Sun Adapter for Batch User's Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 821–0521
September 2009

090917@22749

Contents

Sun Adapter for Batch User's Guide

This guide provides an overview of the Sun Adapter for Batch, and includes details that are necessary to configure and deploy the adapter in a Sun Java™ Composite Application Platform Suite (Java CAPS) project. The Batch Adapter performs FTP and FTP-related operations. The Batch Adapter enables Sun Java CAPS ESB to use an FTP connection to exchange data with other network hosts for the purpose of receiving and delivering objects stored in files.

What You Need to Know

Creating a Sun Adapter for Batch Project

For step-by-step directions for building various Sun Adapter for Batch projects, see the Sun Adapter for Batch Tutorial.

About the Sun Adapter for Batch

This chapter provides a brief overview of the Sun Adapter for Batch.

All Adapters provide a communication bridge between the Java CAPS environment and one or more external systems.

The Batch Adapter performs a variety of FTP and FTP-related operations (depending on your specific needs, network environment, record-processing, file transfer, and external system requirements). The Batch Adapter enables Sun Java CAPS ESB to use an FTP connection to exchange data with other network hosts for the purpose of receiving and delivering objects stored in files.

Batch Adapter OTDs

The Batch Adapter provides Object Type Definitions (OTDs) that enable the creation of file transfer operations using FTP.

The Batch Adapter includes seven specific OTDs:

- **BatchFTP**: The FTP OTD connects to external FTP servers.
- **BatchFTPOverSSL**: The FTP over SSL OTD provides secure data transfer using Secure Sockets Layer (SSL) protocol.
- **BatchSCP**: The SCP OTD provides secure data transfer using Secure Copy Protocol with Secure Shell (SSH) as an underlying protocol.
- **BatchSFTP**: The SFTP OTD provides secure data transfer using SSH File Transfer Protocol (SFTP protocol). SFTP protocol provides a range of operations on remote files, such as directory listings, and remote file removal.
- **BatchLocalFile**: The local file OTD picks up or puts data files to local file systems.
- **BatchRecord**: The record-processing OTD extracts records out of files, parses files into specific records, and defines the content of files as records.
- **BatchInbound**: The inbound OTD receives a file, renames the file with GUID file name, and triggers the Business Process or Collaboration.

Note – The Batch Adapter supports standard FTP in accordance with RFC-959.

Additional Licensing Considerations

Sun Microsystems, Inc. has integrated software from 3SP (J2SSH Maverick) and /n software (IP*Works SSL) with the Batch Adapter.

Use of the integrated software libraries outside of the Sun Java Composite Application Platform Suite Batch Adapter are prohibited and may not be reversed engineered or redistributed.

n/software adds the following restrictions to use of IP*Works SSL:

IP*Works SSL software libraries may only be used by the Sun Java Composite Application Platform Suite Batch Adapter as a run-time component.

Sun Microsystems, Inc. prohibits Sun Java Composite Application Platform Suite Batch Adapter end-user(s) from changing, altering or modifying the licensed software (IP*Works SSL), creating derivative works, translations, reverse assembling, reverse compiling, disassembling, or in any way reverse engineering the licensed software.

Sun Microsystems, Inc. prohibits Sun Java Composite Application Platform Suite Batch Adapter end-user(s) from sublicensing, renting, distributing, leasing or otherwise transferring or assigning any portion of the licensed software (IP*Works SSL). You may not create any derivative works of the licensed software.

Installing the Batch Adapter

This chapter contains installation information for the Batch Adapter.

What's in This Chapter

Batch Adapter System Requirements

System requirements, supported operating systems, and specific issues that affect the Batch Adapter can be found in the following documents:

- Planning for Java CAPS Installation
- Java CAPS 6 Update Release 1 Release Notes

Installing the Batch Adapter

The Sun Java Adapter for Batch is installed by default with the Java Composite Application Platform Suite. For more information about installing this component or other components, see the following documents:

- Installing Additional Components for Java CAPS 6
- Using the Java CAPS 6 Installation GUI
- Using the Java CAPS 6 Installation CLI

Monitoring and Alerts

The **Enterprise Manager**is a Web-based interface that allows you to monitor and manage your Sun Java Composite Application Platform Suite applications.

Using the Enterprise Manager

The following links provide information on how to use the Enterprise Manager to monitor and manage Java CAPS

- Using Enterprise Manager Management Application in Java CAPS
- Monitoring Java CAPS Business Processes
- Alert Codes for Java CAPS Adapters
- Java CAPS Management and Monitoring APIs

The Enterprise Manager requires an adapter specific "plug-in" for each of your installed adapters. These plug-ins enable the Enterprise Manager to target specific alert codes for each adapter type, as well as to start and stop the inbound adapters.

Installing Adapter Enterprise Manager plug-ins

The **Adapter Enterprise Manager plug-ins** are available from the **List of Components to Download** under the Sun Java Composite Application Platform Suite Installer's **DOWNLOADS** tab.

There are two ways to add the Adapter Enterprise Manager plug-ins:

1. From the **Enterprise Manager**:
   a. From the **Enterprise Manager's** Explorer toolbar, click the **Configuration** icon.
   b. Click the **Web Applications Manager** tab, go to the **Auto-Install from Repository** tab, and connect to your Repository.
   c. Select the application plug-ins you require, and click **Install**. The application plug-ins are installed and deployed.
2. From the **Sun Java Composite Application Platform Suite Installer**:
   a. From the **Sun Java Composite Application Platform Suite Installer's Download tab**, select the Plug-Ins you require and save them to a temporary directory.
   b. Log onto the **Sun SeeBeyond Enterprise Manager**. From the **Enterprise Manager's** Explorer toolbar, click the **Configuration** icon.
   c. Click the **Web Applications Manager** tab and go to the **Manage Applications** tab.
   d. Browse for and select the WAR file for the application plug-in that you downloaded, and click **Deploy**. The plug-in is installed and deployed.

Batch Adapter Alert Codes

You can view and delete alerts using the Enterprise Manager. An alert is triggered when a specified condition occurs in a Project component. The purpose of the alert is to warn the administrator or user that a condition has occurred.

▼ To View the Adapter Alert Codes

1. **Add the Adapter Enterprise Manager plug-in for this Adapter.**
2. **From the Enterprise Manager's Explorer toolbar, click the Configuration icon.**
3. **Click the Web Applications Manager tab and go to the Manage Alert Codes tab. Your installed alert codes are displayed under the Results section. If your adapter alert codes are not available displayed under Results, do the following**

   a. **From the Install New Alert Codes section, browse to and select the adapter alert properties file for the application plug-in that you added. The alert properties files are located in the alertcodes folder of your Sun Java Composite Application Platform Suite installation directory.**

   b. **Click Deploy. The available alert codes for your application are displayed under Results. A listing of available this adapter's alert codes is displayed in the following table.**

Alert Code	Description	Detail/Action
BATCH-FTP-EWAY-CONFIG-FAILED	Batch FTP Adapter configuration error, message=[{0}].	An error occurred related to configuration. Check IS log for details. Parameter 0– the error message.
BATCH-FTP-EWAY-CONN-ACQUIRE-FAILED	Batch FTP Adapter error when acquiring connection from pool, message=[{0}].	An error occurred when looking up a connection from the connection pool. Check IS log for details. Parameter 0– the error message.

Alert Code	Description	Detail/Action
BATCH-FTP-EWAY-CONN-INIT-FAILED	Batch FTP Adapter connection initialization failed, message=[{0}].	An error occurred when initializing a connection. Check IS log for details. Parameter 0– the error message.
BATCH-FTP-EWAY-CONNECTION-FAILED	Batch FTP Adapter connection failed, method=[{0}], message=[{1}].	An error occurred when connecting to remote. Check IS log for details. Parameter 0– the method in which the error occurred. Parameter 1– the error message.
BATCH-FTP-EWAY-ERROR	Batch FTP Adapter error, message=[{0}].	An error occurred. Check IS log for details. Parameter 0– the error message.
BATCH-FTP-EWAY-OPERATION-ERROR	Batch FTP Adapter error when doing data transfer operation in [{0}], message=[{1}].	An error occurred when doing a data operation, such as put/get, etc. Parameter 0– the method in which the error occurred. Parameter 1– the error message. Check IS log for details.
BATCH-INBOUND-EWAY-CONFIG-FAILED	Batch Inbound Adapter configuration error, message=[{0}].	An error occurred related to configuration. Check IS log for details. Parameter 0– the error message.

Alert Code	Description	Detail/Action
BATCH-INBOUND-EWAY-ERROR	Batch Inbound Adapter error, message=[{0}].	
BATCH-INBOUND-EWAY-RUNNING	RUNNING Batch Inbound Adapter is running.	Inbound adapter is running. Check IS log for details.
BATCH-INBOUND-EWAY-STARTED	Batch Inbound Adapter started.	Inbound adapter started. Check IS log for details.
BATCH-INBOUND-EWAY-STOPPED	STOPPED Batch Inbound Adapter is stopped.	Batch Inbound Adapter is stopped. Check IS log for details, or restart the Batch Inbound Adapter.
BATCH-INBOUND-EWAY-STOPPING	STOPPING Batch Inbound Adapter is being stopped.	Batch Inbound Adapter is stopping. Check IS log for details, or restart the Batch Inbound Adapter.
BATCH-INBOUND-EWAY-SUSPENDED	SUSPENDED Batch Inbound Adapter is suspended.	Batch Inbound Adapter is suspended. Check IS log for details, or restart the Batch Inbound Adapter.
BATCH-INBOUND-EWAY-SUSPENDING	SUSPENDING Batch Inbound Adapter is suspending.	Batch Inbound Adapter is suspending. Check IS log for details, or restart the Batch Inbound Adapter.
BATCH-LOCALFILE-EWAY-CONFIG-FAILED	Batch Local File Adapter configuration error, message=[{0}].	An error occurred related to configuration. Check IS log for details. Parameter 0– the error message.

Alert Code	Description	Detail/Action
BATCH-LOCALFILE-EWAY-CONN-ACQUIRE-FAILED	Batch Local File Adapter error when acquiring connection from pool, message=[{0}].	An error occurred when looking up a connection from connection pool. Check IS log for details. Parameter 0– the error message.
BATCH-LOCALFILE-EWAY-CONN-INIT-FAILED	Batch Local File Adapter connection initialization failed, message=[{0}].	An error occurred when initializing connection. Check IS log for details. Note: A connection object for local file is only an object pooled in the connection pool. It does not physically connect to any host. Parameter 0– the error message.
BATCH-LOCALFILE-EWAY-ERROR	Batch Local File Adapter error, message=[{0}].	A general error occurred. Check IS log for details. Parameter 0– the error message.
BATCH-LOCALFILE-EWAY-OPERATION-ERROR	Batch Local File Adapter error when doing file operation in [{0}], message=[{1}].	An error occurred when doing local file operations, such as put/get. Check IS log for details. Parameter 0– the method in which the error occurred Parameter 1– the error message.

Alert Code	Description	Detail/Action
BATCH-REC-EWAY-CONFIG-FAILED	Batch Record Adapter configuration error, message=[{0}].	An error occurred related to configuration. Check IS log for details. Parameter 0– the error message.
BATCH-REC-EWAY-CONN-ACQUIRE-FAILED	Batch Record Adapter error when acquiring connection from pool, message=[{0}].	An error occurred when looking up a connection from connection pool. Check IS log for details. Parameter 0– the error message.
BATCH-REC-EWAY-CONN-INIT-FAILED	Batch Record Adapter connection initialization failed, message=[{0}].	An error occurred when initializing connection. Check IS log for details. Note, a connection object for batch record is only an object pooled in the connection pool, it does not physically connect to any host. Parameter 0– the error message.
BATCH-REC-EWAY-ERROR	Batch Record Adapter error, message=[{0}].	A general error occurred. Check IS log for details. Parameter 0– the error message.

Alert Code	Description	Detail/Action
BATCH-REC-EWAY-OPERATION-ERROR	Batch Record Adapter error when doing record operation in [{0}], message=[{1}].	An error occurred when doing a record operation, such as put/get, etc. Parameter 0– the method in which the error occurred. Parameter 1– the error message. Check IS log for details.
FTP-SSL-EWAY-CONFIG-FAILED	Batch FTP Over SSL Adapter configuration error, message=[{0}].	An error occurred related to configuration. Check IS log for details. Parameter 0– the error message.
FTP-SSL-EWAY-CONN-ACQUIRE-FAILED	Batch FTP Over SSL Adapter error when acquiring connection from connection pool, message=[{0}].	Error when looking up a connection from connection pool. Check IS log for details. Parameter 0– the error message.
FTP-SSL-EWAY-CONN-INIT-FAILED	Batch FTP Over SSL Adapter connection initialization error, message=[{0}].	Error when initializing a connection. Check IS log for details. Parameter 0– the error message.
FTP-SSL-EWAY-CONNECTION-FAILED	Batch FTP Over SSL Adapter connection failed, method=[{0}], message=[{1}].	An error occurred when connecting to remote. Check IS log for details. Parameter 0– the method in which the error occurred. Parameter 1– the error message.

Alert Code	Description	Detail/Action
FTP-SSL-EWAY-ERROR	Batch FTP Over SSL Adapter error, message=[{0}].	General error occurred. Check IS log for details. Parameter 0– the error message.
FTP-SSL-EWAY-OPERATION-ERROR	Batch FTP Over SSL Adapter error when doing data transfer operation in [{0}], message=[{1}].	An error occurred when doing a data operation, such as put/get, etc. Parameter 0– the method in which the error occurred. Parameter 1– the error message. Check IS log for details.
SCP-EWAY-CONFIG-FAILED	Batch SCP Adapter configuration error, message=[{0}].	Error occurred related to configuration. Check IS log for details. Parameter 0– the error message.
SCP-EWAY-CONN-ACQUIRE-FAILED	Batch SCP Adapter error when acquiring connection from connection pool, message=[{0}].	Error when looking up a connection from connection pool. Check IS log for details. Parameter 0– the error message.
SCP-EWAY-CONN-INIT-FAILED	Batch SCP Adapter connection initialization error, message=[{0}].	Error when initializing a connection. Check IS log for details. Parameter 0– the error message.

Alert Code	Description	Detail/Action
SCP-EWAY-CONNECTION-FAILED	Batch SCP Adapter connection failed, method=[{0}], message=[{1}].	Error when connecting to remote. Check IS log for details. Parameter 0– the method in which the error occurred. Parameter 1– the error message.
SCP-EWAY-ERROR	Batch SCP Adapter error, message=[{0}].	General error occurred. Check IS log for details. Parameter 0– the error message.
SCP-EWAY-OPERATION-ERROR	Batch SCP Adapter error when doing data transfer operation in [{0}], message=[{1}].	An error occurred when doing a data operation, such as put/get, etc. Parameter 0– the method in which the error occurred. Parameter 1– the error message. Check IS log for details.
SFTP-EWAY-CONFIG-FAILED	Batch SFTP Adapter configuration error, message=[{0}].	An error occurred related to configuration. Check IS log for details. Parameter 0– the error message.
SFTP-EWAY-CONN-ACQUIRE-FAILED	Batch SFTP Adapter error when acquiring connection from connection pool, message=[{0}].	An error occurred when looking up a connection from connection pool. Check IS log for details. Parameter 0– the error message.

Alert Code	Description	Detail/Action
SFTP-EWAY-CONN-INIT-FAILED	Batch SFTP Adapter connection initialization error, message=[{0}].	An error occurred when initializing a connection. Check IS log for details. Parameter 0– the error message.
SFTP-EWAY-CONNECTION-FAILED	Batch SFTP Adapter connection failed, method=[{0}], message=[{1}].	An error occurred when connecting to remote. Check IS log for details. Parameter 0– the method in which the error occurred. Parameter 1– the error message.
SFTP-EWAY-ERROR	Batch SFTP Adapter error, message=[{0}].	A general error occurred. Check IS log for details. Parameter 0– the error message.
SFTP-EWAY-OPERATION-ERROR	Batch SFTP Adapter error when doing data transfer operation in [{0}], message=[{1}].	An error occurred when doing a data operation, such as put/get, etc. Parameter 0– the method in which the error occurred. Parameter 1– the error message. Check IS log for details.

An alert code is a warning that an error has occurred. It is not a diagnostic. The user actions noted above are just some possible corrective measures you may take. Refer to the log files for more information. For information on managing and monitoring alert codes and logs, see the Alert Codes for Java CAPS Adapters.

Configuring the Batch Adapter

This chapter explains how to configure the Batch Adapter properties.

What's in This Chapter

Creating and Configuring Batch Adapters

All adapters contain a set of parameters with properties unique to that adapter type. After the adapters are established and a Batch External System is created in the Project's Environment, the adapter parameters can be modified for your specific system. The Batch Adapter contains Properties templates for the following OTDs:

- **BatchFTP**
- **BatchFTPOverSSL**
- **BatchSCP**
- **BatchSFTP**
- **BatchLocalFile**
- **BatchRecord**
- **BatchInbound**

All Batch Adapters contain properties that are accessed from the Connectivity Map and the Environment Explorer tree. **Connectivity Map** properties most commonly apply to a specific adapter, and may vary from other adapters (of the same type) in the Project. The adapter properties accessed from the **Environment Explorer** tree are commonly global, applying to all adapters (of the same type) in the Project.

Selecting a Batch External Application

To create a Batch Adapter, you must first create a Batch External Application in your Connectivity Map. Batch Adapters are located between a Batch External Application and a Service. Services are containers for Collaborations, Business Processes, eTL processes, and so forth.

▼ To create the Batch External Application

1 From the Connectivity Map toolbar, click the External Applications icon.

2 **Select a Batch External Application from the menu. The selected Batch External Application icon appears on the Connectivity Map toolbar.**

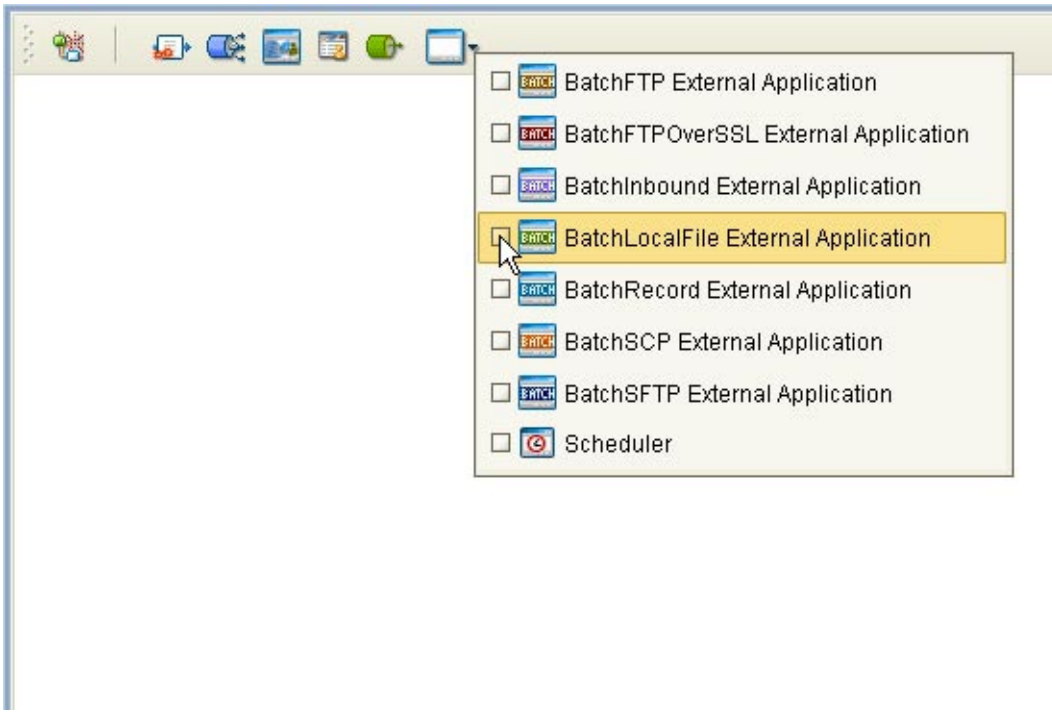


FIGURE 1 External Applications Selection Menu

3 **Drag the new Batch External Application from the toolbar onto the Connectivity Map canvas. This represents an external Batch system.**

From the Connectivity Map, you can associate (bind) the External Application with the Service to establish an adapter (see Figure 2).

FIGURE 2 Adapter Location

When Batch is selected as the External Application, it automatically applies the default Batch Adapter properties, provided by the OTD, to the adapter that connects it to the Service. These properties can then be modified for your specific system, using the **Properties Editor**.

Modifying the Adapter Properties

A Project's properties can be modified after the adapters are established in the Connectivity Map and the Environment is created.

▼ To Modify the Batch Adapter (Connectivity Map) Properties

1. **From the Connectivity Map, double click the adapter icon, located in the link between the associated External Application and the Service.**
2. **The adapter Properties Editor opens to the Adapter Batch Connectivity Map parameters. Make any necessary modifications and click OK to save the settings.**

▼ To Modify the Batch Adapter (Environment Explorer) Properties

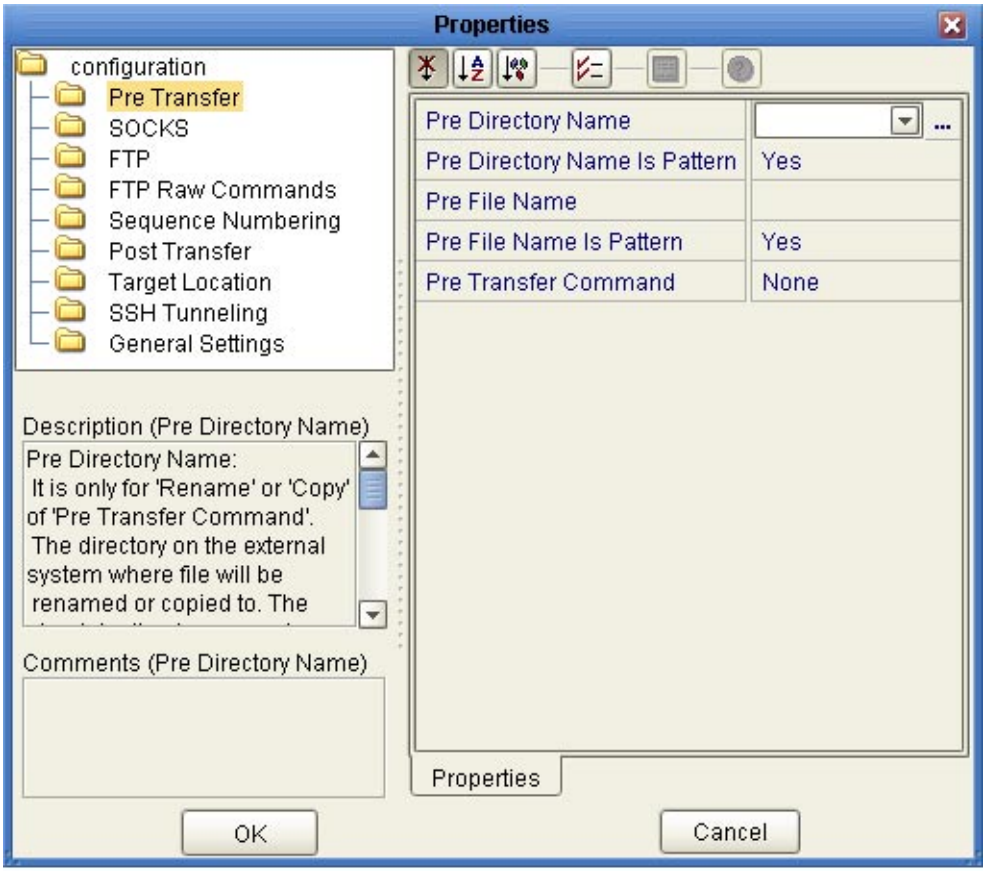
1. **From the Environment Explorer tree, right-click the Batch external system. Select Properties from the shortcut menu. The Properties Editor appears.**
2. **Make any necessary modifications to the Environment parameters of the Batch Adapters, and click OK to save the settings.**

Using the Properties Editor

The Batch Adapter properties are modified using the Batch Adapter Properties Editor.

▼ To Modify the Default Adapter Configuration Properties

1. **Open the Properties Editor for a Batch Adapter (for this example, BatchFTP Connectivity Map Properties.**
2. **From the upper-right pane of the Properties Editor, select a subdirectory of the configuration directory (for this example, select the Pre-Transfer subdirectory). The editable properties contained in that subdirectory are now displayed in the Properties pane (see the figure below).**



3 **Click on any property field to make it editable. For example, click on the Pre Directory Name parameter. You can now type the Pre Directory Name value into the property field. If a parameter's value is true/false or multiple choice, the field reveals a submenu of property options.**

4 **Click on the ellipsis (. . .) in the properties field to open a separate configuration dialog box. This is helpful for large values that cannot be fully displayed in the parameter's property field. Enter the property value in the dialog box and click OK. The value is now displayed in the parameter's property field.**

5 **A description of each parameter is displayed in the Description box when that parameter is selected, providing an explanation of any required settings or options.**

6 **The Comments box, located below the Description box, provides an area for recording notes and information regarding the currently selected parameter. This is saved for future referral.**

7 **After modifying the configuration properties, click OK to close the Properties Editor and save your current changes.**

8 **After modifying the configuration properties, click OK to close the Properties Editor and save the changes.**

Note – Properties set from the Collaboration override the corresponding properties in the adapter's configuration file. Any properties that are not set from the Collaboration retain their configured default settings.

Batch Adapter Properties

The Batch Adapter's Properties are organized as follows:

BatchFTP Adapter Connectivity Map Properties

This section describes the configuration parameters for the **BatchFTP OTD,** accessed from the Connectivity Map.

The BatchFTP Connectivity Map properties include the following sections:

Caution – Several of these configuration options allow you to use regular expressions. This advanced feature is useful but must be used carefully. An improperly formed regular expression can cause the creation of undesired data or even the loss of data. You must have a clear understanding of regular-expression syntax and construction before attempting to use this feature. It is recommended that you test such configurations thoroughly before moving them to production.

Pre Transfer (BatchFTP Connectivity Map)

Pre-transfer operations are those performed before the file transfer. For more information, see "Pre/Post File Transfer Commands" on page 148.

The Pre Transfer section of the BatchFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 1 Connectivity Map - BatchFTP - Pre Transfer

Name	Description	Required Value
Pre Directory Name	Specifies the directory name (path) on the external system to which a file is renamed or copied. The value can be a literal or a pattern name. This setting is only for the **Rename** or **Copy** operations of the **Pre Transfer Command** parameter. For outbound transfers, the directory is created if it does not already exist. See Table 1. See "Using Name Patterns" on page 162.	Enter the exact name of the directory (with the path), enter a pattern name, or select one of the following values: ■ %f ■ %f.%y%y%y%y%M%M%d%d .%h%h%m%m%s%s%S%S%S ■ %f.copy ■ %f.rename

TABLE 1 Connectivity Map - BatchFTP - Pre Transfer *(Continued)*

Name	Description	Required Value
Pre Directory Name Is Pattern	Specifies whether the directory name is interpreted as literal or as a name pattern, as follows: ■ **No**: indicates that the name entered is a literal, an exact match. ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern. See Table 1.	Select **Yes** or **No**. The configured default is **Yes**.
Pre File Name	Specifies the file name on the external system, to which a file is renamed or copied. The value represents the file name. The value can be a literal or pattern name. This setting is only for the **Rename** or **Copy** operations of the **Pre Transfer Command** parameter. Special characters are allowed, for example, the pattern **%f** indicates the original working file name. See Table 1. See "Using Name Patterns" on page 162.	Enter the exact name of the file, enter a pattern name, or select one of the following values: ■ %f ■ %f%# ■ %f.%y%y%y%y%M%M%d%d.%h%h%m%m%s%s%S%S%S ■ %f.copy ■ %f.rename
Pre File Name Is Pattern	Specifies whether the file name represents a literal or a name pattern, as follows: ■ **No**: indicates that the name entered is a literal, an exact match. No pattern matching or name expansion is done. ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern. See Table 1.	Select **Yes** or **No**. The configured default is **Yes**.

TABLE 1 Connectivity Map - BatchFTP - Pre Transfer *(Continued)*

Name	Description	Required Value
Pre Transfer Command	Allows you to execute a desired action directly before the actual file transfer. For an inbound transfer, the file can be made unavailable to other clients polling the target system with the same directory and file pattern or name. For an outbound transfer, you can perform an automatic backup or clean-up of the existing files. The options are: ■ **Rename**: Rename the target file for protection or recovery. ■ **Copy**: Copy the target file for backup or recovery. ■ **None**: Do nothing. To gain proper protection, backup, or recovery, you must choose the appropriate setting that serves your purpose. For example, to recover from failures on an outbound appending transfer, use the **Copy** setting. **Note –** When you are using **Rename** , if the destination file exists, different FTP servers can behave differently. For example, on some UNIX FTP servers, the destination file is overwritten without question. That is, no error or warning message is given. On other FTP servers, a Windows XP server for example, the system generates an error that results in exceptions being thrown in the called OTD method.Be sure you are familiar with the native behavior of the corresponding FTP server. If you are in doubt, try the action at the command prompt. If the action displays an error message, it may result in an exception being thrown in the Collaboration.	Select **Rename**, **Copy**, or **None**. The configured default is **None**. **Note –** The **Copy** option could slow system performance, especially if you are copying a large file.

SOCKS (BatchFTP Connectivity Map)

The BatchFTP SOCKS supports two negotiation methods: NO-AUTHENTICATION and USER/PASSWORD. For more information on SOCKS, see "SOCKS" on page 171.

The SOCKS section of the BatchFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 2 Connectivity Map - BatchFTP - SOCKS

Name	Description	Required Value
Socks Enabled	Specifies whether the FTP command connection goes through a SOCKS server. If you choose **No**, the adapter does not connect to a SOCKS server. In this case, all other parameters under the **SOCKS** section are ignored. **Note –** If this parameter is set to **Yes** , the host name under the **FTP** configuration could fail to resolve some names, such as **localhost** or **127.0.0.1** correctly. Use real IP or machine names to represent the hosts. See Table 11 for more details.	Select **Yes** or **No**. The configured default is **No**.
Socks Version	Specifies the SOCKS server version. If you choose **Unknown**, the adapter detects the actual version for you. **Note –** For the best performance, specify the version number, 4 or 5.	Select **4**, **5**, or **Unknown.** The configured default is **Unknown**.

