Batch e*Way Intelligent Adapter User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

Java Version

Version 20100824152754.

Contents

Chapter 6

Chapter 9

Chapter 10

Chapter 1

Introduction

This guide explains the Batch e*Way™ Intelligent Adapter. This chapter provides an introduction to the guide and the e*Way.

1.1 Batch e*Way User's Guide

This document gives a general overview of the Batch e*Way and explains how to install, configure, and operate it. The guide also explains the e*Way's usability features, as well as how to implement it in a typical e*Gate Integrator environment.

Note: *This e*Way is enabled by the Java programming language.*

1.2 Intended Reader

The reader of this guide is presumed:

- To be a developer or system administrator with the responsibility for maintaining the e*Gate system
- To have high-level knowledge of Windows operations and administration
- To be thoroughly familiar with Windows-style user interface operations
- To have an understanding of how to use the File Transfer Protocol (FTP)

1.3 General e*Way Operation

The Batch e*Way enables the e*Gate system to use an FTP connection to exchange data with other network hosts, for the purpose of receiving and delivering Events stored in files. The e*Way provides an FTP Event Type Definition (ETD) to perform this operation. In addition, the e*Way provides you with a record-processing ETD and a local file ETD.

Note: *The e*Way supports standard FTP according to RFC-959.*

All e*Ways provide a communication bridge between the e*Gate environment and one or more external systems. As a result, e*Ways are two-sided. They communicate with the e*Gate system on one side and externally on the other. The communication between the e*Way and e*Gate is common to all e*Ways, while communication with external systems is different for each e*Way.

The Batch e*Way is specialized to perform a variety of FTP and FTP-related operations, depending on your specific needs, network environment, record-processing, file transfer, and external system requirements.

1.3.1 ETDs and Collaborations

The combination of specialized ETDs, working with e*Way Connections with configurable parameters, allows you to define the characteristics of your own external interfaces.

Essentially, a Batch e*Way ETD is a mirror image of the e*Way Connection and allows you to redefine desired parameters at the Collaboration level, as opposed to doing this operation at every separate e*Way Connection.

Through these specific, specialized ETDs and configurable e*Way Connections, you can create Collaboration Rules that make the e*Way behave as desired. Use the e*Gate Schema Designer's Collaboration Rules Editor to create Collaboration Rules.

Note: *For more information on the Schema Designer, ETDs, and Collaborations/ Collaboration Rules, as well as their use in e*Gate, see the* ***e*Gate Integrator User's Guide****.* **Chapter 7** *provides an overview of e*Gate and its components as part of explaining the e*Way's schema samples.*

1.3.2 Multi-Mode e*Way

The Batch e*Way Intelligent Adaptor employs the Multi-Mode e*Way (with the **stceway.exe** executable file) to communicate with external systems and within the e*Gate system. The Multi-Mode e*Way is a core e*Gate component. One or more Java Collaborations are utilized to maintain communication between the e*Way and external systems.

For information about configuring the Multi-Mode e*Way, see **Chapter 3**.

1.3.3 e*Way Configuration

An e*Way's configuration settings allow you to specify necessary parameters of operation. In turn, these parameters are adopted into the associated ETD's configuration.

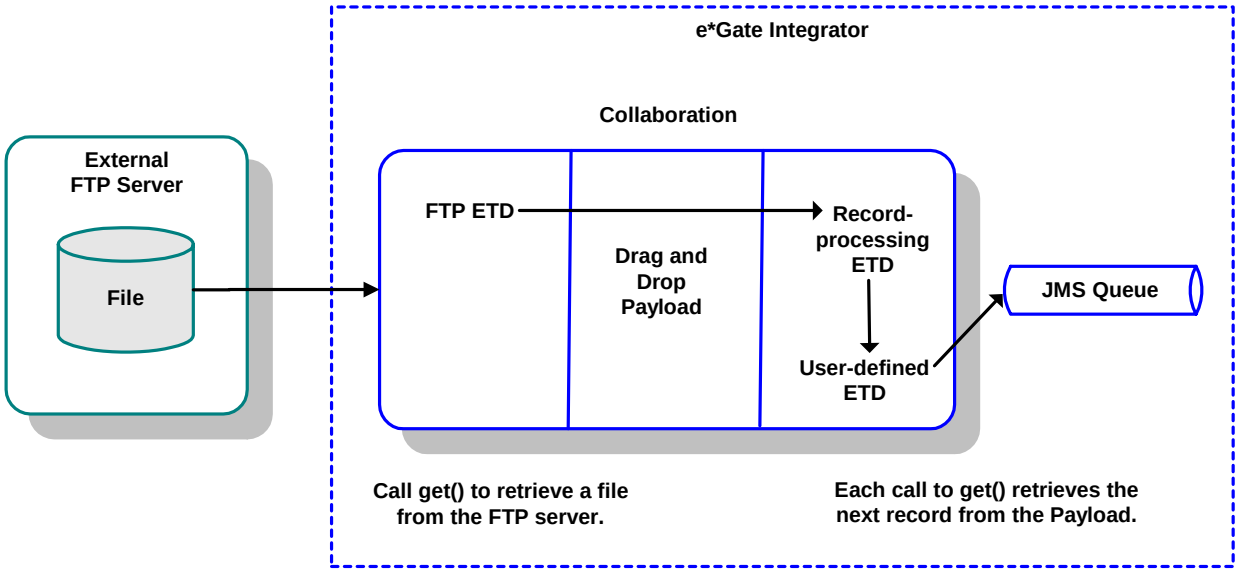
1.3.4 e*Way Overview Diagrams

This section provides general diagrams showing how the e*Way operates in typical use-case situations. See **Chapter 7** for more information on how to implement these types of scenarios, including e*Gate sample schemas.

Case 1: Moving Small Files

Figure 1 shows a diagram of the e*Way set up for use with small files. The e*Way gets files from a remote source and parses them into specified records.

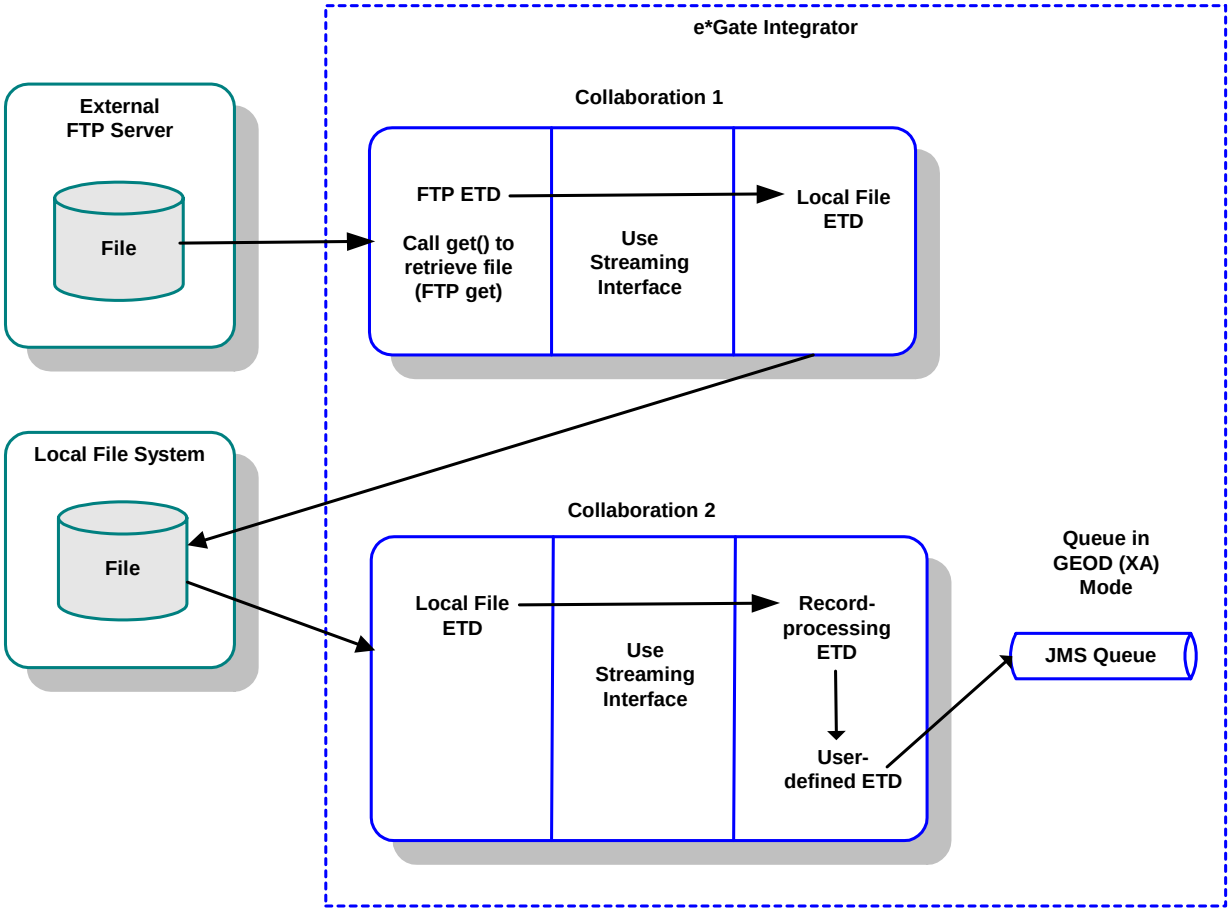
Figure 1 Case 1 Diagram: Moving Small Files



Case 2: Moving Large Files

Figure 2 shows a diagram of the e*Way set up for use with large files. The e*Way gets files from a remote source and parses them into specified records. This setup promotes the exactly-once delivery of Events.

Figure 2 Case 2 Diagram: Moving Large Files



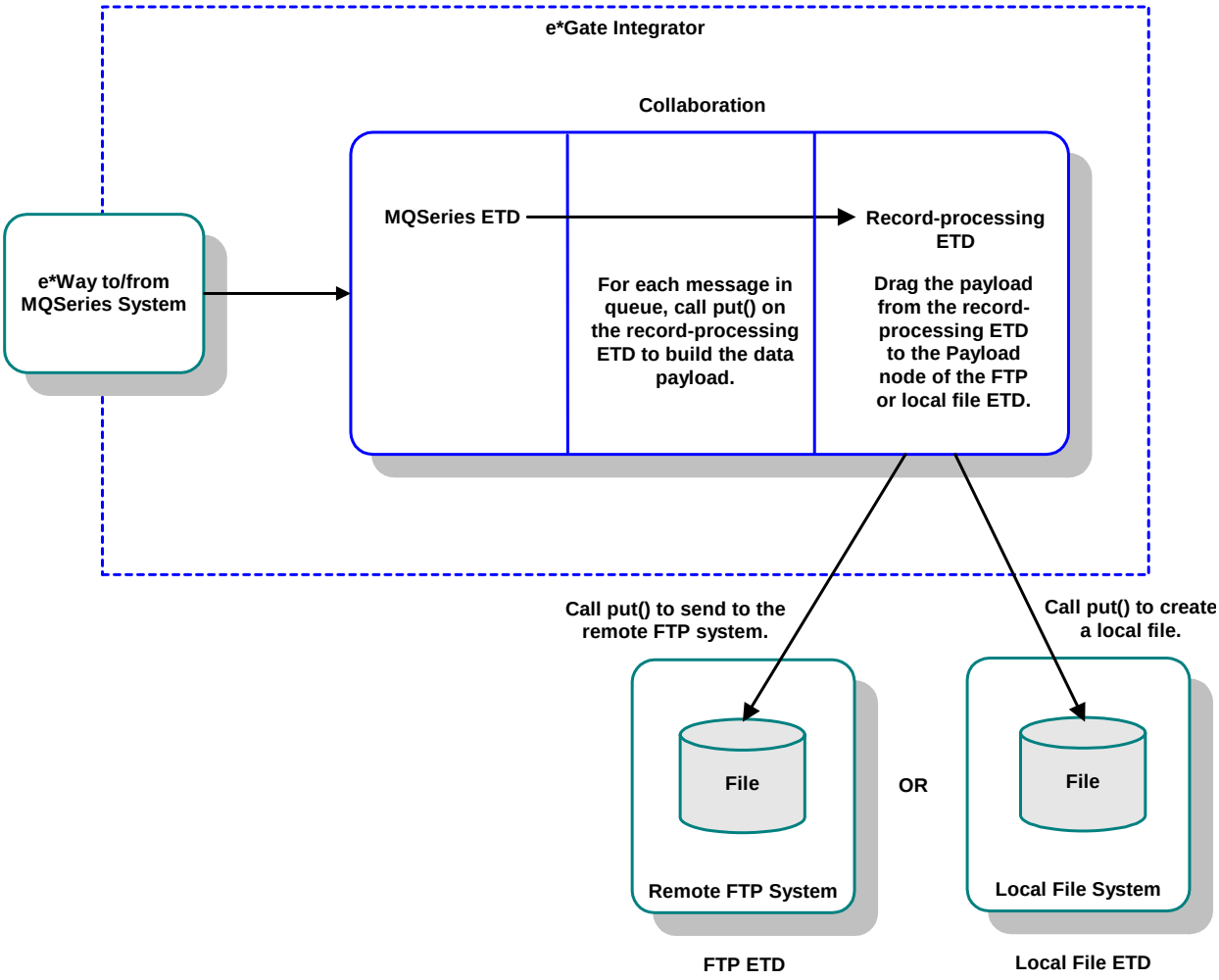
For small files (using a payload-data transfer as shown in case 1) the advantage of using this type of setup is that it provides for optimum performance. If you know ahead of time that the size of the file (payload data) is relatively small, this setup gives you faster operation since the transfer from ETD to ETD is in-memory.

In the case of a large file, however, it is better to use data streaming because this feature does not load the payload (file) into e*Gate's memory.

Case 3: Moving a Data Payload

Figure 3 shows a diagram of the e*Way set up for moving payload data. The e*Way builds a payload and sends it to a local file or remote FTP system.

Figure 3 Case 3 Diagram: Moving a Data Payload



1.3.5 General Features

The Batch e*Way provides the following basic features and implementation applications:

- Receiving and sending individual files
- Receiving files, breaking files up into records, and sending files individually
- Using FTP heuristics to talk to a variety of external system platforms
- Using regular expressions and special characters
- Allowing you to get or put a file synchronously inside a Collaboration
- Retrieving and re-marshaling a file in the same Collaboration

ETDs

Available with the e*Way are extensible ETDs, in which you can override the default configurations. **Chapter 5** details the user-extensible FTP, record-processing, and local file ETDs.

Advanced Features

The following chapters explain how to use the e*Way's advanced features and operations:

- **Chapter 8 "Dynamic Configuration"** Explains how to use the e*Way's Dynamic Configuration feature, including its message-based operations.
- **Chapter 6 "Extending the e*Way"** Discusses ways you can customize the e*Way and extend its functionality.
- **Chapter 9 "Additional Features"** Explains the following features:
  - How to employ payload *data streaming* and minimize memory resource usage.
  - The e*Way's secure FTP features, SOCKS and Secure Shell (SSH) tunneling.
  - How to use the *Guaranteed Exactly Once Delivery* (GEOD) of Events feature, that is, the e*Way's XA-mode functionality.

1.3.6 e*Way Components

The Batch e*Way is made up of the following components:

- Multi-Mode e*Way, a core e*Gate component (uses the **stceway.exe** executable file); see **Chapter 3** for details
- Three custom ETDs, for FTP, record processing, and local file transfer
- Java methods for added functionality; see **Chapter 10** and the Javadoc
- Configuration files, that the e*Gate Schema Designer's e*Way Configuration Editor uses to define configuration parameters; see **Chapter 4** for details
- Additional files necessary for operation, as shown in **Table 1 on page 21** (which provides a complete list of installed files)

1.4 System Requirements

To use the Batch e*Way, you need the following:

- An e*Gate Participating Host.
- A TCP/IP network connection.
- Java SDK (see the **readme.txt** file provided on the installation CD for version information)
- A machine running Windows, on which you can use the e*Gate Schema Designer and its related features, the ETD Editor, e*Way Configuration Editor, and Collaboration Rules Editor.

  The e*Way must be configured and administered using the Schema Designer.
- Additional disk space for e*Way executable, configuration, library, and script files. The disk space is required on both the Participating and the Registry Host. Additional disk space is required to process and queue the data that this e*Way processes. The amount necessary varies based on the type and size of the data being processed and any external applications performing the processing.

For information about supported operating systems, see the **readme.txt** file provided on the installation CD.

1.5 External System Requirements

There are external system requirements if you use Secure Shell (SSH) tunneling. See **"SSH Tunneling" on page 344** for details on these requirements.

Your network must contain an operational SOCKS server if SOCKS will be used for secure data transfer. See **"SOCKS Support" on page 342** for details.

Your network must contain an operational FTP server if FTP operations will be performed.

Chapter 2

Installation

This chapter explains how to install the Batch e*Way Intelligent Adapter.

2.1 Installation on Windows Systems

Pre-installation

- Exit all Windows programs before running the setup program, including any antivirus applications.
- You must have Administrator privileges to install this e*Way.
- Review the **readme.txt** file provided on the installation disc for information about the installation.

2.1.1 e*Way Installation Procedure

To install the Batch e*Way on Windows systems

1. Log on as an Administrator on the workstation where you want to install the e*Way.
2. Insert the e*Way installation CD-ROM into the CD-ROM drive.
3. If the CD-ROM drive's **Auto-run** feature is enabled, the setup application should launch automatically; skip ahead to step 4. Otherwise, use the Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the **setup.exe** file on the CD-ROM drive.
4. After the **InstallShield** setup application launches, follow the on-screen instructions to install the e*Way.

Be sure to install the e*Way files in the suggested **client** installation directory. The installation utility detects and suggests the appropriate installation directory.

Caution: *Unless you are directed to do so by Oracle support personnel, do not change the suggested installation directory setting.*

2.1.2 After Installation

Once you have installed and configured this e*Way, you must incorporate it into a schema before it can perform its intended functions. This is done by defining and incorporating the following:

- Collaborations
- Collaboration Rules
- Intelligent Queues (IQs)
- Event Types
- Event Type Definitions (ETDs)

See the *e*Gate Integrator User's Guide* for details on incorporating the e*Way into a schema.

2.2 UNIX

Pre-installation

You do not require root privileges to install this e*Way. Log on under the name of the user who owns the e*Way files. Be sure that this user has sufficient privileges to create files in the e*Gate directory tree.

2.2.1 Installation Procedure

To install the Batch e*Way on a UNIX system

1. Log on to the workstation containing the CD-ROM drive, and insert the CD-ROM into the drive.
2. If necessary, mount the CD-ROM drive.
3. At the shell prompt, type:

   cd /cdrom/setup
4. Start the installation script by typing:

   setup.sh
5. A menu of options appears. Select the **e*Gate Add-on Applications** option. Then, follow any additional on-screen directions.

Be sure to install the e*Way files in the suggested **client** installation directory. The installation utility detects and suggests the appropriate installation directory.

Caution: *Unless you are directed to do so by Oracle support personnel, do not change the suggested installation directory setting.*

2.2.2 After Installation

Once you have installed and configured this e*Way, you must incorporate it into a schema before it can perform its intended functions. This is done by defining and incorporating the following:

- Collaborations
- Collaboration Rules
- Intelligent Queues (IQs)
- Event Types
- Event Type Definitions (ETDs)

See the *e*Gate Integrator User's Guide* for details on incorporating the e*Way into a schema.

2.3 Files/Directories Created by the Installation

On both Windows and UNIX machines, the Batch e*Way installation installs the files shown in Table 1. Files are installed within the **eGate\client** directory on the Participating Host and committed to the "default" schema on the Registry Host.

Table 1 Files Created by Installation

Directories	Files
classes	stcbatchext.jar stcbatch.jar stcewcommoneway.jar
ThirdParty\NetComponents\classes	NetComponents-1.3.8a.jar
configs\FtpHeuristics	FtpHeuristics.cfg FtpHeuristics.def FtpHeuristics.sc
etd\batchclient	FtpFileETD.xsc
etd\batchclientext	FtpETD.xsc LocalFileETD.xsc BatchRecordETD.xsc
configs\batchclient	FtpFileETD.def
configs\batchclientext	FtpETD.def LocalFileETD.def BatchRecordETD.def
etd	batchftp.ctl localfile.ctl batchrecord.ctl ftpfile.ctl

Note: *The files installed in the* ***etd\batchclient*** *and* ***configs\batchclient*** *directories are from the previous version of the e*Way and are installed to maintain backward compatibility. The files installed in* ***etd\batchclientext*** *and* ***configs\batchclientext*** *are for the newest version of the e*Way.*

Some of the files used by this e*Way are installed by the Monk programming language-enabled version of the e*Way. For information on these files, see the *Batch e*Way Intelligent Adapter User's Guide (Monk Version)*.

Chapter 3

Multi-Mode e*Way Configuration

The Batch e*Way Intelligent Adapter employs the Multi-Mode e*Way (a core e*Gate component) to communicate with external systems and within the e*Gate system.

This chapter describes how to configure the Multi-Mode e*Way.

3.1 Multi-Mode e*Way Properties

You will use the e*Gate Schema Designer to set the Multi-Mode e*Way properties. This section provides instructions for setting the properties you must configure before you can use an e*Way.

Note: *For complete information on how to create and configure the Multi-Mode e*Way, see the* ***Standard e*Way Adapter User's Guide****.* **Chapter 7** *provides more detailed procedures as part of explaining the e*Way's schema samples.*

To set necessary properties for a Multi-Mode e*Way

1. Click the **Components** tab in the Navigator pane of Main window of the Schema Designer.
2. Open the host and Control Broker where you want to create the e*Way.
3. Click the **new e*Way** icon.
4. Enter the name of the new e*Way, then click **OK**.
5. Select the new component, then click the **Properties** icon to edit its properties.

   The **e*Way Properties** dialog box opens
6. Click **Find** beneath the **Executable File** field, and select an executable file (**stceway.exe** is located in the **bin** directory).
7. Under the **Configuration File** field, click **New**.

   The e*Way Configuration Editor window opens.
8. When the **Settings** page opens, set the configuration parameters for this e*Way's configuration file.

Note: *See* **"JVM Settings" on page 24** *and* **"General Settings" on page 27** *for detailed information about these configuration parameters.*

9 When you are finished setting the configuration parameters, click **Save** on the **File** menu. This will save your settings to the configuration (**.cfg**) file.

10 Close the **.cfg** file and e*Way Configuration Editor.

11 Set the properties for the e*Way in the **e*Way Properties** dialog box.

12 Click **OK** to close the dialog box and save the properties.

3.2 JVM Settings

To correctly configure the Batch e*Way Intelligent Adapter, you must configure the Java Virtual Machine (JVM) settings. This section provides detailed information about these configuration parameters, which are displayed in the e*Way Configuration Editor window.

JNI DLL Absolute Pathname

Description

Specifies the absolute path of the installation location of the JNI **.dll** (Windows) or shared library (UNIX) file is installed by the Java SDK on the Participating Host, for example:

C:\eGate\client\bin\Jre or **C:\jdk\jre\bin\server**

This parameter is *mandatory*.

Required Values

A valid path name.

Additional Information

The JNI **.dll** or shared library file name varies, depending on the current operating system (OS). Table 2 lists the file names by OS.

Table 2 JNI.dll and shared library file names

Operating System	Java 2 JNI .dll or Shared Library Name
Windows systems	jvm.dll
Solaris	libjvm.so
HP-UX	libjvm.sl
AIX	libjvm.a
Compaq	libjvm.so
Red Hat Linux	libjvm.so

The value assigned can contain a reference to an environment variable. Enclose the variable name within a pair of "%" symbols, for example:

```
%MY_JNIDLL%
```

Such variables can be used when multiple Participating Hosts are used on different OS/platforms.

Caution: *To ensure that the JNI* ***.dll*** *file loads successfully, the Dynamic Load Library search-path environment variable must be set appropriately to include all the directories under the Java SDK installation directory, which contain shared library or* ***.dll*** *files.*

CLASSPATH Prepend

Description

Specifies the paths to be prepended to the CLASSPATH environment variable for the JVM.

Required Values

An absolute path or an environmental variable. This parameter is optional.

Additional Information

If left unset, no paths are prepended to the CLASSPATH environment variable.

Existing environment variables may be referenced in this parameter by enclosing the variable name in a pair of "%" symbols, for example:

```
%MY_PRECLASSPATH%
```

CLASSPATH Override

Description

Specifies the complete CLASSPATH variable to be used by the JVM. This parameter is optional. If left unset, an appropriate CLASSPATH environment variable (consisting of required e*Gate components concatenated with the system version of CLASSPATH) is set.

Note: *All necessary* ***.jar*** *and* ***.zip*** *files needed by both e*Gate and the JVM must be included. It is recommended that you use the* ***CLASSPATH Prepend*** *parameter.*

Required Values

An absolute path or an environment variable. This parameter is optional.

Additional Information

Existing environment variables may be referenced in this parameter by enclosing the variable name in a pair of "%" symbols, for example:

```
%MY_CLASSPATH%
```

CLASSPATH Append From Environment Variable

Description

Specifies whether the path is appended for the CLASSPATH environmental variable to .**jar** and .**zip** files needed by the JVM.

Required Values

YES or **NO**. The configured default is **YES**.

Initial Heap Size

Description

Specifies the value for the initial heap size in bytes. If this value is set to 0 (zero), the preferred value for the initial heap size of the JVM is used.

Required Values

An integer from **0** to **2147483647**. This parameter is optional.

Maximum Heap Size

Description

Specifies the value of the maximum heap size in bytes. If this value is set to 0 (zero), the preferred value for the maximum heap size of the JVM is used.

Required Values

An integer from **0** to **2147483647**. This parameter is optional.

Maximum Stack Size for Native Threads

Description

Specifies the value of the maximum stack size in bytes for native threads. If this value is set to 0 (zero), the default value is used.

Required Values

An integer from **0** to **2147483647**. This parameter is optional.

Maximum Stack Size for JVM Threads

Description

Specifies the value of the maximum stack size in bytes for JVM threads. If this value is set to 0 (zero), the preferred value for the maximum heap size of the JVM is used.

Required Values

An integer between 0 and 2147483647. This parameter is optional.

Disable JIT

Description

Specifies whether the Just-In-Time (JIT) compiler is disabled.

Required Values

YES or **NO**.

Remote Debugging Port Number

Description

Specifies whether to allow remote debugging of the JVM.

Required Values

YES or **NO**.

Suspend Option for Debugging

Description

Indicates whether to suspend the Option for Debugging on JVM startup.

Required Values

YES or **NO**.

3.3 General Settings

This section contains the parameters for rollback wait and IQ messaging priority.

Note: *For more information on the* ***General Settings*** *configuration parameters, see the* ***e*Gate Integrator User's Guide****.*

3.3.1 Rollback Wait Interval

Description

Specifies the time interval to wait before rolling back the transaction.

Required Values

A number within the range of 0 to 99999999, representing the time interval in milliseconds.

3.3.2 Standard IQ FIFO

Description

Specifies whether the highest priority messages from all Oracle SeeBeyond Standard IQs are delivered in the first-in-first-out (FIFO) order.

Required Values

Select **Yes** or **No**. **Yes** indicates that the e*Way retrieves messages from all Oracle SeeBeyond Standard IQs in the first-in-first-out (FIFO) order. **No** indicates that this feature is disabled; **No** is the default.

Chapter 4

e*Way Connection Configuration

This chapter explains how to configure e*Way Connections for the Batch e*Way Intelligent Adapter.

4.1 Configuring e*Way Connection Properties

You will use the e*Gate Schema Designer to set e*Way connection properties. This section provides instructions for setting the properties you must configure before you can use an e*Way connection with the Batch e*Way.

Note: *For complete information on how to create and configure e*Way Connections, see the **e*Gate Integrator User's Guide**. **Chapter 7** provides more detailed procedures as part of explaining the e*Way's schema samples.*

To set necessary properties for a Batch e*Way Connection

1. Click the Components tab in the Navigator pane of the Main window of the Schema Designer.
2. Select the **e*Way Connections** folder.
3. On the palette, click the **new e*Way Connection** icon.

   The **New e*Way Connection Component** dialog box appears.
4. Enter a name for the **e*Way Connection**, then click **OK**.

   An icon for your new e*Way Connection appears in the Component pane.
5. Double-click the new **e*Way Connection** icon.

   The **e*Way Connection Properties** dialog box appears.
6. Select **Batchext** from the **e*Way Connection Type** drop-down box.
7. Enter the appropriate **Event Type "get" interval** in the dialog box provided.
8. Under **e*Way Connection Configuration File**, click **New**.

Note: *To use an existing e*Way Connection configuration file, click **Find**.*

9 From the **ETD Template Selection** dialog box, select the desired Event Type Definition (ETD). Your options are:

- **BatchRecordETD**: Record-processing ETD
- **FtpETD**: FTP ETD
- **LocalFileETD**: Local file ETD

Note: *If you require backward compatibility, you can select* ***Batch*** *for the e*Way Connection type (step 6 above). Then, you will be able to select the* ***FtpFileETD.xsc*** *ETD template in the ETD Template Selection dialog box. However, if you do not need backward compatibility, it is recommended that you use* ***FtpETD.xsc*** *for FTP operations.*

10 After you make your selection, click **OK**. The e*Way Configuration Editor window appears.

Note: *For complete information on how to use the e*Way Configuration Editor, see the* ***e*Gate Integrator User's Guide****.*

11 Select the appropriate configuration parameters available in the e*Way Configuration Editor window to create a new configuration file for this e*Way Connection (this chapter explains these parameters).

12 When you are finished, save the new configuration file and close the e*Way Configuration Editor.

Note: **Chapter 5** *contains a detailed explanation of the Batch e*Way's ETDs.*

Each Batch e*Way Connection component utilizes one of the e*Way's ETDs. You can use the e*Way Configuration editor to set each component's configuration parameters.

This chapter explains these parameters as follows:

Note: *Individual ETD configuration settings can override e*Way Connection component settings. For more information, see* **Chapter 5**.

In addition, this chapter provides a section that explains how to use the e*Gate Connection Manager: **"Connection Manager" on page 84**.

4.2 BatchRecordETD: Configuration Parameters

This section explains the configuration parameters for the Batch e*Way Connection with the record-processing ETD (**BatchRecordETD.xsc**).

4.2.1 General Settings Configuration

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 e*Way Connection for the record-processing ETD is used. Set this parameter as follows:

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

An instance of the ETD can be used for parsing an inbound payload (only) or for creating an outbound payload (only). A single ETD cannot be used for both purposes at the same time in the same Collaboration.

Required Values

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

4.2.2 Record Configuration

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

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 e*Gate. 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.
- **User defined**: This option allows for user extensibility. If it is chosen, the semantics of what constitutes a record are defined by your own implementation of the parser interface. For more information on the e*Way's extensibility features, see **Chapter 6**.

Note: *If you select **User Defined**, you must enter a Java class name under the **User Class** (see **"User Class Configuration" on page 34**) configuration settings.*

Required Values

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

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

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 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.

4.2.3 User Class Configuration

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 e*Way.

This option is only used when the **User Defined** parameter is selected under the **Record Type** (see **"Record Type" on page 32**) 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 supplied implementations.

Note: *See* **Chapter 6** *for information on how to extend the e*Way.*

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 e*Way.

This option is only used when the **User Defined** parameter is chosen under the **Record Type** (see **"Record Type" on page 32**) settings, and you have supplied a class name for the **User Class** parameter. This parameter is ignored by all 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 e*Gate or the e*Way 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 e*Way at run time.

For more information on the e*Way's extensibility features, see **Chapter 6**.

Required Values

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

4.2.4 Connector Configuration

This section allows you to configure the e*Gate Collaboration engine to identify the e*Way Connection with the record-processing ETD.

Note: *For information on how to use the Connection Manager, see* **"Connection Manager" on page 84***.*

Type

Description

Allows you to specify the type of e*Way Connection.

Required Values

The e*Gate name of the record-processing ETD. The value defaults to **BatchRecordETD**.

Class

Description

Allows you to specify the class name of the Batch e*Way ETD 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.

4.3 FtpETD: Configuration Parameters

This section provides information about the configuration parameters for the Batch e*Way Connection with the FTP ETD (**FtpETD.xsc**).

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.*

4.3.1 General Settings Configuration

This section allows you to configure the **General Settings** parameter, **Transaction Type**.

Transaction Type

Description

Allows you to specify the transaction type, that is, whether to use the XA-compliant Guaranteed Exactly Once Delivery (GEOD) of Events.

Your options are:

- **Non-Transactional**: Do *not* use GEOD (no XA mode).
- **XA-compliant**: Use GEOD; enables the XA mode.

Required Values

Non-Transactional (the default) or **XA-compliant**.

4.3.2 FTP Configuration

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

Directory Listing Style

Description

Allows you to select 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, 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**, or **MPE**.

Note: *For more information, see* **"Using FTP Heuristics" on page 74***.*

Host Name

Description

Allows you to specify the name of the external system that the e*Way 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 Configuration" on page 50** 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 Configuration" on page 48** for details.

Required Values

A valid host name.

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 Configuration" on page 50** 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.

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()**. For complete information on the e*Way's Java methods, see Chapter 10.

Required Values

A valid password.

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 cause the e*Way to enter 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**.

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**.

4.3.3 Target Location Configuration

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

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).

For more information on using regular expressions with FTP and the e*Way, see **"Using Regular Expressions" on page 112**.

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.

For more information on regular expressions you can use with the e*Way, see **"Using Regular Expressions" on page 112**.

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).

For more information on using regular expressions with FTP and the e*Way, see **"Using Regular Expressions" on page 112**.

Required Values

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

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 e*Way 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.

Note: *Append is not supported in the XA mode (GEOD feature).*

Required Values

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

4.3.4 Pre Transfer Configuration

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 103**.

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 file.

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, the system generates an error that results in exception's being thrown in the called ETD 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 exception's being thrown in the Collaboration.

Required Values

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

Pre Directory Name

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 115** 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 115** 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**.

4.3.5 Post Transfer Configuration

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 103**.

Post Transfer Command

Description

Allows you to execute a desired action directly after the actual file transfer or during the "commit" phase when the **Transaction Type** parameter is set to **XA-Compliant**.

Note: *For more information on the e*Way's XA-compliant GEOD features, see* **"Guaranteed Exactly Once Delivery" on page 347**.

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, the system generates an error that results in exception's being thrown in the called ETD 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 exception's being thrown in the Collaboration.

Required Values

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

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 115** 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 115** 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**.

4.3.6 FTP Raw Commands Configuration

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

Note: *If you are using the XA mode for GEOD transmission, you cannot use FTP raw commands.*

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 e*Way in any way.*

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 e*Way in any way.*

4.3.7 Sequence Numbering Configuration

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

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 e*Way 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**.

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 e*Way 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**.

4.3.8 SOCKS Configuration

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

- No-authentication
- User/password

For more information on SOCKS, see **"SOCKS Support" on page 342**.

Socks Enabled

Description

Allows you to specify whether the FTP command connection goes through a SOCKS server.

If you choose **No**, the e*Way 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 37** *for more details.*

Required Values

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

Socks Host Name

Description

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

Required Values

A valid SOCKS server host name.

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 Version

Description

Allows you to specify the SOCKS server version. If you choose **Unknown**, the e*Way 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).

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.

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.

4.3.9 Batch e*Way and SSH Tunneling

This section provides a brief explanation of how the e*Way supports Secure Shell (SSH) tunneling.

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

Additional SSH-supporting Software

The e*Way'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 e*Way Connection.

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

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

Under the e*Way'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 e*Way connects to the FTP server **apple:21** through an SSH tunnel. For more information on SSH tunneling, see **"SSH Tunneling" on page 344**.

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

4.3.10 SSH Tunneling Configuration

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

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 Channel Established

Description

Allows you to specify whether the e*Way needs to launch an SSH subprocess.

Selecting **No** means the SSH channel has not yet been established. The e*Way 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 e*Way, and the e*Way does not need to establish it. For example, you could have issued a command outside of e*Gate, or you could know that another Batch e*Way instance has already established the channel by the time this e*Way 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 e*Gate 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 e*Way'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 e*Way instances located on different e*Gate 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 e*Gate 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 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 e*Way (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.

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.

4.3.11 Extensions Configuration

This section allows you to configure the **Extensions** configuration parameters. These parameters allow you to use the extensibility features of the e*Way.

Provider Class Name

Description

Allows you to enter the name of a Java class you are using to extend the capabilities of the FTP ETD's provider interface.

This is an advanced parameter that allows you to replace e*Way's native implementation of the provider interface (**FtpFileProvider**). For details on extending the e*Ways's capabilities, see **Chapter 6**.

The user class specified here must either:

- Implement the interface **com.stc.eways.batchext.FtpFileProvider**
- Extend the class **com.stc.eways.batchext.FtpFileProviderImpl**

Required Values

You must enter either:

- **com.stc.eways.batchext.FtpFileProviderImpl**
- A user class that implements the **com.stc.eways.batchext.FtpFileProvider** interface or extends **FtpFileProviderImpl**

Client Class Name

Description

Allows you to enter the name of a Java class you are using to extend the capabilities of the FTP ETD's client interface.

This is an advanced parameter that allows you to replace e*Way's native implementation of the client interface (**FtpFileClient**). For details on extending the e*Ways's capabilities, see **Chapter 6**.

The user class specified here must either:

- Implement the interface **com.stc.eways.batchext.FtpFileClient**
- Extend the class **com.stc.eways.batchext.FtpFileClientImpl**

Required Values

You must enter either:

- **com.stc.eways.batchext.FtpFileClientImpl**
- A user class that implements the **com.stc.eways.batchext.FtpFileClient** interface or extends **FtpFileClientImpl**

User Properties File

Description

Allows you to specify the name and path location of a Java properties file.

This is an advanced parameter and is part of the user-extensibility features of the e*Way's FTP ETD. This Java file is one you have used to provide custom properties to any replacement implementation of either of the following interfaces:

- **com.stc.eways.batchext.FtpFileClient**
- **com.stc.eways.batchext.FtpFileProvider**

The fully qualified file name is required. The contents of the file are loaded into a **java.util.Properties** instance during the ETD's initialization. These properties are then available at run time as the **UserProperties** node in the ETD configuration.

For more information on the e*Way's extensibility features, see **Chapter 6**.

Required Values

A valid path location and file name. The file must be a valid Java properties file with key/value pairs.

4.3.12 Connector Configuration

This section allows you to configure the e*Gate Collaboration engine to identify the e*Way Connection with the FTP ETD.

Note: *For information on how to use the Connection Manager, see* **"Connection Manager" on page 84***.*

Type

Description

Specifies the type of e*Way Connection.

Required Values

The e*Gate name of the FTP ETD. The value defaults to **batchftp**.

Class

Description

Specifies the class name of the Batch e*Way ETD connector object.

Required Values

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

Property.Tag

Description

Identifies the data source. This parameter is required by the current **EBobConnectorFactory**.

Required Values

A valid data source package name. Accept the default value.

Connection Establishment Mode

Description

Allows you to specify how a connection to the external system is established and closed:

- **Automatic** indicates that the connection is automatically established when the Collaboration is started, and it keeps the connection alive as needed.
- **OnDemand** indicates that the connection is established on demand, as Business Rules requiring a connection to the external system are performed. The connection is closed after the methods are completed.
- **Manual** indicates that you must explicitly call the connection **connect** and **disconnect** methods in the current Collaboration as Business Rules.

Note: *If you are using the Dynamic Configuration feature, you must be in the* ***Manual*** *mode and call the* ***connect()*** *and* ***disconnect()*** *methods.*

Required Values

Automatic, **OnDemand**, or **Manual**; the default is **Automatic**.

Connection Inactivity Timeout

Description

Allows you to specify the timeout (in milliseconds) for the **Automatic** connection establishment mode, so you only need to set this parameter if you are using this mode.

If you do not set this parameter (or set it to **0**), the connection is not closed by inactivity and is always kept alive. If it goes down, re-establishing the connection is automatically attempted.

If a non-zero value is specified, the e*Gate Connection Manager monitors the e*Way Connection for inactivity, and it is closed if the specified value is reached. For more information on the e*Gate Connection Manager, see **"Connection Manager" on page 84**.

Required Values

An integer (in milliseconds) representing the desired timeout interval.

Connection Verification Interval

Description

This value is used to specify the minimum period of time (milliseconds) between checks for the connection status to the FTP server. If the connection to the server is detected to be down during verification, the relevant Collaboration's **onConnectionDown()** method is called. If the connection comes from a previous connection error, the relevant Collaboration's **onConnectionUp()** method is called.

Required Values

An integer (in milliseconds) representing the desired verification interval; the default is 60,000 ms.

4.3.13 Dynamic Configuration

This section contains the parameters for the e*Way's Dynamic Configuration feature. These options allow you to provide information for your own use, via the FTP ETD's Dynamic Messaging configuration nodes.

Dynamic Messaging allows the e*Way to subscribe to XML messages that determine its activities. These messages can contain all the relevant parameters governing an FTP data transfer, including the data to be sent (if it is an outbound transfer).

Note: *In using this feature, keep in mind that the e*Way is Event-driven, so it does not exchange data based on scheduling.*

For a complete explanation of the Dynamic Configuration feature and its schema template, see **Chapter 8**.

Publish Status Record on Success

Description

If you want to publish a status record on every successful FTP transfer, set this parameter to **Yes**. This setting does *not* cause the e*Way to publish the message. Instead, it only passes the **Yes** setting information to the corresponding configuration node on the FTP ETD. If you set it to **No** (the default), that information is also passed.

You must create this feature yourself, in a Dynamic Configuration Collaboration. For example, you can configure the Collaboration to send a status record to e*Gate, with the format of **batch_eway_error.dtd**, when the payload has been successfully sent to the remote FTP host. In this case, you must configure an inbound topic and make sure this Event is processed.

You can set the **error_code** element of the XML message to zero (0) to indicate no errors and allow the **error_text** to represent the time the file was successfully transferred, for example:

```
Successfully sent on: Fri, 29 Jun 2001 at 14:02:30 PDT
```

Required Values

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

Publish Status Record on Error

Description

If you want to publish a status record on every FTP transfer error, set this parameter to **Yes**. This setting does *not* cause the e*Way to publish the message. Instead, it only passes the **Yes** setting information to the corresponding configuration node on the FTP ETD. If you set it to **No** (the default), that information is also passed.

You must create this feature yourself, in a Dynamic Configuration Collaboration. For example, you can configure the Collaboration to send a status record to e*Gate, with the format of **batch_eway_error.dtd**, whenever there has been an FTP error. In this case, you must configure an inbound topic and make sure this Event is processed.

Required Values

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

Include Order Record in Error Record

Description

If you want to include an order record as part of an error record when the **Publish Status Record on Error** parameter is selected, set this parameter to **Yes**. This setting does *not* cause the e*Way to publish the message. Instead, it only passes the **Yes** setting information to the corresponding configuration node on the FTP ETD. If you set it to **No** (the default), that information is also passed.

You must create this feature yourself, in a Dynamic Configuration Collaboration, along with your configuration for publishing the error status record.

Required Values

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

Include Payload in Error Record

Description

If you want to include the data payload as part of the error status record when the order record command is SEND, set this parameter to **Yes**. This setting does *not* cause the e*Way to publish the message. Instead, it only passes the **Yes** setting information to the corresponding configuration node on the FTP ETD. If you set it to **No** (the default), that information is also passed.

You must create this feature yourself, in a Dynamic Configuration Collaboration, along with your configuration for publishing the error status record.

Required Values

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

Action on Malformed Command

Description

If you want the e*Way to take a specific action whenever there is an FTP transfer error, choose the desired action here. This setting does *not* cause the e*Way to take this action. Instead, it only passes the setting information to the corresponding configuration node on the FTP ETD. The default is **Exit**.

You must configure these actions yourself, in a Dynamic Configuration Collaboration. The options in case of an error are:

- **Exit**: Shut down the e*Way.
- **Ignore**: The e*Way does nothing.
- **Raise Alert**: Send an Alert to the Schema Manager.
- **Publish Error Record**: Publish an XML error status record (see the **Publish Status Record on Error** parameter.

Required Values

Ignore, **Raise Alert**, **Publish Error Record**, or **Exit** (the default).

Caution: *Be careful when using the **Ignore** option, because no action is taken. Data could be lost without your being able to take any recourse action.*

4.4 LocalFileETD: Configuration Parameters

This section provides the information about the configuration parameters for the Batch e*Way Connection with the local file ETD (**LocalFileETD.xsc**).

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.*

4.4.1 General Settings Configuration

This section provides information about configuring the **General Settings** parameter, **Transaction Type**.

Transaction Type

Description

Allows you to specify the transaction type, that is, whether to use the XA-compliant GEOD feature.

Your options are:

- **Non-Transactional**: Do *not* use GEOD (no XA mode).
- **XA-compliant**: Use GEOD; enables the XA mode.

Required Values

Non-Transactional (the default) or **XA-compliant**.

Resume Reading Enabled

Description

Allows you to specify whether the ETD handles the Resume Reading feature as follows:

- **Yes**: Enables the ETD 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.

Note: *This feature is available for inbound data-streaming transfers only. See* **"Resume Reading Feature" on page 106** *for more information.*

Required Values

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

4.4.2 Target Location Configuration

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

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.

For more information on using regular expressions with the e*Way, see **"Using Regular Expressions" on page 112**.

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 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.

For more information on using regular expressions with the e*Way, see **"Using Regular Expressions" on page 112**.

Required Values

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

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 e*Way 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.

Note: *Append is not supported in the XA mode (GEOD feature).*

Required Values

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

4.4.3 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 103**.

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**.

Pre Transfer 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 e*Way 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**.

For more information on special characters you can use with the e*Way, see **"Using Special Characters" on page 115**.

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 Transfer Name Is Pattern

Description

Allows you to specify the meaning of the **Pre Transfer 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.

For more information on using regular expressions with the e*Way, see **"Using Regular Expressions" on page 112**.

Required Values

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

4.4.4 Post Transfer Configuration

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

Note: *For more information on this feature, see* **"Pre/post File Transfer Commands" on page 103**.

Post Transfer Command

Description

Allows you to execute a desired action directly after the actual file transfer or during the "commit" phase when the **Transaction Type** parameter is set to **XA-Compliant**. 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.

Note: *For more information on the e*Way's XA-compliant GEOD features, see* **"Guaranteed Exactly Once Delivery" on page 347**.

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

Post Transfer 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. See **"Using Special Characters" on page 115** for details on using these characters.

If you are entering a path name, use the forward slash (/) instead of the back slash (\) because the e*Way 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.

For more information on using regular expressions with the e*Way, see **"Using Regular Expressions" on page 112**.

Post Transfer 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**.

4.4.5 Sequence Numbering Configuration

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

Starting Sequence Number

Description

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

Max Sequence Number

Description

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

4.4.6 Connector Configuration

This section provides information about configuring the e*Gate Collaboration engine to identify the e*Way Connection with the local file ETD.

Type

Description

Specifies the type of e*Way Connection.

Required Values

The e*Gate name of the local file ETD. The value defaults to **LocalFile**.

Class

Description

Specifies the class name of the Batch e*Way ETD connector object.

Required Values

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

Property.Tag

Description

Identifies the data source. This parameter is required by the current **EBobConnectorFactory**.

Required Values

A valid data source package name. Accept the default value.

4.4.7 Dynamic Configuration

This section contains the parameters for the e*Way's Dynamic Configuration feature. These options allow you to provide information for your own use through the local file ETD's Dynamic Messaging configuration nodes.

Dynamic Messaging allows the e*Way to subscribe to XML messages that determine its activities. These messages can contain all the relevant parameters governing a local file data transfer, including the data to be sent (if it is an outbound transfer).

Note: *In using this feature, keep in mind that the e*Way is Event-driven, so it does not exchange data based on scheduling.*

For a complete explanation of the Dynamic Configuration feature and its schema template, see **Chapter 8**.

Publish Status Record on Success

Description

If you want to publish a status record on every successful local file transfer, set this parameter to **Yes**. This setting does *not* cause the e*Way to publish the message. Instead, it only passes the **Yes** setting information to the corresponding configuration node on the local file ETD. If you set it to **No** (the default), that information is also passed.

You must create this feature yourself, in a Dynamic Configuration Collaboration. For example, you can configure the Collaboration to send a status record to e*Gate, with the format of **batch_eway_error.dtd**, when the payload has been successfully sent. In this case, you must configure an inbound topic and make sure this Event is processed.

You can set the **error_code** element of the XML message to zero (0) to indicate no errors and allow the **error_text** to represent the time the file was successfully transferred, for example:

```
Successfully sent on: Fri, 29 Jun 2001 at 14:02:30 PDT
```

Required Values

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

Publish Status Record on Error

Description

If you want to publish a status record on every data transfer error, set this parameter to **Yes**. This setting does *not* cause the e*Way to publish the message. Instead, it only passes the **Yes** setting information to the corresponding configuration node on the local file ETD. If you set it to **No** (the default), that information is also passed.

You must create this feature yourself, in a Dynamic Configuration Collaboration. For example, you can configure the Collaboration to send a status record to e*Gate, with the format of **batch_eway_error.dtd**, whenever there has been an error. In this case, you must configure an inbound topic and make sure this Event is processed.

Required Values

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

Include Order Record in Error Record

Description

If you want to include an order record as part of an error record when the **Publish Status Record on Error** parameter is selected, set this parameter to **Yes**. This setting does *not* cause the e*Way to publish the message. Instead, it only passes the **Yes** setting information to the corresponding configuration node on the local file ETD. If you set it to **No** (the default), that information is also passed.

You must create this feature yourself, in a Dynamic Configuration Collaboration, along with your configuration for publishing the error status record.

Required Values

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

Include Payload in Error Record

Description

If you want to include the data payload as part of the error status record when the order record command is SEND, set this parameter to **Yes**. This setting does *not* cause the e*Way to publish the message. Instead, it only passes the **Yes** setting information to the corresponding configuration node on the local file ETD. If you set it to **No** (the default), that information is also passed.

You must create this feature yourself, in a Dynamic Configuration Collaboration, along with your configuration for publishing the error status record.

Required Values

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

Action on Malformed Command

Description

If you want the e*Way to take a specific action whenever there is a transfer error, choose the desired action here. This setting does *not* cause the e*Way to take this action. Instead, it only passes the setting information to the corresponding configuration node on the local file ETD. The default is **Exit**.

You must configure these actions yourself, in a Dynamic Configuration Collaboration. The options in case of an error are:

- **Exit**: Shut down the e*Way.
- **Ignore**: The e*Way does nothing.
- **Raise Alert**: Send an Alert to the Schema Manager.
- **Publish Error Record**: Publish an XML error status record (see the **Publish Status Record on Error** parameter.

Required Values

Ignore, **Raise Alert**, **Publish Error Record**, or **Exit** (the default).

Caution: *Be careful when using the **Ignore** option, because no action is taken. Data could be lost without your being able to take any recourse action.*

4.5 FtpFileETD: Configuration Parameters

This section explains the configuration parameters for the Batch e*Way Connection with the FTP file ETD (**FtpFileETD.xsc**).

Note: *This ETD and its corresponding Java implementation is provided for backward compatibility because its functionality has been supplanted by the newer and more functional **FtpETD.xsc**. It is recommended that you use the **FtpETD.xsc** ETD for all new development.*

4.5.1 Connector Configuration

This section provides configuration information for the e*Gate Collaboration engine to identify the e*Way Connection with the FTP file ETD.

Type

Description

Specifies the type of connection.

Required Values

The e*Gate name of the FTP ETD. The value defaults to **ftpfile**.

Class

Description

Specifies the class name of the FTP file connector object.

Required Values

A valid class name. The default is **com.stc.eways.batch.FtpConnector**.

Property.Tag

Description

Identifies the data source. This parameter is required by the current **EBobConnectorFactory**.

Required Values

A valid data source package name.

4.5.2 FTP File Configuration

This section lists the e*Way Connection configuration parameters for the FTP file ETD.

Directory Listing Style

Description

Select 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

From the list provided, select the name of the desired system.

Host Name

Description

The name of the external system that the e*Way connects to.

Required Values

Enter the name of the external host system, for example, **ftphost**.

User Name

Description

When a log on to the external system is required, enter the logon user name to be used.

Required Values

Enter the desired user name.

Password

Description

If a password is required in order to log on to the external system, enter the password that corresponds to the given user name.

Required Values

Enter the required password.

Mode

Description

Allows you to specify the mode used to transfer data to and from the FTP server, using the **Ascii** or **Binary** mode.

Required Values

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

Use PASV

Description

Causes the e*Way to enter the passive or active mode.

Required Values

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

Server Port

Description

The port number to use on the FTP server when connecting to it.

Required Values

Enter the desired port number.

Remote Directory Name

Description

The directory (absolute path location) on the external system where files are retrieved or sent.

Required Values

Enter the desired directory name and path location.

Remote File Name

Description

For inbound (subscriber), it is the remote file name regular expression. For outbound (publisher), it is the remote file name.

For inbound, files in the remote directory that match the regular expression are retrieved to payload, through **get()**, for processing.

For outbound, this is the name of the file as it appears on the remote system for **put()**. Special characters for date and time and sequence numbering expansions may be used, which are expanded by the e*Way before the file is transmitted.

Required Values

Enter the appropriate remote file name, as specified previously. For example, for MVS GDG, this entry can be the version of the data set.

Additional Examples:

- **Remote Directory Name = 'STC.SAMPLE.GDGSET'**
- **Remote File Name = (0)** to indicate the current version

Overwrite Or Append

Description

Select the appropriate parameter, as follows:

- If **Append** is selected and the remote file already exists, then the payload is appended to the existing file.
- If **Overwrite** is selected, then the e*Way overwrites the existing file on the remote system.
- If a file with the same name does not exist, both **Append** and **Overwrite** create a new file on the external host.

This parameter is for outbound only.

Required Values

Select either **Append** or **Overwrite**, as directed previously.

Command After Transfer

Description

After a file has been successfully retrieved from or sent to the external system, the following actions can be performed on the remote copy: delete, rename, archive. Also, no action can be taken at all.

The rename and archive functions may not be available in all cases. In the case of FTP, they rely on the RNFR command being available on the remote FTP daemon.

When retrieving multiple files, use the **Rename** parameter with care. You set this value yourself, so, to use maximum caution, use name sequencing. There is no default.

Required Values

- **Delete**: Delete the file from the remote host.
- **Rename**: Rename the file.
- **Archive**: Move the file to another directory.
- **None**: Do nothing (leaves the file on the remote host intact).

Rename or Archive Name

Description

Depending on the value in the parameter **Command After Transfer**, this command either specifies the name of the file that the remote file is renamed to or the directory it is archived to (see **"Command After Transfer" on page 72**).

Required Values

Enter either the file or directory name, as explained previously.

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

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

Pre Transfer Raw Commands

Description

These are FTP raw commands needed *before* the file transfer command, for example, some SITE commands.

Note: *These commands are sent to the FTP server directly and are not interpreted by the e*Way in any way, so they must be valid FTP raw commands.*

Required Values

Enter the required FTP raw commands. Use semicolons (;) to separate the command set, for example: **PWD;CWD;SITE** (and so on).

Post Transfer Raw Commands

Description

These are FTP raw commands needed *after* the file transfer command, for example, some SITE commands.

Note: *These commands are sent to the FTP server directly and are not interpreted by the e*Way in any way, so they must be valid FTP raw commands.*

Required Values

Enter the required FTP raw commands. Use semicolons (;) to separate the command set, for example: **PWD**;**CWD**;**SITE** (and so on).

Starting Sequence Number

Description

Use this parameter when you have set up the remote file name to contain a sequence number. It tells the e*Way which value to start with in the absence of a sequence number from a previous run.

Also, when the **Max Sequence Number** is reached, the sequence number rolls over to the **Starting Sequence Number**.

This parameter is used for the name pattern %#.

Required Values

The value of the **Starting Sequence Number** *must* be less than the **Max Sequence Number**. The default value is 1.

Max Sequence Number

Description

Use this parameter when you have set up the remote file name to contain a sequence number. It tells the e*Way that when this value (the **Max Sequence Number**) is reached, to reset the sequence number to the **Starting Sequence Number**.

This parameter is used for the name pattern %#.

Required Values

The value of the **Max Sequence Number** *must* be greater than the **Starting Sequence Number**.

4.6 Using FTP Heuristics

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

FTP Heuristics: e*Way Operation

The FTP heuristics are a set of parameters that the e*Way 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' platforms (operating systems).

You do not normally need to change any of the FTP heuristics, since the default parameters have been set up for the most commonly used platforms. However, these parameters are provided in case any changes are necessary to accommodate your site's requirements.

FTP heuristics are stored in the file **FtpHeuristics.cfg**. If you want to modify the e*Way's FTP heuristics to accommodate an additional host type, you must make the addition by manually editing the following files:

- **FtpHeuristics.cfg**
- **FtpHeuristics.def**
- **FtpHeuristics.sc**
- **FtpETD.def**

Note: *If you are using the backward-compatibility **FtpFileETD.xsc** ETD instead of **FtpETD.xsc**, you must edit the **FtpFileETD.def** file instead of **FtpETD.def**. However, it is recommended that you use **FtpETD.xsc** for all new development.*

Platform or File Type Selection

Each platform defined within the FTP heuristics file sets the parameters listed in this section. In the e*Way Configuration Editor, the platform is selected using the appropriate e*Way Connection configuration parameter.

The e*Way's FTP heuristics support the following file types:

- UNIX
- 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

The FTP heuristic methods used for communication with MVS Sequential, MVS GDG, and MVS PDS for the Batch e*Way are designed for FTP servers (at the mainframe) that use the IBM IP stack. See **Chapter 10** for an overview of the e*Way's methods, or you can refer to the Javadoc for complete details.

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 do not work or can require modification.

4.6.1 Configuration Parameters

The section explains the configuration parameters for FTP heuristics feature of the Batch e*Way. The e*Way Configuration Editor allows you to configure this complete set of parameters for each of the platforms listed under **"Platform or File Type Selection" on page 75**.

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 3).

Table 3 Delimiters and Path Naming by Platform

OS	Path Name Format	Delimiter Set				
		1	2	3	4	Enter
UNIX	/dir1/dir2/file.ext	/	/	/		///
Windows	C:\dir1\dir2\file.ext	\\	\\	\\		\\\\\\
VMS	disk1:[dir1.dir2]file.ext;1	[	.	]	;	[.];
MVS PDS	dir1.dir2(member)	\0	.	(	)	\0.()

Table 3 Delimiters and Path Naming by Platform (Continued)

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

* 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 e*Way 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**.

4.7 Connection Manager

The e*Gate Connection Manager allows you to define external connection functionality of an e*Way. You can choose:

- When to make a connection
- When to close a connection and disconnect
- Connection status

4.7.1 Using the Connection Manager

The Connection Manager is controlled in the Batch e*Way's configuration, as explained in **"Connector Configuration" on page 35**.

Table 4 provides additional information on manually controlling the connections.

Table 4 e*Way Connection Control Settings

Method or Action	Automatic	On-demand	Manual
onConnectionUp()	Yes	No	No
onConnectionDown()	Yes	Yes only if the connection attempt fails	No
Automatic Transaction (GEOD/XA)	Yes	No	No
Manual Transaction	Yes	No	No
connect()	No	No	Yes
isConnected()	No	No	Yes
disconnect()	No	No	Yes
Timeout or connect()	Yes	Yes	No
Verify connection interval	Yes	No	No

4.7.2 Controlling Connection Timing and Status

This section explains how you can control when a connection is made, when it is disconnected, and connectivity status.

When a Connection Is Made

Using the **Connector** parameters, you can choose to have an e*Way's connections controlled manually, through the Collaboration, or automatically, through the e*Way Connection's configuration.

If you choose to control a connection, you can specify:

- To connect when the Collaboration is loaded
- To connect when the Collaboration is executed
- To connect by using an additional connection method in the ETD
- To connect by overriding any custom values you have assigned in the Collaboration
- To connect by using the **isConnected()** method (called per connection if your ETD has multiple connections)

When a Connection is Disconnected

In addition to controlling when a connection is made, you can also manually or automatically control when an e*Way's connection is terminated or disconnected.

To control the disconnect, you can specify:

- To disconnect at the end of a Collaboration
- To disconnect at the end of the execution of the Collaboration's Business Rules
- To disconnect during a timeout
- To disconnect after a method call

Connectivity Status

You can control how often an e*Way Connection checks to verify whether its external connection is still alive. You can also set how often it checks. See the following sections for more information:

- **"Connector Configuration" on page 55** for how to set connection control-related configuration parameters
- **Table 4 on page 85** for a list of connection control settings

Chapter 5

e*Way Event Type Definitions

This chapter explains the specialized Event Type Definitions (ETDs) available with the Batch e*Way Intelligent Adapter.

5.1 e*Way ETDs: Overview

The Batch e*Way contains separate ETD components that access the e*Way's basic functions in the e*Gate Integrator system. Each ETD allows you to use a different functional set of the e*Way's features.

5.1.1 Types of ETDs

Table 5 shows the specialized ETDs available with the e*Way.

Table 5 Batch e*Way ETDs

ETD Name	File Name	Description
FTP	FtpETD.xsc	Provides FTP access to remote systems.
Record-processing	BatchRecordETD.xsc	Allows the e*Way to parse or create (or both) payloads of records in specified formats.
Local file	LocalFileETD.xsc	Provides easy access to local file systems.
FTP file	FtpFileETD.xsc	Provides FTP access; supported for backward compatibility with a previous e*Way version.

Each ETD also has its own **.def** file provided for e*Way Connection configuration (see Chapter 4 for details), for example **FtpETD.def**.

Note: *The **FtpFileETD.xsc** ETD and its corresponding Java implementation is from a previous version of the e*Way. It is provided only for backward compatibility. It is recommended that you use the newer **FtpETD.xsc** for all new development.*

This chapter explains each of these ETDs and how to use them with the e*Way.

5.1.2 ETD Components

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

- **ETD Operation**: The ETD itself is an **.xsc** file in e*Gate and can be used in a Collaboration Rule to operate with e*Way Connections.
- **Definition and e*Gate Schema Designer**: An **e*Way Connection Properties** dialog box is available, providing a central location in which you can define the e*Way's properties. The Schema Designer also allows you to access the e*Way Configuration Editor.
- **e*Way Connection**: An e*Way Connection provides access to the information necessary to interface with a specified external connection (more information about e*Way Connections is provided in **Chapter 4**).
- **Configuration**: A **.def** file accompanies each ETD. You can use this file to configure any of the ETD's e*Way Connections. The e*Way Configuration Editor uses **.def** files to set the e*Way's configuration parameters.

All ETDs must be configured and administered using the Schema Designer.

Note: *For complete information on how to use the Schema Designer and the e*Way Configuration Editor, see the* ***e*Gate Integrator User's Guide****.*

Client Components

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

5.2 ETD for FTP Operations

The Batch e*Way includes an ETD that allows you to perform FTP data-transfer functions, the FTP ETD. Although this ETD is a component of the Batch e*Way, you can also use it as a stand-alone module outside of the e*Way to provide FTP functionality for any purpose.

The combination of this ETD, working with a specific e*Way Connection with its own set of configurable parameters, defines the characteristics of the external interface. Using the FTP ETD and one or more e*Way Connections, you can create the Collaboration Rules to make the Batch e*Way behave in specific ways, as desired.

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

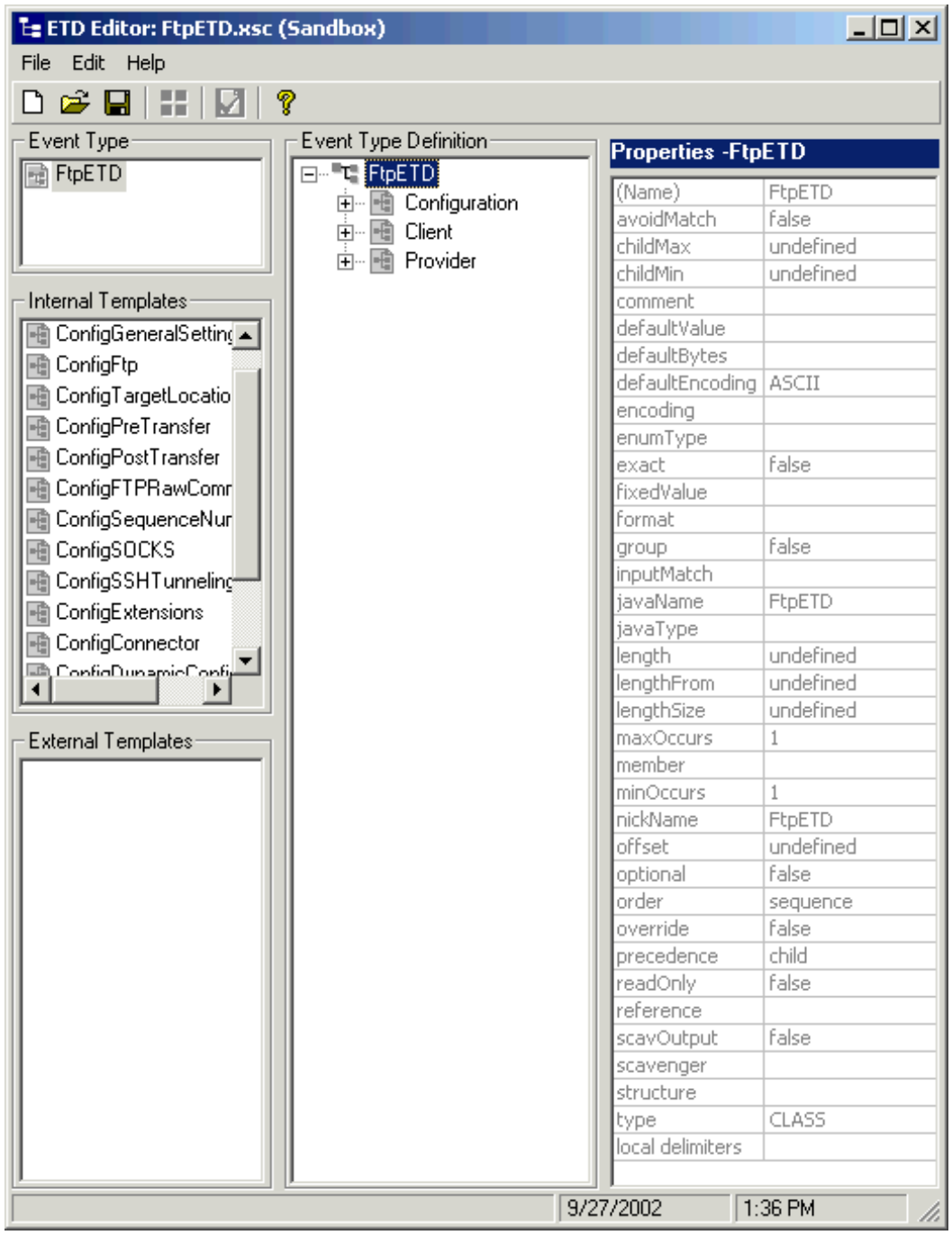
The FTP ETD enables the e*Gate system to exchange data with other network hosts for the purpose of receiving and delivering Events stored in files. The FTP ETD data payload uses a byte array. You must also use a byte array for the payload copy, specifically for binary transfers.

