FTP2

FTP/SFTP/FTPS Component

This component provides access to remote file systems over the FTP and SFTP protocols.

Maven users will need to add the following dependency to their `pom.xml` for this component:

xml<dependency> <groupId>org.apache.camel</groupId> <artifactId>camel-ftp</artifactId> <version>x.x.x</version>See the documentation of the Apache Commons <!-- use the same version as your Camel core version --> </dependency>

More options
See File for more options as all the options from File is inherited.
Absolute paths
Absolute path is **not** supported.

Camel 2.16 will translate absolute paths to relative ones by trimming all leading slashes from `directoryname`. There'll be WARN message printed in the logs.
Consuming from remote FTP server
Make sure you read the section titled *Default when consuming files* further below for details related to consuming files.

URI format

ftp://[username@]hostname[:port]/directoryname[?options] sftp://[username@]hostname[:port]/directoryname[?options] ftps://[username@]hostname[:port]/directoryname[?options]

Where **directoryname** represents the underlying directory. The directory name is a relative path. Absolute paths are **not** supported. The relative path can contain nested folders, such as /inbox/us.

For Camel versions before **Camel 2.16**, the directoryName **must** exist already as this component does not support the `autoCreate` option (which the file component does). The reason is that its the FTP administrator (FTP server) task to properly setup user accounts, and home directories with the right file permissions etc.

For **Camel 2.16**, `autoCreate` option is supported. When consumer starts, before polling is scheduled, there's additional FTP operation performed to create the directory configured for endpoint. The default value for `autoCreate` is `true`.

If no **username** is provided, then `anonymous` login is attempted using no password.
If no **port** number is provided, Camel will provide default values according to the protocol (ftp = 21, sftp = 22, ftps = 2222).

You can append query options to the URI in the following format, `?option=value&option=value&...`

This component uses two different libraries for the actual FTP work. FTP and FTPS uses Apache Commons Net while SFTP uses JCraft JSCH.

The FTPS component is only available in Camel 2.2 or newer.
FTPS (also known as FTP Secure) is an extension to FTP that adds support for the Transport Layer Security (TLS) and the Secure Sockets Layer (SSL) cryptographic protocols.

URI Options

The options below are exclusive for the FTP component.

More options
See File for more options as all the options from File is inherited.
confluenceTableSmall

Name	Default Value	Description
`username`	`null`	Specifies the username to use to log in to the remote file systen.
`password`	`null`	Specifies the password to use to log in to the remote file system.
`account`	`null`	**Camel 2.15.2:** Specified the account to use to login to the remote FTP server (only for FTP and FTP Secure)
`binary`	`false`	Specifies the file transfer mode, BINARY or ASCII. Default is ASCII (`false`).
`disconnect`	`false`	**Camel 2.2:** Whether or not to disconnect from remote FTP server right after use. Can be used for both consumer and producer. Disconnect will only disconnect the current connection to the FTP server. If you have a consumer which you want to stop, then you need to stop the consumer/route instead.

