



# Using FTP Adapters in Productions

Version 2025.1  
2025-06-03

*Using FTP Adapters in Productions*

PDF generated on 2025-06-03

InterSystems IRIS® Version 2025.1

Copyright © 2025 InterSystems Corporation

All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble®, InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™, HealthShare® Health Connect Cloud™, InterSystems® Data Fabric Studio™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Congress Street, Boston, MA 02114, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

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](mailto:support@InterSystems.com)

# Table of Contents

<b>1 Using the FTP Inbound Adapter .....</b>	<b>1</b>
1.1 Overall Behavior .....	1
1.2 Creating a Business Service to Use the Inbound Adapter .....	2
1.3 Implementing the OnProcessInput() Method .....	2
1.3.1 Invoking Adapter Methods .....	3
1.4 Example Business Service Class .....	4
1.5 Adding and Configuring the Business Service .....	5
1.6 See Also .....	5
<b>2 Using the FTP Outbound Adapter .....</b>	<b>7</b>
2.1 Overall Behavior .....	7
2.2 Creating a Business Operation to Use the Outbound Adapter .....	7
2.3 Creating Message Handler Methods .....	8
2.3.1 Calling Adapter Methods from the Business Operation .....	9
2.4 Example Business Operation Class .....	11
2.5 Adding and Configuring the Business Operation .....	12
2.6 See Also .....	12
<b>FTP Adapter Settings .....</b>	<b>13</b>
Settings for the FTP Inbound Adapter .....	14
Settings for the FTP Outbound Adapter .....	22



# 1

## Using the FTP Inbound Adapter

This page describes how to use the FTP inbound adapter (`EnsLib.FTP.InboundAdapter`), which can provide both FTP and SFTP input to a production.

**Tip:** InterSystems IRIS® data platform 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](#).

### 1.1 Overall Behavior

The `EnsLib.FTP.InboundAdapter` enables a production 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.

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 production 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 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 Messages](#).

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](#) and [Example Business Service Class](#).

## 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 Messages](#).

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](#).

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 *SettingSet* 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 a production credentials entry that can authorize a connection to the FTP server. See [Defining Production Credentials](#).

### 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 TLS configuration entry to use to authenticate this connection.

## 1.4 Example Business Service Class

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

### Class Definition

```
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"),
            tFileLocation=pInput.Attributes("FTPDDir"),
            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
}
}

```

Note that this example sets two variables to capture metadata about the incoming stream, *pInput*:

- *tSource* captures the original file name, which is stored in the *Filename* subscript of the *Attributes* property of the incoming stream
- *tFileLocation* captures the complete original file path, which is stored in the *FTPDir* subscript of the same property

## 1.5 Adding and Configuring the Business Service

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

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

## 1.6 See Also

- Using FTP
- Using SSH
- [Using and debugging %Net.SSH.Session for SSH connections](#) (article on the Developer Community)



# 2

## Using the FTP Outbound Adapter

This page describes how to use the FTP outbound adapter (EnsLib.FTP.OutboundAdapter), which can enable a production to send files via both FTP and SFTP.

**Tip:** InterSystems IRIS® data platform 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](#).

### 2.1 Overall Behavior

EnsLib.FTP.OutboundAdapter enables a 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 Messages](#).

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 a request message class, and *ResponseClass* is the name of a 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 Messages](#).

For information on defining the message handler methods, see [Creating Message Handler Methods](#).

- For other options and general information, see [Defining a Business Operation Class](#).

## 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 production 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 a request message class, and *ResponseClass* is the name of a 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 [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 52x or 4xx, 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`, InterSystems IRIS 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](#).

### 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, InterSystems IRIS executes the corresponding *SettingSet* 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 a production credentials entry that can authorize a connection to the FTP server. See [Defining Production Credentials](#).

