Using FTP Adapters with Ensemble

Version 2018.1
2020-11-13

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Using FTP Adapters with Ensemble
Ensemble Version 2018.1 2020-11-13

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

About This Book

This book describes how to configure and use the File Transfer Protocol (FTP) adapters that Ensemble provides (the adapters in the EnsLib.FTP package). This book contains the following sections:

- Using the FTP Inbound Adapter
- Using the FTP Outbound Adapter
- Reference for Settings

For a detailed outline, see the table of contents.

The following books provide related information:

- *Ensemble Best Practices* describes best practices for organizing and developing Ensemble productions.
- *Developing Ensemble Productions* explains how to perform the development tasks related to creating an Ensemble production.
- *Configuring Ensemble Productions* describes how to configure the settings for Ensemble productions, business hosts, and adapters. It provides details on settings not discussed in this book.

For general information, see the *InterSystems Documentation Guide*.

1

Using the FTP Inbound Adapter

This chapter describes how to use the FTP inbound adapter (EnsLib.FTP.InboundAdapter). It contains the following sections:

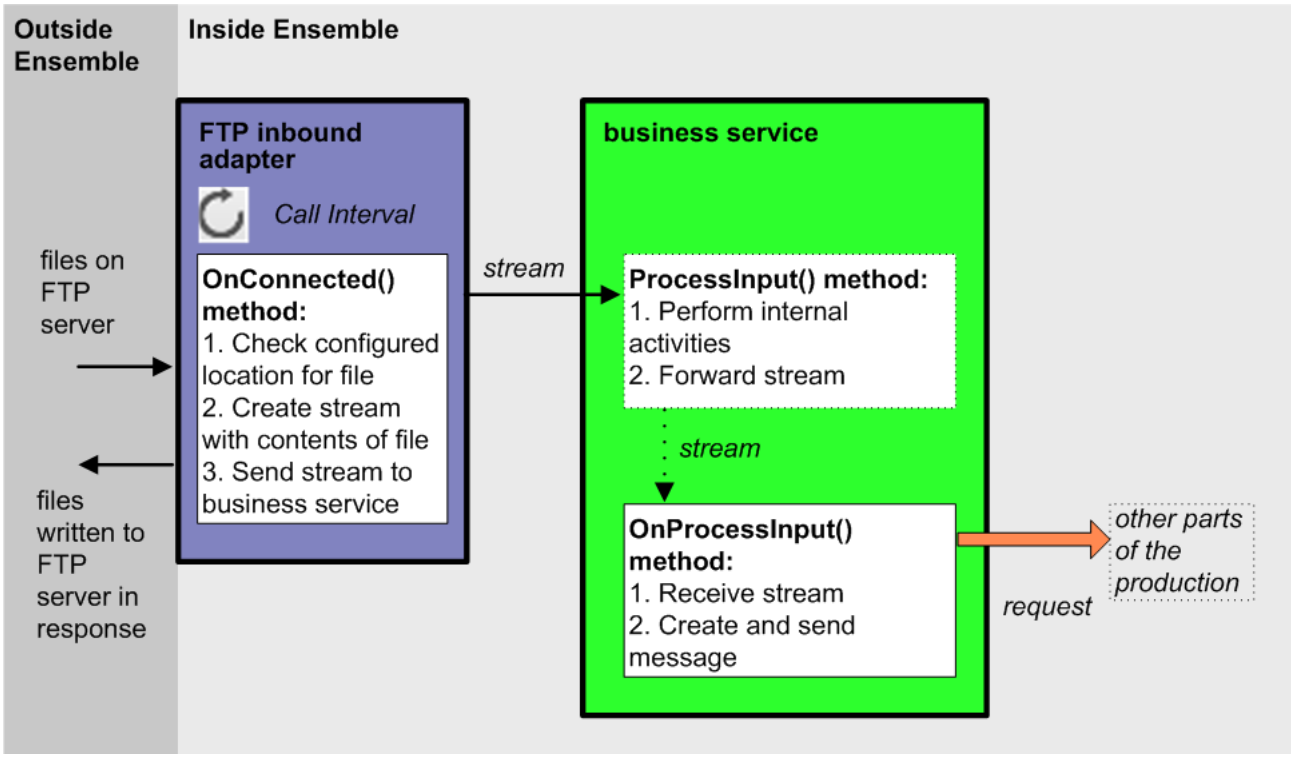
Tip: Ensemble also provides specialized business service classes that use this adapter, and one of those might be suitable for your needs. If so, no programming would be needed. See "Connectivity Options" in *Introducing Ensemble*.

1.1 Overall Behavior

The EnsLib.FTP.InboundAdapter enables Ensemble to receive files via the FTP protocol. The adapter receives FTP input from the configured location, reads the input, and sends the input as a stream to the associated business service. The business service, which you create and configure, uses this stream and communicates with the rest of the production.

If the FTP server expects an acknowledgment or response to its input, the business service is also responsible for formulating this response and relaying it back to the data source, via the EnsLib.FTP.InboundAdapter. The adapter does not evaluate or supplement the response, but relays it, if it is provided.

