WebSphere Adapters

Version 6 Release 1

WebSphere Adapter for FTP User Guide
Version 6 Release 1

WebSphere® Adapters

IBM

Version 6 Release 1

WebSphere Adapter for FTP User Guide
Version 6 Release 1

Note
Before using this information and the product it supports, read the information in "Notices" on page 181.

16 January 2007

This edition applies to version 6, release 1, modification 0 of IBM WebSphere Adapter for FTP 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 FTP 1
New in this release 1
Hardware and software requirements 3
Technical overview of the Adapter for FTP 3
Outbound processing 3
Inbound processing 8
Business objects 16
External service wizard 17
Standards compliance 17
Accessibility 17
Internet Protocol Version 6 (IPv6) 18

Chapter 2. Planning for adapter implementation 19
Before you begin 19
Security 19
Configuring secure socket layers 19
Configuring the adapter for federal information processing standard 140 21
User authentication 22
Deployment options 23
WebSphere Adapters in clustered environments . . 25
Migrating to version 6.1.0 27
Migration considerations 27
Performing the migration 28
Updating but not migrating a version 6.0.2 project 29

Chapter 3. Samples and tutorials . . . 31

Chapter 4. Configuring the module for deployment 33
Roadmap for configuring the module 33
Creating the authentication alias 35
Creating the module 37
Defining business objects 38
Creating a simple service with the adapter pattern wizard 40
Creating the project 46
Configuring the module for outbound processing . 49
Setting deployment and runtime properties . . . 49
Selecting a data type and operation name . . . 51
Configuring the data binding 54
Configuring data handlers 58
Setting interaction specification properties and generating the service 63
Configuring the module for inbound processing . . 66
Setting deployment and runtime properties . . . 66
Selecting a data type and operation name . . . 70
Configuring the data binding 72
Configuring data handlers 76
Generating the service 79

Chapter 5. Changing interaction specification properties using the assembly editor 81

Chapter 6. Deploying the module . . . 83
Deployment environments 83
Deploying the module for testing 83
Generating and wiring a target component for testing inbound processing 83
Adding the module to the server 85
Testing the module for outbound processing using the test client 86
Deploying the module for production 86
Installing the RAR file (for modules using stand-alone adapters only) 86
Exporting the module as an EAR file 88
Installing the EAR file 90

Chapter 7. Administering the adapter module 93
Changing configuration properties for embedded adapters 93
Setting resource adapter properties for embedded adapters 93
Setting managed (J2C) connection factory properties for embedded adapters 95
Setting activation specification properties for embedded adapters 97
Changing configuration properties for stand-alone adapters 99
Setting resource adapter properties for stand-alone adapters 99
Setting managed (J2C) connection factory properties for stand-alone adapters 100
Setting activation specification properties for stand-alone adapters 102
Starting the application that uses the adapter . . . 103
Stopping the application that uses the adapter . . 103
Monitoring performance using Performance Monitoring Infrastructure 104
Configuring Performance Monitoring Infrastructure 104
Enabling tracing with the Common Event Infrastructure (CEI) 106
Viewing performance statistics 107
Troubleshooting and support 108
Configuring logging and tracing 108
First-failure data capture (FFDC) support . . . 111
Business faults 111
XAResourceNotAvailableException 115
org.xml.sax.SAXParseException 116
Self-help resources 116

Chapter 8. Reference information . . . 119

Business object information 119
Business object structure 119
Naming conventions 122
Business object attribute properties 123
Business object operation support 123
Custom business objects 123
Custom file splitting 124
Outbound configuration properties 125
Adapter type properties 126
Resource adapter properties 128
Managed (J2C) connection factory properties 131
Interaction specification properties 139
Inbound configuration properties 148
Adapter type properties 150
Resource adapter properties 151
Activation specification properties 154
Globalization 174
Globalization and bidirectional transformation 174
Properties enabled for bidirectional data transformation 176
Adapter messages 178
Related information 178

Notices 181
Programming interface information 183
Trademarks and service marks 183

Index 185

Chapter 1. Overview of WebSphere Adapter for FTP

With WebSphere Adapter for FTP, you can create integrated processes that use WebSphere Process Server and WebSphere Enterprise Service Bus to access files managed by an FTP server without having to know the details of FTP communications or protocols.

Once configured, the adapter acts like a service provider in a Service oriented architecture (SOA) implementation, providing operations to send and retrieve files. The adapter is part of a module that is deployed to WebSphere Process Server or WebSphere Enterprise Service Bus.

The adapter exposes a service interface that hides the mechanics of how the data or operations are obtained or run. Services outside of the module interact with the adapter instead of directly with the FTP server, so authentication details (such as user name and password) that you provide when you set up a module are shielded from services outside of the module.

What is the benefit? The module, which you create with the external service wizard in WebSphere Integration Developer, is a reusable unit designed to perform a specific inbound or outbound service. Each module uses a consistent interface and standard business objects, so applications consuming the service do not have to understand the lower-level details of the FTP server.

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

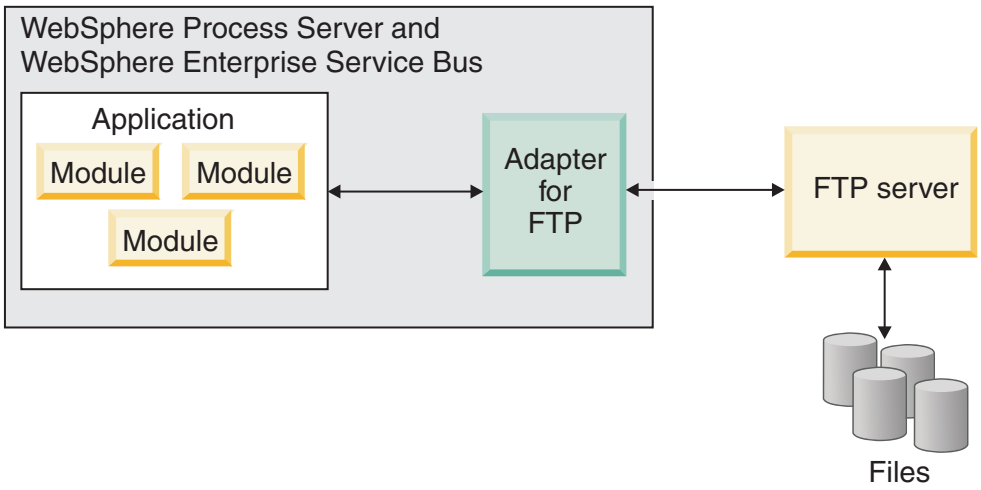


Figure 1. Adapter overview

New in this release

Read about new or enhanced product features and functions.

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/.

New in version 6.1.0:

- Changes to 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.

  The wizard now provides default values for many properties, makes it easier to enter certain information, indicates which properties are required, and lets you configure the module without worrying about advanced properties.
- Business graphs are now optional. 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.
- 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.
- Expanded operating system support. For more information on what operating systems are supported in version 6.1.0, see the hardware and software requirements for WebSphere Adapter for FTP on the IBM Web site at http://www.ibm.com/support/docview.wss?uid=swg27006249.
- Automated migration of WebSphere Adapter for FTP and associated external service wizard artifacts from version 6.0.2 to version 6.1.0.
- The adapter patterns wizard has been added to save time in creating simple services. For more information, see "Creating a simple service with the adapter pattern wizard" on page 40.
- Backward compatibility with 6.0.2 business objects.
- Sub folder search enabled for outbound Exists operations.
- Support for generating unique file names on Create, Append, and Overwrite operations.
- Support for creating a file in the FTP server directory on Append and Overwrite operations.
- Support for deleting a file from the FTP server directory on the Retrieve operation.
- Support for event file sequencing during outbound Create operations.
- Support for file parsing and data transformation for outbound Retrieve operations.
- Support for Secure Sockets Layer (SSL) and Federal Information Processing Standard (FIPS) 140.
- 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 node-level, or stand-alone, deployment of the adapter
- Simplified support for bidirectional script processing
- Support for parameter substitution and error handling in the FTP script file.
- Logging, tracing, and monitoring functionality moved out of the adapter.
- 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.

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 FTP

WebSphere Adapter for FTP provides the means for services running on WebSphere Process Server or WebSphere Enterprise Service Bus to communicate with one or more FTP servers.

The services are contained in a module, which consists of both a project in WebSphere Integration Developer and a unit of deployment to WebSphere Process Server. The module is packaged and deployed to WebSphere Process Server as an enterprise archive (EAR) file.

The module contains components, which are the actual services, imports and exports. Imports identify services outside of a module, making them callable from within the module. Exports allow components in a module to provide their services to external clients. Imports and exports require binding information, which specifies the means of transporting the data from the modules. The assembly editor in WebSphere Integration Developer sets up the imports and exports, lists the supported bindings, and simplifies their creation.

Outbound processing

The Adapter for FTP supports outbound request processing. When the adapter receives a request, which is sent in the form of a business object from the module, it processes the request and returns the result, when applicable, in a business object.

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

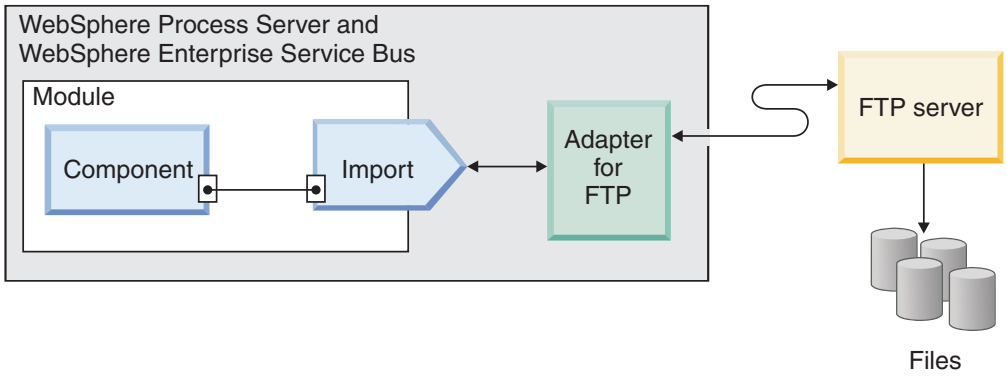


Figure 2. Outbound processing flow

Outbound data transformation

Data transformation during outbound communications refers to the process by which the adapter transforms business objects into an event record created in a native format, such as bytes or a string. The adapter uses an adapter-specific data binding and data handlers to accomplish this.

Data transformation allows external applications to send and receive data in a format that they can understand and process easily. The data bindings and data handlers that the adapter uses to create the event record from the corresponding attributes in a business object are configured using the external service wizard in WebSphere Integration Developer.

Data bindings

Data bindings are responsible for reading the fields in a business object and filling the corresponding fields in an event record. Each data binding is a map that defines how a business object should be formatted. The adapter for FTP uses the FTPFileBaseDataBinding data binding during outbound communication.

During outbound communications, the data binding takes the following fields from a business object and populates the equivalent fields in an event record with their values:

- DirectoryPath
- Filename
- DataConnectionMode
- FileTransferType
- SecondServerDirectory
- SecondServerUsername
- SecondServerPassword
- IncludeEndBODelimiter
- FileInLocalDirectory
- LocalDirectoryPath
- LocalArchivingEnabledForCreate
- LocalArchiveDirForCreate
- StagingDirectory
- GenerateUniqueFile

- SplittingFunctionClassName
- SplitCriteria
- DeleteOnRetrieve
- ArchiveDirectoryForRetrieve
- FileContentEncoding

For data that does not require transformation, the adapter conducts what is called pass-through processing because data passes through the system without being altered.

Data handlers

In addition to data bindings, data transformation requires the use of a data handler. Data handlers perform the conversions between a business object and a native format. In version 6.1.0 of WebSphere Adapter for FTP, the adapter provides the following data handlers:

- XMLDataHandler
- WTXInvokerDataHandler
- WTXMapSelectionDataHandler

Supported operations

An operation is the name of the action that the adapter can perform on remote file systems accessible through an FTP server during outbound processing. The name of the operation typically indicates the type of action that the adapter takes, such as *Create* or *Append*.

During outbound processing, WebSphere Adapter for FTP supports the following operations.

Table 1. Supported outbound operations

Operation	Result
Create	The file with the specified name is created in the specified directory of the FTP server. The content of the file can arrive as either part of the request or it can be retrieved from the local file system. If the file to be created does not exist, the file is created, and the file name is sent back to the calling component indicating a successful file creation. When the file content is received as part of the request, the adapter provides the option to archive the file on the adapter machine before creating it. The file can be created in a staging directory and then sent to the actual directory. If a staging directory is not specified, the file is directly created in actual directory. The adapter provides a feature to generate unique file names. See “Generating Unique File Names” on page 8. The adapter provides a feature to create a file sequence for the output files created. See “Generating a file sequence during Create operations” on page 7. If the file to be created already exists, a DuplicateRecord exception is sent, and no file is created. The existing file is not overwritten.

Table 1. Supported outbound operations (continued)

Operation	Result
Append	The file with the specified name in the specified directory of the FTP server is appended with the content sent across in the request. If the file to be appended exists, the content is appended, and the file name is sent back to the calling component indicating a successful response. The file to be appended is copied from the specified directory to the actual directory, if present, and the content is appended to that file in the staging directory. The file is then moved back to the actual directory. If the CreateIfFileNotExist property is set to `true`, the adapter creates a new file. The adapter provides a feature to generate unique file names. See "Generating Unique File Names" on page 8. If the file to be appended does not exist, a RecordNotFound exception is sent to the calling component.
Delete	The file in the specified directory is deleted on the FTP server and the adapter returns `null` to the calling component indicating a successful deletion. If the file to be deleted does not exist, a RecordNotFound exception is sent to the calling component.
Retrieve	The content of the file or files in the specified request is returned. The file content is split based on SplittingFunctionClassName and SplitCriteria properties. The file content is transformed into a business object based on the configured data handler. If the file or files, specified in the request exist, the content of the file is retrieved and sent as the response. The file content can either be sent back to the calling component or saved to the local file system. The adapter provides an option to delete the file from the FTP server directory after it is retrieved. This is done using the DeleteOnRetrieve property. The adapter supports an option to archive the file on the FTP server before it is deleted using the ArchiveDirectoryForDeleteOnRetrieve property. If the file to be retrieved does not exist, a RecordNotFound exception is sent to the calling component.
Overwrite	Overwrites the file in the directory with the content specified in the request. If the file to be overwritten exists, the content is overwritten, and the file name is sent back to the calling component indicating a successful response. The file to be overwritten is copied from the specified directory to the staging directory, if present, and the content is overwritten for that file in the staging directory. The file is then moved back to the specified directory. If a staging directory is not specified, the content is overwritten on the file in the specified directory. If the CreateIfFileNotExist property is set to `true`, the adapter creates a new file. The adapter provides a feature to generate unique file names. See "Generating Unique File Names" on page 8. If the file to be overwritten does not exist, a RecordNotFound exception is sent to the calling component.

Table 1. Supported outbound operations (continued)

Operation	Result
Exists	If the file name in the request exists in the specified directory or any of the sub folders, the adapter returns `true` and the full path of the file to the calling component. If same file name exists in more than one directory, the adapter returns `true` and the full path of the first file found to the calling component. If the file name does not exist, or the directory does not exist, the adapter returns `false` to the calling component.
List	Returns all file names and directories that are specified in the request to the calling component. If only the directory is specified, all the file names in the directory would be retrieved and sent as a response to the calling component. If the specified directory does not exist, a RecordNotFound exception is sent to the calling component.
ServerToServer FileTransfer	Transfers the specified file from one FTP server directory to another FTP server directory. After transferring the file successfully, `null` is returned to the calling component. If the request does not contain all necessary information about the two servers, the adapter sends an FTPFileServerToServerFileTransfer exception to the calling component.
ExecuteFTPScript	Runs the commands contained in a FTP script file in the adapter machine. The operation runs only the commands that are supported by the FTP server. If the operation fails, the adapter sends an FTPFileExecuteFTPScript exception to the calling component. The script file must not contain connection related commands such as `open` because the adapter uses an already established connection to run the commands. The location of the script file is specified in the DirectoryPath and Filename properties. If the commands in the script file need to be run in a particular directory on the FTP server, then the script file must contain the first command to change to that directory. A list of commands runs and their reply strings are returned to the calling component. The adapter also supports parameter substitution in the FTP script file (replacing parameters %1, %2 with actual values). The values are sent as part of the request.

Generating a file sequence during Create operations

Adapter for FTP supports the generation of a file sequence during an outbound Create operation. The FileSequenceLog property has been introduced to specify the full path of the file where the sequences will be stored. When the FileSequenceLog property value is specified, the adapter generates file sequence numbers to append to the file name of the file that it creates. The sequence number is appended to the file name in the following format: `$FILENAME.$SEQUENCE_NUMBER.$FILE_EXT`. For example, if HostName = `localhost` and Filename = `Customer.txt`, the output files will be `Customer.1.txt, Customer.2.txt, Customer.3.txt`, and so on. The format is the same for all platforms, including z/OS® and i5/OS®. The sequence number continues incrementing after multiple adapter restarts.

When the adapter is operating in a standalone mode, the FileSequenceLog property value must be a file on the local file system. When adapter is operating in a clustered environment, the FileSequenceLog property value must be a file on a mapped drive that is accessible by all of the clusters. You must ensure that the adapter has write permissions for the sequence log file. If not, an `IOException` is sent.

The sequence number must continue incrementing after multiple adapter restarts.

The directory path and file name can be specified in two ways:

1. Setting the HostName and Filename properties, and not specifying **DirectoryPath** and **Filename** fields of the business object request.
2. Setting the **DirectoryPath** and **Filename** fields of the business object request.

Setting the HostName and Filename properties saves time because users do not need to keep setting the values on each business object request. However, the values specified on the business object request take precedence over the values set in the managed connection factory properties.

When the file sequence needs to be reset, you can access the file sequence log file and delete the entry for the file name. A new sequence begins at 1. When the FileSequenceLog property and GenerateUniqueFilename property are both enabled, the GenerateUniqueFilename property value takes precedence, and the log sequence is not generated.

Generating Unique File Names

File name generation can be accomplished in two different ways:

1. The adapter can maintain and use a sequence file to persist sequence numbers, which will be appended to the default file name.
2. Some FTP servers have built-in support for generating unique file names. For this, the server must support the STOU command specified in RFC 1123.

To use the first mechanism, the sequence file, you must specify the location of the sequence file and the target file name. The generated file name will be the target file name with the sequence number appended to it.

Note: The exact names from the wizard should be used here.
These properties exist in three places: the Managed Connection Factory, the Interaction Specification, and the business object. The properties in the business object take precedence over the properties in the interaction specification, which in turn takes precedence over the Managed Connection Factory. To keep things simple, use the properties on the Managed Connection Factory unless you want to handle a particular object differently.

If the default file name has an extension, the sequence number will be appended before the extension. For example, if the default file name is Customer.txt on the Managed Connection Factory, then the output file names that are created are Customer.1.txt, Customer.2.txt, and so forth. The sequence number is maintained independently for each business object type.

To use the second mechanism, the FTP server support, set the GenerateUniqueFile property on the Interaction Specification or in the business object. If you set the GenerateUniqueFilename property to `true`, then the FTP server's mechanism for file name generation will be used, instead of the adapter's mechanism.

Note: The adapter does not support both the GenerateUniqueFile and StagingDirectory options at the same time.

Inbound processing

The Adapter for FTP supports inbound processing of events. The adapter polls a file system associated with an FTP server for events at specified intervals. Each

time a file is created or updated, the adapter tracks this as an event. When the adapter detects an event, it requests a copy of the file, converts the file data into a business object, and sends it to the consuming service.

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

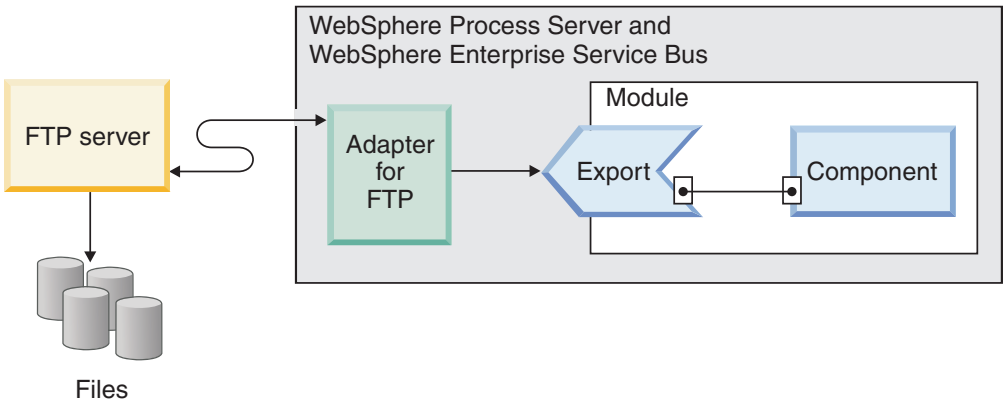


Figure 3. Inbound processing flow

The adapter polls files from the event directory of the FTP server at regular intervals based on the FTPPollFrequency property. When a file arrives in the event directory, the adapter reads the entire file and downloads the file to a local event directory on the adapter server. After the file is downloaded, the adapter either archives the file in the FTP server in an archive directory given by the FTPArchiveDirectory property or deletes it based on your configuration. The event directory, archive directory, poll frequency, and poll quantity (the number of files to poll in a single poll cycle) are all configurable properties.

After the business objects are successfully posted to the export, the events in the local staging directory are either archived in an archive directory on the local file system or deleted, based on your configuration. The adapter must archive or delete the events or they will be polled again.

Inbound event processing consists of the following steps:

1. FTP server generates events in the form of files.
2. The Adapter for FTP polls the event directory.
3. The files are downloaded to the adapter.
4. The files are split based on the SplittingFunctionClassName and SplitCriteria properties. The event file is split into several chunks and each chunk is posted to the export separately. This reduces memory loading during event processing.
   - If splitting is done based on a delimiter, the class that performs this function and the split criteria are provided.
   - If splitting is done based on file size, the class name that performs this function is provided.
   - If splitting is done based on other criteria, you must provide your own file splitting class.
5. The adapter sends the data, including the location of the polled document and the host name of the machine that the file was retrieved from, to the export through a function selector, where the configured data binding is invoked, to convert the text record into a business object.

Supported inbound operation

The adapter supports the emitFTPFile operation, which is taken as the default operation during inbound configuration.

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.

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 2. Examples of matching rules for FilenameFunctionSelector

FileName	ObjectName	Rule
Customer0001.txt	Customer	CUST.*TXT
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 FTPFile 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 emitFTPFile.

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.

Inbound data transformation

Data transformation during inbound communications refers to the process by which the adapter transforms an event record created in a native format, such as bytes or a string, into a business object. The adapter uses an adapter-specific data binding and data handlers to accomplish this.

The data bindings and data handlers that the adapter uses to read the contents of the event record and fill the corresponding attributes in a business object are configured using the external service wizard in WebSphere Integration Developer.

Data bindings

To take fields from an event record, created in a native format, and populate a business object, the adapter needs a data binding. Data bindings are responsible for reading the event record fields and populating the corresponding fields in a business object. The adapter for FTP uses the FTPFileBaseDataBinding data binding during inbound communication.

During inbound communications, the data binding takes the following fields from an event record and populates the following business object attributes with their values:

- Filename
- ChunkInfo
- DirectoryPath

- FileContentEncoding
- FtpServerHostName
- FtpServerEventDirectory

For data that does not require transformation, the adapter conducts what is called pass-through processing because data passes through the system without being altered.

Data handlers

In addition to data bindings, data transformation requires the use of a data handler. A data handler converts data from a native format into a business object. In version 6.1.0 of WebSphere Adapter for FTP, the adapter provides the following data handlers:

- XMLDataHandler
- WTXInvokerDataHandler
- WTXMapSelectionDataHandler

Passing files by reference

The adapter also supports a PassThrough feature, where only the event file name is sent to the export. The event file is appended with a time stamp and is available in the local archive directory. This feature is used when data transformation is not necessary.

Splitting files

The inbound event processing mode supports an optional file splitting feature, where the event file is split into several business objects, also known as chunks, and each business object is posted to the export separately. This reduces memory loading during event processing. File splitting is performed based on either a delimiter or on a file size specified in the SplitCriteria property.

The adapter provides SplitBySize and SplitByDelimiter classes for file splitting. Optionally, you can provide a custom file splitter class and use it by inputting the class name into the SplittingFunctionClassName property.

Splitting files by size

The size value is set in the SplittingFunctionClassName property.

Chunks refer to the resulting files after file splitting is performed. When chunking is enabled, each chunk of the file is posted to the export separately. The number of business objects that are specified in the PollQuantity property is posted to the export. For example, If the value for PollQuantity is 3, then:

The number of business objects polled is 3.

The number of business objects received by the export is 3.

The adapter does not reassemble chunked data. It provides the information about the chunked data for an external application to merge the chunks. The chunking information is set in the chunkInfo property, which is contained in the business object. This information includes the chunk size, in bytes, and the event ID. An example event ID is:

```
AbsolutePathOfTheEventFileNameInLocalEventDirectory_/_yyyy_MM_dd_HH_mm_ss_SSS.
currentBONumber_/_totalBOs
```

Splitting files by delimiter

Delimiters are specified values used for splitting event files. The delimiter is specified in the SplitCriteria property.

The following rules apply to the use of delimiters:

- The specified delimiter must not be the same as any of the data contained within the business object. If it is the same, file splitting can produce incorrect results.
- The delimiter must contain the exact value of new line representation in the event file. If the event file is created on a MAC machine, the new line character is `\r`. On UNIX machines it is `\n`, and for windows machines it is `\r\n`.
- If there is more than one delimiter, each delimiter must be separated by a semicolon `(;)`. If the semicolon is part of the delimiter, the semicolon must be escaped as `\;`. For example, if the delimiter is `##\;##` then it is processed as `##;##`, which means that the semicolon is part of the delimiter.
- 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. The adapter will consider `##$$` the delimiter and skip `"this is the content that will be skipped by the adapter."` For example, if the event file contains a business object in the following format and the delimiter is `##;;$$`, then:

  ```
  Name=Smith
  Company=IBM
  ##this is the content that will be skipped by the adapter$$
  ```

- The delimiter takes any value and there are no restrictions. The following are valid delimiter examples:
  - `####;\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 1:

```
John Doe,123,Washington Ave,222-123-4567
Jane Smith,234,Washington Ave,222-123-4568
```

The separator is the end of line character. In this example you would specify `\r\n` for Windows, `\r` for MAC, and `\n` for Unix.

Example 2:

```
John Doe
123 Washington Ave
222-123-4567
```

```
####
Jane Smith
234 Washington Ave
222-123-4568
```

The separator is `####`.

Event recovery

The adapter supports event recovery for inbound processing in case of abrupt termination. During event processing, the adapter persists the event state in an event persistence table located on the data source. You must set up this data source before you can create the event persistence table.

To use the recovery feature provided by WebSphere Process Server, you must set the AssuredOnceDelivery property in the activation specification to `true`. If it is set to `false`, failed events cannot be recovered. Duplicate events can be delivered if AssuredOnceDelivery is set to `false`. To improve performance, you can set the event recovery, duplicate events, and AssuredOnceDelivery properties to `false`.

Event persistence table

The event persistence table is a persistent cache where events are saved until the adapter can process them. The adapter uses event persistence tables to keep track of inbound requests as they make their way through the system. Each time a file is created, updated, or deleted, the adapter tracks the activity as an event and updates the status of the event in the event persistence table. The status of each event is continually updated by the adapter for recovery purposes until the events are delivered to a configured export.

If the adapter detects that there is no event persistence table, it automatically creates one when the module is deployed to the runtime environment. Each event persistence table created by the adapter is associated with a specific inbound module. The adapter does not support multiple adapter modules pointing to the same event persistence table.

When the adapter polls the FTP server, it creates an entry in the event persistence table 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`. When the adapter copies the event from the FTP server to the in progress folder on the local system, it marks the entry as `IN PROGRESS`. When the adapter sends the event to the function selector for data transformation, it deletes the entry from the event table.

Note: When guaranteed event delivery is not required, the adapter can poll for events without the existence of an event persistence table.

The following table describes each event persistence table value.

Table 3. Event persistence table structure

Column Name	Type	Description
EVNTID	Varchar(255)	A unique event ID for tracking purposes. The adapter uses this ID to track events during inbound processing.

Table 3. Event persistence table structure (continued)

Column Name	Type	Description
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: **NEWEVENT (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.
EVNTDATA	Varchar(255)	Used by the adapter to mark the failed events as `ARCHIVED` to ensure that they are not processed again during adapter startup or recovery.

Event archive

Archived events are stored in the archive directory with a file extension that is specified in the FTPRenameExt property. Event archiving is an optional feature, which provides you with a record of all of the events that have been processed. You can use this information to review whether or not the events were processed successfully.

Event archiving is used differently in different configurations:

- When both the FTPArchiveDirectory and FTPRenameExt values are provided and FTPRenameExt is set to `processed`, the archived file is located in the specified archive directory in the following syntax: *`filename_timestamp`*`.processed`
- When only the FTPArchiveDirectory value is provided, the archived file is located in the specified archive directory in the following syntax: *`filename_timestamp`*
- When neither the FTPArchiveDirectory nor the FTPRenameExt values are provided, the event file is deleted from the event directory of the FTP server after the file is successfully downloaded to the local event directory.
- When only the FTPRenameExt value is provided and is set to `processed`, the archived file is located in the event directory of the FTP server in the following syntax: *`filename_timestamp`*`.processed`

Archiving on MVS™ platforms

Multiple Virtual Storage (MVS) operating systems do not support special characters such as an underscore in dataset or recordset names. On Windows and UNIX platforms, use a time stamp in the original file name while archiving the file. This prevents duplicate file names in an archive folder, thereby preventing the overwriting of an existing file. Use the following format for MVS systems:

```
Event File: Test Archived
```

```
file: Test.TSyyyyMM.TSDDHHMM.TSSsSss
```

Where:

```
yyyy -- year

MM -- month

DD -- date

HH -- hour

MM -- minutes

Ss -- seconds

Sss -- milliseconds
```

The dataset or recordset separator is **.** (decimal) on MVS platforms. The maximum number of **.** (decimals) allowed in a dataset or recordset is six. The dataset or recordset name must not exceed eight characters per **.** (decimal), and the total number of characters must not exceed 44. An example of a file name in this format is:

```
FTPRenameExt: ARCHIVE

Archived File: TEST.TS200304.TS290535.TS42234.ARCHIVE
```

Business objects

A business object is a structure that consists of data, the action to be performed on the data, and additional instructions, if any, for processing the data. The data can represent either a business entity, such as an invoice or an employee record, or unstructured text.

How the adapter uses business objects

The adapter uses business objects to send data to or obtain data from the FTP server. The adapter's main job during inbound operations is to take information from an event record created in a native format, convert it to a business object, and forward it to a service. For outbound operations, this process happens in reverse. The adapter receives a business object from a service, creates an event record from the details it finds in the business object, and then sends the event record to the FTP server.

How data is represented in business objects

Business objects are created using the business object editor in WebSphere Integration Developer, which provides a graphical view of your business objects. As shown in the following illustration, a business object consists of a set of fields and their values. This is a customer business object. As you can see, it records name, address, and phone number information for a customer record. This example uses string values, but many other values are supported by business the object editor.

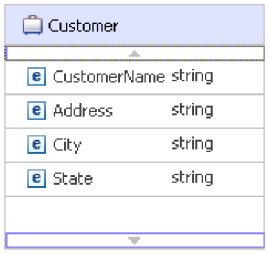


Figure 4. How data is represented in business objects

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 have defined XSD files using the business object editor prior to starting the external service wizard, the adapter will creates business objects from these schemas. For instructions on how to use the business object editor to create business objects, refer to the following link: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/in. After you create your business objects, you can use the business object editor to define the hierarchy of the business objects.

Business graphs

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.

External service wizard

The external service wizard in WebSphere Adapter for FTP is used to create services and to generate business objects from the selected objects. The wizard also generates the service artifacts that enable the adapter to run as a Service Component Architecture (SCA) component.

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 the IBM WebSphere Adapter for FTP, you must plan for inbound and outbound processing and consider security and performance requirements.

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.

