SeeBeyond ICAN Suite

Batch eWay Intelligent Adapter User's Guide

Release 5.0

Contents

Chapter 1

Introducing the Batch eWay

This chapter provides a brief overview of operations and components, general features, configuration, and system requirements for the Batch eWay.

Chapter Topics

1.1 Overview

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.

1.1.1 Batch eWay OTDs

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

The Batch eWay includes an OTD for connecting to external FTP servers, a record-processing OTD, and a local file OTD. The record-processing OTD extracts records out of files, parse files into specific records, and define the content of files as records. The local file OTD picks up or puts data files to local file systems.

Note: *The Batch eWay supports standard FTP according to RFC-959.*

1.2 Supported Operating Systems

The Batch eWay is available on the following operating systems:

- Windows XP SP1a
- Windows Server 2003
- Windows 2000 SP3
- Solaris 8 and 9
- AIX 5.1 and AIX 5.2
- HP-UX 11.0 and HP-UX 11i (RISC)
- HP Tru64 V5.1A
- Red Hat Linux 8 (Intel)
- Red Hat Linux Advanced Server 2.1 (Intel)

1.3 System Requirements

To set up and run the Batch eWay with the eGate Enterprise Designer, you will need the following:

- The Java 2 Software Development Kit (SDK) Version 1.4.1 (Standard Edition) installed on the Enterprise Manager machine.
- Windows XP, Windows Server 2003, or Windows 2000 SP3. This is a requirement for the User Interface.
- Microsoft Internet Explorer 6.0 SP1 or above.
- The JAVA_HOME variable set to the location of the SDK.

Note: *To enable web services, you must install and configure eInsight.*

Chapter 2

Installing the Batch eWay

This chapter contains installation information for the Batch eWay.

Chapter Topics

2.1 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.

2.1.1. Installing the Batch eWay on an eGate Supported System

The Batch eWay is installed during the installation of the eGate Integrator. The eGate installation process includes the following operations:

- Installing the eGate Repository
- Uploading products to the Repository
- Downloading components (such as the SeeBeyond Enterprise Designer and Logical Host)
- Viewing product information home pages

Follow the instructions for installing the eGate Integrator in the *eGate Integrator Installation Guide*, and include the following steps:

1. During the procedures for uploading files to the eGate Repository using the Enterprise Manager, after uploading the **eGate.sar** file, select and upload the following files:
   - **BatcheWay.sar** (to install the Batch eWay)
   - **FileeWay.sar** (to install the File eWay, used in the sample Project)
   - **BatcheWayDocs.sar** (to download the Batch eWay user's guide)

2 Continue installing the eGate Integrator as instructed in the *eGate Integrator Installation Guide*

2.1.2. 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 eWay Properties

This chapter explains how to configure the Batch eWay properties.

Chapter Topics

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** (two properties templates - accessed Connectivity Map and Environment Explorer tree)
- **BatchLocalFile**
- **BatchRecord**

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 BatchFTP eWay also contains properties that must be accessed and configured from the **Environment Explorer tree**. These parameters are commonly global, applying to all eWays (of this same type) in the project.

The properties for the BatchFTP eWay must be set in both locations. BatchLocalFile and BatchRecord properties are only accessed and modified from the Connectivity Map.

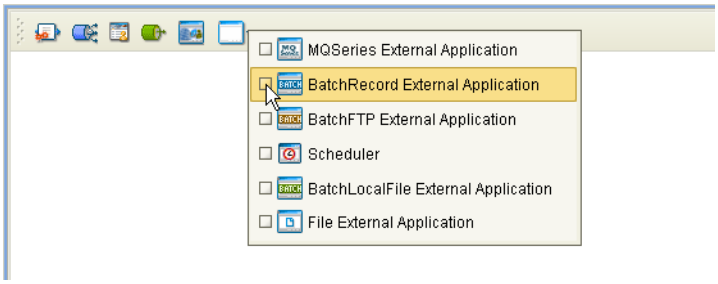
3.1.1 Selecting Batch as the External Application

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 Java 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.

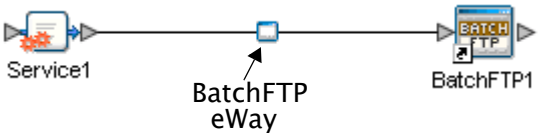
Figure 1 External Applications Selection Menu



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 Sheet** .

3.1.2 Creating Custom Properties for a Batch eWay

A Project's eWay properties can be modified after the eWays have been established in the Connectivity Map and the Environment has been 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 Sheet** 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 Sheet** appears.
2. Make any necessary modifications to the Environment parameters of the Batch eWays, and click **OK** to save the settings.

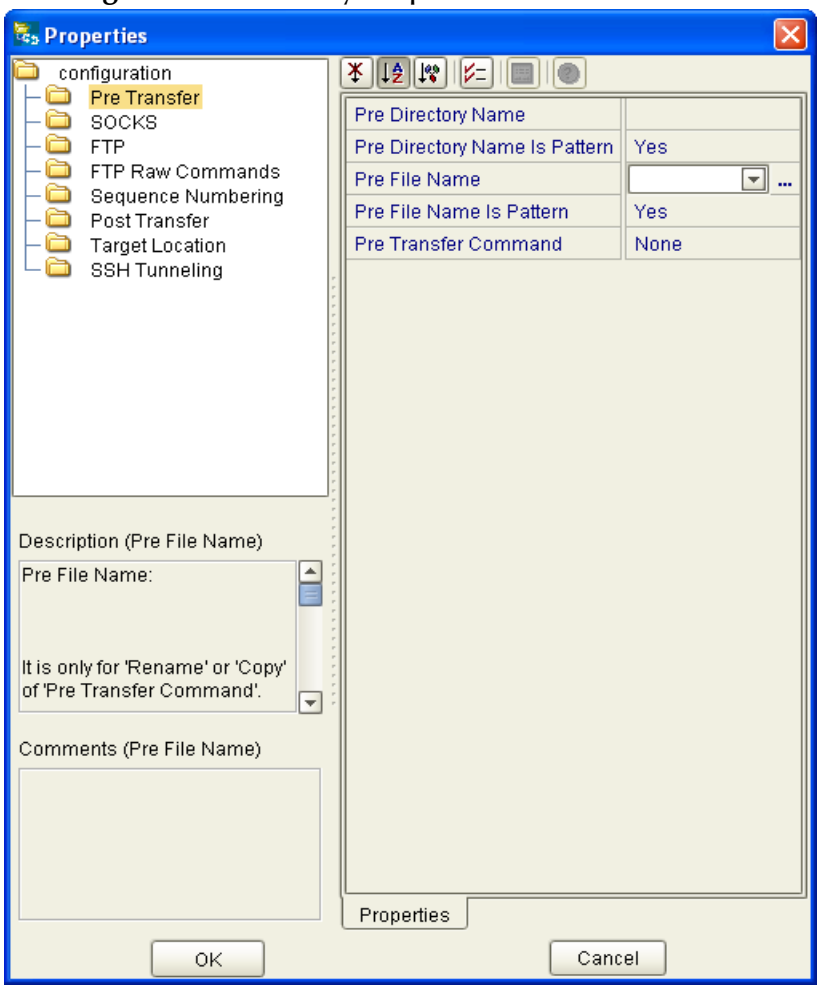
3.1.3. Using the Properties Sheet

Modifications to an eWay's properties are made from the Batch eWay's Properties Sheet .

To modify the default eWay configuration properties

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

Figure 3 Batch eWay Properties Sheet



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 Sheet and save your current changes.

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

3.2 Batch eWay Properties

The Batch eWay's Properties are organized as follows:

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

3.3 BatchFTP eWay Connectivity Map Parameters

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

The BatchFTP Connectivity Map properties include the following sections:

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

3.3.1 Pre Transfer

This section allows you to configure the **Pre Transfer** parameters. Pre-transfer operations are those performed before the file transfer.

Note: *For more information on this feature, see* ***"Pre/post File Transfer Commands" on page 126****.*

Pre Directory Name

Description

Allows you to specify 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 140** 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

Allows you to indicate 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

Allows you to specify 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** means 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 140** 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

Allows you to indicate 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 NT 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

This section provides information about configuring the **SOCKS** secure FTP configuration parameters. The eWay supports the following negotiation methods:

- No-authentication
- User/password

For more information on SOCKS, see **"SOCKS Support" on page 148**.

Socks Enabled

Description

Allows you to specify 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 35** *for more details.*

Required Values

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

Socks Version

Description

Allows you to specify 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

This section allows you to configure the **FTP** parameters.

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, NT 4.0, NT 3.5, HCLFTPD 5.1, HCLFTPD 6.0.1.3, VMS, MSFTPD 2.0, MVS PDS**, **MVS GDG**, **MVS Sequential, VM/ESA, Netware 4.11, AS400**, **AS400-UNIX**, **MPE, User Defined,** or a user named and created style.

Note: *For more information, see* **"Using FTP Heuristics" on page 50***.*

Mode

Description

Allows you to specify 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**.

User Name

Description

When a log on to the external system is required, enter the appropriate user name.

Required Values

A valid user name.

3.3.4 FTP Raw Commands

This section provides information about configuring the **FTP Raw Commands** parameters. FTP raw commands are commands that are sent directly to the FTP server.

Post Transfer Raw Commands

Description

Allows you to specify 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

Allows you to specify 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

This section allows you to configure the **Sequence Numbering** parameters.

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

This section allows you to configure the **Post Transfer** configuration parameters. Post-transfer operations are those performed after the file transfer.

Note: *For more information on this feature, see* **"Pre/post File Transfer Commands" on page 126***.*

Post Directory Name

Description

Allows you to specify 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** means 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 140** 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

Allows you to indicate 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

Allows you to specify 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 140** 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

Allows you to indicate 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 NT 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 Target Location

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

Append

Description

Allows you to specify 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

Allows you to specify 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

Allows you to indicate 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 the home directory of the user unless an absolute path is given.*

Required Values

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

Target File Name

Description

Allows you to specify 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

Allows you to indicate 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**.

3.3.8 BatchFTP OTD and SSH Tunneling

This section provides a brief explanation of how the BatchFTP OTD supports Secure Shell (SSH) tunneling.

Note: *SSH tunneling is also known as port forwarding.*

Additional SSH-supporting Software

The eWay's SSH tunneling feature depends on your using an existing SSH-supporting software application, for example, Plink on Windows or OpenSSH on UNIX.

For different SSH client implementations, the command syntax and environment configuration can be different. See your SSH-supporting application user's guides for details. For more information, see the following Web site:

http://www.openssh.com

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:apple:21 -o BatchMode=yes apple
```

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 **apple:21** through an SSH tunnel. For more information on SSH tunneling, see **"SSH Tunneling" on page 149**.

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

3.3.9 SSH Tunneling

This section provides information for configuring the **SSH Tunneling** secure FTP configuration parameters.

SSH Channel Established

Description

Allows you to specify whether the eWay needs to launch an SSH subprocess.

Selecting **No** means 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**

If you select **No**, setting the following parameters is optional:

- **SSH User Name**
- **SSH Password**

Selecting **Yes** means 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**
- **SSH Listen Port**

Required Values

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

SSH Command Line

Description

Allows you to enter 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 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 Listen Host

Description

Allows you to specify 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

A valid SSH listen host name; the default is **localhost.**

SSH Listen Port

Description

Allows you to specify 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

Allows you to specify 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

A valid SSH password.

SSH Tunneling Enabled

Description

Allows you to specify 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**.

SSH User Name

Description

Allows you to specify 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

A valid SSH user name.

3.4 BatchFTP eWay Environment Explorer 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:

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

This section provides information about configuring the **SOCKS** secure FTP configuration parameters. The eWay supports the following negotiation methods:

- No-authentication
- User/password

For more information on SOCKS, see **"SOCKS Support" on page 148**.

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

Allows you to specify 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

Allows you to enter 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

Allows you to specify 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

This section allows you to configure the **FTP** parameters.

Host Name

Description

Allows you to specify 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" on page 30** 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" on page 21** for details.

Required Values

A valid 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

A valid password.

Server Port

Description

Allows you to specify 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" on page 30** for details.

Required Values

A valid server port number.

User Name

Description

When a log on to the external system is required, enter the appropriate user name.

Required Values

A valid user name.

3.4.3 SSH Tunneling

This section provides information for configuring the **SSH Tunneling** secure FTP configuration parameters.

SSH Listen Host

Description

Allows you to specify 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

A valid SSH listen host name; the default is **localhost.**

SSH Listen Port

Description

Allows you to specify 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

Allows you to specify 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

A valid SSH password.

SSH User Name

Description

Allows you to specify 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

A valid SSH user name.

3.5 BatchLocalFile Configuration Parameters

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

The BatchLocalFile properties include the following sections:

Caution: *Several of these configuration options allow for or 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 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.*

3.5.1 Pre Transfer Configuration

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

Note: *For more information on this feature, see* **"Pre/post File Transfer Commands" on page 126***.*

Pre Directory Name

Description

Allows you to specify 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

Allows you to specify the meaning of the **Pre Directory Name** parameter as follows:

- **Yes** means 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** means 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

Allows you to specify 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

Allows you to specify the meaning of the **Pre File Name** parameter as follows:

- **Yes** means 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** means 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**.

3.5.2 Sequence Numbering

This section allows you to configure the **Sequence Numbering** parameters.

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**.

3.5.3 Post Transfer

This section allows you to configure the **Post Transfer** parameters. Post-transfer operations are those performed after the data transfer

Post Directory Name

Description

Allows you to specify 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

Allows you to specify the meaning of the **Post Transfer Name** parameter as follows:

- **Yes** means that the **Post Transfer Name** (file or directory) represents a pattern to be used for name expansion.
- **No** means 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

Allows you to specify 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

Allows you to specify the meaning of the **Post Transfer Name** parameter as follows:

- **Yes** means that the **Post Transfer Name** (file or directory) represents a pattern to be used for name expansion.
- **No** means 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.

3.5.4 General Settings

This section provides information about configuring the **General Settings** parameters.

Resume Reading Enabled

Description

Allows you to specify 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**: Means 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**.

3.5.5 Target Location

This section provides information about configuring the **Target Location** parameters.

Append

Description

Allows you to specify 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

Allows you to specify 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

Allows you to specify the meaning of the **Target Directory Name** parameter as follows:

- **Yes** means 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** means that the **Target Directory Name** represents the exact directory name to be used for the transfer. No pattern matching of any kind is performed.

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

Required Values

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

Target File Name

Description

Allows you to specify 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

Allows you to specify the meaning of the **Target File Name** parameter as follows:

- **Yes** means 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** means that the **Target File 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 **Yes**.

3.6 BatchRecord Configuration Parameters

This section explains the configuration parameters for the record-processing **BatchRecordOTD,** accessed from the Connectivity Map.

- The BatchRecord properties include the following sections:

3.6.1 General Settings

This section provides you with configuration information for the **General Settings** parameter, **Parse or Create Mode**.

Parse or Create Mode

Description

Allows you to specify 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**.

Parse

Allows you to specify 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**.

Required Values

Parse.

3.6.2 Connector

This section allows you to configure the eGate Collaboration engine to identify the eWay with the record-processing OTD.

Class

Description

Allows you to specify the class name of the Batch eWay OTD connector object.

Required Values

A valid class name. The default is **com.stc.eways.batchext.BatchRecordConnector**.

Property.Tag

Description

Allows you to identify the data source. This parameter is required by the current **EBobConnectorFactory**.

Required Values

A valid data source package name. Accept the default.

Type

Description

Allows you to specify the type of OTD.

Required Values

The eGate name of the record-processing OTD. The value defaults to **BatchRecordOTD**.

3.6.3 Record

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

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

Allows you to enter 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.

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.*

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 you are using escaped ASCII, Hex, or Octal, the "\" character is required.*

Required Values

A valid data record delimiter.

Record Size

Description

Allows you to specify a number indicating the record size. Use this parameter when the **Record Type** is set to **Fixed**, and a number indicating length must be supplied. The number specifies the byte count of each record.

Required Values

An integer from 1 to 2,147,483,647.

Record Type

Description

Allows you to specify the format of the records in the data payload in the Collaboration.

Each payload can contain zero or more records. Using this and related parameters, it is possible to pass the records individually to another component within eGate. Select one of the following options:

- **Delimited**: The records are separated by the delimiter specified under the **Record Delimiter** parameter.
- **Fixed**: The records are all of a given size; the size of each record is specified by the **Record Size** parameter.
- **Single record**: If the payload is to be processed "as-is," select this option.

Note: *If you select **User Defined**, you must enter a Java class name under the **User Class** (see **"User Class" on page 49**) configuration settings.*

Required Values

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

3.6.4 User Class

This section allows you to configure the **User Class** parameters.

User Class

Description

Allows you to specify the name of a Java class you create. This is an advanced parameter and allows for user extensibility of the record-parsing capabilities of the eWay.

This option is only used when the **User Defined** parameter is selected under the **Record Type** (see **"Record Type" on page 49**) settings in the **Record** configuration. In this case, you must enter the full class name of your class, for example:

com.mycompany.batch.MyParser

Required Values

A valid Java class name. This class must either implement the **BatchRecordParser** interface or extend one of the SeeBeyond-supplied implementations.

User Properties

Description