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

5.2.1 ETD Structure and Operation

Figure 4 on page 90 shows the FTP ETD as it appears in the e*Gate Schema Designer ETD Editor's Main window.

Figure 4 FTP ETD Structure: Top-level Nodes



As illustrated in Figure 4, the FTP ETD contains three top-level nodes, Configuration, Client, and Provider. Each is described in the sections to follow. You can expand these nodes in the ETD Editor to reveal additional sub-nodes.

Configuration Node

Each field sub-node in the **Configuration** node of the ETD corresponds to one of the e*Way Connection's FTP configuration parameters.

Client and Provider Nodes

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

- 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 ETD.

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

Note: *For more information on the ETD's client and provider interfaces, as well as ETD extensibility, see* **Chapter 6**.

5.2.2 FTP ETD Node Functions

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

- **FtpETD**: Represents the ETD's root node.
- **Configuration**: Each field sub-node within this node corresponds to an e*Way Connection configuration parameter and contains settings information. See **"FtpETD: Configuration Parameters" on page 36** for details on these parameters and settings.

Note: *This ETD has configuration parameters that can be regular expressions. See* **"Using Regular Expressions" on page 112** *for details.*

- **Client**: This node contains the following sub-nodes, which implement the e*Way's client interface in the ETD (**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 6** for details); in such cases, the node represents the properties specified in the configuration.
  - **InputStreamAdapter** and **OutputStreamAdapter**: Allow you to use and control the ETD's data-streaming features; see **"Streaming Data Between Components" on page 329** 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 ETD.*

- **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 103** for details).

Note: *See **"Resolving Names" on page 116** for more information on these nodes; see **"Using Regular Expressions" on page 112** for more information on regular expressions.*

- **get()**, **put()**, **reset()**, **connect()**, **disconnect()**, and **isConnected()**: See **"Essential FTP ETD Methods" on page 93**.

- **Provider**: The sub-nodes contained in this node are methods that implement the e*Way's provider interface in this ETD (**FtpFileProvider**). These methods allow you to do the general FTP operations that can be performed using the ETD. See **Chapter 10** and the Javadoc for details on these methods.

5.2.3 Using the FTP ETD

Essentially, the FTP ETD nodes are a mirror image of the e*Way Connection. These nodes allow you to configure specific e*Way Connection (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 e*Way Connection component that uses this Collaboration.

Handling Type Conversions

The **Payload** node in the **FtpETD.xsc** structure is predefined as a byte array (**byte[]**). This definition allows the e*Way to handle both binary and character data.

For example, you could be using another ETD (such as an ETD from another e*Way or a user-defined ETD) where the "data" node has been defined as a string (**java.lang.String**). If you were to drag and drop that string to the FTP ETD's **Payload** node, the e*Gate 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 FTP ETD's node structure, see* **"FTP ETD Node Functions" on page 91**.

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 FTP ETD Methods

In addition to the field elements shown in **Figure 4 on page 90**, the FTP ETD's **Client** node contains methods that extend the client interface functionality of the e*Way. These methods are essential to the proper use of the ETD 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 e*Way's data-streaming feature, the **get()** and **put()** methods operate differently. See* **"Streaming Data Between Components" on page 329** *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 FTP and local file ETDs. It must be called when the ETD 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 ETD. If you attempt another transfer without calling **reset()** first, the system throws an exception and makes an entry in the e*Way's error log file.*

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

Note: *See* **Chapter 10** *and the Javadoc for more information on these methods.*

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 e*Way Connection configuration parameters for the ETD.

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

Starting Sequence Number

This parameter tells the e*Way 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 e*Way 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:

FTP ETD

"Sequence Numbering Configuration" on page 47

Local File ETD

This feature is also available with the local file ETD. For more information on these configuration parameters, see **"Sequence Numbering Configuration" on page 65**.

Additional FTP File Transfer Commands

The FTP ETD 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 103** for details.

5.3 ETD for Record Processing

The Batch e*Way's record-processing ETD 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 ETD 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 ETD:

- A large data file that contains a number of SAP IDocs, with each 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 ETD 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.

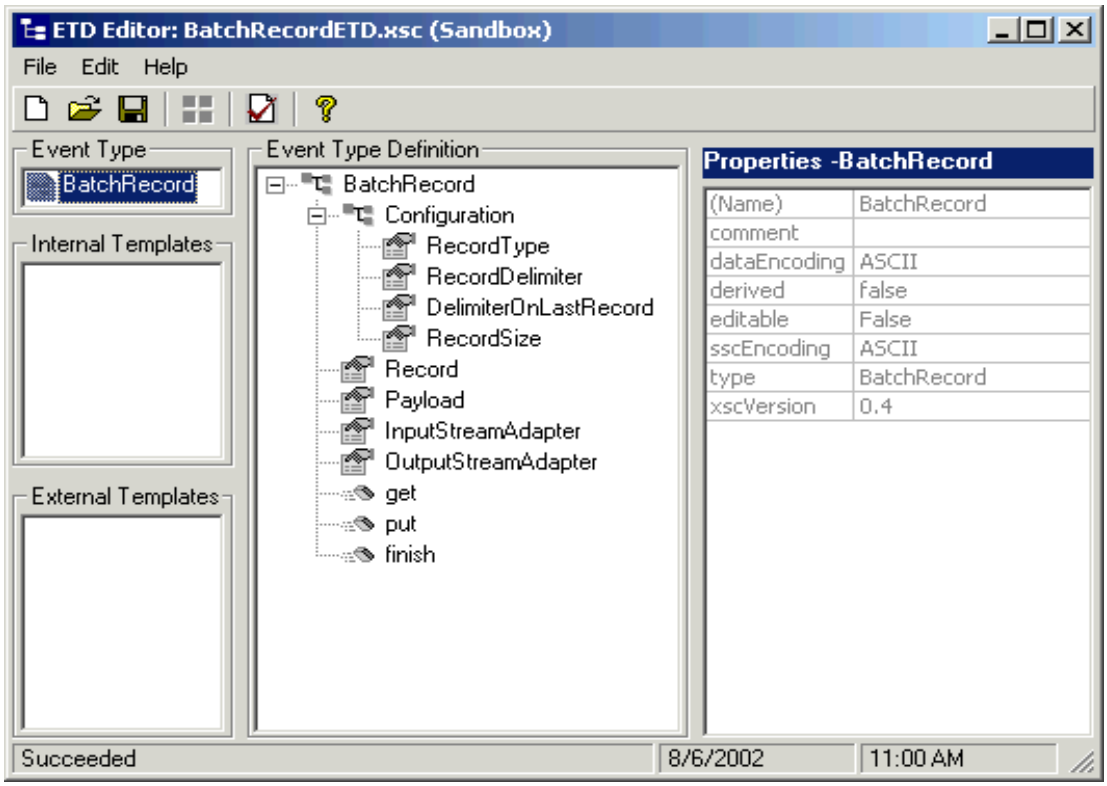
Note: *When using character delimiters with DBCS data, use single byte character(s) or equivalent hex values with hex values that do not coincide with either byte of the double byte character.*

You can also easily extend the ETD to handle other custom record formats. **Chapter 6** contains information on how to extend the ETD's capabilities.

5.3.1 ETD Structure and Operation

Figure 5 shows the record-processing ETD (BatchRecordETD) as it appears in the e*Gate Schema Designer ETD Editor's Main window.

Figure 5 Record-processing ETD Structure



As illustrated in Figure 5, each field node in the **Configuration** node in the ETD (Figure 5) corresponds to one of the e*Way Connection's record-processing configuration parameters.

Note: *For more information on the ETD's extensibility features, see* **Chapter 6**.

5.3.2 Record-processing ETD Node Functions

See Figure 5 for an illustration of this ETD as it appears in the ETD Editor's Main window. The following list explains these primary nodes in the record-processing ETD, including their functions:

- **BatchRecord**: Represents the ETD's root node.
- **Configuration**: Each sub-node within this node corresponds to an e*Way Connection configuration parameter and contains the corresponding settings information, except for the **Parse or Create** parameter. See **"BatchRecordETD: Configuration Parameters" on page 32** for details.

Note: *For the record-processing ETD, 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 ETD. For details on their operation, see **"Streaming Data Between Components" on page 329**.

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 ETD.*

- **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 ETD to clean up any unneeded internal data structures.*

5.3.3 Using the Record-processing ETD

This ETD 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 ETD 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 e*Way Connection'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 ETD's functionality. If you call either method, the record retrieved or added is assumed to be of the type specified in the e*Way Connection 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 e*Way 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 e*Way to throw an exception with an appropriate error message explaining why. Calling **get()** in the create mode would also result in an error.

The e*Way 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 has 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 ETD on the source or destination side of a given Collaboration, as desired, and use the ETD for either parsing or creating a payload. However, you cannot parse and create at the same time. Implement your ETD in a Collaboration using the e*Gate Collaboration Rules Editor.

Creating a Payload

When you want the payload data sent to an external system, you can place the ETD 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 e*Way Connection 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 9** 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 e*Way 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 e*Way.

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 ETD (in the Collaboration Rules Editor). In addition, you can specify an input stream as a source (see **Chapter 9** 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 e*Way Connection'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 ETD 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. See **Chapter 10** of this user's guide for details.

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.

Note: *See* **Chapter 6** *for information on how to extend the ETD's capabilities.*

Use With Data Streaming

If you are using the record-processing ETD with data streaming, you must be careful not to overwrite the output files. If the ETD is continually streaming to a local file ETD that uses the same output file name, the ETD 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 local file ETD 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 109**
- **"Pre/post File Transfer Commands" on page 103**

5.4 ETD for Local File

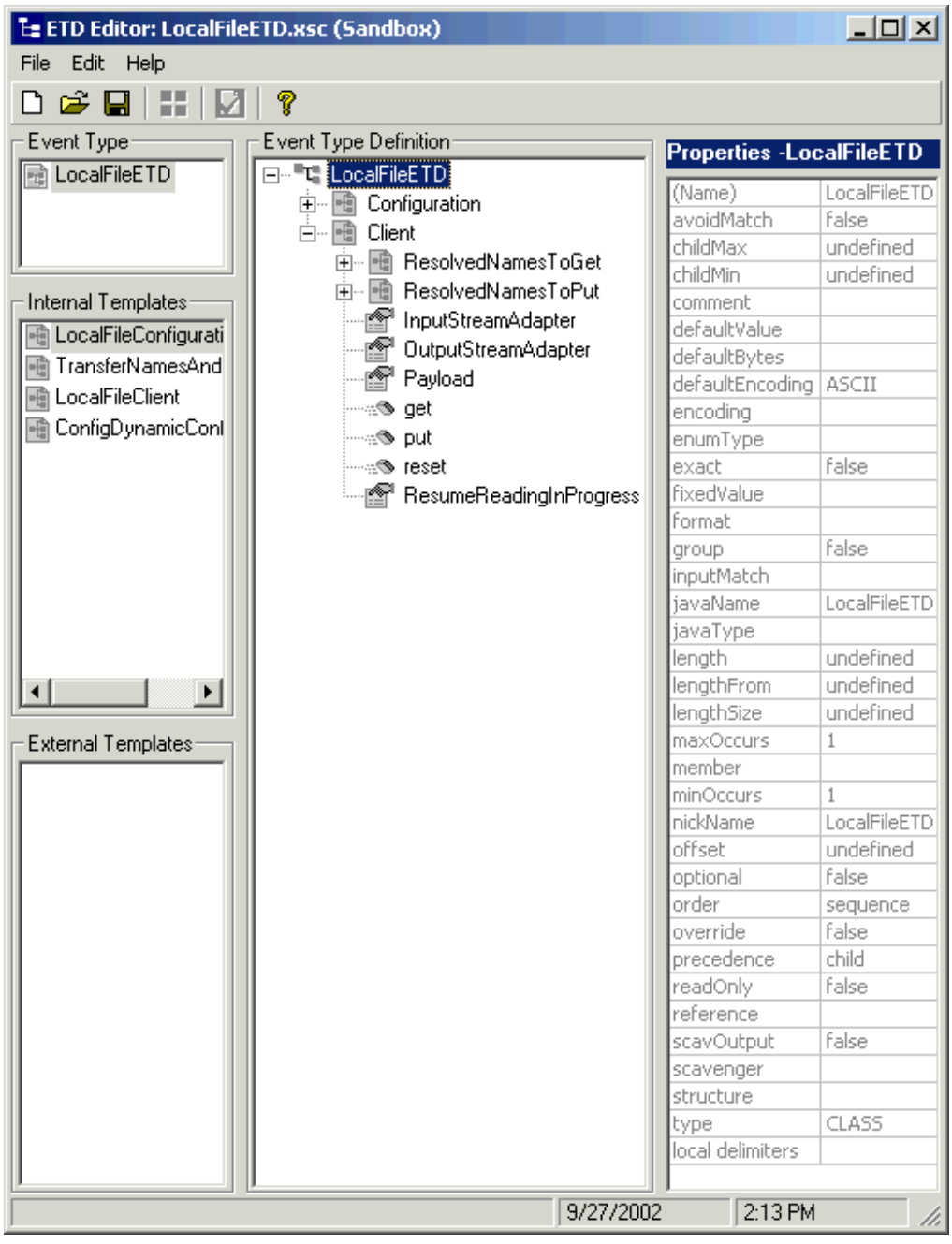
The local file ETD provides access to files on your local system. While file access is not always necessary in e*Gate, it makes sense for the Batch e*Way to have this feature because file processing is one of its core functionalities.

Additional local file features include regular expressions for accessing files and a sequence-numbering scheme for creating files. This section provides information about these features.

5.4.1 ETD Structure and Operation

Figure 6 on page 101 shows the local file ETD (LocalFileETD) as it appears in the e*Gate Schema Designer ETD Editor's Main window.

Figure 6 Local File ETD Structure



Configuration Node

As in the FTP ETD, each field sub-node under the **Configuration** node in the local file ETD (Figure 6) corresponds to one of the e*Way Connection's configuration parameters for that ETD. See **"LocalFileETD: Configuration Parameters" on page 59** for details.

Client Node

This ETD includes an additional top-level node, the **Client**. This node implements its respective functionality interface in the e*Way.

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

5.4.2 Local File ETD Node Functions

The following list explains the nodes in the local file ETD, including primary functions:

- **Configuration**: The field sub-nodes within this node corresponds to an e*Way Connection configuration parameter and contains the corresponding settings information. See **"LocalFileETD: Configuration Parameters" on page 59** for details on these parameters and settings.

Note: *This ETD has configuration parameters that can be regular expressions. See* **"Using Regular Expressions" on page 112** *for details.*

- **Client**: The following sub-nodes, contained in this node, implement the e*Way's client interface in the ETD (**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 112** *and* **"Using Special Characters" on page 115** *for more information on these features.*

  - **InputStreamAdapter** and **OutputStreamAdapter**: Allow you to use and control the ETD's data-streaming features; see **"Streaming Data Between Components" on page 329** 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 103** 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 Local File ETD Methods" on page 106**.
  - **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 ETD.*

5.4.3 Using the Local File ETD

This section explains how to use the local file ETD's features.

Note: *There is no particular order for the calls that can be made on the local file ETD. 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 ETD

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

- **Guaranteed Exactly Once Delivery (GEOD)**: Allows your system to perform file read/write operations using an XA mode, ensuring the file's data integrity when file read/write operations are required in a Collaboration.
- **Data Streaming**: Allows your system to stream data directly to and from a local file system when used together with the FTP ETD or the record-processing ETD. 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.

  This feature is also available in the XA mode. In that case Resume Reading allows for reliably processing large files without overloading the e*Gate system with large amounts of data during a single XA-mode transaction.

Note: *For more information on the e*Way's XA-compliant GEOD features, see* **"Guaranteed Exactly Once Delivery" on page 347***. For more information on the Resume Reading feature, see* **"Resume Reading Feature" on page 106***.*

Pre/post File Transfer Commands

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

These features are available with both the local file ETD and the FTP ETD.

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, the system generates an error that results in exception's being thrown in the called ETD 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

These commands allow you to execute a desired action directly after the actual file transfer or during the "commit" phase when the **Transaction Type** mode is set to **XA-Compliant** (see **"Guaranteed Exactly Once Delivery" on page 347**).

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.

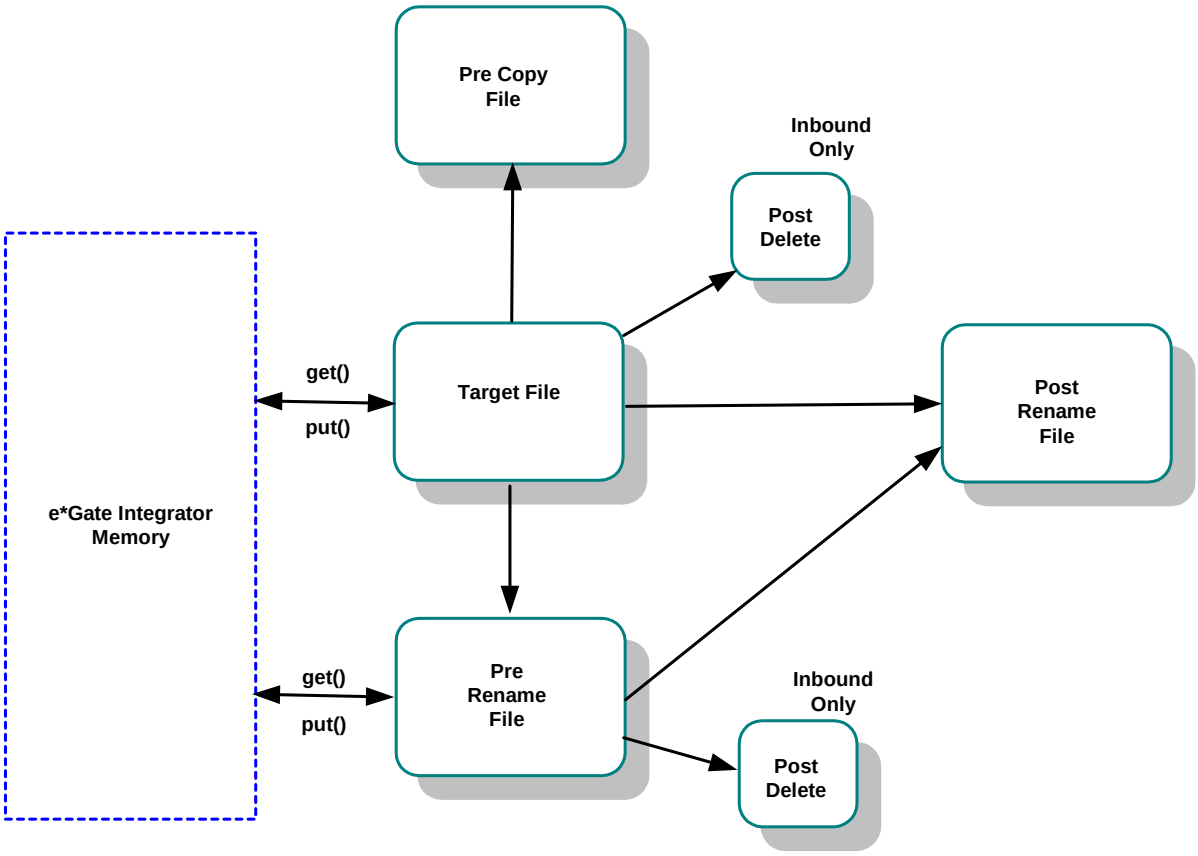
Your post-transfer options are:

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

Figure 7 Pre- and Post-transfer Processes



For information on the e*Way Connection configuration parameters for these commands, see:

FTP ETD

- **"Pre Transfer Configuration" on page 41**
- **"Post Transfer Configuration" on page 44**

Local File ETD

- **"Pre Transfer Configuration" on page 62**
- **"Post Transfer Configuration" on page 64**

Essential Local File ETD Methods

In addition to the field elements shown in **Figure 6 on page 101**, the local file ETD's **Client** node contains methods that extend the client interface functionality of the e*Way. These methods are essential to the proper use of the ETD 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 FTP and local file ETDs. It must be called when the ETD 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 ETD.*

See **Chapter 10** *and the Javadoc for more information on these methods.*

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.

This feature is also available in the XA mode. In that case Resume Reading allows for reliably processing large files without overloading the e*Gate system with large amounts of data during a single XA-mode transaction.See **"Resume Reading Enabled" on page 60** for a description of the e*Way Connection configuration.

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.

Note: *One of the e*Way's implementation sample schemas contains a Collaboration Rule that uses the Resume Reading feature. See* **"Creating Collaboration Rules" on page 202** *for details.*

The **Client** node in the ETD has a read-only property (**ResumeReadingInProgress** node; see **Figure 6 on page 101**) 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.

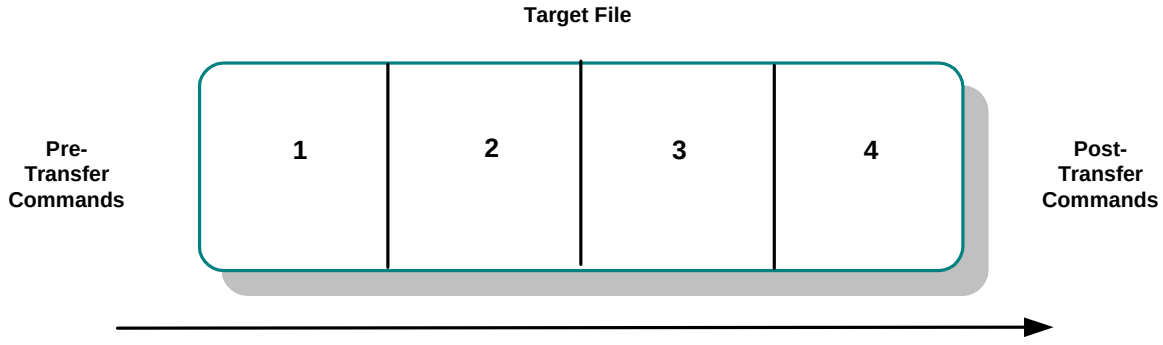
The feature has no special operational requirements besides setting the e*Way Connection configuration option. The e*Way Connection configuration has an option to enable or disable this feature. This option is also accessible at run time.

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

Step-by-step Operation

Figure 8 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 8 Resume Reading Operation



Because the file in **Figure 8 on page 107** 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 e*Way starts reading the file then reaches a break condition after a partial data read (the end of **Part 1**), the e*Way's pre-transfer commands have already been executed. The resume-reading state is stored, and no post-transfer commands are executed. The e*Way 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 e*Way 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 e*Way 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 e*Way 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 e*Way 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 **LocalFileETD.Configuration.ResumeReading** node to **False** before calling **finish()**. This setting tells the local file ETD 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 local file ETD to implement the e*Way's data streaming feature. This feature is also available with the FTP and record-processing ETDs. However, the local file ETD is a data stream-adapter provider, while the other two ETDs are only consumers.

See **"Streaming Data Between Components" on page 329** for details on how to use the ETD's data streaming feature.

Sequence Numbering

This feature in this ETD operates in the same way as sequence numbering for the FTP ETD. See **"Sequence Numbering" on page 94** for details.

Handling Type Conversions

This feature in this ETD operates in the same way as type conversion for the FTP ETD. See **"Handling Type Conversions" on page 92** for details.

5.4.4 Recommended Practice

It is recommended that Collaboration Rules use the record-processing ETD together with the local file ETD to parse records or construct payloads. This usage is a better practice than the use of only the FTP ETD.

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 329** and **"Resume Reading Feature" on page 106**).

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 ETD 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 local file ETD, 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 FTP ETD.

Note: *This practice is especially recommended for use with GEOD transactions in the XA mode. See* **"Guaranteed Exactly Once Delivery" on page 347** *for more information on the e*Way's XA-related features.*

5.4.5 ETD Limitations

The local file ETD 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 e*Way itself does not perform any mapping or mounting.

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

*drive**directory**file_name*

5.5 FTP File ETD

The Batch e*Way contains an ETD for FTP operations called the FTP file ETD. The current Java version of this e*Way supports this ETD as a backward-compatibility feature for a previous version of the e*Way.

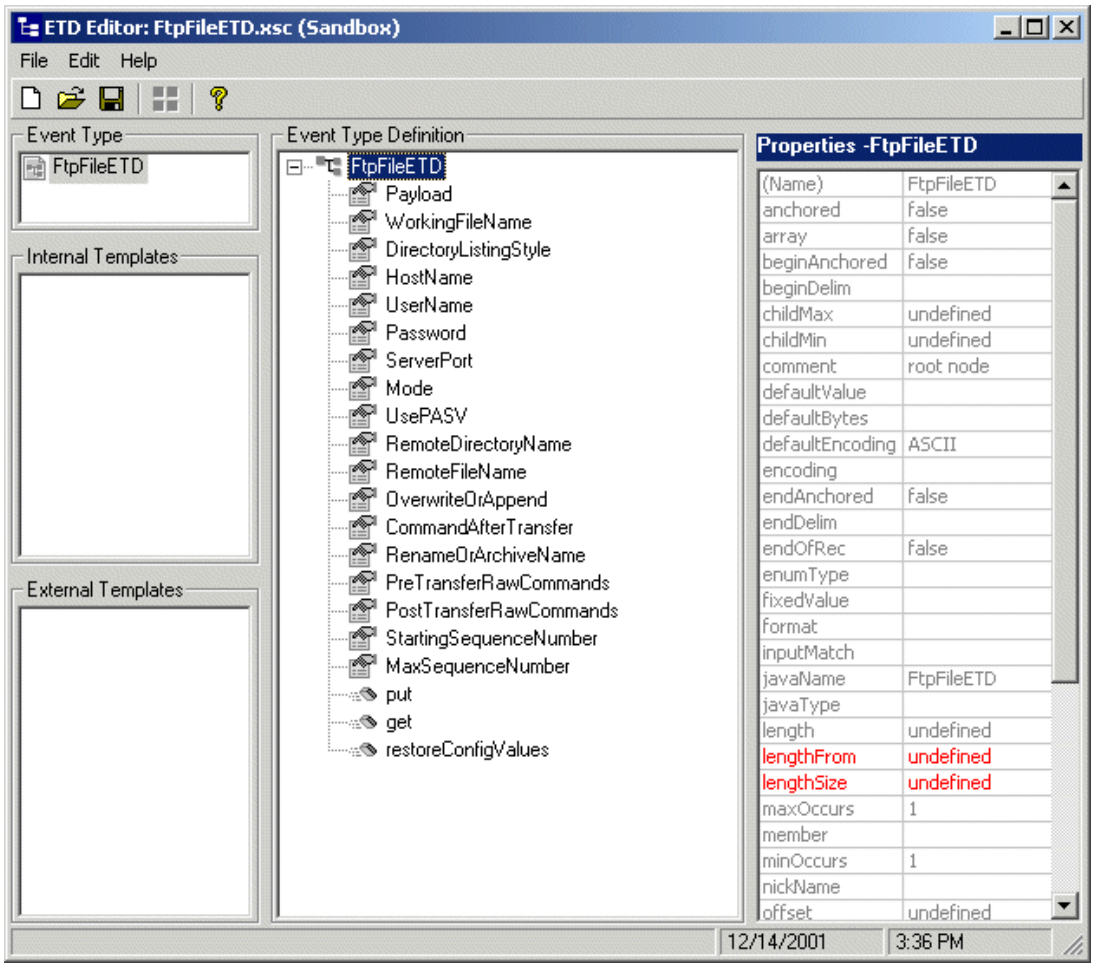
Although this FTP file ETD (**FtpFileETD.xsc**) is still supported in the current version of this e*Way, be careful not to confuse this ETD with the more versatile FTP ETD (**FtpETD.xsc**).

Note: *This ETD and its corresponding Java implementation is provided for backward compatibility because its functionality has been supplanted by the newer and more functional* ***FtpETD.xsc****. It is recommended that you use the* ***FtpETD.xsc*** *ETD for all new development.*

5.5.1 ETD Structure

Figure 9 shows the FTP file ETD as it appears in the ETD Editor's Main window.

Figure 9 FTP File ETD Structure



Note that each field element in the ETD structure corresponds to one of the e*Way Connection's configuration parameters. See **Chapter 4** for an explanation of each of these parameters.

Caution: *You cannot use both the FTP file ETD and the FTP ETD in the same Collaboration.*

5.5.2 ETD Methods

In addition to the field elements shown in **Figure 9 on page 111**, the FTP file ETD contains the following methods:

- **get()**: Retrieves the payload from the FTP server, that is, it retrieves the first matching file based on **Remote Directory Name** and **Remote File Name** to the payload and performs **Command After Transfer**. It also returns a Boolean **true** if the data is retrieved successfully or **false** if no data is available.
- **put()**: Places the payload on the FTP server, that is, it performs an **append** or **put** from the payload to the remote FTP server and performs **Command After Transfer**. It also returns a Boolean **true** if the data is sent successfully or **false** if the operation fails.
- **restoreConfigValues()**: Restores all the values from the e*Way Connection to the appropriate values in the FTP file ETD.

See **Chapter 4** for more information on each of these methods.

Note: **Chapter 10** *and the Javadoc contain more information about the FTP file ETD's methods.*

5.5.3 Handling Type Conversions

The **Payload** node in the **FtpETD.xsc** structure is predefined as a byte array (**byte[]**). This definition allows the e*Way to handle both binary and character data.

See **"Handling Type Conversions" on page 92** for details on this feature.

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

5.5.4 Encrypting Passwords

The FTP file ETD has the method **setPassword()** that accepts the encrypted password as its input. If you want to encrypt the password yourself, you can use the class **com.stc.common.utils.ScEncrypt**.

The method **scEncrypt.encrypt(user, password)** returns the encrypted password.

5.6 Using Regular Expressions

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

5.6.1 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 local file and FTP ETD'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 e*Way, see the following Web site:

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

Regular expressions operate with the local file and FTP ETDs 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 115**.

You can specify an extension, for example, .*\.**dat$**. Then, each time the **get()** method is called, the e*Way gets the next file with a .**dat** extension. The e*Way then retrieves each file into the ETD'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 e*Way

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 e*Way Connection 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.*

5.6.2 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 e*Way. 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 \\ (but, as noted previously, you should use the / character and not \ anyway).

- 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 \\\\.

5.7 Using Special Characters

The Batch e*Way 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 e*Way 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 112**.

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.

5.7.1 Types of Name Expansion

The e*Way 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 6 on page 117** 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 103**), 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.

5.7.2 Resolving Names

Typically, the pre/post names with patterns are resolved during **get()** and **put()** method calls. But sometimes, in using Collaboration Rules, the e*Way 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 FTP ETD, for example:

getResolvedNamesForPut().getTargetFileName()

The previous code yields **file1** based on the pattern **file%#**. In this usage, the ETD nodes can be used to make the desired method call.

Note: *See* **"FTP ETD Node Functions" on page 91** *for more information on FTP ETD nodes.*

5.7.3 Date/time Format Syntax

The e*Way 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 e*Way 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 6 for a complete list.

Table 6 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

Table 6 Time Pattern Strings and Meanings (Continued)

Symbol	Meaning	Presentation	Example
%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 7 shows some examples using the U.S. locale.

Table 7 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 6

Extending the e*Way

This chapter explains how to customize the Batch e*Way Intelligent Adapter's functionality by extending its capabilities.

6.1 Extending e*Way Functionality: Overview

The Batch e*Way has been designed to handle the vast majority of your needs as delivered. However, you may be required to customize one or more functions for your individual requirements. In such cases, you have many options available to easily extend the e*Way's functionality.

6.1.1 Designed With Extensibility In Mind

The e*Way has been designed from the ground up with user extensibility in mind and provides a variety of extensibility features. You can use these features to customize functionality to perform a multitude of specialized tasks. For example, you can override how file operations are performed, how records are processed, and how records are created.

To extend the e*Way's functionality, you can create your own implementations of its Java classes. Also, you can create your own Java properties files. If you want to use these extensions, you must enter your class or file name in the appropriate e*Way Connection configuration.

The use of these features is optional, and Java programming is required to employ them. For this reason, to extend the e*Way's functionality, you need a more advanced programming knowledge than you would if you only used the product "out of the box." However, these extensibility features are readily available in those cases where you need to use them.

6.1.2 Specifying User Classes and Properties Files

Use the e*Way Connection configuration to specify the user classes and Java properties files you are using to extend your e*Way's functionality. If you use these features, you must enter them as configuration parameters in the e*Gate Schema Designer's e*Way Configuration Editor.

For details on how to set these parameters, see the following sections:

Record-processing ETD

- **"User Class" on page 34**
- **"User Properties" on page 35**

FTP ETD

- **"Provider Class Name" on page 54**
- **"Client Class Name" on page 54**
- **"User Properties File" on page 55**

6.1.3 Interface-based e*Way Functionality

Most of the functionality exposed in the e*Way's Event Type Definitions (ETDs) is represented internally by Java interfaces. It is the *implementation* of those interfaces that determines how the ETDs operate.

In programming terms, an interface describes a *contract* and the semantics of how to use a Java *object*. Interfaces are used in many programming languages, including Java. In fact, Java uses them quite heavily. For example, the **java.io.InputStream** interface defines the contract and semantics for any client code that seeks to use the services of an **InputStream** object. This concept is used in the Batch e*Way as well.

The true power of these features is that *you can provide your own implementation for any of these interfaces and use it instead of the default implementation*.

Record-processing ETD

The record-processing ETD uses a *parser* interface to describe the functionality of record parsing and creation semantics. The implementation of that parser interface controls the ETD's functionality. In this case, there are three supplied implementations: delimited, fixed-size, and single-record.

FTP ETD

The FTP ETD uses two interfaces, a *provider* and a *client*. The provider interface represents the ETD's general FTP-related operations. The client interface represents how the functionality of the provider is actually used.

The rest of this chapter explains these interfaces and concepts in detail.

6.2 Extending the Record-processing ETD

This section explains how to extend the record-processing functionality in the e*Way.

6.2.1 Parser Interface Operation

The record-processing ETD uses an interface named **BatchRecordParser** (the *parser interface*). This interface defines the functionality for the record-processing ETD, including parsing and creating a payload of records. Records, in this context, are a section of a file. If you extend the capabilities of the e*Way by implementing your own parser, the way these records (or a section) are processed is up to you.

In terms of operation, the **get()** and **put()** methods in the ETD route to the **get()** and **put()** methods of the e*Way's inner interface implementation. This interface is defined in the **com.stc.eways.batchext** package (see Chapter 10 and the Javadoc for more information).

The e*Way has the following "out of the box" parser implementations:

- Delimited records
- Fixed-size records
- Single records

Keep in mind that, though this interface is called a parser, it can either parse or create a payload. The actual parser implementation used is determined by the settings in the e*Way Connection configuration for the ETD. Based on the record type chosen, an instance of the appropriate parser implementation is created and inserted into the ETD implementation code.

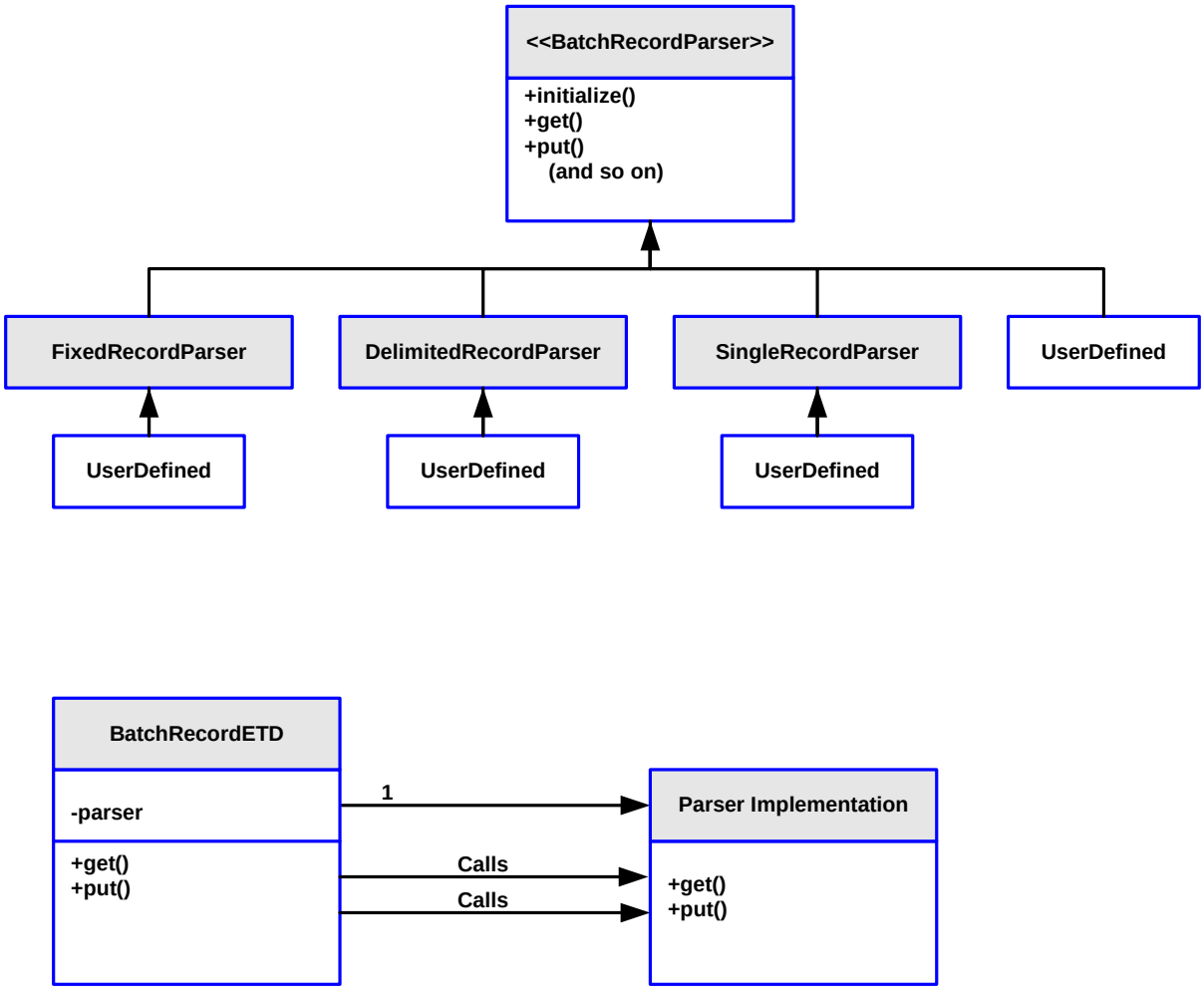
To use your own version of a record-parser implementation in the ETD

- Create your own user-defined class, and optionally, a Java properties file.
- In the e*Way Connection Configuration, go to the **User Class** section in the e*Way Connection configuration for the record-processing ETD.
  - Enter the class name of your own implementation.
  - If desired, you can also supply a full path to a Java properties file to be loaded and passed into the **initialize()** method of the parser immediately after construction (see "Using the initialize() Method" on page 123).

6.2.2 Record-parser Hierarchy

Figure 10 on page 122 shows a simplified Unified Modeling Language (UML) representation of the parser's operation as it relates to the record-processing ETD and its underlying parser interface structure. Unrelated methods are not shown, for clarity.

Figure 10 Diagram: Parser Operation



The top section of Figure 10 shows the parser interface and class hierarchy, and the bottom half shows how the **get()** and **put()** methods in the ETD route to the implementation of those methods in the parser implementation.

Note: *When creating a custom parser implementation, you can create one entirely from scratch, or you can extend one of the supplied implementations, only overriding the method or methods that you need to.*

6.2.3 Deriving From the Parser Interface

Code for the parser interface is in the **com.stc.eways.batchext** package. See the Javadoc section on this interface for an explanation of each of these methods in detail.

This section explains the three most important methods in this interface:

- **get()**
- **put()**
- **initialize()**

Using get() and put() Methods

The **get()** and **put()** methods represent the functionality exposed through the **get()** and **put()** methods in the ETD. They are, consequently, the methods most likely to require your attention when you are extending the e*Way's functionality. Your implementation of these methods determines the nature of a record in a payload.

Using the initialize() Method

When creating your own class, use the **initialize()** method to initialize the parser. This method is called internally by the e*Way immediately after an instance of the parser class has been created. It is called only once, and you can use it to initialize internal operations of your object. The method also allows you to pass user-specified properties into your object, if desired.

The **initialize()** method takes an instance of the **BatchRecordConfiguration** class. This class represents the properties as entered in the e*Way Connection configuration. If user properties were specified in this configuration, those properties are also accessible from this class. See the Javadoc for more information on this class.

6.2.4 Using Your Parser Implementation

After you have created your parser implementation, you must configure your e*Way Connection so that the e*Way knows to use your implementation class. This action is accomplished by:

- Selecting **User Defined** as the **Record Type** parameter under the **Record** section
- Then entering your user-defined class name under the **User Class** section as a parameter

For more information, see **"User Class" on page 34** and **"Record Type" on page 32**.

Actions at Run Time

At run time, the e*Way detects that a user-defined implementation has been requested and takes the following actions:

- Creates an instance of your class
- Loads the user properties file (if a file name was entered)
- Calls the **initialize()** method

From that point on, the other methods in your implementation are called as directed by what you do in the Collaboration. For example, if you make a call to **put()** in the Collaboration it routes to the **put()** method in your implementation.

6.3 Extending the FTP ETD

This section explains how to extend the FTP ETD functionality in the e*Way.

6.3.1 FTP Client and Provider Interfaces

In a sense, extending the FTP ETD is much like extending the record-processing ETD in that most of the functionality shown on the ETD actually routes to the e*Way's inner interface implementation. The FTP ETD has the following interfaces:

- Provider (**FtpFileProvider**)
- Client (**FtpFileClient**)

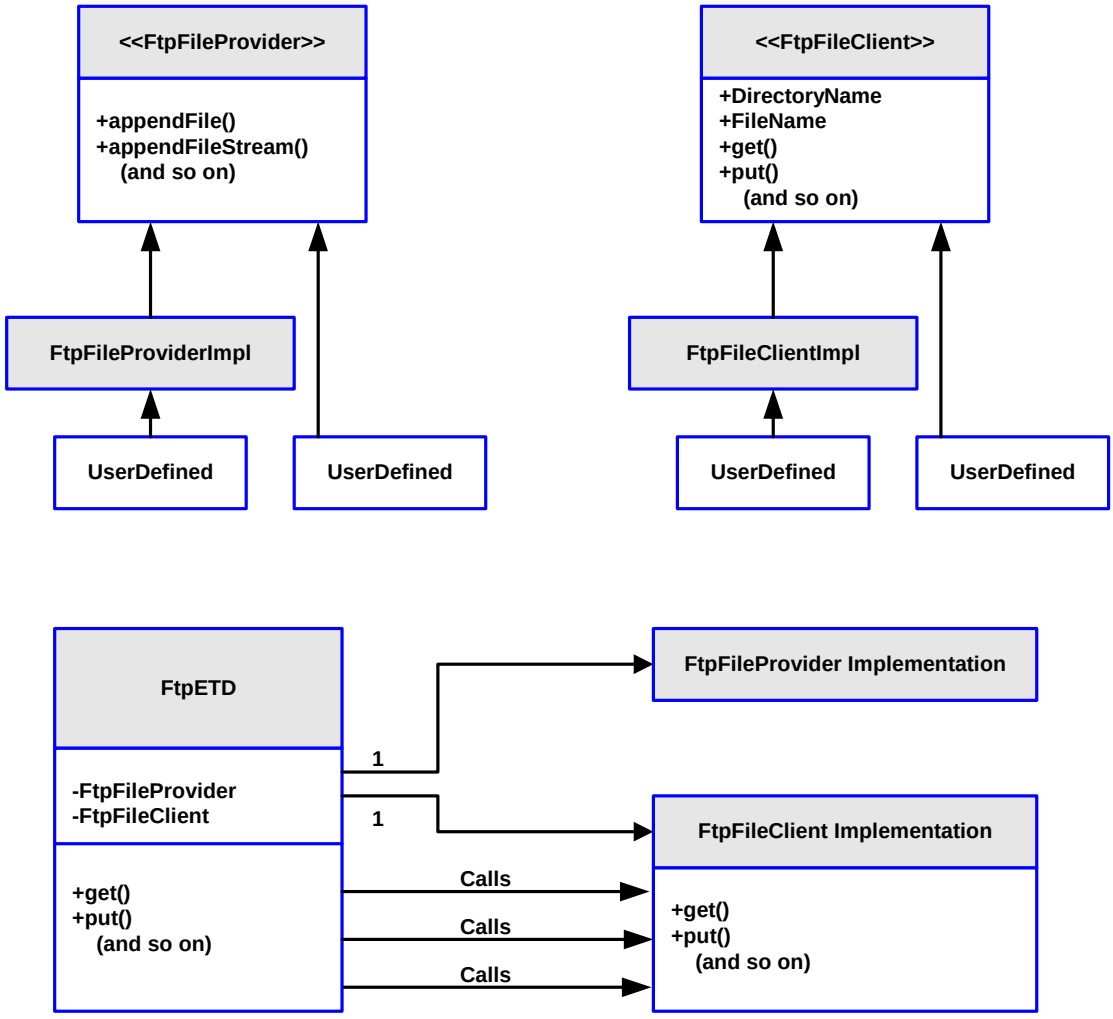
The *provider interface* represents the general FTP-related operations that can be performed in the ETD. That is, it represents the FTP services provided to users who want to utilize them. The *client interface* represents how the functionality of the provider interface is actually used.

You can create your own implementations or override the e*Way's default implementations for either or both of these interfaces.

6.3.2 FTP Client and Provider Hierarchies

Figure 11 on page 125 shows a simplified UML representation of how the client and provider interfaces operate in relation to the FTP ETD and its underlying client/provider interface structure. Unrelated methods are not shown, for clarity.

Figure 11 Diagram: FTP Client and Provider Operation



6.3.3 Deriving From Client and Provider Interfaces

Code for these interfaces is in the **com.stc.eways.batchext** package, and the Javadoc contains explanations for each method.

There are, however, a large number of methods on these interfaces, so it is advised that you derive your own implementations from the supplied implementation. In this case, you only need to override any method or methods when doing so is dictated by your own implementation. Using this approach is much easier than implementing the entire interface from scratch (although you can if desired).

6.3.4 Using Your Client and Provider Implementations

Once you have created your implementation, you need to configure your e*Way Connection so that the e*Way knows to use your implementation class. This action is accomplished by:

- Entering your user-defined provider class name under the **Extensions** section for the **Provider Class Name** parameter
- Entering your user-defined client class name under the **Extensions** section for the **Client Class Name** parameter

For more information, see **"Extensions Configuration" on page 54**.

Supplying User Properties to Your Implementation Class

You can, if desired, supply the fully qualified path name of a Java properties file under the **User Properties File** parameter under the ETD's **Extensions** configuration in the e*Way Connection. If you use a properties file, this file is loaded into a **java.util.Properties** object and is available in the **UserProperties** node in the ETD.

Sample Implementation

A sample e*Gate schema that illustrates these concepts can be found on the installation CD-ROM. The sample implements a custom version of the FTP ETD. See **"Sample Schema: FTP and ETD Extensibility" on page 235** for details on this sample.

Chapter 7

Implementation

This chapter provides information about a series of sample schemas. These will help you understand how to implement the Batch e*Way Intelligent Adapter in a production environment.

7.1 Implementation Overview

This section explains how to implement the Batch e*Way using e*Gate Integrator schema samples included on your installation CD-ROM. You can find these samples on the CD-ROM at the following path location:

\samples\ewbatch\Java

These samples allow you to observe end-to-end data-exchange scenarios involving e*Gate, the e*Way, and sample interfaces. This chapter explains how to implement these sample schemas that use the Batch e*Way.

You can also use the procedures given in this chapter to create your own schemas based on the samples provided. It is recommended that you use a combination of both methods, creating your own schema like each sample, then importing the samples into e*Gate to check your results.

Before Importing or Running a Sample Schema

To import and run a sample schema, the Batch e*Way must be installed, and you must also have access to a remote File Transfer Protocol (FTP) location.

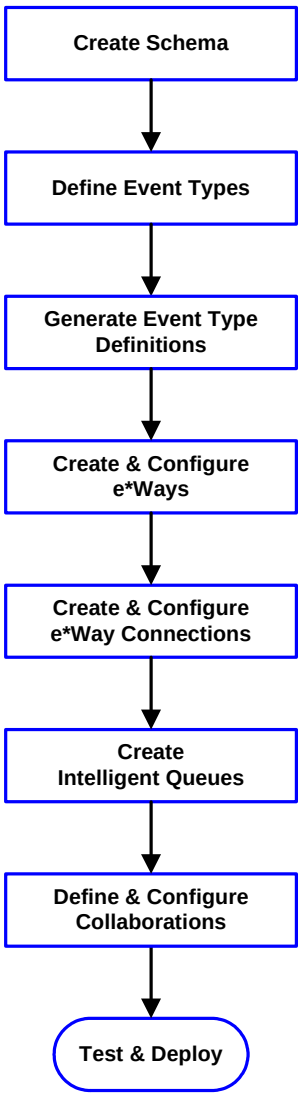
To import a sample schema

1. Copy the desired .**zip** file, for example, **BasicFtpSample.zip,** from the **samples\ewbatch\Java** directory in the install CD-ROM to your desktop or to a temporary directory, then unzip the file.
2. Start the e*Gate Schema Designer.
3. On the **Open Schema from Registry Host** dialog box, click **New**.
4. On the **New Schema** dialog box, click **Create from export**, and then click **Find**.
5. On the **Import from File** dialog box, browse to the directory that contains the sample schema.
6. Click the .**zip** file then click **Open**.

   The schema is installed.

To create the sample schema

Use the following implementation sequence:



1 The first step is to create a new schema. The rest of these steps apply only to this schema.

2 The second step is to create and define the Event Types you are transporting and processing within the schema.

3 You need to associate the Event Types created in the previous step with Event Type Definitions (ETDs) you want to use in the schema.

4 The next step is to create and configure the required e*Ways.

5 You must create and configure the e*Way Connections.

6 Now you need to create Intelligent Queues (IQs) and IQ Managers (if necessary) to hold published Events.

7 You need to create the desired Collaboration Rules for your schema, along with their associated Business Rules.

8 Next, you need to define and configure the necessary Collaborations.

9 Finally, you must check and test your schema. Once you have verified that it is working correctly, you can deploy it in your production environment.

Chapter Organization

Each sample spotlights key features of the e*Way. The first sample section describes a complete end-to-end e*Gate scenario showing how to build a schema using the Batch e*Way, from the beginning. Additional sample descriptions focus on creating and configuring the specialized features each sample seeks to illustrate.

List of Samples

The samples included with this e*Way are:

- **"Sample Schema: Basic FTP With Streaming" on page 129**: Provides a simple, easily implemented schema that illustrates basic FTP operations and data streaming with the e*Way.
- **"Sample Schema: Local File Streaming and GEOD" on page 170**: Shows you how to create and add the record-processing, local file data-streaming, and Guaranteed Exactly Once Delivery (GEOD) of Events features.
- **"Sample Schema: FTP and ETD Extensibility" on page 235**: Illustrates how to extend the FTP ETD, defining your own features.
- **"Sample Schema: Using Secure FTP" on page 270**: Explains how to add secure FTP to the Batch e*Way in a typical e*Gate schema.

7.2 Sample Schema: Basic FTP With Streaming

This section explains how to implement the basic FTP sample schema for the Batch e*Way. The schema demonstrates how to set up the essential features of the e*Way in a typical e*Gate environment, including basic FTP operations, data streaming, and data payload transfer.

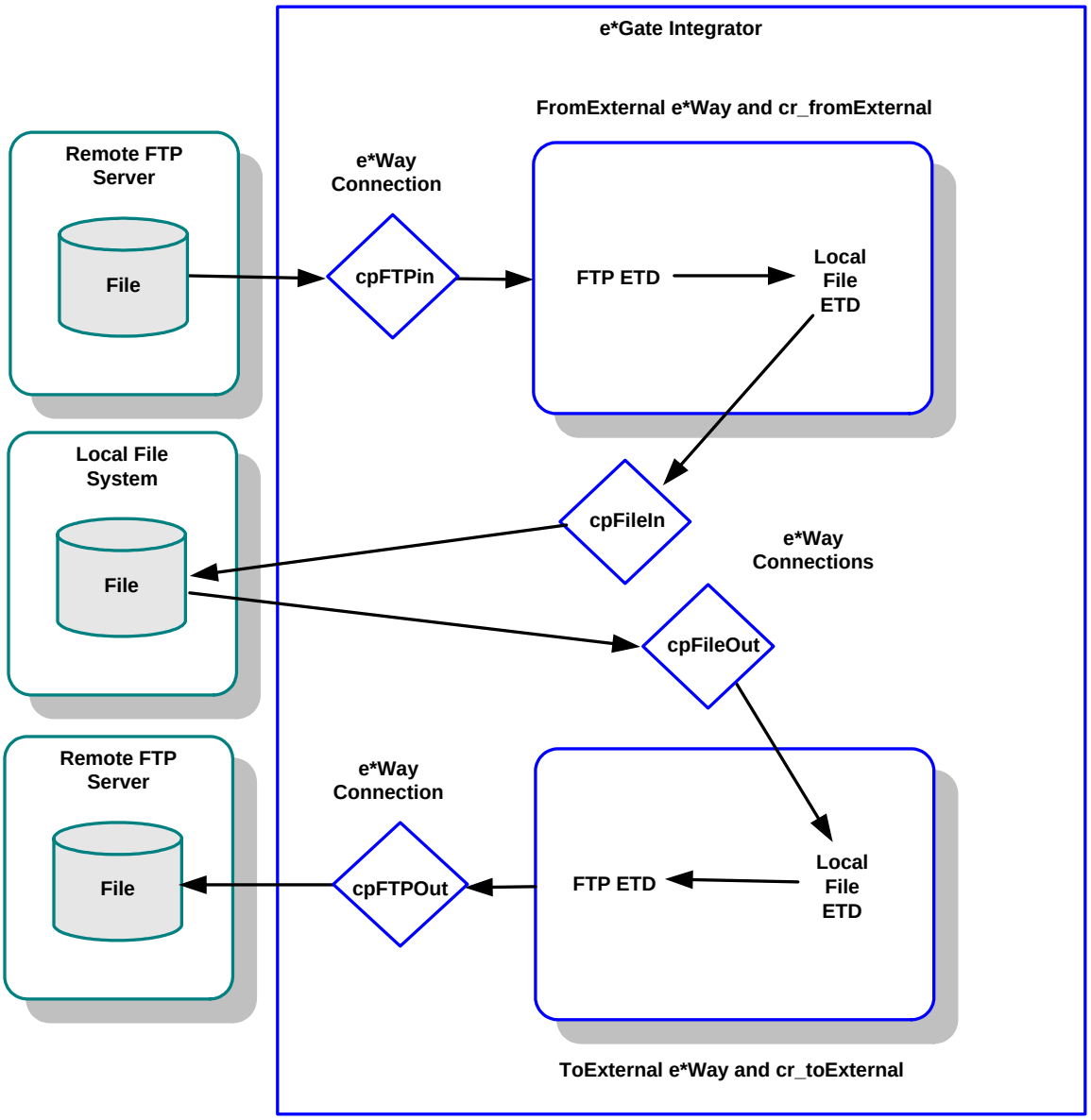
7.2.1 BasicFtpSample Schema Overview

This section provides a general overview of the basic FTP sample schema, its configuration, and how it operates. The name of this schema is BasicFtpSample, and it is contained in the import file **BasicFtpSample.zip**.

Schema Setup

Figure 12 on page 130 shows a diagram of the schema's general architecture. The arrows show the direction of data flow.

Figure 12 BasicFtpSample Schema Diagram



Schema Operation

This sample schema has the following input/output setup:

- **Input**: A file (**ftpSample.dat**) from a remote FTP server.
- **Output**: The same file (renamed to **ftpFileSample.dat**) to a remote FTP server.

This sample schema demonstrates the basic FTP **get()** and **put()** operations of the e*Way's FTP ETD, using both data streaming and payload transfer. The schema also uses the local file ETD for data streaming and local disk file transfers.

To run the sample schema

1. Start only the **FromExternal** e*Way.

   This e*Way returns a file, **ftpSample.dat**, via FTP from a remote FTP system. This file is then renamed on the remote system to **sample.finished**. The returned file is written to the local disk to **C:\eGate\data\ftpFileSample.dat** using data streaming.

2. Shut down the **FromExternal** e*Way.
3. Start the **ToExternal** e*Way.

   This e*Way picks up the local file, **ftpFileSample.dat**, written by the **FromExternal** e*Way and puts it on a remote FTP system via FTP using payload data transfer.

Schema Components

The BasicFtpSample schema with basic FTP implementation consists of the following main e*Gate components:

- **FromExternal**: Inbound Multi-Mode e*Way that brings the into e*Gate from a remote FTP system.
- **ToExternal**: Outbound Multi-Mode e*Way that sends the file to a remote FTP system.
- **collabfrmExt**: Collaboration for the **FromExternal** e*Way.
  - **cr_fromExternal**: Collaboration Rule for **collabfrmExt**.
- **collabToExt**: Collaboration for the **ToExternal** e*Way.
  - **cr_toExternal**: Collaboration Rule for **collabToExt**.
- **localhost_iqmgr**: Oracle **cpFTPout**: e*Way Connection (FTP ETD) for FTP from the **ToExternal** e*Way to the remote system.
- **cpFileOut**: e*Way Connection (local file ETD) for data-streaming the file from the **FromExternal** e*Way to the local file system.
- **cpFileIn**: e*Way Connection (local file ETD) for payload-transferring the file from the local file system to the **ToExternal** e*Way.

7.2.2 Creating the BasicFtpSample Sample Schema

This section explains the basic steps for how to create the sample schema BasicFtpSample.

Note: *For complete information on how to set up an e*Gate schema, see the* ***e*Gate Integrator User's Guide*** *and* ***Creating and End-to-end Scenario with e*Gate Integrator****.*

Creating a New Schema

The first task in deploying the schema sample is to create a new schema name. While it is possible to use the default schema for this implementation example, it is recommended that you create a separate schema for testing purposes.

To create a new schema

1. Start the e*Gate Schema Designer.
2. When the Schema Designer prompts you to log on, select the host that you specified during installation, and enter your password.
3. You are then be prompted to select a schema. Click **New**.
4. Enter a name for the new schema, **BasicFtpSample**.

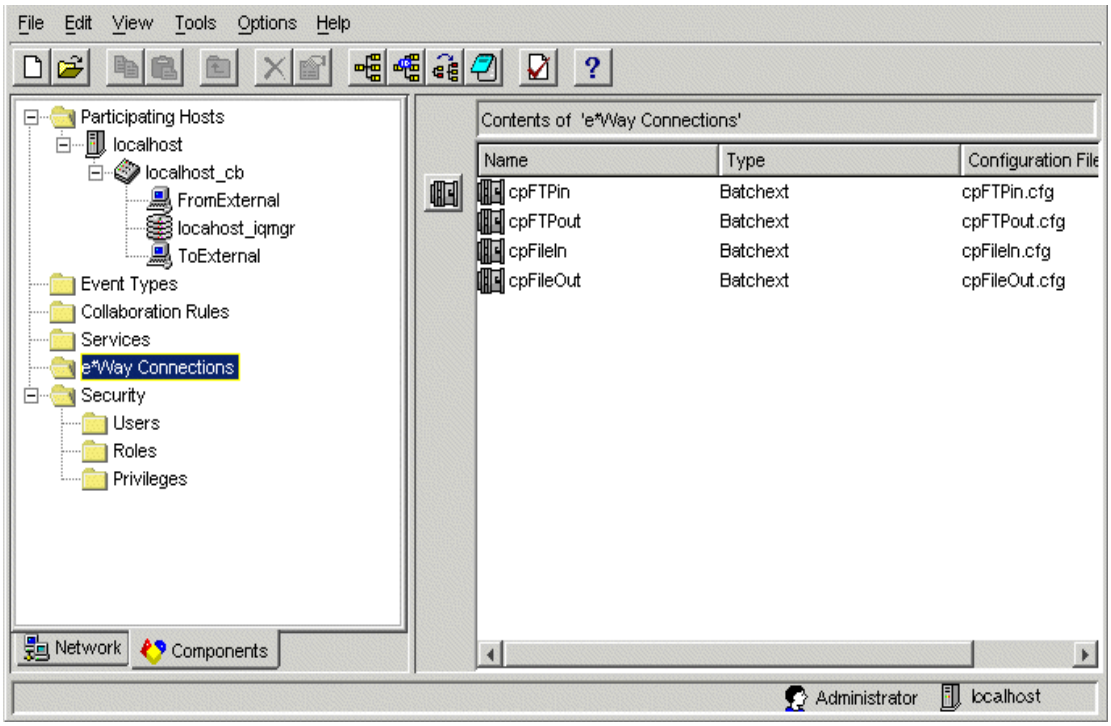
Note: *You can enter any name you want, but it is recommended that you use the same name as the* ***.zip*** *file, to avoid confusion among the different samples for this e*Way.*

5. Click **Open**. The Schema Designer displays a new, unconfigured schema.

The Schema Designer opens under your new schema, with many of the schema's basic components already created. From the Schema Designer, you can access the ETD Editor and Collaboration Rules Editor features. You are now ready to begin creating the necessary components for this sample schema.

Figure 13 on page 133 shows an example of the Schema Designer window with the BasicFtpSample schema already created.

Figure 13 Schema Designer Main Window for BasicFtpSample



Creating Event Types and ETDs

The e*Way installation includes two of the **.xsc** files for the Batch e*Way.

Creating Event Types

Using the Schema Designer, you create the following Event Types:

- **etFileETD** (local file ETD)
- **etFtpETD** (FTP ETD)

Using the ETDs

In this sample schema, you use the following ETDs:

- **LocalFileETD.xsc**
- **FtpETD.xsc**

To create the Event Types and ETDs

1. Highlight the **Event Type** folder on the **Components** tab of the e*Gate Schema Designer.
2. On the palette, click the icon to create a new Event Type.
3. Enter the name of the Event Type (**etFileETD**), then click **OK**.
4. Select the new **Event Type**, then right-click to edit its properties.

5 The **Event Type Properties** dialog box appears (see Figure 14).

Figure 14 etFileETD Event Type Properties Dialog Box

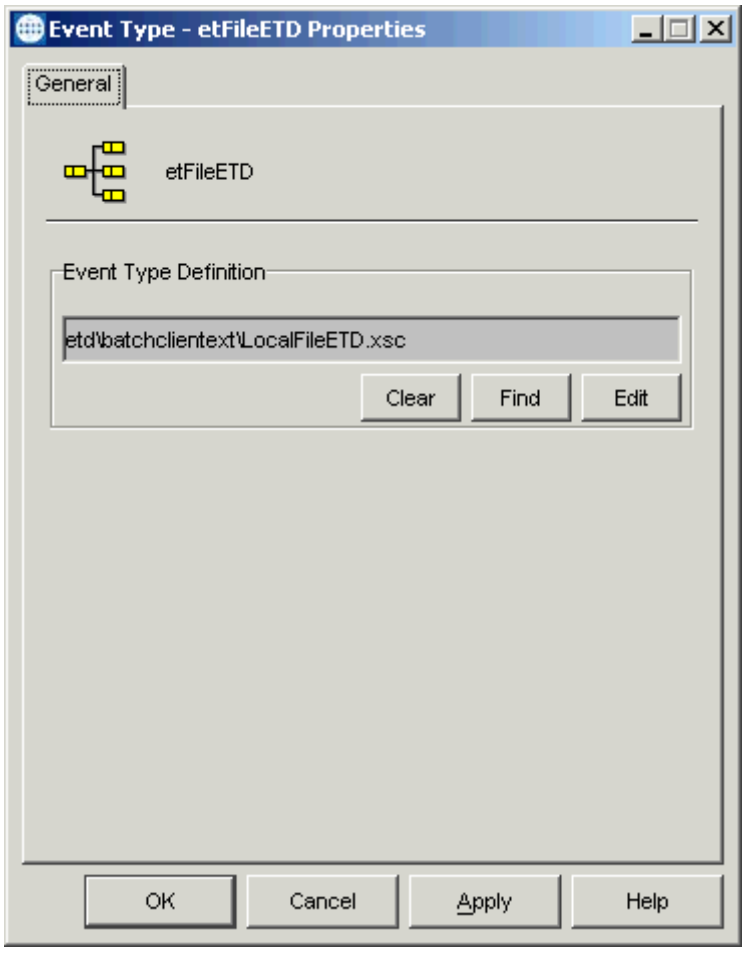


Figure 14 shows the ETD already selected. To select the desired ETD, go to the next step.

6 Click **Find**. The **Event Type Definition Selection** dialog box appears.

7 Navigate to and open the **client\etd\batchclientext** directory then select the **LocalFileETD.xsc** file.

8 Click **Select**. The **LocalFileETD.xsc** file name appears in the Event Type Definition text box, as shown in Figure 14.

