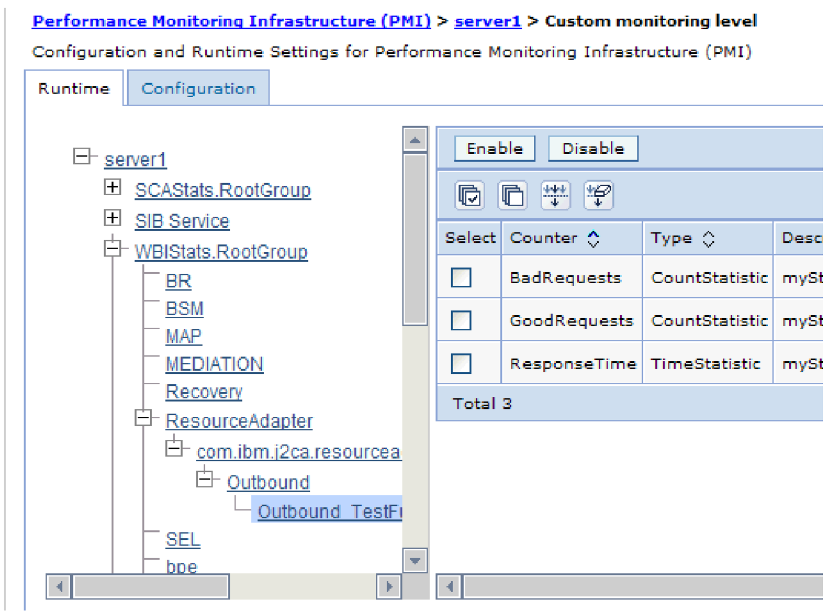
○ Extended
⊞ Provides extended monitoring (Basic + WebSphere components)

○ All
⊞ All statistics are enabled

● Custom
Provides fine-grained control to selectively enable statistics

Figure 57. 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.


Performance Monitoring Infrastructure (PMI) > server1 > Custom monitoring level

Configuration and Runtime Settings for Performance Monitoring Infrastructure (PMI)

Select	Counter	Type	Desc
☐	BadRequests	CountStatistic	mySt
☐	GoodRequests	CountStatistic	mySt
☐	ResponseTime	TimeStatistic	mySt

Total 3

Figure 58. Runtime tab used for configuring PMI

e. Click **WBIStats.RootGroup**. This is a PMI submodule for data collected in the root group. This example uses the name WBIStats for the root group.
f. Click **ResourceAdapter**. This is a submodule 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.

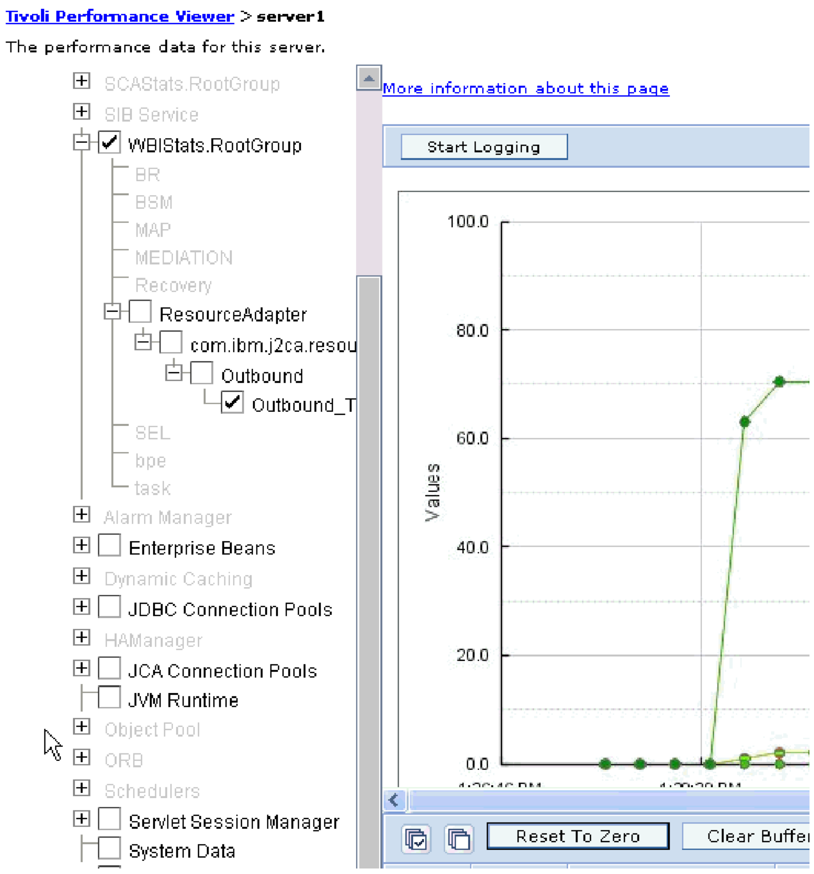


Figure 59. Adapter performance statistics, using graph view

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.

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 all of the 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.

Troubleshooting and support

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

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.

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 take 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.base.***.
   - 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.

Logging Level	Description
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 as a result of 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 primarySetKey attributes as shown in Figure 60 on page 106.

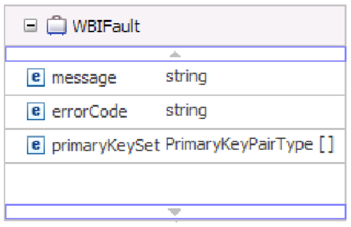


Figure 60. The structure of the WBIFault business object

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 wizard creates the following fault business objects:

- 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, then the adapter throws this fault.

  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.
- MultipleMatchingRecordsFault

  When processing a Retrieve operation, the adapter throws this fault if the query returns more than one record for the keys specified. The business object for this fault has one property, matchCount, which is a string that contains the number of matches.

Configuring the module for fault processing

Before you can configure your module to support business faults, you must have used the external service wizard to configure your module.

To enable fault processing, you must modify the .import and WSDL files for your module. You can configure faults at either the binding level or the method level. If the changes are made at binding level, they apply to all methods in the import. If the changes are made at the method binding level, you can configure a different fault for each method.

Table 10 lists the fault name and fault binding for each fault. Use the fault name and fault binding class when you configure the module.

Table 10. The fault name and fault binding class for each fault

Fault name	**Associated fault binding class**
DUPLICATE_RECORD	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl
MULTIPLE_MATCHING_RECORDS	com.ibm.j2ca.extension.emd.runtime.MatchingFaultDataBinding
MISSING_DATA	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl
RECORD_NOT_FOUND	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl

1. Edit the .import file to configure the fault at either the binding or the method level.
   - To configure the faults at the binding level:
     a. In the binding section, add the faultSelector attribute and the name of the fault selector. The name of the fault selector is com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl.
     b. For each fault that you want to enable, add a <faultBinding> element. In the element, specify the fault name and the fault data binding class name from Table 10 on page 106.