Allows you to specify the fully qualified file name of a Java properties file. This is an advanced parameter and is part of the user-extensibility features of the record-parsing capabilities of the eWay.

This option is only used when the **User Defined** parameter is chosen under the **Record Type** (see **"Record Type" on page 49**) settings, and you have supplied a class name for the **User Class** parameter. This parameter is ignored by all SeeBeyond-supplied parser implementations.

This parameter is optional but, if supplied, the full path must be given. If a file name is supplied, it is loaded and passed to your implementation class as a Java **Properties** object immediately after construction.

The format of the file is totally user-defined and is not interpreted by eGate or the eWay in any way. In this way, you can create the file manually in a text editor ahead of time or dynamically on the fly, as long as it exists before the initialization of the eWay at run time.

Required Values

The fully qualified file name for the Java properties file; this parameter is optional.

3.7 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 Sheet . 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 Sheet 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
- Windows NT 4.0
- Windows NT 3.5
- HCLFTPD 5.1
- HCLFTPD 6.0.1.3
- VMS
- MSFTPD 2.0
- MVS PDS (Partition Data Sets)
- MVS Sequential
- MVS GDG (Generation Data Group)
- VM/ESA
- Netware 4.11
- AS400
- AS400-UNIX
- MPE
- User Defined
- *Additional platforms and properties created by the user if applicable.*

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 the user to modify the properties for an unlisted platform (see **Modifying the FTP Heuristics** on page 52).

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 22).

3.7.1 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.

3.7.2 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 50.

The FTP Heuristics Configuration Parameters are as follows:

Commands Supported by FTP Server

Description

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

Required Values

One or more FTP commands as selected from the list.

Header Lines To Skip

Description

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

Required Values

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

Additional Information

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

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

Header Indication Regex Expression

Description

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

Required Values

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

Additional Information

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

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

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

Trailer Lines To Skip

Definition

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

Required Values

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

Trailer Indication Regex Expression

Definition

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

Required Values

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

Additional Information

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

Directory Indication Regex Expression

Definition

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

Required Values

A regular expression. If there is no reliable way of identifying the 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 1).

Table 1 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.()
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**.

Chapter 4

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.*

Chapter Topics

4.1 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.

4.2 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 BatchLocalFile configuration node:

- read
- write

For business process operations, the Batch eWay has the following operators available under the BatchFTP configuration node:

- get
- put

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.

4.3 The Batch eWay eInsight Sample Projects

This chapter provides three sample Projects that demonstrate how eInsight Business Processes are used with the Batch eWay:

- **The Batch_BP_FTPIn Sample Project** on page 64, that includes the BatchFTP eWay.
- **The Batch_BP_LFIn Sample Project** on page 76, that includes the BatchLocalFile eWay.
- **The Batch_BP_LFOut Sample Project** on page 84, that includes the BatchLocalFile eWay.

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.

4.4 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 uploaded with the eWay's documentation .sar file and downloaded from the Enterprise Manager's Documentation tab. Extract the samples from the Enterprise Manager to a local file.
2. From the File Destination dialog box, select **Import to a new Project**, and enter the name of the Project (for this sample, **Batch_BP_FTPIn**).
3. Browse to the directory that contains the sample project zip file. Select the sample file (for this sample, **Batch_BP_FTPIn.zip**) and click **Open**.
4. When the import has successfully completed, right-click the Repository and select **Refresh All from Repository** from the shortcut menu.
5. Before an imported sample Project can be run you must do the following:
   - Create an **Environment** (see **Creating an Environment** on page 71)
   - Configure the eWay Properties for your specific system (see **Creating and Configuring the Batch eWay** on page 14)
   - Create a **Deployment Profile** (**Creating and Activating the Deployment Profile** on page 74)

4.5 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.

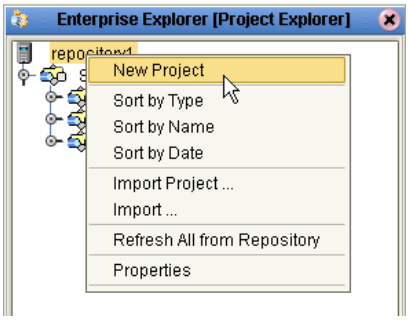
The following pages provide step by step directions for manually creating the Batch_BP_FTPIn_Sample Project.

4.5.1. 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 4 on page 65**). A new Project (Project1) appears on the Project Explorer tree.

Figure 4 Enterprise Explorer - New Project



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

4.5.2 Creating the BP_FTPIn Business Process

Creating the Business Process Flow

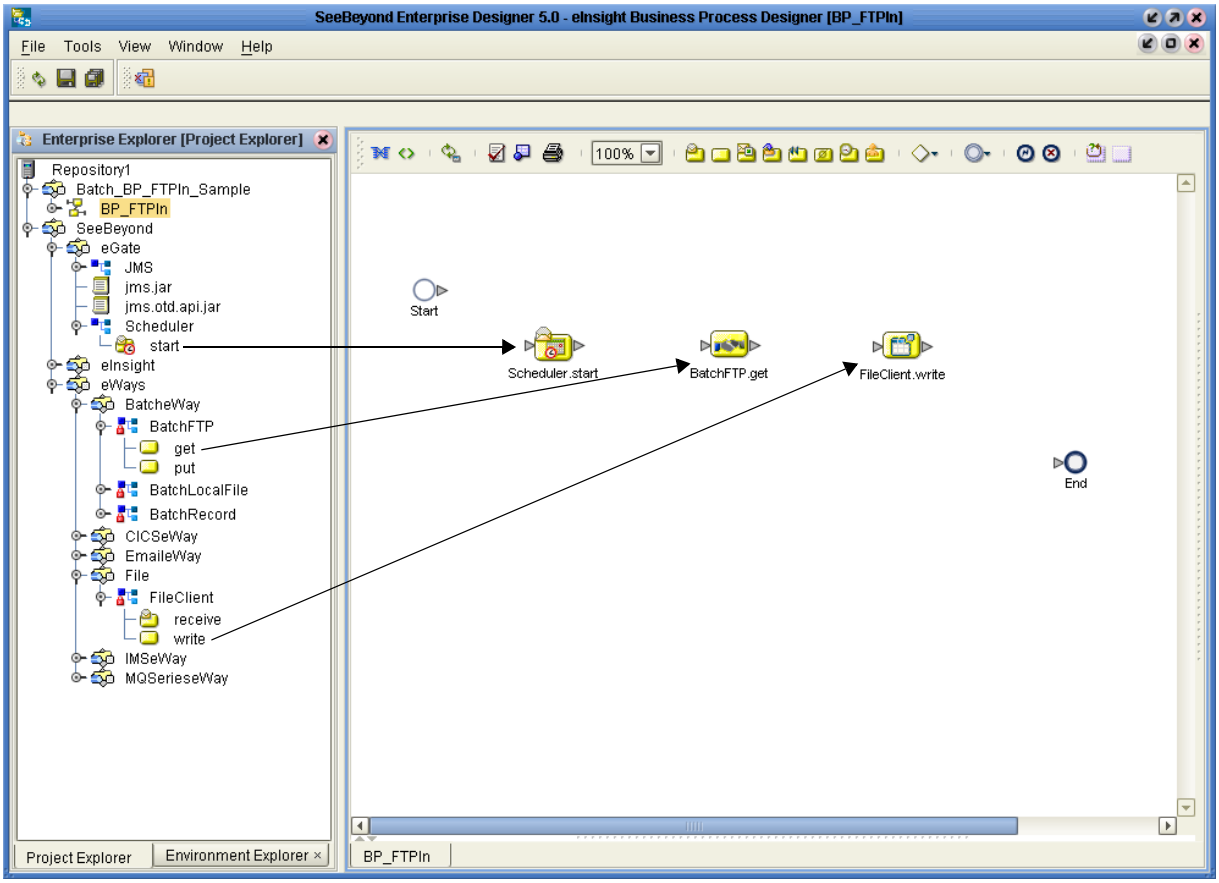
1 From the Enterprise Designer's Project Explorer tree, right-click **Batch_BP FTPIn_Sample**, and select **New** > **Business Process** from the shortcut menu. The eInsight Business Process Designer appears and **BusinessProcess1** is added to the Project Explorer tree. Rename the Business Process to **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 5 on page 66**:

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

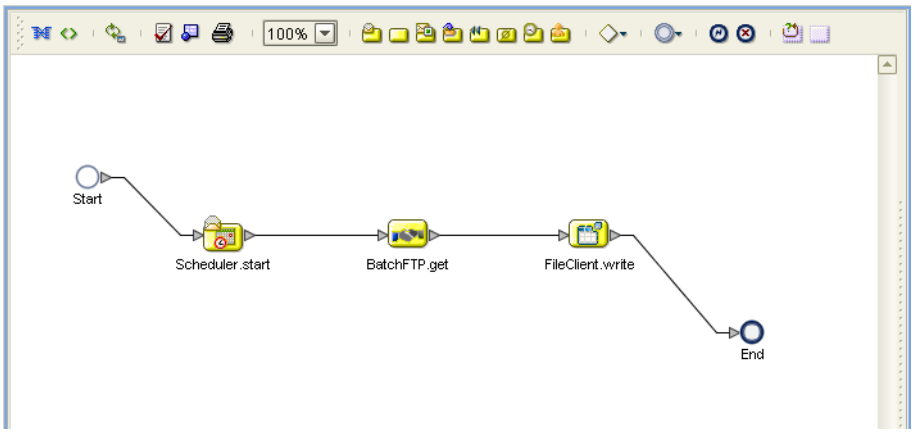
Figure 5 eInsight Business Process Designer - Populate the Canvas



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 6 on page 67**.

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

Figure 6 eInsight Business Process Designer - Link the Modeling Elements

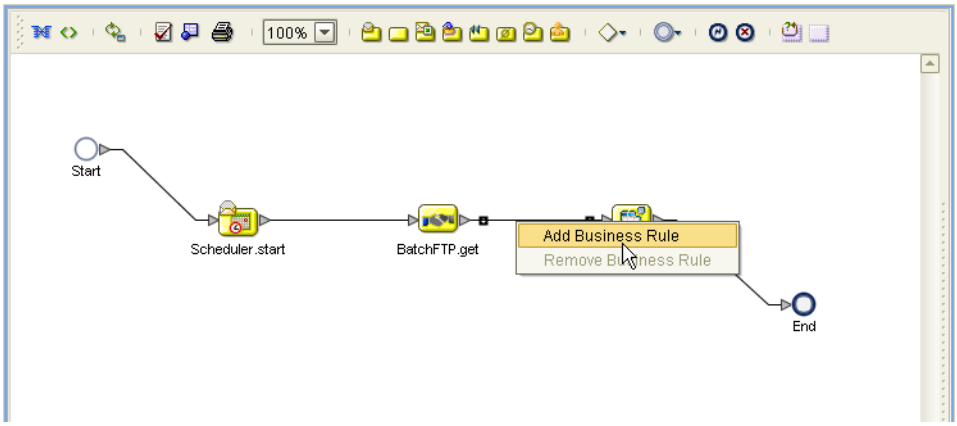


Configuring the Modeling Elements

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 7.

Figure 7 eInsight Business Process Designer - Adding Business Rules



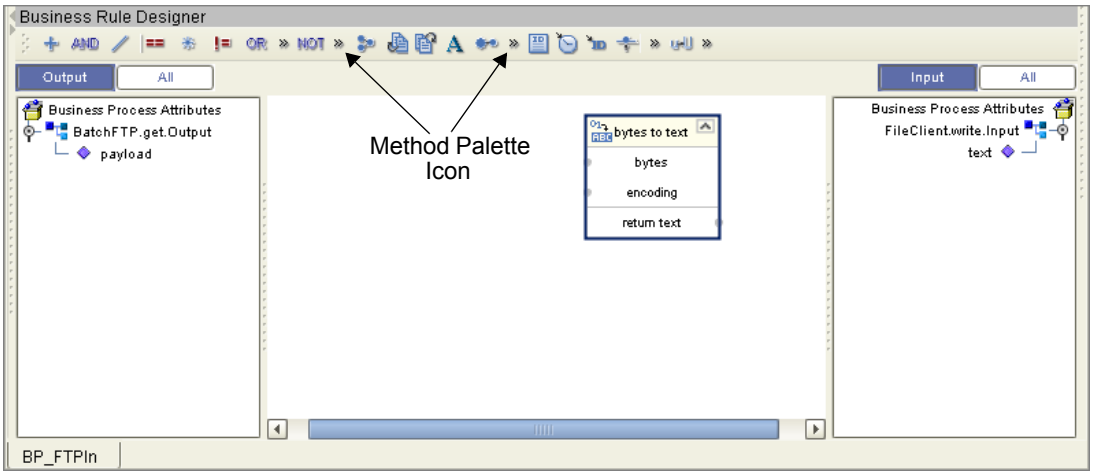
2 From the eInsight Business Process Designer toolbar, click the **Map Business Process Attributes** button. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.

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 toolbar, click the Method Palette icon (see **Figure 8 on page 68**). 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 toolbar, drag and drop the **bytes to text** icon onto the Business Rule Designer canvas. The **bytes to text** method box appears (see Figure 8).

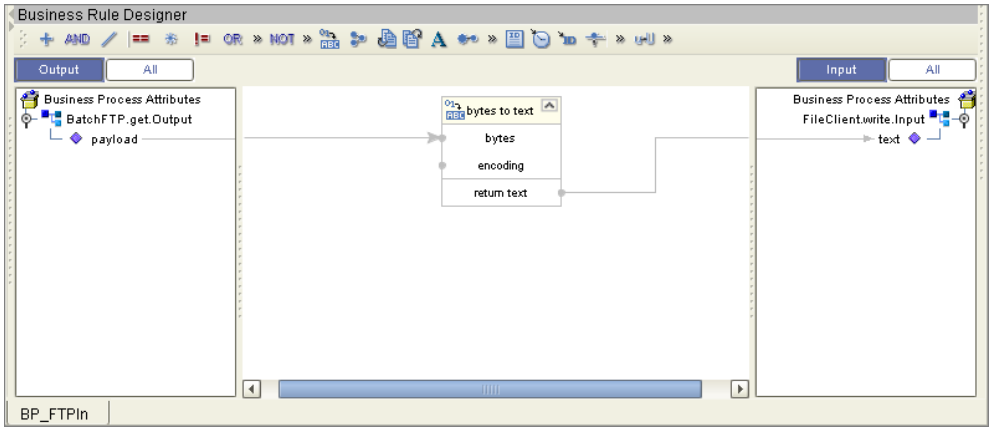
Figure 8 eInsight Business Rule Designer



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. The Business Process Designer (see Figure 9).

Figure 9 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.

4.5.3 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.

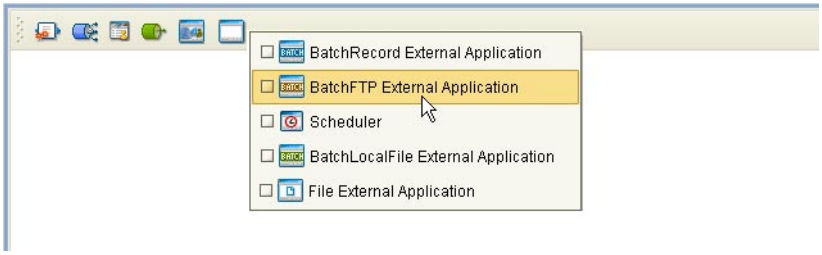
Select the External Applications

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 10).

Figure 10 Connectivity Map - External Applications



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

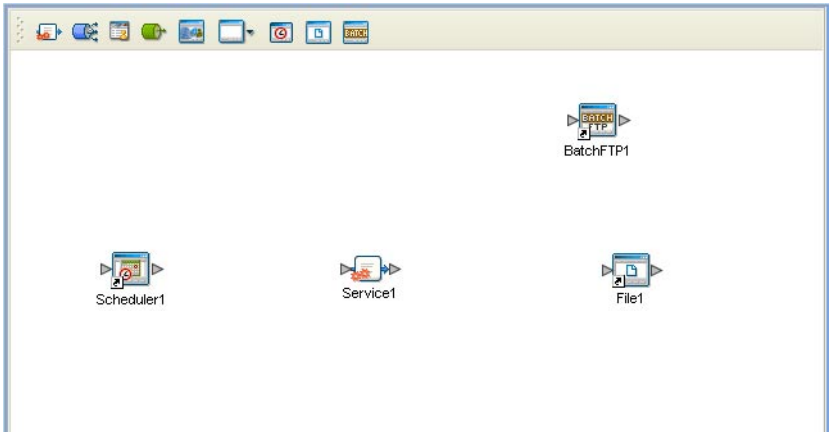
Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

1. Drag the following components onto the Connectivity Map canvas as displayed in **Figure 11 on page 70**:
   - **Scheduler**
   - **Service** (A service is a container for Java 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 11 Connectivity Map with Components



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

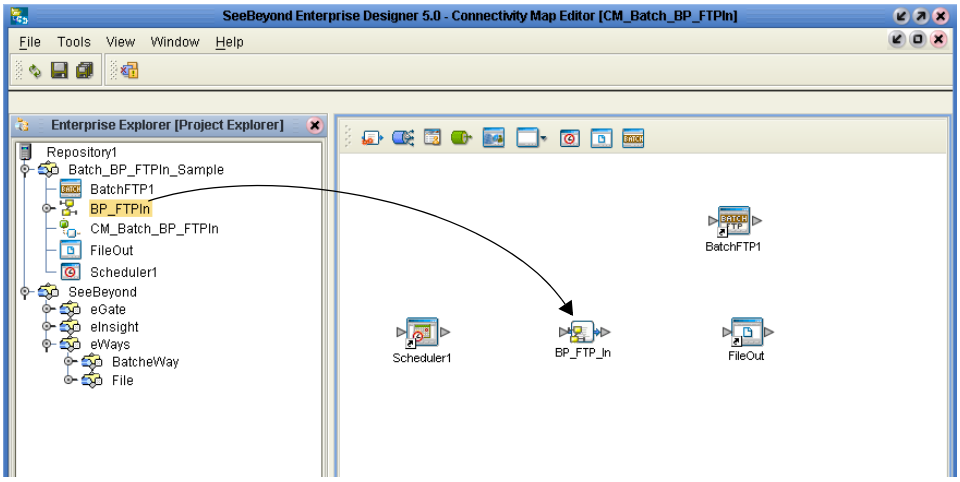
4.5.4. Binding the Project Components

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 12).

Figure 12 Connectivity Map - Binding the Components



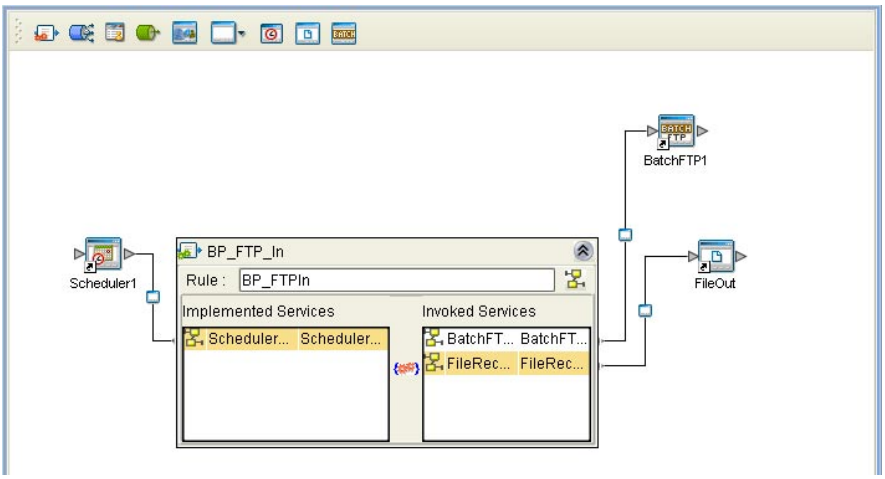
3 Double-click the **BP_FTP_In** service. The **BP_FTP_In** binding box appears with the BP_FTP_In Rule.

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 13).

Figure 13 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.

4.5.5. 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.

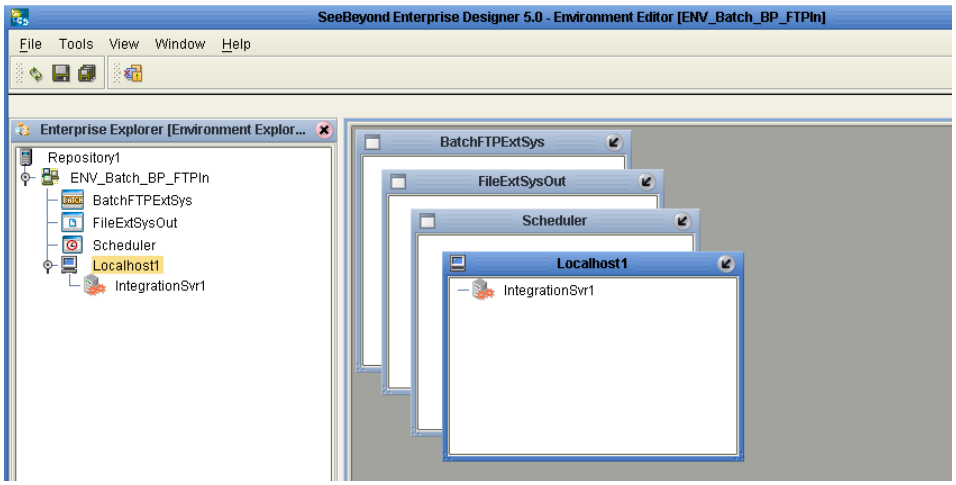
6 From the Project Explorer tree, right-click **ENV_Batch_BP_FTPIn** and select **New Scheduler**. Name the External System **Scheduler**. Click **OK**. The **Scheduler** window is added to the Environment Editor.

7 From the Project Explorer tree, right-click **ENV_Batch_BP_FTPIn** and select **New Logical Host**. Enter **Localhost1** in the **Logical Host Name** field. Select **SeeBeyond JMS IQ Manager** as the System JMS Type. The **Localhost1** window is added to the Environment Explorer tree.

8 From the Environment Explorer tree, right-click **Localhost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under Localhost1.

9 Save changes to the Repository. The Environment Explorer and Environment Editor now appear as displayed in Figure 14.

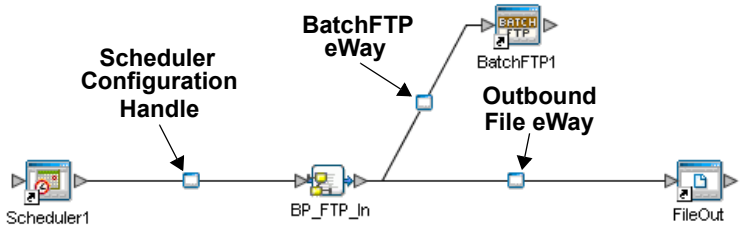
Figure 14 Environment Editor



4.5.6. 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 15).

Figure 15 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 2, and click **OK**. The properties are saved for the eWay.

Table 2 Outbound File eWay Settings

Outbound File eWay Properties	
Directory	C:/temp
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 14 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 3, and click **OK**.

Table 3 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 4, and click **OK**.

Table 4 BatchFTP Environment Explorer eWay Settings

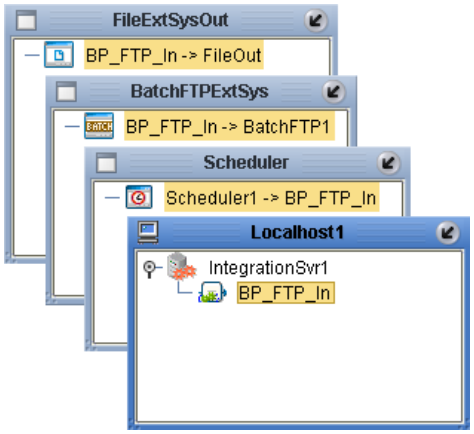
BatchFTP eWay Environment Explorer Properties	
FTP Set as directed, otherwise use the default settings.	
Host Name	*The name of the external system that the eWay connects to.*
Password	*Password required to log into the external system*
Server Port	*Port number to use to connect to the FTP server*
User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

4.5.7 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. From the left pane of the Deployment Editor, drag **BP_FTP_In -> FileOut** (External Application) to the **FileExtSysOut** window.