9 Click **OK** to close the **Event Type Properties** dialog box and save your changes.

10 To create the next Event Type and select its associated ETD, repeat steps 2 through 9. Use the name **etFtpETD** for this Event Type and find the **FtpETD.xsc** ETD file to associate with this Event Type.

11 When you are finished with the dialog box, click **OK** to close it and save your changes.

Creating and Configuring e*Ways

You must create the following Multi-Mode e*Ways:

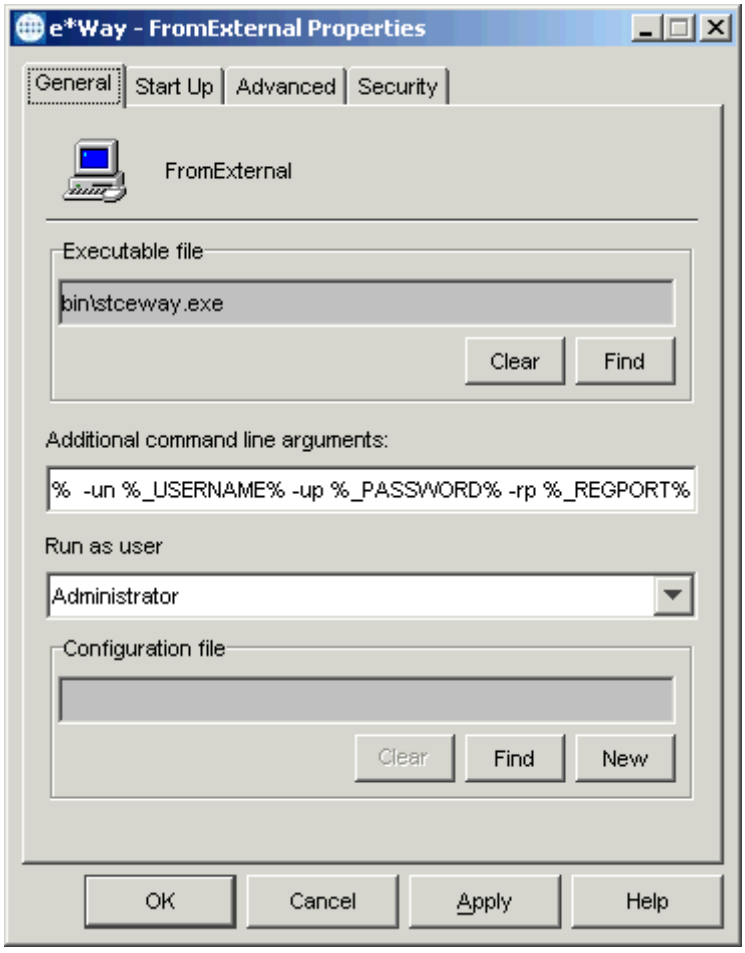
- Inbound: **FromExternal**
- Outbound: **ToExternal**

Note: *For detailed information on the Multi-Mode e*Way, see the* ***Standard e*Way Intelligent Adapter User's Guide****.*

To create the inbound and outbound Multi-Mode e*Ways

1. Select the e*Gate Schema Designer's **Components** tab.
2. Open the host on which you want to create the e*Way.
3. Select the Control Broker that manages the new e*Way.
4. On the palette, click the icon to create a new e*Way.
5. Enter the name of the new e*Way (**FromExternal**), then click **OK**.
6. Select the new component, then double-click to edit its properties.
7. When the **e*Way Properties** dialog box appears, use the default executable file, **stceway.exe** (see **Figure 15 on page 136**).

Figure 15 FromExternal e*Way Properties Dialog Box



Configure the **FromExternal** e*Way properties as shown in the previous figure.

8 To edit the JVM Settings, select **New** under the **Configuration File** text box. Use the default configuration parameters, as shown in the e*Way Configuration Editor.

Note: *See* **Chapter 3** *for more information on how to configure the Multi-Mode e*Way.*

9 Save the **.cfg** file, and exit the e*Way Configuration Editor, returning to the **e*Way Properties** dialog box..

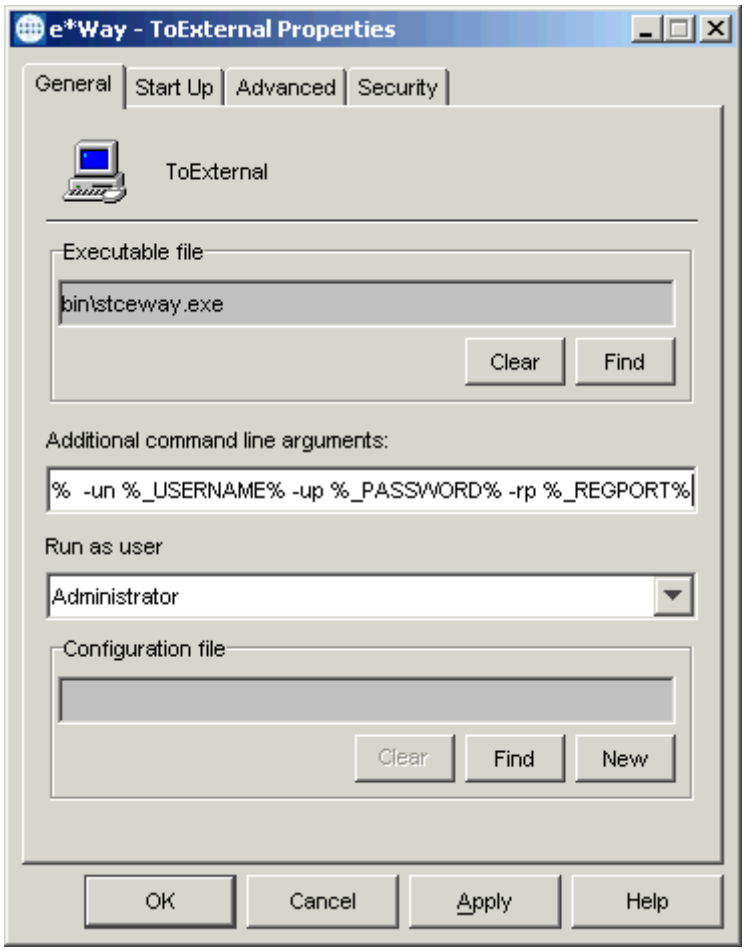
10 Use the **Startup**, **Advanced**, and **Security** tabs to modify the default settings for each.

- A Use the **Startup** tab to specify whether the Multi-Mode e*Way starts automatically, restarts after abnormal termination or due to, for example, scheduling.
- B Use the **Advanced** tab to specify or view the activity and error logging levels, as well as the Event threshold information.
- C Use **Security** to view or set privilege assignments.

11 Select **OK** to close the **e*Way Properties** dialog box and save your settings.

12 Repeat steps 4 through 11 for the **ToExternal** e*Way (see Figure 16).

Figure 16 ToExternal e*Way Properties Dialog Box



Configure the **ToExternal** e*Way properties as shown in the previous figure. Use the default configuration parameters.

Creating and Configuring e*Way Connections

The e*Way Connection configuration file contains the connection information needed to communicate with the local file system and the remote FTP server.

To create and configure the cpFTPin e*Way Connection

1 Highlight the **e*Way Connection** folder on the **Components** tab of the e*Gate Schema Designer.

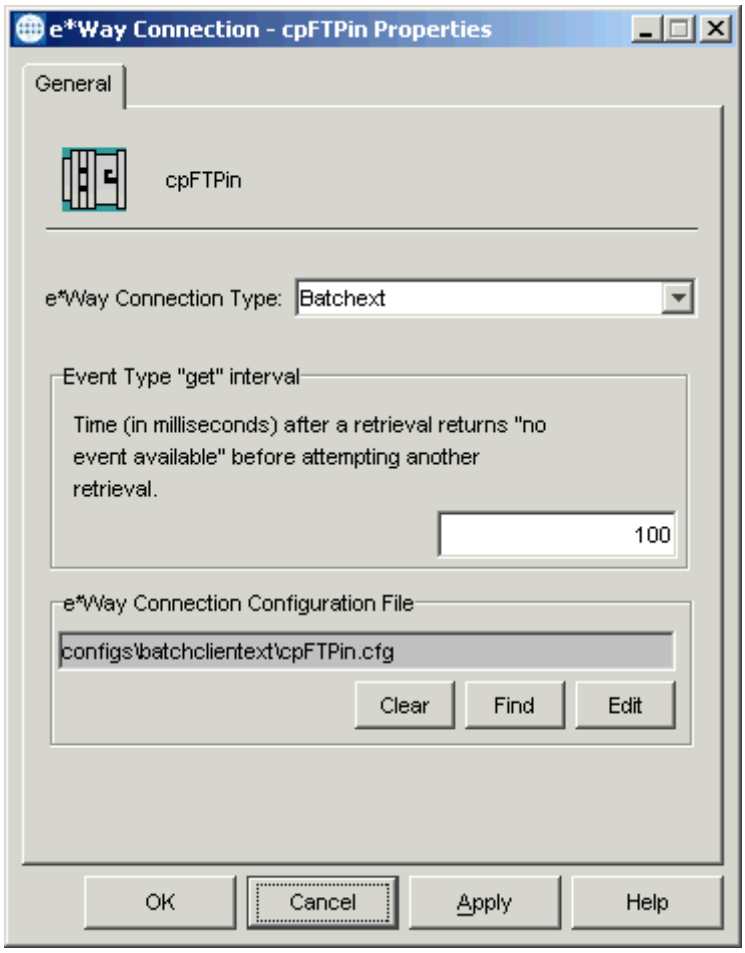
2 On the palette, click the icon to create a new e*Way Connection.

3 Enter the name of the e*Way Connection (**cpFTPin**), then click **OK**.

4 Select the new **e*Way Connection**, then right-click to edit its properties.

5 When the **e*Way Connection Properties** dialog box opens, select **Batchext** from the **e*Way Connection Type** drop-down menu (see Figure 17).

Figure 17 cpFTPin e*Way Connection Properties Dialog Box

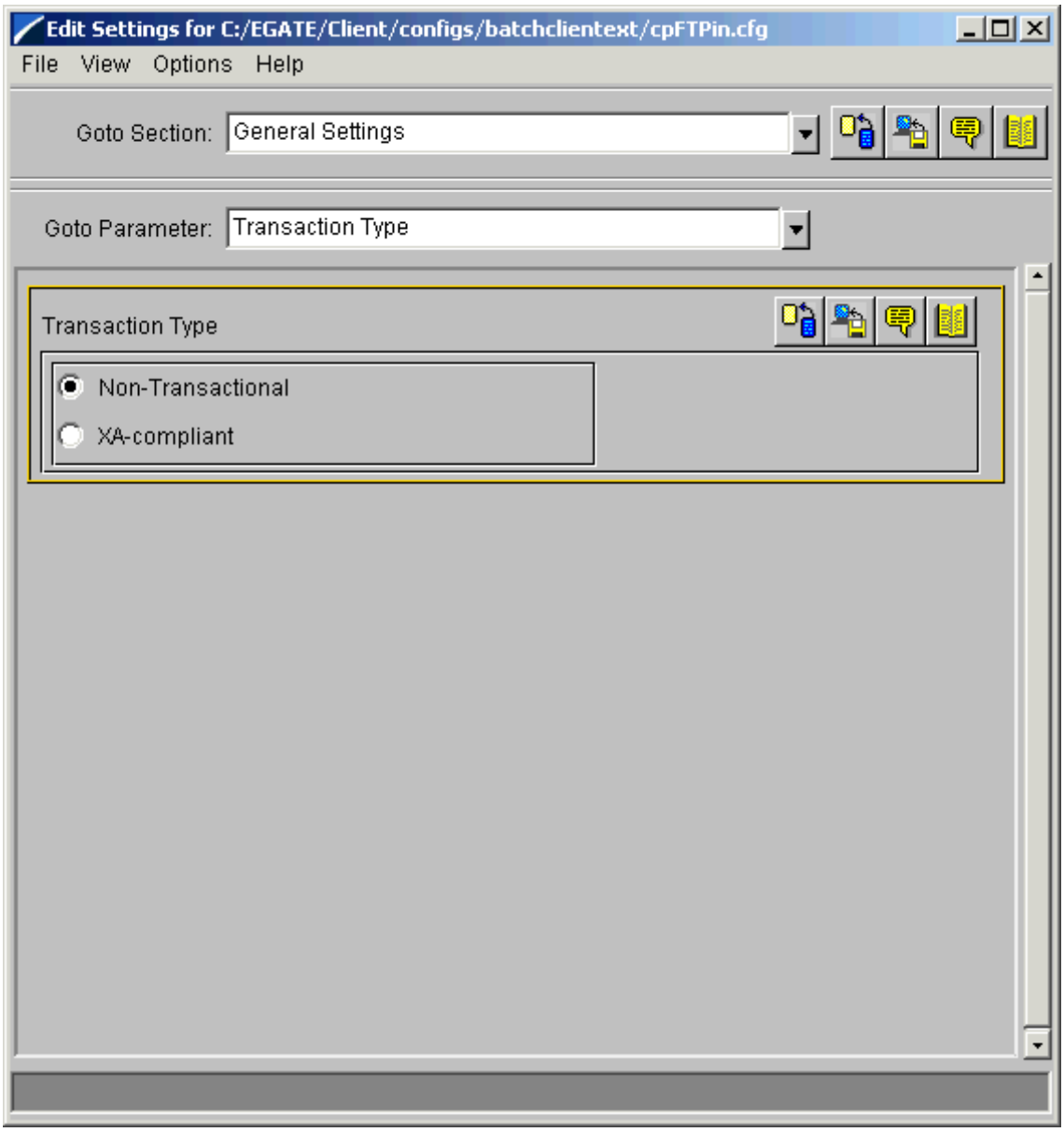


Configure the e*Way Connection properties as shown in the previous figure.

6 Under **e*Way Connection Configuration File**, click **New** (where the **Edit** button is in the previous figure). Select the **FtpETD**.

The e*Way Configuration Editor Main window opens (see **Figure 18 on page 139**).

Figure 18 e*Way Configuration Editor: cpFTPin General Settings



7 Select the desired parameters, including those that correspond to the remote FTP system you are using. See **"FtpETD: Configuration Parameters" on page 36** for details.

Note: *See the* ***e*Gate Integrator User's Guide*** *for complete information on how to use the e*Way Configuration Editor.*

8 Using the e*Way Configuration Editor accept the default settings for all parameters except for:

- **FTP:**
  - **Directory Listing Style**
  - **Host Name**
  - **User Name**
  - **Password**
- **Target Location:**
  - **Target Directory Name**

You must enter your system settings for these parameters (see **Figure 19 on page 141** through **Figure 21 on page 143**).

Figure 19 e*Way Configuration Editor: Necessary cpFTPin Settings (FTP First Set)

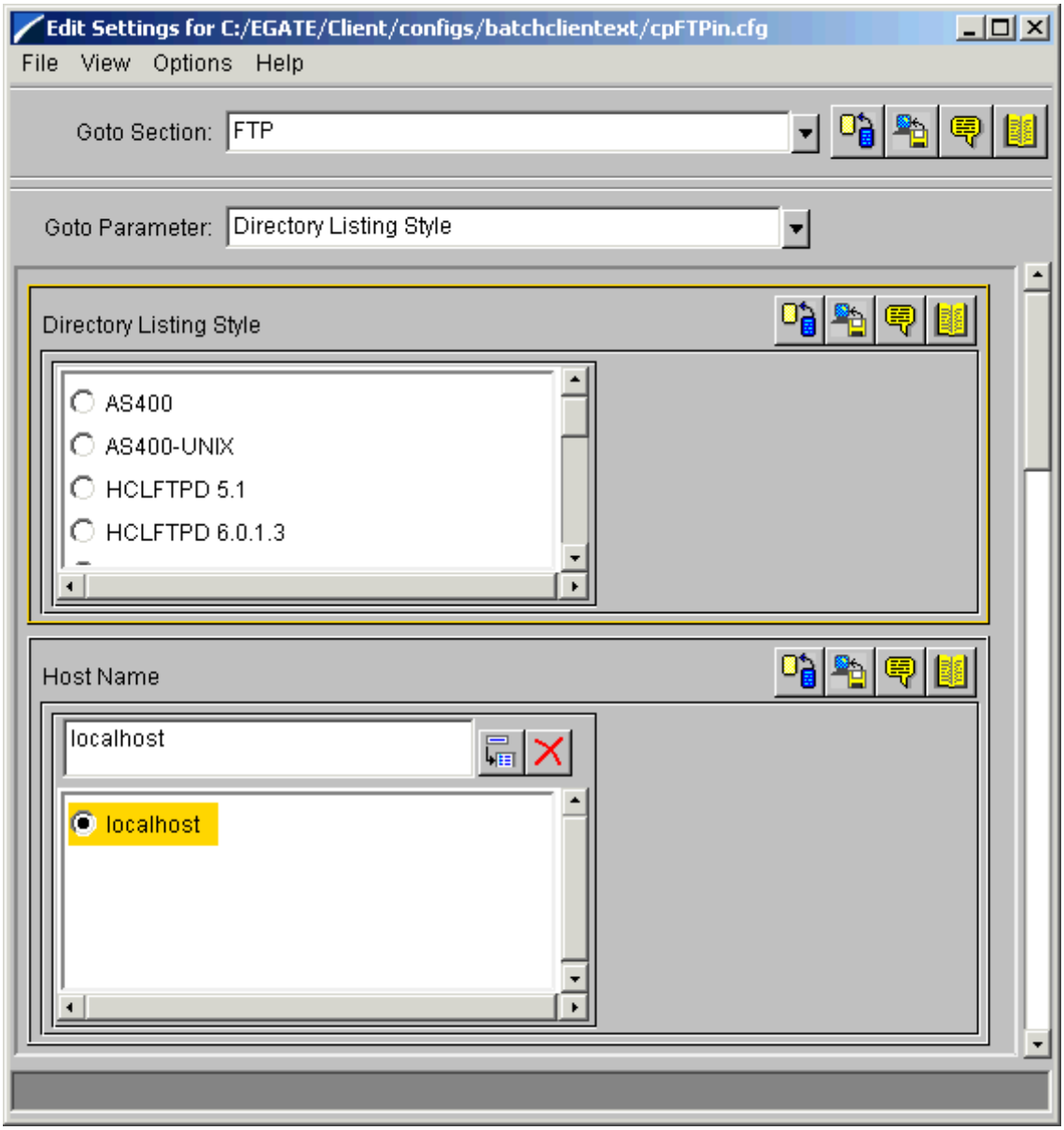


Figure 20 e*Way Configuration Editor: Necessary cpFTPin Settings (FTP Second Set)

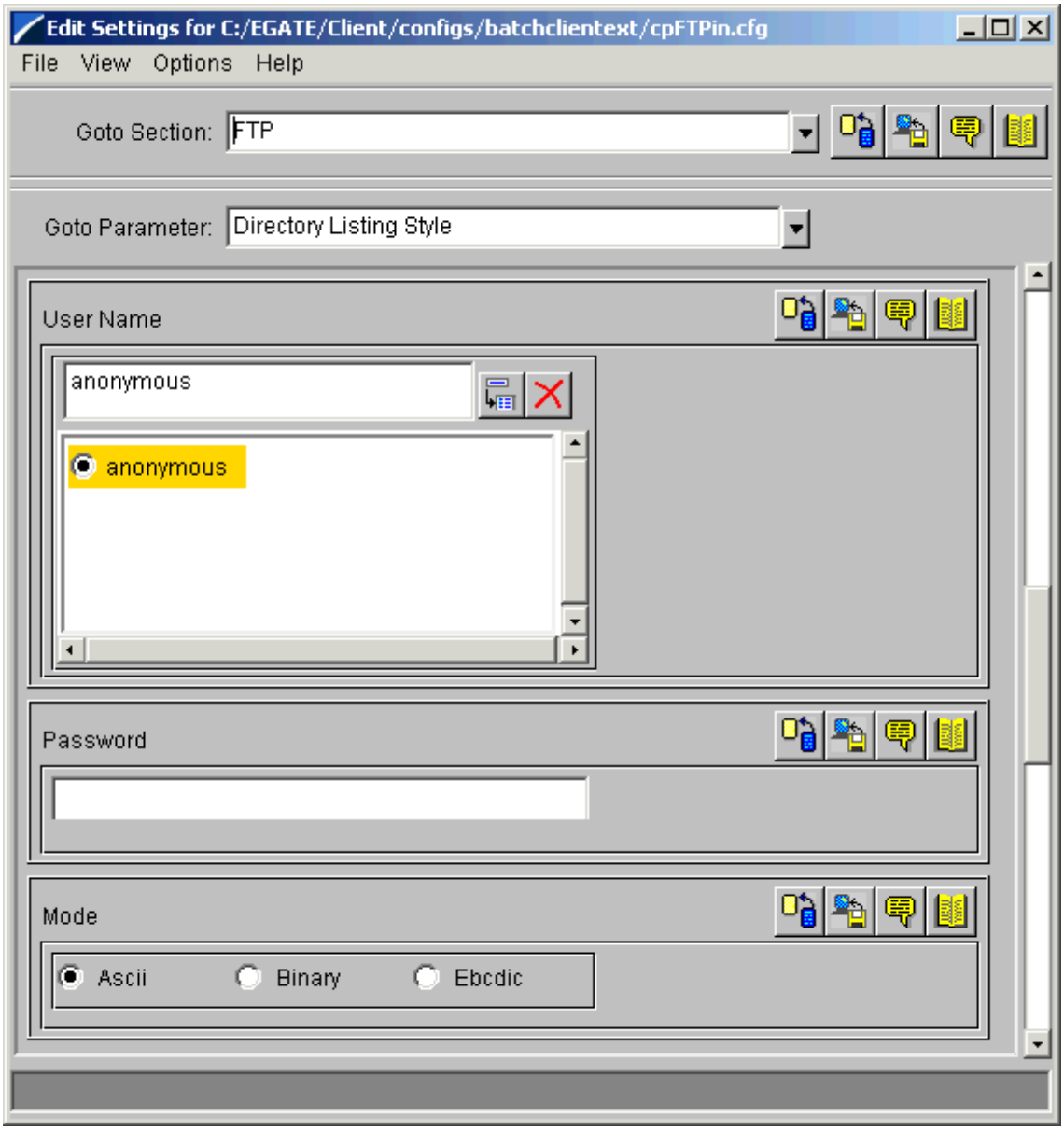
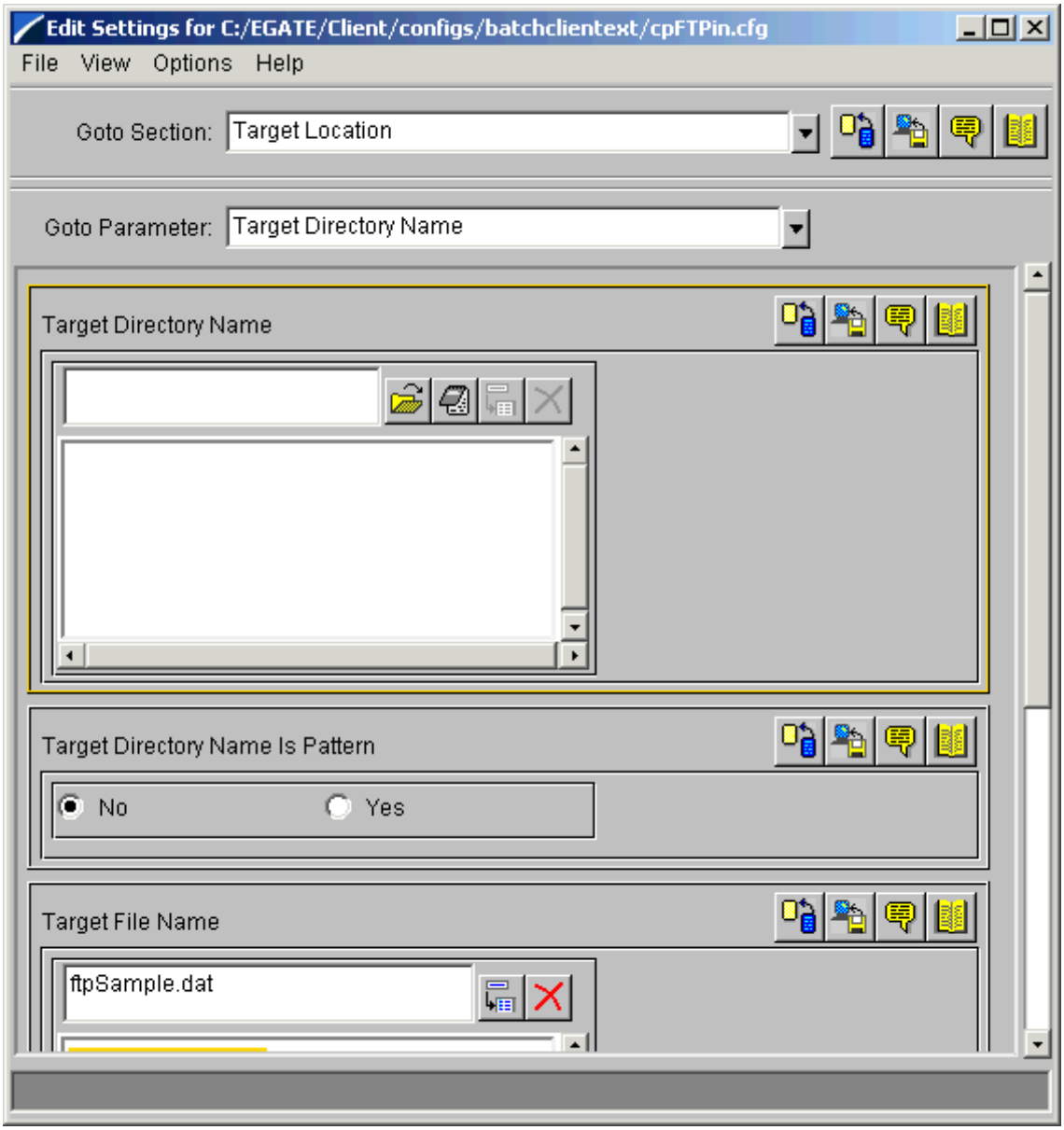


Figure 21 e*Way Configuration Editor: Necessary cpFTPin Settings (Target)



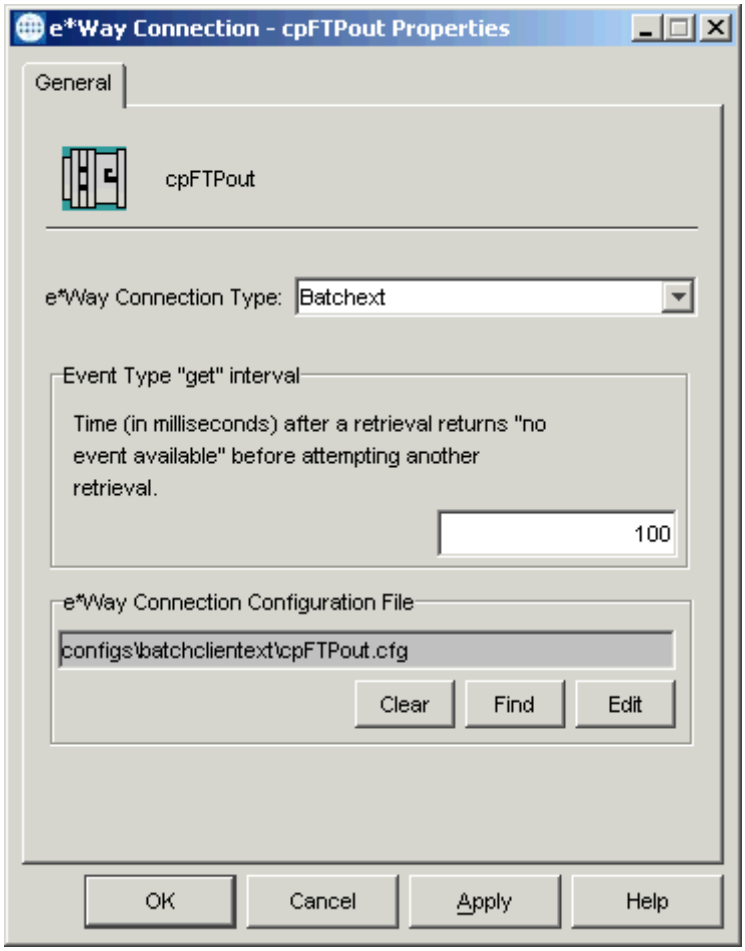
9 When you are finished, save the .**cfg** file as the name of the e*Way Connection, close the e*Way Configuration Editor, and promote the file to run time.

10 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the cpFTPout e*Way Connection

1 Repeat steps 2 through 6 under the **procedure on page 137** for the **cpFTPout** e*Way Connection (see **Figure 22 on page 144**).

Figure 22 cpFTPout e*Way Connection Properties Dialog Box



2 Using the e*Way Configuration Editor, set the parameters for the remote FTP system. See **"FtpETD: Configuration Parameters" on page 36** for details.

   Be sure to provide the appropriate system settings for the same **FTP** and **Target Location** parameters as you did for the **cpFTPin** e*Way Connection. (see the **procedure on page 137**).

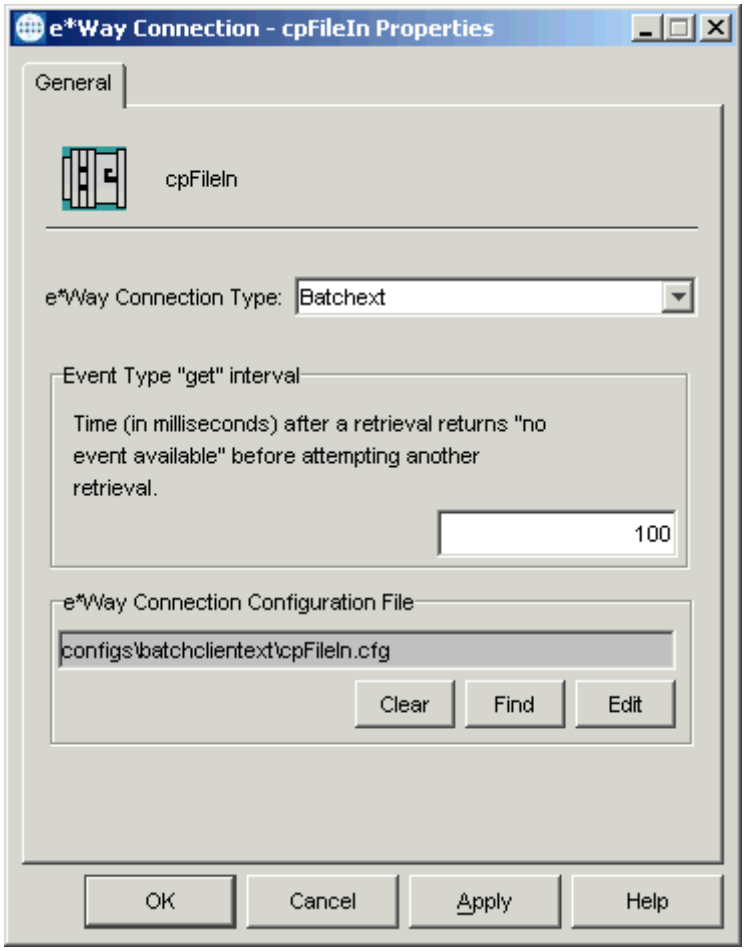
3 When you are finished, save the **.cfg** file, close the e*Way Configuration Editor, and promote the file to run time.

4 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the cpFileIn e*Way Connection

1 Repeat steps 2 through 6 under the **procedure on page 137** for the **cpFileIn** e*Way Connection (see **Figure 23 on page 145**), *except* that you select the **LocalFileETD**.

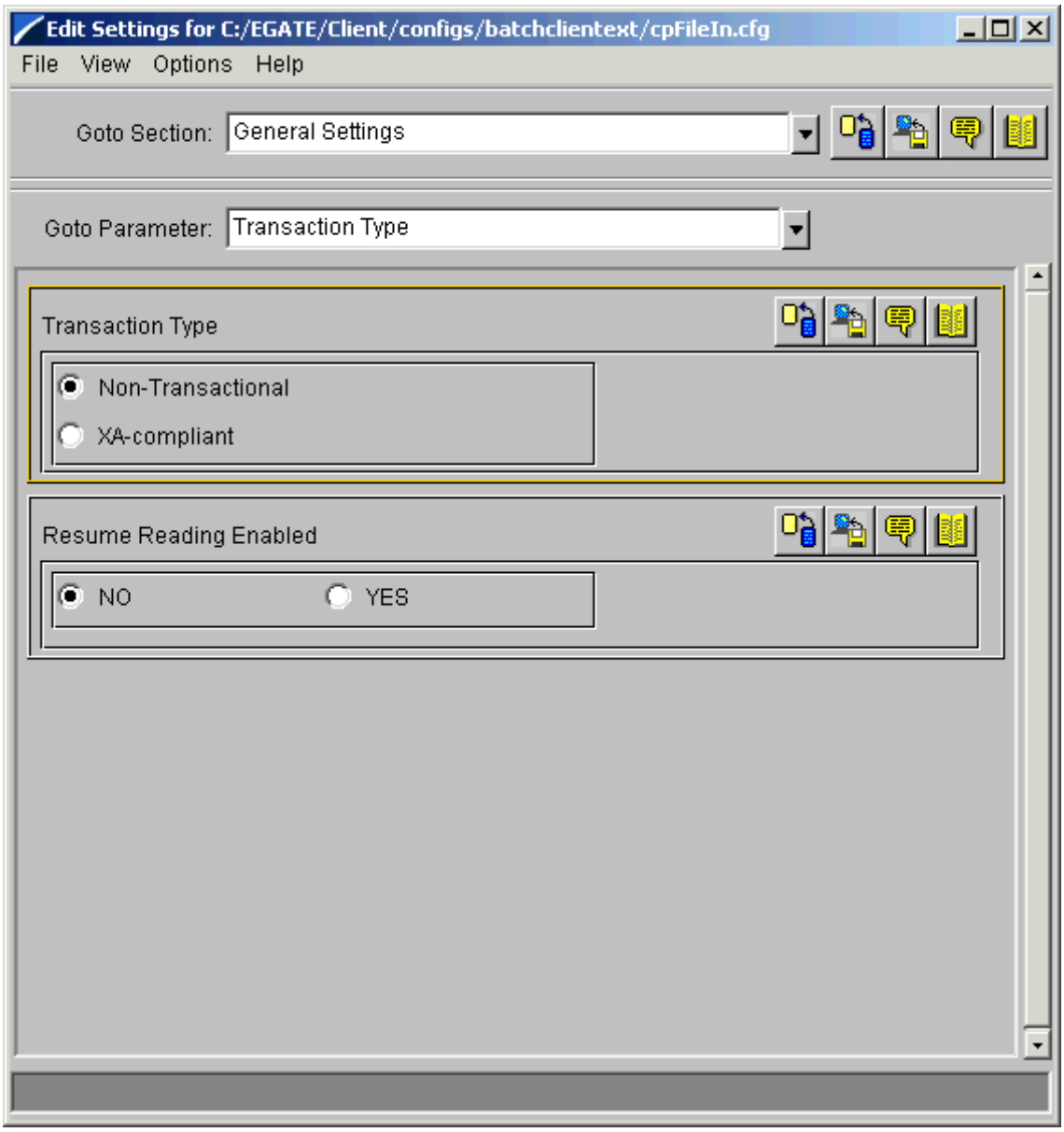
Figure 23 cpFileIn e*Way Connection Properties Dialog Box



2 Using the e*Way Configuration Editor, set the parameters for your local file system as necessary. See **"LocalFileETD: Configuration Parameters" on page 59** for details. For the other settings, you can use the defaults.

Figure 24 on page 146 shows the **General Settings** parameters for this e*Way Connection in the e*Way Configuration Editor.

Figure 24 e*Way Configuration Editor: General Settings for cpFileIn



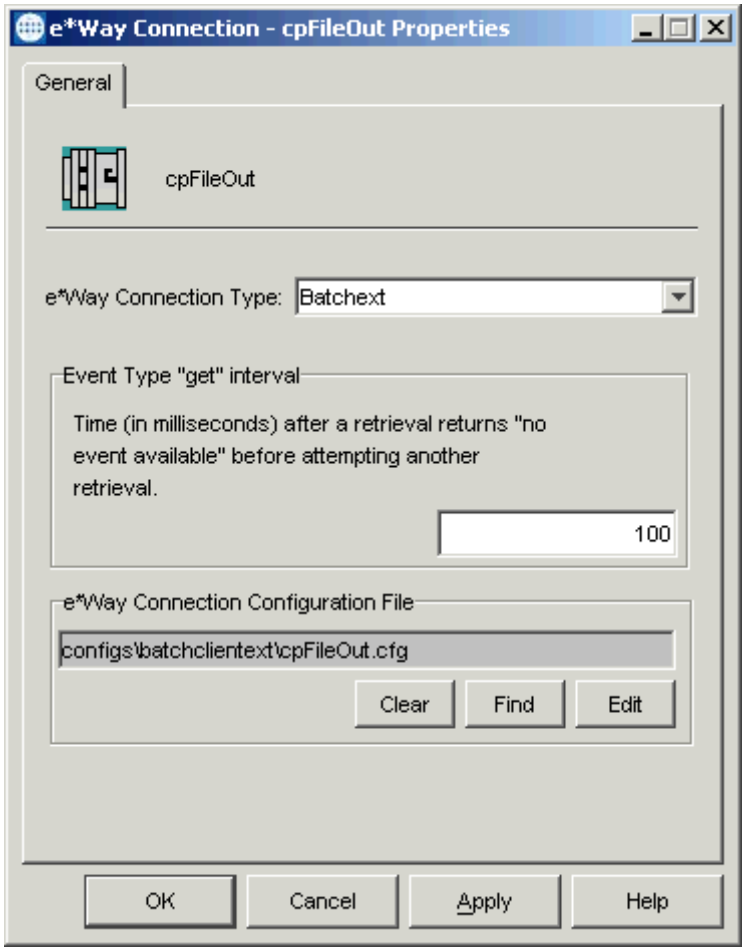
3 When you are finished, save the .**cfg** file, close the e*Way Configuration Editor, and promote the file to run time.

4 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the cpFileOut e*Way Connection

1 Repeat steps 2 through 6 under the **procedure on page 137** for the **cpFileOut** e*Way Connection (see **Figure 25 on page 147**), *except* that you select the **LocalFileETD**.

Figure 25 cpFileOut e*Way Connection Properties Dialog Box



2 Using the e*Way Configuration Editor, set the parameters for your local file system as necessary. See **"LocalFileETD: Configuration Parameters" on page 59** for details. For the other settings, you can use the defaults.

3 When you are finished, save the **.cfg** file, close the e*Way Configuration Editor, and promote the file to run time.

4 Click **OK** to close the **e*Way Connection Properties** dialog box.

Creating Collaboration Rules

The next step is to create the Collaboration Rules that extract and process selected information from the source Event Type defined previously, according to its associated Collaboration Service.

From the Schema Designer Menu bar, click **Options** and select **Default Editor**. For this schema, set the default to **Java**.

To create the cr_fromExternal Collaboration Rules file

1 Select the **Components** tab in the e*Gate Schema Designer.

2 In the Navigation pane, select the **Collaboration Rules** folder.

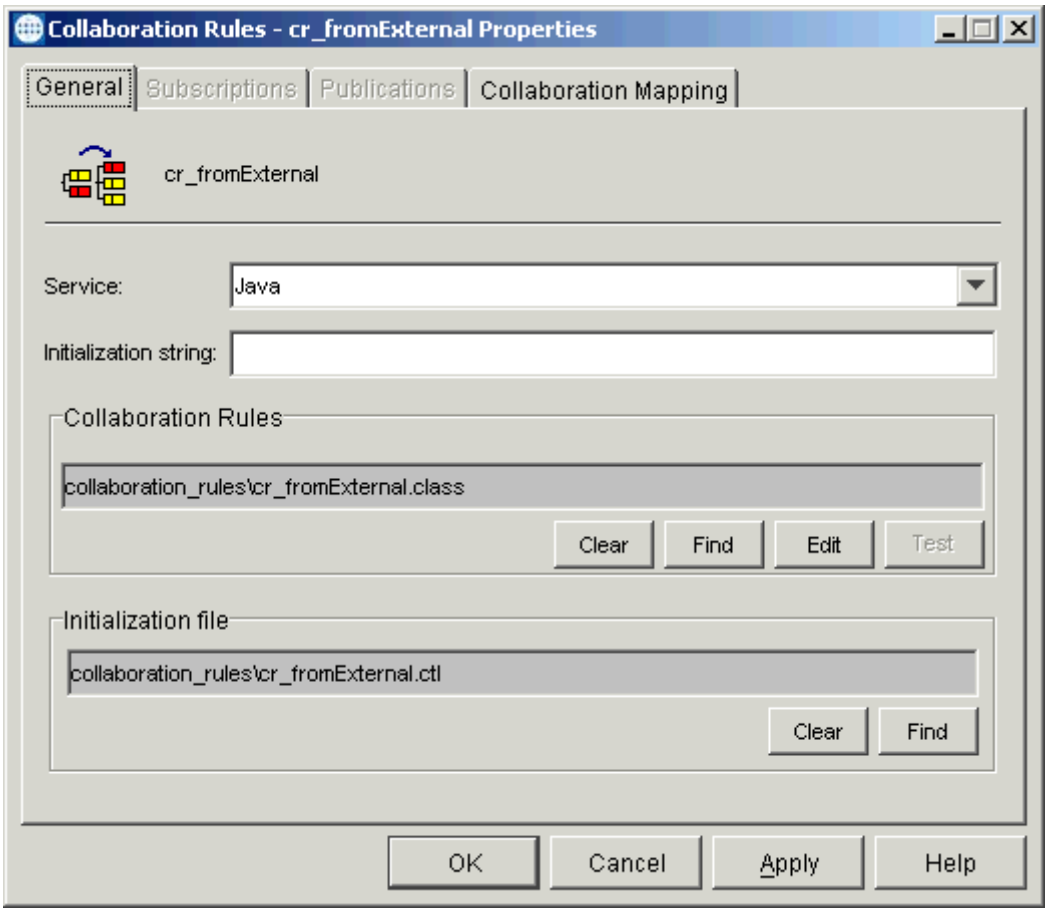
3 On the palette, click the **Collaboration Rules** icon.

4 Enter the name of the new Collaboration Rule, then click **OK.** Use **cr_fromExternal** for this example, for the **FromExternal** e*Way's Collaboration, **collabfrmExt**.

5 Select the new **Collaboration Rule**, then right-click to edit its properties.

6 The **Collaboration Rules Properties** dialog box appears (see Figure 26).

Figure 26 Collaboration Rules Properties Dialog Box for cr_fromExternal: General



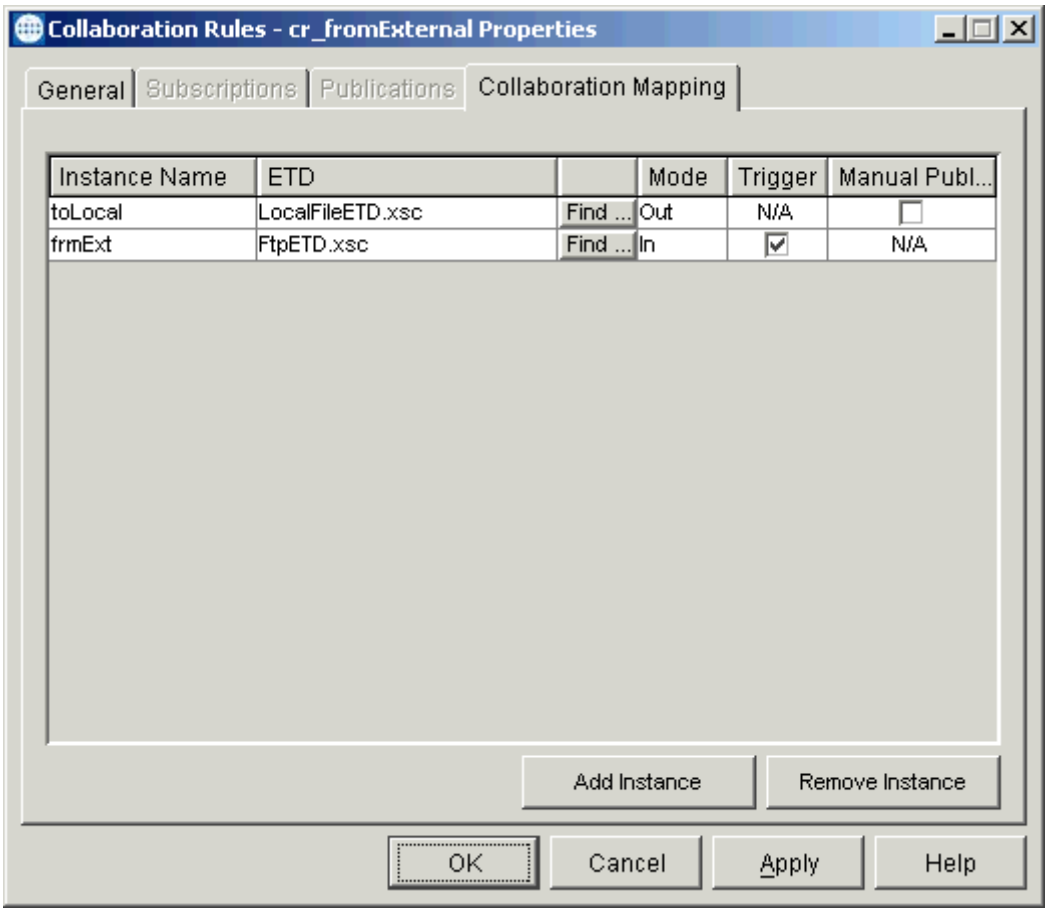
7 On the **General** tab in the dialog box select the **Java Collaboration Service**. In this example, the Collaboration Rules use the e*Gate Java Collaboration Service to manipulate Events or Event data.

8 In the **Initialization String** text box, enter any required initialization string that the Collaboration Service may require. This field can be left blank.

9 Click the **Collaboration Mapping** tab.

10 Using the **Add Instance** button, create instances to coincide with the ETDs (see Figure 27).

Figure 27 Collaboration Rules Properties Dialog Box for cr_fromExternal: Mapping



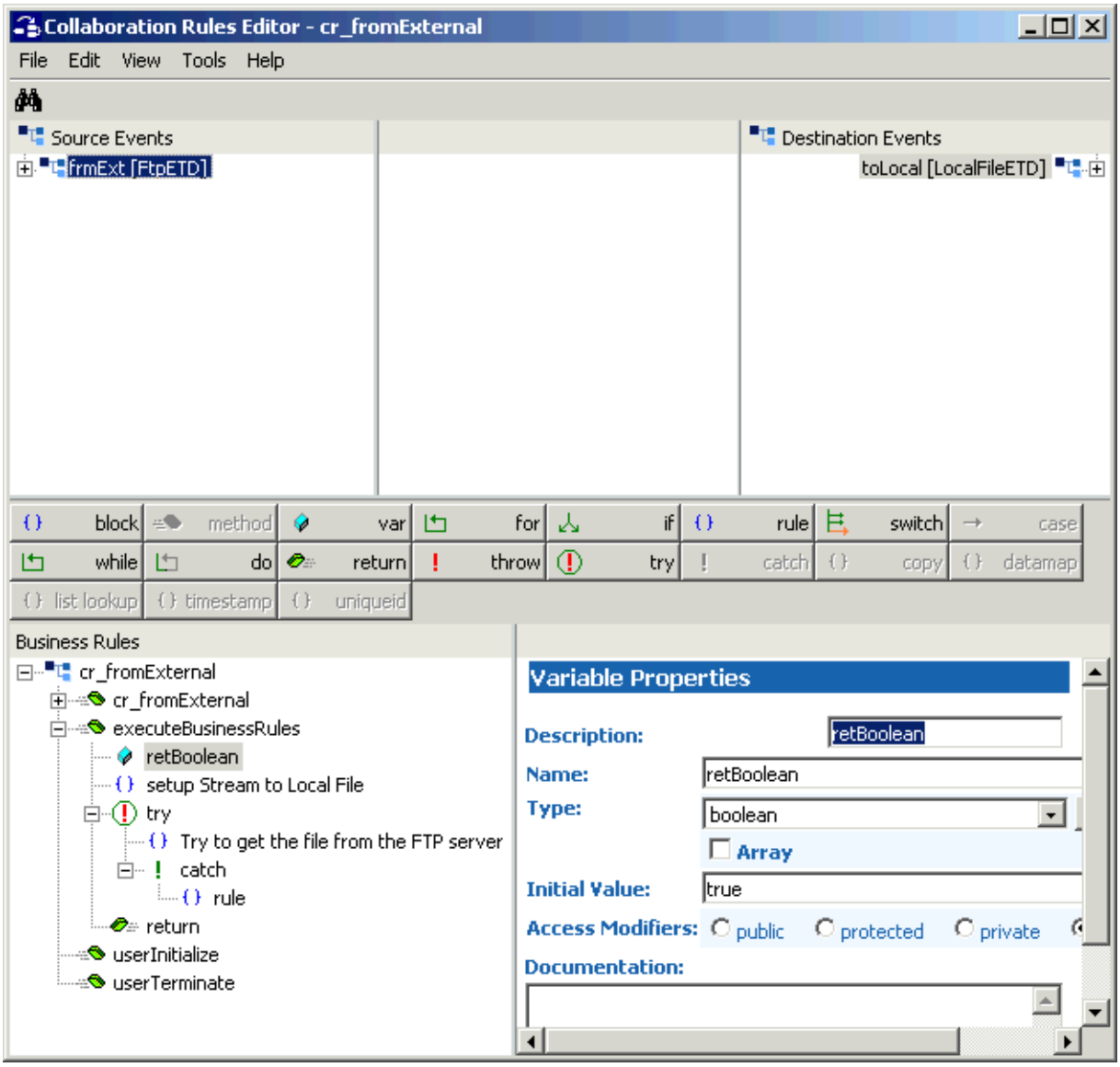
Configure the rest of **cr_fromExternal** as shown in the previous figure.

11 Select the **General** tab again, then click **New** (where the **Edit** button is in **Figure 26 on page 148**).

The Collaboration Rules Editor Main window opens.

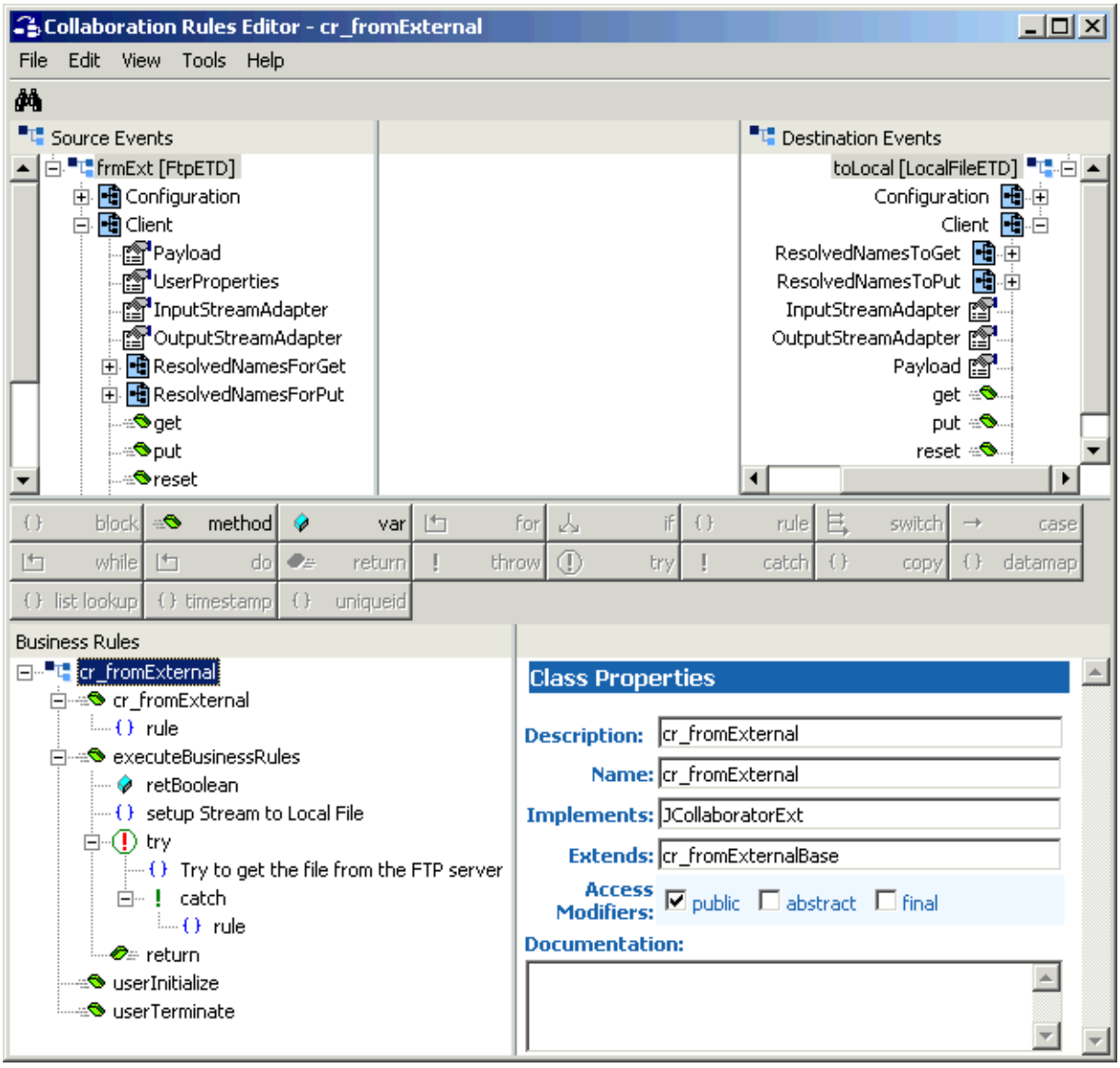
12 Expand the window to full size for optimum viewing (see Figure 28).

Figure 28 Collaboration Rules Editor: cr_fromExternal Start



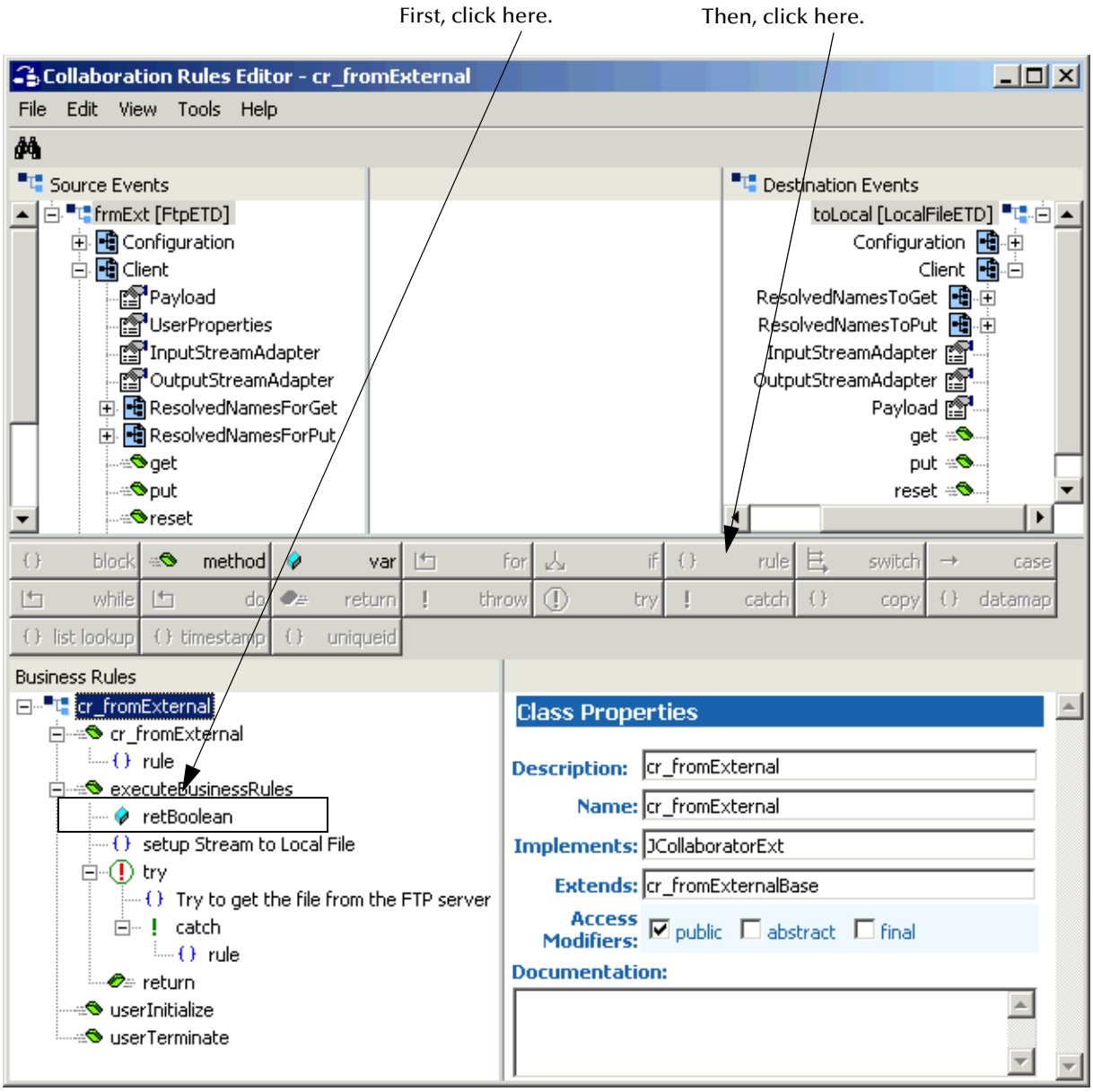
13 Expand the source and destination Events, as well as the Business Rules. Figure 29 shows the results.

Figure 29 Collaboration Rules Editor: cr_fromExternal Expanded



14 To begin creating the first Business Rule, first click the **retBoolean** method in the Business Rules pane (see **Figure 30 on page 152**).

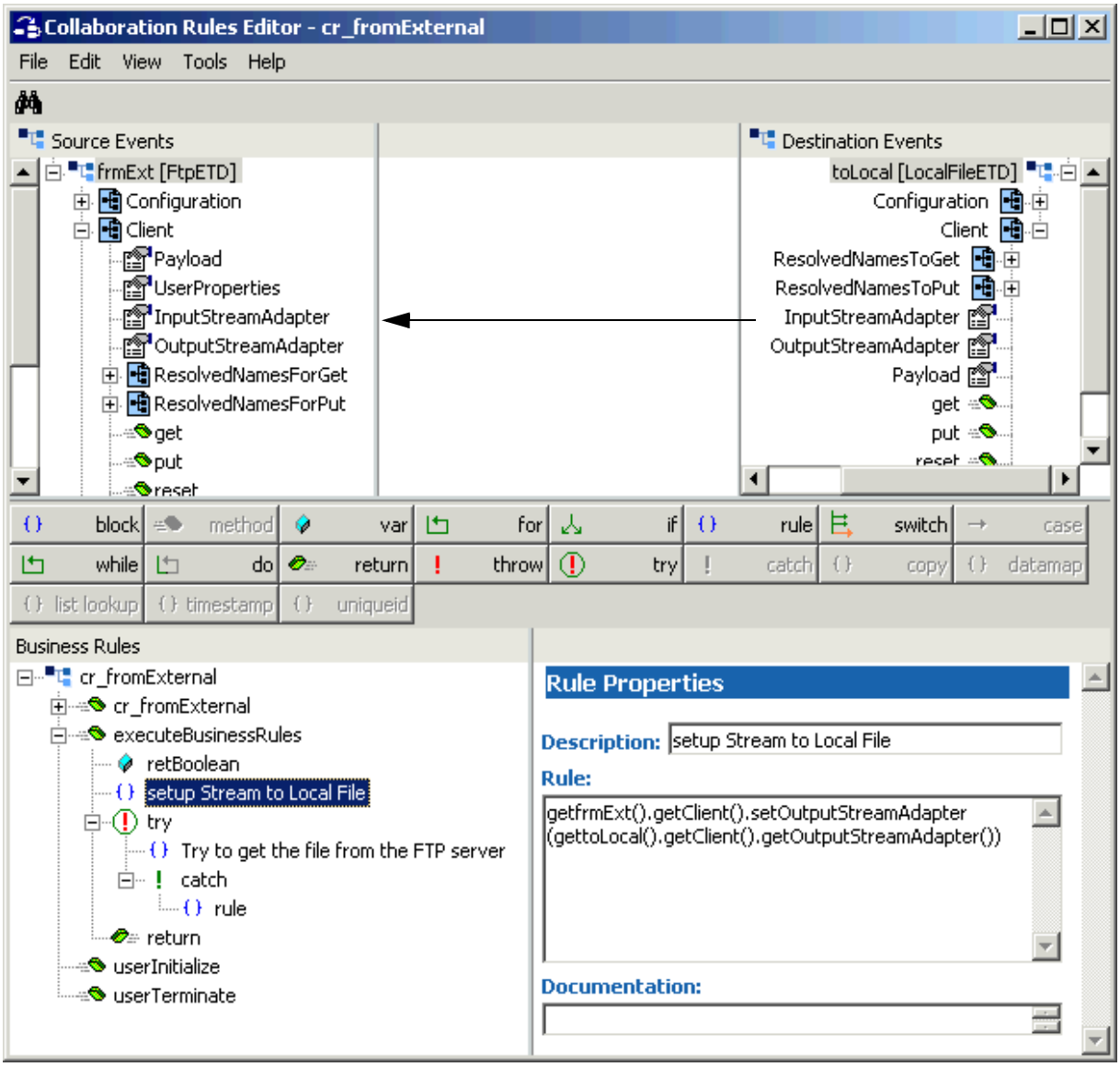
Figure 30 Collaboration Rules Editor: Getting Started



15 Drag the **OutputStreamAdapter** node of the **Destination Event** onto the **OutputStreamAdapter** node of the **Source Event** (see **Figure 31 on page 153**). This action creates the new rule.

16 Name the rule **setup Stream to Local File**.

Figure 31 Collaboration Rules Editor: cr_fromExternal First Rule (Drag/drop)



17 When you first complete this operation, the code shown in the **Rule** scroll box in the **Rule Properties** window is not correct (unlike the previous figure). To complete this implementation, you must delete the code listed in the **Rule** scroll box in the **Rule Properties** window and enter the following code:

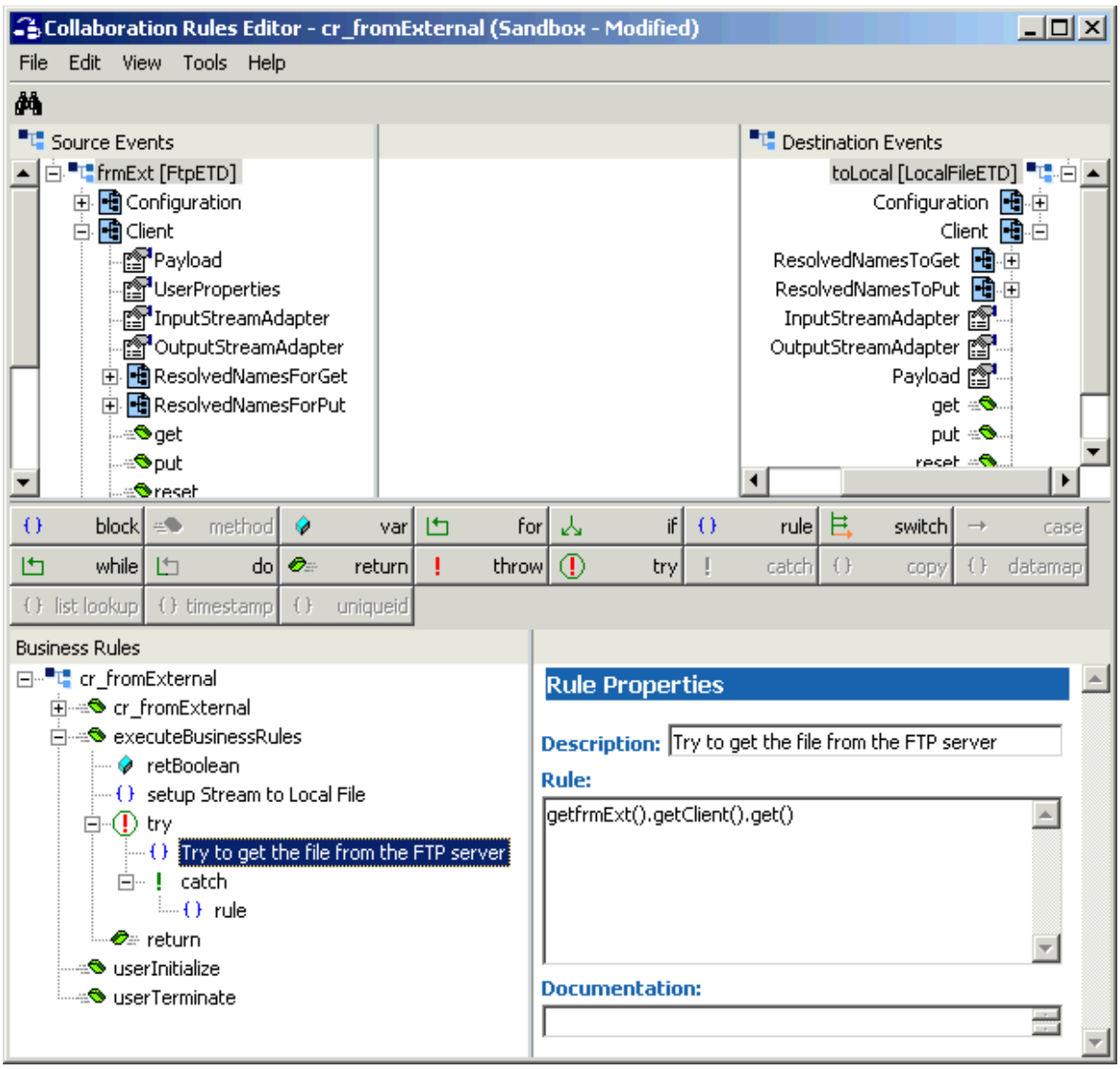
```
getfrmExt().getClient().setOutputStreamAdapter(gettoLocal()
    .getClient().getOutputStreamAdapter())
```

When you are finished typing, the code looks like what is shown in Figure 31. This rule sets up the data-streaming transfer.