To configure and use WebSphere Adapter for FTP, 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.
- The File Transfer Protocol (FTP), the protocol for exchanging files over the Internet.
- The FTP server being used to access the files on a specific file system in your solution.

Security

Secure socket layers (SSL) can be configured to protect the integrity of information being passed between the FTP server and the adapter. For secure communication, a secure FTP server that supports the SSL protocol and contains a private key and certificate must be installed and configured. For users who require it, the adapter can also be configured to run in support of the Federal Information Processing Standard (FIPS) 140.

Configuring secure socket layers

Data that travels across a network can be intercepted by third parties. When this data includes private information such as passwords or credit card numbers, steps should be taken to make this data unintelligible to unauthorized users. By configuring secure socket layers (SSL), you protect the integrity of information being passed between the FTP server and the adapter.

Before you begin

To enable SSL, the following prerequisites must be satisfied:

- The FTP server supports secure communication using SSL.
- The FTP server has its own private key and certificate.
- An FTP client is installed.

- The adapter uses a passive FTP mode of data transfer with a secure FTP server. If there is a firewall between the client and the server, the firewall settings might need to be configured to enable this mode.

About this task

Files passing through the FTP server are vulnerable to third party interference when SSL is not configured for use with the adapter. Using SSL prohibits data from being modified intentionally or unintentionally during transport and protects it from being intercepted. SSL is effective because it uses several cryptographic processes: public key cryptography for authentication with the FTP server and secret key cryptography and digital signatures for privacy and data integrity. SSL allows the adapter to authenticate the identity of the FTP server.

Procedure

1. Set the FTP client trust store. A trust store helps a FTP client decide what it can trust. When using SSL, WebSphere Process Server sends its certificate to the FTP client for verification. The FTP client verifies the certificate to ascertain that it is communicating with the intended FTP server. To enable this verification process, the FTP server's certificate should be present in the client's trust store.
   a. In WebSphere Integration Developer, right-click the server instance and click **Run administrative console**.
   b. Expand **Security**.
   c. Select **SSL certificate and key management**.
   d. Under **Related items**, select **Key stores and certificates**.
   e. Select **NodeDefaultTrustStore**. See the Figure 5 figure below.

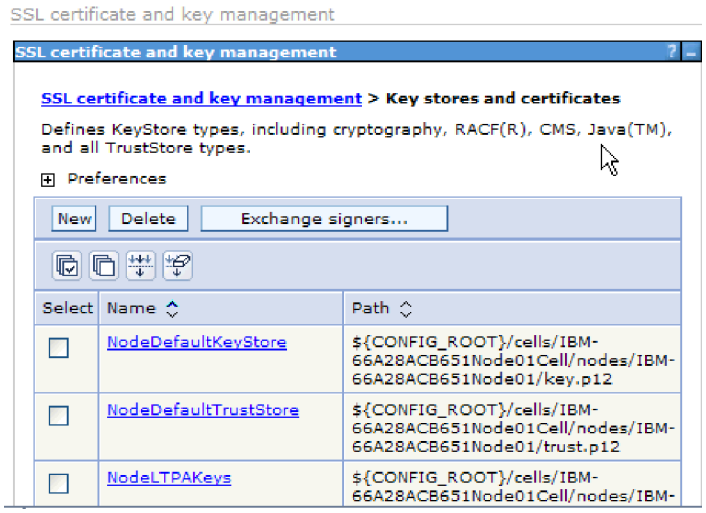


Figure 5. Selecting NodeDefaultTrustStore

   f. Under Additional properties, select **Signer certificates**.
   g. Click **Add**.
   h. In the **Alias** field, type the certificate name. See the Figure 6 on page 21 figure below.

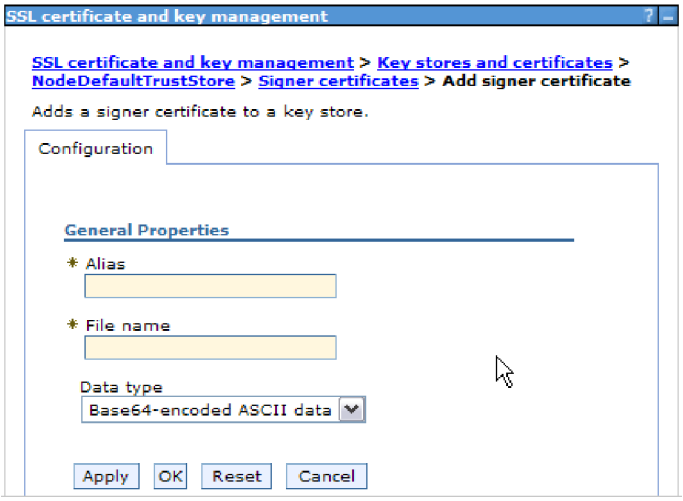


Figure 6. Adding signer certificate properties for the FTP server certificate

i. In the **File name** field, type the full path of the FTP server certificate.
j. Click **OK**

2. Configure SSL properties for the adapter.
   a. In the external service wizard, set **enableSSL** to `True`. By default, **enableSSL** is set to `False`.

Configuring the adapter for federal information processing standard 140

The federal information processing standard 140 (FIPS) is an United States government standard for cryptographic features like encryption, decryption, hashing (message digests), secure socket layers, transport layer security, Internet protocol security, secure shell, signatures, key exchange, and key or certificate generation used in software products and modules. If you are a user working with the United States government who must conform to the FIPS standard, you can configure the adapter to run in FIPS mode.

About this task

Configuring the adapter to run in FIPS mode restricts the adapter to working with modules whose cryptographic features comply with FIPS approved methods and providers. From an adapter perspective, running in FIPS mode restricts the adapter to using the transport layer security (TLS) secure socket protocol.

Note: For the adapter to run in FIPS mode, the FTP server must support SSL v3.1, which is the same as TLS v1.0, and it must be enabled through the wizard of the FTP server. If not properly supported by SSL v3.1, the SSL handshake with the adapter may fail.

To run the adapter in FIPS mode, you must instruct the adapter to use the IBM Java Secure Socket Extension (IBMJSSE2) provider package. The IBMJSSE2 provider is the preregistered Java secure socket extension provider in the Java security file in IBM SDK, version 5.0. IBMJSSE2 uses FIPS-approved packages.

Complete the following steps to run the adapter in FIPS mode:

Procedure

1. In the IBMJSSE2 provider, set the com.ibm.jsse2JSSEFIPS property to `True`.
2. Set the following security properties so the IBMJSSE2 provider will handle all JSSE requests.
   a. Set the ssl.SocketFactory.provider property to `com.ibm.jsse2SSLSocketFactoryImpl`.
   b. Set the ssl.ServerSocketFactory.provider property to `com.ibm.jsse2SSLServerSocketFactoryImpl`.
3. In the security properties file, add the IBMJCEFIPS provider `com.ibm.crypto.fips.provider.IBMJCEFIPS` to the provider list above the IBMJCE provider. Follow the *security.provider.n=providername* format where *n* denotes the order of the provider. The provider with a value of 1 is considered before the provider with a value of 2. Do not remove the IBMJCE provider.
4. Set system properties in the WebSphere Process Server administrative console Java virtual machine (JVM) properties. Follow the -D*propertyname=propertyvalue* format.
5. Set security properties in the Java.security file (located in the *WebSphere Process Server Java virtual machine*/lib/security directory).

For more details on configuring security details, see the security documentation for WebSphere Process Server or WebSphere Enterprise Service Bus.

User authentication

The adapter supports several methods for supplying the user name and password that are needed to connect to the FTP server. Understand the features and limitations of each method to pick a method that provides the appropriate level of security and convenience for your application.

To integrate an adapter into your application, you must provide the user name and password for the adapter to use at run time on WebSphere Process Server or WebSphere Enterprise Service Bus to connect to the FTP server to process outbound requests and inbound events.

At run time, the adapter needs to provide the user name and password to connect to the FTP server. To connect without user intervention, the adapter must access a saved copy of the user information. In a server environment, there are several methods for saving user information. The external service wizard lets you configure the adapter to get the user information using any of the following methods:

- Adapter properties
- Data source
- J2C authentication alias

Saving the user name and password in adapter properties is a direct way to provide this information at run time. You provide this user name and password when you use the external service wizard to configure your module. Although directly specifying the user name and password seems the most straightforward method, it has important limitations. Adapter properties are not encrypted; the password is stored as clear text in fields that are accessible to others on the server. Also, when the password changes, you must update the password in all instances of the adapter that access that FTP server. This includes the adapters embedded in application EAR files as well as adapters that are separately installed on the server.

Using a data source lets you use a connection already established for another application. For example, if multiple applications access the same database with the same user name and password, the applications can be deployed using the same data source. The user name and password can be known only to the first person who deploys an application to that data source or who defines a data source separately.

Using a J2C authentication alias created with the Java Authentication and Authorization Service (JAAS) is a robust, secure way to deploy applications. An administrator creates the authentication alias that is used by one or more applications that need to access a system. The user name and password can be known only to that administrator, who can change the password in a single place when a change is required.

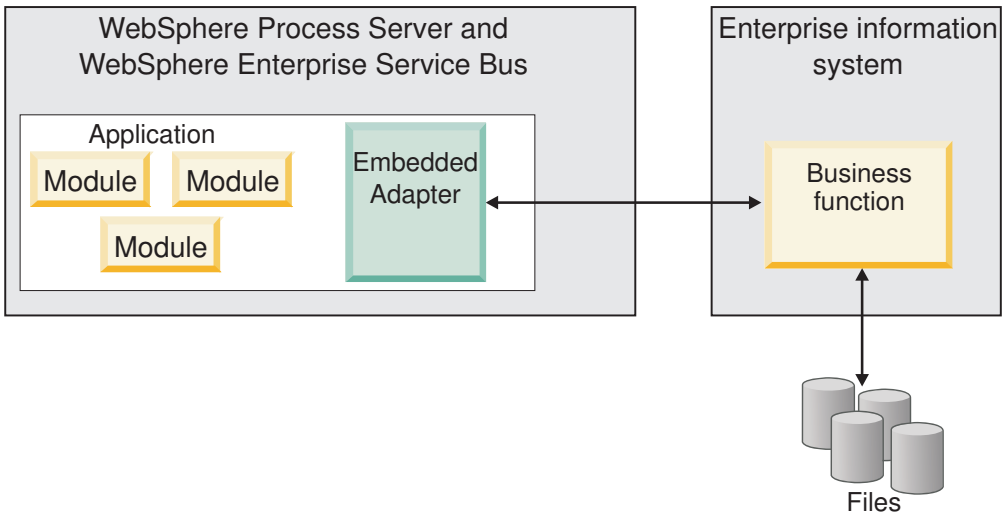
Deployment options

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.

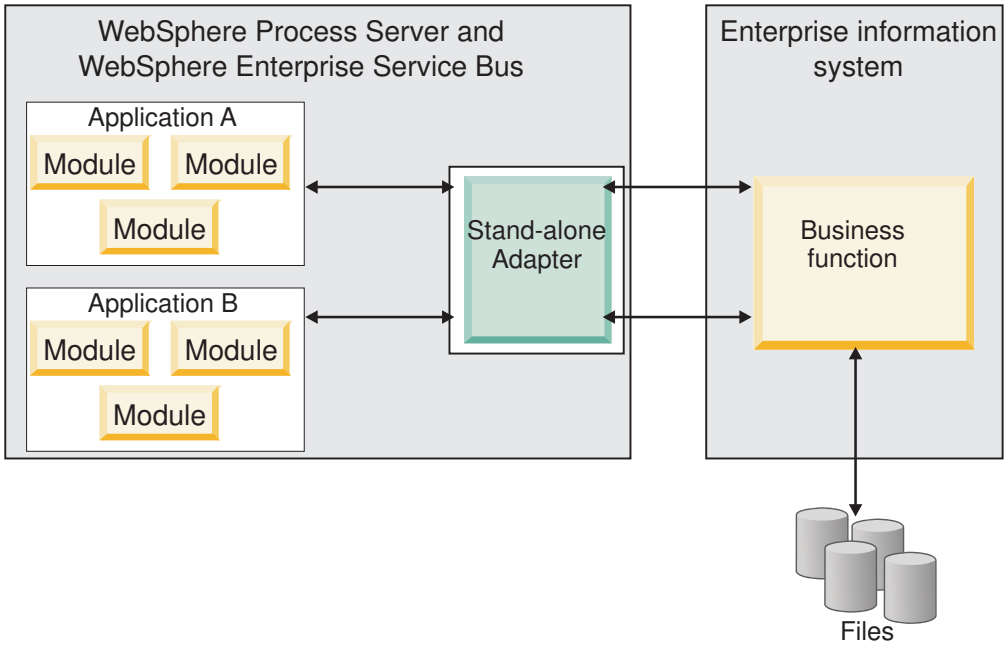
The deployment options are described below:

- **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.



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

Compatibility with earlier versions

WebSphere Adapter for FTP 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 FTP is fully compatible with version 6.02, any of your applications that utilized version 6.0.2 of WebSphere Adapter for FTP 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 FTP, you should always perform the migration in a development environment first and test your applications before deploying the application to a production environment.

The migration wizard is fully integrated with the development environment.

Deprecated features

A deprecated feature is one that is supported but no longer recommended and that might become obsolete. Features from earlier versions of WebSphere Adapter for FTP that have been deprecated in version 6.1.0 include:

- The EventContentType and DefaultObjectName Activation specification properties.
- The FTPURL Managed Connection Factory property.
- The FTPFileDataBinding data binding.
- The annotation tags contained in the XSD files.

Performing the migration

You can migrate a project or EAR file using the version 6.1.0, use the adapter migration wizard. When the tool is finished, the migration is complete and you can work in the project or deploy the module.

Before you begin

Review the information in *Migration considerations*.

About this task

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

Note: After migration is complete, the 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.

Note: You can also migrate in one of the following ways:

- Right-click the project in the J2EE perspective and select **Migrate → Migrate project**.
- From the Problems view, right-click a migration-specific message and select **Quick Fix** to correct the problem.

Procedure

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

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

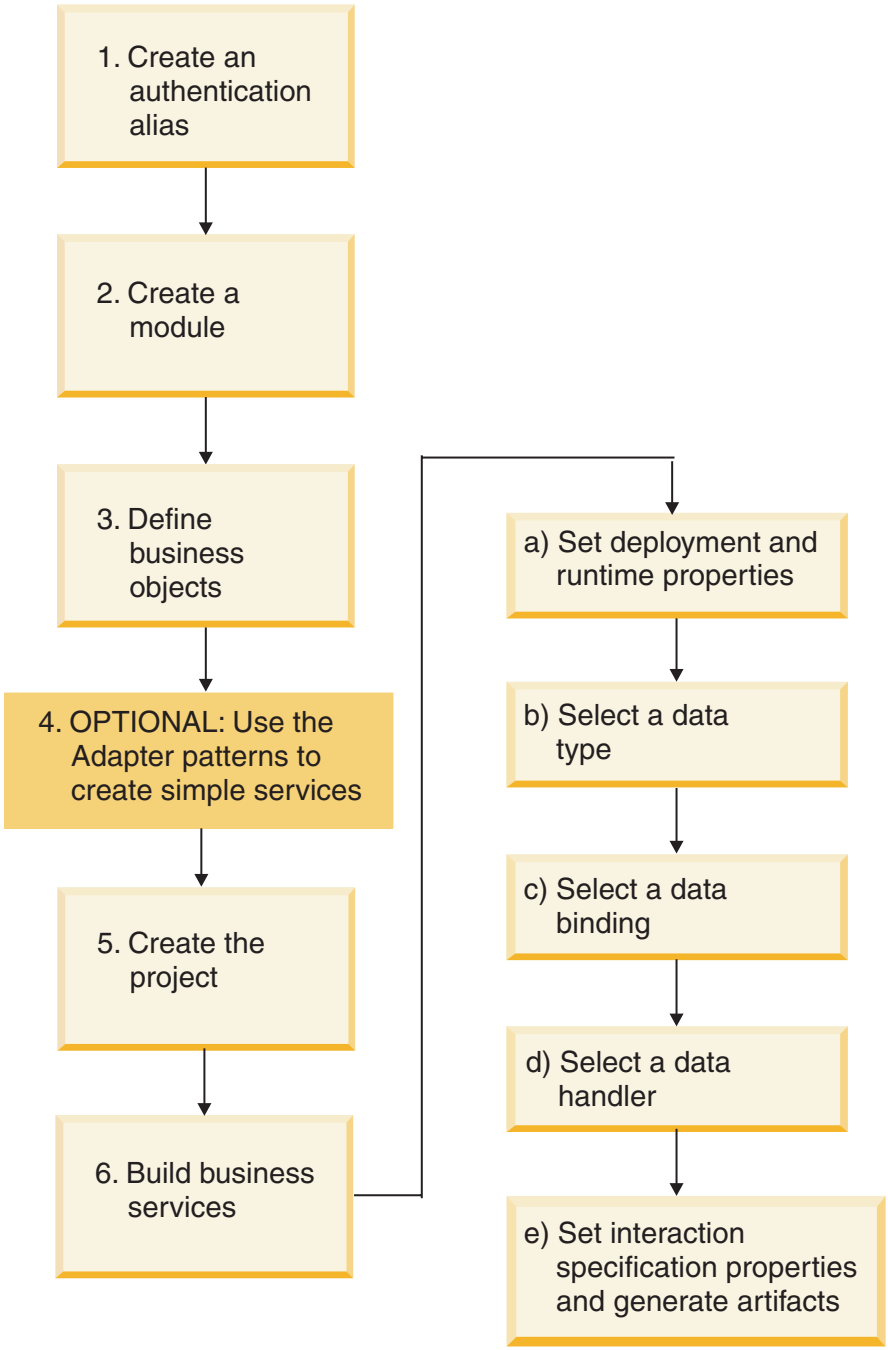


Figure 7. Roadmap for configuring the module

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 an authentication alias to access the FTP server. Perform this step using the administrative console on the server.

2. Create a module in WebSphere Integration Developer. You create business objects in the module.
3. Define the business objects that will be used by the project.
4. Use the Adapter patterns wizard to create simple services. For more information, see “Creating a simple service with the adapter pattern wizard” on page 40.
5. Create a project, which is used to organize the files associated with the adapter using the external service wizard in WebSphere Integration Developer.
6. 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 authentication alias

An authentication alias is a feature that encrypts the password used by the adapter to access the FTP server. After an authentication alias has been created, you can use it when you configure the adapter (instead of directly typing the user ID and password). Adapter properties are not encrypted, and if you directly type password, it is stored as clear text that can be viewed by others. Using the authentication alias is the default choice in the external service wizard.

Before you begin

To create an authentication alias, you must have access to the administrative console of WebSphere Process Server or WebSphere Enterprise Service Bus. The following procedure shows you how to gain access to the administrative console through WebSphere Integration Developer. If you are using the administrative console directly (without going through WebSphere Integration Developer, log in to the administrative console and skip to step 2.

About this task

To create an authentication alias, use the following procedure.

Procedure

1. Start the administrative console.

   To start the administrative console perform the following steps:

   a. Click **Start** → **Programs** → **IBM Software Development Platform** → **WebSphere Integration Developer 6.1** → **WebSphere Integration Developer 6.1**.

   b. If you are prompted to specify a workspace, accept the default value. (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**.

   d. Click the **Servers** tab.

   e. If the server does not show a status of **Started**, right-click the name of the server (for example, **WebSphere Process Server**) and click **Start**.

   f. Right-click the name of the server and click **Run administrative console**.

   g. Log onto the administrative console. If your administrative console requires a user ID and password, type the ID and password and click **Log in**. If the user ID and password are not required, click **Log in**.

2. In the administrative console, click **Security** → **Secure administration, applications, and infrastructure**.

3. Under **Authentication**, click **Java Authentication and Authorization Service** → **J2C authentication data**.

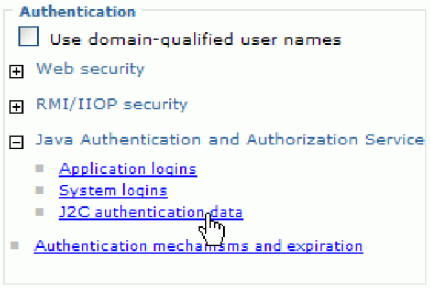


Figure 8. The Authentication section of the Secure administration, applications, and infrastructure window

4. Create an authentication alias.

   a. In the list of J2C authentication aliases that is displayed, click **New**.

   b. Click the **Configuration** tab, then type the name of the authentication alias in the **Alias** field.

   c. Type the user ID and password that are required to establish a connection to the FTP server.

   d. Optional: Type a description of the alias.

   e. Click **OK**.

      The newly created alias is displayed.

      Note the full name of the alias. This full name is the one you use in subsequent configuration windows.

   f. Click **Save**, and then click **Save** again.

5. Click **New**.

Results

You have created an authentication alias, which you will use when you configure the adapter properties.

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.

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 anywhere within the Business Integration section of the WebSphere Integration Developer window.

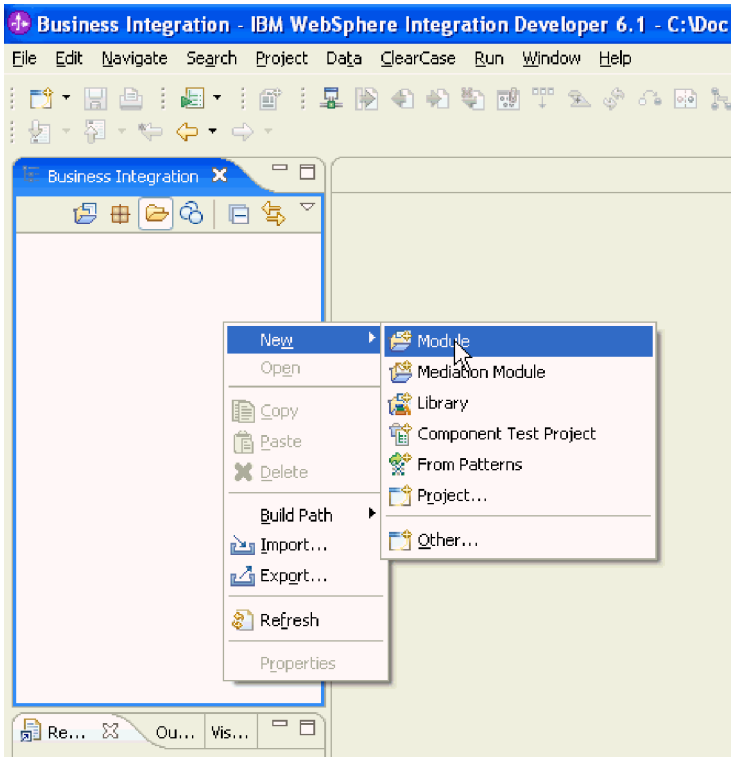


Figure 9. Business Integration section of the window

3. Type a new **Module Name** in the New Module window.

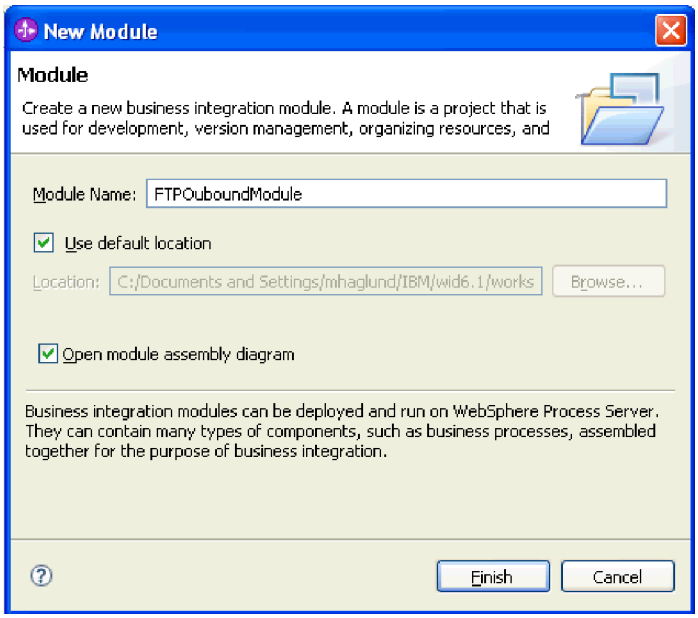


Figure 10. New Module window

4. Click **Finish**.

Results

A new module is listed in the Business Integration window.

What to do next

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

Defining 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.

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**.

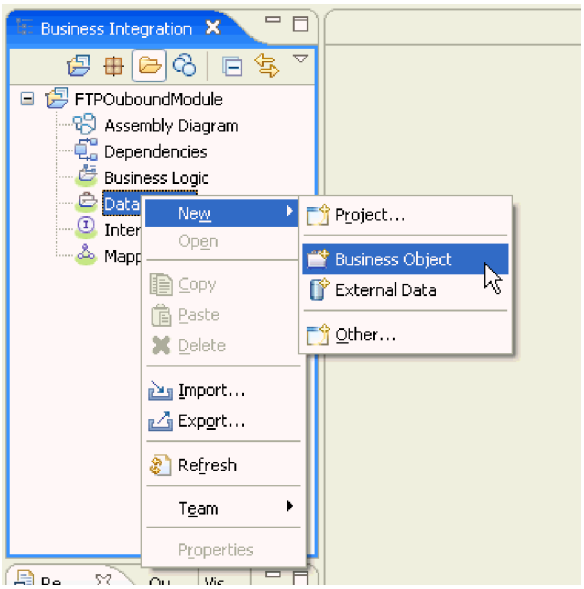


Figure 11. New Business Object selection view

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

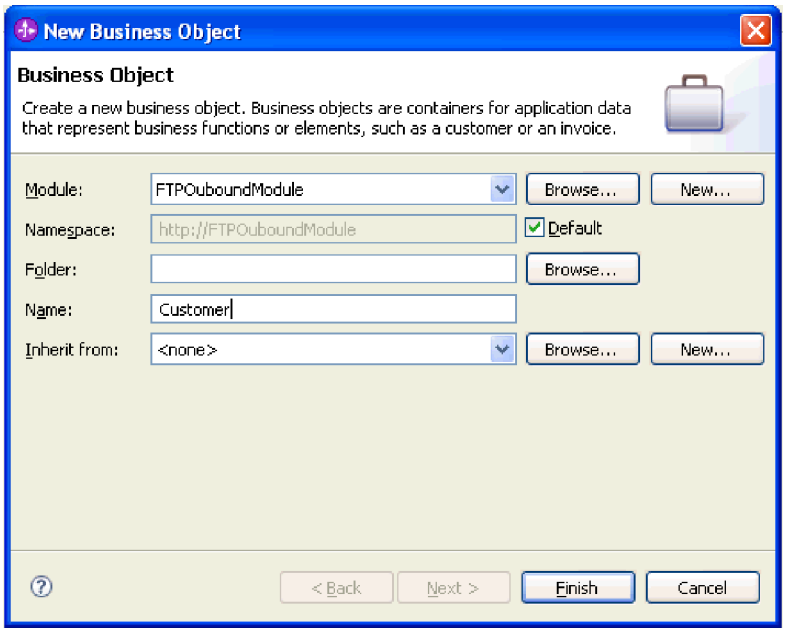


Figure 12. Business Object window

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

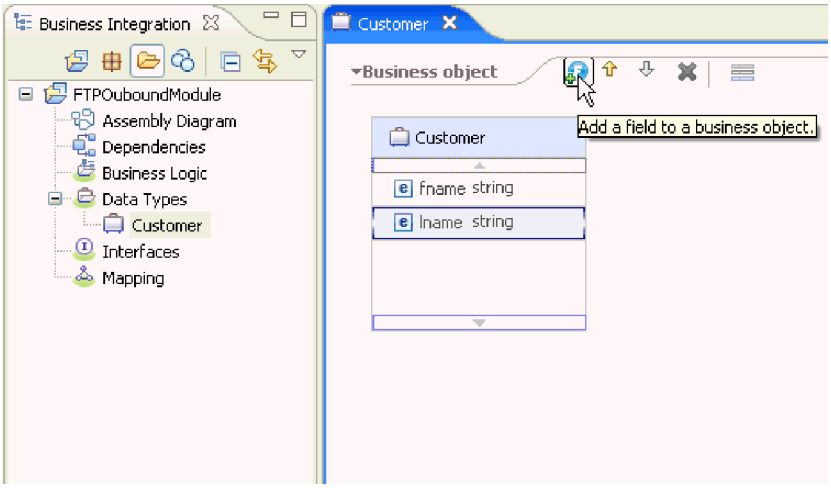


Figure 13. Add Business object fields icon

6. Click the Save icon.
7. Repeat the previous 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 FTP:

Table 4.

Adapter pattern	Description
Inbound FTP pattern	The FTP inbound pattern creates a service that retrieves a file in a specific directory on an FTP server. 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.

Table 4. (continued)

Adapter pattern	Description
Outbound FTP pattern	The FTP outbound pattern creates a service that stores data in a file in a specific directory on an FTP server. 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 FTP 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 FTP service to read from a remote file** and click **Next**.

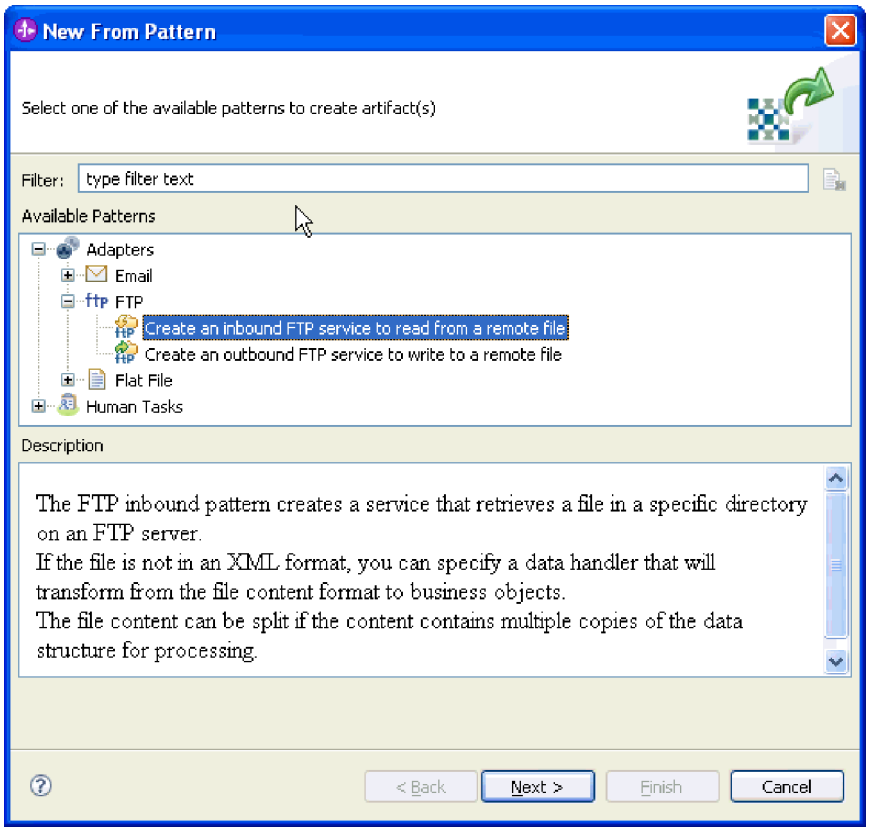


Figure 14. New From Pattern window

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

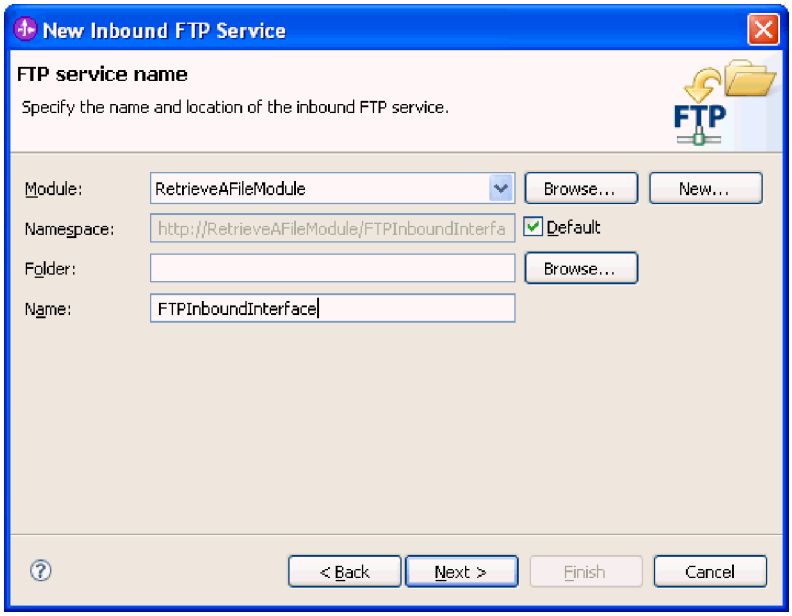


Figure 15. FTP service name window

4. In the Business object and location window, click **Browse** and navigate to the **Customer** business object.
5. Specify both the **Remote directory** and the **Local staging directory** and click **Next**.

