SeeBeyond ICAN Suite

Batch eWay Intelligent Adapter User's Guide

Release 5.0.8

Contents

Chapter 1

Introducing the Batch eWay

This chapter provides a brief overview of the Batch eWay, and presents an outline of the information provided in this user's guide.

What's in This Chapter

1.1 About the Batch eWay Intelligent Adapter

All eWays provide a communication bridge between the eGate environment and one or more external systems.

The Batch eWay Intelligent Adapter, referred to as the Batch eWay throughout this document, 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 eWay enables eGate to use an FTP connection to exchange data with other network hosts for the purpose of receiving and delivering objects stored in files.

1.1.1 Batch eWay OTDs

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

The Batch eWay 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 Shell (SSH) protocol.
- **BatchSFTP**: The SFTP OTD provides secure data transfer using Secure Shell (SSH) protocol.

- **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 eWay supports standard FTP according to RFC-959.*

1.2 What's New in This Release

This release of Batch eWay adds three Object Type Definitions (OTDs) that provide secure data transfer functionality using FTP SSL and FTP SSH protocols. Two new sample Projects have been included that demonstrate the new OTDs.

Two additional samples have also been included to demonstrate how to overcome large-file limitations by using OTDs that expose the InputStreamAdapter and the OutputStreamAdapter fields.

1.3 What's in This Document

This section provides a brief outline of this user's guide.

1.3.1 Organization of Information

This book includes the following chapters:

- Chapter 1 **"Introducing the Batch eWay"** provides an overview of the Batch eWay Intelligent Adapter.
- Chapter 2 **"Installing the Batch eWay"** provides the supported operating systems and system requirements for the Batch eWay. It also includes directions for installing the eWay and accessing the accompanying documentation and sample Projects.
- Chapter 3 **"Configuring the Batch eWay"** describes the process of configuring the Batch eWay to run in your environment.
- Chapter 4 **"Understanding Batch eWay OTDs"** provides an overview of the Object Type Definitions (OTDs) available with the Batch eWay.
- Chapter 5 **"Additional Features"** provides information on the Data Streaming, SSH Tunneling, and SOCKS features of the Batch eWay:
- Chapter 6 **"Using the Batch eWay With eInsight"** describes how to use the Batch eWay with the ICAN Suite's eInsight Business Process Manager and the Web Services interface.

- **Chapter 7** **"Implementing a Batch eWay Project"** describes the features and functionality of the Batch eWay using the eGate Integrator and the Collaboration Editor (Java).

1.3.2 Batch eWay Javadoc

The Batch eWay Javadoc documents the available Java methods provided with the Batch eWay. The Javadoc is uploaded with the eWay's documentation file, **BatcheWayDocs.sar**, and downloaded from the Documentation tab of the Enterprise Manager. To access the full Javadoc, extract the Javadoc to an easily accessible folder, and double-click the **index.html** file.

1.3.3 Scope of the Document

This user's guide provides a description of the Batch eWay Intelligent Adapter. It includes directions for installing the eWay, configuring the eWay properties, and implementing the eWay's sample Projects. This document is also intended as a reference guide, listing available properties, functions, and considerations. For a reference of available Batch eWay Java methods, see the associated Javadoc.

1.3.4 Intended Audience

This guide is intended for experienced computer users who have the responsibility of helping to set up and maintain a fully functioning ICAN Suite system. This person must also understand any operating systems on which the ICAN Suite will be installed (Windows, UNIX, and/or HP NonStop Server), and must be thoroughly familiar with Windows-style GUI operations.

1.3.5 Document Conventions

The following conventions are observed throughout this document.

Table 1 Document Conventions

Text	Convention	Example
Names of buttons, files, icons, parameters, variables, methods, menus, and objects	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
Command line arguments, code samples	`Fixed` font. Variables are shown in ***bold italic***.	`bootstrap -p` ***password***

Table 1 Document Conventions (Continued)

Text	Convention	Example
Hypertext links	Blue text	See **Document Conventions** on page 16
Hypertext links for Web addresses (URLs) or email addresses	Blue underlined text	http://www.seebeyond.com docfeedback@seebeyond.com

1.4 SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.seebeyond.com

1.5 SeeBeyond Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

docfeedback@seebeyond.com

1.6 Additional Licensing Considerations

SeeBeyond Technology Corporation has integrated software from 3SP (J2SSH Maverick) and /n software (IP*Works SSL) are included with the Batch eWay.

Use of the integrated software libraries outside of the ICAN Batch eWay Intelligent 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 ICAN Batch eWay as a run-time component.

SeeBeyond Technology Corporation prohibits ICAN Batch eWay 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.

SeeBeyond Technology Corporation prohibits ICAN Batch eWay 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.

Chapter 2

Installing the Batch eWay

This chapter contains installation information for the Batch eWay.

What's in This Chapter

2.1 Supported Operating Systems

The Batch eWay is available on the following operating systems:

- Windows 2000, Windows XP, and Windows Server 2003
- HP NonStop Server G06.22
- HP Tru64 V5.1A
- HP-UX 11.0, 11i (PA-RISC), and 11i V2.0 (11.23)
- IBM AIX 5.1L and 5.2
- IBM z/OS V1.3 and V1.4
- Red Hat Enterprise Linux AS 2.1 (Intel x86)
- Red Hat Linux 8 (Intel x86)
- Sun Solaris 8 and 9
- Suse Linux Enterprise Server 8 (Intel x86)
- Japanese Windows 2000, Windows XP, and Windows Server 2003
- Japanese HP-UX 11.0, 11i (PA-RISC), and 11i V2.0 (11.23)
- Japanese IBM AIX 5.1L and 5.2
- Japanese Sun Solaris 8 and 9
- Korean Windows 2000, Windows XP, and Windows Server 2003
- Korean HP-UX 11.0, 11i (PA-RISC), and 11i V2.0 (11.23)
- Korean IBM AIX 5.1L and 5.2

- Korean Sun Solaris 8 and 9

WebLogic and WebSphere Application Server Support

- In addition to the operating systems listed above, this eWay is also supported on WebSphere™ and WebLogic™ application servers. This is limited to outbound mode using Java Collaborations. For additional information see the *eGate Integrator User's Guide.*

2.2 System Requirements

- The system requirements for the Batch eWay are the same as those for the eGate Integrator. For more information, refer to the *SeeBeyond ICAN Suite Installation Guide*. It is also helpful to review the Readme.txt for any additional requirements prior to installation. The Readme.txt is located on the installation CD-ROM.
- In addition, the JAVA_HOME variable must be set to the location of the JDK.
- To enable Web Services, you must install and configure the SeeBeyond ICAN Suite eInsight Business Process Manager.

2.3 Installing the Batch eWay

During the eGate Integrator installation process, the Enterprise Manager, a web-based application, is used to select and upload eWays (eWay.sar files) from the eGate installation CD-ROM to the Repository.

When the Repository is running on a UNIX operating system, eGate and the eWays are installed using the Enterprise Manager from a computer running Windows, connected to the Repository server.

2.3.1 Installing the Batch eWay on an eGate Supported System

Installing the Batch eWay on an eGate supported system

The ICAN Suite installation process includes the following operations:

- Install the eGate Repository
- Upload products to the Repository
- Download components (including the eGate Enterprise Designer and Logical Host)

Follow the directions for installing the ICAN Suite in the *SeeBeyond ICAN Suite Installation Guide.* After you have installed eGate and any other purchased core products, do the following:

1. From the Enterprise Manager's **ADMIN** tab, browse to the **Add-ons** directory and select the **ProductsManifest.xml**, and click **Submit**. The available Add-on product list is now displayed.
2. Browse to and select the following files located in the **Add-ons** directory:
   - **FileeWay.sar** (to install the File eWay, used in the sample Project)
   - **BatcheWay.sar** (to install the Batch eWay)
3. Click **upload now** to upload the selected products.
4. Click on the Manifest File field's **Browse** option, browse to the Add-ons > **Documentation** directory, select the **ProductsManifest.xml**, and click **Submit**. The available Add-ons documentation list is now displayed.
5. Select and upload the following file:
   - **BatcheWayDocs.sar** (to upload the e-Mail eWay User's Guide, Javadoc, Readme, and sample Projects to the Enterprise Manager).
6. Click **upload now** to upload the selected product.
7. Continue installing the eGate Integrator as instructed in the *SeeBeyond ICAN Suite Installation Guide.*

Adding the eWay to an Existing ICAN Suite Installation

If you are installing the eWay to an existing ICAN installation, do the following:

1. Complete steps 1 through 5 above.
2. Open the Enterprise Designer and select **Update Center** from the Tools menu. The Update Center Wizard appears.
3. For Step 1 of the wizard, simply click **Next**.
4. For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field. Click **Next**.
5. For Step 3 of the wizard, wait for the modules to download, then click **Next**.
6. The wizard's Step 4 window displays the installed modules. Click **Finish.**
7. When prompted, restart the IDE to complete the installation.

2.3.2 After Installation

Once the eWay is installed and configured, it must then be incorporated into a Project before it can perform its intended functions. See the *eGate Integrator User's Guide* for more information on incorporating the eWay into an eGate Project.

Chapter 3

Configuring the Batch eWay

This chapter explains how to configure the Batch eWay properties.

What's in This Chapter

3.1 Creating and Configuring the Batch eWay

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

- **BatchFTP** — includes Connectivity Map and Environment properties.
- **BatchFTPOverSSL** — includes Connectivity Map and Environment properties.
- **BatchSCP** — includes Connectivity Map and Environment properties.
- **BatchSFTP** — includes Connectivity Map and Environment properties.
- **BatchLocalFile** — includes Connectivity Map and Environment properties.
- **BatchRecord** — includes Connectivity Map properties.
- **BatchInbound** — includes Connectivity Map properties.

All Batch eWays contain properties that are accessed from the **Connectivity Map**. These properties most commonly apply to a specific eWay, and may vary from other eWays (of the same type) in the Project.

The **FTP**, **FTPOverSSL**, **SCP, SFTP**, and **LocalFile** Batch eWays also possess Environment properties that are accessed from the **Environment Explorer tree**. These properties are commonly global, applying to all eWays (of the same type) in the Project.

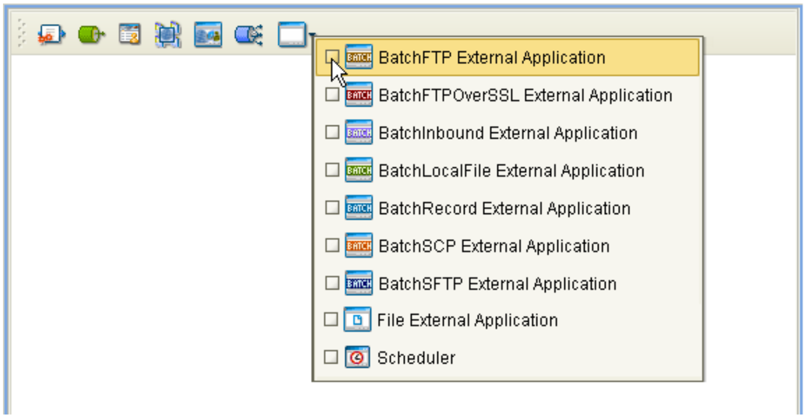
3.1.1 Selecting a Batch External Application

To create a Batch eWay, you must first create a Batch External Application in your Connectivity Map. Batch eWays 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 (see Figure 1). The selected Batch External Application icon appears on the Connectivity Map toolbar.

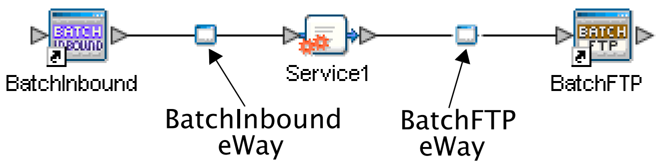
Figure 1 External Applications Selection Menu


BatchFTP External Application
BatchFTPOverSSL External Application
BatchInbound External Application
BatchLocalFile External Application
BatchRecord External Application
BatchSCP External Application
BatchSFTP External Application
File External Application
Scheduler

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 eWay (see Figure 2).

Figure 2 eWay Location.



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

3.1.2 Modifying the eWay Properties

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

Modifying the Batch eWay (Connectivity Map) Properties

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

Modifying the Batch eWay (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 eWays, and click **OK** to save the settings.

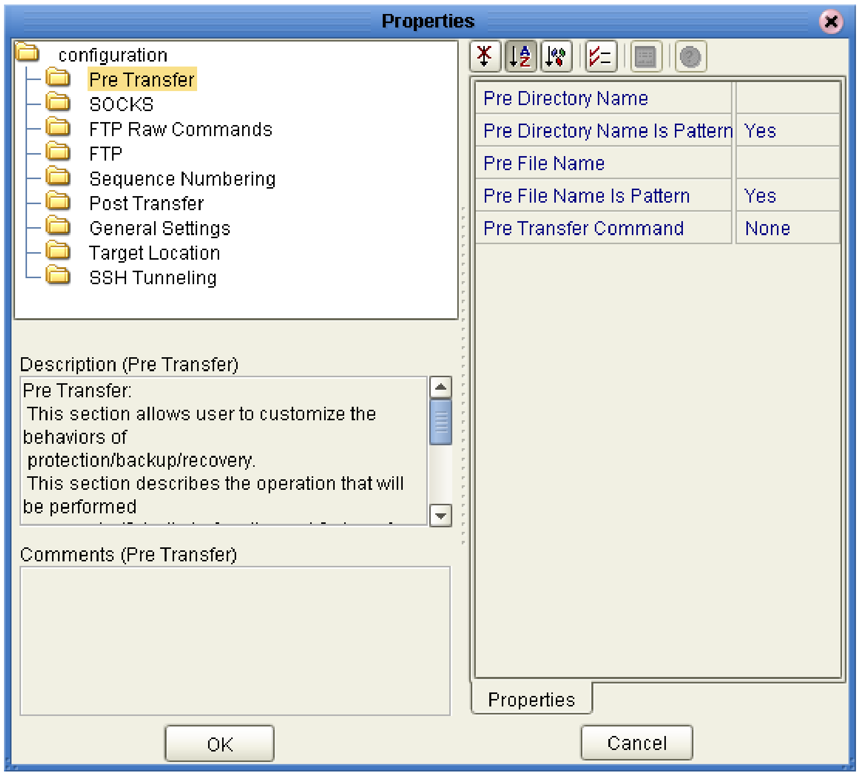
3.1.3 Using the Properties Editor

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

To modify the default eWay configuration properties

1. Open the Properties Editor for a Batch eWay.
2. From the upper-right pane of the Properties Editor, select a subdirectory of the configuration directory. The properties contained in that subdirectory are now displayed in the Properties pane of the Properties Editor. For example, clicking on the **Pre Transfer** subdirectory displays the editable parameters in the right pane, as shown in **Figure 3 on page 23**.

Figure 3 Batch eWay Properties Editor



Pre Transfer:
 This section allows user to customize the behaviors of
 protection/backup/recovery.
 This section describes the operation that will be performed

Pre Directory Name	
Pre Directory Name Is Pattern	Yes
Pre File Name	
Pre File Name Is Pattern	Yes
Pre Transfer Command	None

3 Click on any property field to make it editable. For example, click on the **class** parameter to edit the class value. 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.

3.2 Batch eWay Properties

The Batch eWay's Properties are organized as follows:

- **BatchFTP eWay Connectivity Map Properties** on page 25
- **BatchFTP eWay Environment Properties** on page 49
- **BatchFTPOverSSL eWay Connectivity Map Properties** on page 57
- **BatchFTPOverSSL eWay Environment Properties** on page 64
- **BatchSCP eWay Connectivity Map Properties** on page 70
- **BatchSCP eWay Environment Properties** on page 74
- **BatchSFTP eWay Connectivity Map Properties** on page 79
- **BatchSFTP eWay Environment Properties** on page 84
- **BatchLocalFile Connectivity Map Properties** on page 89
- **BatchLocalFile Environment Properties** on page 101
- **BatchRecord Connectivity Map Properties** on page 102
- **BatchInbound Connectivity Map Properties** on page 106

Note: *Creating customized individual OTD configuration settings can override default eWay OTD configuration settings.*

3.3 BatchFTP eWay 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:

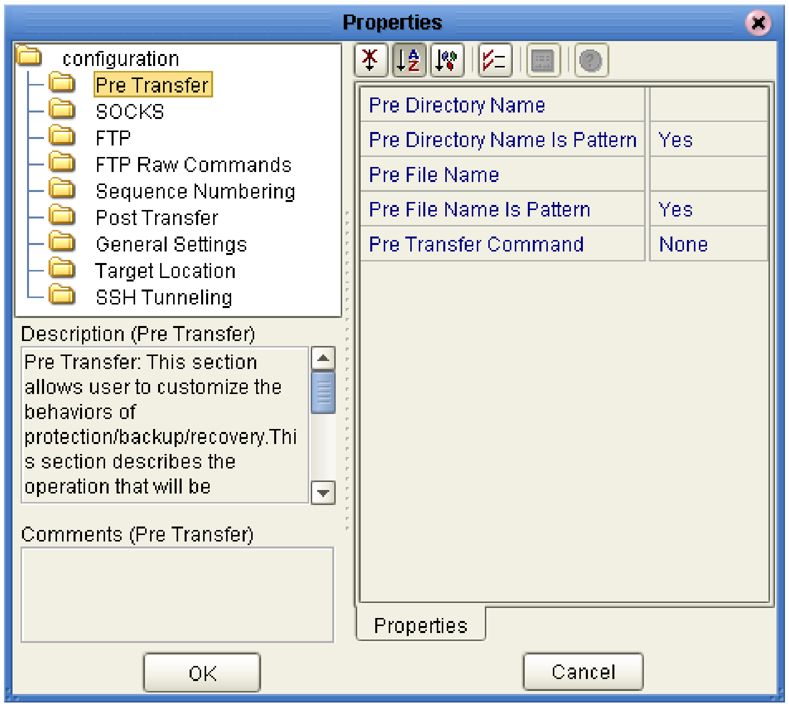
- **Pre Transfer (BatchFTP Connectivity Map)** on page 26
- **SOCKS (BatchFTP Connectivity Map)** on page 29
- **FTP (BatchFTP Connectivity Map)** on page 31
- **FTP Raw Commands (BatchFTP Connectivity Map)** on page 34
- **Sequence Numbering (BatchFTP Connectivity Map)** on page 36
- **Post Transfer (BatchFTP Connectivity Map)** on page 38
- **General Settings (BatchFTP Connectivity Map)** on page 41
- **Target Location (BatchFTP Connectivity Map)** on page 42
- **SSH Tunneling (BatchFTP Connectivity Map)** on page 45

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.*

3.3.1 Pre Transfer (BatchFTP Connectivity Map)

Pre-transfer operations are those performed before the file transfer. The Pre Transfer section contains the top level parameters displayed in Figure 4.

Figure 4 BatchFTP Properties - Pre-Transfer section



Pre Transfer: This section allows user to customize the behaviors of protection/backup/recovery.This section describes the operation that will be

Pre Directory Name	
Pre Directory Name Is Pattern	Yes
Pre File Name	
Pre File Name Is Pattern	Yes
Pre Transfer Command	None

Note: *For more information, see* **Pre/post File Transfer Commands** *on page 135.*

Pre Directory Name

Description

Specifies the directory on the external system in which a file is renamed or copied. The absolute directory name is expected.

For an outbound transfer (publishing), the directory is created if it does not already exist. This setting is only for the **Rename** or **Copy** operations of **Pre 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 **Using Special Characters** on page 150 for details on using these characters.

Required Values

A valid directory name and path location on the target system; special characters are allowed.

Pre Directory Name Is Pattern

Description

Specifies whether the pattern entered for the directory is interpreted literally (if **No**) or as a name expansion (if **Yes**), as follows:

- If you choose **No**, it is assumed that the name entered represents what you want as an exact match. No pattern matching of any kind is done.
- If you choose **Yes**, the value you enter is assumed to be a pattern for name expansion.

Required Values

Yes or **No**; the default is **Yes**.

Pre File Name

Description

Specifies the file name on the external system, to which a file is renamed or copied. The value represents the base file name instead of the full file name. This setting is only for the **Rename** or **Copy** operations of **Pre Transfer Command** parameter.

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 **"Using Special Characters" on page 150** for details on using these characters.

Required Values

A valid file name on the target system; special characters are allowed.

Pre File Name Is Pattern

Description

Specifies whether the pattern entered for the file name is interpreted literally (if **No**) or as a name expansion (if **Yes**), as follows:

- If you choose **No**, it is assumed that the name entered represents what you want as an exact match. No pattern matching of any kind is done.
- If you choose **Yes**, the value you enter is assumed to be a pattern for name expansion.

Required Values

Yes or **No**; the default is **Yes**.

Pre Transfer Command

Description

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.

Note: *The **Copy** option could slow system performance, especially if you are copying a large file.*

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.

Caution: *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.

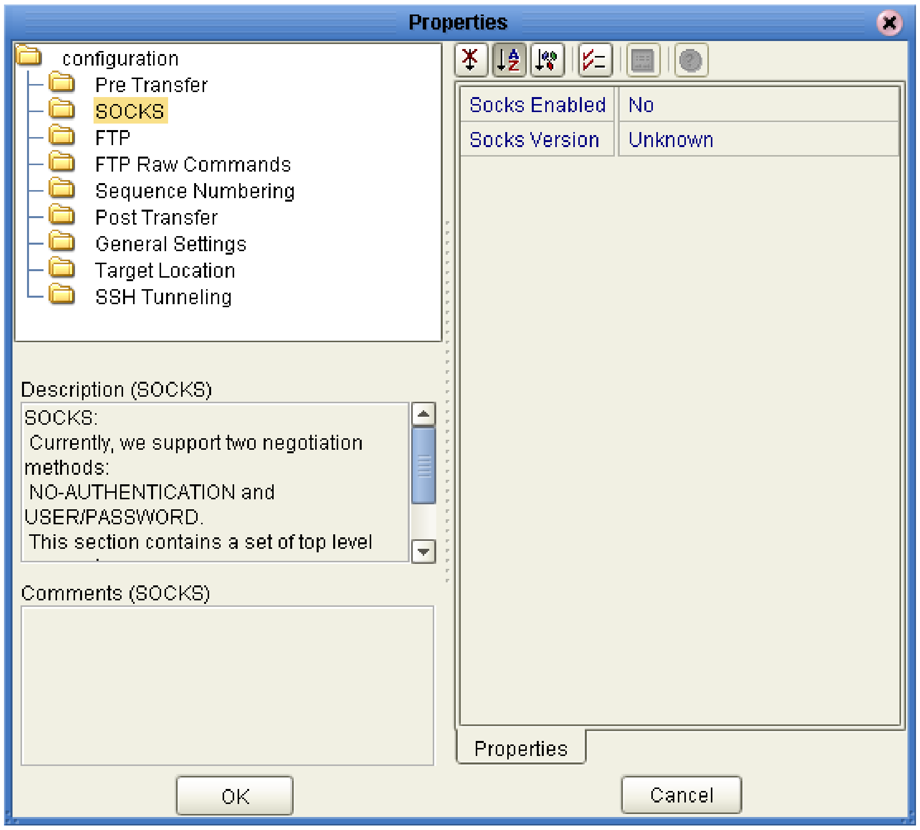
Required Values

Rename, **Copy**, or **None**. The default is **None**.

3.3.2 SOCKS (BatchFTP Connectivity Map)

The **SOCKS** section contains the top level parameters displayed in Figure 5.

Figure 5 BatchFTP Properties - SOCKS section



Description (SOCKS)

SOCKS:
Currently, we support two negotiation methods:
NO-AUTHENTICATION and USER/PASSWORD.
This section contains a set of top level

Socks Enabled	No
Socks Version	Unknown

For more information on SOCKS, see **"SOCKS" on page 157**.

Socks Enabled

Description

Specifies whether the FTP command connection goes through a SOCKS server.

If you choose **No**, the eWay 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 **"Host Name" on page 51** for more details.*

Required Values

Yes or **No**; the default is **No**.

Socks Version

Description

Specifies the SOCKS server version. If you choose **Unknown**, the eWay detects the actual version for you.

Note: *For the best performance, specify the version number, 4 or 5.*

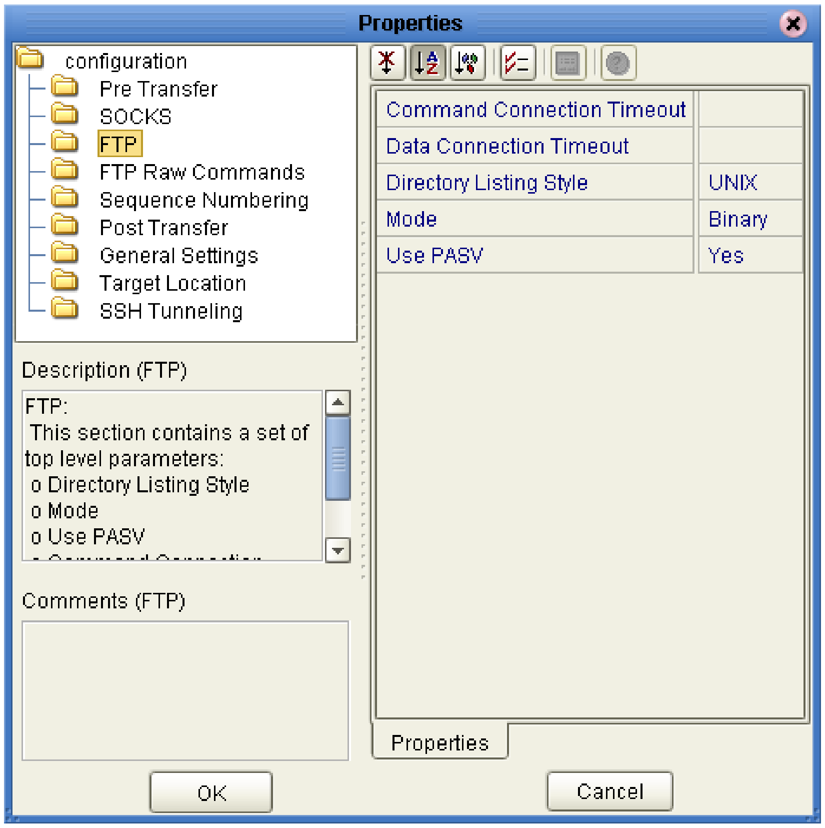
Required Values

Version **4** or **5**, or **Unknown** (the default).

3.3.3 FTP (BatchFTP Connectivity Map)

The **FTP** section contains the top level parameters displayed in Figure 6.

Figure 6 BatchFTP Properties - FTP section



FTP:
This section contains a set of top level parameters:
o Directory Listing Style
o Mode
o Use PASV

Command Connection Timeout	
Data Connection Timeout	
Directory Listing Style	UNIX
Mode	Binary
Use PASV	Yes

Command Connection Timeout

Description

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.

Required Values

An integer from **0** to **2147483647**. The default is **0**.

Data Connection Timeout

Description

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**.

Required Values

An integer from **0** to **2147483647**. The default is **45000**.

Directory Listing Style

Description

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.

Required Values

One of the following values:

- UNIX
- AS400
- AS400-UNIX
- HCLFTPD 6.0.1.3
- HCLFTPD 5.1
- 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
- User Defined
- VM/ESA
- VMS
- VOS3 PDS (Hitachi)
- VOS3 PS (Hitachi)
- VOSK (Hitachi)

Plus, you may define additional platforms and properties, if applicable. For more information, see **"Using FTP Heuristics" on page 109**.

Mode

Description

Specifies the mode used to transfer data to or from the FTP server, using the **Ascii, Binary,** or **Ebcdic** mode.

Required Values

Ascii, Binary, or **Ebcdic**; the default is **Binary**.

Note: *If you choose Ebcdic, make sure that:*

- *Your FTP server supports the EBCDIC mode.*
- *You are processing EBCDIC data.*

Use PASV

Description

Allows you to prompt the eWay 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

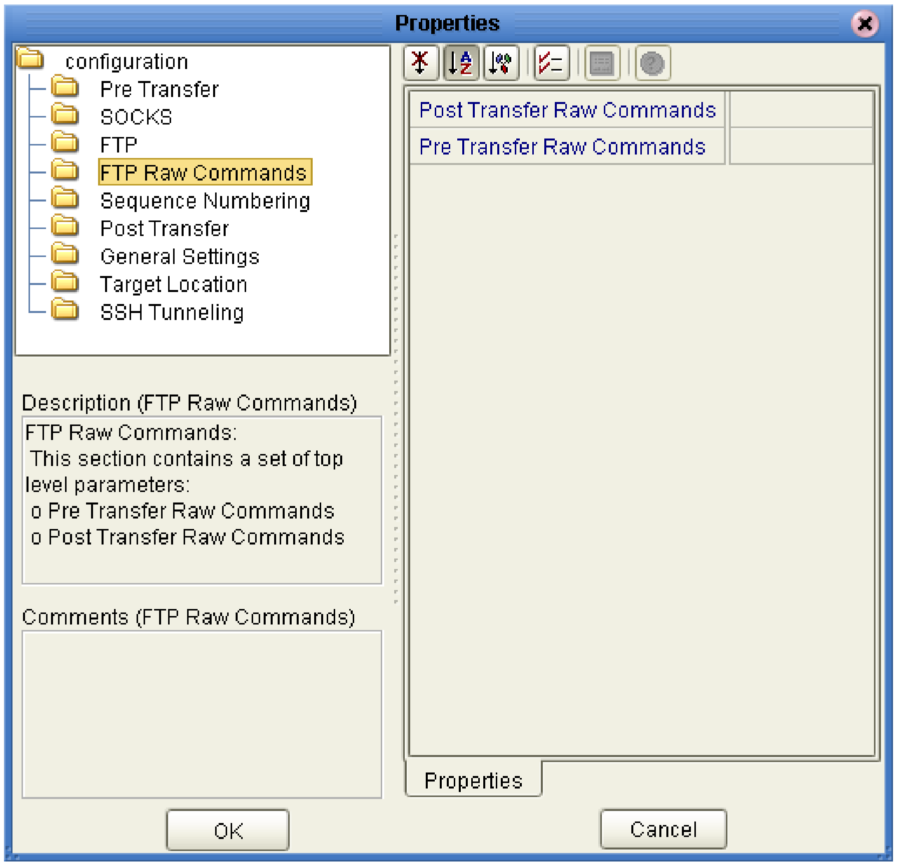
Required Values

Yes or **No**; the default is **Yes**

3.3.4 FTP Raw Commands (BatchFTP Connectivity Map)

FTP raw commands are commands that are sent directly to the FTP server. The **FTP Raw Commands** section contains the top level parameters displayed in Figure 7.

Figure 7 BatchFTP Properties - FTP Raw Commands section



Description (FTP Raw Commands)

FTP Raw Commands:
This section contains a set of top level parameters:
o Pre Transfer Raw Commands
o Post Transfer Raw Commands

Post Transfer Raw Commands

Description

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, for 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.

Caution: *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.*

Required Values

One or more valid FTP raw commands.

Note: *These commands are sent to the FTP server directly and are not interpreted by the eWay in any way.*

Pre Transfer Raw Commands

Description

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, for 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.

Required Values

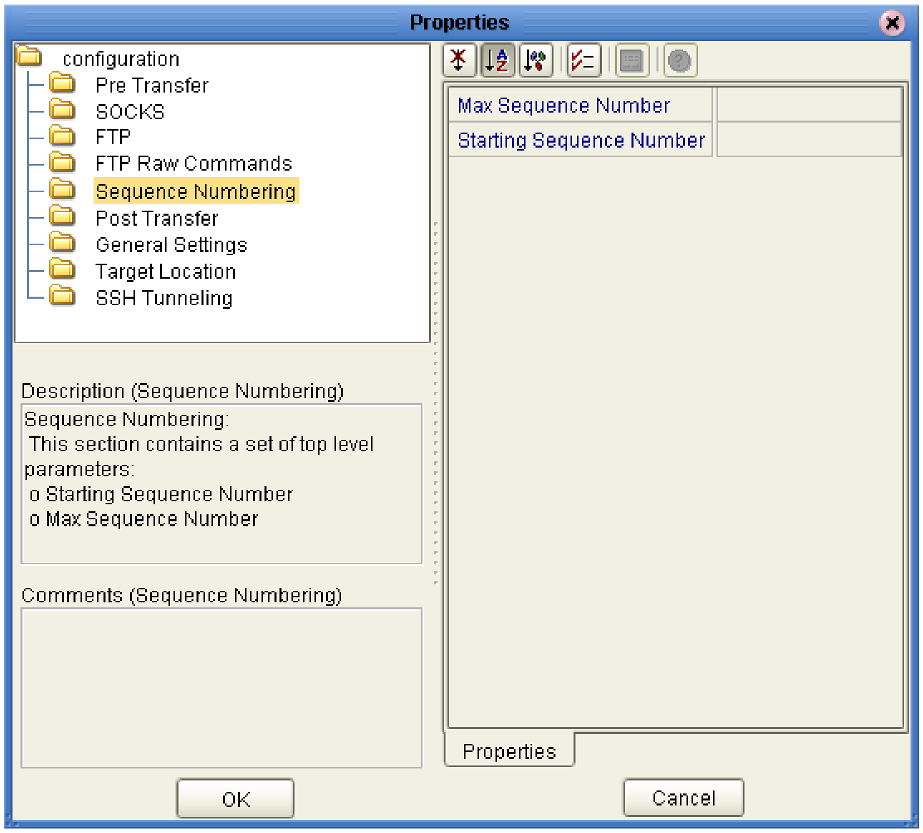
One or more valid FTP raw commands.

Note: *These commands are sent to the FTP server directly and are not interpreted by the eWay in any way.*

3.3.5 Sequence Numbering (BatchFTP Connectivity Map)

The **Sequence Numbering** section contains the top level parameters displayed in Figure 8.

Figure 8 BatchFTP Properties - Sequence Numbering section



Max Sequence Number	
Starting Sequence Number	

Description (Sequence Numbering)

Sequence Numbering:
This section contains a set of top level parameters:
o Starting Sequence Number
o Max Sequence Number

Comments (Sequence Numbering)

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

Max Sequence Number

Description

Use this parameter when you have set up the target directory or file name to contain a sequence number. It tells the eWay 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 %#.

Required Values

An integer from **1** to **2147483647**. The value of **Max Sequence Number** *must* be greater than that of **Starting Sequence Number**. The default value is **999999**.

Starting Sequence Number

Description

Use this parameter when you have set up the target directory or file name to contain a sequence number. It tells the eWay 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.

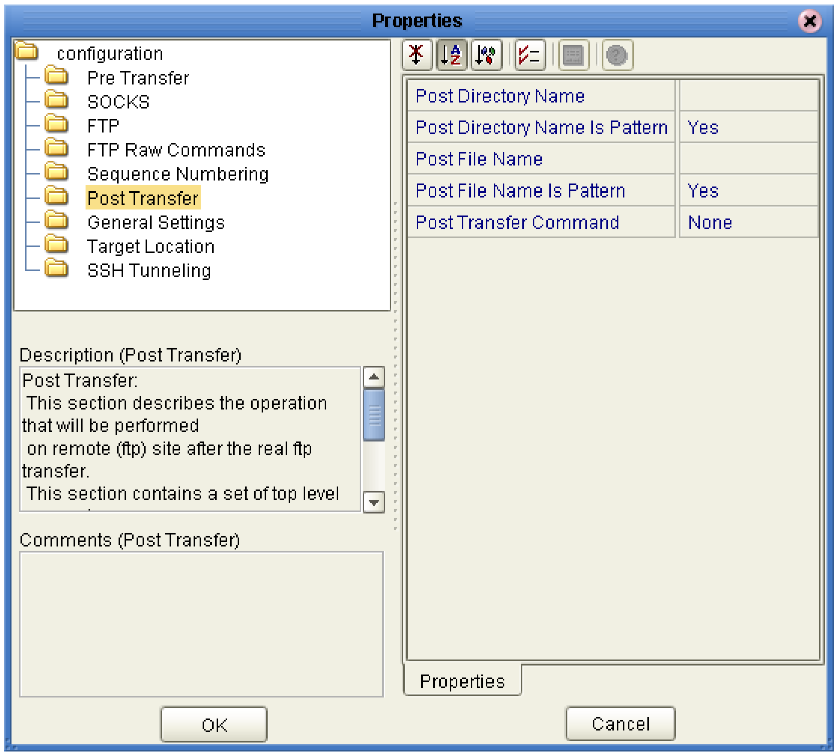
Required Values

An integer from **0** to **2147483647**. The value of the **Starting Sequence Number** *must* be less than the **Max Sequence Number** value. The default value is **1**.

3.3.6 Post Transfer (BatchFTP Connectivity Map)

Post-transfer operations are those performed after the file transfer. The **Post Transfer** section contains the top level parameters displayed in Figure 9.

Figure 9 BatchFTP Properties - Post-Transfer section



Description (Post Transfer)
Post Transfer:
This section describes the operation that will be performed
on remote (ftp) site after the real ftp transfer.
This section contains a set of top level

Post Directory Name	
Post Directory Name Is Pattern	Yes
Post File Name	
Post File Name Is Pattern	Yes
Post Transfer Command	None

Note: *For more information on this feature, see* **"Pre/post File Transfer Commands" on page 135***.*

Post Directory Name

Description

Specifies the directory on the external system in which a file is renamed. The absolute directory name is expected.

For an outbound transfer (publishing), 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 **"Using Special Characters" on page 150** for details on using these characters.

Required Values

A valid directory name and path location on the target system; special characters are allowed.

Post Directory Name Is Pattern

Description

Specifies whether the pattern entered for the directory is interpreted literally (if **No**) or as a name expansion (if **Yes**), as follows:

- If you choose **No**, it is assumed that the name entered represents what you want as an exact match. No pattern matching of any kind is done.
- If you choose **Yes**, the value you enter is assumed to be a pattern for name expansion.

Required Values

Yes or **No**; the default is **Yes**.

Post File Name

Description

Specifies the file name on an external system to which a file is renamed. The value represents the base file name instead of the full file 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. The expansion of any special characters is carried out each time this parameter is used. See **"Using Special Characters" on page 150** for details on using these characters.

Required Values

A valid file name on the target system; special characters are allowed.

Post File Name Is Pattern

Description

Specifies whether the pattern entered for the file name is interpreted literally (if **No**) or as a name expansion (if **Yes**), as follows:

- If you choose **No**, it is assumed that the name entered represents what you want as an exact match. No pattern matching of any kind is done.
- If you choose **Yes**, the value you enter is assumed to be a pattern for name expansion.

Required Values

Yes or **No**; the default is **Yes**.

Post Transfer Command

Description

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.

Caution: *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.

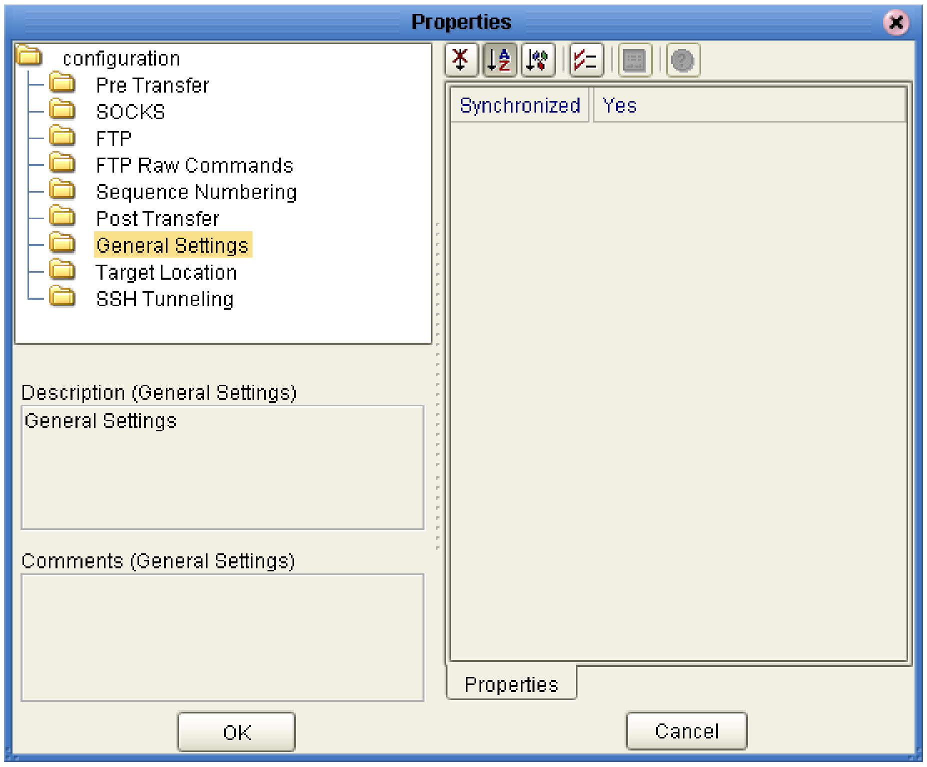
Required Values

Rename, **Delete**, or **None**; the default is **None**.

3.3.7 General Settings (BatchFTP Connectivity Map)

The **General Settings** section contains the top level parameters displayed in Figure 10.

Figure 10 BatchFTP Properties - General Settings section



Synchronized	Yes

Synchronized

Description

Specifically applies to legacy Batch eWay Projects. Provides backward compatibility to allow Projects that were created using the Batch eWay version 5.0.7 or earlier to be imported and deployed without a change in the eWays behavior. The selections are:

- **Yes**: Provides backward compatibility for legacy (pre-5.0.8 Batch eWay) Projects. The eWay preserves the synchronized behavior.
- **No**: For use with new Batch eWay Projects. The eWay creates multiple instances of the Collaboration that run in Parallel.

We recommend that all OTD instances used in a Project maintain that same value for this property.

Required Values

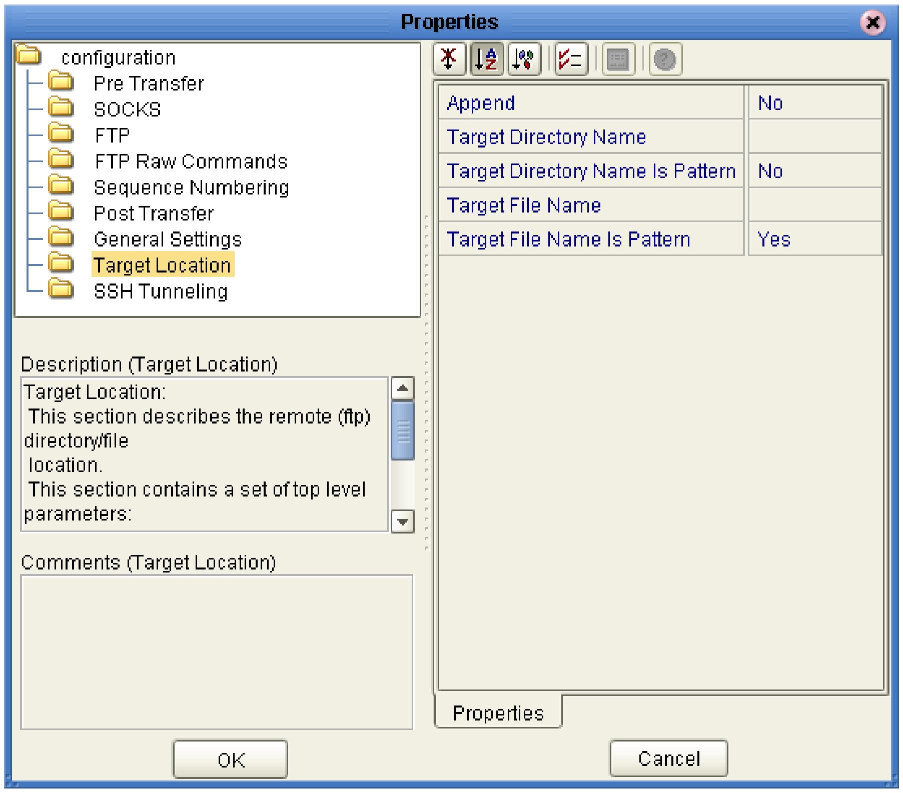
Yes or **No.** The default setting is **Yes**, to provide backward compatibility for existing Projects.

Note: *Synchronized must be set to "Yes" to use Sequence Numbering.*

3.3.8 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 contains the top level parameters displayed in Figure 11.

Figure 11 BatchFTP Properties - Target Location section



Description (Target Location)

Target Location:
This section describes the remote (ftp) directory/file
location.
This section contains a set of top level parameters:

Append	No
Target Directory Name	
Target Directory Name Is Pattern	No
Target File Name	
Target File Name Is Pattern	Yes

Append

Description

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 eWay 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.

Required Values

Yes or **No**; the default is **No**.

Target Directory Name

Description

Specifies the directory on the external system from which files are retrieved or where they are sent. The absolute directory name is preferred, otherwise, this path is relative to the home directory where you are when you log on to the FTP server.

For outbound FTP operations (publishing), the directory is created if it does not already exist.

Required Values

A valid directory name/path location on the target external system.

Target Directory Name Is Pattern

Description

Specifies whether the pattern entered for the directory is interpreted literally (if **No**) or as a regular expression or name expansion (if **Yes**), as follows:

- If you choose **No**, it is assumed that the name entered represents what you want as an exact match. No pattern matching of any kind is done.
- If you choose **Yes**, the value you enter is assumed to be a regular expression (when doing inbound) or name expansion (when doing an outbound FTP to the file system).

Note: *Target directory names are resolved relative to your home directory unless an absolute path is given.*

Required Values

Yes or **No**; the default is **No**.

Target File Name

Description

Specifies the name of the remote FTP file to be retrieved or sent. This parameter can specify the exact file name or a regular-expression pattern. For outbound data (publishing), the file is created if it does not already exist. This parameter represents the base file name instead of the full file name.

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

Required Values

A valid file name or a regular expression or name expansion file name.

Target File Name Is Pattern

Description

Specifies whether the pattern entered for the file name is interpreted literally (if **No**) or as a regular expression or name expansion (if **Yes**), as follows:

- If you choose **No**, it is assumed that the name entered represents what you want as an exact match. No pattern matching of any kind is done.
- If you choose **Yes**, the value you enter is assumed to be a regular expression (when doing inbound) or name expansion (when doing an outbound FTP to the file system).

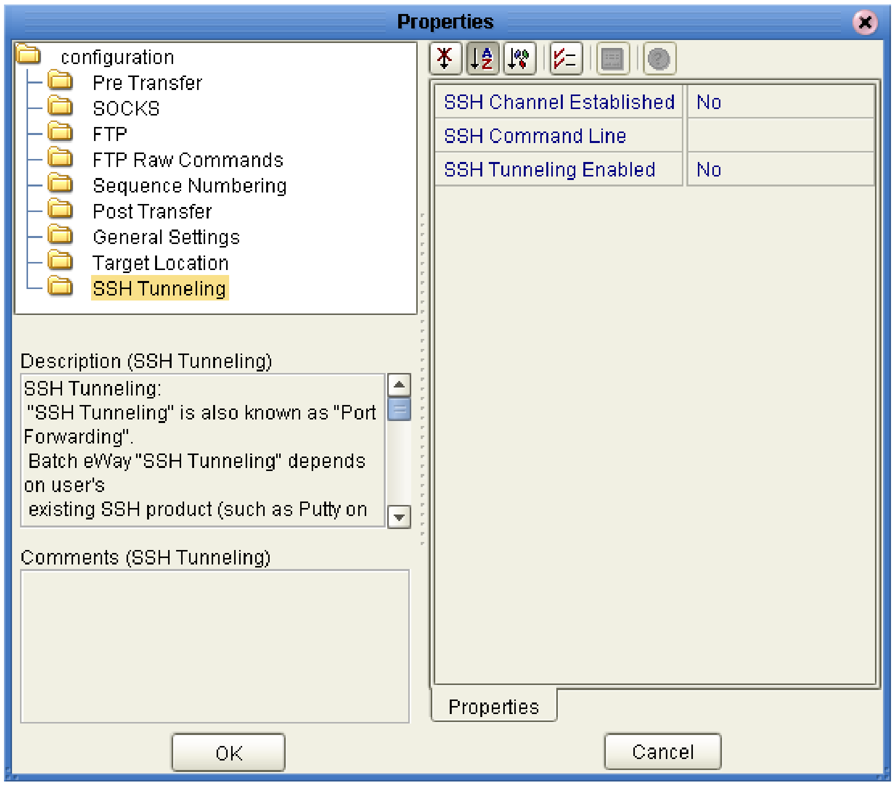
Required Values

Yes or **No**; the default is **Yes**.

3.3.9 SSH Tunneling (BatchFTP Connectivity Map)

This section provides information for configuring the **SSH Tunneling** properties (accessed from the Connectivity Map). 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 contains the top level parameters displayed in Figure 12.

Figure 12 BatchFTP Properties - TSSH Tunneling section



SSH Channel Established	No
SSH Command Line	
SSH Tunneling Enabled	No

Description (SSH Tunneling)

SSH Tunneling:
"SSH Tunneling" is also known as "Port Forwarding".
Batch eWay "SSH Tunneling" depends on user's
existing SSH product (such as Putty on

Comments (SSH Tunneling)

Additional SSH-supporting Software

The eWay'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 159

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 eWay Connection.

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

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

Under the eWay'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 eWay connects to the FTP server **apple:21** through an SSH tunnel. For more information on SSH tunneling, see **"SSH Tunneling Support" on page 159**.

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

SSH Channel Established

Description

Specifies whether the eWay needs to launch an SSH subprocess.

Selecting **No** indicates that the SSH channel has not yet been established. The eWay 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 eWay, and the eWay does not need to establish it. For example, you could have issued a command outside of eGate, or you could know that another Batch eWay instance has already established the channel by the time this eWay runs.

If you select **Yes**, you must set the following parameters:

- **SSH Listen Host** (Environment property)
- **SSH Listen Port** (Environment property)

Required Values

Yes or **No**; the default is **No**.