`localWorkDirectory`	`null`	When consuming, a local work directory can be used to store the remote file content directly in local files, to avoid loading the content into memory. This is beneficial, if you consume a very big remote file and thus can conserve memory. See below for more details.
`passiveMode`	`false`	**FTP and FTPS only**: Specifies whether to use passive mode connections. Default is active mode (`false`).
`securityProtocol`	`TLS`	**FTPS only:** Sets the underlying security protocol. The following values are defined: `TLS`: Transport Layer Security `SSL`: Secure Sockets Layer
`disableSecureDataChannelDefaults`	`false`	**Camel 2.4: FTPS only**: Whether or not to disable using default values for `execPbsz` and `execProt` when using secure data transfer. You can set this option to `true` if you want to be in full control what the options `execPbsz` and `execProt` should be used.
`download`	`true`	**Camel 2.11:** Whether the FTP consumer should download the file. If this option is set to `false`, then the message body will be `null`, but the consumer will still trigger a Camel Exchange that has details about the file such as file name, file size, etc. It's just that the file will not be downloaded.
streamDownload	false	**Camel 2.11:** Whether the consumer should download the entire file up front, the default behavior, or if it should pass an InputStream read from the remote resource rather than an in-memory array as the in body of the Camel Exchange. This option is ignored if download is false or is localWorkDirectory is provided. This option is useful for working with large remote files.
`execProt`	`null`	**Camel 2.4: FTPS only**: Will by default use option `P` if secure data channel defaults hasn't been disabled. Possible values are: `C`: Clear `S`: Safe (SSL protocol only) `E`: Confidential (SSL protocol only) `P`: Private
`execPbsz`	`null`	**Camel 2.4: FTPS only**: This option specifies the buffer size of the secure data channel. If option `useSecureDataChannel` has been enabled and this option has not been explicit set, then value `0` is used.
`isImplicit`	`false`	**FTPS only:** Sets the security mode(implicit/explicit). Default is explicit (`false`).
`knownHostsFile`	`null`	**SFTP only:** Sets the `known_hosts` file, so that the SFTP endpoint can do host key verification.
`useUserKnownHostsFile`	`true`	**SFTP onlly: Camel 2.18:** If knownHostFile has not been explicit configured then use the host file from System.getProperty (user.home)/.ssh/known_hosts
`knownHostsUri`	`null`	**SFTP only: Camel 2.11.1:** Sets the `known_hosts` file (loaded from classpath by default), so that the SFTP endpoint can do host key verification.
`keyPair`	`null`	**SFTP only: Camel 2.12.0:** Sets the Java KeyPair for SSH public key authentication, it supports DSA or RSA keys.
`privateKeyFile`	`null`	**SFTP only:** Set the private key file to that the SFTP endpoint can do private key verification.
`privateKeyUri`	`null`	**SFTP only: Camel 2.11.1:** Set the private key file (loaded from classpath by default) to that the SFTP endpoint can do private key verification.
`privateKey`	`null`	**SFTP only: Camel 2.11.1:** Set the private key as byte[] to that the SFTP endpoint can do private key verification.
`privateKeyFilePassphrase`	`null`	**SFTP only: Deprecated:** use `privateKeyPassphrase` instead. Set the private key file passphrase to that the SFTP endpoint can do private key verification.
`privateKeyPassphrase`	`null`	**SFTP only: Camel 2.11.1:** Set the private key file passphrase to that the SFTP endpoint can do private key verification.
`preferredAuthentications`	`null`	**SFTP only: Camel 2.10.7, 2.11.2,2.12.0:** set the preferred authentications which SFTP endpoint will used. Some example include:password,publickey. If not specified the default list from JSCH will be used.
`ciphers`	`null`	**Camel 2.8.2, 2.9: SFTP only** Set a comma separated list of ciphers that will be used in order of preference. Possible cipher names are defined by JCraft JSCH. Some examples include: aes128-ctr,aes128-cbc,3des-ctr,3des-cbc,blowfish-cbc,aes192-cbc,aes256-cbc. If not specified the default list from JSCH will be used.
`fastExistsCheck`	`false`	**Camel 2.8.2, 2.9:** If set this option to be true, camel-ftp will use the list file directly to check if the file exists. Since some FTP server may not support to list the file directly, if the option is false, camel-ftp will use the old way to list the directory and check if the file exists. Note from **Camel 2.10.1** onwards this option also influences `readLock=changed` to control whether it performs a fast check to update file information or not. This can be used to speed up the process if the FTP server has a lot of files.