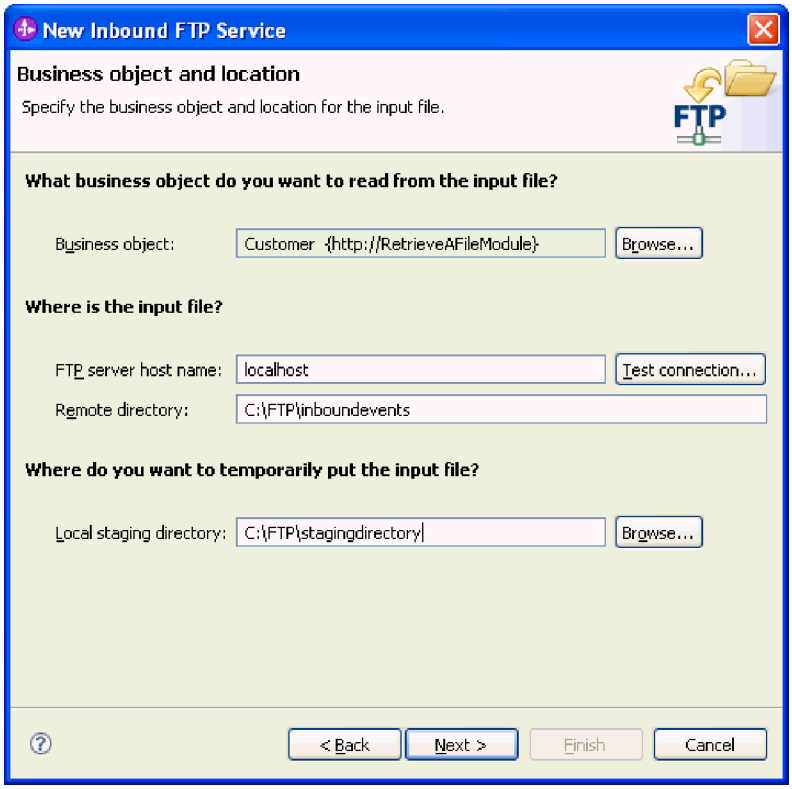


Figure 16. Business object and location window

6. In the FTP server security credential window, select either **Using an existing JAAS alias** or **Using user name and password** and click **Next**.

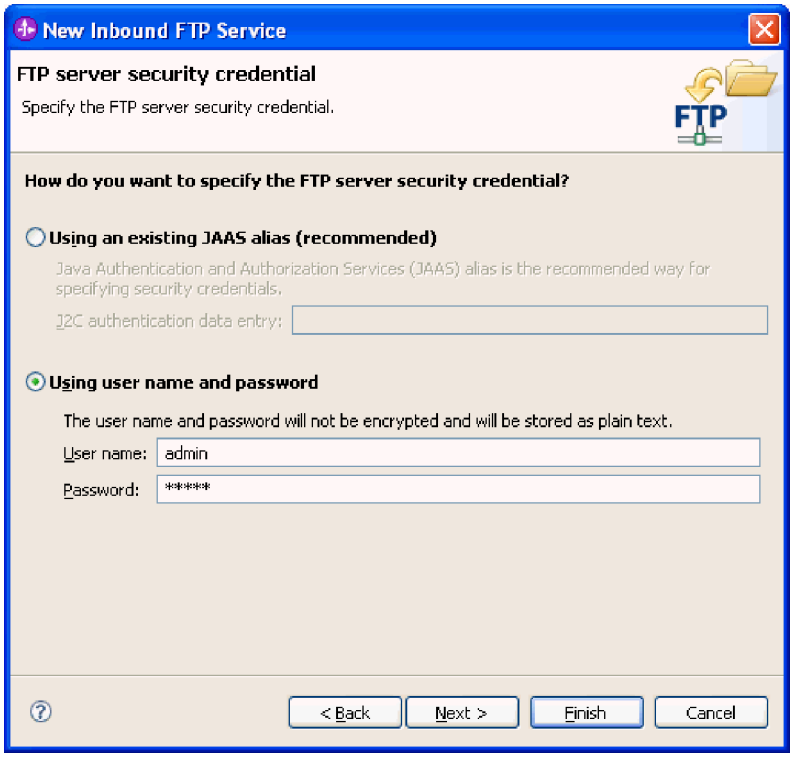


Figure 17. FTP server security credential window

7. 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.
8. Select **Split file content by delimiter** and enter your delimiter, which is `####;\n` in this example. Click **Next**.

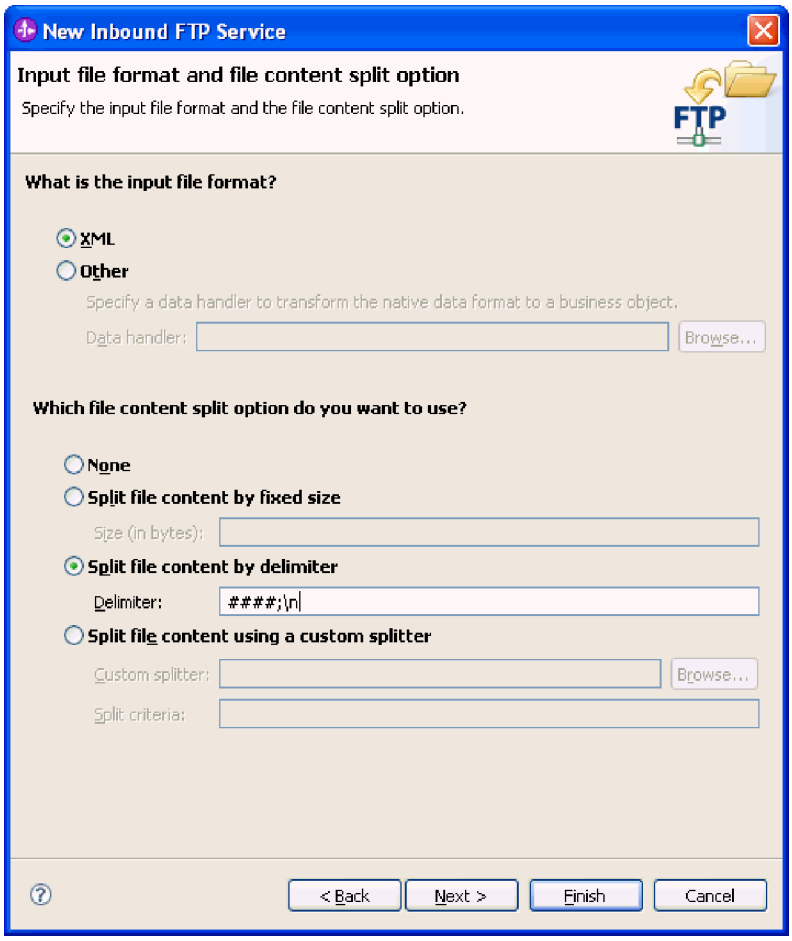


Figure 18. Input file format and file content split option window

9. In the Archive directory and wrapper business object window, specify the **Local archive directory**, which is `FTP\inboundarchive` 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**.

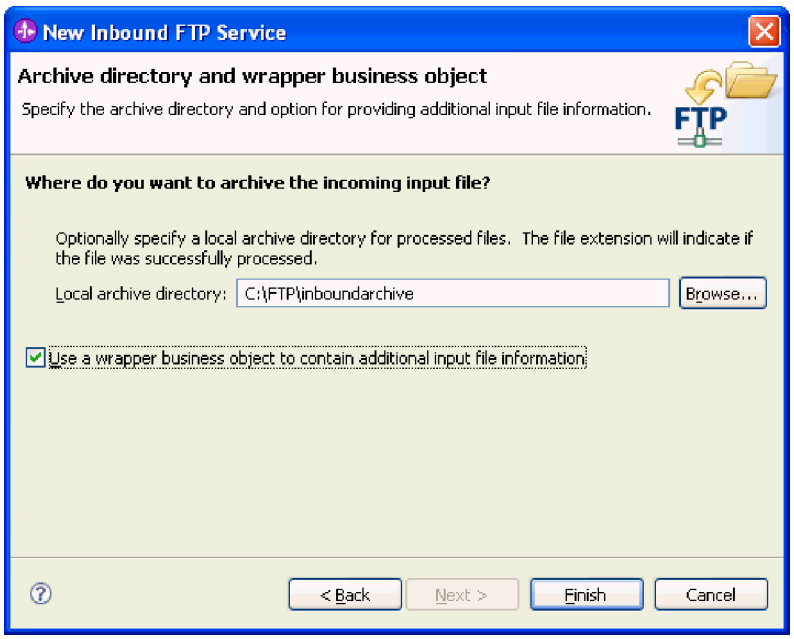


Figure 19. Archive directory and wrapper business object window

Results

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

Table 5.

Artifact	Name	Description
Export	FTPInboundInterface	The export exposes the module externally, in this case, to the WebSphere Adapter for FTP.
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	FTPInboundInterface	This interface contains the operation that can be invoked.
Operation	emitCustomerInput	emitCustomerInput is the only operation in the interface.

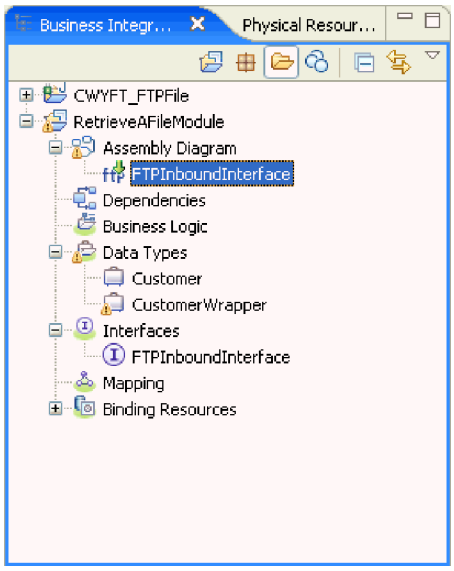


Figure 20. The **Business Integration** *section of the WebSphere Integration Developer window with the new artifacts*

Creating the project

To begin the process of creating and deploying 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.

Before you begin

Make sure you have gathered the information you need to establish a connection to the FTP server. For example, you need the name or IP address of the FTP server and the user ID and password needed to access it.

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** → **WebSphere Integration Developer 6.1** → **WebSphere Integration Developer 6.1**.
   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. 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**.

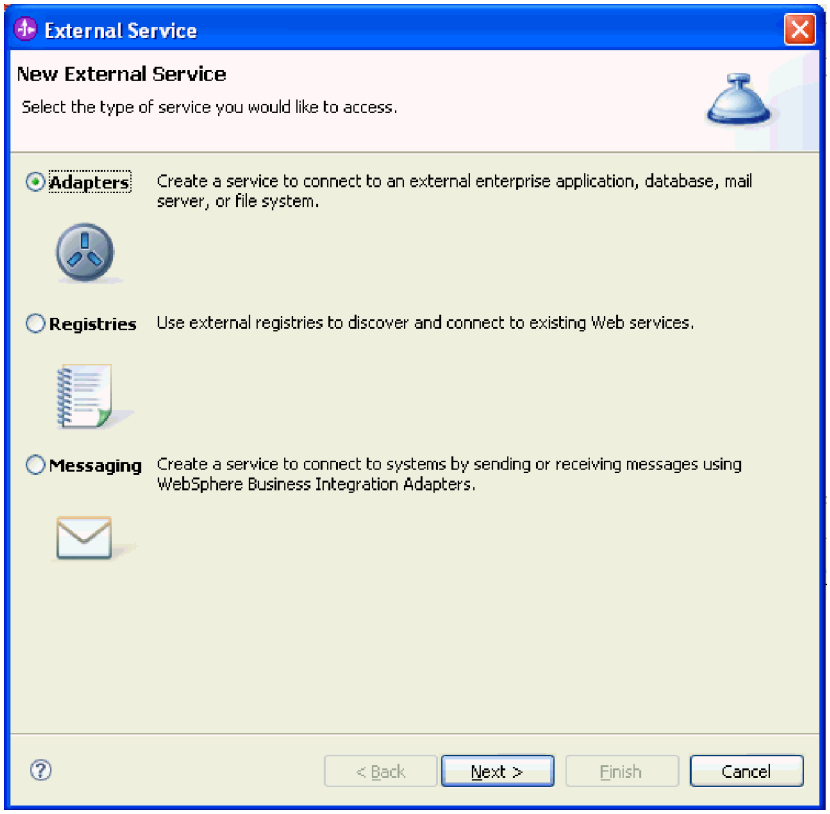


Figure 21. The New external service window

4. In the Select an Adapter window, select **IBM WebSphere Adapter for FTP** and click **Next**.

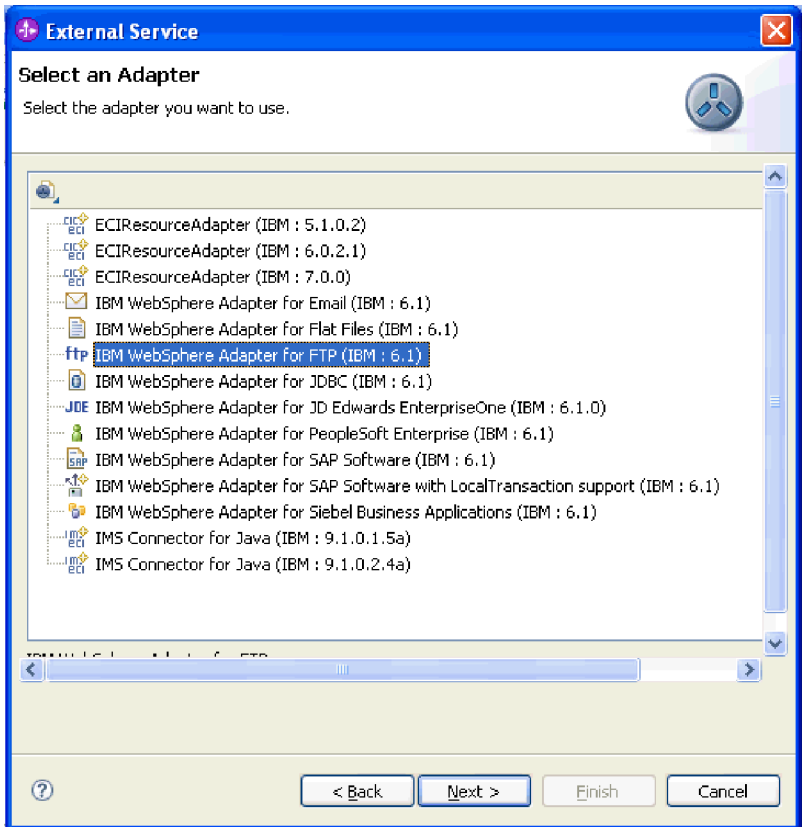


Figure 22. The Select an Adapter window

5. From the Adapter Import window, either keep the default name in the **Connector project** field, or type a new one, and select the **Target runtime**.

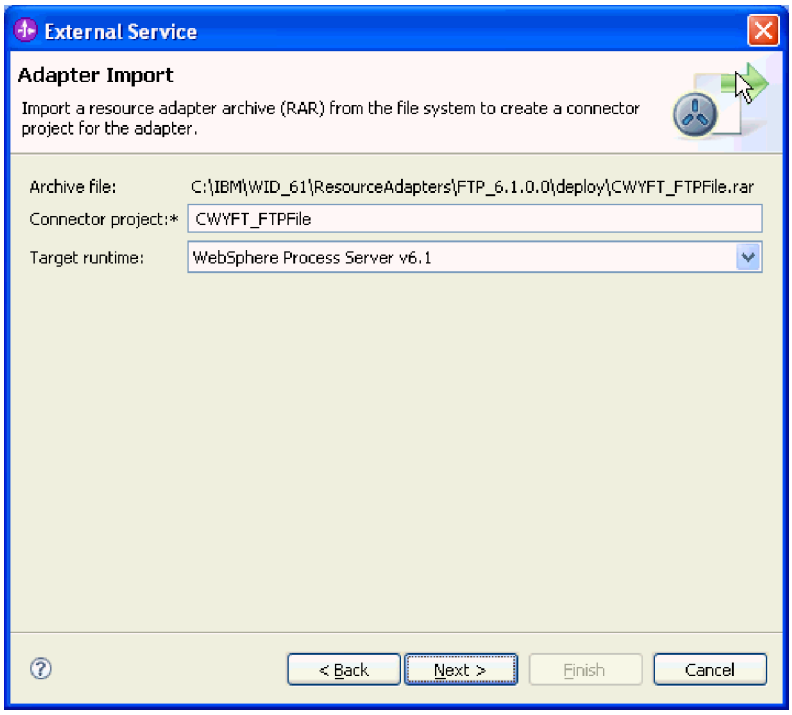


Figure 23. The Adapter Import window

6. Click **Next**.

Results

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

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

Specify deployment and runtime properties that theexternal service wizard uses to connect to the FTP server.

Before you begin

Before you can specify the connection properties, you must have started the external service wizard.

About this task

Theexternal service wizard requires this information to connect to the database for discovery and for creating the service description.

Procedure

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

2. In the Server Configuration Properties window, specify a **J2C Authentication Data Entry**, which is the authentication alias that you set up on the FTP server. The name is case-sensitive. For more details, see ″Creating the authentication alias.″
3. In the **Deploy connector project** field, specify whether to include the adapter files in the module. Choose one of the following options:
   - **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.
4. Define the following FTP system connection information for your module. For more details about the properties in this window, see the topic about managed connection factory properties in this documentation.
   - **Host name** - Specifies the host name of the FTP Server.
   - **Directory** - Specifies the output directory on the FTP server.
   - **Port number** - Specifies the port number of the FTP Server.
   - **Protocol** - Specifies either normal FTP or secure FTP (FTPS).
5. Optional: Click **Advanced** to specify additional properties, such as those that control working with a second FTP server, bidi formatting, a staging directory, and sequence file selection.
6. Optional: In the Service properties section of the window, you can specify a Java Authentication and Authorization Services (JAAS) alias for the adapter to use at run time.
7. In the **Data binding** field, select one of the following:
   - **Use default data binding 'FTPFileBaseDataBinding' for all operations**

     A non-configured data binding for all of the operations used in the service.
   - **Use a data binding configuration for all operations**

     A configured data binding that will be used for all of the operations used in the service.
   - **Specify a data binding for each operation**

     No default binding is specified. You will select a specific data binding for each operation used in the service.
8. Optional: Select the **Change logging properties for wizard** checkbox if you want to define the level of logging for this module.

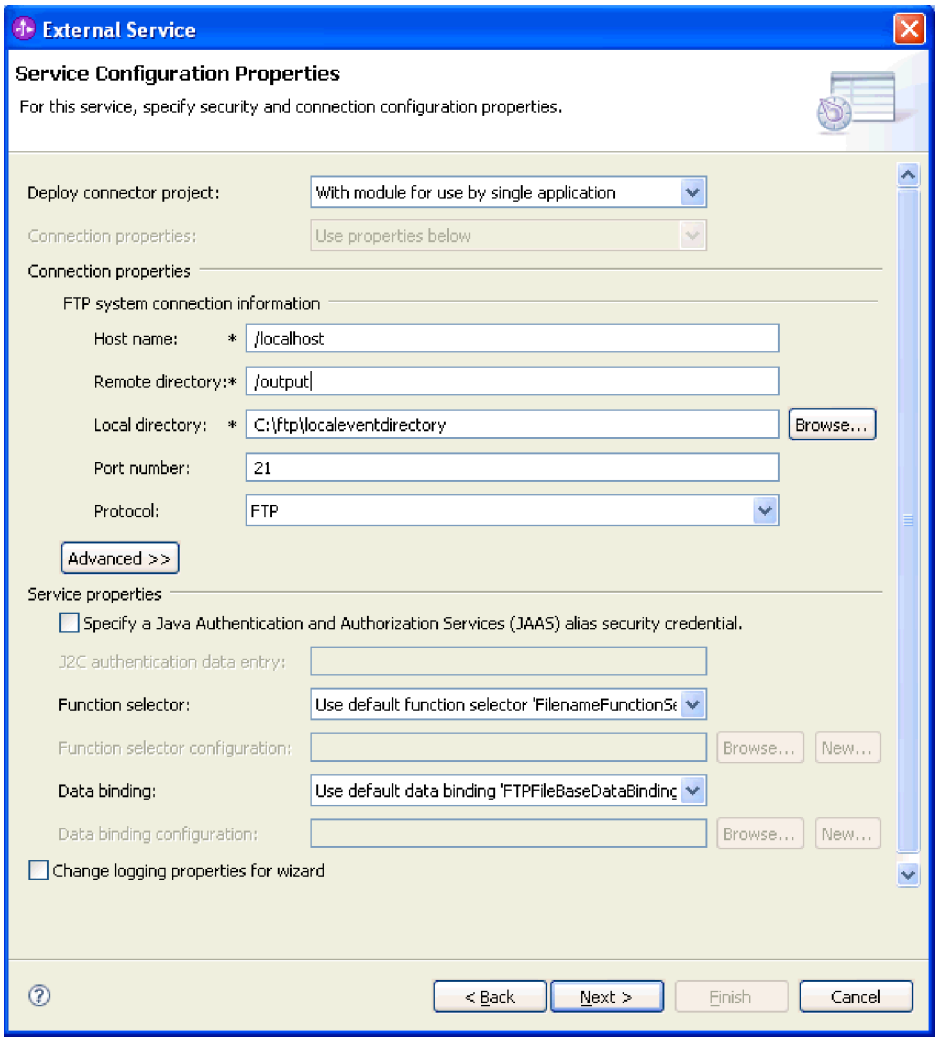


Figure 24. The Server Configuration Properties window

Results

The external service wizard now has the information it needs to connect to the FTP server.

If you selected either the **Use default data binding 'FTPFileBaseDataBinding' for all operations** or **Specify a data binding for each operation** Data binding configuration options, click **Next** to continue to work in the wizard to select a data type and to name the operation associated with the data type.

If you selected the **Use a data binding configuration for all operations** Data binding configuration option, proceed to "Configuring the data binding" on page 54.

Selecting a data type and operation name

Use the external service wizard to select a data type and to name the operation associated with the data type. For outbound communications, the external service wizard gives you the choice of three different data types: user defined type, generic FTP business object, and generic FTP business object with business graph. 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 FTP server before you can complete the following steps.

About this task

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

Procedure

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

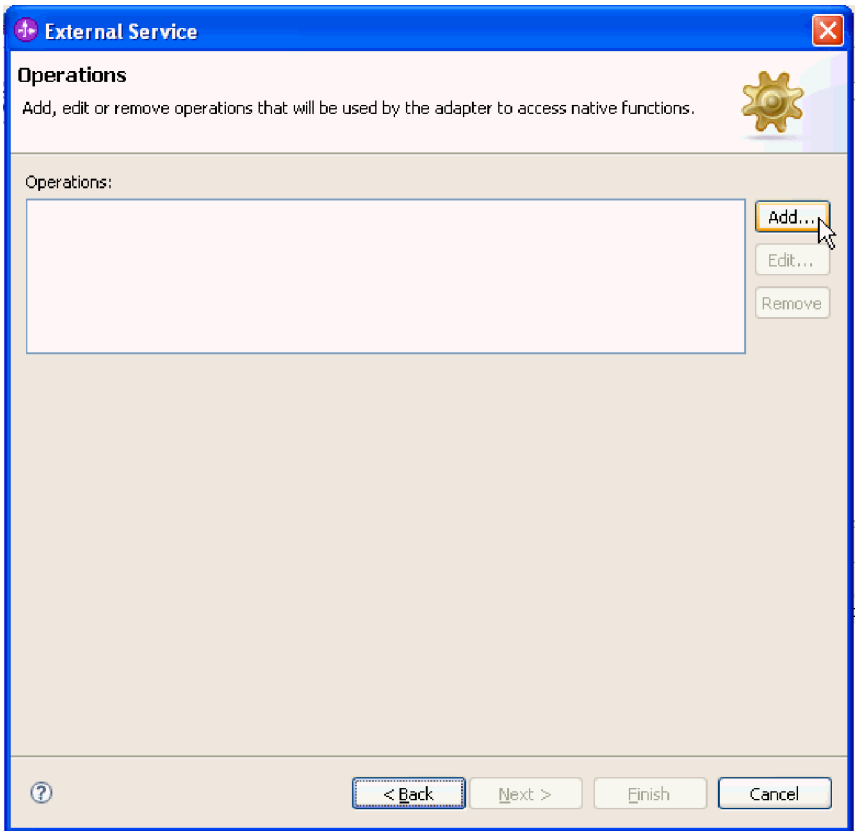


Figure 25. Adding an operation

2. In the Add Operations window, select an **Operation kind** and **The data type for the operation input**, and click **Next**. If you select **User defined type**, you must provide a user-defined data binding to support it. The **Generic FTP business object** provided data binding only supports generic input types for the supported operations.

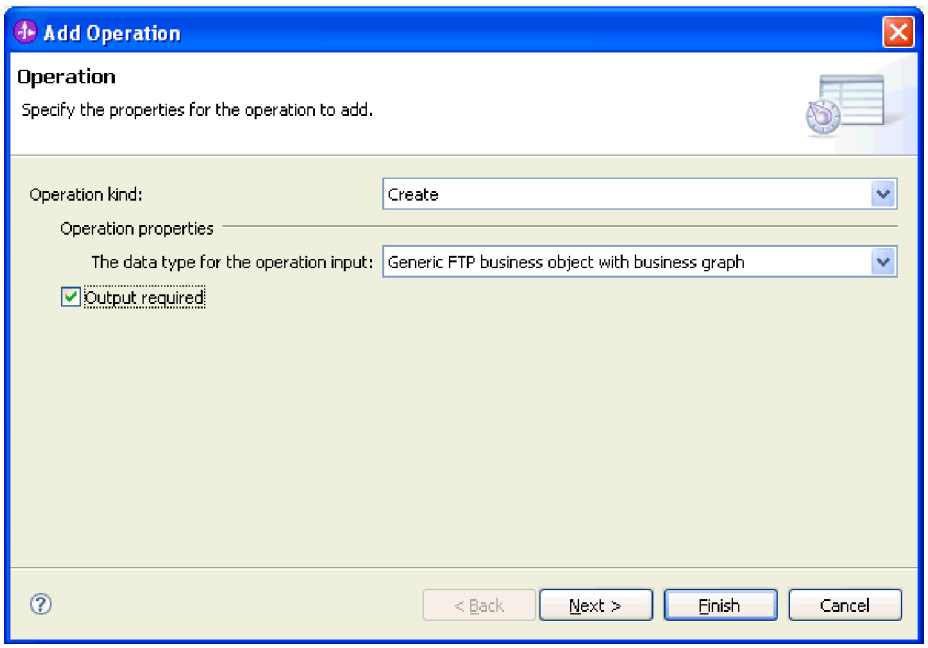


Figure 26. Selecting a data type

3. Optional: 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. Click **Next**.
4. In the Operation window, type in an **Operation name**. Name the operation something meaningful. If this module is going to be used to create a new customer record, name it something like `createCustomer`. For more information on the types of operations the adapter can perform, see the Table 1 on page 5 topic.

   Note: Names cannot contain spaces.

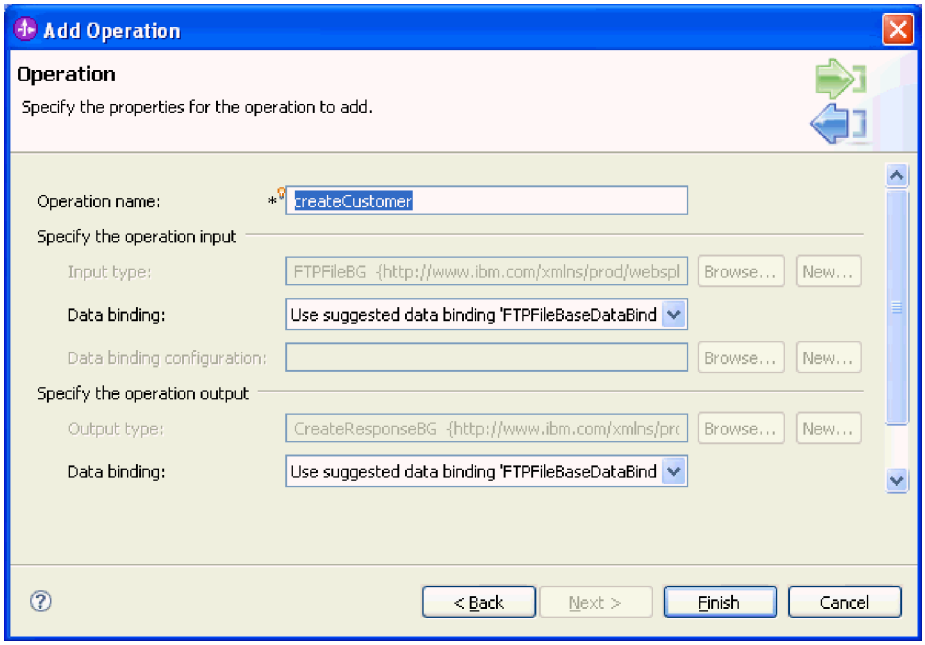


Figure 27. Naming the operation

Results

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

If you chose to use a configured data binding, continue to work from the current external service wizard window, to add and configure a data binding to be used with the module.

If you chose to use the default data binding, proceed to "Setting interaction specification properties and generating the service" on page 63.

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 a 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 a 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.

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 windows described in this documentation.

About this task

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

Procedure

1. Select **New** next to the **Data binding configuration** field in the **Specify the operation input** section of the window.

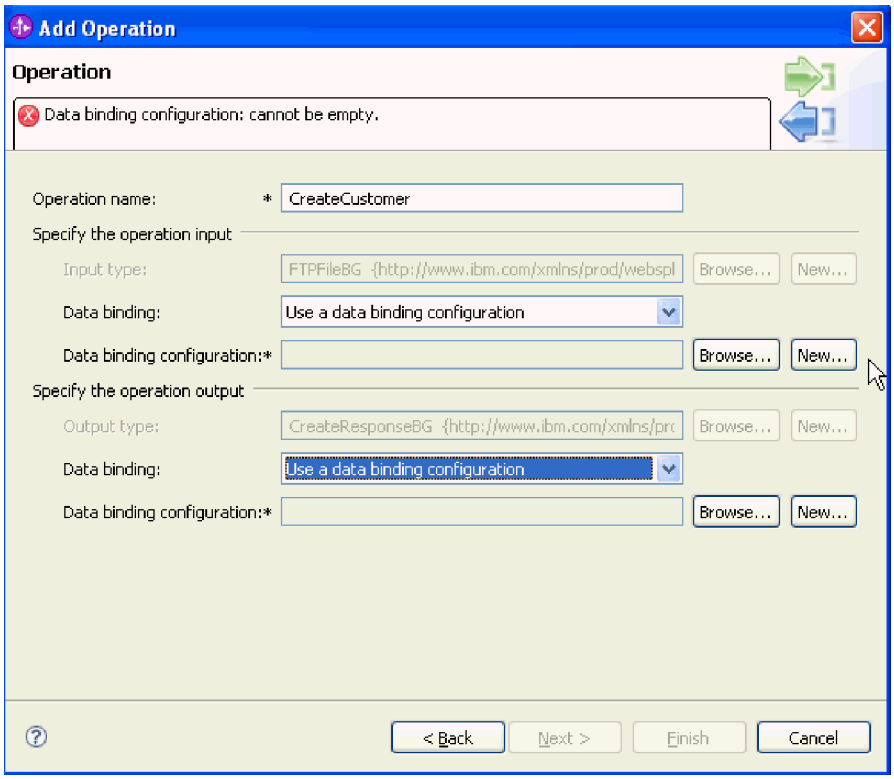


Figure 28. Operation window

2. Type a **Name** for the data binding and click **Next**. A data binding has a pointer to a data handler, so the name should reflect this. For example: `FTPOutboundDB_XML`, or `FTPOutboundDB_Delim1`.