SSH Command Line

Description

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 that it:

- Contains all the necessary arguments
- Is correct in syntax
- Is compliant with your SSH-environment

To verify these requirements, test this command line manually outside of eGate 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 eWay'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 the previous 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 eWay instances located on different eGate 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 also can specify port forwarding in your SSH configuration file.*

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
```

Required Values

A valid SSH command line.

SSH Tunneling Enabled

Description

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.*

Required Values

Yes or **No**; the default is **No**.

3.4 BatchFTP eWay 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 49
- **FTP (BatchFTP Environment)** on page 51
- **General Settings (BatchFTP Environment)** on page 53
- **SSH Tunneling (BatchFTP Environment)** on page 54

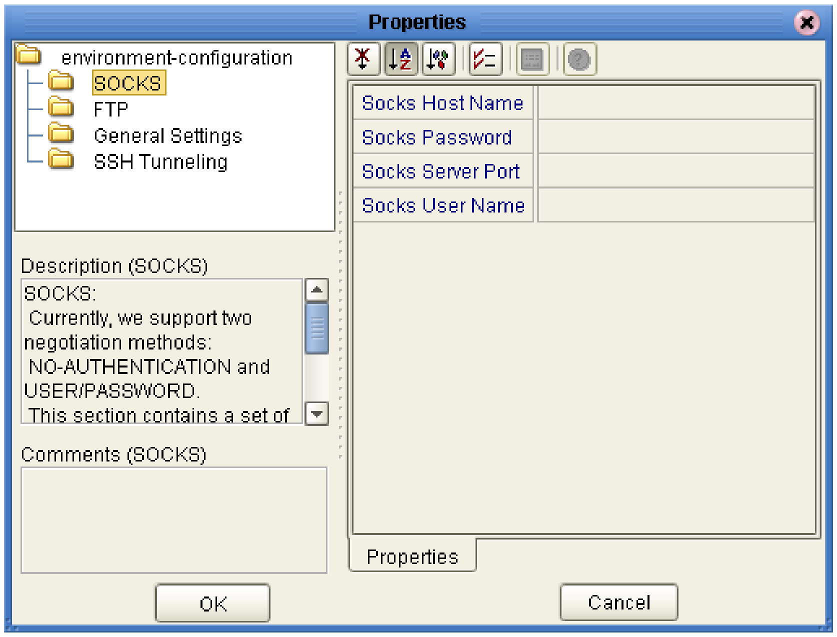
Caution: *Several of these configuration options allow you to use regular expressions. This advanced feature is useful but must be used carefully. An improperly formed regular expression can cause the creation of undesired data or even the loss of data. You must have a clear understanding of regular-expression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

3.4.1 SOCKS (BatchFTP Environment)

This section provides information for configuring the **SOCKS** properties (accessed from the Environment Explorer). The BatchFTP eWay supports the negotiation methods, **No-authentication** and **User/password**. The **SOCKS** section contains the top level parameters displayed in Figure 13.

For more information on SOCKS, see **"SOCKS" on page 157**.

Figure 13 BatchFTP Properties - SOCKS section



SOCKS:
Currently, we support two negotiation methods:
NO-AUTHENTICATION and USER/PASSWORD.
This section contains a set of

Socks Host Name	
Socks Password	
Socks Server Port	
Socks User Name	

Socks Host Name

Description

Specifies the SOCKS server (host) name. If you are communicating with a SOCKS server enter the SOCKS server name in this parameter.

Required Values

The name of the SOCKS server.

Socks Password

Description

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)**.*

Required Values

A valid SOCKS5 password.

Socks Server Port

Description

Specifies the port number to use on the SOCKS server, when connecting to it.

Required Values

An integer from **1** to **65,535**; the default is **1080**.

Socks User Name

Description

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.

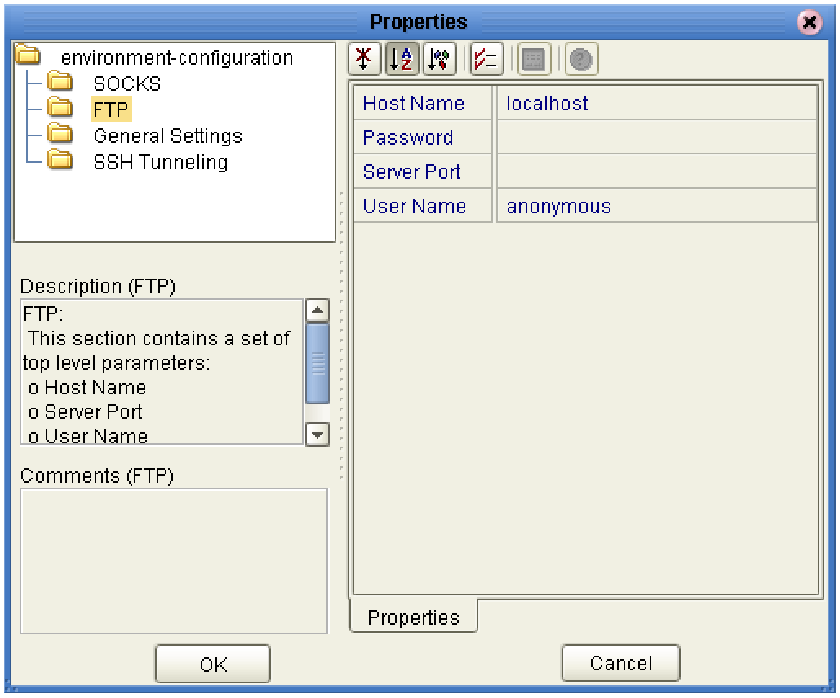
Required Values

A valid SOCKS5 user name.

3.4.2 FTP (BatchFTP Environment)

This section provides information for configuring the **FTP** properties (accessed from the Environment Explorer). The **FTP** section contains the top level parameters displayed in Figure 14.

Figure 14 BatchFTP Properties - FTP section



Description (FTP)

FTP:
This section contains a set of top level parameters:
o Host Name
o Server Port
o User Name

Comments (FTP)

Host Name	localhost
Password	
Server Port	
User Name	anonymous

Host Name

Description

Specifies the name of the external system that the eWay 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 45** 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.

Required Values

The host name.

Password

Description

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()**.

Required Values

The password.

Server Port

Description

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 45** for details.

Required Values

The server port number.

User Name

Description

Specifies the user name used to log onto the external system, when required.

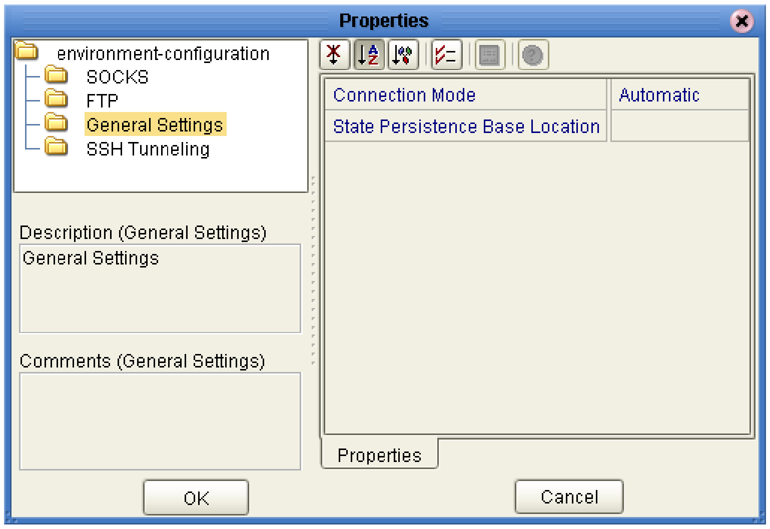
Required Values

A user name that provides access to the external system.

3.4.3 General Settings (BatchFTP Environment)

This section provides information for configuring the **General Settings** properties (accessed from the Environment Explorer). The **General Settings** section contains the top level parameters displayed in Figure 15.

Figure 15 BatchFTP Properties - General Settings section



Description (General Settings)
General Settings

Comments (General Settings)

Connection Mode	Automatic
State Persistence Base Location	

Connection Mode

Description

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).

Required Values

Select **Automatic** or **Manual**. The default is **Automatic**.

State Persistence Base Location

Description

Specifies a directory where the per external connection state file resides, that is, a state file to store state information that needs to be persisted. This is an optional property. If this value is left blank, a default file is used.

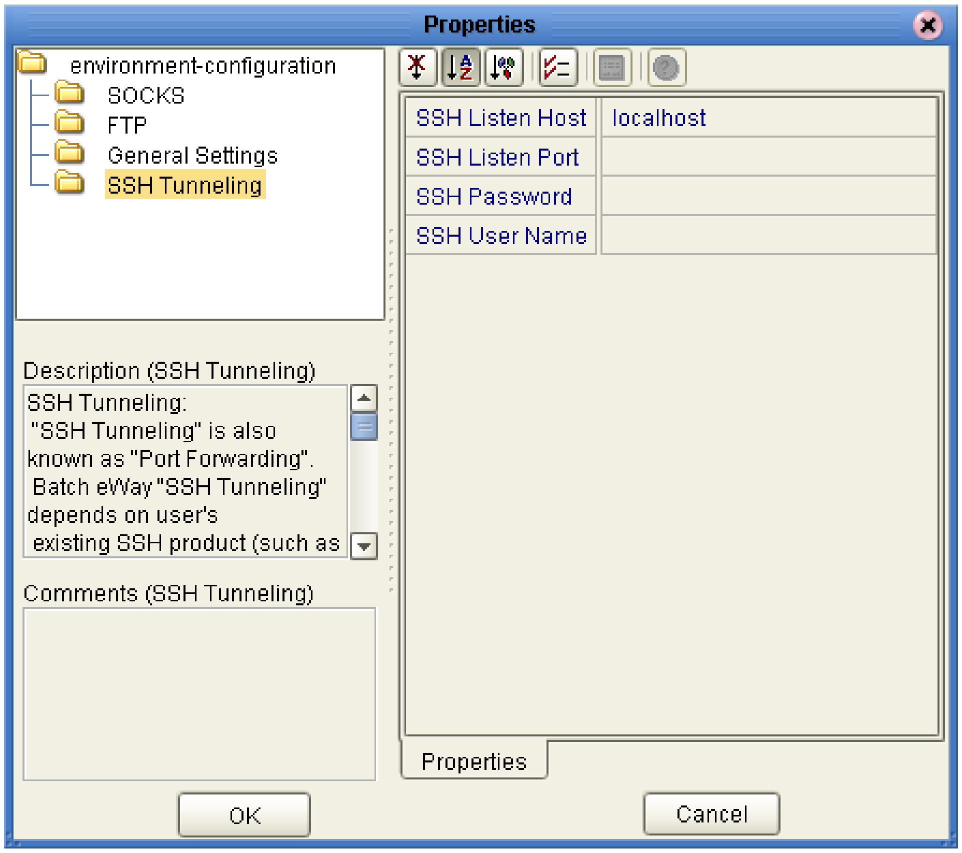
Required Values

A local directory.

3.4.4 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 contains the top level parameters displayed in Figure 16.

Figure 16 BatchFTP Properties - SSH Tunneling section



Description (SSH Tunneling)

SSH Tunneling:
"SSH Tunneling" is also known as "Port Forwarding". Batch eWay "SSH Tunneling" depends on user's existing SSH product (such as

SSH Listen Host	localhost
SSH Listen Port	
SSH Password	
SSH User Name	

SSH Listen Host

Description

Specifies the name of the host where the SSH support software runs, and 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. 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 this 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
```

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
```

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 must be configured to allow connections to other hosts (for some SSH applications, you can use an option **-g**).

Regardless, the transport between the SSH listen host and the FTP server is still secure.

Required Values

The SSH listen host name; the default is **localhost.**

SSH Listen Port

Description

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 eGate system. The corresponding SSH command line uses the following model:

```
ssh -L ListenPort:FtpServerHost:FtpServerPort -o BatchMode=yes
    SSHServer Required Values
```

Required Values

An integer from **1** to **65535**; the default is **4567**.

SSH Password

Description

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**.

Required Values

The SSH password.

SSH User Name

Description

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 eWay (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.

Required Values

The SSH user name.

3.5 BatchFTPOverSSL eWay Connectivity Map Properties

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

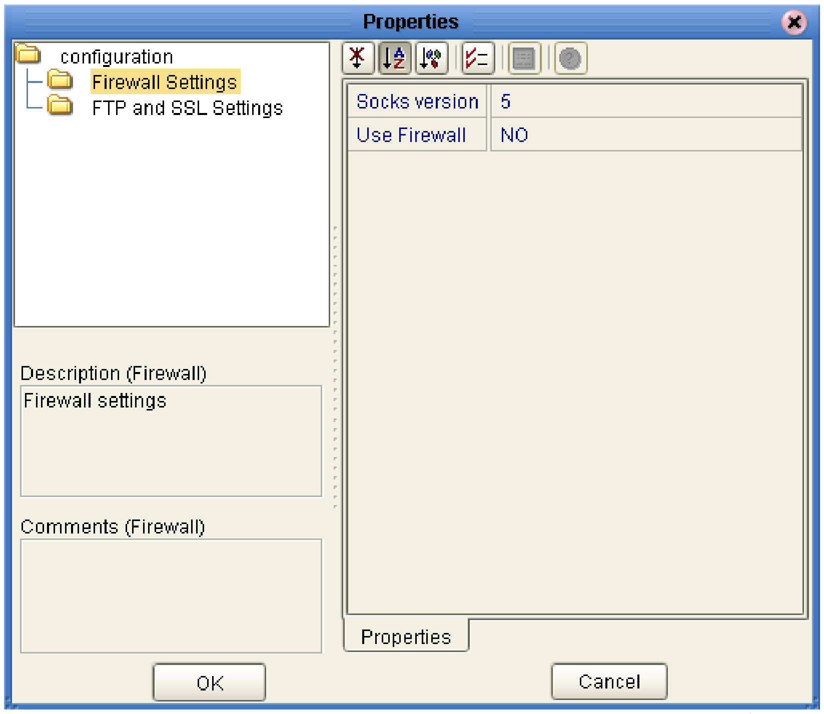
The BatchFTPOverSSL eWay Connectivity Map properties include the following sections:

- **Firewall Settings (BatchFTPOverSSL Connectivity Map)** on page 57
- **FTP and SSL Settings (BatchFTPOverSSL Connectivity Map)** on page 59

3.5.1 Firewall Settings (BatchFTPOverSSL Connectivity Map)

The **Firewall Settings** section contains the top level parameters displayed in Figure 17.

Figure 17 BatchFTPOverSSL Properties - Firewall Settings section



Description (Firewall)
Firewall settings

Comments (Firewall)

Socks version	5
Use Firewall	NO

Socks version

Description

Specifies the SOCKS version of the firewall. The supported options are **4** for SOCKS version 4, or **5** for SOCKS version **5**

Required Values

Select **4** for SOCKS version 4, or **5** for SOCKS version **5**.

Use Firewall

Description

Specifies whether you are using a firewall.

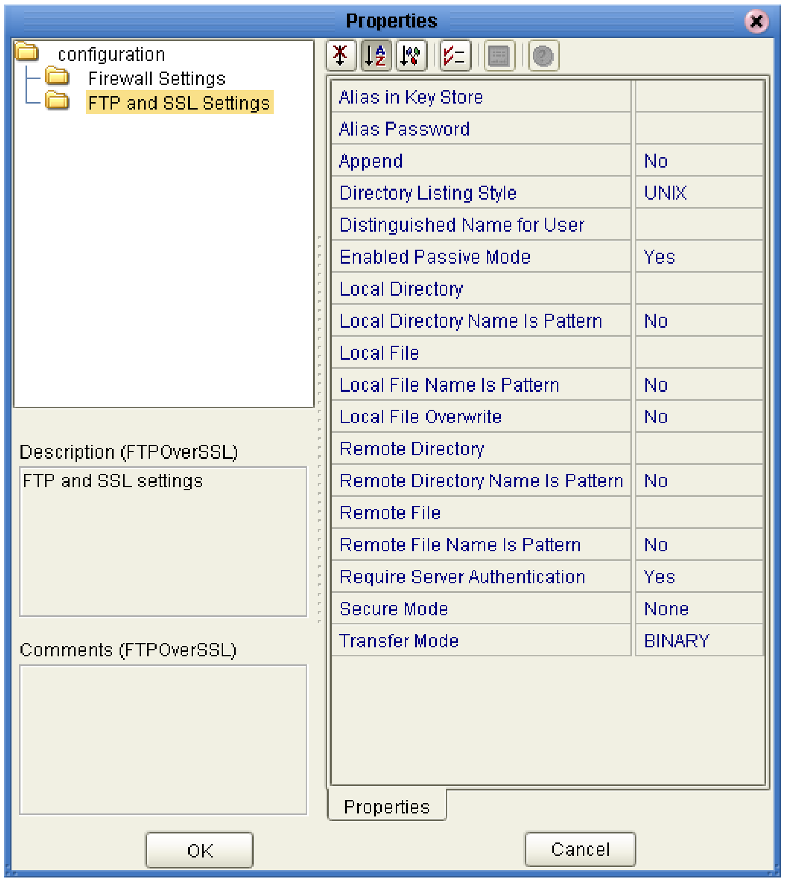
Required Values

Yes or **No**. **Yes** indicates that you are using a firewall. The configured default is **No**.

3.5.2 FTP and SSL Settings (BatchFTPOverSSL Connectivity Map)

The **FTP and SSL Settings** section contains the top level parameters displayed in Figure 18.

Figure 18 BatchFTPOverSSL Properties - FTP and SSL Settings section



FTP and SSL settings

Alias in Key Store	
Alias Password	
Append	No
Directory Listing Style	UNIX
Distinguished Name for User	
Enabled Passive Mode	Yes
Local Directory	
Local Directory Name Is Pattern	No
Local File	
Local File Name Is Pattern	No
Local File Overwrite	No
Remote Directory	
Remote Directory Name Is Pattern	No
Remote File	
Remote File Name Is Pattern	No
Require Server Authentication	Yes
Secure Mode	None
Transfer Mode	BINARY

Alias in Key Store

Description

The **Alias in Key Store** parameter is not used in this release, but is retained for future enhancement.

Alias Password

Description

Specifies the password that protects the key pair entry in the keystore, identified by the alias.

Required Values

The alias password.

Append

Description

Specifies whether new data, transferred to a remote server, is appended to data that was previously transferred. default is No.

Required Values

Yes or **No**. **Yes** indicates that data will be appended. The configured default is **No**.

Directory Listing Style

Description

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.

Required Values

Select **UNIX**, **NT**, or **MVS**. The configured default is **UNIX**.

Distinguished Name for User

Description

The **Distinguished Name for User** parameter is not used in this release, but is retained for future enhancement.

Enabled Passive Mode

Description

Specifies whether FTP passive mode is enabled.

Required Values

Select **Yes** or **No**. **Yes** indicates that FTP passive mode is enabled. The configured default is **Yes**.

Local Directory

Description

Specifies the local directory for files that are sent to or received from a remote system.

Required Values

The local directory name and path.

Local Directory Name is Pattern

Description

Specifies the meaning of the **Local Directory Name** property as follows:

- **Yes**: Indicates that the **Local Directory Name** represents a pattern to be used as a regular expression for pattern matching.
- **No**: Indicates that the **Local Directory Name** represents the exact directory name to be used. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **No**.

Local File

Description

Specifies the local file in the local directory to be sent to the remote server, or received data from the remote.

Required Values

The local file name.

Local File Name is Pattern

Description

Specifies the meaning of the **Local File Name** property as follows:

- **Yes**: Indicates that the **Local File Name** represents a pattern to be used as a regular expression for pattern matching.
- **No**: Indicates that the **Local File Name** represents the exact file name to be used. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **No**.

Local File Overwrite

Description

Specifies whether new data downloaded from the remote will overwrite existing data.

Required Values

Yes or **No**. **Yes** indicates that new data will overwrite existing data. The configured default is **No**.

Remote Directory

Description

Specifies the virtual directory on the FTP server where data is published, or from which data is received. The accessibility of the directory usually depends on the login user.

Required Values

The name of the remote directory.

Remote Directory Name is Pattern

Description

Specifies the meaning of the **Remote Directory Name** property as follows:

- **Yes**: Indicates that the **Remote Directory Name** represents a pattern to be used as a regular expression for pattern matching.
- **No**: Indicates that the **Remote Directory Name** represents the exact directory name to be used. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **No**.

Remote File

Description

Specifies the file name on the remote server that either receives sent data, or holds data to be retrieved.

Required Values

The name of the remote file.

Remote File Name is Pattern

Description

Specifies the meaning of the **Remote File Name** property as follows:

- **Yes**: Indicates that the **Remote File Name** represents a pattern to be used as a regular expression for pattern matching.
- **No**: Indicates that the **Remote File Name** represents the exact file name to be used. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **No**.

Required Server Authentication

Description

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.

Required Values

Yes or **No**. The configured default is **Yes**.

Secure Mode

Description

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.

Required Values

Select **None**, **Implicit SSL**, or **Explicit SSL**. **None** is the configured default.

Transfer Mode

Description

Specifies whether the transfer is binary code or ASCII text.

Required Values

Select **BINARY** or **ASCII**. The configured default is **BINARY**.

3.6 BatchFTPOverSSL eWay Environment Properties

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

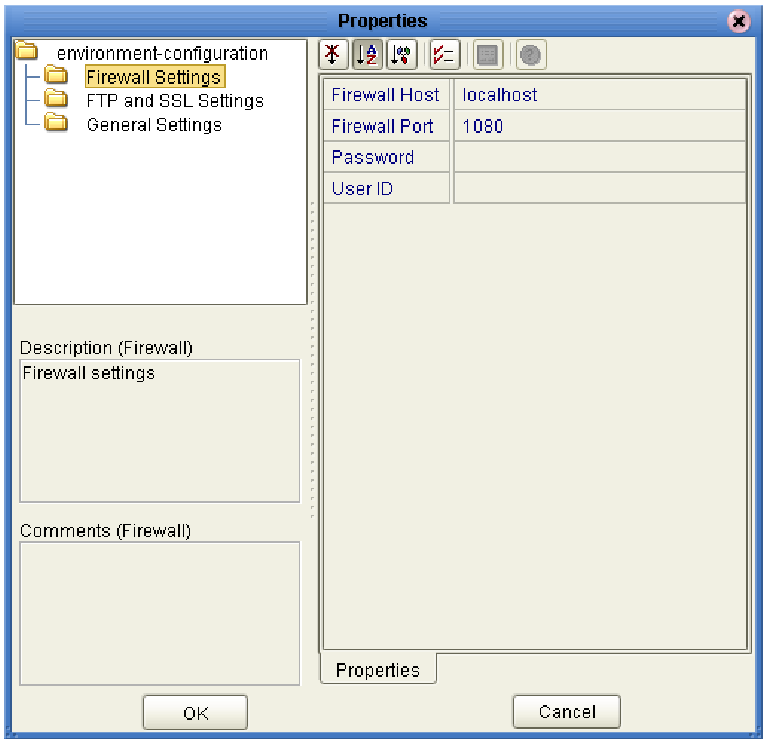
The BatchFTPOverSSL eWay Environment properties include the following sections:

- **Firewall Settings (BatchFTPOverSSL Environment)** on page 64
- **FTP and SSL Settings (BatchFTPOverSSL Environment)** on page 66
- **General Settings (BatchFTPOverSSL Environment)** on page 68

3.6.1 Firewall Settings (BatchFTPOverSSL Environment)

The **Firewall Settings** section contains the top level parameters displayed in Figure 17.

Figure 19 BatchFTPOverSSL Properties - Firewall Settings section



Firewall settings

Firewall Host	localhost
Firewall Port	1080
Password	
User ID	

Firewall Host

Description

Specifies the proxy server host name or IP.

Required Values

The proxy server host name or IP address.

Firewall Port

Description

Specifies the proxy server port.

Required Values

The proxy server port number.

Password

Description

Specifies the password for the user login.

Required Values

The user password.

User ID

Description

Specifies the user login on the proxy server.

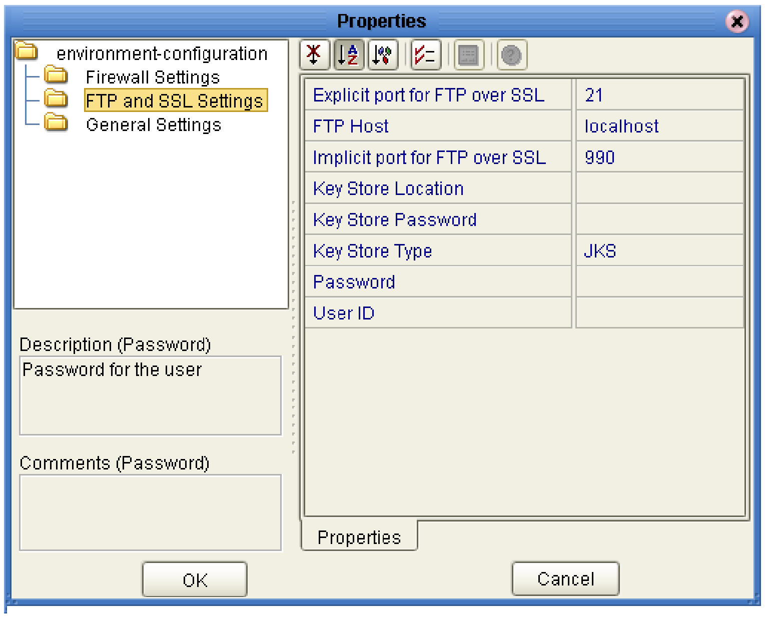
Required Values

User login ID.

3.6.2 FTP and SSL Settings (BatchFTPOverSSL Environment)

The **FTP and SSL Settings** section contains the top level parameters displayed in Figure 20.

Figure 20 BatchFTPOverSSL Properties - FTP and SSL Settings section



Explicit port for FTP over SSL	21
FTP Host	localhost
Implicit port for FTP over SSL	990
Key Store Location	
Key Store Password	
Key Store Type	JKS
Password	
User ID	

Description (Password)
Password for the user

Comments (Password)

Explicit port for FTP over SSL

Description

Specifies the FTP port for explicit SSL. default is 21 (data port 20).

Required Values

The FTP port number for explicit SSL. The configured default is **21** (data port 20).

FTP Host

Description

Specifies the FTP server host name or IP address.

Required Values

The FTP server host name or IP address. The configured default is **localhost**.

Implicit port for FTP over SSL

Description

Specifies the FTP port for implicit SSL.

Required Values

The FTP port for implicit SSL The configured default is **990** (data port 989).

Key Store Location

Description

Specifies the path to the keystore that contains the trusted CA certificates required for server authentication.

Required Values

The fully qualified path to the keystore file.

Key Store Password

Description

Specifies the password to access the keystore file.

Required Values

The password for the keystore.

Key Store Type

Description

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".*

Required Values

Select **JKS**. **JKS** is currently the only supported keystore type, and the configured default.

Password

Description

Specifies the password for the FTP server user login.

Required Values

The password for the FTP server user login.

User ID

Description

Specifies the user login for FTP server.

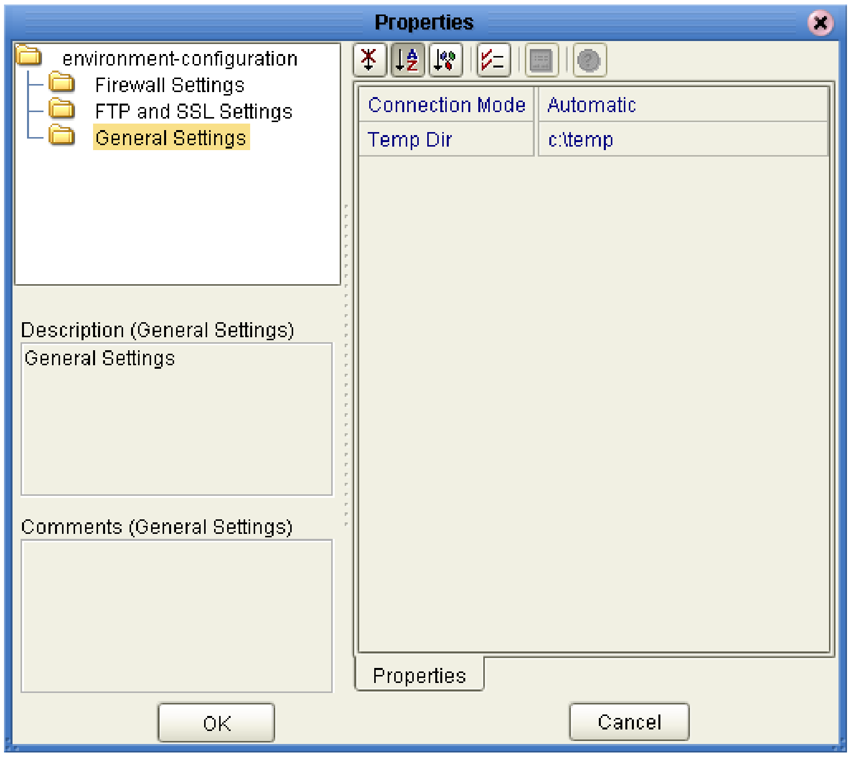
Required Values

The user login name for FTP server.

3.6.3 General Settings (BatchFTPOverSSL Environment)

The **Firewall Settings** section contains the top level parameters displayed in Figure 17.

Figure 21 BatchFTPOverSSL Properties - General Settings section



Connection Mode	Automatic
Temp Dir	c:\temp

Connection Mode

Description

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).

Required Values

Select **Automatic** or **Manual**. The default is **Automatic**.

Temp Dir

Description

Specifies the working directory for holding a payload processed by the OTD. This value is required if the Collaboration sends a payload to a remote server, or receives remote data into a payload, using this OTD.

Required Values

The name and location of a designated temporary directory. The configured default is **c:\temp**.

3.7 BatchSCP eWay Connectivity Map Properties

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

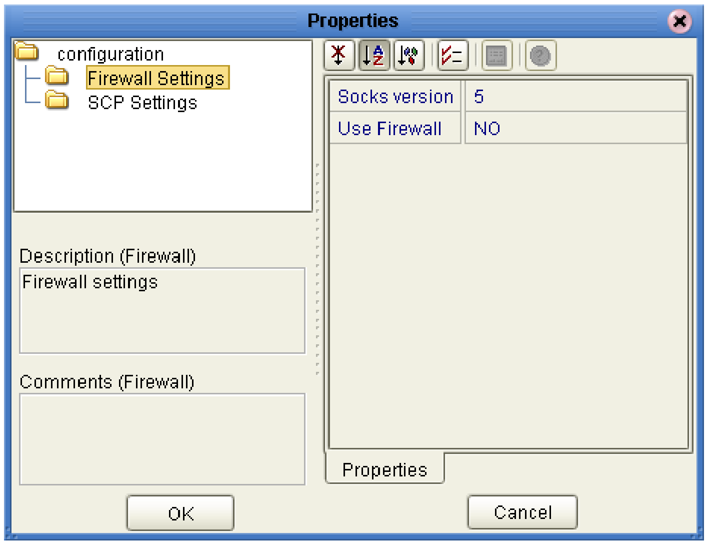
The BatchSCP eWay Connectivity Map properties include the following sections:

- **Firewall Settings (BatchSCP Connectivity Map)** on page 70
- **SCP Settings (BatchSCP Connectivity Map)** on page 71

3.7.1 Firewall Settings (BatchSCP Connectivity Map)

The **Firewall Settings** section contains the top level parameters displayed in Figure 22.

Figure 22 BatchSCP Properties - Firewall Settings section



Description (Firewall)
Firewall settings

Comments (Firewall)

Socks version	5
Use Firewall	NO

Socks version

Description

Specifies the SOCKS version of the firewall. The supported options are **4** for SOCKS version 4, or **5** for SOCKS version **5**

Required Values

Select **4** for SOCKS version 4, or **5** for SOCKS version **5**.

Use Firewall

Description

Specifies whether you are using a firewall.

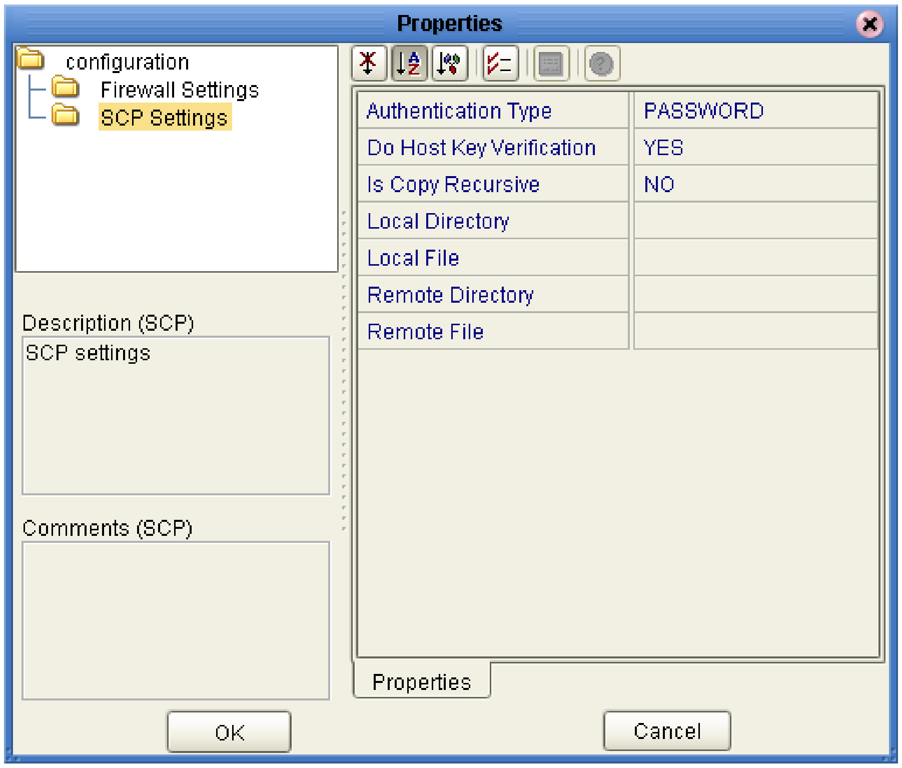
Required Values

Yes or **No**. **Yes** indicates that you are using a firewall. The configured default is **No**.

3.7.2 SCP Settings (BatchSCP Connectivity Map)

The **SCP Settings** section contains the top level parameters displayed in Figure 23.

Figure 23 BatchSCP Properties - SCP Settings section



Authentication Type	PASSWORD
Do Host Key Verification	YES
Is Copy Recursive	NO
Local Directory	
Local File	
Remote Directory	
Remote File	

Authentication Type

Description

Specifies the client authentication type. The options are:

- PASSWORD
- HOSTBASED
- PUBLICKEY

Refer to your specific SSH server documentation for information regarding your authentication type.

Required Values

Select **PASSWORD**, **HOST BASED**, or **PUBLICKEY**. The configured default is **PASSWORD**.

Do Host Key Verification

Description

Specifies whether SSH server authentication by verification of the public key, is enabled.

Required Values

Select **Yes** or **No. Yes** enables SSH server authentication by verifying the public key. The configured default is **Yes**.

Is Copy Recursive

Description

Specifies whether the copy is recursive (for example, copy all sub directories).

Required Values

Select **Yes** or **No. Yes** indicates that the copy is recursive. The configured default is **No**.

Local Directory

Description

Specifies the local directory for files be sent to the remote server, or received from remote server.

Required Values

A local directory.

Local File

Description

Specifies the local file under local directory to be sent to remote, or receive data from remote.

Required Values

The local file.

Remote Directory

Description

Specifies the virtual directory on the SSH (with SFTP sub-system) server where data is published or subscribed to. The accessibility of the directory is usually dependant upon the login user.

Required Values

The remote directory.

Remote File

Description

Specifies the name of a file on the remote server used to either receive published data, or hold data to be retrieved.

Required Values

The remote file.

3.8 BatchSCP eWay Environment Properties

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

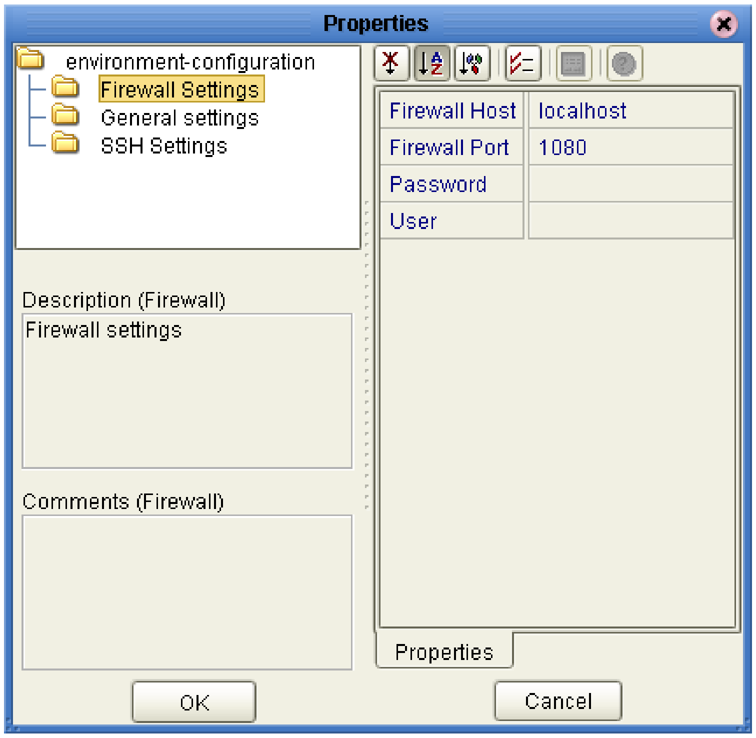
The BatchSCP eWay Environment properties include the following sections:

- **Firewall Settings (BatchSCP Environment)** on page 74
- **General Settings (BatchSCP Environment)** on page 76
- **SSH Settings (BatchSCP Environment)** on page 77

3.8.1 Firewall Settings (BatchSCP Environment)

The **Firewall Settings** section contains the top level parameters displayed in Figure 24.

Figure 24 BatchSCP Properties - Firewall Settings section



Firewall settings

Firewall Host	localhost
Firewall Port	1080
Password	
User	

Firewall Host

Description

Specifies the proxy server host name or IP address.

Required Values

The proxy server host name or IP address. The configured default is **localhost**.

Firewall Port

Description

Specifies the proxy server port number.

Required Values

The proxy server port number. The configured default is **1080**.

Password

Description

Specifies a login password for the proxy server.

Required Values

A password for the proxy server login ID.

User

Description

Specifies a user login ID for the proxy server.

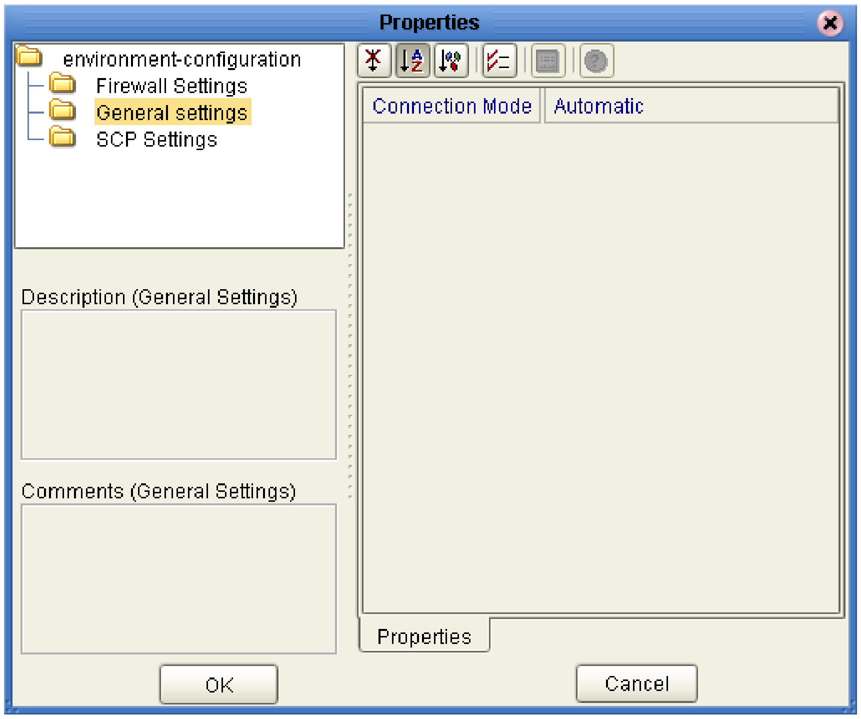
Required Values

A user login ID for the proxy server.

3.8.2 General Settings (BatchSCP Environment)

The **General Settings** section contains the top level parameters displayed in Figure 25.

Figure 25 BatchSCP Properties - General Settings section


Properties
environment-configuration
Firewall Settings
General settings
SCP Settings
Description (General Settings)
Comments (General Settings)
Properties
OK
Cancel

Connection Mode Automatic

Connection Mode

Description

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).

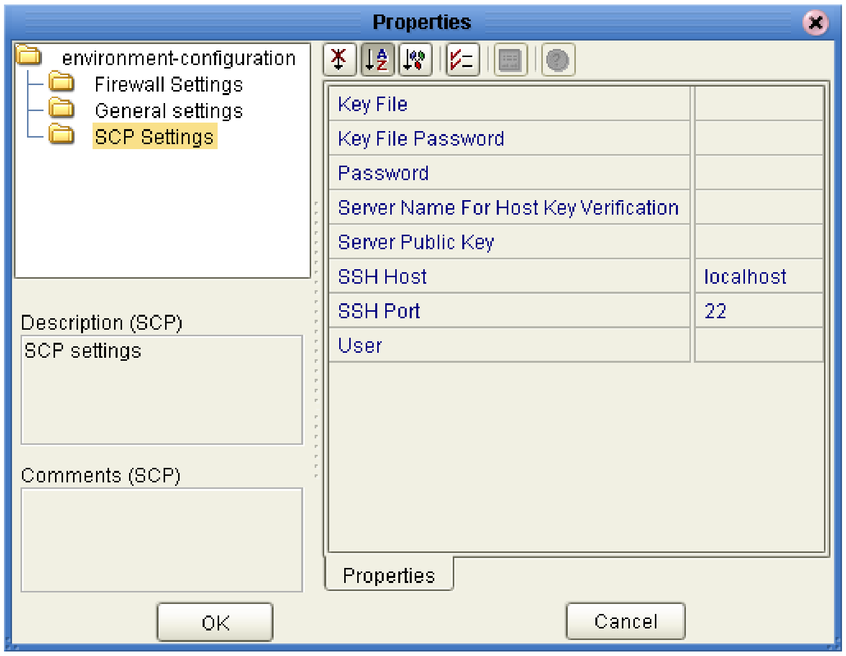
Required Values

Select **Automatic** or **Manual**. The default is **Automatic**.

3.8.3 SSH Settings (BatchSCP Environment)

The **SSH Settings** section contains the top level parameters displayed in Figure 26.

Figure 26 BatchSCP Properties - SSH Settings section



SCP settings

Key File	
Key File Password	
Password	
Server Name For Host Key Verification	
Server Public Key	
SSH Host	localhost
SSH Port	22
User	

Key File

Description

Specifies the location of the file holding the client key pair for client authentication.

Required Values

The location of the file holding the key pair for authentication.

Key File Password

Description

Specifies the password used to protect the key file (key pair).

Required Values

The password.

Password

Description

Specifies a login password for the user.

Required Values

A password.

Server Name For Host Key Verification

Description

Specifies the full domain name of the SSH server used for host key verification.

Required Values

The expected SSH host name when doing host key verification.

Server Public Key

Description

Specifies the public key used by the SSH server. This key is generated on the server and exported, usually into a standard format DER or base 64 encoded DER, and sent to the client through a different, safe channel, and stored on the client machine for host key verification.

See your specific SSH server documentation for more information.

Required Values

The file that contains the server public key for host key verification.

SSH Host

Description

Specifies the host name or IP address of the SSH server.

Required Values

The host name or IP address of the SSH server.

SSH Port

Description

Specifies the port number of the SSH server.

Required Values

The port number of the SSH server.

User

Description

Specifies the user login name for the SSH server.

Required Values

A user login name for the SSH server.

3.9 BatchSFTP eWay Connectivity Map Properties

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

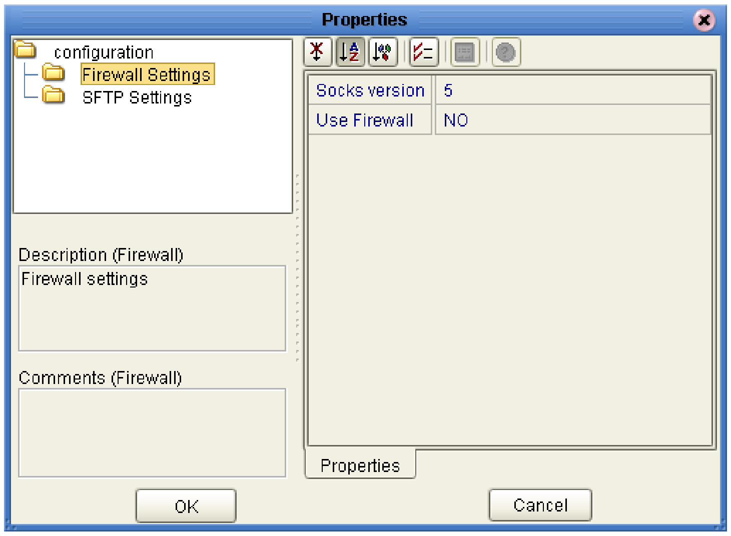
The BatchSFTP eWay Connectivity Map properties include the following sections:

- **Firewall Settings (BatchFTPOverSSL Connectivity Map)** on page 57
- **FTP and SSL Settings (BatchFTPOverSSL Connectivity Map)** on page 59

3.9.1 Firewall Settings (BatchSFTP Connectivity Map)

The **Firewall Settings** section contains the top level parameters displayed in Figure 27.

Figure 27 BatchSFTP Properties - Firewall Settings section



Description (Firewall)
Firewall settings

Comments (Firewall)

Socks version	5
Use Firewall	NO

Socks version

Description

Specifies the SOCKS version of the firewall. The supported options are **4** for SOCKS version 4, or **5** for SOCKS version **5**

Required Values

Select **4** for SOCKS version 4, or **5** for SOCKS version **5**.

Use Firewall

Description

Specifies whether you are using a firewall.

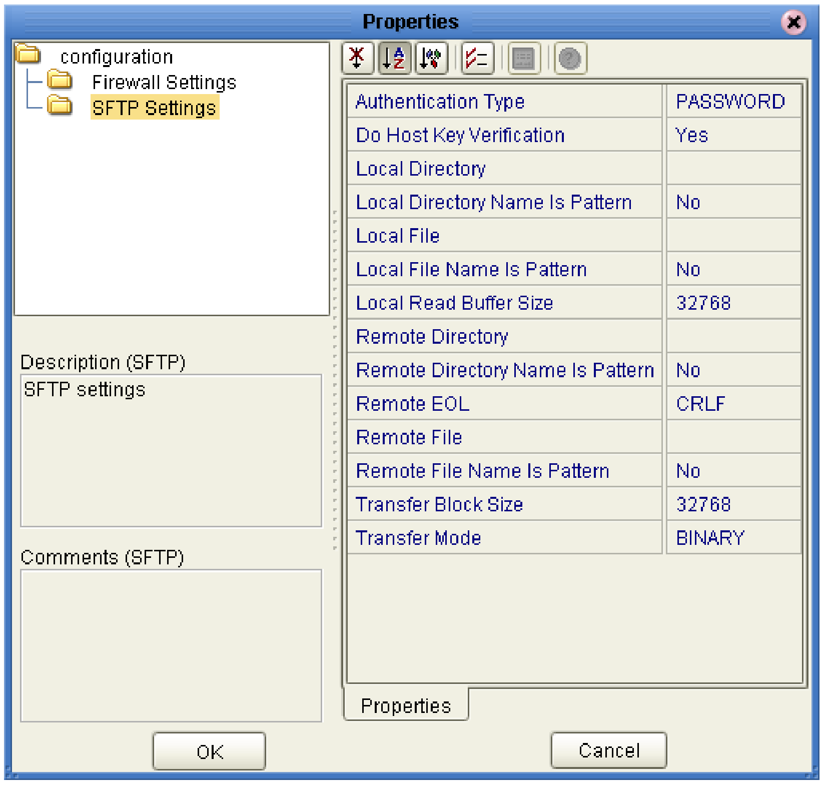
Required Values

Yes or **No**. **Yes** indicates that you are using a firewall. The configured default is **No**.

3.9.2 SFTP Settings (BatchSFTP Connectivity Map)

The **SFTP Settings** section contains the top level parameters displayed in Figure 28.

Figure 28 BatchSFTP Properties - Firewall Settings section



Description (SFTP)
SFTP settings

Comments (SFTP)

Authentication Type	PASSWORD
Do Host Key Verification	Yes
Local Directory	
Local Directory Name Is Pattern	No
Local File	
Local File Name Is Pattern	No
Local Read Buffer Size	32768
Remote Directory	
Remote Directory Name Is Pattern	No
Remote EOL	CRLF
Remote File	
Remote File Name Is Pattern	No
Transfer Block Size	32768
Transfer Mode	BINARY

Authentication Type

Description

Specifies the client authentication type. The options are:

- PASSWORD
- HOSTBASED
- PUBLICKEY

Refer to your specific SSH server documentation for information regarding your authentication type.

Required Values

Select **PASSWORD**, **HOST BASED**, or **PUBLICKEY**. The configured default is **PASSWORD**.

Do Host Key Verification

Description

Specifies whether SSH server authentication by verification of the public key, is enabled.

Required Values

Select **Yes** or **No. Yes** enables SSH server authentication by verifying the public key. The configured default is **Yes**.

Local Directory

Description

Specifies the local directory for files be sent to the remote server, or received from remote server.

Required Values

A local directory.

Local Directory Name is Pattern

Description

Specifies the meaning of the **Local Directory Name** property as follows:

- **Yes**: Indicates that the **Local Directory Name** represents a pattern to be used as a regular expression for pattern matching.
- **No**: Indicates that the **Local Directory Name** represents the exact directory name to be used. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **No**.

Local File

Description

Specifies the local file under local directory to be sent to remote, or receive data from remote.

Required Values

The local file.

Local File Name is Pattern

Description

Specifies the meaning of the **Local File Name** property as follows:

- **Yes**: Indicates that the **Local File Name** represents a pattern to be used as a regular expression for pattern matching.
- **No**: Indicates that the **Local File Name** represents the exact file name to be used. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **No**.

Local Read Buffer Size

Description

Specifies the size (in bytes) of the buffer which is used to read from the local file system. A value of -1 indicates that the whole local file is read at once.

Required Values

A number indicating the size (in bytes) of the local read buffer. A value of -1 indicates that the whole local file is read at once.

Remote Directory

Description

Specifies the virtual directory on the SSH (with SFTP sub-system) server where data is published or subscribed to. The accessibility of the directory is usually dependant upon the login user.

Required Values

The remote directory.

Remote Directory Name is Pattern

Description

Specifies the meaning of the **Remote Directory Name** property as follows:

- **Yes**: Indicates that the **Remote Directory Name** represents a pattern to be used as a regular expression for pattern matching.
- **No**: Indicates that the **Remote Directory Name** represents the exact directory name to be used. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **No**.

Remote EOL

Description

Specifies the remote server - end of line. Options are **CR**, **LF**, **CRLF**.

Required Values

Select **CR**, **LF**, or **CRLF**. **CRLF** is the configured default.

Remote File

Description

Specifies the name of a file on the remote server used to either receive published data, or hold data to be retrieved.

Required Values

The remote file.

Remote File Name is Pattern

Description

Specifies the meaning of the **Remote File Name** property as follows:

- **Yes**: Indicates that the **Remote File Name** represents a pattern to be used as a regular expression for pattern matching.
- **No**: Indicates that the **Remote File Name** represents the exact file name to be used. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **No**.

Transfer Block Size

Description

Specifies the block size used when transferring files. Do not increase, as the remote server may not be able to support higher blocksizes.

Required Values

The block size used when transferring files. The default value is **32768**.

Transfer Mode

Description

Specifies whether the transfer is binary code or ASCII text.

Required Values

Select **BINARY** or **ASCII**. The configured default is **BINARY**.

3.10 BatchSFTP eWay Environment Properties

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

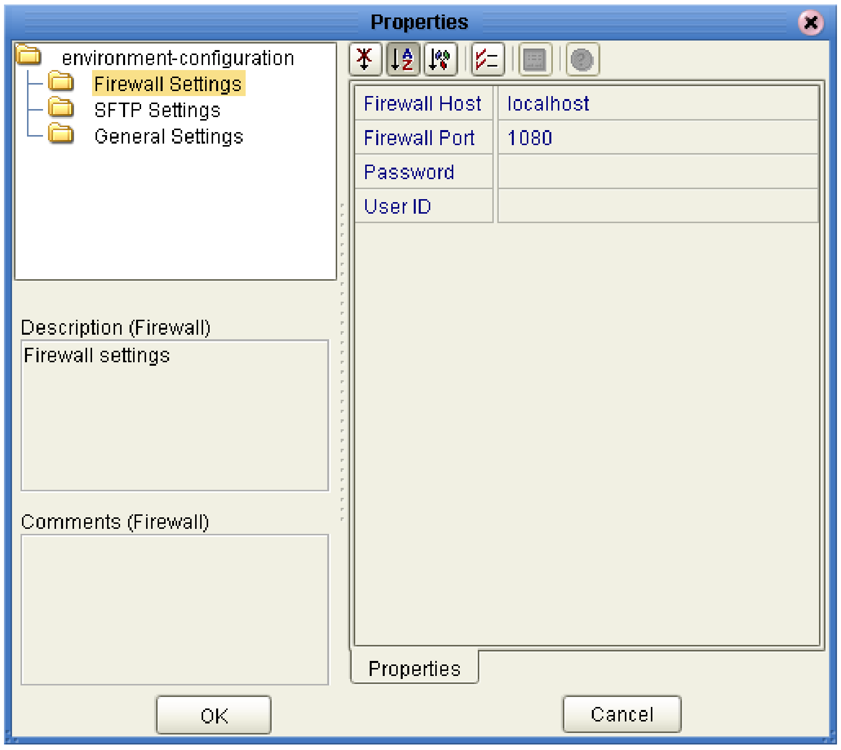
The BatchSFTP eWay Environment properties include the following sections:

- **Firewall Settings (BatchFTPOverSSL Environment)** on page 64
- **FTP and SSL Settings (BatchFTPOverSSL Environment)** on page 66
- **General Settings (BatchFTPOverSSL Environment)** on page 68

3.10.1 Firewall Settings (BatchSFTP Environment)

The **Firewall Settings** section contains the top level parameters displayed in Figure 29.

Figure 29 BatchSFTP Properties - Firewall Settings section



Firewall settings

Firewall Host	localhost
Firewall Port	1080
Password	
User ID	

Firewall Host

Description

Specifies the proxy server host name or IP address.

Required Values

The proxy server host name or IP address. The configured default is **localhost**.

Firewall Port

Description

Specifies the proxy server port number.

Required Values

The proxy server port number. The configured default is **1080**.

Password

Description

Specifies a login password for the proxy server.

Required Values

A password for the proxy server login ID.

User ID

Description

Specifies a user login ID for the proxy server.

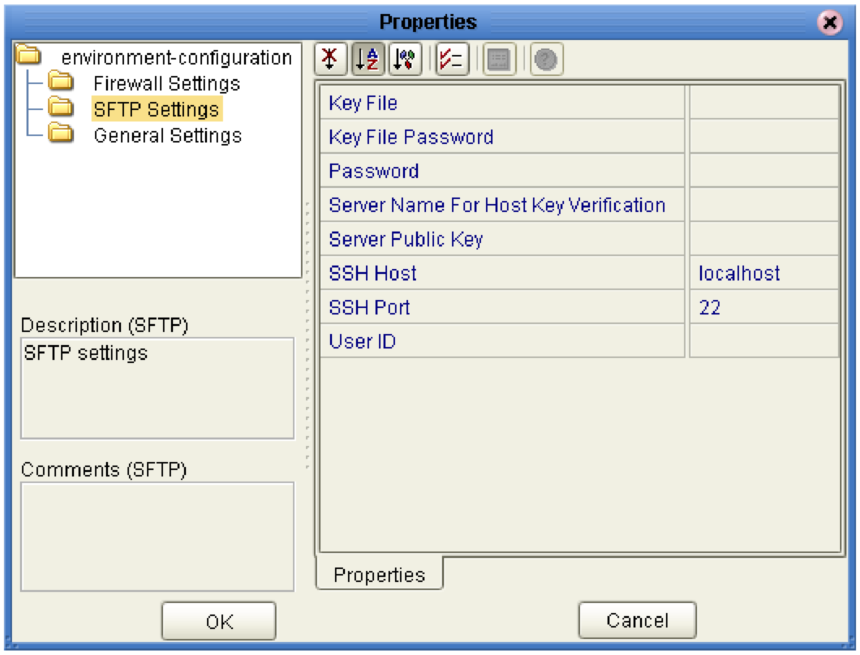
Required Values

A user login ID for the proxy server.

3.10.2 SFTP Settings (BatchSFTP Environment)

The **SFTP Settings** section contains the top level parameters displayed in Figure 29.

Figure 30 BatchSFTP Properties - Firewall Settings section



Key File	
Key File Password	
Password	
Server Name For Host Key Verification	
Server Public Key	
SSH Host	localhost
SSH Port	22
User ID	

Key File

Description

Specifies the location of the file holding the client key pair for client authentication.

Required Values

The location of the file holding the key pair for authentication.

Key File Password

Description

Specifies the password used to protect the key file (key pair).

Required Values

The password.

Password

Description

Specifies a login password for the user.

Required Values

A password.

Server Name For Host Key Verification

Description

Specifies the full domain name of the SSH server used for host key verification.

Required Values

The expected SSH host name when doing host key verification.

Server Public Key

Description

Specifies the public key used by the SSH server. This key is generated on the server and exported, usually into a standard format DER or base 64 encoded DER, and sent to the client through a different, safe channel, and stored on the client machine for host key verification.

See your specific SSH server documentation for more information.

Required Values

The file that contains the server public key for host key verification.

SSH Host

Description

Specifies the host name or IP address of the SSH server.

Required Values

The host name or IP address of the SSH server.

SSH Port

Description

Specifies the port number of the SSH server.

Required Values

The port number of the SSH server.

User ID

Description

Specifies the user login name for the SSH server.

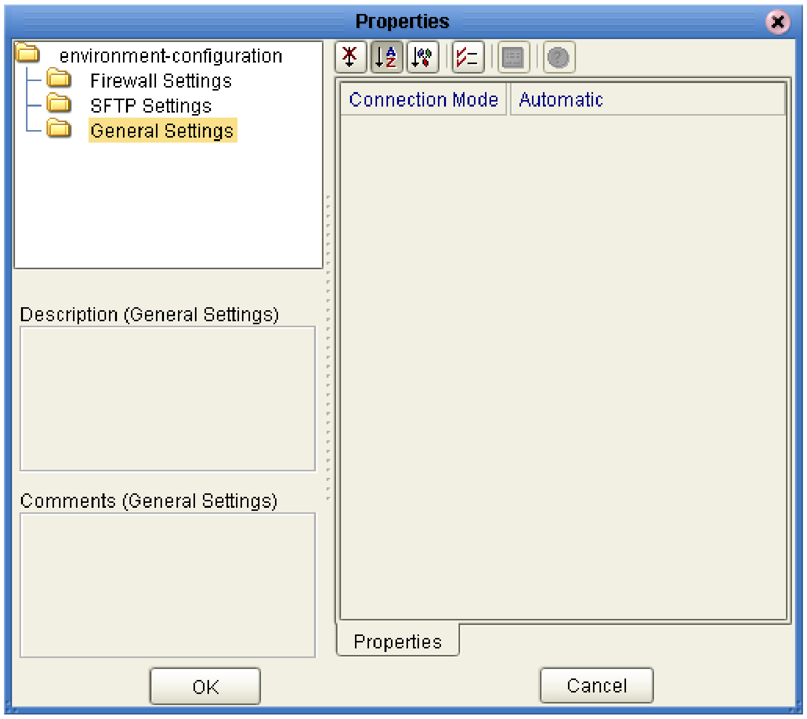
Required Values

A user login name for the SSH server.

3.10.3 General Settings (BatchSFTP Environment)

The **General Settings** section contains the top level parameters displayed in Figure 29.

Figure 31 BatchSFTP Properties - Firewall Settings section


Properties
environment-configuration
Firewall Settings
SFTP Settings
General Settings
Description (General Settings)
Comments (General Settings)
Properties
OK
Cancel

Connection Mode	Automatic

Connection Mode

Description

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).

Required Values

Select **Automatic** or **Manual**. The default is **Automatic**.

3.11 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:

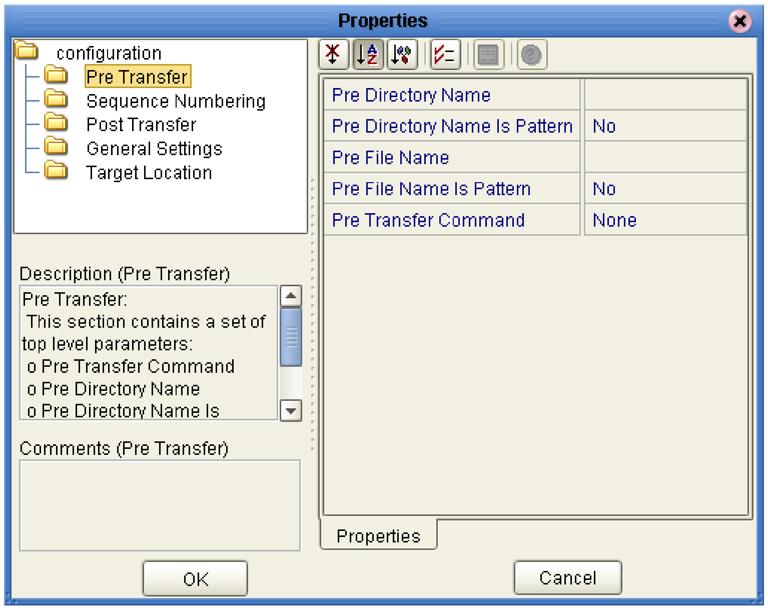
- **Pre Transfer (BatchLocalFile Connectivity Map)** on page 89
- **Sequence Numbering (BatchLocalFile Connectivity Map)** on page 92
- **Post Transfer (BatchLocalFile Connectivity Map)** on page 94
- **General Settings (BatchLocalFile Connectivity Map)** on page 97
- **Target Location (BatchLocalFile Connectivity Map)** on page 99

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.*

3.11.1 Pre Transfer (BatchLocalFile Connectivity Map)

This section provides information about configuring the **Pre Transfer** parameters. Pre-transfer operations are those performed before the data transfer. The **Pre Transfer** section contains the top level parameters displayed in Figure 32.

Figure 32 BatchLocalFile Properties - Pre Transfer section



Description (Pre Transfer)

Pre Transfer:
This section contains a set of top level parameters:
o Pre Transfer Command
o Pre Directory Name
o Pre Directory Name Is

Comments (Pre Transfer)

Pre Directory Name	
Pre Directory Name Is Pattern	No
Pre File Name	
Pre File Name Is Pattern	No
Pre Transfer Command	None

Note: *For more information on this feature, see* **"Pre/post File Transfer Commands" on page 135**.

Pre Directory Name

Description

Specifies either the name of the directory that the remote file is renamed to (**Rename**), or the directory it is moved to (**Move**), depending on the value set in the parameter **Pre Transfer Command**.

Special characters are allowed. The expansion of any special characters are carried out each time this parameter is used.

If you are entering a path name, use the forward slash (/) instead of the back slash (\) because the eWay interprets the back slash as a special character and not a path separator. For example, use **c:/temp/dir** for a path location, *not* **c:\temp\dir**.

Required Values

One of the following values:

- A valid file name or a regular-expression file name
- A valid directory name and path location or the regular-expression directory name and path location on the local system

Pre Directory Name Is Pattern

Description

Specifies the meaning of the **Pre Directory Name** parameter as follows:

- **Yes**: Indicates that the **Pre Transfer Name** (file or directory) represents a pattern to be used as a regular expression for pattern matching on inbound transfers or name expansion on outbound transfers.
- **No**: Indicates that the **Pre Transfer Name** (file or directory) represents the exact directory name to be used for the transfer. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **Yes**.

Pre File Name

Description

Specifies either the name of the file that the remote file is renamed to (**Rename**), or the directory it is moved to (**Move**), depending on the value set in the parameter **Pre Transfer Command**.

Special characters are allowed. The expansion of any special characters are carried out each time this parameter is used.

If you are entering a path name, use the forward slash (/) instead of the back slash (\) because the eWay interprets the back slash as a special character and not a path separator. For example, use **c:/temp/dir** for a path location, *not* **c:\temp\dir**.

Required Values

One of the following values:

- A valid file name or a regular-expression file name
- A valid directory name and path location or the regular-expression directory name and path location on the local system

Pre File Name Is Pattern

Description

Specifies the meaning of the **Pre File Name** parameter as follows:

- **Yes**: Indicates that the **Pre File Name** (file or directory) represents a pattern to be used as a regular expression for pattern matching on inbound transfers or name expansion on outbound transfers.
- **No**: Indicates that the **Pre File Name** (file or directory) represents the exact directory name to be used for the transfer. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **Yes**.

Pre Transfer Command

Description

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.

Your options are:

- **Rename**: Rename the target file.
- **Move**: Move the target file to another directory.
- **None**: Do nothing.

Caution: ***Rename*** *and* ***Move*** *overwrite the file or directory specified by the* ***Pre Transfer Name*** *parameter, if either or both have been entered.*

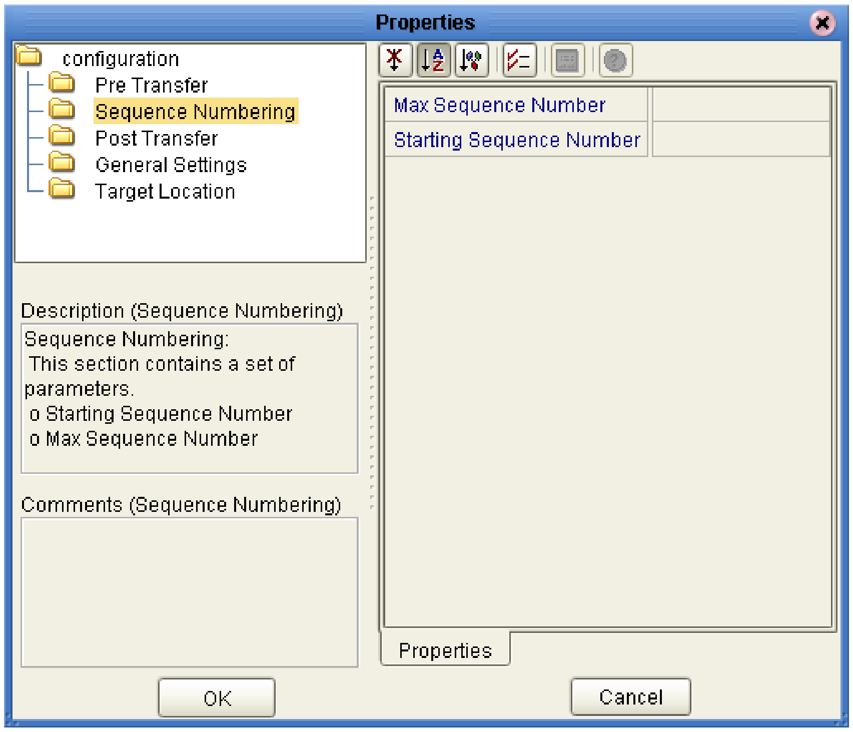
Required Values

Rename, **Move**, or **None**; the default is **None**.

3.11.2 Sequence Numbering (BatchLocalFile Connectivity Map)

The Sequence Numbering section contains the properties displayed in Figure 33.

Figure 33 BatchLocalFile Properties - Sequence Numbering section



Description (Sequence Numbering)

Sequence Numbering:
This section contains a set of parameters.
o Starting Sequence Number
o Max Sequence Number

Comments (Sequence Numbering)

Max Sequence Number	
Starting Sequence Number	

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

Max Sequence Number

Description

Use this parameter when you have set up the target file name to contain a sequence number. It tells the eWay 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 %#.

Required Values

An integer from **1** to **2147483647**. The value of **Max Sequence Number** *must* be greater than that of **Starting Sequence Number**. The default value is **999999**.

Starting Sequence Number

Description

Use this parameter when you have set up the target file name to contain a sequence number. It tells the eWay 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 **%#**.

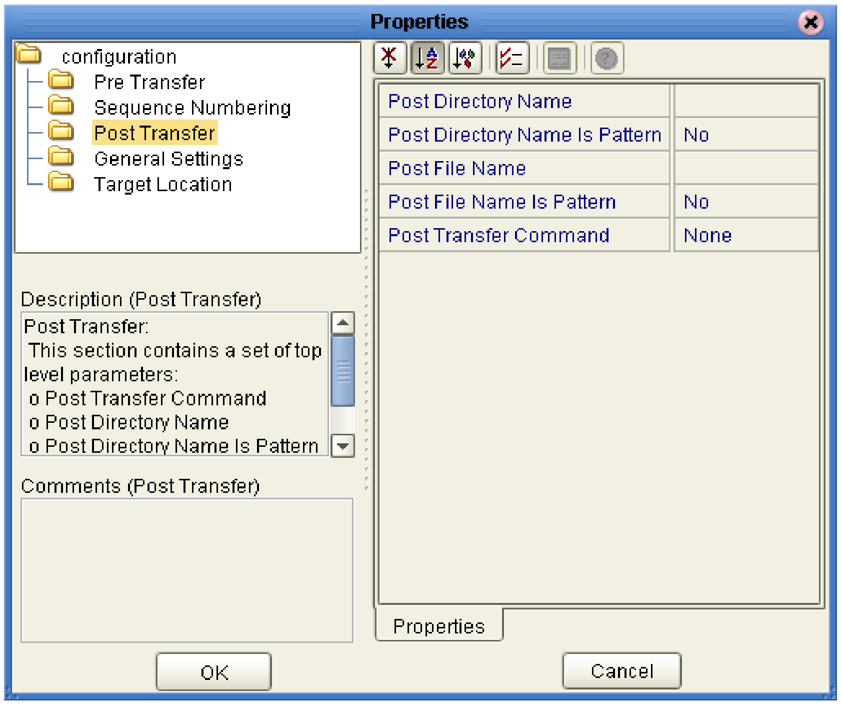
Required Values

An integer from **0** to **2147483647**. The value of the **Starting Sequence Number** *must* be less than the **Max Sequence Number**. The default value is **1**.

3.11.3 Post Transfer (BatchLocalFile Connectivity Map)

Post-transfer operations are those performed after the data transfer. The **Post Transfer** section contains the top level parameters displayed in Figure 34.

Figure 34 BatchLocalFile Properties - Post Transfer section



Post Transfer:
This section contains a set of top level parameters:
o Post Transfer Command
o Post Directory Name
o Post Directory Name Is Pattern

Post Directory Name	
Post Directory Name Is Pattern	No
Post File Name	
Post File Name Is Pattern	No
Post Transfer Command	None

Post Directory Name

Description

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**.

Special characters are allowed. The expansion of any special characters are carried out each time this parameter is used.

If you are entering a path name, use the forward slash (/) instead of the back slash (\) because the eWay interprets the back slash as a special character and not a path separator. For example, use **c:/temp/dir** for a path location, *not* **c:\temp\dir**.

Required Values

One of the following values:

- A valid file name or a regular-expression file name.
- A valid directory name and path location or the regular-expression directory name and path location on the local system.

Post Directory Name Is Pattern

Description

Specifies the meaning of the **Post Transfer Name** parameter as follows:

- **Yes**: Indicates that the **Post Transfer Name** (file or directory) represents a pattern to be used for name expansion.
- **No**: Indicates that the **Post Transfer Name** represents the exact file or directory name to be used for the transfer. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **Yes**.

Post File Name

Description

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**.

Special characters are allowed. The expansion of any special characters are carried out each time this parameter is used.

If you are entering a path name, use the forward slash (/) instead of the back slash (\) because the eWay interprets the back slash as a special character and not a path separator. For example, use **c:/temp/dir** for a path location, *not* **c:\temp\dir**.

Required Values

One of the following values:

- A valid file name or a regular-expression file name.
- A valid directory name and path location or the regular-expression directory name and path location on the local system.

Post File Name Is Pattern

Description

Specifies the meaning of the **Post Transfer Name** parameter as follows:

- **Yes**: Indicates that the **Post Transfer Name** (file or directory) represents a pattern to be used for name expansion.
- **No**: Indicates that the **Post Transfer Name** represents the exact file or directory name to be used for the transfer. No pattern matching of any kind is performed.

Required Values

Yes or **No**; the default is **Yes**.

Post Transfer Command

Description

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 **Move**) or by destroying it permanently (**Delete**). For an outbound transfer, you can make the transferred file available to other clients by renaming or moving it.

The options are:

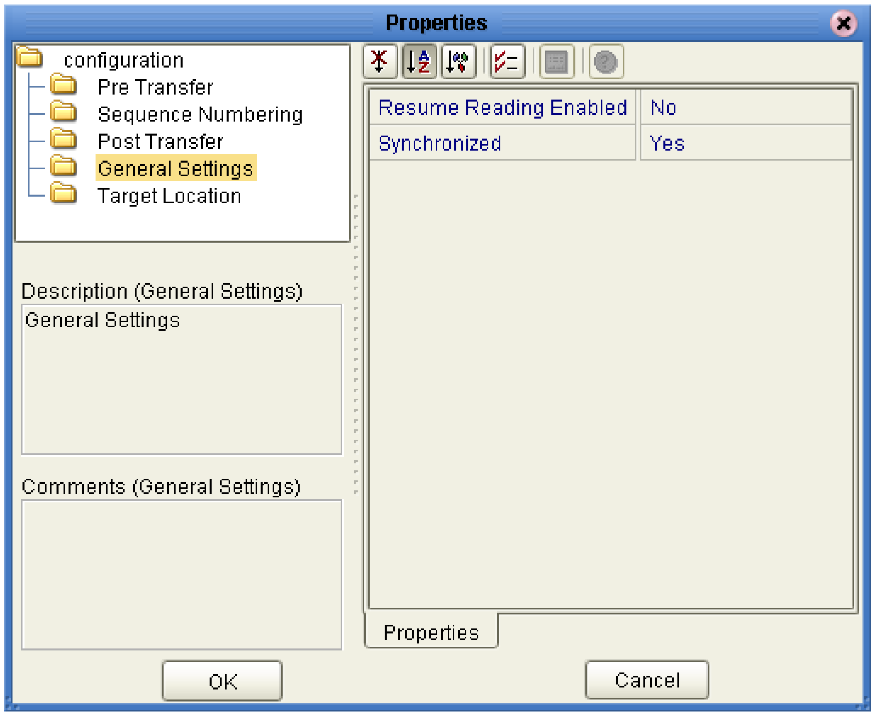
- **Rename**: Rename the transferred file.
- **Move**: Move the target file to another directory.
- **Delete**: Delete the transferred file (inbound transfers only).
- **None**: Do nothing.

The **Rename** and **Move** settings overwrite the file specified under the **Pre Transfer Name** parameter, if one is specified.

3.11.4 General Settings (BatchLocalFile Connectivity Map)

The **General Settings** section contains the top level parameters displayed in Figure 35.

Figure 35 BatchLocalFile Properties - General Settings section



Resume Reading Enabled	No
Synchronized	Yes

Resume Reading Enabled

Description

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.

Required Values

Yes or **No**; the default is **No**.

Note: *Synchronized must be set to “Yes” to use Resume Reading.*

Synchronized

Description

Specifically applies to legacy Batch eWay Projects. Provides backward compatibility to allow Projects that were created using the Batch eWay version 5.0.7 or earlier to be imported and deployed without a change in the eWays behavior. The selections are:

- **Yes**: Provides backward compatibility for legacy (pre-5.0.8 Batch eWay) Projects. The eWay preserves the synchronized behavior.
- **No**: For use with new Batch eWay Projects. The eWay creates multiple instances of the Collaboration that run in Parallel.

We recommend that all OTD instances used in a Project maintain that same value for this property.

Required Values

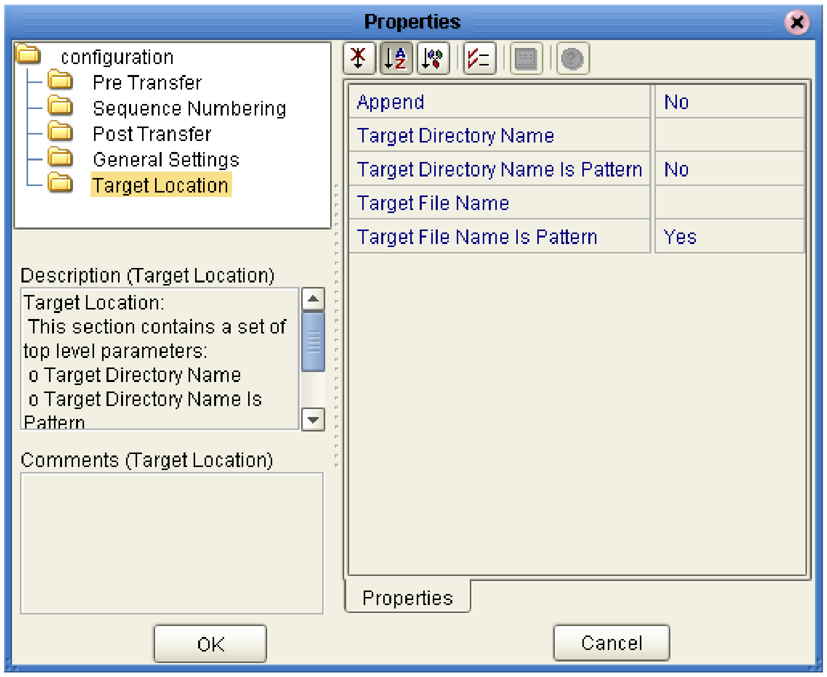
Yes or **No.** The default setting is **Yes**, to provide backward compatibility for existing Projects.

Note: *Synchronized must be set to "Yes" to use Sequence Numbering or Resume Reading.*

3.11.5 Target Location (BatchLocalFile Connectivity Map)

The **Target Location** section contains the top level parameters displayed in Figure 36.

Figure 36 BatchLocalFile Properties - Target Location section



Target Location:
This section contains a set of top level parameters:
o Target Directory Name
o Target Directory Name Is Pattern

Append	No
Target Directory Name	
Target Directory Name Is Pattern	No
Target File Name	
Target File Name Is Pattern	Yes

Append

Description

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:

- If you select **Yes** and the target file already exists, the data is appended to the existing file.
- If you select **No**, the eWay 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.

Required Values

Yes or **No**; the default is **No**.

Target Directory Name

Description

Specifies the directory on the local system from which files are retrieved or where they are sent. This parameter can specify the exact directory path or a regular-expression pattern. For an outbound transfer, the directory is created if it does not already exist.

Required Values

A valid directory name and path location or the regular-expression pattern directory name and path location on the local system.

Target Directory Name Is Pattern

Description

Specifies the meaning of the **Target Directory Name** parameter as follows:

- **Yes**: Indicates that the **Target Directory Name** represents a pattern to be used as a regular expression for pattern matching on inbound transfers or name expansion on outbound transfers.
- **No**: Indicates that the **Target Directory Name** represents the exact directory name to be used for the transfer. No pattern matching of any kind is performed.

Required Values

Yes or **No;** the default is **No**.

Target File Name

Description

Specifies the name of the file on the local system either to be retrieved or sent. This parameter can specify the exact file name or a regular-expression pattern. For outbound data (publishing), the file is created if it does not already exist. This parameter represents the base file name instead of the full file name.

Required Values

A valid file name or a regular-expression file name.

Target File Name Is Pattern

Description

Specifies the meaning of the **Target File Name** parameter as follows:

- **Yes**: Indicates that the **Target File Name** represents a pattern to be used as a regular expression for pattern matching on inbound transfers or name expansion on outbound transfers.
- **No**: Indicates that the **Target File Name** represents the exact file name to be used for the transfer. No pattern matching of any kind is performed.

Required Values

Yes or **No;** the default is **Yes**.

3.12 BatchLocalFile Environment Properties

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

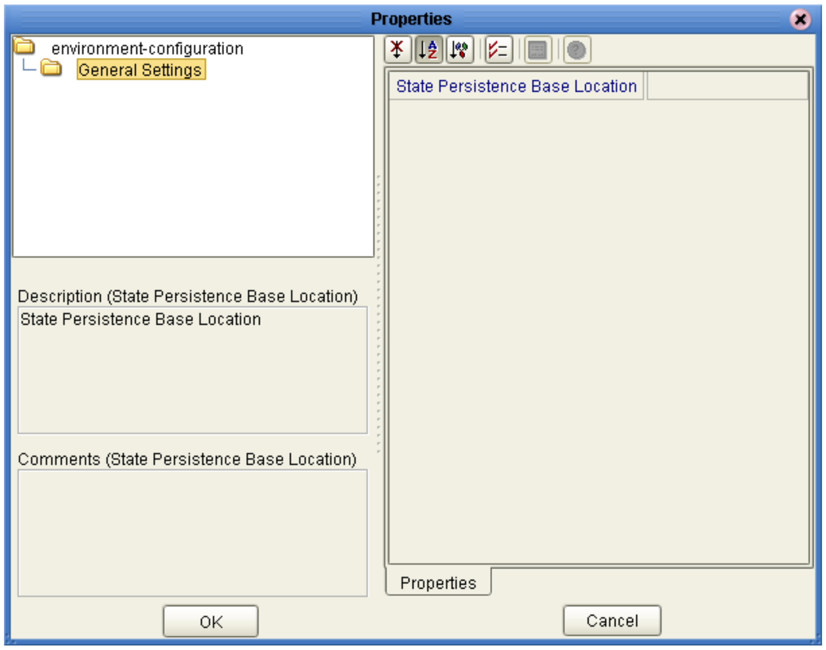
The BatchLocalFile properties include the following sections:

- General Settings (BatchLocalFile Connectivity Map) on page 97

3.12.1 General Settings (BatchLocalFile Environment)

This section provides information about configuring the **General Settings** parameters. The **Pre Transfer** section contains the top level parameters displayed in Figure 37.

Figure 37 BatchLocalFile Properties - General Settings section


Properties
environment-configuration
General Settings
State Persistence Base Location
Description (State Persistence Base Location)
State Persistence Base Location
Comments (State Persistence Base Location)
Properties
OK
Cancel

State Persistence Base Location

Description

Specifies a directory where the per external connection state file resides, that is, a state file to store state information that needs to be persisted. For the BatchLocalFile eWay, this includes sequence number and resume position when **Resume Reading** is turned on. This is an optional property. If this value is left blank, a default file is used.

Required Values

A local directory.

3.13 BatchRecord Connectivity Map Properties

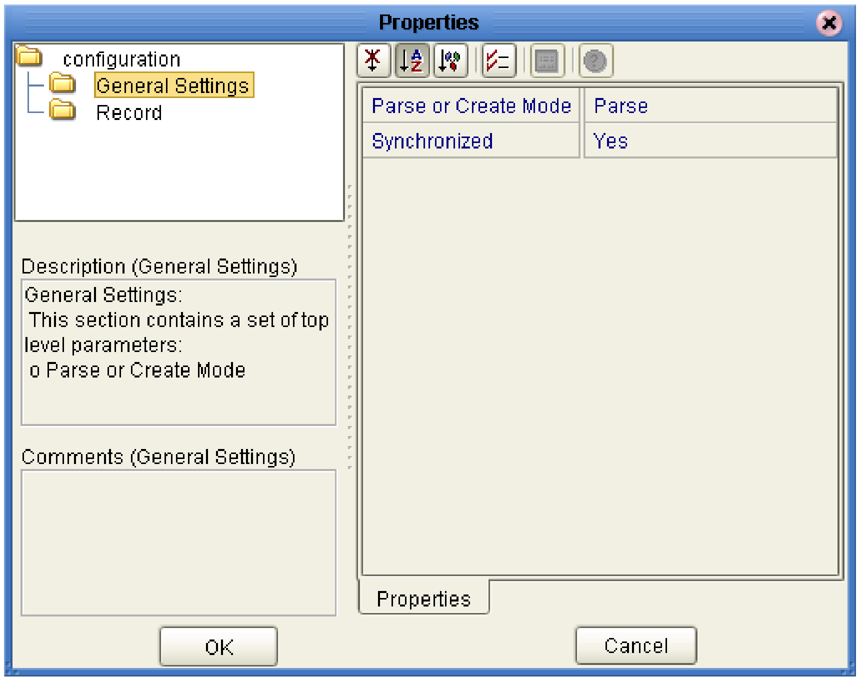
This section explains the properties for the record-processing **BatchRecordOTD**. The BatchRecord properties include the following sections:

- **General Settings (BatchRecord)** on page 102
- **Record (BatchRecord)** on page 104

3.13.1 General Settings (BatchRecord)

The **General Settings** section contains the top level parameters displayed in Figure 38.

Figure 38 BatchRecord Properties - General Settings section



Description (General Settings)
General Settings:
This section contains a set of top level parameters:
o Parse or Create Mode

Comments (General Settings)

Parse or Create Mode	Parse
Synchronized	Yes

Parse or Create Mode

Description

Specifies how this eWay Connection for the record-processing OTD is used. Set this parameter as follows:

- To use the OTD for parsing an inbound payload, choose **Parse**.
- To use the OTD for creating an outbound payload, choose **Create**.

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.

Required Values

Create or **Parse**; the default is **Parse**.

Synchronized

Description

Specifically applies to legacy Batch eWay Projects. Provides backward compatibility to allow Projects that were created using the Batch eWay version 5.0.7 or earlier to be imported and deployed without a change in the eWays behavior. The selections are:

- **Yes**: Provides backward compatibility for legacy (pre-5.0.8 Batch eWay) Projects. The eWay preserves the synchronized behavior.
- **No**: For use with new Batch eWay Projects. The eWay creates multiple instances of the Collaboration that run in Parallel.

We recommend that all OTD instances used in a Project maintain that same value for this property.

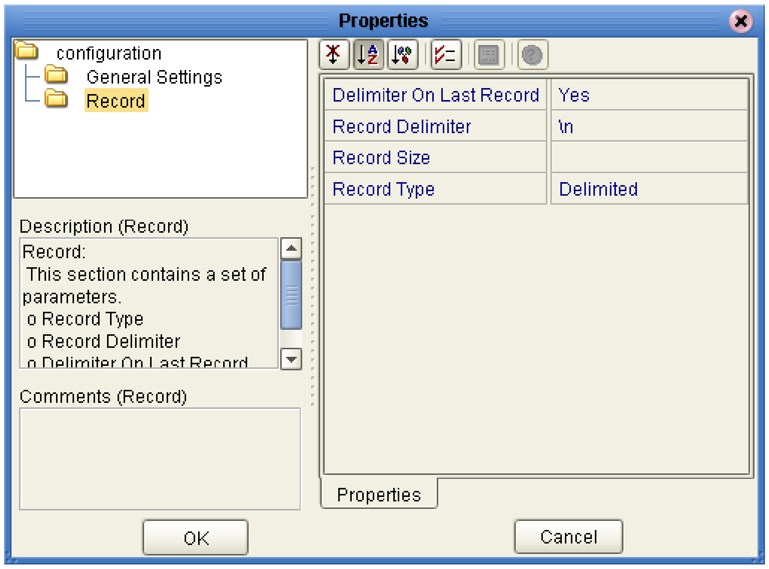
Required Values

Yes or **No.** The default setting is **Yes**, to provide backward compatibility for existing Projects.

3.13.2 Record (BatchRecord)

This section allows you to configure the **Record** parameters, to specify the record characteristics you want the eWay to recognize. The **Record** section contains the top level parameters displayed in Figure 39.

Figure 39 BatchRecord Properties - Record section



Description (Record)

Record:
This section contains a set of parameters.
o Record Type
o Record Delimiter
o Delimiter On Last Record

Comments (Record)

Delimiter On Last Record	Yes
Record Delimiter	\n
Record Size	
Record Type	Delimited

Delimiter on Last Record

Description

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**.

Required Values

No or **Yes** (the default).

Record Delimiter

Description

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 eWay supports all ASCII characters.
    - Example: *,*,* (records separated by ***)
    - Example: | (records separated by a |)

- **Escaped ASCII**: The eWay supports \r, \n, \t, and \f.
  - Example: \r,\n (records separated by CR NL)
  - Example: \n (records separated by NL only)
- **Hex**: The eWay supports 0x00 to 0x7E
  - Example: \0x0D,\0x0A (records separated by CR NL)
- **Octal**: The eWay 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.

Required Values

A valid data record delimiter.

Record Size

Description

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.

Required Values

A number from 1 to 2,147,483,647 indicating the record size (byte count).

Record Type

Description

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 eGate. 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.

Required Values

Delimited (the default), **Fixed**, or **Single Record**.

3.14 BatchInbound Connectivity Map Properties

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

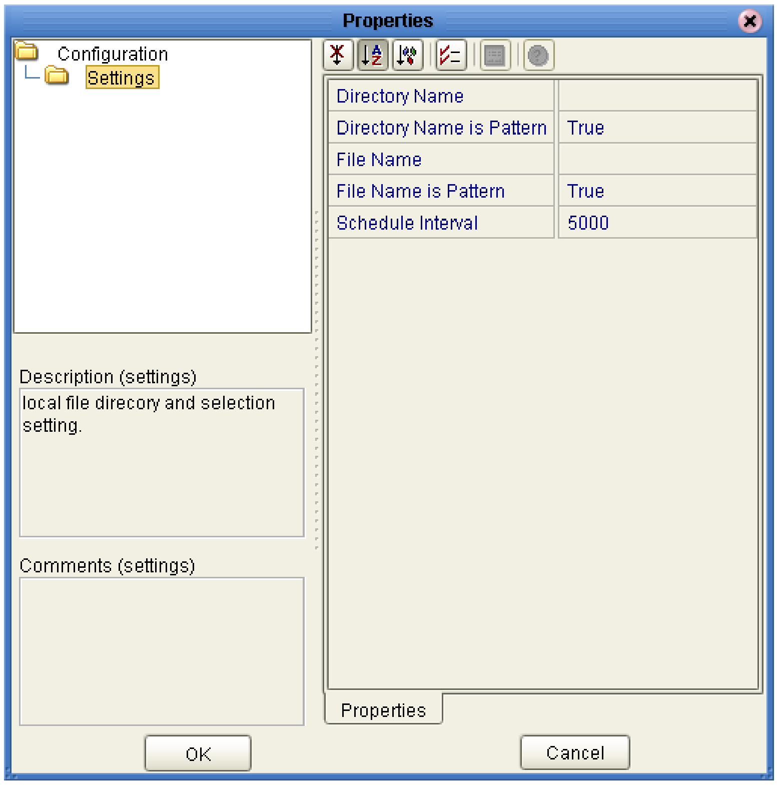
The BatchInbound properties include the following sections:

- **Settings (BatchInbound)** on page 106

3.14.1 Settings (BatchInbound)

The **Settings** section contains the top level parameters displayed in Figure 40.

Figure 40 BatchInbound Properties - Settings section



Description (settings)
local file direcory and selection setting.

Comments (settings)

Directory Name	
Directory Name is Pattern	True
File Name	
File Name is Pattern	True
Schedule Interval	5000

Directory Name

Description

Specifies the input directory name and path (absolute path): for example C:\Temp. It indicates the directory that the BatchInbound eWay polls and from which it transfers input data files.

Required Values

The absolute path (location and name) of the input directory.

Directory Name is Pattern

Description

Specifies whether the **Directory Name** is used as a regular expression to perform pattern matching. Available options:

- **True**: Indicates that the directory name represents a pattern to be used as a regular expression for pattern matching on inbound transfers. For more information on using regular expressions see **Using Regular Expressions** on page 146.
- **False**: Indicates that the directory name represents the exact file name to be used for the transfer. No pattern matching of any kind is performed.

Caution: *Improper use may cause recursive matching.*

Required Values

True or **False**. The configured default is **True**.

File Name

Description

Specifies the input data filename used to select the appropriate input files. This parameter is used in conjunction with **File Name is Pattern**, which specifies whether the value is the exact file name or a regular expression-pattern.

Required Values

A valid **exact file name** (if File Name is Pattern is **False**), or a **regular expression-pattern** (if File Name is Pattern is **True**). For more information on regular expressions see **Using Regular Expressions** on page 146.

File Name is Pattern

Description

Specifies the meaning of the **File Name** parameter as follows:

- **True**: Indicates that the file name represents a pattern to be used as a **regular expression** for pattern matching on inbound file transfers: for example \AMessage(\d){1,3}.txt. For more information on using regular expressions see **Using Regular Expressions** on page 146.
- **False**: Indicates that the file name represents the exact file name to be used for the transfer. No pattern matching of any kind is performed.

Required Values

True or **False**. The default is **True**.

Schedule Interval

Description

Specifies the polling interval, or number of seconds between each poll of the input directory by the eWay for input files.

Required Values

A number indicating the length of time in Milliseconds between eWay polls of the directory. The configured default is 5000 (or 5 seconds).

3.15 Using FTP Heuristics

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

FTP Heuristics: eWay Operation

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

Platform or File Type Selection

The platform is selected from the eWay Properties Editor. To select the specific platform for the eWay do the following:

1. From the Connectivity Map, double-click the BatchFTP eWay. The FTPBatch eWay Properties Editor appears.
2. From the 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 box. The properties for that platform, listed in the FtpHeuristics.cfg file, are now applied to the Directory Listing Style parameter.

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

- UNIX
- AS400
- AS400-UNIX
- HCLFTPD 6.0.1.3
- HCLFTPD 5.1
- 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

- User Defined
- VM/ESA
- VMS
- VOS3 PDS (Hitachi)
- VOS3 PS (Hitachi)
- VOSK (Hitachi)
- Additional platforms and properties you define, if applicable

Note: *When using FTP with an **AS400 UNIX** system some specific FTP configuration settings are required (see **FTP Configuration Requirements for AS400 UNIX** on page 120).*

The FTP Heuristic methods used for communication with **MVS Sequential**, **MVS GDG**, and **MVS PDS** for the Batch eWay are designed for FTP servers (at the mainframe) that use the **IBM IP** stack.

Therefore, when you use FTP to 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 Heuristic features will not work or may require modification.

A **User Defined** platform type is provided in the FtpHeuristics.cfg file which allows you to modify the properties for an unlisted platform (see **Modifying the FTP Heuristics** on page 111).

Additional platforms types may also be added to the FtpHeuristics.cfg file by copying and pasting the User Defined properties (or any of the other platform properties sections), providing a unique name, and modifying the properties for that specific platform. The new platform option is then available to the **Directory Listing Style** parameter (see **Directory Listing Style** on page 32).

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

3.15.1 Modifying the FTP Heuristics

The FTP Heuristics properties are configured by modifying the FtpHeuristics.cfg file for a specific system. To open and modify the FtpHeuristics.cfg file do the following:

1. From the directory **<ican50 directory>\repository\data\files\eWay\BatchFTP** extract **batchadapter.rar** to a temporary directory using a zip program such as WinZip™.
2. From the temporary file, extract the **hostupdate.jar** file in the same manner that you extracted the .rar file in step 1.
3. From the extracted files, open the **configs\FtpHeuristics\FtpHeuristics.cfg** file using any text editor program such as Notepad™.
4. Modify the FTP Heuristics properties as needed for your specific system.
5. Save and repackage the updated files, replacing the FtpHeuristics.cfg file and zipping up the hostupdate.jar and batchadapter.rar files.

The updated FtpHeuristics.cfg file is global for all logical hosts deployed from the repository.

3.15.2 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 or File Type Selection** on page 109.

The FTP Heuristics configuration parameters are as follows:

- **Commands Supported by FTP Server** on page 112
- **Header Lines To Skip** on page 112
- **Header Indication Regex Expression** on page 112
- **Trailer Lines To Skip** on page 113
- **Trailer Indication Regex Expression** on page 113
- **Directory Indication Regex Expression** on page 113
- **File Link Real Data Available** on page 114
- **File Link Indication Regex Expression** on page 114
- **File Link Symbol Regex Expression** on page 114
- **List Line Format** on page 115
- **Valid File Line Minimum Position** on page 115
- **File Name Is Last Entity** on page 116
- **File Name Position** on page 116
- **File Name Length** on page 116
- **File Extension Position** on page 117

- **File Extension Length** on page 117
- **File Size Verifiable** on page 117
- **File Size Position** on page 118
- **File Size Length** on page 118
- **Special Envelope For Absolute Path Name** on page 118
- **Listing Directory Yields Absolute Path Names** on page 119
- **Absolute Path Name Delimiter Set** on page 119
- **Change Directory Before Listing** on page 120
- **Directory Name Requires Terminator** on page 120

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 trailer lines 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

- **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

- **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 2).

Table 2 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	[	.	]	;	[.];
MVS PDS	dir1.dir2(member)	\0	.	(	)	\0.()

Table 2 Delimiters and Path Naming by Platform (Continued)

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

Note: *Where 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 eWay 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**.

3.16 FTP Configuration Requirements for AS400 UNIX

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

- FTP - Use PASV: **No** (see **Use PASV** on page 33)
- FTP Raw Commands - Pre Transfer Raw Commands: **site namefmt 1** (see **Pre Transfer Raw Commands** on page 35)

3.17 Dynamic Configuration

The BatchFTP OTD supports automatic connection during initialization. Automatic connection requires the following properties to be set in the eWay's configuration.

Environment Explorer properties:

- Host Name
- Server Port
- User Name
- Password

Connectivity Map properties:

- Target Directory Name
- Target Directory Name is Pattern (Yes or No)
- Target File Name
- Target File Name is Pattern (Yes or No)

These parameters must be set to valid values prior to using the BatchFTP OTD, so that the eWay can 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 eWay 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 eWay

```
//@map:Client.disconnect
BatchFTP_1.getClient().disconnect();
```

- Set the TargetDirectoryName

```
//@map:Copy "OutDir" to TargetDirectoryName
BatchFTP_1.getConfiguration().setTargetDirectoryName( "OutDir" );
```

- Set the HostName

```
//@map:Copy "myftphostname" to HostName
BatchFTP_1.getConfiguration().setHostName( "myftphostname" );
```

- Connect the eWay

```
//@map:Client.connect
BatchFTP_1.getClient().connect();
```

3 Perform a simple file transfer:

- Get a local file