Note: *For more information on the data-streaming feature, see* **"Streaming Data Between Components" on page 329**.

18 With the previous rule highlighted, click **try** to begin creating another new rule (see Figure 32).

Figure 32 Collaboration Rules Editor: cr_fromExternal (Click try)

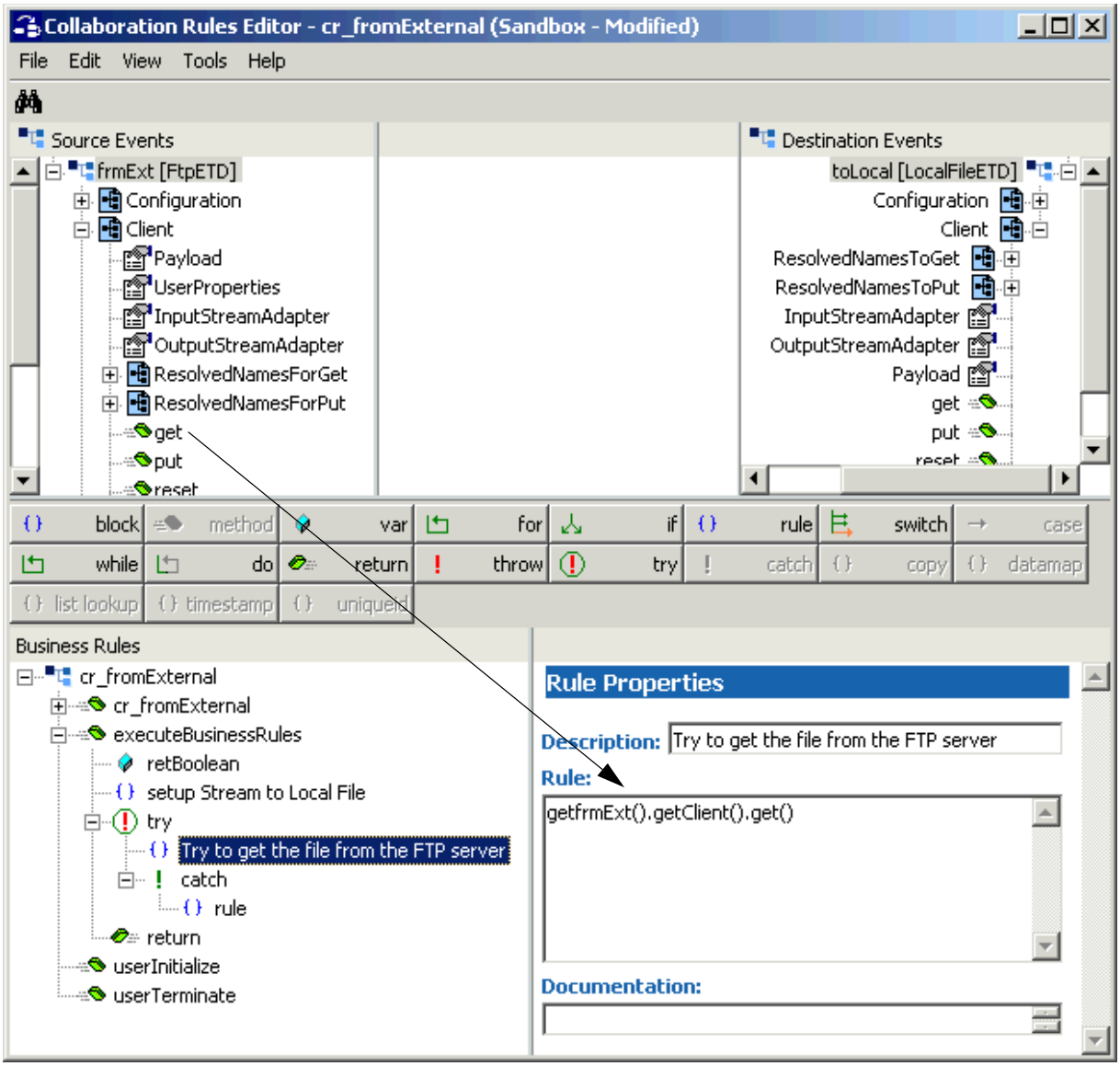


19 Click **rule** (with **try** highlighted) to finish creating this rule. Name the rule **Try to get the file from the FTP server**.

In this rule, you must call the **get** method on the FTP ETD, which consumes the stream adapter, to perform the transfer.

20 Drag the **get** method from the **Source Event** to the **Rule** scroll box in the **Rule Properties** window (see Figure 33).

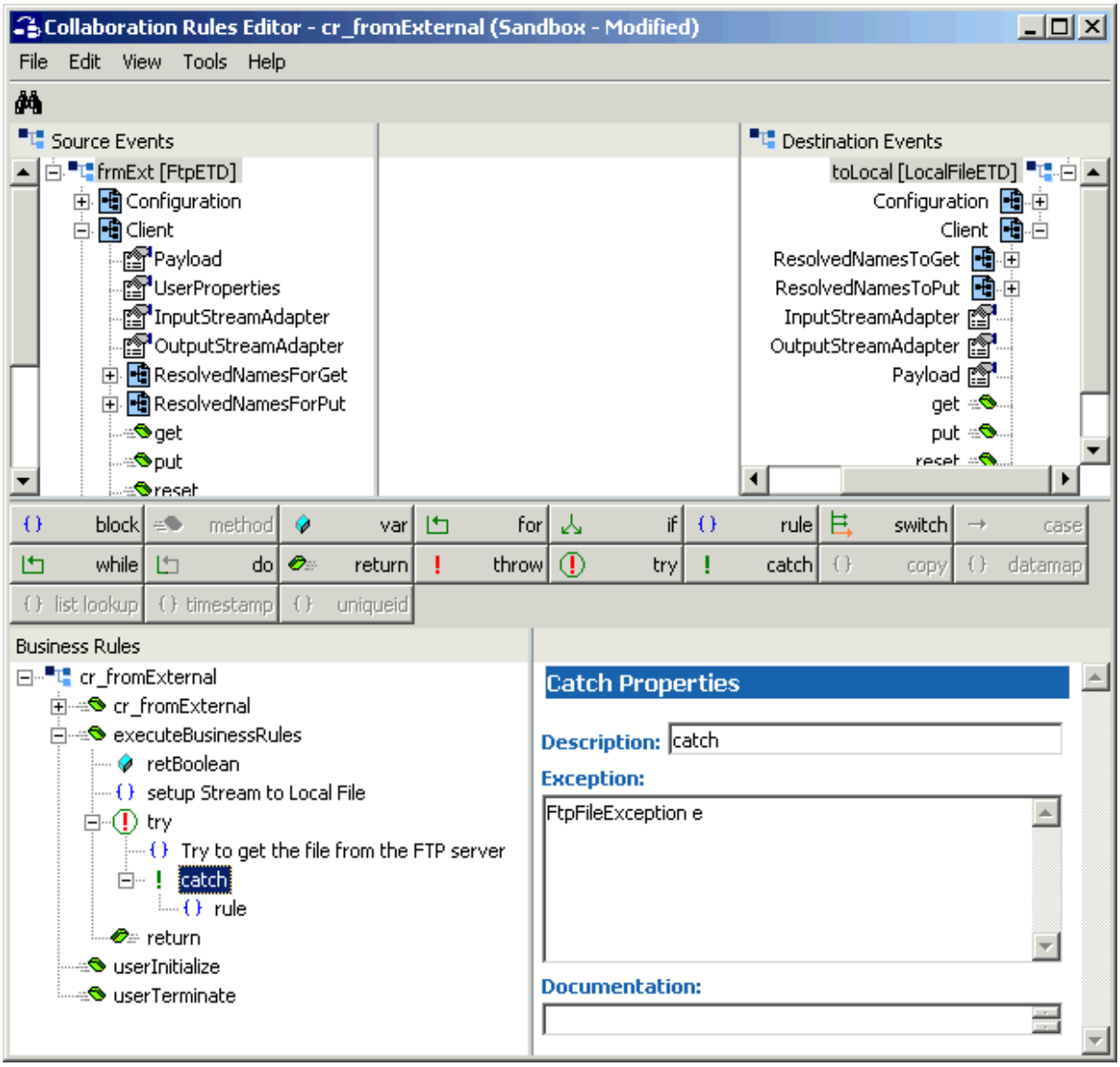
Figure 33 Collaboration Rules Editor: cr_fromExternal Second Rule (Drag get)



This rule completes the data-streaming transfer.

21 Click **catch** (with the previous rule highlighted) to begin creating the last rule (see **Figure 34 on page 156**).

Figure 34 Collaboration Rules Editor: cr_fromExternal (Click catch)

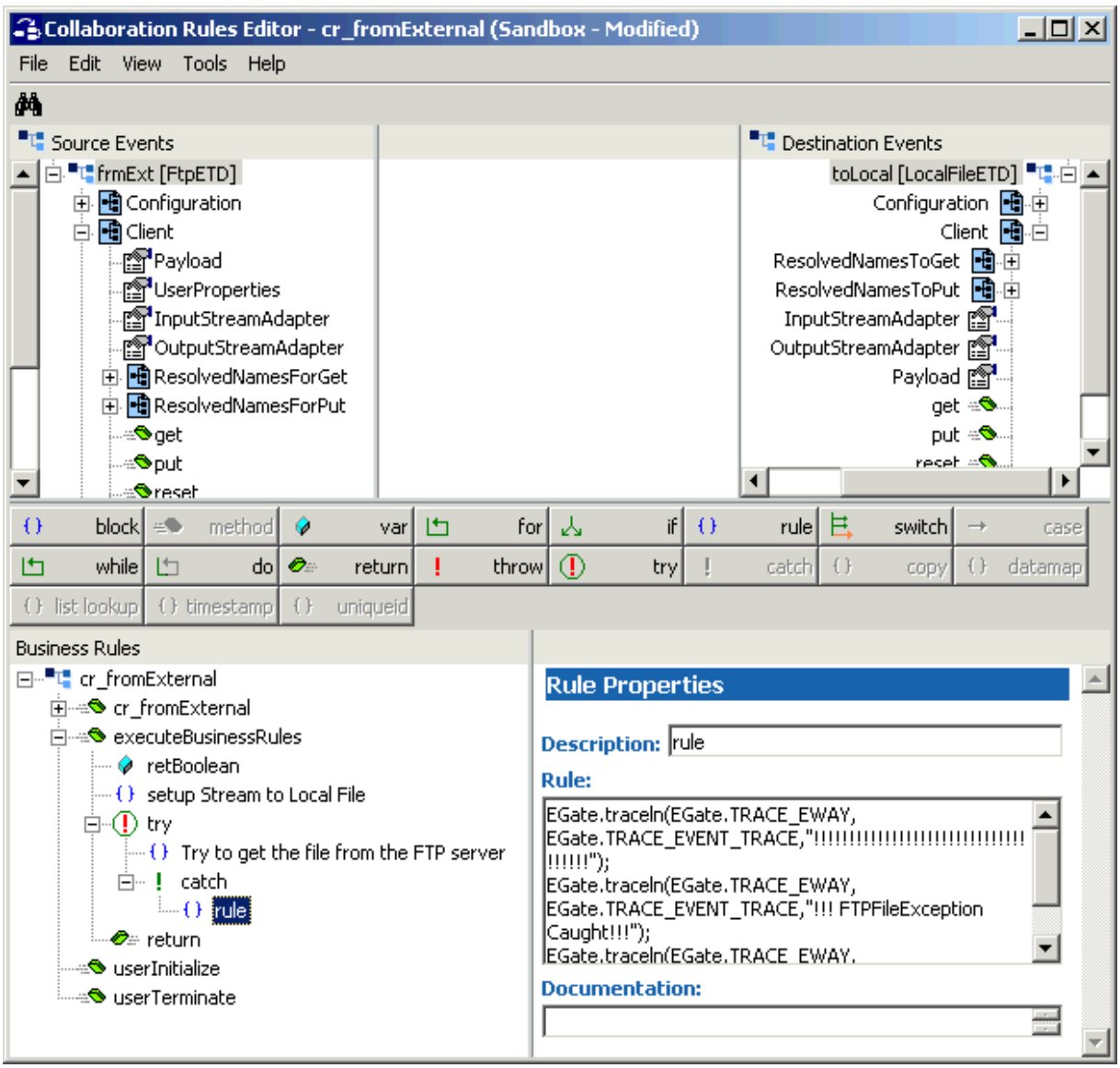


This final rule allows your Collaboration Rule to handle errors.

Note: *See the Javadoc for complete information on the exceptions thrown by the e*Way's methods.*

22 Click **rule** (with **catch** highlighted) to finish creating the final rule (see Figure 35).

Figure 35 Collaboration Rules Editor: cr_fromExternal Third Rule (Error Handling)



23 Type the following text in the **Rule** scroll box (see Figure 35) in the **Rule Properties** window:

```
EGate.traceln(EGate.TRACE_EWAY,
EGate.TRACE_EVENT_TRACE,"!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
EGate.traceln(EGate.TRACE_EWAY, EGate.TRACE_EVENT_TRACE,"!!!
FTPFileException Caught!!!");
EGate.traceln(EGate.TRACE_EWAY,
EGate.TRACE_EVENT_TRACE,"!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
```

Note: *See the Javadoc for details on this code.*

You are now finished creating your Business Rules.

24 You must create a Collaboration Rules class or use one from the sample (**cr_fromExternal.class**).

Note: *See the* ***e*Gate Integrator User's Guide*** *for details on this procedure.*

25 To save the Collaboration Rules file, click **Save** on the **File** menu. The **Save** dialog box appears.

26 Provide a name for the **.xpr** file (for this example, use **cr_fromExternal.xpr**) then click **Save**.

27 Before compiling the code, on the **Tools** menu, click **Options** and verify that all necessary **.jar** files are included (see **"Collaboration Rules Editor: Java Classpaths Dialog Box" on page 255**).

28 When you have finished defining all the desired business logic, compile the Java code by selecting **Compile** from the **File** menu.

If the code compiles successfully, the message **Compile Completed** appears. If the outcome is unsuccessful, a Java Compiler error message appears. If there are any Java errors, be sure to correct them.

29 Once the compilation is complete, you can exit the Collaboration Rules Editor.

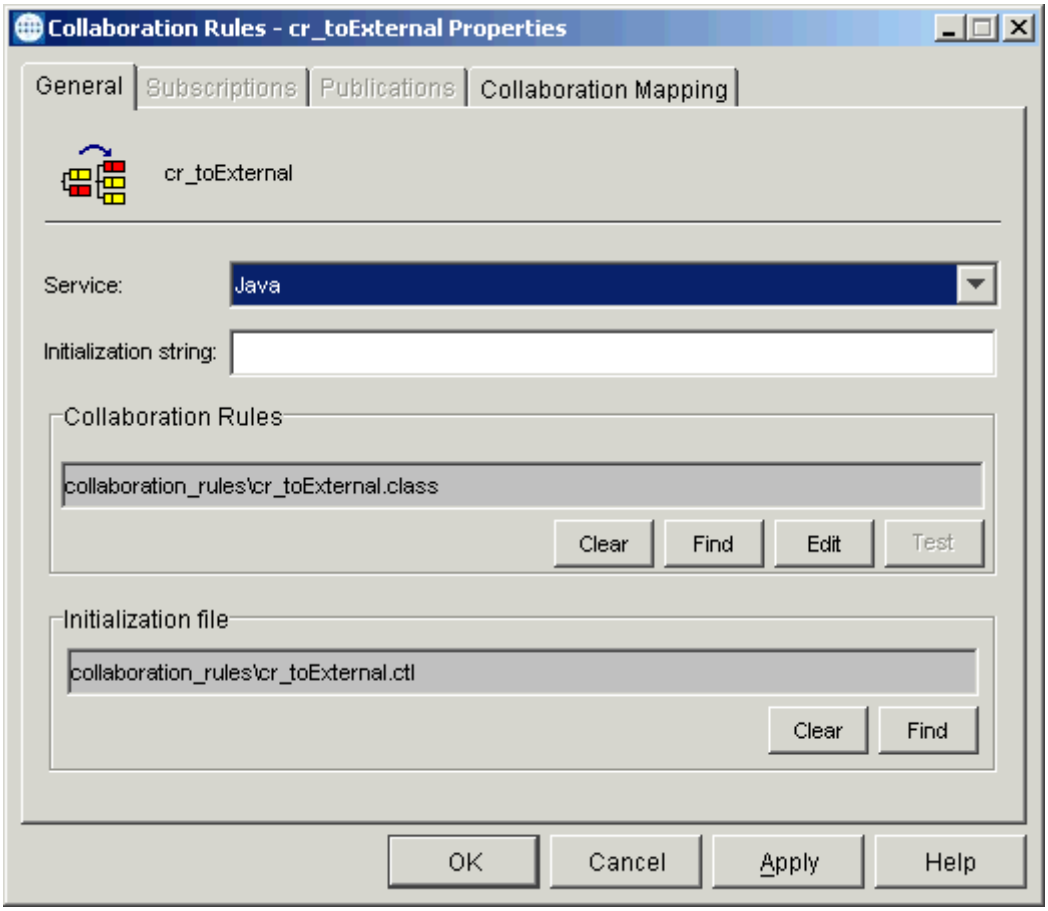
To create the cr_toExternal Collaboration Rules file

1 Repeat steps 3 through 5 under the **procedure on page 147** to create the next Collaboration Rule.

Use **cr_toExternal** as the name for this example, for the **ToExternal** e*Way's Collaboration, **collabToExt**.

2 The **Collaboration Rules Properties** dialog box appears (see **Figure 36 on page 159**).

Figure 36 Collaboration Rules Properties Dialog Box for cr_toExternal: General



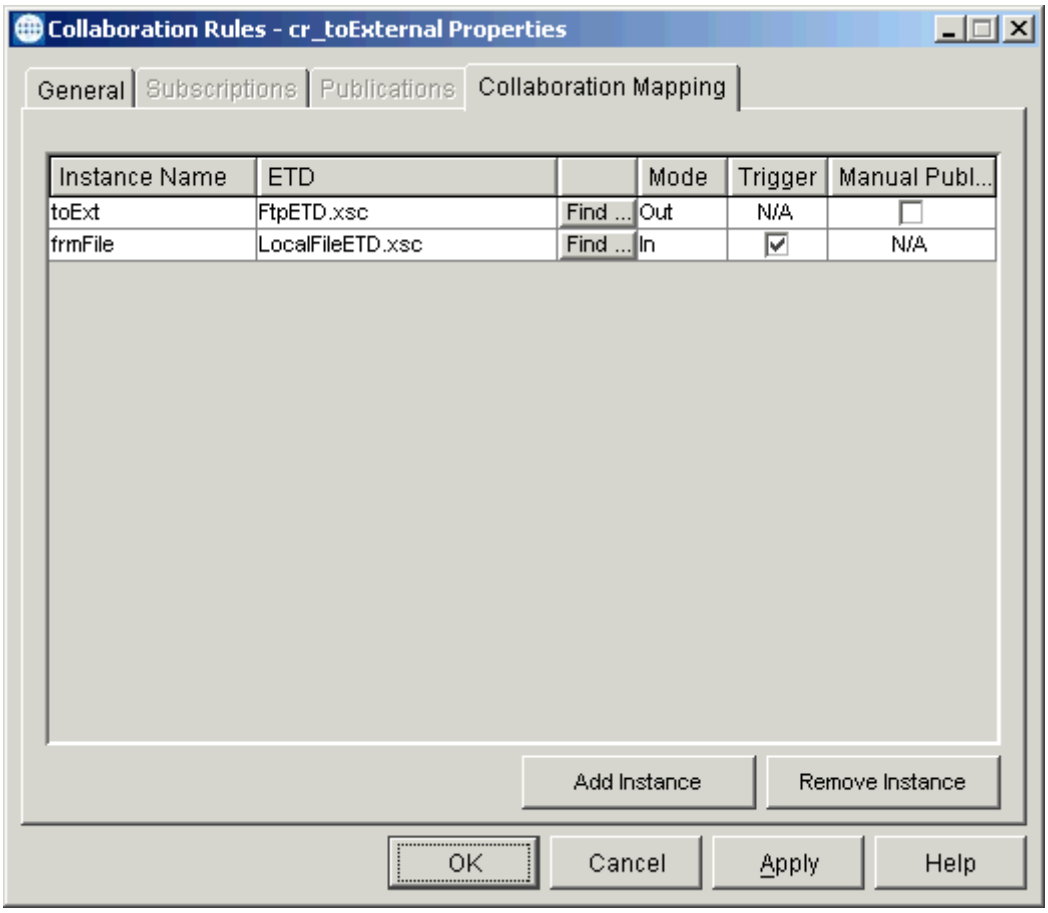
3 On the **General** tab in the dialog box select the **Java Collaboration Service**. In this example, the Collaboration Rules use the e*Gate Java Collaboration Service to manipulate Events or Event data.

4 In the **Initialization String** text box, enter any required initialization string that the Collaboration Service may require. This field can be left blank.

5 Click the **Collaboration Mapping** tab.

6 Using the **Add Instance** button, create instances to coincide with the ETDs (see Figure 37).

Figure 37 Collaboration Rules Properties Dialog Box for cr_toExternal: Mapping



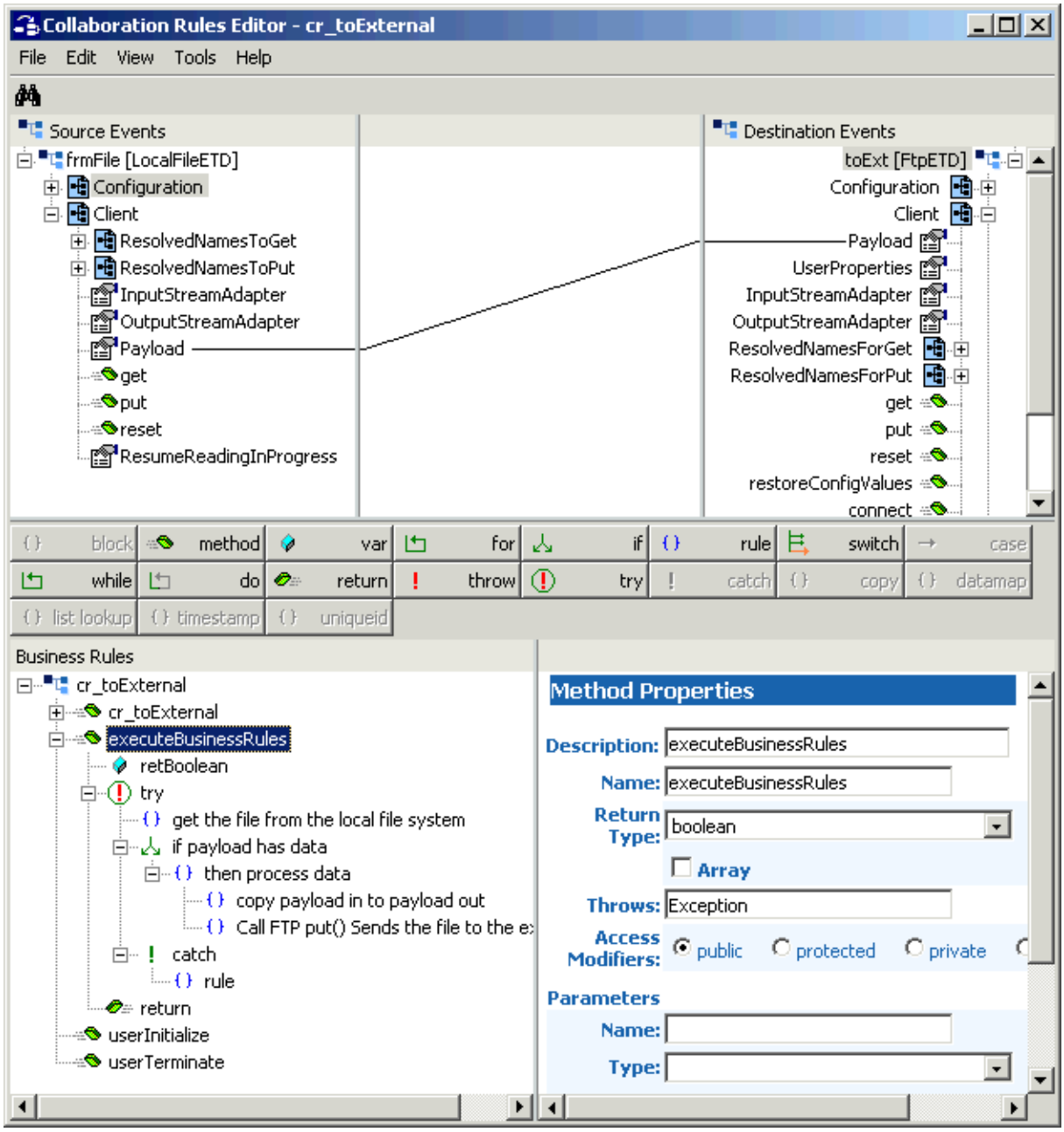
Configure the rest of **cr_toExternal** as shown in the previous figure.

7 Select the **General** tab again, then click **New** (where the **Edit** button is in **Figure 36 on page 159**).

The Collaboration Rules Editor Main window opens.

8 Expand the window to full size for optimum viewing, then expand the source and destination Events, as well as the Business Rules. Figure 38 shows the results.

Figure 38 Collaboration Rules Editor: cr_toExternal Expanded

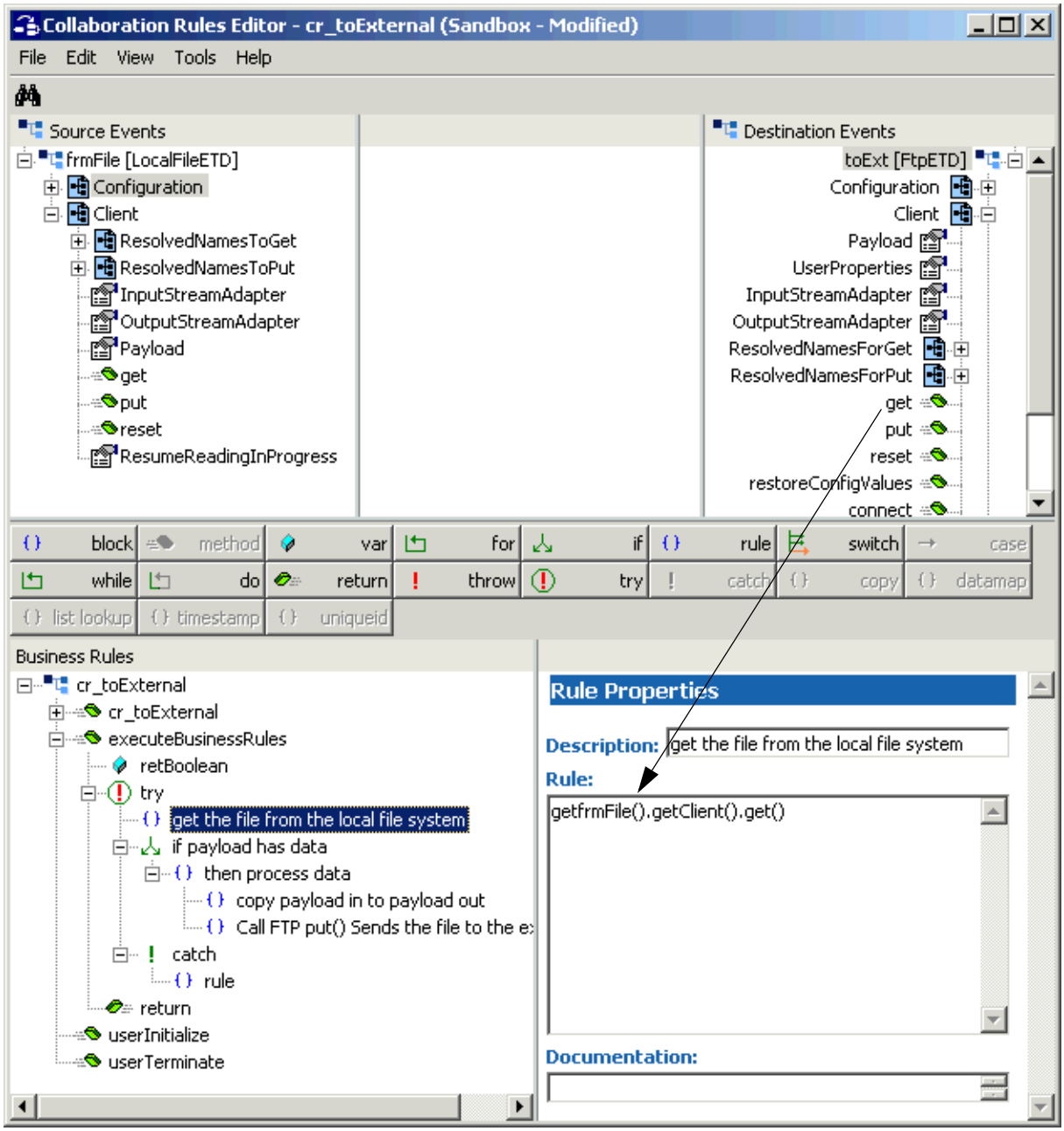


9 To create the first Business Rule, click the **retBoolen** method in the **Business Rules** window then click **try** (you can delete the **finally** rule that appears).

10 With **try** highlighted, click **rule**. Name the new rule **get the file from the local file system**.

11 Drag the **get** method from the **Destination Event** to the **Rule** scroll box in the **Rule Properties** window (see Figure 39).

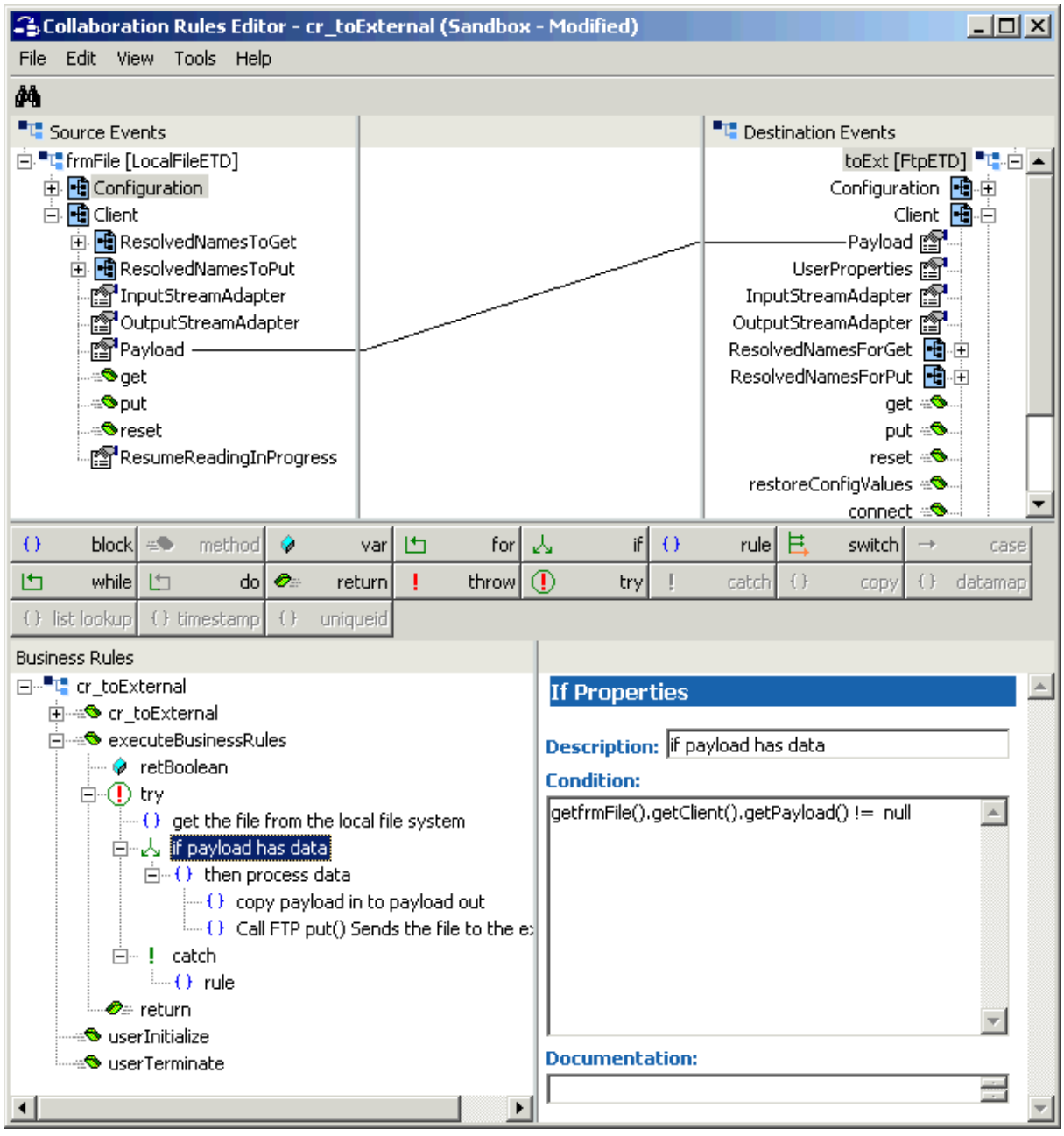
Figure 39 Collaboration Rules Editor: cr_toExternal First Rule (Drag get)



This rule gets the file from the local file system.

12 Click **if** (with the previous rule highlighted) to begin creating another new rule as shown in **Figure 40 on page 163**. Name the new **if** rule **if payload has data**.

Figure 40 Collaboration Rules Editor: cr_toExternal (Click if)



This rule sets up the final payload data transfer and sending to the external FTP system.

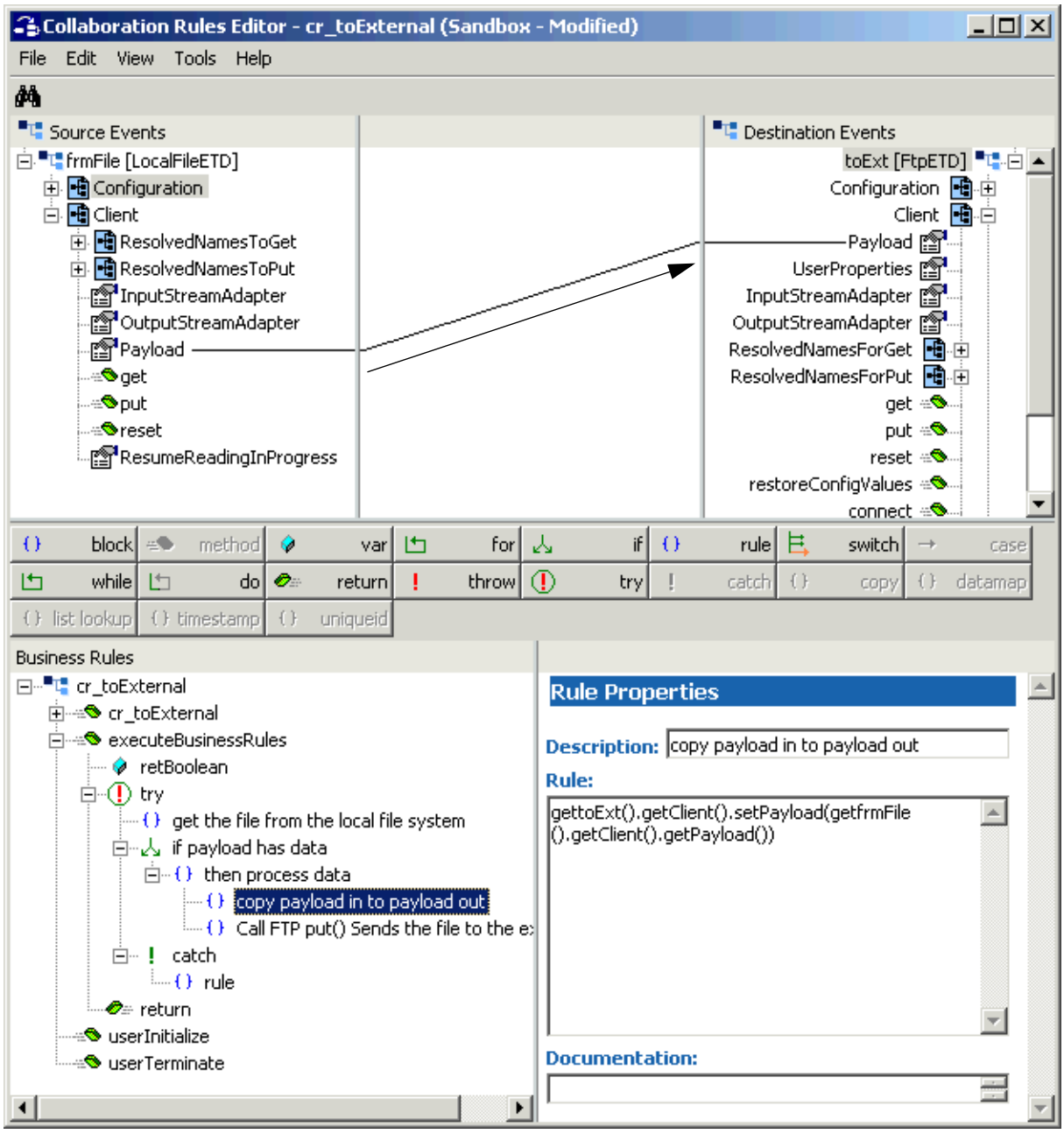
13 As shown in Figure 40, type the following text in the he **Rule** scroll box in the **Rule Properties** window:

```
getfrmFile().getClient().getPayload() !=  null
```

14 Click the **then** statement and name it **then process data**.

15 With the previous rule still selected, drag the **Payload** node of the **Source Event** onto the **Payload** node of the **Destination Event** (see Figure 41). This action creates a new rule. Make sure this rule is a **child** of the **then process data** rule.

Figure 41 Collaboration Rules Editor: cr_toExternal Second Rule (Drag/drop Payload)

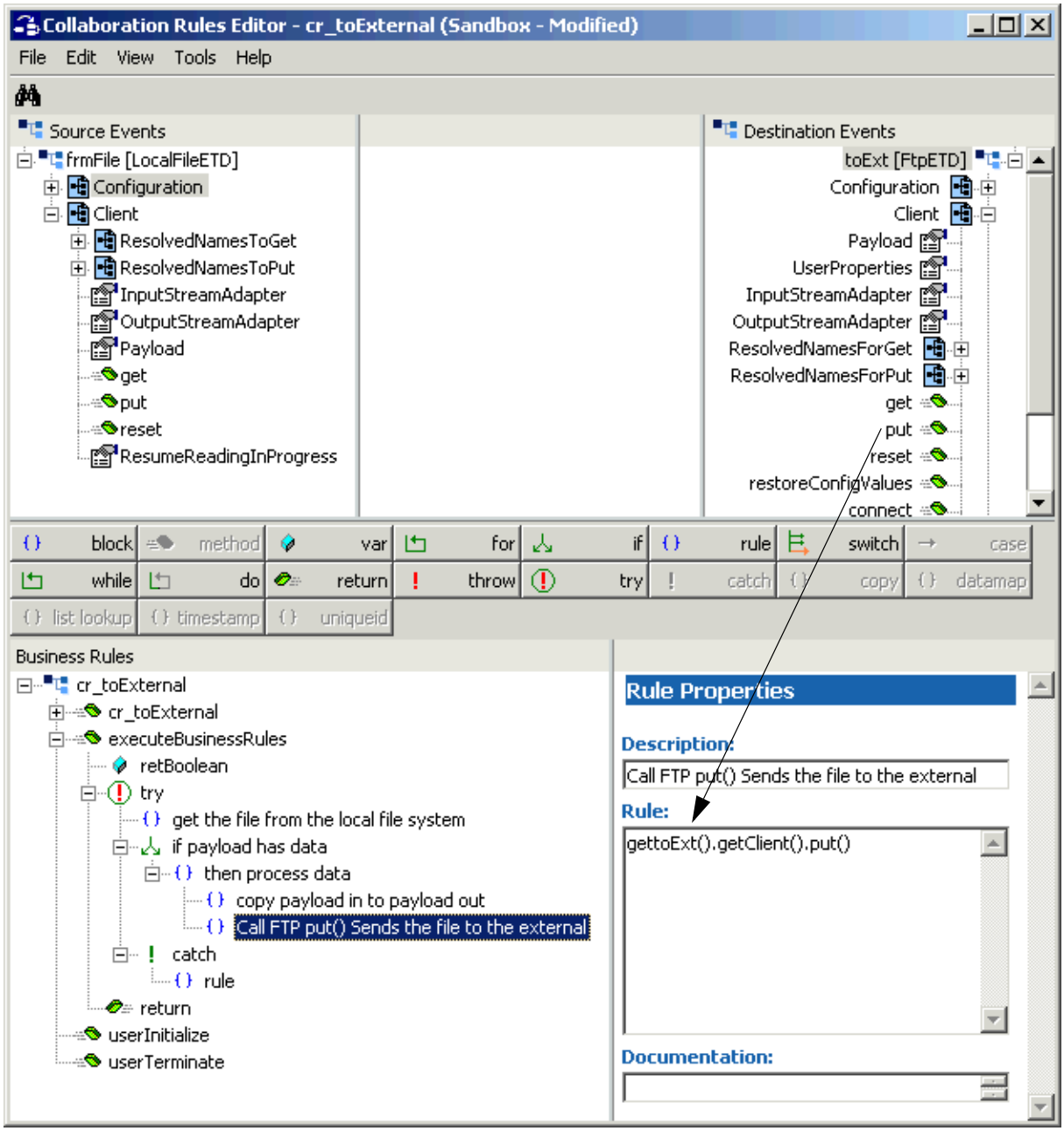


This rule transfers the payload data from one ETD to the other.

16 Name the new rule **copy payload in to payload out**.

17 With the **then process data** rule still selected, drag the **put** method from the **Destination Event** to the **Rule** scroll box in the **Rule Properties** window (see Figure 42). This action creates a new rule. Make sure this rule is a **child** of the **then process data** rule.

Figure 42 Collaboration Rules Editor: cr_toExternal Third Rule (Drag put)

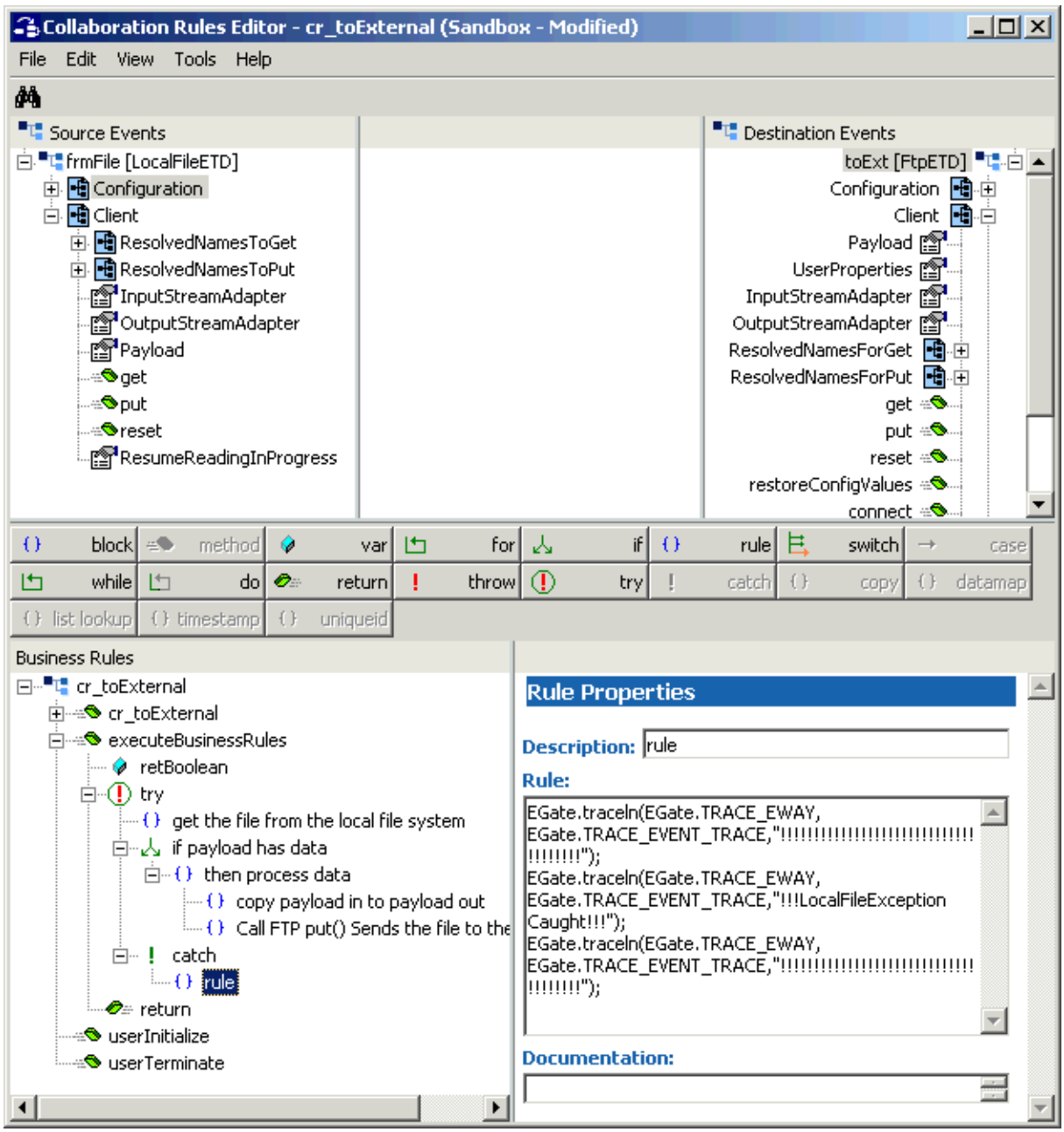


18 Name the new rule **Call FTP put() sends the file to the external**.

19 Click **try** (to highlight it) then **catch** to begin creating the last rule as shown in **Figure 43 on page 166**.

20 With **catch** highlighted, click **rule** (see Figure 43).

Figure 43 Collaboration Rules Editor: cr_toExternal Fourth Rule (Click catch, rule)



This final rule allows your Collaboration Rule to handle errors.

Note: *See the Javadoc for complete information on the exceptions thrown by the e*Way's methods.*

21 Type the following text in the **Rule** scroll box (see **Figure 43 on page 166**) in the **Rule Properties** window:

```
EGate.traceln(EGate.TRACE_EWAY,
EGate.TRACE_EVENT_TRACE,"!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
EGate.traceln(EGate.TRACE_EWAY,
EGate.TRACE_EVENT_TRACE,"!!!LocalFileException Caught!!!");
EGate.traceln(EGate.TRACE_EWAY,
EGate.TRACE_EVENT_TRACE,"!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
```

You are now finished creating your Business Rules.

22 You must create a Collaboration Rules class or use one from the sample (**cr_toExternal.class**).

23 Compile and save this Collaboration Rules file in the same way as you did the previous file for **cr_fromExternal**. Name this file **cr_toExternal.xpr**.

24 When you are finished, exit the Collaboration Rules Editor.

Creating Collaborations

Collaborations are the components that receive and process Event Types, then forward the output to other e*Gate components or an external system.

Collaborations consist of the subscriber, which receives Events of a known type (sometimes from a given source), and the publisher, which distributes transformed Events to a specified recipient.

To create the Collaborations

1 In the e*Gate Schema Designer, select the Navigator's **Components** tab.

2 Open the host on which you want to create the Collaboration.

3 Select the Control Broker for this schema.

4 Select the **FromExternal** e*Way to assign the Collaboration.

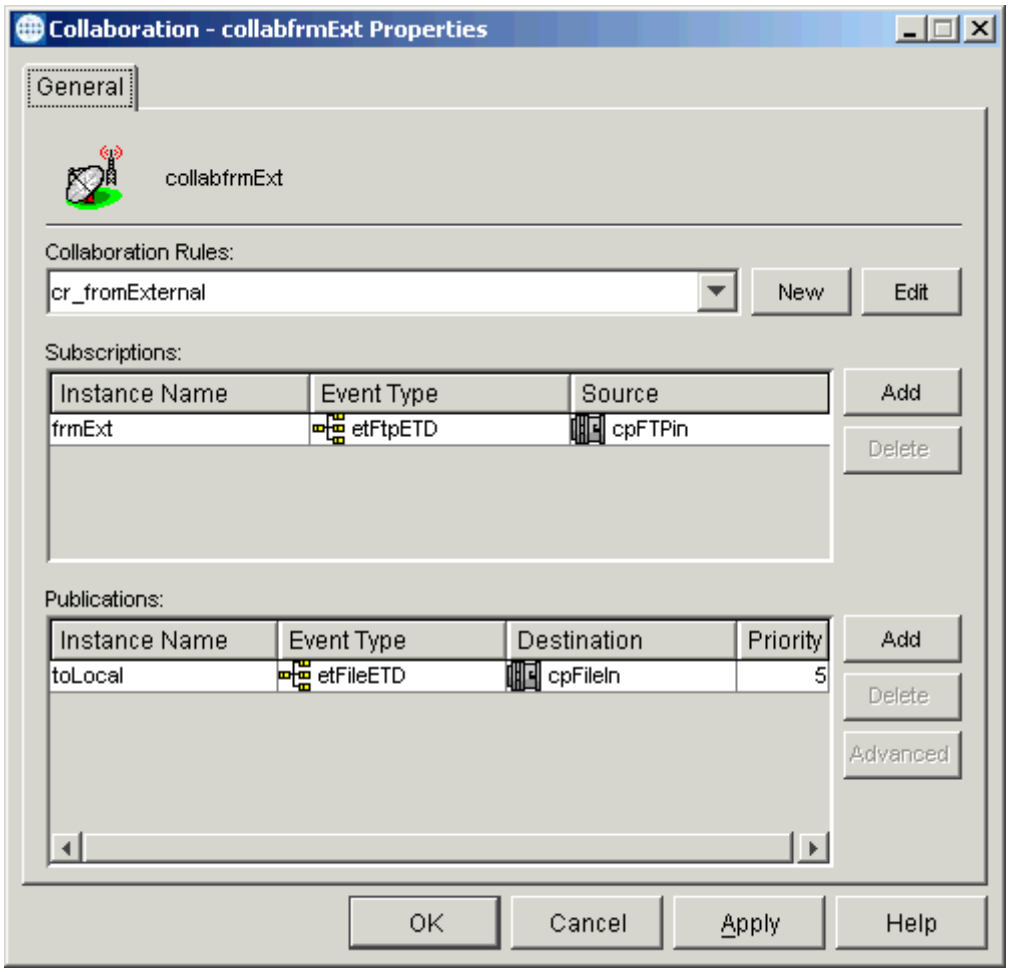
5 On the palette, click the **Collaboration** icon.

6 Enter the name (**collabfrmExt**) of the new Collaboration, then click **OK**.

7 Select the new Collaboration, then right-click to edit its properties.

The **Collaboration Properties** dialog box appears (see **Figure 44 on page 168**).

Figure 44 collabfrmExt Properties Dialog Box



Configure the appropriate Collaboration properties as shown in the previous figure.

8 From the **Collaboration Rules** list, select the first Collaboration Rule that you created previously (**cr_fromExternal**) for this Collaboration.

9 Click **OK** to close the dialog box and save your changes.

10 Select the **ftpOut** e*Way to assign the next Collaboration.

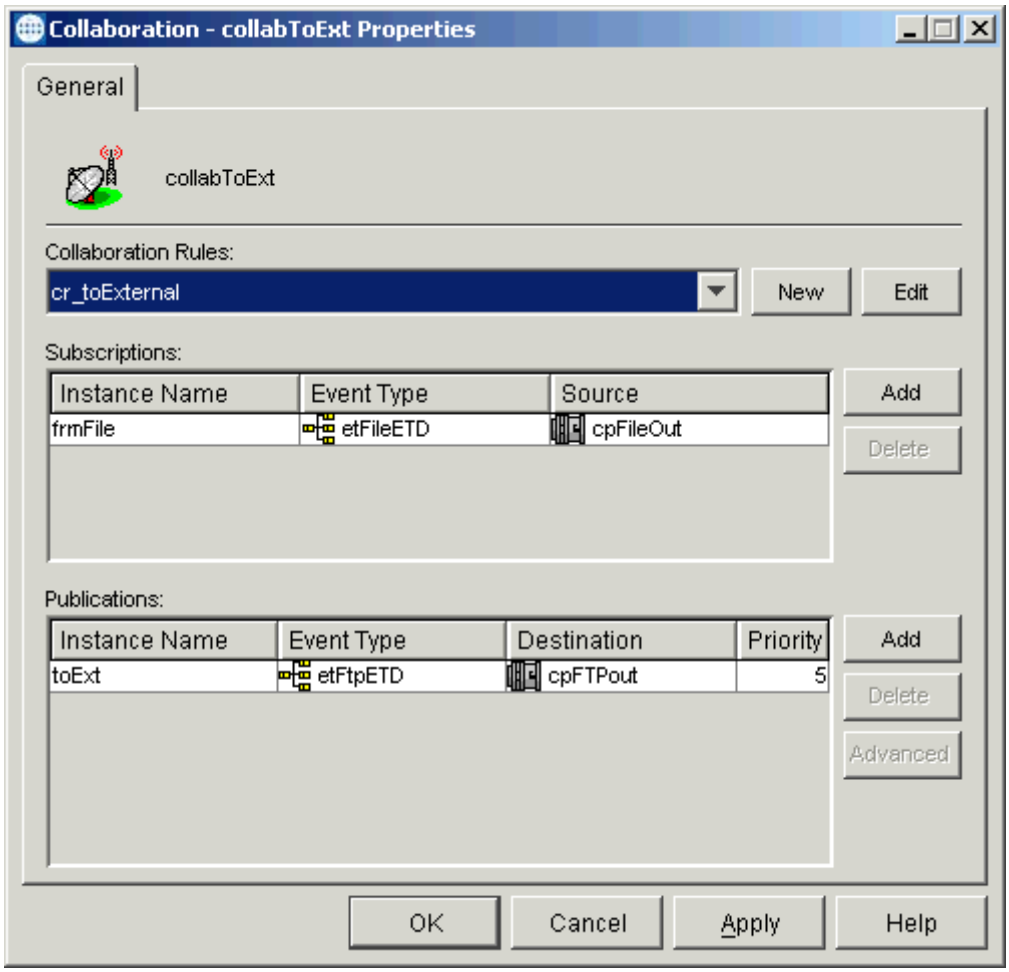
11 On the palette, click the **Collaboration** icon.

12 Enter the name (**collabToExt**) of the new Collaboration, then click **OK**.

13 Select the new Collaboration, then right-click to edit its properties.

The **Collaboration Properties** dialog box appears (see **Figure 45 on page 169**).

Figure 45 collabToExt Properties Dialog Box



Configure the appropriate Collaboration properties as shown in the previous figure.

14 From the **Collaboration Rules** list, select the Collaboration Rule that you created previously (**cr_toExternal**) for this Collaboration.

15 Click **OK** to close the dialog box and save your changes.

Running the Schema

To run the schema

- From the command line prompt, enter on a single line:

```
stccb -rh hostname -rs schemaname -un username
    -up user_password -ln localhost_cb
```

Substitute the appropriate names for ***hostname***, ***username***, ***schemaname***, and ***user_password*** as appropriate.

The schema components start automatically. When there are no more run-time messages, check the output file. If the schema is operating correctly, the remote FTP site contains the payload data in the directory you specified.

7.3 Sample Schema: Local File Streaming and GEOD

This section explains how to implement the local file data-streaming and record-processing features in a sample schema for the Batch e*Way. The schema also employs GEOD (XA mode) along with the local file ETD's Resume Reading feature.

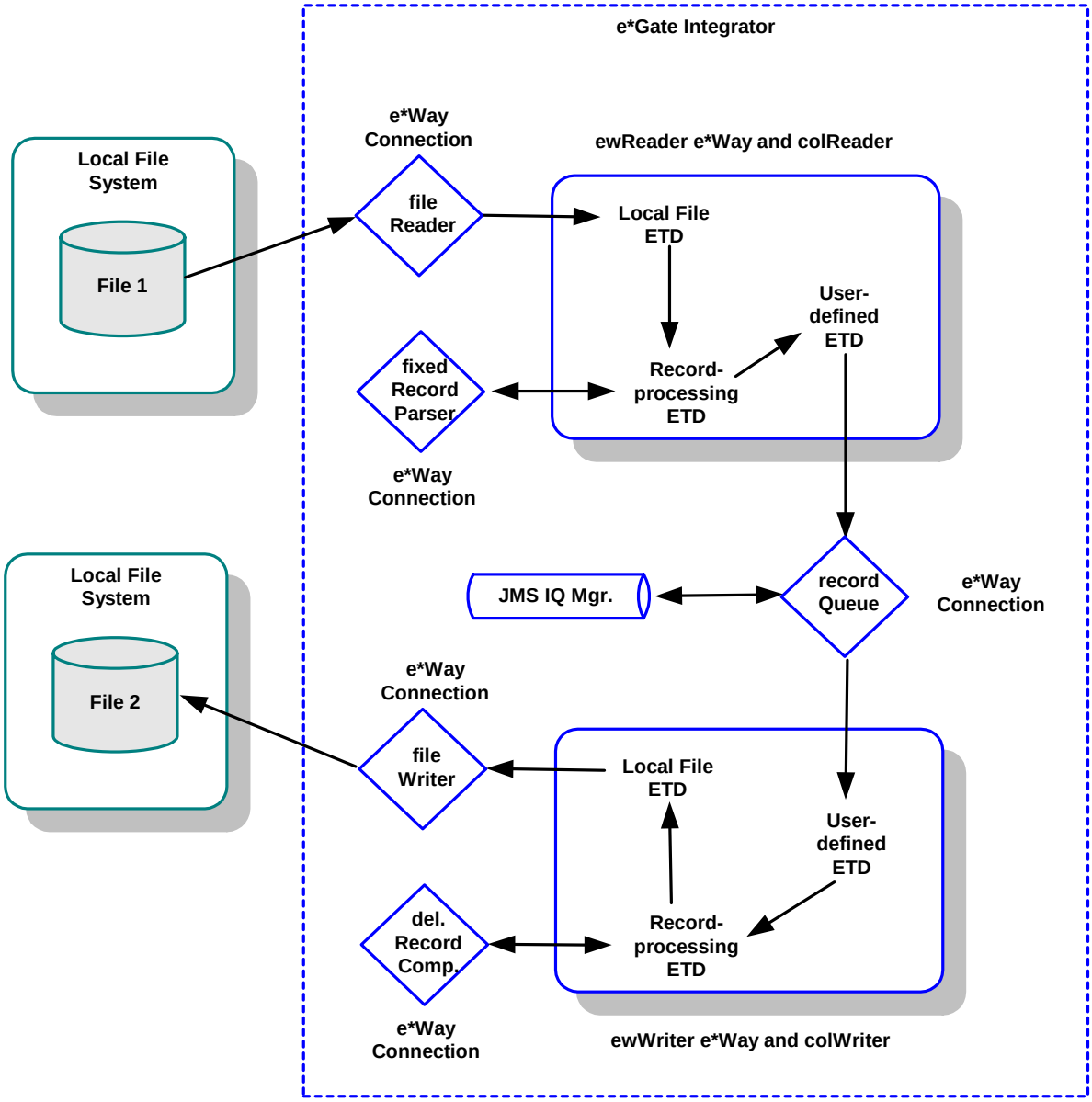
7.3.1 RPStreamingSample Schema Overview

This section provides an overview of the sample schema and how it operates. The name of this schema is RPStreamingSample, and it is contained in the import file **RPStreamingSample.zip**.

Schema Setup

Figure 46 on page 171 shows a diagram of the schema's general architecture. The arrows show the direction of data flow.

Figure 46 RPStreamingSample Schema Diagram



Schema Operation

This sample schema has the following input/output setup:

- **Input**: A simple data file, **fixed50.dat** (provided in a .**zip** file with sample) from a local file system.
- **Output**: A file with the same name, to the same location on a local file system.

This sample schema demonstrates the Batch e*Way's local file access and record-processing features using data-streaming Event Type Definition (ETD) links with the Resume Reading feature in the e*Way's XA mode (using GEOD).

Before Starting

Before you run this schema, make sure that the e*Way Connections have been configured properly. You must pay special attention to the directory and file names specifying the locations on the local file system.

Make sure that the sample input file has been extracted in the correct location, and the e*Gate system has the proper access rights.

Basic Setup

The sample has two Collaboration Rules demonstrating record parsing from a file and constructing files from records.

The first Collaboration Rule reads a file in batches using local-file data streaming with the Resume Reading feature of the local file ETD enabled. Each batch is processed using the record-processing ETD, which extracts fixed-length records. They are in turn converted to strings and posted to an e*Way Connection representing a JMS IQ Manager.

The second Collaboration Rule retrieves the strings from the JMS IQ Manager and writes them as delimited records to another file. A new file is created every time a maximum number of records is reached or when the JMS IQ Manager is empty.

The pre- and post-transfer commands in the local file e*Way Connection are set up in such a way that, after processing, the sample data file ends up with the same name. In this way, the schema becomes self-feeding with input data and runs continually. If you want the schema to process the file only once, select a different post-transfer file or directory name.

All local file and the JMS e*Way Connections are in the XA mode. The XA mode is optional and can be turned off using the corresponding e*Way Connections' XA-related configuration parameter.

Caution: *Disabling the XA mode can cause data loss or duplication.*

Sample Data

The file **RPStreamingSampleData.zip** contains a sample data file. Please extract this .**zip** file in your e*Gate directory. The sample schema configuration assumes that e*Gate has been installed in **C:\eGate**.

Note: *The purpose of this sample schema is only to demonstrate the features of the local file and record-processing ETDs, so there is no error-handling logic in the Collaboration Rules. See* **"Sample Schema: Basic FTP With Streaming" on page 129** *for a sample with error-handling logic.*

Additional Information

For more information on the features demonstrated in this sample schema see the following sections:

- **"ETD for Local File" on page 100**
- **"Streaming Data Between Components" on page 329**
- **"Guaranteed Exactly Once Delivery" on page 347**

Schema Components

The RPStreamingSample schema with data-streaming and record-processing implementations consists of the following main e*Gate components:

- **ewReader**: Inbound Multi-Mode e*Way that brings the local file into e*Gate.
- **ewWriter**: Outbound Multi-Mode e*Way that sends the file back to the local file system.
- **colReader**: Collaboration for the **ewReader** e*Way.
  - **crReader**: Collaboration Rule for **colReader**.
- **colWriter**: Collaboration for the **ewWriter** e*Way.
  - **crWriter**: Collaboration Rule for **colWriter**.
- **localhost_iqmgr**: Oracle SeeBeyond JMS IQ Manager.
- **fileReader**: File-reading (input) e*Way Connection for the **ewReader** e*Way.
- **fixedRecordParser**: Record-processing e*Way Connection for the **ewReader** e*Way.
- **delimitedRecordComposer**: Record-processing e*Way Connection for the **ewWriter** e*Way.
- **fileWriter**: File-writing (output) e*Way Connection for the **ewWriter** e*Way.
- **recordQueue**: e*Way Connection for the **localhost_iqmgr** IQ Manager.

7.3.2 Creating the RPStreamingSample Sample Schema

This section explains the basic steps for how to create the RPStreamingSample schema, including the data-streaming and record-processing features.

Creating a New Schema

This step is the same as for the BasicFtpSample schema. The name of the new schema is **RPStreamingSample**. Follow the procedures provided under **"Creating a New Schema" on page 132**.

Creating Event Types and ETDs

The e*Way installation provides the three **.xsc** files for this schema. You must create Event Types and associate them with ETDs as shown in Table 8.

Table 8 RPStreamingSample Schema ETDs

Event Type	Type of ETD	ETD File Name
file	Local file ETD	LocalFileETD.xsc
record	Record-processing ETD	BatchRecordETD.xsc
transport	User-defined ETD	TransportString.xsc

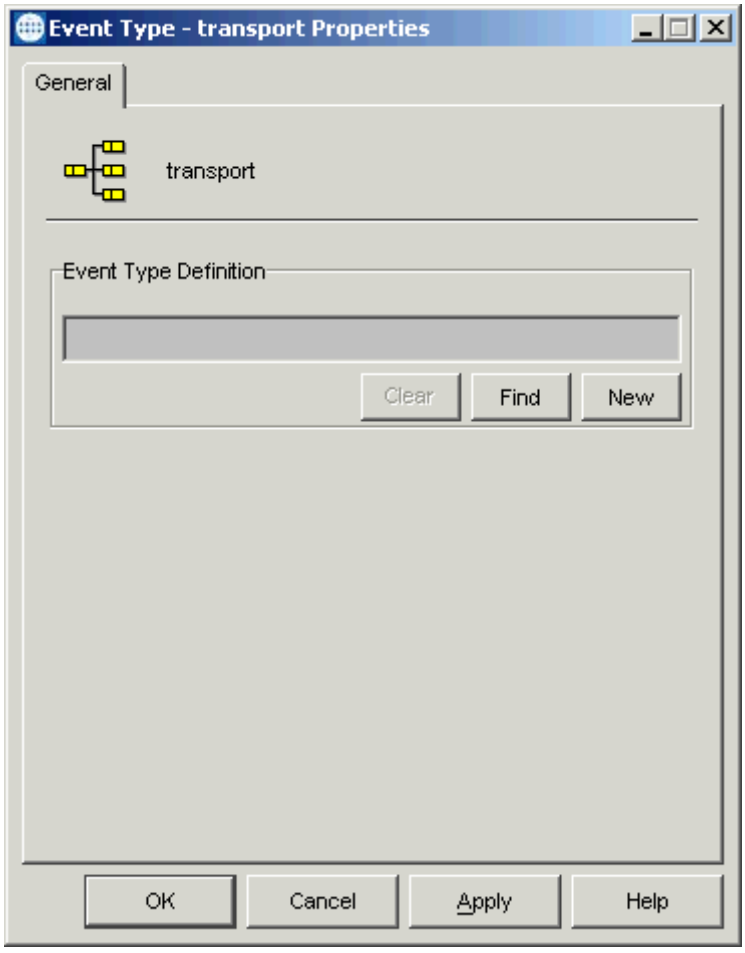
Event Types and ETDs

See **"Creating Event Types and ETDs" on page 133** for instructions on how to create Event Types, locate the ETDs, and how to associate each Event Type with its ETD. The **transport** Event Type requires a user-defined ETD.