        The following .import file shows the DUPLICATE_RECORD and RECORD_NOT_FOUND faults configured for all methods. **Bold face type** indicates changes made to enable fault handling.

```
<esbBinding xsi:type="eis:EISImportBinding"
   dataBindingType="com.ibm.j2ca.flatfile.emd.runtime.FlatFileBaseDataBinding"
   faultSelector="com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl">
    <resourceAdapter
       name="FFOutApp.IBM WebSphere Adapter for Flat Files"
       type="com.ibm.j2ca.flatfile.FlatFileResourceAdapter">
      <properties/>
    </resourceAdapter>
    <faultBinding
       fault="DUPLICATE_RECORD"
       faultBindingType="com.ibm.j2ca.extension.emd.runtime.MatchingFaultDataBinding"/>
    <faultBinding
       fault="RECORD_NOT_FOUND"
       faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl"/>
```

   - To configure the faults at the method level:
     a. In method binding section for the method you want to associate with the fault, add the name of the fault selector. The value for fault selector is com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl.
     b. Add the fault binding elements in the method binding section. Use the fault name and the corresponding fault data binding class name from Table 10 on page 106.

        The following .import file shows the DUPLICATE_RECORD and RECORD_NOT_FOUND faults configured for the createCUSTOMER method. **Bold face type** indicates changes made to enable fault handling.

```
<methodBinding
   inDataBindingType="com.ibm.xmlns.prod.wbi.j2ca.flatfile.customerbg.CustomerBGDataBin
ding"
   method="createCUSTOMER"
   outDataBindingType="com.ibm.xmlns.prod.wbi.j2ca.flatfile.customerbg.CustomerBGDataBind
ing"
   faultSelector="com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl">
     <interaction>
       <properties>
         <functionName>Create</functionName>
       </properties>
     </interaction>
     <faultBinding
         fault="DUPLICATE_RECORD"
         faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl"/>
     <faultBinding
         fault="RECORD_NOT_FOUND"
         faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl"/>
</methodBinding>
```

2. Determine the target namespaces for your faults. For each fault that you want to enable, determine the namespace as follows:
   a. Open the fault schema (XSD file) in a text editor.
   b. Locate the target namespace. The target namespace is shown in **bold face type** in the following portion of a fault schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://com/ibm/j2ca/fault/afcfault"
   xmlns:basefault="http://com/ibm/j2ca/fault">
<import namespace="http://com/ibm/j2ca/fault" schemaLocation="WBIFault.xsd"/>

. . .
```

The faults can all have the same target namespace or they can have different target namespaces.

3. Edit the WSDL file to declare the faults for the service. A sample WSDL file with these changes highlighted is shown at the end of the list.
   a. In the <definitions> element, add a namespace for each fault namespace, using the information you obtained from the fault schema files. If all your fault schemas have the same targetNamespace, add only one alias. If they have different targetNamespaces, add an alias for each unique namespace.
   b. Create an <xsd:import> element to import the schema for each fault you want to enable.
   c. Declare import statements for each fault type. Make sure that you are using the correct alias defined in step 3a to resolve the complex type in `type=`*`alias`*`:`*`faultBOName`*`.xsd`.
   d. Declare the message tags for each of the fault types.
   e. Add the fault declaration to each method where faults should be handled.

   The following WSDL file defines the DUPLICATE_RECORD and RECORD_NOT_FOUND faults. **Bold face type** indicates changes made to enable fault handling.

```
          <definitions
           xmlns="http://schemas.xmlsoap.org/wsdl/"
           xmlns:CustomerBG="http://www.ibm.com/xmlns/prod/wbi/j2ca/flatfile/customerbg"
           xmlns:intf="http://FFOut/FlatFileOutboundInterface"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
Step 3a    xmlns:fault="http://com/ibm/j2ca/fault/afcfault"
            name="FlatFileOutboundInterface.wsdl"
            targetNamespace="http://FFOut/FlatFileOutboundInterface">
            <types>
              <xsd:schema
               xmlns:tns="http://FFOut/FlatFileOutboundInterface"
               xmlns:xsd1="http://www.ibm.com/xmlns/prod/wbi/j2ca/flatfile/customerbg"
               elementFormDefault="qualified"
               targetNamespace="http://FFOut/FlatFileOutboundInterface"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
              <xsd:import
               namespace="http://www.ibm.com/xmlns/prod/wbi/j2ca/flatfile/customerbg"
               schemaLocation="CustomerBG.xsd"/>
Step 3b     <xsd:import namespace="http://com/ibm/j2ca/fault/afcfault"
                  schemaLocation=" DuplicateRecordFault.xsd"/>
             <xsd:import namespace="http://com/ibm/j2ca/fault/afcfault"
                  schemaLocation="RecordNotFoundFault.xsd"/>

          . . .
```

Step 3c on page 108

```
     <xsd:element name="duplicateRecordFaultX">
          <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="duplicateRecordFaultElement"
                          type="fault:DuplicateRecordFault"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>

        <xsd:element name="recordNotFoundFaultX">
          <xsd:complexType>
           <xsd:sequence>
              <xsd:element name="recordNotFoundFaultElement"
                          type="fault:RecordNotFoundFault"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:schema>
    </types>

    . . .
```

Step 3d on page 108

```
  <message name="duplicateRecordFault">
    <part element="intf:duplicateRecordFaultX"
          name="duplicateRecordFaultPart"/>
  </message>
  <message name="recordNotFoundFault">
    <part element="intf:recordNotFoundFaultX"
          name="recordNotFoundFaultPart"/>
  </message>
  <portType name="FlatFileOutboundInterface">

    . . .

  <operation name="createCUSTOMER">
  <input message="intf:createCUSTOMERRequest"
         name="createCUSTOMERRequest"/>
  <output message="intf:createCUSTOMERResponse"
          name="createCUSTOMERResponse"/>
```

Step 3e on page 108