4. From the left pane of the Deployment Editor, drag the **BP_FTP_In -> BatchFTP1** (External Application) to the **BatchFTPExtSys** window.
5. Drag **Scheduler1 -> BP_FTP_In** (Application) to the **Scheduler** window.
6. From the left pane of the Deployment Editor, drag **BP_FTP_In** (Business Process) to **IntegrationSvr1** in the **Localhost1** window (see Figure 16).

Figure 16 Deployment Profile



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

4.5.8. 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\LogicalHost1** directory. You must specify the **LogicalHost1** directory for it to be created.
3. Navigate to **C:\ican50\LogicalHost1\bootstrap\config** directory and open the **logical-host.properties** file using Notepad™.
4. Enter the following information in the appropriate fields:
   - Logical Host root directory: **ican50\LogicalHost1**
   - Repository URL: **http://localhost:*port number/repository name***
   - Repository user name and password: ***Your user name and password***
   - Logical Host Environment name: **ENV_Batch_BP_FTPIn**
   - Logical Host name: **LogicalHost1**

   Save your changes to **logical-host.properties** and close the file.
5. Run the **bootstrap.bat** file in the ican50\LogicalHost1\bootstrap\bin directory.
6. Copy the sample input data file to the input directory.

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

4.5.9 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 and higher. To enable Logging, please see the ***eGate Integrator User's Guide***.

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

4.6 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:

4.6.1. Create a Project

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

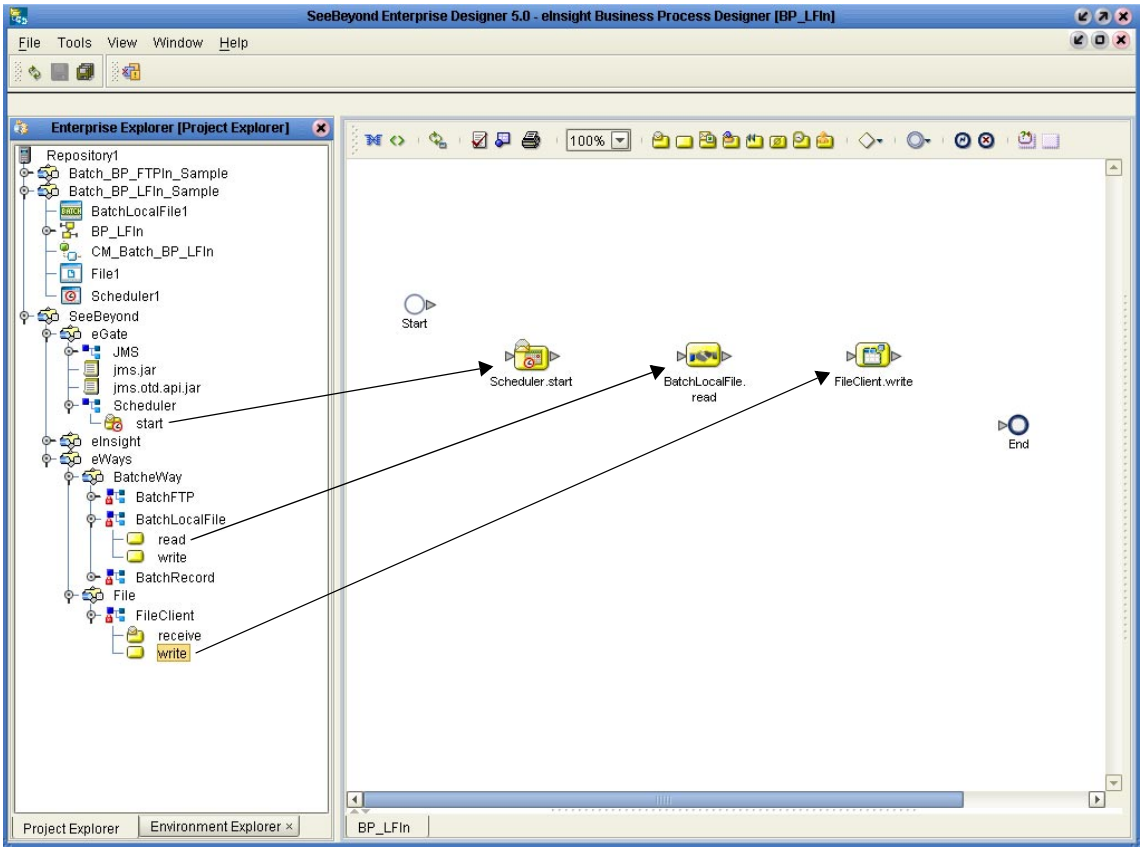
1. Start the Enterprise Designer.
2. From the Enterprise Explorer's Project Explorer tab, select your Repository on the Project Explorer tree, right-click the Repository and select **New Project**. A new Project appears on the Project Explorer tree.
3. Click twice on **Project1** and rename the Project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

4.6.2 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 17 on page 77**:
   - **start**, under SeeBeyond > eGate > Scheduler
   - **read**, under SeeBeyond > eWays > BatcheWay > BatchLocalFile
   - **write**, under SeeBeyond > eWays > FileClient

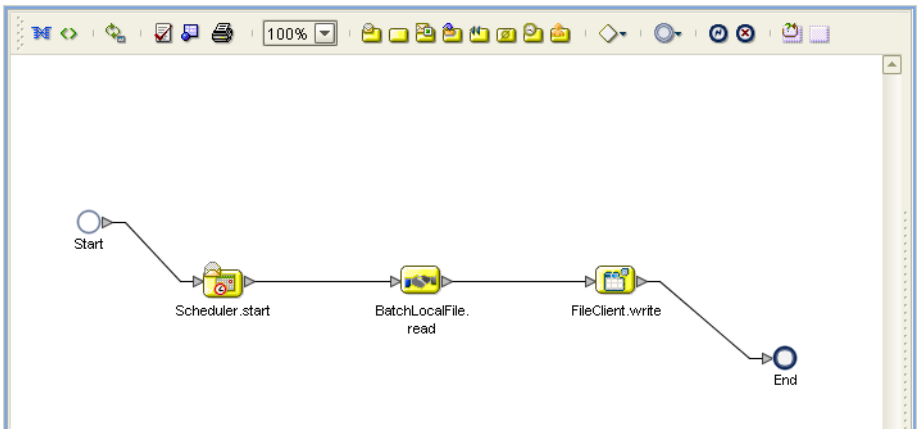
Figure 17 eInsight Business Process Designer - Populate the Canvas



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 18 on page 78**.

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

Figure 18 eInsight Business Process Designer - Link the Modeling Elements

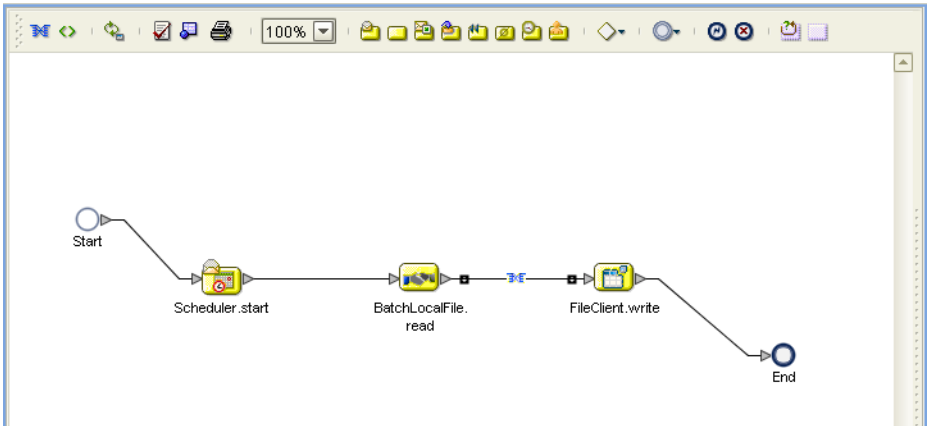


Configuring the Modeling Elements

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 19).

Figure 19 eInsight Business Process Designer - Adding Business Rules



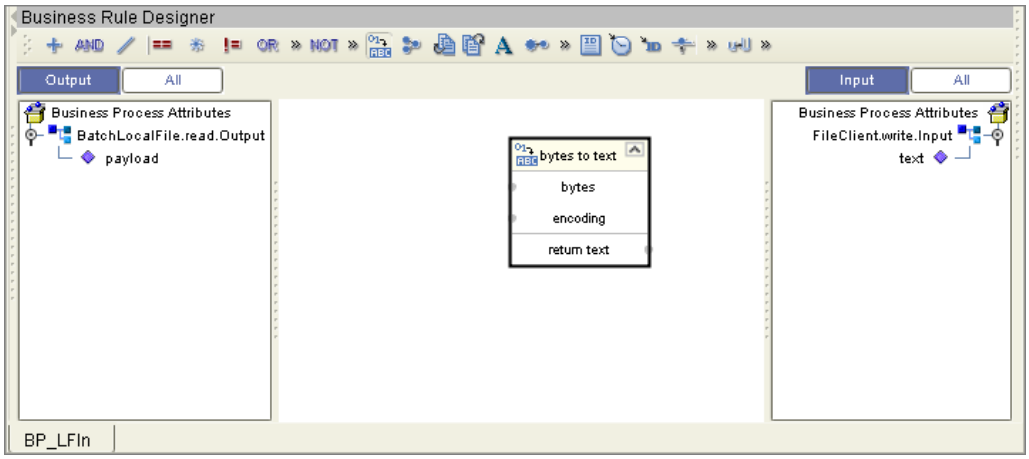
2 From the eInsight Business Process Designer toolbar, click the **Map Business Process Attributes** button. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.

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 toolbar, click the Method Palette icon. 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 toolbar, drag and drop the **bytes to text** icon onto the Business Rule Designer canvas. The **bytes to text** method box appears (see **Figure 20 on page 79**).

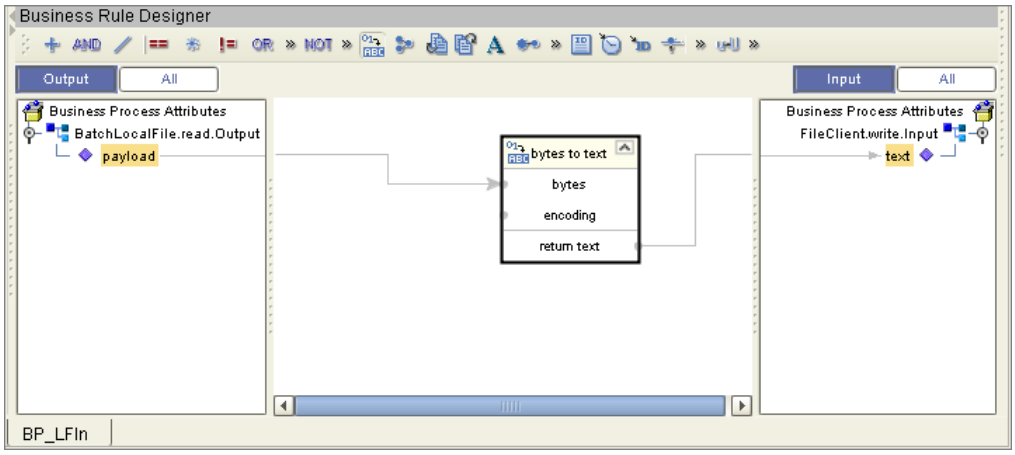
Figure 20 eInsight Business Rule Designer



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 21).

Figure 21 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.

4.6.3 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

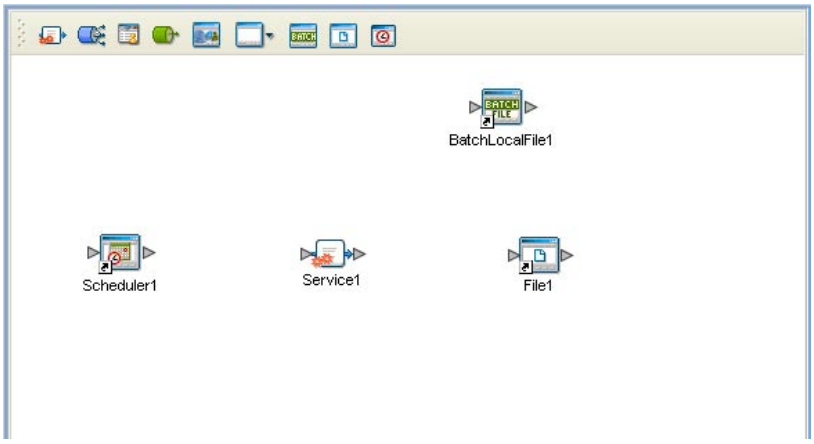
1. Click the **External Application** icon on the Connectivity Map toolbar,
2. Select the applications needed for your Project (for this sample, **Scheduler**, **BatchLocalFile External Application** and **File External Application**). Icons representing the selected applications are added to the Connectivity 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 22:
   - Scheduler
   - Service
   - BatchLocalFile External System
   - File External System

Figure 22 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.

4.6.4. 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.

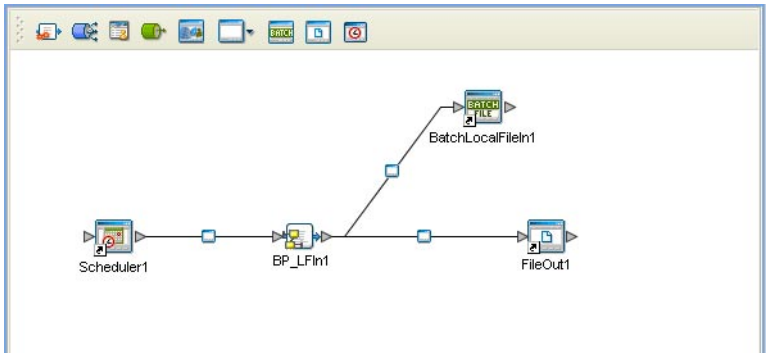
4 From the **BP_LFIn1** binding box, map **SchedulerStart** (under Implemented Services) to the **Scheduler** application.

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 23.

Figure 23 Connectivity Map - Connecting the Project's Components



8 Save the current changes to your Repository.

4.6.5. 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 14 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.

4.6.6. 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**. Enter **Localhost2** in the **Logical Host Name** field. Select **SeeBeyond JMS IQ Manager** as the System JMS Type. The **Localhost2** window is added to the Environment Editor.

8 From the Environment Explorer tree, right-click **Localhost2** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under Localhost2.

9 Save your current changes to the Repository.

4.6.7 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 **Localhost2** window.

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

4.6.8. 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, click on **Local Host**.

2 Extract the file to the **ican50\LogicalHost2** directory. You must specify the **LogicalHost2** directory for it to be created.

3 Navigate to C:\ican50\LogicalHost2\bootstrap\config directory and open the logical-host.properties file using Notepad™.

4 Enter the following information in the appropriate fields:

- Logical Host rood directory: **ican50\LogicalHost2**
- Repository URL: **http://localhost:*port number/repository name***
- Repository user name and password: ***Your user name and password***
- Logical Host Environment name: ENV_Batch_BP_LFIn
- Logical Host name: **LogicalHost2**

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

5 Run the **bootstrap.bat** file in the ican50\LogicalHost2\bootstrap\bin directory.

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

4.7 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 then 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:

4.7.1. 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_LFOut_Sample**).

4.7.2 Creating the BP_LFOut Business Process

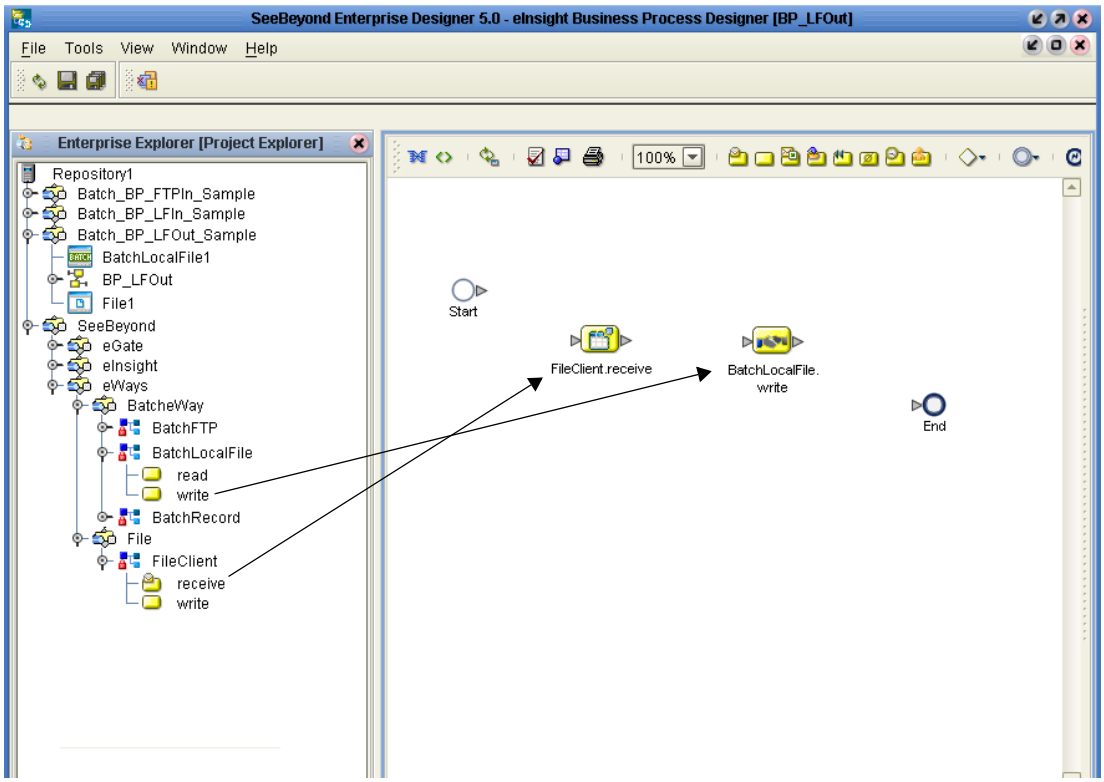
Creating the Business Process Flow

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 17 on page 77**:

- **receive**, under SeeBeyond > eWays > FileClient
- **write**, under SeeBeyond > eWays > BatcheWay > BatchLocalFile

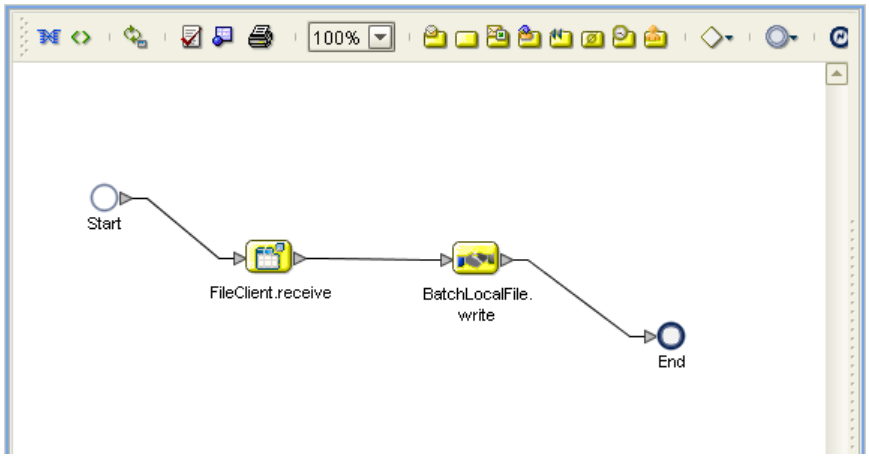
Figure 24 eInsight Business Process Designer - Populate the Canvas



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 25.

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

Figure 25 eInsight Business Process Designer - Link the Modeling Elements

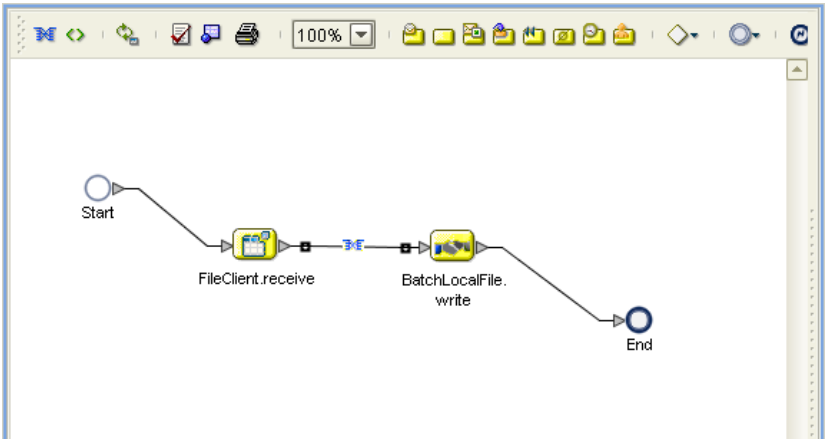


Configuring the Modeling Elements

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 (seeFigure 26).

Figure 26 eInsight Business Process Designer - Adding Business Rules



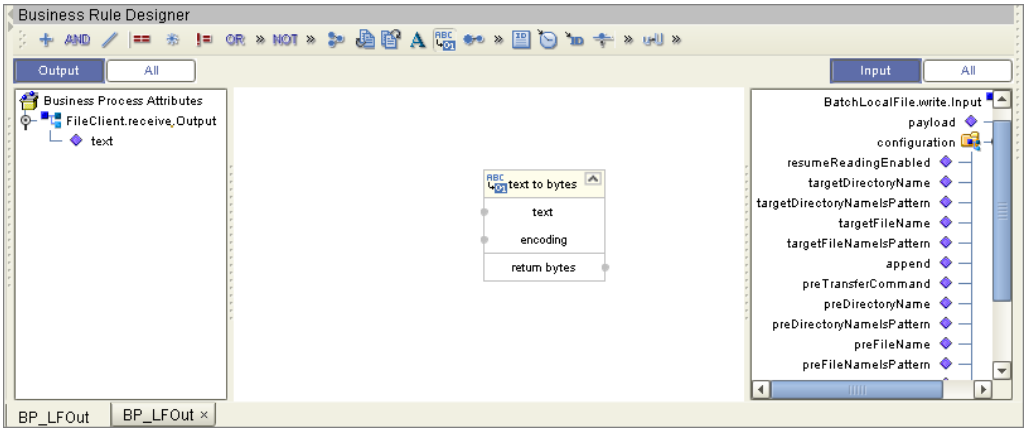
2 From the eInsight Business Process Designer toolbar, click the **Map Business Process Attributes** button. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.

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 toolbar, click the Method Palette icon. 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 toolbar, drag and drop the **text to bytes** icon onto the Business Rule Designer canvas. The **bytes to text** method box appears (see Figure 27).

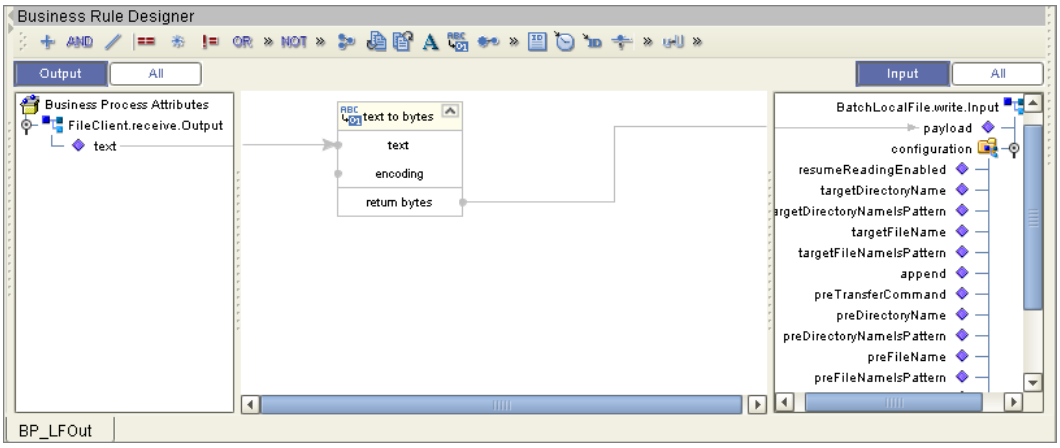
Figure 27 eInsight Business Rule Designer



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.

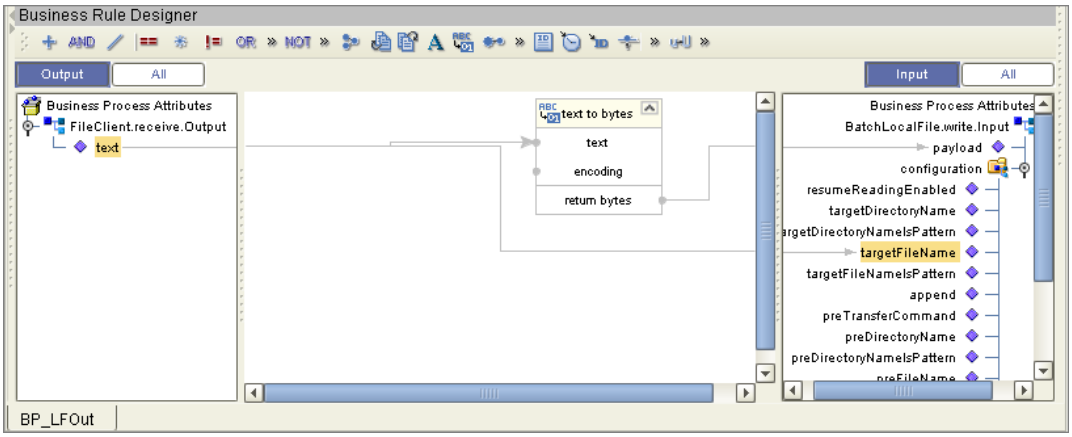
7 Map the **return bytes** output node of the **text to bytes** method box, to bytes to under BatchLocalFile.write.input in the Input pane of the Business Rule Designer. The Business Process Designer (see Figure 28).

Figure 28 eInsight 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 29).

Figure 29 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.

4.7.3 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_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

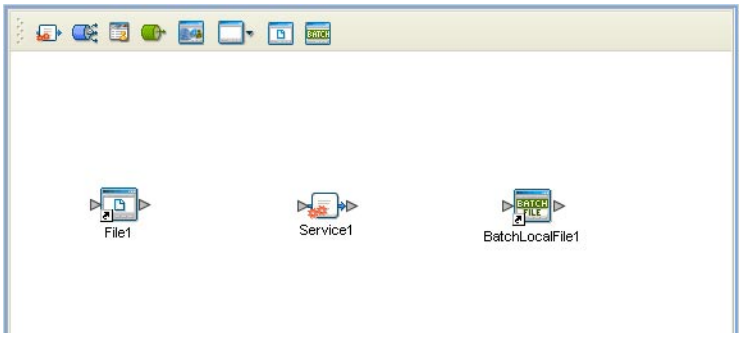
1. Click the **External Application** icon on the Connectivity Map toolbar,
2. Select the applications needed for your Project (for this sample, **BatchLocalFile External Application** and **File External Application**). Icons representing the selected applications are added to the Connectivity Map toolbar.

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 30:
   - File External System
   - Service
   - BatchLocalFile External System

Figure 30 CM_Batch_BP_LFIn Connectivity Map with Components



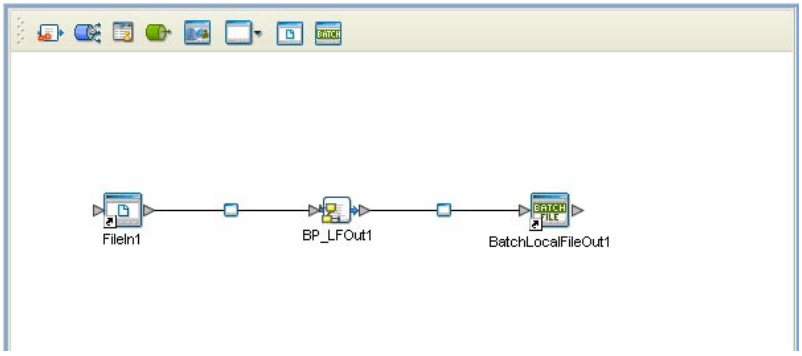
2. Rename **File1** to **FileIn1** by clicking the external application's name once and clicking it again. Enter the new name.
3. Rename **BatchLocalFile1** to **BatchLocalFileOut1**.
4. Rename **Service1** to **BP_LFOut1**.
5. Save your current changes to the Repository.

4.7.4. 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 31.

Figure 31 Connectivity Map - Connecting the Project's Components



7. Save the current changes to your Repository.

4.7.5. 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 14 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.

4.7.6. 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_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**. Enter **Localhost3** in the **Logical Host Name** field. Select **SeeBeyond JMS IQ Manager** as the System JMS Type. The **Localhost3** window is added to the Environment Editor.
7. From the Environment Explorer tree, right-click **Localhost3** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under LogicalHost3.
8. Save your current changes to the Repository.

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

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

5 From the left pane of the Deployment Editor, drag **BP_LFOut1** (Business Process) to **IntegrationSvr1** in the **Localhost3** window.

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

4.7.8. Running the Project

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.*

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\LogicalHost2** directory. You must specify the **LogicalHost2** directory for it to be created.

3 Navigate to C:\ican50\LogicalHost3\bootstrap\config directory and open the logical-host.properties file using Notepad™.

4 Enter the following information in the appropriate fields:

- Logical Host rood directory: **ican50\LogicalHost2**
- Repository URL: **http://localhost:*port number/repository name***
- Repository user name and password: ***Your user name and password***
- Logical Host Environment name: ENV_Batch_BP_LFOut