### 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 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 Definition

```
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 a production, use the Management Portal to do the following:

1. Add an instance of your business operation class to the production.
2. Configure the business service. For information on the settings, see [Reference for Settings](#).
3. Enable the business operation.
4. Run the production.

## 2.6 See Also

- Using FTP
- Using SSH
- [Using and debugging %Net.SSH.Session for SSH connections](#) (article on the Developer Community)



# FTP Adapter Settings

This section provides reference information on the FTP inbound and outbound adapters. Also see [Settings in All 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	<a href="#">External Registry ID</a> , <a href="#">File Path</a> , <a href="#">Delete From Server</a> , <a href="#">Check Modified Before Delete</a> , <a href="#">File Spec Delimiter</a> , <a href="#">File Spec</a> , <a href="#">Archive Path</a> , <a href="#">Call Interval</a> , <a href="#">FTP Server</a> , <a href="#">FTP Port</a> , <a href="#">Credentials</a>
Connection Settings	<a href="#">Protocol</a> , <a href="#">SSL Configuration</a> , <a href="#">SSL Check Server Identity</a> , <a href="#">SSL Use Session Resumption</a> , <a href="#">UsePASV</a> , <a href="#">Stay Connected</a> , <a href="#">Connect Timeout</a>
FTP Settings	<a href="#">Command Translate Table</a>
SFTP Settings	<a href="#">SFTP Authentication Methods</a> , <a href="#">SFTP Public Key File</a> , <a href="#">SFTP Private Key File</a> , <a href="#">SFTP Passphrase Credentials</a> , <a href="#">SFTP Interactive DTL</a> , <a href="#">SFTP Server Character Set</a> , <a href="#">SFTP Local Character Set</a>
Additional Settings	<a href="#">Use FileStream</a> , <a href="#">Server List Style</a> , <a href="#">Retryable Errors</a> , <a href="#">Subdirectory Levels</a> , <a href="#">Charset</a> , <a href="#">Append Timestamp</a> , <a href="#">Confirm Complete</a> , <a href="#">File Access Timeout</a> , <a href="#">Semaphore Specification</a>

The remaining settings are common to all business services. For information, see [Settings for All Business Services](#).

## 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](#).

## Archive Path

Full path name of the directory on the InterSystems IRIS® 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 InterSystems IRIS machine.

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

**Note:** When you purge message bodies from the system, any files in the Archive Path that are referenced by message bodies are deleted. For more information, see [Purging Production Data](#).

## 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 [Defining Business Services](#).

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

## Charset

Specifies the character set of the input file. InterSystems IRIS 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 InterSystems IRIS 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 InterSystems IRIS
- **@*TranslationTable***, where *TranslationTable* is the name of a translation table (in this case InterSystems IRIS uses the given translation table)

For information on character sets and translation tables, see Translation Tables.

## Check Modified Before Delete

When **Delete From Server** is enabled and **Confirm Complete** is **Size**, this optional flag influences checking whether a file has been modified since initial listing for download and then processing.

The size check captures the latest current modified date time during the download of a file. (Note that the precision on a file modified time is limited to minutes.) The date time is used in combination with the file size to determine whether a file has changed on the FTP server since download, that is, whether the file should be deleted, and should be processed as new content. In the case of large files that are synchronously processed (**Archive Path** is empty), significant time may elapse between the download and attempt to delete the file. Additionally the delete action on the FTP server could be deferred due to network interruption. This setting can be one of the following:

- **No**—Never check whether the file has been modified. Just attempt to delete it.
- **Deferred**—If the deletion is deferred, then check the modified value (Default).
- **Yes**—Always check whether the file has been modified before attempting to delete it.

**Note:** If there are limitations in the FTP server view, where the file modified date is unreliable, then this option should not be enabled, to avoid reprocessing content.

## Fatal Errors

For record map services, determines whether the system stops processing a message when it encounters an error such as a validation error in an individual record. When configuring the adapter, choose one of the following options:

- **Any**—This is the default. If InterSystems IRIS encounters an error when saving an individual record, it stops processing the message.
- **ParseOnly**—If InterSystems IRIS encounters an error when saving an individual record, it logs the error, skips the record, and then continues parsing the message. The log includes the position in the stream of the invalid record. Additionally, if **Alert On Error** is enabled, the system generates an alert.

## Header Count

For record map services, determines the number of lines that the service ignores as prefix lines in incoming documents. Ignoring prefix lines enables the services to parse reports and comma-separated values (CSV) files with column headers.

## 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 InterSystems IRIS 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 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, InterSystems IRIS 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).

## Command Translate Table

Specifies the translate table to use for the command channel, specifically for the filename/pathnames. Normally this should not be specified in which case if the FTP server supports UTF8 then the adapter uses it for the filename/pathnames, if the server does not support UTF8 then the adapter uses RAW mode and just reads the bytes as sent. It is possible for a server to provide the filename list in RAW and for it to support UTF8. In this case it might be necessary to set the Command Translate Table to RAW to override the detected UTF8. The values shown in the list are the internal table names.

## Connect Timeout

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

## Credentials

Identifies a production credentials entry that can authorize a connection to the FTP server. See [Defining Production Credentials](#).

## 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.

**Note:** When the business host, production, or instance is restarted, any previously processed files will be processed again.

## 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](#).

## 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. If [SubdirectoryLevel](#) is greater than 0, this setting cannot be blank and must specify the top-level directory where the search begins.

## File Spec

*Required.* Filenames or wildcard file specifications for file(s) to retrieve from the FTP server. Multiple file specifications can be entered separated by a delimiter. It is necessary to enter the delimiter used in the File Spec Delimiter setting. Enter the FileSpec as one line. Maximum length is 2000 characters. For the wildcard specification, use the convention that is appropriate for the operating system on the FTP server machine.

## File Spec Delimiter

If this is nonempty then it will be used as the delimiter to split the File Specification setting into multiple filename/wildcard searches.

## 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.

## Protocol

Indicates whether to use FTP (File Transfer Protocol) or SFTP (SSH File Transfer Protocol). If the protocol is FTP, it is possible to use the **SSL Configuration** setting to configure FTP over TLS. If the protocol is SFTP, then:

- **UsePASV** and **ServerListStyle** settings are ignored.
- **FTP Port** setting should ordinarily be set to 22.
- You must supply a value for the **Credentials** setting.

- If you supply values for the **SFTPPublicKeyFile** and **SFTPPrivateKeyFile** settings, 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. If you do not use the **KeyFile** settings, the adapter attempts only username/password authentication based on the **Credentials** setting.

This setting specifies whether to use FTP or SFTP. If it is left blank, then FTP is used, *unless* the **SSL Configuration** setting is !SFTP. In this case, SFTP is used irrespective of the **Protocol** setting.

## Retryable Errors

Comma-separated list of error types that should be ignored and that should not cause a file to be skipped in the future.

Example: ErrGeneral,AnotherError.

## Semaphore Specification

The Semaphore Specification allows you to indicate that the data file or files are complete and ready to be read by creating an associated second file in the same directory that is used as a semaphore. The inbound file adapter waits until the semaphore file exists before checking the other conditions specified by the Confirm Complete requirements and then processing the data file or files. 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 files as soon as the conditions specified by the Confirm Complete requirements are met.

If you are using the Semaphore Specification feature, consider setting the Confirm Complete field to None.

### Syntax

Semaphore Specification can be an empty string or can be a series of pairs, each of which associates a data filename specification with a semaphore filename pattern. The pairs are separated by semicolons:

```
DataFileSpec=SemaphorePattern;DataFileSpec=SemaphorePattern;...
```

*DataFileSpec* is either a plain filename or a filename specification that includes the \* wildcard (which matches any character). *SemaphorePattern* directly or indirectly specifies the name of the associated semaphore file; it can be either of the following:

- A plain filename (such as `SemaphoreFile.SEM`). In this case, when the system finds a file that matches *DataFileSpec*, the system looks for a file with that exact name.
- A string of the form `*.extension` such as `*.sem`. In this case, when the system finds a file that matches *DataFileSpec*, the system looks for a file with the same name, but with the `sem` extension instead. For example, if the filename `ABCDEF.txt` matches *DataFileSpec*, the system looks for a semaphore file named `ABCDEF.sem`

When looking for a semaphore file based on a data filename, the system looks only at the part of the data filename before the first period. For example, if the filename `test.txt.data.zip.tar` matches *DataFileSpec*, if the semaphore filename pattern is `*.sem` the system looks for a semaphore file named `test.sem`

Notes:

- The semaphore file associated with a given data file (or multiple data files) must be in the same directory as those files.
- *DataFileSpec* and *SemaphorePattern* do not include the directory name.
- *DataFileSpec* is always case-sensitive.
- *SemaphorePattern* is case-sensitive if the operating system is case-sensitive and is not case-sensitive otherwise.
- The pairs are processed left-to-right, and the first matching pair is used; see [How the Inbound File Adapter Uses Semaphore 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.

- If an adapter configured with a FileSpec equal to \*, the adapter usually considers all files in the directory as data files. But if Semaphore Specification is also specified, the adapter can recognize a file as a semaphore file and not treat it as a data file.

For example, consider the following Semaphore Specification, consisting of a single pair:

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

In this case, 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.

In the simplest case, Semaphore Specification can consist only of a single *SemaphoreFileSpec*, which may or may not include a wildcard. This means that the presence or absence of that semaphore file controls whether the adapter processes any files.

### ***Files That Do Not Match a Pattern***

If a Semaphore Specification is specified and a given data file does not match any of the patterns, then the adapter will not process this data file. If this is undesirable, specify a final pair that will match any file and that uses its own semaphore file. For example, consider this Semaphore Specification:

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

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

### ***How the Inbound File Adapter Uses Semaphore Specification***

Within each polling cycle, the inbound file adapter examines all the files found within the configured directory (and any subdirectories). Then for each file:

1. The adapter reads the Semaphore Specification from left-to-right, finding the first specification whose *DataFileSpec* matches the given filename. This indicates the name of the semaphore file to look for.
2. The adapter looks for the semaphore file in the same directory as the file being examined.

Then:

- If it does not find the semaphore file, the adapter skips the file and sets an internal flag that causes the adapter to wait until the next polling cycle.
- If it does find the semaphore file, the adapter processes the file.

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

## **SFTP Authentication Methods**

AuthenticationMethods supported:

- Empty—use public/private key if defined, otherwise use username and password from Credentials.
- p—use username and password from Credentials.
- k—use public/private key.
- i—use Interactive (Challenge/Response).

If you specify multiple flags combine in the order required. For example kp leads to public/private key authentication first followed by username and password.

## SFTP Interactive DTL

If specified, this is the DTL to used to handle the Interactive Authentication—also known as the keyboard-interactive or Challenge/Response authentication. This can be a DTL to create an array of answers for the challenge response authentication. See `Ens.SSH.InteractiveAuth.DTL` for more information. Leave blank to return just Credentials password.

## SFTP Passphrase Credentials

This can be used to specify different credentials entry from which the password will be used as the Key authentication Passphrase. If this is blank, the Credentials setting will be used. If this is not blank, then for the Key authentication Passphrase, the system will use the username and password from the given entry.

This separate setting allows having both Public Key and Password authentication.

See [Defining Production Credentials](#).

## SFTP Private Key File

File path to a file containing the SSH private key. Private keys should be PEM encoded. You need both a private and public key for use with `AuthenticateWithKeyPair`. See `OpenSSH` and `PEM-Encoded Keys`.

## SFTP Public Key File

File path to a file containing the SSH public key. Public keys should be in `OpenSSH` format. You need both a private and public key for use with `AuthenticateWithKeyPair`. See `OpenSSH` and `PEM-Encoded Keys`.

## SFTP Server Character Set

Character set used by the remote system to encode filenames. The default value is `UTF8`. You can set the property to `" "` (an empty string) to indicate that no character set translation is required for filenames. This setting controls the `RemoteCharset` property of the `%Net.SSH.Session` object that represents the relevant SSH session.

## SFTP Local Character Set

Character set used by the local system to encode filenames. The default value for Windows systems is `" "` (an empty string). The default value for UNIX systems is `UTF8`. You can set the property to `" "` (an empty string) to indicate that no character set translation is required for filenames. This setting controls the `LocalCharset` property of the `%Net.SSH.Session` object that represents the relevant SSH session.