The following figure shows the overall flow of inbound messages (but not the responses):



In more detail:

1. Each time the adapter encounters input from its configured data source, it calls the internal **ProcessInput()** method of the business service class, passing the stream as an input argument.
2. The internal ProcessInput() method of the business service class executes. This method performs basic Ensemble tasks such as maintaining internal information as needed by all business services. You do not customize or override this method, which your business service class inherits.
3. The ProcessInput() method then calls your custom OnProcessInput() method, passing the stream object as input. The requirements for this method are described later in "Implementing the OnProcessInput() Method."

   If the data source expects an acknowledgment or response of some kind, the **OnProcessInput()** method of the business service creates it. The inbound adapter simply relays this response back to the external data source.

The response message follows the same path, in reverse.

1.2 Creating a Business Service to Use the Inbound Adapter

To use this adapter in your production, create a new business service class as described here. Later, add it to your production and configure it. You must also create appropriate message classes, if none yet exist. See "Defining Ensemble Messages" in *Developing Ensemble Productions*.

The following list describes the basic requirements of the business service class:

- Your business service class should extend Ens.BusinessService.
- In your class, the *ADAPTER* parameter should equal EnsLib.FTP.InboundAdapter.

- Your class should implement the OnProcessInput() method, as described in "Implementing the OnProcessInput Method."
- For other options and general information, see "Defining a Business Service Class" in *Developing Ensemble Productions*.

1.3 Implementing the OnProcessInput() Method

Within your custom business service class, your **OnProcessInput()** method should have the following signature:

```
Method OnProcessInput(pInput As %CharacterStream,pOutput As %RegisteredObject) As %Status
```

Or:

```
Method OnProcessInput(pInput As %BinaryStream,pOutput As %RegisteredObject) As %Status
```

Where:

- *pInput* is the message object that the adapter will send to this business service. This can be of type %CharacterStream or %BinaryStream, depending on the contents of the expected stream. You use an adapter setting (Charset) to indicate whether the input stream is character or binary; see "Reference for Settings."
- *pOutput* is the generic output argument required in the method signature.

The **OnProcessInput()** method should do some or all of the following:

1. Examine the input stream (*pInput*) and decide how to use it.

   This input type depends on the value of the Charset adapter setting:

   - When the Charset setting has the value Binary, *pInput* is of type %BinaryStream and contains bytes.
   - Otherwise, *pInput* is of type %CharacterStream and contains characters.

     For details about Charset, see "Reference for Settings."

2. Create an instance of the request message, which will be the message that your business service sends.

   For information on creating message classes, see "Defining Ensemble Messages" in *Developing Ensemble Productions*.

3. For the request message, set its properties as appropriate, using values in the input.
4. Call a suitable method of the business service to send the request to some destination within the production. Specifically, call **SendRequestSync()**, **SendRequestAsync()**, or (less common) **SendDeferredResponse()**. For details, see "Sending Request Messages" in *Developing Ensemble Productions*.

   Each of these methods returns a status (specifically, an instance of %Status).

5. Make sure that you set the output argument (`pOutput`). Typically you set this equal to the response message that you have received. This step is required.
6. Return an appropriate status. This step is required.

1.3.1 Invoking Adapter Methods

Within your business service, you might want to invoke the following instance methods of the adapter:

Connect()

```
Method Connect(pTimeout As %Numeric = 30,
               pInbound As %Boolean = 0) As %Status
```

Connect to the FTP server and log in, setting the directory and transfer mode.

Disconnect()

```
Method Disconnect(pInbound As %Boolean = 0) As %Status
```

Disconnect from the FTP server.

ParseFilename()

```
Method ParseFilename(pFilenameLine As %String,
                     Output pTimestamp As %String,
                     Output pSize As %String) As %Boolean
```

TestConnection()

```
Method TestConnection()
```

Correct the properties reflecting the connection state, in case the adapter thinks it is connected but has lost the socket.

The following methods are also available. Each method corresponds to an adapter setting that the user can adjust using the Management Portal. Each time the user clicks **Apply** to accept changes to the value of a *Setting*, the corresponding *Setting*`Set` method executes. These methods provide the opportunity to make adjustments following a change in any setting. For detailed descriptions of each setting, see "Reference for Settings."

ArchivePathSet()

```
Method ArchivePathSet(pInVal As %String) As %Status
```

ArchivePath is the directory where the adapter should place a copy of each file after processing.

CharsetSet()

```
Method CharsetSet(cset As %String) As %Status
```

Charset is the character set of the input file.

ConnectedSet()

```
Method ConnectedSet(pValue As %Boolean) As %Status
```

Connected is an internal property that tracks the adapter's connection to the FTP server.

CredentialsSet()

```
Method CredentialsSet(pInVal As %String) As %Status
```

Credentials is an Ensemble credentials entry that can authorize a connection to the FTP server. For information on creating Ensemble credentials, see *Configuring Ensemble Productions*.

FilePathSet()

```
Method FilePathSet(path As %String) As %Status
```

FilePath is the directory on the FTP server in which to look for files.

FTPPortSet()

```
Method FTPPortSet(port As %String) As %Status
```

FTPPort is the TCP Port on the FTP Server to connect to.