- Logical Host name: **LogicalHost2**

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

5 Run the **bootstrap.bat** file in the ican50\LogicalHost2\bootstrap\bin directory.

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

Chapter 5

Implementing a Batch eWay Project

This chapter provides an introduction to the Batch eWay components used in a Java 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 two sample Batch eWay Projects created using the same procedures as the sample end-to-end Project provided in the *eGate Tutorial*.

For a complete explanation of ICAN terminology and concepts, see the *eGate Integrator User's Guide*.

Chapter Topics

5.1 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 14.

Batch eWay OTDs

Object Type Definitions (OTDs) map input and output message segments at the field level. The Batch eWay has the following three OTDs:

- **BatchRecord OTD**
- **BatchFTP OTD**
- **BatchLocalFile OTD**

For information on the Batch eWay OTDs, see **Overview of Batch OTDs** on page 118.

5.2 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 uploaded with the eWay's documentation .sar file and downloaded from the Enterprise Manager's Documentation tab. Extract the samples from the Enterprise Manager to a local file.
2. From the Enterprise Designer's Project Explorer pane, right-click the Repository and click **Import Project** form the selection menu. The **Select File to Import** dialog box appears.
3. Browse to the directory that contains the sample project zip file. Select the sample file (for this sample, **BatchSample1.zip**) and click **Open**.
4. From the File Destination dialog box, select **Import to a new Project**, enter the name of the Project, and click **OK**.
5. When the import has successfully completed, from the Project Explorer, select the Repository and click **Refresh All from Repository**.
6. Before an imported sample Project can be run you must do the following:
   - Create an **Environment**
   - Configure the eWays for your specific system
   - Create a **Deployment Profile**

5.3 The Batch eWay Sample Projects

This chapter provides step by step directions for two sample Batch eWay Projects.

- **The Batch_FTPIn_LFOut_Sample Project** on page 93
- **The Batch_RecParseStream_Sample Project** on page 108

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.

5.4 The Batch_FTPIn_LFOut_Sample Project

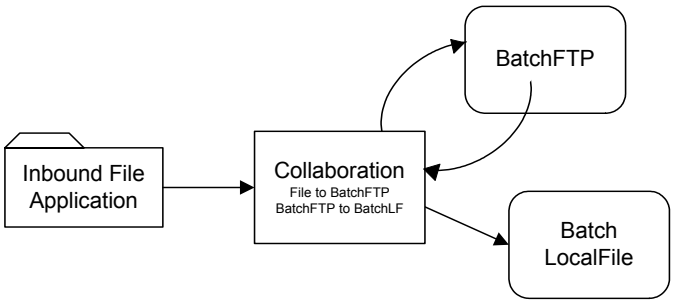
The Batch_FTPIn_LFOut_Sample Project demonstrates how the Batch eWay uses a File eWay to locate and select a file from a local directory. The BatchFTP eWay brings data in and the Collaboration combines the data and writes it to a local file.

The Java Collaboration invokes the Batch FTP eWay to get the payload file, populate the payload with data, and then puts the data into a local file. To accomplish this, the Java Collaboration has three Business Rules that convert incompatible data types by doing the following:

- unmarshaling data from String
- Converting String data to byte array
- Populating a payload with the converted data and writing it to a local file

Figure 32 shows the data flow of the Batch eWay sample Project.

Figure 32 Batch eWay Sample Project

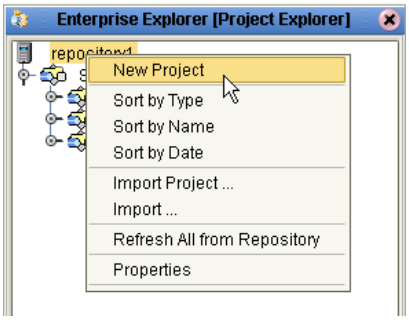


5.4.1. Create a Project

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 33). A new Project (Project1) appears on the Project Explorer tree.

Figure 33 Enterprise Explorer - New Project



3. Click twice (not double-click) on **Project1** and rename the Project (for this sample, **Batch_FTPInLFOut_Sample**).

5.4.2 Creating a Connectivity Map

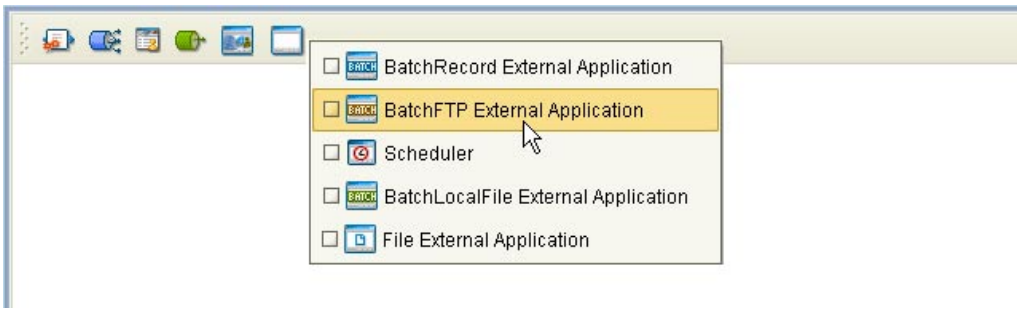
A Connectivity Map provides a canvas for assembling and configuring a Project's components.

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

When creating a Connectivity Map, the 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 34).

Figure 34 Connectivity Map - External Applications



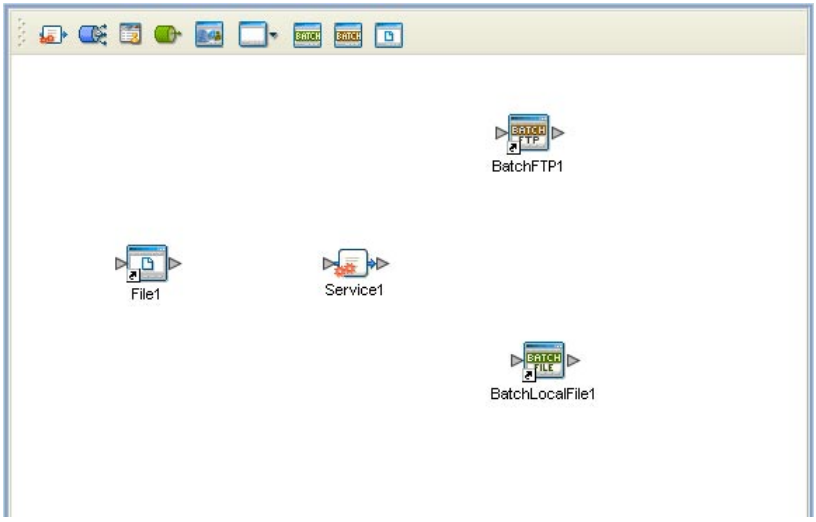
1. Click the **External Application** icon on the Connectivity Map toolbar.
2. Select the External Applications for your Project. For this sample, select **File External Application**, **BatchFTP External Application**, and **BatchLocalFile External Application**. This adds icons representing the **File** and **BatchFTP** and **BatchLocalFile** External Application icons to the Connectivity Map toolbar.

5.4.3 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 35** on page 96:
   - File External Application
   - Service (container for the Java Collaboration)
   - BatchLocalFile External Application
   - BatchFTP External Application

Figure 35 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**

5.4.4 Creating Java Collaboration Definitions

The next step in the sample Project is to create the **jcd_FTPInLFOut** Java Collaboration Definition. A Collaboration Definition contains Business Rules that define the processing and transport of data between eGate components.

The Java Collaboration Definition Wizard is used to create the Java Collaboration Definitions. Once a Collaboration is created, the Java Collaboration Editor is used to create the Business Rules of the Collaboration.

Create the jcd_FTPInLFOut Java Collaboration

The **jcd_FTPInLFOut** Java 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** > **Java Collaboration Definition** from the shortcut menu. The **Java Collaboration Definition Wizard** appears.

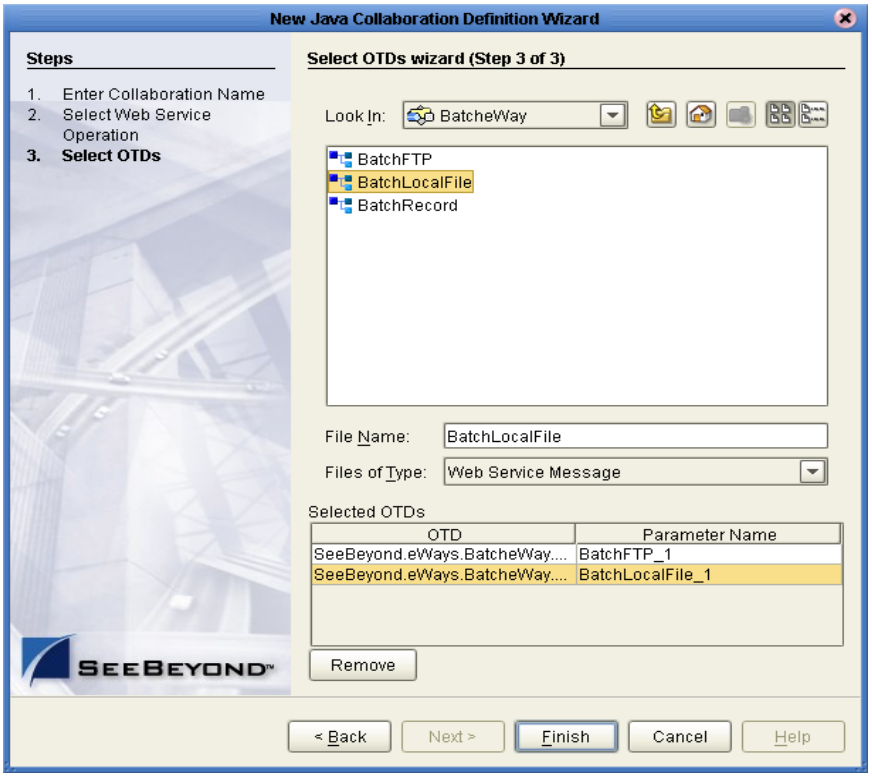
2 Enter a Collaboration Definition name (for this sample **jcd_FTPInLFOut**) and click **Next**.

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.FtpETD** OTD is added to the Selected OTDs field.

5 From the Select OTDs selection window, double-click **BatchLocalFile** (SeeBeyond > eWays > BatcheWay > BatchLocalFile). The **BatchLocalFile.LocalFileETD** OTD is added to the Selected OTDs field (see **Figure 36** on page 97).

Figure 36 Java Collaboration Definition Wizard - jcd_FTPInLFOut - Select OTDs



6 Click **Finish**. The new **jcd_FTPInLFOut** Collaboration appears in the Project Explorer tree.

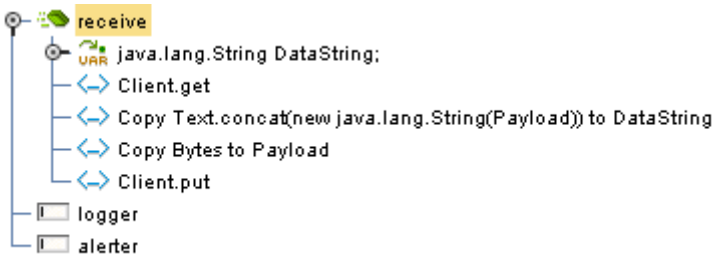
5.4.5 Using the Java Collaboration Editor

The **Batch_FTPInLFOut_Sample** Project uses the **jcd_FTPInLFOut** Collaboration created in the previous section. To complete the Collaboration, use the Java Collaboration Editor to create the Business Rules.

Create the jcd_FTPInLFOut Collaboration Business Rules

Be careful to open all nodes specified in the directions to connect to the correct item. The **jcd_FTPInLFOut** Collaboration contains the Business Rule displayed in **Figure 37** on page 98.

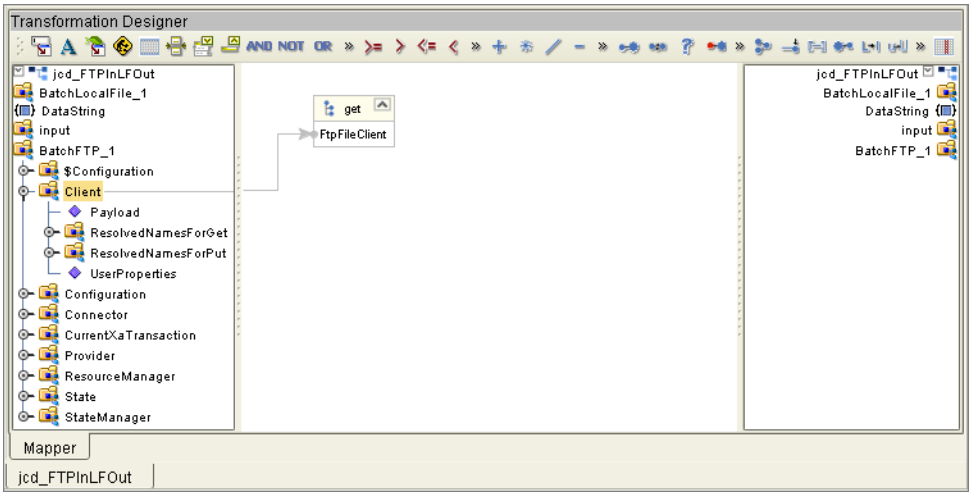
Figure 37 jcd_Passthru Business Rules



To create the **jcd_Passthru** Collaboration Business Rules, do the following:

1. From the Project Explorer tree, double-click **jcd_Passthru**. The Java Collaboration Editor opens to the jcd_Passthru Collaboration.
2. To create the **java.lang.String DataString** (variable) rule do the following:
   - A Right-click **DataString** in the left pane of the Transformation Designer, and select **Create new Variable of this type** from the shortcut menu. The Create a Variable dialog box appears.
   - B Enter **DataString** as the Variable Name and click **OK.**
3. To create the **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 Transformation Designer, and click **Select a 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 Transformation Designer canvas (see Figure 38).

Figure 38 Java Collaboration Editor - Transformation Designer



4. To create the **Copy Text.concat(new java.lang.String(Payload)) to DataString** rule do the following:

D From the Transformation Designer toolbar, drag the **concat** icon to the Transformation Designer canvas. The **concat** method box appears.

E Map **Text** under **input** in the left pane of the Transformation Designer, to the **String** input node of the concat method box.

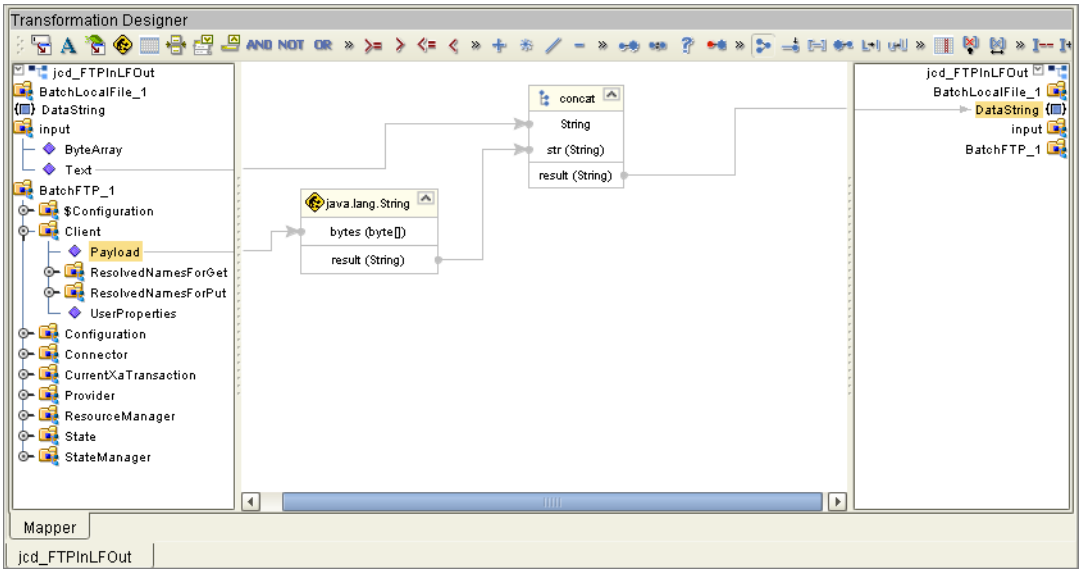
F Map the **result (String)** output node of the **concat** method box to **DataString** in the right pane of the Transformation Designer.

G From the Transformation Designer toolbar, click the **Call New Constructor** icon. The **Call New Constructor** dialog box appears. Select **String** in the **All Classes** window, select **java.lang.String(byte[] bytes)** in the **Constructors** window, and click **OK**. The **java.lang.String** method box appears in the Transformation Designer canvas.

H Map **Payload** under **BatchFTP_1 > Client** in the left pane of the Transformation Designer, to the **bytes (byte[])** input node of the **java.lang.String** method box.

I Map the **result (String)** output node of the **java.lang.String** method box to **the str (String)** input node of the **concat** method box (see .

Figure 39 Java Collaboration Editor - Transformation Designer



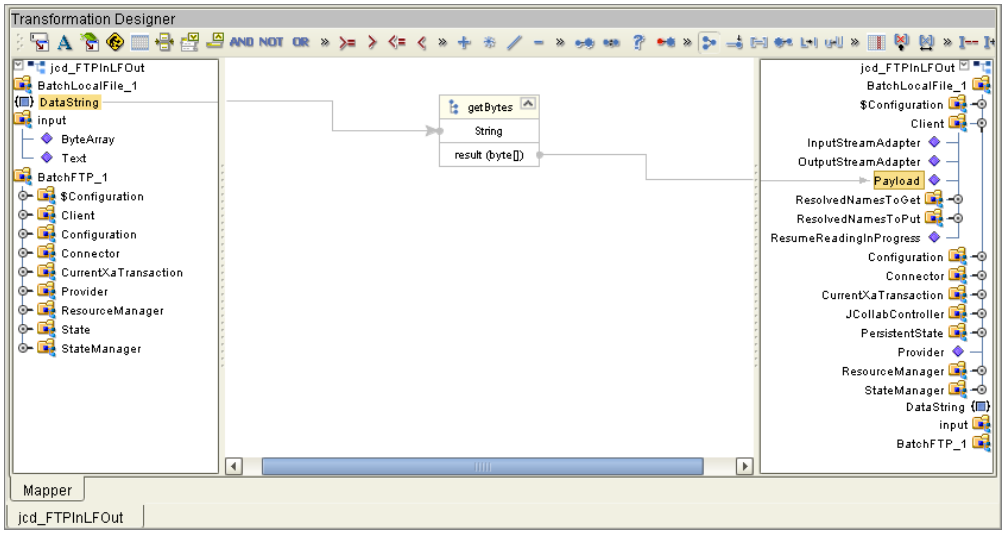
5 To create the **Copy Bytes to Payload** rule do the following:

J Click **rule** on the Business Rules toolbar to add a new rule in the Business Rules pane.

K Right-click **DataString** in the left pane of the Transformation Designer and click **Select a method to call** from the shortcut menu. Select **getBytes()** from the method selection window. The getBytes method box appears.

L Map the **result (byte[])** output node of the **getBytes** method box, to **Payload** under **BatchLocalFile_1 > Client** in the right pane of the Transformation Designer. The Editor now appears as displayed in **Figure 40** on page 100.

Figure 40 Java Collaboration Editor - Transformation Designer



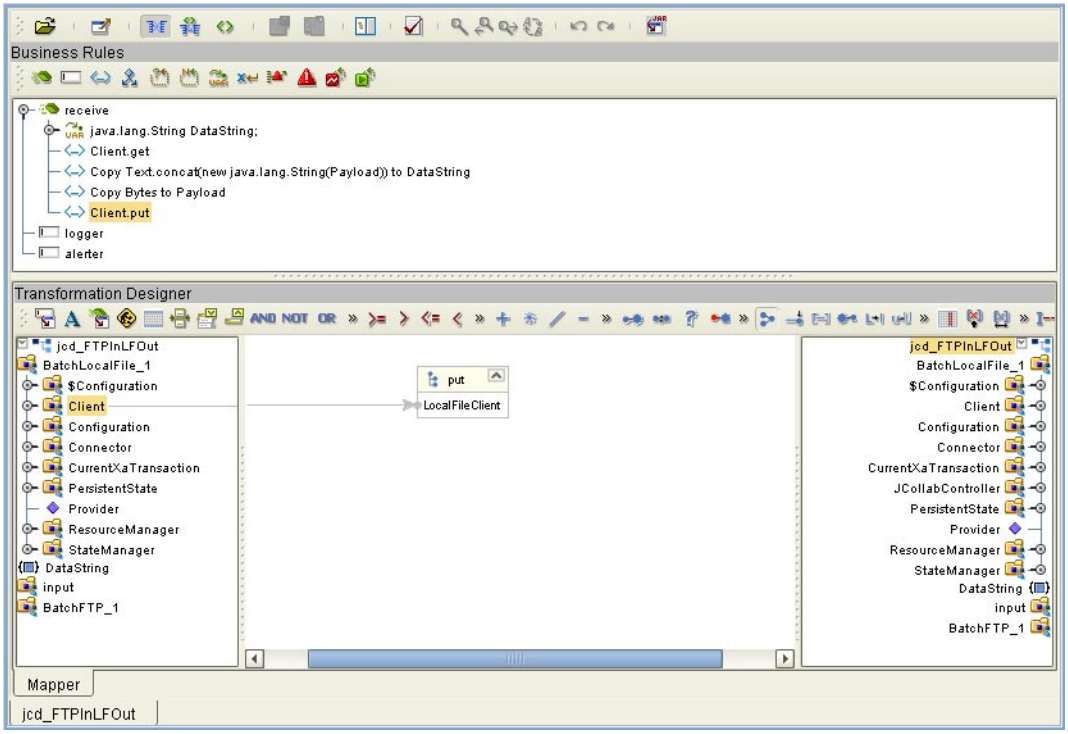
6 To create the **Client.put** rule do the following:

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 Transformation Designer, and click **Select a method to call** from the shortcut menu. The **method selection** window appears.

7 Select **put()** from the method selection window. The get method box appears in the Transformation Designer canvas (see Figure 38).

Figure 41 Java Collaboration Editor - Creating Business Rules



8 From the Editor's toolbar, click **Commit Changes**.

9 Save your current changes to the Repository.

Note: *See the eGate Integrator User's Guide for more information on editing Collaborations.*

5.4.6 Creating Collaboration Bindings

After the Collaboration is complete, associate the components and create Collaboration Bindings by connecting the components in the Connectivity Map.

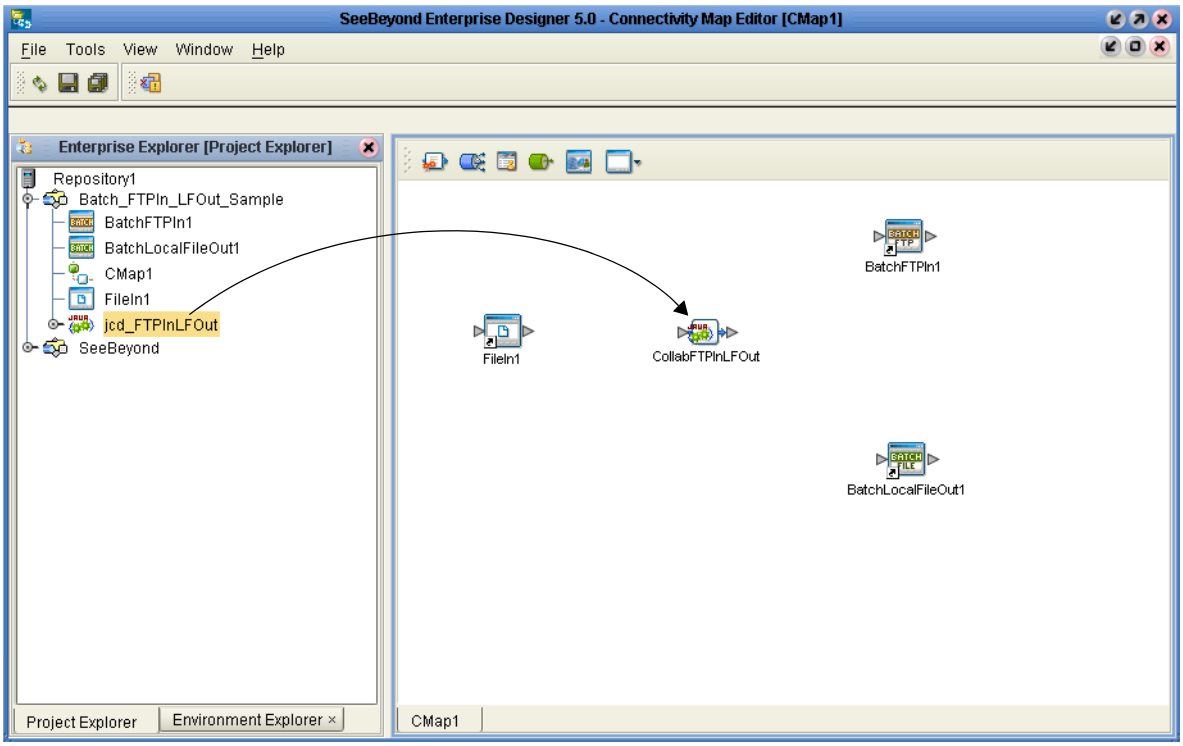
Create the Collaboration Bindings

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 42).

Figure 42 Connectivity Map - Binding the Project components



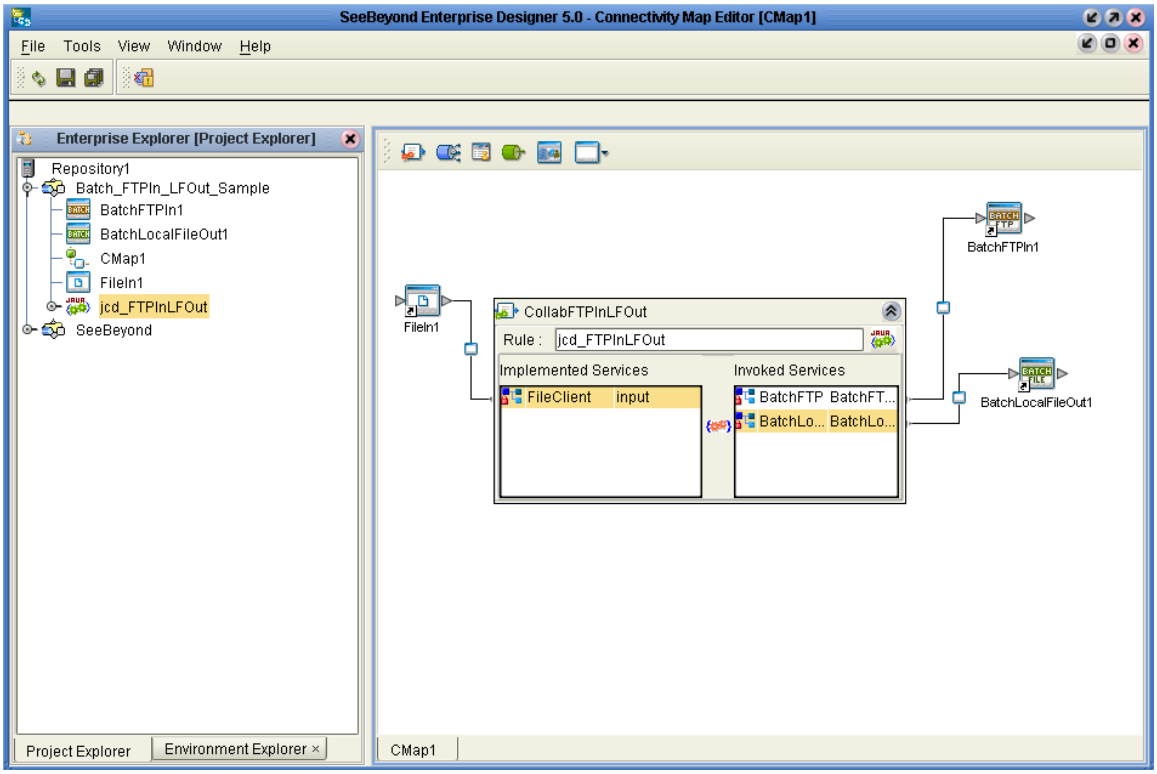
4 From the connectivity Map, double-click **CollabFTPInLFOut**. The **CollabFTPInLFOut** binding dialog box opens with **jcd_FTPInLFOut** as the rule.

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 43**)**.

Figure 43 Connectivity Map - Binding the Project components



8 Minimize the **CollabFTPInLFOut** binding dialog box and save your changes to the repository.