## STFP Key Exchange Algorithm

Specifies the preferred SSH Key Exchange (KEX) algorithm. Available options depend on the host operating system.

## 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

## SSL Config

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

To create and manage TLS configurations, use the Management Portal. See *InterSystems TLS Guide*. The first field on the **Edit SSL/TLS Configuration** page is **Configuration Name**. Use this string as the value for the **SSLConfig** setting.



You can use the [Protocol](#) setting to use SFTP.

## SSL Check Server Identity

When connecting to an FTP server via TLS, the server name in the certificate must match the DNS name used to connect to the server. This match is based on the rules in section 3.1 of RFC 2818.

## SSL Use Session Resumption

If this setting is enabled, then when making the SSL connection for the data channel, the adapter reuses session parameters from the command channel. This option requires OpenSSL v1.1.x+.

## 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. If the value is greater than `0`, the **FilePath** setting must specify the top-level directory where the search begins.

## 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	<a href="#">External Registry ID</a> , <a href="#">FTP Server</a> , <a href="#">FTP Port</a> , <a href="#">Credentials</a> , <a href="#">File Path</a>
SFTP Settings	<a href="#">SFTP Append Mode</a> , <a href="#">SFTP Authentication Methods</a> , <a href="#">SFTP Public Key File</a> , <a href="#">SFTP Private Key File</a> , <a href="#">SFTP Server Character Set</a> , <a href="#">SFTP Local Character Set</a> , <a href="#">SFTP Passphrase Credentials</a> , <a href="#">SFTP Interactive DTL</a> , <a href="#">SFTP File Access Mode</a>
Connection Settings	<a href="#">Protocol</a> , <a href="#">SSL Configuration</a> , <a href="#">SSL Check Server Identity</a> , <a href="#">UsePASV</a> , <a href="#">Stay Connected</a> , <a href="#">Connect Timeout</a>
FTP	<a href="#">Command Translate Table</a>
Additional Settings	<a href="#">Overwrite</a> , <a href="#">Charset</a>

The remaining settings are common to all business operations. For information, see [Settings for All Business Operations](#).

## Charset

Specifies the desired character set for the output file. InterSystems IRIS® automatically translates the characters to this character encoding. See [Charset](#).

## Command Translate Table

Specifies the translate table to use for the command channel, specifically for the filename/pathnames. Normally this should not be specified in which case if the FTP server supports UTF8 then the adapter uses it for the filename/pathnames, if the server does not support UTF8 then the adapter uses RAW mode and just reads the bytes as sent. It is possible for a server to provide the filename list in RAW and for it to support UTF8. In this case it might be necessary to set the Command Translate Table to RAW to override the detected UTF8. The values shown in the list are the internal table names.

## Connect Timeout

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

## Credentials

Identifies a production credentials entry that can authorize a connection to the FTP server. See [Credentials](#).

## 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.

## Protocol

Indicates whether to use FTP (File Transfer Protocol) or SFTP (SSH File Transfer Protocol). If the protocol is FTP, it is possible to use the **SSL Configuration** setting to configure FTP over TLS. If the protocol is SFTP, then:

- **UsePASV** and **ServerListStyle** settings are ignored.
- **FTP Port** setting should ordinarily be set to 22.
- You must supply a value for the **Credentials** setting.
- If you supply values for the **SFTPPublicKeyFile** and **SFTPPrivateKeyFile** settings, 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. If you do not use the **KeyFile** settings, the adapter attempts only username/password authentication based on the **Credentials** setting.

This setting specifies whether to use FTP or SFTP. If it is left blank, then FTP is used, *unless* the **SSL Configuration** setting is **!SFTP**. In this case, SFTP is used irrespective of the **Protocol** setting.

## 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.

## 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.

## SFTP Private Key File

File path to a file containing the SSH private key certificate. Private keys should be PEM encoded. You need both a private and public key for use with AuthenticateWithKeyPair. See OpenSSH and PEM-Encoded Keys.

## SFTP Public Key File

File path to a file containing the SSH public key certificate. Public keys should be in OpenSSH format. You need both a private and public key for use with `AuthenticateWithKeyPair`. See [OpenSSH](#) and [PEM-Encoded Keys](#).

## SSL Config

The name of an existing client TLS configuration to use to authenticate this connection. Choose a client rather than a server TLS configuration, because the adapter initiates the communication. See [SSL Config](#) in [Settings for the FTP Inbound Adapter](#).

For prior version compatibility, a value of `!SFTP` indicates the protocol is SFTP. However, the recommendation is to use the [Protocol](#) setting to use SFTP.

## SSL Check Server Identity

When connecting to an FTP server via TLS, the server name in the certificate must match the DNS name used to connect to the server. This match is based on the rules in section 3.1 of RFC 2818.

## SSL Use Session Resumption

If this setting is enabled, then when making the SSL connection for the data channel, the adapter reuses session parameters from the command channel. This option requires OpenSSL v1.1.x+.

## 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.

## 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.

## Use SetPermissions Call

Applicable when the adapters uses `Put` to write a file to the server; in this case, the SFTP server may apply a umask. A separate `SetPermissions` call that might not be modified by a umask can be made to apply the requested permissions. The default is to use an extra `SetPermissions` call. If the SFTP File Access Mode is 0600 a `SetPermissions` call will not be made irrespective of this setting. Note - this call may not succeed if the target file has already been collected on the server. A Warning will be issued if the `SetPermissions` call errors.