FTPServerSet()

```
Method FTPServerSet(server As %String) As %Status
```

FTPServer is the FTP Server to connect to. This can be the IP address or server name, as long as the domain host controller can resolve the name.

SSLConfigSet()

```
Method SSLConfigSet(sslcfg As %String) As %Status
```

SSLConfig is the SSL/TLS configuration entry to use to authenticate this connection.

1.4 Business Service Class

The following code example shows a business service class that references the EnsLib.FTP.InboundAdapter.

```
Class EnsLib.FTP.PassthroughService Extends Ens.BusinessService
{

Parameter ADAPTER = "EnsLib.FTP.InboundAdapter";

/// Configuration item(s) to which to send file stream messages
Property TargetConfigNames As %String(MAXLEN = 1000);

Parameter SETTINGS = "TargetConfigNames";

/// Wrap the input stream object in a StreamContainer message object and send
/// it. If the adapter has a value for ArchivePath, send async; otherwise send
/// synchronously to ensure that we don't return to the Adapter and let it
/// delete the file before the target Config Item is finished processing it.

Method OnProcessInput(pInput As %Stream.Object,
                      pOutput As %RegisteredObject) As %Status
{
  Set tSC=$$$OK, tSource=pInput.Attributes("Filename"),
                 pInput=##class(Ens.StreamContainer).%New(pInput)
  Set tWorkArchive=(""'=..Adapter.ArchivePath)

  For iTarget=1:1:$L(..TargetConfigNames, ",") {
    Set tOneTarget=$ZStrip($P(..TargetConfigNames,",",iTarget),"<>W")
    Continue:""=tOneTarget
    $$$sysTRACE("Sending input Stream ...")

    If tWorkArchive {
      Set tSC1=..SendRequestAsync(tOneTarget,pInput)
      Set:$$$ISERR(tSC1) tSC=$$$ADDSC(tSC,tSC1)
    } Else {
      #; If not archiving send Sync to avoid Adapter deleting file before
      #; Operation gets it
      Set tSC1=..SendRequestSync(tOneTarget,pInput)
      Set:$$$ISERR(tSC1) tSC=$$$ADDSC(tSC,tSC1)
      }
    }
    Quit tSC
  }
}
```

This example sets the *tSource* variable to the original file name which is stored in the *Filename* subscript of the Attributes property of the incoming stream (*pInput*).

1.5 Adding and Configuring the Business Service

To add your business service to an Ensemble production, use the Management Portal to do the following:

1. Add an instance of your business service class to the Ensemble production.
2. Configure the business service. For information on the settings, see "Reference for Settings."
3. Enable the business service.
4. Run the production.

2

Using the FTP Outbound Adapter

This chapter describes how to use the FTP outbound adapter (EnsLib.FTP.OutboundAdapter). It contains the following sections:

- Overall Behavior
- Creating a Business Operation to Use the Outbound Adapter
- Creating Message Handler Methods
- Example Business Operation Class
- Adding and Configuring the Business Operation

Tip: Ensemble also provides specialized business service classes that use this adapter, and one of those might be suitable for your needs. If so, no programming would be needed. See "Connectivity Options" in *Introducing Ensemble*.

2.1 Overall Behavior

EnsLib.FTP.OutboundAdapter enables an Ensemble production to send files via the FTP protocol. To use this adapter, you create and configure a business operation that uses the adapter. The business operation receives a message from within the production, looks up the message type, and executes the appropriate method. This method usually executes methods of the associated adapter.

2.2 Creating a Business Operation to Use the Outbound Adapter

To create a business operation to use the EnsLib.FTP.OutboundAdapter, you create a new business operation class. Later, add it to your production and configure it.

You must also create appropriate message classes, if none yet exist. See "Defining Ensemble Messages" in *Developing Ensemble Productions*.

The following list describes the basic requirements of the business operation class:

- Your business operation class should extend Ens.BusinessOperation.

- In your class, the *ADAPTER* parameter should equal EnsLib.FTP.OutboundAdapter.
- In your class, the *INVOCATION* parameter should specify the invocation style you want to use, which must be one of the following.
  - **Queue** means the message is created within one background job and placed on a queue, at which time the original job is released. Later, when the message is processed, a different background job is allocated for the task. This is the most common setting.
  - **InProc** means the message will be formulated, sent, and delivered in the same job in which it was created. The job will not be released to the sender's pool until the message is delivered to the target. This is only suitable for special cases.
- Your class should define a *message map* that includes at least one entry. A message map is an XData block entry that has the following structure:

```
XData MessageMap
{
<MapItems>
  <MapItem MessageType="messageclass">
    <Method>methodname</Method>
  </MapItem>
  ...
</MapItems>
}
```

- Your class should define all the methods named in the message map. These methods are known as *message handlers*. Each message handler should have the following signature:

  ```
  Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status
  ```

  Here *Sample* is the name of the method, *RequestClass* is the name of an Ensemble request message class, and *ResponseClass* is the name of an Ensemble response message class. In general, the method code will refer to properties and methods of the Adapter property of your business operation.

  For information on defining message classes, see "Defining Ensemble Messages" in *Developing Ensemble Productions*.