```
  <fault message="intf:duplicateRecordFault"
            name="duplicateRecordFaultFault" />
  <fault message="intf:recordNotFoundFault"
         name="recordNotFoundFaultFault" />
  </operation>
  </portType>
  </definitions>
```

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 whitepapers
- 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. For information about installing or using IBM Support Assistant, visit this address:

http://www.ibm.com/software/support/isa/

Chapter 8. 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.

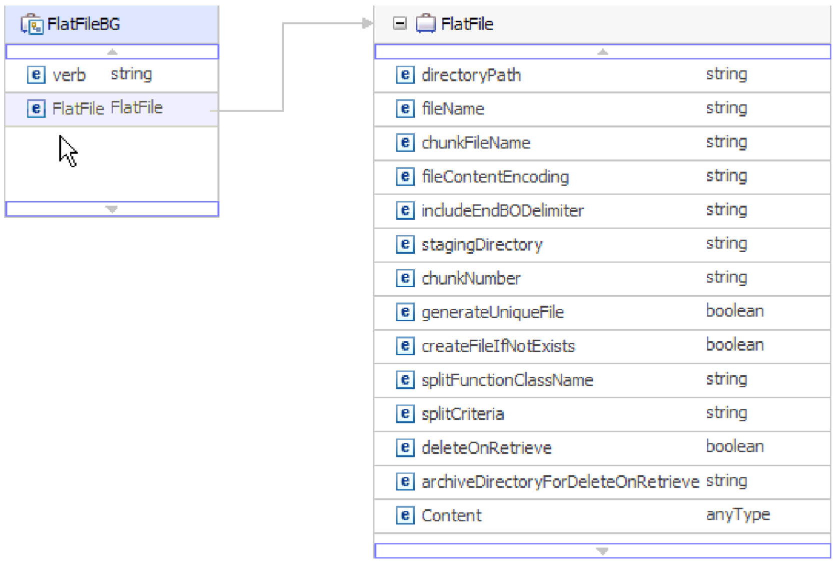
Business object structures

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:


FlatFileBG

verb	string
FlatFile	FlatFile

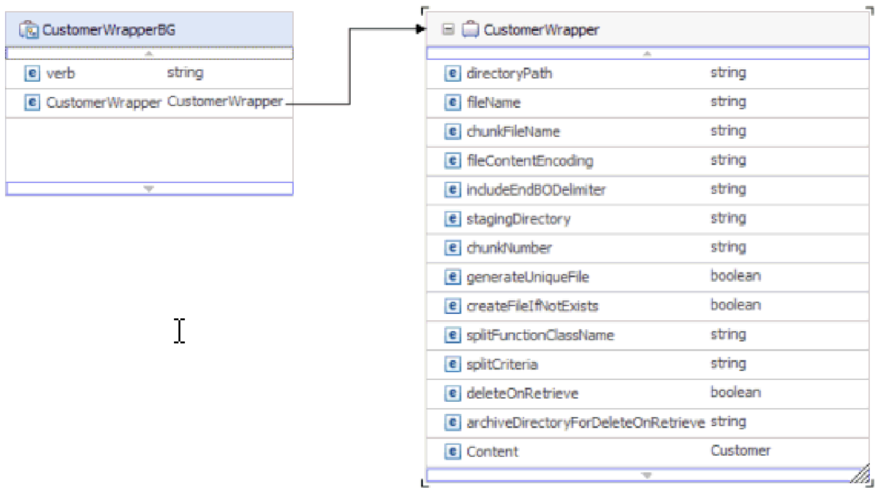
FlatFile

directoryPath	string
fileName	string
chunkFileName	string
fileContentEncoding	string
includeEndBODelimiter	string
stagingDirectory	string
chunkNumber	string
generateUniqueFile	boolean
createFileIfNotExists	boolean
splitFunctionClassName	string
splitCriteria	string
deleteOnRetrieve	boolean
archiveDirectoryForDeleteOnRetrieve	string
Content	anyType

Figure 61. 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:



CustomerWrapperBG

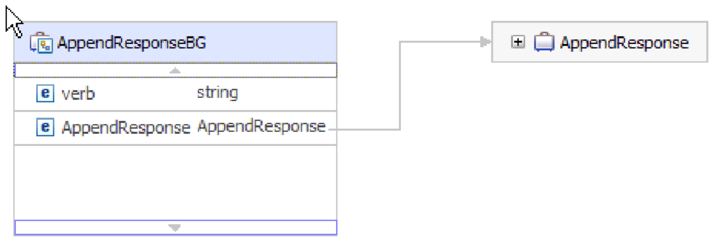
verb	string
CustomerWrapper	CustomerWrapper

CustomerWrapper

directoryPath	string
fileName	string
chunkFileName	string
fileContentEncoding	string
includeEndBODelimiter	string
stagingDirectory	string
chunkNumber	string
generateUniqueFile	boolean
createFileIfNotExists	boolean
splitFunctionClassName	string
splitCriteria	string
deleteOnRetrieve	boolean
archiveDirectoryForDeleteOnRetrieve	string
Content	Customer

Figure 62. The CustomerWrapperBG business object structure

Append operation response business object

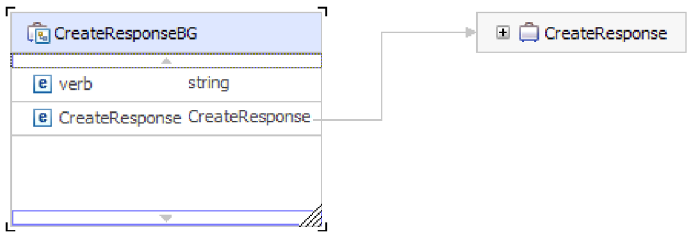


AppendResponseBG

verb	string
AppendResponse	AppendResponse

Figure 63. Structure of the Append operation response business object

Create operation response business object



CreateResponseBG

verb	string
CreateResponse	CreateResponse

Figure 64. Structure of the Create operation response business object

Exists operation response business object

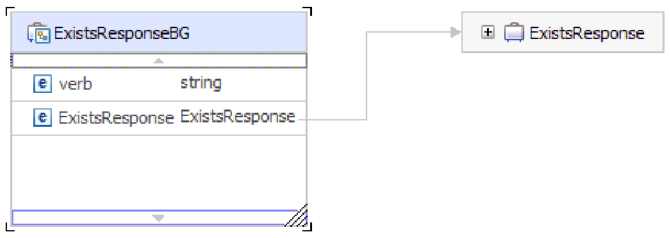


Figure 65. Structure of the Exists operation response business object

List operation response business object

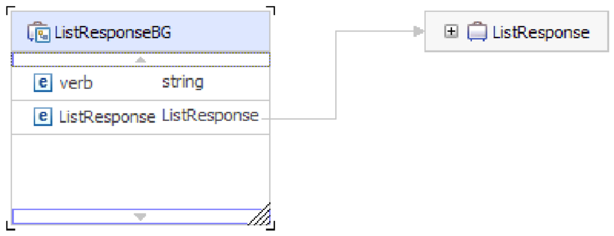


Figure 66. Structure of the List operation response business object

Overwrite operation response business object

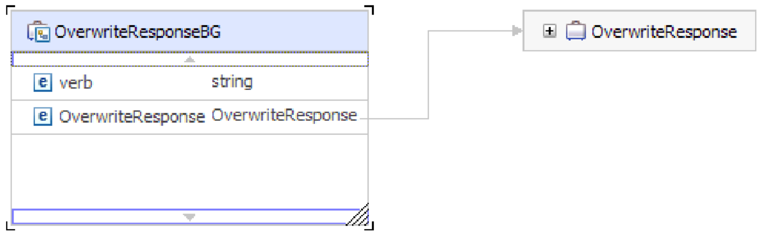


Figure 67. Structure of the Overwrite operation response business object

Retrieve operation response business object

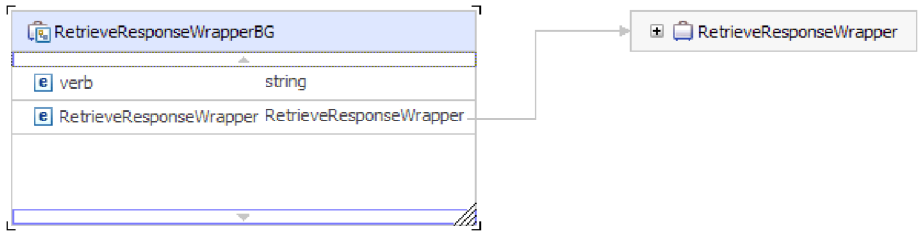

RetrieveResponseWrapperBG
verb string
RetrieveResponseWrapper RetrieveResponseWrapper
RetrieveResponseWrapper

Figure 68. 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 11. 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 12. 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 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.

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 using WebSphere Integration Developer or the WebSphere Process Server 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	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. Removing a default value from a required field on the external service wizard *will not change that default value*. 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. Possible values are **Yes** and **No**. 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, • Yes, when the EventQueryType property is set to Dynamic • Yes, for Oracle databases
Possible values	Lists and describes the possible values that you can select for the property.
Default	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. The word `None` is an acceptable default value, and does not mean that there is no default value.
Unit of measure	Specifies how the property is measured, for example in kilobytes or seconds.

Row	Explanation
Property type	Describes the property type. Valid property types include the following: • Boolean • String • Integer
Usage	Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented: For WebSphere Application Server version 6.40 or earlier, the password: • Must be uppercase • Must be 8 characters in length For versions of WebSphere Application Server later than 6.40, the password: • Is not case sensitive • Can be up to 40 characters in length. This section lists other properties that affect this property or that are affected by this property and describes the nature of the conditional relationship.
Example	Provides sample property values, for example: ″If Language is set to JA (Japanese), Codepage number is set to `8000`″.
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 **Yes** and **No**.
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 **Yes** and **No**.

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 118.

Table 13. Connection properties for the external service wizard

Property name in the wizard	Description
“Bidi format string” on page 120	The bidi format string of the content data
“Data binding” on page 120	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 120	During inbound processing, the name of the function selector configuration to be used.
“Log file output location” on page 121	The full path name of the log file generated by the external service wizard

Table 13. Connection properties for the external service wizard (continued)

Property name in the wizard	**Description**
"Logging level" on page 121	The level of logging to be used by the adapter
"NameSpace" on page 122	The namespace of the business object that is generated
"Operation name" on page 122	The operation defined in the external service wizard
"Processing Direction" on page 122	The processing direction, `Inbound` or `Outbound`

Bidi format string

The bidi format string of the content data.

Table 14. 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 15. Data binding details

Required	No
Default	Use default data binding 'FlatFileBaseDataBinding' for all operations
Usage	The value of this property can be: • Use default data binding 'FlatFileBaseDataBinding' for all operations • Use a data binding configuration for all operations • Specify a data binding for each operation
Globalized	No
Bidi supported	No

Function selector

During inbound processing, the name of the function selector configuration to be used.

Table 16. Function selector details

Required	Yes
Default	`FilenameFunctionSelector`
Property type	String

Table 16. Function selector details (continued)

Usage	The function selector returns the appropriate operation to be called on the service. The adapter provides two function selectors, `FilenameFunctionSelector` and `EmbeddedNameFunctionSelector`. • `FilenameFunctionSelector` is a rule-based function selector that matches a regular expression on a file name to an object name. Use `FilenameFunctionSelector` for generic FlatFile business objects, where the object name cannot be determined from the event file. `FilenameFunctionSelector` is represented in properties as a two-column table with *N* rows. For any event file with a `.txt` extension, the corresponding object name is FlatFile, and the endpoint method name generated by the function selector is emitFlatFile. You must set this same name in the EISFunctionName property after you add the operation. 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. • Use `EmbeddedNameFunctionSelector` for content-specific business objects, where the object name is embedded in the event file. `EmbeddedNameFunctionSelector` returns the function name based on the desired content data, not the wrapper. For example, if the content-specific business object is CustomerWrapperBG, the function returned by the function selector is emitCustomer. You must configure `EmbeddedNameFunctionSelector` 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.