`strictHostKeyChecking`	`no`	**SFTP only: Camel 2.2:** Sets whether to use strict host key checking. Possible values are: `no`, `yes` and `ask`. `ask` does not make sense to use as Camel cannot answer the question for you as its meant for human intervention. **Note:** The default in Camel 2.1 and below was `ask`.
`maximumReconnectAttempts`	`3`	Specifies the maximum reconnect attempts Camel performs when it tries to connect to the remote FTP server. Use 0 to disable this behavior.
`reconnectDelay`	`1000`	Delay in millis Camel will wait before performing a reconnect attempt.
`connectTimeout`	`10000`	**Camel 2.4:** Is the connect timeout in millis. This corresponds to using `ftpClient.connectTimeout` for the FTP/FTPS. For SFTP this option is also used when attempting to connect.
`soTimeout`	`null / 30000`	**FTP and FTPS Only: Camel 2.4:** Is the `SocketOptions.SO_TIMEOUT` value in millis. A good idea is to configure this to a value such as 300000 (5 minutes) to not hang a connection. On SFTP this option is set as timeout on the JSCH Session instance. Also SFTP from **Camel 2.14.3/2.15.3/2.16** onwards. From **Camel 2.16** onwards the default is 300000 (300 sec).
`timeout`	`30000`	**FTP and FTPS Only: Camel 2.4:** Is the data timeout in millis. This corresponds to using `ftpClient.dataTimeout` for the FTP/FTPS. For SFTP there is no data timeout.
`throwExceptionOnConnectFailed`	`false`	**Camel 2.5:** Whether or not to thrown an exception if a successful connection and login could not be establish. This allows a custom `pollStrategy` to deal with the exception, for example to stop the consumer or the likes.
`siteCommand`	`null`	**FTP and FTPS Only: Camel 2.5:** To execute site commands after successful login. Multiple site commands can be separated using a new line character (\n). Use `help site` to see which site commands your FTP server supports.
`stepwise`	`true`	**Camel 2.6:** Whether or not stepwise traversing directories should be used or not. Stepwise means that it will CD one directory at a time. See more details below. You can disable this in case you can't use this approach.
`separator`	`UNIX`	**Camel 2.6:** Dictates what path separator char to use when uploading files. `Auto` = Use the path provided without altering it. `UNIX` = Use unix style path separators. `Windows` = Use Windows style path separators. Since **Camel 2.15.2**: The default value is changed to UNIX style path, before **Camel 2.15.2**: The default value is `Auto`.
`chmod`	`null`	**SFTP Producer Only: Camel 2.9:** Allows you to set chmod on the stored file. For example `chmod=640`.
`compression`	`0`	**SFTP Only: Camel 2.8.3/2.9:** To use compression. Specify a level from 1 to 10. **Important:** You must manually add the needed JSCH zlib JAR to the classpath for compression support.
`receiveBufferSize`	`32768`	**FTP/FTPS Only: Camel 2.15.1:** The buffer size for downloading files. The default size is 32kb.
`ftpClient`	`null`	**FTP and FTPS Only: Camel 2.1:** Allows you to use a custom `org.apache.commons.net.ftp.FTPClient` instance.
`ftpClientConfig`	`null`	**FTP and FTPS Only: Camel 2.1:** Allows you to use a custom `org.apache.commons.net.ftp.FTPClientConfig` instance.
`ftpClientConfig.XXX`		**FTP and FTPS Only:** To configure various options on the FTPClient instance from the uri. For example: `ftpClientConfig.receiveDataSocketBufferSize=8192&ftpClientConfig.sendDataSocketBufferSize=8192`
`serverAliveInterval`	`0`	**SFTP Only: Camel 2.8** Allows you to set the serverAliveInterval of the sftp session
`serverAliveCountMax`	`1`	**SFTP Only: Camel 2.8** Allows you to set the serverAliveCountMax of the sftp session
`ftpClient.trustStore.file`	`null`	**FTPS Only:** Sets the trust store file, so that the FTPS client can look up for trusted certificates.
`ftpClient.trustStore.type`	`JKS`	**FTPS Only:** Sets the trust store type.