To create the transport Event Type and ETD

1. Highlight the **Event Type** folder on the **Components** tab of the e*Gate Schema Designer.
2. On the palette, click the icon to create a new Event Type.
3. Enter the name of the Event Type (**transport**), then click **OK**.
4. Select the new **Event Type**, then right-click to edit its properties.
5. The **Event Type Properties** dialog box appears (see Figure 47).

Figure 47 transport Event Type Properties Dialog Box



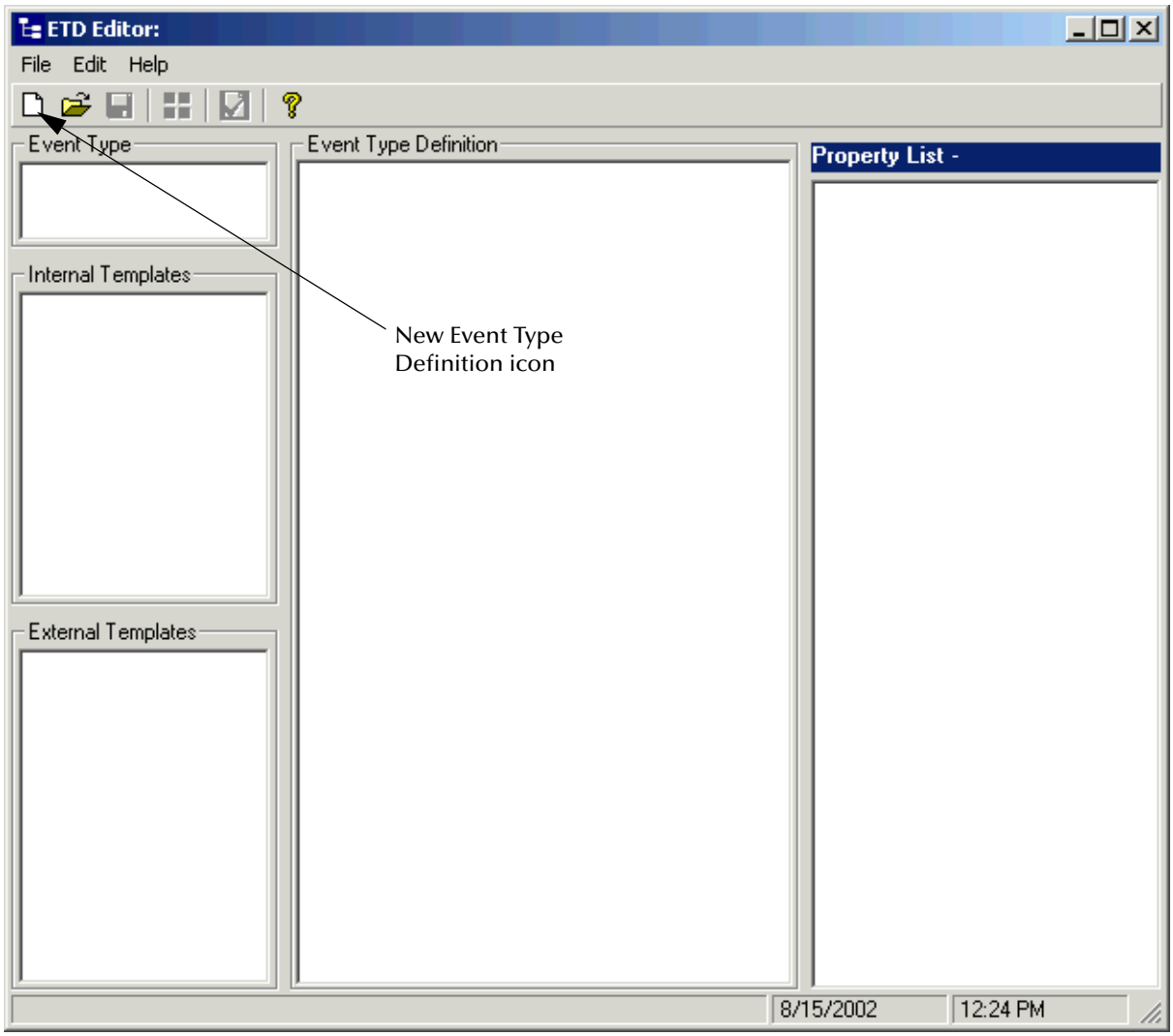
Because this Event Type is for a user-defined ETD, you must use the ETD Editor feature of the Schema Designer to create this ETD.

6. From the Schema Designer menu bar, select **Options** and click **Default Editor**. For this schema, set the default to **Java**.

7 Click the icon in the toolbar for the ETD Editor.

The ETD Editor Main window appears (see **Figure 48 on page 175**).

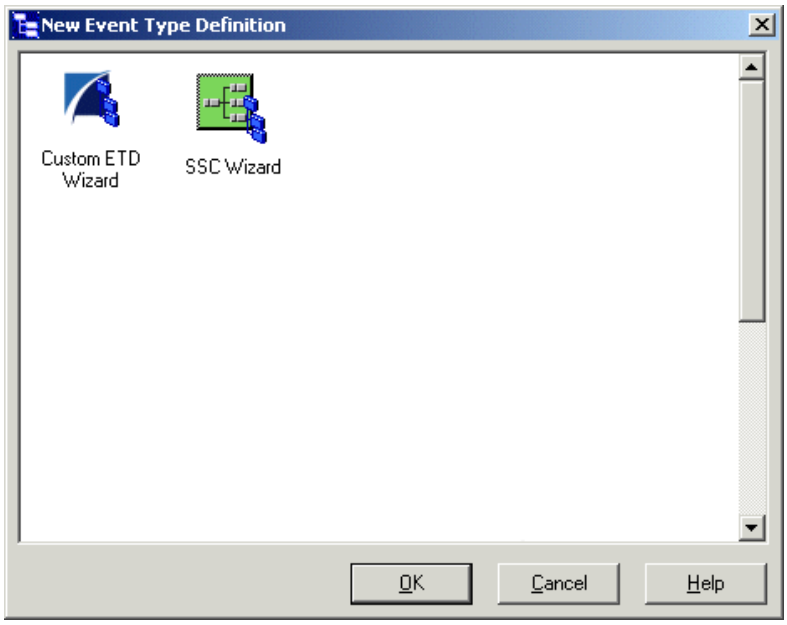
Figure 48 ETD Editor Main Window



8 Click the **New Event Type Definition** icon in the window's tool bar.

The **New Event Type Definition** dialog box appears (see **Figure 49 on page 176**).

Figure 49 New Event Type Definition Dialog Box



9 Double-click the **Custom ETD Wizard** icon.

The **Introduction** dialog box for the Custom ETD wizard appears (see Figure 50).

Figure 50 Custom ETD Wizard: Introduction

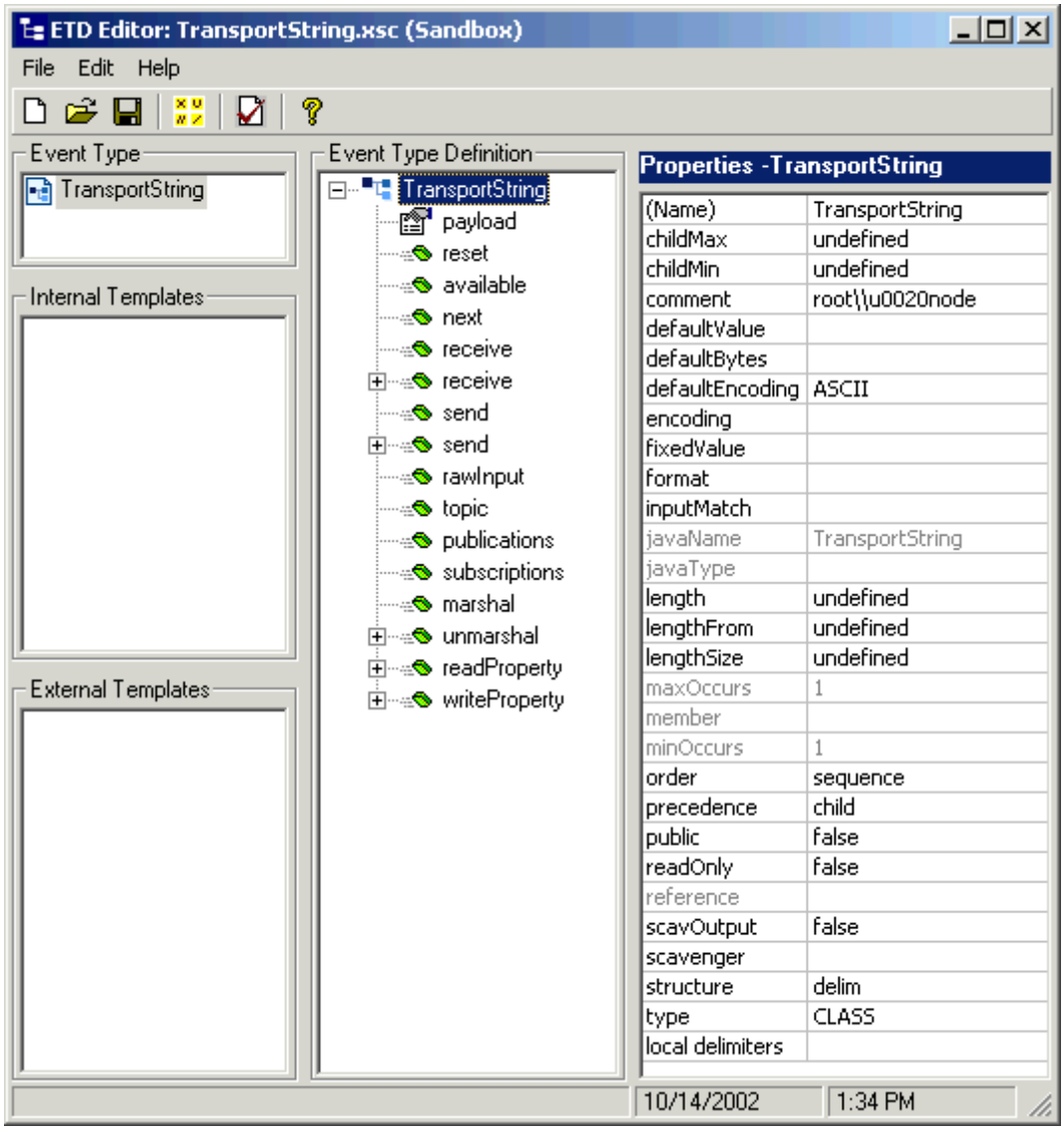
10 Follow the instructions given by the wizard to create a new ETD and name its root node **TransportString**. Be sure to give it the appropriate Java package name.

Note: *For complete instructions on how to use the Custom ETD wizard to create a new ETD, see the* ***e*Gate Integrator User's Guide****.*

The contents of the file appear in the ETD Editor window.

11 Assign the new ETD the properties shown in Figure 51.

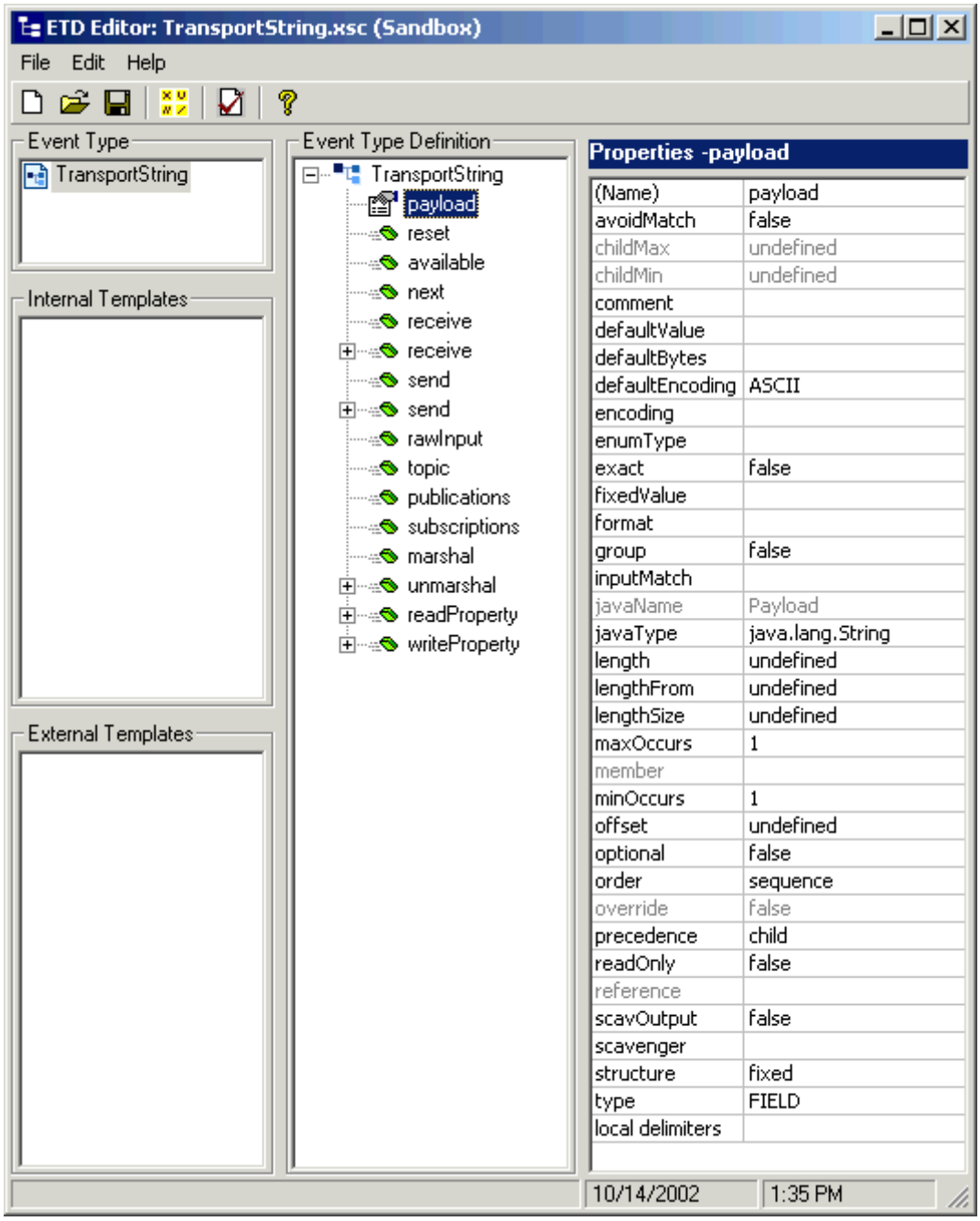
Figure 51 TransportString.xsc in ETD Editor: Root Node



Note: *For complete instructions on how to use the ETD Editor, see the* ***e*Gate Integrator User's Guide****.*

12 Add **payload** as a child node under the root node. Be sure that you give this node the properties shown in Figure 52.

Figure 52 TransportString.xsc in ETD Editor: ByteData Node



13 Finish building the ETD until it looks like the one shown in Figure 52.

14 When you are finished editing the ETD, save your changes and close the ETD Editor. Be sure to create the ETD (**TransportString.xsc**) in the **client\etd** directory.

15 From the Schema Designer, open the **Event Type Properties** dialog box for the **transport** Event Type and associate it with the **TransportString.xsc** ETD you created (see **Figure 47 on page 174**).

16 When you are finished with the dialog box, click **OK** to close it and save your changes.

Creating and Configuring e*Ways

You must create the following Multi-Mode e*Ways:

- Inbound: **ewReader** with **ewReader.cfg** configuration file.
- Outbound: **ewWriter** with **ewWriter.cfg** configuration file.

For details on how to create and configure e*Ways, see **"Creating and Configuring e*Ways" on page 135**.

Figure 53 and **Figure 54 on page 180** show the **e*Way Properties** dialog boxes for the **ewReader** and **ewWriter** e*Ways. Configure and name these e*Ways as shown in both of the figures.

Figure 53 ewReader e*Way Properties Dialog Box

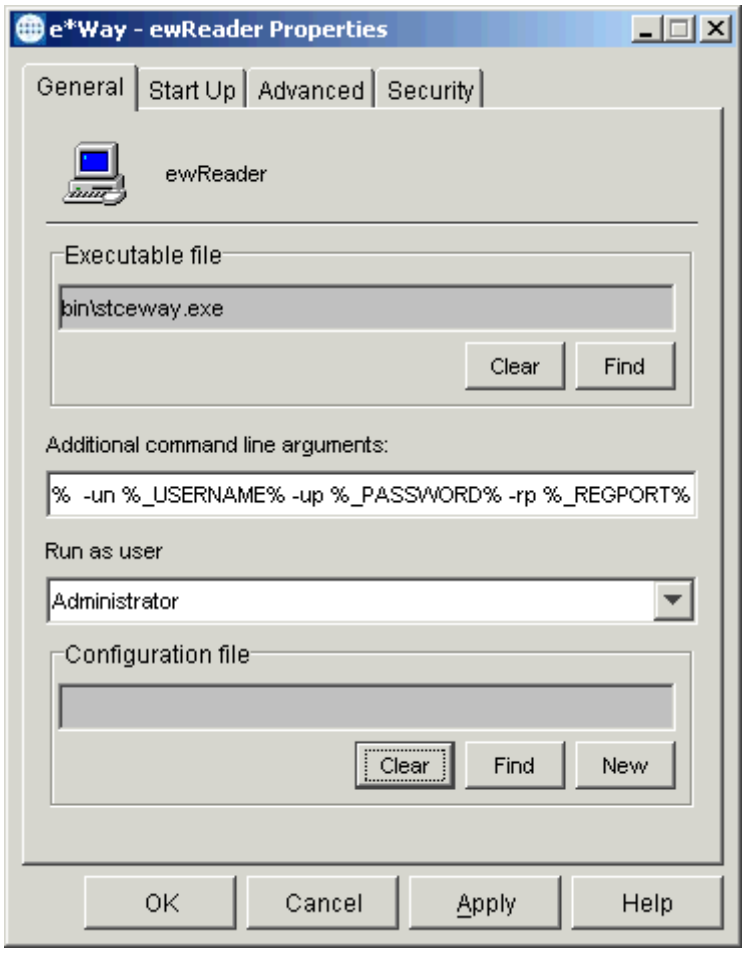
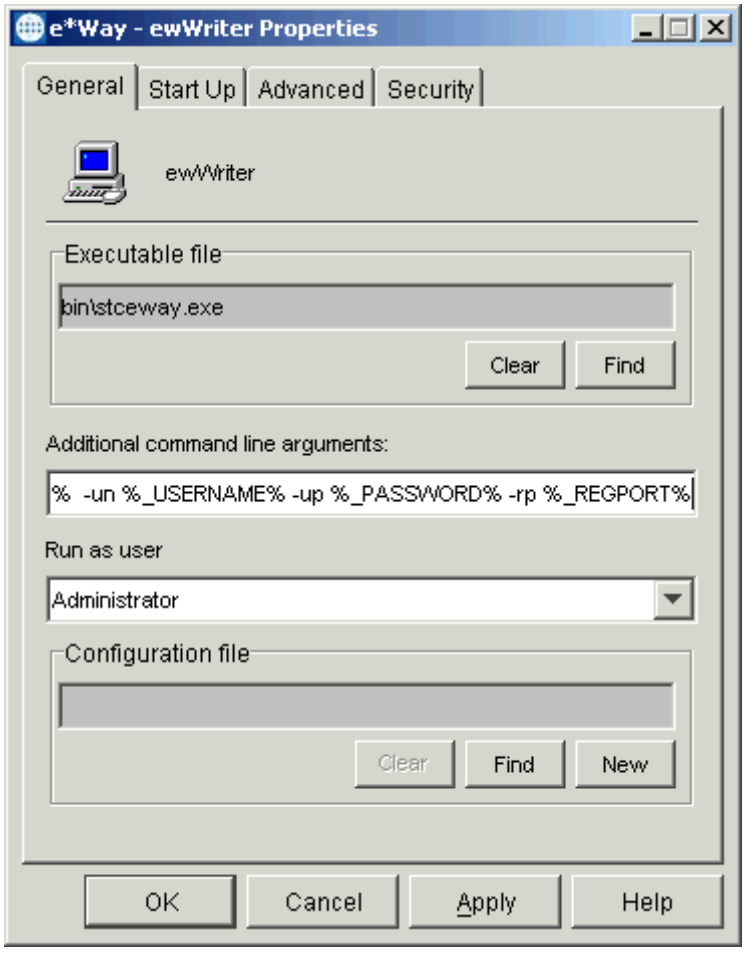


Figure 54 ewWriter e*Way Properties Dialog Box



Creating and Configuring e*Way Connections

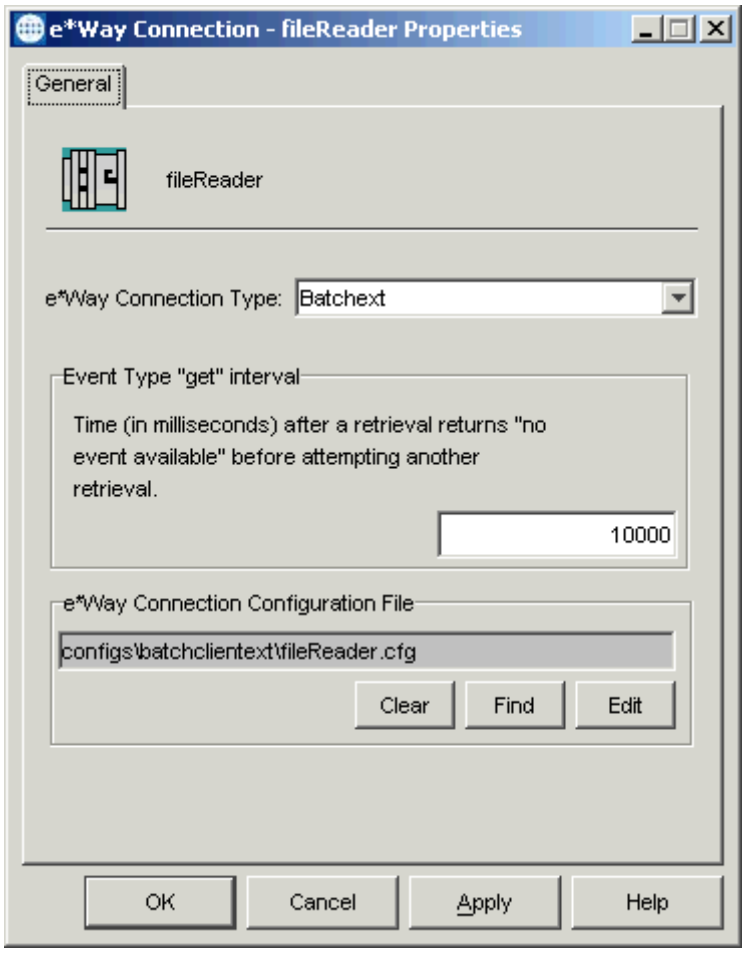
The e*Way Connection configuration file contains the connection information needed to communicate with the local file system and the JMS IQ Manager.

To create and configure the fileReader e*Way Connection

1. Highlight the **e*Way Connection** folder on the **Components** tab of the e*Gate Schema Designer.
2. On the palette, click the icon to create a new e*Way Connection.
3. Enter the name of the e*Way Connection (**fileReader**), then click **OK**.
4. Select the new **e*Way Connection**, then right-click to edit its properties.

5 When the **e*Way Connection Properties** dialog box opens, select **Batchext** from the **e*Way Connection Type** drop-down menu (see Figure 55).

Figure 55 fileReader e*Way Connection Properties Dialog Box

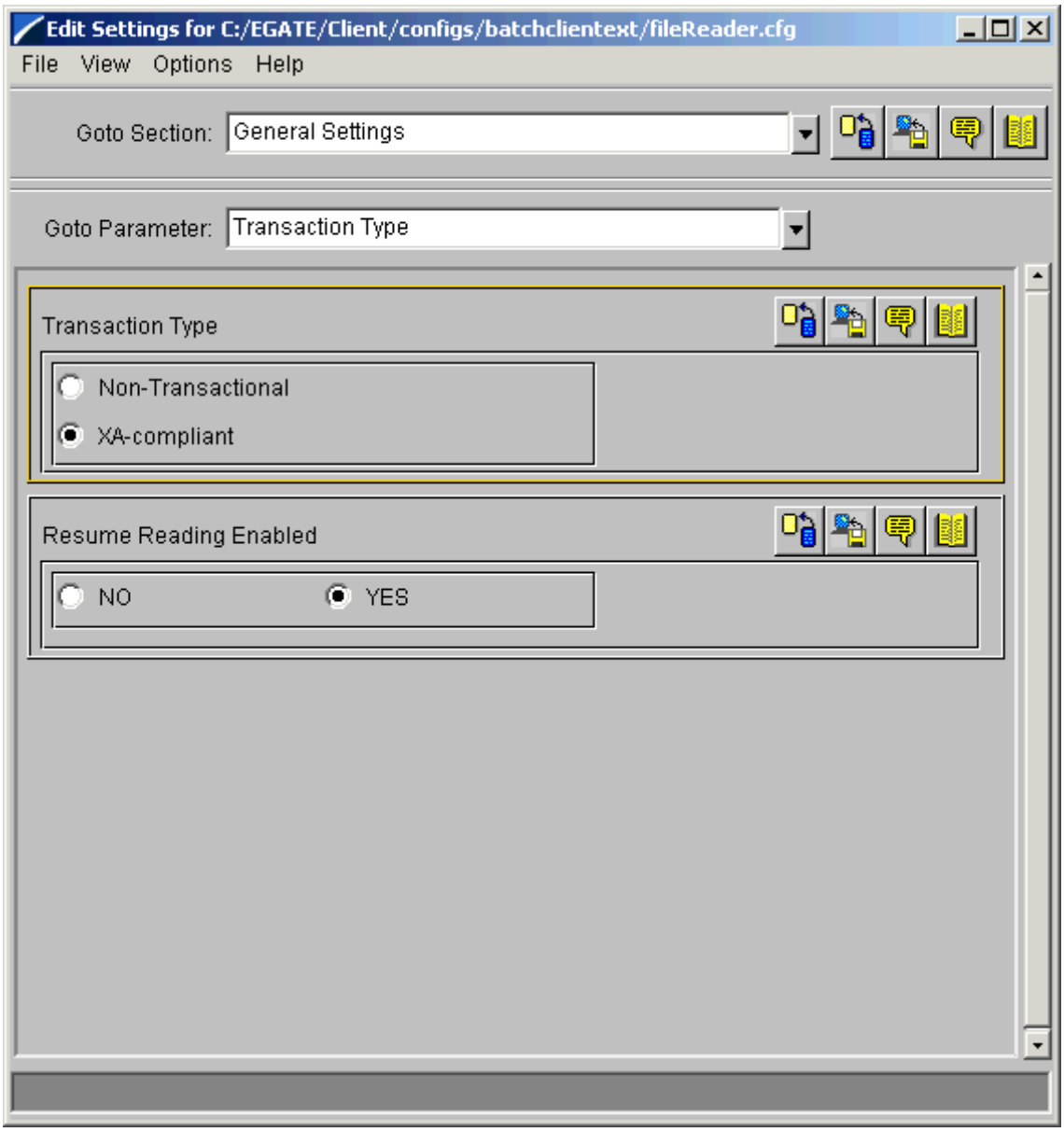


Configure the e*Way Connection properties as shown in the previous figure.

6 Under **e*Way Connection Configuration File**, click **New** (where the **Edit** button is in the previous figure). Select the **LocalFileETD**.

The e*Way Configuration Editor Main window opens (see **Figure 56 on page 182**).

Figure 56 e*Way Configuration Editor: fileReader General Settings



7 Select the following parameters:

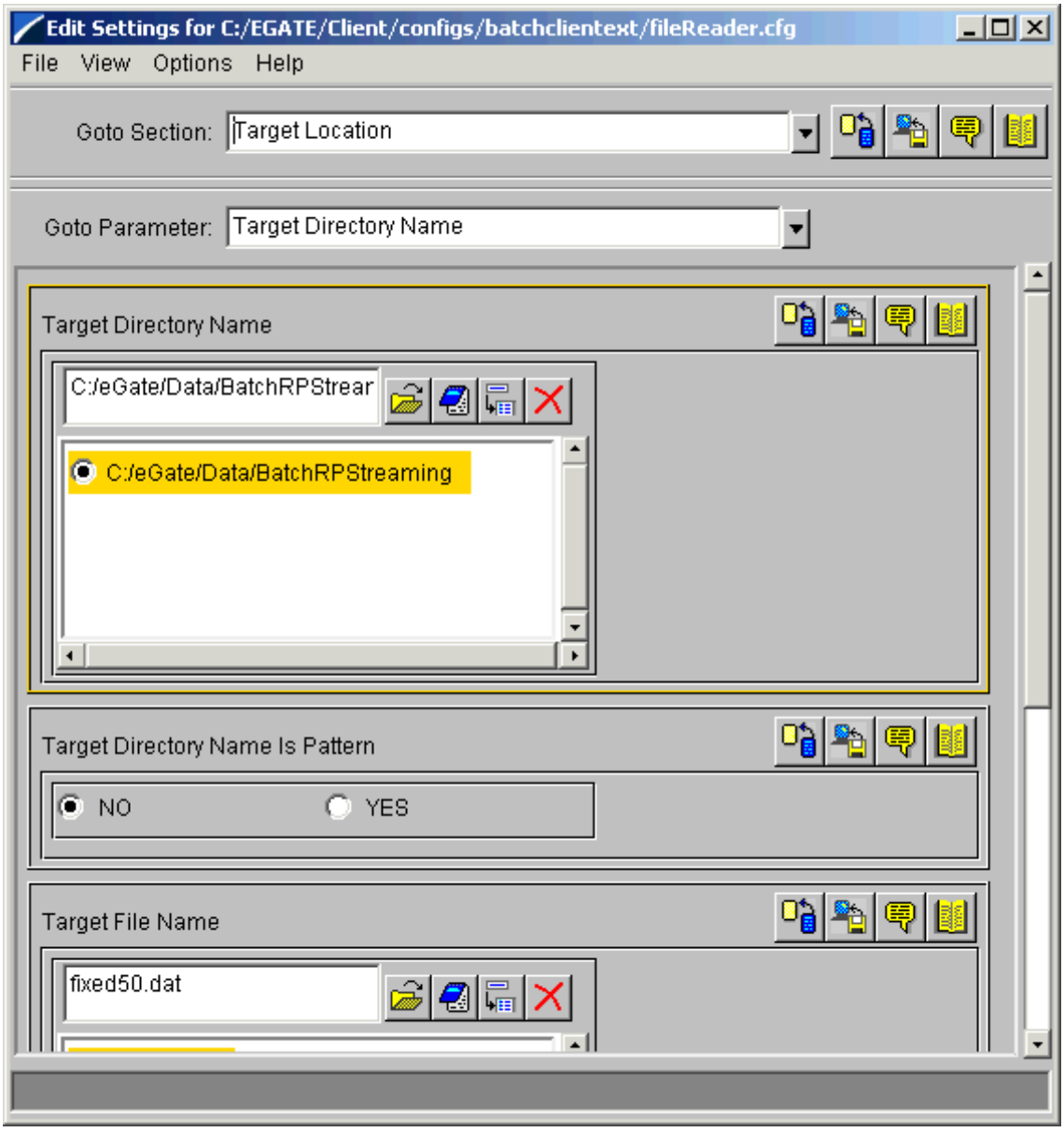
- **XA-compliant** for the **Transaction Type**; this parameter enables GEOD for the e*Way Connection and places it in the XA mode.
- **Yes** for **Resume Reading**; this parameter enables the Resume Reading feature for the e*Way Connection.

Note: *See the **e*Gate Integrator User's Guide** for complete information on how to use the e*Way Configuration Editor.*

8 Select the **Target Location** settings (see Figure 57). Under this section, set the appropriate parameters for the target directory and file name as shown in the figure. Enter the information that corresponds to your local file system.

In addition, do not use pattern matching or appending.

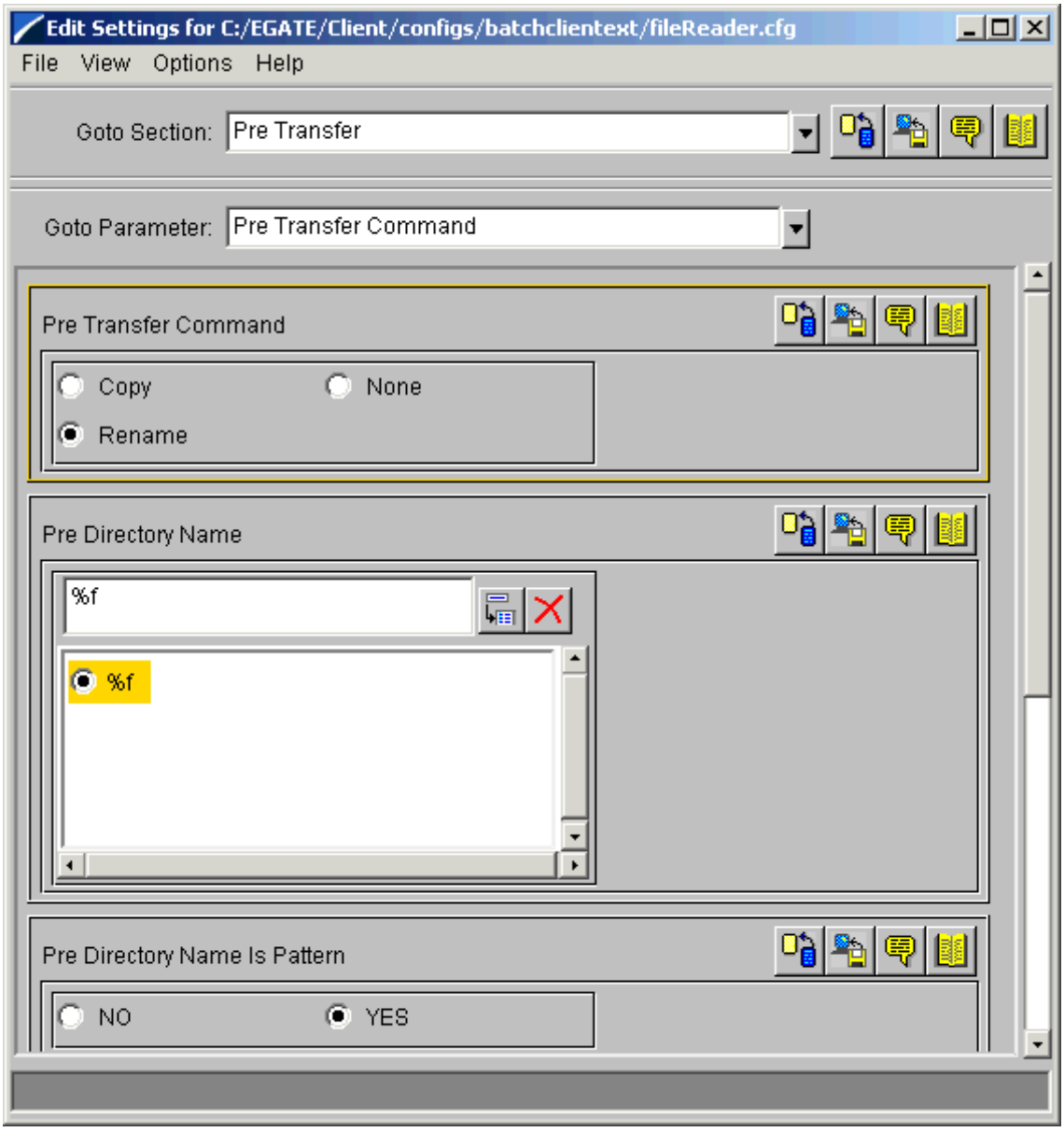
Figure 57 e*Way Configuration Editor: fileReader Target Location Settings



9 Select the **Pre Transfer** settings (see Figure 58). Under this section, set the appropriate parameters for the commands you want to execute before the file transfer as shown in the figure.

In addition, enter **%f.proc** for the pre file name and use pattern matching.

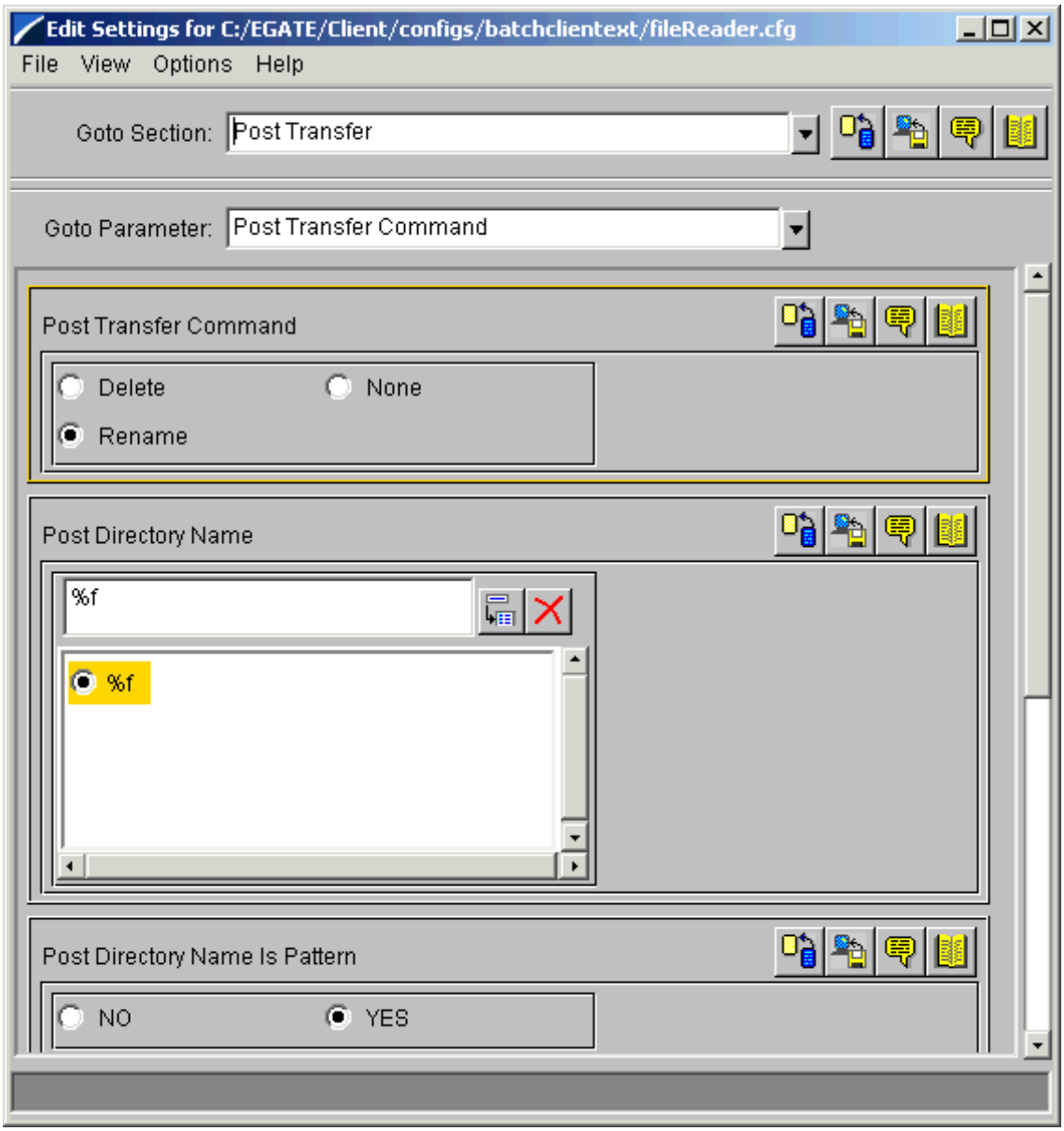
Figure 58 e*Way Configuration Editor: fileReader Pre Transfer Settings



10 Select the **Post Transfer** settings (see Figure 59). Under this section, set the appropriate parameters for the commands you want to execute after the file transfer as shown in the figure.

In addition, enter **%f** for the post file name and use pattern matching for the file name as well.

Figure 59 e*Way Configuration Editor: fileReader Post Transfer Settings



11 For the rest of the parameters, use the defaults. See **"LocalFileETD: Configuration Parameters" on page 59** for details.

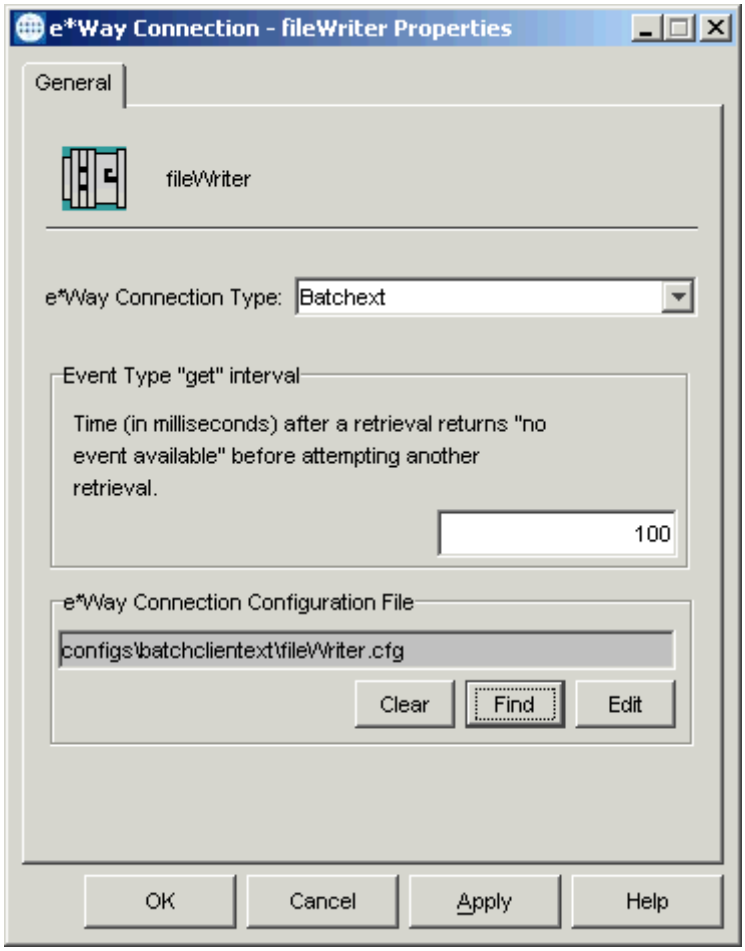
12 When you are finished, save the **.cfg** file as the name of the e*Way Connection, close the e*Way Configuration Editor, and promote the file to run time.

13 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the fileWriter e*Way Connection

1. Follow the same procedures as you did previously to create the new e*Way Connection and name the new component **fileWriter**.
2. When the **e*Way Connection Properties** dialog box opens, select **Batchext** from the **e*Way Connection Type** drop-down menu (see Figure 60).

Figure 60 fileWriter e*Way Connection Properties Dialog Box

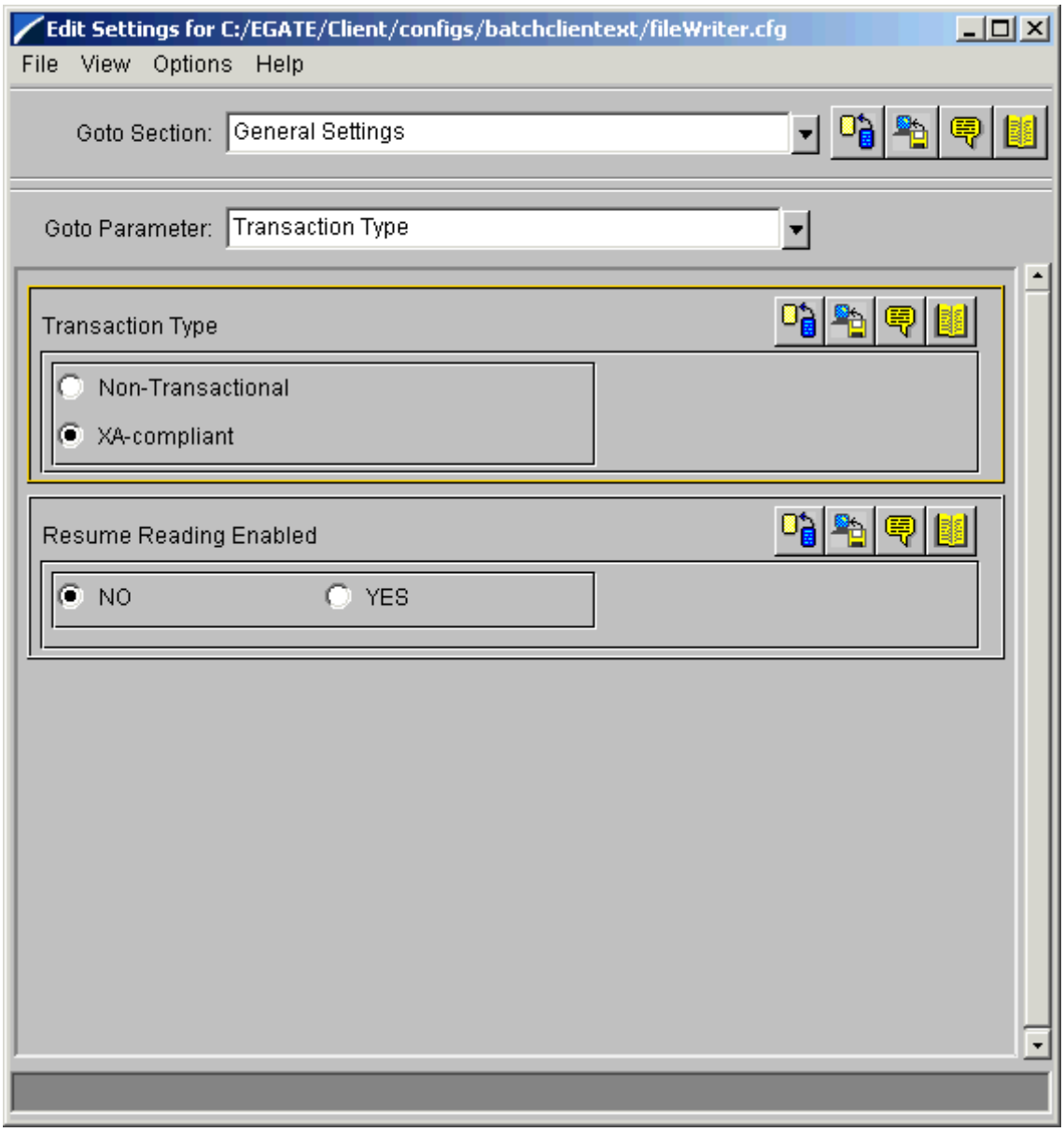


Configure the e*Way Connection properties as shown in the previous figure.

3. Under **e*Way Connection Configuration File**, click **New** (where the **Edit** button is in the previous figure). Select the **LocalFileETD**.

   The e*Way Configuration Editor Main window opens (see **Figure 61 on page 187**).

Figure 61 e*Way Configuration Editor: fileWriter General Settings



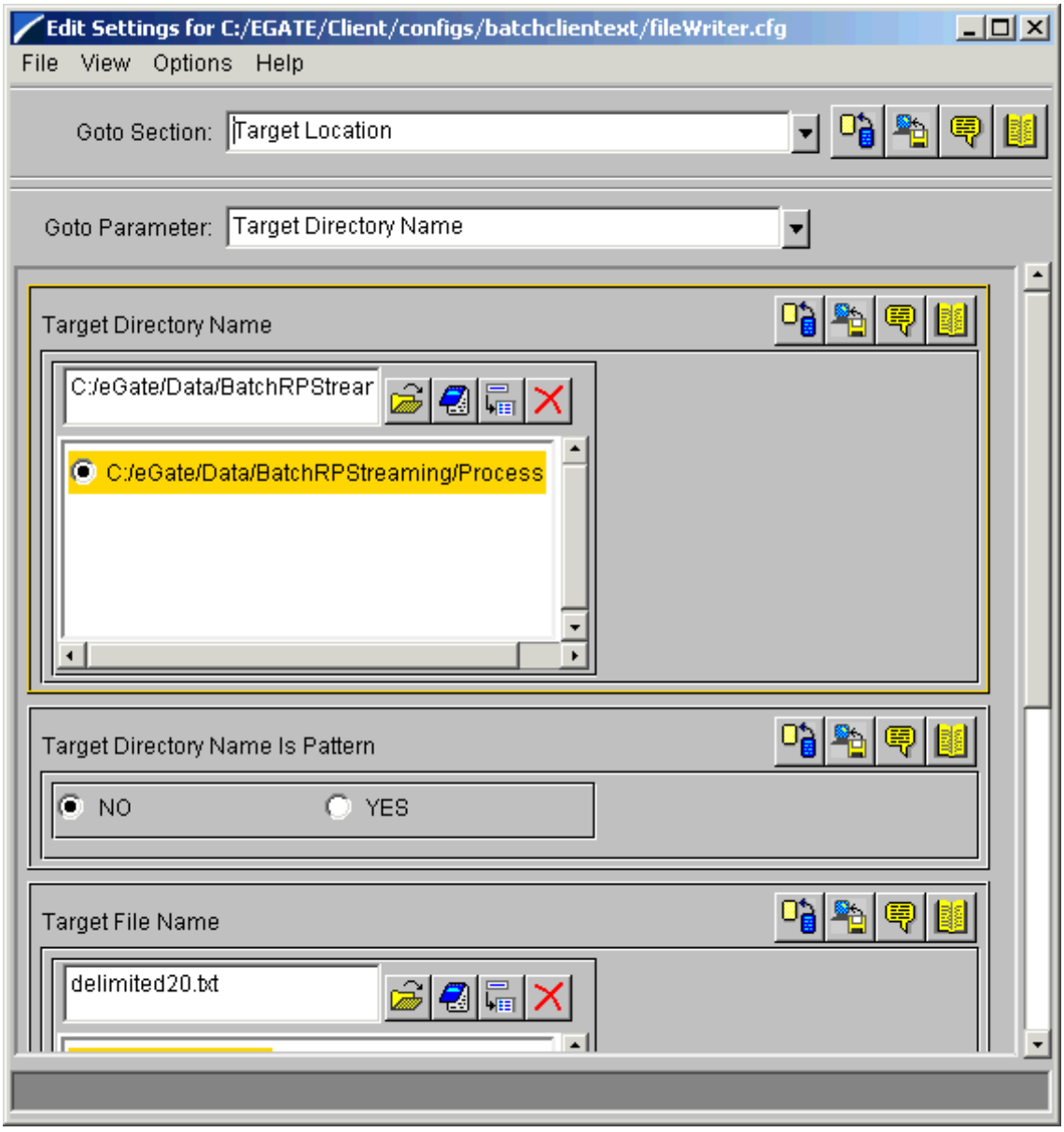
4 Select the following parameters:

- **XA-compliant** for the **Transaction Type**; this parameter enables GEOD for the e*Way Connection and places it in the XA mode.
- **No** for **Resume Reading**; you do not want to enable the Resume Reading feature for the e*Way Connection.

5 Select the **Target Location** settings (see Figure 62). Under this section, set the appropriate parameters for the target directory and file name as shown in the figure. Enter the information that corresponds to your local file system.

In addition, do not use pattern matching or appending.

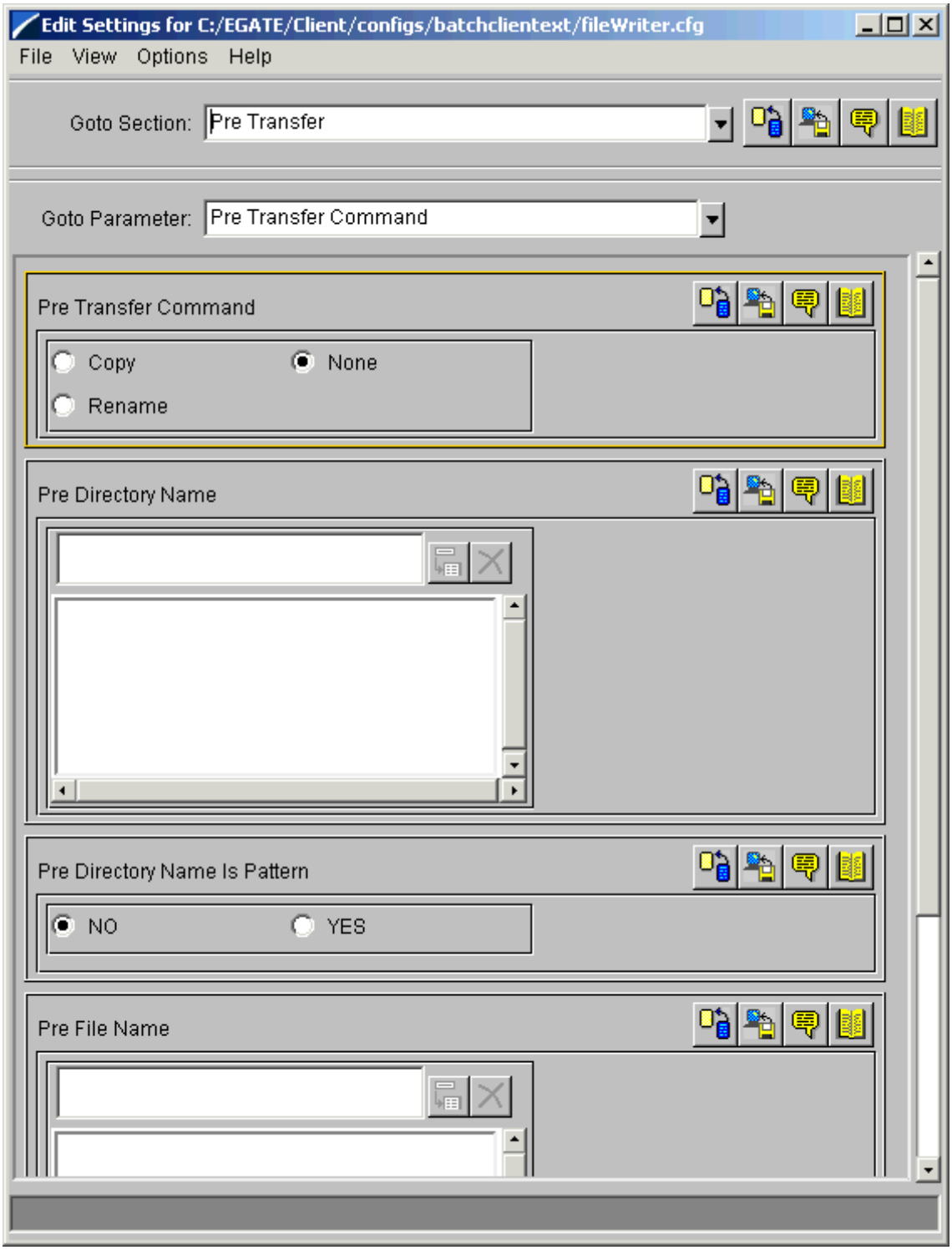
Figure 62 e*Way Configuration Editor: fileWriter Target Location Settings



6 Select the **Pre Transfer** settings (see Figure 63 on page 189). Under this section, set the appropriate parameters for the commands you want to execute before the file transfer as shown in the figure.

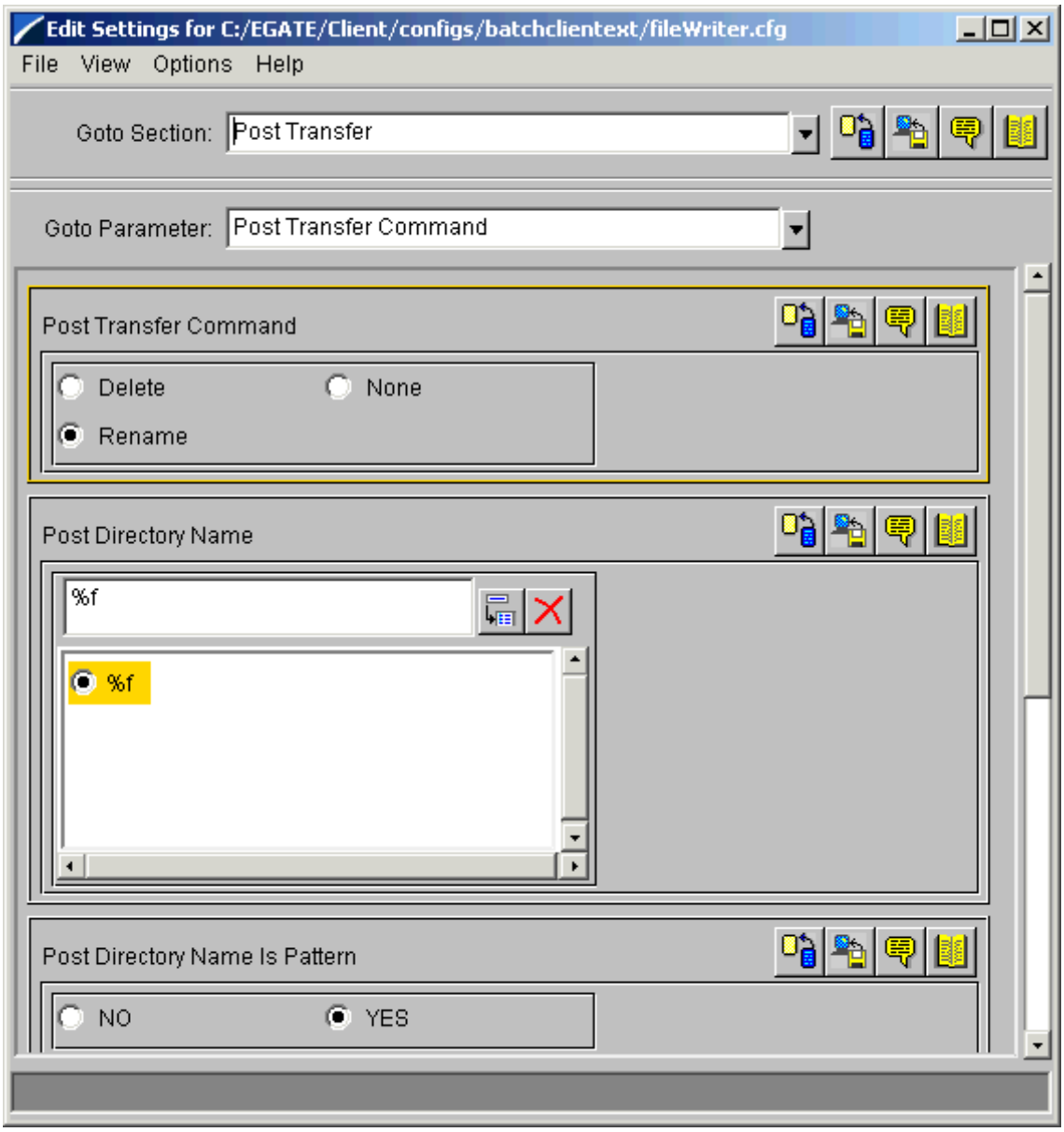
In addition, make sure that there is no pre file name, and be sure **Pre File Name Is Pattern** is set to **NO**.

Figure 63 e*Way Configuration Editor: fileWriter Pre Transfer Settings



7 Select the **Post Transfer** settings (see Figure 59). Under this section, set the appropriate parameters for the commands you want to execute after the file transfer as shown in the figure. In addition, enter **%#-%f** for the post file name and use pattern matching.

Figure 64 e*Way Configuration Editor: fileWriter Post Transfer Settings



8 For the rest of the parameters, use the defaults. See **"LocalFileETD: Configuration Parameters" on page 59** for details.

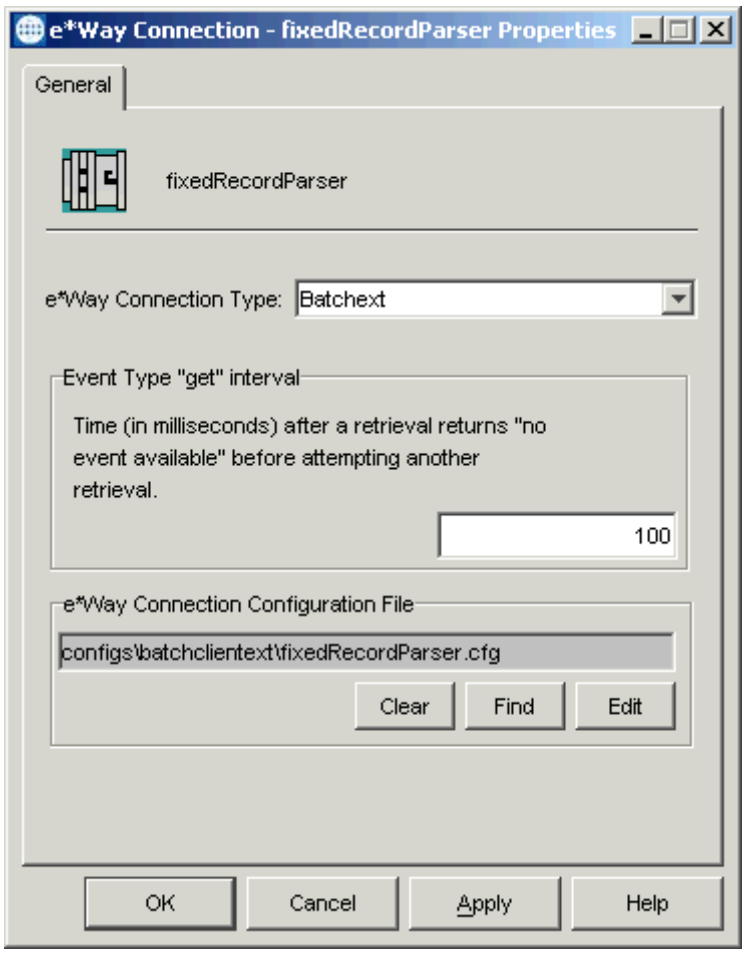
9 When you are finished, save the **.cfg** file as the name of the e*Way Connection, close the e*Way Configuration Editor, and promote the file to run time.

10 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the fixedRecordParser e*Way Connection

1. Follow the same procedures as you did previously to create the new e*Way Connection and name the new component **fixedRecordParser**.
2. When the **e*Way Connection Properties** dialog box opens, select **Batchext** from the **e*Way Connection Type** drop-down menu (see Figure 65).

Figure 65 fixedRecordParser e*Way Connection Properties Dialog Box



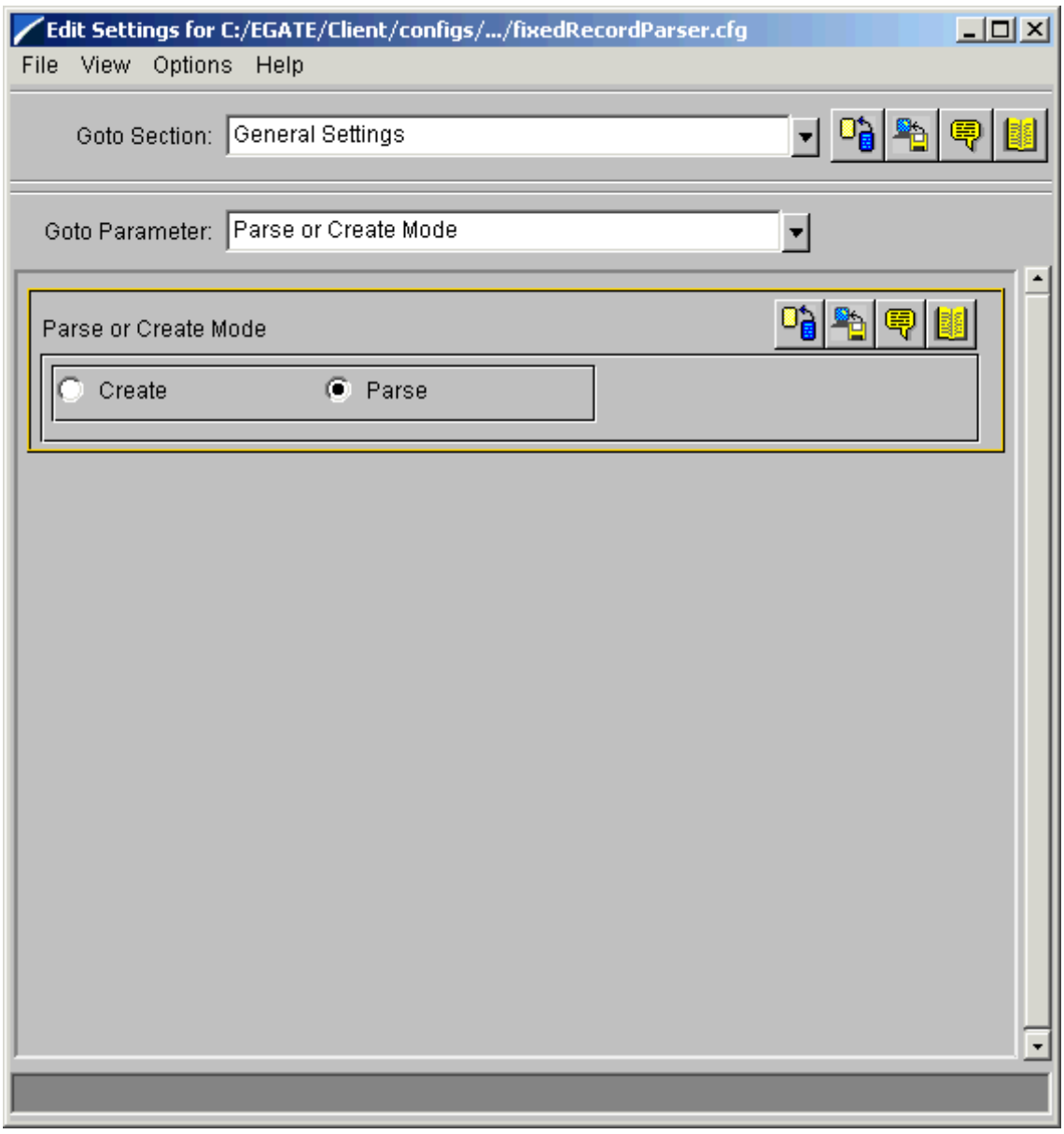
Configure the e*Way Connection properties as shown in the previous figure.