Globalized	Yes
Bidi supported	No

Log file output location

The full path name of the log file generated by the external service wizard.

Table 17. 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 18. Logging level details

Required	No

Possible values	`Severe`
	`Warning`
	`Audit`
	`Info`
	`Config`
	`Detail`

Default	`Severe`
Property type	List of values
Globalized	No

Table 18. Logging level details (continued)

Bidi supported	No

NameSpace

The namespace of the business object that is generated.

Table 19. NameSpace details

Required	Yes
Default	http://www.ibm.com/xmlns/prod/websphere/j2ca/flatfile
Property type	String
Globalized	Yes
Bidi supported	No

Operation name

The name you give to the operation defined for this module.

Table 20. 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 21. Processing Direction details

Required	Yes
Possible values	`Outbound` `Inbound`
Default	`Outbound`
Property type	String
Globalized	No
Bidi supported	No

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 activation specification 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 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 118.

Note: The external service wizard refers to these properties as managed connection factory properties and the WebSphere Process Server administrative console refers to them as (J2C) connection factory properties.

Table 22. Managed connection factory properties

Property name		
In the wizard	**In the administrative console**	**Description**
"Default target file name"	OutputFileName	The name of the file that is created in the output directory
"Output directory"	OutputDirectory	The full path name of the directory where the adapter creates files during outbound operations
"Sequence file" on page 124	FileSequenceLog	The full path name of the file where sequences are stored during outbound Create operations
"Staging directory" on page 124	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

Default target file name

The name of the file that is created in the output directory.

Table 23. Default target file name details

Required	No
Default	None
Property type	String
Usage	If a value for OutputFileName is specified in the record object, this value is overridden.
Globalized	Yes
Bidi supported	Yes

Output directory

The full path name of the directory where the adapter creates files during outbound operations.

Table 24. Output directory details

Required	No
Default	None
Property type	String
Usage	The output directory is used by the adapter to write the final output files for Create, Append, and Overwrite operations.
Globalized	Yes

Table 24. Output directory details (continued)

Bidi supported	Yes

Sequence file

This property specifies the full path name of the file where sequences are stored during outbound Create operations.

Table 25. Sequence file details

Required	No
Default	None
Property type	String
Usage	The adapter checks the file sequence log when it receives a Create request to see whether a file with that 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 *n* 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*n*. **Note:** All sequences begin with 1. If this property is not specified and the adapter receives a request to create a file with a file name that already exists, the adapter generates a `DuplicateRecordException` error. To generate file sequencing for a particular type of request, set the output directory and the file name at the managed connection level, to avoid having to set them repeatedly in the business object for each request. **Note:** The directory path and file name, if specified in the business object, take precedence over the values specified at the managed connection level When the adapter is working in a clustered environment, make sure that the sequence file specified by the FileSequenceLog property 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 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 continues to increment after an adapter restart. You can reset the file sequence by changing the sequence value in the sequence file.
Globalized	Yes
Bidi supported	Yes

Staging directory

The full path name of the directory where the adapter temporarily stores the initial output files for Create and Overwrite operations, to avoid write conflicts during outbound processing.

Table 26. Staging directory details

Required	No
Default	None
Property type	String

Table 26. Staging directory details (continued)

Usage	If this property is specified, the output file is first written to a staging directory and then renamed to the output directory.
Globalized	Yes
Bidi supported	Yes

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 118.

Table 27. Resource adapter properties for the Adapter for Flat Files

Name		
In the wizard	**In the administrative console**	**Description**
Adapter ID	AdapterID	Identifies the adapter instance for CEI and PMI events with respect to logging and tracing.
(Not available)	Enable HA support	Do not change this property.
(Not available)	LogFileMaxSize	Supported for compatibility with earlier versions
(Not available)	LogFilename	Supported for compatibility with earlier versions
(Not available)	LogNumberOfFiles	Supported for compatibility with earlier versions
(Not available)	TraceFileMaxSize	Supported for compatibility with earlier versions
(Not available)	TraceFileName	Supported for compatibility with earlier versions
(Not available)	TraceNumberOfFiles	Supported for compatibility with earlier versions

Adapter ID to use for logging and tracing (AdapterID)

Use this property to identify a specific deployment, or instance, of the adapter.

Table 28. Adapter ID to use for logging and tracing details

Required	Yes
Default	CWYFF_FlatFile

Table 28. Adapter ID to use for logging and tracing details (continued)

Property type	String
Usage	This property is used to identify the adapter instance for PMI events. If you are deploying multiple instances of an adapter, set this property to a unique value for each adapter instance. For inbound processing this property is retrieved from the resource adapter properties. For outbound processing, it is retrieved form the managed connection factory properties.
Globalized	Yes
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 118.

Table 29. Interaction specification properties

Property name		
In the wizard	**In the administrative console**	**Description**
“Archive directory for retrieve operation” on page 127	ArchiveDirectoryFor DeleteOnRetrieve	The directory where retrieved files are stored before they are deleted, if the DeleteOnRetrieve property is set to `true`
“Create a new file if the file does not exist” on page 127	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 128	OutputFileName	The name of the output file that is created or modified
“Delete the file after retrieve operation” on page 128	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 128	IncludeEndBODelimiter	The file content is appended with this value.

Table 29. Interaction specification properties (continued)

Property name		Description
In the wizard	In the administrative console	
“File content encoding” on page 128	FileContentEncoding	The encoding that is used to write to the file
“Generate a unique file” on page 129	GenerateUniqueFile	Specifies that the adapter will create a unique file during Create, Append, and Overwrite operations
“Output directory” on page 129	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 129	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 130	SplittingFunctionClassName	Specifies how the retrieved file is to be split, by delimiter or by size, during an outbound Retrieve operation