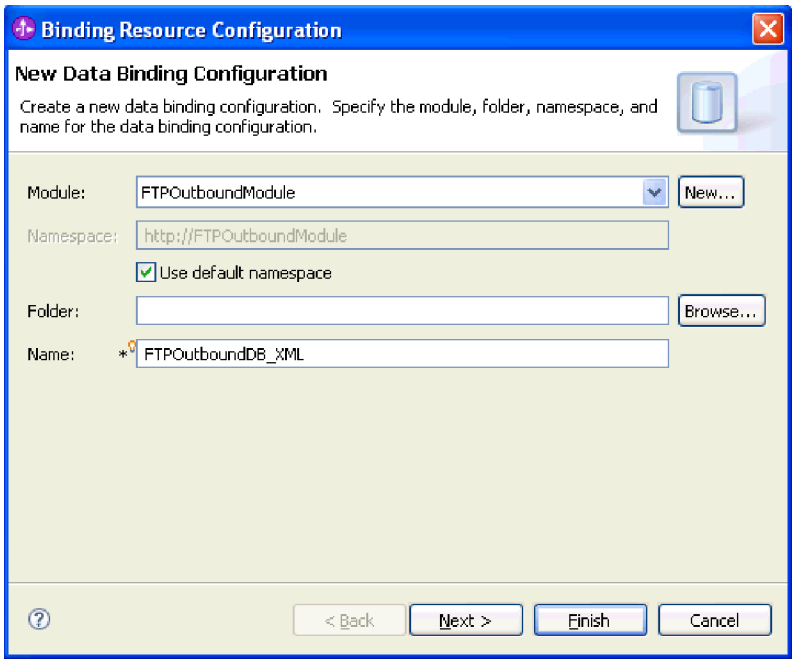


Figure 29. Naming the data binding

3. On the Select a configuration type window, leave the **Data binding** radio button selected.
4. Click **Browse** to select a class name. The term "class" here refers to the data binding class associated with the data binding you are in the process of creating for this module.
5. On the Data Binding Selection window, click the **Show data binding classes** radio button selected.
6. Select the correct data binding class for your data type and click **OK**.

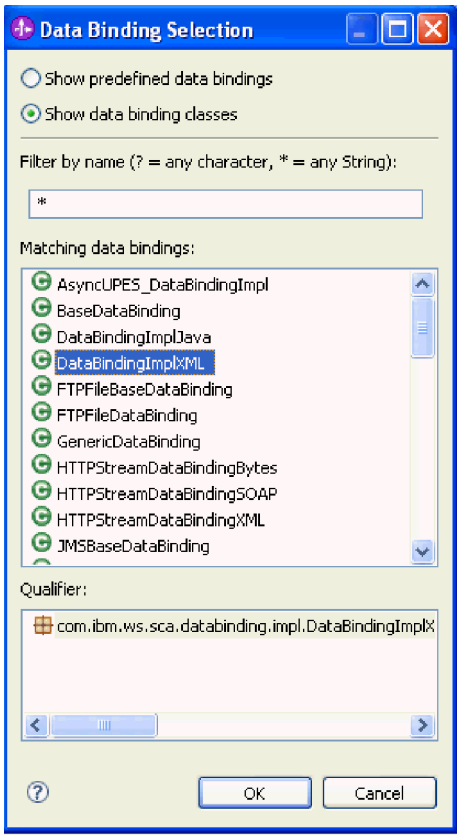


Figure 30. Selecting a data binding

The external service wizard will default to the correct data binding class for your data type. For more information on data bindings, see the topic devoted to outbound data transformation in this documentation. The data binding class name will show on the Select a configuration type window.

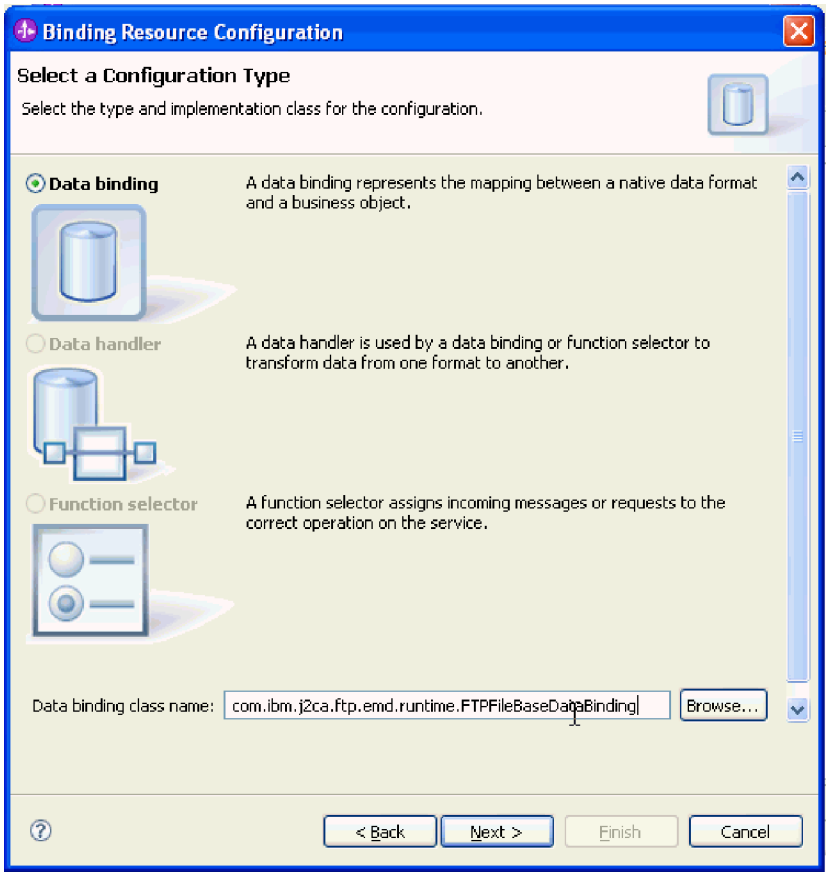


Figure 31. Data binding class is populated in the Select a Configuration Type window

7. Click **Next**.

Results

A data binding is configured for use with the module.

What to do next

From the current external service wizard window, you will proceed to select a data handler for the module.

Configuring data handlers

When you select a data type that contains business objects, you need to specify the data handler that will perform the conversions between a business object and a native format.

Before you begin

You must have created a data binding before specifying the data handlers for the module.

About this task

To specify data handlers, follow this procedure.

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 windows described in this documentation.

Procedure

1. Select **New** for the **DataHandlerConfigurationName** in the Data Binding Properties window.
2. In the **Name** field of the New Data Handler Configuration window, type a name for the data handler.

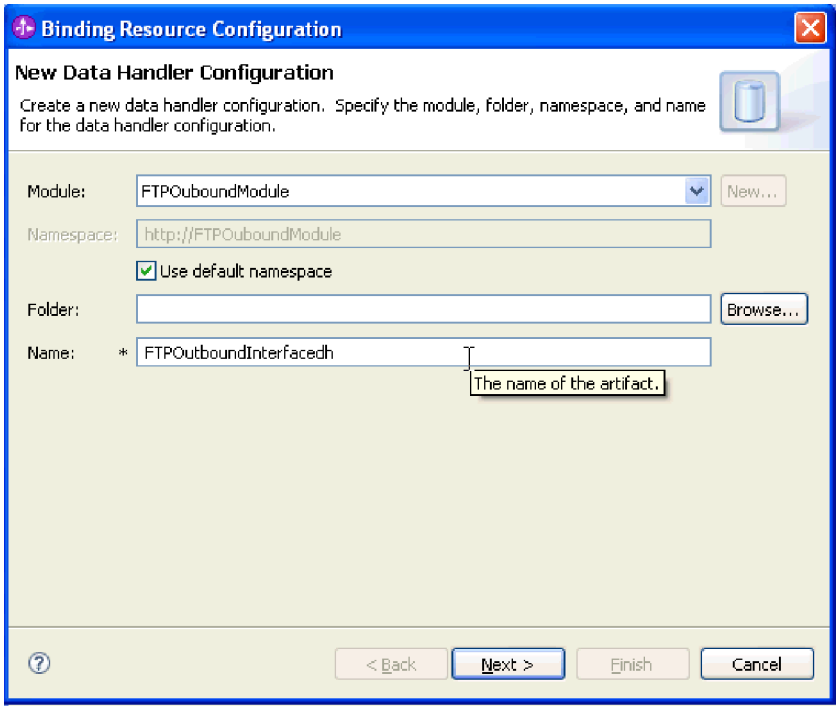


Figure 32. Specifying the data handler name

3. Click **Next**.
4. On the Select a configuration type window, leave the **Data Handler** radio button selected and click **Browse**.

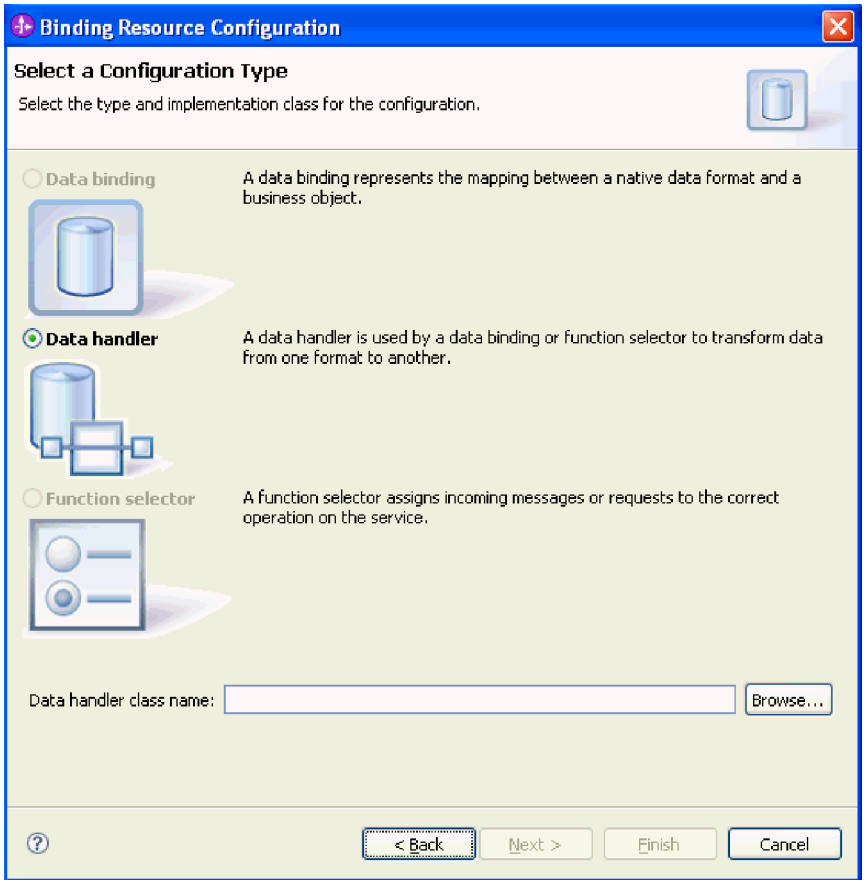


Figure 33. Choosing the data handler configuration type

5. In the Data Handler Selection window, select the correct data handler for the type of transformation required by your business objects and click **OK**.

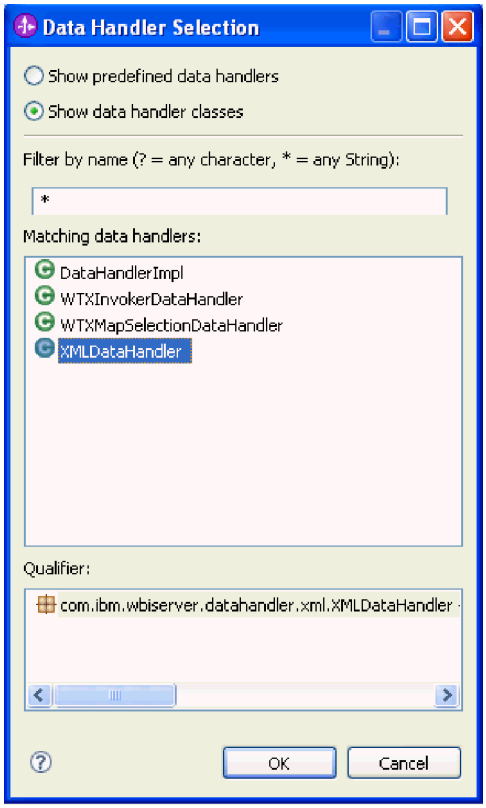


Figure 34. Selecting the data handler class

6. On the Select a configuration type window, the **Data handler class name field** is populated. Click **Next** to continue.
7. On the Specify Properties window, type in a value for the **encoding** field and click **Finish**. This value indicates the type of character encoding the adapter will use during data transformation. For more information on the encoding property, see the topic about FTP business object properties in this documentation.
8. Click **Finish** in the Data Binding Properties window.
9. On the Operation window, select **New** for the **DataBinding type** field in the **Specify the details of the operation output** section of the window.

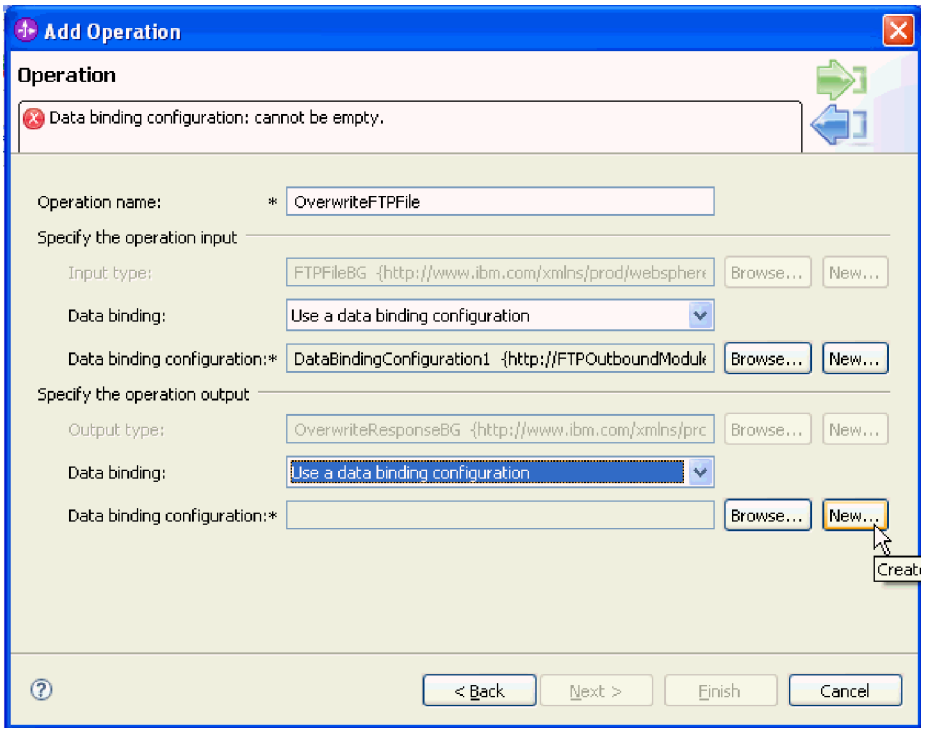


Figure 35. Add Operation window

10. In the **Name** field, type a name for the data binding and click **Next**.

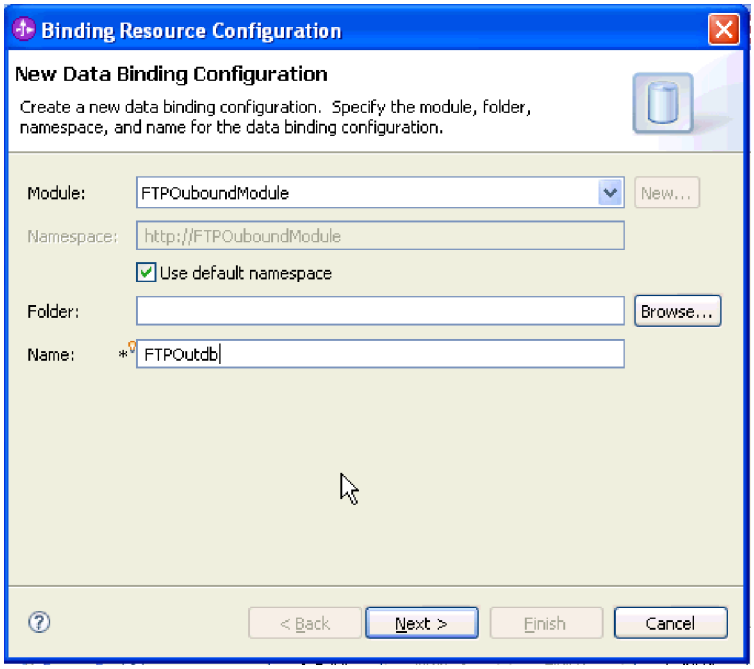


Figure 36. Naming the data binding

11. On the Select a Configuration Type window, leave the **Data binding** radio button selected.
12. Click **Finish**.
13. Click **Finish** on the Add operation window.

Results

Data handlers are created.

What to do next

In the wizard, continue to specify interaction specification properties and generate artifacts for the module.

Setting interaction specification properties and generating the service

Interaction specification properties are optional. If you choose to set them, the values you specify will appear as defaults in all parent FTP business objects generated by the external service wizard. Interaction specification properties control the interaction for an operation. 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 topic about it in this documentation.

Procedure

1. Optional: To set interaction specification properties, populate the fields in the Operations window. You can also click **Advanced** to add additional property details.
   a. Type values for any fields you want to set as defaults.
   b. Click **Next**.

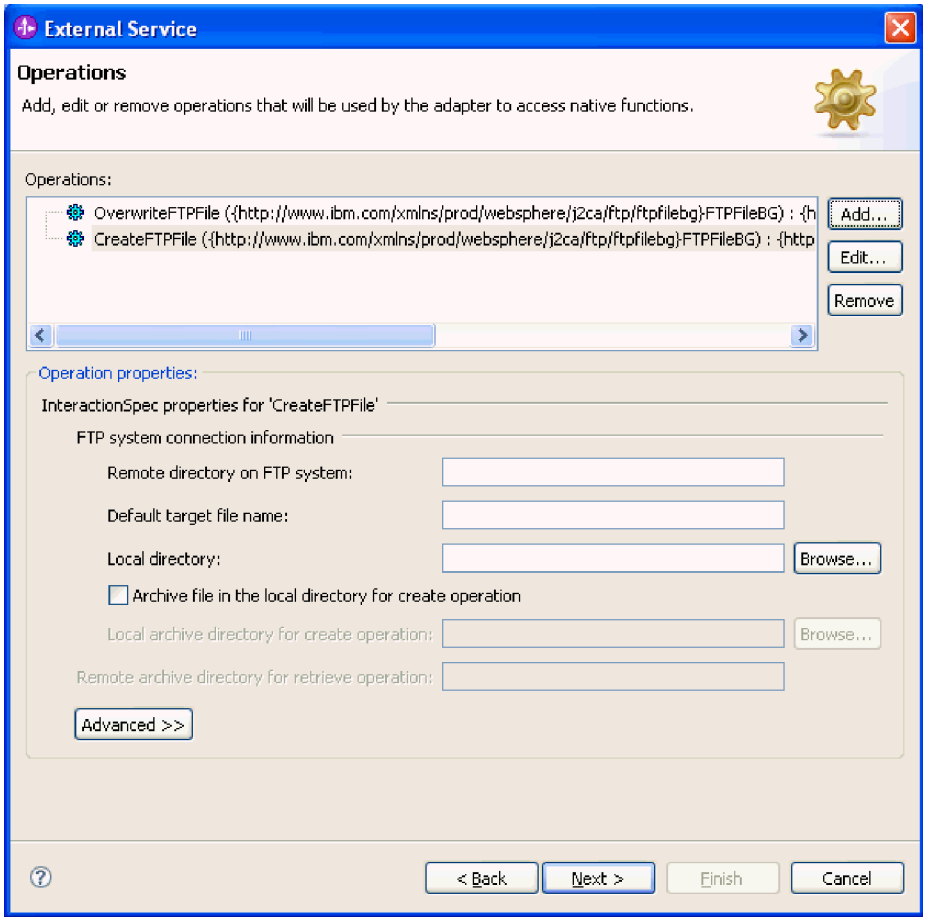


Figure 37. Interaction specification properties

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

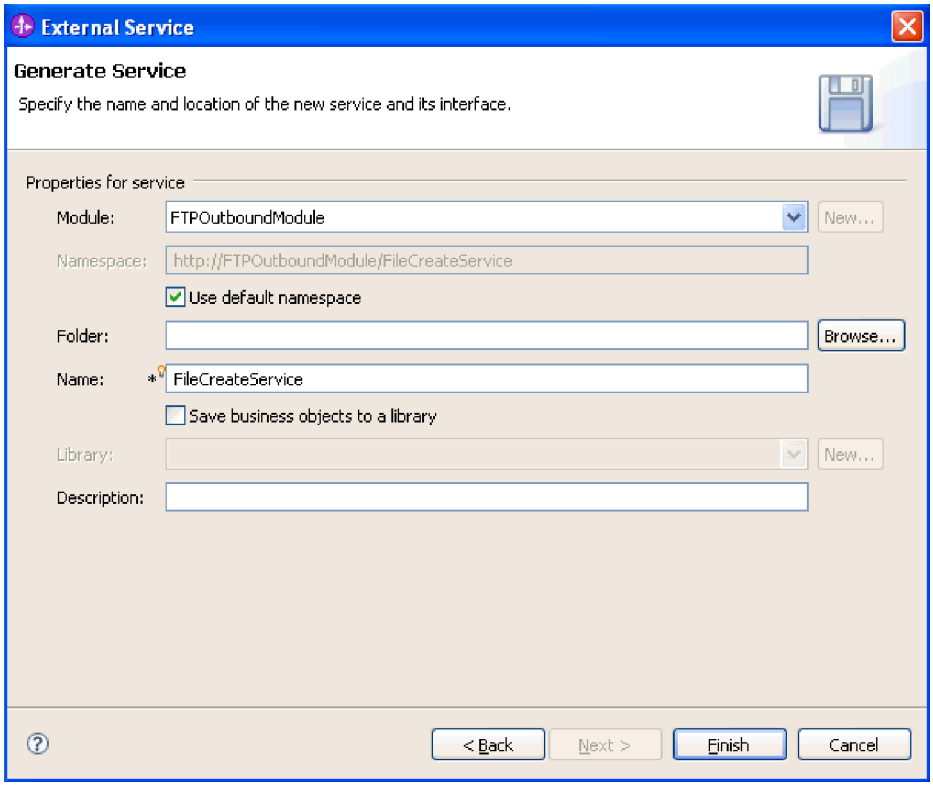


Figure 38. Naming the interface

3. Click **Finish**. The WebSphere Integration Developer assembly diagram opens and the interface you created is displayed.

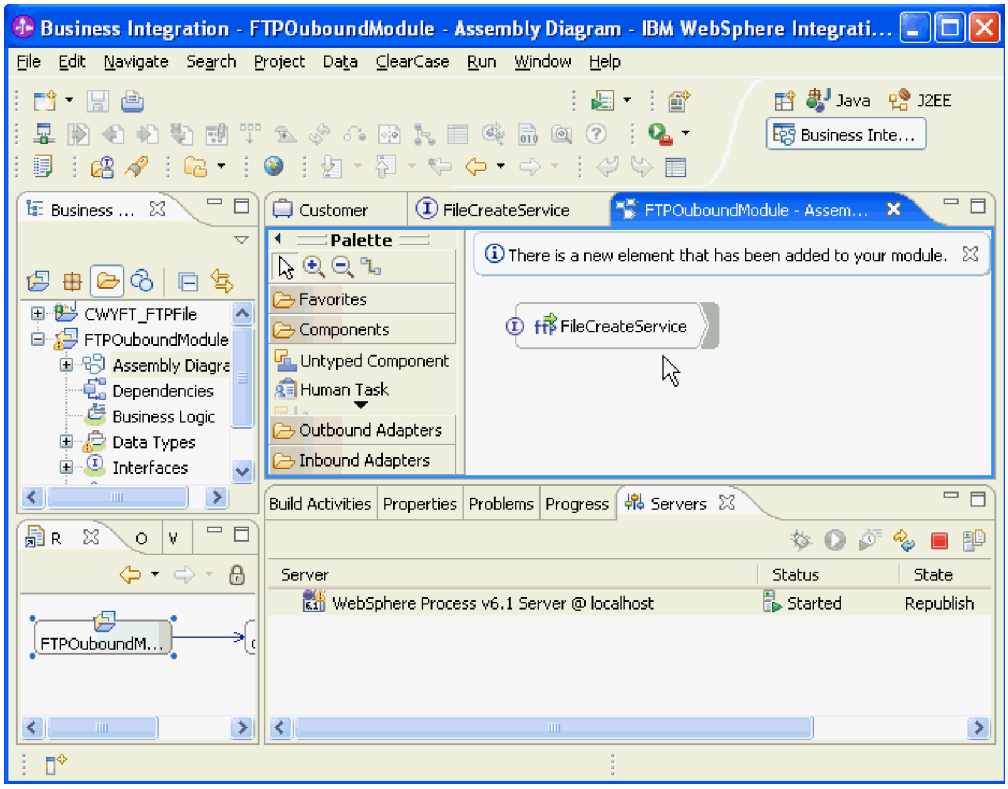


Figure 39. Interface in WebSphere Integration Developer

4. Optional: Repeat the previous steps to add all other required operations, including the bindings, data handlers, and interaction specifications.

Results

WebSphere Integration Developer generates the artifacts and the 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 to the server.

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

Specify deployment and runtime properties that the external service wizard uses to connect to the FTP server.

Before you begin

Before you can specify the connection properties, you must have started the external service wizard.

About this task

The external service wizard requires this information to connect to the database for discovery and for creating the service description.

Procedure

1. In the Processing Direction window, select **Inbound** and click **Next**.
2. In the **Deploy connector project** field, specify whether to include the adapter files in the module. Choose one of the following options:
   - **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.
3. Define the following FTP system connection information for your module. For more details about the properties in this window, see the topic about managed connection factory properties in this documentation.
   - **Host name** - Specifies the host name of the FTP Server.
   - **Remote directory** - Specifies the directory on the FTP server where the adapter polls and picks up files.
   - **Local directory** - Specifies the directory on the adapter workstation where the event files are downloaded from the FTP server.
   - **Port number** - Specifies the port number of the FTP Server.
   - **Protocol** - Specifies either normal FTP or secure FTP (FTPS).

   Click **Advanced** to specify additional properties, such as those that control event polling and persistence, archiving, bidi formatting, and logging and tracing.

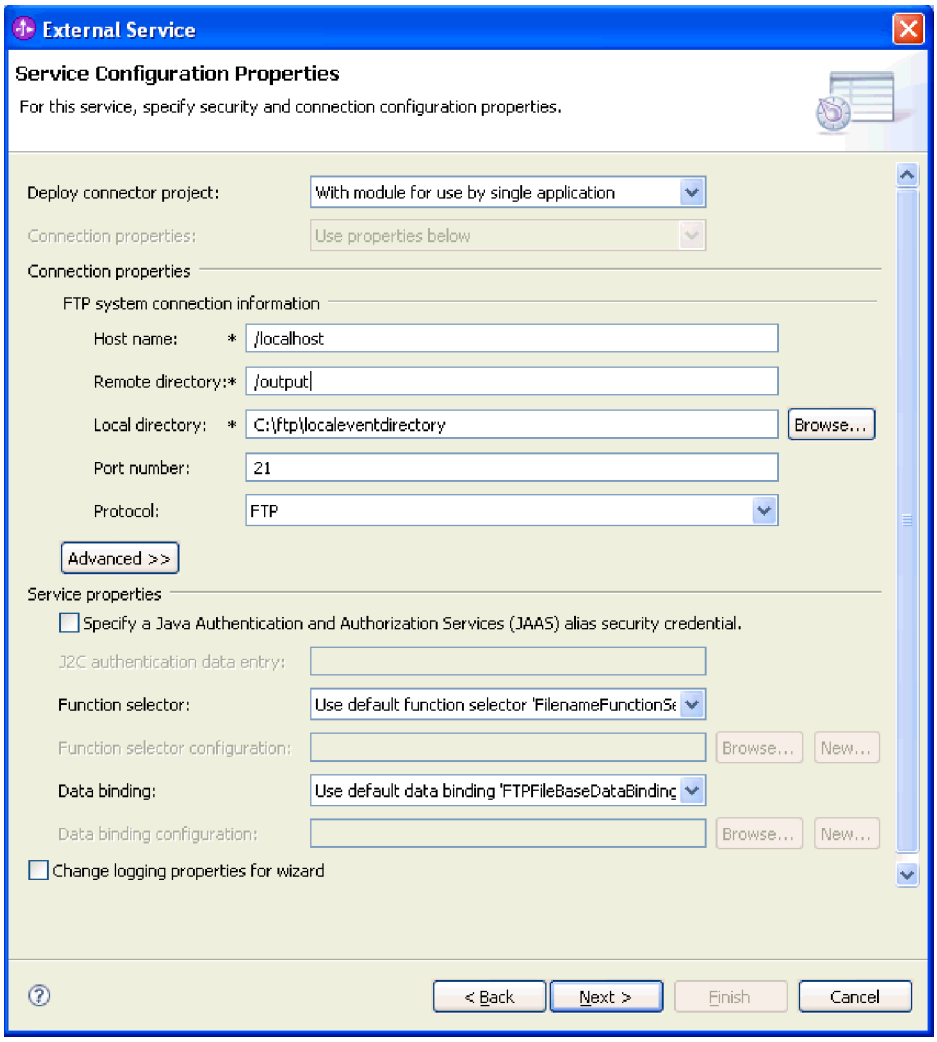


Figure 40. The Server Configuration Properties window

4. In the Service properties section of the window, select one of the following options from the **Function selector** field. A function selector assigns incoming messages or requests to the correct operation on the service.
   - **Use default function selector 'FilenameFunctionSelector'**

     If choosing to use this option, click **Next**.
   - **Use a function selector configuration**

     If choosing this option, complete the following steps:

     a. Click **New** next to the **Function Selector Configuration** field.

     b. In the New Function Selector Configuration window, specify a **Name** for the function selector. Click **Next**

     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, you can edit it using the assembly editor.

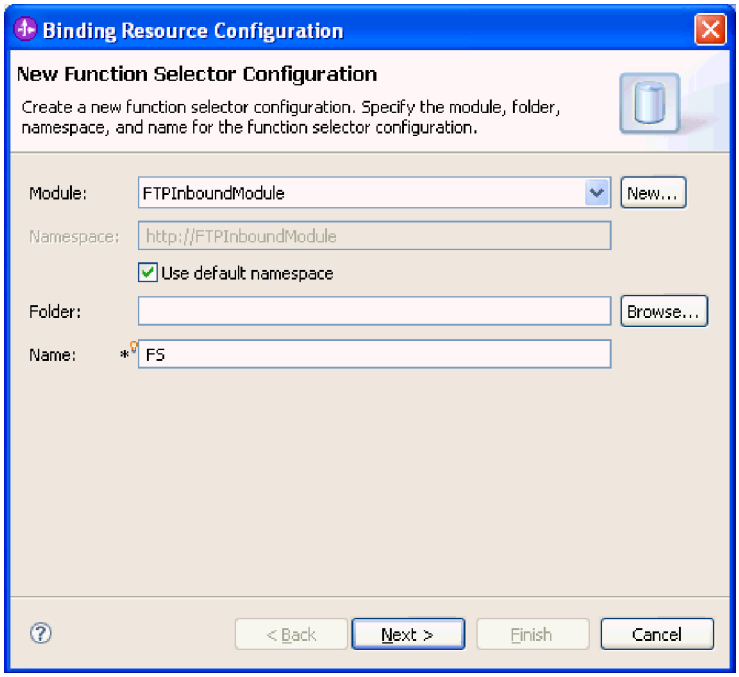


Figure 41. The New Function Selector Configuration window

5. From the Select a Configuration Type window, click **Browse** next to the **Function selector class name** field.
6. From the Function Selector Selection window, choose the function selector. Click **OK**.