FTP (BatchFTP Connectivity Map)

The FTP section of the BatchFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 3 Connectivity Map - BatchFTP - Pre Transfer

Name	Description	Required Value
Command Connection Timeout	Allows you to set the timeout of the FTP command/control connection socket. Normally, the larger the file you are transferring, the higher this value must be. Of course, the quality of the network connection also affects this setting. The value is in milliseconds. A timeout of zero is interpreted as an infinite timeout.	An integer from **0** to **2147483647**. The configured default is **45000**.

TABLE 3 Connectivity Map - BatchFTP - Pre Transfer *(Continued)*

Name	Description	Required Value
Data Connection Timeout	Allows you to set the timeout of the FTP data connection socket. Normally, a slow or busy network connection requires a higher timeout setting. The value is in milliseconds. A timeout of zero is interpreted as an infinite timeout. For setting the timeout of the command/control connection socket, see the parameter **Command Connection Timeout**.	An integer from **0** to **2147483647**. The configured default is **45000**.
Directory Listing Style	Specifies the system that reflects the remote host. This parameter is used to determine the format in which the **LIST** command returns file-listing information.The Directory Listing Style values include User Defined1 - User Defined10 values. These user defined properties allow you to create multiple user-defined FTP heuristic configurations, and make these selectable from the BatchFTP Adapter properties. You can create corresponding heuristic configurations in the FtpHeuristics.cfg file under the User Defined sections. For more information on setting user defined FTP heuristic properties, see "To Modify the FTP Heuristics Configuration File" on page 109). **Note –** This property is superseded by any value specified in the User Defined Directory Listing Style property (see Table 3). The User Defined Directory Listing Style property value must be blank (empty) to enable the Directory Listing Style property	One of the following values: ■ UNIX ■ AS400 ■ AS400-UNIX ■ HCLFTPD 6.0.1.3 ■ HCLFTPD 5.1 ■ HP NonStop/Tandem ■ MPE ■ MSFTPD 2.0 ■ MSP PDS (Fujitsu) ■ MSP PS (Fujitsu) ■ MVS GDG ■ MVS PDS ■ MVS Sequential ■ Netware 4.11 ■ NT 3.5 ■ NT 4.0 ■ UNIX ■ UNIX (EUC-JP ■ UNIX (SJIS) ■ User Defined ■ User Defined (1-10) ■ VM/ESA ■ VMS ■ VOS3 PDS (Hitachi) ■ VOS3 PS (Hitachi) ■ VOSK (Hitachi) For more information, see "Using FTP Heuristics" on page 105.

TABLE 3 Connectivity Map - BatchFTP - Pre Transfer *(Continued)*

Name	Description	Required Value
User Defined Directory Listing Style.	Specifies the name of a user-defined directory listing style (heuristics) that is available in the user-created FTP heuristics configuration file located on the logical host. This property works in conjunction with the properties Table 3 and Table 11. For details on how to use the User Defined Directory Listing Style see"To Create a Custom Heuristics Configuration File" on page 108 **Note –** The BatchFTP OTD will generate an exception if a selected User Defined Directory Listing Style or the User Defined Heuristics Configuration File path is not defined correctly. If a User Defined Directory Listing Style is specified, a corresponding value must also be provided for the User Defined Heuristics Configuration File property.	A text string value (default to blank) representing the directory listing style (heuristics) name which is defined in a user supplied heuristics configuration file.
Use PASV	Allows you to prompt the adapter to enter either the passive or active mode. Normally, when you connect to an FTP site, the site establishes the data connection to your computer. However, some FTP sites allow passive transfers, meaning that your computer establishes the data connection. By default, the passive mode is used. It is recommended that you use this mode for transfers to and from FTP sites that support it. The passive mode can be required in the following situations: ■ For users on networks behind some types of router-based firewalls ■ For users on networks behind a gateway requiring passive transfers ■ If transfers are erratic ■ If data-channel errors are prevalent in your environment	Select **Yes** or **No**. The configured default is **Yes**.
Mode	Specifies the mode used to transfer data to or from the FTP server, using the **Ascii, Binary,** or **Ebcdic** mode. If you choose Ebcdic, make sure of the following: ■ Your FTP server supports the EBCDIC mode. ■ You are processing EBCDIC data.	Select **Ascii, Binary,** or **Ebcdic.** The configured default is **Binary**.

FTP Raw Commands (BatchFTP Connectivity Map)

FTP raw commands are commands that are sent *directly* to the FTP server.

The **FTP Raw Commands** section of the BatchFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 4 Connectivity Map - BatchFTP - FTP Raw Commands

Name	Description	Required Value
Post Transfer Raw Commands	Specifies the FTP raw commands to be used directly after the file-transfer command. For example, some SITE commands use a ; (semi-colon) to separate the command set, as displayed in this example: `SITE RECFM=FB;SITE LRECL=50;SITE BLOCKSIZE=32750;SITE TRACKS;SITE PRI=5;SITE SEC=5` These commands are sent one by one, in the sequence they are listed. **Note –** Certain combinations of post-transfer raw commands can cause the loss of data if there is a failure on the FTP server. For example, if the inbound post-transfer command is **Delete** , and your post-transfer raw commands fail, the deleted file is not recoverable.	One or more valid FTP raw commands. **Note –** These commands are sent to the FTP server directly and are not interpreted by the adapter in any way.
Pre Transfer Raw Commands	Specifies the FTP raw commands to be used directly before the file-transfer command. For example, some SITE commands use a ; (semi-colon) to separate the command set: `SITE RECFM=FB;SITE LRECL=50;SITE BLOCKSIZE=32750;SITE TRACKS;SITE PRI=5;SITE SEC=5` These commands are sent one by one, in the sequence they are listed.	One or more valid FTP raw commands. **Note –** These commands are sent to the FTP server directly and are not interpreted by the adapter in any way.

Sequence Numbering (BatchFTP Connectivity Map)

The Sequence Numbering section of the BatchFTP Connectivity Map properties contains the top-level parameters displayed in this table.

Note – The Synchronized property, under General Settings, must be set to "Yes" to use Sequence Numbering.

TABLE 5 Connectivity Map - BatchFTP - Sequence Numbering

Name	Description	Required Value
Max Sequence Number	Use this parameter when you have set up the target directory or file name to contain a sequence number. It tells the adapter that when this value (the **Max Sequence Number**) is reached, to reset the sequence number to the **Starting Sequence Number** value. This parameter is used for the name pattern **%#**. See "Using Name Patterns" on page 162.	An integer from **1** to **2147483647**. The value of **Max Sequence Number** *must* be greater than that of **Starting Sequence Number**.
Starting Sequence Number	Use this parameter when you have set up the target directory or file name to contain a sequence number. It tells the adapter which value to start with in the absence of a sequence number from the previous run. This parameter is used for the name pattern **%#**. When the **Max Sequence Number** value is reached, the sequence number rolls over to the **Starting Sequence Number** value. See "Using Name Patterns" on page 162.	An integer from **0** to **2147483647**. The value of the **Starting Sequence Number** *must* be less than the **Max Sequence Number** value.

Post Transfer (BatchFTP Connectivity Map)

Post-transfer operations are those performed on remote (ftp) site after the real ftp transfer. For more information on this feature, see "Pre/Post File Transfer Commands" on page 148.

The Post Transfer section of the BatchFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 6 Connectivity Map - BatchFTP - Post Transfer

Name	Description	Required Value
Post Directory Name	Specifies the directory name (path) on the external system to which a file is renamed. The value can be a literal or pattern name. For an outbound transfer (to destination), the directory is created if it does not already exist. This setting is only for the **Rename** operation of the **Post Transfer Command** parameter. Special characters are allowed, for example, the pattern **%f** indicates the original working directory name. The expansion of any special characters is carried out each time this parameter is used. See Table 6. See "Using Name Patterns" on page 162.	Enter the exact name of the directory (with the path), enter a pattern name, or select one of the following values: ■ %f ■ %f%# ■ %f.%y%y%y%y%M%M%d %d.%h%h%m%m%s%s%S%S%S ■ %f.rename
Post Directory Name Is Pattern	Specifies whether the pattern entered for the directory represents a literal or a name pattern, as follows: ■ **No**: indicates that the name entered is a literal, an exact match. ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern. See Table 6.	Select **Yes** or **No**. The configured default is **Yes**.
Post File Name	Specifies the file name to which a file on an external system is renamed. The value represents the file name. The value can be a literal, or pattern name. This setting is only for **Rename** operation of **Post Transfer Command** parameter. Special characters are allowed. For example, the pattern **%f** indicates the original working file name. See Table 6. See "Using Name Patterns" on page 162.	Enter the exact name of the file, enter a pattern name, or select one of the following values: ■ %f ■ %f.%y%y%y%y%M%M%d%d.%h%h%m%m%s%s%S%S%S ■ %f.rename
Post File Name Is Pattern	Specifies whether the pattern entered for the file name is interpreted as literal or as a name pattern , as follows: ■ **No**: indicates that the name entered is literal, an exact match. No pattern matching or name expansion is done. ■ **Yes**: indicates that the name value you enter is a name pattern. See Table 6.	Select **Yes** or **No**. The configured default is **Yes**

TABLE 6 Connectivity Map - BatchFTP - Post Transfer *(Continued)*

Name	Description	Required Value
Post Transfer Command	Allows you to execute a desired action directly after the actual file transfer or during the "commit" phase. For an inbound transfer, you can mark the transferred file as "consumed" by making an automatic backup (**Rename**) or by destroying it permanently (**Delete**). For an outbound transfer, you can make the transferred file available to other clients by renaming it. The options are: ■ **Rename**: Rename the transferred file. ■ **Delete**: Delete the transferred file (inbound transfers only). ■ **None**: Do nothing. **Note –** When you are using **Rename** , if the destination file exists, different FTP servers can behave differently. For example, on some UNIX FTP servers, the destination file is overwritten without question. That is, no error or warning message is given. On other FTP servers, a Windows XP server for example, the system generates an error that results in exceptions being thrown in the called OTD method.Be sure you are familiar with the native behavior of the corresponding FTP server. If you are in doubt, try the action at the command prompt. If the action displays an error message, it is likely to result in an exceptions being thrown in the Collaboration.	**Select Rename**, **Delete**, or **None**. The configured default is **None**.

Target Location (BatchFTP Connectivity Map)

The **Target Location** section allows you to configure the parameters for the **Target Location** (remote location) of the FTP directories and files.

The **Target Location** section of the BatchFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 7 Connectivity Map - BatchFTP - Target Location

Name	Description	Required Value
Append	Specifies whether to overwrite or append the data to the existing file. Use this parameter for outbound FTP transfers only. Choose the appropriate setting as follows: ■ If you select **Yes** and the target file already exists, the data is appended to the existing file. ■ If you select **No**, the adapter overwrites the existing file on the remote system. If a file with the same name does not exist, both **Yes** and **No** create a new file on the external host.	**Select Yes** or **No.** The configured default is **No**.
Target Directory Name	Specifies the directory on the external system from which files are retrieved or sent. The directory name and path is preferred, otherwise, the path is relative to your home directory when you log on to the FTP server. The value can be a literal, regular expression (source), or pattern name (destination). For outbound FTP operations (destination), the directory is created if it does not already exist. See Table 7. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A directory name and path on the target external system.
Target Directory Name Is Pattern	Specifies whether the directory name is represented as literal, or as a regular expression or name pattern, as follows: ■ **No**: indicates that the name entered is a literal, an exact match. ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. See Table 7.	**Select Yes** or **No.** The configured default is **No**.

TABLE 7 Connectivity Map - BatchFTP - Target Location *(Continued)*

Name	Description	Required Value
Target File Name	Specifies the name of the remote FTP file to be retrieved or sent. The value can be a literal, regular expression (get), or pattern name (put). For MVS GDG systems, the target file name can be the version of the data set, for example: ■ Target directory name = "**STC.SAMPLE.GDGSET**' ■ Target file name = (**0**) to indicate the current version See Table 7. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	For inbound: a literal file name or a regular expression. For outbound: a literal file name or name pattern.
Target File Name Is Pattern	Specifies whether the target file name represents a literal, or as a regular expression or name pattern, as follows: ■ **No**: indicates that the name entered is a literal, an exact match. ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. See Table 7.	Select **Yes** or **No.** The configured default is **Yes**.

SSH Tunneling (BatchFTP Connectivity Map)

The SSH Tunneling section provides information for configuring the **SSH Tunneling** properties. If Secure FTP (FTP over SSH or FTP over SSL) is required, use the Secure FTP OTDs (BatchFTPOverSSL, BatchSFTP, and BatchSCP).

The SSH Tunneling section of the BatchFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 8 Connectivity Map - BatchFTP - SSH Tunneling

Name	Description	Required Value
SSH Channel Established	Specifies whether the adapter needs to launch an SSH subprocess. Selecting **No** indicates that the SSH channel has not yet been established. The adapter spawns a subprocess internally then establishes the channel on your behalf. If you select **No**, you must set the following parameters: ■ **SSH Command Line** ■ **SSH Listen Port** (Environment property) If you select **No**, setting the following parameters is optional: ■ **SSH User Name** (Environment property) ■ **SSH Password** (Environment property) Selecting **Yes** indicates that an SSH channel has already been established. That is, the channel has already been started outside the adapter, and the adapter does not need to establish it. For example, you could have issued a command outside of Java CAPS, or you could know that another Batch Adapter instance has already established the channel by the time this adapter runs. If you select **Yes**, you must set the following parameters: ■ **SSH Listen Host** (Environment property) ■ **SSH Listen Port** (Environment property)	**Select Yes** or **No.** The configured default is **No**.

TABLE 8 Connectivity Map - BatchFTP - SSH Tunneling *(Continued)*

Name	Description	Required Value
SSH Command Line	Specifies the command line used to establish an SSH channel. This parameter is required only when you set the **SSH Channel Established** parameter to **No**. This entry must be the complete, correct command line required by the additional software application you are using to support SSH tunneling. This command line is executed as it is, so you must be sure of the following: ■ It contains all the necessary arguments ■ The syntax is correct ■ It is compliant with your SSH-environment To verify these requirements, test this command line manually outside of Java CAPS to make sure it works correctly. Execute the command line from the shell and ensure that it does not prompt for any additional user input. If it does, continue to add whatever additional parameters are required until it no longer prompts for additional input, then use that command line in the adapter's configuration. You can specify any other options that are based on your SSH-environment. However, if you do so, you must still be sure this command line is correct and complete. For example, port forwarding could be specified using the following command-line option: -L `ListenPort:FtpServerHost:FtpServerPort` In this example, *ListenPort* must be the same value as that given for the parameter **SSH Listen Port**. The value given for `FtpServerHost` overwrites the parameter setting for **Host Name** under the **FTP** parameters. The value given for `FtpServerPort` overwrites the parameter setting for **Server Port** under the **FTP** parameters. All other settings under the **FTP** parameters operate for the specified FTP server: **FtpServerHost:FtpServerPort**. If the SSH channel established by an SSH command line must be shared by other Batch Adapter instances located on different Java CAPS client hosts, you must configure SSH port forwarding to allow non-local connections from other hosts. For some SSH clients, you can use the option **-g**. **Note –** You can also specify port forwarding in your SSH configuration file. (*Continued on the next page*)	A valid SSH command line.

TABLE 8 Connectivity Map - BatchFTP - SSH Tunneling *(Continued)*

Name	Description	Required Value
SSH Command Line (continued)	(*Continued from last page*) The command-line syntax can differ, depending on the type of SSH client implementation you are using. See your SSH-tunneling support software user documentation for details. Examples: ssh -L 3456:ftp.sun.com:21 -o BatchMode=yes apple ssh -L 4567:apple:21 -o BatchMode=yes apple ssh -L 5678:orange:21 -o BatchMode=yes apple ssh -L 6789:orange:21 -g -o BatchMode=yes apple plink -L 4567:apple:21 apple plink -L 5678:orange:21 apple plink -L 6789:orange:21 -g apple	
SSH Tunneling Enabled	Specifies whether the FTP command connection is secured through an SSH tunnel. If you choose **No**, all other parameters in this section are ignored. **Note –** If you want to use the SSH port-forwarding feature, you may need to reconfigure your FTP server, depending on what kind of server you are using and how it is currently configured. See your SSH documentation for more information.	**Select Yes** or **No.** The configured default is **No**.

Additional SSH-supporting Software

The adapter's SSH tunneling (also known as port forwarding) feature utilizes additional existing SSH-supporting software applications, for example, Plink on Windows or OpenSSH on UNIX (see "Additional Software Requirements" on page 174

For different SSH client implementations, the command syntax and environment configuration may vary. See your SSH-supporting application's user guide for details.

Port-forwarding Configuration

SSH tunneling provides secure FTP command connections. This mechanism is based on an existing SSH port-forwarding configuration. You must configure SSH port forwarding on the *SSH listen host* before you configure the supporting adapter connection.

For example, on the Java CAPS client host **localhost**, you can issue a command, such as:

```
ssh -L 4567:apple:21 -o BatchMode=yes apple
```

Under the adapter's configuration for the previous example, you must specify:

- **localhost** for the Environment parameter **SSH Listen Host**
- **4567** for the Environment parameter **SSH Listen Port**

In this case, the adapter connects to the FTP server **apple:21** through an SSH tunnel. For more information on SSH tunneling, see "SSH Tunneling Support" on page 174.

Note – It is possible to use SOCKS and SSH tunneling at the same time. However, this practice is not recommended.

General Settings (BatchFTP Connectivity Map)

The General Settings section of the BatchFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 9 Connectivity Map - BatchFTP - General Settings

Name	Description	Required Value
Synchronized	Specifically applies to legacy Batch Adapter Projects. Provides backward compatibility to allow Projects that were created using the Batch Adapter version 5.0.7 or earlier to be imported and deployed without a change in the adapters behavior. The selections are: ■ Yes: Provides backward compatibility for legacy (pre-5.0.8 Batch Adapter) Projects. The adapters run in synchronized mode, one instance of the Collaboration after the other. ■ No: For use with new Batch Adapter Projects. The adapter run in parallel, creating multiple instances of the Collaboration that run in parallel. All OTD instances used in a Project should have the same value for this property. **Note –** Synchronized must be set to "Yes" to use Sequence Numbering.	**Yes** or **No.** The default setting is **Yes, simulating Projects created with** Batch Adapter version 5.0.7 or earlier.

BatchFTP Adapter Environment Properties

This section describes the configuration properties for the **BatchFTP OTD accessed from the Environment Explorer tree**.

The BatchFTP Environment Explorer properties include the following sections:

- "SOCKS (BatchFTP Environment)" on page 43

Caution – Several of these configuration options allow you to use regular expressions. This advanced feature is useful but must be used carefully. An improperly formed regular expression can cause the creation of undesired data or even the loss of data. You must have a clear understanding of regular-expression syntax and construction before attempting to use this feature. It is recommended that you test such configurations thoroughly before moving them to production.

SOCKS (BatchFTP Environment)

This section provides information for configuring the **SOCKS** properties (accessed from the Environment Explorer). The BatchFTP Adapter supports the negotiation methods, No-authentication and User/password. For more information on SOCKS, see "SOCKS" on page 171.

The SOCKS section of the BatchFTP Environment properties contains the top-level parameters displayed in this table.

TABLE 10 Environment - BatchFTP - SOCKS

Name	Description	Required Value
Socks Host Name	Specifies the SOCKS server (host) name. If you are communicating with a SOCKS server enter the SOCKS server name in this parameter.	The name of the SOCKS server.
Socks Server Port	Specifies the port number to use on the SOCKS server, when connecting to it.	An integer from **1** to **65,535**. The configured default is **1080**.
Socks User Name	Specifies the user name to use (together with the password specified under the **Socks Password** parameter) for authentication with a SOCKS5 server, if necessary. This parameter is used for the user/password negotiation method.	A valid SOCKS5 user name.

TABLE 10 Environment - BatchFTP - SOCKS *(Continued)*

Name	Description	Required Value
Socks Password	Specifies the password to use (together with the user name specified under the **Socks User Name** parameter) for authentication with a SOCKS5 server, if necessary. This parameter is used for the user/password negotiation method. **Note –** The corresponding Java accessors are **getSocksPassword**() , **setSocksPassword(java.lang.String p)** and **setSocksEncryptedPassword(java.lang.String p)**.	A valid SOCKS5 password.

FTP (BatchFTP Environment)

The **FTP** section of the BatchFTP Environment properties contains the top-level parameters displayed in this table.

TABLE 11 Environment - BatchFTP - FTP

Name	Description	Required Value
Host Name	Specifies the name of the external system that the adapter connects to. If the parameter **SSH Tunneling Enabled** under the **SSH Tunneling** configuration settings is set to **Yes**, the parameters **Host Name** and **Server Port,** under the FTP settings, are ignored. In this case, the FTP host name is determined by an SSH option, according to the following model: **ssh -L ListenPort:FtpServerHost:FtpServerPort SSHServer** In the previous example, the FTP feature communicates with the FTP server *FtpServerHost:FtpServerPort* using an existing SSH tunnel. See "SSH Tunneling (BatchFTP Connectivity Map)" on page 38 for details. If the parameter **Socks Enabled** under the **SOCKS** configuration parameters is set to **Yes**, the host name under the **FTP** configuration could fail to resolve some names, for example, **localhost** or **127.0.0.1** correctly. Use real IP or machine names to represent the hosts. See "SOCKS (BatchFTP Connectivity Map)" on page 29 for details.	The host name.

TABLE 11 Environment - BatchFTP - FTP *(Continued)*

Name	Description	Required Value
Server Port	Specifies the port number to use on the FTP server when connecting to it. If the parameter **SSH Tunneling Enabled** under the **SSH Tunneling** configuration is set to **Yes**, the parameters **Host Name** and **Server Port** under the **FTP** configuration are ignored. In this case, the FTP server port number is determined by an SSH option, according to the following model: **ssh -L ListenPort:FtpServerHost:FtpServerPort SSHServer** In the previous example, the FTP feature communicates with the FTP server *FtpServerHost:FtpServerPort* using an existing SSH tunnel. See "SSH Tunneling (BatchFTP Connectivity Map)" on page 38 for details.	The server port number.
User Name	Specifies the user name used to log onto the external system, when required.	A user name that provides access to the external system.
Password	If a password is required to log on to an external system, enter the password that corresponds to the user name. The corresponding Java accessor methods are `getPassword()`, `setPassword()`, and `setEncryptedPassword()`.	The password.
User Defined Heuristics Configuration File	Specifies the name and location of the user defined FTP heuristics configuration file. The format of the files content must be in the same form as that of the FTPHeuristics.cfg file. See "To Create a Custom Heuristics Configuration File" on page 108 for more details. This property works in conjunction with the property Table 3 User Defined Directory Listing Style property **Note –** The BatchFTP OTD will generate an exception if a selected User Defined Directory Listing Style or the User Defined Heuristics Configuration File path is not defined correctly. If a User Defined Directory Listing Style is specified, a corresponding value must also be provided for the User Defined Heuristics Configuration File property.	The location and name of the user defined FTP heuristics configuration file on the local host.

General Settings (BatchFTP Environment)

The **General Settings** section of the BatchFTP Environment properties contains the top-level parameters displayed in this table.

TABLE 12 Environment - BatchFTP - General Settings

Name	Description	Required Value
State Persistence Base Location	Specifies a working directory for storing intermediary results. Options: ■ Leave value blank: BatchFTP will use a default folder as the working directory. ■ Specify a path to a local file system folder with read/write permissions.	A working directory with read/write permissions, or leave blank (no value) to accept a default directory.
Connection Mode	Specifies whether a physical connection is established when an external connection is instantiated. The options are: ■ **Automatic:** Establishes a physical connection when an external connection is instantiated. ■ **Manual:** Does not automatically establish a physical connection when an external connection is instantiated. If a physical connection is not automatically established, a physical connection must be established from the Collaboration (for example, by calling the connect() method).	Select **Automatic** or **Manual**. The configured default is Automatic.

SSH Tunneling (BatchFTP Environment)

This section provides information for configuring the **SSH Tunneling** properties (accessed from the Environment Explorer). If Secure FTP (FTP over SSH or FTP over SSL) is required, use the Secure FTP OTDs (**BatchFTPOverSSL**, **BatchSFTP**, and **BatchSCP**). SSH Tunneling is supported for compatibility purposes.

The **SSH Tunneling** section of the BatchFTP Environment properties contains the top-level parameters displayed in this table.

TABLE 13 Environment - BatchFTP - SSH Tunneling

Name	Description	Required Value
SSH Listen Host	Specifies the name of the host where the SSH support software runs, and to which the host listens. This parameter is required when and only when **SSH Channel Established** is set to **Yes**. The reason for this is, if you choose **No** this Listen Host will always be **localhost** because the SSH client will always be started from localhost. For optimum security, it is recommended that you use **localhost** as your choice. The connection to the corresponding port number on this host is forwarded to the FTP server through an SSH-secure channel. On the listen host, the SSH support software must be configured and started with the **Port-Forwarding** option. The FTP command connection is forwarded through the secure tunnel. The corresponding SSH command uses the following model: ssh -L **ListenPort:FtpServerHost:FtpServerPort** -o BatchMode=yes **SSHServer** If this host name is not **localhost**, the data transport between the local host and the SSH listen host is not secure. Also, your SSH support software should be configured to allow connections to other hosts (for some SSH clients, it is an option **-g**). For example, on an SSH listen host, you could issue a command, such as: ssh -L 4567:apple:21 -o BatchMode=yes apple or ssh -L 5678:orange: 21 -o BatchMode=yes apple Regardless, the transport between the SSH listen host and the FTP server is still secure.	The SSH listen host name. The configured default is **localhost.**

TABLE 13 Environment - BatchFTP - SSH Tunneling *(Continued)*

Name	Description	Required Value
SSH Listen Port	Specifies the port number that the SSH-tunneling support software uses to check for incoming connections. This port number can be any unused port number on the SSH listen host. The connection to this port is forwarded to the FTP server through an SSH-secure channel. This parameter is required and it must be exactly same as the `ListenPort` value in the SSH command you issue either inside or outside the Java CAPS system. The corresponding SSH command line uses the following model: ssh -L **ListenPort:FtpServerHost:FtpServerPort** -o BatchMode=yes **SSHServer Required Values**	An integer from **1** to **65535**.
SSH User Name	Specifies an SSH user name. This parameter can be required when the setting for the **SSH Channel Established** parameter is **No**. This parameter is required only if the SSH support software is started from within the adapter (refer to the corresponding SSH command line). Even then, it is only required if your SSH implementation executes in an interactive way that requires you to enter a user name. Again, this requirement depends on how you specify the SSH command line and how your SSH environment is configured.	The SSH user name.
SSH Password	Specifies an SSH password corresponding to the user name entered under **SSH User Name**. This parameter can be required only when the setting for the **SSH Channel Established** parameter is **No**. For more information, see **SSH User Name**.	The SSH password.

Connection Pool Settings (BatchFTP Environment)

The Connection Pool Settings section of the BatchFTP Environment properties contains the top-level parameters displayed in this table.

TABLE 14 Environment - BatchFTP - Connection Pool Settings

Name	Description	Required Value
Steady pool size	Specifies the minimum number of physical EIS connections that the pool keeps available at all times.	An integer indicating the maximum number of connections available at all times. A value of **0** (zero) indicates that there are no physical connections in the pool and that new connections are created as needed. The configured default is **2**.
Maximum pool size	**Specifies the maximum number of physical EIS connections the pool contains.**	An integer indicating the maximum number of connections allowed. A value of 0 **(zero) indicates that there is no maximum**. The default value is **10**.
Max Idle Timeout In Second	**Specifies the maximum idle timeout (in seconds)**. This is a hint to the server. A timer thread periodically removes unused connections. This parameter defines the interval at which this thread runs. This thread removes unused connections after the specified idle time expires. It allows the user to specify the maximum amount of time that a connection can remain in the pool. .	An integer indicating **the maximum idle timeout in seconds**. When this is set to a number greater than 0 (zero), the container removes or destroys any connections that are idle for the specified duration. A value of 0 specifies that idle connections can remain in the pool indefinitely. The configured default is **300** (5 minutes).

Connection Retry Settings (BatchFTP Environment)

The Connection Retry Settings section The Connection Pool Settings section of the BatchFTP Environment properties contains the top-level parameters displayed in this table.

TABLE 15 Environment - BatchFTP - Connection Retry Settings

Name	Description	Required Value
Connection Retries	Specifies the number of retries to establish a connection upon failure to acquire a connection.	An integer indicating the maximum number of retries to establish a connection upon failure to acquire a connection. The Configured default value is **0**.

TABLE 15 Environment - BatchFTP - Connection Retry Settings *(Continued)*

Name	Description	Required Value
Connection Retry Interval	Specifies the length of time (in milliseconds) between each reattempt to access the destination file. This is used in conjunction with the Connection Retries setting.	An integer indicating length of the pause (in milliseconds). The configured default value is **1000** (1 second).

BatchFTPOverSSL Adapter Connectivity Map Properties

The BatchFTPOverSSL Adapter Connectivity Map properties include the following sections:

Pre Transfer (BatchFTPOverSSL Connectivity Map)

The Pre Transfer section allows user to customize the behaviors of protection/backup/recovery. This section describes the operation that will be performed on remote end or locally before the real file transfer.

The Pre Transfer section of the BatchFTPOverSSL Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 16 Connectivity Map - BatchFTPOverSSL - Pre Transfer

Name	Description	Required Value
Remote Dir Name	Specifies the directory and path on the remote external system where file the is renamed or copied. This is only for **Rename** or **Copy** of the **Remote Pre Command**. The value can be a literal, regular expression (source), or pattern name (destination). When specifying a destination directory, the directory is created if it doesn't already exist. Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used. For example, the pattern **%f** means the original working directory name. See Table 16. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A directory name and path location on the target system. Special characters are allowed.
Remote Dir Name Is Pattern	Specifies whether the pattern entered for the directory represents a literal, or a name pattern or regular expression, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: Indicates that the name value entered represents a literal, an exact match. No pattern matching or name expansion is done. See Table 16.	Select **Yes** or **No**. The configured default is **Yes**.
Remote File Name	Specifies the file name on the external system, to which a file is renamed or copied. The value represents the file name without the path. This setting is only for the **Rename** or **Copy** operations of **Pre Transfer Command** parameter. The value can be a literal, regular expression (get), or pattern name (put). Special characters are allowed, for example, the pattern **%f** indicates the original working file name. The expansion of any special characters is carried out each time this parameter is used. See Table 16. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A remote file name.