“Staging directory” on page 130	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 30. 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 31. Create a new file if the file does not exist details

Required	No

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 error `RecordNotFoundException`.
Globalized	No
Bidi supported	No

Default target file name

The name of the output file that is created or modified.

Table 32. 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 33. 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.
Globalized	No
Bidi supported	No

Delimiter between business objects in the file

The file content is appended with this value.

Table 34. Delimiter between business objects in the file details

Required	No
Default	None
Property type	String
Usage	This property is used during outbound Create, Append, and Overwrite operations.
Globalized	Yes
Bidi supported	Yes

File content encoding

The encoding that is used to write to the file.

Table 35. File content encoding details

Required	No
Possible values	Any Java-supported encoding set; for example, UTF-8.
Default	None

Table 35. File content encoding details (continued)

Property type	String
Usage	If you do not specify this property, the adapter uses the regional settings of the operating system.
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 36. 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. During Append and Overwrite operations, if this property is set to `True` and the CreateFileIfNotExists property is set to `True`, the adapter creates a unique file.
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 37. 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 38. Specify criteria to split file content details

Required	No
Possible values	A delimiter or a valid number
Default	`0`

Table 38. Specify criteria to split file content details (continued)

Property type	String
Usage	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: • If the SplittingFunctionClassName property is set to `com.ibm.j2ca.utils.filesplit.SplitByDelimiter`, the SplitCriteria property must contain the delimiter that separates the business objects in the retrieved file. • If the SplittingFunctionClassName property is set to `com.ibm.j2ca.utils.filesplit.SplitBySize`, 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. If the SplitCriteria property is set to 0, chunking is disabled. 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 `\r`, and the SplitCriteria property must contain `\r`. The platform-specific newline characters are as follows: Macintosh - **\r** Microsoft Windows - **\r\n** UNIX - **\n** 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 `\;`. For example, if the delimiter given is `##\;##`. it is evaluated to `##;##`.
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 39. 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 40. 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

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 WebSphere Process Server 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	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. Removing a default value from a required field on the external service wizard *will not change that default value*. 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. Possible values are **Yes** and **No**. 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, • Yes, when the EventQueryType property is set to Dynamic • Yes, for Oracle databases
Possible values	Lists and describes the possible values that you can select for the property.
Default	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. The word `None` is an acceptable default value, and does not mean that there is no default value.

Row	Explanation
Unit of measure	Specifies how the property is measured, for example in kilobytes or seconds.
Property type	Describes the property type. Valid property types include the following: • Boolean • String • Integer
Usage	Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented: For WebSphere Application Server version 6.40 or earlier, the password: • Must be uppercase • Must be 8 characters in length For versions of WebSphere Application Server later than 6.40, the password: • Is not case sensitive • Can be up to 40 characters in length. This section lists other properties that affect this property or that are affected by this property and describes the nature of the conditional relationship.
Example	Provides sample property values, for example: ″If Language is set to JA (Japanese), Codepage number is set to `8000`″.
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 **Yes** and **No**.
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 **Yes** and **No**.

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 118.

Table 41. Connection properties for the external service wizard

Property name in the wizard	Description
“Bidi format string” on page 133	The bidi format string of the content data
“Data binding” on page 133	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 133	During inbound processing, the name of the function selector configuration to be used.

Table 41. Connection properties for the external service wizard (continued)

Property name in the wizard	Description
“Log file output location” on page 134	The full path name of the log file generated by the external service wizard
“Logging level” on page 134	The level of logging to be used by the adapter
“NameSpace” on page 135	The namespace of the business object that is generated
“Operation name” on page 135	The operation defined in the external service wizard
“Processing Direction” on page 135	The processing direction, `Inbound` or `Outbound`

Bidi format string

The bidi format string of the content data.

Table 42. 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 43. Data binding details

Required	No
Default	Use default data binding ’FlatFileBaseDataBinding’ for all operations
Usage	The value of this property can be: • Use default data binding ’FlatFileBaseDataBinding’ for all operations • Use a data binding configuration for all operations • Specify a data binding for each operation
Globalized	No
Bidi supported	No

Function selector

During inbound processing, the name of the function selector configuration to be used.

Table 44. Function selector details

Required	Yes
Default	`FilenameFunctionSelector`
Property type	String

Table 44. Function selector details (continued)

Usage	The function selector returns the appropriate operation to be called on the service. The adapter provides two function selectors, `FilenameFunctionSelector` and `EmbeddedNameFunctionSelector`. • `FilenameFunctionSelector` is a rule-based function selector that matches a regular expression on a file name to an object name. Use `FilenameFunctionSelector` for generic FlatFile business objects, where the object name cannot be determined from the event file. `FilenameFunctionSelector` is represented in properties as a two-column table with *N* rows. For any event file with a `.txt` extension, the corresponding object name is FlatFile, and the endpoint method name generated by the function selector is emitFlatFile. You must set this same name in the EISFunctionName property after you add the operation. 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. • Use `EmbeddedNameFunctionSelector` for content-specific business objects, where the object name is embedded in the event file. `EmbeddedNameFunctionSelector` returns the function name based on the desired content data, not the wrapper. For example, if the content-specific business object is CustomerWrapperBG, the function returned by the function selector is emitCustomer. You must configure `EmbeddedNameFunctionSelector` 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.