3. Under **e*Way Connection Configuration File**, click **New** (where the **Edit** button is in the previous figure). Select the **BatchRecordETD**. The e*Way Configuration Editor appears.

Note: *See* **"BatchRecordETD: Configuration Parameters" on page 32** *for details on the record-processing ETD's configuration.*

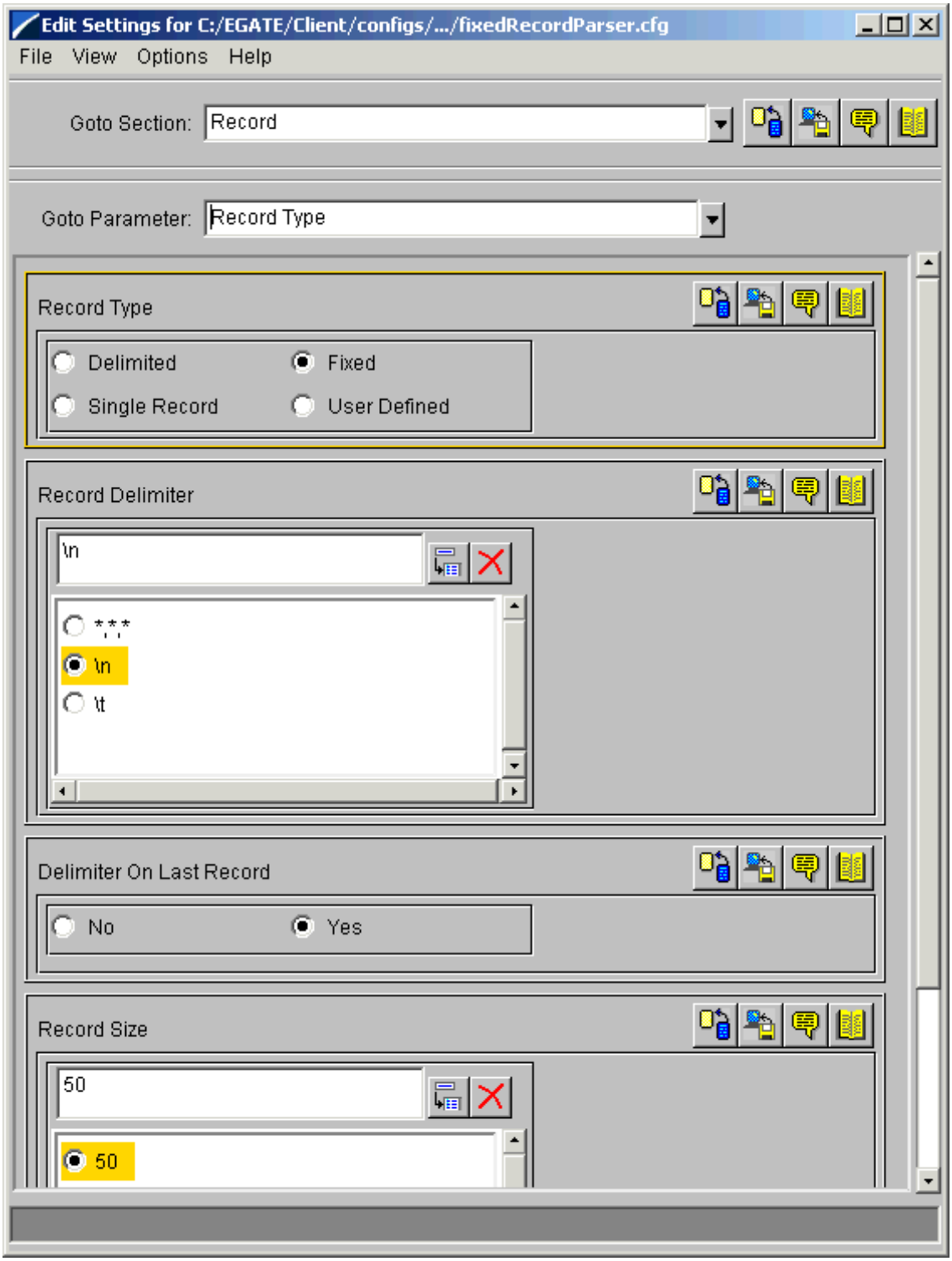
4 Under the **General Settings** section, set the **Parse or Create Mode** to **Parse** (see Figure 66).

Figure 66 e*Way Configuration Editor: fixedRecordParser General Settings



5 Under the **Record** section, set the parameters as shown in Figure 67.

Figure 67 e*Way Configuration Editor: fixedRecordParser General Settings



For the rest of the parameters, use the defaults.

6 When you are finished, save the **.cfg** file as the name of the e*Way Connection, close the e*Way Configuration Editor, and promote the file to run time.

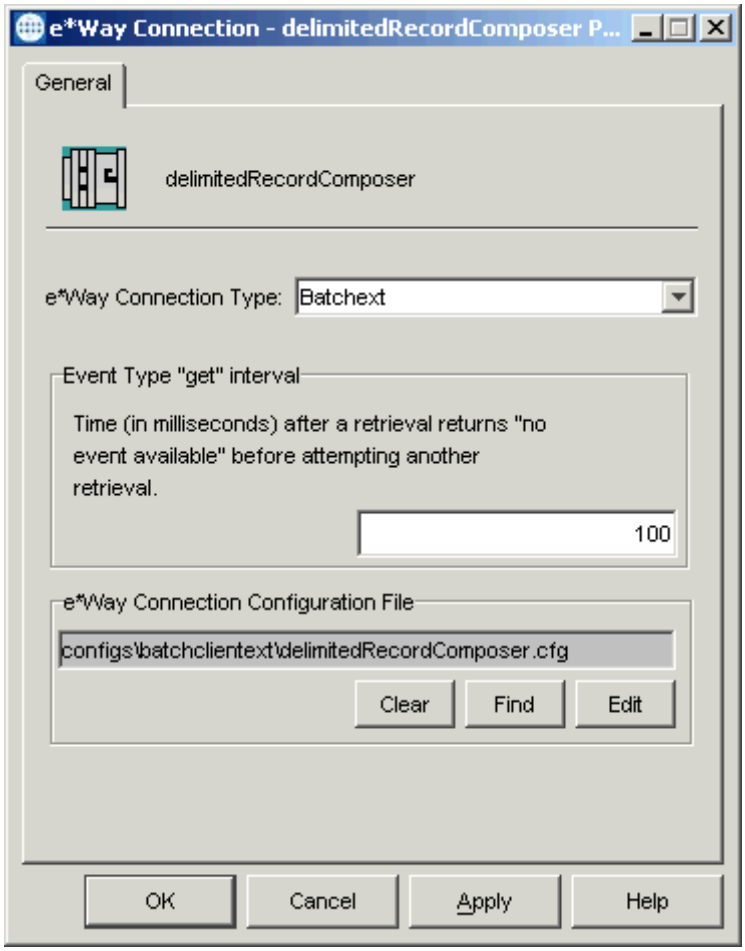
7 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the delimitedRecordComposer e*Way Connection

1 Follow the same procedures as you did previously to create the new e*Way Connection and name the new component **delimitedRecordComposer**.

2 When the **e*Way Connection Properties** dialog box opens, select **Batchext** from the **e*Way Connection Type** drop-down menu (see Figure 68).

Figure 68 delimitedRecordComposer e*Way Connection Properties Dialog Box

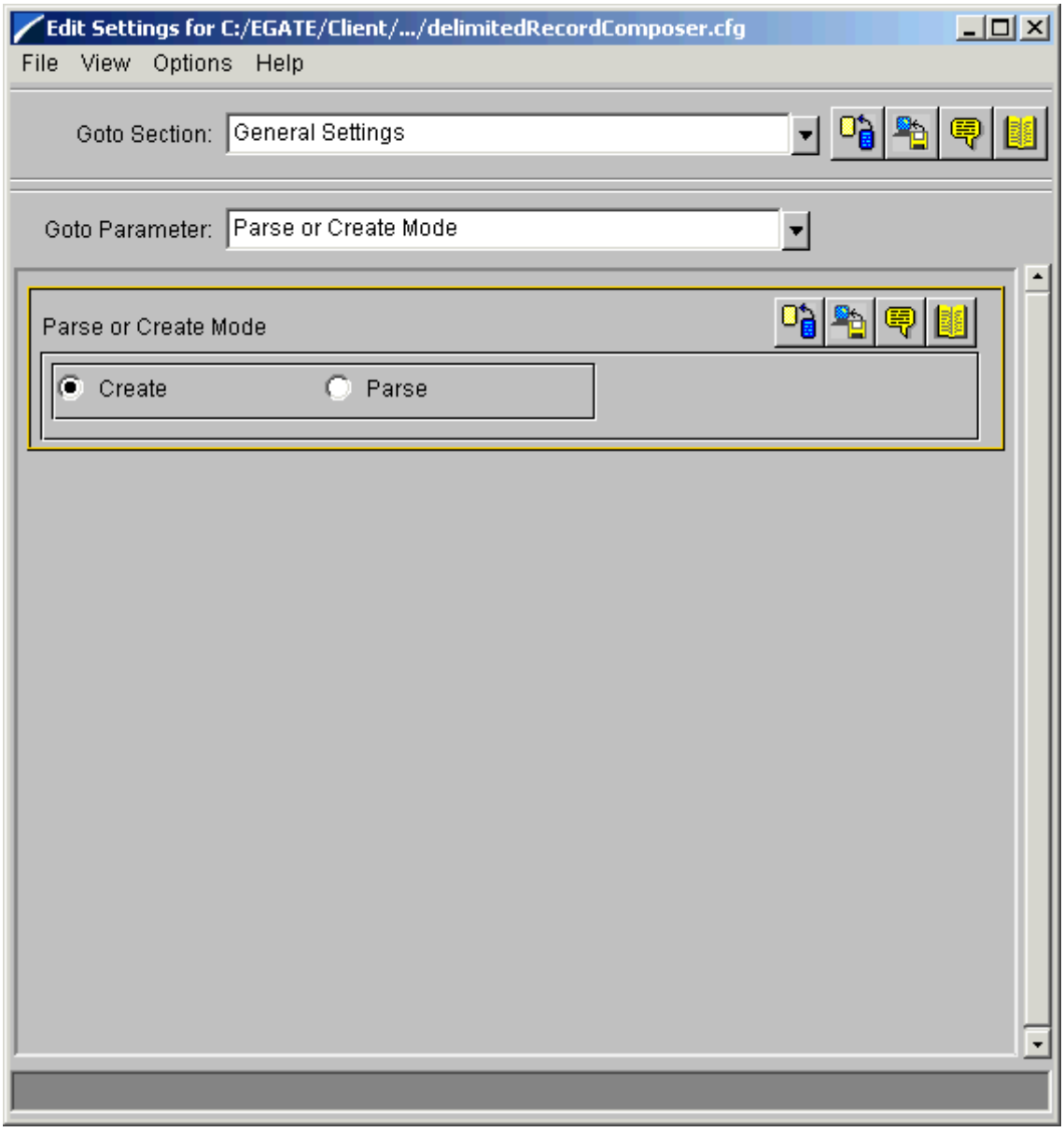


Configure the e*Way Connection properties as shown in the previous figure.

3 Under **e*Way Connection Configuration File**, click **New** (where the **Edit** button is in the previous figure). Select the **BatchRecordETD**. The e*Way Configuration Editor appears.

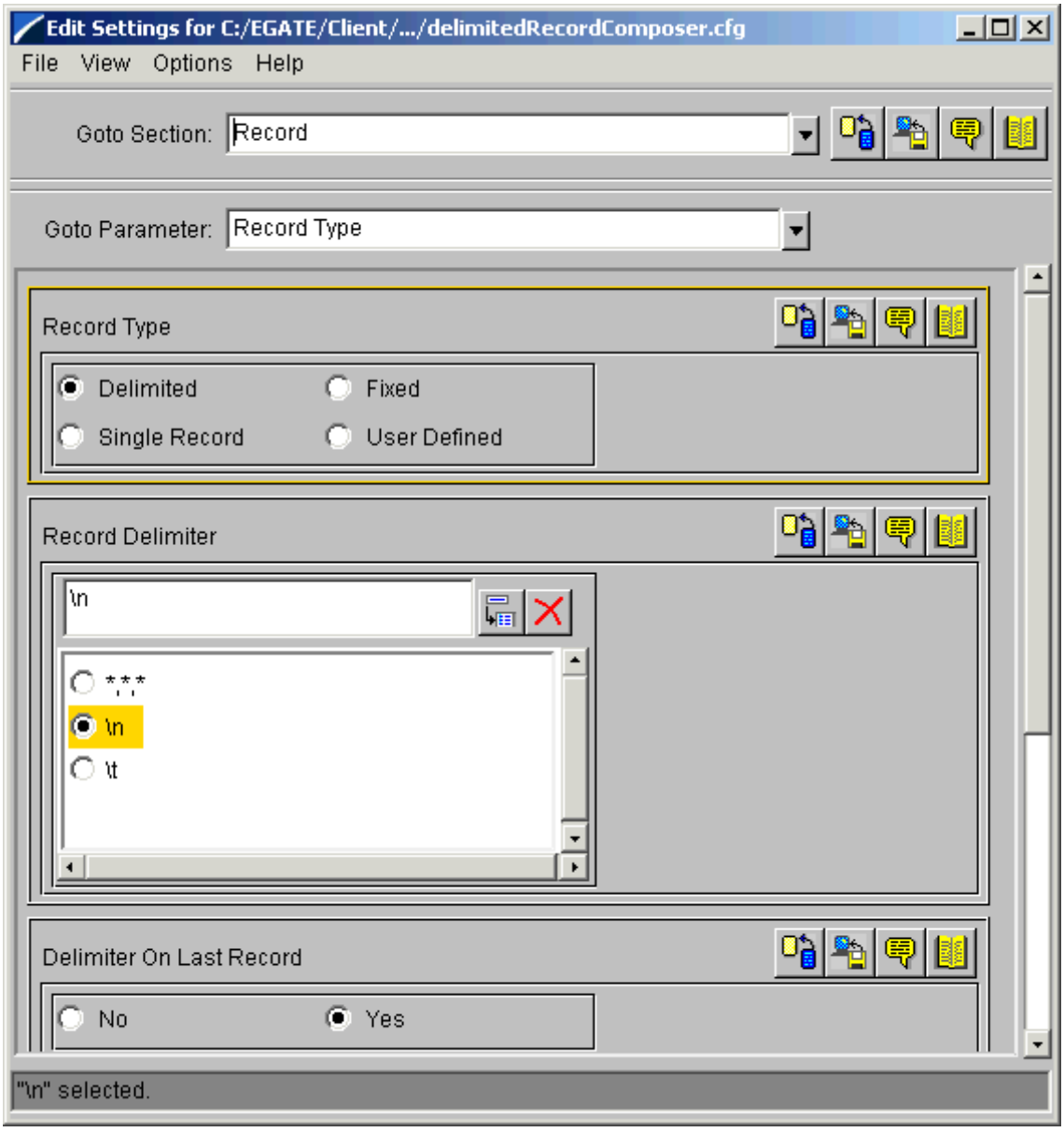
4 Under the **General Settings** section, set the **Parse or Create Mode** to **Create** (see Figure 69).

Figure 69 e*Way Configuration Editor: delimitedRecordComposer General Settings



5 Under the **Record** section, set the parameters as shown in Figure 70. Leave the **Record Size** blank; you do not need to set this parameter for delimited records.

Figure 70 e*Way Configuration Editor: delimitedRecordComposer General Settings



For the rest of the parameters, use the defaults.

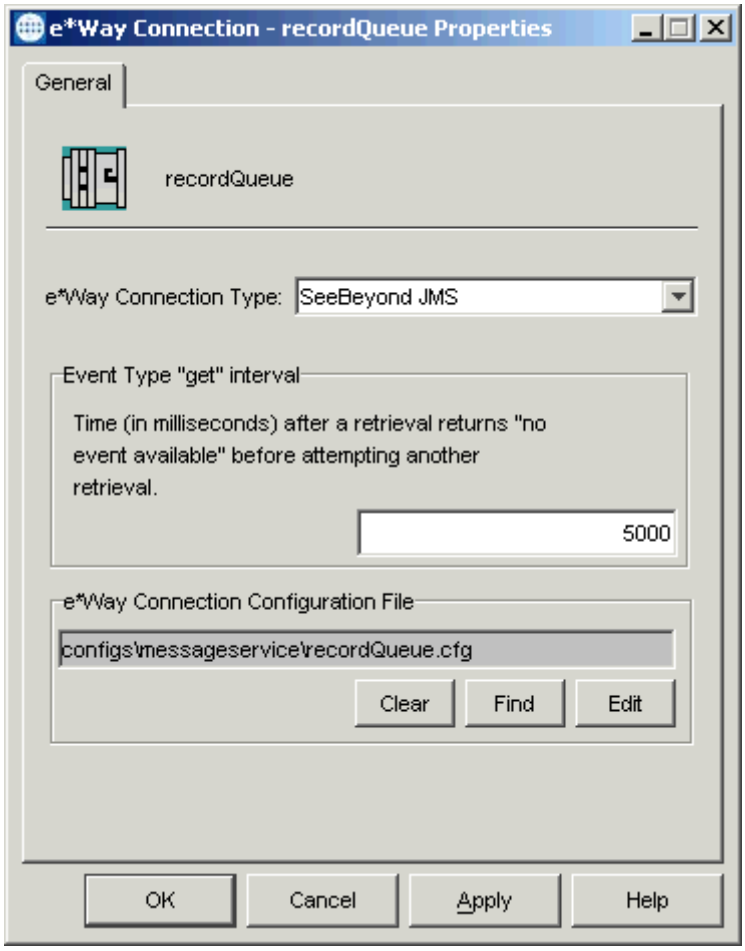
6 When you are finished, save the **.cfg** file as the name of the e*Way Connection, close the e*Way Configuration Editor, and promote the file to run time.

7 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the recordQueue e*Way Connection

1 Follow the same procedures as you did previously to create the new e*Way Connection *except* that you do not need to select an ETD. Name the new component **recordQueue** (see Figure 71).

Figure 71 recordQueue e*Way Connection Properties Dialog Box



Configure the e*Way Connection properties as shown in the previous figure. Be sure to select **SeeBeyond JMS** as the **e*Way Connection Type**.

2 Using the e*Way Configuration Editor accept the default settings for all parameters except for:

- **General Settings** (all parameters)

You must enter your system settings for these parameters as shown in **Figure 72 on page 198** and **Figure 73 on page 199**.

Figure 72 e*Way Configuration Editor: Necessary recordQueue Settings 1

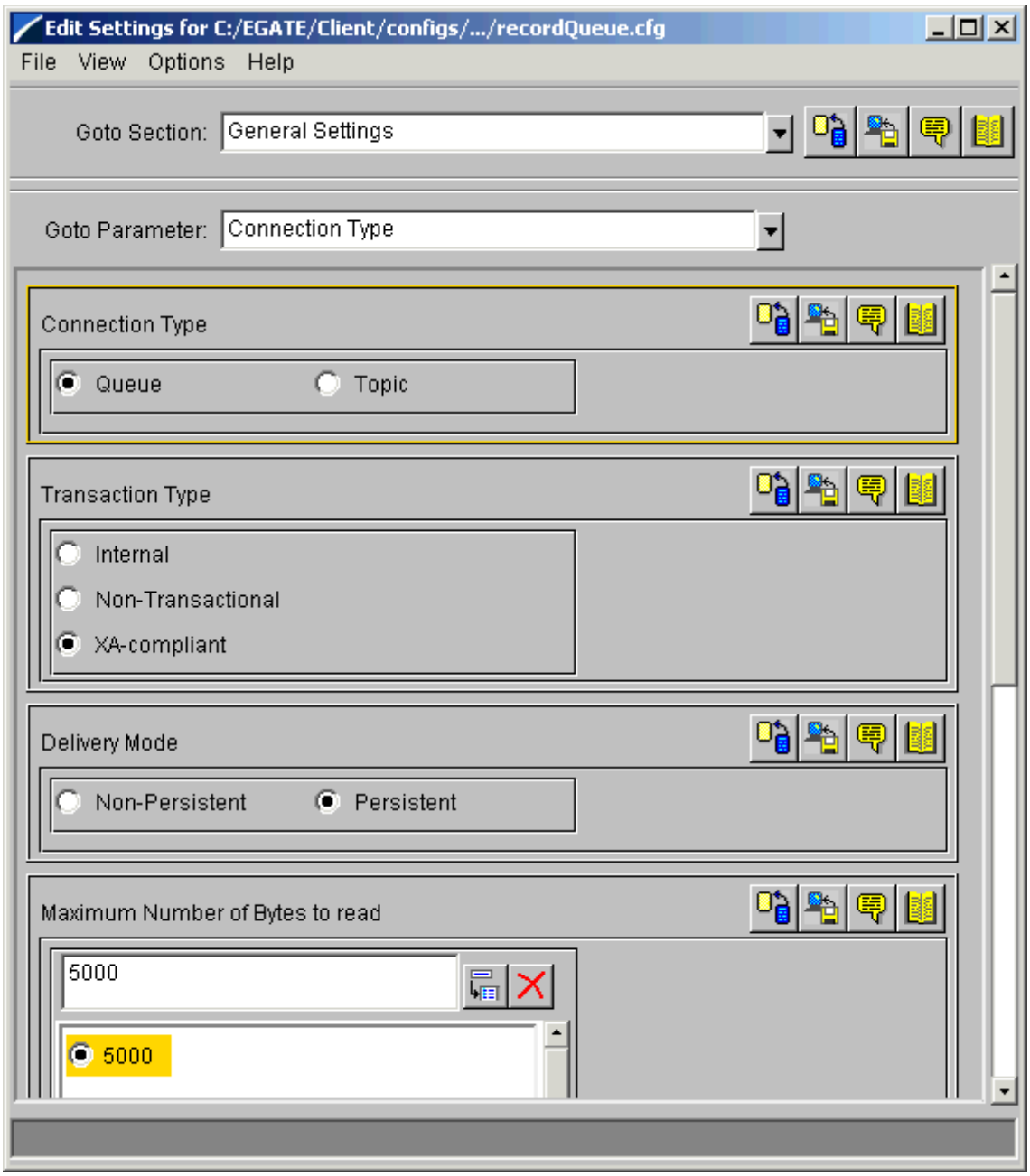
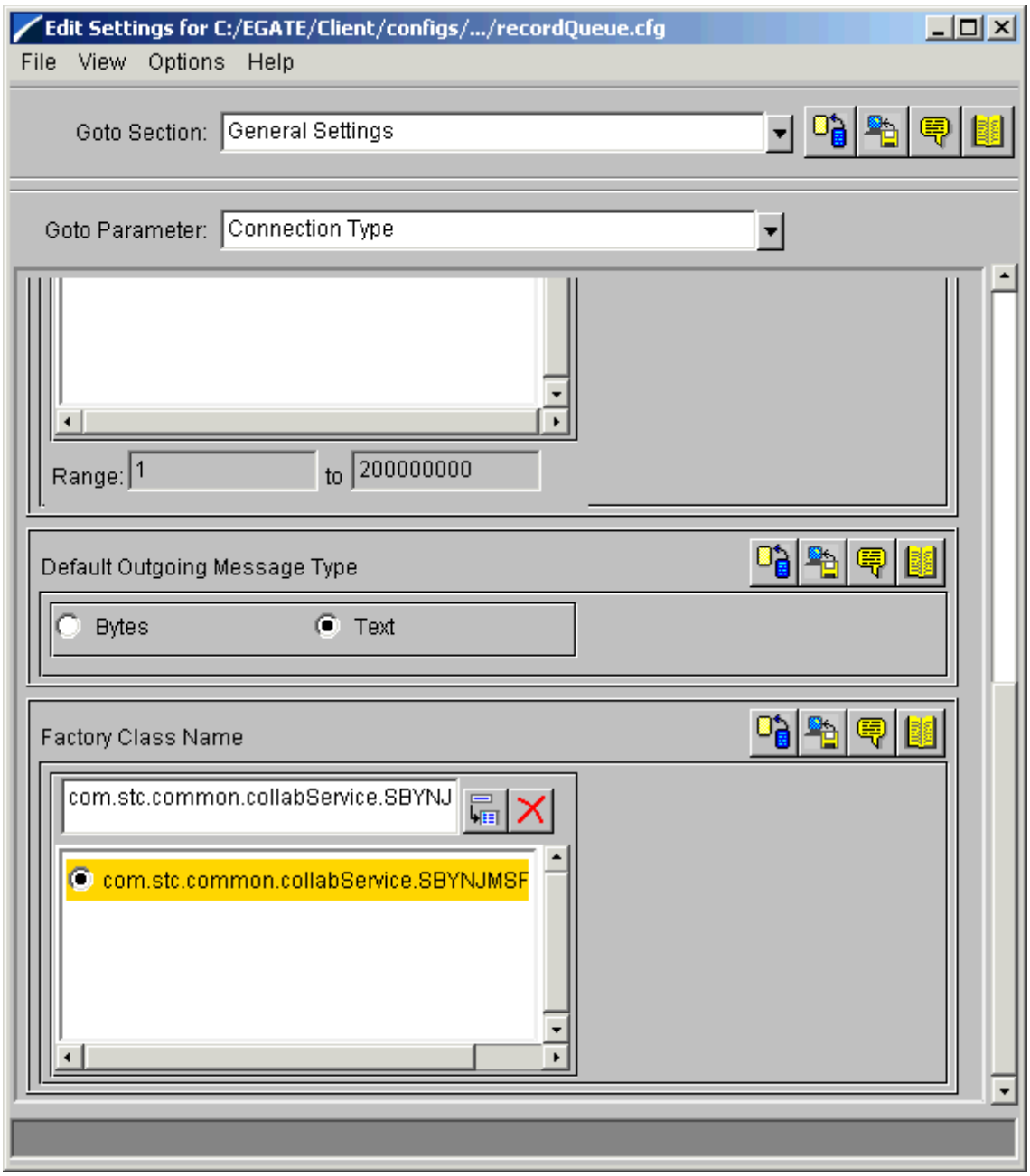


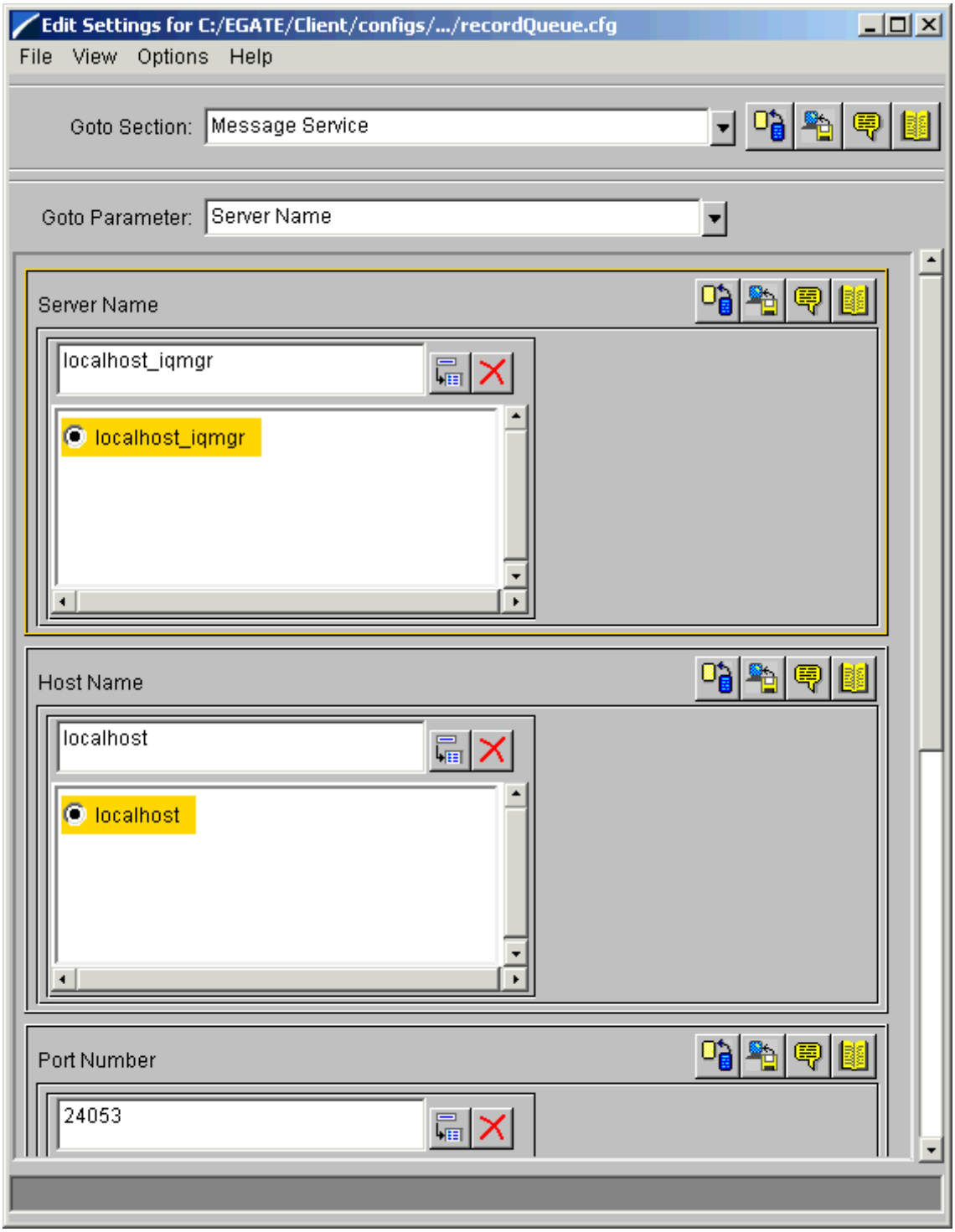
Figure 73 e*Way Configuration Editor: Necessary recordQueue Settings 2



In addition, set the following parameters (see **Figure 74 on page 200**) to match your own system:

- **Message Service:**
  - **Server Name**
  - **Host Name**
  - **Port Number**

Figure 74 e*Way Configuration Editor: Necessary recordQueue Settings 3



3 When you are finished, save the .**cfg** file as the name of the e*Way Connection, close the e*Way Configuration Editor, and promote the file to run time.

4 Click **OK** to close the **e*Way Connection Properties** dialog box.

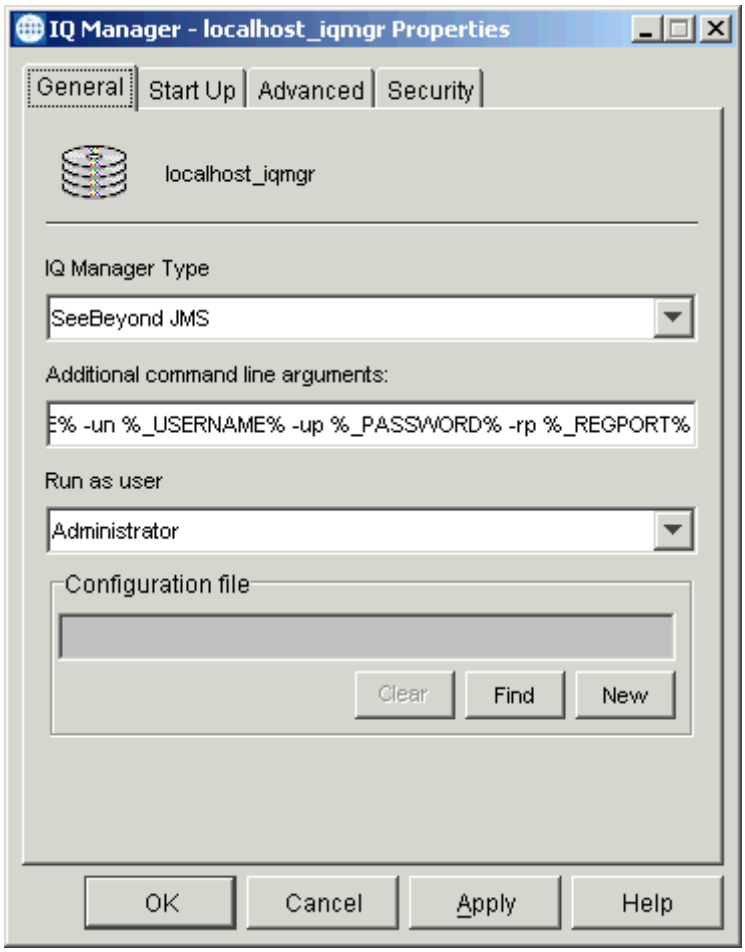
Checking the IQ Manager

By default, the IQ Manager in your schema is configured to use the Oracle SeeBeyond Java Messaging Service (JMS) IQ Service. The system has already named the IQ Manager for you, so you can keep that name if desired. In the sample, the component is named **localhost_iqmgr**.

IQs use IQ Services to transport data. IQ Services provide the mechanism for moving Events between IQs, handling the low-level implementation of data exchange (such as system calls to initialize or reorganize a database).

For an illustration of the IQ Manager Properties dialog box for this schema, see Figure 75.

Figure 75 IQ Manager Properties Dialog Box



Note: *See the **Oracle SeeBeyond JMS Intelligent Queue User's Guide** for more information on the JMS IQ Manager feature.*

Creating Collaboration Rules

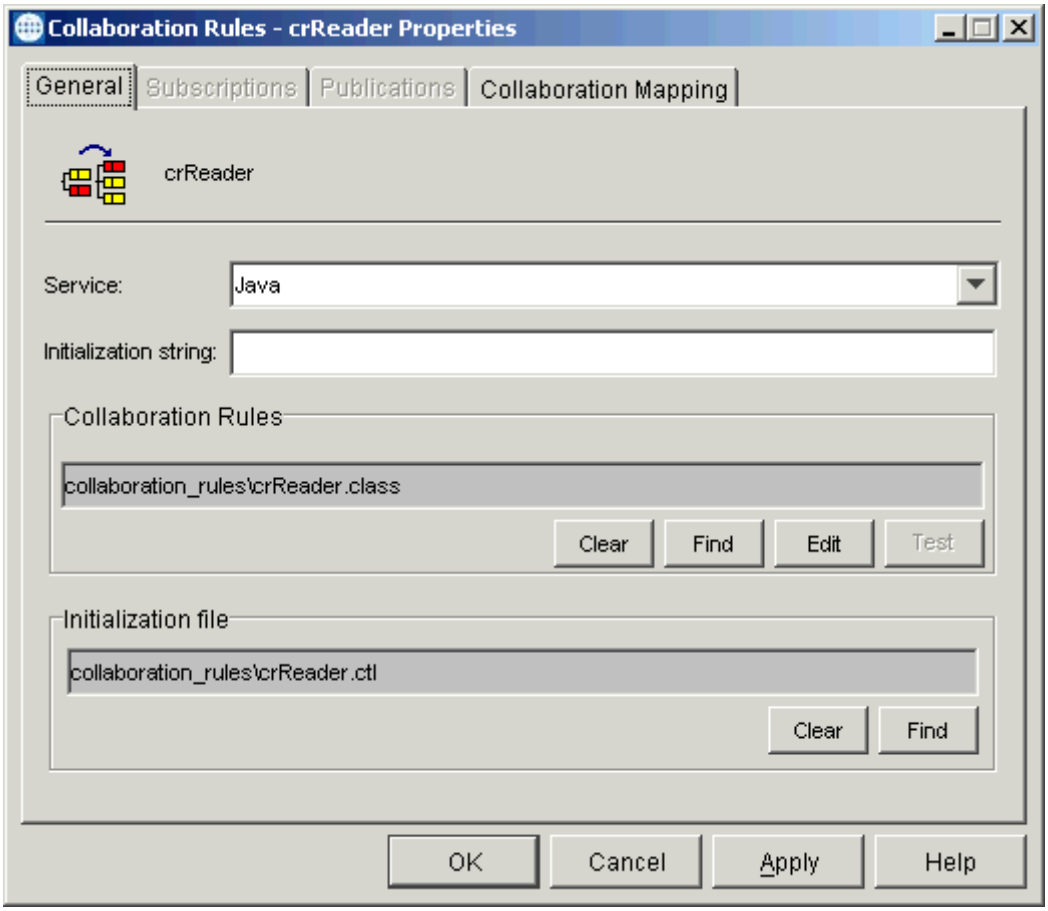
The next step is to create the Collaboration Rules that extract and process selected information from the source Event Type defined previously, according to its associated Collaboration Service.

From the Schema Designer menu bar, select **Options** and click **Default Editor**. For this schema, set the default to **Java**.

To create the crReader Collaboration Rules file

1. Select the **Components** tab in the e*Gate Schema Designer.
2. In the Navigation pane, select the **Collaboration Rules** folder.
3. On the palette, click the **Collaboration Rules** icon.
4. Enter the name of the new Collaboration Rule, then click **OK.** Use **crReader** for this example, for the **ewReader** e*Way's Collaboration, **colReader**.
5. Select the new **Collaboration Rule**, then right-click to edit its properties.
6. The **Collaboration Rules Properties** dialog box appears (see Figure 76).

Figure 76 Collaboration Rules Properties Dialog Box for crReader: General Tab



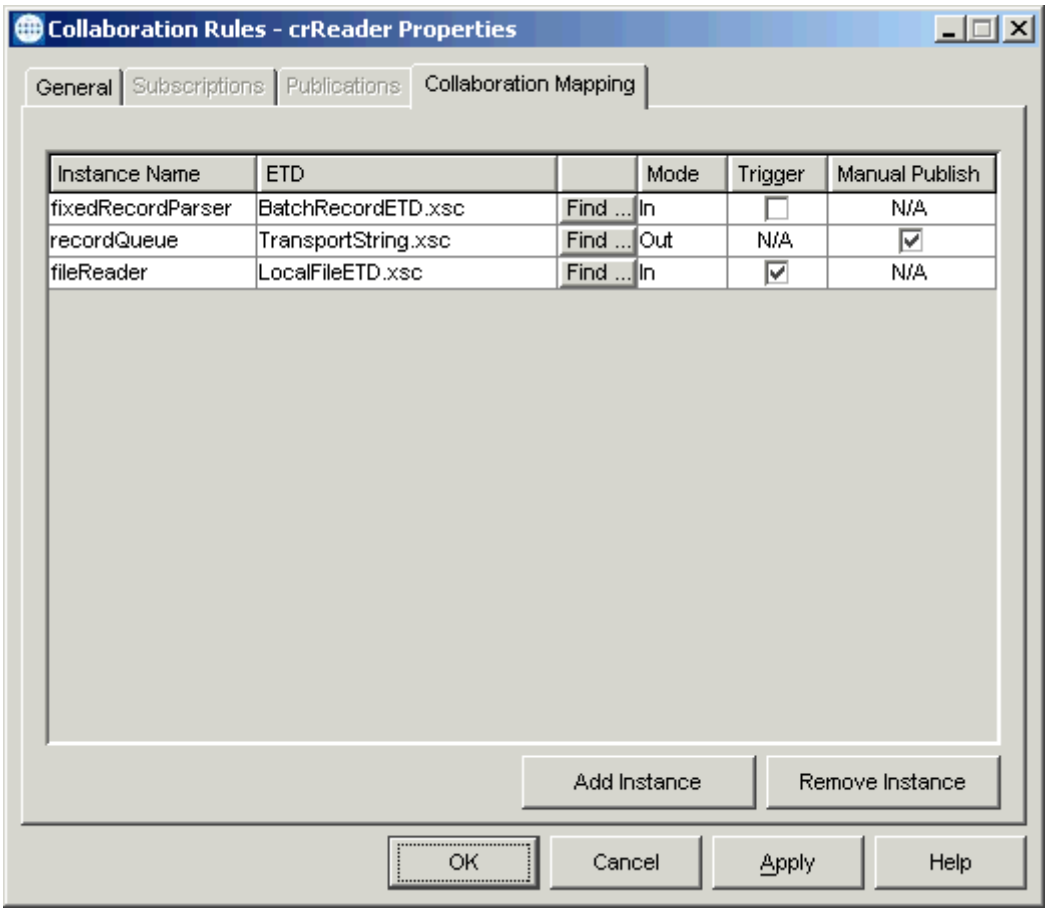
7 On the **General** tab in the dialog box select the **Java Collaboration Service**. In this example, the Collaboration Rules use the e*Gate Java Collaboration Service to manipulate Events or Event data.

8 In the **Initialization String** text box, enter any required initialization string that the Collaboration Service may require. This field can be left blank.

9 Click the **Collaboration Mapping** tab.

10 Using the **Add Instance** button, create instances to coincide with the ETDs (see Figure 77).

Figure 77 Collaboration Rules Properties Dialog Box for crReader: Mapping Tab



Configure the rest of **crReader** as shown in the previous figure.

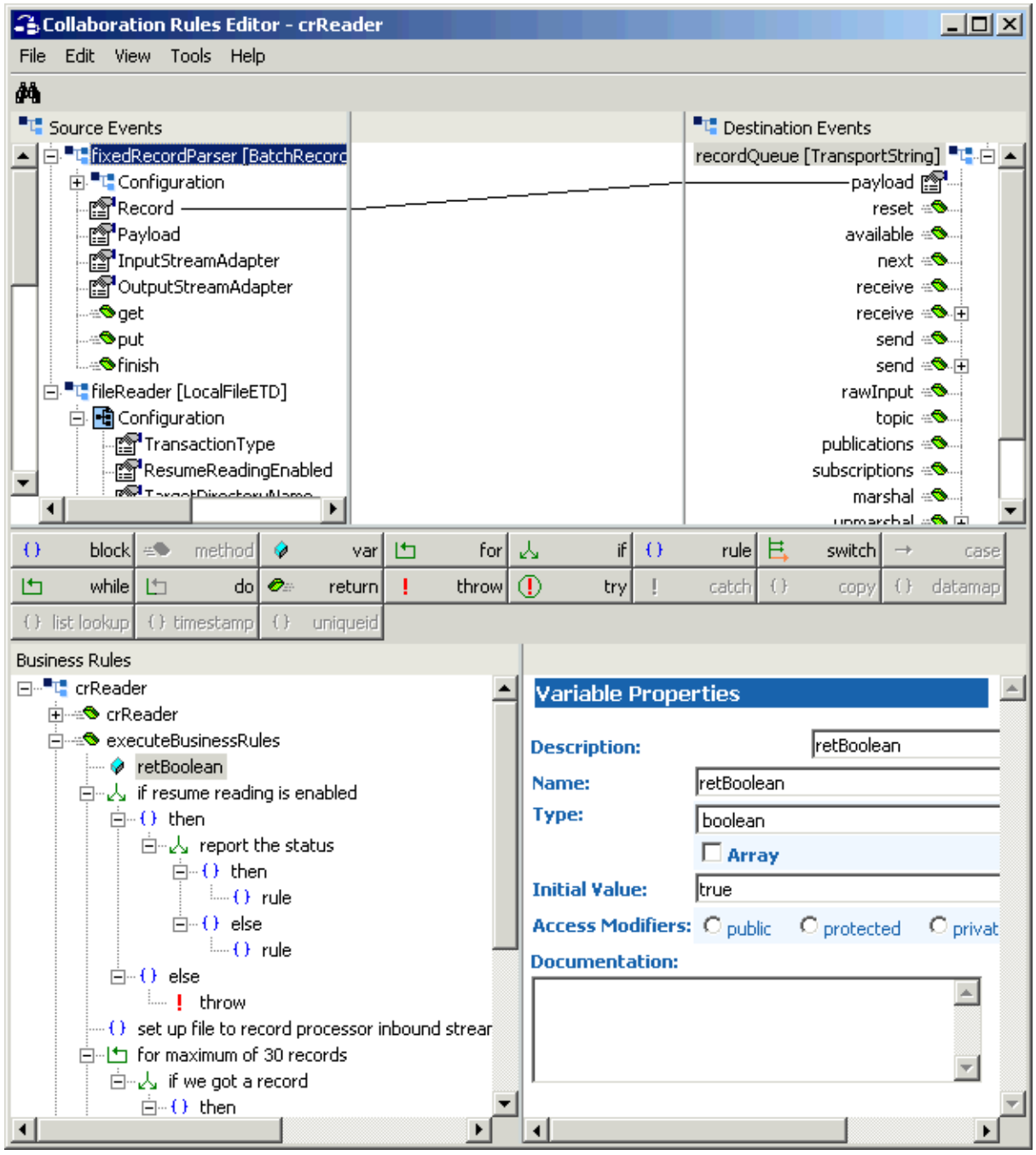
Note: *You can use the **transport.xsc** ETD from the sample for your user-defined ETD or create one of your own that fits the example.*

11 Select the **General** tab again, then click **New** (where the **Edit** button is in **Figure 76 on page 202**).

The Collaboration Rules Editor Main window opens.

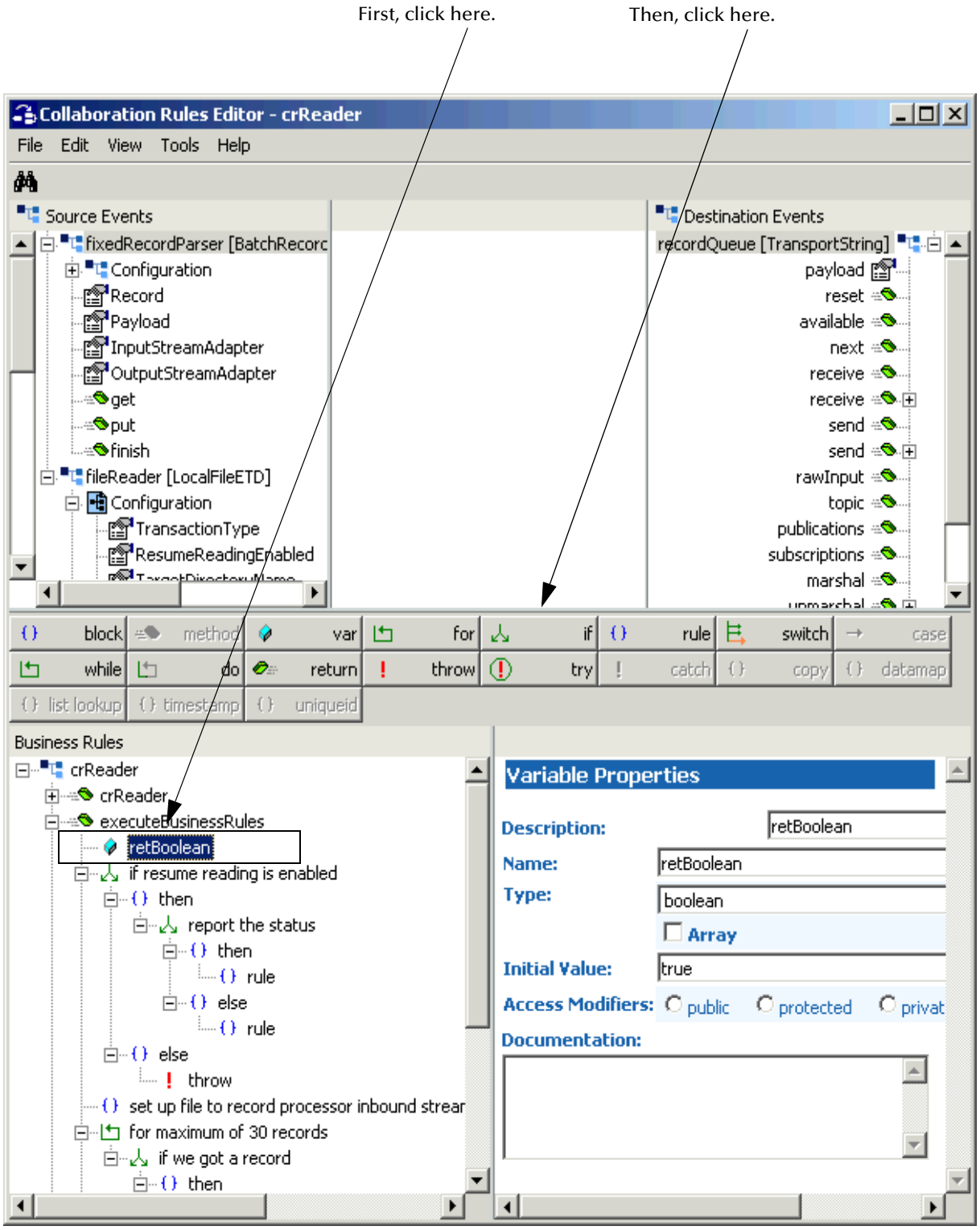
12 Expand the window for optimum viewing then expand the source and destination Events, as well as the Business Rules. Figure 78 shows the results.

Figure 78 Collaboration Rules Editor: crReader Expanded



13 To create the first Business Rule (**if resume reading is enabled**), first click the **retBoolean** method in the Business Rules pane then click the **if** button (see **Figure 79 on page 205**).

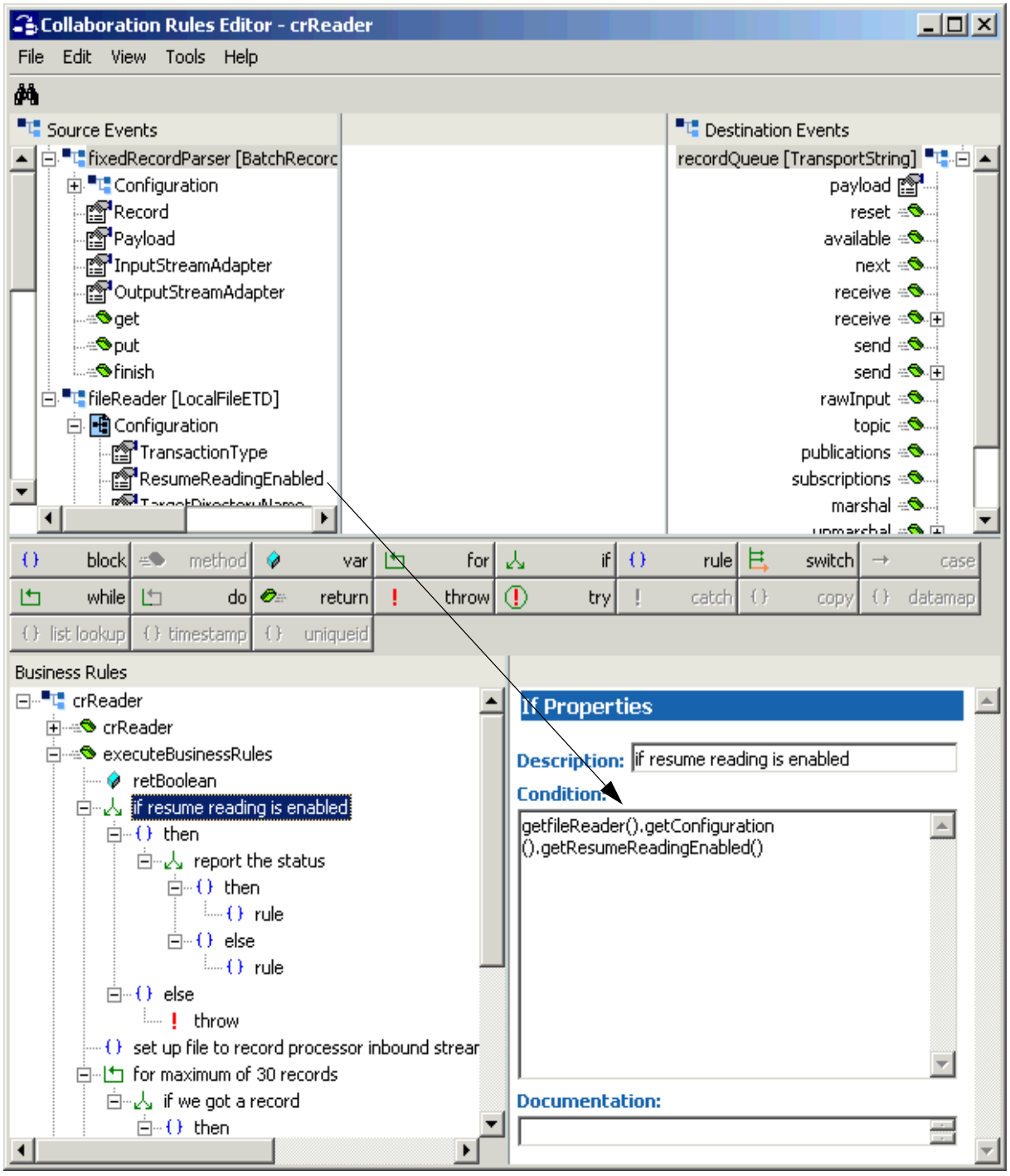
Figure 79 Collaboration Rules Editor: Getting Started



14 Click the new **if** rule and give it the name if **resume reading is enabled**.

15 Drag the **ResumeReadingEnabled** configuration parameter from the local file **Source Event** to the **Rule** scroll box in the **Rule Properties** window (see Figure 80).

Figure 80 Collaboration Rules Editor: crReader if Rule

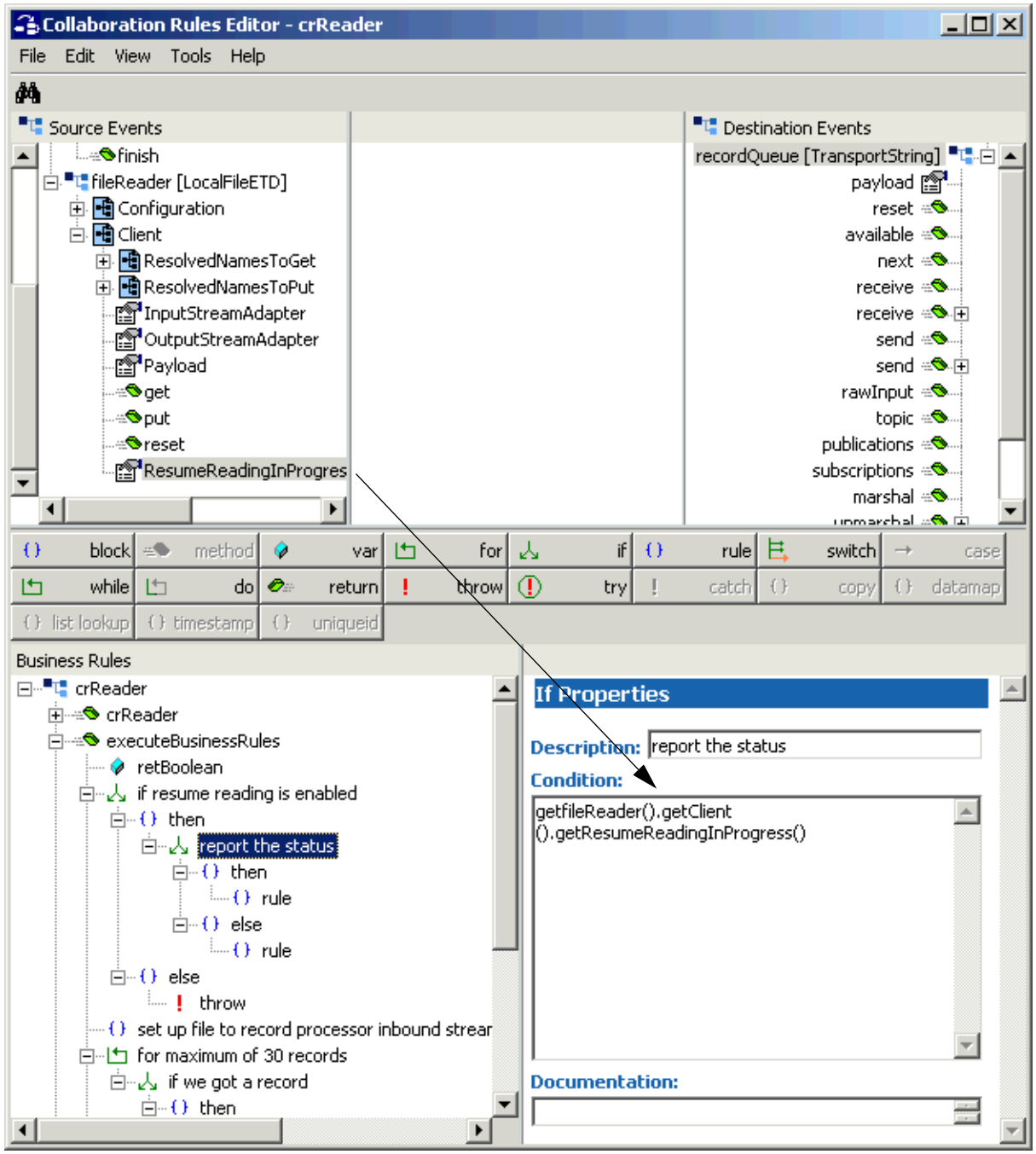


This action enables the Resume Reading feature for the ETD.

16 To create the next portion of this rule, first click **then** in the **Business Rules** window then click **if** to create a nested **if** rule (give it the name, **report the status**).

17 With **report the status** selected, drag the **ResumeReadingInProgress** node of the local file ETD **Source Event** into the **Rule** scroll box in the **Rule Properties** window (see Figure 81).

Figure 81 Collaboration Rules Editor: crReader then Statement

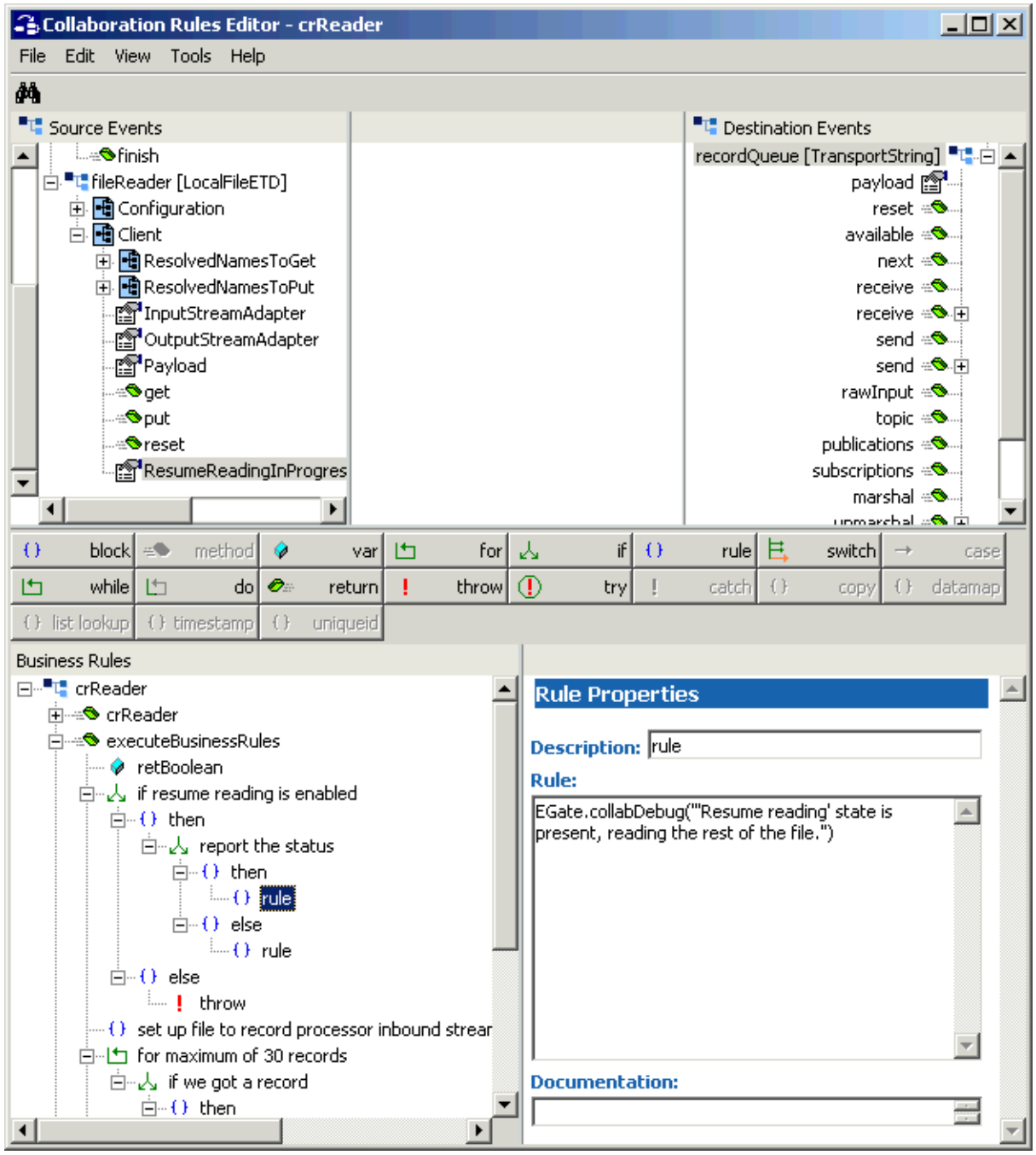


18 To create the **then** rule for this nested **if** rule, first click **then** and next click **rule**. This rule and the next give you messages for the Resume Reading feature.

19 With **rule** selected, as shown in Figure 82, type the following text in the **Rule** scroll box in the **Rule Properties** window:

```
EGate.collabDebug("'Resume reading' state is present, reading the
    rest of the file.")
```

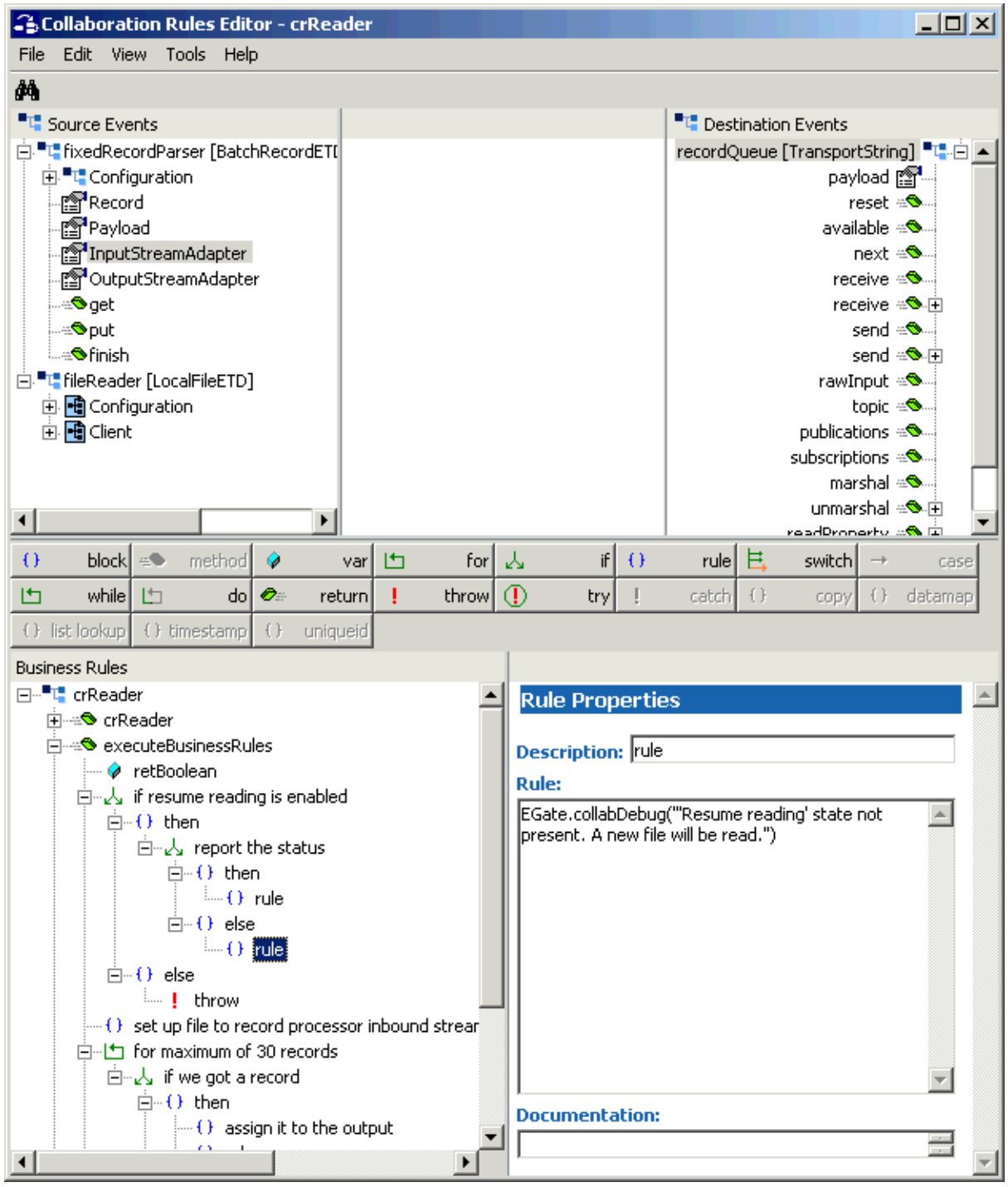
Figure 82 Collaboration Rules Editor: crReader then Rule



20 Click **else** (see Figure 83) then **rule** and repeat the same actions for the **else** rule, typing:

```
EGate.collabDebug("'Resume reading' state not present. A new file
    will be read.")
```

Figure 83 Collaboration Rules Editor: crReader else Rule



21 Click **else** in the main rule then **throw** to create the final **else** rule and complete this Business Rule. This rule provides an error message.