TABLE 16 Connectivity Map - BatchFTPOverSSL - Pre Transfer *(Continued)*

Name	Description	Required Value
Remote File Name Is Pattern	Specifies whether the pattern entered for the file name represents a literal, or a name pattern or regular expression, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ No: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 16.	Select **Yes** or **No**. The configured default is **Yes**.
Remote Pre Command	Allows you to execute a desired action directly before the actual file transfer. For an inbound transfer, the file can be made unavailable to other clients polling the target system with the same directory and file pattern or name. For an outbound transfer, you can perform an automatic backup/clean-up of the existing files. The options are: ■ **Rename**: Rename the target file for protection or recovery. ■ **Copy**: Copy the target file for backup or recovery. ■ **None**: Do nothing. To gain proper protection, backup, or recovery, you must choose the appropriate setting that serves your purpose. For example, to recover from failures on an outbound appending transfer, use the **Copy** setting. **Note –** When you are using **Rename** , if the destination file exists, different FTP servers can behave differently. For example, on some UNIX FTP servers, the destination file is overwritten without question. That is, no error or warning message is given. On other FTP servers, a Windows XP server for example, the system generates an error that results in exceptions being thrown in the called OTD method.Be sure you are familiar with the native behavior of the corresponding FTP server. If you are in doubt, try the action at the command prompt. If the action displays an error message, it may result in an exception being thrown in the Collaboration.	Select **Rename**, **Copy**, or **None**. The configured default is **None**. **Note –** The **Copy** option could slow system performance, especially if you are copying a large file.

TABLE 16 Connectivity Map - BatchFTPOverSSL - Pre Transfer *(Continued)*

Name	Description	Required Value
Local Dir Name	Specifies the directory name (path) to be used by **Rename** or **Copy**. The value can be a literal, regular expression (source), or pattern name (destination). Special characters are allowed (name pattern). The expansion of any special characters is carried out each time this parameter is used. See Table 16. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162. **Note –** When entering a path separator, use the forward slash " / " instead of the back slash " \ ". The adapter interprets the back slash as a special character. For example, use `c:/temp/dir`.	A directory name.
Local Dir Name Is Pattern	Specifies whether the Local Directory Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 16.	Select **Yes** or **No**. The configured default is **No**.
Local File Name	Specifies the file name to be used by **Rename** or **Copy**. The value can be a literal, regular expression (get), or pattern name (put). Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162. See Table 16.	A file name.
Local File Name Is Pattern	Specifies whether the Local File Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ No: indicates that the name value entered is a literal, an exact match. See Table 16.	Select **Yes** or **No**. The configured default is **No**.

TABLE 16 Connectivity Map - BatchFTPOverSSL - Pre Transfer *(Continued)*

Name	Description	Required Value
Local Pre Command	Allows you to execute a desired action directly before the actual file transfer. For an inbound transfer, the file can be made unavailable to other clients polling the target system with the same directory and file pattern or name. For an outbound transfer, you can perform an automatic backup of the existing files. The options are: ■ **Rename**: Rename the target file for protection or recovery. ■ **Copy**: Copy the target file for backup or recovery. ■ **None**: Do nothing. To gain proper protection, backup, or recovery, you must choose the appropriate setting that serves your purpose. For example, to recover from failures on an outbound appending transfer, use the **Copy** setting. **Note –** Rename and Copy overwrite the file specified by the Local Dir Name and Local File Name properties, if they exist.	Select **Rename**, **Copy**, or **None**. The configured default is **None**. **Note –** The **Copy** option could slow system performance, especially if you are copying a large file.

FTP and SSL Settings (BatchFTPOverSSL Connectivity Map)

The FTP and SSL Settings section of the BatchFTPOverSSL Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 17 Connectivity Map - BatchFTPOverSSL - FTP and SSL Settings

Name	Description	Required Value
Secure Mode	Specifies the secure mode. Selections are: ■ **None:** FTP is in clear text. ■ **Implicit SSL:** The SSL handshake is started right after the socket connection is done. ■ **Explicit SSL:** The SSL handshake is started by the client sending `AUTH SSL/TLS FTP` command.	Select **None**, **Implicit SSL**, or **Explicit SSL**. **None** is the configured default.
Directory Listing Style	Specifies the directory listing style of the FTP Server as **UNIX**, **NT**, or **MVS**. This provides a "hint" to the client side for parsing the directory listing response from the FTP Server.	Leave as **UNIX**. Currently the only supported option is UNIX. The configured default is **UNIX**.

TABLE 17 Connectivity Map - BatchFTPOverSSL - FTP and SSL Settings *(Continued)*

Name	Description	Required Value
Enabled Passive Mode	Specifies whether FTP passive mode is enabled.	**Select Yes** or **No**. Yes indicates that FTP passive mode is enabled. The configured default is **Yes**.
Transfer Mode	Specifies whether the transfer is binary code or ASCII text.	**Select BINARY or ASCII.** The configured default is **BINARY**.
Append	Specifies whether new data transferred to a remote server is appended to data that was previously transferred.	**Select Yes** or **No**. Yes indicates that data will be appended. The configured default is **No**.
Required Server Authentication	Specifies whether server authentication is required. The selections are: ■ **Yes:** Indicates that server authentication is required, and that all parameters used for authentication (for example, **Key Store Location** , **Key Store Password**, **Key Store Type**, and so forth) must be set correctly so that the server certificate can be verified against the local trusted CA certificates. ■ **No:** Indicates that server authentication is not required.	**Select Yes** or **No**. The configured default is **Yes**.
Distinguished Name for User	Specifies the distinguished name (DN) for the login user. This is imported from a CSR reply, and used to configure client authentication.	The Distingusihed Name, as in X.509.
Alias in Key Store	Specifies the alias for a key pair in a JKS type Key Store. This value is used to configure client authentication.	The alias.
Alias Password	Specifies the password that protects the key pair entry in the keystore, identified by the alias.	The alias password.
Remote Directory	Specifies a directory on the FTP server where data is sent or received. The accessibility of the directory usually depends on the login user. The value can be a literal, regular expression (source), or pattern name (destination). See Table 17. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The name of the remote directory.

TABLE 17 Connectivity Map - BatchFTPOverSSL - FTP and SSL Settings *(Continued)*

Name	Description	Required Value
Remote Directory Name is Pattern	Specifies whether the Remote Directory Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. See Table 17.	Select **Yes** or **No.** The configured default is **No**.
Remote File	Specifies the file name on the remote server. ■ The value can be a literal, regular expression (get), or pattern name (put). See Table 17. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The name of the remote file.
Remote File Name is Pattern	Specifies whether the Remote File Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. See Table 17.	Select **Yes** or **No.** The configured default is **No**.
Local Directory	Specifies the local directory (path) for files that are sent to or received from a remote system. The value can be a literal, regular expression (source), or pattern name (destination). See Table 17. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The local directory name.

TABLE 17 Connectivity Map - BatchFTPOverSSL - FTP and SSL Settings *(Continued)*

Name	Description	Required Value
Local Directory Name is Pattern	Specifies whether the Local Directory name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 17.	Select **Yes** or **No.** The configured default is **No**.
Local File	Specifies the local file name. The value can be a literal, regular expression (get), or pattern name (put). See Table 17. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The local file name.
Local File Name is Pattern	Specifies whether the Local File name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 17.	Select **Yes** or **No.** The configured default is **No**.
Local File Overwrite	Specifies whether new data downloaded from the remote will overwrite existing data.	Select **Yes** or **No.** The configured default is **No**.

Post Transfer (BatchFTPOverSSL Connectivity Map)

The Post Transfer section of the BatchFTPOverSSL Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 18 Connectivity Map - BatchFTPOverSSL - Post Transfer

Name	Description	Required Value
Remote Dir Name	Specifies the directory name (path) on the remote external system where the file the will be renamed or copied. This is only for **Rename** or **Copy** of the Post Transfer Command. The value can be a literal, regular expression (source), or pattern name (destination). For outbound (destination), the directory is created if it doesn't already exist. Special characters are allowed. For example, the pattern **%f** means the original working directory name. The expansion of any special characters is carried out each time this parameter is used. See Table 18. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A directory name and path on the external system. Special characters are allowed.
Remote Dir Name Is Pattern	Specifies whether the Remote Directory Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 18.	Select **Yes** or **No**. The configured default is **Yes**.
Remote File Name	Specifies the file name on the external system, to which a file is renamed or copied. This setting is only for the **Rename** or **Copy** operations of **Post Transfer Command** parameter. The value can be a literal, regular expression (get), or pattern name (put). Special characters are allowed, for example, the pattern **%f** indicates the original working file name. The expansion of any special characters is carried out each time this parameter is used. See Table 18. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The file name.

TABLE 18 Connectivity Map - BatchFTPOverSSL - Post Transfer *(Continued)*

Name	Description	Required Value
Remote File Name Is Pattern	Specifies whether the Remote File Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. See Table 18.	Select **Yes** or **No**. The configured default is **Yes**.
Remote Post Command	Allows you to execute a desired action directly after the actual file transfer. For an inbound transfer, it can be applied to mark the transferred file as consumed by making an automatic backup (**Rename**) or by destroying it permanently (**Delete**). For an outbound transfer, it can be applied to make the transferred file available to other clients by renaming it. ■ **Rename**: Rename the transferred file. ■ **Delete**: Delete the transferred file. ■ **None**: Do nothing. Note: For **Rename**, if the destination file exists, different FTP servers may behave differently. For example, on some UNIX FTP servers, the destination file will be overwritten without extra message. On an NT FTP server, this will fail and get an exception. It does not define unified behavior, rather, it follows the native behavior of the corresponding FTP server.	Select **Rename**, **Delete**, or **None**. The configured default is **None**.
Local Dir Name	Specifies the directory name (path) to be used by **Rename**. The value can be a literal, regular expression (source), or pattern name (destination). Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used. **Note –** For path separator, use the forward slash " / " instead of the back slash " \ ". The adapter interprets the back slash as a special character. For example, use `c:/temp/dir`. See Table 18. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The local directory name.

TABLE 18 Connectivity Map - BatchFTPOverSSL - Post Transfer *(Continued)*

Name	Description	Required Value
Local Dir Name Is Pattern	Specifies whether the Local Directory Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 18.	Select **Yes** or **No**. The configured default is **No**.
Local File Name	Specifies the file name to be used by **Rename**. The value can be a literal, regular expression (get), or pattern name (put). Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used. **Note –** For path seperator, use the forward slash " / " instead of the back slash " \ ". The adapter interprets the back slash as a special character. For example, use `c:/temp/dir`. See Table 18. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A file name.
Local File Name Is Pattern	Specifies whether the Local File Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 18.	Select **Yes** or **No**. The configured default is **No**.

TABLE 18 Connectivity Map - BatchFTPOverSSL - Post Transfer *(Continued)*

Name	Description	Required Value
Local Post Command	Allows you to execute a desired action directly after the actual file transfer. For an inbound transfer, the target file can be marked as **consumed** by making an automatic backup (**Rename**) or by destroying it permanently (**Delete**). For an outbound transfer the target file can be made available to other clients by renaming it. The options are: ■ **Rename**: Rename the target file. ■ **Delete**: Delete the target file (inbound transfers only). ■ **None**: Do nothing. **Note –** Rename overwrites the file specified by the Local Dir Name and Local File Name properties, if they exist.	Select **Rename**, **Delete**, or **None**. The configured default is **None**.

Firewall Settings (BatchFTPOverSSL Connectivity Map)

The Firewall Settings section of the BatchFTPOverSSL Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 19 Connectivity Map - BatchFTPOverSSL - Firewall Settings

Name	Description	Required Value
Use Firewall	Specifies whether you are using a firewall. If a firewall is used, supports **SOCKS 4** and **5**.	**Yes** or **No**. **Yes** indicates that you are using a firewall. The configured default is **No**.
SOCKS version	Specifies the SOCKS version of the firewall. The supported options are **4** for SOCKS version 4, or **5** for SOCKS version **5.**	Select **4** for **SOCKS version 4**, or **5** for **SOCKS version 5** .

Synchronization (BatchFTPOverSSL Connectivity Map)

The Synchronization section of the BatchFTPOverSSL Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 20 Connectivity Map - BatchFTPOverSSL - Synchronization

Name	Description	Required Value
Synchronized	Specifies whether the adapter simulates the pre- version 5.1 adapter behavior in which the adapter runs synchronized or in parallel. The selections are: ■ **Yes**: The adapter runs in synchronized mode, one instance of the Collaboration after the other. ■ **No**: The adapter runs in parallel, creating multiple instances of the Collaboration that run in parallel. **Note –** All OTD instances used in a Project should have the same value for this property.	Select **Yes** or **No.** The configured default is **No**.

BatchFTPOverSSL Adapter Environment Properties

This section describes the configuration properties for the **BatchFTPOverSSL OTD,** accessed from the Environment Explorer.

The BatchFTPOverSSL Adapter Environment properties include the following sections:

FTP and SSL Settings (BatchFTPOverSSL Environment)

The FTP and SSL Settings section of the BatchFTPOverSSL Environment properties contains the top-level parameters displayed in this table.

TABLE 21 Environment - BatchFTPOverSSL - FTP and SSL Settings

Name	Description	Required Value
FTP Host	Specifies the **FTP server host name or IP address.**	The FTP server host name or IP address. The configured default is **localhost**.
Explicit port for FTP over SSL	Specifies the **FTP port for explicit SSL. default is 21 (data port 20).**	The FTP port number for explicit SSL. The configured default is **21** (data port 20).

TABLE 21 Environment - BatchFTPOverSSL - FTP and SSL Settings *(Continued)*

Name	Description	Required Value
Implicit port for FTP over SSL	Specifies the **FTP port for implicit SSL.**	The **FTP port for implicit SSL** **The configured default is 990 (data port 989).**
User ID	Specifies the user login for FTP server.	The user login name for FTP server.
Password	Specifies the password for the FTP server user login.	The password for the FTP server user login.
Key Store Location	Specifies the path to the keystore that contains the trusted CA certificates required for server authentication.	The fully qualified path to the keystore file**.**
Key Store Password	Specifies the password to access the keystore file.	**The password for the keystore.**
Key Store Type	Specifies the keystore format type. Selections include **JKS** or **other**. **Note –** JKS is currently the only supported keystore type. If "other" is selected as the value, an exception is thrown when the OTD is initialized, with the error message, "Unknown type key store".	Select JKS. **JKS** is currently the only supported keystore type, and the configured default.

Firewall Settings (BatchFTPOverSSL Environment)

The Firewall Settings section of the BatchFTPOverSSL Environment properties contains the top-level parameters displayed in this table.

TABLE 22 Environment - BatchFTPOverSSL - Firewall Settings

Name	Description	Required Value
Firewall Host	Specifies **the proxy server host name or IP.**	The proxy server host name or IP address.
Firewall Port	Specifies **the proxy server port.**	The proxy server port number.
User ID	Specifies **the user login on the proxy server.**	User login ID.
Password	Specifies **the password for the user login.**	The user password.

General Settings (BatchFTPOverSSL Environment)

The General Settings section of the BatchFTPOverSSL Environment properties contains the top-level parameters displayed in this table.

TABLE 23 Environment - BatchFTPOverSSL - General Settings

Name	Description	Required Value
Connection Mode	Specifies whether a physical connection is established when an external connection is instantiated. Options are: ■ **Automatic**: Establishes a physical connection when an external connection is instantiated. ■ **Manual**: Does not automatically establish a physical connection when an external connection is instantiated. If a physical connection is not automatically established, a physical connection must be established from the Collaboration (for example, by calling the `connect()` method). If BatchFTPOverSSL Connection Mode is set to **Automatic**, the following Environment parameters must be set with valid values: FTP and SSL Settings ⇒ **FTP Host** FTP and SSL Settings ⇒ **Explicit port for FTP over SSL** (this must be set if the BatchFTPOverSSL Connectivity Map property, **FTP and SSL Settings** ⇒ **Secure Mode** is set to **Explicit SSL**) FTP and SSL Settings ⇒ **Implicit port for FTP over SSL** (this must be set if the BatchFTPOverSSL Connectivity Map property, **FTP and SSL Settings** ⇒ **Secure Mode** is set to **Implicit SSL**) FTP and SSL Settings ⇒ **User ID** FTP and SSL Settings ⇒ **Password** FTP and SSL Settings ⇒ **Key Store Location** (this must be set if the BatchFTPOverSSL Connectivity Map property, **FTP and SSL Settings** ⇒ **Require Server Authentication** is set to **Yes**, or if the BatchFTPOverSSL Connectivity Map property, **FTP and SSL Settings** ⇒ **Distinguished Name for User** contains a value, indicating that client authentication is required) FTP and SSL Settings ⇒ **Key Store Password** (this must be set if **Key Store Location** is set) Also, if a firewall is enabled and the Connectivity Map property, **Firewall Settings** ⇒ **Use Firewall**, is set to **Yes**, the following Environment parameters must be set with valid values: Firewall Settings ⇒ **Firewall Host** Firewall Settings ⇒ **Firewall Port** Firewall Settings ⇒ **User** Firewall Settings ⇒ **Password**	Select **Automatic** or **Manual**. The default is **Automatic**.

TABLE 23 Environment - BatchFTPOverSSL - General Settings *(Continued)*

Name	Description	Required Value
Temp Dir	Specifies a working directory for storing intermediary results. Options include: ■ **Leave value blank**: BatchFTPOverSSL will use a default location as the temporary directory. ■ Specify a path to a local file system folder with read/write permissions.	A working directory with read/write permissions, or leave blank (no value) to accept a default directory.

Connection Pool Settings (BatchFTPOverSSL Environment)

The Connection Pool Settings section of the BatchFTPOverSSL Environment properties contains the top-level parameters displayed in this table.

TABLE 24 Environment - BatchFTPOverSSL - Connection Pool Settings

Name	Description	Required Value
Steady pool size	Specifies the minimum number of physical connections the pool keeps available at all times.	An integer indicating the maximum number of connections available at all times. A value of **0** (zero) indicates that there are no physical connections in the pool and that new connections are created as needed. The configured default is **2**.
Maximum pool size	**Specifies the maximum number of physical connections the pool contains. 0 (zero) indicates that there is no maximum.**	An integer indicating the maximum number of connections allowed. A value of 0 **(zero) indicates that there is no maximum**. The configured default is **10**.

TABLE 24 Environment - BatchFTPOverSSL - Connection Pool Settings *(Continued)*

Name	Description	Required Value
Max Idle Timeout In Seconds	**Specifies the maximum idle timeout (in seconds)**. This is a hint to the server. A timer thread periodically removes unused connections. This parameter defines the interval at which this thread runs. This thread removes unused connections after the specified idle time expires. It allows the user to specify the amount of time a connection can remain idle in the pool.	An integer indicating **the maximum idle timeout in seconds**. When this is set to a number greater than **0**, the container removes or destroys any connections that are idle for the specified duration. A value of **0** specifies that idle connections can remain in the pool indefinitely. The configured default is **300** (5 minutes).

Connection Retry Settings (BatchFTPOverSSL Environment)

The Connection Retry Settings section of the BatchFTPOverSSL Environment Map properties contains the top-level parameters displayed in this table.

TABLE 25 Environment - BatchFTPOverSSL - Connection Retry Settings

Name	Description	Required Value
Connection Retries	Specifies the number of retries to establish a connection upon failure to acquire a connection.	An integer indicating the maximum number of retries to establish a connection upon failure to acquire a connection. The default value is **0**.
Connection Retry Interval	**Specifies the length of the pause (in milliseconds) between each reattempt to access the destination file. Used in conjunction with the Connection Retries setting.**	An integer indicating **length of the pause (in milliseconds)**. The default value is **1000** (1 second).

BatchSCP Adapter Connectivity Map Properties

This section describes the configuration properties for the **BatchSCP OTD,** accessed from the Connectivity Map.

The BatchSCP Adapter Connectivity Map properties include the following sections:

- "SCP Settings (BatchSCP Connectivity Map)" on page 68
- "Firewall Settings (BatchSCP Connectivity Map)" on page 68
- "Synchronization (BatchSCP Connectivity Map)" on page 69

SCP Settings (BatchSCP Connectivity Map)

The SCP Settings section of the BatchSCP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 26 Connectivity Map - BatchSCP - SCP Settings

Name	Description	Required Value
Authentication Type	Specifies the client authentication type. The options are: ■ PASSWORD ■ HOSTBASED ■ PUBLICKEY Refer to your specific SSH server documentation for information regarding your authentication type.	Select **PASSWORD**, **HOST BASED**, or **PUBLICKEY**. The configured default is **PASSWORD**.
Do Host Key Verification	Specifies whether SSH server authentication by verification of the public key, is enabled.	Select **Yes** or **No**. **Yes** enables SSH server authentication by verifying the public key. The configured default is **Yes**.
Remote Directory	Specifies the directory on the SSH (with SFTP sub-system) server where data is sent or received. The accessibility of the directory is usually dependent upon the login user..	The remote directory.
Remote File	Specifies the name of a file on the remote server used to either receive published data, or hold data to be retrieved.	The remote file.
Local Directory	Specifies the local directory for files be sent to the remote server, or received from remote server.	A local directory.
Local File	Specifies the local file under local directory to be sent to remote, or receive data from remote.	The local file.
Is Copy Recursive	Specifies whether the copy is recursive (for example, copy all sub directories).	Select **Yes** or **No**. **Yes** indicates that the copy is recursive. The configured default is No.

Firewall Settings (BatchSCP Connectivity Map)

The Firewall Settings section of the BatchSCP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 27 Connectivity Map - BatchSCP - Firewall Settings

Name	Description	Required Value
Use Firewall	Specifies whether a firewall is used.	**Yes** or **No**. Yes indicates that you are using a firewall. The configured default is **No**.
SOCKS Version	Specifies the SOCKS version required by the firewall. The supported options are **4** for SOCKS version **4**, or **5** for SOCKS version **5**	Select **4** for SOCKS version 4, or **5** for SOCKS version **5**. The configured default is **5**.

Synchronization (BatchSCP Connectivity Map)

The **Synchronization** section of the BatchSCP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 28 Connectivity Map - BatchSCP - Synchronization

Name	Description	Required Value
Synchronized	Specifies whether the adapter simulates the pre- version 5.1 adapter behavior in which the adapter runs synchronized or in parallel. The selections are: ■ **Yes**: The adapter runs in synchronized mode, one instance of the Collaboration after the other. ■ **No**: The adapter runs in parallel, creating multiple instances of the Collaboration that run in parallel. **Note –** All OTD instances used in a Project should have the same value for this property.	**Yes** or **No.** The default setting is **No**.

BatchSCP Adapter Environment Properties

This section describes the configuration properties for the **BatchSCP OTD,** accessed from the Environment Explorer.

The BatchSCP Adapter Environment properties include the following sections:

SSH Settings (BatchSCP Environment)

The SSH Settings section of the BatchSCP Environment properties contains the top-level parameters displayed in this table.

TABLE 29 Environment - BatchSCP - SSH Settings

Name	Description	Required Value
SSH Host	Specifies the host name or IP address of the SSH server.	The host name or IP address of the SSH server.
SSH Port	Specifies the port number of the SSH server.	The port number of the SSH server.
User	Specifies the user login name for the SSH server.	A user login name for the SSH server.
Password	Specifies a login password for the user.	A login password.
Key File	Specifies the folder (path) that holds the private key for client authentication.	The folder (path) that holds the private key for client authentication.
Key File Password	Specifies the password used to protect the key file (key pair).	The Key File password.
Server Public Key	Specifies the folder (path) that holds the public key used by the SSH server. This key is generated on the server and sent to the client through a, safe channel, and stored on the client machine for host key verification. See your specific SSH server documentation for more information.	The the folder (path) that contains the server public key for host key verification.
Server Name For Host Key Verification	Specifies the full domain name of the SSH server used for host key verification.	The expected SSH host name when doing host key verification.
Preferred Public Key Algorithm	Specifies the preferred public key algorithm for SSH authentication. The options are **DSA** and **RSA**.	**DSA** or **RSA**. The configured default is DSA.

Firewall Settings (BatchSCP Environment)

The Firewall Settings section of the BatchSCP Environment properties contains the top-level parameters displayed in this table.

TABLE 30 Environment - BatchSCP - Firewall Settings

Name	Description	Required Value
Firewall Host	Specifies **the proxy server host name or IP address.**	The proxy server host name or IP address. The configured default is **localhost**.
Firewall Port	Specifies **the proxy server port number.**	The proxy server port number. The configured default is **1080**.
Password	Specifies **a login password for the proxy server.**	A password for the proxy server login ID.
User	Specifies **a user login ID for the proxy server.**	A **user login ID for the proxy server.**

General Settings (BatchSCP Environment)

The General Settings section of the BatchSCP Environment properties contains the top-level parameters displayed in this table.

TABLE 31 Environment - BatchSCP - General Settings

Name	Description	Required Value
Connection Mode	Specifies whether a physical connection is established when an external connection is instantiated. Options are: ■ **Automatic**: Establishes a physical connection when an external connection is instantiated. ■ **Manual:** Does not automatically establish a physical connection when an external connection is instantiated. If a physical connection is not automatically established, a physical connection must be established from the Collaboration (for example, by calling the `connect()` method). If BatchSCP Connection Mode is set to **Automatic**, the following Environment parameters must be set with valid values: SCP Settings ⇒ **SSH Host** SCP Settings ⇒ **SSH Port** SCP Settings ⇒ **User** SCP Settings ⇒ **Password** SCP Settings ⇒ **Key File** (this must be set if the BatchSCP Connectivity Map property, **SCP Settings** ⇒ **Authentication Type** is set to **HOSTBASED** or **PUBLICKEY**) SCP Settings ⇒ **Key File Password** (required by the **Key File** property) SCP Settings ⇒ **Server Public Key** (this must be set if the BatchSCP Connectivity Map property, **SCP Settings** ⇒ **Do Host Key Verification** is set to **Yes**) SCP Settings ⇒ **Server Name for Host Key Verification** (this must be set if the BatchSCP Connectivity Map property, **SCP Settings** ⇒ **Do Host Key Verification** is set to **Yes**) Also, if a firewall is enabled and the Connectivity Map property, **Firewall Settings** ⇒ **Use Firewall**, is set to **Yes**, the following Environment parameters must be set with valid values: Firewall Settings ⇒ **Firewall Host** Firewall Settings ⇒ **Firewall Port** Firewall Settings ⇒ **User** Firewall Settings ⇒ **Password**	Select **Automatic** or **Manual**. The configured default is **Automatic**.

Connection Pool Settings (BatchSCP Environment)

The **Connection Pool Settings** section of the BatchSCP Environment properties contains the top-level parameters displayed in this table.

TABLE 32 Environment - BatchSCP - Connection Pool Settings

Name	Description	Required Value
Steady pool size	Specifies the minimum number of physical connections the pool keeps available at all times.	An integer indicating the maximum number of connections available at all times. A value of **0** (zero) indicates that there are no physical connections in the pool and that new connections are created as needed. The configured default is **2**.
Maximum pool size	Specifies the maximum number of physical connections the pool contains. 0 (zero) indicates that there is no maximum.	An integer indicating the maximum number of connections allowed. A value of **0 (zero) indicates that there is no maximum**. The configured default is **10**.
Max Idle Timeout In Seconds	Specifies the maximum idle timeout (in seconds). This is a hint to the server. A timer thread periodically removes unused connections. This parameter defines the interval at which this thread runs. This thread removes unused connections after the specified idle time expires. It allows the user to specify the amount of time a connection can remain idle in the pool.	An integer indicating **the maximum idle timeout in seconds**. When this is set to a number greater than **0**, the container removes or destroys any connections that are idle for the specified duration. A value of **0** specifies that idle connections can remain in the pool indefinitely. The configured default is **300** (5 minutes).

Connection Retry Settings (BatchSCP Environment)

The Connection Retry Settings section contains the following top level parameters:

The General Settings section of the BatchSCP Environment Map properties contains the top-level parameters displayed in this table.

TABLE 33 Environment - BatchSCP - Connection Retry Settings

Name	Description	Required Value
Connection Retries	Specifies the number of retries to establish a connection upon failure to acquire a connection.	An integer indicating the maximum number of retries to establish a connection upon failure to acquire a connection. The default value is **0**.
Connection Retry Interval	Specifies the length of the pause (in milliseconds) between each reattempt to access the destination file. Used in conjunction with the Connection Retries setting.	An integer indicating **length of the pause (in milliseconds)**. The default value is **1000** (1 second).

BatchSFTP Adapter Connectivity Map Properties

This section describes the configuration properties for the **BatchSFTP OTD,** accessed from the Connectivity Map.

The BatchSFTP Adapter Connectivity Map properties include the following sections:

Pre Transfer (BatchSFTP Connectivity Map)

The Pre Transfer section allows user to customize the behaviors of protection/backup/recovery. This section describes the operation that will be performed on remote end or locally before the real file transfer.

The Pre Transfer section of the BatchSFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 34 Connectivity Map - BatchSFTP - Pre Transfer

Name	Description	Required Value
Remote Dir Name	Specifies the directory name (path) on the remote external system where the file the is renamed or copied. This is only for **Rename** or **Copy** of the Remote Pre Command. The value can be a literal, regular expression (source), or pattern name (destination). For outbound (destination), the directory is created if it doesn't already exist. Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used. For example, the pattern **%f** means the original working directory name. See Table 34. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A directory name.
Remote Dir Name Is Pattern	Specifies whether the Remote Directory Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 34.	Select **Yes** or **No**. The configured default is **Yes**.
Remote File Name	Specifies the file name on the external system, to which a file is renamed or copied. This setting is only for the **Rename** or **Copy** operations of **Pre Transfer Command** parameter. The value can be a literal, regular expression (get), or pattern name (put). Special characters are allowed, for example, the pattern **%f** indicates the original working file name. The expansion of any special characters is carried out each time this parameter is used. See Table 34. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The file name.