Globalized	Yes
Bidi supported	No

Log file output location

The full path name of the log file generated by the external service wizard.

Table 45. 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 46. Logging level details

Required	No

Possible values	`Severe`
	`Warning`
	`Audit`
	`Info`
	`Config`
	`Detail`

Default	`Severe`
Property type	List of values
Globalized	No

Table 46. Logging level details (continued)

Bidi supported	No

NameSpace

The namespace of the business object that is generated.

Table 47. NameSpace details

Required	Yes
Default	http://www.ibm.com/xmlns/prod/websphere/j2ca/flatfile
Property type	String
Globalized	Yes
Bidi supported	No

Operation name

The name you give to the operation defined for this module.

Table 48. 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 49. 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 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 118.

Table 50. Activation specification properties

Property name		
In the wizard	**In the administrative console**	**Description**
"Archive directory" on page 137	ArchiveDirectory	The directory where the adapter archives processed event files.
(Not available)	ArchivingProcessed	Supported for compatibility with earlier versions
"Auto create event table" on page 138	EP_Create Table	Determines whether the event persistence table is created automatically or manually.
(Not available)	DefaultObjectName	Supported for compatibility with earlier versions
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 138	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	Supported for compatibility with earlier versions
"Event directory" on page 139	EventDirectory	The directory where the event files are stored.
"Event recovery data source (JNDI) name" on page 140	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 140	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
"Failure file extension for archive" on page 141	FailedArchiveExtension	The file extension used to archive unsuccessfully processed business objects in the input event file.
"File content encoding" on page 141	FileContentEncoding	The encoding of the files that are read by the adapter.
"File extension for archive" on page 141	OriginalArchiveExtension	The file extension used to archive the original event file.

Table 50. Activation specification properties (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
"Include business object delimiter in the file content" on page 142	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
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 143	FilePassByReference	Specifies whether the adapter delivers the file content to the export.
"Password used to connect to event data source" on page 143	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
"Retrieve files in sorted order" on page 144	SortEventFiles	The sorting order of polled event files.
"Retrieve files with pattern" on page 144	EventFileMask	The file filter for the event files.
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 145	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 145	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 146	StopPollingOnError	Specifies whether the adapter stops polling for events when it encounters an error during polling
"Success file extension for archive" on page 146	SuccessArchiveExtension	The file extension used to archive successfully processed business objects.
"User name used to connect to event data source" on page 146	EP_UserName	The user name used by event persistence processing to obtain the JDBC database connection from the data source.

Archive directory

This property specifies the directory where the adapter archives processed event files.

Table 51. Archive directory details

Required	No
Default	None

Table 51. Archive directory details (continued)

Property type	String
Globalized	Yes
Bidi supported	Yes

Auto create event table

This property determines whether the event persistence table is created automatically or manually.

Table 52. 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

Database schema name

This property specifies the schema name of the database used by event persistence processing.

Table 53. Database schema name details

Required	No
Default	None
Property type	String
Globalized	Yes
Bidi supported	Yes

Delivery type (DeliveryType)

This property specifies the order in which events are delivered by the adapter to the export.

Table 54. Delivery type details

Required	No
Possible values	`ORDERED` `UNORDERED`
Default	`ORDERED`
Property type	String
Usage	The following values are supported: • `ORDERED`: The adapter delivers events to the export one at a time. • `UNORDERED`: The adapter delivers all events to the export at once.

Table 54. Delivery type details (continued)

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 55. Do not process events that have a timestamp in the future details

Required	Yes
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 be delivered. If set to `False`, the adapter delivers all events.
Globalized	No
Bidi supported	No

Ensure once-only event delivery (AssuredOnceDelivery)

This property specifies whether to provide ensure once-only event delivery for inbound events.

Table 56. Ensure once-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 57. Event directory details

Required	Yes
Default	None
Property type	String
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 58. 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 59. 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 60. Event types to process details

Required	No
Possible values	A comma-delimited (,) list of business object types
Default	`null`
Property type	String

Table 60. Event types to process details (continued)

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

Failure file extension for archive

This property specifies the file extension used to archive unsuccessfully processed business objects in the input event file.

Table 61. 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 62. File content encoding details

Required	No
Default	None
Property type	String
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.
Globalized	No
Bidi supported	No

File extension for archive

This property specifies the file extension used to archive the original event file.

Table 63. 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 64. 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 `com.ibm.j2ca.extension.utils.filesplit.SplitByDelimiter`. **Note:** 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 65. 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 66. Maximum events in polling period details

Required	Yes
Default	`10`
Property type	Integer

Table 66. Maximum events in polling period details (continued)

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, less 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 67. Number of times to retry the system connection details

Required	No
Possible values	Positive integers
Default	`0`
Property type	Integer
Usage	Only positive values are valid. When the adapter encounters an error related to the inbound connection, this property specifies the number of times the adapter tries to restart the connection. A value of `0` indicates an infinite number of retries.
Globalized	Yes
Bidi supported	No

Pass only file name and directory, not the content

Table 68. Pass only file name and directory, not the content details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	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. **Note:** 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.