`ftpClient.trustStore.algorithm`	`SunX509`	**FTPS Only:** Sets the trust store algorithm.
`ftpClient.trustStore.password`	`null`	**FTPS Only:** Sets the trust store password.
`ftpClient.keyStore.file`	`null`	**FTPS Only:** Sets the key store file, so that the FTPS client can look up for the private certificate.
`ftpClient.keyStore.type`	`JKS`	**FTPS Only:** Sets the key store type.
`ftpClient.keyStore.algorithm`	`SunX509`	**FTPS Only:** Sets the key store algorithm.
`ftpClient.keyStore.password`	`null`	**FTPS Only:** Sets the key store password.
`ftpClient.keyStore.keyPassword`	`null`	**FTPS Only:** Sets the private key password.
`sslContextParameters`	`null`	**FTPS Only: Camel 2.9:** Reference to a `org.apache.camel.util.jsse.SSLContextParameters` in the Registry. This reference overrides any configured SSL related options on ftpClient as well as the securityProtocol (SSL, TLS, etc.) set on FtpsConfiguration. See Using the JSSE Configuration Utility.
`proxy`	`null`	**SFTP Only: Camel 2.10.7, 2.11.1:** Reference to a `com.jcraft.jsch.Proxy` in the Registry. This proxy is used to consume/send messages from the target SFTP host.
`useList`	`true`	**FTP/FTPS Only: Camel 2.12.1:** Whether the consumer should use FTP LIST command to retrieve directory listing to see which files exists. If this option is set to `false`, then `stepwise=false` must be configured, and also `fileName` must be configured to a fixed name, so the consumer knows the name of the file to retrieve. When doing this only that single file can be retrieved. See further below for more details.
`ignoreFileNotFoundOrPermissionError`	`false`	**Camel 2.12.1:** Whether the consumer should ignore when a file was attempted to be retrieved but did not exist (for some reason), or failure due insufficient file permission error. **Camel 2.14.2:** This option now applies to directories as well.
`sendNoop`	`true`	**Camel 2.16:** Producer only. Whether to send a noop command as a pre-write check before uploading files to the FTP server. This is enabled by default as a validation of the connection is still valid, which allows to silently re-connect to be able to upload the file. However if this causes problems, you can turn this option off.
`jschLoggingLevel`	`WARN`	**SFTP Only: Camel 2.15.3/2.16:** The logging level to use for JSCH activity logging. As JSCH is verbose at by default at INFO level the threshold is WARN by default.
`bulkRequest`		**SFTP Only: Camel 2.17.1:** Specifies how many requests may be outstanding at any one time. Increasing this value may slightly improve file transfer speed but will increase memory usage.
`disconnectOnBatchComplete`	false	**Camel 2.18:** Whether or not to disconnect from remote FTP server after a Batch is complete. Can be used for both consumer and producer. Disconnect will only disconnect the current connection to the FTP server. If you have a consumer which you want to stop, then you need to stop the consumer/route instead.
`activePortRange`		**Camel 2.18:** Set the client side port range in active mode. The syntax is: minPort-maxPort. Both port numbers are inclusive, eg 10000-19999 to include all 1xxxx ports.

FTPS component default trust store

When using the `ftpClient.` properties related to SSL with the FTPS component, the trust store accepts all certificates. If you only want trust selective certificates, you have to configure the trust store with the `ftpClient.trustStore.xxx` options or by configuring a custom `ftpClient`.

When using `sslContextParameters`, the trust store is managed by the configuration of the provided SSLContextParameters instance.

You can configure additional options on the `ftpClient` and `ftpClientConfig` from the URI directly by using the `ftpClient.` or `ftpClientConfig.` prefix.

For example to set the `setDataTimeout` on the `FTPClient` to 30 seconds you can do:

from("ftp://foo@myserver?password=secret&ftpClient.dataTimeout=30000").to("bean:foo");

You can mix and match and have use both prefixes, for example to configure date format or timezones.

from("ftp://foo@myserver?password=secret&ftpClient.dataTimeout=30000&ftpClientConfig.serverLanguageCode=fr").to("bean:foo");

You can have as many of these options as you like.

See the documentation of the Apache Commons FTP FTPClientConfig for possible options and more details. And as well for Apache Commons FTP FTPClient.

If you do not like having many and long configuration in the url you can refer to the `ftpClient` or `ftpClientConfig` to use by letting Camel lookup in the Registry for it.

For example:

<bean id="myConfig" class="org.apache.commons.net.ftp.FTPClientConfig"> <property name="lenientFutureDates" value="true"/> <property name=" serverLanguageCode" value="fr"/> </bean>

And then let Camel lookup this bean when you use the # notation in the url.

from("ftp://foo@myserver?password=secret&ftpClientConfig=#myConfig").to("bean:foo");

More URI options

title:More options
See File2 as all the options there also applies for this component.

Examples

```
ftp://someone@someftpserver.com/public/upload/images/holiday2008?password=secret&binary=true
ftp://someoneelse@someotherftpserver.co.uk:12049/reports/2008/password=secret&binary=false
ftp://publicftpserver.com/download
```

FTP Consumer does not support concurrency
The FTP consumer (with the same endpoint) does not support concurrency (the backing FTP client is not thread safe).
You can use multiple FTP consumers to poll from different endpoints. It is only a single endpoint that does not support concurrent consumers.

The FTP producer does **not** have this issue, it supports concurrency.
More information
This component is an extension of the File component. So there are more samples and details on the File component page.

Default when consuming files

The FTP consumer will by default leave the consumed files untouched on the remote FTP server. You have to configure it explicitly if you want it to delete the files or move them to another location. For example you can use `delete=true` to delete the files, or use `move=.done` to move the files into a hidden done sub directory.