5.4.7. 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.

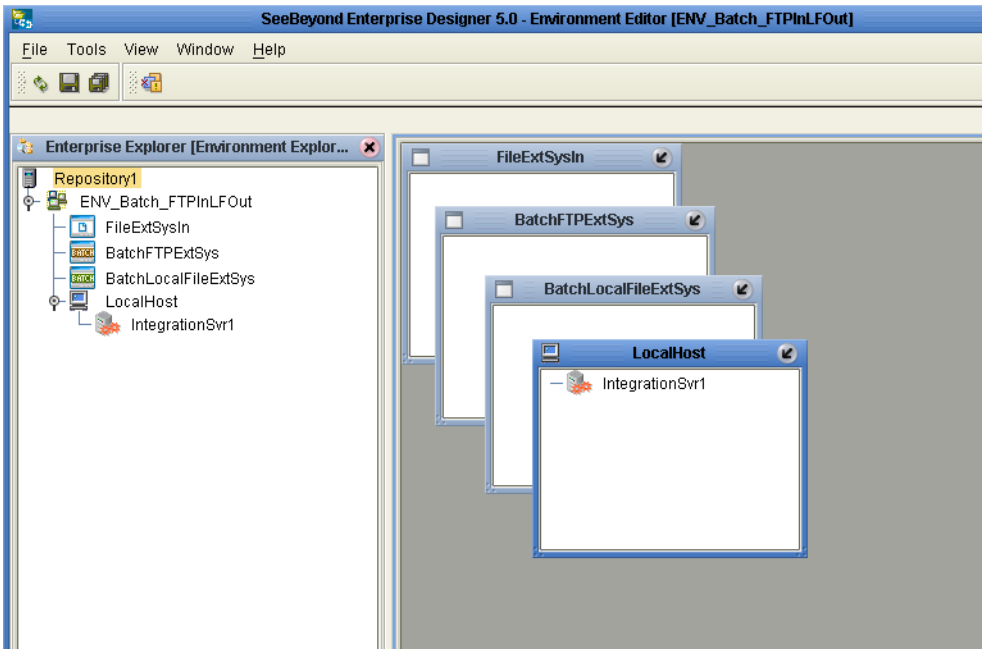
6 From the Project Explorer tree, right-click **ENV_Batch_FTPInLFOut** and select **New BatchLocalFile External System**. Name the External System **BatchLocalFileExtSys**. Click **OK**. The **BatchLocalFileExtSys** window is added to the Environment Editor.

7 From the Project Explorer tree, right-click **ENV_Batch_FTPInLFOut** and select **New Logical Host**. Enter **LocalHost** in the **Logical Host Name** field. Select **STC Message Server** as the System JMS Type. The **LocalHost** window is added to the Environment Editor.

8 From the Environment Explorer tree, right-click **LocalHost** 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 44.

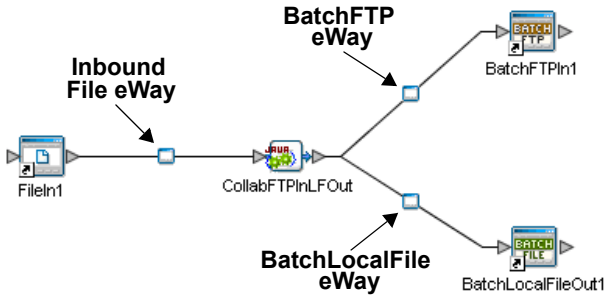
Figure 44 Environment Editor



5.4.8. 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 45** on page 104).

Figure 45 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 5, and click **OK**. The properties are saved for the eWay.

Table 5 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 6, and click **OK**. The properties are saved for the eWay.

Table 6 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 14 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 7** on page 105, and click **OK**.

Table 7 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 8, and click **OK**.

Table 8 BatchFTP eWay Environment Explorer Properties

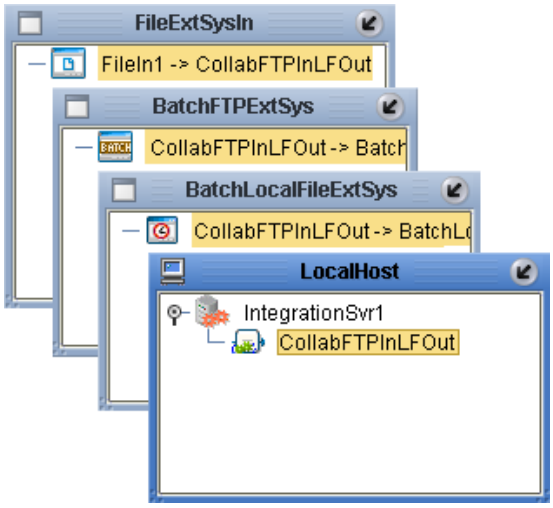
BatchFTP eWay Environment-Configuration Parameters	
FTP Set as directed, otherwise use the default settings.	
Host Name	*The name of the external system to which the eWay connects*
Password	*Password required to log into the external system*
Server Port	*Port number to use to connect to the FTP server*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

5.4.9 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. From the left pane of the Deployment Editor, drag the **FileIn1 -> CollabFTPInLFOut** (External Application) to the **FileExtSysOut** window.
4. From the left pane of the Deployment Editor, drag the **CollabFTPInLFOut -> Batch_FTPIn** (External Application) to the **BatchFTPExtSys** window.
5. From the left pane of the Deployment Editor, drag the **CollabFTPInLFOut -> Batch_LocalFileOut** (External Application) to the **BatchLocalFileExtSys** window.
6. From the left pane of the Deployment Editor, drag **CollabFTPInLFOut** (Java Collaboration) to **IntegrationSvr1** in the **LocalHost** window (see Figure 46).

Figure 46 Deployment Profile



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

5.4.10. 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, click on **Logical Host - for win32**.
2. Extract the file to the **ican50\LogicalHost3** directory. You must specify the **LogicalHost3** directory for it to be created.

3 Navigate to **C:\ican50\LogicalHost3\bootstrap\config** directory and open the **logical-host.properties** file using Notepad™.

4 Enter the following information in the appropriate fields:

- Logical Host root directory: **ican50\LogicalHost3**
- Repository URL: **http://localhost:*port number/repository name***
- Repository user name and password: ***Your user name and password***
- Logical Host Environment name: **ENV_Batch_FTPInLFOut**
- Logical Host name: **LogicalHost3**

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

5 Run the **bootstrap.bat** file in the ican50\LogicalHost3\bootstrap\bin directory.

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

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*.

5.4.11 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 and higher. To enable Logging, please see the ***eGate Integrator User's Guide***.

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

5.5 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 then publishes the data file to an external directory.

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

To create the Sample Project do the following:

5.5.1. 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**).

5.5.2 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.

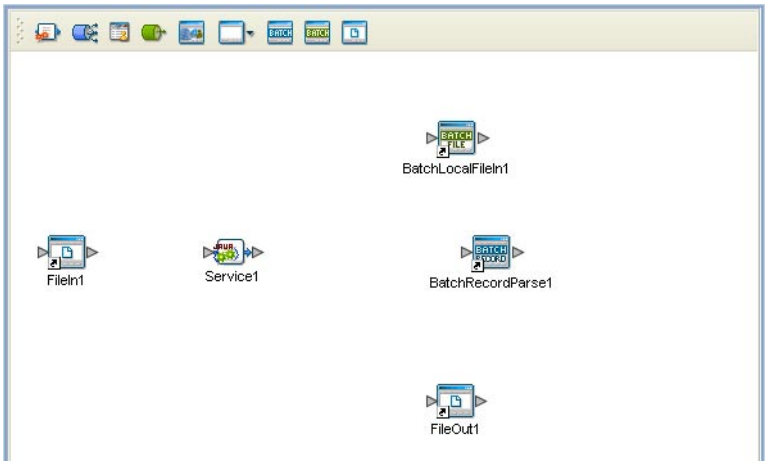
Populate the Connectivity Map

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 directed displayed in Figure 47:

- (2) File External Application - **FileIn1**, **FileOut1**
- Service - **Service1**
- BatchLocalFile External System - **BatchLocalFileIn1**
- BatchRecord External System - **BatchRecordParse1**

Figure 47 CM_RecParseStream_Sample Connectivity Map with Components



2 Save your current changes to the Repository.

5.5.3. Creating a Java Collaboration Definition

The next step in the sample is to create a Java Collaboration using the Java Collaboration Definition Wizard. Once a Collaboration Definition has been created, the Business Rules of the Collaboration are written using the Java Collaboration Editor.

The jcd_RecParse_Stream Java Collaboration

1 From the Project Explorer, right-click the **Batch_RecParseStream_Sample** Project and select **New > Java Collaboration Definition** from the shortcut menu. The **Java Collaboration Definition Wizard** appears.

2 Enter a Collaboration Definition name (for this sample **jcd_RecParse_Stream**) and click **Next**.

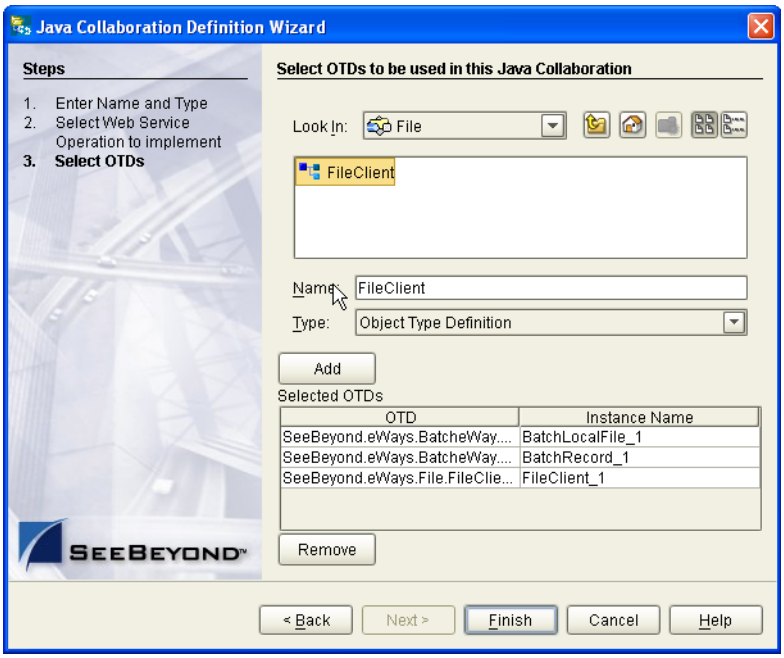
3 For Step 2 of the Wizard, from the Web Services Interfaces selection window, double-click **SeeBeyond** > **eWays** > **File** > **FileClient** > **receive**. The File Name field now displays **receive.** Click **Next**.

4 For Step 3 of the Wizard, from the Select OTDs selection window, double-click **BatchLocalFile** (SeeBeyond > eWays > BatcheWay > BatchLocalFile). The **BatchLocalFile.LocalFileETD** OTD is added to the Selected OTDs field.

5 From the Select OTDs selection window, double-click **SeeBeyond** > **eWays** > **BatcheWay > BatchRecord**. The **BatchRecordETD** 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 48 Java Collaboration Definition Wizard - Select OTDs



7 Click **Finish**. The Java Collaboration Editor opens to the new Collaboration in the right pane of the Enterprise Designer.

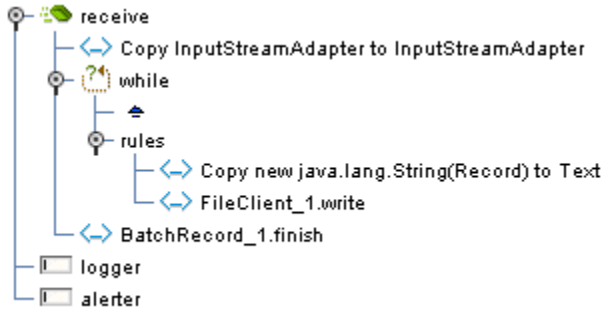
5.5.4. Using the Java Collaboration Editor

The **Batch_FTPInLFOut_Sample** Project uses the **jcd_FTPInLFOut** Collaboration created in the previous section. To complete the Collaboration, use the Java 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 to the correct item. The **jcd_RecParse_Stream** Collaboration contains the Business Rule displayed in Figure 49 on page 111.

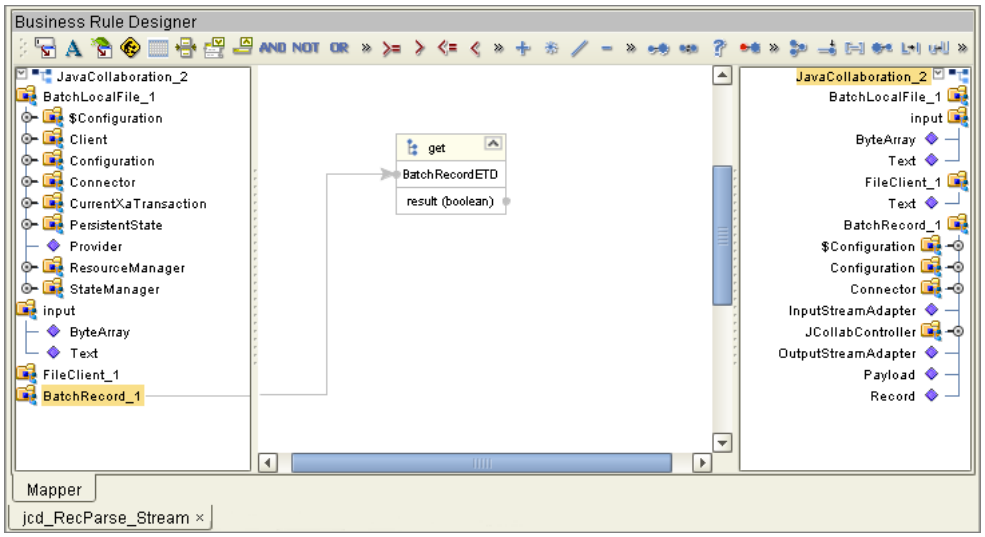
Figure 49 jjcd_RecParse_Stream Business Rules



To create the **jcd_RecParse_Stream** Collaboration Business Rules, do the following:

1. From the Project Explorer tree, double-click **jcd_RecParse_Stream**. The Java Collaboration Editor opens to the **jcd_RecParse_Stream** Collaboration.
2. To create the **Copy InputStreamAdapter to InputStreamAdapter** rule do the following:
   - A. Map **InputStreamAdapter** under **BatchLocalFile_1 > Client** in the left pane of the Transformation Designer, to **InputStreamAdapter** under **BatchRecord_1** in the right pane of the Transformation Designer. This is done by clicking on and dragging InputStreamAdapter to the destination InputStreamAdapter, and releasing the cursor.
3. 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. Select the Condition under the while statement (this appears as an up-arrow under the while statement) in the Business Rules pane. Right-click **BatchRecord_1** the left pane of the Transformation Designer, and click **Select a 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 Transformation Designer canvas (see Figure 50).

Figure 50 Java Collaboration Editor - Business Rules



4 To create the first rule under the **while** statement do the following:

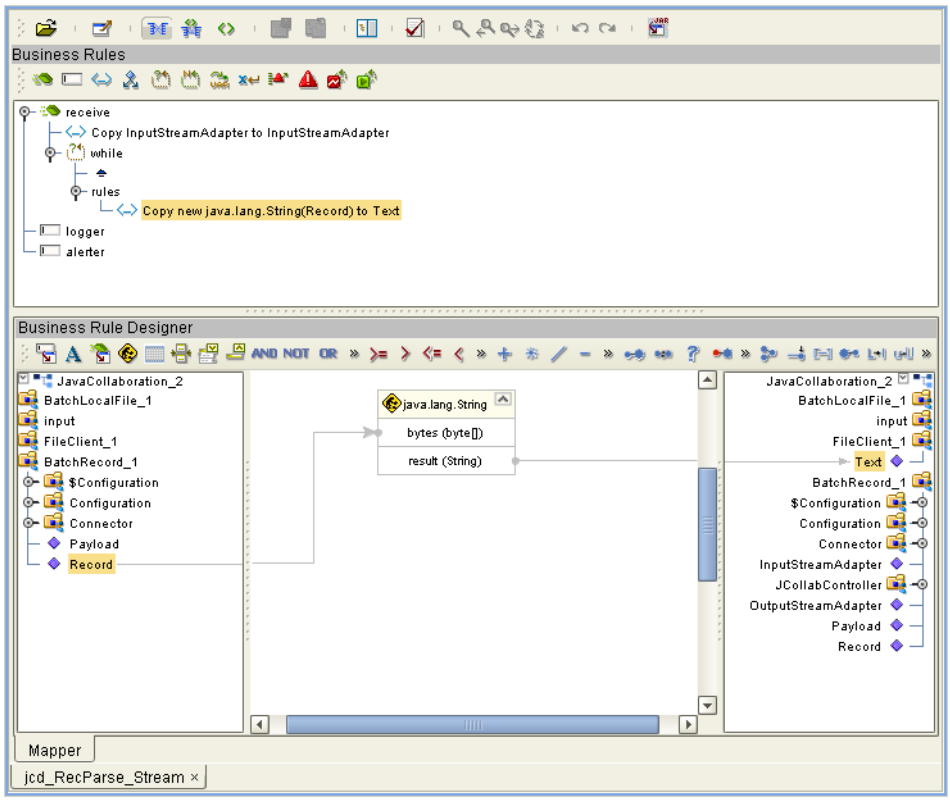
A Select the first rule (**Copy new java.lang.String(Record) to Text**) under the **while** statement in the Business Rules pane.

B From the Transformation Designer toolbar, click on the **Call New Constructor** icon. The **Call New Constructor** dialog box appears. Select **String** from the **All Classes** box and **java.lang.String(byte[] bytes)** from the Constructors box. Click **OK**. The **java.lang.String** method box appears in the Transformation Designer canvas.

C Map **Record**, under **BatchRecord_1** in the left pane of the Transformation Designer, to the **bytes (byte[])** input node of the java.lang.String method box.

D Map the result (String) output node of the java.lang.String method box, to **Text** under **FileClient_1** in the right pane of the Transformation Designer (see Figure 51).

Figure 51 Java Collaboration Editor - Business Rules



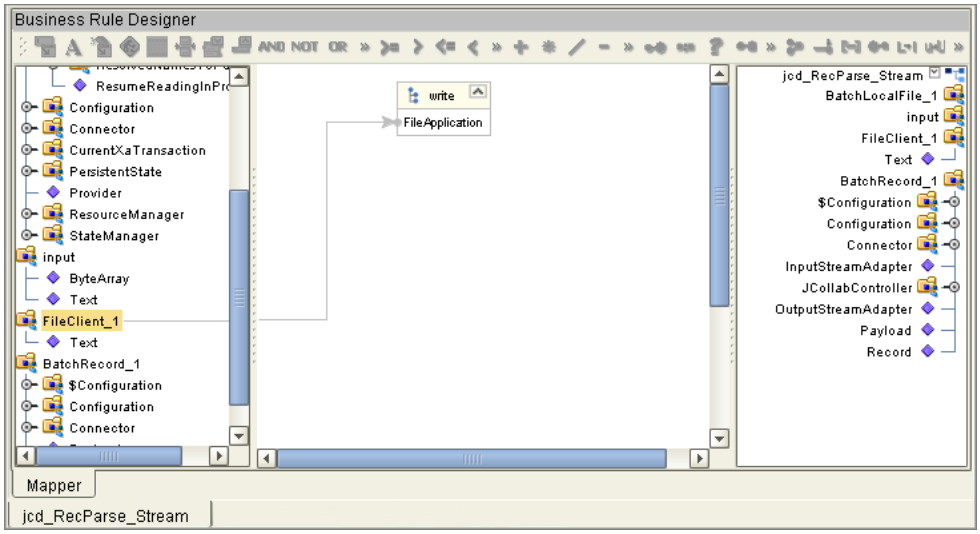
5 To create the second rule under the **while** statement do the following:

A Select the rules node in the Business Rules pane, and from the Business Rules toolbar, click the rule icon. A new rule is added to the Business Rules tree under the while statement.

B Right-click FileClient_1 in the left pane of the Transformation Designer, and click **Select a 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 Transformation Designer canvas (see Figure 52).

Figure 52 Java Collaboration Editor - Business Rules



6 To create the **BatchRecord_1.finish** do the following:

A Click **rule** on the Business Rules toolbar to add a new rule in the Business Rules pane.

B Right-click **BatchRecord_1** in the left pane of the Transformation Designer, and click **Select a 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 in the Transformation Designer canvas

7 Right-click the Repository in the Project Explorer and select **Save changes to Repository** to save your changes.

For more information on how to create Business Rules using the Java Collaboration Editor see the *eGate Integrator User's Guide*.

5.5.5. 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** Java 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.

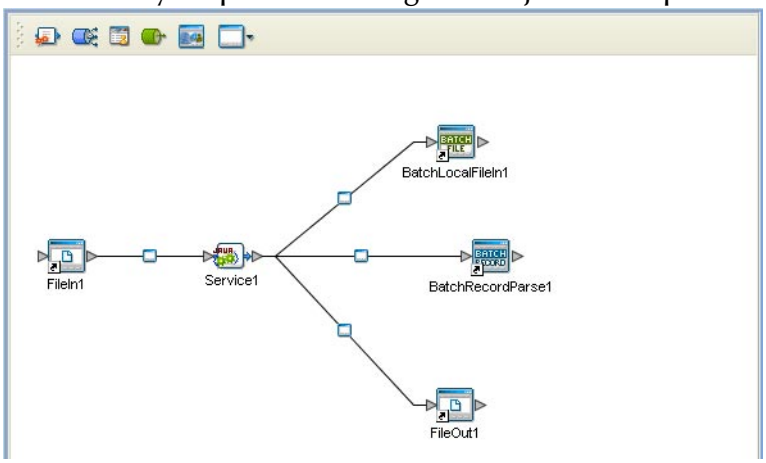
5 From the **Service1** binding box, map the **BatchLocalFile** (under Invoked Services) to the **BatchLocalFileIn1** External Application.

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 53.

Figure 53 Connectivity Map - Connecting the Project's Components



9 Save your current changes to your Repository.

5.5.6. 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**. Enter **Localhost4** in the **Logical Host Name** field. Select **STC Message Server** as the System JMS Type. The **Localhost4** window is added to the Environment Editor.

9 From the Environment Explorer tree, right-click **Localhost4** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under Localhost4.

10 Save your current changes to the Repository.

5.5.7. 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 14.

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.

5.5.8 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 left pane of the Deployment Editor, drag the **FileIn1 -> Service1** (External Application) to the **FileExtSysIn** window.