TABLE 34 Connectivity Map - BatchSFTP - Pre Transfer *(Continued)*

Name	Description	Required Value
Remote File Name Is Pattern	Specifies whether the Remote File Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ No: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 34.	Select **Yes** or **No**. The configured default is **Yes**.
Remote Pre Command	Allows you to execute a desired action directly before the actual file transfer. For an inbound transfer, the file can be made unavailable to other clients polling the target system with the same directory and file pattern or name. For an outbound transfer, you can perform an automatic backup/clean-up of the existing files. The options are: ■ **Rename**: Rename the target file for protection or recovery. ■ **Copy**: Copy the target file for backup or recovery. ■ **None**: Do nothing. To gain proper protection, backup, or recovery, you must choose the appropriate setting that serves your purpose. For example, to recover from failures on an outbound appending transfer, use the **Copy** setting. **Note –** When you are using **Rename**, if the destination file exists, different FTP servers can behave differently. For example, on some UNIX FTP servers, the destination file is overwritten without question. That is, no error or warning message is given. On other FTP servers, a Windows XP server for example, the system generates an error that results in exceptions being thrown in the called OTD method.Be sure you are familiar with the native behavior of the corresponding FTP server. If you are in doubt, try the action at the command prompt. If the action displays an error message, it may result in an exception being thrown in the Collaboration.	Select **Rename**, **Copy**, or **None**. The configured default is **None**. **Note –** The **Copy** option could slow system performance, especially if you are copying a large file.

TABLE 34 Connectivity Map - BatchSFTP - Pre Transfer *(Continued)*

Name	Description	Required Value
Local Dir Name	Specifies the local directory name (path) to be used by **Rename** or **Copy**. The value can be a literal, regular expression (source), or pattern name (destination). Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used. See Table 34. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162. **Note –** When entering a path separator, use the forward slash " / " instead of the back slash " \ ". The adapter interprets the back slash as a special character. For example, use `c:/temp/dir`.	A directory name.
Local Dir Name Is Pattern	Specifies whether the Local Directory Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. See Table 34.	Select **Yes** or **No**. The configured default is **No**.
Local File Name	Specifies the local file name to be used by **Rename** or **Copy**. The value can be a literal, regular expression (get), or pattern name (put). Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used. See Table 34. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A file name.
Local File Name Is Pattern	Specifies whether the Local File Name name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 34.	Select **Yes** or **No**. The configured default is **No**.

TABLE 34 Connectivity Map - BatchSFTP - Pre Transfer *(Continued)*

Name	Description	Required Value
Local Pre Command	Allows you to execute a desired action directly before the actual file transfer. For an inbound transfer, the file can be made unavailable to other clients polling the target system with the same directory and file pattern or name. For an outbound transfer, you can perform an automatic backup of the existing files. The options are: ■ **Rename**: Rename the target file for protection or recovery. ■ **Copy**: Copy the target file for backup or recovery. ■ **None**: Do nothing. To gain proper protection, backup, or recovery, you must choose the appropriate setting that serves your purpose. For example, to recover from failures on an outbound appending transfer, use the **Copy** setting. **Note –** Rename and Copy overwrite the file specified by the Local Dir Name and Local File Name properties, if they exist. See "Using Name Patterns" on page 162.	Select **Rename**, **Copy**, or **None**. The configured default is **None**. **Note –** The **Copy** option could slow system performance, especially if you are copying a large file.

SFTP Settings (BatchSFTP Connectivity Map)

The SFTP Settings section of the BatchSFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 35 Connectivity Map - BatchSFTP - SFTP Settings

Name	Description	Required Value
Transfer Mode	Specifies whether the transfer is binary code or ASCII text.	**Select BINARY or ASCII.** The configured default is **BINARY**.
Remote EOL	Specifies the remote server - end of line. Options are **CR**, **LF**, **CRLF**.	Select **CR**, **LF**, or **CRLF**. CRLF is the configured default.
Transfer Block Size	Specifies the block size used when transferring files. Do not increase the default, as the remote server may not be able to support higher blocksizes.	An integer indicating the block size used when transferring files. The configured default is **32768**.

TABLE 35 Connectivity Map - BatchSFTP - SFTP Settings *(Continued)*

Name	Description	Required Value
Local Read Buffer Size	Specifies the size (in bytes) of the buffer which is used to read from the local file system.	An integer indicating the size (in bytes) of the local read buffer. A value of **-1** indicates that the whole local file is read at once.
Authentication Type	Specifies the client authentication type. The options are as follows: ■ PASSWORD ■ HOSTBASED ■ PUBLICKEY Refer to your specific SSH server documentation for information regarding your authentication type.	Select **PASSWORD**, **HOST BASED**, or **PUBLICKEY**. The configured default is **PASSWORD**.
Do Host Key Verification	Specifies whether SSH server authentication by verification of the public key, is enabled.	Select **Yes** or **No**. **Yes** enables SSH server authentication by verifying the public key. The configured default is **Yes**.
Remote Directory	Specifies the directoryname (path) on the SSH (with SFTP sub-system) server where data is sent or received. The accessibility of the directory usually depends on the login user. The value can be a literal, regular expression (source), or pattern name (destination). See Table 35. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The remote directory name.
Remote Directory Name is Pattern	Specifies whether the Remote Directory Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 35.	**Select Yes** or **No.** The configured default is **No**.

TABLE 35 Connectivity Map - BatchSFTP - SFTP Settings *(Continued)*

Name	Description	Required Value
Remote File	Specifies the name of a file on the remote server. The value can be a literal, regular expression (get), or pattern name (put). See Table 35. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The remote file.
Remote File Name is Pattern	Specifies whether the Remote File Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 35.	**Select Yes** or **No.** The configured default is **No.**
Local Directory	Specifies the local directory name (path) for sending or recieve files on the remote server. The value can be a literal, regular expression (source), or pattern name (destination). See Table 35. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A local directory.
Local Directory Name is Pattern	Specifies the meaning of the **Local Directory Name** property as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 35.	**Select Yes** or **No.** The configured default is **No**.
Local File	Specifies the local file to be sent or received on the remote server. The value can be a literal, regular expression (get), or pattern name (put). See Table 35. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The local file.

TABLE 35 Connectivity Map - BatchSFTP - SFTP Settings *(Continued)*

Name	Description	Required Value
Local File Name is Pattern	Specifies whether the local file name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 35.	**Select Yes** or **No.** The configured default is **No**.

Post Transfer (BatchSFTP Connectivity Map)

The Post Transfer section of the BatchSFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 36 Connectivity Map - BatchSFTP - Post Transfer

Name	Description	Required Value
Remote Dir Name	Specifies the directory name (path) on the remote system where the file the will be renamed or copied. T This is only for **Rename** or **Copy** of the Post Transfer Command. The value can be a literal, regular expression (source), or pattern name (destination). For outbound (destination), the directory is created if it doesn't already exist. Special characters are allowed. For example, the pattern **%f** means the original working directory name. The expansion of any special characters is carried out each time this parameter is used. See Table 36. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A directory name.

TABLE 36 Connectivity Map - BatchSFTP - Post Transfer *(Continued)*

Name	Description	Required Value
Remote Dir Name Is Pattern	Specifies whether the Remote Directory Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 36.	Select **Yes** or **No**. The configured default is **Yes**.
Remote File Name	Specifies the file name on the external system. This setting is only for the **Rename** or **Copy** operations of **Post Transfer Command** parameter. The value can be a literal, regular expression (get), or pattern name (put). Special characters are allowed, for example, the pattern **%f** indicates the original working file name. The expansion of any special characters is carried out each time this parameter is used. See Table 36. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A file name.
Remote File Name Is Pattern	Specifies whether the Remote File Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ No: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 36.	Select **Yes** or **No**. The configured default is **Yes**.

TABLE 36 Connectivity Map - BatchSFTP - Post Transfer *(Continued)*

Name	Description	Required Value
Remote Post Command	Allows you to execute a desired action directly after the actual file transfer. For an inbound transfer, it can be applied to mark the transferred file as consumed by making an automatic backup (**Rename**) or by destroying it permanently (**Delete**). For an outbound transfer, it can be applied to make the transferred file available to other clients by renaming it. ■ **Rename**: Rename the transferred file. ■ **Delete**: Delete the transferred file. ■ **None**: Do nothing. **Note –** For **Rename** , if the destination file exists, different FTP servers may behave differently. For example, on some UNIX FTP servers, the destination file will be overwritten without extra message. On an NT FTP server, this will fail and get an exception. It does not define unified behavior, rather, it follows the native behavior of the corresponding FTP server.	Select **Rename**, **Delete**, or **None**. The configured default is **None**.
Local Dir Name	Specifies the local directory name (path) to be used by **Rename**. The value can be a literal, regular expression (source), or pattern name (destination). Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used. See Table 36. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162. When entering a path separator, use the forward slash "/" instead of the back slash "\". The adapter interprets the back slash as a special character. For example, use `c:/temp/dir`.	A directory name.
Local Dir Name Is Pattern	Specifies whether the Local Directory Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 36.	Select **Yes** or **No**.The configured default is **No**.

TABLE 36 Connectivity Map - BatchSFTP - Post Transfer *(Continued)*

Name	Description	Required Value
Local File Name	Specifies the local file name to be used by **Rename**. The value can be a literal, regular expression (get), or pattern name (put). Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used. See Table 36. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	A file name.
Local File Name Is Pattern	Specifies whether the Local File Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 36.	Select **Yes** or **No**. The configured default is **No**.
Local Post Command	Allows you to execute a desired action directly after the actual file transfer. For an inbound transfer, the target file can be marked as consumed by making an automatic backup (**Rename**) or by destroying it permanently (**Delete**). For an outbound transfer the target file can be made available to other clients by renaming it. The options are: ■ **Rename**: Rename the target file. ■ **Delete**: Delete the target file (inbound transfers only). ■ **None**: Do nothing. **Note –** Rename overwrites the file specified by the Local Dir Name and Local File Name properties, if they exist.	Select **Rename**, **Delete**, or **None**. The configured default is **None**.

Firewall Settings (BatchSFTP Connectivity Map)

The Firewall Settings section of the BatchSFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 37 Connectivity Map - BatchSFTP - Firewall Settings

Name	Description	Required Value
Use Firewall	Specifies whether a firewall is used.	Select **Yes** or **No**. Yes indicates that you are using a firewall. The configured default is **No**.
SOCKS Version	Specifies the SOCKS version of the firewall. The supported options are **4** for SOCKS version **4**, or **5** for SOCKS version **5**	Select **4** for SOCKS version 4, or **5** for SOCKS version **5**. The configured default is **5**

Synchronization (BatchSFTP Connectivity Map)

The **Synchronization** section of the BatchSFTP Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 38 Connectivity Map - BatchSFTP - Synchronization

Name	Description	Required Value
Synchronized	Specifies whether the adapter simulates the pre- version 5.1 adapter behavior in which the adapter runs synchronized or in parallel. The selections are: ■ **Yes**: The adapter runs in synchronized mode, one instance of the Collaboration after the other. ■ **No**: The adapter runs in parallel, creating multiple instances of the Collaboration that run in parallel. **Note –** All OTD instances used in a Project should have the same value for this property.	**Select Yes** or **No.** The default setting is **No**.

BatchSFTP Adapter Environment Properties

This section describes the configuration properties for the **BatchSFTP OTD,** accessed from the Environment Explorer.

The BatchSFTP Adapter Environment properties include the following sections:

SFTP Settings (BatchSFTP Environment)

The SFTP Settings section of the BatchSFTP Environment properties contains the top-level parameters displayed in this table.

TABLE 39 Environment - BatchSFTP - SFTP Settings

Name	Description	Required Value
SSH Host	Specifies the host name or IP address of the SSH server.	The host name or IP address of the SSH server.
SSH Port	Specifies the port number of the SSH server.	The port number of the SSH server.
User ID	Specifies the user login name for the SSH server.	A user login name for the SSH server.
Password	Specifies a login password for the user.	A password.
Key File	Specifies the folder (path) that holds the private key for client authentication of type PUBLIC KEY or HOST BASED..	The folder (path) that holds the private key for client authentication.
Key File Password	Specifies the password used to protect the key file.	The Key File password.
Server Public Key	Specifies the folder (path) that holds the public key used by the SSH server. This key is generated on the server and sent to the client through a, safe channel, and stored on the client machine for host key verification.	The the folder (path) that contains the server public key for host key verification.
Server Name For Host Key Verification	Specifies the full domain name of the SSH server used for host key verification.	The expected SSH host name when doing host key verification.
Preferred Public Key Algorithm	Specifies the preferred public key algorithm for SSH authentication. The options are **DSA** and **RSA**	**DSA** or **RSA**. The configured default is DSA.

Firewall Settings (BatchSFTP Environment)

The Firewall Settings section of the BatchSFTP Environment properties contains the top-level parameters displayed in this table.

TABLE 40 Environment - BatchSFTP - Firewall Settings

Name	Description	Required Value
Firewall Host	Specifies **the proxy server host name or IP address.**	The proxy server host name or IP address. The configured default is **localhost**.
Firewall Port	Specifies **the proxy server port number.**	The proxy server port number. The configured default is **1080**.
User ID	Specifies **a user login ID for the proxy server.**	A **user login ID for the proxy server.**
Password	Specifies **a login password for the proxy server.**	A password for the proxy server login ID.

General Settings (BatchSFTP Environment)

The General Settings section of the BatchSFTP Environment properties contains the top-level parameters displayed in this table.

TABLE 41 Environment - BatchSFTP - General Settings

Name	Description	Required Value
Connection Mode	Specifies whether a physical connection is established when an external connection is instantiated. Options: ■ **Automatic:** Establishes a physical connection when an external connection is instantiated. ■ **Manual:** Does not automatically establish a physical connection when an external connection is instantiated. If a physical connection is not automatically established, a physical connection must be established from the Collaboration (for example, by calling the `connect()` method). If BatchSFTP Connection Mode is set to **Automatic**, the following Environment parameters must be set with valid values: SFTP Settings ⇒ **SSH Host** SFTP Settings ⇒ **SSH Port** SFTP Settings ⇒ **User** SFTP Settings ⇒ **Password** SFTP Settings ⇒ **Key File** (this must be set if the BatchSFTP Connectivity Map property, **SFTP Settings ⇒ Authentication Type**, is set to **HOSTBASED** or **PUBLICKEY**) SFTP Settings ⇒ **Key File Password** (required by the **Key File** property) SFTP Settings ⇒ **Server Public Key** (this must be set if the BatchSFTP Connectivity Map property, **SFTP Settings ⇒ Do Host Key Verification** is set to **Yes**) SFTP Settings ⇒ **Server Name for Host Key Verification** (this must be set if the BatchSFTP Connectivity Map property, **SFTP Settings ⇒ Do Host Key Verification** is set to **Yes**) Also, if a firewall is enabled and the Connectivity Map property, **Firewall Settings > Use Firewall**, is set to **Yes**, the following Environment parameters must be set with valid values: Firewall Settings ⇒ **Firewall Host** Firewall Settings ⇒ **Firewall Port** Firewall Settings ⇒ **User** Firewall Settings ⇒ **Password**	Select **Automatic** or **Manual**. The configured default is **Automatic**.

Connection Pool Settings (BatchSFTP Environment)

The **Connection Pool Settings** section of the BatchSFTP Environment properties contains the top-level parameters displayed in this table.

TABLE 42 Environment - BatchSFTP - Connection Pool Settings

Name	Description	Required Value
Steady pool size	Specifies the minimum number of physical connections the pool keeps available at all times.	An integer indicating the maximum number of connections available at all times. A value of **0** (zero) indicates that there are no physical connections in the pool and that new connections are created as needed. The configured default is **2**.
Maximum pool size	**Specifies the maximum number of physical connections the pool contains. 0 (zero) indicates that there is no maximum.**	An integer indicating the maximum number of connections allowed. A value of 0 **(zero) indicates that there is no maximum**. The configured default is **10**.
Max Idle Timeout In Seconds	**Specifies the maximum idle timeout (in seconds)**. This is a hint to the server. A timer thread periodically removes unused connections. This parameter defines the interval at which this thread runs. This thread removes unused connections after the specified idle time expires. It allows the user to specify the amount of time a connection can remain idle in the pool.	An integer indicating **the maximum idle timeout in seconds**. When this is set to a number greater than **0**, the container removes or destroys any connections that are idle for the specified duration. A value of **0** specifies that idle connections can remain in the pool indefinitely. The configured default is **300** (5 minutes).

Connection Retry Settings (BatchSFTP Environment)

The Connection Retry Settings section contains the following top level parameters:

The General Settings section of the BatchSFTP Environment Map properties contains the top-level parameters displayed in this table.

TABLE 43 Environment - BatchSFTP - Connection Retry Settings

Name	Description	Required Value
Connection Retries	Specifies the number of retries to establish a connection upon failure to acquire a connection.	An integer indicating the maximum number of retries to establish a connection upon failure to acquire a connection. The default value is **0**.
Connection Retry Interval	**Specifies the length of the pause (in milliseconds) between each reattempt to access the destination file. Used in conjunction with the Connection Retries setting.**	An integer indicating **length of the pause (in milliseconds)**. The default value is **1000** (1 second).

BatchLocalFile Connectivity Map Properties

This section explains the properties for the **BatchLocalFile OTD, accessed from the Connectivity Map**.

The BatchLocalFile properties include the following sections:

Caution – Several of these configuration options allow regular expressions to be used. This advanced feature is useful but must be used carefully. An improperly formed regular expression can cause undesired data or loss of data. You must have a clear understanding of regular-expression syntax and construction before attempting to use this feature. It is recommended that you test such configurations thoroughly before moving them to production.

Pre Transfer (BatchLocalFile Connectivity Map)

This section provides information about configuring the **Pre Transfer** parameters. Pre-transfer operations are those operations executed right before the actual data transfer.

The Pre Transfer section of the BatchLocalFile Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 44 Connectivity Map - BatchLocalFile - Pre Transfer

Name	Description	Required Value
Pre Directory Name	Specifies the directory name (path) on the external system in which a file is renamed or copied. This setting is only for the **Rename** or **Copy** operations of **Pre Transfer Command** parameter. The value can be a literal, or pattern name. For outbound transfers, the directory is created if it does not already exist. Special characters are allowed. For example, the pattern **%f** indicates the original working directory name. The expansion of any special characters is carried out each time this parameter is used. See **Note –** For path separator, use the forward slash " / " instead of the back slash " \ ". The adapter interprets the back slash as a special character. For example, use `c:/temp/dir`. See Table 44. See "Using Name Patterns" on page 162.	Enter the exact name of the directory (with the path), enter a pattern name, or select one of the following values: ■ %f ■ %f.%y%y%y%y%M%M%d%d.%h%h%m%m%s%s%S%S%S ■ %f.copy ■ %f.rename
Pre Directory Name Is Pattern	Specifies whether the Pre Directory Name represents a literal or a name pattern, as follows: ■ Yes: indicates that the name value you enter is assumed to be a name pattern. ■ No: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 44.	Select **Yes** or **No**. The configured default is **No**.
Pre File Name	Specifies the file name on the external system, to which a file is renamed or copied. This setting is only for the **Rename** or **Copy** operations of **Pre Transfer Command** parameter. The value can be a literal or pattern name. Special characters are allowed, for example, the pattern **%f** indicates the original working file name. The expansion of any special characters is carried out each time this parameter is used. See Table 44. See "Using Name Patterns" on page 162	Enter the exact name of the file, enter a pattern name, or select one of the following values: ■ %f ■ %f%# ■ %f.%y%y%y%y%M%M%d%d.%h%h%m%m%s%s%S%S%S ■ %f.copy ■ %f.rename

TABLE 44 Connectivity Map - BatchLocalFile - Pre Transfer *(Continued)*

Name	Description	Required Value
Pre File Name Is Pattern	Specifies whether the Pre File Name represents a literal or a name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 44.	Select **Yes** or **No**. The configured default is **No**.
Pre Transfer Command	Allows you to determine the action executed directly before the actual file transfer. In the case of an inbound file transfer, you can make the file unavailable to other clients polling the target system via the same directory and file pattern or name. In the case of an outbound transfer, you can make an automatic backup of the existing file. The options are as follows: ■ **Rename**: Rename the target file. ■ **Copy**: Copy the target file. ■ **None**: Do nothing.	Select **Rename**, **Copy**, or **None**; the default is **None**. **Note – Rename** and **Copy** overwrite the file or directory specified by the **Pre Directory Name** and **Pre Transfer Name** parameter, if it exists.

Note – For more information on this feature, see "Pre/Post File Transfer Commands" on page 148 .

Sequence Numbering (BatchLocalFile Connectivity Map)

The **Sequence Numbering** section of the BatchLocalFile Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 45 Connectivity Map - BatchLocalFile - Sequence Numbering

Name	Description	Required Value
Max Sequence Number	Use this parameter when you have set the target file name to contain a sequence number. It tells the adapter to reset the sequence number to the **Starting Sequence Number** value when this value (the **Max Sequence Number**) has been reached. This parameter is used for the name pattern %#. See "Using Name Patterns" on page 162.	An integer from **1** to **2147483647**. The value of **Max Sequence Number** *must* be greater than that of **Starting Sequence Number**. The configured default value is **999999**.
Starting Sequence Number	Use this parameter when you have set up the target file name to contain a sequence number. It tells the adapter which value to start with in the absence of a sequence number from a previous run. Also, when the **Max Sequence Number** value is reached, the sequence number rolls over to the **Starting Sequence Number** value. This parameter is used for the name pattern %#. See "Using Name Patterns" on page 162.	An integer from **0** to **2147483647**. The value of the **Starting Sequence Number** *must* be less than the **Max Sequence Number**. The configured default value is **1**.

Note – The Synchronized property, under General Settings, must be set to "Yes" to use Sequence Numbering.

Post Transfer (BatchLocalFile Connectivity Map)

Post-transfer operations are those performed after the data transfer.

The Post Transfer section of the BatchLocalFile Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 46 Connectivity Map - BatchLocalFile - Post Transfer

Name	Description	Required Value
Post Directory Name	Specifies the directory name (path) on the external system in which a file is renamed. This setting is only for the **Rename** operation of the **Post Transfer Command** parameter. The value can be a literal or pattern name. For outbound transfers, the directory is created if it does not already exist. Special characters are allowed, for example, the pattern **%f** indicates the original working directory name. The expansion of any special characters is carried out each time this parameter is used. **Note –** For path separator, use the forward slash " / " instead of the back slash " \ ". The adapter interprets the back slash as a special character. For example, use `c:/temp/dir`. See Table 46. See "Using Name Patterns" on page 162.	Enter the exact name of the directory (with the path), enter a pattern name, or select one of the following values: ■ %f ■ %f%# ■ %f.%y%y%y%y%M%M%d%d.%h%h%m%m%s%s%S%S%S ■ %f.copy ■ %f.rename
Post Directory Name Is Pattern	Specifies whether the Post Directory Name represents a literal or a name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 46.	Select **Yes** or **No.** The configured default is **No**.
Post File Name	Specifies either the name of the file that the transferred file is renamed to (Rename) or the directory it is moved to (Move), depending on the setting in the parameter **Post Transfer Command**. The value can be a literal or pattern name. Special characters are allowed. The expansion of any special characters are carried out each time this parameter is used. See Table 46. See "Using Name Patterns" on page 162.	Enter the exact name of the file, enter a pattern name, or select one of the following values:Select one of the following values: ■ %f ■ %f.%y%y%y%y%M%M%d%d.%h%h%m%m%s%s%S%S%S ■ %f.copy ■ %f.rename

TABLE 46 Connectivity Map - BatchLocalFile - Post Transfer *(Continued)*

Name	Description	Required Value
Post File Name Is Pattern	Specifies whether the Post File Name represents a literal or a name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 46.	Select **Yes** or **No.** The configured default is **No**.
Post Transfer Command	Allows you to execute a desired action directly after the actual file transfer. For an inbound transfer, you can mark the transferred file as "consumed" by making an automatic backup (**Rename**) or by destroying it permanently (**Delete**). For an outbound transfer, you can make the transferred file available to other clients by renaming it. The options are as follows: ■ **Rename**: Rename the target file. **Rename** overwrites the file specified by the **Post File Name** and **Post Directory Name**, if it exists ■ **Copy**: Copy the target file. ■ **Delete**: Delete the target file (inbound transfers only). ■ **None**: Do nothing.	Select **Rename**, **Copy**, **Delete**, or **None**. The configured default is **None**.

General Settings (BatchLocalFile Connectivity Map)

The General Settings section of the BatchLocalFile Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 47 Connectivity Map - BatchLocalFile - General Settings

Name	Description	Required Value
Resume Reading Enabled	Specifies whether the OTD handles the Resume Reading feature as follows: ■ **Yes**: Enables the OTD to store any state information necessary to resume reading from the current file in a subsequent execution of the Collaboration Rule. ■ **No**: Indicates that the file is considered "consumed" even if the streaming consumer did *not* read until the end of file.	**Yes** or **No** The configured default is **No**. **Note –** Synchronized must be set to "Yes" to use Resume Reading.
Synchronized	Specifically applies to legacy Batch Adapter Projects. Provides backward compatibility to allow Projects that were created using the Batch Adapter version 5.0.7 or earlier to be imported and deployed without a change in the adapter's behavior. The selections are: ■ Yes: Provides backward compatibility for legacy (pre-5.0.8 Batch Adapter) Projects. The adapter runs in synchronized mode, one instance of the Collaboration after the other. ■ No: For use with new Batch Adapter Projects. The adapter run in parallel, creating multiple instances of the Collaboration that run in parallel. **Note –** All OTD instances used in a Project should have the same value for this property.	**Yes** or **No.** The default setting is **Yes, simulating Projects created with** Batch Adapter version 5.0.7 or earlier. **Note –** Synchronized must be set to "Yes" to use Sequence Numbering or Resume Reading.

Target Location (BatchLocalFile Connectivity Map)

The Target Location section of the BatchLocalFile Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 48 Connectivity Map - BatchLocalFile - Target Location

Name	Description	Required Value
Append	Specifies whether to overwrite or append the data to the existing file. Use this parameter for outbound file transfers only. Choose the appropriate setting as follows: ■ **Yes:** If the target file already exists, the data is appended to the existing file. ■ **No:** The adapter overwrites the existing file on the remote system. If a file with the same name does not exist, both **Yes** and **No** create a new file on the external host.	Select **Yes** or **No** The configured default is **No**.
Target Directory Name	Specifies the directory name (path) on the local system from which files are retrieved or where they are sent. The value can be a literal, regular expression (source), or pattern name (destination). For outbound transfer (destination), the directory is created if it does not already exist. **Note –** For path separator, use the forward slash " / " instead of the back slash " \ ". The adapter interprets the back slash as a special character. For example, use `c:/temp/dir`. See Table 48. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162.	The directory name.
Target Directory Name Is Pattern	Specifies whether the Target Directory Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 48.	Select **Yes** or **No** The configured default is **Yes**.
Target File Name	Specifies the name of the file on the local system either to be retrieved or sent. The value can be a literal, regular expression (get), or pattern name (put). See Table 48. See "Using Regular Expressions" on page 159 or "Using Name Patterns" on page 162	A file name.

TABLE 48 Connectivity Map - BatchLocalFile - Target Location *(Continued)*

Name	Description	Required Value
Target File Name Is Pattern	Specifies whether the Target File Name represents a literal, or a regular expression or name pattern, as follows: ■ **Yes**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **No**: indicates that the name value entered is a literal, an exact match. See Table 48.	Select **Yes** or **No** The configured default is **No**.

BatchLocalFile Environment Properties

This section explains the properties for the **BatchLocalFile OTD, accessed from the Environment Explorer**.

The BatchLocalFile properties include the following sections:

- "General Settings (BatchLocalFile Environment)" on page 98
- "Connection Pool Settings (BatchLocalFile Environment)" on page 99

General Settings (BatchLocalFile Environment)

The **General Settings** section of the BatchLocalFile Environment properties contains the top-level parameters displayed in this table.

TABLE 49 Environment - BatchLocalFile - General Settings

Name	Description	Required Value
State Persistence Base Location	Specifies a working directory for storing intermediary results. Options: ■ Leave value blank: BatchLocalFile will use a default folder as the working directory. ■ Specify a path to a local file system folder with read/write permissions.	A working directory with read/write permissions, or leave blank (no value) to accept a default directory.

Connection Pool Settings (BatchLocalFile Environment)

The **Connection Pool Settings** section of the BatchLocalFile Environment properties contains the top-level parameters displayed in this table.

TABLE 50 Environment - BatchLocalFile - Connection Pool Settings

Name	Description	Required Value
Steady pool size	Specifies the minimum number of physical connections the pool keeps available at all times.	An integer indicating the maximum number of connections available at all times. A value of **0** (zero) indicates that there are no physical connections in the pool and that new connections are created as needed. The configured default is **2**.
Maximum pool size	**Specifies the maximum number of physical connections the pool contains. 0 (zero) indicates that there is no maximum.**	An integer indicating the maximum number of connections allowed. A value of 0 **(zero) indicates that there is no maximum**. The configured default is **10**.
Max Idle Timeout In Seconds	**Specifies the maximum idle timeout (in seconds)**. This is a hint to the server. A timer thread periodically removes unused connections. This parameter defines the interval at which this thread runs. This thread removes unused connections after the specified idle time expires. It allows the user to specify the amount of time a connection can remain idle in the pool.	An integer indicating **the maximum idle timeout in seconds**. When this is set to a number greater than **0**, the container removes or destroys any connections that are idle for the specified duration. A value of **0** specifies that idle connections can remain in the pool indefinitely. The configured default is **300** (5 minutes).

BatchRecord Connectivity Map Properties

This section explains the properties for the record-processing **BatchRecordOTD**. The BatchRecord properties include the following sections:

- "General Settings (BatchRecord Connectivity Map)" on page 100
- "Record (BatchRecord Connectivity Map)" on page 100

General Settings (BatchRecord Connectivity Map)

The General Settings section of the BatchRecord Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 51 Connectivity Map - BatchRecord - General Settings

Name	Description	Required Value
Parse or Create Mode	Specifies how this adapter Connection for the record-processing OTD is used. Set this parameter as follows: ■ **Parse**: To use the OTD for parsing an inbound payload. ■ **Create**: To use the OTD for creating an outbound payload. An instance of the OTD can be used for parsing an inbound payload (only) or for creating an outbound payload (only). A single OTD cannot be used for both purposes at the same time in the same Collaboration.	Select **Create** or **Parse**. The configured default is **Parse**.
Synchronized	Specifically applies to legacy Batch Adapter Projects. Provides backward compatibility to allow Projects that were created using the Batch Adapter version 5.0.7 or earlier to be imported and deployed without a change in the adapters behavior. The selections are: ■ **Yes**: Provides backward compatibility for legacy (pre-5.0.8 Batch Adapter) Projects. The adapter runs in synchronized mode, one instance of the Collaboration after the other. ■ **No**: For use with new Batch Adapter Projects. The adapter runs in parallel, creating multiple instances of the Collaboration that run in parallel. **Note –** All OTD instances used in a Project should have the same value for this property.	Select **Yes** or **No.** The default setting is **Yes**, simulating Projects created with Batch Adapter version 5.0.7 or earlier.