22 As shown in the previous steps and figures, type the following text in the **Rule** scroll box in the **Rule Properties** window:

```
new CollabConnException("'Resume reading' required for this
collaboration rule, because it may perform partial reads from the
file.")
```

23 To create the next Business Rule, click the previous rule.

24 Click the **rule** button again and give it the name **set up file to record processor inbound streaming link** (see **Figure 84 on page 211**).

25 First, drag the **InputStreamAdapter** node from the record-processing ETD **Source Event** to the **Rule** scroll box in the **Rule Properties** window (see **Figure 84 on page 211**). Next, do the same with the **get** method from the same ETD.

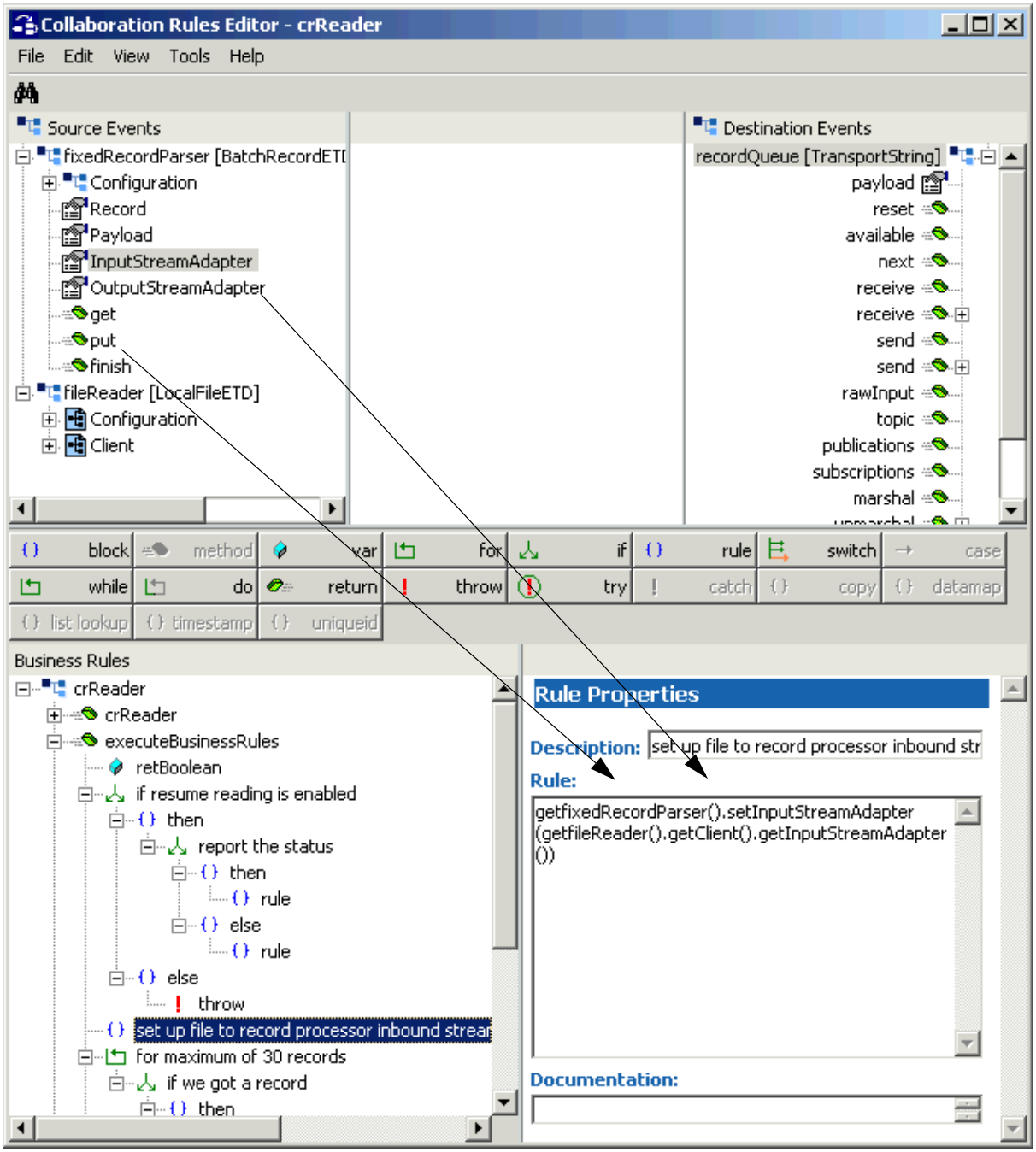
26 You must change one **get** to **set** in the **Rule** scroll box in the **Rule Properties** window to make it read as follows:

```
getfixedRecordParser().setInputStreamAdapter(getfileReader()
    .getClient().getInputStreamAdapter())
```

See **Figure 84 on page 211** for details. You have now created the data-streaming setup.

27 To create the next Business Rule, click the previous rule.

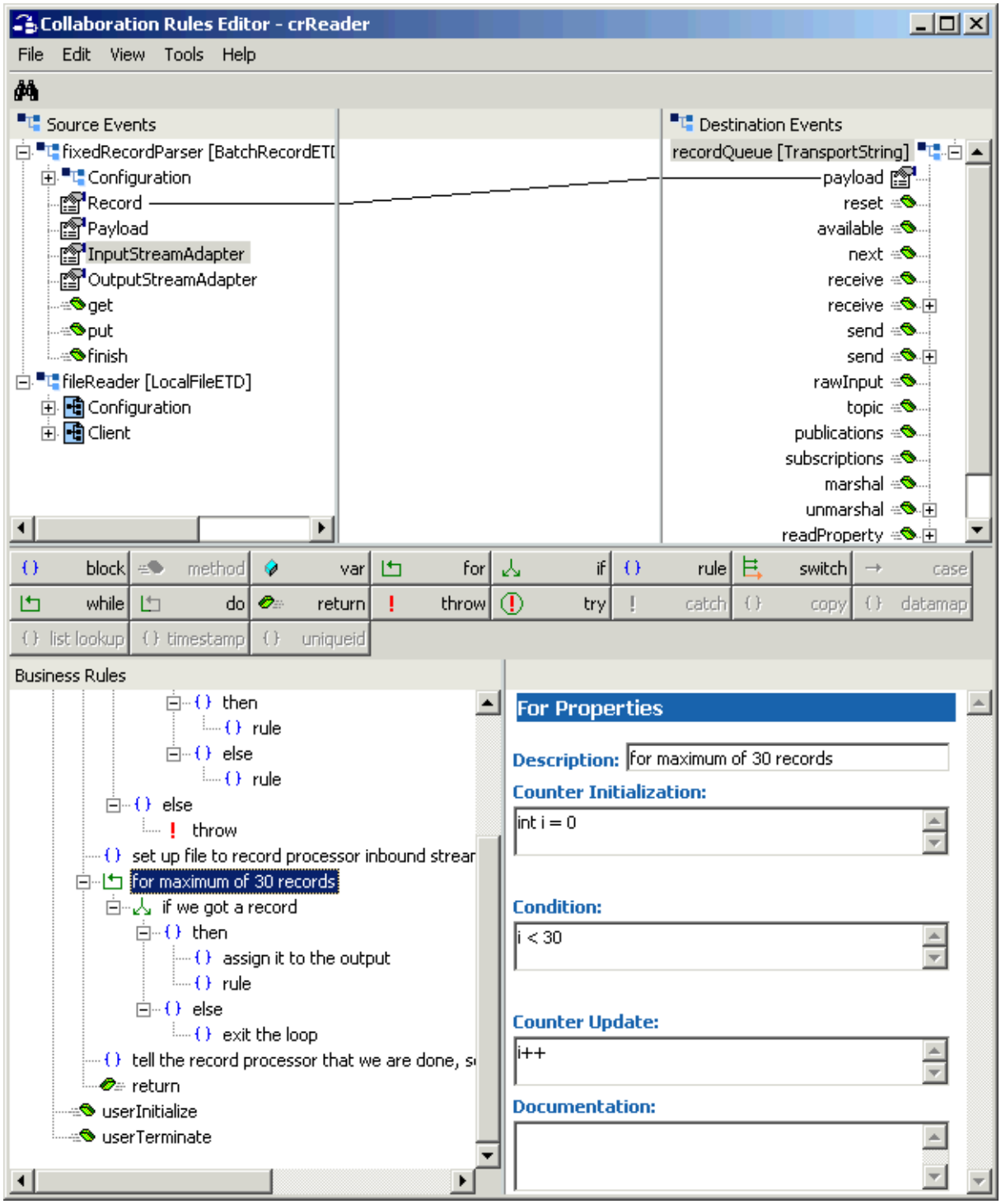
Figure 84 Collaboration Rules Editor: crReader Streaming Setup



28 Click **for** to create a **for** rule (see Figure 85 on page 212). Name it **for maximum of 30 records**. This rule sets up the Resume Reading feature to operate with the Collaboration Rule. Each part of a file transferred by a Business Rule must be 30 records or less long.

29 Make sure the information in the **For Properties** window matches the information shown in Figure 85 on page 212.

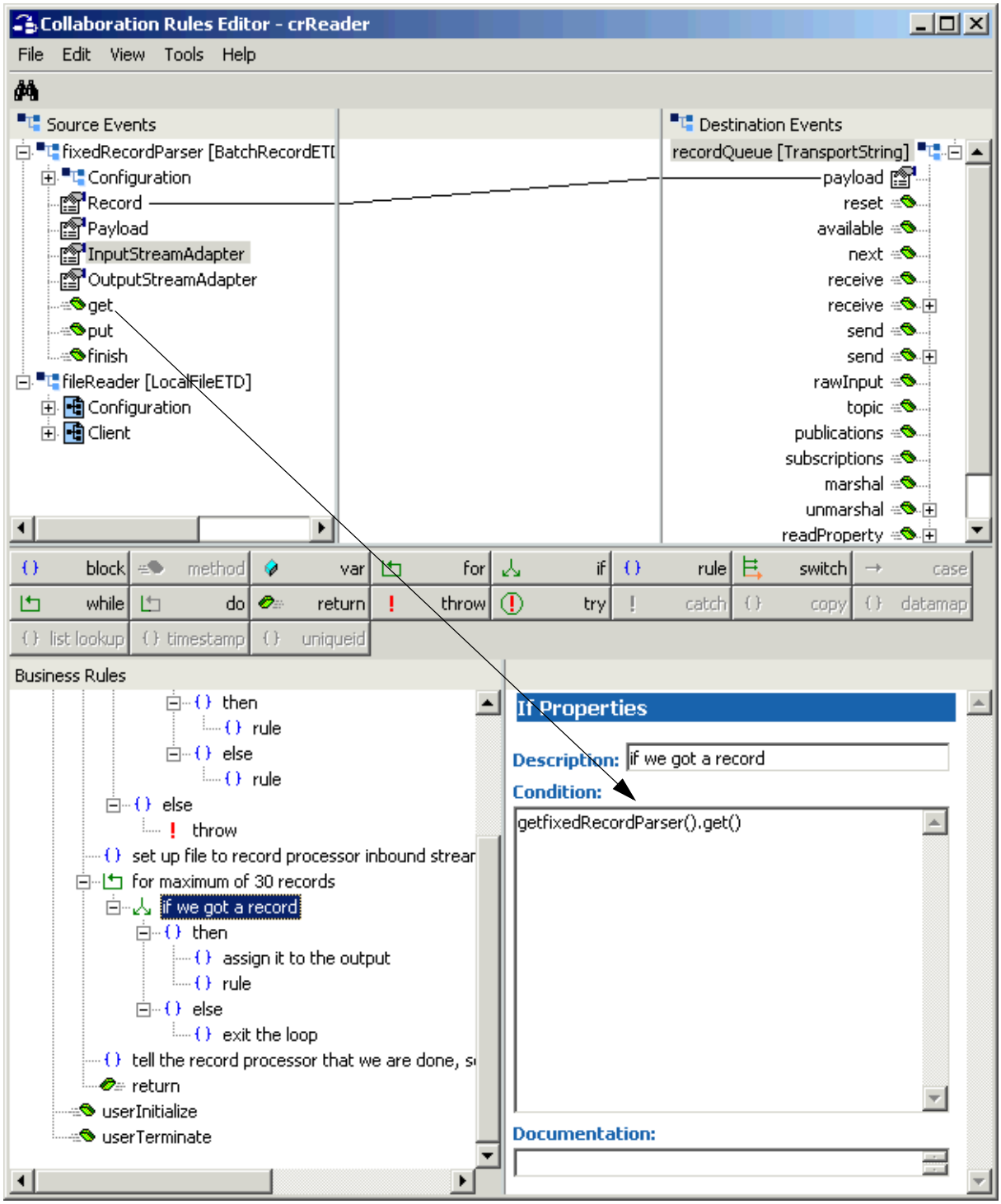
Figure 85 Collaboration Rules Editor: crReader while Rule



30 Click **if** to create another nested **if** rule (see **Figure 86 on page 213**). Name this rule **if we got a record**.

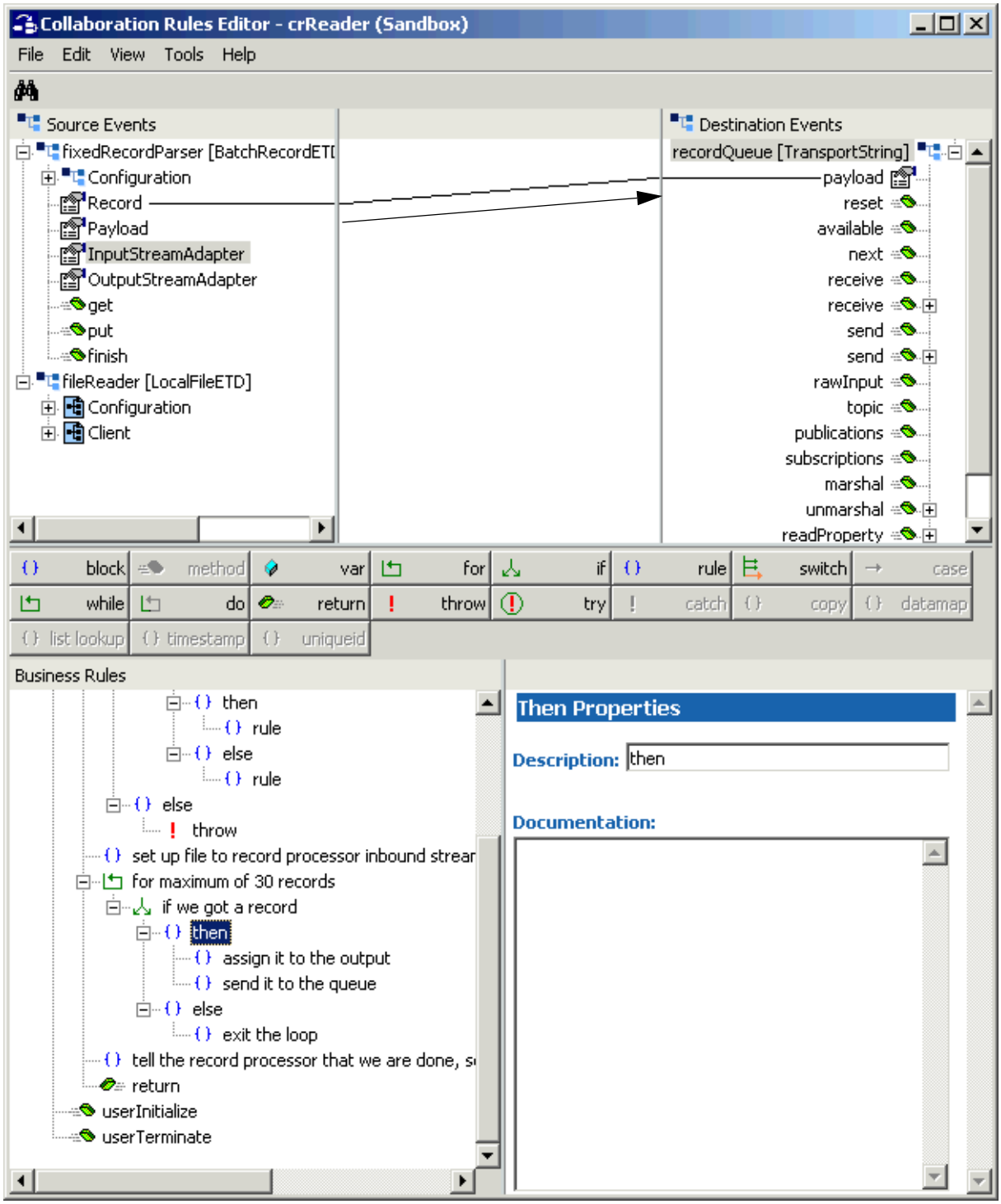
31 Drag the **get** method from the record-processing **Source Event** to the **Rule** scroll box in the **Rule Properties** window (see Figure 86).

Figure 86 Collaboration Rules Editor: crReader get Method



32 Click **then** and drag the **Record** node from the record-processing ETD **Source Event** to the **payload** node of the **Destination Event** (see Figure 87).

Figure 87 Collaboration Rules Editor: crReader Assigning Record to Output

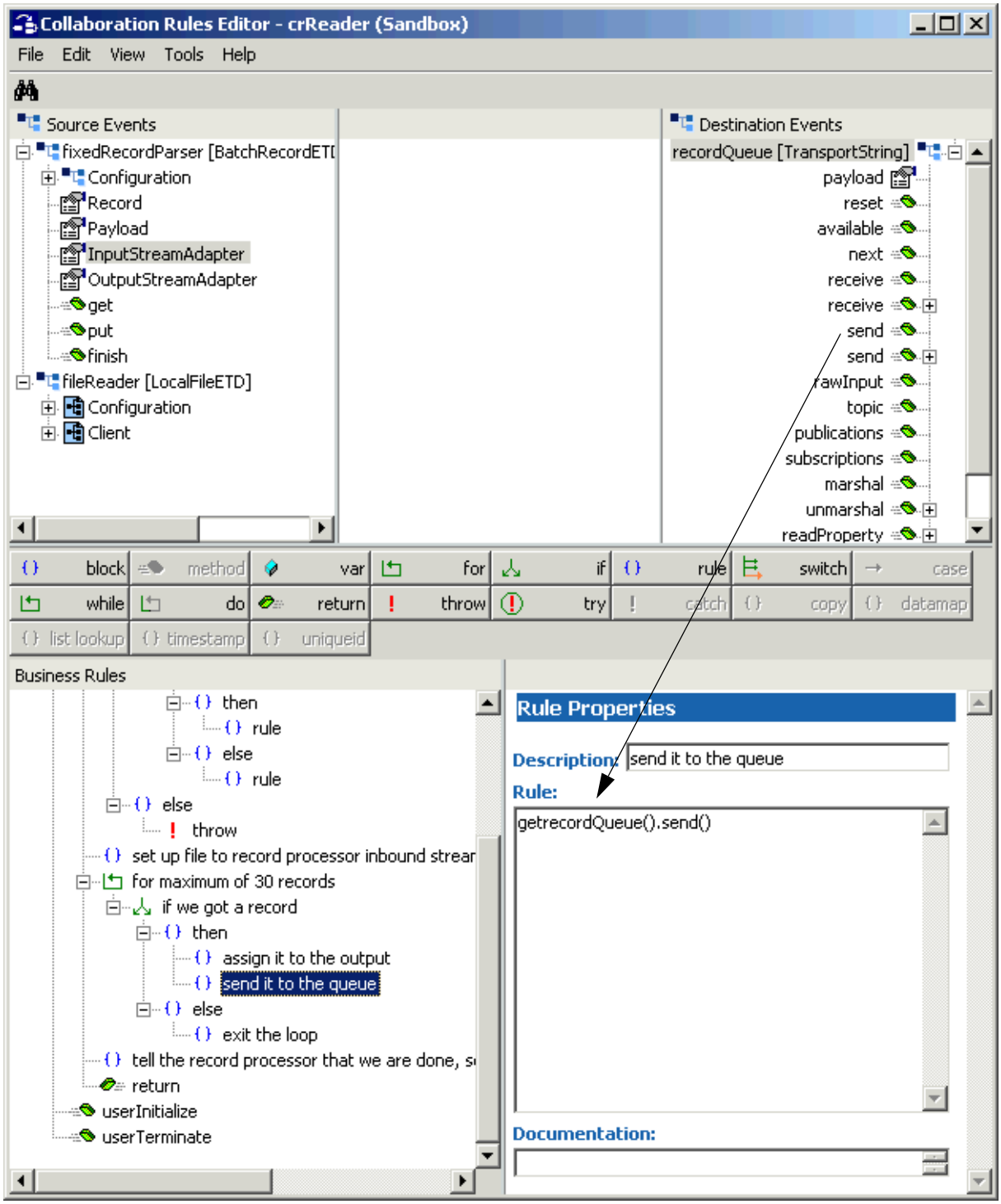


33 Name the new rule **assign it to the output**.

34 With the previous rule still selected, click **rule** to create a new rule.

35 Drag the first **send** method from the **Destination Event** to the **Rule** scroll box in the **Rule Properties** window (see Figure 88).

Figure 88 Collaboration Rules Editor: crReader Sending to IQ Manager

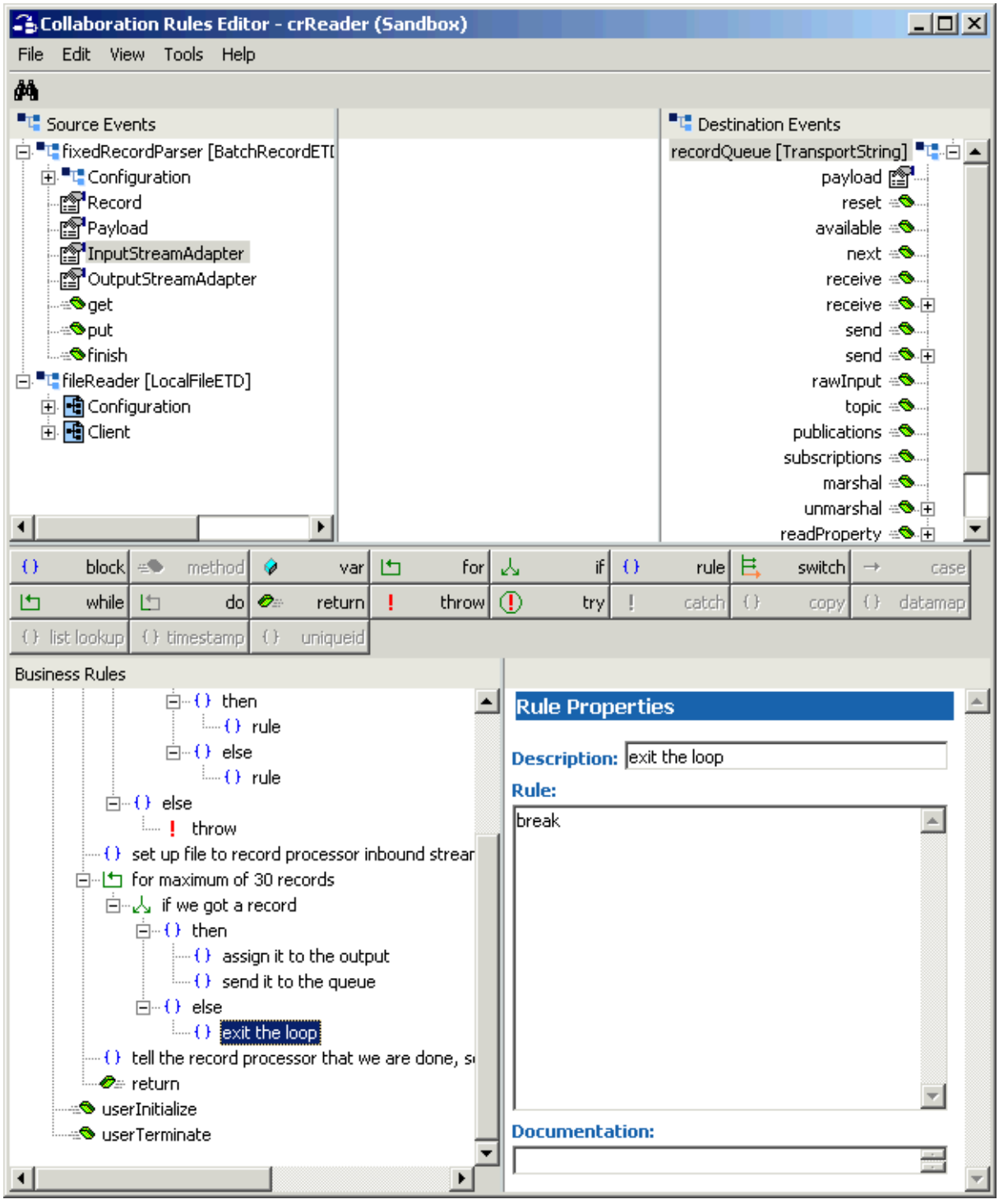


36 Name the new rule **send it to the queue** (see Figure 88).

37 Click **else** then click **rule** again. Name the new rule **exit the loop** (see Figure 89).

38 Enter a **break** in the **Rule** scroll box in the **Rule Properties** window (see Figure 89).

Figure 89 Collaboration Rules Editor: crReader Exiting the Loop

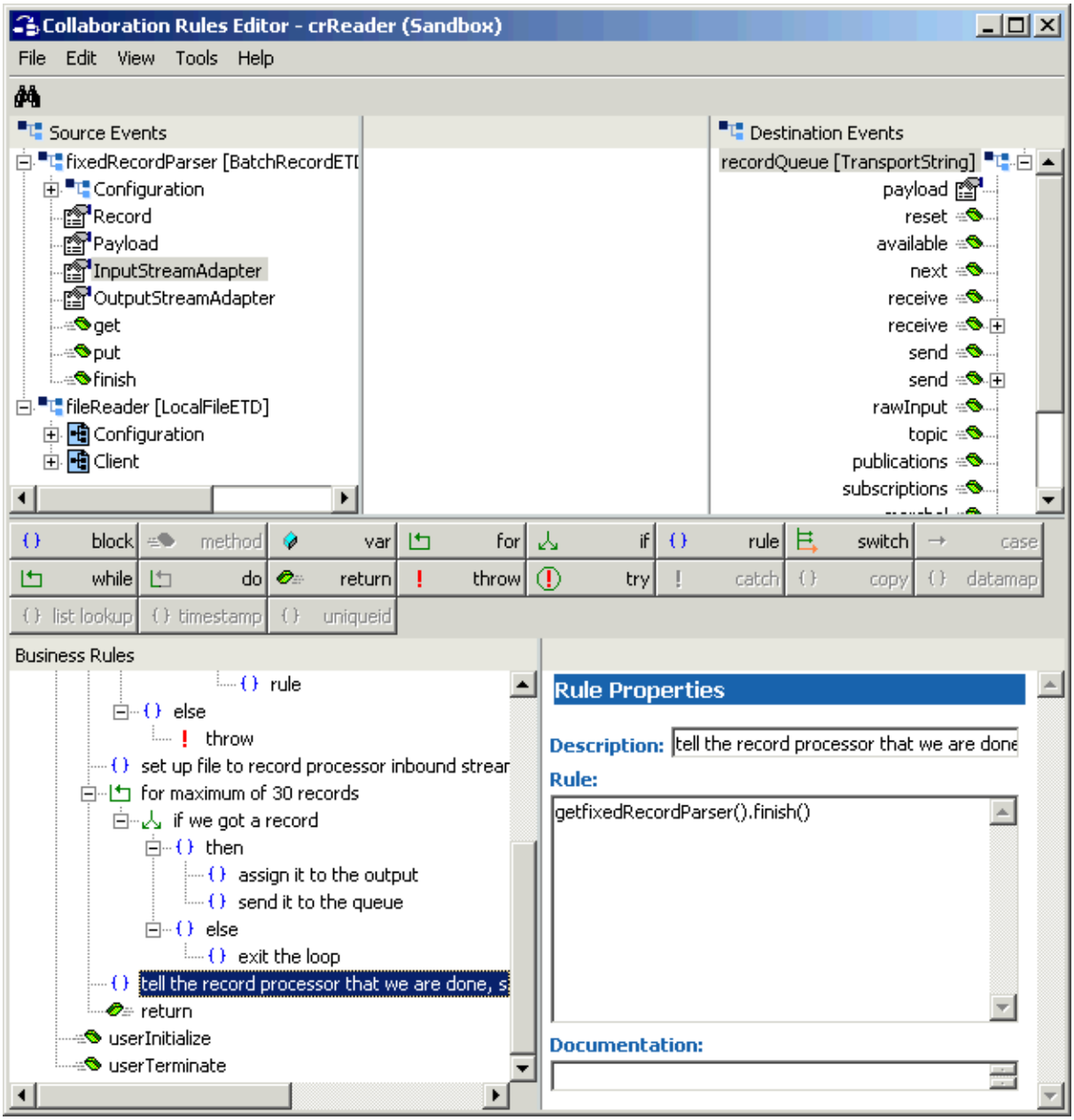


39 To create the next Business Rule, click **for maximum of 30 records**.

40 Click **rule** and name the new rule **tell the record processor that we are done, so it can successfully release the streaming link** (see Figure 90). Use the pop-up dialog box to make the rule a **Sibling** rule.

41 Drag the **finish** method from the record-processing **Source Event** to the **Rule** scroll box in the **Rule Properties** window (see Figure 90).

Figure 90 Collaboration Rules Editor: crReader Finishing Record Processing



You have now finished creating the Business Rules.

42 You must create a Collaboration Rules class or use one from the sample.

Note: *See the* ***e*Gate Integrator User's Guide*** *for details on this procedure.*

43 To save the Collaboration Rules file, click **Save** on the **File** menu. The **Save** dialog box appears.

44 Provide a name for the **.xpr** file (for this example, use **crReader.xpr**) then click **Save**.

45 Before compiling the code, on the **Tools** menu, click **Options** and verify that all necessary **.jar** files are included (see **"Collaboration Rules Editor: Java Classpaths Dialog Box" on page 255**).

46 When you have finished defining all the desired business logic, compile the Java code by selecting **Compile** from the **File** menu.

If the code compiles successfully, the message **Compile Completed** appears. If the outcome is unsuccessful, a Java Compiler error message appears. If there are any Java errors, be sure to correct them.

47 Once the compilation is complete, you can exit the Collaboration Rules Editor.

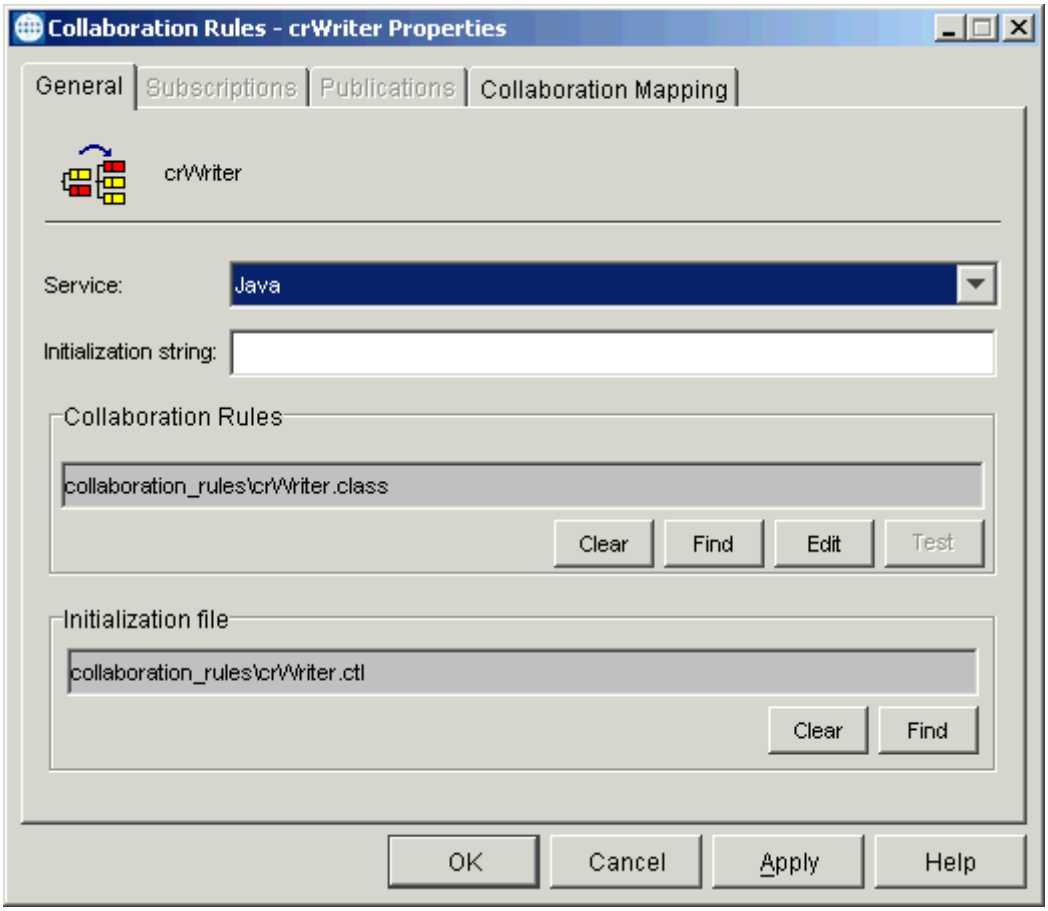
To create the crWriter Collaboration Rules file

1 Repeat steps 3 through 5 under the **procedure on page 147** to create the next Collaboration Rule.

Use **crWriter** as the name for this example, for the **ewWriter** e*Way's Collaboration, **colWriter**.

2 Double-click on the **crWriter** icon. The **Collaboration Rules Properties** dialog box appears (see **Figure 91 on page 219**).

Figure 91 Collaboration Rules Properties Dialog Box for crWriter: General Tab



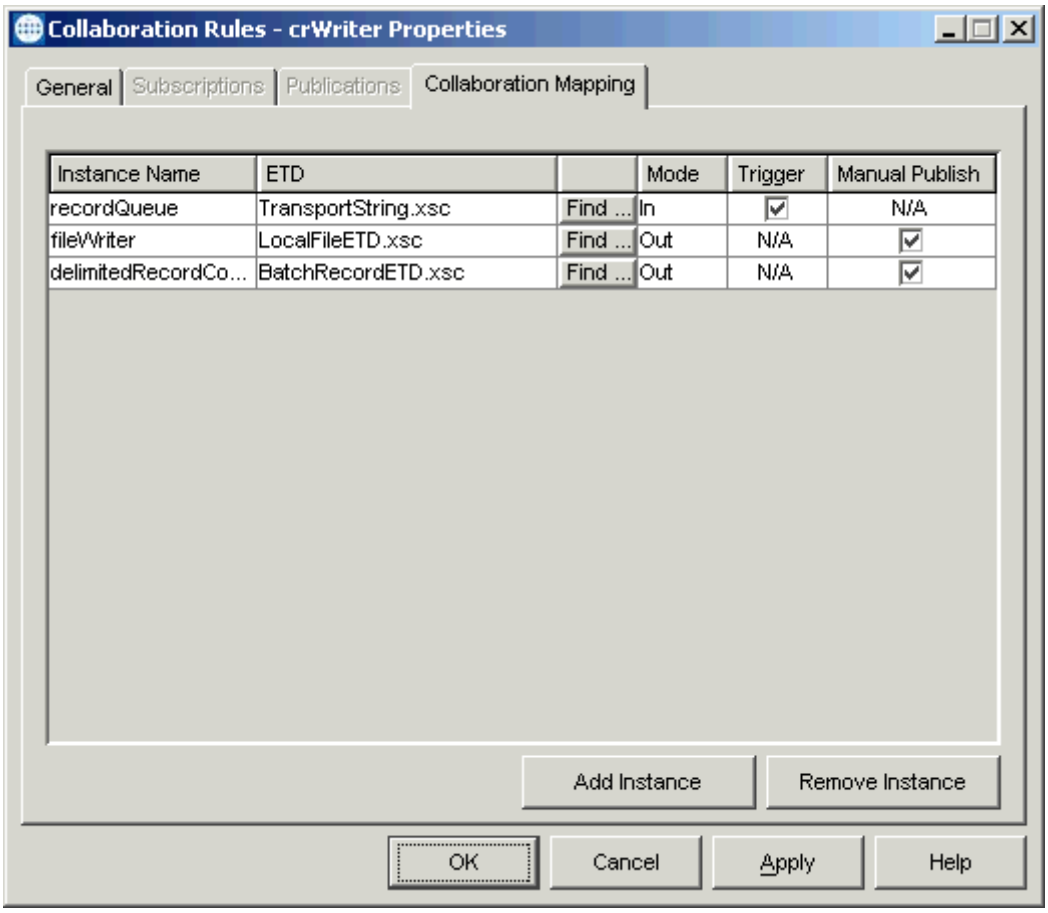
On the **General** tab in the dialog box select the **Java Collaboration Service**.

3 In the **Initialization String** text box (optional), enter any required initialization string that the Collaboration Service may require.

4 Click the **Collaboration Mapping** tab.

5 Using the **Add Instance** button, create instances to coincide with the ETDs (see Figure 92).

Figure 92 Collaboration Rules Properties Dialog Box for crWriter: Mapping Tab

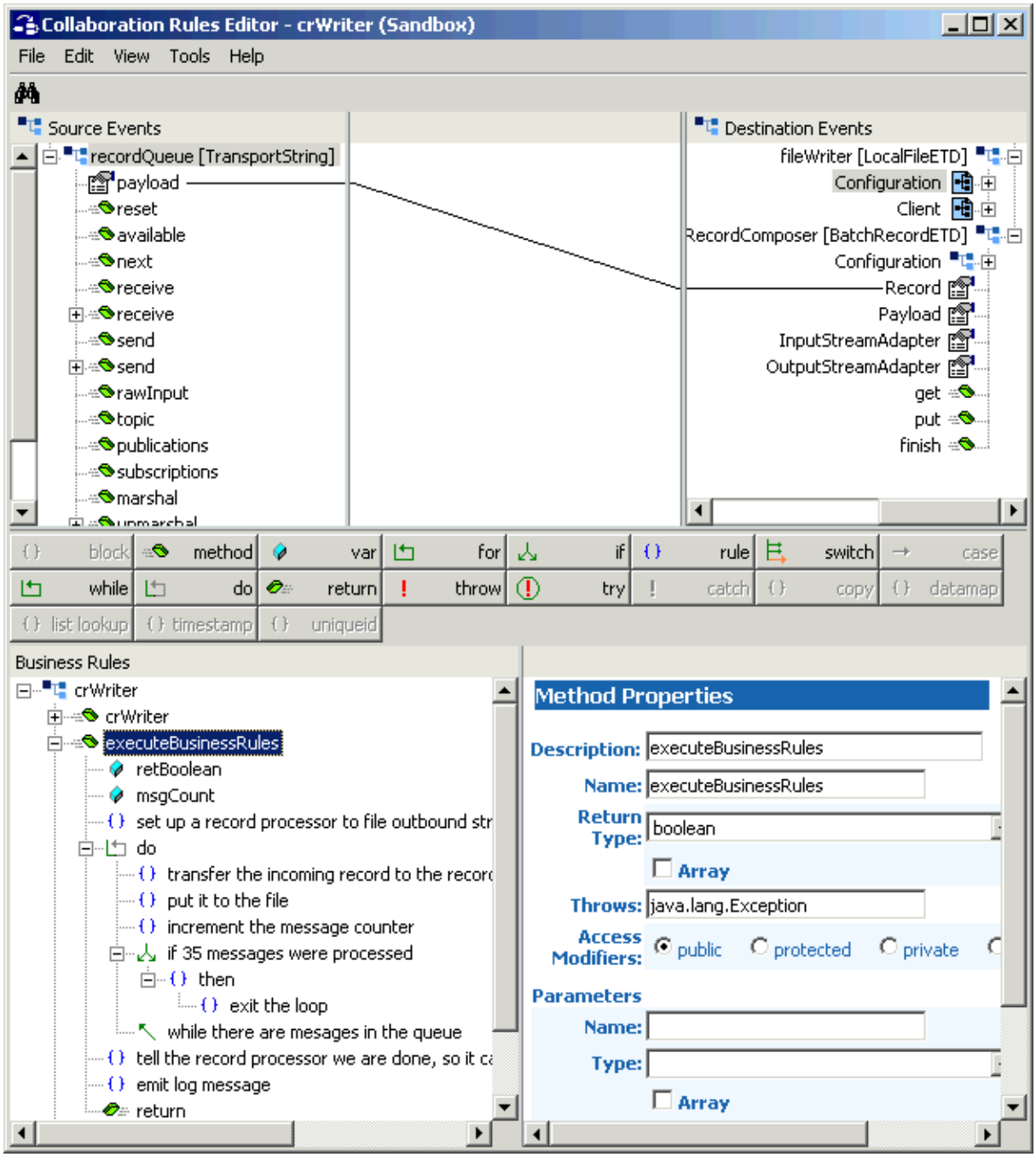


Configure the rest of **crWriter** as shown in the previous figure.

6 Select the **General** tab again, then click **New**.

The Collaboration Rules Editor Main window opens. Expand the source and destination Events, as well as the Business Rules. **Figure 93 on page 221** shows the results.

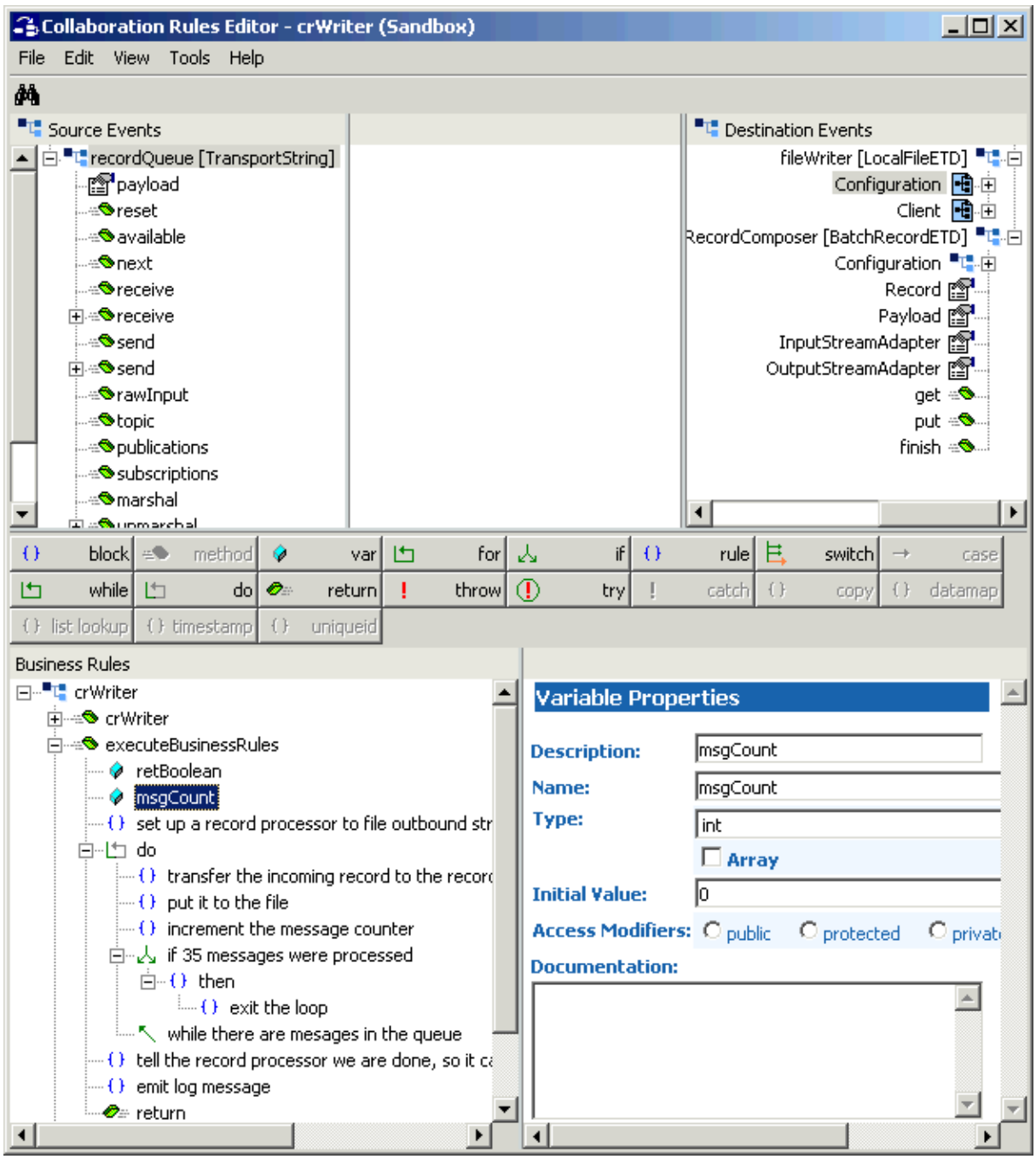
Figure 93 Collaboration Rules Editor: crWriter Expanded



7 Click **retBoolean** then click **var** to create a variable rule. Name the variable **msgCount** (Figure 94 on page 222).

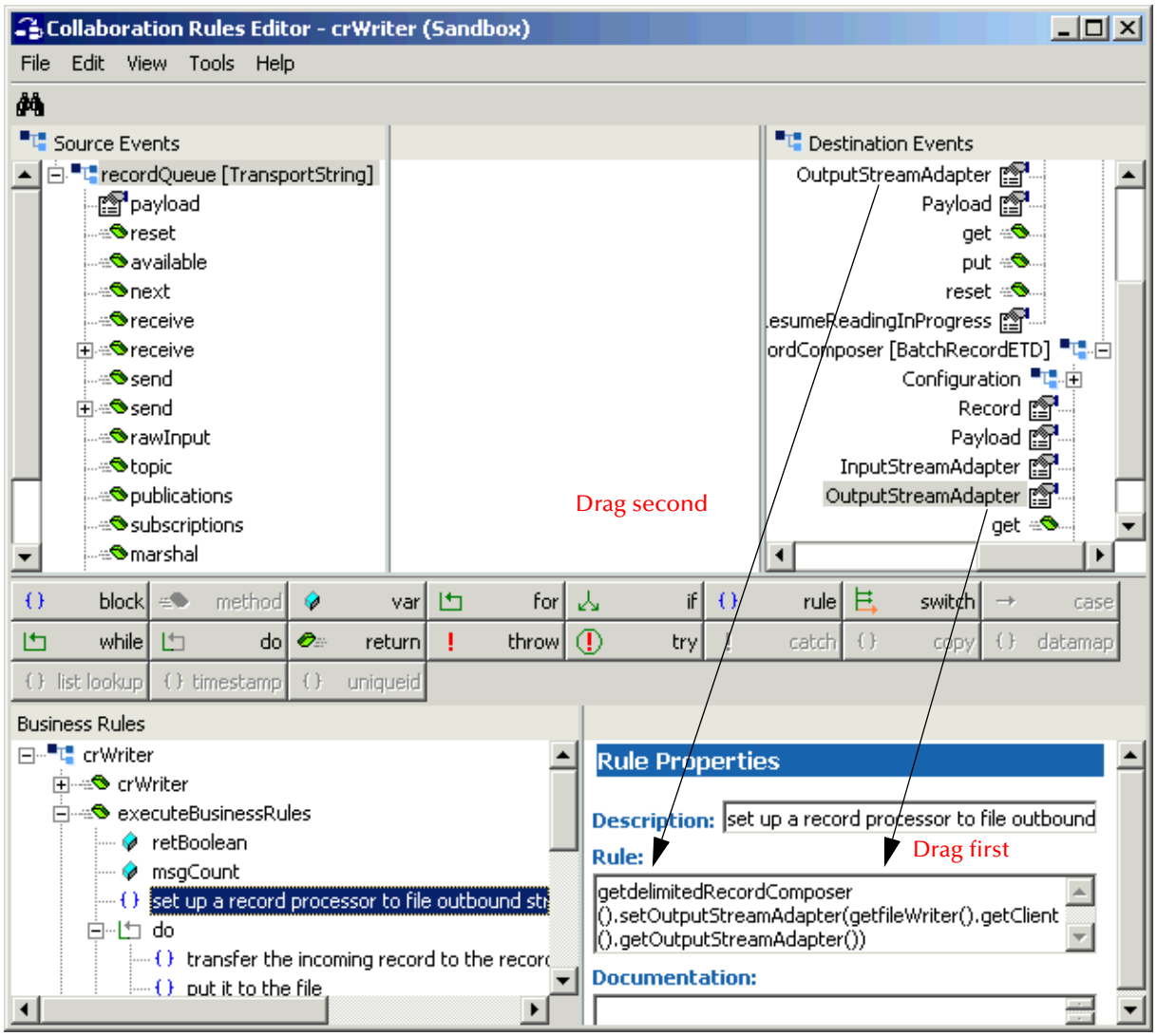
8 Enter information in the **Variable Properties** window as shown in Figure 94.

Figure 94 Collaboration Rules Editor: crWriter msgCount Variable



9 First, drag the **OutputStreamAdapter** node from the record-processing ETD **Destination Event** to the **Rule** scroll box in the **Rule Properties** window (see Figure 95 on page 223). Next, do the same with the **OutputStreamAdapter** method from the local file ETD. These actions create a new rule.

Figure 95 Collaboration Rules Editor: crWriter Streaming Setup



10 Name the new rule **set up a record processor to file outbound streaming link** (see Figure 95).

11 You must edit (change **set** to **get**) the text in the **Rule** scroll box in the **Rule Properties** window to make it read as follows:

```
getdelimitedRecordComposer().setOutputStreamAdapter(getfileWriter
    ().getClient().getOutputStreamAdapter())
```

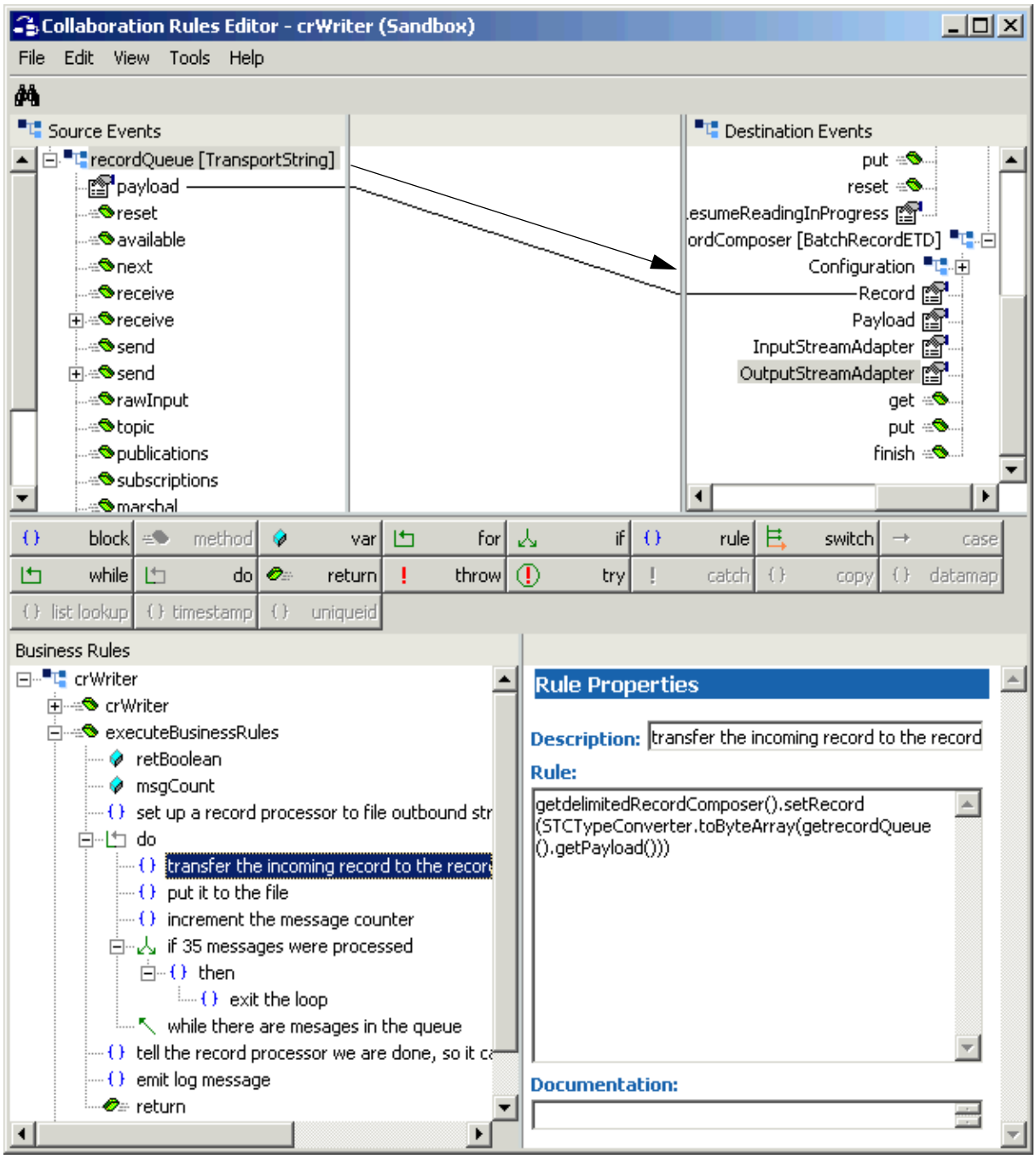
See Figure 95 for details. You have now created the data-streaming setup.

12 To create the next Business Rule, click the previous rule.

13 Click **do** to create a **do** rule then click **rule** to create a new rule under **do**.

14 Drag the **payload** node from the **Source Event** onto the **Record** node of the **Destination Event** record-processing ETD and select **Child** node in the pop-up dialog box (see Figure 96).

Figure 96 Collaboration Rules Editor: crWriter Transferring Records

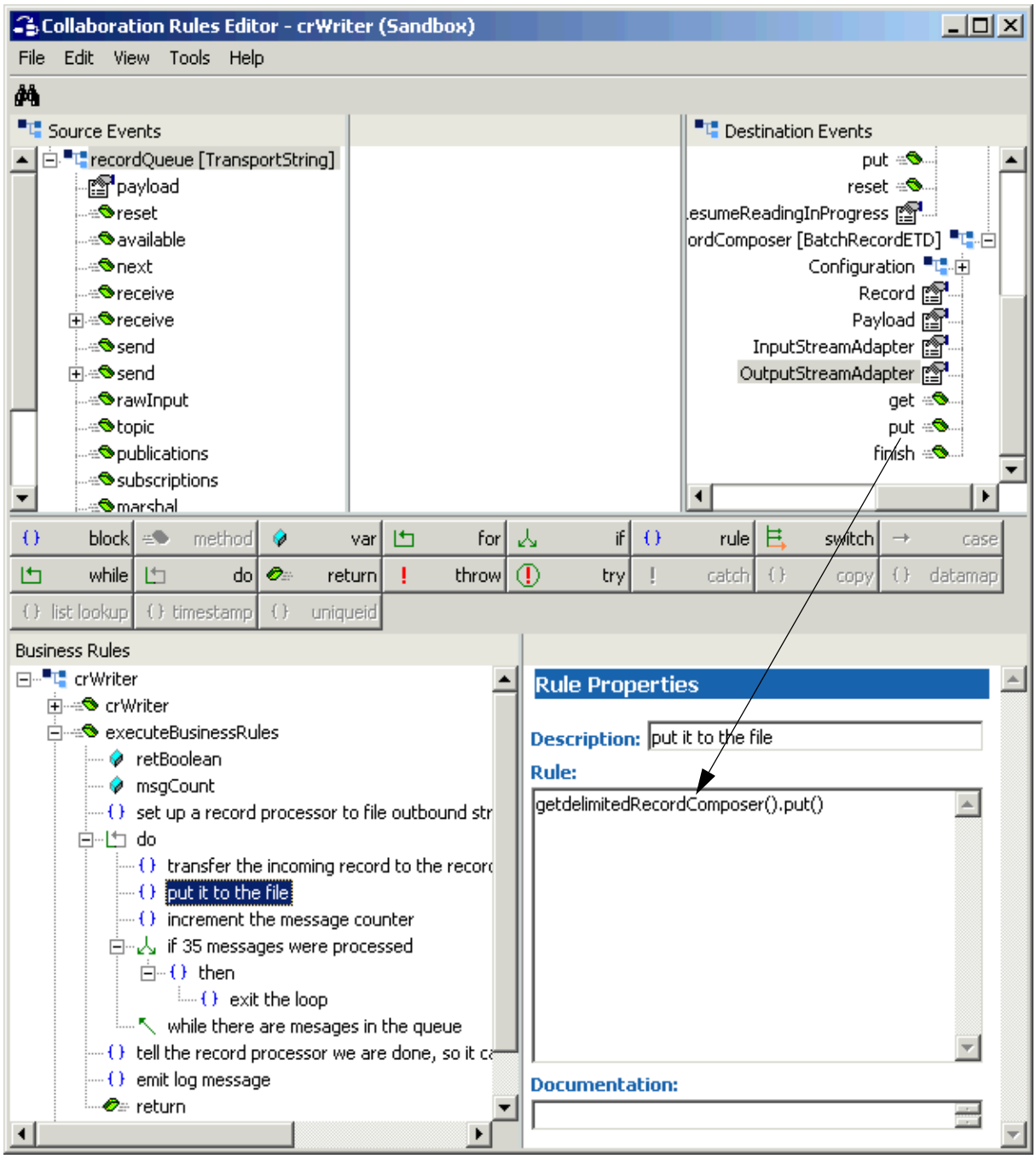


15 Name the new rule **transfer the incoming record to the record processor**.

16 Click **rule** again to create another rule. Name the new rule **put it to the file** (see Figure 97).

17 Drag the **put** method from the **Source Event** into the **Rule** scroll box in the **Rule Properties** window (see Figure 97).

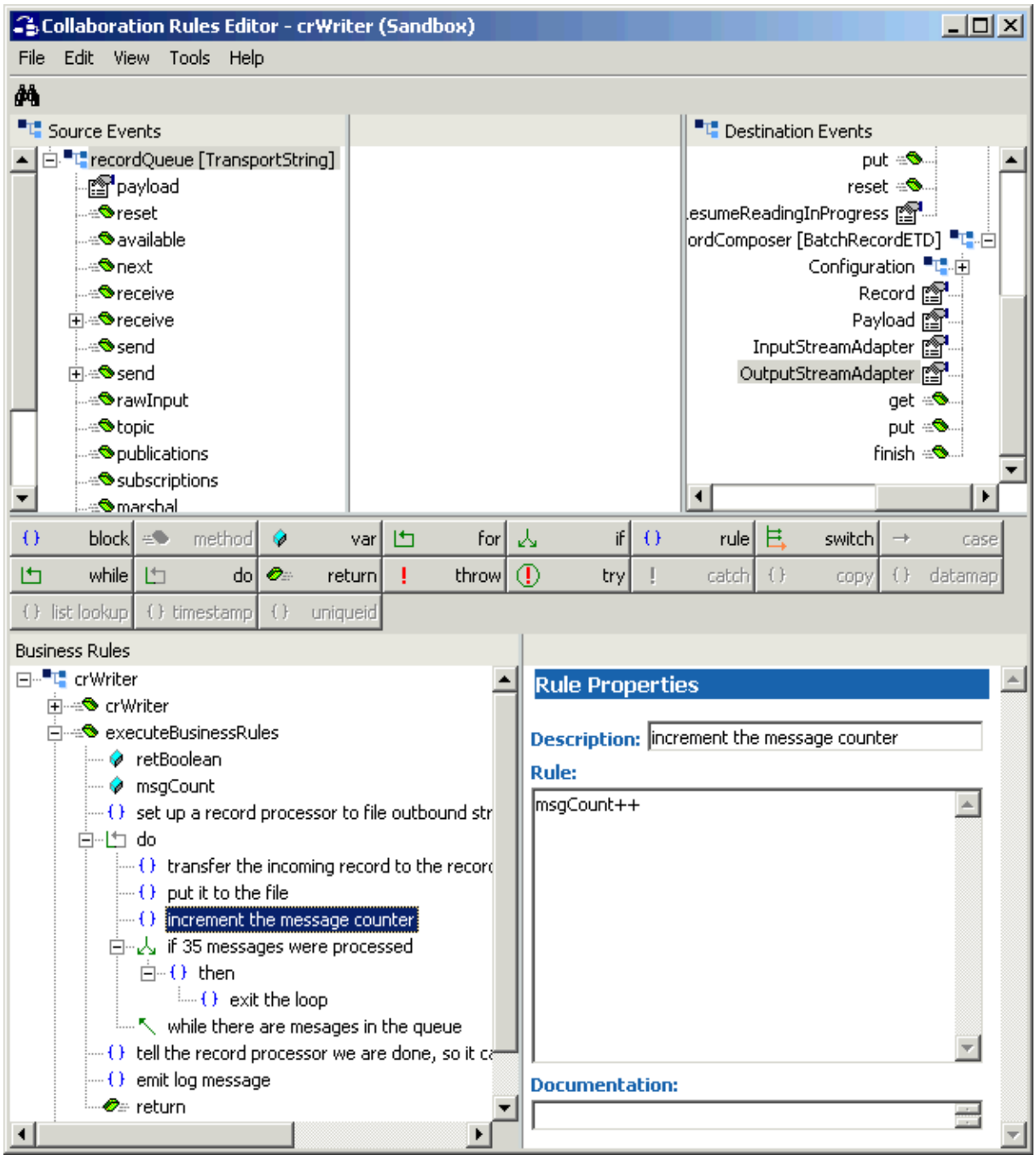
Figure 97 Collaboration Rules Editor: crWriter put to File



18 Click **rule** again to create another rule. Name the new rule **increment the message counter** (see **Figure 98 on page 226**).

19 Enter information in the **Rule** scroll box in the **Rule Properties** window as shown in Figure 98.

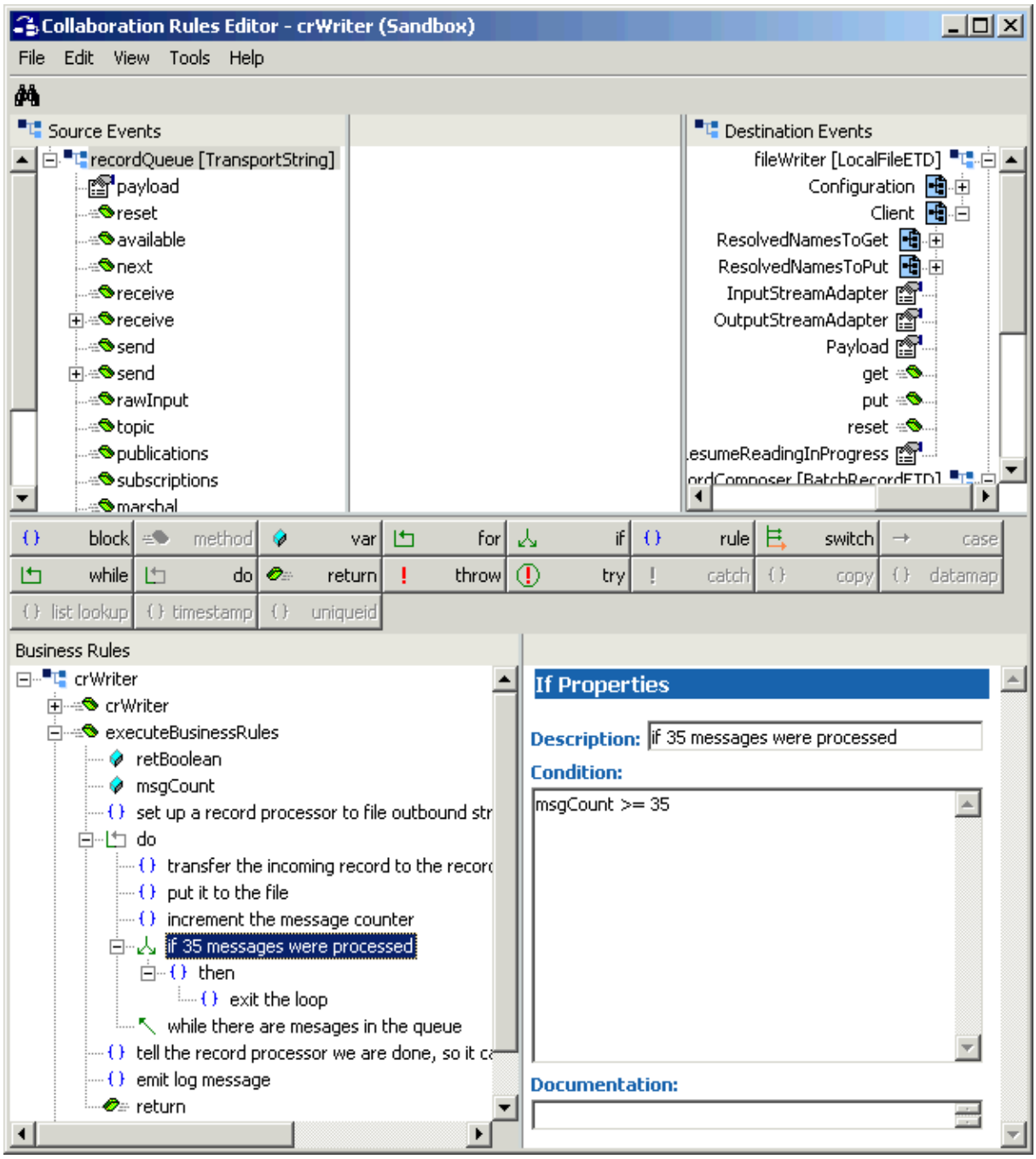
Figure 98 Collaboration Rules Editor: crWriter Message Counter



20 Click **do** to create another rule under it then click **if** to create an **if** rule (see Figure 99 on page 227). Name the new rule **if 35 messages were processed**.