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 69. Password used to connect to event data source details

Required	No

Table 69. Password used to connect to event data source details (continued)

Default	None
Property type	String
Globalized	Yes
Bidi supported	Yes

Retrieve files in sorted order

This property specifies the sorting order of polled event files.

Table 70. 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 71. 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 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 72. Retry interval details

Required	Yes
Default	`2000`
Unit of measure	Milliseconds
Property type	Integer

Table 72. Retry interval details (continued)

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 73. Specify criteria to split file content details

Required	No
Default	`0`
Property type	String
Usage	This property specifies either 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: • If the SplittingFunctionClassName property is set to `com.ibm.j2ca.extension.utils.filesplit.SplitByDelimiter`, the SplitCriteria property must contain the delimiter that separates the business objects in the event file. • If the SplittingFunctionClassName property is set to `com.ibm.j2ca.utils.filesplit.SplitBySize`, 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. When the SplitCriteria property value is set to `0`, file splitting is disabled. **Note:** 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.
Globalized	Yes
Bidi supported	Yes

Split function class name

This property determines how the event file is to be split.

Table 74. Split function class name details

Required	No
Possible values	`com.ibm.j2ca.extension.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. **Note:** If the EventContentType property is null, the SplittingFunctionClassName property is automatically set to `com.ibm.j2ca.utils.filesplit.SplitBySize`.
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 75. 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 76. 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 77. User name used to connect to event data source details

Required	No
Default	None
Property type	String
Globalized	Yes
Bidi supported	Yes

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 118.

Table 78. Resource adapter properties for the Adapter for Flat Files

Name		
In the wizard	**In the administrative console**	**Description**
Adapter ID	AdapterID	Identifies the adapter instance for CEI and PMI events with respect to logging and tracing.
(Not available)	Enable HA support	Do not change this property.
(Not available)	LogFileMaxSize	Supported for compatibility with earlier versions
(Not available)	LogFilename	Supported for compatibility with earlier versions
(Not available)	LogNumberOfFiles	Supported for compatibility with earlier versions
(Not available)	TraceFileMaxSize	Supported for compatibility with earlier versions
(Not available)	TraceFileName	Supported for compatibility with earlier versions
(Not available)	TraceNumberOfFiles	Supported for compatibility with earlier versions

Adapter ID to use for logging and tracing (AdapterID)

Use this property to identify a specific deployment, or instance, of the adapter.

Table 79. Adapter ID to use for logging and tracing details

Required	Yes
Default	CWYFF_FlatFile
Property type	String
Usage	This property is used to identify the adapter instance for PMI events. If you are deploying multiple instances of an adapter, set this property to a unique value for each adapter instance. For inbound processing this property is retrieved from the resource adapter properties. For outbound processing, it is retrieved form the managed connection factory properties.
Globalized	Yes
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, and 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 andWebSphere 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 and 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 80. Bidirectional data format attributes and values

Letter position	Purpose	Values	Description	Default setting
1	Order schema	`I` or `V`	Implicit (Logical) or Visual	`I`

2	Direction	`L`
		`R`
		`C`
		`D`
			Left-to-Right,
			Right-to-Left
			Contextual Left-to-Right
			Contextual Right-to-Left
				`L`

3	Symmetric Swapping	`Y` or `N`	Symmetric Swapping is on or off	`Y`

4	Shaping	`S`
		`N`
		`I`
		`M`
		`F`
		`B`
			Shaped text
			Unshaped text
			Initial shaping
			Middle shaping
			Final shaping
			Isolated shaping
				`N`

5	Numeric Shaping	`H`
		`C`
		`N`
			Hindi
			Contextual
			Nominal
				`N`

Bidirectional properties that identify data for transformation

To identify business data subject to transformation, set the BiDiContextEIS property. Do this by specifying values for each of the five bidirectional format attributes (listed in Table 80) for the property. The BiDiContextEIS property can be set for the managed connection factory and the activation specification.

To identify event persistence data subject to transformation, set the BiDiFormatEP property. Do this by specifying values for each of the five bidirectional format attributes (listed in Table 1) for the property. The BiDiFormatEP 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.

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/v6r1mx/topic/com.ibm.wbit.610.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

The WebSphere Integration Developer online samples/tutorials gallery includes samples and tutorials to help you use WebSphere Adapters. You can access the online samples/tutorials gallery as follows:
- 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 6.1.0, information center, which includes WebSphere Process Server, WebSphere Enterprise Service Bus, and WebSphere Integration Developer information: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp
- WebSphere Adapters, version 6.0.2, information center: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.wsadapters602.doc/welcome_top_wsa602.html
- WebSphere Adapters, version 6.0, information center: http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/topic/com.ibm.wsadapters.doc/welcome_wsa.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

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

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

Index

A

accessibility
  administrative console 25
  external service wizard 25
  IBM Accessibility Center 25
  keyboard 25
  shortcut keys 25
activation specification properties
  Archive directory 135
  Auto create event table 135
  Database schema name 135
  Delivery type 135
  Do not process events that have a timestamp in the future 135
  Ensure once only event delivery 135
  Event directory 135
  Event recovery data source (JNDI) name 135
  Event recovery table name 135
  Event types to process 135
  Failure file extension for archive 135
  File content encoding 135
  File extension for archive 135
  Include business object delimiter in the file content 135
  Interval between polling periods 135
  Number of times to retry the system connection 135
  Pass only file name and directory, not the content 135
  Password used to connect to event data source 135
  Poll quantity 135
  Retrieve files in sorted order 135
  Retrieve files with pattern 135
  Retry interval if connection fails 135
  setting in administrative console 91, 96
  Specify criteria to split file content 135
  Split function class name 135
  Stop the adapter when an error is encountered while polling 135
  Success file extension for archive 135
  User name used to connect to event data source 135
adapter application
  starting 97
  stopping 97
Adapter for Flat Files
  accessibility 25
  administering 87
  standards compliance 24
Adapter for Flat Files module