```
//@map:
BatchLocalFile_1.getClient().get();
```

- Assign the Payload

```
//@map:Copy Payload to Payload
BatchFTP_1.getClient().setPayload(BatchLocalFile_1.getClient().getPay
load() );
```

- 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.

3.18 Alerting and Logging

eGate provides an alerting and logging feature. This allows monitoring of messages, and captures any adverse messages in order of severity based on configured severity level.

Logging Categories for the Batch eWay

The following logging categories apply to the Batch eWay. The level of logging for each category can be modified from the Enterprise Manager's Monitor.

- Batch eWay
- eWay management framework
- eWay resource adapter framework

For information on enabling or modifying the level of logging for the each logging category, see the *eGate Integrator System Administration Guide*.

Note: *The alerts/status notifications for the Batch eWay are currently limited to Started, Running, Stopping, and Stopped.*

Chapter 4

Understanding Batch eWay OTDs

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

What's in This Chapter

4.1 Overview of the Batch OTDs

An OTD contains a set of rules that define an object. The object encodes data as it travels through eGate. 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 eWay. The Batch eWay 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, batch record, or local file data exchange.

4.1.1 Types of Batch eWay OTDs

Table 3 shows the specialized OTDs available with the eWay.

Table 3 Batch eWay 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 FTP over (Secure Shell) protocol.
BatchSFTP	Provides secure data transfer using FTP over (Secure Shell) protocol.
BatchLocalFile	Provides easy access to local file systems.
BatchRecord	Allows the eWay 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 explains each of these OTDs and how to use them with the eWay.

4.1.2 OTD Components

Each of the OTDs is made up of the following components:

- **OTD Operation**: The OTD is used in a Collaboration Rule to operate with eWays.
- **Definition and eGate Enterprise Designer**: An **eWay Properties Sheet** is available, providing a central location in which you can define the eWay's properties.
- **eWay**: An eWay provides access to the information necessary to interface with a specified external application.

All OTDs must be configured and administered using the Enterprise Designer.

Note: *For complete information on how to use the Enterprise Designer and the eWay Properties Sheet, see the* ***eGate Integrator User's Guide****.*

Client Components

Any client components relevant to these OTDs have their own requirements. See the client system's documentation for details.

4.2 FTP Operations OTD

The Batch eWay includes four OTDs that allows you to perform FTP data-transfer functions. The BatchFTP OTD enables the eGate 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 offer enhanced Security features using SSL and SSH protocol.

The BatchFTP OTD data uses a byte array. You must also use a byte array for the payload copy, specifically for binary transfers.

Caution: *It is recommended to use a byte array in every case. Failure to do so can cause loss of data.*

4.2.1 OTD Structure and Operation

The BatchFTP OTD contains three top-level nodes, Configuration, Client, and Provider. Each is described in the sections to follow. You can expand these nodes in the OTD Editor to reveal additional sub-nodes.

Configuration Node

Each field sub-node in the **Configuration** node of the OTD corresponds to one of the eWay'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 eWay.

- The *client interface* represents how the functionality of the provider is actually used.
- The *provider interface* represents all the general FTP operations that can be performed in the OTD.

These operations are the FTP services provided to those who want to use them to create their own implementation.

4.2.2 BatchFTP OTD Node Functions

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

- **BatchFTPOTD**: Represents the OTD's root node.
- **Configuration**: Each field sub-node within this node corresponds to an eWay configuration parameter and contains settings information. See **FTP Operations OTD** on page 126 for details on these parameters and settings.

Note: *This OTD has configuration parameters that can be regular expressions. See* **Using Regular Expressions** *on page 146 for details.*

- **Client**: This node contains the following sub-nodes, which implement the eWay'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 (see **Chapter 5** for details); in such cases, the node represents the properties specified in the configuration.
  - **InputStreamAdapter** and **OutputStreamAdapter**: Allow you to use and control the OTD's data-streaming features; see **Streaming Data Between Components** on page 153 for details.

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 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 135 for details).

Note: *See* **Resolving Names** *on page 151 for more information on these nodes; see* **Using Regular Expressions** *on page 146 for more information on regular expressions.*

  - **get()**, **put()**, **reset()**, **connect()**, **disconnect()**, and **isConnected()**: See **Essential BatchFTP OTD Methods** on page 129.
- **Provider**: The sub-nodes contained in this node are methods that implement the eWay's provider interface in this OTD (**FtpFileProvider**). These methods allow you to do the general FTP operations that can be performed using the OTD.

4.2.3 Using the BatchFTP OTD

The BatchFTP OTD nodes allow you to configure specific eWay configuration parameters for the Collaboration controlling the FTP process. Once you have set the configuration parameters as desired, you do not have to define the same parameters in each corresponding eWay 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 eWay to handle both binary and character data.

For example, you could be using another OTD (such as an OTD from another eWay or a user-defined OTD) where the “data” node has been defined as a string (**java.lang.String**). If you were to drag and drop that string to the BatchFTP OTD’s **Payload** node, the eGate Collaboration Rules Editor can do an automatic type conversion and create code similar to that shown in the next example.

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 Rules Editor uses the Java Virtual Machine (JVM) default encoding to do the conversion to code, as shown in the previous examples.

Note: *For more information on the BatchFTP OTD’s node structure, see* **BatchFTP OTD Node Functions** *on page 126.*

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 eWay. 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 eWay's data-streaming feature, the* ***get()*** *and* ***put()*** *methods operate differently. See* **Streaming Data Between Components** *on page 153 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 eWay's error log file.*

- **restoreConfigValues()**: Allows you to restore the configuration parameter defaults to the related eWay 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 eWay configuration parameters for the OTD.

This parameter is used for the name pattern %#.

Starting Sequence Number

This parameter tells the eWay which value to start with in the absence of a sequence number from the previous run.

When the maximum sequence number is reached, the sequence number rolls over to the starting sequence number.

Maximum Sequence Number

This parameter tells the eWay when this value (the maximum sequence number) is reached, to reset the sequence number to the starting sequence number.

Note: *Keep in mind that the maximum sequence number* ***must*** *be greater than the starting sequence number.*

This feature is also available with the **BatchLocalFile OTD**. For more information on these configuration parameters, see **Sequence Numbering (BatchFTP Connectivity Map)** on page 36.

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 135 for details.

4.3 Batch Secure FTP OTDs

The Batch Secure FTP OTDs: **BatchFTPOverSSL**, **BatchSCP**, and **BatchSFTP**, are in most respects, structured similar to the **BatchFTP** OTD, but include additional configuration properties to enable secure data transfer features.

Secure FTP is a generalized term covering FTP over SSH as well as FTP over SSL. In the Batch eWay, the **BatchSFTP and BatchSCP** OTDs are used to transmit files using FTP over SSH. The **BatchFTPOverSSL** OTD is used to transmit files using FTP over SSL.

Secure Copy or SCP is a means of securely transferring computer files between a local and a remote host or between two remote hosts, using the Secure Shell (SSH) 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. The **BatchSCP** OTD uses SCP to copy a file to or from a remote host. SFTP may be used in place of SCP.

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

Configuration Node

As in the BatchFTP OTD, each field sub-node under the **Configuration** node in the OTDs corresponds to one of the eWay's configuration parameters for that OTD.

Client APIs

Each eWays Client node contains sub-nodes that implement the eWay's client interface in the OTDs. The following section lists the Client API for each of the secure FTP OTDs.

FTPOverSSLClient API

The BatchFTPOverSSL OTD's Client API includes the following methods:

- **connect**: connects to the FTP server and does authentication as configured.
- **disconnect**: disconnects from the FTP server.
- **isConnected**: determines if ICAN is connected to the FTP server.
- **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.
- **deleteDir**: deletes the remote directory as specified by the argument.
- **deleteFile**: deletes the remote file as specified by the argument.
- **mkdir**: creates a directory on the remote. The name of the directory is specified in the configuration parameters.
- **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**).
- **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**).
- **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.
- **getEntryCount**: returns the count of directory entries, as the result of invoking **listDir** or **listDirLong**.
- **getEntry**: gets the directory entry at the specified index (arg).
- **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.
- **nextEntry**: returns the next entry in the result list.

- **resetEntries**: resets the result list iterator so that it can be iterated again.
- **renameFile**: renames the file specified by the configuration parameters **Remote Directory** and **Remote File** to a new name (arg).
- **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.
- **put**: same as **get**, but the data transfer is from local to remote.
- **getPayload**: returns the payload.
- **setpayload**: sets the payload.

SFTPClient API

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

- **connect**: connects to the SSH server and does authentication as configured.
- **disconnect**: disconnects from the SSH server.
- **isConnected**: determines if ICAN is connected to the SSH server.
- **cd**: changes the remote current directory to the specified path.
- **delete**: deletes the remote file or directory as specified in the properties.
- **delete**: deletes the remote file or directory as specified by the configuration parameters, **Remote Directory** and **Remote File**.
- **mkdir**: creates a directory on the remote. The name of the directory is specified in the properties.
- **mkdir**: creates a directory on the remote using the name specified in the configuration parameters, **Remote Directory** and **Remote File**.
- **pwd**: returns the remote current path.
- **lcd**: changes the local current directory.
- **lpwd**: returns the local current path.
- **rename**: renames the file or directory specified by the old name (first argument), to new name (second argument).
- **rename**: renames the file or directory specified by configuration parameters, **Remote Directory** and **Remote File**, to new name (argument).
- **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.
- **put**: same as **get** but the data transfer is from local to remote.
- **getPayload**: returns the payload.
- **setpayload**: sets the payload.

SCPClient API

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

- **connect**: connects to the SSH server and does authentication as configured.
- **disconnect**: disconnects from the SSH server.
- **isConnected**: determines if ICAN is connected to 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.
- **put**: same as **get**, but the data transfer is from local to remote.
- **getPayload**: returns the payload.
- **setpayload**: sets the payload.
- **getRecursive**: the recursive counterpart to **get()**.
- **putRecursive**: the recursive counterpart to **put()**.

4.4 Local File OTD

The BatchLocalFile OTD provides access to files on your local system. While file access is not always necessary in eGate, it makes sense for the Batch eWay 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. This section provides information about these features.

4.4.1 OTD Structure and Operation

Configuration Node

As in the BatchFTP OTD, each field sub-node under the **Configuration** node in the BatchLocalFile OTD corresponds to one of the eWay's configuration parameters for that OTD. See **BatchLocalFile Connectivity Map Properties** on page 89 for details.

Client Node

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

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.

4.4.2 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 eWay configuration parameter and contains the corresponding settings information. See **BatchLocalFile Connectivity Map Properties** on page 89 for details on these parameters and settings.

Note: *This OTD has configuration parameters that can be regular expressions. See* **Using Regular Expressions** *on page 146 for details.*

- **Client**: The following sub-nodes, contained in this node, implement the eWay'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.

Note: *See* **Using Regular Expressions** *on page 146 and* **Using Special Characters** *on page 150 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 153 for details.

    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 135 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.

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

  - **get()**, **put()**, and **reset()**: See **Essential BatchLocalFile OTD Methods** on page 138.
  - **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.*

4.4.3 Using the BatchLocalFile OTD

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

Note: *There is no 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 Rules execution. An example of this usage is 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.*

Advantages of Using the OTD

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

- **Data Streaming**: Allows your system 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 system 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.

Note: *For more information on the Resume Reading feature, see* **Resume Reading Feature** *on page 138.*

Pre/post File Transfer Commands

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

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

Caution: *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 the throwing of an exception in the Collaboration.

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.
- **Copy**: Copy the target file for backup or recovery; you must enter a desired directory and file name.
- **None**: Do nothing.

Note: *The directory is created if it does not already exist.*

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.

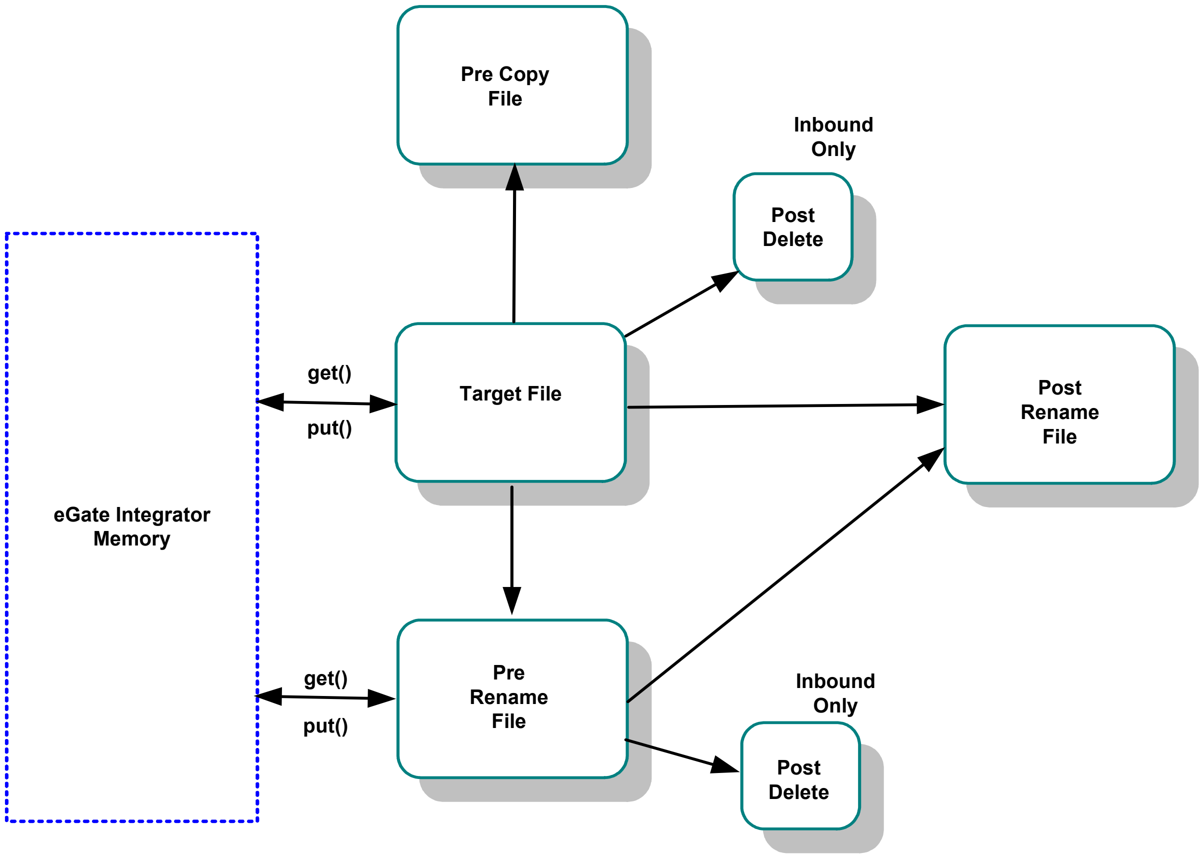
Your post-transfer options are:

- **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.*

Figure 41 shows a diagram of how the different pre- and post-file-transfer commands operate in carrying out **get()** and **put()** method calls.

Figure 41 Pre- and Post-transfer Processes



For information on the eWay configuration parameters for these commands, see:

BatchFTP OTD

- **Pre Transfer (BatchFTP Connectivity Map)** on page 26
- **Post Transfer (BatchFTP Connectivity Map)** on page 38

BatchLocalFile OTD

- **Pre Transfer (BatchFTP Connectivity Map)** on page 26
- **Post Transfer (BatchFTP Connectivity Map)** on page 38

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 eWay. 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**.

Note: *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**.

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

The method throws an exception (**LocalFileException**) if there is a problem.

- **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 the system 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.

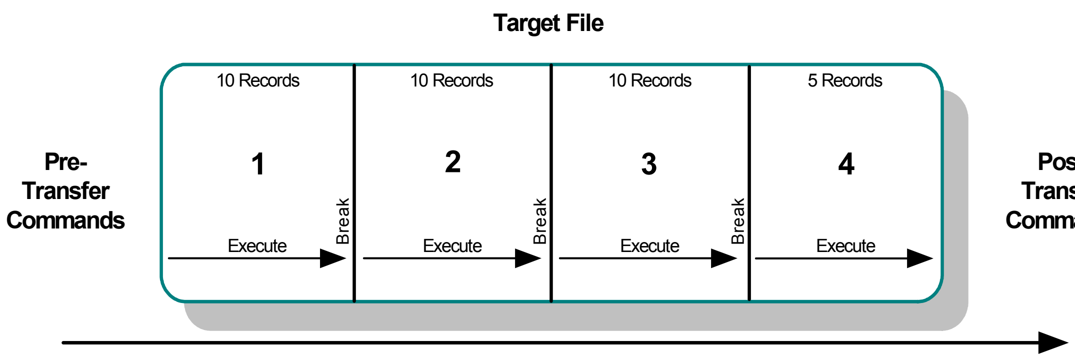
The feature has no special operational requirements besides setting the eWay configuration option. The eWay configuration has an option to enable or disable this feature. This option is also accessible at run time.

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

Step-by-step Operation

Figure 42 shows a diagram of 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 42 Resume Reading Operation



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

1. The eWay starts reading the file then reaches a break condition after a partial data read (the end of **Part 1**), the eWay's pre-transfer commands have already been executed. The resume-reading state is stored, and no post-transfer commands are executed. The eWay 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 eWay 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 eWay 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 eWay 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 eWay 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 eWay'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 153 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 129 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 128 for details.

4.4.4 Recommended Practice

It is recommended that Collaboration Rules use the record-processing OTD together with the BatchLocalFile OTD to parse records or construct payloads. This usage is a better practice than the use of only 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 153 and **Resume Reading Feature** on page 138.

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.

4.4.5 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 eWay itself does not perform any mapping or mounting.

The OTD supports Universal Reference Identifiers (URIs) but the scheme must be left off as follows:

\\drive\directory\file_name

4.5 Batch Record Processing OTD

The Batch eWay'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 types of records 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.

Note: *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.*

4.5.1 OTD Structure and Operation

Each field node in the **Configuration** node in the OTD corresponds to one of the eWay's record-processing configuration parameters.

4.5.2 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 eWay configuration parameter and contains the corresponding settings information, except for the **Parse or Create** parameter. See **BatchRecord Connectivity Map Properties** on page 102 for details.

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 idea to use a byte array in all cases. Failure to do so can cause loss of data.*

- **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 153.

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.*

- **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 it populates the **Record** node with the record retrieved. The method 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 allow the OTD to clean up any unneeded internal data structures.*

4.5.3 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 eWay'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 eWay 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 eWay 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 eWay to throw an exception with an appropriate error message explaining why. Calling **get()** in the create mode would also result in an error.

The eWay 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 eGate 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 eWay 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 (see **Chapter 5** for details on payload streaming).

When you are building a data payload, you must take into account the type and format of the data you are sending. The eWay 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 eWay.

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 drag and drop the data onto the payload node in the OTD (in the Collaboration Rules Editor). In addition, you can specify an input stream as a source (see **Chapter 5** for details on payload streaming).

Extracting Records

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 eWay'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.

Parser Interface

The functionality underlying the record-processing component is described in the parser interface (**BatchRecordParser**). This interface is defined in the **com.stc.eways.batchext** package.

If you want to write your own record-parsing implementation, you can either implement this interface from scratch or derive it from one of our implementations changing only the method or methods you need to change.

Use 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 140
- **Pre/post File Transfer Commands** on page 135

4.6 Batch Input (Trigger) File OTD

Polls for input file, renames the file to a GUID, and triggers the Business Process or Collaboration.

The Batch eWay's **BatchInbound** OTD acts similar to the inbound File eWay, in that it regularly polls an input directory for inbound target files. But unlike the File eWay, when a file with the appropriate name is received by the BatchInbound eWay, the target file is immediately locked so no other process can access it, and renamed to the form: ***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.

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

4.6.1 OTD Structure and Operation

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

4.7 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 eWay.

4.7.1 Regular Expressions: Overview

Regular expressions allow you to specify wildcard patterns for the file name and directory name.

Note: *The full scope of regular expressions is not covered here. For a good explanation of regular expressions, see the book "**sed and awk**" by Dale Dougherty and Arnold Robbins (published by O'Reilly).*

The BatchLocalFile, BatchFTP OTD's, and BatchInbound configurations allow you to use regular expressions, for example, if you want to access all files with the same extension. For more information on available characters and supported syntax when using regular expressions with the eWay, see the following Web site:

http://www.cacas.org/java/gnu/regexp/syntax.html

Regular expressions operate with the BatchLocalFile, BatchInbound, BatchFTP OTDs 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.

Note: *For more information on name patterns, using special characters, see* **Using Special Characters** *on page 150.*

You can specify an extension, for example, .*\.**dat$**. Then, each time the **get()** method is called, the eWay gets the next file with a **.dat** extension. The eWay 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 eWay

You must exercise great care when using regular expressions. This tool can give the new, inexperienced user problems.

Note that there is a **File Name Is Pattern** or **Directory Name Is Pattern** configuration parameter in the eWay configuration interface, after every parameter where you can choose whether to enter a regular expression. This feature allows you to specify that the pattern entered is a regular expression or just a static text entry to be interpreted literally.

Important: *Regular expressions resolve even with a partial match to the file name. The resolution process searches for what the file name contains instead of what the file name is.*

4.7.2 Rules for Directory Regular Expressions

There are special considerations you must take into account when you are using regular expressions for directories. This section explains the general rules and guidelines for using directory regular expressions with the Batch eWay. It also provides some examples.

Basic Directory Regular Expression Rules

The following are the general rules for directory regular expressions:

- 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. So, 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.
- Different platforms require different regular expression patterns, for example:
  - With Windows platforms, use the following pattern:

    drive:\\regexp1\\regexp2\\regexp3 ...
  - With UNIX platforms, including mounted directories, use the following pattern:

    /regexp1/regexp2/regexp3 ...
  - With Windows UNC platforms, use the following pattern:

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

Directory Regular Expression Examples

Several examples of directory regular expression usage follow:

Windows Examples

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

The expression **\D** means any non-digit character.

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

The symbol “.” means any character

UNIX Examples

/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.

Windows UNC Example

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

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

4.8 Using Special Characters

The Batch eWay allows you to use *special characters* to symbolize often-used information in a short-hand way. You can use these character combinations to specify place holders for this information. Using these symbols, you can quickly convey date/time, number, and file-name information.

Special characters are utilities the eWay uses for file-name expansion. 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.

Note: *For information on regular expressions, see* **Using Regular Expressions** *on page 146.*

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.

4.8.1 Types of Name Expansion

The eWay provides the following types of name expansion:

Date/Time stamp

- Uses the format **%[GyMdhHmsSEDFwWakKz]**, for example, **abc%y%y%y%y** means **abc2001** (see **Table 4 on page 151** 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 135), 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.

Additional Examples

- **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.

4.8.2 Resolving Names

Typically, the pre/post names with patterns 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.

Note: *See* **BatchFTP OTD Node Functions** *on page 126 for more information on BatchFTP OTD nodes.*

4.8.3 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 expansion, 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 4 for a complete list.

Table 4 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 4 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.

Examples

Table 5 shows some examples using the U.S. locale.

Table 5 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

Chapter 5

Additional Features

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

What's in This Chapter

5.1 Streaming Data Between Components

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

5.1.1 Introduction to Data Streaming

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

Each OTD component in the eWay 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.

You will 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.*

5.1.2 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.

5.1.3 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 eWay 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 eWay's different types of OTDs, see* **Chapter 4**.

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 input stream adapter to read from.

Data Streaming Setups

The eWay provides four basic data-streaming setups, allowing you to transfer data:

- From a local file system to a record-processing setup (uses **InputStreamAdapter** node in OTD)
- From a record-processing setup to a local file system (uses **OutputStreamAdapter** node in OTD)
- From a local file system to a remote FTP server (uses **InputStreamAdapter** node in OTD)
- From a remote FTP server to a local file system (uses **OutputStreamAdapter** node in OTD)

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

- Release any references to the stream.
- 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.

5.1.4 Stream-adapter Interfaces

This section provides the Batch eWay'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) throws
    StreamingException;
}
```