Record (BatchRecord Connectivity Map)

This section allows you to configure the **Record** parameters, to specify the record characteristics you want the adapter to recognize.

The Record section of the BatchRecord Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 52 Connectivity Map - BatchRecord - Record

Name	Description	Required Value
Delimiter on Last Record	Allows you to supply the delimiter to be used with the final record. Use this parameter only when the **Record Type** is set to **Delimited**. Some message formats insist that the final message in a record set has no trailing delimiter. However, in most cases, you can safely leave this parameter set to **Yes**.	Select **Yes** or **No.** The configured default setting is **Yes**.
Record Delimiter	Specifies the delimiter to be used for records. Use this parameter when the **Record Type** is set to **Delimited**. The value entered is interpreted as a sequence of one or more bytes. If there are multiple bytes in the delimiter, each must be separated by a comma. You can enter the delimiters in the following formats: ■ **ASCII Characters**: The adapter supports all ASCII characters.**Example**: *,*,* (records separated by ***)**Example**: | (records separated by a |) ■ **Escaped ASCII**: The adapter supports \r, \n, \t, and \f.**Example**: \r,\n (records separated by CR NL)**Example**: \n (records separated by NL only) ■ **Hex**: The adapter supports 0x00 to 0x7E**Example**: \0x0D,\0x0A (records separated by CR NL) ■ **Octal**: The adapter supports 000 to 0177.**Example**: \015,\012 (same as \0x0D,\0x0A) **Note –** When using character delimiters with DBCS data, use single byte character(s), or equivalent hex values that do not coincide with either byte of the double byte characters.When using escaped ASCII, Hex, or Octal, the "\" character is required.	A valid character to use as data record delimiter.
Record Size	Specifies a number indicating the record size (byte count). Use this parameter when the **Record Type** is set to **Fixed**, and a number indicating length must be supplied.	A number from **1** to **2,147,483,647** indicating the record size (byte count).

TABLE 52 Connectivity Map - BatchRecord - Record *(Continued)*

Name	Description	Required Value
Record Type	Specifies the format of the records in the data payload in the Collaboration. Each payload contains zero or more records. Using this and related parameters, you can pass records individually to another component within Java CAPS. Available options: ■ **Delimited**: The records are separated by the delimiter specified under the **Record Delimiter** parameter. ■ **Fixed**: The records are all of a given size, specified by the **Record Size** parameter. ■ **Single Record**: If the payload is to be processed "as-is," select this option. ■ User Defined: This option is not supported.	**Delimited**, **Fixed**, **or Single Record**. The configured default is **Delimited.**

BatchRecord Environment Properties

This section explains the properties for **BatchRecord, accessed from the Environment Explorer**.

The BatchRecord properties include the following:

■ "Connection Pool Settings (BatchRecord Environment)" on page 102

Connection Pool Settings (BatchRecord Environment)

The **Connection Pool Settings** section of the BatchRecord Environment properties contains the top-level parameters displayed in this table.

TABLE 53 Environment - BatchRecord - Connection Pool Settings

Name	Description	Required Value
Steady pool size	Specifies the minimum number of physical connections the pool keeps available at all times.	An integer indicating the maximum number of connections available at all times. A value of **0** (zero) indicates that there are no physical connections in the pool and that new connections are created as needed. The configured default is **2**.
Maximum pool size	**Specifies the maximum number of physical connections the pool contains. 0 (zero) indicates that there is no maximum.**	An integer indicating the maximum number of connections allowed. A value of 0 **(zero) indicates that there is no maximum**. The configured default is **10**.
Max Idle Timeout In Seconds	**Specifies the maximum idle timeout (in seconds)**. This is a hint to the server. A timer thread periodically removes unused connections. This parameter defines the interval at which this thread runs. This thread removes unused connections after the specified idle time expires. It allows the user to specify the amount of time a connection can remain idle in the pool.	An integer indicating **the maximum idle timeout in seconds**. When this is set to a number greater than **0**, the container removes or destroys any connections that are idle for the specified duration. A value of **0** specifies that idle connections can remain in the pool indefinitely. The configured default is **300** (5 minutes).

BatchInbound Connectivity Map Properties

This section explains the configuration parameters for the **BatchInbound Adapter (OTD), accessed from the Connectivity Map** (there are no Environment properties for BatchInbound).

The BatchInbound properties include the following sections:

- "Settings (BatchInbound Connectivity Map)" on page 103

Settings (BatchInbound Connectivity Map)

The Settings section of the BatchInbound Connectivity Map properties contains the top-level parameters displayed in this table.

TABLE 54 Connectivity Map - BatchInbound- Settings

Name	Description	Required Value
Directory Name	Specifies the input directory name (path). It indicates the directory that the BatchInbound Adapter polls for trigger or data files. The value can be a literal or a regular expression. **Note –** For path separator, use the forward slash " / " instead of the back slash " \ ". The adapter interprets the back slash as a special character. For example, use `c:/temp/dir`. See Table 54. See "Using Regular Expressions" on page 159 .	The directory name.
Directory Name is Pattern	Specifies whether the Directory Name represents a literal, or a regular expression, as follows: ■ **True**: indicates that the name value you enter is assumed to be a regular expression ■ **False**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 54. **Note –** Improper use may cause recursive matching.	Select **True** or **False.** The configured default is **False.**
File Name	Specifies the input filename. The value can be a literal or a regular expression. See Table 54. See "Using Regular Expressions" on page 159 .	A file name.
File Name is Pattern	Specifies whether the target file name represents a literal or a regular expression, as follows: ■ **True**: indicates that the name value you enter is assumed to be a name pattern or regular expression. ■ **False**: indicates that the name value entered is a literal, an exact match. No pattern matching or name expansion is done. See Table 54.	Select **True** or **False.** The configured default is True.

TABLE 54 Connectivity Map - BatchInbound- Settings *(Continued)*

Name	Description	Required Value
Schedule Interval	Specifies the polling interval, or number of seconds between each poll of the input directory by the adapter for input files.	A number indicating the length of time in Milliseconds between adapter polls of the directory. The configured default is **5000** (or 5 seconds).

BatchInbound Environment Properties

This section explains the configuration parameters for the **BatchInbound Adapter (OTD), accessed from the Environment Explorer**.

The BatchInbound properties include the following sections:

- "MDB Settings (BatchInbound Environment)" on page 105

MDB Settings (BatchInbound Environment)

The **MDB** Settings section of the BatchInbound Environment properties contains the top-level parameters displayed in this table.

TABLE 55 Connectivity Map - BatchInbound- Settings

Name	Description	Required Value
Max Pool Size	Specifies the maximum size of the MDB (Message Driven Bean) pool.	**An integer indicating the maximum MDB pool size.** The configured default is **1000**.

Using FTP Heuristics

Due to the diversified nature of FTP server directory listing styles, that is, the format in which the list of directory entries are returned to the client as the result of a LIST command, the directory listing parser requires FTP server information regarding its specific format, to parse the result. These server specific parameters that provide format information are called FTP heuristics. When these heuristic parameters are set for a specific platform, it is called a directory listing style.

This section provides a general explanation of how the FTP Heuristics feature of the adapter operates, as well as some basic information on how to use it. It also explains the FTP Heuristics configuration parameters for the adapter.

FTP Heuristics

FTP heuristics are a set of parameters that the adapter uses to interact with external FTP daemons on a platform-specific level. The FTP heuristics create and parse both path and file names in the style required by the external system's platform (operating system).

Platform Selection

The BatchFTP OTD provides a number of built-in heuristics configuration styles. The style is selected from the BatchFTP Adapter Connectivity Map property, **FTP ⇒ Directory Listing Style**. To select a style for your specific target platform do the following:

1. From the Connectivity Map, double-click the BatchFTP Adapter. The FTPBatch Adapter Properties Editor appears.
2. From the Editor's left pane, under the configuration tree, select **FTP**. The **FTP** parameters are now displayed in the Properties pane.
3. Click the **Directory Listing Style** field and select a system platform from the drop-down menu. The properties available from the drop-down menu, reflect the styles listed in the `FtpHeuristics.cfg` file.

The adapter's FTP Heuristics support the following platform types:

- UNIX
- AS400
- AS400-UNIX
- HCLFTPD 6.0.1.3
- HCLFTPD 5.1
- HP NonStop/Tandem
- MPE
- MSFTPD 2.0
- MSP PDS (Fujitsu)
- MSP PS (Fujitsu)
- MVS GDG (Generation Data Group)
- MVS PDS (Partitioned Data Sets)
- MVS Sequential
- NetWare 4.11

- Windows NT 3.5
- Windows NT 4.0
- UNIX
- UNIX (EUC-JP)
- UNIX (SJIS)
- User Defined
- User Defined1-10 (See "Creating User Defined Heuristic Directory Listing Styles" on page 108)
- VM/ESA
- VMS
- VOS3 PDS (Hitachi)
- VOS3 PS (Hitachi)
- VOSK (Hitachi)
- *User defined name and heuristic configuration (See* "Creating User Defined Heuristic Directory Listing Styles" on page 108

Note – When using FTP with an AS400 UNIX (UFS) system, some specific FTP configuration settings are required (see "FTP Configuration Requirements for AS400 UNIX (UFS)" on page 124).

The following systems support Japanese mainframes:

- MSP PDS (Partitioned Data Sets)
- MSP PS (Physical Sequential)
- VOS3 PDS (Partition Data Sets)
- VOS3 PS (Physical Sequential)
- VOSK

IBM IP stack required for MVS Sequential, MVS GDG, and MVS PD

The FTP Heuristic methods used to interface with **MVS Sequential**, **MVS GDG**, and **MVS PDS**, are designed for FTP servers (at the mainframe) that use the **IBM IP stack**. Therefore, when you use FTP with an **MVS Sequential, MVS GDG**, or **MVS PDS** file system on a mainframe computer, you need to make sure that the FTP server is using an **IBM IP stack**. If any other IP stack is in place, the FTP heuristics feature will not work or may require modification.

Creating User Defined Heuristic Directory Listing Styles

You can create "user defined" heuristic configurations that allow you to interface with other platforms that are not listed in the Directory Listing Styles. The Batch Adapter includes a mechanism that allows you to configure a set of heuristic properties so that the underlying parser can parse the LIST command result correctly. These properties are described under "FTP Heuristics Configuration Parameters" on page 113.

There are two methods for creating custom user defined directory listing styles:

- **Create a Custom Heuristics Configuration File**: You can create a custom user defined heuristics configuration file, listing the style names and parameters in the same format as the `FtpHeuristics.cfg` file. This file is then located on the logical host. The configuration file location and the style name are then specified in the BatchFTP configuration properties (see "To Create a Custom Heuristics Configuration File" on page 108).
- **Modify the FTP Heuristics Configuration File**: You can open `FtpHeuristics.cfg` file, add your user-defined style, and repackage the file. This method requires you to unzip a JAR file, add your custom style, and repackage the files (see "To Modify the FTP Heuristics Configuration File" on page 109). In many cases, this method may be more intrusive and cumbersome than the method listed above.

▼ To Create a Custom Heuristics Configuration File

1. **Using a text editor, create a user defined configuration file containing the property settings required to interface with your target platform. You can do this by copying a section (style) from the FtpHeuristics.cfg file that is similar to the style (platform parameter settings) that you are creating, or you can copy the format provided under "Heuristics Configuration File Format" on page 110.**

2. **Save your user defined configuration, as a CFG file, to a safe location on the logical host.**

3. **From the BatchFTP Environment properties, select the FTP ⇒ User Defined Heuristics Configuration File property, and enter the location and name of your user defined heuristics configuration file (for example `C:\USER_DEFINED_HEURISTICS\UDH.cfg`).**

4. **From the BatchFTP Connectivity Map properties, select FTP ⇒ User Defined Directory Listing Style, and enter the name of your user-named style (for example MY AS400-UNIX). You are allowed to list one user-named style. This style is now the configured Directory Listing Style, superseding the value of the Directory Listing Style property.**

   You can use this method to create multiple user-named styles by adding the styles to your user defined configuration file, and entering the different user defined style names in the Connectivity Map properties for each of your various FTPBatch component adapters.

You can also create multiple user defined configuration files if necessary, but this requires the creation of additional BatchFTP External Systems in the Environment. If you chose this method, you must copy your Environment components (drag-and-drop) to the correct BatchFTP External System before applying Automap.

Considerations

If you decide to use this method for creating custom user defined heuristic configurations, take note of the following:

- The BatchFTP Connectivity Map property, **User Defined Directory Listing Style,** supersedes the **Directory Listing Style** property. When a **User Defined Directory Listing Style** is specified, it is used as the heuristic configuration for the corresponding BatchFTP Adapter (OTD). To use the **Directory Listing Style** property value as the applied heuristic style, the **User Defined Listing Style** property value must be left blank.
- Setting the **User Defined Directory Listing Style** property value to blank (no value) makes the selected **Directory Listing Style** property value (built-in heuristic configuration) the current enabled style.
- At runtime, the user defined heuristics configuration file must exist on the logical host, and possess appropriate permission settings to allow the heuristic configuration parameters to be accessed by the deployed application.
- An error message is generated by the BatchFTP OTD when a **User Defined Directory Listing Style** is specified, but the **User Defined Heuristics Configuration File** property value is blank, or associated the user defined heuristics configuration file is not accessible or does not contain a corresponding style configuration.
- Setting the value of the **User Defined Directory Listing Style** triggers the loading of the corresponding heuristics configuration file specified by the **User Defined Heuristics Configuration File** property. If you make changes to the heuristics configuration file, set the **User Defined Heuristics Configuration File** property before setting the **User Defined Directory Listing Style**.

▼ To Modify the FTP Heuristics Configuration File

To modify the `FtpHeuristics.cfg` file to include your user defined heuristic configuration styles, do the following:

1. **The `FtpHeuristics.cfg` file is contained by the `stcbatch.jar` file, which is found in the following location:**

   `<JavaCAPS51>``\edesigner\usrdir\modules\ext\batcheway\stcbatch.jar`

   where *JavaCAPS51* is the Sun Java Composite Application Platform Suite install directory.

2. **Unzip `stcbatch.jar` and locate the `FtpHeuristics.cfg` file.**

3 **Open `FtpHeuristics.cfg` with a text editor and add your user defined heuristic configuration styles.**

▼ To Add User Defined Heuristic Configuration Styles

1 **Copy the User Defined section (or any other section), and paste it to the bottom of `FtpHeuristics.cfg`.**

2 **Rename the section and each property name with your user-defined name or one of the available listings (User Defined1, User Defined2, and so forth). See the example provided under "Heuristics Configuration File Format" on page 110. In this example, the user defined name is MY AS400-UNIX). Only one style with a user-defined name can be specified, but 10 configuration styles can be named as User Defined1-10.**

3 **Modify the new section's properties for your target platform. See "FTP Heuristics Configuration Parameters" on page 113 for property descriptions.**

4 **Repeat steps 2-4 above to create additional User Defined configurations.**

▼ To Repackage the `FtpHeuristics.cfg` File.

1 **Zip the `stcbatch.jar` file (including the updated `FtpHeuristics.cfg` file) and copy `stcbatch.jar` back to it's original location.**

2 **From the BatchFTP Configuration Map properties, select FTP ⇒ User Defined ⇒ Directory Listing Style, and enter the name of your user-named style (for example MY AS400-UNIX), or you can select any one of the 10 User Defined properties from the Directory Listing Style dropdown list (see "Creating User Defined Heuristic Directory Listing Styles" on page 108).**

3 **Your configuration changes will be applied to any Projects that are built and deployed with this Enterprise Designer.**

Heuristics Configuration File Format

This example includes two user-named styles (MY AS400-UNIX, and UDH NT 4.0).

```
#
# ------------------------------------------------------------------------
#   Section:    MY AS400-UNIX
# ------------------------------------------------------------------------
```

```
#

MY AS400-UNIX!Commands Supported By FTP
Server!value=APPE%CWD%DELE%LIST%MKD%NOOP%PASS%QUIT%RETR%RNFR%RNTO
%SITE%STOR%TYPE%USER!set=APPE%CWD%DELE%LIST%MKD%NOOP%PASS%QUIT%RETR
%RNFR%RNTO%SITE%STOR%TYPE%USER

MY AS400-UNIX!Header Lines To Skip!value=0!set=0

MY AS400-UNIX!Header Indication Regex Expression!value=!set=

MY AS400-UNIX!Trailer Lines To Skip!value=0!set=0

MY AS400-UNIX!Trailer Indication Regex Expression!value=!set=

MY AS400-UNIX!Directory Indication Regex Expression!value=!set=

MY AS400-UNIX!File Link Real Data Available!value=No!set=No%Yes

MY AS400-UNIX!File Link Indication Regex Expression!value=!set=

MY AS400-UNIX!File Link Symbol Regex Expression!value=!set=

MY AS400-UNIX!List Line Format!value=Fixed!set=Blank Delimited%Fixed

MY AS400-UNIX!Valid File Line Minimum Position!value=52!set=52

MY AS400-UNIX!File Name Is Last Entity!value=Yes!set=No%Yes

MY AS400-UNIX!File Name Position!value=52!set=52

MY AS400-UNIX!File Name Length!value=0!set=0

MY AS400-UNIX!File Extension Position!value=0!set=0

MY AS400-UNIX!File Extension Length!value=0!set=0

MY AS400-UNIX!File Size Verifiable!value=No!set=No%Yes

MY AS400-UNIX!File Size Position!value=0!set=0

MY AS400-UNIX!File Size Length!value=0!set=0

MY AS400-UNIX!Special Envelope For Absolute Pathname!value=!set="

MY AS400-UNIX!Listing Directory Yields Absolute Pathnames!value=No!set=No%Yes

MY AS400-UNIX!Absolute Pathname Delimiter Set!value=///!set=///
```

```
MY AS400-UNIX!Change Directory Before Listing!value=Yes!set=No%Yes

MY AS400-UNIX!Directory Name Requires Terminator!value=No!set=No%Yes

#

#

# ------------------------------------------------------------------------

#   Section:   UDH NT 4.0

# ------------------------------------------------------------------------

#

UDH NT 4.0!Commands Supported By FTP
 Server!value=APPE%CWD%DELE%LIST%MKD%NOOP%PASS%QUIT%RETR%RNFR%RNTO%SITE%
STOR%TYPE%USER!set=APPE%CWD%DELE%LIST%MKD%NOOP%PASS%QUIT%RETR%RNFR%RNTO%SITE%
STOR%TYPE%USER

UDH NT 4.0!Header Lines To Skip!value=0!set=0

UDH NT 4.0!Header Indication Regex Expression!value=!set=

UDH NT 4.0!Trailer Lines To Skip!value=0!set=0

UDH NT 4.0!Trailer Indication Regex Expression!value=!set=

UDH NT 4.0!Directory Indication Regex Expression!value=<DIR>!set=<DIR>

UDH NT 4.0!File Link Real Data Available!value=No!set=No%Yes

UDH NT 4.0!File Link Indication Regex Expression!value=\.lnk$!set=\.lnk$

UDH NT 4.0!File Link Symbol Regex Expression!value=!set=

UDH NT 4.0!List Line Format!value=Blank Delimited!set=Blank Delimited%Fixed

UDH NT 4.0!Valid File Line Minimum Position!value=4!set=4

UDH NT 4.0!File Name Is Last Entity!value=Yes!set=No%Yes

UDH NT 4.0!File Name Position!value=4!set=4

UDH NT 4.0!File Name Length!value=0!set=0
```

```
UDH NT 4.0!File Extension Position!value=0!set=0

UDH NT 4.0!File Extension Length!value=0!set=0

UDH NT 4.0!File Size Verifiable!value=Yes!set=No%Yes

UDH NT 4.0!File Size Position!value=3!set=3

UDH NT 4.0!File Size Length!value=0!set=0

UDH NT 4.0!Special Envelope For Absolute Pathname!value=!set=

UDH NT 4.0!Listing Directory Yields Absolute Pathnames!value=No!set=No%Yes

UDH NT 4.0!Absolute Pathname Delimiter Set!value=\\\\\\!set=\\\\\\

UDH NT 4.0!Change Directory Before Listing!value=No!set=No%Yes

UDH NT 4.0!Directory Name Requires Terminator!value=No!set=No%Yes
```

FTP Heuristics Configuration Parameters

This section describes the configuration parameters for the Batch FTP Heuristics located in the `FtpHeuristics.cfg` file. The Batch FTP Heuristics configuration file (`FtpHeuristics.cfg`) contains the full set of parameters for each of the platforms listed under "Platform Selection" on page 106.

The FTP Heuristics configuration parameters are as follows:

Commands Supported by FTP Server

Description

Specifies the commands that the FTP server on the given host supports.

Required Values

One or more FTP commands as selected from the list.

Header Lines To Skip

Description

Specifies the number of beginning lines from a **LIST** command to be considered as a potential header (subject to the **Header Indication Regex Expression** configuration parameter, discussed below) and skipped.

Required Values

A non-negative integer. Enter zero if there are no headers.

Additional Information

In the example below, the line "total 6" comprises a one-line header.

```
total 6
-rw-r-----   1 ed        usr           110 Apr 15 13:43 AAA
-rw-r--r--   1 ed        usr           110 Apr 15 13:33 aaa
```

Header Indication Regex Expression

Description

Specifies a regular expression used to help identify lines which comprise the header in the output of a LIST command. All the declared lines of the header (see **Header Lines To Skip**, above) must match the regular expression.

Required Values

A regular expression. The default varies based on the FTP server's operating system. If there is no reliable way of identifying the header lines in the **LIST** command's output, leave this parameter undefined.

Additional Information

The regular expression "^ *total" indicates that each line in the header starts with "total," possibly preceded by blanks, for example:

```
total 6
-rw-r-----   1 ed        usr           110 Apr 15 13:43 AAA
-rw-r--r--   1 ed        usr           110 Apr 15 13:33 aaa
```

If the regular expression is undefined, then the header is solely determined by the value of the configuration parameter **Header Lines To Skip**.

Trailer Lines To Skip

Definition

Specifies the number of ending lines from a **LIST** command that are to be considered as a potential Trailer (subject to the **Trailer Indication Regex Expression**) and skipped.

Required Values

A non-negative integer. Enter zero if there are no trailers.

Trailer Indication Regex Expression

Definition

Specifies the regular expression used to help identify lines which comprise the trailer in the output of a **LIST** command. All the declared lines of the trailer (see **Trailer Lines To Skip**) must match the regular expression.

Required Values

A regular expression. If there is no reliable way of identifying the trailer lines in the **LIST** output, then leave this parameter undefined.

Additional Information

If the regular expression is undefined, then the header is determined solely by the value of the **Trailer Lines To Skip** configuration parameter.

Directory Indication Regex Expression

Definition

Specifies a regular expression used to identify external directories in the output of a **LIST** command. Directories cannot be retrieved and must be filtered out of the file list.

Required Values

A regular expression. If there is no reliable way of identifying the directory in the **LIST** output, then leave this parameter undefined.

Additional Information

The regular expression "^ *d" specifies that a directory is indicated by a line starting with the lowercase "d,' possibly preceded by blanks, for example:

```
drwxr-xr-x   2 ed    usr     2048 Apr 17 17:43 public_html
```

File Link Real Data Available

Definition

Specifies whether a file may be a file link (a pointer to a file) on those operating systems whereon an FTP server will return the data for the real file as opposed to the content of the link itself.

Required Values

Yes or **No**.

File Link Indication Regex Expression

Definition

Specifies a regular expression that identifies external file links in the output of a **LIST** command. File links are pointers to the real file and usually have some visual symbol, such as- >, mixed in with the file name in the output of the **LIST** command. Only the link name is desired within the returned list.

Required Values

A regular expression. If there is no reliable way of identifying a file link within a **LIST** output, then leave this parameter undefined.

Additional Information

The regular expression "^ *l" specifies that a file link is indicated by a line starting with the lowercase "l," preceded possibly by blanks, for example:

```
lrwxr-xr-x   2 ed        usr   2048 Apr 17 17:43 p ->       public_html
```

File Link Symbol Regex Expression

Definition

Specifies a regular expression that parses the external file link name in the output of a **LIST** command. Only the link name is required for the file list to be returned.

Required Values

A regular expression. If there is no reliable way of identifying a file link within a **LIST** output, then leave this parameter undefined.

Additional Information

The regular expression "[ ] ->[ ]" defines that a file link symbol is represented by an arrow surrounded by spaces (" -> "). When parsed, only the file name to the right of the symbol is used.

In the following example, only the **public_html** would be used, not the "p" character:

```
lrwxrwxrwx   2 ed        usr  4 Apr 17 17:43 p -> public_html
```

List Line Format

Definition

Specifies whether fields in each line are blank delimited or fixed, that is, whether information always appears at certain columns.

Required Values

Blank Delimited or **Fixed**.

Additional Information

Even though some lines appear to be blank delimited, be wary of certain fields continuing their maximum value when juxtaposed with the next field without any separating blank. In such a case, we recommend you declare the line as "Fixed," for example:

```
-rw-r--r--   1 ed        usr           110 Apr 15 13:33 aaa
^^^^^^^^^^   ^ ^^        ^^^           ^^^ ^^^ ^^ ^^^^^ ^^^
     1       2  3         4             5   6  7    8    9
```

Valid File Line Minimum Position

Definition

Specifies the minimum number of positions (inclusive) a listing line must have in order to be considered as a possible valid file name line.

Required Values

For a **Fixed** list line format, enter a value equal to the number of columns, counting the first column at the far left as column 1. For a **Blank Delimited** list line format, enter a value equal to the number of fields, counting the first field on the far left as field 1.

For either case, if no minimum can be determined, set this value to zero (0).

Additional Information

For example, in the **Blank Delimited** line below, the minimum number of fields is 9:

```
-rw-r--r--   1 ed        usr          110 Apr 15 13:33 aaa
^^^^^^^^^^   ^ ^^        ^^^          ^^^ ^^^ ^^ ^^^^^ ^^^
     1       2  3         4            5   6  7    8    9
                                                File Name
```

Note – The URL FTP Proxy will fail on ascertaining file names that have leading blanks, trailing blanks, or both.

File Name Is Last Entity

Definition

Specifies whether the file name is the last entity on each line. This allows the file name to have imbedded blanks (however, leading or trailing blanks are not supported).

Required Values

Yes or **No**.

File Name Position

Definition

Specifies the starting position (inclusive) of a file name.

Required Values

For **Fixed** list line format, enter the column number, counting the first column on the far left as column 1. For **Blank Delimited** list line format, enter the field number, counting the first field on the extreme left as field 1.

Additional Information

For **Blank Delimited** List Line Format only, if the file name has imbedded blanks, then it can span over several fields, for example:

```
-rw-r--r--   1 ed        usr          110 Apr 15 13:33 aaa
^^^^^^^^^^   ^ ^^        ^^^          ^^^ ^^^ ^^ ^^^^^ ^^^
    1        2  3         4            5   6  7    8    9
                                                File Name
```

File Name Length

Definition

Represents the maximum width of a file name; valid only for **Fixed** list line format.

Required Values

Enter one of the following:

- **An Integer:** Positive lengths imply that the file name is right-justified within the maximum field width, and thus leading-blanks are discarded.
  - **Negative Lengths:** That is, compared to the absolute length, imply that the file name is left-justified and trailing-blanks are discarded.
  - **Zero (0) Value Length:** If the file name is at the end of a file listing line, this value implies that the file name field extends to the end of the line.

Note – For Blank Delimited list line format, this value is usually zero (0). However, if the File Name Length parameter is supplied even though a Blank Delimited list line format is specified, this implies that if the file name field exceeds the given length, then the rest of the List Line data occurs on the following line.

File Extension Position

Definition

Specifies the left-most position of the file extension for those operating systems that present the file name extension separated from the main file name.

Required Values

For **Fixed** list line format, enter the column number, counting the first column at the extreme left as column 1. For **Blank Delimited** list line format, enter the field number, counting the first field at the far left as field 1. If there is no file extension (as on UNIX systems) set the value to zero (0).

File Extension Length

Definition

Specifies the maximum width of the file extension; valid only for **Fixed** list line format.

Required Values

Enter one of the following:

- **An Integer**

- **Positive Lengths:** Imply that the file extension is right-justified within the maximum field width and therefore leading-blanks are discarded.
- **Negative Lengths:** Imply that the file extension is left-justified and trailing-blanks are discarded (the absolute length is used).
- **Value of Zero (0):** *Always* for the **Blank Delimited** list line format.

File Size Verifiable

Definition

Specifies whether the file size is verifiable, significant, and accurate within a directory listing.

Required Values

Yes or **No**. The **File Size Stability Check** configurable parameter must also be enabled.

Additional Information

Even if the file size field of a listing line is not significant (that is, it is there but only represents an approximate value), the value of this parameter must be **No**. However, the file size location must still be declared in the **File Size Position** parameter below to assist determining which line of listing represents a valid file name, for example:

```
-rw-r--r--   1 ed        usr           110 Apr 15 13:33 aaa
                                       ^^^
                                       File Size
```

Note – Use of this parameter does not guarantee that the file is actually stable. As this feature is intended only for backward compatibility with previous FTP implementations, we do not recommend that you rely on this functionality for critical data.

File Size Position

Definition

Specifies the left-most position in the listing line that represents the size of the file. Even though for some operating systems the value shown might not truly reflect the file size, this position is still important in ascertaining that the line contains a valid file name.

Required Values

A non-negative integer. For **Fixed** list line format, the position value is the column number (starting with one (1) on the far left). For **Blank Delimited**, this value represents the field number (starting with one (1) on the far left). If the **LIST** line does not have a size field, set this parameter to zero (0).