  exporting as EAR file 82
  installing EAR file on server 84
  starting 97
  stopping 97
adapter messages 150
adapter patterns wizard 43
adapter performance 98
adapter technotes 152
append 4
Append 4
artifacts, generating 61

B

backward compatibility
  project interchange files 34
  projects 34
business faults 105
business object information 113
business object, predefining 39, 41
business objects 4, 22
  attribute properties 116
  naming conventions 116
  structure 113

C

CEI (Common Event Infrastructure) 101
clustered environment
  deploying in 30
  description 30
  inbound processes 30
  outbound processes 31
Common Event Infrastructure (CEI) 101
compatibility matrix 2
compatibility with earlier versions 31
configuring
  logging 102
  Performance Monitoring Infrastructure (PMI) 98
  tracing 102
configuring the data binding, inbound 68
configuring the data binding, outbound 56
connection properties, inbound 63
connection properties, outbound 50
create 4
Create 5
custom properties
  activation specification 91, 96
  managed connection factory 89, 94
  resource adapter 87, 93

D

data transformation (inbound) 21
data transformation (outbound) 7
debugging
  org.xml.sax.SAXParseException exception 110
  self-help resources 111
  XAResourceNotAvailableException exception 109
delete 4
Delete 6
deployment
  environments 77
  options 27
  to production environment 80
  to test environment 77
deprecated features 31
developerWorks 151
developerWorks resources, WebSphere Adapters 151

E

EAR file
  exporting 82
  installing on server 84
education, WebSphere Adapters 151
embedded adapter
  activation specification properties, setting 91
  considerations for using 29
  description 27
  managed connection factory properties, setting 89
  resource adapter properties, setting 87
EmbeddedNameFunctionSelector 17
enableHASupport property 30
Event archival values 17
event store
  overview 15
  structure 16
exceptions
  org.xml.sax.SAXParseException 110
  XAResourceNotAvailableException 109
exists 4
Exists 6
exporting module as EAR file 82
external service
  generating inbound artifacts 72
  overview 23
external service connection properties
  Bidi format string 119, 132
  Data binding 119, 132
  Function selector 119, 132
  Log file output location 119, 132
  Logging level 119, 132
  NameSpace 119, 132
  Operation name 119, 132
  Processing Direction 119, 132
external service discovery, connection properties 50, 63
external service wizard
  accessibility 25
  starting 48

F

faults
  description 105
FFDC (first-failure data capture) 105
file splitting
  delimiter-based 9, 18
  size-based 9, 18
FilenameFunctionSelector 17
files
  SystemOut.log log file 104
  trace.log trace file 104
first-failure data capture (FFDC) 105
function selector 17

G

generating artifacts 61

H

hardware and software requirements 2
hardware requirements 2
high-availability environment
  deploying in 30

high-availability environment *(continued)*
  description 30
  inbound processes 30
  outbound processes 31

I

IBM WebSphere Adapter Toolkit 151
implementation, Java 78
inbound configuration properties 131
installing EAR file 84
interaction specification properties
  Archive directory for retrieve operation 126
  changing 75
  Create a new file if the file does not exist 126
  Default target file name 126
  Delete the file after retrieve operation 126
  Delimiter between business objects in the file 126
  File content encoding 126
  Generate a unique file 126
  Output directory 126
  Specify criteria to split file content 126
  Split function class name 126
  Staging directory 126
Internet Protocol Version 6.0 (IPv6) 25
IPv6 25

J

Java implementation 78

K

keyboard 25

L

list 4
List 6
Log Analyzer 102
log files
  changing file name 104
  disabling 102
  enabling 102
  level of detail 102
  location 104
logging
  configuring properties with administrative console 102

M

managed (J2C) connection factory properties
  setting in administrative console 89, 94
managed connection properties
  Default target file name 122
  Output directory 122
  Sequence file 122
  Staging directory 122
matrix, compatibility 2
messages, adapter 150
migration considerations 31
module, creating 39
monitoring performance 98

N

naming conventions for business objects 116

O

operations 4, 5, 6, 7
org.xml.sax.SAXParseException 110
outbound 4, 5, 6, 7
 processing 4
 supported operations 4
outbound configuration properties 118
outbound operations
 append 4
 create 4
 delete 4
 exists 4
 list 4
 overwrite 4
 retrieve 4
overwrite 4
Overwrite 6

P

package files for adapters 103
patterns 43
Performance Monitoring Infrastructure (PMI)
 configuring 98
 description 98
 viewing performance statistics 100
performance statistics 100
PMI (Performance Monitoring Infrastructure)
 configuring 98
 description 98
 viewing performance statistics 100
problem determination
 org.xml.sax.SAXParseException exception 110
 self-help resources 111
 XAResourceNotAvailableException exception 109
project interchange (PI) file
 updating without migrating 34
project, creating 48
properties
 activation specification 91, 96
 configuration properties
  inbound 131
  outbound 118
 inbound configuration 131
 managed (J2C) connection factory 89, 94
 outbound configuration 118
 resource adapter 87, 93

R

RAR (resource adapter archive) file
 description 81
 installing on server 81
Redbooks, WebSphere Adapters 151
related information 151
related products, information 151
requirements, hardware and software 2
resource adapter archive (RAR) file
 description 81
 installing on server 81
resource adapter properties
 Adapter ID 125, 147
 details 125, 147
 Enable HA support 125, 147
 setting in administrative console 87, 93
retrieve 4
Retrieve 7
Retry limit property 143
roadmap for configuring the module 37
runtime environment
 deploying EAR file to 80

S

samples 35
security 27
self-help resources 111
shortcut keys 25
software requirements 2
stand-alone adapter
 activation specification properties, setting 96
 considerations for using 29
 description 27
 managed connection factory properties, setting 94
 resource adapter properties, setting 93
standards compliance 24
starting adapter applications 97
stopping adapter applications 97
support
 overview 102
 self-help resources 111
 technical 152
supported operations 4, 5, 6, 7
SystemOut.log file 104

T

target component 77
technical overview 3
technical support 152
technotes 2, 111, 152
technotes, WebSphere Adapters 151
test environment
 adding module to 79
 deploying to 77, 79
 testing modules 80
trace files
 changing file name 104
 disabling 102
 enabling 102
 level of detail 102
 location 104
trace.log file 104
tracing
 configuring properties with administrative console 102
troubleshooting
 org.xml.sax.SAXParseException exception 110
 overview 102
 self-help resources 111
 XAResourceNotAvailableException exception 109
tutorials 35

U

unique file names, generating 12

W

WebSphere Adapter for Flat Files 122, 125, 147
- inbound processing 13
- introduction 1
- outbound processing 4
- planning adapter implementation 27
- security 27
- technical overview 3

WebSphere Adapters, version 6.0, information 151
WebSphere Adapters, version 6.0.2, information 151
WebSphere Application Server information 151
WebSphere Business Integration Adapters information 151
WebSphere Business Process Management, version 6.1.0, information 151
WebSphere Enterprise Service Bus
- deploying to 80
- information 151

WebSphere Extended Deployment 30
WebSphere Integration Developer
- information 151
- starting 39, 41, 48
- test environment 77

WebSphere Process Server
- deploying to 80
- information 151

wiring components 77

X

XAResourceNotAvailableException 109

Printed in USA