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;
}
```

5.2 SOCKS FTP Support

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

5.2.1 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 eWay 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 clientss
  - 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 eWay. To enable support, the following properties must be specified in the Batch eWay 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 158**.

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 functionality 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 eWay

Negotiation Methods

The BatchFTP eWay 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 eWay contains a number of properties used to configure SOCKS with the BatchFTP eWay. 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 eWay 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 eWay Connectivity Map Properties** on page 25, and **BatchFTP eWay Environment Properties** on page 49.

5.3 SSH Tunneling Support

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

The Batch eWay 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 is supported for compatibility purposes.

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 eWay 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 eWay makes use of additional software applications. The eWay 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 eWay

To use SSH tunneling to provide for secure logon IDs and passwords, the BatchFTP eWay uses the additional SSH-tunneling software (see **Additional Software Requirements** on page 159).

Enabling SSH Tunneling

To enable SSH tunneling, select **Yes** under the **SSH Tunneling Enabled** parameter in the eWay Connection configuration (see **"SSH Tunneling Configuration Parameters" on page 161**). 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 eWay's use

Using an Existing Channel

To use an existing channel, select **Yes** under the **SSH Channel Established** parameter in the configuration. The eWay 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 eWay automatically uses that channel.

Using an Internal Channel

If you choose **No**, under the **SSH Channel Established** parameter, the eWay launches a process within eGate 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 eWay. 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 eWay Connection.

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

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

Under the eWay'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 eWay connects to the FTP server **atlas:21** through an SSH tunnel.

SSH Tunneling Configuration Parameters

This section lists the SSH tunneling parameters you must set to configure the eWay Connection. For more information, see **"SSH Tunneling Configuration Parameters" on page 161**.

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 eWay 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 eWay 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**.

Chapter 6

Using the Batch eWay With eInsight

This chapter describes how to use the Batch eWay with ICAN Suite's eInsight Business Process Manager and its engine's Web Services interface.

Note: *You must have the **eInsight.sar** file installed to use the Web Services interface.*

What's in This Chapter

6.1 eInsight Engine and eGate Components

You can deploy an eGate component as an Activity in an eInsight Business Process. Once you associate the desired component with an Activity, eInsight invokes it using a Web Services interface. eGate components that can interface with eInsight in this include the following:

- Java Messaging Service (JMS)
- Object Type Definitions (OTDs)
- eWays
- Collaborations

Using the eGate Enterprise Designer and eInsight, you can add an Activity to a Business Process, then associate that Activity with an eGate component, for example, an eWay. Then, when eInsight runs the Business Process, it automatically invokes that component via its Web Services interface.

See the *eInsight Business Process Manager User's Guide* for details.

6.2 Batch eWay With eInsight

You can associate an eInsight Business Process Activity with eGate during the system design phase. To make this association, select the desired operator under the eWay in the Enterprise Explorer and drag it onto the eInsight Business Process canvas.

For Business Process operations, the Batch eWay has the following operators available under the for Batch configuration nodes:

BatchFTP

- get
- put

BatchFTPOverSSL

- get
- put

BatchSFTP

- get
- put

BatchSCP

- get
- put

BatchLocalFile

- read
- write

BatchInbound

- receive

BatchRecord

Not Applicable

The operator automatically changes to an Activity with an icon identifying the component that is the basis for the Activity.

At run time, the eInsight engine invokes each step in the order defined in the Business Process. Using the engine’s Web Services interface, the Activity invokes the Batch eWay.

6.3 Considerations

The following items must be considered when implementing a Batch eWay Project:

- An error may occur upon activation for existing or imported Projects that use the **BatchLocalFile** eWay. This is due to a new BatchLocalFile Environment property that has been added to the eWay. To resolve this, refresh (open and close) your Project's BatchLocalFile Environment properties, and save the Project to the repository before activating the Project.
- When using FTP with an **AS400 UNIX** system, the following FTP configuration settings are required:
  - FTP - Use PASV: **No** (see **Use PASV** on page 33)
  - FTP Raw Commands - Pre Transfer Raw Commands: **site namefmt 1** (see **Pre Transfer Raw Commands** on page 35)

6.4 Importing a Sample Project

Sample eWay Projects are included as part of the installation CD-ROM package. To import a sample eWay Project to the Enterprise Designer do the following:

1. The sample files are first uploaded with the Batch eWay's documentation .sar file (**BatcheWayDocs.sar**) and downloaded from the Enterprise Manager's Documentation tab. Extract the samples from the Enterprise Manager to a local file.
2. Save all unsaved work before importing a Project.
3. From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import** from the shortcut menu. The **Import Manager** appears.
4. Browse to the directory that contains the sample Project zip file. Select the sample file (for example, **Batch_BP_FTPIn_Sample.zip**) and click **Import**. After the sample Project is successfully imported, click **Close**.
5. Before an imported sample Project can be run you must do the following:
   - Create an **Environment** (see **Creating an Environment** on page 173)
   - Configure the eWay Properties for your specific system (see **Creating and Configuring the Batch eWay** on page 21)
   - Create a **Deployment Profile** (see **Creating and Activating the Deployment Profile** on page 176)

6.5 The Batch eWay eInsight Sample Projects

The following sample Projects demonstrate how eInsight Business Processes are used with the Batch eWay:

- **The Batch_BP_FTPIn Sample Project** on page 167
- **The Batch_BP_LFIn Sample Project** on page 178
- **The Batch_BP_LFOut Sample Project** on page 186
- **The Batch_Security_Sample_BP Project** on page 193

Sample data files

Sample data files for the Batch eWay Projects are included with the samples. See Input_Files_Readme.txt included with the sample data files for more information.

6.5.1 Sample Project Descriptions

The Batch_BP_FTPIn Sample Project

The **Batch_BP_FTPIn_Sample** Project demonstrates the following:

- The eGate Scheduler prompts the BatchFTP eWay to pick up a file from an external directory.
- The data is converted from bytes to text and published to the File eWay.
- The File eWay then publishes the data file to an external directory.

The Batch_BP_LFIn Sample Project

The **Batch_BP_LFIn** Project demonstrates the following:

- The eGate Scheduler prompts the BatchLocalFile eWay to pick up a file from an external directory.
- The data is converted from bytes to text and published to the File eWay.
- The File eWay then publishes the data file to an external directory

The Batch_BP_LFOut Sample Project

The **Batch_BP_LFOut** Project demonstrates the following:

- The File eWay picks up a file from an external directory and publishes the data to the BatchLocalFile eWay.
- The data is converted from text to bytes.
- The payload node of the BatchLocalFile eWay publishes the converted file to a target directory with the same name as the file content.

The Batch_Security_Sample_BP Project

The **Batch_Security_Sample_BP** Project uses the BatchFTPOverSSL, BatchSFTP, and BatchSCP OTDs to demonstrate the SSL and SSH secure file transfer functions of the eWays.

- The project uses a BatchInbound eWay to poll an external directory for a specific input file. When the file is present it triggers the Business Process to do the following:
- The BatchFTPOverSSL eWay "gets" a file from a remote directory and publishes it to a local directory.
- The BatchFSTP eWay "puts" a file from a remote directory to a local directory.
- The BatchSCP eWay "puts" a file from a remote directory to a local directory.

6.6 The Batch_BP_FTPIn Sample Project

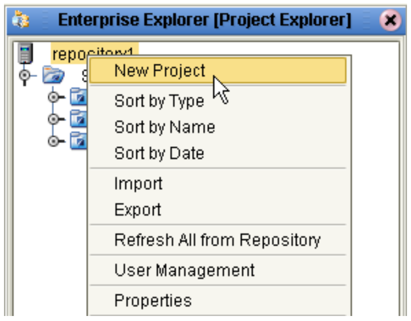
The following pages provide step by step directions for manually creating the **Batch_BP_FTPIn_Sample** Project.

6.6.1 Create a Project

The first step is to create a new Project in the SeeBeyond Enterprise Designer.

1. Start the Enterprise Designer.
2. From the Enterprise Explorer's Project Explorer tab, right-click the Repository and select **New Project** (see **Figure 43 on page 167**). A new Project (Project1) appears on the Project Explorer tree.

Figure 43 Enterprise Explorer - New Project



3. Click twice on **Project1** and rename the Project (for this sample, **Batch_BP_FTPIn_Sample**).

6.6.2 Creating the BP_FTPIn Business Process

Creating the Business Process Flow

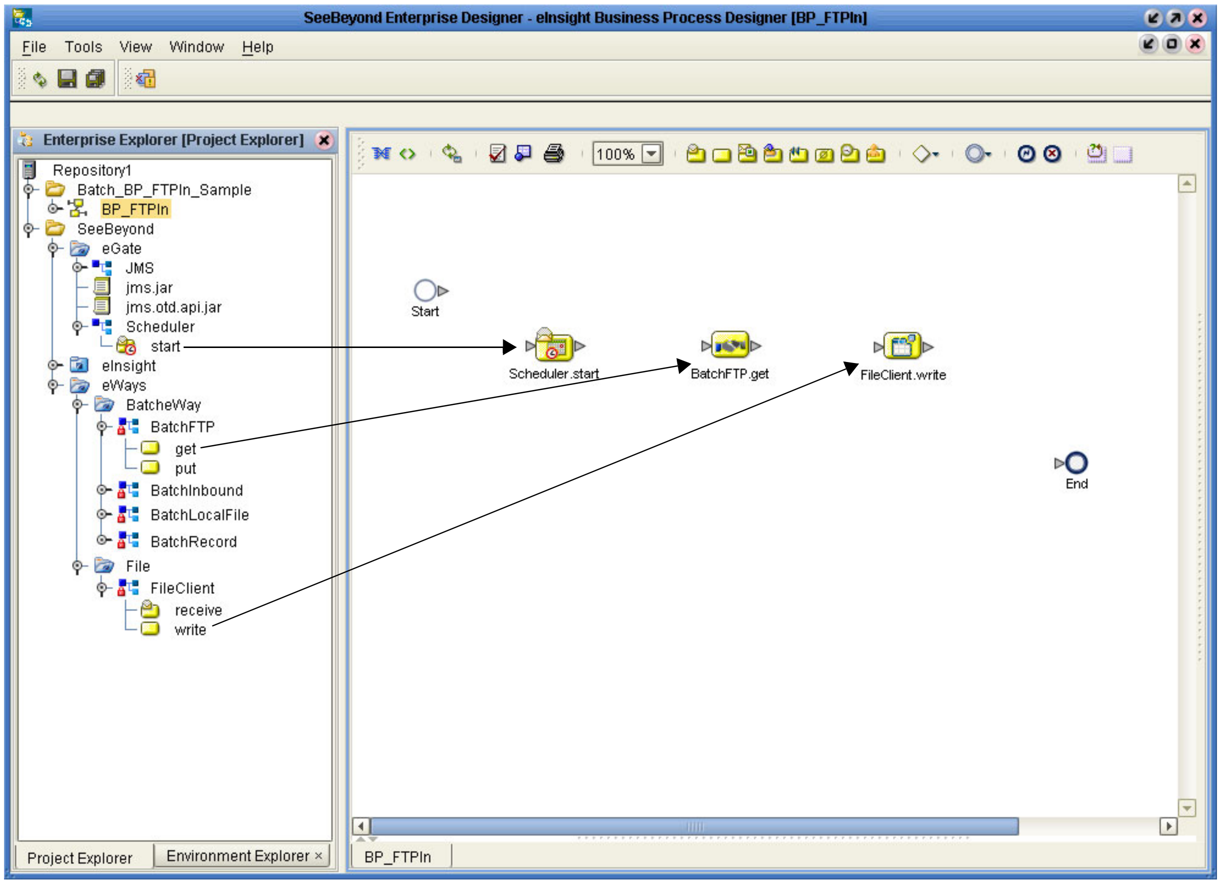
1 From the Enterprise Designer's Project Explorer tree, right-click **Batch_BP FTPIn_Sample**, and select **New** > **Business Process** from the shortcut menu. The eInsight Business Process Designer appears and **BusinessProcess1** is added to the Project Explorer tree. Rename the Business Process to **BP_FTPIn**.

2 From the Project Explorer tree, expand the **SeeBeyond > eGate > Scheduler**, **eWays > BatcheWay > BatchFTP**, and **File > FileClient** nodes to expose the available Business Process elements.

3 Populate the eInsight Business Process Designer's modeling canvas with the following elements from the Project Explorer tree, as displayed in Figure 44 on page 168:

- **start**, under SeeBeyond > eGate > Scheduler
- **get**, under SeeBeyond > eWays > BatcheWay > BatchFTP
- **write**, under SeeBeyond > eWays > File > FileClient

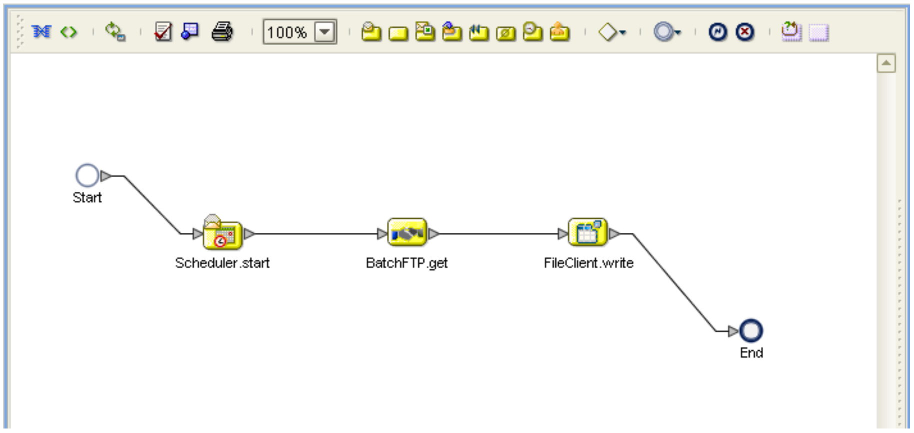
Figure 44 eInsight Business Process Designer - Populate the Canvas



4 Link the modeling elements by clicking on the element's connector and dragging the cursor to the next element's connector, making the following links as displayed in **Figure 45 on page 169**.

- Start -> Scheduler.start
- Scheduler.start -> BatchFTP.get
- BatchFTP.get -> FileClient.write
- FileClient.write -> End

Figure 45 eInsight Business Process Designer - Link the Modeling Elements

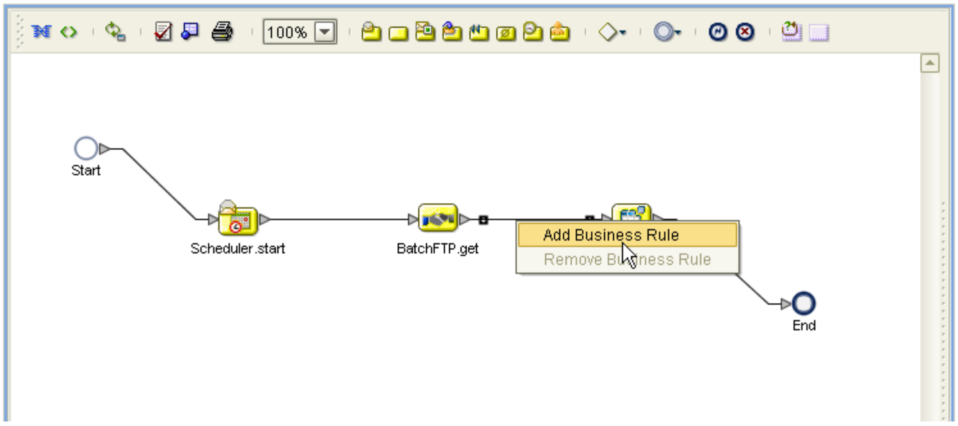


Configuring the Modeling Elements

Business Rules, created between the Business Process elements, define the relationship between the input and output Attributes of the elements.

1 Right-click the link between the **BatchFTP.get** and **FileClient.write** Activities and select **Add Business Rule** from the shortcut menu as displayed in Figure 46.

Figure 46 eInsight Business Process Designer - Adding Business Rules



2 From the eInsight Business Process Designer toolbar, click the **Display Business Rules Designer** button. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.

3 Click on the **Business Rule** icon in the link between **BatchFTP.get** and **FileClient.write** to display the Business Rule Output and Input Attributes in the Business Rule Designer. These Attributes can now be modified.

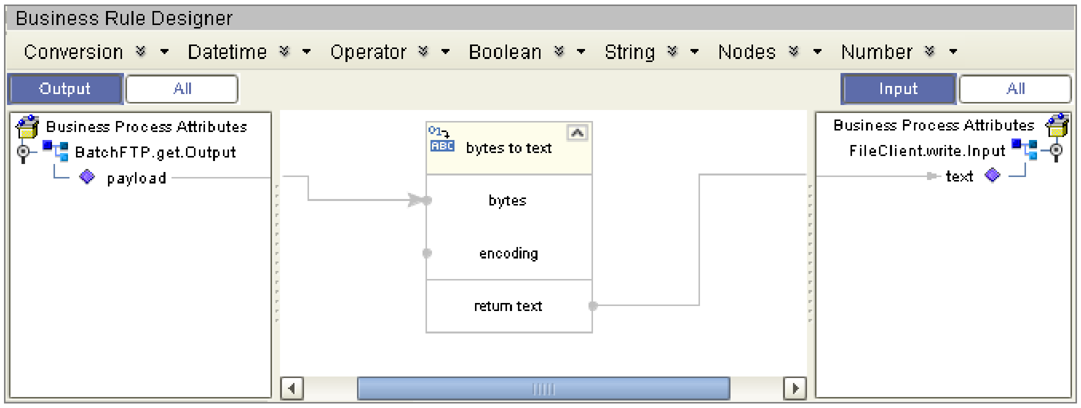
4 From the Business Rule Designer String menu, select **Settings**. The Method Palette appears. From the **String** tab of the Method Palette, select **bytes to text** and click **Close**. The **bytes to text** icon is added to the toolbar.

5 From the Business Rule Designer String menu, select **bytes to text**. The **bytes to text** method box appears.

6 Map **payload,** under BatchFTP.get.Output in the Output pane of the Business Rule Designer, to the **bytes** input node of the **bytes to text** method box. This is done by clicking on **payload** and dragging the cursor to the **bytes** input node of the **bytes to text** method box.

7 Map the **return text** output node of the **bytes to text** method box to text, under FileClient.write.input in the Input pane of the Business Rule Designer (see Figure 47).

Figure 47 eInsight Business Rule Designer



8 From the Business Process Designer toolbar, click the **Synchronize Graphical Model and Business Process Code** icon to synchronize the graphical interface to the Business Process code.

9 Click the Enterprise Designer's **Save All** icon to save your current changes.

6.6.3 Create a Connectivity Map

The Connectivity Map provides a canvas for assembling and configuring a Project's components.

1 In Enterprise Explorer's Project Explorer, right-click the **Batch_BP_FTPIn_Sample** Project and select **New > New Connectivity Map** from the shortcut menu.

2 The New Connectivity Map appears and a node for the Connectivity Map is added to the Project Explorer tree labeled **CMap1**. Rename the Connectivity Map to **CM_Batch_BP_FTPIn**.

The icons in the toolbar represent the available components used to populate the Connectivity Map canvas.

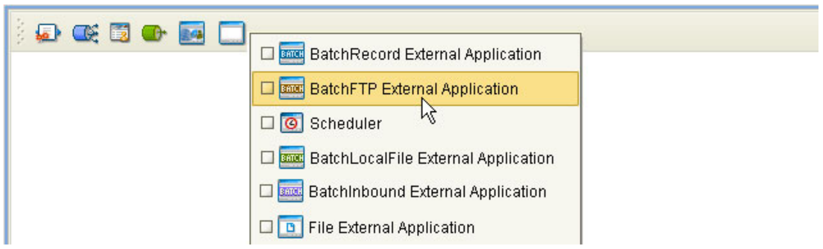
Select the External Applications

When creating a Connectivity Map, the eWays are associated with External Systems. For example, to establish a connection to BatchFTP eWay, you must first select BatchFTP as an External System to use in your Connectivity Map.

To create the External Applications used by the Batch_BP_FTPIn_Sample Project do the following:

1. Click the **External Application** icon on the Connectivity Map toolbar (see Figure 48).

Figure 48 Connectivity Map - External Applications


BatchRecord External Application
BatchFTP External Application
Scheduler
BatchLocalFile External Application
BatchInbound External Application
File External Application

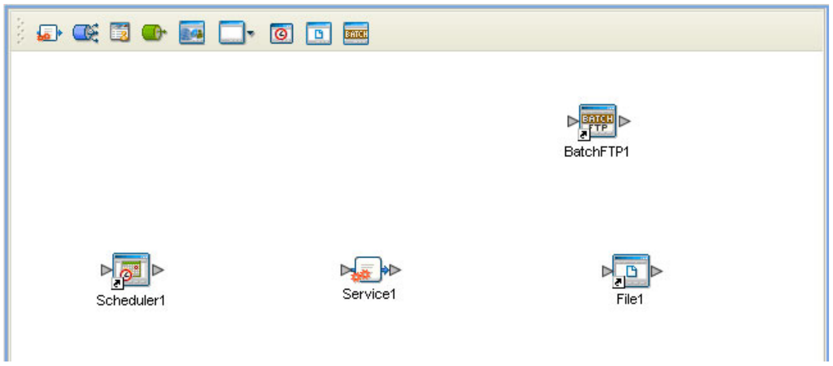
2. Select the applications needed for your Project (for this sample, **Scheduler, File External Application, and BatchFTP External Application**). Icons representing the selected applications are added to the Connectivity Map toolbar.

Populate the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

1. Drag the following components onto the Connectivity Map canvas as displayed in Figure 49:
   - **Scheduler**
   - **Service** (A service is a container for Collaborations, Business Processes, eTL processes, and so forth)
   - **BatchFTP External Application**
   - **File External Application**
2. From the Connectivity Map, rename the **File1** application to **FileOut**, and rename the **Service1** container to **BP_FTP_In.**

Figure 49 Connectivity Map with Components



3 Click the **Save All** icon to save your current changes.

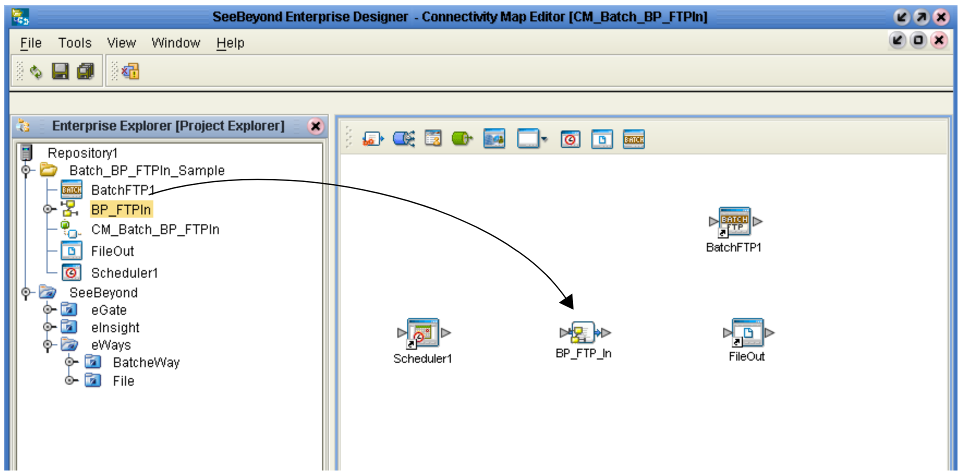
6.6.4 Binding the Project Components

After the Business Processes have been completed, the components are associated and the bindings are created using the Connectivity Map.

1 From the Project Explorer, double-click the Connectivity Map, **CM_Batch_BP_FTPIn**, to display the Connectivity Map canvas.

2 Drag and drop the **BP_FTPIn** Business Process from the Project Explorer to the **BP_FTP_In** service (see Figure 50).

Figure 50 Connectivity Map - Binding the Components



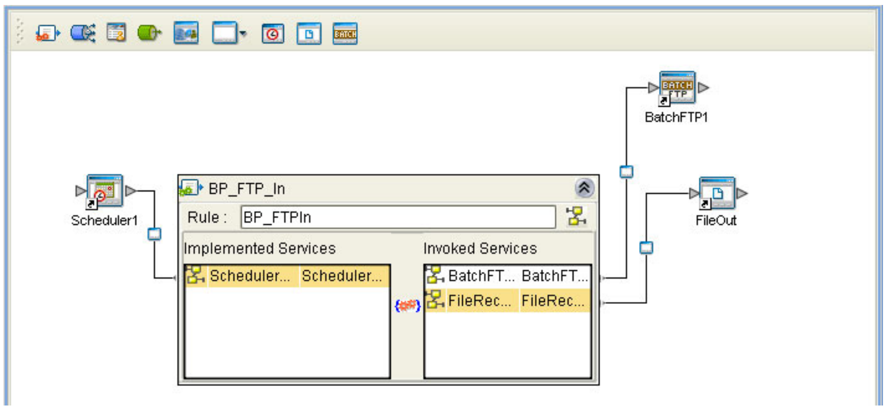
3 Double-click the **BP_FTP_In** service. The **BP_FTP_In** binding box appears with the BP_FTP_In Rule.

4 From the **BP_FTP_In** binding box, map **SchedulerStart** (under Implemented Services) to the **Scheduler1** application.

5 From the **BP_FTP_In** binding box, map **BatchFTPReceiver** (under Invoked Services) to the **BatchFTP1** External Application.

6 From the **BP_FTP_In** binding box, map **FileReceiver** to **FileOut** (see Figure 51).

Figure 51 Connectivity Map - Binding the Components



7 Minimize the **BP_FTP_In** binding box by clicking the chevrons in the upper-right corner and save your current changes.

6.6.5 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and JMS IQ Managers used by a Project and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Explorer and Environment Editor.

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **ENV_Batch_BP_FTPIn**.

4 From the Project Explorer tree, right-click **ENV_Batch_BP_FTPIn** and select **New BatchFTP External System**. Name the External System **BatchFTPExtSys**. Click **OK**. The **BatchFTPExtSys** window is added to the Environment Editor.

5 From the Project Explorer tree, right-click **ENV_Batch_BP_FTPIn** and select **New File External System**. Name this External System **FileExtSysOut** and select **Outbound File eWay** as the External System Type. The **FileExtSysOut** window is added to the Environment Editor.

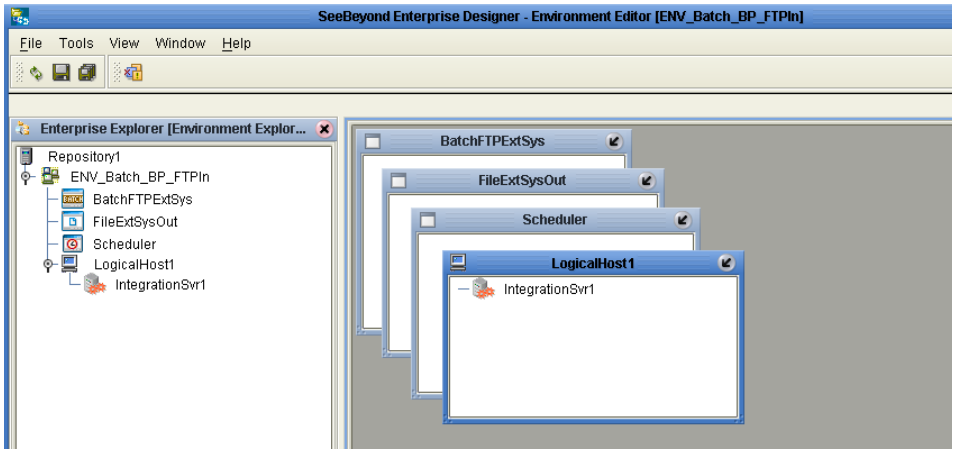
6 From the Project Explorer tree, right-click **ENV_Batch_BP_FTPIn** and select **New Scheduler**. Name the External System **Scheduler**. Click **OK**. The **Scheduler** window is added to the Environment Editor.

7 From the Project Explorer tree, right-click **ENV_Batch_BP_FTPIn** and select **New Logical Host**. The **LogicalHost1** window is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

8 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1**.

9 Save changes to the Repository. The Environment Explorer and Environment Editor now appear as displayed in Figure 52.

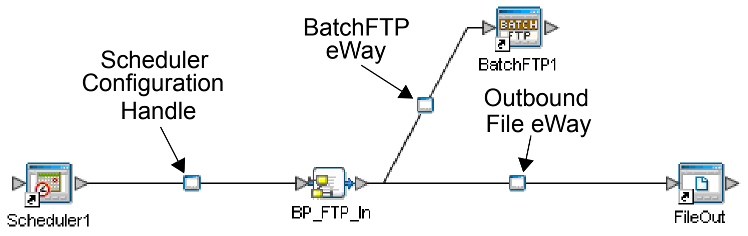
Figure 52 Environment Editor



6.6.6 Configuring the eWay Properties

The Batch_BP_FTPIn_Sample Project contains two eWays, each represented in the Connectivity Map as a node between an External Application and a service. eWays facilitate communication and movement of data between the external applications and the eGate system (see Figure 53).

Figure 53 eWay Configurations



The File eWay configuration parameters are configured from the Connectivity Map. The BatchFTP eWay configuration parameters are set from both the Project Explorer's Connectivity Map and the Environment Explorer tree. To configure the eWays do the following:

Configuring the File eWay Properties

1 Double-click the **Outbound File eWay**, select **Outbound File eWay** in the Templates dialog box and click **OK**.

2 The Properties Sheet opens to the Outbound File eWay properties. Modify the properties for your system, including the settings for the **Outbound File eWay** in Table 6, and click **OK**. The properties are saved for the eWay.

Table 6 Outbound File eWay Settings

Outbound File eWay Properties	
Directory	C:/temp

Outbound File eWay Properties	
Output file name	output%d.dat

Configuring the BatchFTP eWay Properties

The BatchFTP eWay configuration parameters must be set in both the Project Explorer and Environment Explorer. For more information on the BatchFTP eWay properties and the Properties Sheet, see **Creating and Configuring the Batch eWay** on page 21 or see the *eGate Integrator User's Guide*.

For the Batch_BP_FTPIn_Sample Project, do the following:

Modifying the BatchFTP eWay Connectivity Map Properties

1. From the Connectivity Map, double-click the **BatchFTP** eWay. The Properties Sheet opens to the BatchFTP eWay Connectivity Map properties.
2. Modify the **BatchFTP** eWay properties for your system, including the settings in Table 7, and click **OK**.

Table 7 BatchFTP Connectivity Map eWay Settings

BatchFTP eWay Connectivity Map Properties	
Target Location Set as directed, otherwise use the default settings	
Target Directory Name	The directory (absolute path) on the external system from which files will be retrieved or sent.
Target File Name	Retrieved file name

Modifying the BatchFTP eWay Environment Explorer Properties

1. From the **Environment Explorer** tree, right-click the BatchFTP External System (**BatchFTPExtSys** in this sample), and select **Properties**. The Properties Sheet opens to the BatchFTP eWay Environment properties.
2. Modify the BatchFTP eWay Environment properties for your system, including the settings in Table 8, and click **OK**.

Table 8 BatchFTP Environment Explorer eWay Settings

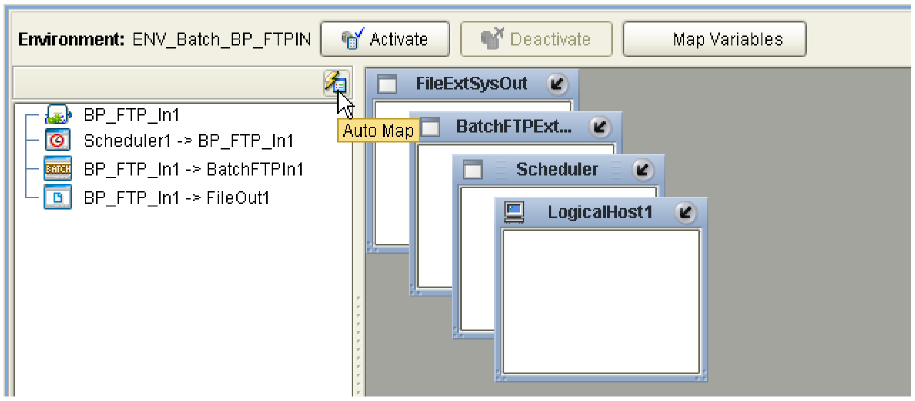
BatchFTP eWay Environment Explorer Properties	
FTP Set as directed, otherwise use the default settings.	
Host Name	*The name of the external system that the eWay connects to.*
Password	*Password required to log into the external system*
Server Port	*Port number to use to connect to the FTP server*
User Name	*User ID used to login to the external system*

6.6.7 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Business Processes and message destinations to the integration server and JMS IQ Manager. Deployment Profiles are created using the Deployment Editor.

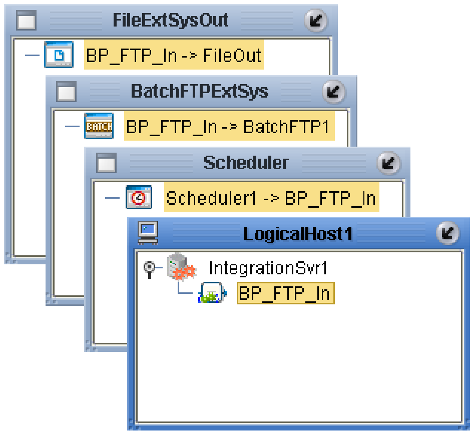
1. From the Enterprise Explorer's Project Explorer, right-click the Project (**Batch_BP_FTPIn_Sample**) and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample DP_Batch_BP_FTPIn). Make sure that the selected Environment is **ENV_Batch_BP_FTPIN**. Click **OK**.
3. Click the **Auto Map** icon as displayed in Figure 54. The Projects components are automatically mapped to their system window as seen in **Figure 55 on page 177**. If any of the Project components are not mapped automatically after Auto Map is used, those component can be mapped manually by following the appropriate steps below. Once all components are mapped, proceed to step 8

Figure 54 Deployment Profile - Auto Map



4. From the left pane of the Deployment Editor, drag **BP_FTP_In -> FileOut** (External Application) to the **FileExtSysOut** window.
5. From the left pane of the Deployment Editor, drag the **BP_FTP_In -> BatchFTP1** (External Application) to the **BatchFTPExtSys** window.
6. Drag **Scheduler1 -> BP_FTP_In** (Application) to the **Scheduler** window.
7. From the left pane of the Deployment Editor, drag **BP_FTP_In** (Business Process) to **IntegrationSvr1** in the **LogicalHost1** window (see Figure 55).

Figure 55 Deployment Profile



8 Click **Activate**. When activation succeeds, save the changes to the Repository.

6.6.8 Running the Project

The following directions assume that the Enterprise Designer was downloaded to **C:\ican50**. If this is not the case, replace that location in the following directions with the appropriate location.

1 From the Enterprise Manager Downloads tab, download **Logical Host - for win32**.

2 Extract the file to the **ican50** directory.

3 Navigate to **C:\ican50\logicalhost\bootstrap\config** directory and open the **logical-host.properties** file using a text editor.

4 Enter the following information in the appropriate fields:

- Logical Host root directory: **ican50\LogicalHost1\logicalhost**
- Repository URL: **http://localhost:*port number*/*repository name***
- Repository user name and password: ***Your user name and password***
- Logical Host Environment name: ***Your Project's Environment name***
- Logical Host name: **logicalhost**

Save your changes to **logical-host.properties** and close the file.

5 Run the **bootstrap.bat** file in the **ican50\logicalhost\bootstrap\bin** directory.

6 Copy the sample input data file to the input directory.

Note: *For UNIX, be sure to upload the appropriate LogicalHost*

For more information on running a Project that utilizes eInsight from the SeeBeyond Enterprise Designer see the *eInsight Business Process Manager User's Guide* and the *eGate Integrator User's Guide*.

6.7 The Batch_BP_LFIn Sample Project

The **Batch_BP_LFIn** Project demonstrates the following: The eGate Scheduler prompts the BatchLocalFile eWay to pick up a file from an external directory. The data is converted from bytes to text and published to the File eWay. The File eWay then publishes the data file to an external directory.

The following pages provide step by step directions for manually creating the Batch_BP_LFIn sample Project components.

To create the **Batch_BP_LFIn** Project do the following:

6.7.1 Create a Project

The first step is to create and name a new Project in eGate Enterprise Designer.

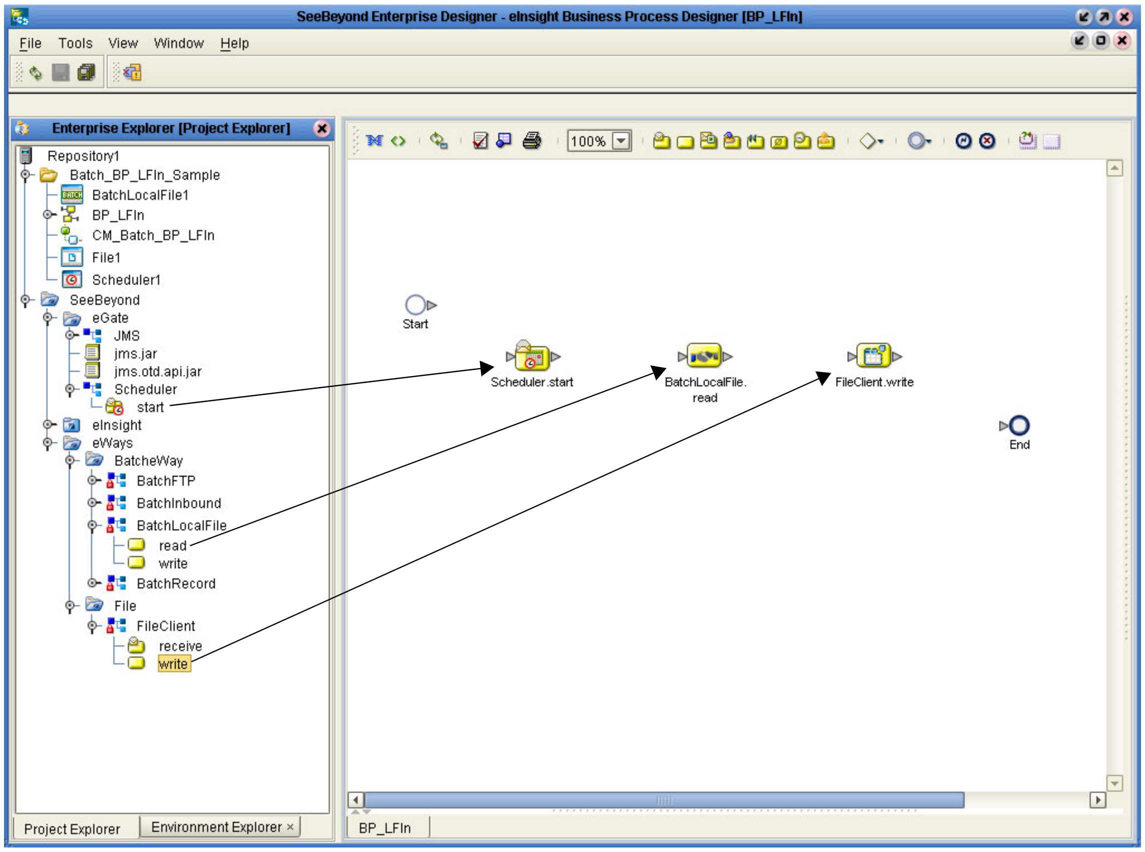
1. Start the Enterprise Designer.
2. From the Enterprise Explorer's Project Explorer tab, select your Repository on the Project Explorer tree, right-click the Repository and select **New Project**. A new Project appears on the Project Explorer tree.
3. Click twice on **Project1** and rename the Project (for this sample, **Batch_BP_LFIn_Sample**).

6.7.2 Creating the BP_LFIn Business Process

Creating the Business Process Flow

1. From the Enterprise Designer's Project Explorer tree, right-click **Batch_BP LFIn_Sample**, and select **New** > **Business Process** from the shortcut menu. The eInsight Business Process Designer appears and **BusinessProcess1** is added to the Project Explorer tree. Rename the Business Process to **BP_LFIn**.
2. Populate the eInsight Business Process Designer's modeling canvas with the following elements from the Project Explorer tree, as displayed in **Figure 56 on page 179**:
   - **start**, under SeeBeyond > eGate > Scheduler
   - **read**, under SeeBeyond > eWays > BatcheWay > BatchLocalFile
   - **write**, under SeeBeyond > eWays > FileClient

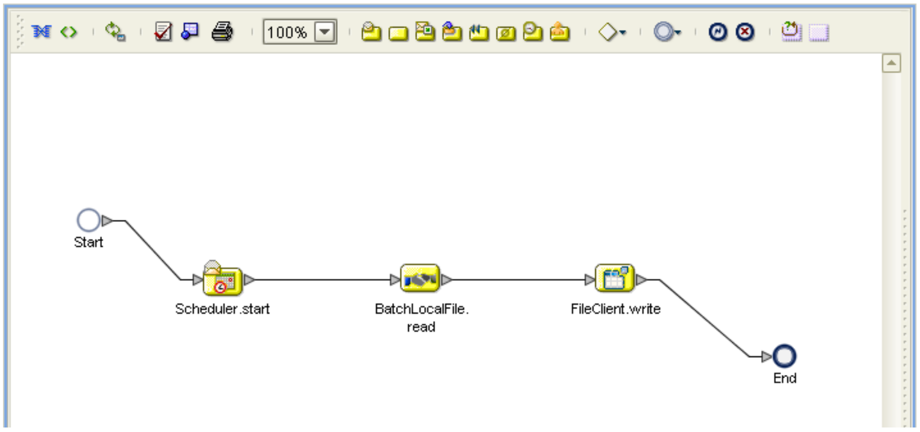
Figure 56 eInsight Business Process Designer - Populate the Canvas



3 Link the modeling elements by clicking on the element's connector and dragging the cursor to the next element's connector, making the following links as displayed in **Figure 57 on page 180**.

- Start -> Scheduler.start
- Scheduler.start -> BatchLocalFile.read
- BatchLocalFile.read -> FileClient.write
- FileClient.write -> End

Figure 57 eInsight Business Process Designer - Link the Modeling Elements

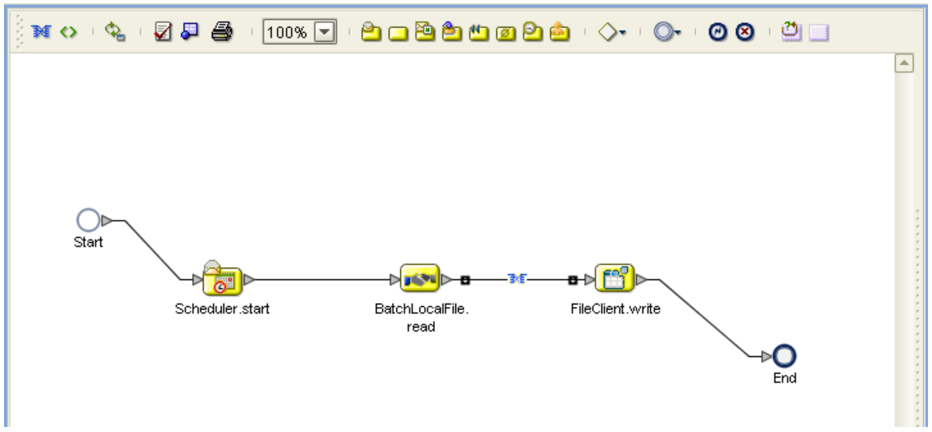


Configuring the Modeling Elements

Business Rules, created between the Business Process elements, define the relationship between the input and output Attributes of the elements.

1 Right-click the link between the **BatchLocalFile.read** and **FileClient.write** Activities and select **Add Business Rule** from the shortcut menu (see Figure 58).

Figure 58 eInsight Business Process Designer - Adding Business Rules



2 From the eInsight Business Process Designer toolbar, click the **Display Business Rule Designer** button. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.

3 Click on the **Business Rule** icon in the link between **BatchLocatFile.read** and **FileClient.write** to display the Business Process Output and Input Attributes in the Business Rule Designer. These Attributes can now be modified.

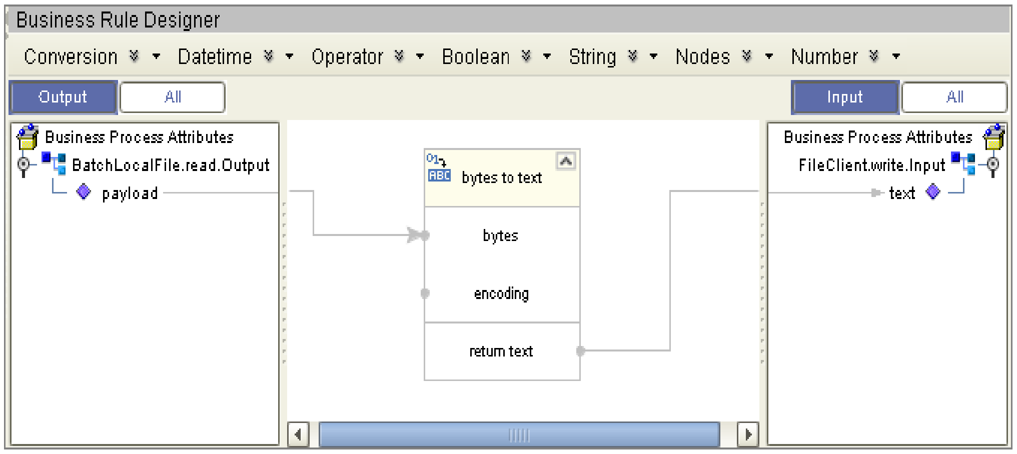
4 From the Business Rule Designer's String menu, select **Settings**. The Method Palette appears. From the **String** tab of the Method Palette, select **bytes to text** and click **Close**. The **bytes to text** icon is added to the toolbar.

5 From the Business Rule Designer String menu, select **bytes to text**. The **bytes to text** method box appears.

6 Map **payload,** under BatchLocalFile.read.Output in the Output pane of the Business Rule Designer, to the **bytes** input node of the **bytes to text** method box. This is done by clicking on **payload** and dragging the cursor to the **bytes** input node of the **bytes to text** method box.

7 Map the **return text** output node of the **bytes to text** method box, to **text** under FileClient.write.input in the Input pane of the Business Rule Designer (see Figure 59).

Figure 59 eInsight Business Rule Designer



8 When the Business Process is complete, from the Business Process Designer toolbar, click the **Synchronize Graphical Model and Business Process Code** icon to synchronize the graphical interface to the Business Process code.

9 Click the Enterprise Designer’s **Save All** icon to save your current changes.

6.7.3 Create a Connectivity Map

The Connectivity Map provides a canvas for configuring a Project’s components.

1 In Enterprise Explorer’s Project Explorer, right-click the new Project (**Batch_BP_LFIn_Sample**) and select **New > New Connectivity Map** from the shortcut menu.

2 The New Connectivity Map appears and a node for the Connectivity Map is added under the Project on the Project Explorer tree labeled **CMap1**. Rename the Connectivity Map **CM_Batch_BP_LFIn**.

Select the External Applications

1 Click the **External Application** icon on the Connectivity Map toolbar,

2 Select the applications needed for your Project (for this sample, **Scheduler**, **BatchLocalFile External Application** and **File External Application**). Icons representing the selected applications are added to the Connectivity Map toolbar.

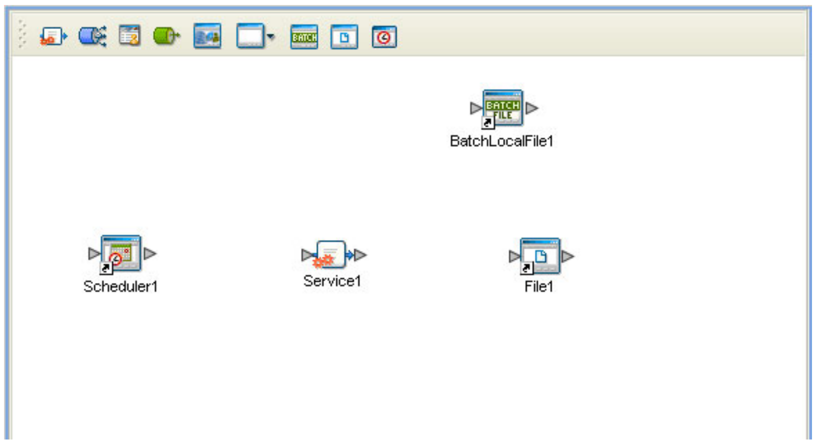
Populate the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

1 For the **Batch_BP_LFIn_Sample** Project, drag the following components onto the Connectivity Map canvas as displayed in Figure 60:

- Scheduler
- Service
- BatchLocalFile External System
- File External System

Figure 60 CM_Batch_BP_LFIn Connectivity Map with Components



2 Rename **File1** to **FileOut1** by clicking the external application's name once and clicking it again. Enter the new name.

3 Rename **BatchLocalFile1** to **BatchLocalFileIn1**.

4 Rename **Service1** to **BP_LFIn1**.

5 Save your current changes to the Repository.

6.7.4 Binding the Project Components

The components are associated and the bindings are created in the Connectivity Map.

1 From the Project Explorer, double-click the Connectivity Map **CM_Batch_BP_LFIn**. The Enterprise Designer canvas now displays the **CM_Batch_BP_LFIn** Connectivity Map.

2 Drag and drop the **BP_LFIn** Business Process from the Project Explorer onto **BP_LFIn1** in the **CM_Batch_BP_LFIn** Connectivity Map.

3 Double-click **BP_LFIn1**. The **BP_LFIn1** binding dialog box appears.

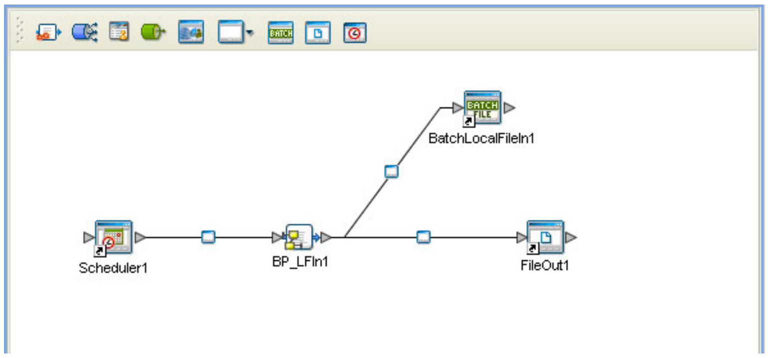
4 From the **BP_LFIn1** binding box, map **SchedulerStart** (under Implemented Services) to the **Scheduler** application.

5 From the **BP_LFIn1** binding box, map the **BatchLocalFileReceiver** (under Invoked Services) to the **BatchLocalFileIn1** External Application.

6 From the **BP_LFIn1** binding box, map the **FileReceiver** (under Invoked Services) to the **FileOut1** External Application.

7 Minimize the **BP_LFIn1** binding box. The Connectivity Map now appears similar to the Connectivity Map displayed in Figure 61.

Figure 61 Connectivity Map - Connecting the Project's Components



8 Save the current changes to your Repository.

6.7.5 Configuring the eWay Properties

The Batch_BP_LFIn_Sample Project uses two eWays, each represented in the Connectivity Map as a node between an External Application and a Collaboration.

The File eWay configuration parameters are configured from the Connectivity Map. The BatchLocalFile eWay unlike the BatchFTP eWay, is only configured from the Project Explorer's Connectivity Map, and not from the Environment Explorer. To configure the eWays do the following:

Configuring the File eWay Properties

1 Double-click the **Outbound File eWay** and select **Outbound File eWay** in the Templates dialog box and click **OK**.

2 The Properties Sheet opens to the Outbound File eWay configuration. Modify the properties for your system.

Configuring the BatchLocalFile eWay Properties

The BatchLocalFile eWay properties are set from the Project Explorer's Connectivity Map. For more information on the BatchLocalFile eWay properties and the Properties Sheet, see **Creating and Configuring the Batch eWay** on page 21 or see the *eGate Integrator User's Guide*.

For the Batch_BP_LFIn_Sample Project, do the following:

Modifying the BatchLocalFile eWay Connectivity Map Properties

1 From the **Connectivity Map**, double-click the **BatchLocalFile** eWay. The Properties Sheet opens to the BatchLocalFile eWay properties.

2 Modify the BatchLocalFile eWay Connectivity Map properties for your system.

6.7.6 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and JMS IQ Managers used by a Project and contain the configuration information for these components.

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **ENV_Batch_BP_LFIn**.

4 Right-click **ENV_Batch_BP_LFIn** and select **New BatchLocalFile External System**. Name the External System **BatchLocalFileExtSys**. Click **OK**. The **BatchLFExtSys** window is added to the Environment Editor.

5 Right-click **ENV_Batch_BP_LFIn** and select **New File External System**. Name this External System **FileExtSysOut** and select **Outbound File eWay** as the External System Type. The **FileExtSysOut** window is added to the Environment Editor.

6 Right-click **ENV_Batch_BP_LFIn** and select **New Scheduler**. Name the External System **Scheduler**. Click **OK**. The **Scheduler** window is added to the Environment Editor.

7 Right-click **ENV_Batch_BP_LFIn** and select **New Logical Host**. The **LogicalHost1** window is added to the Environment Editor.

8 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1**.

9 Save your current changes to the Repository.

6.7.7 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Collaborations and message destinations to the integration server and JMS IQ Manager. Deployment profiles are created using the Deployment Editor.

1 From the Enterprise Explorer's Project Explorer, right-click the Project (**Batch_BP_LFIn_Sample**) and select **New** > **Deployment Profile**.

2 Enter a name for the Deployment Profile (for this sample **DP_Batch_BP_LFIn**). Make sure that the selected Environment is **ENV_Batch_BP_LFIn**. Click **OK**.

3 From the left pane of the Deployment Editor, drag **BP_LFIN1 -> FileOut1** (External Application) to the **FileExtSysOut** window.

4 From the left pane of the Deployment Editor, drag the **BP_LFIn1 -> BatchLocalFileIn1** (External Application) to the **BatchLocalFileExtSys** window.

5 Drag **Scheduler1 -> BP_LFIn1** (Application) to the **Scheduler** window.

6 From the left pane of the Deployment Editor, drag **BP_LFIn1** (Business Process) to **IntegrationSvr1** in the **LogicalHost1** window.

7 Click **Activate**. When activation succeeds, save the changes to the Repository.

6.7.8 Running the Project

To run the Project follow the directions provided in **"Running the Project" on page 177**.

6.8 The Batch_BP_LFOut Sample Project

The **Batch_BP_LFOut** Project demonstrates the following:

- The File eWay picks up a file from an external directory and publishes the data to the BatchLocalFile eWay.
- The data is converted from text to bytes.
- The payload node of the BatchLocalFile eWay publishes the converted file to a target directory with the same name as the file content.

The following pages provide step by step directions for manually creating the **Batch_BP_LFOut** sample Project components.

To create the **Batch_BP_LFOut** Project do the following:

6.8.1 Create a Project

The first step is to create and name a new Project in eGate Enterprise Designer.

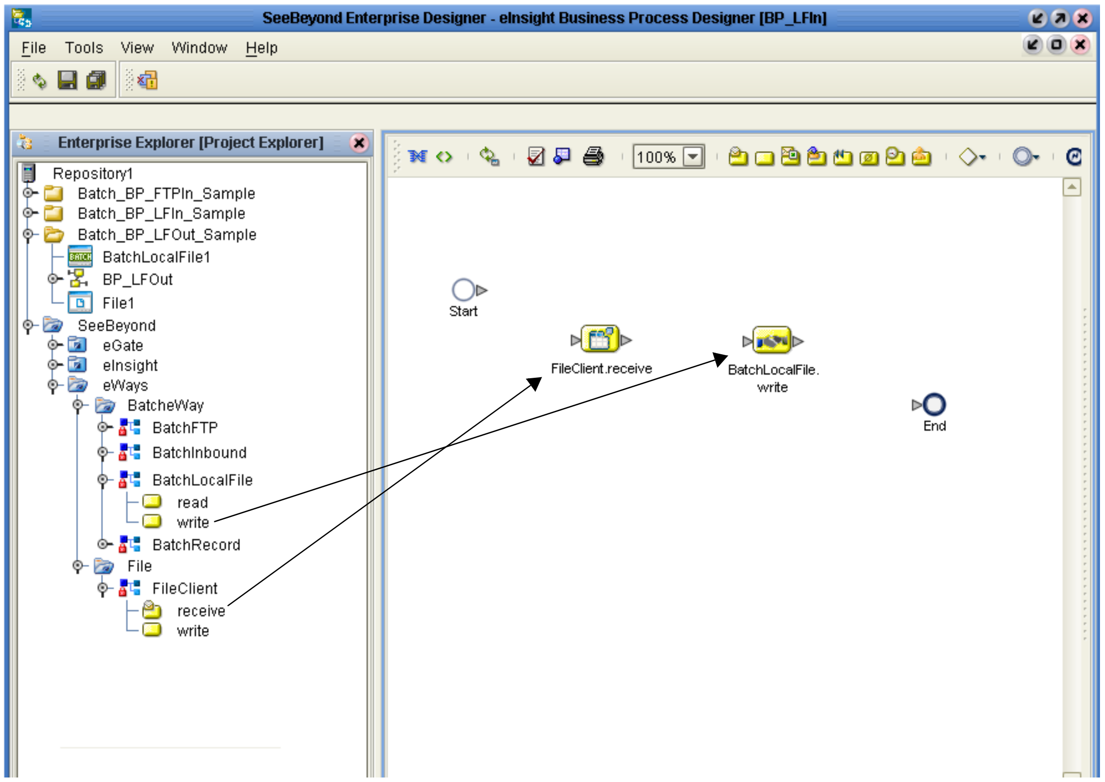
1. Start the Enterprise Designer.
2. From the Enterprise Explorer's Project Explorer tab, select your Repository on the Project Explorer tree, right-click the Repository and select **New Project**. A new Project appears on the Project Explorer tree.
3. Click twice on **Project1** and rename the Project (for this sample, **Batch_BP_LFOut_Sample**).

6.8.2 Creating the BP_LFOut Business Process

Creating the Business Process Flow

1. From the Enterprise Designer's Project Explorer tree, right-click **Batch_BP LFOut_Sample**, and select **New** > **Business Process** from the shortcut menu. The eInsight Business Process Designer appears and **BusinessProcess1** is added to the Project Explorer tree. Rename the Business Process to **BP_LFOut**.
2. Populate the eInsight Business Process Designer's modeling canvas with the following elements from the Project Explorer tree, as displayed in **Figure 62 on page 187**:
   - **receive**, under SeeBeyond > eWays > FileClient
   - **write**, under SeeBeyond > eWays > BatcheWay > BatchLocalFile

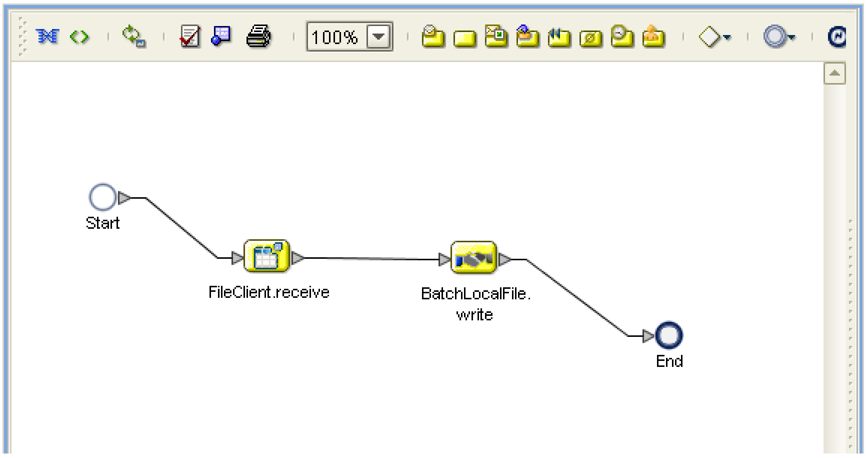
Figure 62 eInsight Business Process Designer - Populate the Canvas



3 Link the modeling elements by clicking on the element's connector and dragging the cursor to the next element's connector, making the following links as displayed in Figure 63.

- Start -> FileClient.receive
- FileClient.receive -> BatchLocalFile.write
- BatchLocalFile.write -> End

Figure 63 eInsight Business Process Designer - Link the Modeling Elements

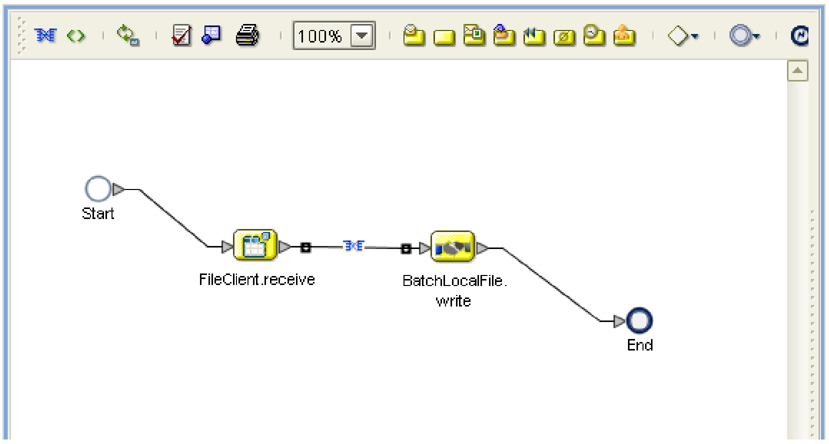


Configuring the Modeling Elements

Business Rules, created between the Business Process elements, define the relationship between the input and output attributes of the elements.

1 Right-click the link between the **FileClient.receive** and **BatchLocalFile.write** Activities and select **Add Business Rule** from the shortcut menu (see Figure 64).

Figure 64 eInsight Business Process Designer - Adding Business Rules



2 From the eInsight Business Process Designer toolbar, click the **Display Business Rules Designer** button. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.

3 Click on the **Business Rule** icon in the link between **FileClient.receive** and **BatchLocatFile.write** to display the Business Rule Output and Input Attributes in the Business Rule Designer. These Attributes can now be modified.

4 From the Business Rule Designer String menu, select **Settings**. The Method Palette appears. From the **String** tab of the Method Palette, select **text to bytes** and click **Close**. The **text to bytes** icon is added to the toolbar.

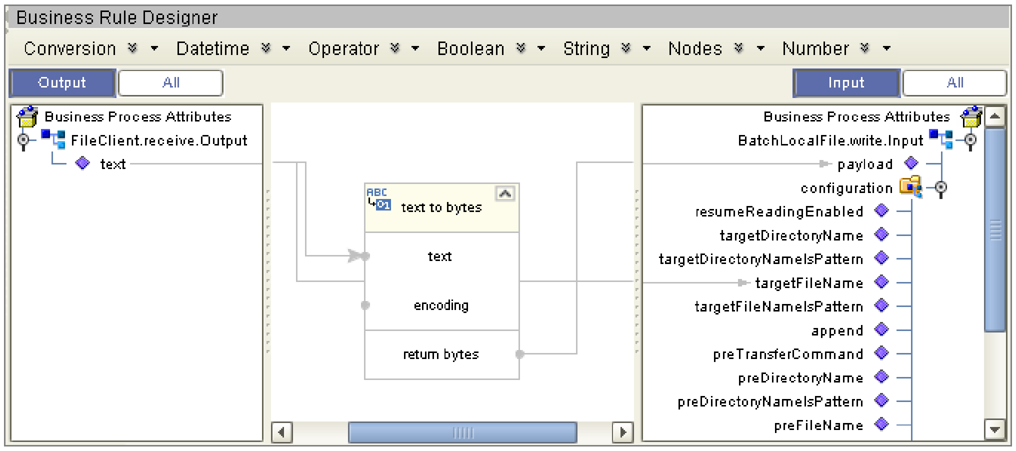
5 From the Business Rule Designer String menu, select **text to bytes**. The **bytes to text** method box appears.

6 Map **text,** under FileClient.receive.Output in the Output pane of the Business Rule Designer, to the **text** input node of the **text to bytes** method box. This is done by clicking on **text** and dragging the cursor to the **text** input node of the **text to bytes** method box.

7 Map the **return bytes** output node of the **text to bytes** method box, to bytes to **payload** under BatchLocalFile.write.input in the Input pane of the Business Rule Designer.

8 Map **text,** under FileClient.receive.Output in the Output pane of the Business Rule Designer, to **targetFileName** under Configuration > BatchLocalFile.write.input in the Input pane of the Business Rule Designer (see Figure 65 on page 189).

Figure 65 eInsight Business Rule Designer



9 When the Business Process is complete, from the Business Process Designer toolbar, click the **Synchronize Graphical Model and Business Process Code** icon to synchronize the graphical interface to the Business Process code.

10 Click the Enterprise Designer's **Save All** icon to your current changes.

6.8.3 Create a Connectivity Map

The Connectivity Map provides a canvas for configuring a Project's components.

1 In Enterprise Explorer's Project Explorer, right-click the new Project (**Batch_BP_LFOut_Sample**) and select **New > New Connectivity Map** from the shortcut menu.

2 The New Connectivity Map appears and a node for the Connectivity Map is added under the Project on the Project Explorer tree labeled **CMap1**. Rename the Connectivity Map **CM_Batch_BP_LFOut**.

Select the External Applications

1 Click the **External Application** icon on the Connectivity Map toolbar,

2 Select the applications needed for your Project (for this sample, **BatchLocalFile External Application** and **File External Application**). Icons representing the selected applications are added to the Connectivity Map toolbar.

Populate the Connectivity Map

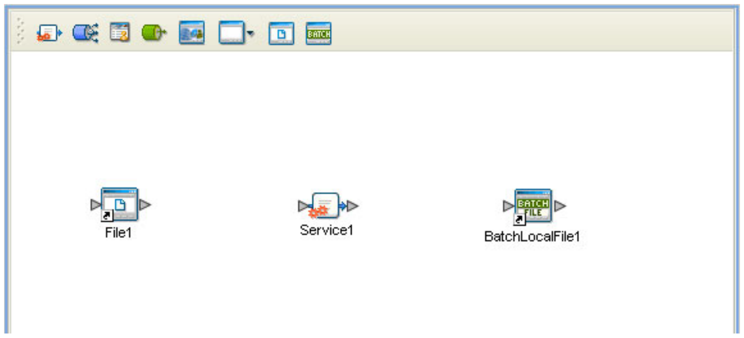
Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

1 For the Batch_BP_LFIn_Sample Project, drag the following components onto the Connectivity Map canvas as displayed in Figure 66 on page 190:

- File External System
- Service

- BatchLocalFile External System

Figure 66 CM_Batch_BP_LFIn Connectivity Map with Components



2 Rename **File1** to **FileIn1** by clicking the external application's name once and clicking it again. Enter the new name.

3 Rename **BatchLocalFile1** to **BatchLocalFileOut1**.

4 Rename **Service1** to **BP_LFOut1**.

5 Save your current changes to the Repository.

6.8.4 Binding the Project Components

Components are associated and the bindings are created in the Connectivity Map.

1 From the Project Explorer, double-click the Connectivity Map **CM_Batch_BP_LFOut**. The Enterprise Designer canvas now displays the **CM_Batch_BP_LFOut** Connectivity Map.

2 Drag and drop the **BP_LFOut** Business Process from the Project Explorer onto **BP_LFOut1** in the **CM_Batch_BP_LFOut** Connectivity Map.

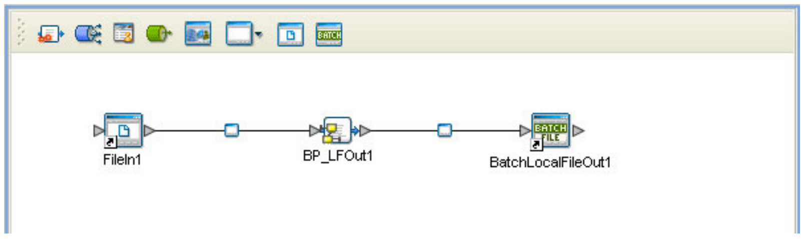
3 Double-click **BP_LFOut1**. The **BP_LFOut1** binding dialog box appears.

4 From the **BP_LFOut1** binding box, map **FileSender** (under Implemented Services) to the **FileIn1** External Application.

5 From the **BP_LFOut1** binding box, map the **BatchLocalFileReceiver** (under Invoked Services) to the **BatchLocalFileOut1** External Application.

6 Minimize the **BP_LFOut1** binding box. The Connectivity Map now appears as displayed in Figure 67.

Figure 67 Connectivity Map - Connecting the Project's Components



7 Save the current changes to your Repository.

6.8.5 Configuring the eWays

The Batch_BP_LFOut_Sample Project uses two eWays, each represented in the Connectivity Map as a node between an External Application and a Service.

The File eWay properties are configured from the Connectivity Map. The BatchLocalFile eWay, unlike the BatchFTP eWay, is only configured from the Project Explorer's Connectivity Map, and not from the Environment Explorer. To configure the eWays, do the following:

Configuring the File eWays

1. Double-click the **Inbound File eWay** and select **Inbound File eWay** in the Templates dialog box and click **OK**.
2. The Properties Sheet opens to the Inbound File eWay properties. Modify the properties for your system.

Configuring the BatchLocalFile eWay

The BatchLocalFile eWay properties are set from the Project Explorer's Connectivity Map. For more information on the BatchLocalFile eWay properties and the Properties Sheet, see **Creating and Configuring the Batch eWay** on page 21 or see the *eGate Integrator User's Guide*.

For the Batch_BP_LFOut_Sample Project, do the following:

Modifying the BatchLocalFile eWay Connectivity Map Properties

1. From the **Connectivity Map**, double-click the **BatchLocalFile** eWay. The Properties Sheet opens to the BatchLocalFile eWay properties.
2. Modify the BatchLocalFile eWay properties for your system.

6.8.6 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and JMS IQ Managers used by a Project and contain the configuration information for these components.

1. From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.
2. Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.
3. Rename the new Environment to **ENV_Batch_BP_LFOut**.
4. Right-click **ENV_Batch_BP_LFOut** and select **New BatchLocalFile External System**. Name the External System **BatchLocalFileExtSys**. Click **OK**. The **BatchLFExtSys** window is added to the Environment Editor.
5. Right-click **ENV_Batch_BP_LFOut** and select **New File External System**. Name this External System **FileExtSysIn** and select **Inbound File eWay** as the External System Type. The **FileExtSysIn** window is added to the Environment Editor.

6 Right-click **ENV_Batch_BP_LFOut** and select **New Logical Host**. The **LogicalHost1** window is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

7 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1**.

8 Save your current changes to the Repository.

6.8.7 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Collaborations and message destinations to the integration server and JMS IQ Manager. Deployment profiles are created using the Deployment Editor.

1 From the Enterprise Explorer's Project Explorer, right-click the Project (**Batch_BP_LFOut_Sample**) and select **New** > **Deployment Profile**.

2 Enter a name for the Deployment Profile (for this sample **DP_Batch_BP_LFOut**). Make sure that the selected Environment is **ENV_Batch_BP_LFOut**. Click **OK**.

3 From the Deployment Editor toolbar, click the **Auto Map** icon. The Projects components are automatically mapped to their system windows. If any of the Project components are not mapped automatically after Auto Map is used, those component can be mapped manually by following the appropriate steps below. Once all components are mapped, proceed to step 7.

4 From the left pane of the Deployment Editor, drag **FileIn1 -> BP_LEFOut1** (External Application) to the **FileExtSysIn** window.

5 From the left pane of the Deployment Editor, drag the **BP_LFOut1 -> BatchLocalFileOut1** (External Application) to the **BatchLocalFileExtSys** window.

6 From the left pane of the Deployment Editor, drag **BP_LFOut1** (Business Process) to **IntegrationSvr1** in the **LogicalHost1** window.

7 Click **Activate**. When activation succeeds, save the changes to the Repository.

6.8.8 Running the Project

To run the Project follow the directions provided in **"Running the Project" on page 177**.

Note: *For the Batch_BP_LFOut_Sample, make sure that the file content and target directory name are the same, including spaces, carriage returns, and so forth.*

6.9 The Batch_Security_Sample_BP Project

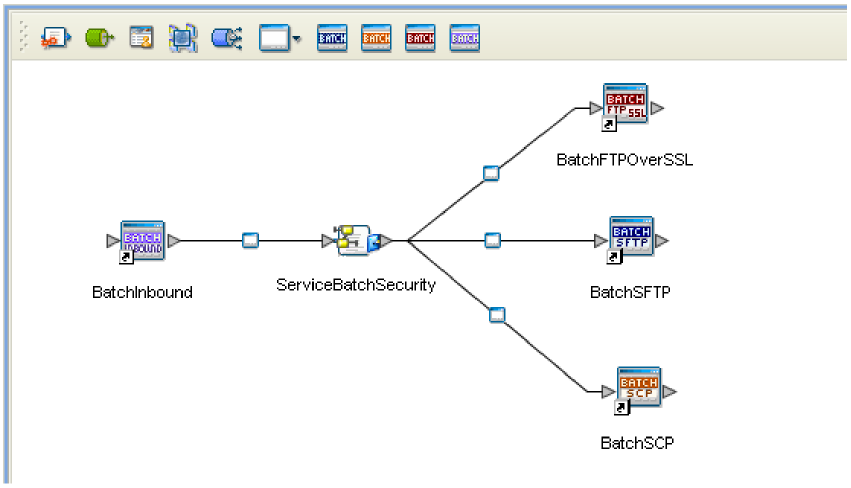
This section provides an overview of the Batch_Security_Sample_BP Project, and describes how to run the imported project. The sample is created in the same manner as the previous samples in this chapter.

The Batch_Security_Sample_BP Project contains one Business Process that employees four Batch OTDS that enable SSL or SSH secure file transfers.

The Batch_Security_Sample_BP Project Components

The Batch_Security_Sample_BP Project uses four External Applications, four component eWays, and one Java Collaboration, as seen in the Project's Connectivity Map (see Figure 68).

Figure 68 cmBatchSecurity Connectivity Map



These four Batch eWays perform the following functions

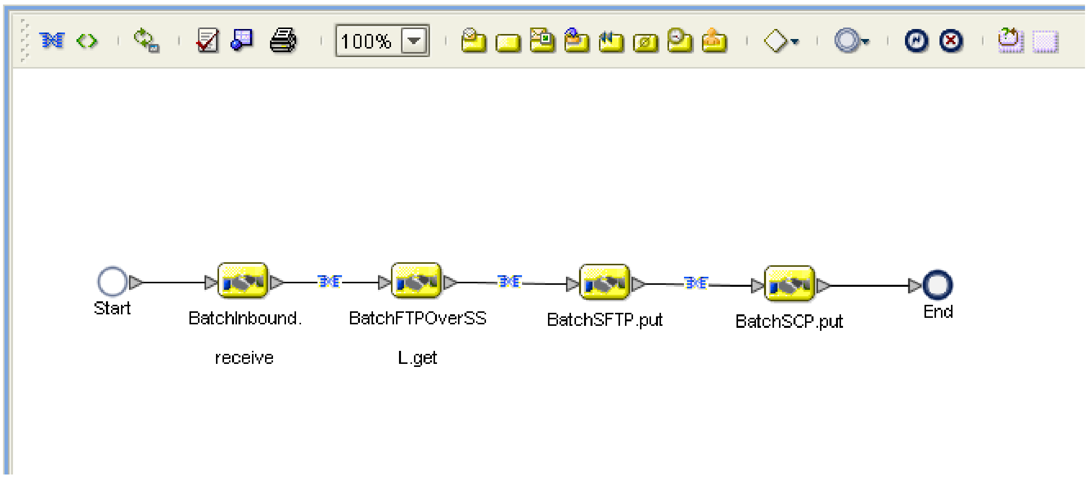
- **BatchInbound**: acts as a trigger for the Project. The BatchInbound eWay polls an external directory for a specific input file. When the input file is present, the BatchInbound eWay triggers the Business Process.
- **BatchFTPOverSSL**: "get" files from a remote directory and publish them to a local directory.
- **BatchSFTP**: "put" files from a local directory to a remote directory.
- **BatchSCP**: "put" files from a local directory to a remote directory.

6.9.1 The bpBatchSecurity Business Process

The Batch_Security_Sample_BP Project's **bpBatchSecurity** Business Process is created using the eInsight Business Process Designer. It uses the **BatchInbound receive,**

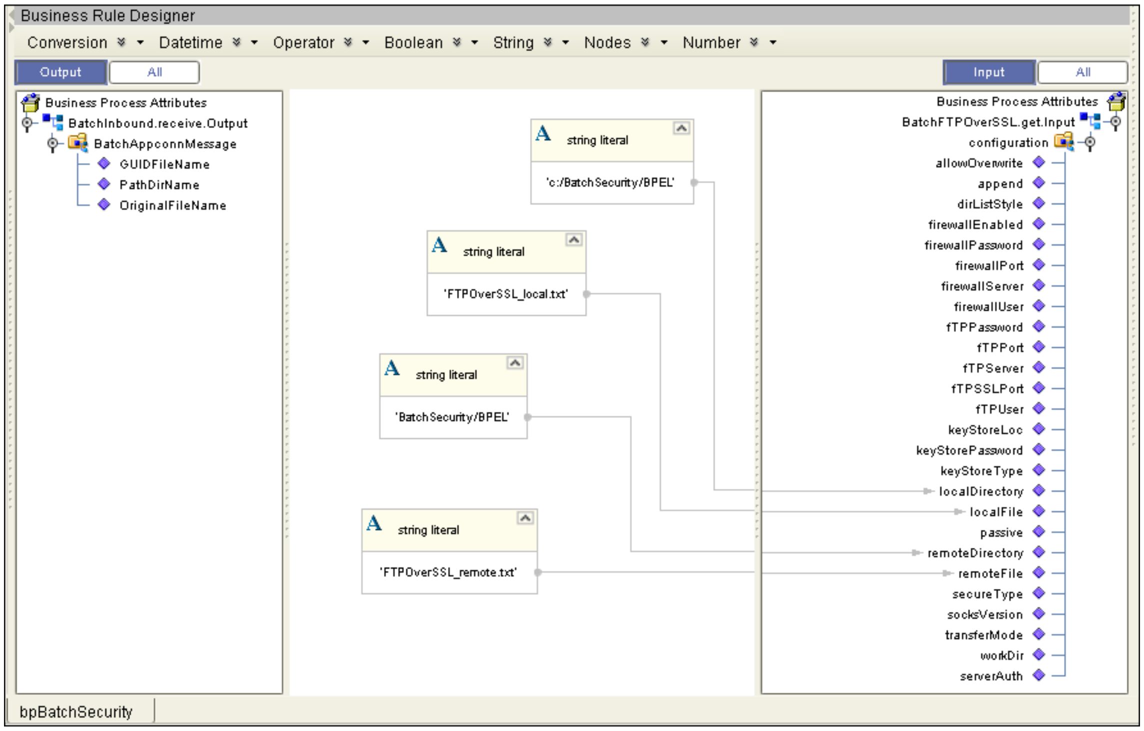
BatchFTPOverSSL.get, **BatchSFTP.put**, **BatchSCP.put**as the component Activities to create the Business Rules (see Figure 69).

Figure 69 bpBatchSecurity Business Process



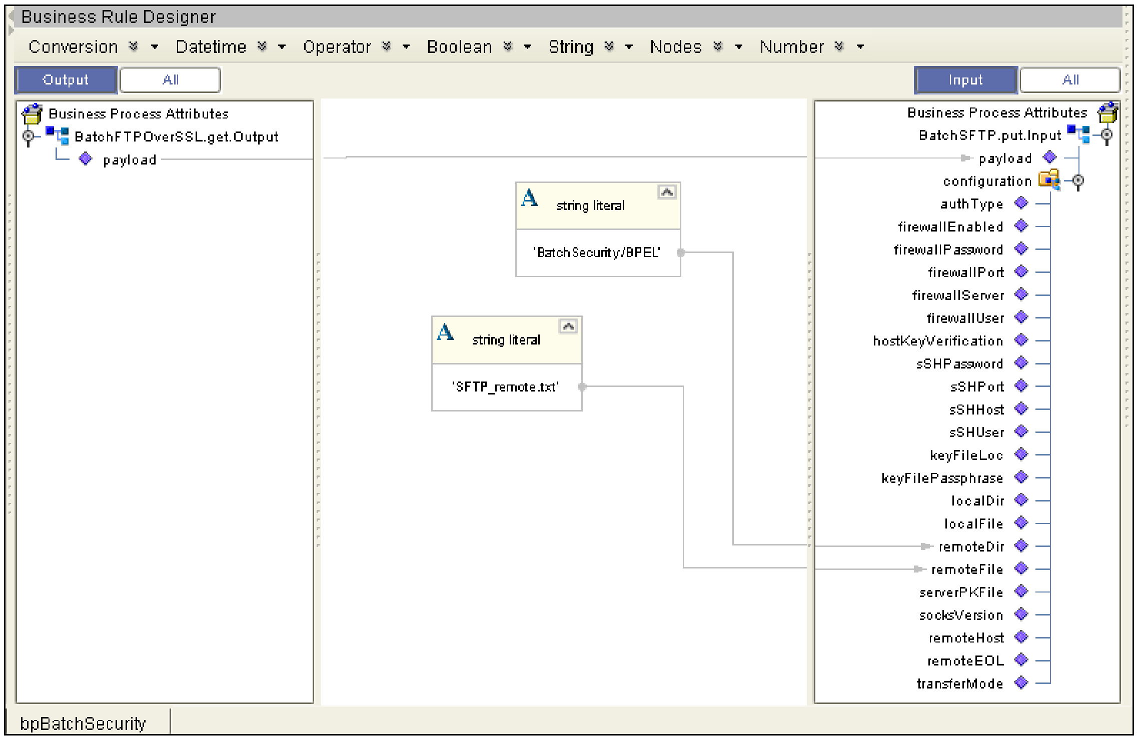
The bpBatchSecurity Business Process includes three Business Rules. The first, is located between the **BatchInbound.receive** and **BatchFTPOverSSL.get** Activities. This Business Rule, as displayed in the Business Rule Designer in Figure 70, when triggered by the BatchInbound eWay, "gets" a file from a remote directory and publishes it to a local directory.

Figure 70 BatchInbound.receive - BatchFTPOverSSL.get Business Rule



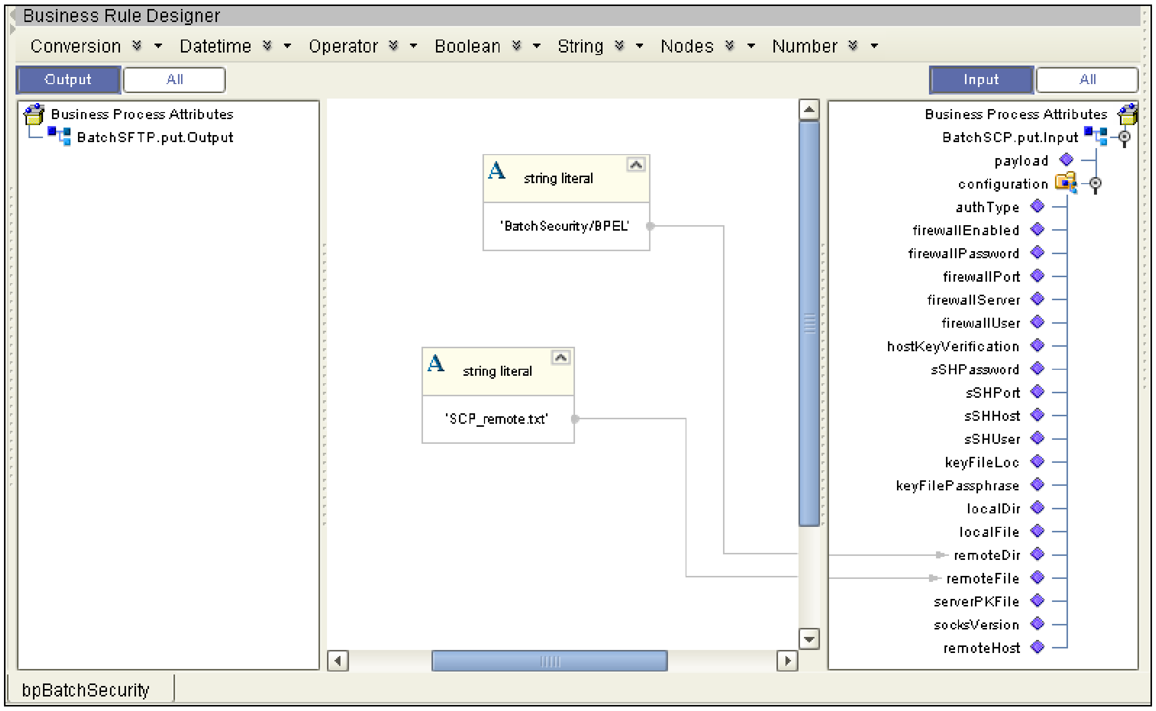
The second Business Rule, located between the **BatchFTPOverSSL.get** and the **BatchSFTP.put** Activities (see Figure 71 on page 195), "puts" a file from a local directory to a remote directory.

Figure 71 BatchFTPOverSSL.get - BatchSFTP.put Business Rule



The third Business Rule, located between the **BatchSFTP.put** and the **BatchSCP.put** Activities (see Figure 72), “puts” a file from a local directory to a remote directory.

Figure 72 BatchSFTP.put - BatchSCP.put Business Rule



6.9.2 Completing the Batch_Security_Sample_BP Project

Import the Batch_Security_Sample_BP Project as directed in **"Importing a Sample Project" on page 165**. After the sample Project has been imported and appears in your Project Explorer tree, Create the Project's Environment.

6.9.3 Creating an Environment

Environments include the External Systems, Logical Hosts, Integration Servers and Message Servers used by a Project and contain the configuration information for these components.

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **BP_BatchSecuritySample_Env**.

4 Right-click **BP_BatchSecuritySample_Env** and select **New BatchInbound External System**. Name this External System **BatchInboundExtSys**. The **BatchInboundExtSys** window is added to the Environment Editor.

5 Right-click **BP_BatchSecuritySample_Env** and select **New BatchFTPOverSSL External System**. Name this External System **BatchFTPOverSSLExtSys**. The **BatchFTPOverSSLExtSys** window is added to the Environment Editor.

6 Right-click **BP_BatchSecuritySample_Env** and select **New BatchSFTP External System**. Name this External System **BatchSFTPExtSys**. The **BatchSFTPExtSys** window is added to the Environment Editor.

7 Right-click **BP_BatchSecuritySample_Env** and select **New BatchSCP External System**. Name this External System **BatchSCPExtSys**. The **BatchSCPExtSys** window is added to the Environment Editor.

8 Right-click **BP_BatchSecuritySample_Env** and select **New Logical Host**. The **LogicalHost1 box** is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

9 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under Localhost1.

10 Save your current changes to the Repository.

6.9.4 Configuring the eWay Properties

The **Batch_Security_Sample_BP** Project uses four component eWays, each represented in the Connectivity Map as a node between an External Application and the Service.

The **BatchFTPOverSSL**, **BatchSFTP**, and **BatchSCP** component eWays contain properties which are configured from the Connectivity Map, and Environment Explorer. The **BatchInbound** component eWay only possess Environment properties.

Configure the eWay properties for your system. For more information about the various properties, see **"Batch eWay Properties" on page 24**.

Create the Sample Directories

The sample Project uses a number of directories. The default directories in the sample configuration properties are:

- The input directory (BatchInbound): **c:/BatchSecurity**
- **Local Directory** (BatchFTPOverSSL, BatchSCP, BatchSFTP): **c:/BatchSecurity**
- **Remote Directory** (BatchFTPOverSSL, BatchSCP, BatchSFTP): **BatchSecurity**

Create these directories on your system, or change the properties to reference your existing directories.

Create the Sample Files

The Sample also uses an input file named **BatchInbound.txt**. Create a text file with this name to use as a trigger file.

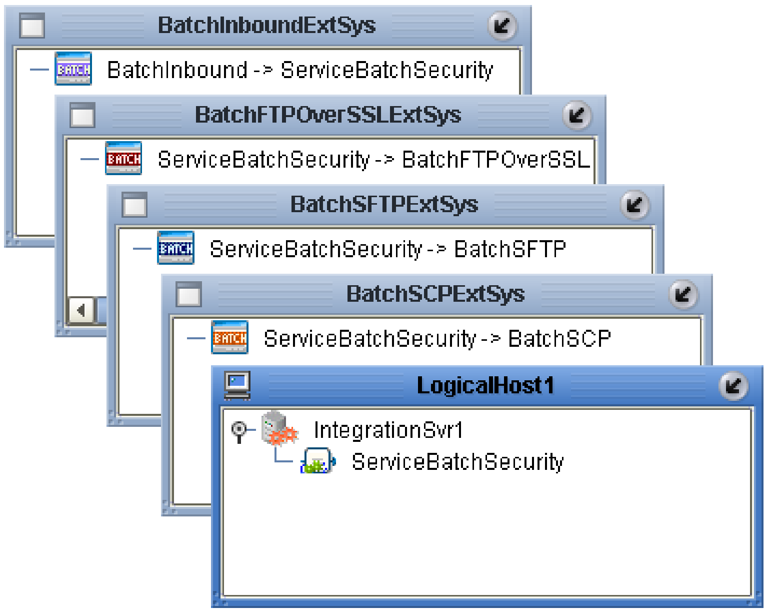
In addition, the sample uses a local file (the default name is **BatchSCP_local.txt**) and a remote file (the default name is **BatchSCP_remote.txt**). You can use any .txt files for these. For the remote file, rename your text file to **BatchSCP_remote.txt** and place it in the Remote Directory. For the local file, rename your text file to **BatchSCP_local.txt** and place it in the Local Directory.

6.9.5 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Collaborations and message destinations to the integration server and message server. Deployment profiles are created using the Deployment Editor.

1. From the Enterprise Explorer's Project Explorer, right-click the Project (**Batch_Security_Sample_BP**) and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **BP_BatchSecuritySample_DP**). Make sure that the selected Environment is **BP_BatchSecuritySample_Env**. Click **OK**.
3. From the Deployment Editor toolbar, click the **Auto Map** icon. The Projects components are automatically mapped to their system windows. If any of the Project components are not mapped automatically after Auto Map is used, drag and drop those components to the appropriate environment window, as displayed in Figure 73. Once all components are mapped, proceed to step 4.

Figure 73 BP_BatchSecuritySample_DP Deployment Profile


BatchInboundExtSys
BatchInbound -> ServiceBatchSecurity
BatchFTPOverSSLExtSys
ServiceBatchSecurity -> BatchFTPOverSSL
BatchSFTPExtSys
ServiceBatchSecurity -> BatchSFTP
BatchSCPExtSys
ServiceBatchSecurity -> BatchSCP
LogicalHost1
IntegrationSvr1
ServiceBatchSecurity

4 Click **Activate**. When activation succeeds, save Your current changes to the Repository.

6.9.6 Running the Project

To run the Project follow the directions provided in **"Running the Project" on page 177**.

Chapter 7

Implementing a Batch eWay Project

This chapter provides an introduction to the Batch eWay components used in a Collaboration based Project. This chapter assumes that you are already familiar with ICAN concepts and that you understand how to create a Project using the SeeBeyond Enterprise Designer.

This chapter presents four sample Batch eWay Projects created using the same procedures as the sample end-to-end Project provided in the *eGate Integrator Tutorial*.

For a complete explanation of ICAN terminology and concepts, see the *eGate Integrator User's Guide*.

What's in This Chapter

7.1 Batch eWay Components

eWay components that are unique to the Batch eWay include the following:

Batch eWay Configuration Files

The properties files for the Batch eWay contain the parameters used to connect with a specific external system. These properties are set using the Properties Sheet. For more information about the Batch eWay properties files and the Properties Sheet see **Creating and Configuring the Batch eWay** on page 21.

Batch eWay OTDs

Object Type Definitions (OTDs) map input and output message segments at the field level. The Batch eWay has the following seven OTDs:

- **BatchFTP OTD**: supports connections to external FTP servers.
- **BatchFTPOverSSL OTD**: supports secure data transfer using Secure Sockets Layer (SSL) protocol.
- **BatchSCP OTD**: supports secure data transfer using Secure Shell (SSH) protocol.
- **BatchSFTP OTD**: supports secure data transfer using Secure Shell (SSH) protocol.

- **BatchLocalFile OTD:** supports data file publish and subscribe functions for local file systems.
- **BatchRecord OTD:** provides functions for extracting records out of files, parsing files into specific records, and defining the content of files as records.
- **BatchInbound OTD**: provides functionality for receiving files, renaming files with GUID file names, and triggering a Business Process or Collaboration.

For information on the Batch eWay OTDs, see **Overview of the Batch OTDs** on page 124.

7.2 Considerations

The following items must be considered when implementing a Batch eWay Project:

- An error may occur upon activation for existing or imported Projects that use the **BatchLocalFile** eWay. This is due to a new BatchLocalFile Environment property that has been added to the eWay. To resolve this, refresh (open and close) your Project's BatchLocalFile Environment properties, and save the Project to the repository before activating the Project.
- When using FTP with an **AS400 UNIX** system, the following FTP configuration settings are required:
  - FTP - Use PASV: **No** (see **Use PASV** on page 33)
  - FTP Raw Commands - Pre Transfer Raw Commands: **site namefmt 1** (see **Pre Transfer Raw Commands** on page 35)

7.3 Importing a Sample Project

Sample eWay Projects are included as part of the installation CD-ROM package. To import a sample eWay Project to the Enterprise Designer do the following:

1. The sample files are first uploaded with the Batch eWay's documentation .sar file (**BatcheWayDocs.sar**) and downloaded from the Enterprise Manager's Documentation tab. Extract the samples from the Enterprise Manager to a local file.
2. Save all unsaved work before importing a Project.
3. From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import** from the shortcut menu. The **Import Manager** appears.
4. Browse to the directory that contains the sample Project zip file. Select the sample file (for example, **BatchInbound_Sample.zip**) and click **Import**. After the sample Project is successfully imported, click **Close**.
5. Before an imported sample Project can be run you must do the following:
   - Create an **Environment** (see **Creating an Environment** on page 213)

- Configure the eWays for your specific system (see **Configuring the eWays Properties** on page 214)
- Create a **Deployment Profile** (see **Creating and Activating the Deployment Profile** on page 217)

7.4 The Batch eWay Sample JCD Projects

This chapter provides step by step directions for manually creating the sample Batch eWay Projects that use Java Collaboration Definitions.

- **The Batch_FTPIn_LFOut_Sample Project** on page 203,
- **The Batch_RecParseStream_Sample Project** on page 219
- **The BatchInbound_Sample Project** on page 228
- **The Batch_Security_Sample_JCD Project** on page 237
- **The Batch_Stream_Get Sample Project** on page 243
- **The Batch_Stream_Put Sample Project** on page 247

Sample data files

Sample data files for the Batch eWay Projects are included with the samples. See Input_Files_Readme.txt included with the sample data files for more information.

7.4.1 Sample Project Descriptions

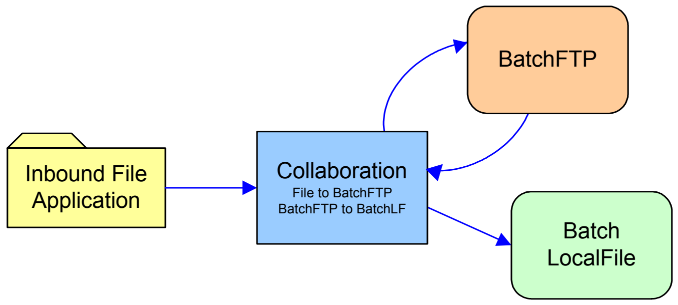
The Batch_FTPIn_LFOut_Sample Project

The Batch_FTPIn_LFOut_Sample Project demonstrates the following:

- The Batch eWay, using a File eWay, locates and selects a file from a local directory.
- The Collaboration (Java) invokes the Batch FTP eWay to get the payload file, populates the payload with data, and then puts the data into a local file.
- The Collaboration uses three Business Rules to convert incompatible data types by doing the following:
  - Unmarshaling data from a String
  - Converting String data to byte array
  - Populating a payload with the converted data and writing it to a local file

Figure 74 shows the data flow of the Batch eWay sample Project.

Figure 74 Batch_FTPIn_LFOut_Sample Project



The Batch_RecParseStream_Sample Project

The **Batch_RecParseStream_Sample** Project demonstrates the following:

- The eGate Scheduler prompts the BatchLocalFile eWay to pick up a file from an external directory.
- The data is converted from bytes to text and published to the file eWay.
- The File eWay publishes the data file to an external directory.

The BatchInbound_Sample Project

The **BatchInbound_Sample** Project demonstrates the following:

- The BatchInbound eWay polls the input directory periodically. When it sees a specified file, it renames the file with a GUID file name, and triggers the Collaboration.
- The BatchLocalfileIn eWay gets the file with the GUID file name.
- The files contents is copied from BatchLocalfileIn payload to the FileOut payload.
- The FileOut eWay writes the payload to an output file.

The Batch_Security_Sample_JCD Project

The **Batch_Security_Sample_JCD** Project uses the BatchFTPOverSSL, BatchSFTP, and BatchSCP OTDs to demonstrate the SSL and SSH secure file transfer functions of the eWays.

The project uses a BatchInbound eWay to poll an external directory for a specific input file. When the file is present it triggers the Collaboration to do the following:

FTP Over SSL

- Gets a file from a remote directory and publishes it to a local directory.
- Publishes a file from local directory to a remote directory.
- Downloads a file from a remote directory to a local directory.
- Uploads a file from a local directory to remote directory.
- Creates remote directories and list them.

- Deletes remote directories and list the results.

SFTP

- Gets a file from remote directory and publishes it to a local directory.
- Creates a new remote directory
- Changes directories to the newly created remote directory, and publishes a file there.
- Renames the published file.
- Deletes an unwanted file from remote directory.

SCP

- Recursively gets a file from a remote directory and publishes it to a local directory.
- Recursively puts a file from local directory to a remote directory.

The Batch_Stream_Get Sample Project

The **BATCH_STREAM_005** Project demonstrates how to download a remote file to a local file using data-streaming. The following occurs in this sample Project:

- The BatchInbound eWay component scans the directory for the trigger file.
- The BatchLocalFile eWay component sends a byte stream out to the BatchFTP eWay component using the OutputStreamAdapter of the BatchLocalFile eWay component.
- The Client element of the BatchFTP consumes the stream adapter. In this case, the **com.stc.eways.batchext.FtpFileClient** instance invokes the **get()** method to complete the data transfer.

The Batch_Stream_Put Sample Project

The **Batch_Stream_Put** Project demonstrates how to upload a remote file to a local file using data-streaming. The following occurs in this sample Project:

- The BatchInbound eWay component scans the directory for the trigger file.
- The BatchLocalFile eWay component sends a byte stream out to the BatchFTP eWay component using the InputStreamAdapter of the BatchLocalFile eWay component.
- The Client element of the BatchFTP consumes the stream adapter. In this case, the **com.stc.eways.batchext.FtpFileClient** instance invokes the **put()** method to complete the data transfer.

7.5 The Batch_FTPIn_LFOut_Sample Project

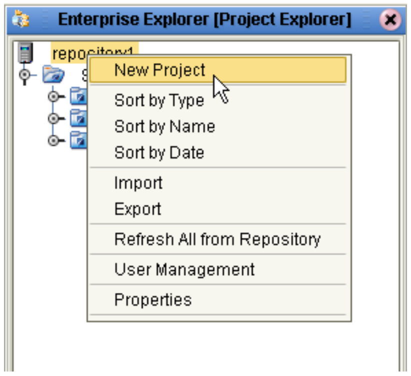
This section describes how the components of the Batch_FTPIn_LFOut_Sample Project are created. To manually create the Batch_FTPIn_LFOut_Sample Project, do the following:

7.5.1 Create a Project

The first step is to create a new Project in the SeeBeyond Enterprise Designer.

1. Start the Enterprise Designer.
2. From the Enterprise Explorer's Project Explorer tab, right-click the Repository and select **New Project** (see Figure 75). A new Project (Project1) appears on the Project Explorer tree.

Figure 75 Enterprise Explorer - New Project



3. Rename the Project (for this sample, **Batch_FTPInLFOut_Sample**).

7.5.2 Creating a Connectivity Map

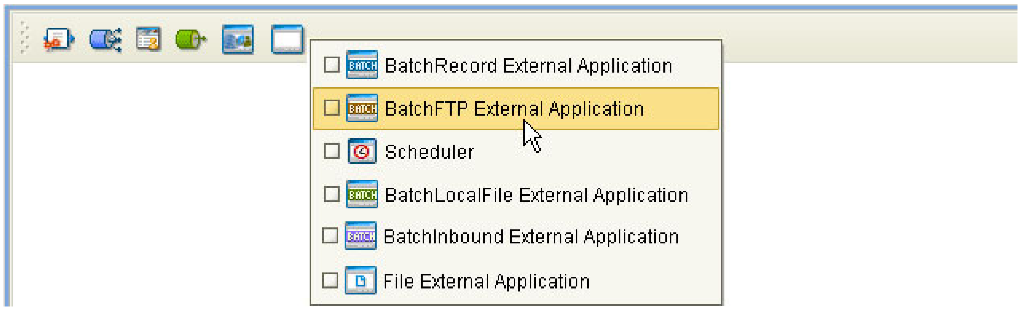
A Connectivity Map provides a canvas for assembling and configuring a Project's components.

1. In the Enterprise Explorer's Project Explorer, right-click **Batch_FTPInLFOut_Sample** and select **New > Connectivity Map** from the shortcut menu.
2. The new Connectivity Map appears and adds a node on the Project Explorer tree labeled **CMap1**.

Selecting the External Applications

In the Connectivity Map, eWays are associated with External Applications. For example, to establish a connection to an external FTP server, you must first select BatchFTP as an External System to use in the Connectivity Map (see Figure 76).

Figure 76 Connectivity Map - External Applications


BatchRecord External Application
BatchFTP External Application
Scheduler
BatchLocalFile External Application
BatchInbound External Application
File External Application

1 Click the **External Application** icon on the Connectivity Map toolbar.

2 Select the External Applications for your Project. For this sample, select **File External Application**, **BatchFTP External Application**, and **BatchLocalFile External Application**. This adds icons representing the **File** and **BatchFTP** and **BatchLocalFile** External Application icons to the Connectivity Map toolbar.

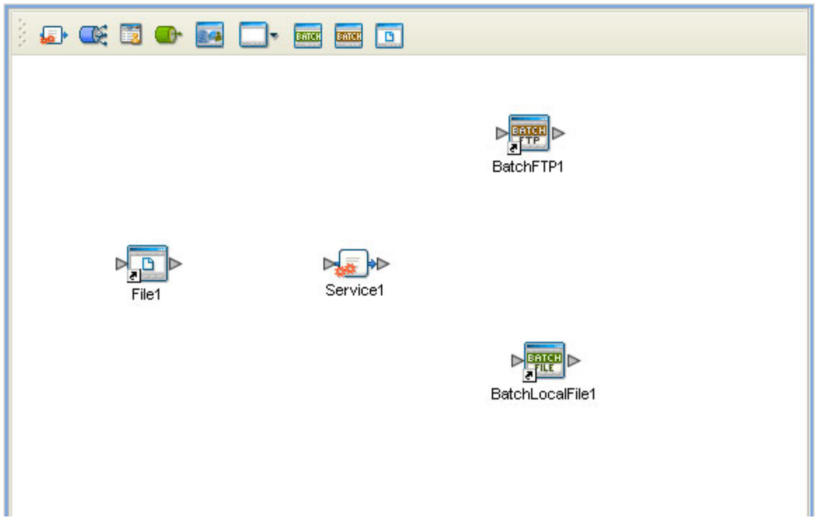
7.5.3 Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the Connectivity Map toolbar to the canvas.

1 For the **Batch_FTPInLFOut_Sample** sample, drag and drop the following components onto the Connectivity Map canvas as displayed in **Figure 77** on page 206:

- File External Application
- Service (container for the Collaboration)
- BatchLocalFile External Application
- BatchFTP External Application

Figure 77 Batch_FTPInLFOut_Sample Connectivity Map



2 Rename the items in the Connectivity Map by right-clicking the item, selecting Rename from the shortcut menu, and typing the new name in. Rename the items as follows:

- **File1** to **FileIn1**
- **BatchFTP1** to **BatchFTPIn1**
- **BatchLocalFile1** to **BatchLocalFileOut1**
- **Service1** to **CollabFTPInLFOut**

7.5.4 Creating Collaboration Definitions

The next step in the sample Project is to create the **jcd_FTPInLFOut** Collaboration Definition. A Collaboration Definition contains Business Rules that define the processing and transport of data between eGate components.

The Collaboration Definition Wizard (Java) is used to create the Java Collaboration Definitions. Once a Collaboration is created, the Collaboration Editor is used to create the Business Rules of the Collaboration.

Create the jcd_FTPInLFOut Collaboration

The **jcd_FTPInLFOut** Collaboration defines how data is transferred between the FileIn application, the Inbound BatchFTP application and the Outbound BatchLocalFile.

1 From the Project Explorer, right-click **Batch_FTPInLFOut_Sample** and select **New** > **Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.

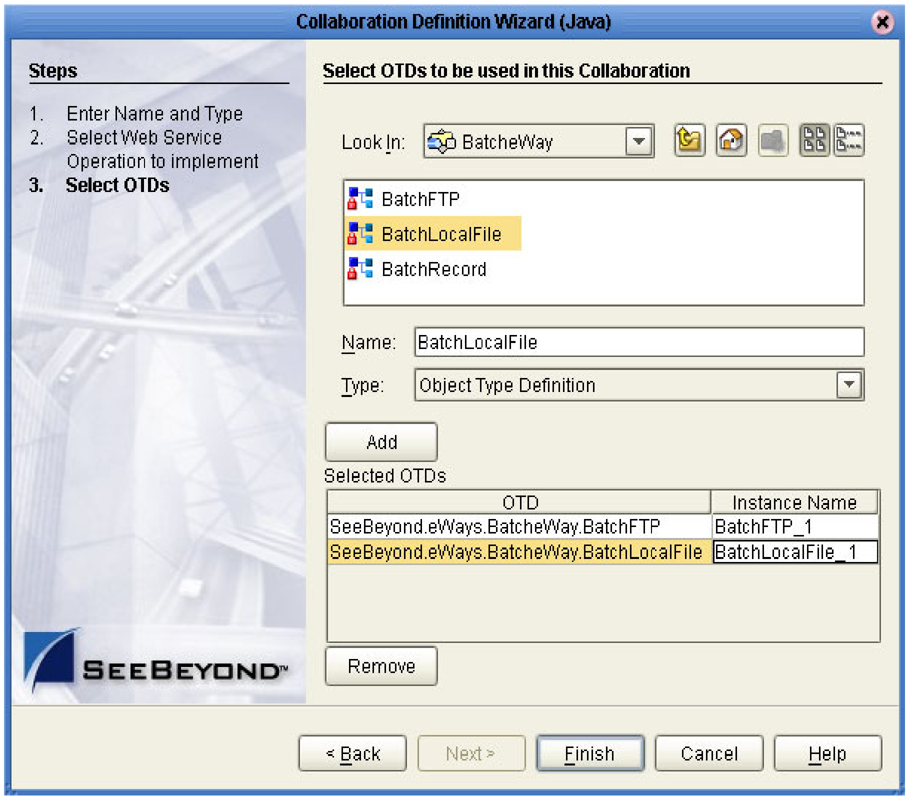
2 Enter a Collaboration Definition name (for this sample **jcd_FTPInLFOut**) and click **Next**.

3 For Step 2 of the Wizard, double-click **SeeBeyond** > **eWays** > **File** > **FileClient** > **receive**. The File Name field now displays **receive**. Click **Next**.

4 For Step 3 of the Wizard, from the Select OTDs selection window, double-click **SeeBeyond** > **eWays** > **BatcheWay > BatchFTP**. The **BatchFtp** OTD is added to the Selected OTDs field.

5 From the Select OTDs selection window, double-click **BatchLocalFile** (SeeBeyond > eWays > BatcheWay > BatchLocalFile). The **BatchLocalFile** OTD is added to the Selected OTDs field (see **Figure 78** on page 207).

Figure 78 Collaboration Definition Wizard (Java) - jcd_FTPInLFOut - Select OTDs



Name: BatchLocalFile

Type: Object Type Definition

Selected OTDs

OTD	Instance Name
SeeBeyond.eWays.BatcheWay.BatchFTP	BatchFTP_1
SeeBeyond.eWays.BatcheWay.BatchLocalFile	BatchLocalFile_1

6 Click **Finish**. The new **jcd_FTPInLFOut** Collaboration appears in the Project Explorer tree.

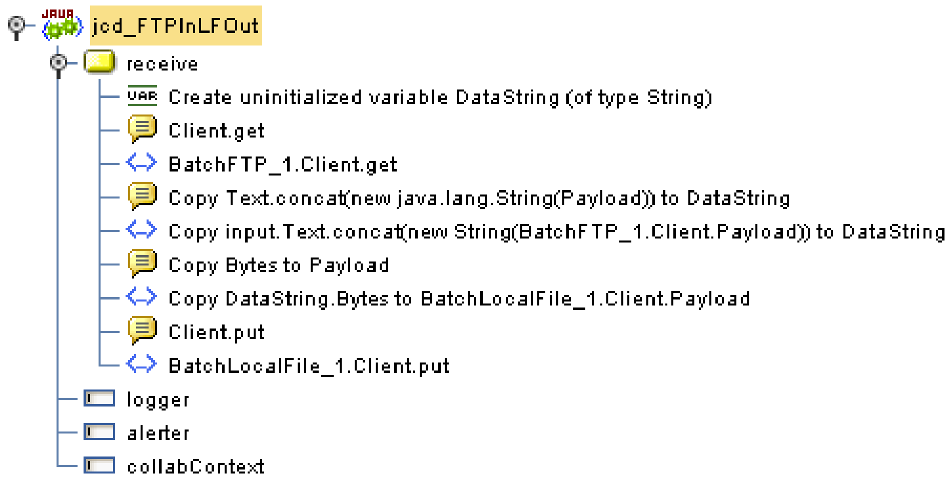
7.5.5 Using the Collaboration Editor (Java)

The **Batch_FTPInLFOut_Sample** Project uses the **jcd_FTPInLFOut** Collaboration created in the previous section. To complete the Collaboration, use the Collaboration Editor (Java) to create the Business Rules.

Create the jcd_FTPInLFOut Collaboration Business Rules

Be careful to open all nodes specified in the directions to connect the correct items. The **jcd_FTPInLFOut** Collaboration contains the Business Rule displayed in **Figure 79** on page 208.

Figure 79 jcd_FTPInLFOut Business Rules