Example

```
 -rw-r--r--   1 ed        usr           110 Apr 15 13:33 aaa
 ^^^^^^^^^^   ^ ^^        ^^^           ^^^ ^^^ ^^ ^^^^^ ^^^
      1        2  3         4             5   6  7    8    9
                                      File
                                         Size
```

The following text represents valid number representations of file sizes:

```
1234 or 1,234,567 or -12345 or +12345 or ’  1234  ’ or 12/34       or 1,234/56
```

The following text represents invalid number representations of file sizes (the ^ indicates where the error occurs):

```
’12 34’ or 123,45,678 or 123-456-789 or --123 or 123-
   ^          ^                  ^        ^          ^
or 12345678901 or any number > 4294967295 or < -2147483647
   ^ (too large)
or 123.45 or 12AB34 or 0x45 or ,123,456 or 12//34
      ^        ^        ^      ^              ^
or /123 or 123/ or 12,3/45
   ^          ^        ^
```

File Size Length

Definition

Specifies the maximum width (number of columns) of the file size field, only valid for **Fixed** List Line Format.

Required Values

A non-negative integer. For **Blank Delimited** list line format, set this value to zero (0).

Special Envelope For Absolute Path Name

Definition

Specifies special enveloping characters required to surround an absolute path name (for example, single quotes are used in MVS). Only use a single quote at the start of the directory name.

Required Values

A pair of enveloping characters. Even if the leading and trailing character is identical, enter it twice.

If no enveloping characters are required for an operating system, leave this parameter undefined.

Note – On UNIX, this parameter is always undefined.

Listing Directory Yields Absolute Path Names

Definition

Specifies whether, when the **DIR** command is used on a directory name, the resulting file names are absolute.

Required Values

Yes or **No**.

Note – On UNIX, this character is always set to No.

Absolute Path Name Delimiter Set

Definition

Specifies any absolute path requiring certain delimiters to separate directory names (or their equivalent) from each other and from the file name.

Required Values

Enter the delimiters for the absolute path, starting from the left, for:

- Initial (left-most) directory delimiter
- Intermediate directory delimiters
- Initial (left-most) file name delimiter
- Optionally, the ending (right-most) file name delimiter

Wherever there is no specific delimiter, use "\0" (backslash zero) to act as a placeholder. Delimiters that are backslashes need to be escaped with another backslash (see Table 56).

TABLE 56 Delimiters and Path Naming by Platform

OS	Path Name Format	Delimiter Set				
		1	2	3	4	Enter
UNIX	`/dir1/dir2/file.ext`	/	/	/		///
Windows	`C:\dir1\dir2\file.ext`	\\	\\	\\		\\\\\\
VMS	`disk1:[dir1.dir2]file.ext;1`	[	.	]	;	[.];

TABLE 56 Delimiters and Path Naming by Platform *(Continued)*

OS	Path Name Format	Delimiter Set				
		1	2	3	4	Enter
MVS PDS	`dir1.dir2`(member)	\0	.	(	)	\0.()
MVS Sequential	`dir1.dir2.filename`	\0	.	.		\0..
MVS GDG	`dir1.dir2.file(version#)` (see Note)	\0	.	.		\0..
AS400	`dir1/file.ext`	\0	/	.		\0/.

Above, version # = 0 for current, +1 for new, -1 (-2, -3, etc.) for previous generations.

Change Directory Before Listing

Definition

Determines whether a change directory (`cd`) command needs to be done before issuing the `DIR` command to get a listing of files under the desired directory.

Required Values

Yes or **No**.

Note – The current Batch Adapter implementation does not rely on this parameter.

Directory Name Requires Terminator

Definition

Determines whether a directory name that is not followed immediately by a file name requires the ending directory delimiter as a terminator (for example, as on VMS).

Required Values

Yes or **No**.

FTP Configuration Requirements for AS400 UNIX (UFS)

When using FTP with an **AS400 UNIX (UFS)** system, the following FTP configuration settings are required:

- FTP - Use PASV: **No** (see Table 3)
- FTP Raw Commands - Pre Transfer Raw Commands: **site namefmt 1** (see Table 4)

Dynamic Configuration

The **BatchFTP**, **BatchFTPOverSSL**, **BatchSCP** and **BatchSFTP** OTDs support automatic connection during initialization. Each of these OTDs require a number of properties to be set with valid values when Connection Mode is set to Automatic. This includes, but is not limited to, the following:

Environment Properties:

- Host Name
- Server Port
- User Name
- Password
- Any additional properties that are required for a successful connection.

These parameters must be set to valid values prior to using the BatchFTP OTD to allow the adapter to initialize successfully. After the initialization is successful, the parameters can be reconfigured from within the Collaboration Rule.

Dynamic configuration allows you to change configuration settings (based on the data input or Collaboration Rule logic) on the fly. Changes are made to the Collaboration using the Collaboration Editor. Make any necessary changes to the configuration settings and perform the **put** or **get**. The Project disconnects, reconnects with the new configuration settings, and performs the transfer.

Dynamic Configuration Sample

The following sample code demonstrates how to dynamically configure the adapter and perform a simple file transfer.

1. From BatchLocalFile:

   Set the TargetDirectoryName

```
//@map:Copy "InDir" to TargetDirectoryName

BatchLocalFile_1.getConfiguration().setTargetDirectoryName( "InDir" );
```

2. From BatchFTP:

   Disconnect the adapter

   ```
   //@map:Client.disconnect

   BatchFTP_1.getClient().disconnect();
   ```

3. Set the TargetDirectoryName

   ```
   //@map:Copy "OutDir" to TargetDirectoryName

   BatchFTP_1.getConfiguration().setTargetDirectoryName( "OutDir" );
   ```

4. Set the HostName

   ```
   //@map:Copy "myftphostname" to HostName

   BatchFTP_1.getConfiguration().setHostName( "myftphostname" );
   ```

5. Connect the adapter

   ```
   //@map:Client.connect

   BatchFTP_1.getClient().connect();
   ```

6. Perform a simple file transfer:

   Get a local file

   ```
   //@map:

   BatchLocalFile_1.getClient().get();
   ```

7. Assign the Payload

   ```
   //@map:Copy Payload to Payload

   BatchFTP_1.getClient().setPayload(BatchLocalFile_1.getClient().getPay
   load() );
   ```

8. Put a file on the FTP server

   ```
   //@map:Client.put

   BatchFTP_1.getClient().put();
   ```

To view the Collaboration Editor's Java Source Editor, click the **Advance mode** or **Source Code mode** icon, available on the Collaboration Editor toolbar.

Dynamic Configurable Parameters for Secure FTP OTDs

The secure Batch FTP OTDs contain several dynamic configurable parameters, which include (but are not limited to) the following:

BatchFTPOverSSL:

- CM (Connectivity Map) Link configuration ⇒ FTP and SSL Settings ⇒ Remote Directory
- CM Link configuration ⇒ FTP and SSL Settings ⇒ Remote Directory Name Is Pattern
- CM Link configuration ⇒ FTP and SSL Settings ⇒ Remote File
- CM Link configuration ⇒ FTP and SSL Settings ⇒ Remote File Name Is Pattern
- CM Link configuration ⇒ FTP and SSL Settings ⇒ Local Directory
- CM Link configuration ⇒ FTP and SSL Settings ⇒ Local Directory Name Is Pattern
- CM Link configuration ⇒ FTP and SSL Settings ⇒ Local File
- CM Link configuration ⇒ FTP and SSL Settings ⇒ Local File Name Is Pattern
- CM Link configuration ⇒ FTP and SSL Settings ⇒ Transfer Mode
- CM Link configuration ⇒ FTP and SSL Settings ⇒ Append
- CM Link configuration ⇒ FTP and SSL Settings ⇒ Local File Overwrite

BatchSCP:

- CM Link configuration ⇒ SCP Settings ⇒ Remote Directory
- CM Link configuration ⇒ SCP Settings ⇒ Remote File
- CM Link configuration ⇒ SCP Settings ⇒ Local Directory
- CM Link configuration ⇒ SCP Settings ⇒ Local File
- CM Link configuration ⇒ SCP Settings ⇒ Transfer Mode
- CM Link configuration ⇒ SCP Settings ⇒ Copy Recursive

BatchSFTP:

- CM Link configuration ⇒ SFTP Settings ⇒ Remote Directory
- CM Link configuration ⇒ SFTP Settings ⇒ Remote Directory Name Is Pattern
- CM Link configuration ⇒ SFTP Settings ⇒ Remote File
- CM Link configuration ⇒ SFTP Settings ⇒ Remote File Name Is Pattern
- CM Link configuration ⇒ SFTP Settings ⇒ Local Directory
- CM Link configuration ⇒ SFTP Settings ⇒ Local Directory Name Is Pattern
- CM Link configuration ⇒ SFTP Settings ⇒ Local File
- CM Link configuration ⇒ SFTP Settings ⇒ Local File Name Is Pattern
- CM Link configuration ⇒ SFTP Settings ⇒ Transfer Mode

Configuration Parameters that Accept Integer Values

The configuration parameters listed below can be configured from the Collaboration Editor by entering the specified integer values in the method parameters. The classes, **com.stc.connect.ssl.FTPSSLConstants** and **com.stc.connect.ssh.SSHConstants**, do not allow "incremental completion," that is, you must enter the value using the fully qualified name to access the constant.

For example, to set the BatchFTPOverSSL **Secure Mode** to **Explicit SSL**, from the Collaboration Editor, do the following:

1. From the Collaboration Editor toolbar, click **Source Code Mode**. The Collaboration Editor's **Java Source Editor** opens.
2. From the Business Rules tree (Business Rules pane) select the rule that contains the parameter or method that you want to configure. Selecting the rule highlights the corresponding code in the Java Source Editor. Find the code you wish to modify.
3. From the Java Source Editor, enter the value for the setting you require. For example, to set the BatchFTPOverSSL **SecureType** method to **Explicit SSL**, type **com.stc.connector.ssl.FTPSSLConstants.FTP_SECURE_TYPE_SSL_EXPLICIT** as the parameter value (see example below):

```
public void receive( com.stc.connector.appconn.file.FileTextMessage input,
 com.stc.connector.batchadapter.appconn.ftps.FTPOverSSL BatchFTPOverSSL_1 )

        throws Throwable

    {

        if (!BatchFTPOverSSL_1.getClient().isConnected()) {

            logger.error( "Collab Start NOT CONNECTED ============ DO CONNECT" );

            BatchFTPOverSSL_1.getClient().connect();

            BatchFTPOverSSL_1.getConfiguration().setSecureType(
com.stc.connector.ssl.FTPSSLConstants.FTP_SECURE_TYPE_SSL_EXPLICIT );

        }

        BatchFTPOverSSL_1.getClient().get();

        if (BatchFTPOverSSL_1.getClient().isConnected()) {

            logger.error( "Collab End IS CONNECTED ============ DO DISCONNECT" );

            BatchFTPOverSSL_1.getClient().disconnect();
```

```
        }

    }
```

4. Once you have made your changes to the Collaboration, click the Commit Changes icon (from the Java Source Editor toolbar).

The OTD parameters listed below accept the following specified values:

- **BatchFTPOverSSL:**
  - CM Link configuration > FTP and SSL Settings > SecureType
    - None: **com.stc.connector.ssl.FTPSSLConstants.FTP_SECURE_TYPE_NONE**
    - Implicit SSL:**com.stc.connector.ssl.FTPSSLConstants.FTP_SECURE_TYPE_IMPLICIT**
    - Explicit SSL:**com.stc.connector.ssl.FTPSSLConstants.FTP_SECURE_TYPE_SSL_EXPLICIT**
  - Environment Link configuration > FTP and SSL Settings > KeyStoreType
    - JKS:**com.stc.connector.ssl.FTPSSLConstants.KEY_STORE_TYPE_JKS** (only one valid choice)
    - Other: (this is a place holder - reserved for future enhancement)
  - CM Link configuration > FTP and SSL Settings > TransferMode
    - ASCII:**com.stc.connector.ssl.FTPSSLConstants.FTP_TRANS_MODE_ASCII**
    - BINARY:**com.stc.connector.ssl.FTPSSLConstants.FTP_TRANS_MODE_BINARY**
- **BatchSFTP:**
  - CM Link configuration > SFTP Settings > TransferMode
    - ASCII: **com.stc.connector.ssh.SSHConstants.TRANS_MODE_ASCII**
    - BINARY: **com.stc.connector.ssh.SSHConstants.TRANS_MODE_BINARY**

Understanding Batch Adapter OTDs

This chapter provides an overview of the Object Type Definitions (OTDs) available with the Batch Adapter. The OTDs include fields that contain methods, properties, and their application.

What's in This Chapter

Overview of the Batch OTDs

An OTD contains a set of rules that define an object. The object encodes data as it travels through Java CAPS. OTDs are used as the basis for creating Collaboration Definitions for a Project.

Each OTD acts as a template with a unique set of features of the Adapter. The Batch Adapter OTD template is not customizable and cannot be edited.

The four parts of an OTD are:

- **Element: An element** is the highest level in the OTD tree. The element is the basic container that holds the other parts of the OTD. The element can contain fields and methods.
- **Field**: Fields are used to represent data. A field can contain data in any of the following formats: string, boolean, int, double, or float.
- **Method**: Method nodes represent actual Java methods.
- **Parameter**: Parameter nodes represent the Java methods' parameters.

A high-level view of the OTD folder structure shows methods and attributes you can use in creating Business Rules that invoke FTP, secure FTP, batch record, or local file data exchange.

Types of Batch Adapter OTDs

This table shows the specialized OTDs available with the adapter.

TABLE 57 Batch Adapter OTDs

OTD Name	Description
BatchFTP	Provides FTP access to remote systems.
BatchFTPOverSSL	Provides secure data transfer using FTP over SSL.
BatchSCP	Provides secure data transfer using Secure Copy Protocol with Secure Shell (SSH) as an underlying protocol.
BatchSFTP	Provides secure data transfer using SSH File Transfer Protocol or SFTP protocol.
BatchLocalFile	Provides easy access to local file systems.

TABLE 57 Batch Adapter OTDs *(Continued)*

OTD Name	Description
BatchRecord	Allows the adapter to parse or create (or both) payloads of records in specified formats.
BatchInbound	Polls for input file, renames the file to a GUID, and triggers the Business Process or Collaboration.

This chapter describes each of these OTDs and how to use them with the adapter.

OTD Functions

OTDs provide the following functions:

- OTD are used in Collaborations to provide platform, system, and program specific functionality that allow you to create Business Rules.
- OTDs contain system specific parameters that can be configured using the Properties Editor.
- OTDs provide access to the information required to interface with specific external application.

All OTDs must be configured and administered using the Enterprise Designer. Any client components relevant to these OTDs have their own requirements. See the client system's documentation for details.

For the **BatchFTP**, **BatchLocalFile**, and **BatchRecord** OTDs, only those nodes for which there is a corresponding section in the Environment or Connectivity Map properties (From the Properties Editor) are implemented on the OTD. The remaining nodes are not implemented and are reserved for potential future use.

For the **BatchFTP**, **BatchLocalFile**, and **BatchRecord** OTDs, only those configuration parameters which appear in the Environment or Connectivity Map properties (From the Properties Editor) are supported for use on the OTD. The remaining configuration parameters are not implemented, and are reserved for potential future use. Even though an implemented configuration parameter might be accessed and used from a non-implemented node, such use is not recommended.

BatchFTP OTD

The Batch Adapter includes four OTDs that allow you to perform FTP data-transfer functions. The **BatchFTP** OTD enables the Java CAPS system to exchange data with other network hosts for the purpose of receiving and delivering objects stored in files. In addition, the **BatchFTPOverSSL**, **BatchSCP**, and **BatchSFTP** OTDs enable secure data transfer between the local host and a remote host using FTP over SSL and SSH.

BatchFTP OTD Structure

The BatchFTP OTD contains three top-level nodes, **Client, Configuration**, **Provider**, **State**, and **StateManager**. Expand these nodes to reveal additional sub-nodes.

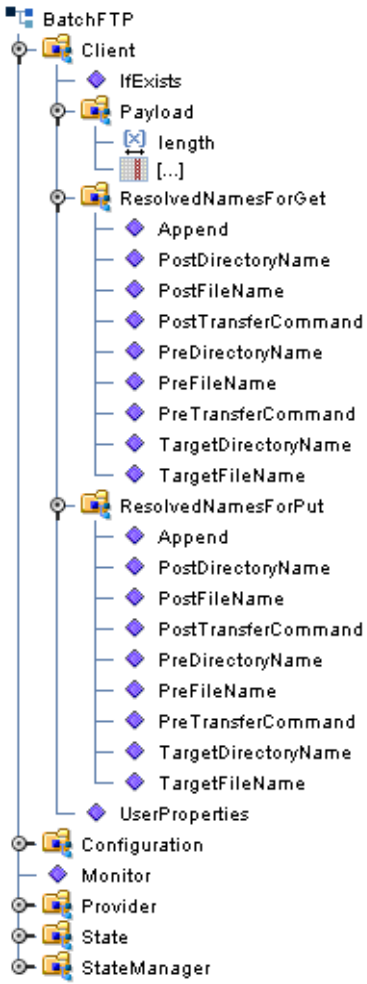


FIGURE 3 BatchFTP OTD Structure

Configuration Node

Each field sub-node in the **Configuration** node of the OTD corresponds to one of the adapter's FTP configuration parameters.

Client and Provider Nodes

This OTD includes two additional top-level nodes, the **Client** and **Provider**. These nodes implement their respective functionality interfaces in the adapter.

- The *client interface* represents how the functionality of the provider is actually used.

- The *provider interface* represents all of the general FTP operations that can be performed by the OTD.

BatchFTP OTD Node Functions

The following list provides an explanation of various nodes in the BatchFTP OTD, including primary functions:

- **BatchFTP**: Represents the OTD's root node.
- **Configuration**: Each field sub-node within this node corresponds to an adapter configuration parameter and contains settings information.
  - **InputStreamAdapter** and **OutputStreamAdapter**: Allow you to use and control the OTD's data-streaming features; see "Streaming Data Between Components" on page 166 for details.

    Note – This OTD has configuration parameters that can be regular expressions. See "Using Regular Expressions" on page 159 for details.

- **Client**: This node contains the following sub-nodes, which implement the adapter's client interface in the OTD (FtpFileClient):
  - **Payload**: An in-memory buffer that contains the payload or message data you want to transfer by FTP, in the form of a byte array.
  - **UserProperties**: Only used if you have provided a user-defined implementation of the **FtpFileClient** interface; in such cases, the node represents the properties specified in the configuration.

    Note – You can transfer data using the Payload node or by using data streaming (InputStreamAdapter and OutputStreamAdapter nodes), but you cannot use both methods in the same OTD.

  - **ResolvedNamesForGet** and **ResolvedNamesForPut**: Allow you to get the real file or directory name used during a transfer and perform an operation with it. For example, you could do a file transfer, with `get()` or `put()`, using the real name. You are able to retrieve the real file or directory name, even if these names have been expressed using regular expressions or special characters.

    These nodes contain sub-nodes that allow you to resolve file and directory names for target destinations, as well as names for pre- and post-transfer commands. See "Pre/Post File Transfer Commands" on page 148, "Resolving Names" on page 163 for more information on these nodes. Also see "Using Regular Expressions" on page 159 for more information on regular expressions.

- `get()`, `put()`, `reset()`, `connect()`, `disconnect()`, and `isConnected()`: See "Essential BatchFTP OTD Methods" on page 135.

- **Provider**: The sub-nodes contained in this node are methods that implement the adapter's provider interface in this OTD (**FtpFileProvider**). These methods allow you to do the general FTP operations that can be performed using the OTD.

Using the BatchFTP OTD

The BatchFTP OTD nodes allow you to configure specific adapter configuration parameters for the Collaboration controlling the FTP process. Once you have set the configuration parameters, you do not have to define the same parameters in each corresponding adapter component that uses this Collaboration.

Handling Type Conversions

The **Payload** node in the BatchFTP OTD is predefined as a byte array (**byte**[]). This definition allows the adapter to handle both binary and character data.

For example, you could use another OTD (such as an OTD from another adapter or a user-defined OTD) where the "data" node has been defined as a String (**java.lang.String**). If you were to map that String to the BatchFTP OTD's **Payload** node, the Java CAPS Collaboration Editor can do an automatic type conversion and create code similar to that shown in the following example.

Note – You must use care with this feature. While it works in many situations, there can be occasions when the default encoding causes errors in the translation.

Code Conversion and Generation

For example, in a string-to-byte array conversion (or vice versa), the generated Java code could be:

```
getoutput().setPayload(STCTypeConverter.toByteArray
   (getinput().getBlob()));
```

or

```
getinput().setBlob(STCTypeConverter.toString
   (getoutput().getPayload()));
```

If you define the blob data as a byte array, no type conversion is necessary. When there is a conversion, the Collaboration Editor uses the Java Virtual Machine (JVM) default encoding to do the conversion to code, as shown in the previous examples.

Type Conversion Troubleshooting

As explained previously, the default encoding and translation works for many situations. There are cases, however (for example, binary data such as a `.zip` file), when the encoding could cause errors in the translation. Depending on the data character set and JVM default encoding, you *must* choose the appropriate encoding. In most cases, using the encoding string "ISO-8859-1" is the best choice.

To use this encoding, you can modify the code manually by adding the encoding String. Taking the previous examples, the resulting code using "ISO-8859-1" is:

```
getoutput().setPayload(STCTypeConverter.toByteArray
   (getinput().getBlob(), "ISO-8859-1"));
```

or

```
getinput().setBlob(STCTypeConverter.toString
   (getoutput().getPayload(), "ISO-8859-1"));
```

Using this String solves this type conversion problem. For more information, see the appropriate JVM encoding reference manuals.

Essential BatchFTP OTD Methods

In addition to the field elements, the BatchFTP OTD's **Client** node contains methods that extend the client interface functionality of the adapter. These methods are essential to the proper use of the OTD and require some additional explanation. They are:

- `get()`: Retrieves a file from the remote FTP server then stores its contents as a data payload. The method retrieves the first matching file based on the **Target Directory Name** and **Target File Name** parameters and stores the contents as a data payload (a byte array). It then performs any **Post Transfer Command**.

  Note – After this method call, you can get the payload's contents via the method `getPayload()`.

  If no qualified file is available for retrieving, you get the exception containing `java.io.FTPFileException` as a nested exception.

- **put()**: Places the payload data on the FTP server, that is, it performs an append or put action from the **Payload** node to the remote FTP server and performs any **Post Transfer Command**.

  If no qualified file is available for sending, you get the exception containing `java.io.FTPFileException` as a nested exception.

Note – When you are using the adapter's data-streaming feature, the `get()` and `put()` methods operate differently. See "Streaming Data Between Components" on page 166 for details on this operation.

- `reset()`: Allows you to return the **Client** node to its state immediately after the previous initialization.

Note – The reset() method is available in the Batch FTP OTDs and BatchLocalFile OTD. The reset method must be called when the OTD has to be reused for another transfer during the same execution of `executeBusinessRules()` (for example, when you are using the Dynamic Configuration feature). The `reset()` method resets the content of the Client node without resetting the whole OTD. If you attempt another transfer without calling reset() first, the system throws an exception and makes an entry in the adapter's error log file.

- `restoreConfigValues()`: Allows you to restore the configuration parameter defaults to the related adapter configuration.
- `connect()`, `disconnect()`, and `isConnected()`: Perform connection-related operations with respect to the FTP server.

Sequence Numbering

The sequence numbering feature allows you to set up the FTP target directory or file name to contain a sequence number. You can set the starting and maximum sequence numbers using the adapter configuration parameters for the OTD.

This parameter is used for the name pattern `%#`.

This feature is also available with the BatchLocalFile OTD. For more information on these configuration parameters, see "Sequence Numbering (BatchFTP Connectivity Map)" on page 33.

Additional FTP File Transfer Commands

The BatchFTP OTD also allows you to enter commands to be executed directly before and after the file transfer operation. See "Pre/Post File Transfer Commands" on page 148 for details.

BatchFTPOverSSL OTD

The Batch Secure FTP over SSL OTD (BatchFTPOverSSL) provides secure data transfer using Secure Sockets Layer (SSL) protocol.

For information about configuring external FTP servers, SSL servers, and so forth, refer to the application's documentation.

BatchFTPOverSSL OTD Structure

The BatchFTPOverSSL OTD contains two top-level nodes, **Client** and **Configuration**. Expand these nodes to reveal additional sub-nodes.

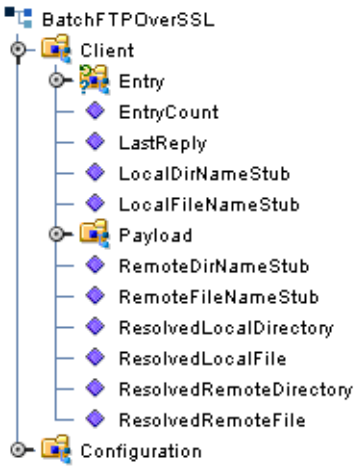


FIGURE 4 BatchFTPOverSSL OTD Structure

BatchFTPOverSSL OTD Node Functions

The following list provides an explanation of various nodes in the BatchFTPOverSSL OTD, including primary functions:

BatchFTPOverSSL: Represents the OTD's root node.

Configuration Node

The BatchFTPOverSSL sub-nodes under the **Configuration** node correspond to BatchFTPOverSSL Adapter's Connectivity Map and Environment configuration parameters.

BatchFTPOverSSL Client Node

The Client node contains sub-nodes that implement the adapter's client interface in the OTDs. The BatchFTPOverSSL Client node includes the following methods:

- `append()`: transfers data to the remote in append mode. The source of the data is determined by the configuration parameters **Local Directory** and **Local File**. If both are empty, then the data is from pay load.
- `connect()`: connects to the FTP server and does authentication as configured.
- `deleteDir(String dir)`: deletes the remote directory as specified by the argument.
- `deleteFile(String path)`: deletes the remote file as specified by the argument.
- `disconnect()`: disconnects from the FTP server.
- `doRawCommands(String commands)`: specifies the raw commands.
- `download()`: downloads data from the remote (specified in configuration parameters **Remote Directory** and **Remote File**) to the local (specified in configuration parameters **Local Directory** and **Local File**).
- `get()`: copies the file or directory specified by the configuration parameters **Remote Directory** and **Remote File**, to the local, as specified by the configuration parameters **Local Directory** and **Local File**. If the configuration parameter, **Is Copy Recursive**, is set to **Yes**, the copy will be recursive.
- `getEntry()`: gets the index entry in the current entry list.
- `getEntryCount()`: returns the count of directory entries, as the result of invoking **listDir** or **listDirLong**.
- `getLastReply()`: gets the FTP response code as a String.
- `getPayload()`: returns the payload.
- `hasEntry()`: returns **false** if there is more to the entry in the directory listing result list (when the result list is exhausted), and calls **resetEntries** to re-iterate the result list again.
- `getResolvedLocalDirectory()`: returns the resolved local directory name.
- `getResolvedLocalFile()`: returns the resolved local file name.
- `getResolvedRemoteDirectory()`: returns the resolved remote directory name.
- `getResolvedRemoteFile()`: returns the resolved remote file name.
- `hasEntry()`: returns whether the current entry list has entries.
- `isConnected()`: determines if the Java Integration Suite is connected to the FTP server.
- `listDir()`: returns the entry under the remote directory (specified in configuration parameters **Remote Directory** and **Remote File**). The entry only contains the name. After this method is invoked, use methods such as **hasEntry**, **nextEntry**, **getEntry**, **getEntryCount**, and so forth. to iterate the entry information.

- `listDirLong()`: returns the entries under the remote directory (specified in configuration parameters **Remote Directory** and `Remote File`). The entry contains details such as **name**, **size**, **time stamp**, **is directory or not**, and so forth. After this method is invoked, use methods such as **hasEntry**, **nextEntry**, **getEntry**, **getEntryCount**, and so forth, to iterate the entry information.
- `mkdir(String dir)`: creates a directory on the remote. The name of the directory is specified in the configuration parameters.
- `nextEntry()`: returns the next entry in the result list.
- `put()`: same as **get**, but the data transfer is from local to remote.
- `renameFile(String newName)`: renames the file specified by the configuration parameters **Remote Directory** and **Remote File** to a new name (arg).
- `reset()`: resets the internal life cycle methods, such as discard payload buffer.
- `resetEntries()`: resets the result list iterator so that it can be iterated again.
- `resolveLocalAsDestination()`: resolves the local directory name and local file name if they are patterns (used to generate real directory and file name for data transfer destination), upon the success of the resolution.
- `resolveLocalAsSource()`: resolves the local directory and file if they are regular expressions (filters for data transfer source); upon the success of the resolution.
- `resolveRemoteAsDestination()`: resolves the remote directory name and remote file name if they are patterns (used to generate real directory and file name for data transfer destination), upon the success of the resolution.
- `resolveRemoteAsSource()`: resolves the remote directory and file if they are regular expressions (filters for data transfer source); upon the success of the resolution.
- `setpayload(byte[] newPayload)`: sets the payload as specified by the argument.
- `setResolvedLocalDirectory(String s)`: sets the resolved local directory name.
- `setResolvedLocalFile(String s)`: sets the resolved local file name.
- `setResolvedRemoteDirectory(String s)`: sets the resolved remote directory name.
- `setResolvedRemoteFile(String s)`: sets the resolved remote file name.
- `upload()`: uploads data from the local (specified in configuration parameters **Local Directory** and **Local File**) to the remote (specified in configuration parameters **Remote Directory** and **Remote File**).

Note – See the Batch Adapter Javadoc for a list of all exposed FTPOverSSLClient methods.

BatchSFTP OTD

The Batch Secure SFTP OTD (BatchSFTP) provides secure data transfer using SSH File Transfer Protocol or SFTP. SFTP provides a range of operations on remote files, such as resuming interrupted transfers, directory listings, and remote file removal.

SFTP is one means of securely transferring computer files between a local and a remote host or between two remote hosts, using the Secure Shell (SSH) protocol. The BatchSFTP OTD uses SFTP to copy a file to or from a remote host.

BatchSFTP OTD Structure

The BatchSFTP OTD contains two top-level nodes, **Client** and **Configuration**. Expand these nodes to reveal additional sub-nodes.