  For information on defining the message handler methods, see "Creating Message Handler Methods," later in this chapter.
- For other options and general information, see "Defining a Business Operation Class" in *Developing Ensemble Productions*.

Note: Studio provides a wizard that you can use to create a business operation stub similar to the preceding. To access this wizard, click **File —> New** and then click the **Production** tab. Then click **Business Operation** and click **OK**.

2.3 Creating Message Handler Methods

When you create a business operation class for use with EnsLib.FTP.OutboundAdapter, typically your biggest task is writing message handlers for use with this adapter, that is, methods that receive Ensemble messages and then communicate via FTP.

Each message handler method should have the following signature:

```
Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status
```

Here *Sample* is the name of the method, *RequestClass* is the name of an Ensemble request message class, and *ResponseClass* is the name of an Ensemble response message class.

In general, the method should do the following:

1. Examine the inbound request message.
2. When you want to call one of the methods in EnsLib.FTP.OutboundAdapter, use the Adapter property to identify the adapter object. The following example calls the method **PutStream()** from a business operation method:

   ```
   Quit ..Adapter.PutStream(pFilename,..%TempStream)
   ```

   You can use similar syntax to call any of the methods described in the section "Calling Adapter Methods from the Business Operation. "
3. Make sure that you set the output argument (`pOutput`). Typically you set this equal to the response message. This step is required.
4. Return an appropriate status. This step is required.

2.3.1 Calling Adapter Methods from the Business Operation

When your business operation class references the EnsLib.FTP.OutboundAdapter, the following adapter methods become available. You can call these adapter methods from the methods in your business operation class using the Adapter property as described in the previous topic:

Connect()

```
Method Connect(pTimeout As %Numeric = 30,
               pInbound As %Boolean = 0) As %Status
```

Connects to the FTP server and log in, setting the directory and transfer mode.

The FTP outbound adapter has a retry cycle, which it invokes to reestablish the connection when:

- A TCP socket failure or timeout occurs while it is trying to establish a TCP connection to the FTP server
- The local client responds with a network code (a timeout)
- The local client responds with a retryable code (FTP codes 52*x* or 4*xx*, such as code 529 for a timeout)

CreateTimestamp()

```
ClassMethod CreateTimestamp(pFilename As %String = "",
                            pSpec As %String = "")
                            As %String
```

Using the *pFilename* string as a starting point, incorporates the time stamp specifier provided in *pSpec* and returns the resulting string. The default time stamp specifier is `%f_%Q` where:

- `%f` is the name of the data source, in this case the input filename
- _ is the literal underscore character, which will appear in the output filename
- `%Q` indicates ODBC format date and time

In substituting a value for the format code `%f`, Ensemble strips out any of the characters |,?,\,/,:,[,],<,>,&,,,;,NUL,BEL,TAB,CR,LF, replacing spaces with underscores (_), slashes (/) with hyphens (-), and colons (:) with dots (.).

For full details about time stamp conventions, see "Time Stamp Specifications for Filenames" in *Configuring Ensemble Productions*.

Delete()

```
Method Delete(pFilename As %String) As %Status
```

Deletes a named file from an FTP server. Server, Username, Password, and File Path are already configured as settings for this adapter. Returns a status value indicating the success of the FTP operation.

Disconnect()

```
Method Disconnect(pInbound As %Boolean = 0) As %Status
```

Disconnects from the FTP server.

GetStream()

```
Method GetStream(pFilename As %String,
                 ByRef pStream As %Stream.Object = $$$NULLOREF) As %Status
```

Retrieves a named file from an FTP server and returns it as a stream. Server, Username, Password, FilePath and Transfer mode (Charset) are already configured as settings for this adapter. This method returns a status value indicating the success of the FTP operation. If the caller provides a stream, it must be the appropriate type of stream for the transfer (character or binary). This method will create the stream if none is provided.

NameList()

```
Method NameList(Output pFileList As %ListOfDataTypes) As %Status
```

Gets a list of files on an FTP server. Server, Username, Password, and FilePath are already configured as settings for this adapter. The filenames are returned in a %ListOfDataTypes object. This method returns a status value indicating the success of the FTP operation.

PutStream()

```
Method PutStream(pFilename As %String,
                 pStream As %Stream.Object) As %Status
```

Stores a stream to an FTP server as a named file. Server, Username, Password, FilePath and Transfer mode (Charset) are already configured as settings for this adapter. This method returns a status value indicating the success of the FTP operation.

Rename()

```
Method Rename(pFilename As %String,
                 pNewFilename As %String,
                 pNewPath As %String = "") As %Status
```

Renames a file on an FTP server. Server, Username, Password, and FilePath are already configured as settings for this adapter. This method returns a status value indicating the success of the FTP operation.

TestConnection()

```
Method TestConnection()
```

Correct the properties reflecting the connection state, in case the adapter thinks it is connected but has lost the socket.

The following methods are also available. Each method corresponds to an adapter setting that the user can adjust using the Management Portal. Each time the user applies changes of a setting, Ensemble executes the corresponding *Setting*`Set` method. These methods provide the opportunity to make adjustments following a change in any setting. For detailed descriptions of each setting, see "Reference for Settings."