The regular File consumer is different as it will by default move files to a `.camel` sub directory. The reason Camel does **not** do this by default for the FTP consumer is that it may lack permissions by default to be able to move or delete files.

limitations

The option **readLock** can be used to force Camel **not** to consume files that are currently being written. However, this option is turned off by default, as it requires that the user has write access. See the options table at File2 for more details about read locks.
There are other solutions to avoid consuming files that are currently being written over FTP; for instance, you can write to a temporary destination and move the file after it has been written.

When moving files using `move` or `preMove` option the files are restricted to the FTP_ROOT folder. That prevents you from moving files outside the FTP area. If you want to move files to another area you can use soft links and move files into a soft linked folder.

Message Headers

The following message headers can be used to affect the behavior of the component

confluenceTableSmall

Header	Description

`CamelFileName`	Specifies the output file name (relative to the endpoint directory) to be used for the output message when sending to the endpoint. If this is not present and no expression either, then a generated message ID is used as the filename instead.
`CamelFileNameProduced`	The actual filepath (path + name) for the output file that was written. This header is set by Camel and its purpose is providing end-users the name of the file that was written.
`CamelBatchIndex`	Current index out of total number of files being consumed in this batch.
`CamelBatchSize`	Total number of files being consumed in this batch.
`CamelFileHost`	The remote hostname.
`CamelFileLocalWorkPath`	Path to the local work file, if local work directory is used.

In addition the FTP/FTPS consumer and producer will enrich the Camel `Message` with the following headers

confluenceTableSmall

Header	Description
`CamelFtpReplyCode`	**Camel 2.11.1:** The FTP client reply code (the type is a integer)
`CamelFtpReplyString`	**Camel 2.11.1:** The FTP client reply string

About timeouts

The two set of libraries (see top) have different APIs for setting timeout. You can use the `connectTimeout` option for both of them to set a timeout in millis to establish a network connection. An individual `soTimeout` can also be set on the FTP/FTPS, which corresponds to using `ftpClient.soTimeout`. Notice SFTP will automatically use `connectTimeout` as its `soTimeout`. The `timeout` option only applies for FTP/FTSP as the data timeout, which corresponds to the `ftpClient.dataTimeout` value. All timeout values are in millis.

Using Local Work Directory

Camel supports consuming from remote FTP servers and downloading the files directly into a local work directory. This avoids reading the entire remote file content into memory as it is streamed directly into the local file using `FileOutputStream`.

Camel will store to a local file with the same name as the remote file, though with `.inprogress` as extension while the file is being downloaded. Afterwards, the file is renamed to remove the `.inprogress` suffix. And finally, when the Exchange is complete the local file is deleted.

So if you want to download files from a remote FTP server and store it as files then you need to route to a file endpoint such as:

javafrom("ftp://someone@someserver.com?password=secret&localWorkDirectory=/tmp").to("file://inbox"); Optimization by renaming work file
The route above is ultra efficient as it avoids reading the entire file content into memory. It will download the remote file directly to a local file stream. The `java.io.File` handle is then used as the Exchange body. The file producer leverages this fact and can work directly on the work file `java.io.File` handle and perform a `java.io.File.rename` to the target filename. As Camel knows it's a local work file, it can optimize and use a rename instead of a file copy, as the work file is meant to be deleted anyway.

Stepwise changing directories

Camel FTP can operate in two modes in terms of traversing directories when consuming files (eg downloading) or producing files (eg uploading)

- stepwise
- not stepwise

You may want to pick either one depending on your situation and security issues. Some Camel end users can only download files if they use stepwise, while others can only download if they do not. At least you have the choice to pick (from Camel 2.6 onwards).

In Camel 2.0 - 2.5 there is only one mode and it is:

- before 2.5 not stepwise
- 2.5 stepwise

From Camel 2.6 onwards there is now an option `stepwise` you can use to control the behavior.

Note that stepwise changing of directory will in most cases only work when the user is confined to it's home directory and when the home directory is reported as `"/"`.

The difference between the two of them is best illustrated with an example. Suppose we have the following directory structure on the remote FTP server we need to traverse and download files:

/ /one /one/two /one/two/sub-a /one/two/sub-b

And that we have a file in each of sub-a (a.txt) and sub-b (b.txt) folder.