4. From the left pane of the Deployment Editor, drag **Service1 -> FileOut1** (External Application) to the **FileExtSysOut** window.
5. From the left pane of the Deployment Editor, drag the **Service1 -> BatchLocalFileIn1** (External Application) to the **BatchLocalFileExtSys** window.
6. From the left pane of the Deployment Editor, drag the **Service1 -> BatchRecordParse1** (External Application) to the **BatchRecordExtSys** window.
7. From the left pane of the Deployment Editor, drag **Service1** (Java Collaboration) to **IntegrationSvr1** in the **Localhost2 >** window.
8. Click **Activate**. When activation succeeds, save Your current changes to the Repository.

5.5.9. 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, click on **Logical Host - for win32**.
2. Extract the file to the **ican50\LogicalHost4** directory. You must specify the **LocalHost4** directory for it to be created.
3. Navigate to C:\ican50\LogicalHost4\bootstrap\config directory and open the logical-host.properties file using Notepad™.
4. Enter the following information in the appropriate fields:
   - Logical Host rood directory: **ican50\LogicalHost4**
   - Repository URL: **http://localhost:*port number/repository name***
   - Repository user name and password: ***Your user name and password***
   - Logical Host Environment name: **ENV_Batch_RecParseStream**
   - Logical Host name: **LogicalHost4**

   Save your changes to **logical-host.properties** and close the file.
5. Run the **bootstrap.bat** file in the ican50\LogicalHost4\bootstrap\bin directory.
6. Copy the sample input data file to the input directory.

Chapter 6

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.

Chapter Topics

6.1 Overview of 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** – This 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.

6.1.1 Types of Batch eWay OTDs

Table 9 shows the specialized OTDs available with the eWay.

Table 9 Batch eWay OTDs

OTD Name	Description
BatchFTP	Provides FTP access to remote systems.
BatchLocalFile	Provides easy access to local file systems.
BatchRecord	Allows the eWay to parse or create (or both) payloads of records in specified formats.

This chapter explains each of these OTDs and how to use them with the eWay.

6.1.2 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.

6.2 OTD for FTP Operations

The Batch eWay includes an OTD that allows you to perform FTP data-transfer functions, the BatchFTP OTD.

The combination of this OTD, working with a specific eWay with its own set of configurable parameters, defines the characteristics of the external interface. Using the BatchFTP OTD and one or more eWays, you can create the Collaboration Rules to make the Batch eWay behave in specific ways.

Note: *Create Collaboration Rules using the eGate Enterprise Designer's Collaboration Rules Editor. For more information on this feature, see the* ***eGate Integrator User's Guide****.*

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. 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 all cases. Failure to do so can cause loss of data.*

6.2.1 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.

6.2.2 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 **"OTD for FTP Operations" on page 119** for details on these parameters and settings.

Note: *This OTD has configuration parameters that can be regular expressions. See* **"Using Regular Expressions" on page 137** *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 7** 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 144** 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 126** for details).

Note: *See* **"Resolving Names" on page 141** *for more information on these nodes; see* **"Using Regular Expressions" on page 137** *for more information on regular expressions.*

  - **get()**, **put()**, **reset()**, **connect()**, **disconnect()**, and **isConnected()**: See **"Essential BatchFTP OTD Methods" on page 123**.
- **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.

6.2.3 Using the BatchFTP OTD

The BatchFTP OTD nodes allow you to configure specific eWay configuration parameters for the Java 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 120**.

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 144** *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 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, if 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 that 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.*

For information on the parameter settings see:

BatchFTP OTD

"Sequence Numbering" on page 24

BatchLocalFile OTD

This feature is also available with the BatchLocalFile OTD. For more information on these configuration parameters, see **"Sequence Numbering" on page 24**.

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 126** for details.

6.3 OTD for Local File

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.

6.3.1 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 Configuration Parameters" on page 39** 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.

6.3.2 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 Configuration Parameters" on page 39** for details on these parameters and settings.

Note: *This OTD has configuration parameters that can be regular expressions. See* **"Using Regular Expressions" on page 137** *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 137** *and* **"Using Special Characters" on page 140** *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 144** 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 126** 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 129**.
- **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.*

6.3.3 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.*

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 129**.

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 NT 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.

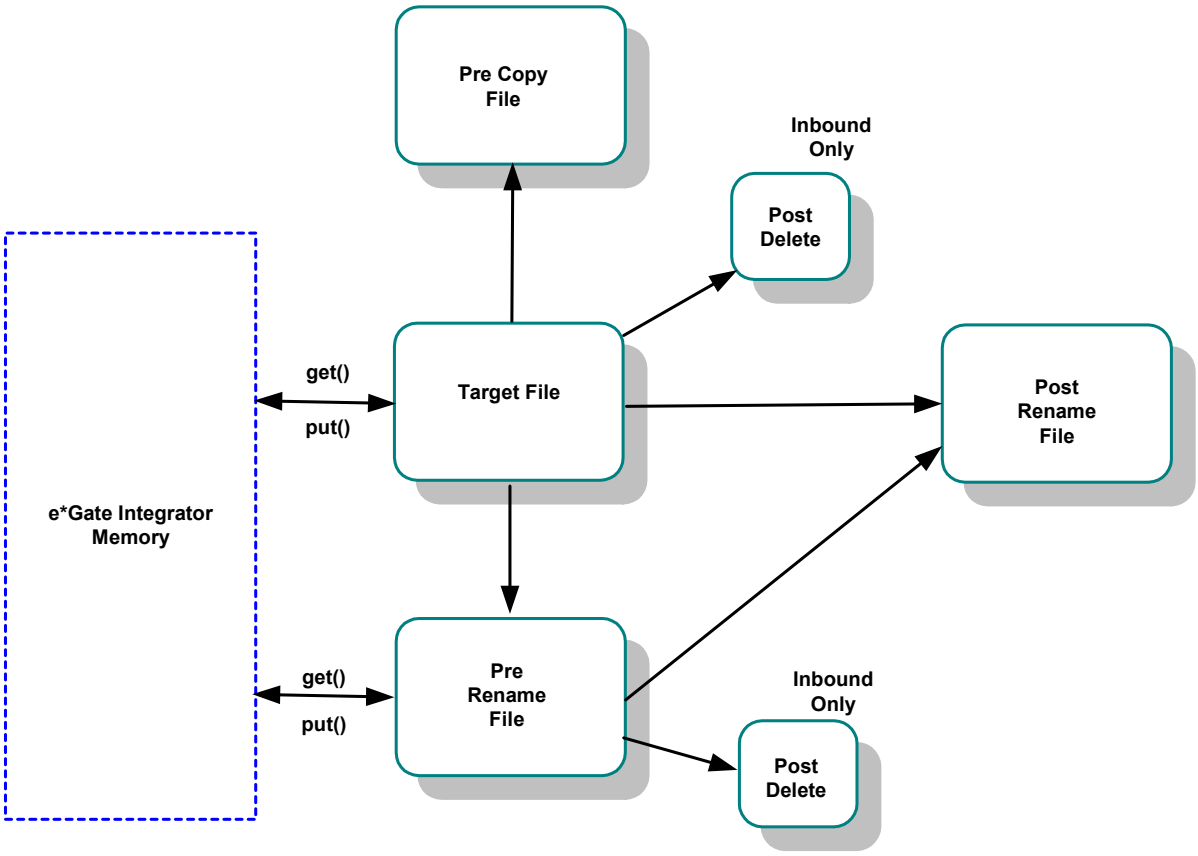
Your post-transfer options are:

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

Figure 54 Pre- and Post-transfer Processes



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

BatchFTP OTD

- **"Pre Transfer" on page 18**
- **"Post Transfer" on page 25**

BatchLocalFile OTD

- **"Pre Transfer" on page 18**
- **"Post Transfer" on page 25**

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 and require some additional explanation. They are:

- **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, if 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.

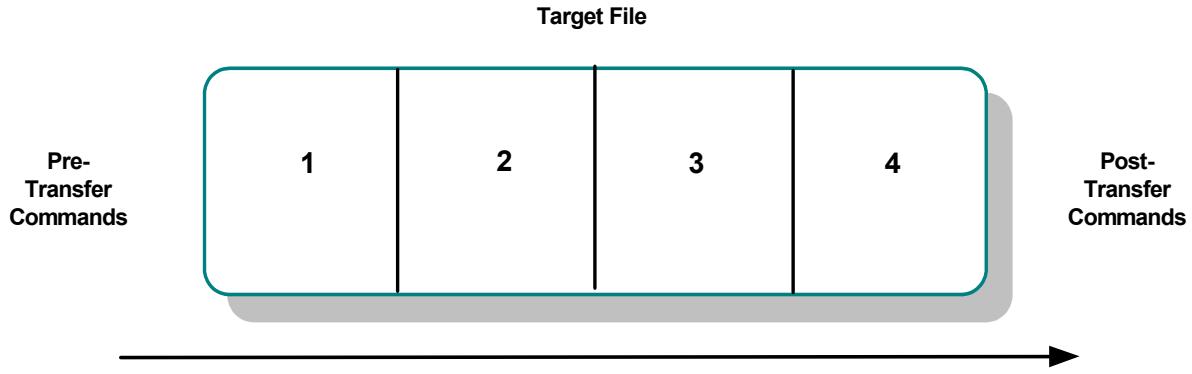
The feature has no special operational requirements besides setting the eWay configuration option. The eWay configuration has an option to enable or disable this feature. This option is also accessible at run time.

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 55 shows a diagram of how the Resume Reading feature operates along with pre- and post-file-transfer commands. This Collaboration Rule has four Business Rules, each of which reads a part of the file.

Figure 55 Resume Reading Operation



Because the file in **Figure 55 on page 130** is read in four parts, there are three instances of the break condition. The lines at the end of **Parts 1, 2, and 3** represent these conditions.

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

- 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.
- 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.
- 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 144** for details on how to use the OTD's data streaming feature.

Sequence Numbering

This feature in this OTD operates in the same way as sequence numbering for the BatchFTP OTD. See **"Sequence Numbering" on page 123** for details.

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 122** for details.

6.3.4 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 144** and **"Resume Reading Feature" on page 129**).

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.

6.3.5 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*

6.4 OTD for Batch Record Processing

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.*

6.4.1 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.

6.4.2 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 Configuration Parameters" on page 46** 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 144**.

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.*

6.4.3 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 7 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 7 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 132**
- **"Pre/post File Transfer Commands" on page 126**

6.5 Using Regular Expressions

This section explains some basic guidelines on how to use regular expressions with the Batch eWay.

6.5.1 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).*

Both the BatchLocalFile and BatchFTP OTD's configurations allow you to use regular expressions, for example, if you want to access all files with the same extension. For more information on how to use 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 and 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 140**.

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.

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**. You can use these types of regular expression patterns 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.*

6.5.2 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 \\\\.

6.6 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 137***.*

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.

6.6.1 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 10 on page 142** 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 126**), 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.

6.6.2 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 120** *for more information on BatchFTP OTD nodes.*

6.6.3 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 10 for a complete list.

Table 10 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)
%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 11 shows some examples using the U.S. locale.

Table 11 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 7

Additional Features

This chapter explains the following additional features of the Batch eWay:

- **Data streaming**: The eWay's ability to stream data between Object Type Definition (OTD) components.
- **Secure FTP:** The eWay supports SOCKS and SSH tunneling to provide secure FTP data transmission.

Chapter Topics

7.1 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.

7.1.1 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.

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.*

7.1.2 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.

7.1.3 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, allowing 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 6**.

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.

7.1.4 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;
}
```