CharsetSet()

```
Method CharsetSet(cset As %String) As %Status
```

Charset is the character set of the input file.

ConnectedSet()

```
Method ConnectedSet(pValue As %Boolean) As %Status
```

Connected is an internal property that tracks the adapter's connection to the FTP server.

CredentialsSet()

```
Method CredentialsSet(pInVal As %String) As %Status
```

Credentials is an Ensemble credentials entry that can authorize a connection to the FTP server. For information on creating Ensemble credentials, see *Configuring Ensemble Productions*.

FilePathSet()

```
Method FilePathSet(path As %String) As %Status
```

FilePath is the directory on the FTP server in which to look for files.

FTPPortSet()

```
Method FTPPortSet(port As %String) As %Status
```

FTPPort is the TCP Port on the FTP Server to connect to.

FTPServerSet()

```
Method FTPServerSet(server As %String) As %Status
```

FTPServer is the FTP Server to connect to. This can be the IP address or server name, as long as the domain host controller can resolve the name.

SSLConfigSet()

```
Method SSLConfigSet(sslcfg As %String) As %Status
```

SSLConfig is the SSL/TLS configuration entry to use to authenticate this connection.

2.4 Example Business Operation Class

The following code example shows a business operation class that references the EnsLib.FTP.OutboundAdapter:

```
Class EnsLib.FTP.PassthroughOperation Extends Ens.BusinessOperation
{
Parameter ADAPTER = "EnsLib.FTP.OutboundAdapter";

/// Name of file to output the document(s) to. May include timestamp
/// specifiers. The %f specifier if present will be replaced with the
/// name of the document's original source filename (stripped of
/// characters illegal in filenames).  See the method
/// Ens.Util.File.CreateTimestamp() for documentation of timestamping options.

Property Filename As %String(MAXLEN = 1000);

Parameter SETTINGS As %String = "Filename";
```

```
Method OnMessage(pRequest As Ens.StreamContainer,
                 Output pResponse As %Persistent) As %Status
  {
  Set tFilename=
  ..Adapter.CreateTimestamp(##class(%File).GetFilename(
    pRequest.Stream.Attributes("Filename")),..Filename)
  Quit ..Adapter.PutStream(tFilename, pRequest.Stream)
  }
}
```

2.5 Adding and Configuring the Business Operation

To add your business operation to an Ensemble production, use the Management Portal to do the following:

1. Add an instance of your business operation class to the Ensemble production.
2. Configure the business service. For information on the settings, see “Reference for Settings.”
3. Enable the business operation.
4. Run the production.

Reference for Settings

This section provides the reference information on the FTP inbound and outbound adapters. Also see "Settings in All Productions" in *Managing Productions*.

Settings for the FTP Inbound Adapter

Provides reference information for settings of the FTP inbound adapter, EnsLib.FTP.InboundAdapter.

Summary

The inbound FTP adapter has the following settings:

Group	Settings
Basic Settings	External Registry ID, Delete From Server, File Path, File Spec, Archive Path, Call Interval, FTP Server, FTP Port, Credentials
Connection Settings	SSL Configuration, UsePASV, Stay Connected, Connect Timeout
SFTP Settings	SFTPPublicKeyFile, SFTPPrivateKeyFile
Additional Settings	Use FileStream, Server List Style, Subdirectory Levels, Charset, Append Timestamp, Semaphore Specification, Confirm Complete, File Access Timeout

The remaining settings are common to all business services. For information, see "Settings for All Business Services" in *Configuring Ensemble Productions*.

Append Timestamp

Append a time stamp to filenames in the **Archive Path** and **Work Path** directories; this is useful to prevent possible name collisions on repeated processing of the same filename.

- If this value is empty or 0, no time stamp is appended.
- If this setting is 1, then the standard template '%f_%Q' is appended.
- For other possible values, see "Time Stamp Specifications for Filenames" in *Configuring Ensemble Productions*.

Archive Path

Full path name of the directory on the Ensemble server to save a copy of each file received from the FTP server. This directory must exist, and it must be accessible through the file system on the local Ensemble machine.

If not specified, Ensemble stores the local copy of the file in a temporary location and then deletes it after processing is completed.

Call Interval

The polling interval for the adapter, in seconds. This is the time interval at which the adapter checks for input files in the specified locations.

Upon polling, if the adapter finds a file, it links the file to a stream object and passes the stream object to the associated business service. If several files are detected at once, the adapter sends one request to the business service for each individual file until no more files are found.

If the business service processes each file synchronously, the files will be processed sequentially. If the business service sends them asynchronously to a business process or business operation, the files might be processed simultaneously.

After processing all the available files, the adapter waits for the polling interval to elapse before checking for files again. This cycle continues whenever the production is running and the business service is enabled and scheduled to be active.

It is possible to implement a callback in the business service so that the adapter delays for the duration of the **CallInterval** between inputs. For details, see the chapter "Defining Business Services" in *Developing Ensemble Productions*.

The default CallInterval is 5 seconds. The minimum is 0.1 seconds.