```
jcd_FTPInLFOut
  receive
    Create uninitialized variable DataString (of type String)
    Client.get
    BatchFTP_1.Client.get
    Copy Text.concat(new java.lang.String(Payload)) to DataString
    Copy input.Text.concat(new String(BatchFTP_1.Client.Payload)) to DataString
    Copy Bytes to Payload
    Copy DataString.Bytes to BatchLocalFile_1.Client.Payload
    Client.put
    BatchLocalFile_1.Client.put
  logger
  alerter
  collabContext
```

To create the **jcd_FTPInLFOut** Collaboration Business Rules, do the following:

1 From the Project Explorer tree, double-click **jcd_Passthru**. The Collaboration Editor (Java) opens to the jcd_Passthru Collaboration.

2 To create comments for the Business Rules, from the Business Rules toolbar, click the **comment** icon. The **Enter a Comment** dialog box appears. Enter the comment and click **OK**. The comment is placed on the Business Rules tree under the last selected item. Once the Comment is created, it can be moved by clicking the comment and dragging it up or down the Business Rules tree to a new location.

3 To create the **Create uninitialized variable DataString (of type String)** variable rule do the following:

   A From the Business Rules toolbar, click the **Local Variable** icon. The **Create Variable** dialog box appears.

   B From the Create Variable dialog box, Enter **DataString** as the **Name**. For **Type**, select **Class** and click the ellipsis (...) button. The **Class Browser** dialog box appears.

   C From the Class Browser dialog box, select **String** under **All Classes** and click **Select**. From the Create Variable dialog box, click **OK**. The new variable is added to the Business Rules tree.

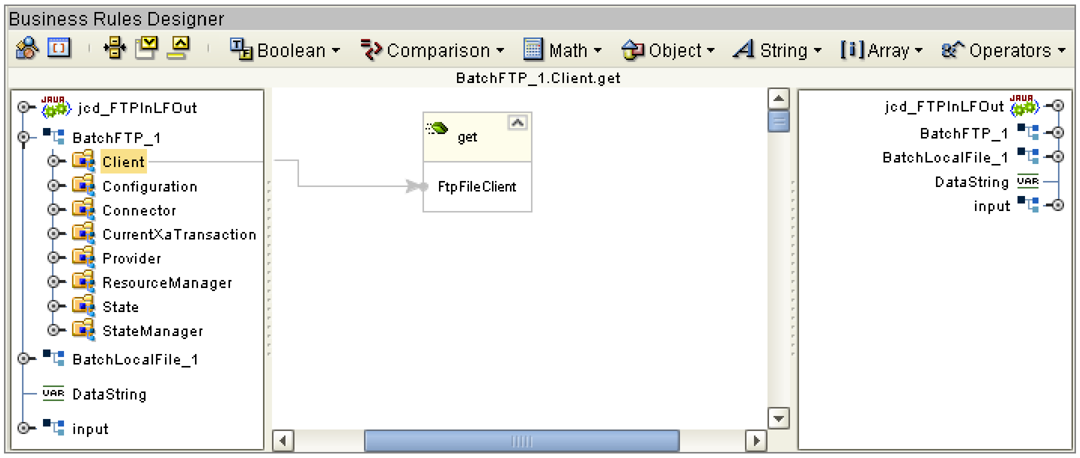
4 To create the **BatchFTP_1.Client.get** rule do the following:

   A Click **rule** on the Business Rules toolbar to add a new rule in the Business Rules pane.

   B Right-click **Client** under the **BatchFTP_1** node in the left pane of the Business Rules Designer, and select **Select method to call** from the shortcut menu. The method selection window appears.

   C Select **get()** from the method selection window. The get method box appears in the Business Rules Designer canvas (see Figure 80).

Figure 80 Collaboration Editor (Java) - Business Rules Designer



5 To create the **Copy input.Text.concat(new String(BatchFTP_1.Client.Payload)) to DataString** rule do the following:

A From the Business Rules Designer String menu, select **concat**. The **concat** method box appears.

B From the Business Rules Designer toolbar, click the Class Browser icon. The Class Browser dialog box appears. From the Class Browser dialog box, select **String** under **All Classes**, select **String(byte[] bytes)** under **String**, and click **Select**. The String method box appears.

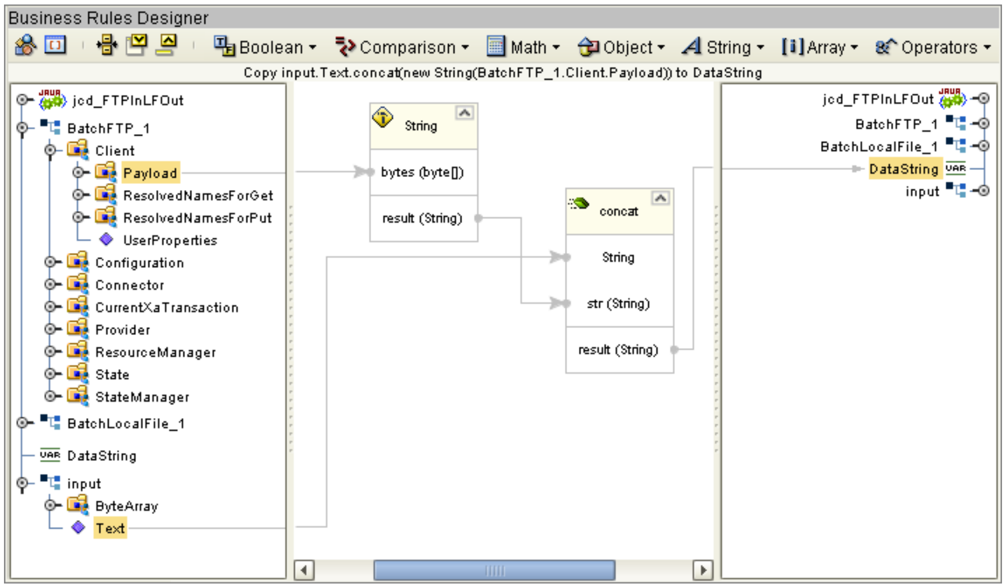
C Map **Payload** under BatchFTP_1 > Client in the left pane of the Business Rules Designer, to the **bytes (byte[])** input node of the **String** method box.

D Map **Text** under **input** in the left pane of the Business Rules Designer, to the **String** input node of the concat method box.

E Map the **result (String)** output node of the **String** method box to the **str (String)** input node of the **concat** method box.

F Map the **result (String)** output node of the **concat** method box to **DataString** in the right pane of the Business Rules Designer (see **Figure 81** on page 210).

Figure 81 Collaboration Editor (Java) - Business Rules Designer



6 To create the **Copy DataString.Bytes to BatchLocalFile_1.Client.Payload** rule do the following:

A Click **rule** on the Business Rules toolbar to add a new rule in the Business Rules pane.

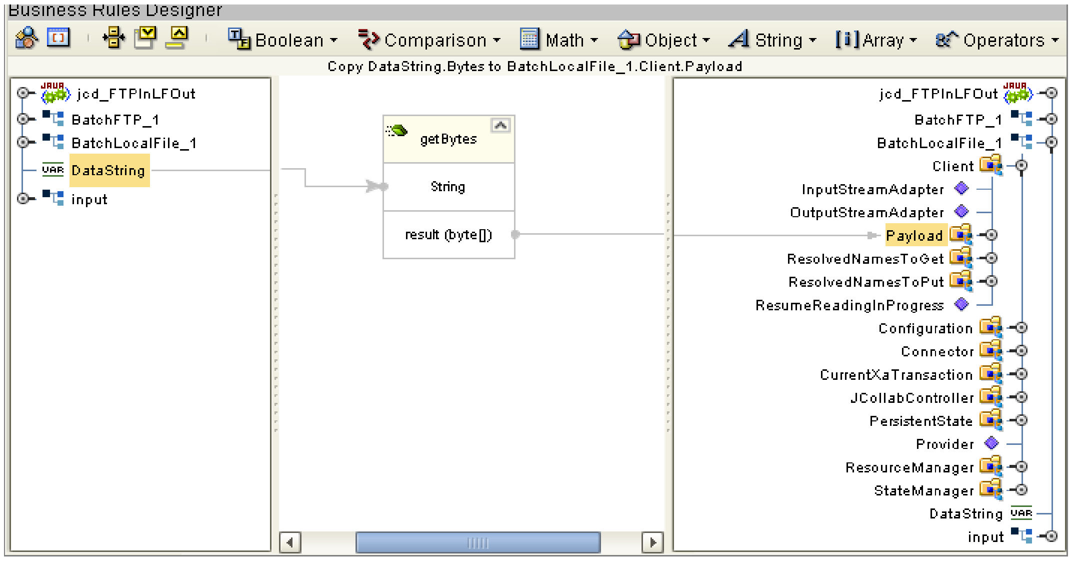
B Right-click **DataString** in the left pane of the Business Rules Designer and click **Select method to call** from the shortcut menu.

C Select **getBytes()** from the method selection window. The **getBytes** method box appears.

Note: *For eGate 5.0.4, the getBytes() method may not be available from the method selection window. To create this method, click the Call Java Method icon on the Business Rules Designer toolbar. From the Call Method dialog box, select String as the value for All Classes, select getBytes() as the Methods value, and click OK. The getBytes method box appears. Map DataString in the left pane of the Business Rules Designer, to the String input node of the getBytes method box.*

D Map the **result (byte[])** output node of the **getBytes** method box, to **Payload** under **BatchLocalFile_1 > Client** in the right pane of the Business Rules Designer. The Editor now appears as displayed in Figure 82 on page 211.

Figure 82 Collaboration Editor (Java) - Business Rules Designer



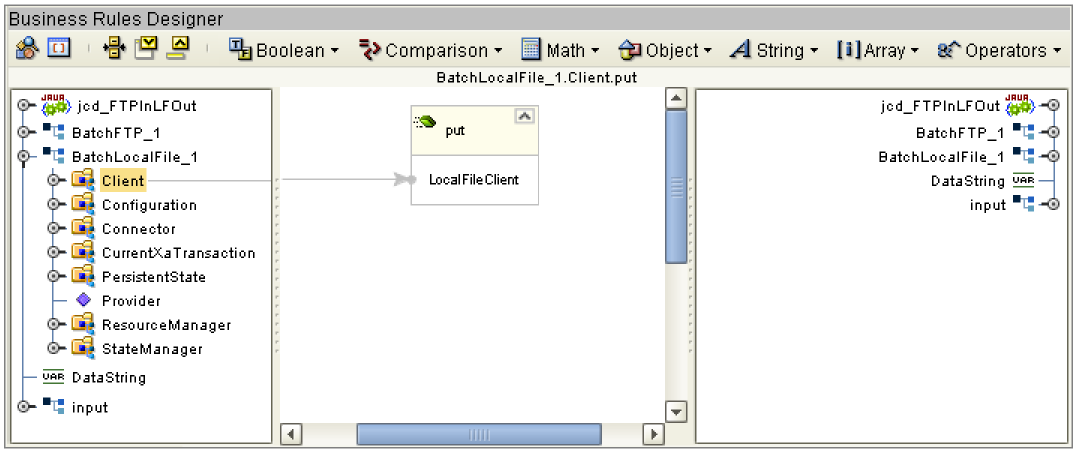
7 To create the **BatchLocalFile_1.Client.put** rule do the following:

A Click **rule** on the Business Rules toolbar to add a new rule in the Business Rules pane.

B .Right-click **Client** under the **BatchLocalFile_1** node in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The method selection window appears.

C Select **put()** from the method selection window. The put method box appears in the Business Rules Designer canvas (see Figure 83).

Figure 83 Collaboration Editor (Java) - Creating Business Rules



8 From the editor's toolbar, click **Validate** to check the Collaboration for errors.

9 Save your current changes to the Repository.

Note: *See the eGate Integrator User's Guide for more information on editing Collaborations.*

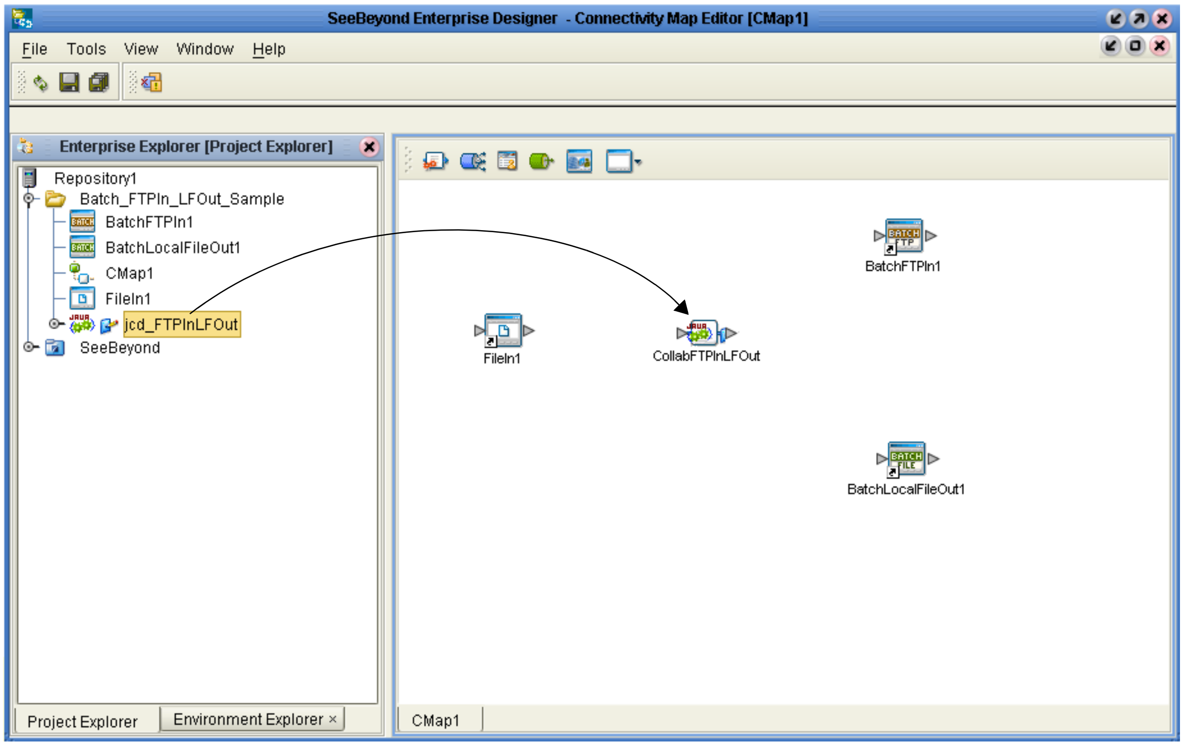
7.5.6 Creating Collaboration Bindings

After the Collaboration is complete, associate the components and create Collaboration Bindings by connecting the components in the Connectivity Map.

Create the Collaboration Bindings

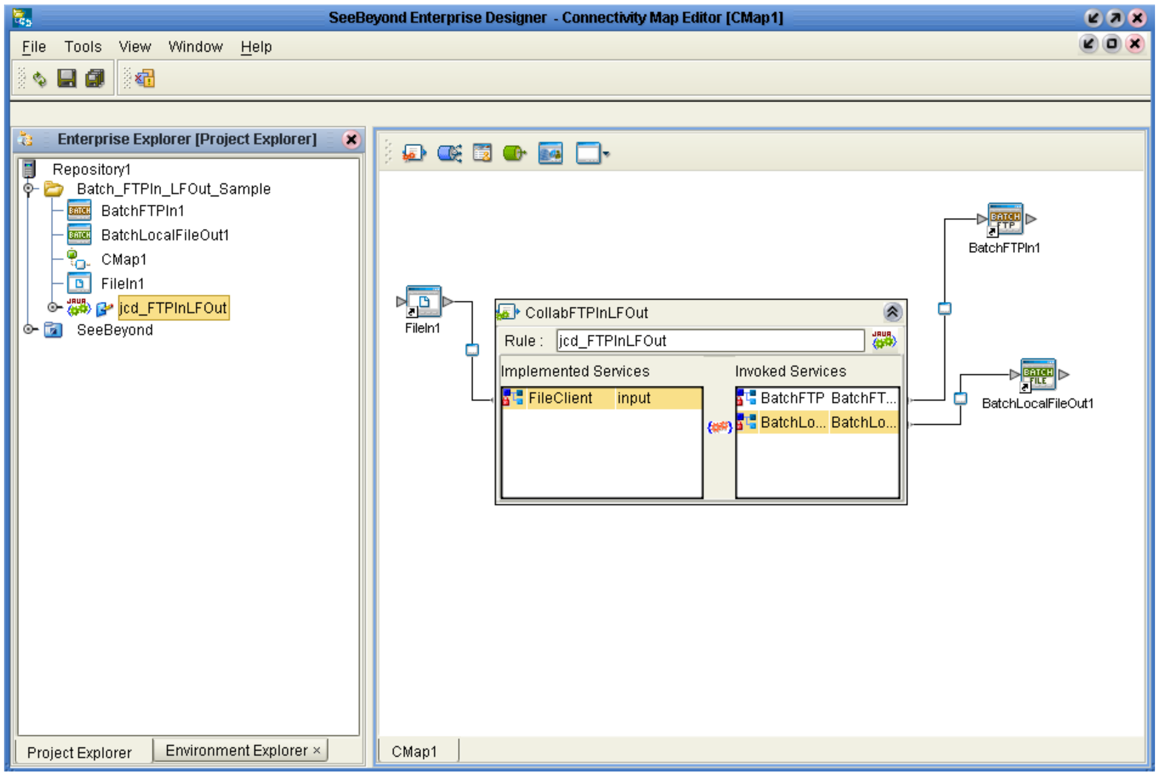
1. From the Project Explorer, double-click the Connectivity Map **CMap1**.
2. Drag and drop the **jcd_FTPInLFOut** Collaboration from the Project Explorer onto **Collaboration1**. If the Collaboration is successfully associated the "gears" icon changes from red to green.
3. From the Project Explorer, drag and drop the **jcd_FTPInLFOut** Collaboration, onto **CollabFTPInLFOut** in the connectivity Map canvas (see Figure 84).

Figure 84 Connectivity Map - Binding the Project components



4. From the connectivity Map, double-click **CollabFTPInLFOut**. The **CollabFTPInLFOut** binding dialog box opens with **jcd_FTPInLFOut** as the rule.
5. From the Connectivity Map, map **FileClient input** under Implemented Services in the **CollabFTPInLFOut** binding dialog box, to **FileIn1**.
6. Map **BatchFTP** under Invoked Services in the **CollabFTPInLFOut** binding dialog box, to **BatchFTPIn1**.
7. Map **BatchLocalFileOut1** under Invoked Services in the **CollabFTPInLFOut** binding dialog box, to **BatchLocalFileOut1 (see** Figure 85**)**.

Figure 85 Connectivity Map - Binding the Project components



8 Minimize the **CollabFTPInLFOut** binding dialog box and save your changes to the Repository.

7.5.7 Creating an Environment

Environments include the External Systems, Logical Hosts, Integration Servers and Message Servers used by a Project and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Explorer and Environment Editor.

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **ENV_Batch_FTPInLFOut**.

4 From the Project Explorer tree, right-click **ENV_Batch_FTPInLFOut** and select **New File External System**. Name this External System **FileExtSysIn** and select **Inbound File eWay** as the External System Type. The **FileExtSysIn** window is added to the Environment Editor.

5 From the Project Explorer tree, right-click **ENV_Batch_FTPInLFOut** and select **New BatchFTP External System**. Name the External System **BatchFTPExtSys**. Click **OK**. The **BatchFTPExtSys** window is added to the Environment Editor.

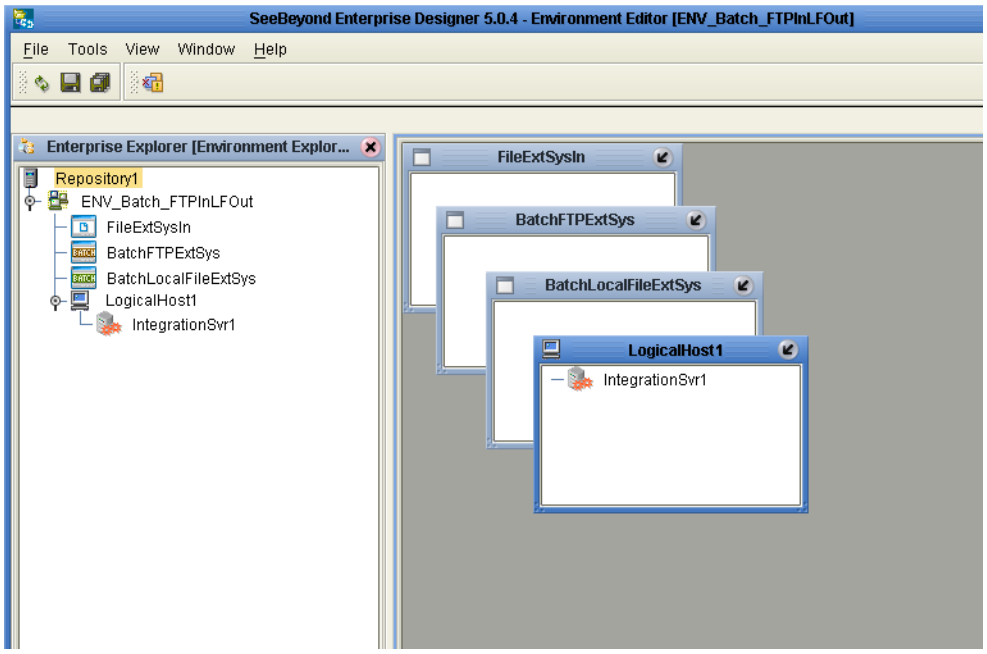
6 From the Project Explorer tree, right-click **ENV_Batch_FTPInLFOut** and select **New BatchLocalFile External System**. Name the External System **BatchLocalFileExtSys**. Click **OK**. The **BatchLocalFileExtSys** window is added to the Environment Editor.

7 From the Project Explorer tree, right-click **ENV_Batch_FTPInLFOut** and select **New Logical Host**. The **LogicalHost1 box** is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

8 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1**.

9 Save changes to the Repository. The Environment Explorer and Environment Editor now appear as displayed in Figure 86.

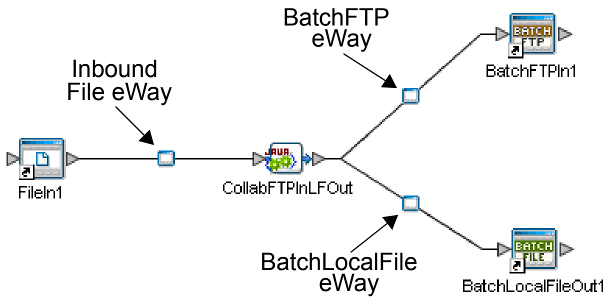
Figure 86 Environment Editor



7.5.8 Configuring the eWays Properties

The **ENV_Batch_FTPInLFOut_Sample** Project contains three eWays, each represented in the Connectivity Map as a node between an External Application and a Collaboration. The eWays facilitate communication and movement of data between the external applications and the eGate system (see **Figure 87** on page 215).

Figure 87 eWay Properties



The **File eWay** and the **BatchLocalFile eWay** properties are accessed and set from the Connectivity Map. The **BatchFTP eWay** properties must be set from both the Project Explorer's Connectivity Map and the Environment Explorer tree. To configure the eWays do the following:

Configuring the File eWay Properties

1. Double-click the **Inbound File eWay**, select **Inbound File eWay** in the Templates dialog box and click **OK**.
2. The Properties Sheet opens to the Inbound File eWay properties. Modify the properties for your system, including the settings for the **Inbound File eWay** in Table 9, and click **OK**. The properties are saved for the eWay.

Table 9 Inbound File eWay Properties

Inbound File eWay Properties	
Directory	C:/temp
Output file name	output%d.dat

Configuring the BatchLocalFile eWay Properties

1. Double-click the **BatchLocalFile eWay**. The Properties Sheet opens to the Outbound File eWay properties.
2. Modify the properties for your system, including the settings in Table 10, and click **OK**. The properties are saved for the eWay.

Table 10 BatchLocalFile eWay Settings

BatchLocalFile eWay Properties	
Append	No
Target Directory Name	The directory on the file system where files are retrieved or sent.
Target Directory Name is Pattern	No
Target File Name	The name of the file to be retrieved or sent.
Target File Name is Pattern	No

Configuring the BatchFTP eWay Properties

The BatchFTP eWay properties must be set in both Connectivity Map and Environment Explorer. For more information on the BatchFTP eWay properties and the Properties Sheet, see **Creating and Configuring the Batch eWay** on page 21 or see the *eGate Integrator User's Guide*.

Modifying the BatchFTP eWay Connectivity Map Properties

1 From the Connectivity Map, double-click the **BatchFTP** eWay. The Properties Sheet opens to the BatchFTP eWay Connectivity Map properties.

2 Modify the **BatchFTP** eWay (Project Explorer) configuration for your system, including the settings in Table 11, and click **OK**.

Table 11 BatchFTP eWay Connectivity Map Properties

BatchFTP eWay Connectivity Map Properties	
Target Location Set as directed, otherwise use the default settings	
Target Directory Name	The directory on the external system (absolute path) from which files are retrieved or sent
Target File Name	The FTP remote file name which is retrieved from or sent to

Modifying the BatchFTP eWay (Environment Explorer) Properties

1 From the **Environment Explorer** tree, right-click the BatchFTP External System (**BatchFTPExtSys** in this sample), and select **Properties**. The Properties Sheet opens to the BatchFTP eWay environment-configuration properties.

2 Modify the BatchFTP eWay environment-configuration properties for your system, including the settings in Table 12, and click **OK**.

Table 12 BatchFTP eWay Environment Explorer Properties

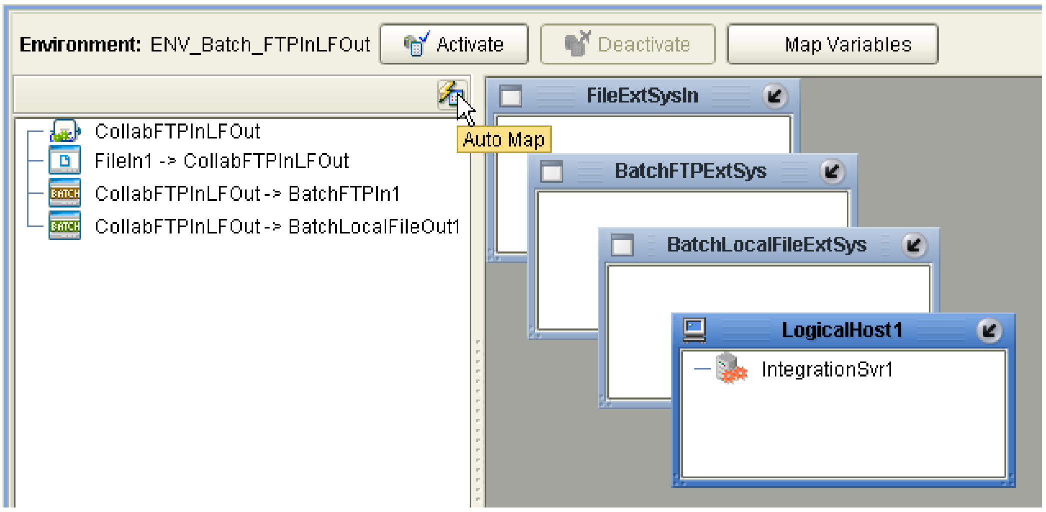
BatchFTP eWay Environment-Configuration Parameters	
FTP Set as directed, otherwise use the default settings.	
Host Name	*The name of the external system to which the eWay connects*
Password	*Password required to log into the external system*
Server Port	*Port number to use to connect to the FTP server*
User Name	*User ID used to login to the external system*

7.5.9 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Collaborations and message destinations to the integration server and message server. Deployment profiles are created using the Deployment Editor.

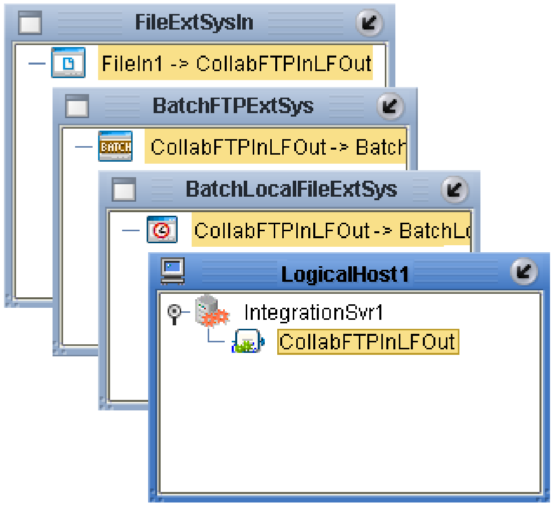
1. From the Enterprise Explorer's Project Explorer, right-click the Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **DP_Batch_FTPInLFOut**). Make sure that the selected Environment is **ENV_Batch_FTPInLFOut**. Click **OK**.
3. Click the **Auto Map** icon as displayed in Figure 88. The Projects components are automatically mapped to their system windows as seen in **Figure 89** on page 218. If any of the Project components are not mapped automatically after Auto Map is used, those component can be mapped manually by following the appropriate steps below. Once all components are mapped, proceed to step 8.

Figure 88 Deployment Profile - Auto Map


Environment: ENV_Batch_FTPInLFOut
Activate
Deactivate
Map Variables
Auto Map
CollabFTPInLFOut
FileIn1 -> CollabFTPInLFOut
CollabFTPInLFOut -> BatchFTPIn1
CollabFTPInLFOut -> BatchLocalFileOut1
FileExtSysIn
BatchFTPExtSys
BatchLocalFileExtSys
LogicalHost1
IntegrationSvr1

4. From the left pane of the Deployment Editor, drag the **FileIn1 -> CollabFTPInLFOut** (External Application) to the **FileExtSysOut** window.
5. From the left pane of the Deployment Editor, drag the **CollabFTPInLFOut -> Batch_FTPIn** (External Application) to the **BatchFTPExtSys** window.
6. From the left pane of the Deployment Editor, drag the **CollabFTPInLFOut -> Batch_LocalFileOut** (External Application) to the **BatchLocalFileExtSys** window.
7. From the left pane of the Deployment Editor, drag **CollabFTPInLFOut** (Collaboration) to **IntegrationSvr1** in the **LogicalHost1** window (see Figure 89).

Figure 89 Deployment Profile



8 Click **Activate**. When activation succeeds, save your current changes to the Repository.

7.5.10 Running the Project

The following directions assume that the Enterprise Designer was downloaded to **C:\ican50**. If this is not the case, replace that location in the following directions with the appropriate location.

1 From the Enterprise Manager Downloads tab, download **Logical Host - for win32**.

2 Extract the file to the **ican50** directory.

3 Navigate to **C:\ican50\logicalhost\bootstrap\config** directory and open the **logical-host.properties** file using a text editor.

4 Enter the following information in the appropriate fields:

- Logical Host root directory: **ican50\LogicalHost1\logicalhost**
- Repository URL: **http://localhost:*port number/repository name***
- Repository user name and password: ***Your user name and password***
- Logical Host Environment name: ***Your Project's Environment name***
- Logical Host name: **logicalhost**

Save your changes to **logical-host.properties** and close the file.

5 Run the **bootstrap.bat** file in the **ican50\logicalhost\bootstrap\bin** directory.

6 Copy the sample input data file to the input directory.

Note: *For UNIX, be sure to upload the appropriate LogicalHost*

7.6 The Batch_RecParseStream_Sample Project

This section describes how the components of the Batch_RecParseStream_Sample Project are created. To create the Project manually, do the following:

7.6.1 Create a Project

The first step is to create and name a new Project in eGate Enterprise Designer.

1. Start the Enterprise Designer.
2. From the Enterprise Explorer's Project Explorer tab, select your Repository on the Project Explorer tree, right-click the Repository and select **New Project**. A new Project appears on the Project Explorer tree.
3. Click twice on **Project1** and rename the Project (for this sample, **Batch_RecParseStream_Sample**).

7.6.2 Create a Connectivity Map

The Connectivity Map provides a canvas for configuring a Project's components.

1. In Enterprise Explorer's Project Explorer, right-click the new Project (**Batch_RecParseStream_Sample**) and select **New > Connectivity Map** from the shortcut menu.
2. The New Connectivity Map appears and a node for the Connectivity Map is added under the Project on the Project Explorer tree labeled **CMap1**. Rename the Connectivity Map **CM_RecParseStream_Sample**.

Select the External Applications

1. Click the **External Application** icon on the Connectivity Map toolbar,
2. Select the following applications for your Project:
   - (2) File External Application
   - BatchLocalFile External System
   - BatchRecord External System
3. Icons representing the selected applications are added to the Connectivity Map toolbar.

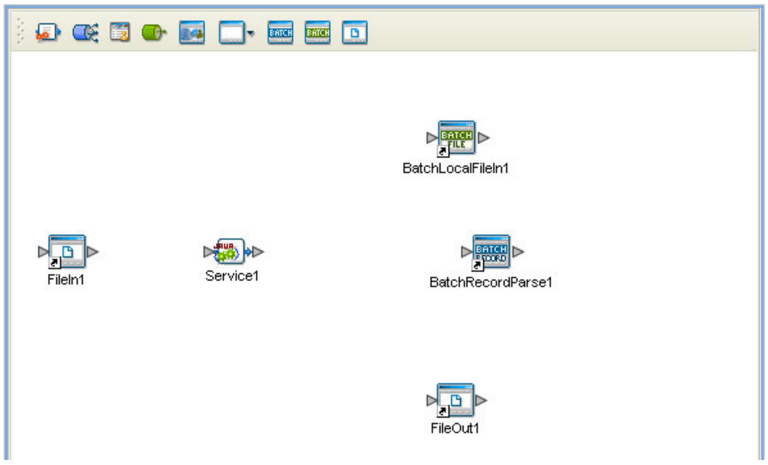
Populate the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

1. For the Batch_RecParseStream_Sample Project, drag the following components onto the Connectivity Map canvas, and name each as displayed in Figure 90:
   - (2) File External Application - **FileIn1**, **FileOut1**

- Service - **Service1**
- BatchLocalFile External System - **BatchLocalFileIn1**
- BatchRecord External System - **BatchRecordParse1**

Figure 90 CM_RecParseStream_Sample Connectivity Map with Components



2 Save your current changes to the Repository.

7.6.3 Creating a Collaboration Definition

The next step in the sample is to create a Java Collaboration using the Collaboration Definition Wizard (Java). Once a Collaboration Definition has been created, the Business Rules of the Collaboration are written using the Collaboration Editor.

The jcd_RecParse_Stream Collaboration

1 From the Project Explorer, right-click the **Batch_RecParseStream_Sample** Project and select **New > Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.

2 Enter a Collaboration Definition name (for this sample **jcd_RecParse_Stream**) and click **Next**.

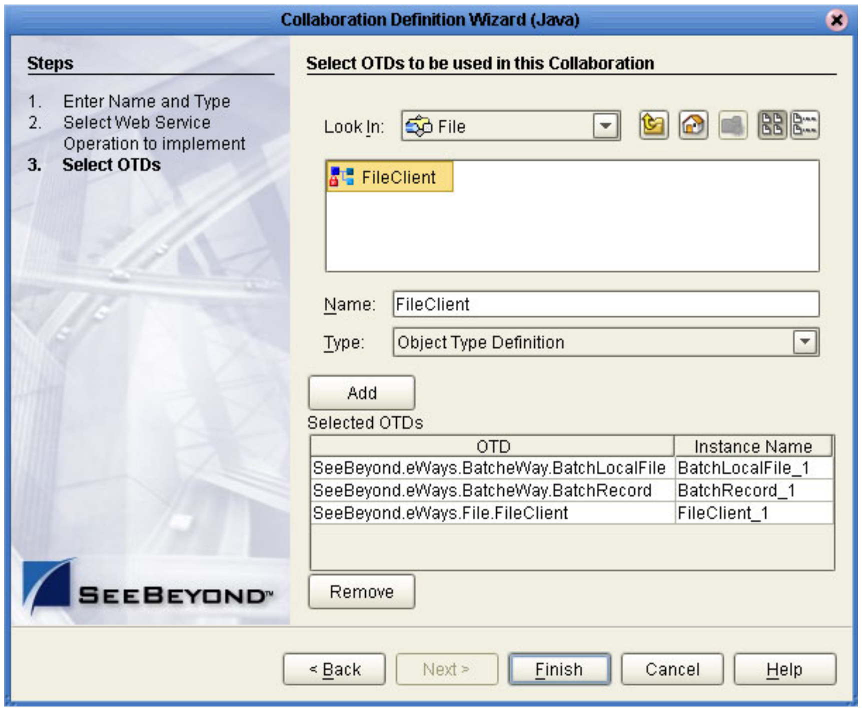
3 For Step 2 of the Wizard, from the Web Services Interfaces selection window, double-click **SeeBeyond** > **eWays** > **File** > **FileClient** > **receive**. The File Name field now displays **receive.** Click **Next**.

4 For Step 3 of the Wizard, from the Select OTDs selection window, double-click **SeeBeyond** > **eWays** > **BatcheWay** > **BatchLocalFile**. The **LocalFile** OTD is added to the Selected OTDs field.

5 From the Select OTDs selection window, double-click **SeeBeyond** > **eWays** > **BatcheWay > BatchRecord**. The **BatchRecord** OTD is added to the Selected OTDs field.

6 Click the **Up One Level** button to return to the Repository directory. Double-click **SeeBeyond** > **eWays** > **File** > **FileClient**. The **FileClient** OTD is added to the Selected OTDs field.

Figure 91 Collaboration Definition Wizard (Java) - Select OTDs


Collaboration Definition Wizard (Java)
Steps
1. Enter Name and Type
2. Select Web Service Operation to implement
3. Select OTDs
SEEBEYOND
Select OTDs to be used in this Collaboration
Look In:
File
FileClient
Add
Selected OTDs
Remove
< Back
Next >
Finish
Cancel
Help

Name: FileClient

Type: Object Type Definition

OTD	Instance Name
SeeBeyond.eWays.BatcheWay.BatchLocalFile	BatchLocalFile_1
SeeBeyond.eWays.BatcheWay.BatchRecord	BatchRecord_1
SeeBeyond.eWays.File.FileClient	FileClient_1

7 Click **Finish**. The Collaboration Editor (Java) opens to the new Collaboration in the right pane of the Enterprise Designer.

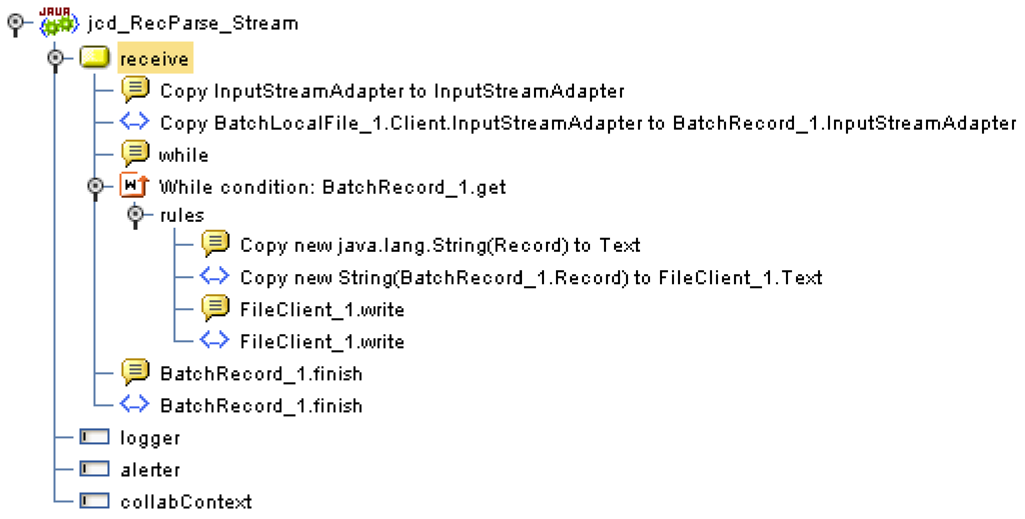
7.6.4 Using the Collaboration Editor (Java)

The **Batch_FTPInLFOut_Sample** Project uses the **jcd_FTPInLFOut** Collaboration created in the previous section. To complete the Collaboration, use the Collaboration Editor to create the Business Rules.

Create the jcd_RecParse_Stream Collaboration Business Rules

Be careful to open all nodes specified in the directions to connect the correct items. The **jcd_RecParse_Stream** Collaboration contains the Business Rule displayed in **Figure 92** on page 222.

Figure 92 jjcd_RecParse_Stream Business Rules