7.2 Secure FTP and the eWay

This section explains the secure FTP features available with the Batch eWay. The Batch eWay supports the following types of secure FTP:

- SOCKS versions 4 and 5
- Secure shell (SSH) tunneling

The rest of this chapter discusses these features and how the Batch eWay uses them.

7.2.1 SOCKS Support

SOCKS is an IETF (Internet Engineering Task Force)-approved standard (RFC 1928) generic, proxy protocol for TCP/IP-based networking applications. The SOCKS protocol provides a flexible framework for developing secure communications 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, version 4 and version 5 (called SOCKSv4 and SOCKSv5). The SOCKSv4 protocol performs the following functions:

- Makes connection requests
- Sets up proxy circuits
- Relays application data

In addition to the functions listed above, the SOCKSv5 protocol also provides authentication.

The Batch eWay supports both SOCKSv4 and SOCKSv5. To enable SOCKS support, the following must be specified in the eWay Connection's configuration file:

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

Details of these configuration parameters are provided under **"SOCKS Configuration Parameters" on page 149**.

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 includes two components, the SOCKS server and SOCKS client. The SOCKS server is implemented at the application layer, while the SOCKS client is implemented between the application and transport layers.

Primarily, the protocol allows hosts on one side of a SOCKS server to gain access to systems on the other side of a SOCKS server, without requiring direct IP-accessibility.

SOCKS Proxy Server

When an application client needs to connect to an application server, the client connects to a SOCKS proxy server. The proxy server connects to the application server on behalf of the client and relays data between the client and the application server. For the application server, the proxy server is the client.

SOCKS and the Batch eWay

The eWay can use SOCKS to provide for secure FTP during data transmission.

Negotiation Methods

The eWay supports the following negotiation methods:

- No-authentication
- User/password

SOCKS Configuration Parameters

This section lists the SOCKS parameters you must set to configure the eWay Connection. For more information, see **"SOCKS Configuration Parameters" on page 149**.

Socks Enabled

Allows you to specify 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.

Socks Host Name

Allows you to enter the SOCKS host name. When you are communicating with a SOCKS server, enter the SOCKS server name in this parameter.

Socks Server Port

Allows you to enter the port number to use on the SOCKS server, when connecting to it.

Socks Version

Allows you to specify the SOCKS server version. For the best performance, it is a good idea to specify a version number instead of **Unknown**.

Socks User Name

Allows you to specify the user name to use along with the password for authentication with a SOCKS5 server. This parameter is used for the user/password negotiation method.

Socks Password

Allows you to specify the password to use along with the user name for authentication with a SOCKS5 server. This parameter is used for the user/password negotiation method.

7.2.2 SSH Tunneling

This section explains the Batch eWay's Secure Shell (SSH) tunneling features.

Note: *SSH tunneling is also called SSH port forwarding.*

SSH Tunneling: Overview

Developed by SSH Communications Security Ltd., SSH is a program that logs on to another computer over a network. Once you have used SSH to log on, you can execute commands in a remote machine and move files from one machine to another. SSH provides strong authentication and secure communications over insecure channels. This feature is a replacement for **rlogin**, **rsh**, **rcp**, and **rdist**.

SSH protects a network from attacks such as Internet protocol (IP) spoofing, IP source routing, and Domain Name System (DNS) spoofing. An attacker who has managed to take over a network can only force SSH to disconnect. The attacker cannot play back the traffic or hijack the connection as long as 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 an additional software application for the SSH tunneling. The eWay supports either of the following applications:

- **OpenSSH**: For details, see:

  http://www.openssh.org

- **Plink.exe**: Plink is a Win32-only command-line interface to the PuTTY back ends and is available from the PuTTY distribution at:

  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

The eWay can use SSH tunneling to provide for secure logon IDs and passwords. The eWay makes use of additional SSH-tunneling software for this functionality.

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 152**). 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 152**.

SSH Tunneling Enabled

Allows you to specify whether the FTP command connection is secured through an SSH tunnel.

- If you choose **No**, all other parameters in this section are ignored.

SSH Channel Established

Allows you to specify whether the eWay needs to launch an SSH subprocess.

- **No** means there is no existing SSH channel for an FTP transfer.
- **Yes** means an SSH channel has been established, so the eWay does not need to spawn an SSH subprocess. If you select **Yes**, the following parameters are required:
  - **SSH Listen Host**
  - **SSH Listen Port**

SSH Command Line

Allows you to enter 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

Allows you to specify 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

Allows you to specify 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

Allows you to specify an SSH user name. This parameter can be required when the setting for the **SSH Channel Established** parameter is **No**.

SSH Password

Allows you to specify 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 8

Using eWay Java Classes and Methods

This chapter provides an overview of the Java classes and methods contained in the Batch eWay. These methods are used to extend the functionality of the eWay.

8.1 Batch eWay Classes and Methods: Overview

Java methods allow you to set information in the Batch eWay Object Type Definitions (OTDs), as well as get information from them.

The nature of this data transfer depends on the configuration parameters you set for the eWay in the eGate Enterprise Designer. For more information, see **Chapter 3**.

Note: *For more information on the Batch eWay OTD structures, their nodes, and attributes, see* **Chapter 6**.

8.1.1 Java Classes

Java methods are organized into related classes.The methods for the Batch eWay are organized into the following Java classes:

- **BatchException**
- **BatchRecordConfiguration**
- **BatchRecordOTD**
- **BatchRecordParser**
- **BatchOTD**
- **FtpFileClient**
- **FtpFileConfiguration**
- **FtpFileException**
- **FtpFileProvider**
- **FtpHeuristics**
- **InputStreamAdapter**
- **LocalFileClient**
- **LocalFileConfiguration**

- **LocalFileOTD**
- **LocalFileException**
- **NestedException**
- **OutputStreamAdapter**
- **StreamingException**

8.1.2 Batch eWay Javadoc

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.

Index

P

R

S

T

U

W