Charset

Specifies the character set of the input file. Ensemble automatically translates the characters from this character encoding. The setting value is not case-sensitive. Use `Binary` for binary files, or for any data in which newline and line feed characters are distinct or must remain unchanged, for example in HL7 Version 2 or EDI messages. Other settings may be useful when transferring text documents. Choices include:

- `Binary` — Binary transfer (the default for the FTP adapters)
- `Ascii` — Ascii mode FTP transfer but no character encoding translation
- `Default` — The default character encoding of the local Ensemble server
- `Latin1` — The ISO Latin1 8-bit encoding
- `ISO-8859-1` — The ISO Latin1 8-bit encoding
- `UTF-8` — The Unicode 8-bit encoding
- `UCS2` — The Unicode 16-bit encoding
- `UCS2-BE` — The Unicode 16-bit encoding (Big-Endian)
- Any other alias from an international character encoding standard for which NLS (National Language Support) is installed in Ensemble
- *@TranslationTable*, where *TranslationTable* is the name of an Ensemble translation table (in this case Ensemble uses the given translation table)

For background information on character translation in Caché, see "Localization Support" in the *Caché Programming Orientation Guide*.

Semaphore Specification

The Semaphore Specification allows you to indicate that the data file is complete and ready to be read by creating a second file that is used as a semaphore. The inbound FTP adapter waits until the semaphore file exists before checking the other conditions specified by the Confirm Complete requirements and then processing the data file. This allows the application creating the data file to ensure that the adapter waits until the data file is complete before processing it. The adapter tests only for the existence of the semaphore file and does not read the semaphore file contents.

If the Semaphore Specification is an empty string, the adapter does not wait for a semaphore file and processes the data file as soon as the conditions specified by the Confirm Complete requirements are met. If you are using a semaphore file to control when the adapter processes the data file, you should consider setting the Confirm Complete field to None.

The Semaphore Specification allows you to specify individual semaphore files for each data file or a single semaphore file to control multiple data files. You can use wildcards to pair semaphore files with data files, and can specify a series of patterns matching semaphore files to data files. The adapter always looks for a matching semaphore file in the same directory as the data file. If the adapter is looking for data files in subdirectories, the semaphore file must be in the same subdirectory level as its corresponding data file.

The general format for specifying the Semaphore Specification is:

```
[DataFileSpec=] SemaphoreFileSpec [;[DataFileSpec=] SemaphoreFileSpec]...
```

For example, if the Semaphore Specification is:

```
ABC*.TXT=ABC*.SEM
```

It means that the ABCTest.SEM semaphore file controls when the adapter processes the ABCTest.TXT file and that the ABCdata.SEM semaphore file controls when the adapter processes the ABCdata.txt file.

Note: In a semaphore specification, the * (asterisk) matches any character except dot. In a file specification, the asterisk matches any character including the dot.

You can have one semaphore file control multiple data files. For example, if the Semaphore Specification is:

```
*.DAT=DATA.SEM
```

The DATA.SEM semaphore file controls when the adapter processes all *.DAT files in the same directory. When the adapter is looking for data files and corresponding semaphore files, it loops through all the data files at a polling interval. With the previous Semaphore Specification, if it started looking for DATA.SEM for the ABC.DAT file and does not find it, it continues looking for the semaphore files for the other files. But, if during this process DATA.SEM is created and it is looking for a match for XYZ.DAT, it finds the corresponding semaphore file. But the adapter defers processing XYZ.DAT until the next polling interval because a preceding data file, ABC.DAT, was waiting for the same semaphore file.

If you specify multiple pairings, separate them with a ; (semicolon). For example, if the Semaphore Specification is:

```
*.TXT=*.SEM;*.DAT=*.READY
```

The semaphore file MyData.SEM controls when the adapter processes MyData.TXT, but the semaphore file MyData.READY controls when it processes MyFile.DAT.

The adapter finds the corresponding semaphore file for each data file by reading the Semaphore Specification from left to right. Once it determines the corresponding semaphore file, it stops reading the Semaphore Specification for that file. For example, if the Semaphore Specification is:

```
VIData.DAT=Special.SEM; *.DAT=*.SEM
```

The adapter looks for the semaphore file Special.SEM before it processes VIData.DAT, but it does not consider VIData.SEM as a semaphore file for VIData.DAT. It does consider stuff.SEM as the semaphore file for stuff.DAT because stuff.DAT did not match an earlier specification. Consequently, if you are including multiple specifications that can match the same file, you should specify the more specific specification before the more general ones.

The data file target pattern is case-sensitive and the semaphore pattern case sensitivity is operating system dependent, that is *.TXT=*.SEM .is only applied to target files found ending with capitalized .TXT but the operating system may not differentiate between *.SEM and *.sem. If the operating system is not case-sensitive, the adapter treats semaphore files ending in any case combination of *.SEM and *.sem as equivalent but only uses them as the semaphore for data files named *.TXT. It cannot distinguish case in the semaphore files but can distinguish it in the data files.

If you only specify a single file specification and omit the = (equals) sign, the adapter treats that as the Semaphore Specification for all data files. For example, if the Semaphore Specification is:

```
*.SEM
```

This is equivalent to specifying a single wildcard to the left of the = (equals) sign:

```
*=*.SEM
```

In this case, the semaphore file MyFile.SEM controls the data file MyFile.txt and the semaphore file BigData.SEM controls the data file BigData.DAT.