Using stepwise=true (default mode)

TYPE A 200 Type set to A PWD 257 "/" is current directory. CWD one 250 CWD successful. "/one" is current directory. CWD two 250 CWD successful. "/one/two" is current directory. SYST 215 UNIX emulated by FileZilla PORT 127,0,0,1,17,94 200 Port command successful LIST 150 Opening data channel for directory list. 226 Transfer OK CWD sub-a 250 CWD successful. "/one/two/sub-a" is current directory. PORT 127,0,0,1,17,95 200 Port command successful LIST 150 Opening data channel for directory list. 226 Transfer OK CDUP 200 CDUP successful. "/one/two" is current directory. CWD sub-b 250 CWD successful. "/one/two/sub-b" is current directory. PORT 127,0,0,1,17,96 200 Port command successful LIST 150 Opening data channel for directory list. 226 Transfer OK CDUP 200 CDUP successful. "/one/two" is current directory. CWD / 250 CWD successful. "/" is current directory. PWD 257 "/" is current directory. CWD one 250 CWD successful. "/one" is current directory. CWD two 250 CWD successful. "/one/two" is current directory. PORT 127,0,0,1,17,97 200 Port command successful RETR foo.txt 150 Opening data channel for file transfer. 226 Transfer OK CWD / 250 CWD successful. "/" is current directory. PWD 257 "/" is current directory. CWD one 250 CWD successful. "/one" is current directory. CWD two 250 CWD successful. "/one/two" is current directory. CWD sub-a 250 CWD successful. "/one/two/sub-a" is current directory. PORT 127,0,0,1,17,98 200 Port command successful RETR a.txt 150 Opening data channel for file transfer. 226 Transfer OK CWD / 250 CWD successful. "/" is current directory. PWD 257 "/" is current directory. CWD one 250 CWD successful. "/one" is current directory. CWD two 250 CWD successful. "/one/two" is current directory. CWD sub-b 250 CWD successful. "/one/two/sub-b" is current directory. PORT 127,0,0,1,17,99 200 Port command successful RETR b.txt 150 Opening data channel for file transfer. 226 Transfer OK CWD / 250 CWD successful. "/" is current directory. QUIT 221 Goodbye disconnected.

As you can see when stepwise is enabled, it will traverse the directory structure using CD xxx.

Using stepwise=false

230 Logged on TYPE A 200 Type set to A SYST 215 UNIX emulated by FileZilla PORT 127,0,0,1,4,122 200 Port command successful LIST one/two 150 Opening data channel for directory list 226 Transfer OK PORT 127,0,0,1,4,123 200 Port command successful LIST one/two/sub-a 150 Opening data channel for directory list 226 Transfer OK PORT 127,0,0,1,4,124 200 Port command successful LIST one/two/sub-b 150 Opening data channel for directory list 226 Transfer OK PORT 127,0,0,1,4,125 200 Port command successful RETR one/two/foo.txt 150 Opening data channel for file transfer. 226 Transfer OK PORT 127,0,0,1,4,126 200 Port command successful RETR one/two/sub-a/a.txt 150 Opening data channel for file transfer. 226 Transfer OK PORT 127,0,0,1,4,127 200 Port command successful RETR one/two/sub-b/b.txt 150 Opening data channel for file transfer. 226 Transfer OK QUIT 221 Goodbye disconnected.

As you can see when not using stepwise, there are no CD operation invoked at all.

Samples

In the sample below we set up Camel to download all the reports from the FTP server once every hour (60 min) as BINARY content and store it as files on the local file system.{snippet:id=e1|lang=java|url=camel/trunk/components/camel-ftp/src/test/java/org/apache/camel/component/file/remote/FromFtpToBinarySampleTest.java}And the route using Spring DSL:

xml <route> <from uri="ftp://scott@localhost/public/reports?password=tiger&binary=true&delay=60000"/> <to uri="file://target/test-reports"/> </route>

Consuming a remote FTPS server (implicit SSL) and client authentication

from("ftps://admin@localhost:2222/public/camel?password=admin&securityProtocol=SSL&isImplicit=true &ftpClient.keyStore.file=./src/test/resources/server.jks &ftpClient.keyStore.password=password&ftpClient.keyStore.keyPassword=password") .to("bean:foo");

Consuming a remote FTPS server (explicit TLS) and a custom trust store configuration

from("ftps://admin@localhost:2222/public/camel?password=admin&ftpClient.trustStore.file=./src/test/resources/server.jks&ftpClient.trustStore.password=password") .to("bean:foo");

Filter using `org.apache.camel.component.file.GenericFileFilter`

Camel supports pluggable filtering strategies. This strategy can be provided by implementing `org.apache.camel.component.file.GenericFileFilter` in Java. You can then configure the endpoint with such a filter to skip certain filters before being processed.