```
jcd_RecParse_Stream
  receive
    Copy InputStreamAdapter to InputStreamAdapter
    Copy BatchLocalFile_1.Client.InputStreamAdapter to BatchRecord_1.InputStreamAdapter
    while
    While condition: BatchRecord_1.get
      rules
        Copy new java.lang.String(Record) to Text
        Copy new String(BatchRecord_1.Record) to FileClient_1.Text
        FileClient_1.write
        FileClient_1.write
    BatchRecord_1.finish
    BatchRecord_1.finish
  logger
  alerter
  collabContext
```

To create the **jcd_RecParse_Stream** Collaboration Business Rules, do the following:

1 From the Project Explorer tree, double-click **jcd_RecParse_Stream**. The Collaboration Editor (Java) opens to the **jcd_RecParse_Stream** Collaboration.

2 To create comments for the Business Rules, from the Business Rules toolbar, click the **comment** icon. The **Enter a Comment** dialog box appears. Enter the comment and click **OK**. The comment is placed on the Business Rules tree under the last selected item. Once the Comment is created, it can be moved by clicking the comment and dragging it up or down the Business Rules tree to a new location.

3 To create the **Copy BatchLocalFile_1.Client.InputStreamAdapter to BatchRecord_1.InputStreamAdapter** rule do the following:

   A Map **InputStreamAdapter** under BatchLocalFile_1 > Client in the left pane of the Business Rules Designer, to **InputStreamAdapter** under BatchRecord_1 in the right pane of the Business Rules Designer. This is done by clicking on and dragging InputStreamAdapter to the destination InputStreamAdapter, and releasing the cursor.

4 To create the **while** statement do the following:

   A Click **while** on the Business Rules toolbar to add a new while statement in the Business Rules pane.

   B Right-click **BatchRecord_1** the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The **method selection** window appears.

   C Select **get()** from the method selection window. The get method box appears in the Business Rules Designer canvas.

5 To create the **Copy new String(BatchRecord_1.Record) to FileClient_1.Text** rule under the **while** statement do the following:

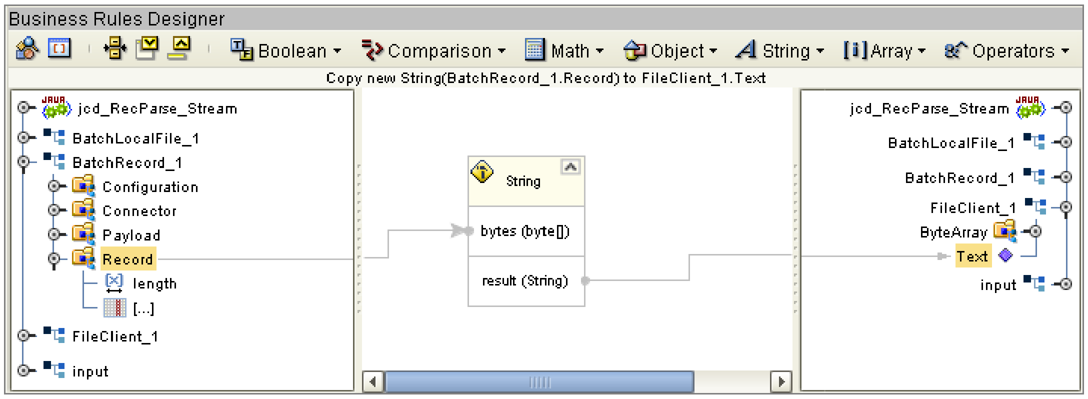
   A Select the **Copy new java.lang.String(Record) to Text** rule under the **while** statement in the Business Rules pane.

B From the Business Rules Designer toolbar, click on the **Class Browser** icon. The **Class Browser** dialog box appears. Select **String** under **All Classes** and **String(byte[] bytes)** under **String**. Click **Select**. The **String** method box appears.

C Map **Record**, under **BatchRecord_1** in the left pane of the Business Rules Designer, to the **bytes (byte[])** input node of the String method box.

D Map the result (String) output node of the String method box, to **Text** under **FileClient_1** in the right pane of the Business Rules Designer (see Figure 93).

Figure 93 Collaboration Editor (Java) - Business Rules



6 To create the **FileClient_1.write** rule under the **while** statement do the following:

A Click **rule** on the Business Rules toolbar to add a new **rule** under the while statement.

B Right-click FileClient_1 in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The method selection window appears.

C Select **write()** from the method selection window. The write method box appears in the Business Rules Designer canvas.

7 To create the **BatchRecord_1.finish** rule do the following:

A Select the While statement on the Business Rules tree and click **rule** on the Business Rules toolbar. A new rule is added to the Business Rules tree.

B Right-click **BatchRecord_1** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The method selection window appears.

C Select **finish()** from the method selection window. The **finish** method box appears.

8 From the editor's toolbar, click **Validate** to check the Collaboration for errors.

9 Save your changes to the Repository.

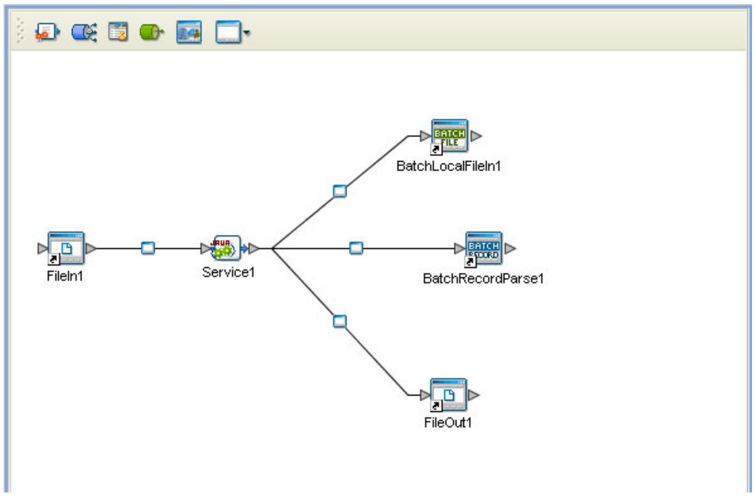
For more information on how to create Business Rules using the Collaboration Editor see the *eGate Integrator User's Guide*.

7.6.5 Binding the Project Components

The components are associated and the bindings are created in the Connectivity Map.

1. From the Project Explorer, double-click the Connectivity Map **CM_RecParseStream_Sample**. The Enterprise Designer canvas now displays the **CM_RecParseStream_Sample** Connectivity Map.
2. Drag and drop the **jcd_RecParse_Stream** Collaboration from the Project Explorer tree onto **Service1** in the **CM_RecParseStream_Sample** Connectivity Map.
3. Double-click **Service1**. The **Service1** binding dialog box appears with **jcd_RecParse_Stream** as the rule.
4. From the **Service1** binding box, map **FileClient input** (under Implemented Services) to the **FileIn1** application.
5. From the **Service1** binding box, map the **BatchLocalFile** (under Invoked Services) to the **BatchLocalFileIn1** External Application.
6. From the **Service1** binding box, map the **BatchRecord** (under Invoked Services) to the **BatchRecordParse1** External Application.
7. From the **Service1** binding box, map the **FileClient** (under Invoked Services) to the **FileOut1** External Application.
8. Minimize the **Service1** binding box. The Connectivity Map now appears as displayed in Figure 94.

Figure 94 Connectivity Map - Connecting the Project's Components



9. Save your current changes to your Repository.

7.6.6 Creating an Environment

Environments include the External Systems, Logical Hosts, Integration Servers and Message Servers used by a Project and contain the configuration information for these components.

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **ENV_Batch_RecParseStream**.

4 Right-click **ENV_Batch_RecParseStream** and select **New File External System**. Name this External System **FileExtSysIn** and select **Inbound File eWay** as the External System Type. The **FileExtSysIn** window is added to the Environment Editor.

5 Right-click **ENV_Batch_RecParseStream** and select **New File External System**. Name this External System **FileExtSysOut** and select **Outbound File eWay** as the External System Type. The **FileExtSysOut** window is added to the Environment Editor.

6 Right-click **ENV_Batch_RecParseStream** and select **New BatchLocalFile External System**. Name the External System **BatchLocalFileExtSys**. Click **OK**. The **BatchLocalFileExtSys** window is added to the Environment Editor.

7 Right-click **ENV_Batch_RecParseStream** and select **New BatchRecord External System**. Name the External System **BatchRecordExtSys**. Click **OK**. The **BatchRecordExtSys** window is added to the Environment Editor.

8 Right-click **ENV_Batch_RecParseStream** and select **New Logical Host**. The **LogicalHost1 box** is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

9 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1**.

10 Save your current changes to the Repository.

7.6.7 Configuring the eWay Properties

The **Batch_RecParseStream_Sample** Project uses three eWays, each represented in the Connectivity Map as a node between an External Application and a Service.

The **File**, **BatchLocalFile**, and **BatchRecord** eWay properties are configured from the Connectivity Map. To configure the eWays do the following:

Configuring the File eWay Properties

1 Double-click the **Outbound File eWay** and select **Outbound File eWay** in the Templates dialog box and click **OK**.

2 The Properties Sheet opens to the Outbound File eWay properties. Modify the configuration for your system.

Configuring the BatchLocalFile eWay Properties

The BatchLocalFile and BatchRecord eWay properties are set from the Project Explorer's Connectivity Map. For more information on the BatchLocalFile eWay properties and the Properties Sheet, see **Creating and Configuring the Batch eWay** on page 21.

For the Batch_RecParseStream_Sample Project, do the following:

Modifying the BatchLocalFile and BatchRecord eWay (Project Explorer) Properties

1 From the **Connectivity Map**, double-click the eWay. The Properties Sheet opens to the eWay Connectivity Map properties.

2 Modify the eWay properties for your system.

3 Save the current changes to your Repository.

7.6.8 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Collaborations and message destinations to the integration server and message server. Deployment profiles are created using the Deployment Editor.

1 From the Enterprise Explorer's Project Explorer, right-click the Project (**Batch_RecParseStream_Sample**) and select **New** > **Deployment Profile**.

2 Enter a name for the Deployment Profile (for this sample **DP_Batch_RecParseStream**). Make sure that the selected Environment is **ENV_Batch_RecParseStream**. Click **OK**.

3 From the Deployment Editor toolbar, click the **Auto Map** icon. The Projects components are automatically mapped to their system windows. If any of the Project components are not mapped automatically after Auto Map is used, those component can be mapped manually by following the appropriate steps below. Once all components are mapped, proceed to step 9.

4 From the left pane of the Deployment Editor, drag the **FileIn1 -> Service1** (External Application) to the **FileExtSysIn** window.

5 From the left pane of the Deployment Editor, drag **Service1 -> FileOut1** (External Application) to the **FileExtSysOut** window.

6 From the left pane of the Deployment Editor, drag the **Service1 -> BatchLocalFileIn1** (External Application) to the **BatchLocalFileExtSys** window.

7 From the left pane of the Deployment Editor, drag the **Service1 -> BatchRecordParse1** (External Application) to the **BatchRecordExtSys** window.

8 From the left pane of the Deployment Editor, drag **Service1** (Collaboration) to **IntegrationSvr1** in the **LogicalHost1** window.

9 Click **Activate**. When activation succeeds, save Your current changes to the Repository.

7.6.9 Running the Project

To run the Project follow the directions provided in **"Running the Project" on page 218**.

7.7 The BatchInbound_Sample Project

The **BatchInbound_Sample** Project demonstrates the following:

1. The BatchInbound eWay polls the input directory periodically. When it sees a specified file, it renames the file with a GUID file name, and triggers the Collaboration.

2. The BatchLocalfileIn eWay gets the file with the GUID file name.

3. The files contents is copied from BatchLocalfileIn payload to the FileOut payload.
4. The FileOut eWay writes the payload to an output file.

The following pages provide step by step directions for manually creating the **BatchInbound_Sample** Project components.

7.7.1 Create a Project

The first step is to create and name a new Project in eGate Enterprise Designer.

1. Start the Enterprise Designer.
2. From the Enterprise Explorer's Project Explorer tab, select your Repository on the Project Explorer tree, right-click the Repository and select **New Project**. A new Project appears on the Project Explorer tree.
3. Rename the Project (for this sample, **BatchInbound_Sample**).

7.7.2 Create a Connectivity Map

The Connectivity Map provides a canvas for configuring a Project's components.

1. In Enterprise Explorer's Project Explorer, right-click the new Project (**BatchInbound_Sample**) and select **New > Connectivity Map** from the shortcut menu.
2. The New Connectivity Map appears and a node for the Connectivity Map is added under the Project, on the Project Explorer tree, labeled **CMap1**. Rename the Connectivity Map **CMap_BatchInbound**.

Select the External Applications

1. Click the **External Application** icon on the Connectivity Map toolbar,
2. Select the following applications for your Project:
   - BatchInbound External Application
   - BatchLocalFile External System
   - File External System

3 Icons representing the selected applications are added to the Connectivity Map toolbar.

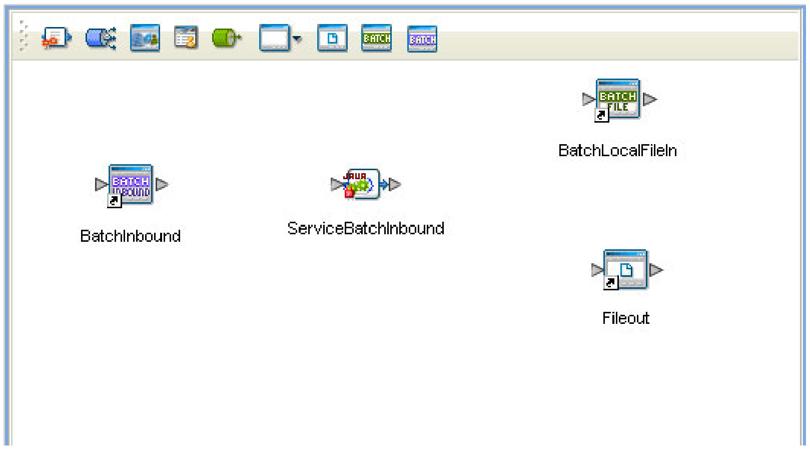
Populate the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

1 For the BatchInbound_Sample Project, drag the following components onto the Connectivity Map canvas, and name each as displayed in Figure 90:

- BatchInbound External Application (rename to **BatchInbound**)
- Service (rename to **ServiceBatchInbound**)
- BatchLocalFile External System (rename to **BatchLocalFileIn**)
- File External Application (rename to **Fileout**)

Figure 95 CMap_BatchInbound Connectivity Map with Components



2 Save your current changes to the Repository.

7.7.3 Creating a Collaboration Definition

The next step in the sample is to create a Java Collaboration using the Collaboration Definition Wizard (Java). Once a Collaboration Definition has been created, the Business Rules of the Collaboration are written using the Collaboration Editor.

The CollabBatchInbound Collaboration

1 From the Project Explorer, right-click the **BatchInbound_Sample** Project and select **New > Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.

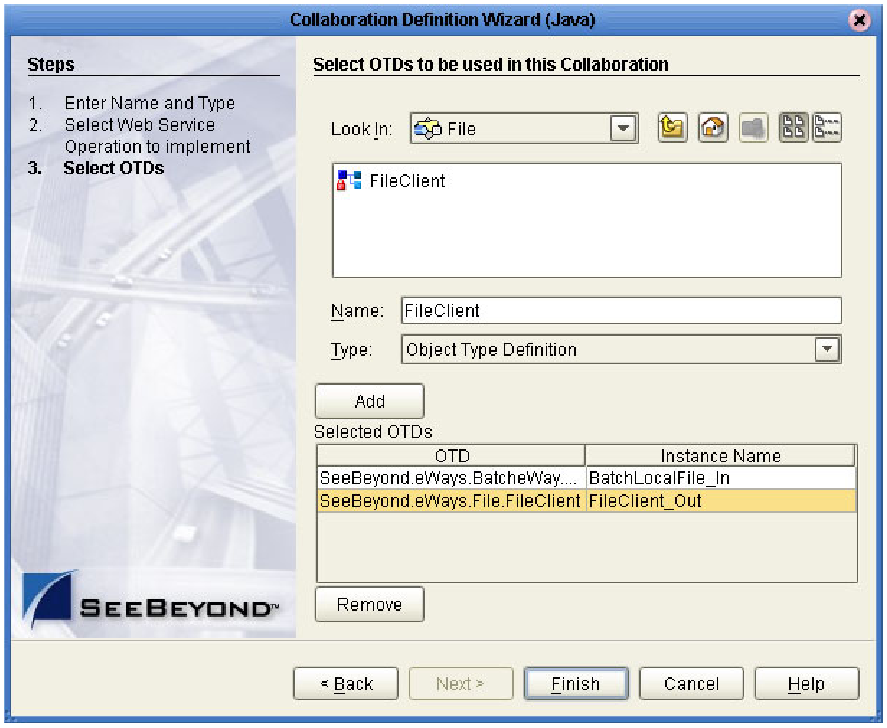
2 Enter a Collaboration Definition name (for this sample **CollabBatchInbound**) and click **Next**.

3 For Step 2 of the Wizard, from the Web Services Interfaces selection window, double-click **SeeBeyond** > **eWays** > **BatcheWay** > **BatchInbound**> **receive**. The Name field now displays **receive.** Click **Next**.

4 For Step 3 of the Wizard, from the Select OTDs selection window, double-click **SeeBeyond** > **eWays** > **BatcheWay** > **BatchLocalFile**. The **BatchLocalFile.LocalFile** OTD is added to the Selected OTDs field. Double-click the **BatchLocalFile_1** Instance Name and rename the instance to **BatchLocalFile_In**.

5 Click the **Up One Level** button to return to the Repository directory. Double-click **SeeBeyond** > **eWays** > **File** > **FileClient**. The **FileClient** OTD is added to the Selected OTDs field. Double-click the **FileClient_1** Instance Name and rename the instance to **FileClient_Out**.

Figure 96 Collaboration Definition Wizard (Java) - Select OTDs


Collaboration Definition Wizard (Java)
Steps
1. Enter Name and Type
2. Select Web Service Operation to implement
3. Select OTDs
Select OTDs to be used in this Collaboration
Look In:
File
FileClient
Add
Selected OTDs
Remove
< Back
Next >
Finish
Cancel
Help
SEEBEYOND

Name: FileClient
Type: Object Type Definition

OTD	Instance Name
SeeBeyond.eWays.BatcheWay....	BatchLocalFile_In
SeeBeyond.eWays.File.FileClient	FileClient_Out

6 Click **Finish**. The Collaboration Editor (Java) opens to the new Collaboration in the right pane of the Enterprise Designer.

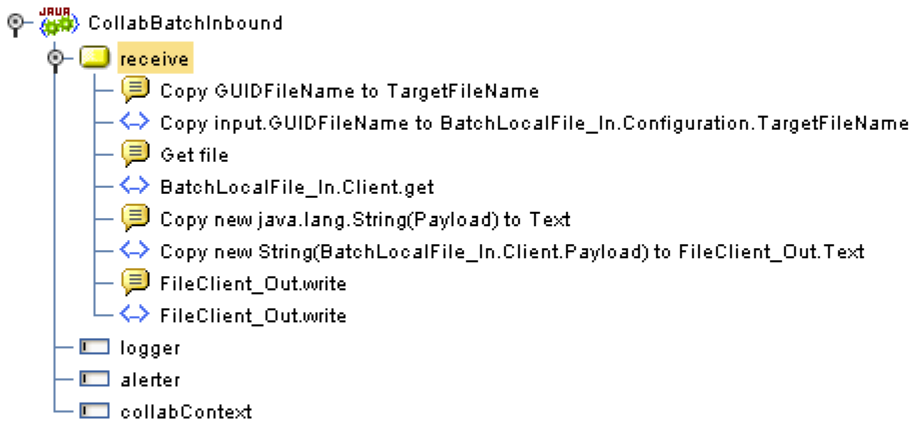
7.7.4 Using the Collaboration Editor (Java)

The **BatchInbound_Sample** Project uses the **CollabBatchInbound** Collaboration created in the previous section. To complete the Collaboration, use the Collaboration Editor to create the Business Rules.

Create the CollabBatchInbound Collaboration Business Rules

Be careful to open all nodes specified in the directions to connect the correct items. The **CollabBatchInbound** Collaboration contains the Business Rule displayed in **Figure 97** on page 231.

Figure 97 CollabBatchInbound Business Rules