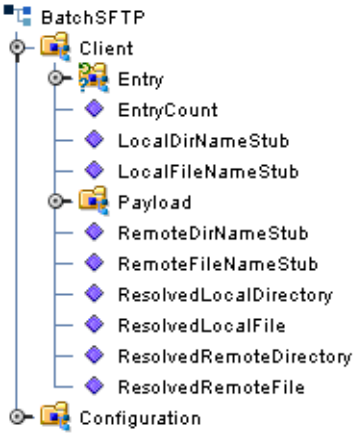


FIGURE 5 BatchSFTP OTD Structure

BatchSFTP OTD Node Functions

The following list provides an explanation of various nodes in the BatchSFTP OTD, including primary functions:

BatchSFTP: Represents the OTD's root node.

Configuration Node

The BatchSFTP sub-nodes under the **Configuration** node correspond to the BatchSFTP Adapter's Connectivity Map and Environment configuration parameters.

BatchSFTP Client Node

The BatchSFTP OTD's Client node includes the following methods:

- `cd(String dir)`: changes the remote current directory to the specified path.
- `connect()`: connects to the remote SSH server and does authentication as configured.
- `delete()`: deletes remote files specified by configuration parameters *RemoteDirectory* and *RemoteFile*.
- `delete(String file)`: deletes remote files specified by file.
- `disconnect()`: disconnects the client from the remote SSH server.
- `get()`: copies data from a remote SSH server (specified by configuration parameters RemoteDirectory and RemoteFile) to the local machine. Depending on the current status of the configuration, the remote data can be stored into the payload (in memory buffer) or a local file specified by configuration parameters LocalDirectory and LocalFile.

  Note – Note that the remote could be a folder. In this case, if the configuration parameter Recursive is "Yes", the folder hierarchy is copied to the local destination.

- `getEntry(int index)`: gets the index entry in the current entry list.
- `getEntryCount()`: returns the number of entries in the current entry list.
- `getPayload()`: returns the payload.

  Note – If you call `getPayload()` when using the BatchSFTP OTD, and the Local Directory and Local Filename are set, `getPayload()` returns null, even if a file has been retrieved.

- `getResolvedLocalDirectory()`: returns the resolved local directory name.
- `getResolvedLocalFile()`: returns the resolved local file name.
- `getResolvedRemoteDirectory()`: returns the resolved remote directory name.
- `getResolvedRemoteFile()`: returns the resolved remote file name.
- `hasEntry()`: returns whether the current entry list has entries.
- `isConnected()`: determines if the Java Integration Suite is connected to the SSH server.
- `lcd(String dir)`: changes the local current directory.
- `listDir()`: lists all the entries under remote current directory.
- `lpwd()`: returns the local current path.
- `mkdir()`: creates a directory on the remote. The name of the directory is specified in the properties.

- `mkdir(String dir)`: creates a directory on the remote using the name specified in the configuration parameters, **Remote Directory** and **Remote File**.
- `nextEntry()`: returns the next entry in the current entry list.
- `put()`: copies local data (specified by configuration parameters LocalDirectory and LocalFile) to the remote SSH server (specified by configuration parameters RemoteDirectory and RemoteFile). Depending on the current status of the configuration, the local data can be from either a payload or local file.

Note – Note that the local could be a folder. In this case, if the configuration parameter Recursive is "Yes", the folder hierarchy is copied to the remote destination.

- `pwd()`: returns the remote current path.
- `rename(String newPath)`: renames the file or directory specified by the old name (first argument), to new name (second argument).
- `rename(String oldPath, String newPath)`: renames the file or directory specified by configuration parameters, **Remote Directory** and **Remote File**, to new name (argument).
- `reset()`: resets the internal life cycle methods, such as discard payload buffer.
- `resetEntries()`: resets the current entry list so that next call to `nextEntry()` will return the first entry in the list.
- `resolveLocalAsDestination()`: resolves the local directory name and local file name if they are patterns (used to generate real directory and file name for data transfer destination), upon the success of the resolution.
- `resolveLocalAsSource()`: resolves the local directory and file if they are regular expressions (filters for data transfer source); upon the success of the resolution.
- `resolveRemoteAsDestination()`: resolves the remote directory name and remote file name if they are patterns (used to generate real directory and file name for data transfer destination), upon the success of the resolution.
- `resolveRemoteAsSource()`: resolves the remote directory and file if they are regular expressions (filters for data transfer source); upon the success of the resolution.
- `setpayload(byte[] newPayload)`: sets the payload.
- `setResolvedLocalDirectory(String s)`: sets the current resolved local directory name.
- `setResolvedLocalFile(String s)`: sets the current local file name to s, should not be invoked directly from user Collaboration.
- `setResolvedRemoteDirectory(String s)`: sets the current resolved remote directory name.
- `setResolvedRemoteFile(String s)`: sets the current resolved remote file name.

Note – See the Batch Adapter Javadoc for a list of all exposed SFTPClient methods.

BatchSCP OTD

The Batch Secure SCP OTD (BatchSCP) provides secure data transfer using Secure Copy Protocol with Secure Shell (SSH) as an underlying protocol. SCP is similar to RCP (remote copy), but the file is copied over secure shell (SSH) rather than RSH (remote shell). When files are copied using SCP the data is encrypted during transfer.

For information about configuring external FTP servers, SSH servers, and so forth, refer to the application's documentation.

BatchSCP OTD Structure

The BatchSCP OTD contains two top-level nodes, **Client** and **Configuration**. Expand these nodes to reveal additional sub-nodes.

FIGURE 6 BatchSCP OTD Structure

BatchSCP OTD Node Functions

The following list provides an explanation of various nodes in the BatchSCP OTD, including primary functions:

BatchSCP: Represents the OTD's root node.

Configuration Node

The BatchSCP sub-nodes under the **Configuration** node correspond to BatchSCP Adapter's Connectivity Map and Environment configuration parameters.

BatchSCP Client Node

The BatchSCP OTD's Client node includes the following methods:

- `connect()`: connects to the SSH server and does authentication as configured.
- `disconnect()`: disconnects from the SSH server.
- `get()`: copies the file or directory specified by the configuration parameters, **Remote Directory** and **Remote File**, to the local, as specified by configuration parameters **Local Directory** and **Local File**. If the configuration parameter, **Is Copy Recursive**, is set to **Yes**, the copy will be recursive.
- `getPayload()`: returns the payload buffer as a byte array.
- `getRecursive()`: copies data from remote to local with configuration parameter Recursive set to "Yes".
- `isConnected()`: checks whether the client is connected to the SSH server.
- `put()`: Copies local data (specified by configuration parameters *LocalDirectory* and *LocalFile*) to the remote SSH server (specified by configuration parameters RemoteDirectory and RemoteFile).
- `putRecursive()`: copies data from local to remote with the configuration parameter Recursive set to "Yes".
- `setpayload(byte[] newPayload)`: sets the payload.

Note – See the Batch Adapter Javadoc for a list of all exposed SCPClient methods.

BatchLocalFile OTD

The BatchLocalFile OTD provides access to files on your local system. While file access is not always necessary in Java CAPS, it makes sense for the Batch Adapter to have this feature because file processing is one of its core functions.

Additional BatchLocalFile features include regular expressions for accessing files and a sequence-numbering scheme for creating files.

BatchLocalFile OTD Structure

The BatchLocalFile OTD contains four top-level nodes, **Client**, **Configuration**, **PersistentState**, and **State Manager**. Expand these nodes to reveal additional sub-nodes.

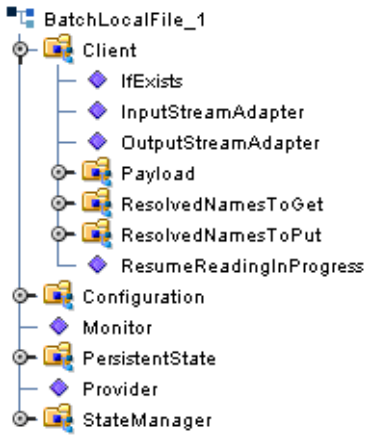


FIGURE 7 BatchLocalFile OTD Structure

Configuration Node

As it iw with each of the Batch OTDs, each field sub-node under the **Configuration** node in the BatchLocalFile OTD corresponds to one of the adapter's configuration parameters for that OTD. See "BatchLocalFile Connectivity Map Properties" on page 90 for details.

Client Node

This OTD includes an additional top-level node, the **Client**. This node implements its respective functionality interface in the adapter.

The *client interface* represents how the functionality of the OTD is actually used. This functionality includes the basic operations and features of the OTD. The client interface provides the OTD's the file services those who want to use them.

BatchLocalFile OTD Node Functions

The following list explains the nodes in the BatchLocalFile OTD, including primary functions:

- **Configuration**: The field sub-nodes within this node corresponds to an adapter configuration parameter and contains the corresponding settings information. See "BatchLocalFile Connectivity Map Properties" on page 90 for details on these parameters and settings.

Note – This OTD has configuration parameters that can be regular expressions. See "Using Regular Expressions" on page 159 for details.

- **Client**: The following sub-nodes, contained in this node, implement the adapter's client interface in the OTD (**LocalFileClient**):
  - **ResolvedNamesToGet** and **ResolvedNamesToPut**: Allow you to get the real file or directory name used during a transfer and perform an operation with it. For example, you could do a local file transfer, with `get()` or `put()`, using the real name. You are able to retrieve the real file or directory name, even if these names have been expressed using regular expressions or special characters.These nodes contain sub-nodes allowing you to resolve file and directory names for target destinations, as well as names for pre- and post-transfer commands (see "Pre/Post File Transfer Commands" on page 148 for details).

    See "Using Regular Expressions" on page 159 and "Using Name Patterns" on page 162 for more information on these features.
  - **InputStreamAdapter** and **OutputStreamAdapter**: Allow you to use and control the OTD's data-streaming features; see "Streaming Data Between Components" on page 166 for details.
  - **Payload:** An in-memory buffer that contains the payload or message data you want to transfer by local file, in the form of a byte array.
  - `get()`, `put()`, and `reset()`: See "Essential BatchLocalFile OTD Methods" on page 150.
  - **ResumeReadingInProgress**: This node allows you to resume a data-streaming file transfer operation that was interrupted for whatever reason. These transfers occur piece by piece and usually involve large files. This feature allows you to resume at the same point where the transfer left off when it stopped.

    Note – You can transfer data using the Payload node or by using data streaming (InputStreamAdapter and OutputStreamAdapter nodes), but you cannot use both methods in the same OTD.

Using the BatchLocalFile OTD

This section explains how to use the BatchLocalFile OTD's features.

There is no required particular order for the calls that can be made on the BatchLocalFile OTD. The only required call is `reset()` after a transfer, if it is used for more than one transfer per Collaboration Rule execution. An example of this would be a dynamic batch order with multiple files to be transferred.

If you are using a Java Collaboration to generate multiple files with sequence numbers, using the BatchLocalFile interface, you must call the `reset()` method to indicate the end of one file and the start of the next one.

BatchLocalFile Specific Features

Using the BatchLocalFile OTD to read records from a local file has the following advantages:

- **Data Streaming**: Allows your application to stream data directly to and from a local file system when used together with the BatchFTP OTD or the record-processing OTD. This feature minimizes the required RAM when large files are read, because the entire file is never loaded in memory.
- **Resume Reading**: Allows your application to read large files in a number of subsequent Business Rule executions, when you are using data streaming. This operation is achieved by persisting information about the current successful file read operation and resuming the next read operation from that last stored position.

For more information on the Resume Reading feature, see "Resume Reading Feature" on page 150.

Pre/Post File Transfer Commands

The adapter has features that allow you to execute desired actions directly before or after the actual file transfer. You can enter these settings at the adapter configuration parameters or in the Configuration node of the desired OTD.

These features are available with both the BatchLocalFile OTD and the BatchFTP OTD.

Pre Commands

For an inbound transfer, the file can be made unavailable to other clients polling the target system with the same directory and file pattern or name (Rename). For an outbound transfer, you can perform an automatic backup of the existing file (Copy).

Your pre-transfer options are:

- **Rename**: Rename the target file for protection or recovery; you must provide a desired directory and file name. The directory is created if it does not already exist.
- **Copy**: Copy the target file for backup or recovery; you must enter a desired directory and file name.
- **None**: Do nothing.

Caution – Each FTP server can behave differently when you are using Rename and a destination file already exists. For example, for some UNIX FTP servers, the destination file is overwritten without question. That is, no error or warning message is given. On other FTP servers, a Windows XP server for example, the system generates an error that results in exceptions being thrown in the called OTD method.Be sure you are familiar with the native behavior of the corresponding FTP server. If you are in doubt, try the action at the command prompt. If the action displays an error message, it is likely to result in the throwing of an exception in the Collaboration.

To gain proper protection, backup, or recovery, you must choose the appropriate setting that serves your purpose. For example, to recover from failures on an outbound appending transfer, use the **Copy** setting. When specifying file and directory names you can use regular expressions, special characters, or both.

Post Commands

For an inbound transfer, you can mark the transferred file as "consumed" by making an automatic backup (**Rename**) or by destroying it permanently (**Delete**). For an outbound transfer, you can make the transferred file available to other clients by renaming it. When specifying file and directory names you can use regular expressions, special characters, or both.

Your post-transfer options are:

- **Rename**: Rename the transferred file; you must provide a desired directory and file name.
- **Delete**: Delete the transferred file (inbound transfers only).
- **None**: Do nothing.

Note – For an outbound transfer (publishing), the directory is created if it does not already exist.

For more information on Pre and Post Transfer commands see the following:

BatchFTP OTD

- "Pre Transfer (BatchFTP Connectivity Map)" on page 27
- "Post Transfer (BatchFTP Connectivity Map)" on page 34

BatchLocalFile OTD

- "Pre Transfer (BatchLocalFile Connectivity Map)" on page 90
- "Post Transfer (BatchLocalFile Connectivity Map)" on page 93

Essential BatchLocalFile OTD Methods

In addition to the field elements, the BatchLocalFile OTD's **Client** node contains methods that extend the client interface functionality of the adapter. These methods are essential to the proper use of the OTD. These methods include:

- `get()`: Retrieves a local file then stores its contents as a data payload. The method retrieves the first matching file based on the **Target Directory Name** and **Target File Name** parameters and stores the contents as a data payload (a byte array). It then performs any **Post Transfer Command**.

  After this method call, you can get the payload's contents via the method `getPayload()`.
- `put()`: Stores the data payload (as a byte array) to a file. It then performs any **Post Transfer Command**.

  Before using this method call, you must set the file contents using the method `setPayload()`.

  The method throws an exception (**LocalFileException**) if an error occurs.
- `reset()`: Allows you to return the **Client** node to its state immediately after the previous initialization.

Note – The `reset()` method is available in both BatchFTP and BatchLocalFile OTDs. It must be called when the OTD has to be reused for another transfer during the same execution of executeBusinessRules() (for example, when you are using the Dynamic Configuration feature). The `reset()` method resets the content of the Client node without resetting the whole OTD.

Resume Reading Feature

The purpose of this feature is to allow an application to read large files in parts instead of processing the whole file at once. Resume Reading allows your system to read files in a number of subsequent Business Rule executions, when you are using data streaming.

General Operation

The Resume Reading feature's operation is achieved by keeping persistent information about the current successful file read operation, breaking, then resuming the next read operation from that last stored break position. As a result, the current file is read in parts, and the beginning and end of each part is determined by a predefined *break condition*.

You determine the break condition through the definition of your Business Rules. Since the Resume Reading feature operates based on reading one part of a file at a time per Business Rule, these rules must determine the break. Each Business Rule executes reading a part of the file, breaks, then passes to the next rule, which reads the next part up to the break, and so on, until the entire file is read.

A break condition can be any type of stopping point you determine in your Collaboration Rules. For example, this condition could be a fixed number of records, a delimiter, or reaching a specific character string.

The **Client** node in the OTD has a read-only property (**ResumeReadingInProgress** node) indicating whether there is a resume-reading operation in progress. This node is for informational purposes only. Also, the Resume Reading feature is available in the data-streaming mode only.

The feature has no special operational requirements besides setting the adapter configuration option. The adapter configuration has an option to enable or disable this feature. This option is also accessible at run time.

Note – If this feature is enabled, the adapter always checks first for a resume-reading operation in progress. If this feature is not in progress, the adapter determines the next file based on the adapter configuration settings.

Step-by-step Operation

The diagram below illustrates how the Resume Reading feature operates along with pre- and post-file-transfer commands. In this example, the Collaboration executes the same Business Rule four times. Each execution causes the Collaboration to read another section of the file. When the Collaboration reads the final records, it executes the Post-Transfer commands

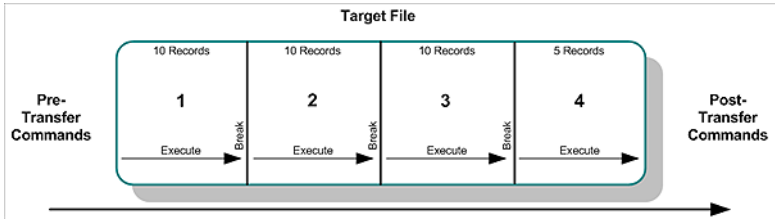


FIGURE 8 Resume Reading Operation

In this example, the reading happens in the following steps:

1. The adapter starts reading the file then reaches a break condition after a partial data read (the end of **Part 1**), the adapter's pre-transfer commands have already been executed. The resume-reading state is stored, and no post-transfer commands are executed. The adapter is waiting for the next execution of the Business Rule.
2. The resume-reading operation is in progress but still attains only partial data reads. The adapter reads from one break condition to the next (**Part 2** and **Part 3** in the figure) The resume-reading state is stored in each case, and the adapter executes the Business Rule once per each part.

3. The resume-reading operation is in progress and completes its data read during the final execution of the Business Rule (**Part 4**). The adapter reads from a break condition to the end of a file. No resume-reading state is stored, and any post-transfer commands are then executed.

In all of the previous steps, the Business Rule is executed repeatedly, and the current read position in the file changes on each execution. If the file is smaller than **Part 1** in the figure, the adapter does not reach a break condition. The operation is normal, and no resume-reading state is stored. The pre- and post-transfer commands are executed.

Operation Without Resume Reading Enabled

If the Resume Reading feature is not enabled:

- **Data-read Stop Then Restart**: Any unread data at the end of the file is ignored.
- **Resume Reading in Progress**: If there is a resume-reading operation in progress from a previous execution, an error is generated, and an exception is thrown.

Note – If there is a resume-reading operation in progress it cannot be interrupted and must be completed. The `executeBusinessRules()` method must be called enough times to fully consume the file. In other words, do not discontinue processing the file before it has been completely consumed.

To Avoid Storing a Resume Reading State

Sometimes a partial data-stream read is necessary even when the Resume Reading feature is enabled. For example, there could be some application logic on top of the record parsers, which might abandon the rest of the file because of a corrupted record and close the file successfully after reading only part of the file's content.

In this case, you must set the **LocalFileOTD.Configuration.ResumeReading** node to **False** before calling **finish()**. This setting tells the BatchLocalFile OTD to complete the operation without storing a resume-reading state. You can set up the Collaboration Rule to then send notifications or take other measures, as desired.

Data Stream-adapter Provider

You can use the BatchLocalFile OTD to implement the adapter's data streaming feature. This feature is also available with the FTP and record-processing OTDs. However, the BatchLocalFile OTD is a data stream-adapter provider, while the other two OTDs are only consumers. See "Streaming Data Between Components" on page 166 for details on how to use the OTD's data streaming feature.

Sequence Numbering

Sequence numbering for the BatchLocalFile OTD operates similar to sequence numbering for the BatchFTP OTD. See "Sequence Numbering" on page 136 for details.

Generating Multiple Files with Sequence Numbering

When using a Java Collaboration to generate multiple files that have sequence numbering using the BatchLocalFile interface, reset() must be called to signify the end of one file and the start of the next.

Handling Type Conversions

This feature in this OTD operates in the same way as type conversion for the BatchFTP OTD. See "Handling Type Conversions" on page 134 for details.

Recommended Practice

To parse records or construct payloads in your Collaboration, we recommend that you use the record-processing BatchRecord OTD together with the BatchLocalFile OTD. Using the two OTD together provides a number of advantages over just using the BatchFTP OTD.

Example 1: Parsing a Large File

For example, you have set up a Collaboration Rule to parse a large file and submit the records to a database or a JMS IQ Manager. If something goes wrong during the parsing process, the whole file needs to be transmitted again from the FTP server.

In contrast, streaming from a local file system can avoid later FTP transfers of the same file in case of error. This approach has the advantage of allowing you to use data streaming and the Resume Reading feature with large files (see "Streaming Data Between Components" on page 166 and "Resume Reading Feature" on page 150.

Example 2: Slow, Complex Query

Another scenario could be a case where a slow, complex SQL query is used to retrieve a number of records. The Collaboration Rule packs them into a **Payload** node using the record-processing OTD then sends them via FTP to an external system. If the FTP transfer fails, the SQL query must be executed again.

In contrast, if the data payload has been stored locally with the BatchLocalFile OTD, the FTP transfer can be repeated without the need to re-execute the SQL query. In such cases, you can also use data streaming and local-file appending.

In both cases, the use of a data-streaming link can significantly reduce the memory requirements compared to the in-memory data-payload transfer used with the BatchFTP OTD.

OTD Limitations

The BatchLocalFile OTD supports mapped drives and NFS mounted drives. It does not, however, support the mapping of the drives. That is, the drive must already be mapped or mounted. The adapter itself does not perform any mapping or mounting.

The OTD supports Universal Reference Identifiers (URIs) but the scheme must be left off, for example,`\\drive\directory\file_name`.

BatchRecord OTD

BatchRecord, the Batch Adapter's record-processing OTD, allows you to *parse* (extract) *records* from an incoming *payload* (payload data) or to create an outgoing payload consisting of records. Understanding the operation of this OTD and how to use it requires an explanation of some of these terms.

The word *payload* here refers to an in-memory buffer, that is, a sequence of bytes or a stream. Also, *records* in this context are not records in the database sense. Instead, a record simply means a sequence of bytes with a known and simple structure, for example, fixed-length or delimited records.

For example, each of the following record types can be parsed or created by this OTD:

- A large data file that contains a number of SAP IDocs, each with 1024 bytes in length.
- A data file that contains a large number of X12 purchase orders, each terminated by a special sequence of bytes.

The record-processing OTD can handle records in the following formats:

- **Fixed length**: Each record in the payload is exactly the same size.
- **Delimited**: Each record is followed by a specific sequence of bytes, for example, CR,LF.
- **Single**: The entire payload is the record.

When using character delimiters with DBCS data, use single byte character(s) or equivalent hex values with hex values that do not coincide with either byte of the double byte character.

BatchRecord OTD Structure

The BatchRecord OTD contains two top-level nodes, **Client**, **Configuration**, **PersistentState**, and **StateManager**. Expand these nodes to reveal additional sub-nodes.

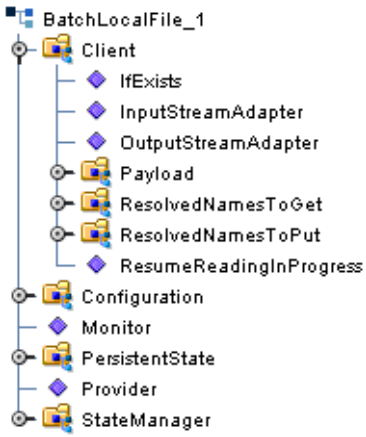


FIGURE 9 BatchRecord OTD Structure

OTD Structure and Operation

Each field node under the **Configuration** node of the OTD, corresponds to one of the adapter's record-processing configuration parameters.

Record-processing OTD Node Functions

The following list explains these primary nodes in the record-processing OTD, including their functions:

- **BatchRecord**: Represents the OTD's root node.
- **Configuration**: Each sub-node within this node corresponds to an adapter configuration parameter and contains the corresponding settings information, except for the **Parse or Create** parameter. See "BatchRecord Connectivity Map Properties" on page 99 for details.
  - **InputStreamAdapter** and **OutputStreamAdapter**: Allow you to use and control the data-streaming features of the OTD. For details on their operation, see "Streaming Data Between Components" on page 166.

Note – You can transfer data using the Payload node or by using data streaming (InputStreamAdapter and OutputStreamAdapter nodes), but you cannot use both methods in the same OTD.

Note – For the record-processing OTD, these configuration nodes are read-only. They are provided only for the purpose of accessing and checking the configuration information at run time.

- **Record**: A properties node that represents either:
  - The current record just retrieved via the `get()` method, if the call succeeded
  - The current record to be added to the data payload when `put()` is called
- **Payload**: The in-memory buffer containing the data payload byte array you are parsing or creating.

Caution – It is a good practice to use a byte array in all cases. Failure to do so can cause loss of data.

- `put()`: Adds whatever is currently in the **Record** node to the data payload. The method returns **true** if the call is successful.
- `get()`: Retrieves the next record from the data payload (or stream), and populates the **Record** node with the record retrieved. `get()` returns **true** if the call is successful.
- `finish()`: Allows you to indicate a successful completion of either a parse or create loop for both `put()` and `get()`.

Note – Use `reset()` to indicate any errors and to allow the OTD to clean up any unneeded internal data structures.

Using the Record-processing OTD

This OTD has the following basic uses:

- **Parsing a payload**: When the payload comes from an external system
- **Creating a payload**: Before sending the payload to an external system

A single instance of the OTD is not designed to be used for both purposes at the same time in the same Collaboration. To enforce this restriction, there is a setting under the adapter's General Settings parameters called **Parse or Create Mode, for which you can select either Parse or Create**.

Using `get()` and `put()`

The `get()` and `put()` methods are the heart of the OTD's functionality. If you call either method, the record retrieved or added is assumed to be of the type specified in the adapter configuration, for example, fixed-length or delimited.

The `get()` method can throw an exception, but generally this action only happens when there is a severe failure. One such failure is an attempt to call `get()` before the payload data (or stream if you are streaming) has been set. However, the best practice is to code the Collaboration to check the return value from a `get()` call. A return of **true** means a successful get operation; a **false** means the opposite.

Choosing the Parse or Create Mode

The adapter checks to ensure that the proper calls are made according to your mode setting. For example, calling `put()` in a parse-mode environment would cause the adapter to throw an exception with an appropriate error message explaining why. Calling `get()` in the create mode would also result in an error.

The adapter requires these restrictions because:

- If you are processing an inbound payload, you are calling `get()` to extract records from the payload (parsing). In this situation it makes little sense to call `put()`. Doing so at this point would alter the payload while you are trying to extract records from it. Calling `put()` would overwrite the payload and destroy the data you are trying to obtain.
- Conversely, when you are creating a payload by calling `put()`, you have no need to extract or parse data at this point. Therefore, you cannot call `get()`.

As a result, you can place the OTD on the source or destination side of a given Collaboration, as desired, and use the OTD for either parsing or creating a payload. However, you cannot parse and create at the same time. Implement your OTD in a Collaboration using the Java CAPS Collaboration Rules Editor.

Creating a Payload

When you want the payload data sent to an external system, you can place the OTD on the outbound side of the Collaboration interfacing with that system. Successive calls to `put()` build up the payload data in the format defined in the adapter configuration.

Once all the records have been added to the payload, you can drag and drop the payload onto the node or nodes that represent the Collaboration's outbound destination. Also, you can set an output stream as the payload's destination.

When you are building a data payload, you must take into account the type and format of the data you are sending. The adapter allows you to use the following formats:

- **Single Record**: This type of payload represents a single record to be sent. Each successive call to `put()` has the effect of growing the payload by the size of the data being put, and the payload is one contiguous stream of bytes.
- **Fixed-size Records**: This type of payload is made up of records, with each being exactly the same size. An attempt to `put()` a record that is not of the size specified causes an exception to be thrown.

- **Delimited Records**: This type of payload is made up of records that have a delimiter at the end. Each record can be a different size. Do not add any delimiters to this data type when it is passed to `put()`. The delimiters are added automatically by the adapter.
- **User Defined**: In this type of payload, the semantics are fully controlled by your own implementation.

Parsing a Payload

To represent payload data inbound from an external system, you map the data to the payload node in the OTD (from the Collaboration Rules Editor). In addition, you can specify an input stream as a source.

Either way, each successive call to `get()` extracts the next record from the payload. The type of record extracted depends on the parameters you set in the adapter's configuration, for example, fixed size or delimited.

You must design the parsing Collaboration with instructions on what to do with each record extracted. Normally, the record can be sent to another Collaboration where a custom OTD describes the record format and carries on further processing.

Fully Consuming a Payload

It is possible to fully consume a payload. That is, after a number of successive calls to `get()`, you can retrieve all the records in the payload. After this point, successive calls to `get()` return the Boolean **false**. You must design the business rules in the subject Collaboration to take this possibility into account.

Using Record Processing with Data Streaming

If you are using the record-processing OTD with data streaming, you must be careful not to overwrite the output files. If the OTD is continually streaming to a BatchLocalFile OTD that uses the same output file name, the OTD can write over files on the output side.

To avoid this problem, you must use either file sequence numbering or change the output file names in the Collaboration Rules. Sequence numbering allows the BatchLocalFile OTD to distinguish individual files by adding a sequence number to them. If you use target file names, post-transfer file names, or both, you can change the name of the output file to a different file name.

For more information on how to use these features, see "Sequence Numbering" on page 153 and "Pre/Post File Transfer Commands" on page 148.

BatchInbound OTD

The BatchInbound Input (Trigger) File adapter polls for an input file, renames the file with a GUID prefix, and triggers the Business Process or Collaboration.

The Batch adapter's **BatchInbound** OTD acts similar to the inbound File adapter, in that it regularly polls an input directory for inbound target files. But unlike the File adapter, when a file with the appropriate name is received by the BatchInbound adapter, the target file is renamed with the following pattern: *GUID.original_filename* to ensure that the file is not over-written and is only sent once. A GUID (Globally Unique Identifier) provides a unique, formatted string that represents a 128-bit value.

Note that the BatchInbound OTD does not read the file, but renames the file in such a way that it provides the name of the file that triggers the Business Process or Collaboration

BatchInbound OTD Structure

The BatchInbound OTD contains one top-level node, **BatchAppconnMessage**, with three fields, **PathDirName**, **OriginalFileName**, and **GUIDFileName**. These nodes provide the external input directory, original file name, and the GUID file name.