If no wildcard is used in the Semaphore Specification then it is the complete fileSpec for the semaphore file. For example, if the Semaphore Specification is:

```
*.DAT=DataDone.SEM
```

Then the DataDone.SEM semaphore file controls when the adapter reads any data file with the .DAT file extension.

If a Semaphore Specification is specified and a data file does not match any of the patterns, then there is no corresponding semaphore file and the adapter will not process this data file. You can avoid this situation by specifying * as the last data file in the Semaphore Specification. For example, if the Semaphore Specification is:

```
*.DAT=*.SEM; *.DOC=*.READY; *=SEM.LAST
```

The `SEM.LAST` is the semaphore file for all files that do not end with `.DAT` or `.DOC`.

If an adapter configured with a FileSpec equal to `*`, the adapter usually considers all files in the directory as data files. But, if the adapter also has a Semaphore Specification and it recognizes a file as a semaphore file, it does not treat it as a data file.

After the adapter has processed through all the data files in a polling cycle, it deletes all the corresponding semaphore files.

Confirm Complete

Confirm complete receipt of each file, if possible. This setting handles the case where the file is not completely available on the server at the time downloading begins. When configuring the adapter, choose one of the following options:

- `None` — May provide fastest performance for small files because no extra FTP directory listing request needs to be done for each file download attempt
- `Size` — This is the default. `Size` means keep reading more data for a file until the size reported for it in the server directory listing does not increase. This option is only reliable when the **Charset** is `Binary`, because in text mode, the file position used for downloading may become corrupted by the insertion or removal of line feed characters.
- `Rename` — Keep trying to read more data for a file until the server allows us to rename it. This option only works if the FTP server grants the rename privilege to Ensemble for the download directory, using the **Credentials** configured on this adapter, and if the file permissions on the file itself are set so that the FTP server has privilege to rename it. If not then the Rename attempt always fails, and the Ensemble download never completes.
- `Size & Rename` — The `Size` option alone may not be sufficient when the FTP servers or source application is sluggish. If the server reports the same size for the file twice, sequentially, 2 seconds apart, Ensemble considers the download complete. Therefore, the `Size & Rename` setting is preferable if the server supports rename.

If you are working with the ConfirmComplete property programmatically, each option has an integer value from 0 (None) to 3 (Size & Rename). The default is Size (integer value 1).

Connect Timeout

Number of seconds to wait on each attempt to connect to the FTP server. The default is 5 seconds.

Credentials

Identifies an Ensemble credentials entry that can authorize a connection to the FTP server. For information on creating Ensemble credentials, see *Configuring Ensemble Productions*.

Delete From Server

True or False. If True, delete files from the FTP server after successful processing. If False, the adapter ignores files already processed until something else deletes them from the FTP server. The default is True.

External Registry ID

ID of External Service Registry entry. Leave blank if you are not using the external registry The registry endpoint sets the FTPServer, FTPPort, FilePath, and SSLConfig properties. For more information, see "Configuring ESB Services and Operations" in *Using Ensemble as an ESB*.

File Access Timeout

Amount of time in seconds that the system waits for information from the source application before confirming the complete receipt of a file. For more information, see Confirm Complete.

If you supply a decimal value, the system rounds the value up to the nearest whole number. The default value is 2.

File Path

Full pathname of the directory on the FTP server in which to look for files. This directory must exist, and it must be accessible using the Credentials provided.

File Spec

Filename or wildcard file specification for file(s) to retrieve from the FTP server. For the wildcard specification, use the convention that is appropriate for the operating system on the FTP server machine.

FTP Port

Identifies the TCP port on the FTP server to connect to. The default value is 21.

FTP Server

Identifies the FTP server to connect to. This can be the IP address or server name, as long as the domain host controller can resolve the name.

Server List Style

Identifies the operating system on the FTP server. This determines the type of listing format that the FTP server returns. Choose one of the following:

- `UNIX` (the default)
- `MSDOS`

SFTPPrivateKeyFile

File path to a file containing the SSH private key. Private keys should be PEM encoded.

SFTPPublicKeyFile

File path to a file containing the SSH public key. Public keys should be in OpenSSH format.

SSL Config

The name of an existing SSL/TLS configuration to use to authenticate this connection. Choose a client SSL/TLS configuration, because the adapter initiates the communication.

To create and manage SSL/TLS configurations, use the Management Portal. See the chapter "Using SSL/TLS with Caché" in the *Caché Security Administration Guide*. The first field on the **Edit SSL/TLS Configuration** form is **Configuration Name**. Use this string as the value for the **SSLConfig** setting.

You can also use this setting to indicate you are using the SFTP mode. To do so, specify `!SFTP` for the setting value. See the description of the *SSLConfig* property in the EnsLib.FTP.InboundAdapter entry in the *InterSystems Class Reference* for details.

Once you indicate you are using SFTP, you can then configure the SFTPPublicKeyFile and SFTPPrivateKeyFile settings. If you supply values for both **SFTP Public Key File** and **SFTP Private Key File**, the adapter attempts key pair authentication. It does this in conjunction with the username and password supplied via the Credentials setting, using the password in the Credentials as the passphrase for the private key.