```
CollabBatchInbound
  receive
    Copy GUIDFileName to TargetFileName
    Copy input.GUIDFileName to BatchLocalFile_In.Configuration.TargetFileName
    Get file
    BatchLocalFile_In.Client.get
    Copy new java.lang.String(Payload) to Text
    Copy new String(BatchLocalFile_In.Client.Payload) to FileClient_Out.Text
    FileClient_Out.write
    FileClient_Out.write
  logger
  alerter
  collabContext
```

To create the **CollabBatchInbound** Collaboration Business Rules, do the following:

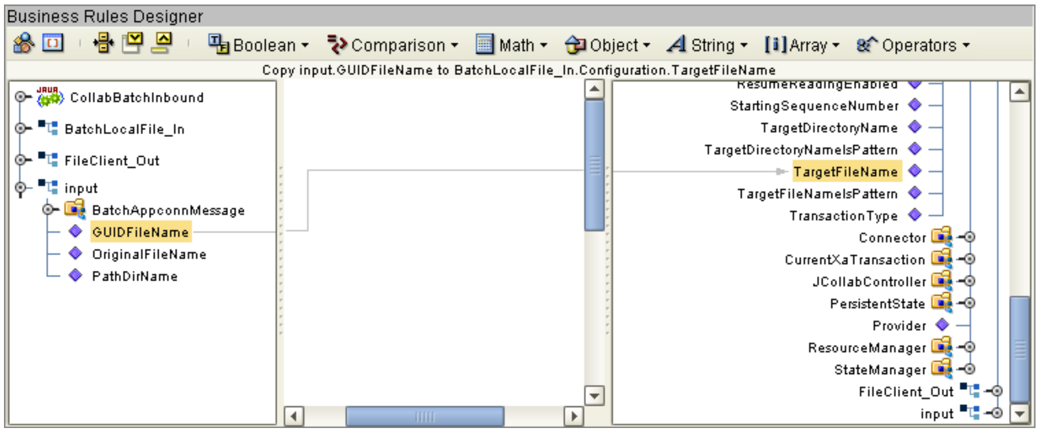
1. From the Project Explorer tree, double-click **CollabBatchInbound**. The Collaboration Editor (Java) opens to the **CollabBatchInbound** Collaboration.

2. To create comments for the Business Rules, from the Business Rules toolbar, click the **comment** icon. The **Enter a Comment** dialog box appears. Enter the comment and click **OK**. The comment is placed on the Business Rules tree under the last selected item. Once the Comment is created, it can be moved by clicking the comment and dragging it up or down the Business Rules tree to a new location.

3. To create the **Copy input.GUIDFileName to BatchLocalFile_In.Configuration.TargetFileName** Business Rule, do the following:

   A Map **GUIDFileName** under BatchAppconnMessage > input in the left pane of the Business Rules Designer, to **TargetFileName** under Configuration > BatchLocalFile_In in the right pane of the Business Rules Designer. To do this, click on **GUIDFileName** in the left pane of the Business Rules Designer, and drag the cursor to **TargetFileName** in the right pane of the Business Rules Designer, and release the cursor (see Figure 98).

Figure 98 Collaboration Editor (Java) - Business Rules



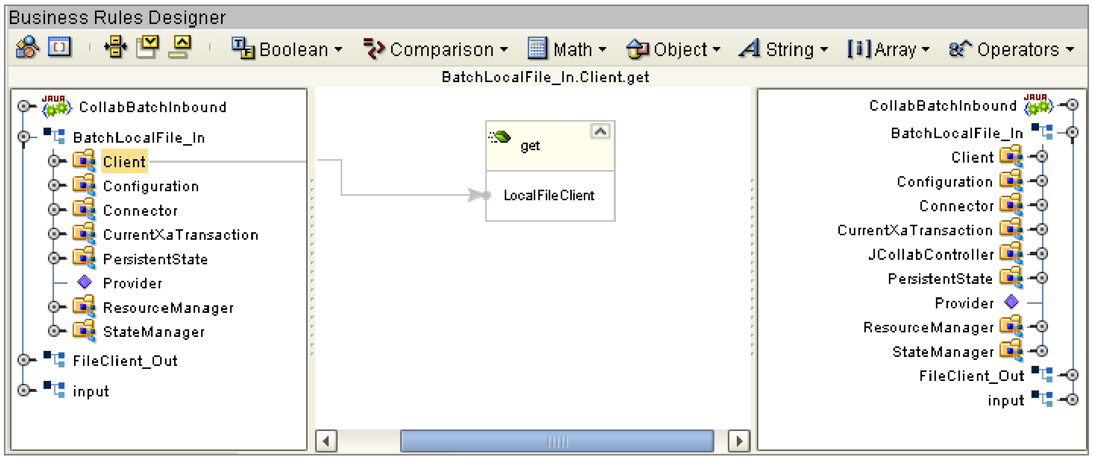
4. To create the **BatchLocalFile_In.Client.get** rule, do the following:

A Click **rule** on the Business Rules toolbar to add a new rule in the Business Rules pane.

B Right-click **Client** under **BatchLocalFile_In** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The method selection window appears.

C Select **get()** from the method selection window. The **get** method box appears in the Business Rules Designer canvas as displayed in Figure 99.

Figure 99 Collaboration Editor (Java) - Business Rules



5 To create the **Copy new String(BatchLocalFile_In.Client.Payload) to FileClient_Out.Text** rule do the following:

A Click **rule** on the Business Rules toolbar to add a new rule in the Business Rules pane.

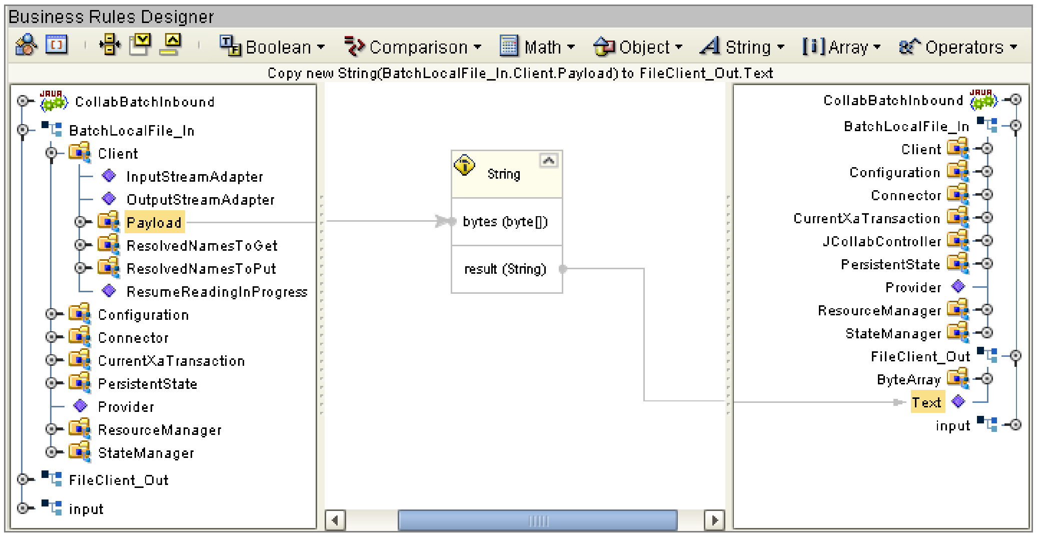
B Map **Payload** under **Client** > **BatchAppconnMessage** in the left pane of the Business Rules Designer, to **ByteArray** under **FileClient_Out** in the right pane of the Business Rules Designer (see Figure 100).

C From the Business Rules Designer toolbar, click the **Class Browser** icon. The **Class Browser** dialog box appears. From the Class Browser dialog box, select **String** under **All Classes**, select **String(byte[] bytes)** under **String**, and click **Select**. The String method box appears.

D Map **Payload** under BatchLocalFile_In > Client in the left pane of the Business Rules Designer, to the **bytes (byte[])** input node of the **String** method box.

E Map **result (String)** output node of the String method box, to **Text** under BatchLocalFile_In in the right pane of the Business Rules Designer (See **Figure 100** on page 233.

Figure 100 Collaboration Editor (Java) - Business Rules



6 To create the **FileClient_Out.write** rule do the following:

A Click **rule** on the Business Rules toolbar to add a new rule in the Business Rules pane.

B Right-click **FileClient_Out** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The method selection window appears.

C Select **write()** from the method selection window. The **write** method box appears in the Business Rules Designer canvas.

7 From the editor's toolbar, click **Validate** to check the Collaboration for errors.

8 Save your changes to the Repository.

For more information on how to create Business Rules using the Collaboration Editor see the *eGate Integrator User's Guide*.

7.7.5 Binding the Project Components

The components are associated and the bindings are created in the Connectivity Map.

1 From the Project Explorer, double-click the Connectivity Map **CMap_BatchInbound** to display the Connectivity Map in the Enterprise Designer.

2 Drag and drop the **CollabBatchInbound** Collaboration from the Project Explorer tree onto the **ServiceBatchInbound** service in the Connectivity Map.

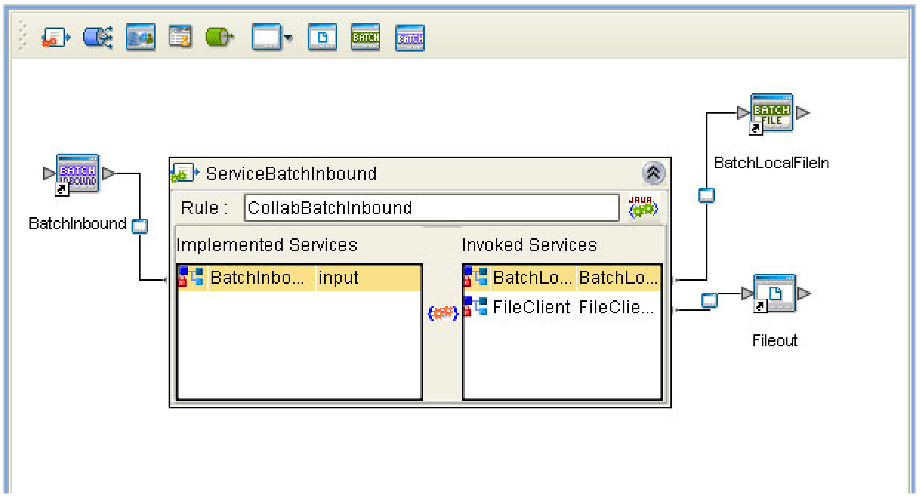
3 Double-click **ServiceBatchInbound**. The **ServiceBatchInbound** binding dialog box appears with **CollabBatchInbound** as the rule.

4 From the **ServiceBatchInbound** binding box, map **BatchInbound input** (under Implemented Services) to the **BatchInbound** application.

5 From the **ServiceBatchInbound** binding box, map the **BatchLocalFile** (under Invoked Services) to the **BatchLocalFileIn** External Application.

6 From the **ServiceBatchInbound** binding box, map the **FileClient** (under Invoked Services) to the **Fileout** External Application (see **Figure 101** on page 234).

Figure 101 Connectivity Map - Connecting the Project's Components



7 Minimize the **ServiceBatchInbound** binding box.

8 Save your current changes to your Repository.

7.7.6 Creating an Environment

Environments include the External Systems, Logical Hosts, Integration Servers and Message Servers used by a Project and contain the configuration information for these components.

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **Env_BatchInbound**.

4 Right-click **Env_BatchInbound** and select **New BatchInbound External System**. Name this External System **BatchInboundExtSysIn**. The **BatchInboundExtSysIn** window is added to the Environment Editor.

5 Right-click **Env_BatchInbound** and select **New File External System**. Name this External System **FileExtSysOut** and select **Outbound File eWay** as the External System Type. The **FileExtSysOut** window is added to the Environment Editor.

6 Right-click **Env_BatchInbound** and select **New BatchLocalFile External System**. Name the External System **BatchLocalFileExtSys**. Click **OK**. The **BatchLocalFileExtSys** window is added to the Environment Editor.

7 Right-click **Env_BatchInbound** and select **New Logical Host**. The **LogicalHost1 box** is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

8 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1**.

9 Save your current changes to the Repository.

7.7.7 Configuring the eWay Properties

The **BatchInbound_Sample** Project uses three eWays, each represented in the Connectivity Map as a node between an External Application and a Service.

The **BatchInbound**, **BatchLocalFile**, and **File** eWay properties are configured from the Connectivity Map. To configure the eWays do the following:

Configuring the File eWay Properties

The File eWay properties are set from the Project Explorer's Connectivity Map.

1 Double-click the **Outbound File eWay** and select **Outbound File eWay** in the Templates dialog box and click **OK**.

2 The Properties Sheet opens to the Outbound File eWay properties. Modify the configuration for your system.

Configuring the BatchLocalFile eWay Properties

The BatchLocalFile and BatchInbound eWay properties are set from the Project Explorer's Connectivity Map. For more information on the eWay properties and the Properties Sheet, see **Creating and Configuring the Batch eWay** on page 21.

1 From the **Connectivity Map**, double-click the eWay. The Properties Sheet opens to the eWay Connectivity Map properties.

2 Modify the eWay properties for your system.

3 Save the current changes to your Repository.

7.7.8 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Collaborations and message destinations to the integration server and message server. Deployment profiles are created using the Deployment Editor.

1 From the Enterprise Explorer's Project Explorer, right-click the Project (**BatchInbound_Sample**) and select **New** > **Deployment Profile**.

2 Enter a name for the Deployment Profile (for this sample **DP_BatchInbound**). Make sure that the selected Environment is **Env_BatchInbound**. Click **OK**.

3 From the Deployment Editor toolbar, click the **Auto Map** icon. The Projects components are automatically mapped to their system windows. If any of the Project components are not mapped automatically after Auto Map is used, those component can be mapped manually by following the appropriate steps below. Once all components are mapped, proceed to step 8.

4 From the left pane of the Deployment Editor, drag **ServiceBatchInbound -> FileOut** (External Application) to the **FileExtSysOut** window.

5 From the left pane of the Deployment Editor, drag the **BatchInbound -> ServiceBatchInbound** (External Application) to the **BatchInboundExtSys** window.

6 From the left pane of the Deployment Editor, drag the **ServiceBatchInbound -> BatchLocalFileIn** (External Application) to the **BatchLocalFileExtSys** window.

7 From the left pane of the Deployment Editor, drag **ServiceBatchInbound** (Collaboration) to **IntegrationSvr1** in the **LogicalHost1** window.

8 Click **Activate**. When activation succeeds, save Your current changes to the Repository.

7.7.9 Running the Project

To run the Project follow the directions provided in **"Running the Project" on page 218**.

7.8 The Batch_Security_Sample_JCD Project

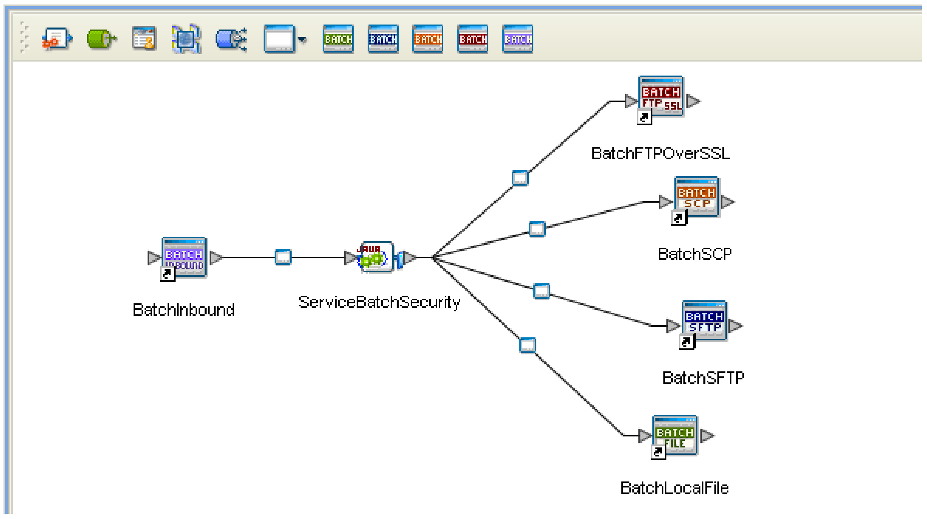
This section provides an overview of the Batch_Security_Sample_JCD Project, and describes how to run the imported project. The sample is created in the same manner as the previous samples in this chapter.

The Batch_Security_Sample_JCD Project contains one Java Collaboration Definition that employs three Batch OTDS that enable SSL or SSH secure file transfers.

The Batch_Security_Sample_JCD Project Components

The Batch_Security_Sample_JCD Project uses five External Applications, five component eWays, and one Java Collaboration, as seen in the Project's Connectivity Map (see Figure 102).

Figure 102 cmBatchSecurity Connectivity Map



These five Batch eWays perform the following functions

- **BatchInbound**: acts as a trigger for the Project. The BatchInbound eWay polls an external directory for a specific input file. When the input file is present, the BatchInbound eWay triggers the Collaboration.
- **BatchFTPOverSSL**: is used by the Collaboration to:
  - get files from a remote directory and publish them to a local directory
  - get files from a local directory and publish them to the remote directory
  - download files from the remote directory to a local directory
  - upload files from a local directory to a remote directory
  - create and list remote directories
  - delete remote directories and list the results

  These functions can be seen in the **FTP over SSL** section of the **jcdBatchSecurity** Collaboration Definition (see **Figure 104** on page 239). The BatchFTPOverSSL eWay supports SSL protocol.

- **BatchSFTP**: is used by the Collaboration to:
    - get files from a remote directory and publish them to a local directory
    - create a new remote directory
    - change directories to the new directory and publish a file to that directory
    - rename the published file
    - delete an unwanted file from the remote directory

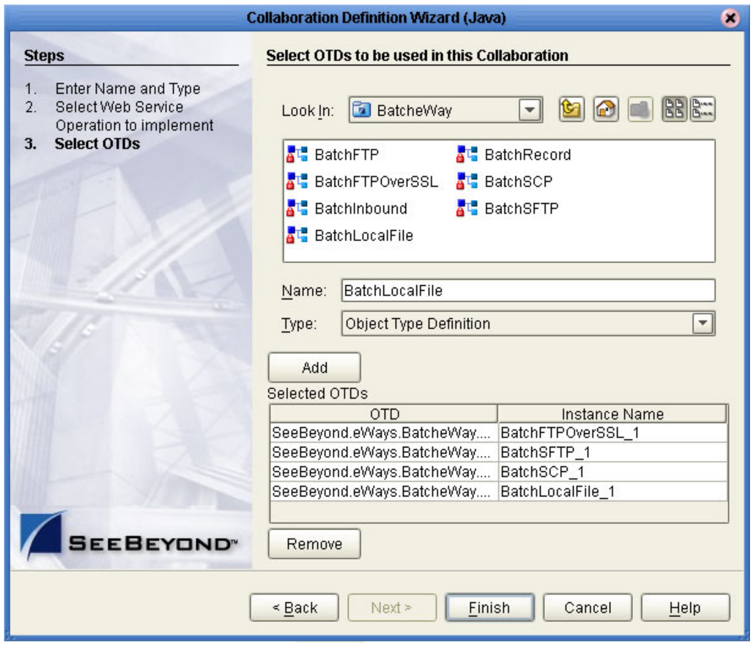
  These functions can be seen in the **SFTP** section of the **jcdBatchSecurity** Collaboration Definition (see **Figure 105** on page 240).
- **BatchSCP**: is used by the Collaboration to:
    - recursively get a file from the remote directory and publish the file to a local directory
    - recursively put a file from the local directory to a remote directory

  These functions can be seen in the **SCP** section of the **jcdBatchSecurity** Collaboration Definition (see **Figure 106** on page 240).
- **BatchLocalFile**: is used to write a list of directories to a local directory.

7.8.1 The jcdBatchSecurity Collaboration Definition

The Batch_Security_Sample_JCD Project's **jcdBatchSecurity** Collaboration Definition is created with the Collaboration Definition Wizard (Java). It uses the **BatchInbound OTD** as the Web Service, and the **BatchFTPOverSSL**, **BatchSFTP**, **BatchSCP**, and **BatchLocalFile** OTDs for the necessary API to create the Business Rules (see Figure 103).

Figure 103 jcdBatchSecurity Collaboration Definition - SFTP Section

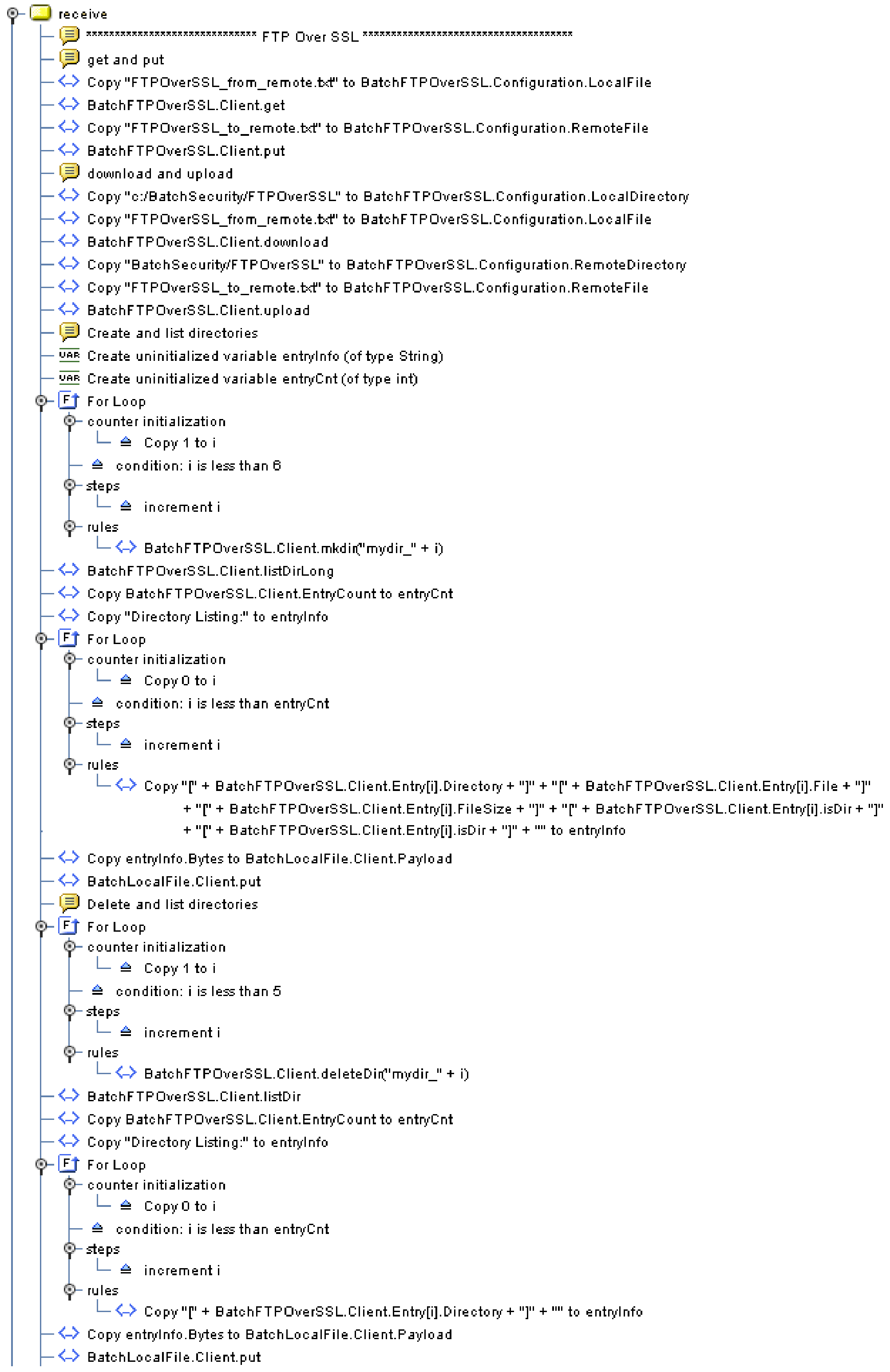


Name: BatchLocalFile
Type: Object Type Definition

OTD	Instance Name
SeeBeyond.eWays.BatcheWay....	BatchFTPOverSSL_1
SeeBeyond.eWays.BatcheWay....	BatchSFTP_1
SeeBeyond.eWays.BatcheWay....	BatchSCP_1
SeeBeyond.eWays.BatcheWay....	BatchLocalFile_1

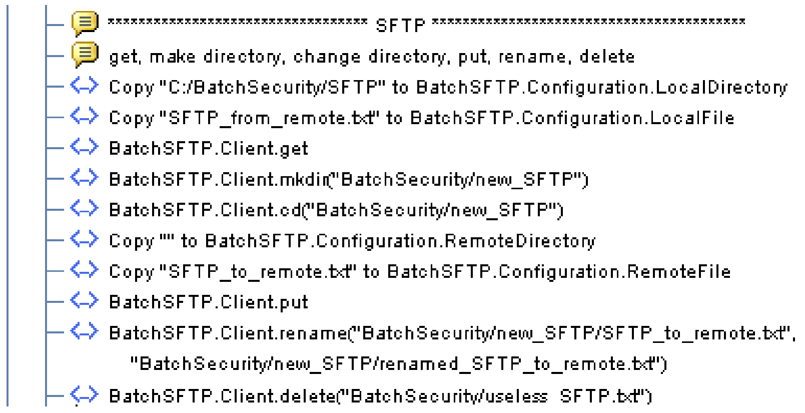
The Collaboration's Business Rules, as seen from the Java Collaboration Editor, have been separated into three sections to highlight the different OTDs used to perform the various functions: **FTPOverSSL** - Figure 104, **SFTP** - Figure 105, and **SCP** - Figure 106.

Figure 104 jcdBatchSecurity Collaboration Definition - FTP over SSL Section



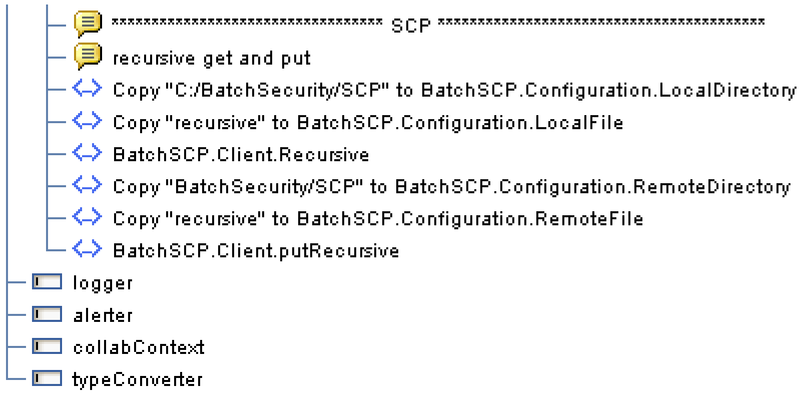
```
receive
  ****************************** FTP Over SSL ************************************
  get and put
  Copy "FTPOverSSL_from_remote.txt" to BatchFTPOverSSL.Configuration.LocalFile
  BatchFTPOverSSL.Client.get
  Copy "FTPOverSSL_to_remote.txt" to BatchFTPOverSSL.Configuration.RemoteFile
  BatchFTPOverSSL.Client.put
  download and upload
  Copy "c:/BatchSecurity/FTPOverSSL" to BatchFTPOverSSL.Configuration.LocalDirectory
  Copy "FTPOverSSL_from_remote.txt" to BatchFTPOverSSL.Configuration.LocalFile
  BatchFTPOverSSL.Client.download
  Copy "BatchSecurity/FTPOverSSL" to BatchFTPOverSSL.Configuration.RemoteDirectory
  Copy "FTPOverSSL_to_remote.txt" to BatchFTPOverSSL.Configuration.RemoteFile
  BatchFTPOverSSL.Client.upload
  Create and list directories
  Create uninitialized variable entryInfo (of type String)
  Create uninitialized variable entryCnt (of type int)
  For Loop
    counter initialization
      Copy 1 to i
    condition: i is less than 6
    steps
      increment i
    rules
      BatchFTPOverSSL.Client.mkdir("mydir_" + i)
  BatchFTPOverSSL.Client.listDirLong
  Copy BatchFTPOverSSL.Client.EntryCount to entryCnt
  Copy "Directory Listing:" to entryInfo
  For Loop
    counter initialization
      Copy 0 to i
    condition: i is less than entryCnt
    steps
      increment i
    rules
      Copy "[" + BatchFTPOverSSL.Client.Entry[i].Directory + "]" + "[" + BatchFTPOverSSL.Client.Entry[i].File + "]"
            + "[" + BatchFTPOverSSL.Client.Entry[i].FileSize + "]" + "[" + BatchFTPOverSSL.Client.Entry[i].isDir + "]"
            + "[" + BatchFTPOverSSL.Client.Entry[i].isDir + "]" + "" to entryInfo
  Copy entryInfo.Bytes to BatchLocalFile.Client.Payload
  BatchLocalFile.Client.put
  Delete and list directories
  For Loop
    counter initialization
      Copy 1 to i
    condition: i is less than 5
    steps
      increment i
    rules
      BatchFTPOverSSL.Client.deleteDir("mydir_" + i)
  BatchFTPOverSSL.Client.listDir
  Copy BatchFTPOverSSL.Client.EntryCount to entryCnt
  Copy "Directory Listing:" to entryInfo
  For Loop
    counter initialization
      Copy 0 to i
    condition: i is less than entryCnt
    steps
      increment i
    rules
      Copy "[" + BatchFTPOverSSL.Client.Entry[i].Directory + "]" + "" to entryInfo
  Copy entryInfo.Bytes to BatchLocalFile.Client.Payload
  BatchLocalFile.Client.put
```

Figure 105 jcdBatchSecurity Collaboration Definition - SFTP Section



```
*********************************** SFTP ******************************************
get, make directory, change directory, put, rename, delete
Copy "C:/BatchSecurity/SFTP" to BatchSFTP.Configuration.LocalDirectory
Copy "SFTP_from_remote.txt" to BatchSFTP.Configuration.LocalFile
BatchSFTP.Client.get
BatchSFTP.Client.mkdir("BatchSecurity/new_SFTP")
BatchSFTP.Client.cd("BatchSecurity/new_SFTP")
Copy "" to BatchSFTP.Configuration.RemoteDirectory
Copy "SFTP_to_remote.txt" to BatchSFTP.Configuration.RemoteFile
BatchSFTP.Client.put
BatchSFTP.Client.rename("BatchSecurity/new_SFTP/SFTP_to_remote.txt",
   "BatchSecurity/new_SFTP/renamed_SFTP_to_remote.txt")
BatchSFTP.Client.delete("BatchSecurity/useless SFTP.txt")
```

Figure 106 jcdBatchSecurity Collaboration Definition - SCP Section



```
*********************************** SCP ******************************************
recursive get and put
Copy "C:/BatchSecurity/SCP" to BatchSCP.Configuration.LocalDirectory
Copy "recursive" to BatchSCP.Configuration.LocalFile
BatchSCP.Client.Recursive
Copy "BatchSecurity/SCP" to BatchSCP.Configuration.RemoteDirectory
Copy "recursive" to BatchSCP.Configuration.RemoteFile
BatchSCP.Client.putRecursive
logger
alerter
collabContext
typeConverter
```

7.8.2 Completing the Batch_Security_Sample_JCD Project

Import the Batch_Security_Sample_JCD Project as directed in **"Importing a Sample Project" on page 200**. After the sample Project has been imported and appears in your Project Explorer tree, create the Project's Environment.

7.8.3 Creating an Environment

Environments include the External Systems, Logical Hosts, Integration Servers and Message Servers used by a Project and contain the configuration information for these components.

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **BatchSecuritySample_Env**.

4 Right-click **BatchSecuritySample_Env** and select **New BatchInbound External System**. Name this External System **BatchInboundExtSys**. The **BatchInboundExtSys** window is added to the Environment Editor.

5 Right-click **BatchSecuritySample_Env** and select **New BatchFTPOverSSL External System**. Name this External System **BatchFTPOverSSLExtSys**. The **BatchFTPOverSSLExtSys** window is added to the Environment Editor.

6 Right-click **BatchSecuritySample_Env** and select **New BatchSFTP External System**. Name this External System **BatchSFTPExtSys**. The **BatchSFTPExtSys** window is added to the Environment Editor.

7 Right-click **BatchSecuritySample_Env** and select **New BatchSCP External System**. Name this External System **BatchSCPExtSys**. The **BatchSCPExtSys** window is added to the Environment Editor.

8 Right-click **BatchSecuritySample_Env** and select **New BatchLocalFile External System**. Name the External System **BatchLocalFileExtSys**. Click **OK**. The **BatchLocalFileExtSys** window is added to the Environment Editor.

9 Right-click **BatchSecuritySample_Env** and select **New Logical Host**. The **LogicalHost1 box** is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

10 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under Localhost1.

11 Save your current changes to the Repository.

7.8.4 Configuring the eWay Properties

The **Batch_Security_Sample_JCD** Project uses five component eWays, each represented in the Connectivity Map as a node between an External Application and the Service.

The **BatchFTPOverSSL**, **BatchSFTP**, **BatchSCP,** and **BatchLocalFile** component eWays contain properties which are configured from the Connectivity Map, and Environment Explorer. The **BatchInbound** component eWay only possess Environment properties.

Configure the eWay properties for your system. For more information about the various properties, see **"Batch eWay Properties" on page 24**.

Create the Sample Directories

The sample Project uses a number of directories. The default directories in the sample configuration properties are:

- The input directory (BatchInbound): **c:/BatchSecurity**
- **Local Directory** (BatchFTPOverSSL, BatchSCP, BatchSFTP): **c:/BatchSecurity**
- **Remote Directory** (BatchFTPOverSSL, BatchSCP, BatchSFTP): **BatchSecurity**
- **Target Directory** (BatchLocalFile): **c:/BatchSecurity**

Create these directories on your system, or change the properties to reference your existing directories.

Create the Sample Files

The Sample also uses an input file named **BatchInbound.txt**. Create a text file with this name to use as a trigger file.

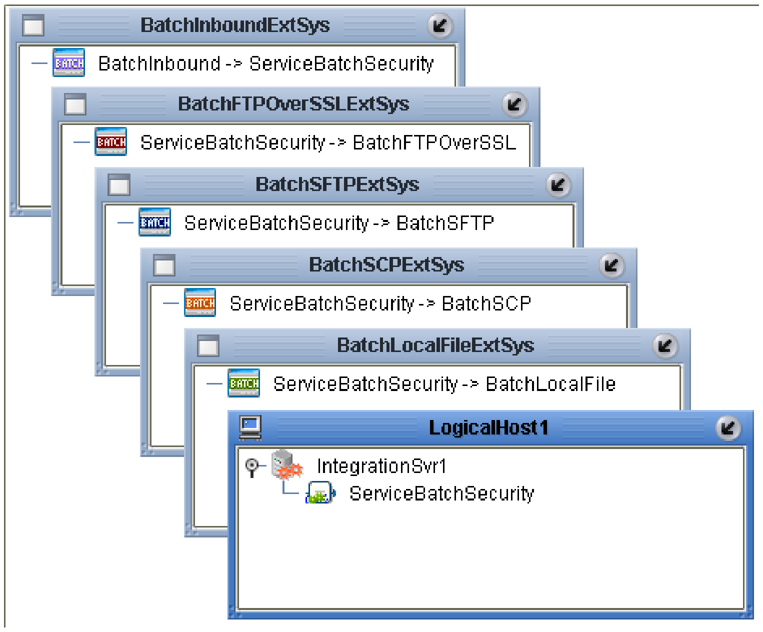
In addition, the sample uses a local file (the default name is **BatchSCP_local.txt**) and a remote file (the default name is **BatchSCP_remote.txt**). You can use any .txt files for these. For the remote file, rename your text file to **BatchSCP_remote.txt** and place it in the Remote Directory. For the local file, rename your text file to **BatchSCP_local.txt** and place it in the Local Directory.

7.8.5 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Collaborations and message destinations to the integration server and message server. Deployment profiles are created using the Deployment Editor.

1. From the Enterprise Explorer's Project Explorer, right-click the Project (**Batch_Security_Sample_JCD**) and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **BatchSecuritySample_DP**). Make sure that the selected Environment is **BatchSecuritySample_Env**. Click **OK**.
3. From the Deployment Editor toolbar, click the **Auto Map** icon. The Projects components are automatically mapped to their system windows. If any of the Project components are not mapped automatically after Auto Map is used, drag and drop those components to the appropriate environment window, as displayed in Figure 107. Once all components are mapped, proceed to step 4.

Figure 107 BatchSecuritySample_DP Deployment Profile


BatchInboundExtSys
BatchInbound -> ServiceBatchSecurity
BatchFTPOverSSLExtSys
ServiceBatchSecurity -> BatchFTPOverSSL
BatchSFTPExtSys
ServiceBatchSecurity -> BatchSFTP
BatchSCPExtSys
ServiceBatchSecurity -> BatchSCP
BatchLocalFileExtSys
ServiceBatchSecurity -> BatchLocalFile
LogicalHost1
IntegrationSvr1
ServiceBatchSecurity

4. Click **Activate**. When activation succeeds, save Your current changes to the Repository.

7.8.6 Running the Project

To run the Project follow the directions provided in **"Running the Project" on page 218**.

7.9 The Batch_Stream_Get Sample Project

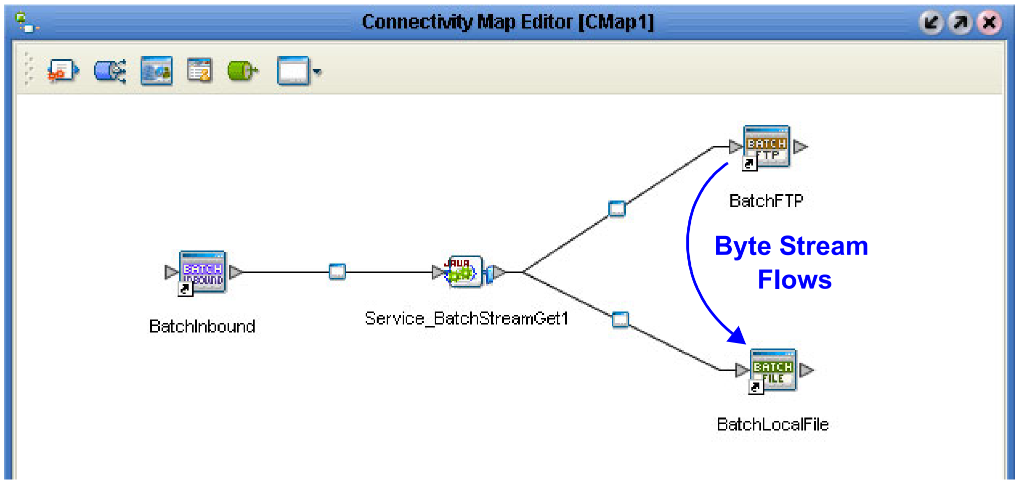
This section provides an overview of the **Batch_Stream_Get** sample Project, and describes how to run the imported project. The sample is created in the same manner as the previous samples in this chapter.

The Batch_Stream_Get sample Project contains one Java Collaboration Definition that employs two Batch OTDs that enable data streaming using the **OutputStreamAdapter** in the **BatchFTP** and **BatchLocalFile** OTDs.

The Batch_Stream_Get Sample Project Components

The **Batch_Stream_Get** sample Project uses three External Applications, three component eWays, and one Java Collaboration, as seen in the Project's Connectivity Map (see Figure 108).

Figure 108 CMap_BatchStreamGet Connectivity Map



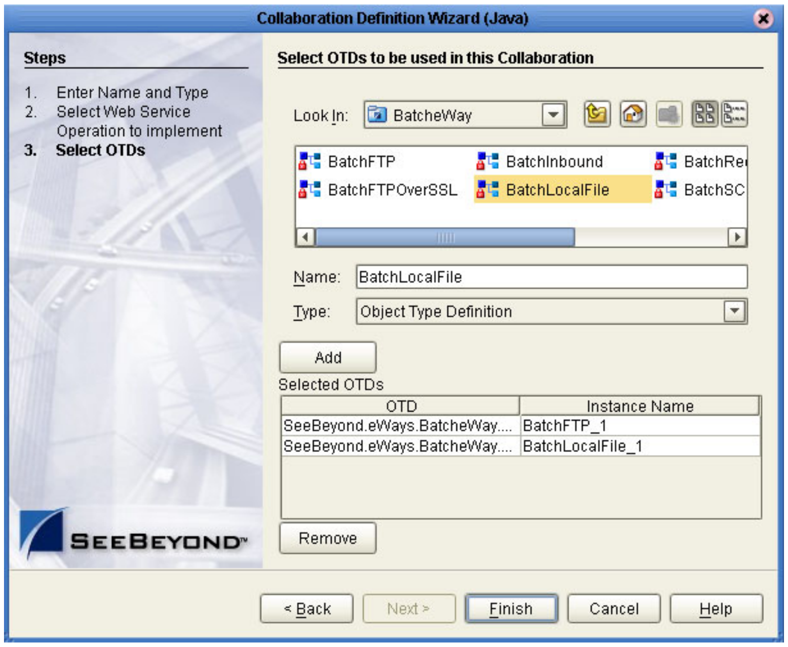
These three Batch eWays perform the following functions

- **BatchInbound**: acts as a trigger for the Project. The BatchInbound eWay polls an external directory for a specific input file. When the input file is present, the BatchInbound eWay triggers the Collaboration.
- **BatchFTP**: is used by the Collaboration to transfer data to or from a remote FTP server.
- **BatchLocalfile**: is used by the Collaboration to write the byte stream from the external system BatchFTP to a local file.

7.9.1 The jcd_BatchStreamGet Collaboration Definition

The **Batch_Stream_Get** Project's **jcd_BatchStreamGet** Collaboration Definition is created with the Collaboration Definition Wizard (Java). It uses the **BatchFTP** OTD and the **BatchLocalFile** OTD for the necessary API to create the Business Rules (see Figure 109).

Figure 109 jcd_BatchStreamGet Collaboration Definition - Select OTDs



Name: BatchLocalFile
Type: Object Type Definition

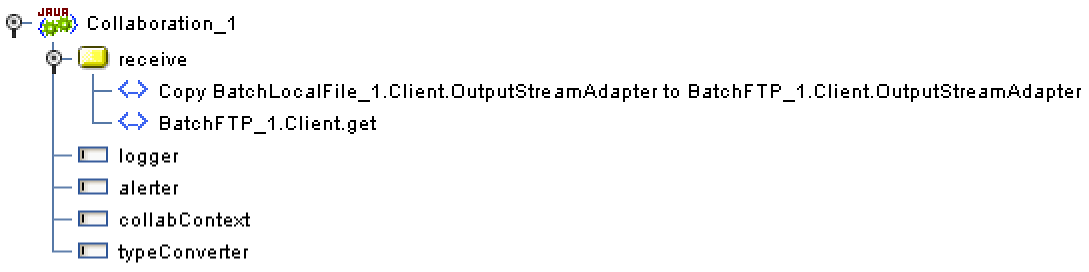
Selected OTDs

OTD	Instance Name
SeeBeyond.eWays.BatcheWay....	BatchFTP_1
SeeBeyond.eWays.BatcheWay....	BatchLocalFile_1

The Collaboration's Business Rules, as seen from the Java Collaboration Editor in Figure 110, highlight the two OTDs used to perform the data transfer. The Business Rules within this Java Collaboration definition include the following:

- The BatchInbound eWay component scans the directory for the trigger file.
- The BatchLocalFile eWay component receives a byte stream from the BatchFTP eWay component using the OutputStreamAdapter of the BatchLocalFile eWay component.
- The Client element of the BatchFTP consumes the stream adapter. In this case, the **com.stc.eways.batchext.FtpFileClient** instance invokes the **get()** method to complete the data transfer.

Figure 110 jcd_BatchStreamGet Collaboration Definition



```
Collaboration_1
  receive
    Copy BatchLocalFile_1.Client.OutputStreamAdapter to BatchFTP_1.Client.OutputStreamAdapter
    BatchFTP_1.Client.get
  logger
  alerter
  collabContext
  typeConverter
```

7.9.2 Completing the Batch_Stream_Get Project

Import the **Batch_Stream_Get** sample Project as directed in **"Importing a Sample Project" on page 200**. After the sample Project has been imported and appears in your Project Explorer tree, create the Project's Environment.

7.9.3 Creating an Environment

Environments include the External Systems, Logical Hosts, Integration Servers and Message Servers used by a Project and contain the configuration information for these components.

1. From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.
2. Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.
3. Rename the new Environment to **BatchStreamGet_Env**.
4. Right-click **BatchStreamGet_Env** and select **New BatchInbound External System**. Name this External System **BatchInboundExtSys**. The **BatchInboundExtSys** window is added to the Environment Editor.
5. Right-click **BatchStreamGet_Env** and select **New BatchFTP External System**. Name this External System **BatchFTPExtSys**. The **BatchFTPExtSys** window is added to the Environment Editor.
6. Right-click **BatchStreamGet_Env** and select **New BatchLocalFile External System**. Name the External System **BatchLocalFileExtSys**. Click **OK**. The **BatchLocalFileExtSys** window is added to the Environment Editor.
7. Right-click **BatchStreamGet_Env** and select **New Logical Host**. The **LogicalHost1 box** is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.
8. From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under Localhost1.
9. Save your current changes to the Repository.

7.9.4 Configuring the eWay Properties

The **Batch_Stream_Get** sample Project uses three component eWays, each represented in the Connectivity Map as a node between an External Application and the Service.

The **BatchFTP**, **BatchInbound**, and **BatchLocalFile** component eWays contain properties which are configured from the Connectivity Map, and Environment Explorer.

Configure the eWay properties for your system. For more information about the various properties, see **"Batch eWay Properties" on page 24**.

Create the Sample Directories

The sample Project uses a number of directories. The default directories in the sample configuration properties are:

- **The Directory Name** (BathInbound): C:\EGATE_DATA\BATCH_SAMPLE\IN
- **Target Directory Name** (BatchLocalFile): C:\EGATE_DATA\BATCH_SAMPLE\OUTPUT
- **Target Directory Name** (BatchFTP): BATCH_FTP_DIR

Create these directories on your system, or change the properties to reference your existing directories.

Create the Sample Files

The sample also uses an input file named **input.txt**. Create a text file with this name to use as a trigger file.

The sample also uses the following target file names.

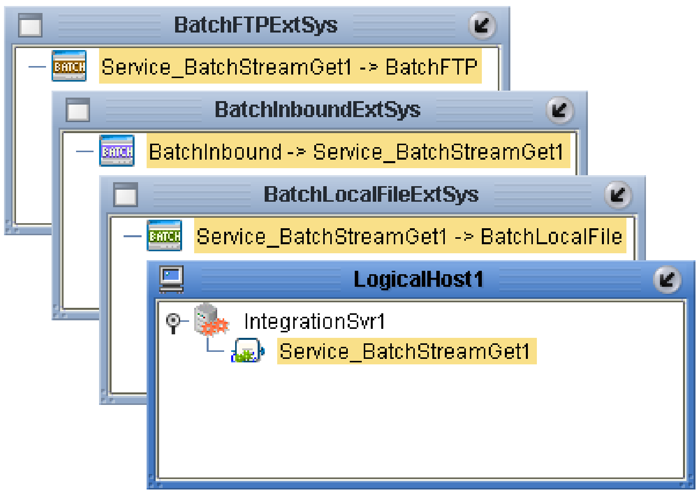
- The **BatchFTP** component eWay uses a Target File Name of INPUT_FOR_STREAM.txt.
- The **BatchLocalFile** component eWay uses a Target File Name of DATA_STREAM_OUT.txt

7.9.5 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Collaborations and message destinations to the integration server and message server. Deployment profiles are created using the Deployment Editor.

1. From the Enterprise Explorer's Project Explorer, right-click the Project (**Batch_Stream_Get**) and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **BatchStreamGet_DP**). Make sure that the selected Environment is **BatchStreamGet_Env**. Click **OK**.
3. From the Deployment Editor toolbar, click the **Auto Map** icon. The Projects components are automatically mapped to their system windows (see **Figure 111** on page 247).

Figure 111 BatchStreamGet_DP Deployment Profile


BatchFTPExtSys
Service_BatchStreamGet1 -> BatchFTP
BatchInboundExtSys
BatchInbound -> Service_BatchStreamGet1
BatchLocalFileExtSys
Service_BatchStreamGet1 -> BatchLocalFile
LogicalHost1
IntegrationSvr1
Service_BatchStreamGet1

4 Click **Activate**. When activation succeeds, save Your current changes to the Repository.

7.9.6 Running the Project

To run the Project follow the directions provided in **"Running the Project" on page 218**.

7.10 The Batch_Stream_Put Sample Project

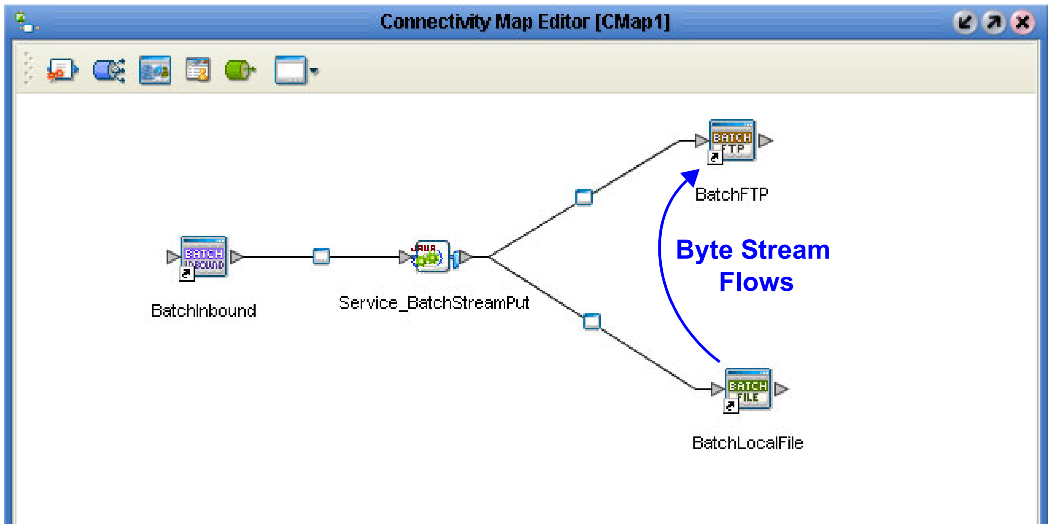
This section provides an overview of the **Batch_Stream_Put** Project, and describes how to run the imported project. The sample is created in the same manner as the previous samples in this chapter.

The Batch_Stream_Put Project contains one Java Collaboration Definition that employs two Batch OTDS that enable data streaming using the **InputStreamAdapter** in the **BatchFTP** and **BatchLocalFile** OTDs.

The Batch_Stream_Put Project Components

The Batch_Stream_Put Project uses three External Applications, three component eWays, and one Java Collaboration, as seen in the Project's Connectivity Map (see **Figure 112** on page 248).

Figure 112 CMap_BatchStreamPut Connectivity Map



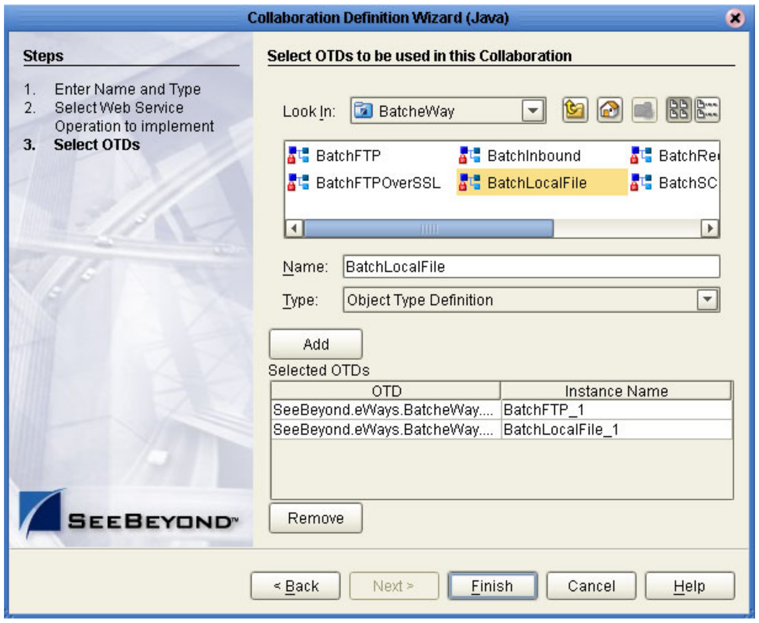
These three Batch eWays perform the following functions

- **BatchInbound**: acts as a trigger for the Project. The BatchInbound eWay polls an external directory for a specific input file. When the input file is present, the BatchInbound eWay triggers the Collaboration.
- **BatchFTP**: is used by the Collaboration to provide support for transfers to an external system:
- **BatchLocalFile**: reads a local file and provides the that local file as a byte stream to the external system BatchFTP.

7.10.1 The jcd_BatchStreamPut Collaboration Definition

The **Batch_Stream_Put** Sample Project's **jcd_BatchStreamPut** Collaboration Definition is created with the Collaboration Definition Wizard (Java). It uses the **BatchFTP** OTD and the **BatchLocalFile** OTD for the necessary API to create the Business Rules (see Figure 113 on page 249).

Figure 113 jcd_BatchStreamPut Collaboration Definition - Select OTDs



Name: BatchLocalFile
Type: Object Type Definition

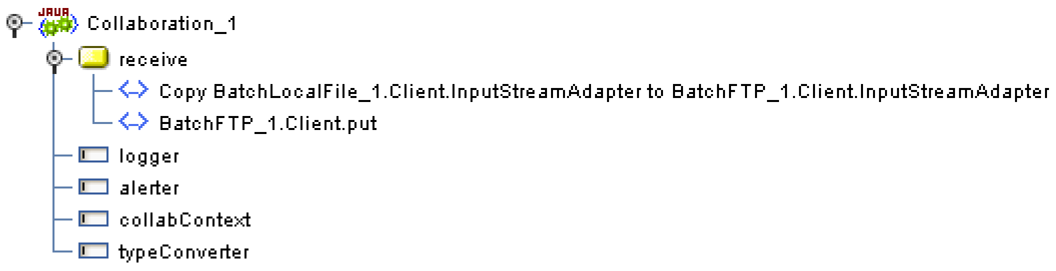
Selected OTDs

OTD	Instance Name
SeeBeyond.eWays.BatcheWay....	BatchFTP_1
SeeBeyond.eWays.BatcheWay....	BatchLocalFile_1

The Collaboration's Business Rules, as seen from the Java Collaboration Editor in Figure 114, highlight the two OTDs used to perform the data transfer. The Business Rules within this Java Collaboration definition include the following:

- The BatchInbound eWay component scans the directory for the trigger file.
- The BatchLocalFile eWay component sends a byte stream to the BatchFTP eWay component using the InputStreamAdapter of the BatchLocalFile eWay component.
- The Client element of BatchFTP consumes the stream adapter. In this case, the **com.stc.eways.batchext.FtpFileClient** instance invokes the **put()** method to complete the data transfer.

Figure 114 jcd_BatchStreamPut Collaboration Definition



```
Collaboration_1
  receive
    Copy BatchLocalFile_1.Client.InputStreamAdapter to BatchFTP_1.Client.InputStreamAdapter
    BatchFTP_1.Client.put
  logger
  alerter
  collabContext
  typeConverter
```

7.10.2 Completing the Batch_Stream_Put Project

Import the **Batch_Stream_Put** Project as directed in **"Importing a Sample Project" on page 200**. After the sample Project has been imported and appears in your Project Explorer tree, create the Project's Environment.

7.10.3 Creating an Environment

Environments include the External Systems, Logical Hosts, Integration Servers and Message Servers used by a Project and contain the configuration information for these components.

1. From the Enterprise Explorer, click the **Environment Explorer** tab.
2. Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.
3. Rename the new Environment to **BatchStreamPut_Env**.
4. Right-click **BatchStreamPut_Env** and select **New BatchInbound External System**. Name this External System **BatchInboundExtSys**. The **BatchInboundExtSys** window is added to the Environment Editor.
5. Right-click **BatchStreamPut_Env** and select **New BatchFTP External System**. Name this External System **BatchFTPExtSys**. The **BatchFTPExtSys** window is added to the Environment Editor.
6. Right-click **BatchStreamPut_Env** and select **New BatchLocalFile External System**. Name the External System **BatchLocalFileExtSys**. Click **OK**. The **BatchLocalFileExtSys** window is added to the Environment Editor.
7. Right-click **BatchStreamPut_Env** and select **New Logical Host**. **LogicalHost1** is added to the Environment Editor.
8. From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under Localhost1.
9. Save your current changes to the Repository.

7.10.4 Configuring the eWay Properties

The **Batch_Stream_Put** Sample Project uses three component eWays, each represented in the Connectivity Map as a node between an External Application and the Service.

The **BatchFTP**, **BatchInbound**, and **BatchLocalFile** component eWays contain properties which are configured from the Connectivity Map, and Environment Explorer.

Configure the eWay properties for your system. For more information about the various properties, see **"Batch eWay Properties" on page 24**.

Create the Sample Directories

The sample Project uses a number of directories. The default directories in the sample configuration properties (as noted with regular expressions) are:

- **The Directory Name** (BatchInbound): C:\\EGATE_DATA\\BATCH_SAMPLE\\^IN[0-9]
- **Target Directory Name** (BatchLocalFile): C:\EGATE_DATA\BATCH_SAMPLE\DATA
- **Target Directory Name** (BatchFTP): BATCH_FTP_DIR

Create these directories on your system, or change the properties to reference your existing directories.

Create the Sample Files

For this sample, the BatchInbound eWay File Name property value is the regular expression **^input\.txt**. Any file you create that adheres to the rules of the regular expression noted in the file name acts as a trigger for the sample Project.

The sample also uses the following target file names.

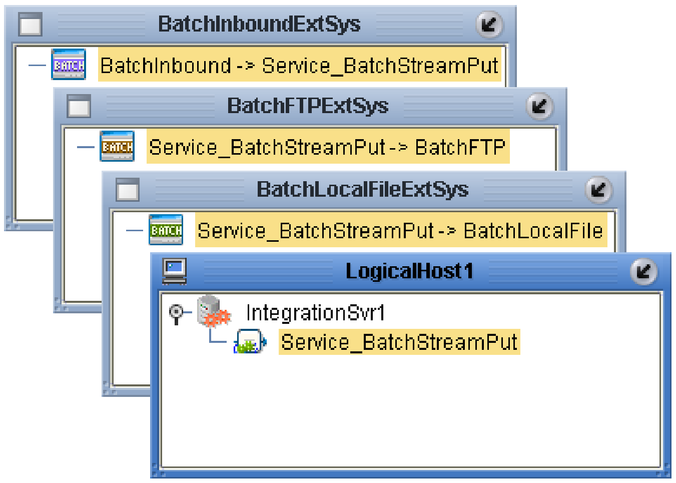
- The **BatchFTP** component eWay uses a Target File Name of DATA_FOR_STREAM.out.
- The **BatchLocalFile** component eWay uses a Target File Name of DATA_TOBE_STREAMED_UP.txt

7.10.5 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Collaborations and message destinations to the integration server and message server. Deployment profiles are created using the Deployment Editor.

1. From the Enterprise Explorer's Project Explorer, right-click the Project (**Batch_Stream_Put**) and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **BatchStreamPut_DP**). Make sure that the selected Environment is **BatchStreamPut_Env**. Click **OK**.
3. From the Deployment Editor toolbar, click the **Auto Map** icon. The Projects components are automatically mapped to their system windows.

Figure 115 BatchStreamPut_DP Deployment Profile


BatchInboundExtSys
BatchInbound -> Service_BatchStreamPut
BatchFTPExtSys
Service_BatchStreamPut -> BatchFTP
BatchLocalFileExtSys
Service_BatchStreamPut -> BatchLocalFile
LogicalHost1
IntegrationSvr1
Service_BatchStreamPut

4. Click **Activate**. When activation succeeds, save your current changes to the Repository.

7.10.6 Running the Project

To run the Project follow the directions provided in **"Running the Project" on page 218**.

Index