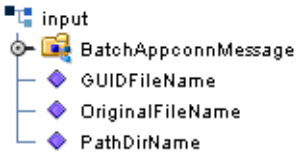


FIGURE 10 BatchInbound OTD Structure

Using Regular Expressions

A regular expression is a character string in which some characters provide special meaning in regard to matching patterns. This section explains some basic guidelines on how to use regular expressions with the Batch Adapter.

Regular Expressions: Overview

Regular expressions allow you to specify patterns for file names and directory names. Regular expressions are used for "get" operations (receiving or source), as opposed to name patterns which are used for "put" operations (sending or destination).

The BatchFTP, Batch FTPOverSSL, BatchSFTP, BatchLocalFile, and BatchInbound OTDs allow you to use regular expressions, for example, if you want to access all files with a specific extension.

Regular expressions operate as follows:

- The directory/file names can be defined as either:
  - Actual file names (everywhere)
  - Name patterns (all names for "put" operations and pre/post transfer names for get operations)
  - Regular expressions (target names for "get" operations)

  The difference between the regular expressions and name patterns is:
  - Regular expressions are used to match existing names on the FTP server or the local file system.
  - Name patterns are used to create names by replacing the special characters in the pattern.

    For more information on name patterns using special characters, see "Using Name Patterns" on page 162.

    You can specify an extension, for example, `.*\.dat$`. Then, each time the `get()` method is called, the adapter gets the next file with a `.dat` extension. The adapter then retrieves each file into the OTD's **Payload** node and updates the working file-name attribute with the name of the file currently being accessed.

    For another example, you can use the file-matching the pattern `data\.00[1-9]` to get the files `data.001`, then `data.002`, and so on. Note that in each case the "." is escaped, which is consistent with regular-expression syntax. It also matches to `xyzdata.001` and `xyz.data.001`, because it does not exclude anything before "data". To make "data" the exact start of the matching pattern you must use `^data\.00[1-9]` or `\A data\.00[1-9]`.

Caution – The use of regular expressions is an advanced feature and must be implemented carefully. An improperly formed regular expression can cause undesired data or even the loss of data. You must have a clear understanding of regular-expression syntax and construction before attempting to use this feature. It is recommended that you test such configurations thoroughly before moving them to production.

Entering Regular Expressions

You can enter a regular expression for the FTP or local file name in a variety of ways, for example, `.*\.dat$` or `^xyz.*\.dat$`. The first case indicates all files with an extension of `.dat`. The second case indicates all file names with an extension of `.dat` whose names start with `xyz`.

Another example could be `file[0-9]\.dat`. This expression specifies `file0.dat`, `file1.dat`, `file2.dat`, and so on, through `file9.dat`. This will also match `xyz.file0.dat`,

`xyz.file1.dat`, and so on. This type of expression will not exclude anything in front of "file". To exclude any characters before "file" (to make "file" the exact beginning) use `^file[0-9].dat` or `\Afile[0-9].dat`.

These types of regular expression patterns can be used for a get operation.

Regular Expressions and the Adapter

Note that the adapter provides a **File Name Is Pattern** or **Directory Name Is Pattern** configuration parameter after every property that allows a regular expression as an option. This feature allows you to specify that the pattern entered is a regular expression or just a static text entry to be interpreted literally.

Note – Regular expressions will resolve even with a partial match to the file name. The resolution process searches for the file name contents rather than the file name.

Rules for Directory Regular Expressions

There are special considerations you must be aware of when you use regular expressions for directory names. This section describes these restrictions and provides some examples.

Restrictions for Using Regular Expressions as Directory Names

The following restrictions apply when using regular expressions as directory names:

- The directory root, the drive name, and directory separators must be expressed exclusively. That is, do not express any of these elements as a regular expression. Only folder names are expected to appear as regular expressions.
- A regular expression must not span over the directory separators. If you use a regular expression between two directory separators, it must be one whole expression.
- Escape all directory separators in a directory pattern if the separator conflicts with a regular expression special character (that is, `" * [ ] ( ) | + { } : . ^ $ ? \`"). The back slash (\) is the special character used to escape other special characters in regular expressions. For Windows platforms, the directory separator is the back slash, so it must be escaped as \\.
- For the Windows Universal Naming Convention (UNC), the directory root (including the computer name and the shared root folder name) must be expressed exclusively. That is, do not express the computer name and shared root folder as a regular expression.
- Directory separators are platform dependent, for example:
  - For **Windows** platforms, the directory path follows this pattern:

    ```
    drive:\\regexp1\\regexp2\\regexp3 ...
    ```

    or for **Windows UNC** notation, the directory path follows this pattern:

```
\\\\machineName\\shared_folder\\regexp1\\regexp2\\regexp3 ...
```

- For **UNIX** platforms, including mounted directories, the directory path follows this pattern:

```
/regexp1/regexp2/regexp3 ...
```

Regular Expression Directory Name Examples

The following are several examples of regular expression directory name usage:

- Windows:

```
c:\\eGate$\\^client\\collab\D\\ ...
```

  The expression `\D` indicates any non-digit character.

```
d:\\a.b\\c.d\\e.f\\g.h\\[0-9]\\ ...
```

  The symbol "." means any character

- Windows UNC notation:

```
\\\\My_Machine\\public\\xyz$\\^abc
```

  The prefix for Windows UNC notation is `\\`. After escaping, it becomes `\\\\`.

- UNIX:

```
/abc\d/def/ghi/ ...
```

  The expression `\d` means any digit character.

```
/^PRE[0-9]{5}\.dat$/ ...
```

  This expression means to begin with `PRE` followed by a five-digit number and use a `.dat` extension. The symbol `\.` means to interpret the real character (a period) instead of any character. Therefore, `PRE12345.dat` does match, but `PRE123456dat` does not.

Using Name Patterns

The Batch Adapter allows you to use a Name Pattern, that is, special characters that symbolize often-used information as *short-hand*. You can use these character combinations to specify place holders for this specific information. Using these characters, you can quickly convey date/time, number, and file-name information.

The BatchFTP, Batch FTPOverSSL, BatchSFTP, BatchLocalFile, and BatchInbound OTDs allow you to use special characters or specify a name pattern. A name pattern allows you to specify

patterns for file names and directory names. Name patterns are used for "put" operations (sending or destination) , as opposed to regular expressions which are used for "get" operations (receiving or source).

Special characters are utilities the eWays use for file-name pattern. The general rules for their use are:

- Use `%` to indicate the special character that needs to be expanded.
- Use `%%` to indicate the escaped character `%`; for example, `abc%%d` means `abc%d`, and the `%d` is not expanded again.

For example, for a put operation, a pattern such as `file%#.dat` can be used. This pattern uses the sequence number setting in the configuration, and each put creates successive files named `file1.dat`, `file2.dat`, and so on.

For information on regular expressions, see "Using Regular Expressions" on page 159.

Types of Name Patterns

The eWay provides the following types of name patterns:

- **Date/Time stamp**: Uses the format `%[GyMdhHmsSEDFwWakKz]`, for example, `abc%y%y%y%y` means `abc2001` (see Table 58 for more information).
- **Sequence number**: Uses the format `%#, %5#`, for example, `abc%#` means `abc1`, `abc2`, `abc3`, and so on; for another example, `abc%5#` (zero-padded) means `abc00001`, `abc00002`, `abc00003`, ..., `abc00010`, ..., `abc00100`, and so on.
- **Working-file name**: Uses the format `%f`; normally, it is used for pre- or post-file-transfer commands (see "Pre/Post File Transfer Commands" on page 148), for example, `%f.abc` means `working_filename.abc`.

The sequence of expansion operates in the reverse order of the previous list, that is, first the file name is expanded, then the sequence number, and finally the time stamp.

Some additional examples of name pattern:

- `abc.%y%y%y%y%M%M%d%d.%h%h%m%m%s%s%S%S%S` means `abc.20011112.162532678`
- `abc%#.def%#` means `abc2.def3`
- `%f.%#` means `xxxxx.4`, `xxxxx.5`, ... (Where `xxxxx` is the working-file name)

Resolving Names

Typically, the pre/post names with name patterns or regular expressions are resolved during `get()` and `put()` method calls. But sometimes, in using Collaboration Rules, the eWay has to get the resolved names before the actual `get()` or `put()` call.

In such cases, you can get the resolved names in this way through the **ResolvedNamesForGet** and **ResolvedNamesForPut** nodes in the BatchFTP OTD, for example:

```
getResolvedNamesForPut().getTargetFileName()
```

The previous code yields `file1` based on the pattern `file%#`. In this usage, the OTD nodes can be used to make the desired method call. See "BatchFTP OTD Node Functions" on page 133 for more information on BatchFTP OTD nodes.

Date/time Format Syntax

The eWay uses the Java simple default date and time format syntax (U.S. locale). To specify these formats for name pattern, you must use a time pattern string.

Note – The eWay uses the Java standard for date/time stamps from the Java class java.text.SimpleDateFormat. Some of these formats can differ from the list given here, depending on the Java SDK version you are using.

In these patterns, all ASCII letters are reserved as pattern letters. See Table 58 for a complete list.

TABLE 58 Time Pattern Strings and Meanings

Symbol	Meaning	Presentation	Example
%G	Era designator	Text	AD
%y	Year	Number	1996
%M	Month in year	Text and number	July & 07
%d	Day in month	Number	10
%h	Hour in a.m./p.m. (1 through12)	Number	12
%H	Hour in day (0 through 23)	Number	0
%m	Minute in hour	Number	30
%s	Second in minute	Number	55
%S	Millisecond	Number	978
%E	Day in week	Text	Tuesday
%D	Day in year	Number	189
%F	Day of week in month	Number	2 (second Wednesday in July)

TABLE 58 Time Pattern Strings and Meanings *(Continued)*

Symbol	Meaning	Presentation	Example
%w	Week in year	Number	27
%W	Week in month	Number	2
%a	Marker for a.m./p.m.	Text	PM
%k	Hour in day (1 through 24)	Number	24
%K	Hour in a.m./p.m. (0 through 1)	Number	0
%z	Time zone	Text	Pacific Standard Time

The general rules for date/time formats are:

- **Text**: The count of pattern letters determines the format as follows:
  - For four or more pattern letters, use the full form.
  - For fewer than four, use the short or abbreviated form if one exists.

  Number: The minimum number of digits as follows:
  - Shorter numbers are zero-padded to this amount.
  - The year is handled differently; that is, if the count of "y" is two, the year is truncated to two digits.

  Text and number: For three or more pattern letters, use text; otherwise use a number.
- **Quotes and delimiters**: Use these symbols as follows:
  - Enclose literal text you want rendered within single quotes.
  - Use double quotes to mean single quotes.
  - Use commas for delimiters.

TABLE 59 U.S. Locale Date/time Patterns

Format Pattern	Result
yyyy.MM.dd, G, 'at' hh:mm:ss, z	1996.07.10 AD at 15:08:56 PDT
E, M, dd, "yy	Wednesday, July 10, '96
h:mm, a	12:08 PM
h, 'o"clock' a, z	12 o'clock PM., Pacific Daylight Time
K:mm a, z	0:00 p.m., PST
yyyyy.M.dd, G, hh:mm, a	1996.July.10 AD 12:08 PM

Additional Batch Adapter Features

This chapter explains additional features of the Batch Adapter, including Data Streaming, SOCKS, SSH Tunneling, and Ensuring Secure FTP Data Transfers.

What's in This Chapter

Streaming Data Between Components

Components in the Batch Adapter implement a feature for *data streaming*. This chapter explains data streaming, how it works, and how to use it with the adapter and eGate Integrator.

Introduction to Data Streaming

Data streaming provides a means for interconnecting any two components of the adapter by means of a *data stream channel*. This channel provides an alternate way of transferring the data between the Batch Adapter components. Streaming is available between BatchLocalFile and BatchFTP, or BatchLocalFile and BatchRecord.

Each OTD component in the adapter has a **Payload** node. This node represents the in-memory data and is used when the data is known to be relatively small in size or has already been loaded into memory. The node can represent, for example, the buffer in the record-processing OTD, as it is being built or parsed, or the contents of a file read into memory.

Instead of moving the data all at once between components in eGate's memory, you can use a *data-stream channel* to provide for streaming the data between them a little at a time, outside of eGate.

Data streaming was designed primarily to handle large files, but you can use it for smaller data sizes as well.

Use the eGate Enterprise Designer's Collaboration Rules Editor to set up data-streaming operations. The rest of this section explains the data streaming feature and how to set it up.

Note – Payload-based and streaming-based transfers are mutually exclusive. You can use one or the other but not both for the same data.

Overcoming Large-file Limitations

The primary advantage of using data streaming is that it helps to overcome the limitations of dealing with large files. For example, if you have a 1-gigabyte file that contains a large number of records, you need a large amount of resources to load the payload into memory just to parse it.

Streaming allows you to read from the file, little by little, using a data-streaming mechanism. This way, you do not need to load the file into the eGate system's memory. Using streaming is not as fast as using in-memory operations, but it is far less resource-intensive.

Using Data Streaming

Each data-streaming transfer involves two OTDs in a Collaboration as follows:

- One provides the *stream adapter*.
- The other consumes the stream adapter to perform the data transfer.

Caution – Implementing InputStream must support skip() with negative numbers as an argument.

This section explains how the two data-streaming OTDs operate to effect the transfer of data.

Data-streaming Operation

Each of the OTDs in the Batch Adapter exposes stream adapter nodes that enable any OTD to participate in data-streaming transfers. The nodes are named **InputStreamAdapter** and **OutputStreamAdapter**. You can associate the stream adapters by using the drag-and-drop features of the eGate Enterprise Designer.

The **InputStreamAdapter** (highlighted) and **OutputStreamAdapter** nodes in the OTD are used for data streaming. This feature operates as follows:

- **Stream-adapter consumers**: The FTP and the record-processing OTDs can only *consume* stream adapters. Therefore, their stream-adapter nodes are *write-only*. Their node values can be *set* (modified).
- **Stream-adapter provider**: The local file OTD can only *provide* stream adapters, so its stream-adapter nodes are *read-only*. Its node value can only be *retrieved*.

The local file OTD is always the stream provider, and the FTP and record-processing OTDs are the consumers.

Note – For an explanation of the adapter's different types of OTDs, see Understanding Batch Adapter OTDs .

Data Streaming Versus Payload Data Transfer

Use of the **InputStreamAdapter** and **OutputStreamAdapter** nodes is an alternative to using the **Payload** node as follows:

- Use these stream adapter nodes to transfer data if you want data streaming.
- Use the **Payload** node for a data transfer *without* data streaming (payload data transfer).

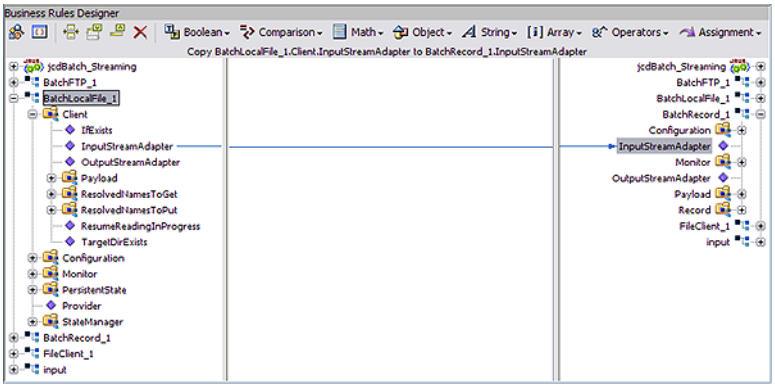
All operations that, in payload data transfer, *read* from the **Payload** node require the **InputStreamAdapter** node when you are setting up data streaming. Using the same logic, all operations that, in payload data transfer, *write* to the **Payload** node require **OutputStreamAdapter** node for data streaming.

Do *not* confuse the stream adapter nodes with the `get()` and `put()` methods on the OTDs. For example, the BatchFTP OTD's client interface `get()` method *writes* to the **Payload** node during a payload transfer, so it requires an **OutputStreamAdapter** node to write to for data streaming. In contrast, the record-processing OTD's `get()` method *reads* from the **Payload** node during a payload transfer, so for data streaming, `get()` requires an inputStreamAdapter node to read from.

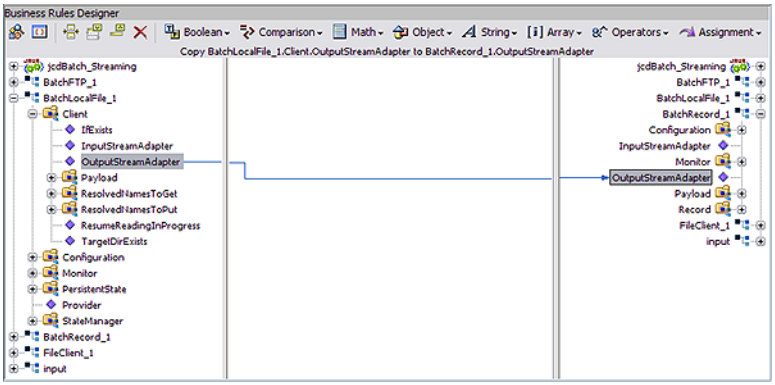
Data Streaming Scenarios

The four typical data-streaming scenarios are:

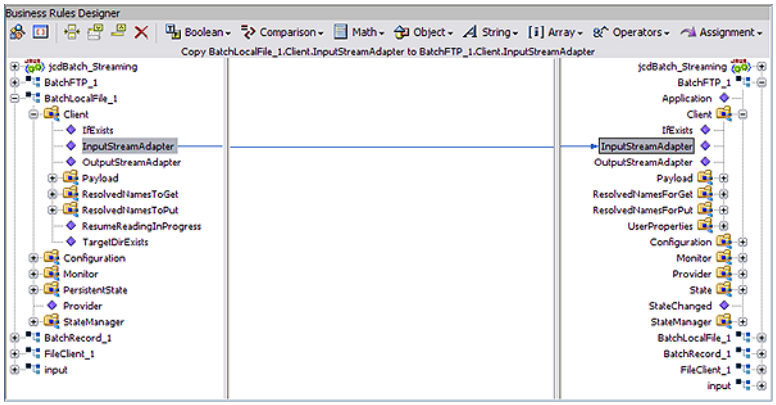
- Transfer data from a local file system (BatchLocalFile) to a record-processing (BatchRecord) setup. This scenario uses the **InputStreamAdapter** OTD node (see "Data Streaming Scenarios" on page 168).



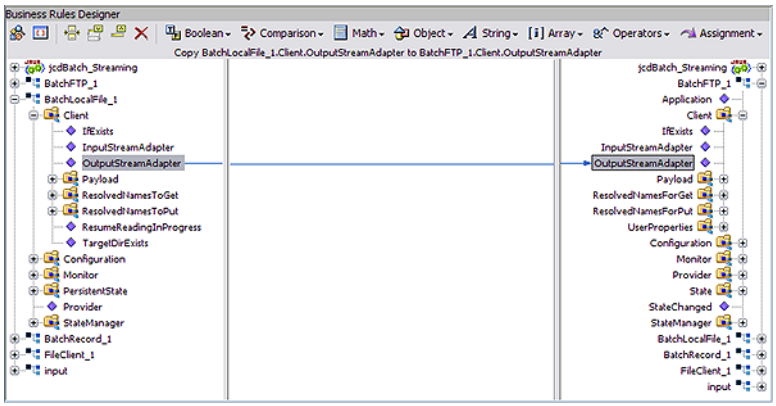
- Transfer data from a record-processing (BatchRecord) setup to a local file system (BatchLocalFile) using the **OutputStreamAdapter** OTD node (see "Data Streaming Scenarios" on page 168).



- Transfer data from a local file system (BatchLocalFile) to a remote FTP (BatchFTP) setup. This scenario uses the **InputStreamAdapter** OTD node (see "Data Streaming Scenarios" on page 168).



- Transfer data from a remote FTP server (BatchFTP) to a local file system (BatchLocalFile) using the **OutputStreamAdapter** OTD node (see "Data Streaming Scenarios" on page 168).



Consuming-stream Adapters

This section explains how to use consuming-stream adapters.

▼ To obtain a stream

- **Use the `requestXXStream()` method to obtain the corresponding XX stream.**

▼ To Use a Stream

- **Perform the transfer using the methods provided by the stream.**

▼ To Dispose of a Stream

1. **Release any references to the stream.**
2. **Release the stream (XX) using the `releaseXXStream()` method. Some of the OTDs support post-transfer commands. The Success parameter indicates whether these commands are executed. Do not close the stream.**

Stream-adapter Interfaces

This section provides the Batch Adapter's OTD stream-adapter Java interfaces. This information is only for advanced users familiar with Java programming, who want to provide custom OTD implementations for stream-adapter consumers or providers.

Inbound Transfers

The following Java programming-language interface provides support for inbound transfers from an external system:

```
public interface com.stc.eways.common.eway.streaming.InputStreamAdapter {
public java.io.InputStream requestInputStream() throws StreamingException;
public void releaseInputStream(boolean success) throwsStreamingException;
}
```

Outbound Transfers

The following Java interface provides support for outbound transfers to an external system:

```
public interface com.stc.eways.common.eway.streaming.OutputStreamAdapter {
public java.io.OutputStream requestOutputStream() throws StreamingException;
public void releaseOutputStream(boolean success) throws StreamingException;
}
```

SOCKS FTP Support

This section explains the SOCKS FTP features available for the Batch Adapter.

SOCKS

SOCKS is an Internet Engineering Task Force (IETF) -approved standard (RFC 1928) generic, proxy protocol for TCP/IP-based network applications. This simple protocol supports a flexible framework for developing secure communications. SOCKS accomplish this by easily integrating other security technologies.

Note – The adapter only supports SOCKS protocols that conform to this IETF standard.

There are two versions of the SOCKS protocol.

- SOCKSv4 (version 4), that provides the following functions:
  - requests connections
  - Setup proxy clients
  - transmits application data
- SOCKSv5 (version 5) that includes all the functionality of version 4 and also provides authentication

Both the SOCKSv4 and SOCKSv5 protocols are supported by the Batch Adapter. To enable support, the following properties must be specified in the Batch Adapter Properties Sheet:

- SOCKS server name
- SOCKS server port number
- User name
- Encrypted password

Details of these configuration parameters are provided under "SOCKS Configuration Properties" on page 173.

Note – In the Collaboration Rules, make sure you set the SOCKS version number to 4, 5, or- 1 (unknown). Do not set this value to any other number.

SOCKS: Overview

SOCKS embodies two components, the SOCKS Server (implemented at the application layer), and the SOCKS Client (implemented between the application and transport layers).

In essence, the purpose of the SOCKS protocol is to allow a host on one side of a SOCKS Server to interact with a host on the other side of the Server, subject to authentication, without passing IP packets directly between the two.

SOCKS Proxy Server

The SOCKS proxy server connects to the application server on behalf of the application client and relays data between the client and an application server. From the application server's perspective, the SOCKS proxy is the client.

SOCKS and the Batch Adapter

Negotiation Methods

The BatchFTP Adapter supports the following methods used to define the negotiation phase of authentication between the Socks Client and Server:

- No-authentication (no authentication required)
- User/password (user name and password)

SOCKS Configuration Properties

The Batch Adapter contains a number of properties used to configure SOCKS with the BatchFTP Adapter. These properties are configured using the BatchFTP Properties Sheet accessed from the Connectivity Map and the Environment Explorer.

- **Socks Enabled:** Specifies whether the FTP command connection goes through a SOCKS server. A value of No indicates that the adapter is not connecting to a SOCKS server. In this case, all other parameters under the SOCKS section are ignored.
- **Socks Host Name**: Specifies the SOCKS host name. When you are communicating with a SOCKS server, enter the SOCKS server name in this parameter.
- **Socks Server Port**: Specifies the port number of the SOCKS server.
- **Socks Version**: Specifies the SOCKS server version. A value of 4 or 5 for SOCKSv4 or SOCKSv5 provides the best performance, but the default value **Unknown can if the version is in question.**
- **Socks User Name**: Specifies the user name that matches the associated password used for authentication with a SOCKS5 server. This parameter is applied when user/password negotiation method is used.
- **Socks Password**: Specifies the password to use along with the user name for authentication with a SOCKS5 server. This parameter is applied when user/password negotiation method is used.

For information on the BatchFTP configuration properties, see "BatchFTP Adapter Connectivity Map Properties" on page 26, and "BatchFTP Adapter Environment Properties" on page 42.

SSH Tunneling Support

This section explains the Batch Adapter's Secure Shell (SSH) tunneling features. SSH tunneling is also called SSH port forwarding.

The Batch Adapter encrypts the command channel of FTP utilizing SSH. To encrypt data, you can encrypt a file prior to sending it, using your preferred method or that of the receiver. The received file can then be decrypted by the recipient. If Secure FTP (FTP over SSH or FTP over SSL) is required, use the Secure FTP OTDs ((BatchFTPOverSSL, BatchSFTP, and BatchSCP).

SSH Tunneling: Overview

Developed by SSH Communications Security Ltd., Secure Shell (SSH) is a program that allows a computer to log onto another computer over a network to move files over the network and execute commands. SSH is intended as a replacement for **rlogin**, **rsh**, **rcp**, and **rdist**.

SSH provides strong authentication and secure communications over non-secure channels. SSH protects a network from attacks such as IP and DNS spoofing, IP source routing, and interception of plaintext passwords and authentication data. If an attacker manages to take over a network, he can only force SSH to disconnect. The content and the connection are secure when encryption is enabled.

When you are using the SSH **slogin** (instead of **rlogin**), the entire logged-on session, including the transmission of the password, is encrypted. As a result, it is almost impossible for an outsider to collect passwords.

Note – For improved security, the number of times the adapter can log on during a single session is limited because, during a disconnect, the SSH tunnel is not closed. This method of operation allows you to establish another connection without logging on.

For more information on SSH and how to use it, see the following Web site:

`http://www.openssh.com`

Additional Software Requirements

The adapter makes use of additional software applications. The adapter also supports either of the following applications for SSH tunneling:

- **OpenSSH**: an encryption and authentication tool for UNIX. For more information go to:

  `http://www.openssh.org`

- `Plink.exe`: Plink is a Win32-only command-line interface to the PuTTY Telnet/SSH client. For more information visit:

```
http://www.chiark.greenend.org.uk/~sgtatham/putty
```

In either case, the you are responsible for downloading, installing, and properly configuring the necessary software. You must refer to the appropriate software provider for support and documentation.

SSH Tunneling and the Batch Adapter

To use SSH tunneling to provide for secure logon IDs and passwords, the BatchFTP Adapter uses the additional SSH-tunneling software (see "Additional Software Requirements" on page 174).

Enabling SSH Tunneling

To enable SSH tunneling, select **Yes** under the **SSH Tunneling Enabled** parameter in the adapter connection configuration (see "SSH Tunneling Configuration Parameters" on page 176). You can use the SSH-tunneling software in either of the following ways:

- By using an existing SSH channel where a secure connection has already been established
- By internally launching an SSH process for the adapter's use

Using an Existing Channel

To use an existing channel, select **Yes** under the **SSH Channel Established** parameter in the configuration. The adapter then operates under the assumption that you have already established the SSH channel using the additional software. Once you set this parameter to **Yes**, the adapter automatically uses that channel.

Using an Internal Channel

If you choose **No**, under the **SSH Channel Established** parameter, the adapter launches a process within Java CAPS to establish a channel. In this case, you must specify, under the **SSH Command Line** parameter, a full and correct command-line statement for your SSH-tunneling application and environment.

Note – You can obtain this information from the SSH-tunneling application's configuration. See the application's documentation for details.

You must enter a correct and complete command-line statement. That is, all necessary command line parameters must be provided so that the SSH-tunneling software can run correctly without requiring further interaction.

Check the accuracy of this information by executing the command line from the shell. If the software prompts for more information, add the required information to the command line and try again. Continue this process until the software starts and operates properly without additional action.

Note – You may need to launch the application at least once from the shell before using it in the adapter. This requirement depends on the SSH-tunneling application and platform. Some applications prompt for trust-related information on the first attempt, to connect to a remote host.

Port-forwarding Configuration

Through SSH tunneling, the FTP command connection is protected. This mechanism is based on an existing SSH port-forwarding configuration. You must configure SSH port forwarding on the *SSH listen host* before you configure the supporting adapter Connection.

For example, on the Java CAPS client host **localhost**, you can issue a command, such as:

```
ssh -L 4567:atlas:21 -o BatchMode=yes atlas
```

Under the adapter's configuration for the previous example, you must specify:

- **localhost** for the parameter **SSH Listen Host**
- **4567** for the parameter **SSH Listen Port**

In this case, the adapter connects to the FTP server **atlas:21** through an SSH tunnel.

SSH Tunneling Configuration Parameters

You must set the following SSH tunneling parameters to configure the adapter Connection:

- **SSH Tunneling Enabled**: Specifies whether the FTP command connection is secured through an SSH tunnel:
  - **No:** indicates that all other parameters in this section are ignored.
- **SSH Channel Established**: Specifies whether the adapter needs to launch an SSH subprocess:
  - **No**: indicates that there is no existing SSH channel for an FTP transfer.
  - **Yes**: indicates that an SSH channel has been established, so it is not necessary for the adapter to spawn an SSH subprocess. If you select **Yes**, the following parameters are required:
  - **SSH Listen Host**
  - **SSH Listen Port**
- **SSH Command Line**: Specifies the command line used to establish an SSH channel. This parameter is required only when you set the **SSH Channel Established** parameter to **No**.

  The command-line syntax can be different, depending on the specific SSH client implementation. See your SSH-tunneling support software user's guides for details.
- **SSH Listen Host**: Specifies the host name where the SSH support software runs, as well as the host it listens to.

This parameter is required only when you set the **SSH Channel Established** parameter to **Yes**. If you choose **No**, the **Listen Host** is always **localhost** because the SSH support software is always started from the local host.

- **SSH Listen Port**: Specifies the port number that the SSH-tunneling support software uses to check for incoming connections. This port number can be any unused port number on the SSH listen host.
- **SSH User Name**: Specifies an SSH user name. This parameter can be required when the setting for the **SSH Channel Established** parameter is **No**.
- **SSH Password**: Specifies an SSH password corresponding to the user name entered under **SSH User Name**. This parameter can be required only when the setting for the **SSH Channel Established** parameter is **No**. For more information, see **SSH User Name**.

For more information, see "SSH Tunneling Configuration Parameters" on page 176.