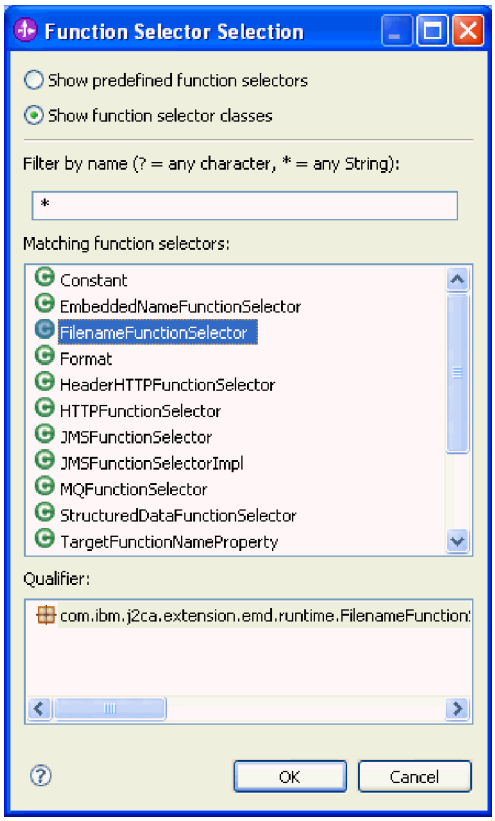


Figure 42. The Function Selector Selection window

7. Click **Next** in the Select a Configuration Type window.
8. Click **Finish** in the Function selector properties window.
9. Click **Next** in the Service Configuration Properties window.

Results

The external service wizard now has the information it needs to connect to the FTP server.

If you selected either the **Use default data binding 'FTPFileBaseDataBinding' for all operations** or **Specify a data binding for each operation** Data binding configuration options, click **Next** to continue to work in the wizard to select a data type and to name the operation associated with the data type.

If you selected the **Use a data binding configuration for all operations** Data binding configuration option, proceed to "Configuring the data binding" on page 54.

Selecting a data type and operation name

Use the external service wizard to select a data type and to name the operation associated with the data type. For inbound communications, the external service wizard gives you the choice of three different data types: user defined type, generic FTP business object, and generic FTP business object with business graph. 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 FTP server before you can complete the following steps.

About this task

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

Procedure

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

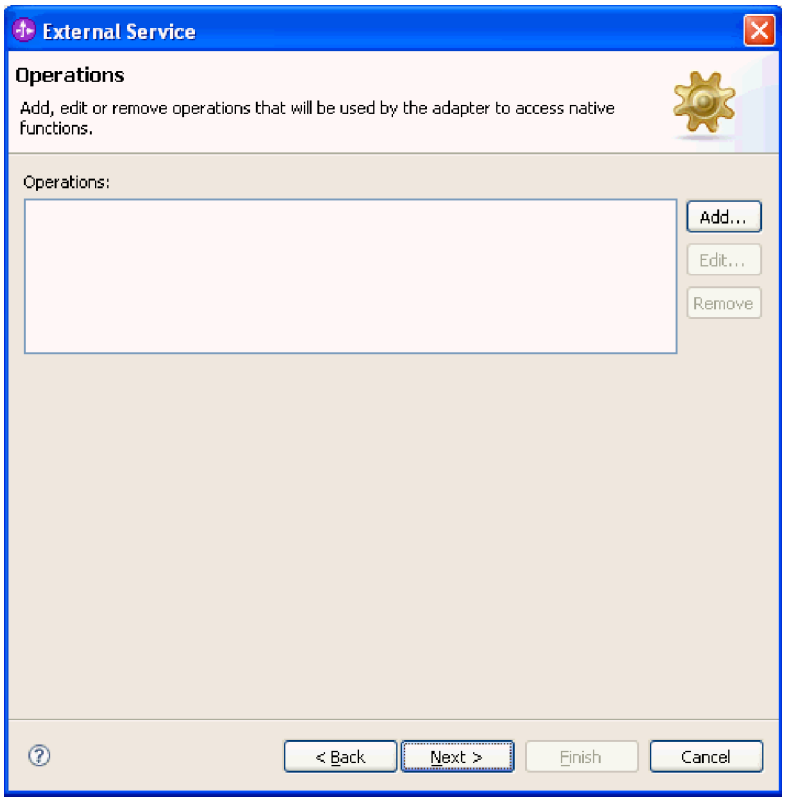


Figure 43. Adding an operation

2. In the Add Operations window, select **The data type for the operation input**, and click **Next**. If you select **User defined type**, you must provide a user-defined data binding to support it. The **Generic FTP business object** provided data binding only supports generic input types for the supported operations.
3. In the Operation window, type a name in the **Operation name** field or keep the default `emitFTPFile` name.

   Note: Names cannot contain spaces.

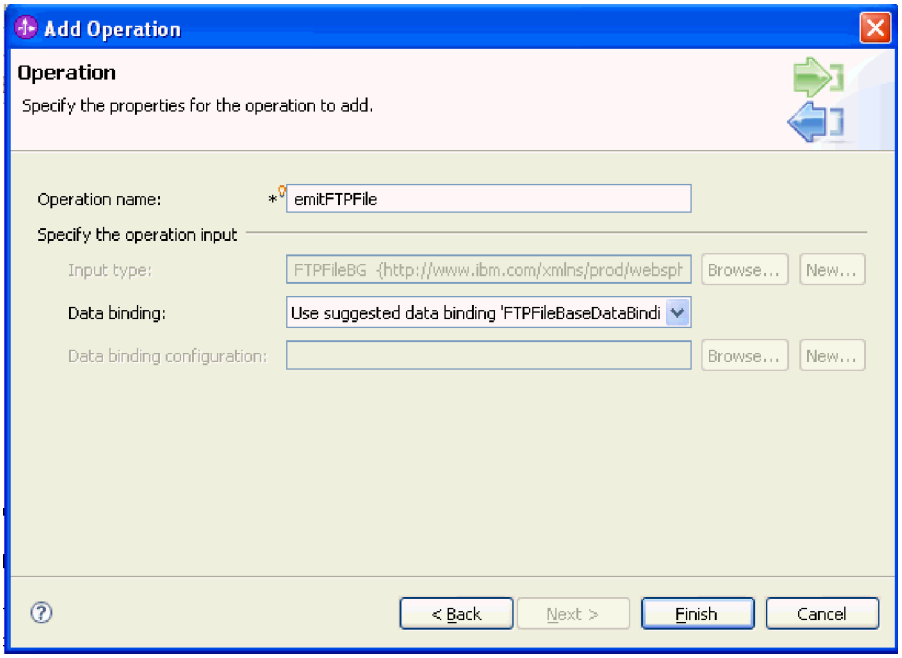


Figure 44. Naming the operation

Results

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

If you chose to use a configured data binding, continue to work from the current external service wizard window, to add and configure a data binding to be used with the module.

If you chose to use the default data binding, proceed to "Generating the service" on page 79.

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 in the corresponding fields in a 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 a 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 windows described in this documentation.

Procedure

1. Select **New** next to the **Data binding configuration** field in the **Service properties** section of the window.

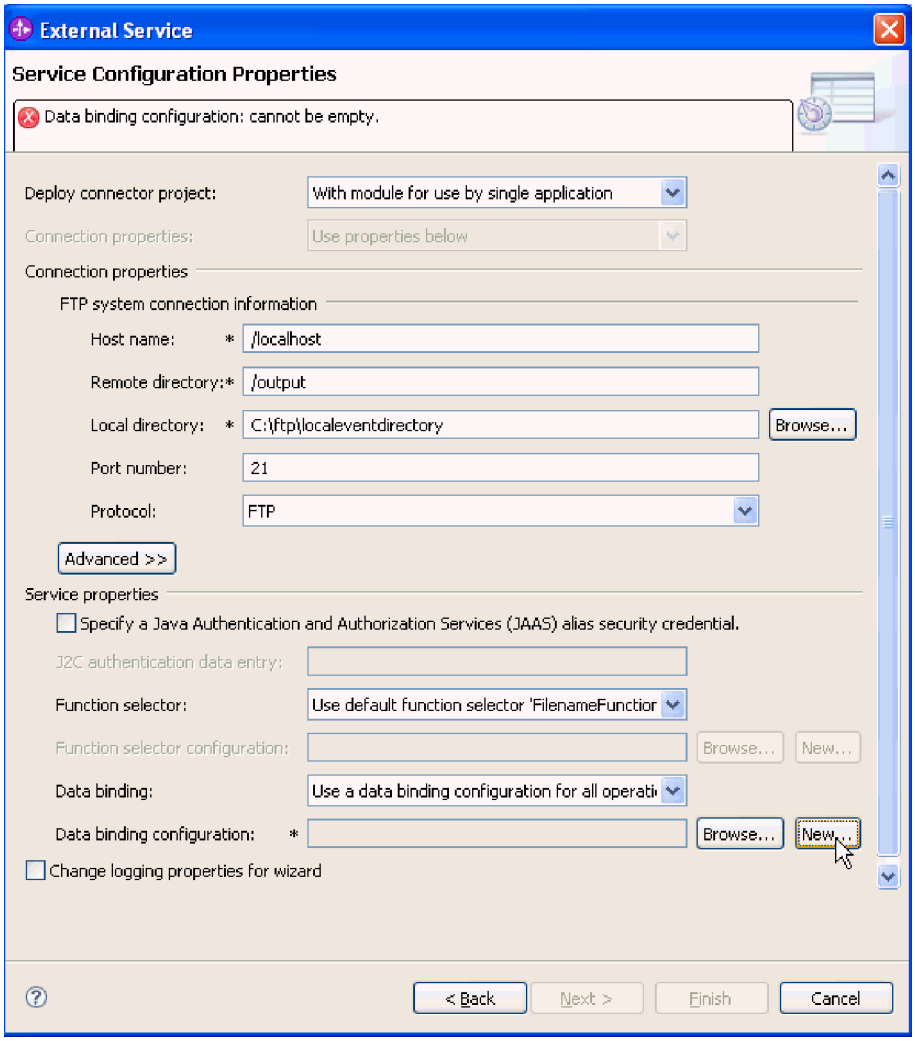


Figure 45. Service Configuration Properties window

2. In the **Name** field, type a name for the data binding and click **Next**. A data binding has a pointer to a data handler, so the name should reflect this. For example: `FTPInboundDB_XML`.

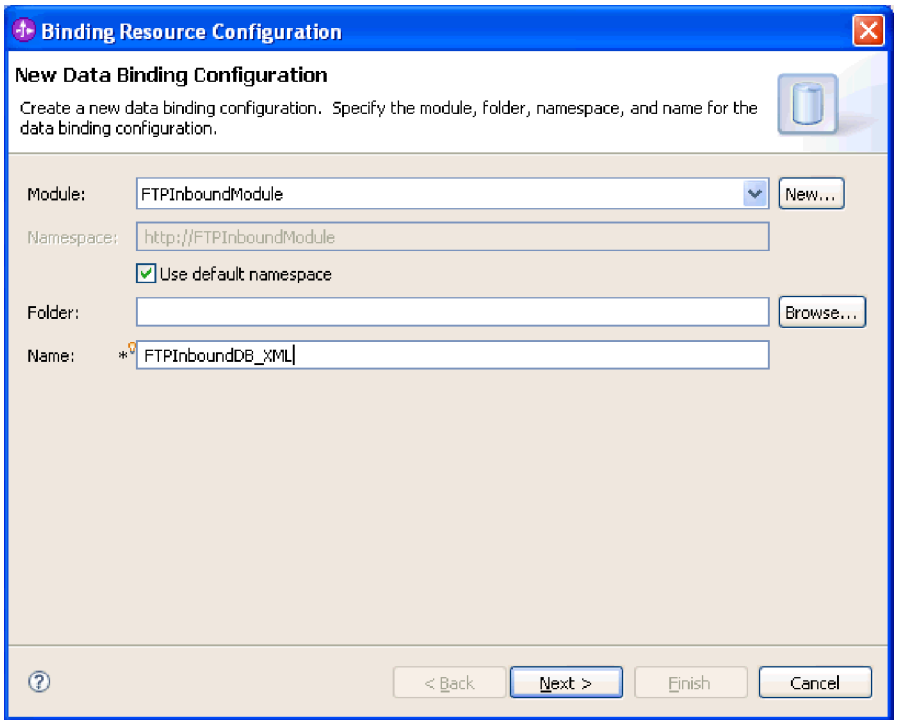


Figure 46. Naming the data binding

3. Click **Next**.
4. In the Select a configuration type window, leave the **Data binding** radio button selected.
5. Click **Browse** to select a class name. The term "class" here refers to the data binding class associated with the data binding you are creating for this module.
6. In the Data Binding Selection window, click the **Show data binding classes** radio button selected.
7. Select the correct data binding class for your data type and click **OK**.

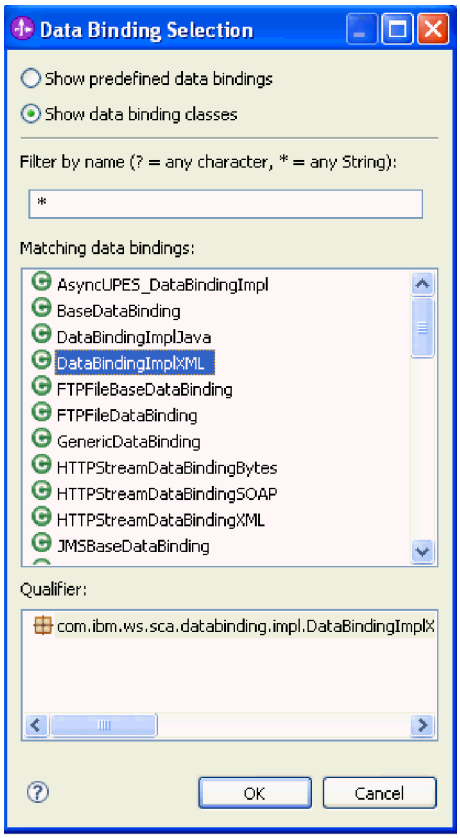


Figure 47. Selecting a data binding

The external service wizard will default to the correct data binding class for your data type. For more information on data bindings, see the topic about outbound data transformation in this documentation. The data binding class name will show on the Select a configuration type window.

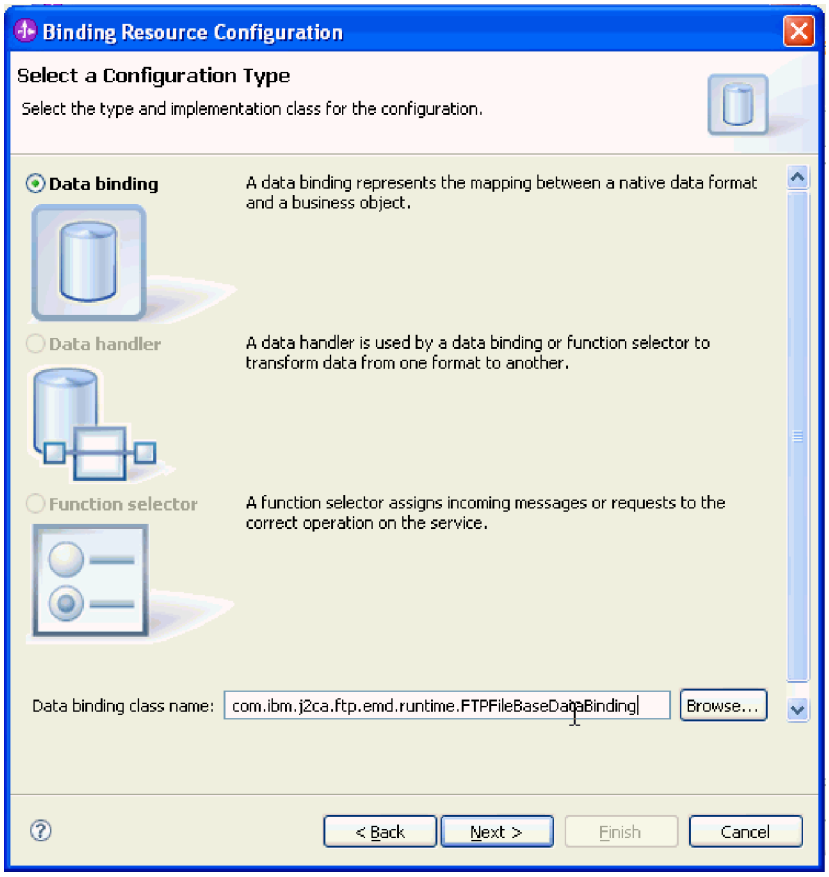


Figure 48. Data binding class is populated in the Select a Configuration Type window

8. Click **Next**.

Results

A data binding is configured for use with the module.

What to do next

From the current external service wizard window, you will proceed to select a data handler for the module.

Configuring data handlers

When you select a data type that contains business objects, you need to specify the data handler that will perform the conversions between a business object and a native format.

Before you begin

You must have created a data binding before specifying the data handlers for the module.

About this task

To specify data handlers, follow this procedure.

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 windows described in this documentation.

Procedure

1. Select **New** for the **DataHandlerConfigurationName** in the Data Binding Properties window.
2. In the **Name** field of the New Data Handler Configuration window, type a name for the data handler.
3. Click **Next**.
4. In the Select a configuration type window, leave the **Data Handler** radio button selected and click **Browse**.

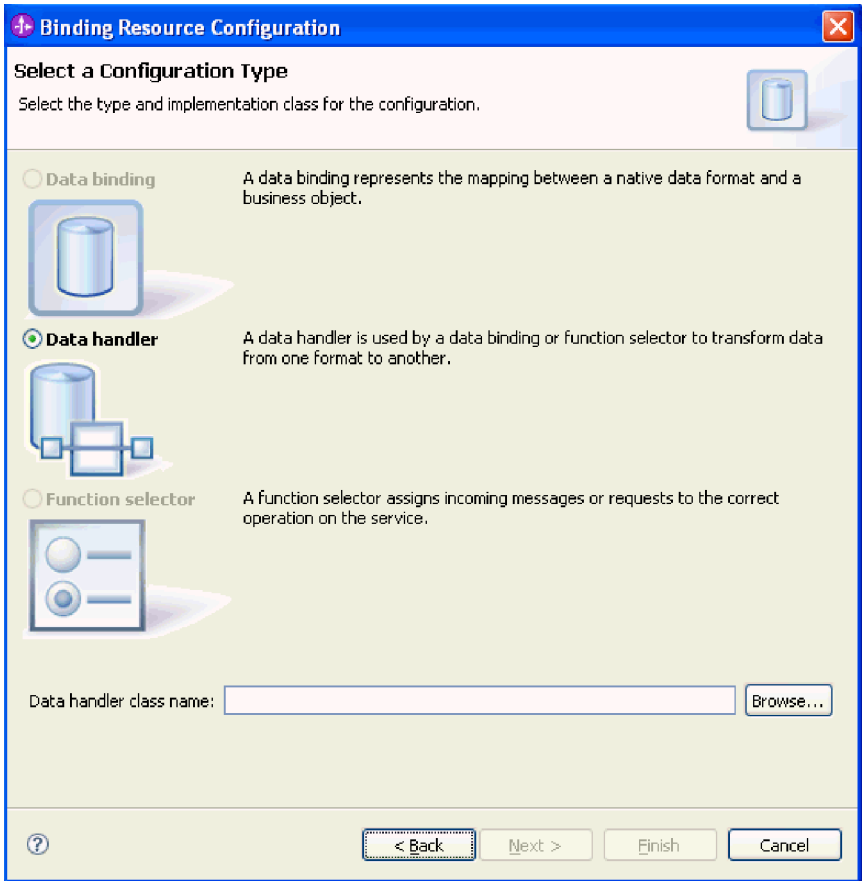


Figure 49. Choosing the data handler configuration type

5. In the Data handler selection window, select the correct data handler for the type of transformation required by your business objects and click **OK**.

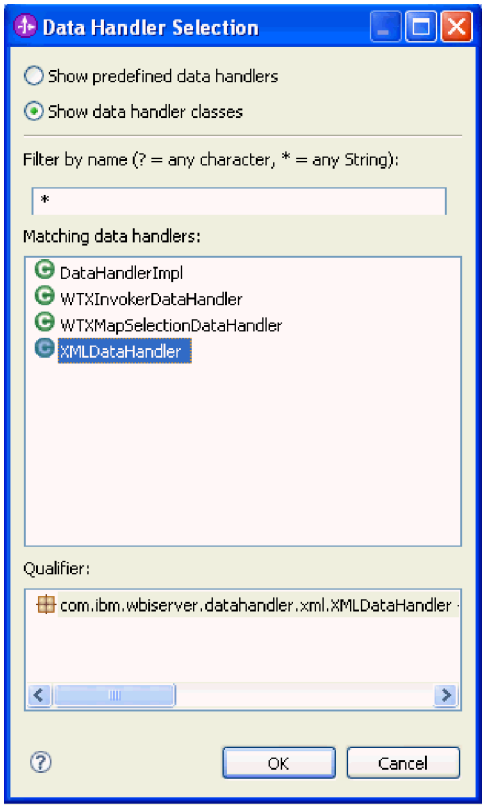


Figure 50. Selecting the data handler class

6. In the Select a configuration type window, the **Data handler class name field** is populated. Click **Next** to continue.
7. In the Specify Properties window, type a value for the **encoding** field and click **Finish**. This value indicates the type of character encoding the adapter will use during data transformation. For more information on the encoding property, see the topic about FTP business object properties in this documentation.
8. Click **Finish** in the Data Binding Properties window.
9. In the Operation window, select **New** for the **DataBinding type** field in the **Specify the details of the operation outnput** section of the window.
10. Type a **Name** for the data binding and click **Next**.

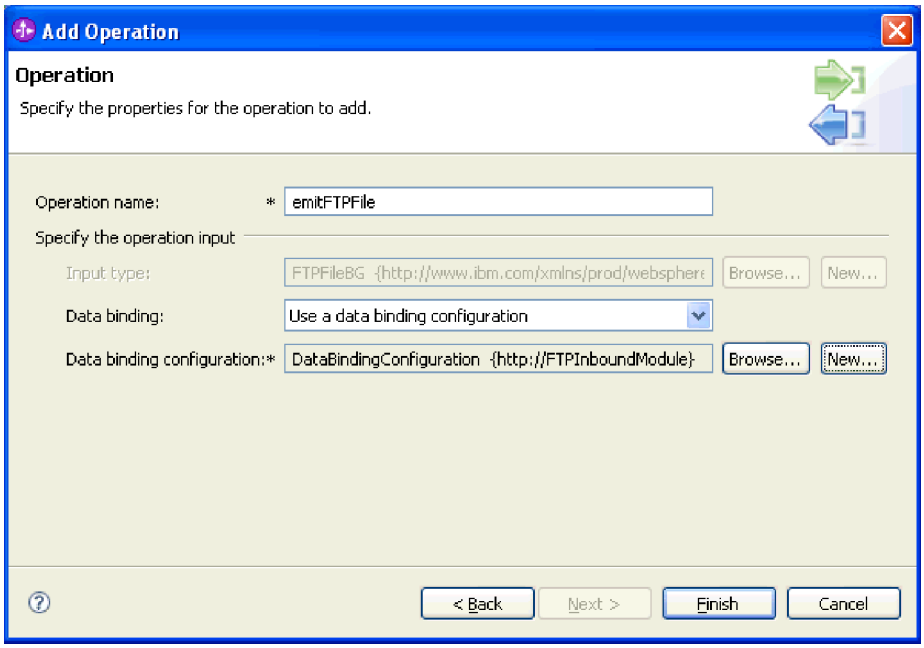


Figure 51. Naming the data binding

11. In the Select a configuration type window, leave the **Data binding** radio button selected.
12. Click **Finish**.
13. Click **Finish** on the Operation window.

Results

Data handlers are created.

What to do next

Continue in the wizard to specify interaction specification properties and generate artifacts for the module.

Generating the 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.

About this task

To generate artifacts, follow this procedure.

Procedure

1. Click **Next** in the Operations window.
2. In the Generate Service window, supply a name for the interface. This is the name that will display in the WebSphere Integration Developer assembly diagram.
3. Click **Finish**. The WebSphere Integration Developer assembly diagram opens and the interface you created is displayed.

Results

The WebSphere Integration Developer generates the artifacts and an export. 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 to the server.

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.

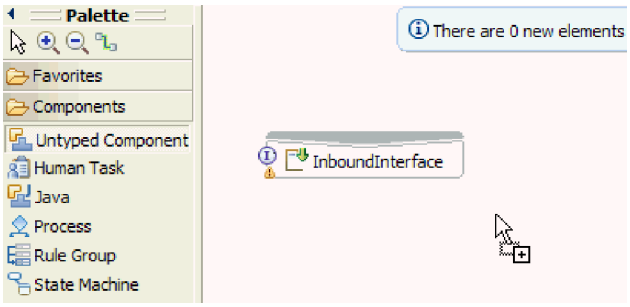


Figure 52. Adding a component to the Assembly Diagram

      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:

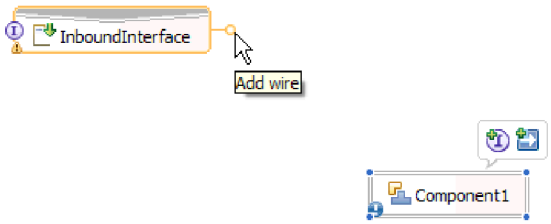


Figure 53. Selecting the wire icon

   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**.

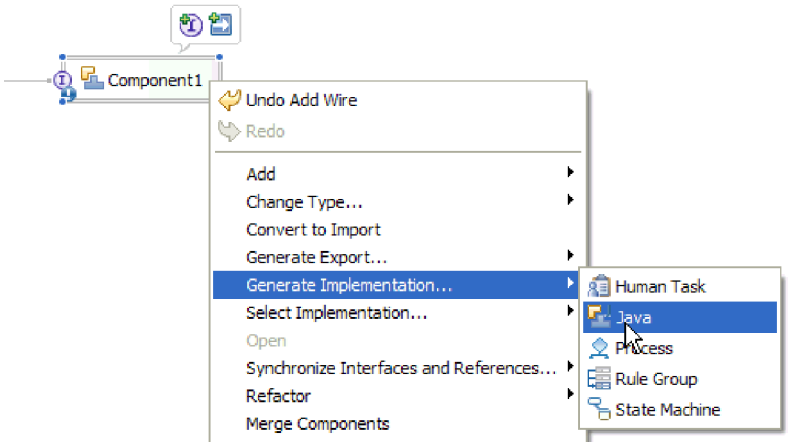


Figure 54. Generating a Java implementation

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**.

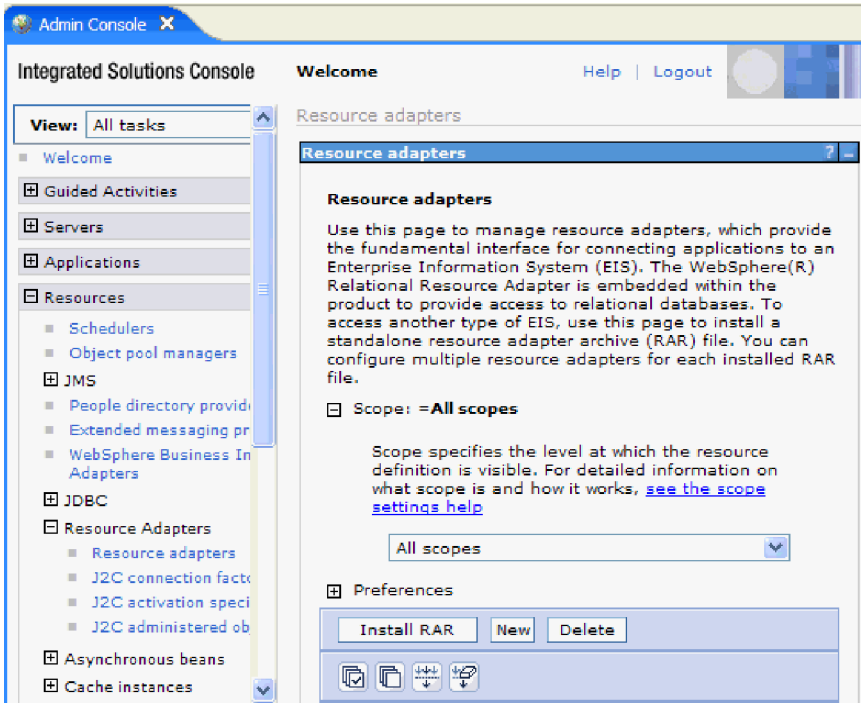


Figure 55. The Install RAR button on the Resource adapters page

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**.

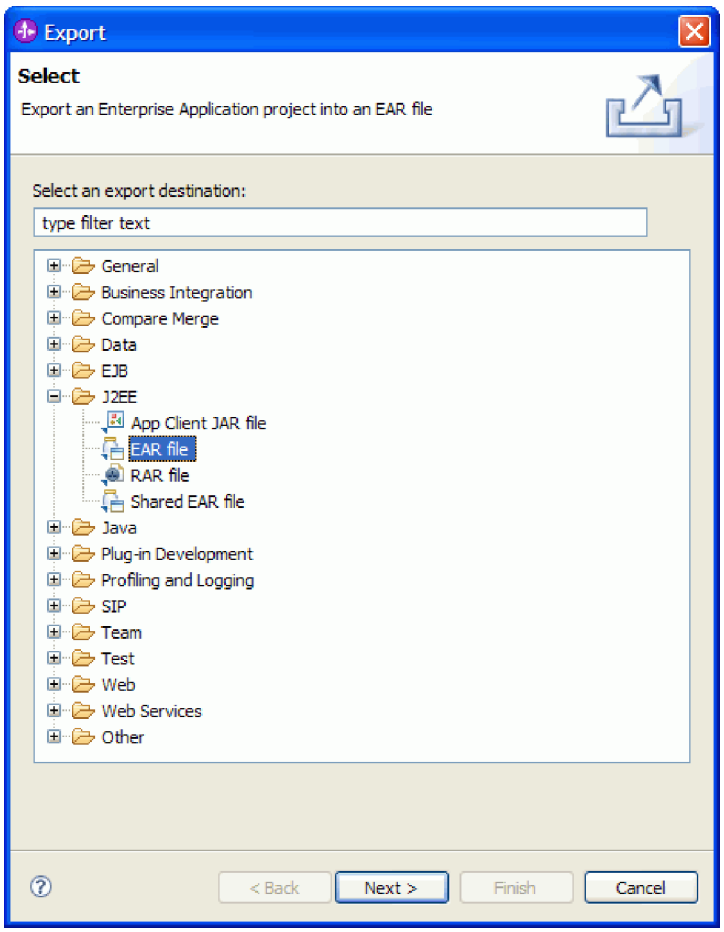


Figure 56. Selecting **EAR file** *from the Select window*

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**.

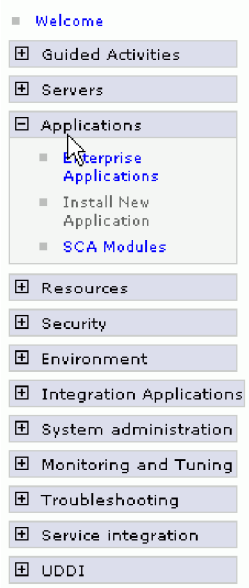


Figure 57. Preparing for the application installation window

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**.