Stay Connected

A zero value means to disconnect immediately after every input event. The default of –1 means to stay permanently connected, even during idle times. Adapters are assumed idle at startup and therefore only auto-connect if they are configured with a StayConnected value of –1. The only StayConnected values that are useful in the normal case are 0 and –1. However, StayConnected can also be a positive number. If the StayConnected time is longer than the CallInterval, the adapter stays connected all the time. If the StayConnected time is shorter than the CallInterval, the adapter disconnects and reconnects at each CallInterval.

Subdirectory Levels

Number of levels of subdirectory depth under the given directory that should be searched for files.

Use FileStream

Specifies whether the adapter should use file stream for the data received. If this is false, the adapter uses a global stream. Note that regardless of this setting, a file stream will be used if Archive Path or ArchiveIO is set.

UsePASV

Use Passive FTP mode: server returns a data port address and the client connects to it. Most firewalls are more tolerant of Passive mode FTP because both the control and data TCP connections are initiated by the client.

Settings for the FTP Outbound Adapter

Provides reference information for settings of the FTP outbound adapter, EnsLib.FTP.OutboundAdapter.

Summary

The outbound FTP adapter has the following settings:

Group	Settings
Basic Settings	External Registry ID, FTP Server, FTP Port, Credentials, File Path
SFTP Settings	SFTP AppendMode, SFTPPublicKeyFile, SFTPPrivateKeyFile, SFTPSetFileAccessMode
Connection Settings	SSL Configuration, UsePASV, Stay Connected, Connect Timeout
Additional Settings	Overwrite, Charset, TempFileName

The remaining settings are common to all business operations. For information, see "Settings for All Business Operations" in *Configuring Ensemble Productions*.

Charset

Specifies the desired character set for the output file. Ensemble automatically translates the characters to this character encoding. See "Charset" in "Settings for the FTP Inbound Adapter."

Connect Timeout

Number of seconds to wait on each attempt to connect to the FTP server. The default is 5 seconds.

Credentials

Identifies an Ensemble credentials entry that can authorize a connection to the FTP server. See "Credentials" in "Settings for the FTP Inbound Adapter."

File Path

Full pathname of the directory on the FTP server in which to write files. This directory must exist, and it must be accessible using the Credentials provided.

FTP Port

Identifies the TCP port on the FTP server to connect to. The default value is 21.

FTP Server

Identifies the FTP server to connect to. This can be the IP address or server name, as long as the domain host controller can resolve the name.

Overwrite

True or False. If True, and the output file exists, overwrite it. If False, and the output file exists, append to it. The default is True.

SFTP Append Mode

Controls whether to use the SFTP server append (Server mode) or to emulate an append on the client (Client mode). Some SFTP servers do not support append access. If you are using a server that does not support append, select **Client mode** and the FTP outbound adapter reads the file from the SFTP server, appends the data to the file and then sends the file back to the SFTP server overwriting the existing file. This setting is only meaningful if SSL Configuration is set to !SFTP and Overwrite is set to false.

A consequence of emulating an append is that if another process modifies the same file on the SFTP server between the time that the adapter reads the file and the time that it writes the file with the appended data, the updates from the other process are lost.

Note: If you are unsure whether the SFTP server supports append access you can use a server append test that is accessible through a link on the **SFTP Append Mode** popup help text.

SFTPPrivateKeyFile

File path to a file containing the SSH private key certificate.

SFTPPublicKeyFile

File path to a file containing the SSH public key certificate.

SFTPSetFileAccessMode

SFTP File Access Mode specifies the access permissions to assign to the file on the remote system when transferred. It can be specified as either octal such as 0600, or symbolic such as u+rw,g+r. The default is 0600. If specifying octal, four digits are required. For symbolic and specifying “all,” use 'ugo'.

If an error occurs with the SetPermissions call (for example if the target file has been removed), the error is logged as a Warning and is not used to indicate the Put failed.

SSL Config

Either the name of an existing client SSL/TLS configuration to use to authenticate this connection or !SFTP to use an SFTP server. If you specify the name of an existing SSL/TLS configuration, choose a client rather than a server SSL/TLS configuration, because the adapter initiates the communication. See “SSL Config” in “Settings for the FTP Inbound Adapter.”

Stay Connected

If a positive value, the adapter stays connected to the remote system for this number of seconds after completing an operation. A zero value means to disconnect immediately after every operation. The default of -1 means to stay permanently connected, even during idle times. Adapters are assumed idle at startup and therefore only auto-connect if they are configured with a value of -1.

TempFileName

Name of a temporary file to output the document(s) to. Upon completion of the upload the file with this name to the FTP server it is renamed to the name specified in the Filename setting. If this setting is blank no temporary file will be used. May include timestamp specifiers. The %f specifier, if present, will be replaced with the name of the document's original source filename (stripped of characters illegal in target filenames). See the method **Ens.Util.File.CreateTimestamp()** for documentation of timestamping options.

UsePASV

Use Passive FTP mode: server returns a data port address and the client connects to it. Most firewalls are more tolerant of Passive mode FTP because both the control and data TCP connections are initiated by the client.