In the sample we have built our own filter that only accepts files starting with report in the filename.{snippet:id=e1|lang=java|url=camel/trunk/components/camel-ftp/src/test/java/org/apache/camel/component/file/remote/FromFtpRemoteFileFilterTest.java}And then we can configure our route using the **filter** attribute to reference our filter (using # notation) that we have defined in the spring XML file:

xml <!-- define our sorter as a plain spring bean --> <bean id="myFilter" class="com.mycompany.MyFileFilter"/> <route> <from uri="ftp://someuser@someftpserver.com?password=secret&filter=#myFilter"/> <to uri="bean:processInbox"/> </route>

Filtering using ANT path matcher

The ANT path matcher is a filter that is shipped out-of-the-box in the **camel-spring** jar. So you need to depend on **camel-spring** if you are using Maven. The reason is that we leverage Spring's AntPathMatcher to do the actual matching.

The file paths are matched with the following rules:

- ? matches one character
- * matches zero or more characters
- ** matches zero or more directories in a path

The sample below demonstrates how to use it:{snippet:id=example|lang=xml|url=camel/trunk/tests/camel-itest/src/test/resources/org/apache/camel/itest/ftp/SpringFileAntPathMatcherRemoteFileFilterTest-context.xml}

Using a proxy with SFTP

To use an HTTP proxy to connect to your remote host, you can configure your route in the following way:

xml<!-- define our sorter as a plain spring bean --> <bean id="proxy" class="com.jcraft.jsch.ProxyHTTP"> <constructor-arg value="localhost"/> <constructor-arg value="7777"/> </bean> <route> <from uri="sftp://localhost:9999/root?username=admin&password=admin&proxy=#proxy"/> <to uri=" bean:processFile"/> </route>

You can also assign a user name and password to the proxy, if necessary. Please consult the documentation for `com.jcraft.jsch.Proxy` to discover all options.

Setting preferred SFTP authentication method

If you want to explicitly specify the list of authentication methods that should be used by `sftp` component, use `preferredAuthentications` option. If for example you would like Camel to attempt to authenticate with private/public SSH key and fallback to user/password authentication in the case when no public key is available, use the following route configuration:

from("sftp://localhost:9999/root?username=admin&password=admin&preferredAuthentications=publickey,password"). to("bean:processFile");

Consuming a single file using a fixed name

When you want to download a single file and know the file name, you can use `fileName=myFileName.txt` to tell Camel the name of the file to download. By default the consumer will still do a FTP LIST command to do a directory listing and then filter these files based on the `fileName` option. Though in this use-case it may be desirable to turn off the directory listing by setting `useList=false`. For example the user account used to login to the FTP server may not have permission to do a FTP LIST command. So you can turn off this with `useList=false`, and then provide the fixed name of the file to download with `fileName=myFileName.txt`, then the FTP consumer can still download the file. If the file for some reason does not exist, then Camel will by default throw an exception, you can turn this off and ignore this by setting `ignoreFileNotFoundOrPermissionError=true`.

For example to have a Camel route that picks up a single file, and deletes it after use you can write

from("ftp://admin@localhost:21/nolist/?password=admin&stepwise=false&useList=false&ignoreFileNotFoundOrPermissionError=true&fileName=report. txt&delete=true") .to("activemq:queue:report");

Notice that we have used all the options we talked above.

You can also use this with `ConsumerTemplate`. For example to download a single file (if it exists) and grab the file content as a String type:

String data = template.retrieveBodyNoWait("ftp://admin@localhost:21/nolist/? password=admin&stepwise=false&useList=false&ignoreFileNotFoundOrPermissionError=true&fileName=report.txt&delete=true", String.class);

Debug logging

This component has log level **TRACE** that can be helpful if you have problems.

Endpoint See Also

- File2