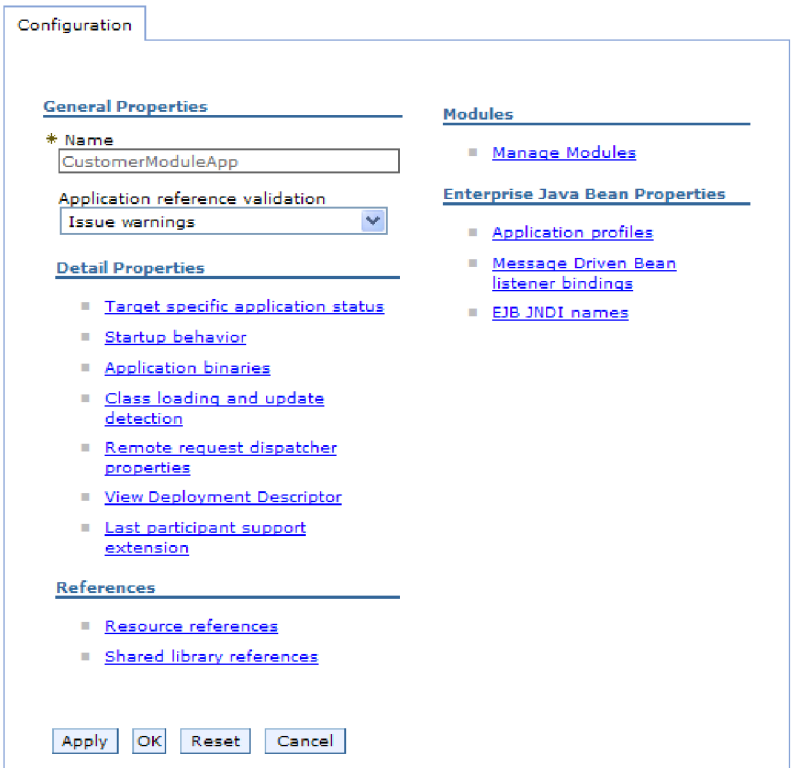


Figure 58. The Manage Modules selection in the Configuration tab

5. Click **IBM WebSphere Adapter for FTP**.
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 128 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:

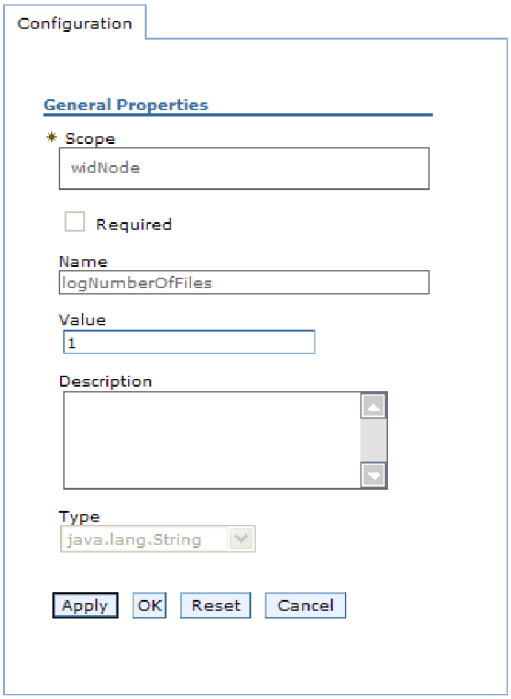


Figure 59. The Configuration tab for the logNumberOfFiles property

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 FTP server 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**.

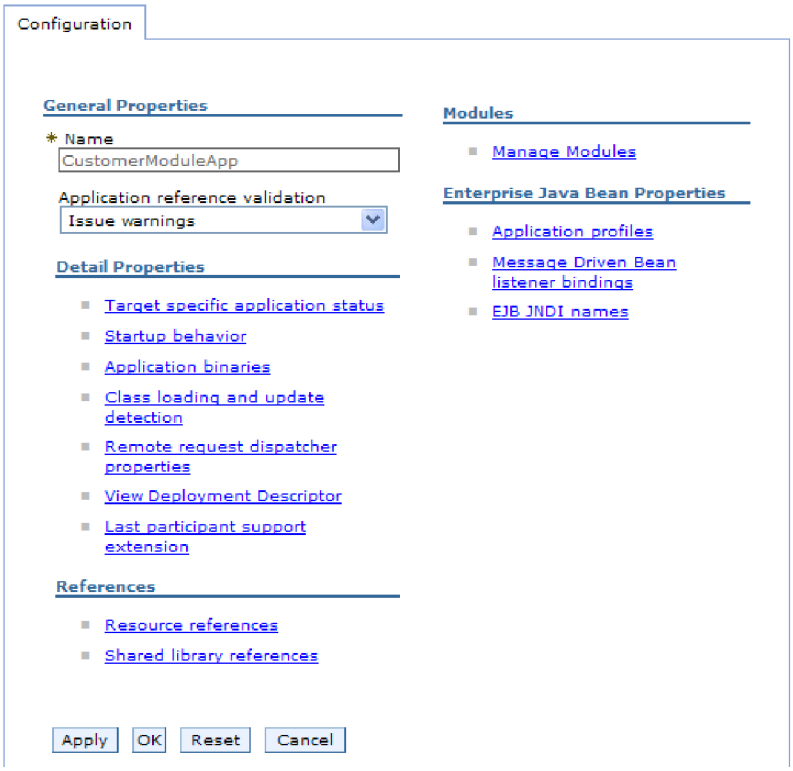


Figure 60. The Manage Modules selection in the Configuration tab

5. Click **IBM WebSphere Adapter for FTP**.
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 FTP. 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 (J2C) connection factory properties" on page 131 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**.

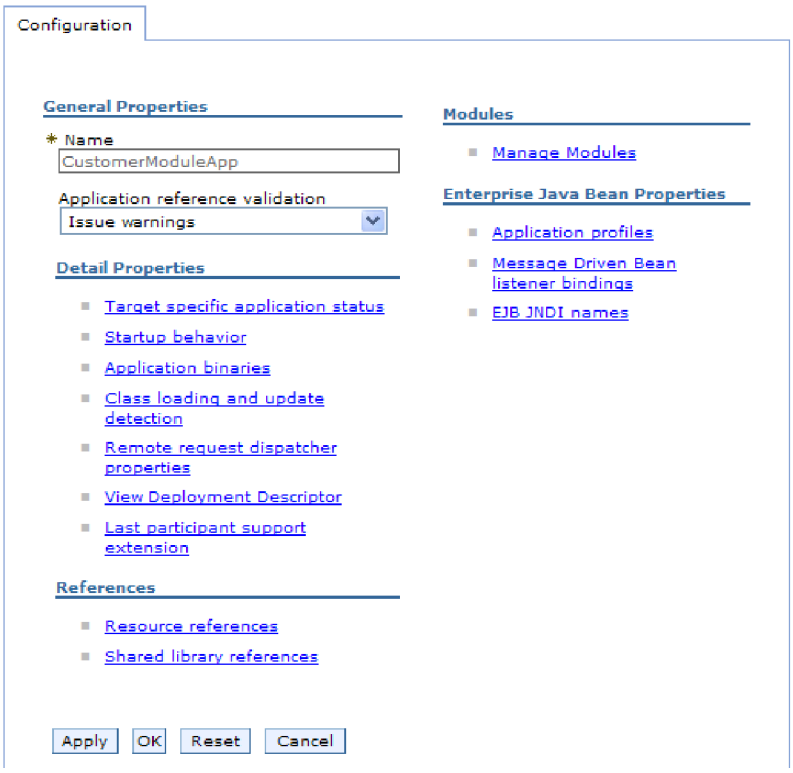


Figure 61. The Manage Modules selection in the Configuration tab

5. Click **IBM WebSphere Adapter for FTP**.
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 154 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 FTP**.
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 128 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:

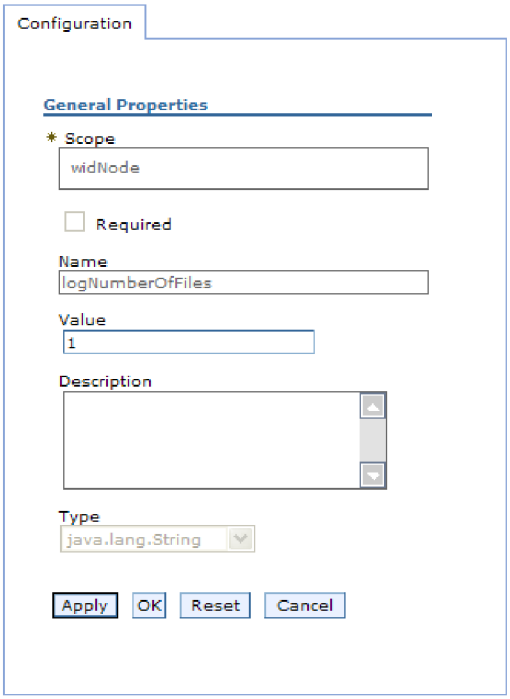


Figure 62. The Configuration tab for the logNumberOfFiles property

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 FTP server 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 FTP**.
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.

Figure 63. 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 FTP. 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 (J2C) connection factory properties" on page 131 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 FTP**.
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 154 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 FTP. 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 106.

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.

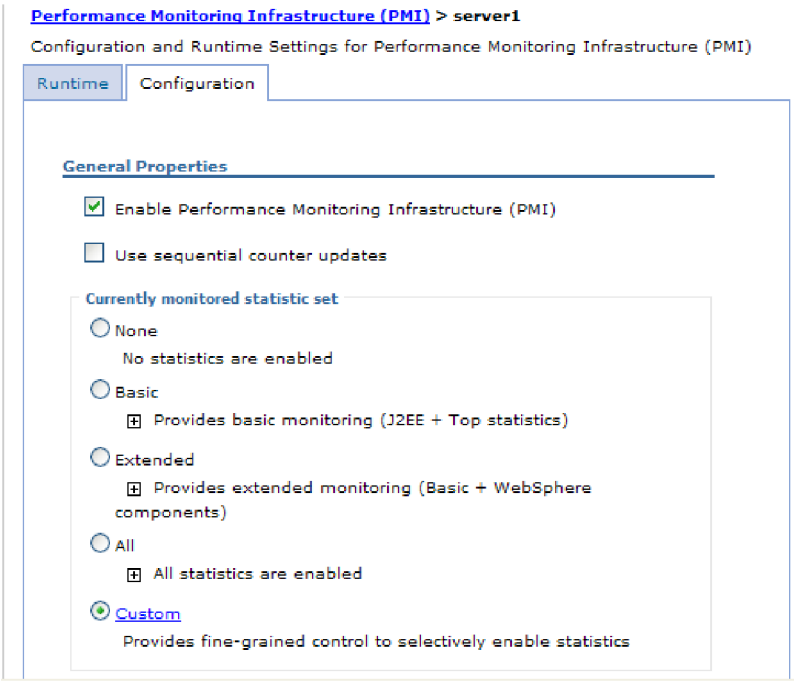


Figure 64. Enabling Performance Monitoring Infrastructure

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

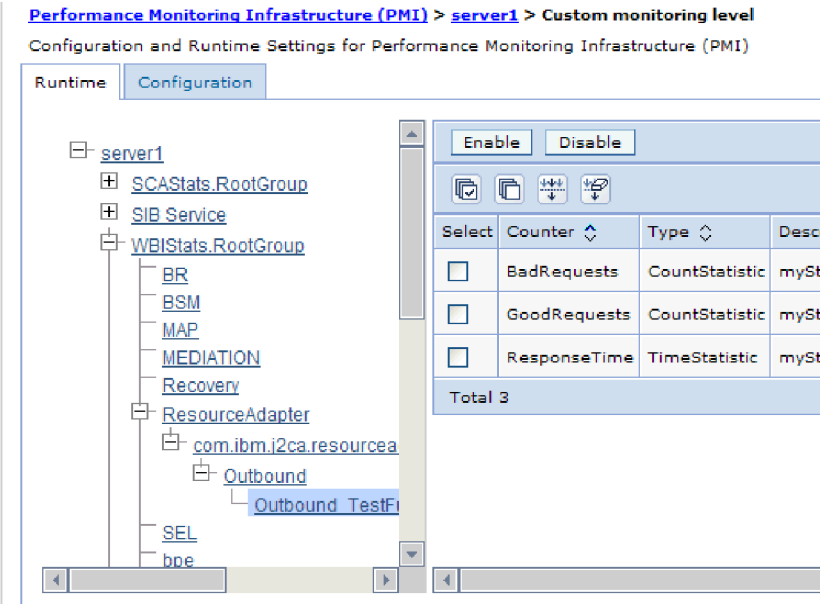


Figure 65. Runtime tab used for configuring PMI

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.

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.

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.

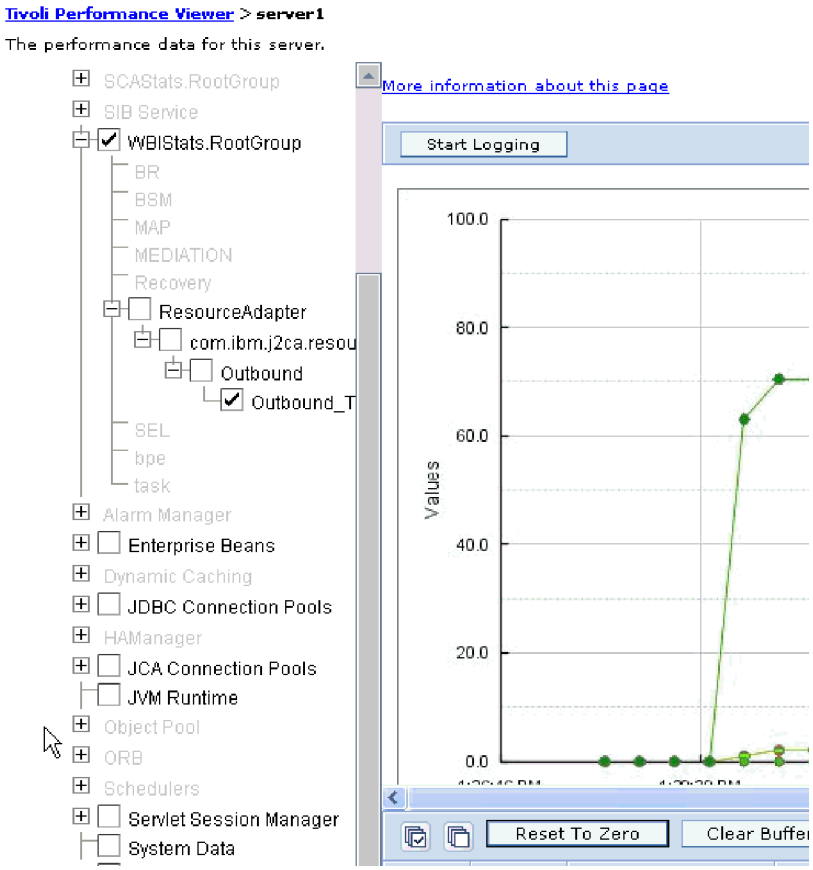


Figure 66. Adapter performance statistics, using graph view

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 FTP only, select the **com.ibm.j2ca.ftp** 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.

Logging Level	Description
Audit	A significant event has occurred that affects the server state or resources.
Info	The task is running. This logging level includes general information outlining the overall progress of a task.
Config	The status of a configuration is reported or a configuration change has occurred.
Detail	The subtask is running. This logging level includes general information detailing the progress of a subtask.

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

Results

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

Changing the log and trace file names

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

Before you begin

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

About this task

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

Log and trace files are in the *install_root*/profiles/*profile_name*/logs/*server_name* folder.

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

Procedure

1. In the navigation pane of the administrative console, select **Applications > Enterprise Applications**.
2. In the Enterprise Applications list, click the name of the adapter application. This is the name of the EAR file for the adapter, but without the .ear file extension. For example, if the EAR file is named Accounting_OutboundApp.ear, then click **Accounting_OutboundApp**.
3. In the Configuration tab, in the Modules list, click **Manage Modules**.
4. In the list of modules, click IBM WebSphere Adapter for FTP.
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 67.

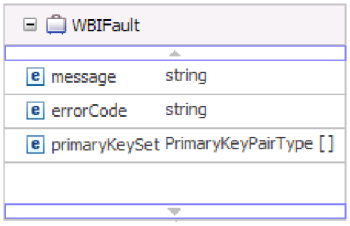


Figure 67. The structure of the WBIFault business object

The wizard creates the following fault business objects:

- DuplicateRecordFault

  The adapter throws this fault when processing an outbound Create operation when an error occurs because the file specified already exists in the specified directory path.
- RecordNotFoundFault

  The adapter throws this fault when processing the Append, Delete, Overwrite, Retrieve, ExecuteFTPScript, and ServerToServerFileTransfer operations when the file directory path or script file does not exist in the specified directory path.
- MissingDataFault

  The adapter throws this fault when required values are not provided, such as when the file content is null or the file name or directory path is empty. During a Retrieve operation, the adapter throws this fault when an error occurs because the delimiter is null or not valid.

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

Table 6. 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
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 6 on page 112.

   The following .import file shows the DUPLICATE_RECORD, MISSING_DATA, 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.ftpfile.emd.runtime.FTPFileBaseDataBinding"
  faultSelector="com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl">
   <resourceAdapter name="FTPOutApp.IBM WebSphere Adapter for FTP Files"
                    type="com.ibm.j2ca.ftpfile.FTPFileResourceAdapter">
      <properties/>
   </resourceAdapter>
   <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"/>
   <faultBinding
      fault="MISSING_DATA"
      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 6 on page 112.

     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.ftp.customerbg.CustomerBGDataBinding"
 method="createCUSTOMER"
 outDataBindingType="com.ibm.xmlns.prod.wbi.j2ca.ftp.customerbg.CustomerBGDataBinding"
 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/ftpfile/customerbg"
  xmlns:intf="http://FTPOut/FTPFileOutboundInterface"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
Step 3a
  xmlns:fault="http://com/ibm/j2ca/fault/afcfault"
    targetNamespace="http://FTPOut/FTPFileOutboundInterface">
  <types>
  <xsd:schema
     xmlns:tns="http://FTPOut/FTPFileOutboundInterface"
     xmlns:xsd1="http://www.ibm.com/xmlns/prod/wbi/j2ca/ftpfile/customerbg"
     elementFormDefault="qualified"
     targetNamespace="http://FTPOut/FTPFileOutboundInterface"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:import
     namespace="http://www.ibm.com/xmlns/prod/wbi/j2ca/ftpfile/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
        <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 114

```
<message name="duplicateRecordFault">
  <part element="intf:duplicateRecordFaultX"
        name="duplicateRecordFaultPart"/>
</message>
<message name="recordNotFoundFault">
  <part element="intf:recordNotFoundFaultX"
        name="recordNotFoundFaultPart"/>
</message>
<operation name="createCUSTOMER">
<input message="intf:createCUSTOMERRequest" name="createCUSTOMERRequest"/>
<output message="intf:createCUSTOMERResponse" name="createCUSTOMERResponse"/>
```

Step 3e on page 114

```
<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 FTP, 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 FTP 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 FTP server. The name typically reflects the operation to be performed and the structure of the business object.

Business object structure

The adapter supports three different types of business object structures. A generic business object, which is used to pass unstructured data. A generic business object with a business graph, which contains the action to be performed on the data and the connection-specific information. A user-defined type, which is a content-specific business object that supports very specific business object structures (such as customer and order business objects).

Business graphs are optional and can be selected in the external service wizard.

The FTPFileBG, FTPFile and UnstructuredContent generic business object definitions are automatically generated. Depending on the custom complex types selected when you create external services, the corresponding business object or objects' definitions will also be generated. For example, if we select Customer, including the optional business graph, the CustomerWrapperBG and CustomerWrapper business objects will be generated.

FTPFileBG

The FTPFileBG business object is a generic business object that contains the verb (the action to be performed on the data) and the FTPFile business object as a child. The following graphic illustrates this relationship.

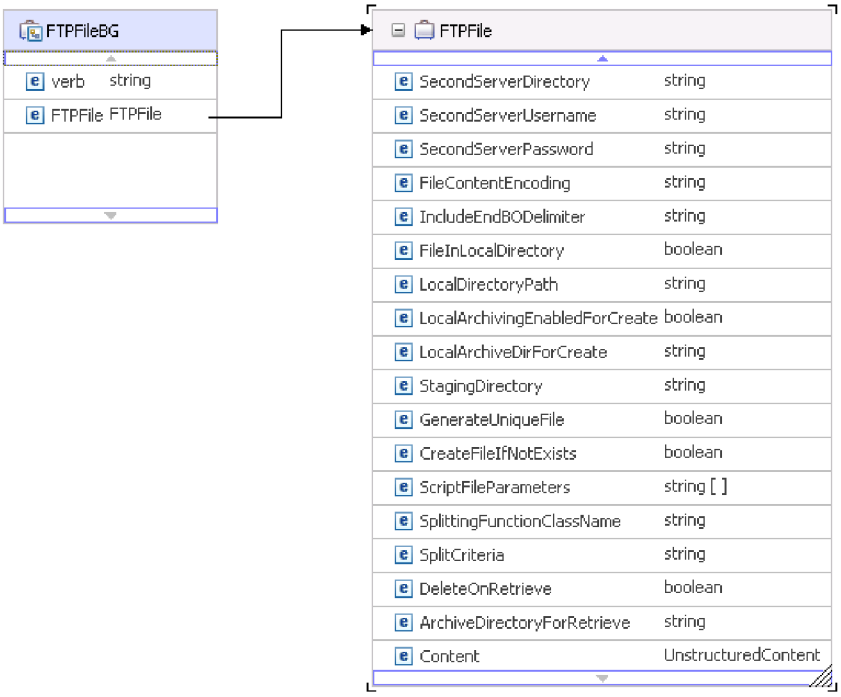


Figure 68. FTPFileBG business object

FTPFile

The FTPFile business object contains all necessary connection information, and an UnstructuredContent business object as a child. The following graphic illustrates this relationship.

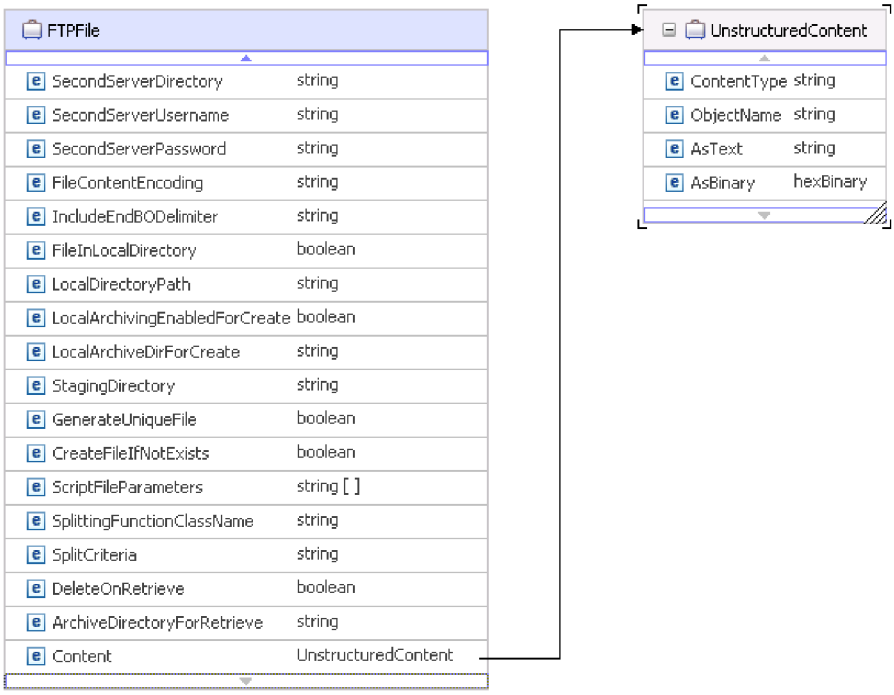


Figure 69. FTPFile business object

CustomerWrapperBG

The CustomerWrapperBG is a business object that contains the verb (the action to be performed on the data) and the CustomerWrapper business object as a child. The following graphic illustrates this relationship.

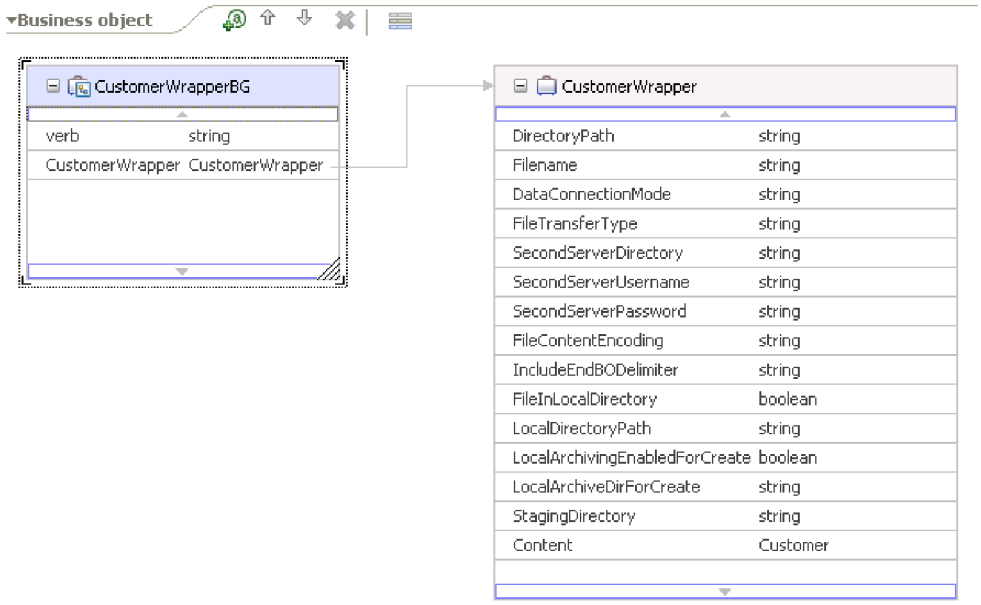


Figure 70. CustomerWrapperBG business object

CustomerWrapper

The CustomerWrapper business object is a business object that contains all necessary connection information and the content-specific Customer business object as a child. The following graphic illustrates this relationship.

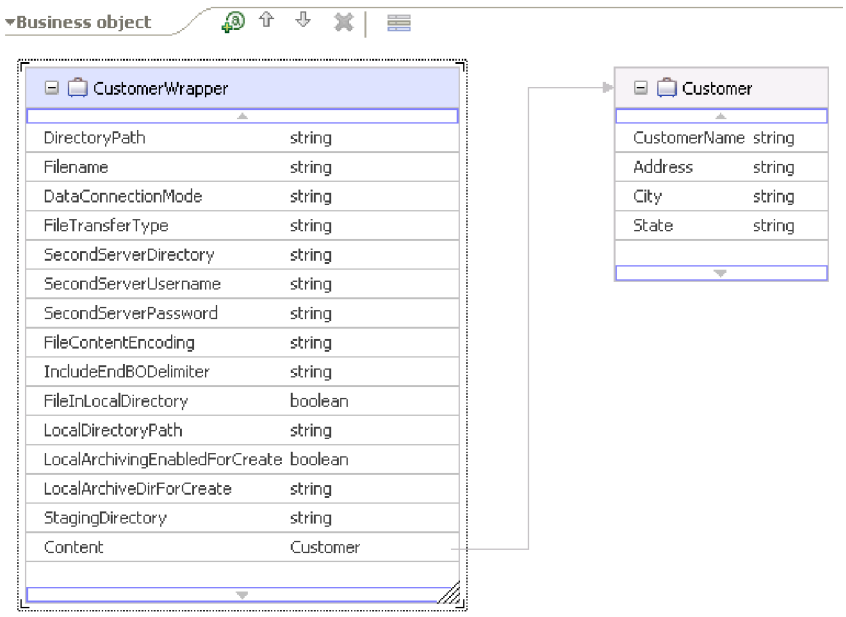


Figure 71. CustomerWrapper business object

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 FTP server that it uses to build the business object. Use the Business Object Editor to create user defined objects.

External service wizard 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 FTP server 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.

Note: Business graph generation is optional and is supported for WebSphere Process Server only.

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 and how the adapter interprets them.

Table 7. Business object attribute properties

Property	Description
Cardinality	For simple attributes, 1 is used. For container attributes, depending on the method requirements, `n` is used.
Foreign Key	The adapter does not have any specific elements representing Foreign Keys.
Key	The adapter does not have any specific elements representing a Key.
Name	This property represents the unique name of the attribute, if it is a simple attribute, or the name of the business object, if it is a child business object.
Required	This property specifies whether an attribute must contain a value.
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 the name of another business object.

Business object operation support

An operation is the name of the action that is performed on the business object by the adapter. Every business object has an operation associated with it. The name of the operation typically indicates the type of action that is taken on the business object.

The following table defines the operations that the adapter supports.

Table 8. Supported operations of business objects

Operation	Result
Create	Creates a file with the specified file name in the specified directory with the content sent across in the request.
Append	Appends the content in the request to the end of the file.
Retrieve	Returns the content of the file specified in the request.
Delete	Deletes the file from the directory specified in the request.
Overwrite	Overwrites the file in the directory with the content specified in the request.
Exists	Returns a successful response if the file in the request exists in the specified directory or sub directories.
List	Returns all the file names in the specified directory.
ServerToServerFileTransfer	Transfers the file from one FTP server to another FTP server.
ExecuteFTPScript	Runs an FTP script file in the specified directory.

Custom business objects

If you use custom business objects, you must create predefined business objects using WebSphere Integration Developer business object wizard before running the external service wizard. The business object definitions created by the wizard are stored as xsd files on your local system. When the external service wizard creates

business objects, it looks for the predefined business objects created in the business object wizard and populates them with data specific to the module.

For more information on how to create predefined business objects, refer to the WebSphere Integration Developer documentation.

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 FTP 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 FTP 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.
Property type	Describes the property type. Valid property types include the following: • Boolean • String • Integer

Row	Explanation
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**.

Adapter type properties

Adapter type properties provide the external service wizard with the adapter details. These properties are configured using the external service wizard before deployment or with the WebSphere Application Server administrative console after deployment.

Note: If you set any of these adapter type properties using bidirectional script, you must set values that identify the format of the bidirectional script entered for that property.

The adapter type properties and their purpose are described in the following table. A complete 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 understanding property details.

Table 9. Adapter type properties

Property name		
In the wizard	In the administrative console	Description
"Description property (Description)" on page 127	`Description`	Adapter description.
"Display Name property (DisplayName)" on page 127	`DisplayName`	Adapter display name.

Table 9. Adapter type properties (continued)

"ID property (ID)"	`ID`	ID for the adapter type.
"Vendor property (Vendor)"	`Vendor`	Name of the vendor providing the adapter.
"Version property (Version)"	`Version`	Adapter version.

Description property (Description)

Adapter description.

Table 10. Description property characteristics

Required	Yes
Default	`IBM WebSphere Adapter for FTP`
Property type	String

Display Name property (DisplayName)

Adapter display name.

Table 11. DisplayName property characteristics

Required	Yes
Default	`IBM WebSphere Adapter for FTP`
Property type	String

ID property (ID)

ID for the adapter type.

Table 12. ID property characteristics

Required	Yes
Default	`FTP`
Property type	String

Vendor property (Vendor)

Name of the vendor providing the adapter.

Table 13. Vendor property characteristics

Required	Yes
Default	`IBM`
Property type	String

Version property (Version)

Adapter version.

Table 14. Version property characteristics

Required	Yes
Default	`6.1.0`
Property type	String

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, but are supported 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 details tables in the sections that follow, see Guide to understanding property details.

Table 15. Resource adapter properties for the Adapter for FTP

Property 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.
"EISEncoding (EISEncoding)" on page 129	`EISEncoding`	Encoding of the FTP Server.
(Not available)	enableHASupport	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 16. Adapter ID to use for logging and tracing details

Required	Yes

Table 16. Adapter ID to use for logging and tracing details (continued)

Default	CWYFT_FTPFile
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

EISEncoding (EISEncoding)

This property specifies the encoding of the FTP Server. Sets the encoding for the control connection while communicating with the FTP Server. Set the property if the FTP server's directories or file names contain globalized characters.

Table 17. EISEncoding characteristics

Required	No
Default	`None`
Property type	String
Examples	`UTF-8, ISO-8859-1`

Enable high availability support (enableHASupport)

Do not change this property. It must be set to `true`.

Log file maximum size (LogFileMaxSize)

This property specifies the size of the log files in kilobytes.

Table 18. Log file maximum size details

Required	No
Default	`0`
Property type	Integer
Usage	When the log file reaches it maximum size, the adapter start using a new log file. If the file size is specified as `0` or no maximum size is specified, the file does not have a maximum size.
Globalized	Yes
Bidi supported	No

Log file name (LogFilename)

This property specifies the full path name of the log file.

Table 19. Log file name details

Required	No
Default	No default value
Property type	String

Table 19. Log file name details (continued)

Usage	This property is deprecated.
Globalized	Yes
Bidi supported	Yes

Log number of files (LogNumberOfFiles)

This property specifies the number of log files.

Table 20. Log number of files details

Required	No
Default	1
Property type	Integer
Usage	When a log file reaches its maximum size, the adapter starts using another log file. If no value is specified, the adapter creates a single log file.
Globalized	Yes
Bidi supported	No

Trace file maximum size (TraceFileMaxSize)

This property specifies the size of the trace files in kilobytes.

Table 21. Trace file maximum size details

Required	No
Default	`0`
Property type	Integer
Usage	If no value is specified, then the trace file has no maximum size.
Globalized	Yes
Bidi supported	No

Trace file name (TraceFilename)

This property specifies the full path of the trace file.

Table 22. Trace file name details

Required	No
Default	No default value
Unit of measure	Kilobytes
Property type	String
Usage	This property is deprecated.
Globalized	Yes
Bidi supported	Yes

Trace number of files (TraceNumberOfFiles)

This property specifies the number of trace files to use. When a trace file reaches its maximum size, the adapter starts using another trace file.

Table 23. Trace number of files details

Required	No
Default	1
Property type	Integer
Usage	If no value is specified, the adapter uses a single trace file.
Globalized	Yes
Bidi supported	No

Managed (J2C) connection factory properties

Managed connection factory properties are used by the adapter at run time to create an outbound connection instance with the FTP server.

You set the managed connection factory properties using the external service wizard and can change them using the WebSphere Integration Developer Assembly Editor, or after deployment through the WebSphere Process Server administrative console.

The following table lists the managed connection factory properties. A complete 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 understanding property details.

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 24. Managed connection factory properties

Property name		Description
In the wizard	In the administrative console	
"Custom parser class name property (CustomParserClassName)" on page 132	`CustomParserClassName`	Specifies the fully qualified class name of the custom parser that is used to parse the `ls -l` output.
"Default target file name property (Filename)" on page 133	`Filename`	Specifies the name of the file to be used during outbound Create operations.
"Directory property (OutputDirectory)" on page 133	`OutputDirectory`	Specifies the output directory in the FTP Server.
"Encoding used by FTP server property (EISEncoding)" on page 133	`EISEncoding`	Specifies the encoding of the FTP server.
"Host name property (HostName)" on page 134	`HostName`	Specifies the hostname of the FTP Server.

Table 24. Managed connection factory properties (continued)

"Host name property (SecondServerHostName)" on page 134	HostName	Specifies the hostname of the second FTP Server.
"Host name property (SocksProxyHost)" on page 134	SocksProxyHost	Specifies the name of the workstation that is used as a proxy server.
"Password property (Password)" on page 134	Password	Specifies the password of the user with privileges to connect to the FTP server and perform FTP operations.
"Password property (SecondServerPassword)" on page 135	SecondServerPassword	Specifies the password of the Second FTP server to which the file is transferred during a server to server file transfer outbound operation.
"Password property (SocksProxyPassword)" on page 135	SocksProxyPassword	Specifies the password used to authenticate the proxy server.
"Port number property (PortNumber)" on page 135	PortNumber	Specifies the port number of the FTP server.
"Port number property (SecondServerPortNumber)" on page 135	SecondServerPortNumber	Specifies the port number of the second FTP server.
"Port number property (SocksProxyPort)" on page 136	SocksProxyPort	Specifies the port number of the proxy server.
"Protocol property (Protocol)" on page 136	Protocol	Specifies either a normal or secure FTP connection.
"Protocol property (SecondServerProtocol)" on page 136	SecondServerProtocol	Specifies either a normal or secure FTP connection for the second server.
"Second Server Directory property (SecondServerDirectory)" on page 137	SecondServerDirectory	Specifies the directory path of the second FTP server to which the ServerToServerFileTransfer outbound operation is performed.
"Sequence file property (FileSequenceLog)" on page 137	FileSequenceLog	Specifies the full path of the file where the sequence number will be stored for outbound Create processing.
"Staging directory property (StagingDirectory)" on page 138	StagingDirectory	Specifies the directory that the file is first created in to.
"User name property (SecondServerUserName)" on page 138	SecondServerUserName	Specifies the user name of the second FTP server to which the file is transferred during a server to server file transfer outbound operation.
"User Name property (SocksProxyUserName)" on page 138	SocksProxyUserName	Specifies the user name used to authenticate the proxy server.
"User name property (Username)" on page 138	Username	Specifies the name of the user.

Custom parser class name property (CustomParserClassName)

Fully qualified class name of the custom parser that is used to parse the `ls –l` output. Only used when the `ls –l` output deviates from standard output.

Table 25. Custom parser class name property characteristics

Required	No
Default	None
Property type	String
Globalized	No

Default target file name property (Filename)

Name of the file to be used during outbound Create operations. This value, combined with FtpUrl, determines the sequence.

For example:

FtpUrl = `ftp://localhost:21/out` and Filename = `Customer.txt`, the output files are Customer.1.txt, Customer.2.txt, Customer.3.txt and so on.

Table 26. Default target file name property characteristics

Required	Yes
Default	`Yes`
Property type	String
Globalized	No

Directory property (OutputDirectory)

Output directory in the FTP Server that the outbound operation is performed on.

Table 27. Directory property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Encoding used by FTP server property (EISEncoding)

Encoding of the FTP server. Use this value to set the encoding for the control connection to the FTP server.

- When both EISEncoding at the adapter level and EISEncoding at the MCF level are not set (both are null), nothing is set on the control connection while communicating with the FTP server.
- When EISEncoding at the adapter level is set and EISEncoding at the MCF level is not set, the value at adapter level is set on the control connection while communicating with the FTP server. This is helpful when using multiple MCF's and the same encoding is set. In this case, set the value at the adapter level so that all the connections will have the same encoding for the control connection.
- When EISEncoding at the adapter level is not set and EISEncoding at the MCF level is set, the value at MCF level is set on the control connection while communicating with the FTP server. Since the value is at the MCF level, this is applicable for only that MCF.

- When both EISEncoding at the adapter level and EISEncoding at the MCF level are set, the value at the MCF level takes precedence.

Specify any Java-supported encoding set for this attribute.

Table 28. Encoding used by FTP server property characteristics

Required	No
Default	None
Property type	String
Globalized	No

Host name property (HostName)

Hostname of the FTP Server to which the connection is established during an outbound operation.

Table 29. Host name property characteristics

Required	Yes
Default	None
Property type	String
Globalized	Yes

Host name property (SecondServerHostName)

Hostname of the second FTP Server to which the connection is established during an outbound operation

Table 30. Host name property characteristics

Required	Yes
Default	None
Property type	String
Globalized	Yes

Host name property (SocksProxyHost)

Host name of the workstation that is used as a proxy server through which the adapter requests are routed to the FTP server.

Table 31. Host name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Password property (Password)

Password of the user with privileges to connect to the FTP server and perform FTP operations. You do not need to specify a value for this attribute if the Password is included in the URL specified in the FtpUrl property.

Table 32. Password property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Password property (SecondServerPassword)

Password of the Second FTP server to which the file is transferred during a server to server file transfer outbound operation.

Table 33. Password property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Password property (SocksProxyPassword)

Password used to authenticate the proxy server.

Table 34. Password property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Port number property (PortNumber)

Port number of the FTP server through which the connection is established during an outbound operation.

Table 35. Port number property characteristics

Required	Yes
Default	21
Property type	Integer
Globalized	No

Port number property (SecondServerPortNumber)

Port number of the second FTP server through which the connection is established during an outbound operation.

Table 36. Port number property characteristics

Required	Yes
Default	21
Property type	Integer

Table 36. Port number property characteristics (continued)

Globalized	No

Port number property (SocksProxyPort)

Port number of the proxy server through which the adapter requests are routed to the FTP server.

Table 37. Port number property characteristics

Required	No
Default	`1080`
Property type	Integer
Globalized	No

Protocol property (Protocol)

Protocol that determines whether the connection to be established is a normal FTP connection or a secure FTP connection.

For example:

Normal connection: `FTP`

Secure connection: `FTPS`

Table 38. Protocol property characteristics

Required	Yes
Default	`FTP`
Property type	String
Globalized	No

Protocol property (SecondServerProtocol)

Protocol that determines whether the second connection to be established is a normal FTP connection or a secure FTP connection.

For example:

Normal connection: `FTP`

Secure connection: `FTPS`

Table 39. Protocol property characteristics

Required	Yes
Default	`FTP`
Property type	String
Globalized	No

Second Server Directory property (SecondServerDirectory)

Directory path of the second FTP server to which the ServerToServerFileTransfer outbound operation is performed.

Syntax for specifying the directory path is: ftp://
[UserId:password@]FTPserver[:port]Directory
ForSecondServer

The following information can also be specified:

- The username and password of a user with privileges to connect to the second FTP server and perform FTP operations. If not specified here, must be specified in the SecondServerUsername and SecondServerPassword properties.
- The FTP port. If not specified here, the adapter uses the default FTP port.
- The remote event directory. If not specified here, the adapter transfers the file to the directory to which the connection is established in the FTP server.

Table 40. Second Server Directory property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Sequence file property (FileSequenceLog)

Specifies the full path of the file where the sequence number will be stored for outbound Create processing.

When the FileSequenceLog property is specified, the adapter generates a unique sequence of numbers to insert into the file names when processing the Create operation.

The sequence of numbers will continue to increment after multiple adapter restarts.

The sequence number is inserted into the file name in the following format:

```
filename.number.extension
```

For example `Customer.3.txt`

When the FileSequenceLog property is not specified or contains an invalid value, no sequence number is generated.

Table 41. Sequence file property characteristics

Required	No
Default	None
Property type	String
Globalized	No

Staging directory property (StagingDirectory)

The directory that the file is first created in to during an outbound create operation. After creation, the file is moved to the directory specified in the DirectoryPath property. The staging directory is also used for Append and Overwrite operations where the specified file is copied to StagingDirectory (if present), then appended or overwritten with content and moved back to the original specified directory. If StagingDirectory is not present, the operation is run in the actual required directory. The benefit of working with a staging directory is to avoid file writing conflicts, which can occur when people are reading the file while it is still being written during append and updates.

Table 42. Staging directory property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

User name property (SecondServerUserName)

User name of the second FTP server to which the file is transferred during a server to server file transfer outbound operation.

Table 43. User name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

User Name property (SocksProxyUserName)

User name used to authenticate the proxy server.

Table 44. User Name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

User name property (Username)

Name of the user with privileges to connect to the FTP server and perform FTP operations. You do not need to specify a value for this attribute if the Username is included in the URL specified in the FtpUrl property.

Table 45. User name property characteristics

Required	No
Default	None
Property type	String

Table 45. User name property characteristics (continued)

Globalized	Yes

Interaction specification properties

Interaction specification properties control the interaction for an operation. The external service wizard sets the interaction specification properties when you configure the adapter. You can change some, but not all, of these properties. However, some properties for outbound operations can be changed by the user. Use the assembly editor to change these properties, which 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 details tables in the sections that follow, see Guide to understanding property details.

Table 46. Interaction specification properties

Property name		Description
In the wizard	In the administrative console	
"Remote archive directory for retrieve operation property (ArchiveDirectoryForRetrieve)" on page 141	`ArchiveDirectory ForRetrieve`	The adapter optionally archives the file to this folder before it is deleted during a Retrieve operation.
"Create new file if the file does not exist property (CreateFileIfNotExists)" on page 141	`CreateFileIfNotExists`	If the file does not exist on the FTP server, the adapter creates the file when this property is set to `true` during Append and Overwrite operations.
"FTP server connection mode property (DataConnectionMode)" on page 142	`DataConnectionMode`	Data connection mode used by the FTP server during file transfers.
(Not available)	`DeleteOnRetrive`	The adapter deletes the file from the FTP Server after it is retrieved when this property is set to `true`.
"Remote directory on FTP system property (DirectoryPath)" on page 142	`DirectoryPath`	Absolute path of the directory on the FTP server where the outbound operation needs to be performed.
"File content encoding property (FileContentEncoding)" on page 143	`FileContentEncoding`	Encoding used while writing to the file.
"File in local directory property (FileInLocalDirectory)" on page 143	`FileInLocalDirectory`	If set to `true` during a create operation, the file content is picked from the local directory path of the adapter machine.
"Default target file name property (Filename)" on page 143	`Filename`	Name of the file in the directory provided by the DirectoryPath property.
"File transfer type property (FileTransferType)" on page 143	`FileTransferType`	File transfer type used during outbound operations.
(Not available)	`GenerateUniqueFile`	The adapter creates a unique file name when this property is set to `true`.

Table 46. Interaction specification properties (continued)

"Host name property (SecondServerHostName)" on page 144	SecondServerHostName	Hostname of the second FTP server.
"Include business object delimiter in the file content property (IncludeEndBODelimiter)" on page 144	IncludeEndBODelimiter	File content is appended with this value.
"Local archive directory for create operation property (LocalArchiveDirForCreate)" on page 145	LocalArchiveDirForCreate	When LocalArchivingEnabledForCreateis set to true during a create operation, the file is saved to the local workstation in this directory.
"Archive file in the local directory for create operation property" on page 145	LocalArchivingEnabled ForCreate	When set to true, the file is saved to the local workstation during a create operation.
"Local directory property (LocalDirectoryPath)" on page 145	LocalDirectoryPath	The file is picked from this directory.
"Port number property (SecondServerPortNumber)" on page 145	SecondServerPortNumber	Port number of the second FTP server.
"Protocol property (SecondServerProtocol)" on page 146	SecondServerProtocol	Determines whether the connection is normal FTP or secure FTP.
"Script File Parameters property (ScriptFileParameters)" on page 146	ScriptFileParameters	The parameters required by the FTP script file.
"Directory property (SecondServerDirectory)" on page 141	SecondServerDirectory	Directory path of the second FTP server during a ServerToServerFileTransfer operation.
"Password property (SecondServerPassword)" on page 146	SecondServerPassword	Password of the second FTP server during a ServerToServerFileTransfer operation.
"User name property (SecondServerUsername)" on page 147	SecondServerUsername	Username of the second FTP server during a ServerToServerFileTransfer operation.
"Specify criteria to split file content property (SplitCriteria)" on page 147	SplitCriteria	The delimiter that separates the business objects in the event file.
"Split function class name property (SplittingFunctionClassName)" on page 147	SplittingFunctionClassName	The fully qualified class name of the class file to be used to enable file splitting.
"Staging directory property (StagingDirectory)" on page 148	StagingDirectory	The file is first created into this directory.

Remote archive directory for retrieve operation property (ArchiveDirectory ForRetrieve)

During an outbound Retrieve operation, the adapter optionally archives the file to this folder before it is deleted. The archive directory must exist.

Table 47. Remote archive directory for retrieve operation property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Create new file if the file does not exist property (CreateFileIfNotExists)

During outbound Append and Overwrite operations, if the file does not exist on the FTP server, the adapter creates the file when this property is set to `true`. If this property is `false` and file does not exist, the adapter sends an error.

Table 48. Create new file if the file does not exist property characteristics

Required	No
Default	`false`
Property type	Boolean
Globalized	No

Directory property (SecondServerDirectory)

Directory of the second FTP server to which the server to server file transfer outbound operation is performed.

Syntax for specifying the directory path is: ftp://
[UserId:password@]FTPserver[:port]DirectoryForSecondServer

The following information can also be specified:

- Username and password of a user with privileges to connect to the second FTP server and perform FTP operations. If not specified here, it must be specified in the SecondServerUsername and SecondServerPassword.properties.
- FTP port. If not specified in here, the adapter uses the default FTP port.
- Remote event directory. If not specified in here, the adapter transfers the file to the directory to which the connection is established in the FTP server.

Table 49. Directory property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

FTP server connection mode property (DataConnectionMode)

Data connection mode used by the FTP server during file transfers. Takes either `active` or `passive`. This value is used only when a file transfer is taking place. This property is not used when performing a server to server file transfer outbound operation.

Table 50. FTP server connection mode property characteristics

Required	No
Default	`active`
Property type	String
Possible values	`active` or `passive`
Globalized	No

Default Object Name property (DefaultObjectName)

Name of the business object used by the Data Transformation Framework to access the ASI information required for data transformation.

The adapter gets the DataBinding class name corresponding to the RetrieveContentType value from the annotation of the business object.

Table 51. Default Object Name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes
Example	The generic business object used by the adapter is FTPFile.

DeleteOnRetrive

During an outbound Retrieve operation the adapter deletes the file from the FTP Server after it is retrieved when this property is set to `true`.

Table 52. Delete the file after retrieve operation property characteristics

Required	No
Default	`false`
Property type	Boolean
Globalized	No

Remote directory on FTP system property (DirectoryPath)

Absolute path of the directory on the FTP server where the outbound operation needs to be performed for all operations except ExecuteFTPScript, or the directory path on the local adapter workstation for the ExecuteFTPScript operation only. The directory must already exist.

Table 53. Remote directory on FTP system property characteristics

Required	Yes

Table 53. Remote directory on FTP system property characteristics (continued)

Default	None
Property type	String
Globalized	Yes

File content encoding property (FileContentEncoding)

Encoding used while writing to the file. If this property is not specified, the adapter tries to read without using any specific encoding. You can specify any Java supported encoding set.

Table 54. File content encoding property characteristics

Required	No
Default	None
Property type	String
Globalized	No

File in local directory property (FileInLocalDirectory)

During outbound create operations, if this property is set to `true`, the file content is not available in the business object. The file is retrieved from the local directory on the adapter workstation. During outbound retrieve operations, if this property is set to `true`, the file content is not sent to the J2EE application as part of the business object. The file is saved to the local directory of the adapter workstation.

Table 55. File in local directory property characteristics

Required	No
Default	`false`
Property type	Boolean
Globalized	No

Default target file name property (Filename)

Name of the file in the directory provided by the DirectoryPath property on which the outbound operations are performed. This value is required for all outbound operations except LIST.

Table 56. Default target file name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

File transfer type property (FileTransferType)

File transfer type used during outbound operations. Takes either ASCII or binary.

Table 57. File transfer type property characteristics

Required	No
Default	`binary`
Property type	String
Globalized	No

GenerateUniqueFile

During outbound Create, Append, and Overwrite operations the adapter creates a unique file name when this property is `true`. The adapter ignores any value that is set for the `Filename` property when this property is set to `true`.

The adapter also creates a unique file name when both this property and the `CreateFileIfNotExists` property are set to `true` during Append and Overwrite operations.

Note: The adapter does not support both GenerateUniqueFile and StagingDirectory options at the same time.

Table 58. Generate a unique file property characteristics

Required	No
Default	`false`
Property type	Boolean
Globalized	No
Restrictions	The FTP server must support RFC1123 to use this feature.

Host name property (SecondServerHostName)

Hostname of the second FTP server to which the connection is established during an outbound operation.

Table 59. Host name property characteristics

Required	Yes
Default	None
Property type	String
Globalized	Yes

Include business object delimiter in the file content property (IncludeEndBODelimiter)

File content is appended with this value. Used during the outbound create, append, and overwrite operations.

Table 60. Include business object delimiter in the file content property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Local archive directory for create operation property (LocalArchiveDirForCreate)

During outbound create operations, when the file content is coming as part of the business object and LocalArchivingEnabledForCreate is set to `true`, the file is saved to the local workstation in this directory.

Table 61. Local archive directory for create property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Archive file in the local directory for create operation property

During outbound create operations, when the file content is coming as part of the business object from a J2EE application and this property is set to `true`, the file is saved to the local workstation in the LocalArchiveDirForCreate directory before performing the outbound operation.

Table 62. Archive file in the local directory for create operation property characteristics

Required	No
Default	`false`
Property type	Boolean
Globalized	No

Local directory property (LocalDirectoryPath)

During outbound create operations, when FileInLocalDirectory property is set to `true`, the file content is not available in the business object. Instead the file is picked from this directory. During outbound retrieve operations, when FileInLocalDirectory property is set to `true`, the file content is not sent as part of business object. The file is saved to this directory.

Table 63. Local directory property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Port number property (SecondServerPortNumber)

Port number of the second FTP server through which the connection is established during an outbound operation.

Table 64. Port number property characteristics

Required	Yes
Default	`21`
Property type	Integer

Table 64. Port number property characteristics (continued)

Globalized	No

Protocol property (SecondServerProtocol)

Protocol that determines whether the connection to be established is a normal FTP connection or a secure FTP connection.

For example:

Normal connection: FTP

Secure connection: FTPS

Table 65. Protocol property characteristics

Required	Yes
Default	`FTP`
Property type	String
Globalized	No

Retrieve Content Type property (RetrieveContentType)

Content/MIME type of the retrieved file that is sent to the Data Transformation Framework to invoke the right data handler while converting the native format to a business object. During PassThrough this value is set to `NULL`.

Table 66. RetrieveContentType property characteristics

Required	No
Default	None
Property type	String
Globalized	No
Example	`text/xml, text/delimited, text/namevalue`

Script File Parameters property (ScriptFileParameters)

During an outbound ExecuteFTPScript operation, the parameters required by the FTP script file are set in this property. During runtime the adapter replaces the parameters with these values.

Table 67. Script File Parameters property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Password property (SecondServerPassword)

Password of the second FTP server to which the file is transferred during a server to server file transfer outbound operation.

Table 68. Password property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

User name property (SecondServerUsername)

Username of the second FTP server to which the file is transferred during a server to server file transfer outbound operation.

Table 69. User name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Specify criteria to split file content property (SplitCriteria)

Takes different values based on the value of the SplittingFunctionClassName property.

- If the SplittingFunctionClassName property specifies that files are split based on a delimiter, then SplitCriteria contains the delimiter that separates the business objects in the event file.
- If SplittingFunctionClassName is set to a value which does splitting based on size, then the SplitCriteria property contains a valid number that represents the size in bytes.
  - If the event file size is greater than this value, the adapter splits the file into chunks of this size and the chunks are posted.
  - If the event file size is less than this value, the entire event file is posted. When SplitCriteria=0, chunking is disabled.

Table 70. Specify criteria to split file content property characteristics

Required	No
Default	0
Property type	String
Globalized	Yes

Split function class name property (SplittingFunction ClassName)

Takes the fully qualified class name of the class file to be used to enable file splitting. Requires two values:

- The com.ibm.j2ca.extension.utils.filesplit.SplitByDelimiter class that splits the event file based on delimiter.
- The com.ibm.j2ca.extension.utils.filesplit.SplitBySize class that splits the event file based on the event file size.

The delimiter or file size is provided in the SplitCriteria property. If the RetrieveContentType property is set to `null`, it is automatically set to a class name that performs splitting based on file size.

Table 71. Split function class name property characteristics

Required	No
Default	`com.ibm.j2ca.extension.utils.filesplit.SplitBySize`
Property type	String
Globalized	No

Staging directory property (StagingDirectory)

During outbound create operations, the file will first be created into this directory. When the file creation is complete, the file is copied to the directory specified in the DirectoryPath property. This staging directory is also used for Append and Overwrite operations where the specified file is copied to the StagingDirectory, if present. The appended or overwritten content is then moved back to the original specified directory. If StagingDirectory is not specified, the operation is run in the actual required directory.

Note: The adapter does not support both StagingDirectory and GenerateUniqueFile options at the same time.

Table 72. Staging directory property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Inbound configuration properties

WebSphere Adapter for FTP 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 FTP 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.
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**.

Adapter type properties

Adapter type properties provide the external service wizard with the adapter details. These properties are configured using the external service wizard before deployment or with the WebSphere Application Server administrative console after deployment.

Note: If you set any of these adapter type properties using bidirectional script, you must set values that identify the format of the bidirectional script entered for that property.

The adapter type properties and their purpose are described in the following table. A complete 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 understanding property details.

Table 73. Adapter type properties

Property name		Description
In the wizard	In the administrative console	
"Description property (Description)"	`Description`	Adapter description.
"Display Name property (DisplayName)"	`DisplayName`	Adapter display name.
"ID property (ID)" on page 151	`ID`	ID for the adapter type.
"Vendor property (Vendor)" on page 151	`Vendor`	Name of the vendor providing the adapter.
"Version property (Version)" on page 151	`Version`	Adapter version.

Description property (Description)

Adapter description.

Table 74. Description property characteristics

Required	Yes
Default	`IBM WebSphere Adapter for FTP`
Property type	String

Display Name property (DisplayName)

Adapter display name.

Table 75. DisplayName property characteristics

Required	Yes
Default	`IBM WebSphere Adapter for FTP`
Property type	String

ID property (ID)

ID for the adapter type.

Table 76. ID property characteristics

Required	Yes
Default	`FTP`
Property type	String

Vendor property (Vendor)

Name of the vendor providing the adapter.

Table 77. Vendor property characteristics

Required	Yes
Default	`IBM`
Property type	String

Version property (Version)

Adapter version.

Table 78. Version property characteristics

Required	Yes
Default	`6.1.0`
Property type	String

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, but are supported 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 details tables in the sections that follow, see Guide to understanding property details.

Table 79. Resource adapter properties for the Adapter for FTP

Property 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.
"EISEncoding (EISEncoding)"	`EISEncoding`	Encoding of the FTP Server.
(Not available)	enableHASupport	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 80. Adapter ID to use for logging and tracing details

Required	Yes
Default	CWYFT_FTPFile
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

EISEncoding (EISEncoding)

This property specifies the encoding of the FTP Server. Sets the encoding for the control connection while communicating with the FTP Server. Set the property if the FTP server's directories or file names contain globalized characters.

Table 81. EISEncoding characteristics

Required	No
Default	`None`
Property type	String
Examples	`UTF-8, ISO-8859-1`

Enable high availability support (enableHASupport)

Do not change this property. It must be set to `true`.

Log file maximum size (LogFileMaxSize)

This property specifies the size of the log files in kilobytes.

Table 82. Log file maximum size details

Required	No
Default	`0`
Property type	Integer
Usage	When the log file reaches it maximum size, the adapter start using a new log file. If the file size is specified as `0` or no maximum size is specified, the file does not have a maximum size.
Globalized	Yes
Bidi supported	No

Log file name (LogFilename)

This property specifies the full path name of the log file.

Table 83. Log file name details

Required	No
Default	No default value
Property type	String
Usage	This property is deprecated.
Globalized	Yes
Bidi supported	Yes

Log number of files (LogNumberOfFiles)

This property specifies the number of log files.

Table 84. Log number of files details

Required	No
Default	`1`
Property type	Integer
Usage	When a log file reaches its maximum size, the adapter starts using another log file. If no value is specified, the adapter creates a single log file.
Globalized	Yes
Bidi supported	No

Trace file maximum size (TraceFileMaxSize)

This property specifies the size of the trace files in kilobytes.

Table 85. Trace file maximum size details

Required	No
Default	`0`
Property type	Integer
Usage	If no value is specified, then the trace file has no maximum size.

Table 85. Trace file maximum size details (continued)

Globalized	Yes
Bidi supported	No

Trace file name (TraceFilename)

This property specifies the full path of the trace file.

Table 86. Trace file name details

Required	No
Default	No default value
Unit of measure	Kilobytes
Property type	String
Usage	This property is deprecated.
Globalized	Yes
Bidi supported	Yes

Trace number of files (TraceNumberOfFiles)

This property specifies the number of trace files to use. When a trace file reaches its maximum size, the adapter starts using another trace file.

Table 87. Trace number of files details

Required	No
Default	1
Property type	Integer
Usage	If no value is specified, the adapter uses a single trace file.
Globalized	Yes
Bidi supported	No

Activation specification properties

Activation specification properties are properties that hold the inbound event processing configuration information for a message endpoint.

Activation specification properties are used during endpoint activation to notify the adapter of eligible event listeners. During inbound processing, the adapter uses these event listeners to receive events before forwarding them to the endpoint (a message driven bean).

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.

The following table lists the activation 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 details tables in the sections that follow, see Guide to understanding property details.

Table 88. Activation specification properties

Property name		
In the wizard	In the administrative console	Description
"Ensure once-only event delivery (AssuredOnceDelivery)" on page 158	AssuredOnceDelivery	Specifies whether the adapter provides assured once delivery of events
"Auto create event table property (CreateTable)" on page 159	`CreateTable`	Tells the adapter whether or not to create the Event Persistence table.
"Create Table property (CreateTable)" on page 159	`CreateTable`	When set to `true`, the event table and related indexes are created.
"Custom parser class name property (CustomParserClassName)" on page 159	`CustomParserClassName`	Fully qualified class name of the custom parser which is used to parse the `ls –l` output.
"Database Password property (DatabasePassword)" on page 159	`DatabasePassword`	Password used by event persistence for retrieving the JDBC database connection from the data source.
"Database schema name property (SchemaName)" on page 159	`SchemaName`	Schema name of the database used by event persistence.
"Database Username property (DatabaseUsername)" on page 160	`DatabaseUsername`	Username used by event persistence for retrieving the JDBC database connection from the data source.
"FTP server connection mode property (DataConnectionMode)" on page 160	`DataConnectionMode`	Data connection mode used by the FTP server during file transfers.
(Not available)	`DefaultObjectName`	Supported for compatibility with earlier versions.
"Delivery type (DeliveryType)" on page 160	DeliveryType	Determines the order in which events are delivered by the adapter to the export
"Encoding used by FTP server property (EISEncoding)" on page 161	`EISEncoding`	Encoding of the FTP server.
(Not available)	`EventContentType`	Supported for compatibility with earlier versions.
"Event recovery data source (JNDI) name property (DataSourceJNDIName)" on page 161	`DataSourceJNDIName`	JNDI name of the data source used by event persistence to get the JDBC database connection.
"Event recovery table name property (EventTableName)" on page 161	`TableName`	Name of the table that is used by the adapter for event persistence.
"Event types to process (EventTypeFilter)" on page 162	EventTypeFilter	A delimited list of event types that indicates to the adapter which events it should deliver
"Failure file extension for local archive property (FailedArchiveExt)" on page 162	`FailedArchiveExt`	File extension used to archive business objects in the event file that are not successfully processed.

Table 88. Activation specification properties (continued)

"File content encoding property (FileContentEncoding)" on page 162	`FileContentEncoding`	Encoding used to read the event files.
"File extension for remote archive property (FTPRenameExtension)" on page 163	`FTPRenameExt`	File extension or suffix that the adapter uses to rename the remote FTP file
"Pass only file name and directory, not the content property (FilePassByReference)" on page 163	`FilePassByReference`	Specifies that the file content of the event file is not sent to the export.
"File transfer type property (FileTransferType)" on page 163	`FileTransferType`	File transfer type used during inbound processing.
"Do not process events that have a timestamp in the future (FilterFutureEvents)" on page 163	FilterFutureEvents	Specifies whether the adapter filters out future events by comparing the timestamp on each event with the system time
"Number of files to get at a time property (FTPGetQuantity)" on page 164	`FTPGetQuantity`	Determines the number of files retrieved from the remote FTP URL.
"Number of poll periods between downloads property (FTPPollFrequency)" on page 164	`FTPPollFrequency`	Determines how frequently the adapter polls the FTP server.
"Run FTP script file after downloading files property" on page 164	`FTPScriptFileExecuted AfterInbound`	Specifies the path of the script file that is run after every inbound poll cycle.
"Run FTP script file before downloading files property" on page 165	`FTPScriptFileExecuted BeforeInbound`	Specifies the path of the script file that is run prior to every inbound poll cycle.
"Host name property (HostName)" on page 165	`HostName`	Hostname of the FTP Server to which the connection is established.
"Include business object delimiter in the file content property (IncludeEndBODelimiter)" on page 165	`IncludeEndBODelimiter`	When set to `true`, the delimiter is sent with the business object content for further processing.
"Local archive directory property (LocalArchiveDirectory)" on page 165	`LocalArchiveDirectory`	Absolute path of the local Archive directory.
"Local directory property (LocalEventDirectory)" on page 166	`LocalEventDirectory`	Local system directory into which the adapter downloads event files from the FTP site.
"Maximum connections (MaximumConnections)" on page 166	MaximumConnections	The maximum number of connections that the adapter can use for inbound event delivery

Table 88. Activation specification properties (continued)

"Minimum connections (MinimumConnections)" on page 166	MinimumConnections	The minimum number of connections that the adapter can use for inbound event delivery
"File extension for local archive property (OriginalArchiveExt)" on page 167	`OriginalArchiveExt`	File extension used to archive the original event file.
"Password property (Password)" on page 167	`Password`	Password of the user who has privileges to connect to the FTP server and perform FTP operations.
"Password used to connect to event data source property (Password)" on page 167	`Password`	Password used during event persistence.
"Interval between polling periods (PollPeriod)" on page 167	PollPeriod	The length of time that the adapter waits between polling periods
"Maximum events in polling period (PollQuantity)" on page 168	PollQuantity	The number of events that the adapter delivers to the export during each poll period
"Port number property (PortNumber)" on page 168	`PortNumber`	Port number of the FTP server.
"Protocol property (Protocol)" on page 168	`Protocol`	Determines whether the connection is normal FTP or secure FTP.
"Retrieve files with this pattern property (EventFileMask)" on page 169	`EventFileMask`	Filter for the event files.
"Retry interval if connection fails (RetryInterval)" on page 170	RetryInterval	The length of time that the adapter waits between attempts to establish a new connection after an error during inbound operations
"Number of times to retry the system connection (RetryLimit)" on page 170	RetryLimit	The number of times the adapter tries to reestablish an inbound connection after an error
"Remote archive directory property (FTPArchiveDirectory)" on page 169	`FTPArchiveDirectory`	Relative path of the archive directory on the FTP server.
"Remote directory property (EventDirectory)" on page 169	`EventDirectory`	Remote directory of the FTP server from where the event files are retrieved for inbound processing.
"Host name property (SocksProxyHost)" on page 170	`SocksProxyHost`	Host name of the machine used as a proxy server.
"Password property (SocksProxyPassword)" on page 170	`SocksProxyPassword`	Password used to authenticate the proxy server.
"Port number property (SocksProxyPort)" on page 171	`SocksProxyPort`	Port number of the proxy server.
"User name property (SocksProxyUserName)" on page 171	`SocksProxyUserName`	User name used to authenticate the proxy server.

Table 88. Activation specification properties (continued)

"Sort event files property (SortEventFiles)" on page 171	`SortEventFiles`	Determines the sorting order of event files being polled.
"Specify criteria to split file content property (SplitCriteria)" on page 172	`SplitCriteria`	Takes different values based on the value of the SplittingFunctionClassName property.
"Splitting function class name property" on page 172	`SplittingFunctionClassName`	Takes the fully qualified class name of the class file to be used to enable file splitting.
"Stop the adapter when an error is encountered while polling (StopPollingOnError)" on page 172	StopPollingOnError	Specifies whether the adapter stops polling for events when it encounters an error during polling
"Success file extension for local archive property (SuccessArchiveExt)" on page 173	`SuccessArchiveExt`	File extension used to archive all of the successfully processed business objects.
"User name property (UserName)" on page 173	`UserName`	Name of the user who has privileges to connect to the FTP server and perform FTP operations.
"User name used to connect to event data source property (UserName)" on page 173	`UserName`	Username used by event persistence for getting the database connection.

Ensure once-only event delivery (AssuredOnceDelivery)

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

Table 89. 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

Auto create event table property (CreateTable)

Tells the adapter whether or not to create the Event Persistence table. If the value is `true` and table does not exist then the adapter creates the table. If the value is `false` the adapter does not create the table.

Table 90. Auto create event table property characteristics

Required	No
Default	`true`
Property type	Boolean
Globalized	No

Create Table property (CreateTable)

When set to `true`, the event table and related indexes are created. For troubleshooting table creation errors, set this property to `false`. The table and indexes can then be created manually.

Table 91. Create Table property characteristics

Required	No
Default	`true`
Property type	Boolean
Globalized	No

Custom parser class name property (CustomParserClassName)

Fully qualified class name of the custom parser which is used to parse the `ls –l` output. Used only when the `ls –l` output deviates from standard output.

Table 92. Custom parser class name property characteristics

Required	No
Default	None
Property type	String
Globalized	No

Database Password property (DatabasePassword)

Password used by event persistence for retrieving the JDBC database connection from the data source.

Table 93. Database Password property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Database schema name property (SchemaName)

Schema name of the database used by event persistence.

Table 94. Database schema name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Database Username property (DatabaseUsername)

Username used by event persistence for retrieving the JDBC database connection from the data source.

Table 95. Database Username property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

FTP server connection mode property (DataConnectionMode)

Data connection mode used by the FTP server during file transfers. Accepts either active or passive settings.

Table 96. FTP server connection mode property characteristics

Required	No
Default	`active`
Property type	String
Globalized	No

Delivery type (DeliveryType)

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

Table 97. 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.
Globalized	No
Bidi supported	No

Encoding used by FTP server property (EISEncoding)

Encoding of the FTP server. Use this value to set the encoding for the control connection to the FTP server.

- When both EISEncoding at the adapter level and EISEncoding at the activation specification level are not set (both are null), nothing is set on the control connection while communicating with the FTP server.
- When EISEncoding at the adapter level is set and EISEncoding at the activation specification level is not set, the value at adapter level is set on the control connection while communicating with the FTP server. This is helpful when using multiple activation specification's and the same encoding is set. In this case, set the value at the adapter level so that all the connections will have the same encoding for the control connection.
- When EISEncoding at the adapter level is not set and EISEncoding at the activation specification level is set, the value at activation specification level is set on the control connection while communicating with the FTP server. Since the value is at the activation specification level, this is applicable for only that activation specification.
- When both EISEncoding at the adapter level and EISEncoding at the activation specification level are set, the value at the activation specification level takes precedence.

Specify any Java-supported encoding set for this attribute.

Table 98. Encoding used by FTP server property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Event recovery data source (JNDI) name property (DataSourceJNDIName)

JNDI name of the data source used by event persistence to get the JDBC database connection. The data source must be created in WebSphere Process Server. The database name specified while creating the data source must already exist.

Table 99. Event recovery data source (JNDI) name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Event recovery table name property (EventTableName)

Name of the table that is used by the adapter for event persistence. When using multiple activation specifications, this value must be unique for each. The same table name must not be used by other instances of same adapter or a different adapter. If the table does not exist in the database, the adapter will create the table.

Table 100. Event recovery table name property characteristics

Required	No

Table 100. Event recovery table name property characteristics (continued)

Default	`FTPTABLE`
Property type	String
Globalized	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 101. Event types to process details

Required	No
Possible values	A comma-delimited (,) list of business object types
Default	`null`
Property type	String
Usage	Events are filtered by business object type. If the property is set, the adapter delivers only those events that are in the list. A value of `null` indicates that no filter will be applied and that all events will be delivered to the export.
Example	To receive only events relating to the Customer and Order business objects, specify this value: `Customer,Order`
Globalized	No
Bidi supported	No

Failure file extension for local archive property (FailedArchiveExt)

File extension used to archive business objects in the event file that are not successfully processed. This property is used only when LocalArchiveDirectory is valid and exists.

Table 102. Failure file extension for local archive property characteristics

Required	No
Default	`fail`
Property type	String
Globalized	Yes

File content encoding property (FileContentEncoding)

Encoding used to read the event files based on the EndBODelimiter property and during string to byte[] conversions. If not specified, the adapter attempts to read without any specific encoding. You can specify any Java supported encoding set.

Table 103. File content encoding property characteristics

Required	No
Default	None
Property type	String
Globalized	No

File extension for remote archive property (FTPRenameExtension)

File extension or suffix that the adapter uses to rename the remote FTP file after the connector has polled for it. Renaming the file prevents the connector from polling the same file in the next poll cycle. The adapter can be configured to rename the processed event file and move it to an archive directory.

Table 104. File extension for remote archive property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Pass only file name and directory, not the content property (FilePassByReference)

Specifies that the file content of the event file is not sent to the export.

If set to `true`, the file is appended with a timestamp and sent to the LocalArchiveDirectory. The timestamp prevents errors and overwrites to the file when another file with the same name is received. This property can be set to `true` only when the LocalArchiveDirectory property is set and the specified directory exists. This property is used only for PassThrough inbound processing. When enabled, the file is not split into chunks.

Table 105. Pass only file name and directory, not the content property characteristics

Required	No
Default	`false`
Property type	Boolean
Globalized	No

File transfer type property (FileTransferType)

File transfer type used during inbound processing. Accepts either ASCII or binary.

Table 106. File transfer type property characteristics

Required	No
Default	`binary`
Property type	String
Globalized	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 107. Do not process events that have a timestamp in the future details

Required	Yes

Table 107. Do not process events that have a timestamp in the future details (continued)

Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	If set to `True`, the adapter compares the time of each event to the system time. If the event time is later than the system time, the event is not be delivered. If set to `False`, the adapter delivers all events.
Globalized	No
Bidi supported	No

Number of files to get at a time property (FTPGetQuantity)

Determines the number of files retrieved from the remote FTP URL with each remote poll.

Table 108. Number of files to get at a time property characteristics

Required	Yes
Default	`10`
Property type	Integer
Globalized	No

Number of poll periods between downloads property (FTPPollFrequency)

Determines how frequently the adapter polls the FTP server, measured in the number of standard poll cycles. For example, if PollPeriod is set to 10000, and FTPPollFrequency is set to 6, the adapter polls the LocalEventDirectory every 10 seconds and polls the remote EventDirectory every 60 seconds. The adapter performs FTP polling only if you specify a value for this property. If PollPeriod is 0, we consider it as 1 for calculation. If the calculation evaluates to 0, the adapter does not perform FTP polling.

Table 109. Number of poll periods between downloads property characteristics

Required	Yes
Default	`5`
Property type	Integer
Globalized	No

Run FTP script file after downloading files property

Specifies the path of the script file that is run after every inbound poll cycle. This feature can be used to perform additional actions on the FTP server after each poll cycle. For example, it can be used to set file permissions.

Table 110. Run FTP script file after downloading files property characteristics

Required	No
Default	None

Table 110. Run FTP script file after downloading files property characteristics (continued)

Property type	String
Globalized	Yes

Run FTP script file before downloading files property

Specifies the path of the script file that is run prior to every inbound poll cycle. This feature can be used to perform additional actions on the FTP server prior to each poll cycle. For example, it can be used to set file permissions.

Table 111. Run FTP script file before downloading files property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Host name property (HostName)

Hostname of the FTP Server to which the connection is established during inbound processing.

Table 112. Create Table property characteristics

Required	Yes
Default	None
Property type	String
Globalized	Yes

Include business object delimiter in the file content property (IncludeEndBODelimiter)

When set to `true`, the delimiter is sent with the business object content for further processing. This property is valid only when splitting the event files based on a delimiter.

Table 113. Include business object delimiter in the file content property characteristics

Required	No
Default	`false`
Property type	String
Globalized	No

Local archive directory property (LocalArchiveDirectory)

Absolute path of the local Archive directory. The directory must be valid and already exist.

Table 114. Local archive directory property characteristics

Required	No
Default	None

Table 114. Local archive directory property characteristics (continued)

Property type	String
Globalized	Yes

Local directory property (LocalEventDirectory)

Local system directory into which the adapter downloads event files from the FTP site. You must specify a value for this property to enable the adapter to process events.

Table 115. Local directory property characteristics

Required	Yes
Default	None
Property type	String
Globalized	Yes

Maximum connections (MaximumConnections)

This property specifies the maximum number of connections that the adapter can use for inbound event delivery.

Table 116. Maximum connections details

Required	No
Default	1
Property type	Integer
Usage	Only positive values are valid. The adapter considers any positive entry less than 1 to be equal to 1. Typing a negative value or 1 for this property may result in run time errors.
Globalized	No
Bidi supported	No

Minimum connections (MinimumConnections)

This property specifies the minimum number of connections that the adapter can use for inbound event delivery.

Table 117. Minimum connections details

Required	No
Default	1
Property type	Integer
Usage	Only positive values are valid. Any value less than 1 is treated as 1 by the adapter. Typing a negative value or 1 for this property may result in run time errors.
Globalized	No
Bidi supported	No

File extension for local archive property (OriginalArchiveExt)

File extension used to archive the original event file. This preserves the entire event file for reference in case any of its business objects fail. This property is used only when LocalArchiveDirectory is valid and exists.

Table 118. File extension for local archive property characteristics

Required	No
Default	`original`
Property type	String
Globalized	Yes

Password property (Password)

Password of the user who has privileges to connect to the FTP server and perform FTP operations. You do not need to specify a value for this property if the password is included in the URL specified in the EventDirectory property.

Table 119. Password property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Password used to connect to event data source property (Password)

The password used during event persistence to get the database connection from the data source.

Table 120. Password used to connect to event data source property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Interval between polling periods (PollPeriod)

This property specifies the length of time that the adapter waits between polling periods.

Table 121. 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

Table 121. Interval between polling periods details (continued)

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 122. Maximum events in polling period details

Required	Yes
Default	`10`
Property type	Integer
Usage	The value must be greater than 0. If this value is increased, more events are processed per polling period and the adapter may perform less efficiently. If this value is decreased, less events are processed per polling period and the adapter's performance may improve slightly.
Globalized	No
Bidi supported	No

Port number property (PortNumber)

Port number of the FTP server through which the connection is established during inbound processing.

Table 123. Port number property characteristics

Required	Yes
Default	`21`
Property type	Integer
Globalized	No

Protocol property (Protocol)

Protocol that determines whether the connection to be established is a normal FTP connection or a secure FTP connection.

For example:

Normal connection: FTP

Secure connection: FTPS

Table 124. Protocol property characteristics

Required	Yes
Default	`FTP`
Property type	String

Table 124. Protocol property characteristics (continued)

Globalized	No

Remote archive directory property (FTPArchiveDirectory)

Relative path of the archive directory on the FTP server. The directory must already exist. There are several options for using this property to specify archiving:

- Specifying a value for this property, but no value for the FTPRenameExt property causes the adapter to append a timestamp to the event file name and move it to the FTP server archive directory specified in this property.
- Specifying a value for this property and the FTPRenameExt property causes the adapter to rename the processed event file name with a timestamp and the value specified in FTPRenameExt and moves it to the FTP server archive directory specified in this property.
- Specifying no value either for this property or the FTPRenameExt property causes the adapter to delete the processed event file without archiving it.
- Specifying no value for this property but specifying a value for the FTPRenameExt property causes the adapter to rename the processed event file, adding a timestamp and the value specified in FTPRenameExt.

Table 125. Remote archive directory property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Remote directory property (EventDirectory)

Remote directory of the FTP server from where the event files are retrieved for inbound processing.

Table 126. Remote directory property characteristics

Required	Yes
Default	`*.*`
Property type	String
Globalized	Yes

Retrieve files with this pattern property (EventFileMask)

Filter for the event files. The file filter is a well-qualified expression consisting of alphanumeric characters and the `*` and `?` wild cards.

Table 127. Retrieve files with this pattern property characteristics

Required	Yes
Default	`*.*`
Property type	String
Globalized	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 128. Retry interval details

Required	Yes
Default	`2000`
Unit of measure	Milliseconds
Property type	Integer
Usage	Only positive values are valid. When the adapter encounters an error related to the inbound connection, this property specifies the length of time the adapter waits before trying to establish a new connection.
Globalized	Yes
Bidi supported	No

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 129. 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

Host name property (SocksProxyHost)

Host name of the machine used as a proxy server through which the adapter requests are routed to the FTP server.

Table 130. Host name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Password property (SocksProxyPassword)

Password used to authenticate the proxy server.

Table 131. Password property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Port number property (SocksProxyPort)

Port number of the proxy server through which the adapter requests are routed to the FTP server.

Table 132. Port number property characteristics

Required	No
Default	`1080`
Property type	Integer
Globalized	No

User name property (SocksProxyUserName)

User name used to authenticate the proxy server.

Table 133. User name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

Sort event files property (SortEventFiles)

Determines the sorting order of event files being polled. Supported values are:

- by file name – sort ascending on file name
- by timestamp – sort ascending on last modified timestamp
- no sort – not sorted

Event file ordering from which events need to be delivered is valid only if the activation specification DeliveryType property is set to `ORDERED`. file name sorting is provided based on the locale of the FTP server. The ICU4J package is used to track the locales and their corresponding rules.

Table 134. Sort event files property characteristics

Required	No
Default	<blank> (= not sorted)
Property type	String
Globalized	No

Specify criteria to split file content property (SplitCriteria)

This property takes different values based on the value of the SplittingFunctionClassName property. For example: To specify that a file is to be split every 5KB, set the SplitCriteria property to `5000`.

- If the SplittingFunctionClassName property specifies that files are split based on a delimiter, then SplitCriteria contains the delimiter that separates the business objects in the event file.
- If SplittingFunctionClassName is set to a value which does splitting based on size, then the SplitCriteria property contains a valid number that represents the size in bytes.
  - If the event file size is greater than this value, the adapter splits the file into chunks of this size and the chunks are posted.
  - If the event file size is less than this value, the entire event file is posted. When SplitCriteria=0, chunking is disabled.

When FilePassByReference is enabled during inbound PassThrough, the event file is not split.

Table 135. Specify criteria to split file content property characteristics

Required	No
Default	`0`
Property type	String
Globalized	Yes

Splitting function class name property

This value takes the fully qualified class name of the class file to be used to enable file splitting. Requires two values:

- The com.ibm.j2ca.extension.utils.filesplit.SplitByDelimiter class that splits the event file based on delimiter.
- The com.ibm.j2ca.extension.utils.filesplit.SplitBySize class that splits the event file based on the event file size.

Optionally, you can provide a custom file splitter class and use it by inputting the class name into the SplittingFunctionClassName property.

The delimiter or file size is provided in the SplitCriteria property. If the EventContentType property is set to null, it is automatically set to a class name that performs splitting based on file size.

Table 136. Splitting function class name property characteristics

Required	No
Default	`com.ibm.j2ca.extension.utils.filesplit.SplitBySize`
Property type	String
Globalized	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 137. 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 local archive property (SuccessArchiveExt)

File extension used to archive all of the successfully processed business objects. This property is used only when LocalArchiveDirectory is valid and exists. For example, `12345.order > 12345.order.success`

Table 138. Success file extension for local archive property characteristics

Required	No
Default	`success`
Property type	String
Globalized	Yes

User name property (UserName)

Name of the user who has privileges to connect to the FTP server and perform FTP operations. You do not need to specify a value for this property if the Username is included in the URL specified in the EventDirectory property.

Table 139. User name property characteristics

Required	No
Default	None
Property type	String
Globalized	Yes

User name used to connect to event data source property (UserName)

Username used by event persistence for getting the database connection from the data source.

Table 140. User name used to connect to event data source property characteristics

Required	No
Default	None
Property type	String

Table 140. User name used to connect to event data source property characteristics (continued)

Globalized	Yes

Globalization

WebSphere Adapter for FTP 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 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 transformation, which refers 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.

Globalization

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). Components in the WebSphere Business Integration system are written in Java. Therefore, when data is transferred between WebSphere Business 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 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, standards are used to display and process it. WebSphere Process Server and WebSphere Enterprise Service Bus use the Windows standard format, but an enterprise information system exchanging data with WebSphere Process Server or WebSphere Enterprise Service Bus can use a different format. WebSphere Adapters transform bidirectional script data passed between the two systems so that it is accurately processed and displayed on both sides of a transaction.

Bidirectional format

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 prior to 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 141. Bidirectional format attributes

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	Text is shaped Text is not shaped Initial shaping Middle shaping Final shaping Isolated shaping	N
5	Numeric Shaping	H C N	Hindi Contextual Nominal	N

The adapter transforms data into a logical, left-to-right format before sending the data to WebSphere Process Server or WebSphere Enterprise Service Bus.

Using bidirectional properties

You can use multiple bidirectional properties to control the transformation of both content data and metadata. You can set special bidirectional properties to exclude either content data or metadata from bidirectional transformation, or to identify data that requires special treatment during a transformation.

The following table describes four types of bidirectional properties.

Table 142. Bidirectional property types

Property type	Data transformations
EIS	Controls the format for content data, or data that is sent by the enterprise information system.
Metadata	Controls the format for metadata, or data that provides information about the content data.
Skip	Identifies content or metadata to exclude from transformation.
Special Format	Identifies certain text, such as file paths or URLs, that require different treatment during the transformation process. Can be set for either content data or metadata.

You can set properties that control bidirectional transformation in three areas.

- **Resource adapter properties:** These properties store default configuration settings, including the TurnBiDiOff property, which controls whether the adapter instance performs bidirectional transformation or not. Use the administrative console of the server to configure these properties.

- **Managed (J2C) connection factory properties:** These properties are used at run time to create an outbound connection instance with an enterprise information system. Once the managed connection factory properties are created, they are stored in the deployment descriptor.
- **Activation specification properties:** These properties hold the inbound event processing configuration information for a message endpoint. Set them as you perform external service, or use the administrative console of the server.

Business object annotations

Some adapters allow you to annotate bidirectional properties within a business object. Do this to add information that specifically controls the transformation of a business object or part of a business object. Use business object editor, a tool within WebSphere Integration Developer, to add annotations at these levels:

- Business object
- Business object application-specific attribute
- Business object attribute
- Business object attribute application-specific attribute

Property scope and lookup mechanism

After you set values for bidirectional properties for an adapter and annotate business objects where appropriate, the adapter performs bidirectional transformations. It does so by using logic that relies on a hierarchical inheritance of property settings and a lookup mechanism.

Properties defined within the resource adapter are at the top of the hierarchy, while those defined within other areas or annotated within a business object are at lower levels of the hierarchy. So for example, if you only set values for EIS-type bidirectional properties for the resource adapter, those values are inherited and used by transformations that require a defined EIS-type bidirectional property whether they arise from an inbound (activation specification) transaction or an outbound (managed connection factory) transaction.

However, if you set values for EIS-type bidirectional properties for both the resource adapter and the activation specification, a transformation arising from an inbound transaction uses the values set for the activation specification.

The processing logic uses a lookup mechanism to search for bidirectional property values to use during a transformation. The lookup mechanism begins its search at the level where the transformation arises and searches upward through the hierarchy for defined values of the appropriate property type. It uses the first valid value it finds. It searches the hierarchy from child to parent only; siblings are not considered in the search.

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 (J2C) connection factory properties

The following managed (J2C) connection properties control bidirectional transformation.

- FTPURL
- FileName
- StagingDirectory
- SecondServerUsername
- SecondServerPassword
- SecondServerDirectory
- SocksProxyUsername
- SocksProxyPassword
- FileSequenceLog

Activation specification properties

The following activation specification properties control bidirectional transformation.

- EventDirectory
- EventFileMask
- FTPArchiveDirectory
- LocalEventDirectory
- LocalArchiveDirectory
- FTPScriptFileExecutedBeforeInbound
- FTPScriptFileExecutedAfterInbound
- FTPRenameExt
- FailedArchiveExt
- OriginalArchiveExt
- SuccessArchiveExt
- SplitCriteria
- SocksProxyUsername
- SocksProxyPassword

Deployment Descriptor configuration properties

The following Deployment Descriptor configuration properties control bidirectional transformation.

- EPDataSourceJNDIName
- EPEventTableName
- EPDatabaseUsername
- EPDatabasePassword
- EPDatabaseSchemaName

Wrapper business object properties

The following wrapper business object properties control bidirectional transformation.

- DirectoryPath

- Filename
- ChunkInfo
- FtpServerEventDirectory
- SecondServerDirectory
- SecondServerUsername
- SecondServerPassword
- IncludeEndBODelimiter
- LocalDirectoryPath
- LocalArchiveDirForCreate
- StagingDirectory
- ScriptFileParameters
- SplitCriteria
- ArchiveDirectoryForRetrieve

Adapter messages

View the messages issued by WebSphere Adapter for FTP 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 CWYFT are issued by WebSphere Adapter for FTP
- 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 FTP.

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 FTP, 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

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

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

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

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

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

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

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

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

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

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

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

IBM Corporation
Department 2Z4A/SOM1
294 Route 100
Somers, NY 10589-0100
U.S.A.

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: (c) (your company name) (year). Portions of

this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

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

Programming interface information

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

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

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

Warning:

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

Trademarks and service marks

IBM, the IBM logo, developerWorks, i5/OS, Redbooks, Tivoli, ViaVoice, WebSphere, and z/OS are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

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

Microsoft and Windows are registered trademark of Microsoft Corporation in the United States, other countries, or both.

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

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

This product includes software developed by the Eclipse Project (http://www.eclipse.org).

Index

A

accessibility
  administrative console 18
  external service wizard 18
  IBM Accessibility Center 18
  keyboard 18
  shortcut keys 18
activation specification properties
  list of 154
  setting in administrative console 97, 102
Activation specification properties 154
adapter application
  starting 103
  stopping 103
adapter architecture 3
Adapter for FTP
  accessibility 18
  administering 93
  standards compliance 17
Adapter for FTP module
  exporting as EAR file 88
  installing EAR file on server 90
  starting 103
  stopping 103
Adapter implementation 19
adapter messages 178
adapter patterns wizard 1, 40
adapter performance 104
adapter technotes 179
annotation 4
Append 5
archive, event 15
ArchiveDirectoryForRetrieve 141
artifacts, generating 63
assembly editor 76
authentication
  description 22
  run time 22
authentication alias 35

B

backward compatibility
  project interchange files 29
  projects 29
business faults 111
business graph 4
business object attribute properties 123
business object operation support 123
business object structure 119
business object, custom 124
business object, predefine 37, 38
business objects 16, 119
  naming conventions 122

C

CEI (Common Event Infrastructure) 106
CharacterSet 49, 66
chunking 12
clustered environment
  deploying in 25
  description 25
  inbound processes 26
  outbound processes 26
Common Event Infrastructure (CEI) 106
compatibility matrix 3
compatibility with earlier versions 27
configuring
  logging 108
  Performance Monitoring Infrastructure (PMI) 104
  tracing 108
Create 5
CreateFileIfNotExists 141
custom business objects 124
custom properties
  activation specification 97, 102
  managed connection factory 95, 100
  resource adapter 93, 99
CustomerWrapper 4
CustomerWrapper business object 119
CustomerWrapperBG 4, 119

D

data transformation framework: inbound 11
data transformation framework: outbound 4
DataConnectionMode 142
debugging
  org.xml.sax.SAXParseException exception 116
  self-help resources 116
  XAResourceNotAvailableException exception 115
DefaultObjectName 142
Delete 5
DeleteOnRetrive 142
delimiter 11, 12
deployment
  environments 83
  options 23
  to production environment 86
  to test environment 83
deprecated features 27
Description 127, 150
developerWorks 179
developerWorks resources, WebSphere Adapters 178
Directory Path 142
DisplayName 127, 150

E

EAR file
  exporting 88
  installing on server 90
education, WebSphere Adapters 178
embedded adapter
  activation specification properties, setting 97
  considerations for using 24
  description 23
  managed connection factory properties, setting 95
  resource adapter properties, setting 93

EmbeddedNameFunctionSelector 10
enableHASupport property 26
endpoints 76
event archive, archiving on MVS platforms 15
event recovery 12
event store 14
exceptions
  org.xml.sax.SAXParseException 116
  XAResourceNotAvailableException 115
ExecuteFTPScript 5
Exists 5
exporting module as EAR file 88
External service connection properties 126, 150
external service wizard
  accessibility 18
  connection properties 67
  start up 67
  starting 46

F

faults
  description 111
Federal information processing standard 21
FFDC (first-failure data capture) 111
File splitting 12
FileContentEncoding 143
FileInLocalDirectory 143
Filename 143
FilenameFunctionSelector 10
files
  SystemOut.log log file 110
  trace.log trace file 110
FileTransferType 143
firewall 19
first-failure data capture (FFDC) 111
FTPFile 4
FTPFileBG 4
FTPFileBG business object 119
function selector 10

G

GenerateUniqueFile 144
generating artifacts 63
generating artifacts, inbound 79

H

hardware and software requirements 3
hardware requirements 3
high-availability environment
  deploying in 25
  description 25
  inbound processes 26
  outbound processes 26

I

IBM WebSphere Adapter Toolkit 179
ID 127, 151
implementation, Java 84
inbound configuration properties 148
Inbound event processing 9
Inbound processing 9
IncludeEndBODelimiter 144
installing EAR file 90
interaction specification properties
  changing 81
Interaction specification properties 139
Internet Protocol Version 6.0 (IPv6) 18
introduction 1
IPv6 18

J

J2EE component 58
Java implementation 84

K

keyboard 18

L

List 5
LocalArchiveDirForCreate 145
LocalArchivingEnabledForCreate 145
LocalDirectoryPath 145
Log Analyzer 109
log files
  changing file name 110
  disabling 108
  enabling 108
  level of detail 108
  location 110
logging
  configuring properties with administrative console 108

M

managed (J2C) connection factory properties
  setting in administrative console 95, 100
Managed (J2C) connection factory properties 131
matrix, compatibility 3
messages, adapter 178
migration considerations 27
module, creating 37
monitoring performance 104

N

naming conventions for business objects 122

O

org.xml.sax.SAXParseException 116
outbound configuration properties 125
Outbound processing 3
overview 1
Overwrite 5

P

package files for adapters 109
passive FTP mode 19
Passthrough processing 12
patterns 1, 40

Performance Monitoring Infrastructure (PMI)
  configuring 104
  description 104
  viewing performance statistics 107
performance statistics 107
PMI (Performance Monitoring Infrastructure)
  configuring 104
  description 104
  viewing performance statistics 107
pre-defined business objects 124
problem determination
  org.xml.sax.SAXParseException exception 116
  self-help resources 116
  XAResourceNotAvailableException exception 115
product overview 1
project interchange (PI) file
  updating without migrating 29
project, creating 46
properties
  activation specification 97, 102
    list of 154
  configuration properties
    inbound 148
    outbound 125
  inbound configuration 148
  managed (J2C) connection factory 95, 100
  outbound configuration 125
  resource adapter 93, 99

R

RAR (resource adapter archive) file
  description 87
  installing on server 87
recovery feature 12
Redbooks, WebSphere Adapters 178
reference bindings 58
reference bindings, inbound 76
related information 178
related products, information 178
requirements, hardware and software 3
resource adapter archive (RAR) file
  description 87
  installing on server 87
resource adapter properties
  details 128, 151
  setting in administrative console 93, 99
Retrieve 5
RetrieveContentType 146
Retry limit property 170
roadmap for configuring the module 33
runtime environment
  authentication in 22
  deploying EAR file to 86

S

samples 31
ScriptFileParameters 146
SecondServerDirectory 141
SecondServerHostName 144
SecondServerPassword 146
SecondServerPortNumber 145
SecondServerProtocol 146
SecondServerUsername 147
secure FTP 19
Secure socket layer (SSL) 19
Security 19
Selecting business objects and services: Outbound 52
selecting business objects, inbound 72
self-help resources 116
ServerToServerFileTransfer 5
Setting connection properties 49, 66
shortcut keys 18
software requirements 3
SplitByDelimiter 12
SplitBySize 12
SplitCriteria 11, 12, 147
SplittingFunctionClassName 147
SSL communication 19
StagingDirectory 148
stand-alone adapter
  activation specification properties, setting 102
  considerations for using 25
  description 23
  managed connection factory properties, setting 100
  resource adapter properties, setting 99
stand-alone reference 58
  wiring 76
standards compliance 17
starting adapter applications 103
stopping adapter applications 103
support
  overview 108
  self-help resources 116
  technical 179
Supported outbound operations 5
SystemOut.log file 110

T

target component 83
Technical overview 3
technical support 179
technotes 3, 116, 179
technotes, WebSphere Adapters 178
test environment
  adding module to 85
  deploying to 83, 85
  testing modules 86
trace files
  changing file name 110
  disabling 108
  enabling 108
  level of detail 108
  location 110
trace.log file 110
tracing
  configuring properties with administrative console 108
troubleshooting
  org.xml.sax.SAXParseException exception 116
  overview 108
  self-help resources 116
  XAResourceNotAvailableException exception 115
trust store 19, 21
tutorials 31

V

Vendor 127, 151
Version 127, 151

W

WebSphere Adapters, version 6.0, information 179
WebSphere Adapters, version 6.0.2, information 179
WebSphere Application Server information 179
WebSphere Business Integration Adapters information 179
WebSphere Business Process Management, version 6.1.0,
information 179
WebSphere Enterprise Service Bus
deploying to 86
information 179
WebSphere Extended Deployment 25
WebSphere Integration Developer
information 179
starting 37, 38, 46
test environment 83
WebSphere Process Server
deploying to 86
information 179
wiring 58
wiring components 83
wrapper business object 4

X

XAResourceNotAvailableException 115
xsd files 119

Printed in USA