21 Enter information in the **Rule** scroll box in the **Rule Properties** window as shown in Figure 99.

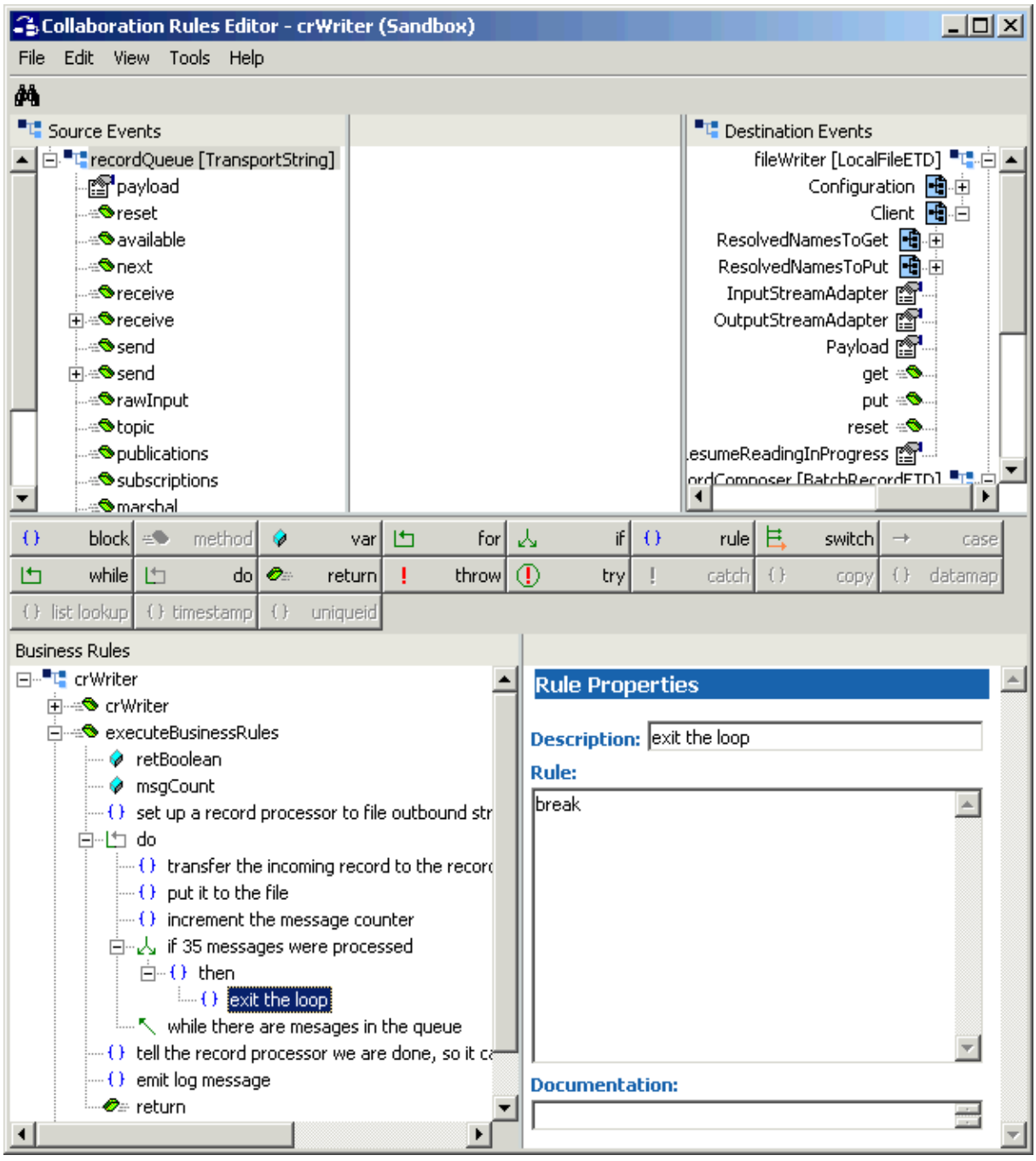
Figure 99 Collaboration Rules Editor: crWriter if Rule



22 Click **then** and next click **rule** to create a rule under **then** (see **Figure 100 on page 228**). Name the new rule **exit the loop**.

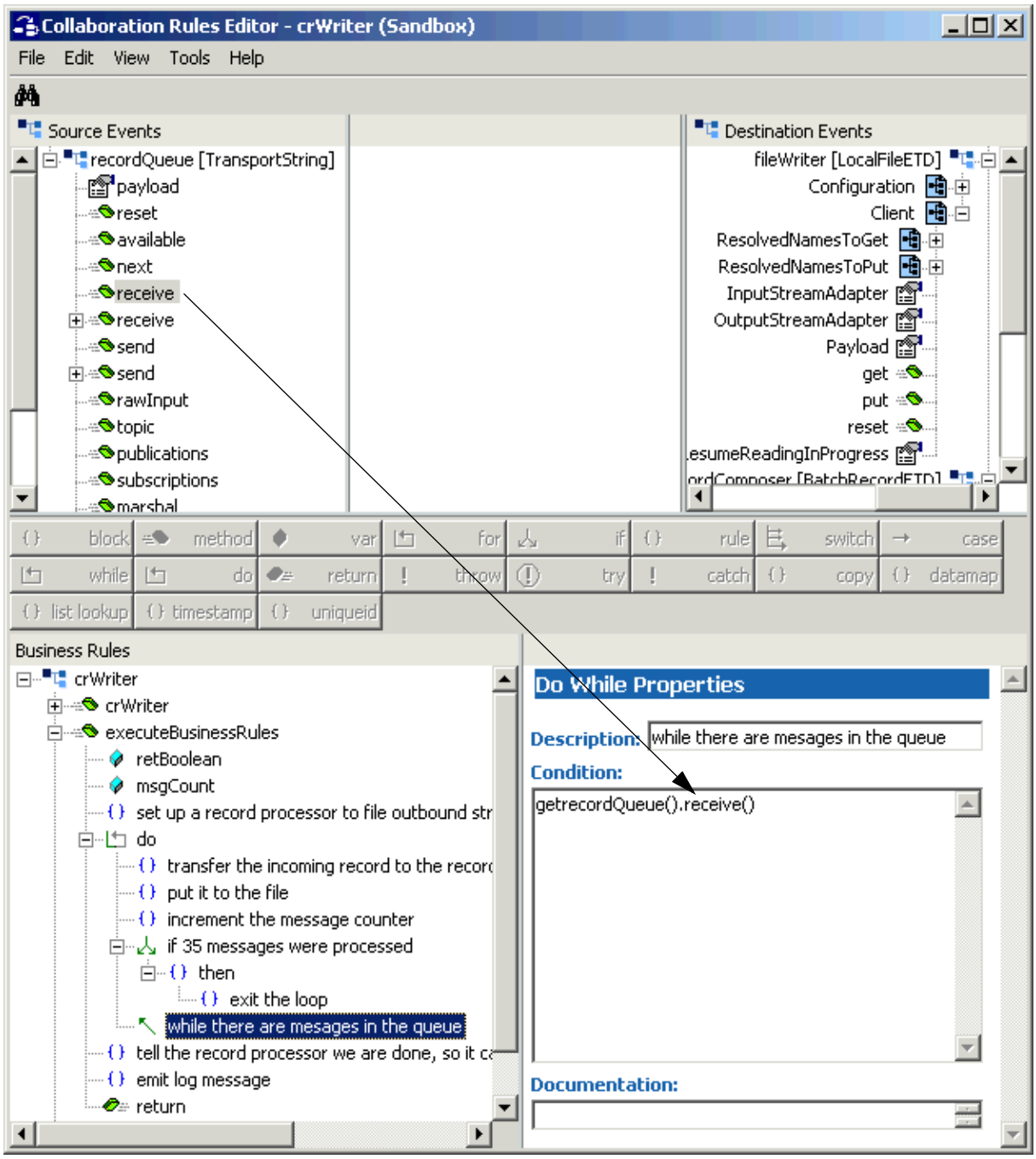
23 Enter a **break** in the **Rule** scroll box in the **Rule Properties** window as shown in Figure 100.

Figure 100 Collaboration Rules Editor: crWriter Exiting the Loop



24 Drag the first **receive** method under the **Source Event** into the **Rule** scroll box in the **Rule Properties** window (see Figure 101).

Figure 101 Collaboration Rules Editor: crWriter receive Method



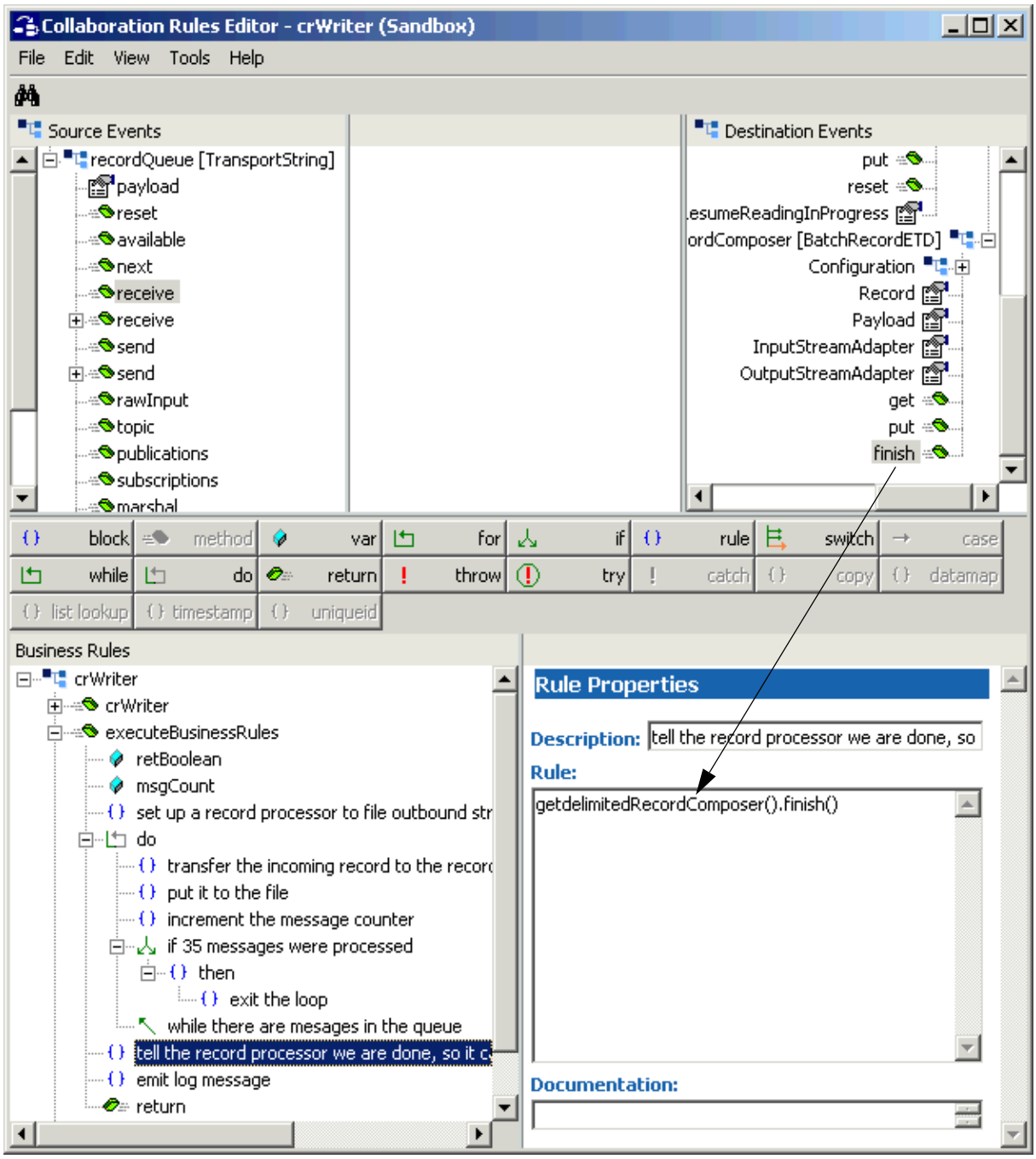
25 Name the new rule **while there are messages in the queue**.

26 To create the next Business Rule, click the **do** rule and select **Sibling** in the pop-up.

27 Name the new rule **tell the record processor we are done, so it can release the streaming link successfully** (see Figure 102 on page 230).

28 Drag the **finish** method from the Destination Event into the **Rule** scroll box in the **Rule Properties** window (see Figure 102).

Figure 102 Collaboration Rules Editor: crWriter finish Method

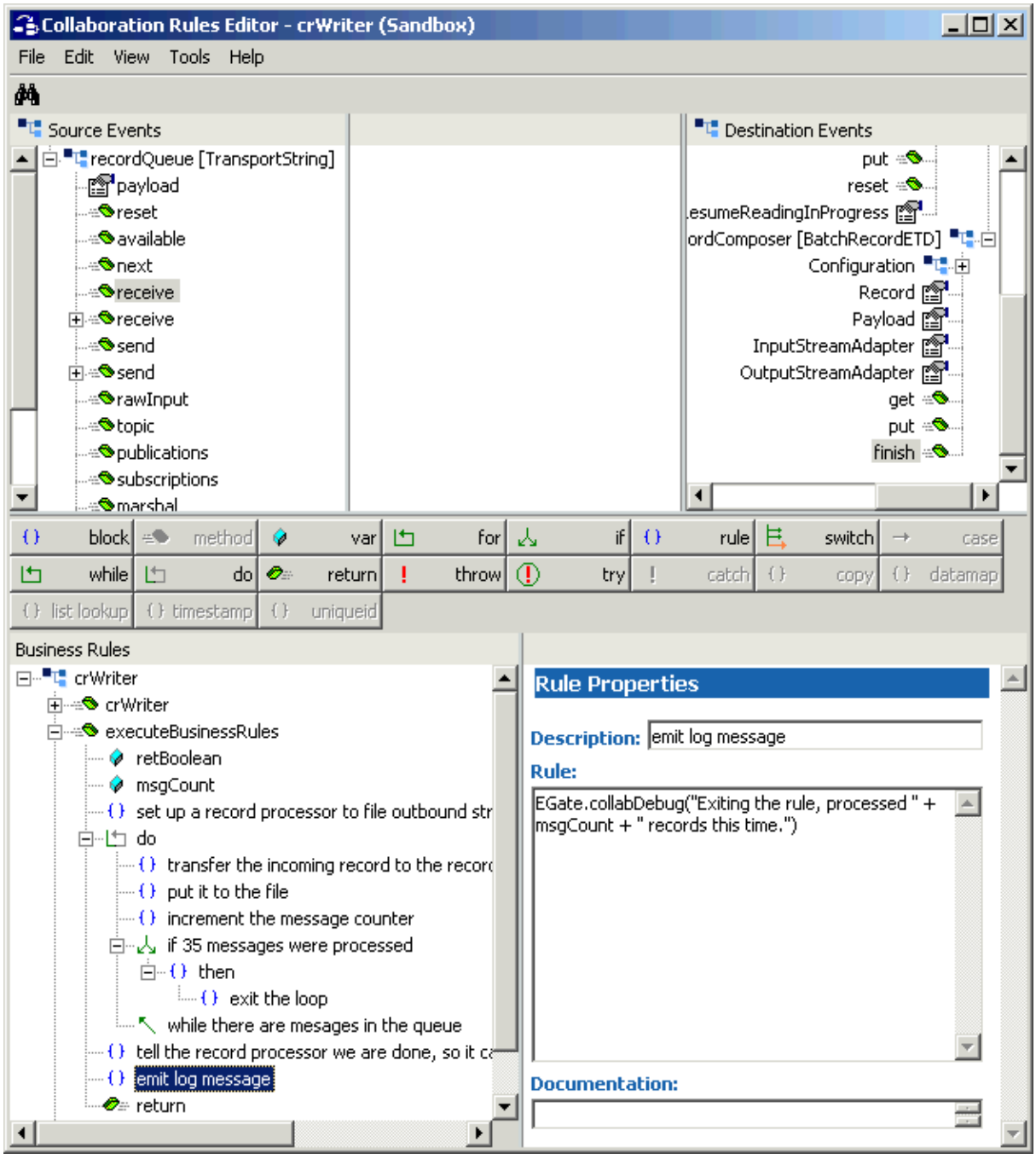


29 Click **rule** and name the new rule **emit log message** (see **Figure 103 on page 231**).

30 With **rule** selected, as shown in Figure 103, type the following text in the **Rule** scroll box in the **Rule Properties** window:

```
EGate.collabDebug("Exiting the rule, processed " + msgCount +
    " records this time.")
```

Figure 103 Collaboration Rules Editor: crWriter Log Message



You have now finished creating the Business Rules.

31 You must create a Collaboration Rules class or use one from the sample.

32 To save the Collaboration Rules file, click **Save** on the **File** menu. The **Save** dialog box appears.

33 Provide a name for the **.xpr** file (for this example, use **crWriter.xpr**) then click **Save**.

34 Compile and save this Collaboration Rule in the same way as you did the previous one.

35 Once the compilation is complete, you can exit the Collaboration Rules Editor.

Creating Collaborations

Collaborations are the components that receive and process Event Types, then forward the output to other e*Gate components or an external system.

Collaborations consist of the subscriber, which receives Events of a known type (sometimes from a given source), and the publisher, which distributes transformed Events to a specified recipient.

To create the Collaborations

1 In the e*Gate Schema Designer, select the Navigator's **Components** tab.

2 Open the host on which you want to create the Collaboration.

3 Select the Control Broker for this schema.

4 Select the **ewReader** e*Way to assign the Collaboration.

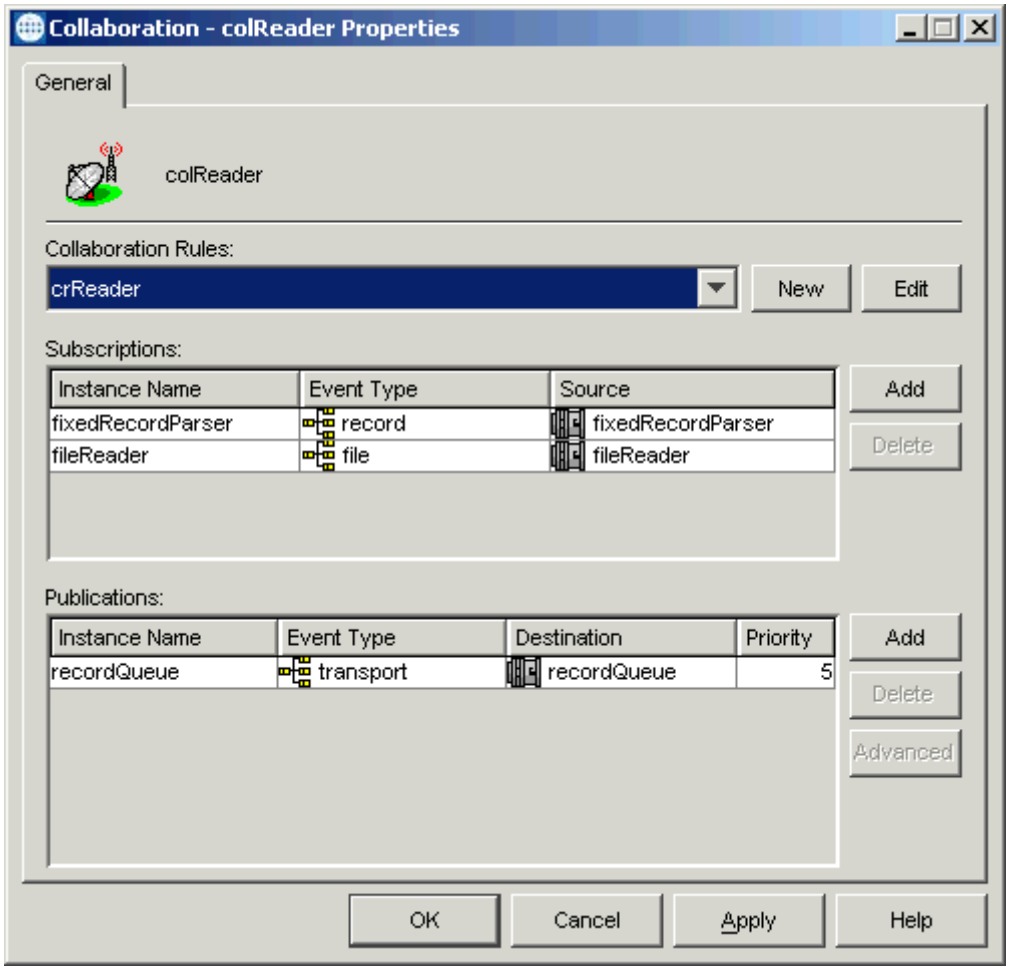
5 On the palette, click the **Collaboration** icon.

6 Enter the name (**colReader**) of the new Collaboration, then click **OK**.

7 Select the new Collaboration, then right-click to edit its properties.

8 The **Collaboration Properties** dialog box appears (see Figure 104).

Figure 104 colReader Properties Dialog Box



Configure the appropriate Collaboration properties as shown in the previous figure.

9 From the **Collaboration Rules** list, select the first Collaboration Rule that you created previously (**crReader**) for this Collaboration.

10 Click **OK** to close the dialog box and save your changes.

11 Select the **ewWriter** e*Way to assign the next Collaboration.

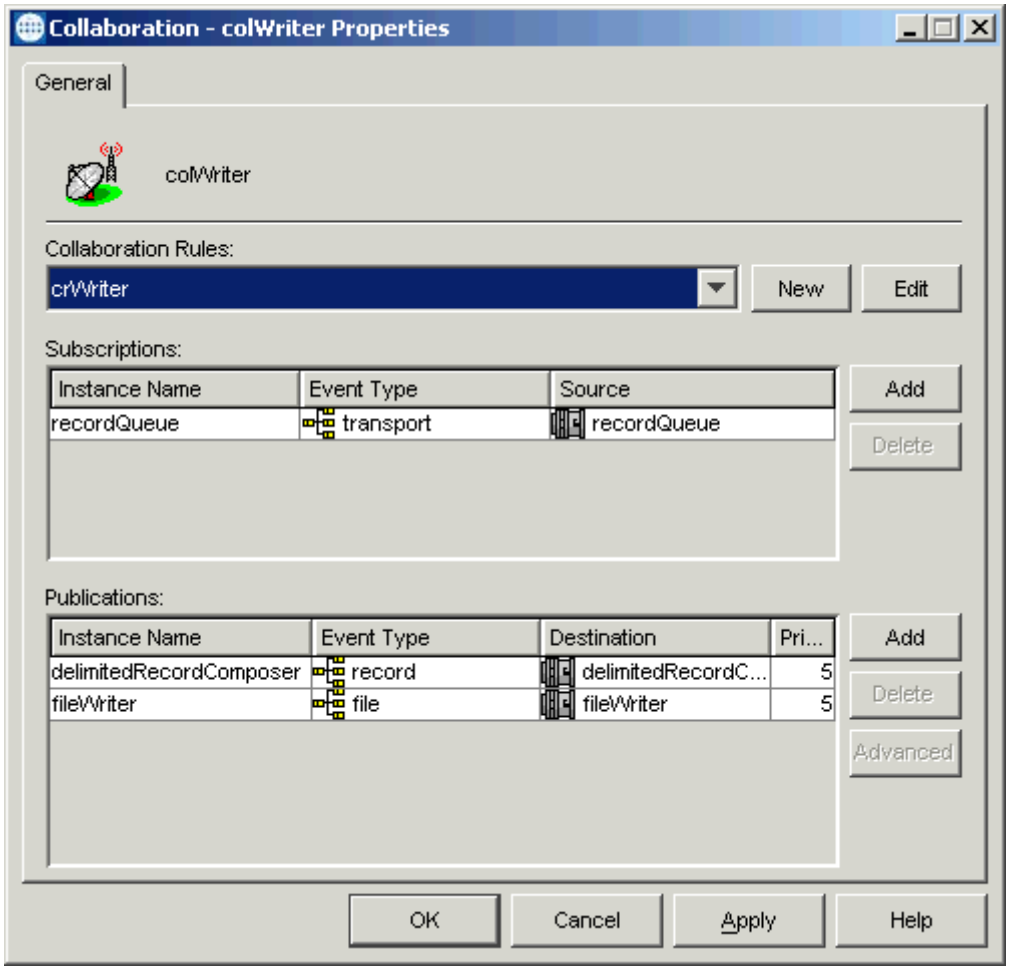
12 On the palette, click the **Collaboration** icon.

13 Enter the name (**colWriter**) of the new Collaboration, then click **OK**.

14 Select the new Collaboration, then right-click to edit its properties.

15 The **Collaboration Properties** dialog box appears (see Figure 105).

Figure 105 colWriter Properties Dialog Box



Configure the appropriate Collaboration properties as shown in the previous figure.

16 From the **Collaboration Rules** list, select the Collaboration Rule that you created previously (**crWriter**) for this Collaboration.

17 Click **OK** to close the dialog box and save your changes.

Running the Schema

For an explanation of how to run the schema see **"Running the Schema" on page 169**.

7.4 Sample Schema: FTP and ETD Extensibility

This section explains how to implement the FTP ETD with its extensibility features. In this schema, the ETD's FTP functionality has been modified by the user to customize specific features.

Note: *For more information on ETD extensibility and adding user-defined features, see* **Chapter 6**.

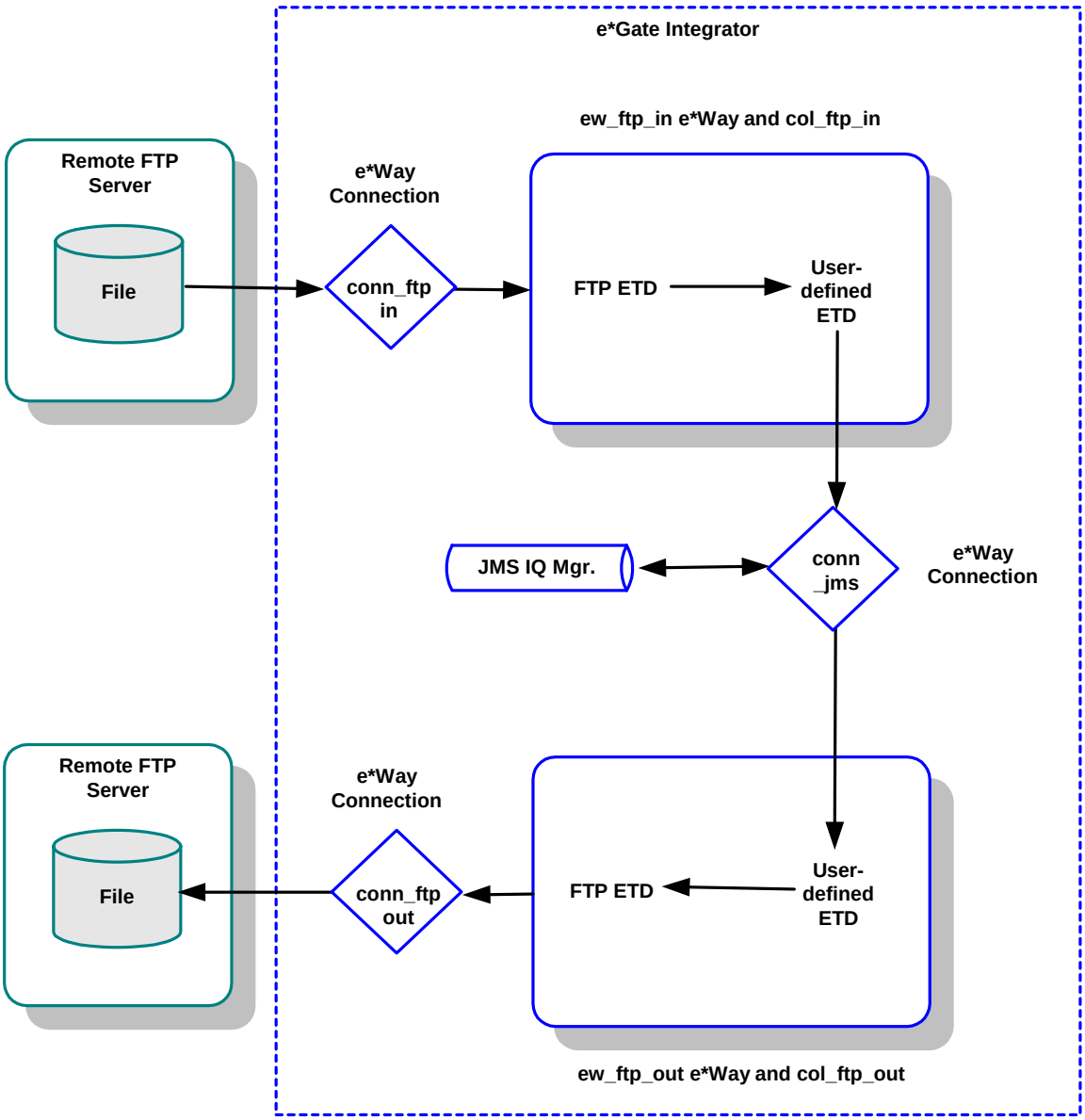
7.4.1 FtpExtensibilitySample Schema Overview

This section provides an overview of the sample schema and how it operates. The name of this schema is FtpExtensibilitySample, and it is contained in the import file **FtpExtensibilitySample.zip**.

Schema Setup

Figure 106 on page 236 shows a diagram of the schema's general architecture. The arrows show the direction of data flow. The remote FTP locations can be the same or two different FTP servers.

Figure 106 FtpExtensibilitySample Schema Diagram



Schema Operation

This sample schema has the following input/output setup:

- **Input**: An inbound e*Way retrieves a remote file from a remote FTP location and sends it as an Event to a JMS IQ Manager.
- **Output**: An outbound e*Way gets the Event from the IQ Manager and stores it as another remote FTP file.

Before you run this schema, make sure an FTP input file is provided and is accessible to e*Gate. Because no post-transfer command is specified, the input file is transferred continually until you shut down the e*Way.

The user-defined Java classes for this schema are:

- **FtpFileClientImplSample0**: For the inbound e*Way's client interface implementation.
- **FtpFileProviderImplSample0**: For the outbound e*Way's provider interface implementation.

The appropriate e*Way Connections have been configured to use these classes, under the **Extensions** parameters section. See **"Creating and Configuring e*Way Connections" on page 246** for details.

Once the schema starts running, e*Gate displays log messages, such as:

```
FtpFileClientImplSample0.get():
```

The previous message is for an FTP **get()** operation. Messages like it mean the user-defined class **FtpFileClientImplSample0** is being utilized.

Note: *The purpose of this sample schema is only to demonstrate the extensibility of the FTP ETD, so there is no error-handling logic in the Collaboration Rules. See* **"Sample Schema: Basic FTP With Streaming" on page 129** *for a sample with error-handling logic.*

Schema Components

The FtpExtensibilitySample schema with a user-defined FTP extensibility implementation consists of the following main e*Gate components:

- **ew_ftp_in**: Inbound Multi-Mode e*Way that brings the file from a remote FTP server into e*Gate.
- **ew_ftp_out**: Outbound Multi-Mode e*Way that sends the file from e*Gate to a remote FTP server.
- **col_ftp_in**: Collaboration for the **ew_ftp_in** e*Way.
  - **cr_ftp_in**: Collaboration Rule for **col_ftp_in**.
- **col_ftp_out**: Collaboration for the **ew_ftp_out** e*Way.
  - **cr_ftp_out**: Collaboration Rule for **col_ftp_out**.
- **localhost_iqmgr**: Oracle SeeBeyond JMS IQ Manager.
- **conn_ftp_in**: e*Way Connection for the **ew_ftp_in** e*Way.
- **conn_jms**: e*Way Connection for the JMS IQ Manager **localhost_iqmgr**.
- **conn_ftp_out**: e*Way Connection for the **ew_ftp_out** e*Way.

7.4.2 Creating the FtpExtensibilitySample Schema

This section explains the basic steps in creating the sample schema, focusing on how to add and configure the FTP extensibility feature to the schema.

Creating a New Schema

This step is the same as for the BasicFtpSample schema. The name of the new schema is **FtpExtensibilitySample**. Follow the procedures provided under **"Creating a New Schema" on page 132**.

Creating Event Types and ETDs

The FtpExtensibilitySample schema uses the following Java-based ETDs:

- **FtpETD.xsc**
- **transport.xsc**

The e*Way installation includes the **FtpETD.xsc** ETD. You must create the **transport.xsc** ETD yourself.

Event Types and ETDs

Using the Schema Designer, create the Event Types and associate them with ETDs as shown in Table 9.

Table 9 FtpExtensibilitySample Schema Event Types and ETDs

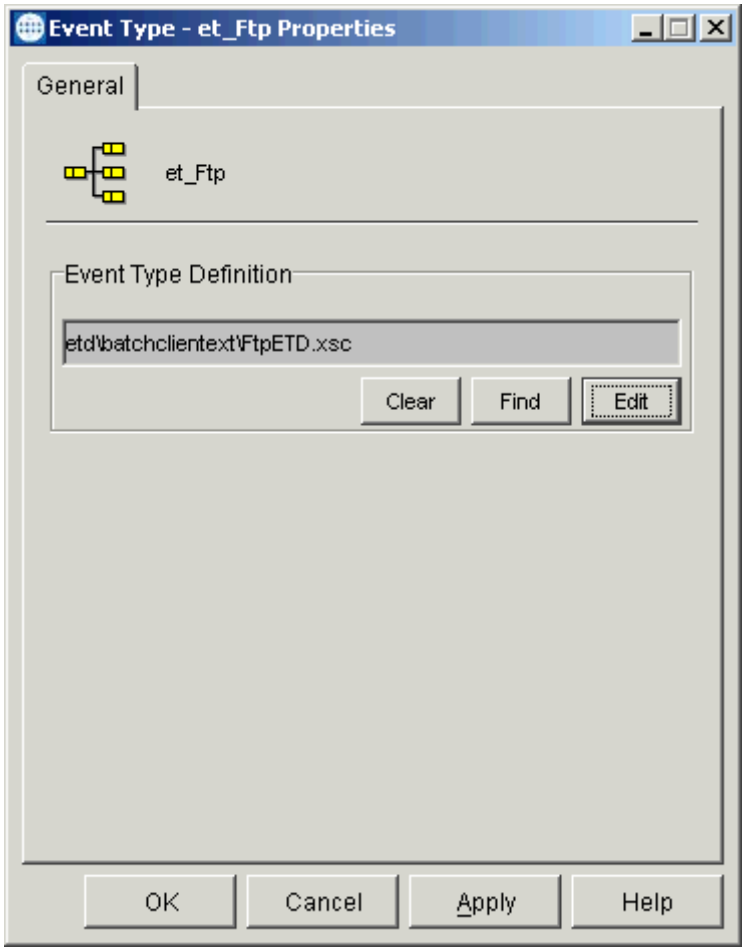
Event Type	Type of ETD	ETD File Name
et_Ftp	FTP ETD	FtpETD.xsc
et_transport	User-defined ETD	transport.xsc

See **"Creating Event Types and ETDs" on page 133** for instructions on how to create Event Types, locate, and how to associate each Event Type with its ETD.

To create the et_Ftp Type

1. Highlight the **Event Type** folder on the **Components** tab of the e*Gate Schema Designer.
2. On the palette, click the icon to create a new Event Type.
3. Enter the name of the Event Type (**et_Ftp**), then click **OK**.
4. Select the new **Event Type**, then right-click to edit its properties.
5. The **Event Type Properties** dialog box appears (see **Figure 107 on page 239**).

Figure 107 et_Ftp Event Type Properties Dialog Box



6 Click **Find** and from the resulting dialog box select the **FtpETD.xsc** ETD to associate with the new Event Type you created.

7 When you have found the file, click **Select**.

8 When you are finished with the **Event Type Properties** dialog box, click **OK** to close it and save your changes.

To create the et_transport Event Type and ETD

1 Highlight the **Event Type** folder on the **Components** tab of the e*Gate Schema Designer.

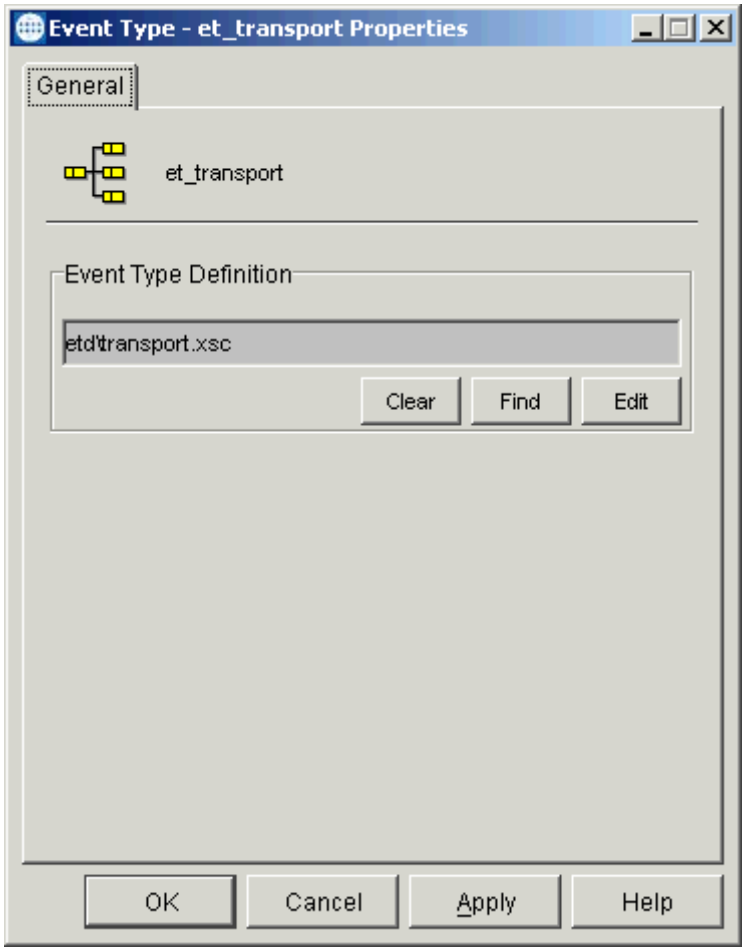
2 On the palette, click the icon to create a new Event Type.

3 Enter the name of the Event Type (**et_transport**), then click **OK**.

4 Select the new **Event Type**, then right-click to edit its properties.

5 The **Event Type Properties** dialog box appears (see **Figure 108 on page 240**).

Figure 108 et_transport Event Type Properties Dialog Box



Because this Event Type is for a user-defined ETD, you must use the ETD Editor feature of the Schema Designer to create this ETD.

6 From the Schema Designer menu bar, select **Options** and click **Default Editor**. For this schema, set the default to **Java**.

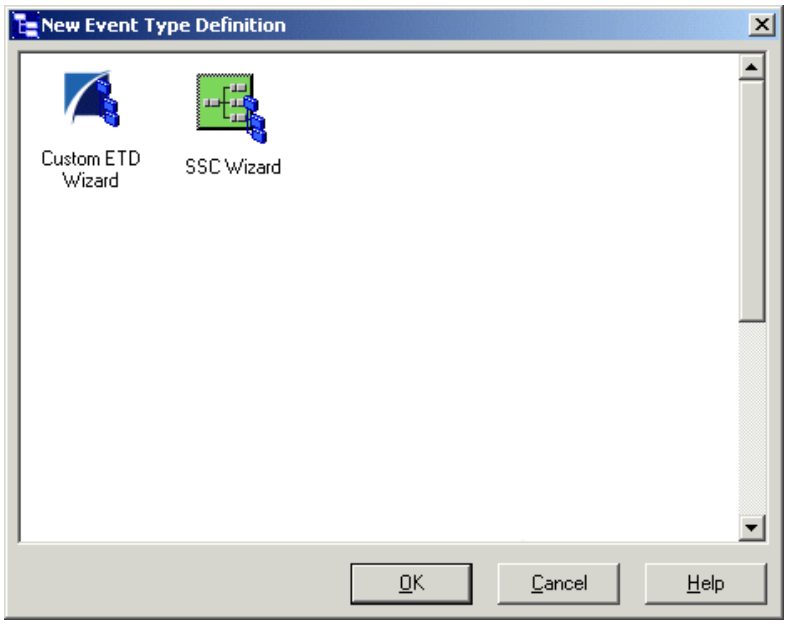
7 Click the icon in the toolbar for the ETD Editor.

The ETD Editor Main window appears.

8 Click the **New Event Type Definition** icon in the window's tool bar.

The **New Event Type Definition** dialog box appears (see **Figure 109 on page 241**).

Figure 109 New Event Type Definition Dialog Box



9 Double-click the **Custom ETD Wizard** icon.

The **Introduction** dialog box for the Custom ETD wizard appears (see Figure 110).

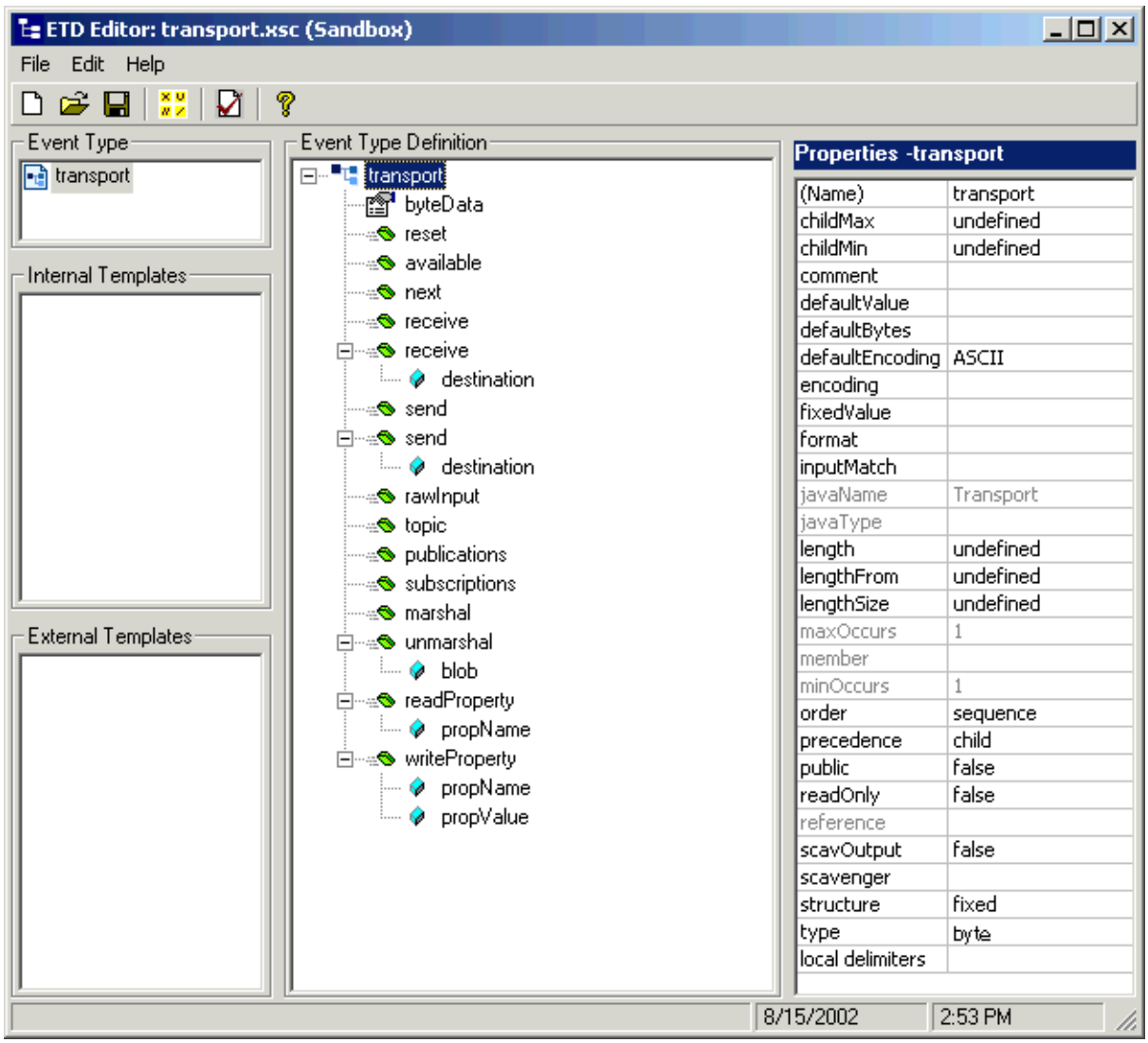
Figure 110 Custom ETD Wizard: Introduction

10 Follow the instructions given by the wizard to create a new ETD and name the root node **transport**. Be sure to give it an appropriate Java package name.

The contents of the file appear in the ETD Editor window.

11 Assign the new ETD the properties shown in Figure 111.

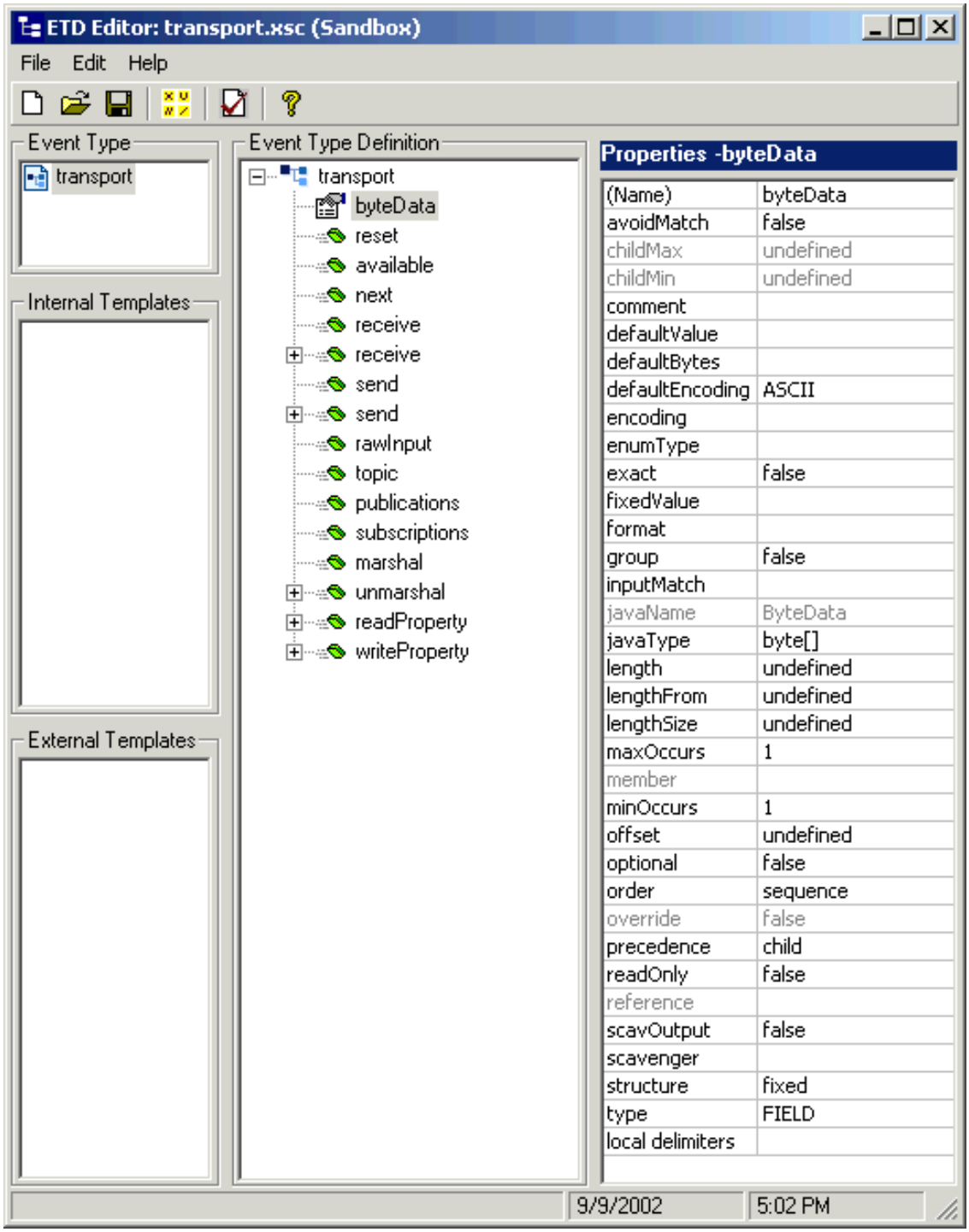
Figure 111 transport.xsc in ETD Editor: Root Node



Note: *For complete instructions on how to use the ETD Editor, see the* ***e*Gate Integrator User's Guide****.*

12 Add **byteData** as a child node under the root node. Be sure that you give this node the properties shown in Figure 112.

Figure 112 transport.xsc in ETD Editor: ByteData Node



13 Finish building the ETD until it looks like the one shown in Figure 112.

14 When you are finished editing the ETD, save your changes and close the ETD Editor. Be sure to create the ETD (**transport.xsc**) in the **client\etd** directory.

15 From the Schema Designer, open the **Event Type Properties** dialog box for the **et_transport** Event Type and associate it with the **transport.xsc** ETD you created.

16 When you are finished with the dialog box, click **OK** to close it and save your changes.

Creating and Configuring e*Ways

You must create the following Multi-Mode e*Ways:

- Inbound: **ew_ftp_in**
- Outbound: **ew_ftp_out**

For details on how to create and configure e*Ways, see **"Creating and Configuring e*Ways" on page 135**.

Figure 113 on page 245 and **Figure 114 on page 246** show the **e*Way Properties** dialog boxes for the **ew_ftp_in** and **ew_ftp_out** e*Ways. Configure and name these e*Ways as shown in both of the figures. Give each **.cfg** file the name of its e*Way.

Figure 113 ew_ftp_in e*Way Properties Dialog Box

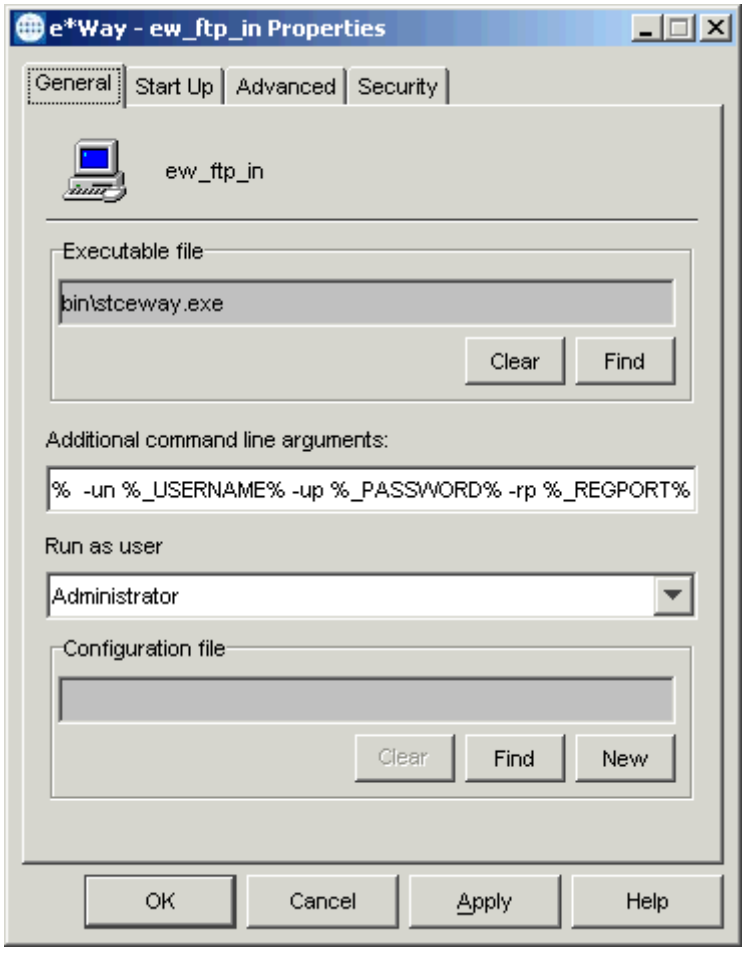
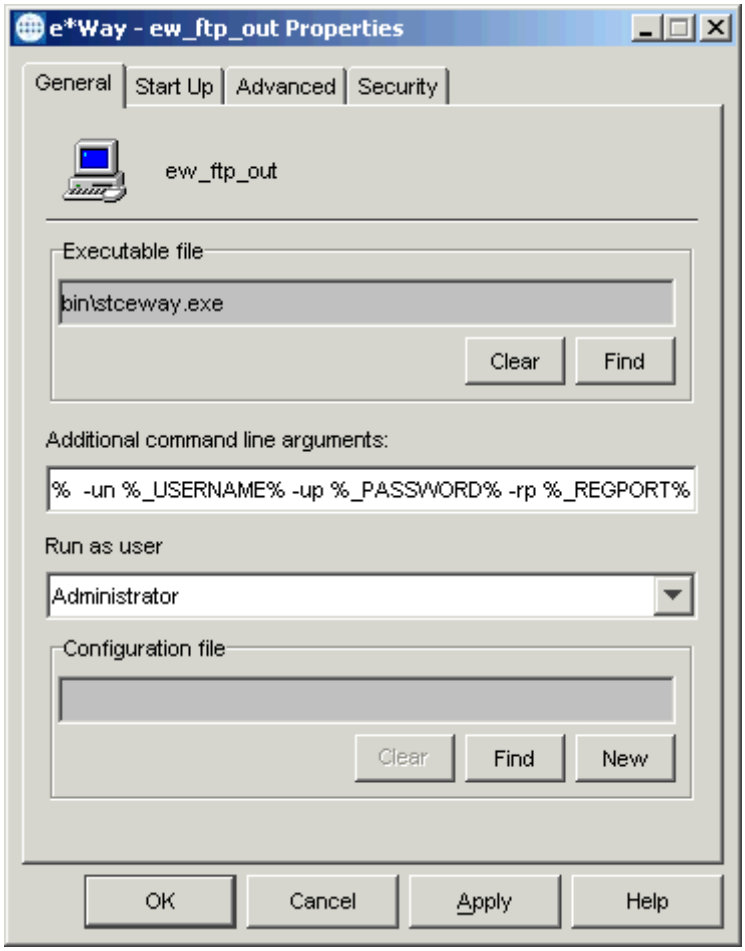


Figure 114 ew_ftp_out e*Way Properties Dialog Box



Creating and Configuring e*Way Connections

In general, you can create and configure the e*Way Connections for this schema in the same way as those for the other sample schemas. The e*Way Connection configuration file contains the connection information needed to communicate with the remote FTP server and the JMS IQ Manager.

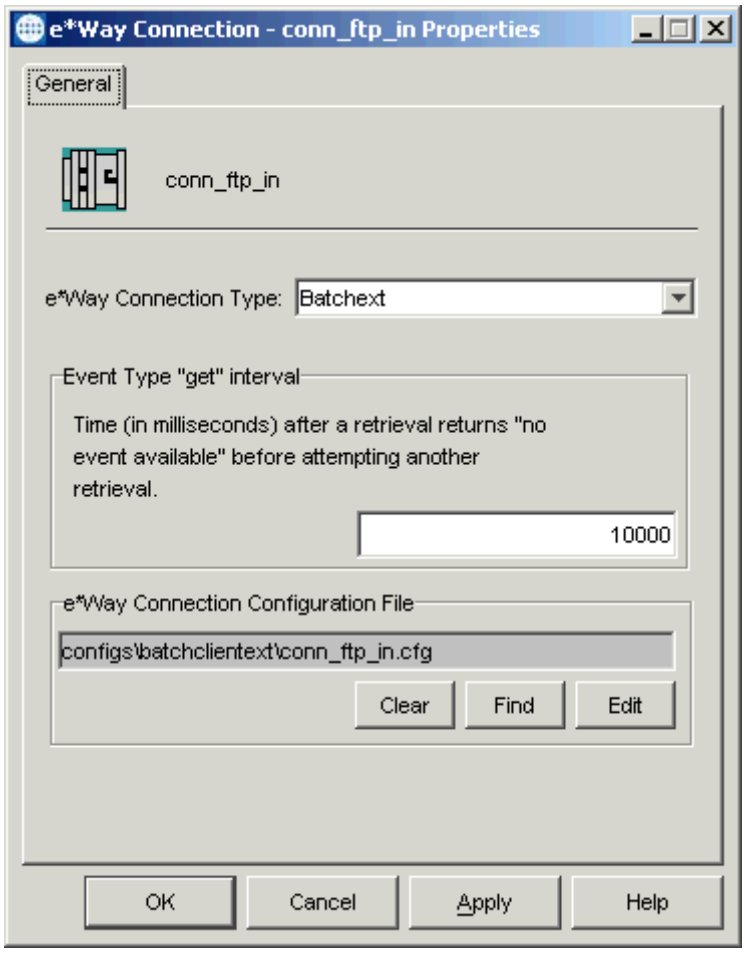
You must create and configure the following e*Way Connections:

To create and configure the conn_ftp_in e*Way Connection

1. Highlight the **e*Way Connection** folder on the **Components** tab of the e*Gate Schema Designer.
2. On the palette, click the icon to create a new e*Way Connection.
3. Enter the name of the e*Way Connection (**conn_ftp_in**), then click **OK**.
4. Select the new **e*Way Connection**, then right-click to edit its properties.

5 When the **e*Way Connection Properties** dialog box opens, select **Batchext** from the **e*Way Connection Type** drop-down menu (see Figure 115).

Figure 115 conn_ftp_in e*Way Connection Properties Dialog Box



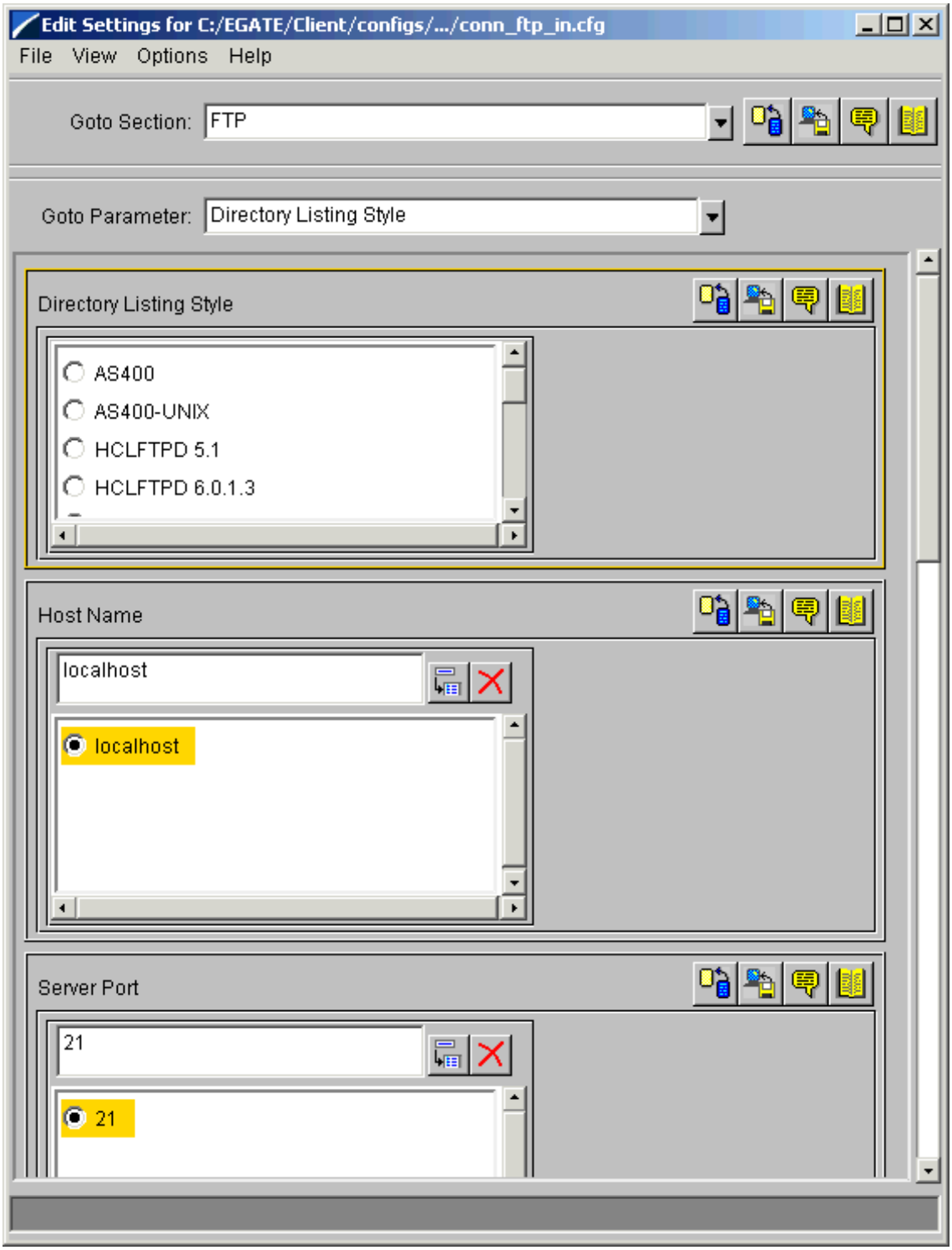
Configure the e*Way Connection properties as shown in the previous figure.

6 Under **e*Way Connection Configuration File**, click **New** (where the **Edit** button is in the previous figure). Select the **FtpETD**.

The e*Way Configuration Editor Main window opens. Use this interface to select the desired parameters, including those that correspond to the remote FTP system you are using. See **"FtpETD: Configuration Parameters" on page 36** for details.

7 Select the **FTP** settings (see **Figure 116 on page 248**).

Figure 116 e*Way Configuration Editor: conn_ftp_in FTP Settings



Note: *See the* ***e*Gate Integrator User's Guide*** *for complete information on how to use the e*Way Configuration Editor.*

8 Under the **FTP** settings, accept the default settings for all parameters except for:

- **Directory Listing Style**
- **Host Name**
- **Server Port**
- **User Name**
- **Password**

You must enter your system settings for these parameters.

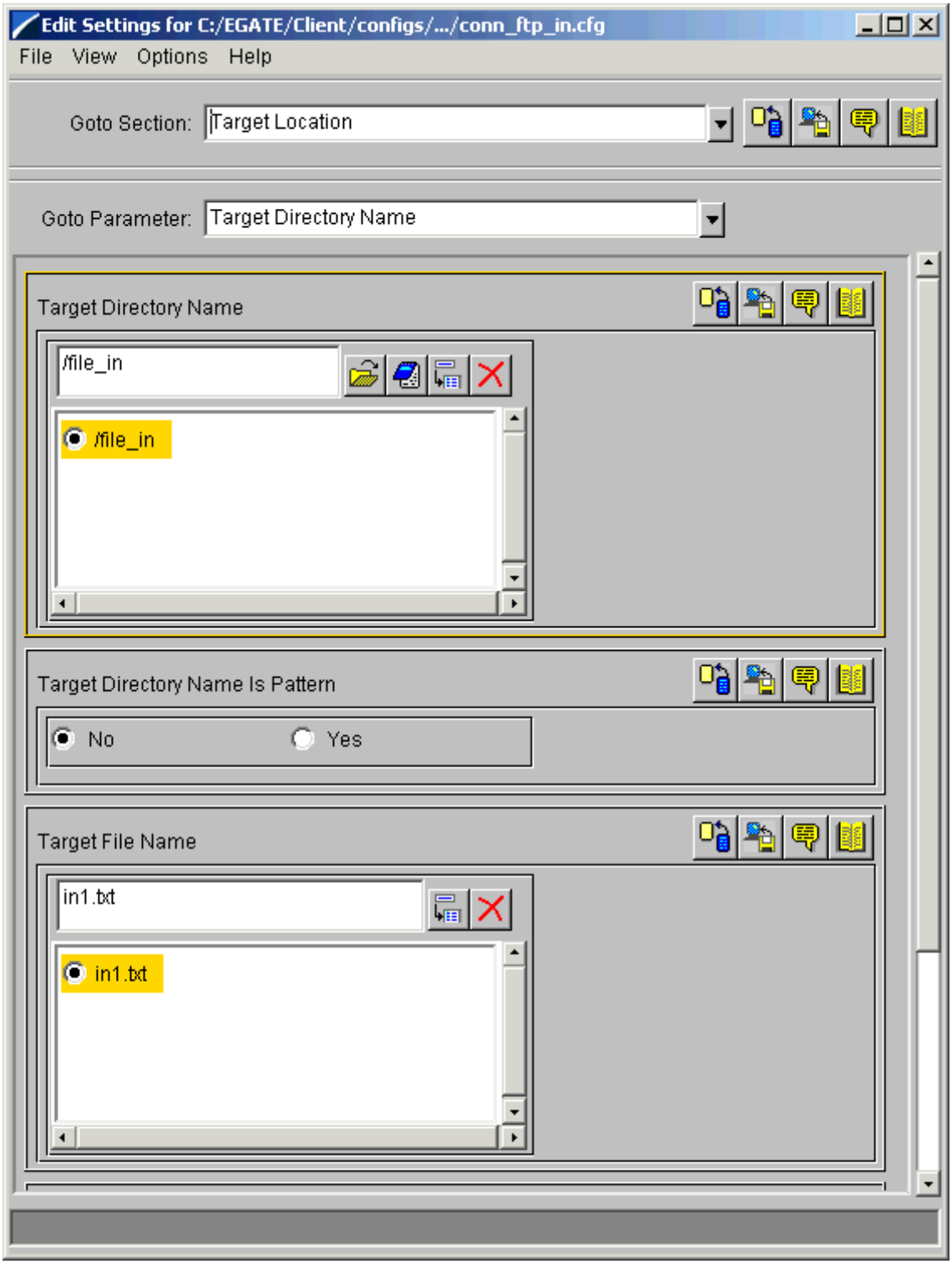
9 Select the **Target Location** settings (see **Figure 117 on page 250**).

10 Under the **Target Location** settings, accept the default settings for all parameters except for:

- **Target Directory Name**
- **Target File Name**

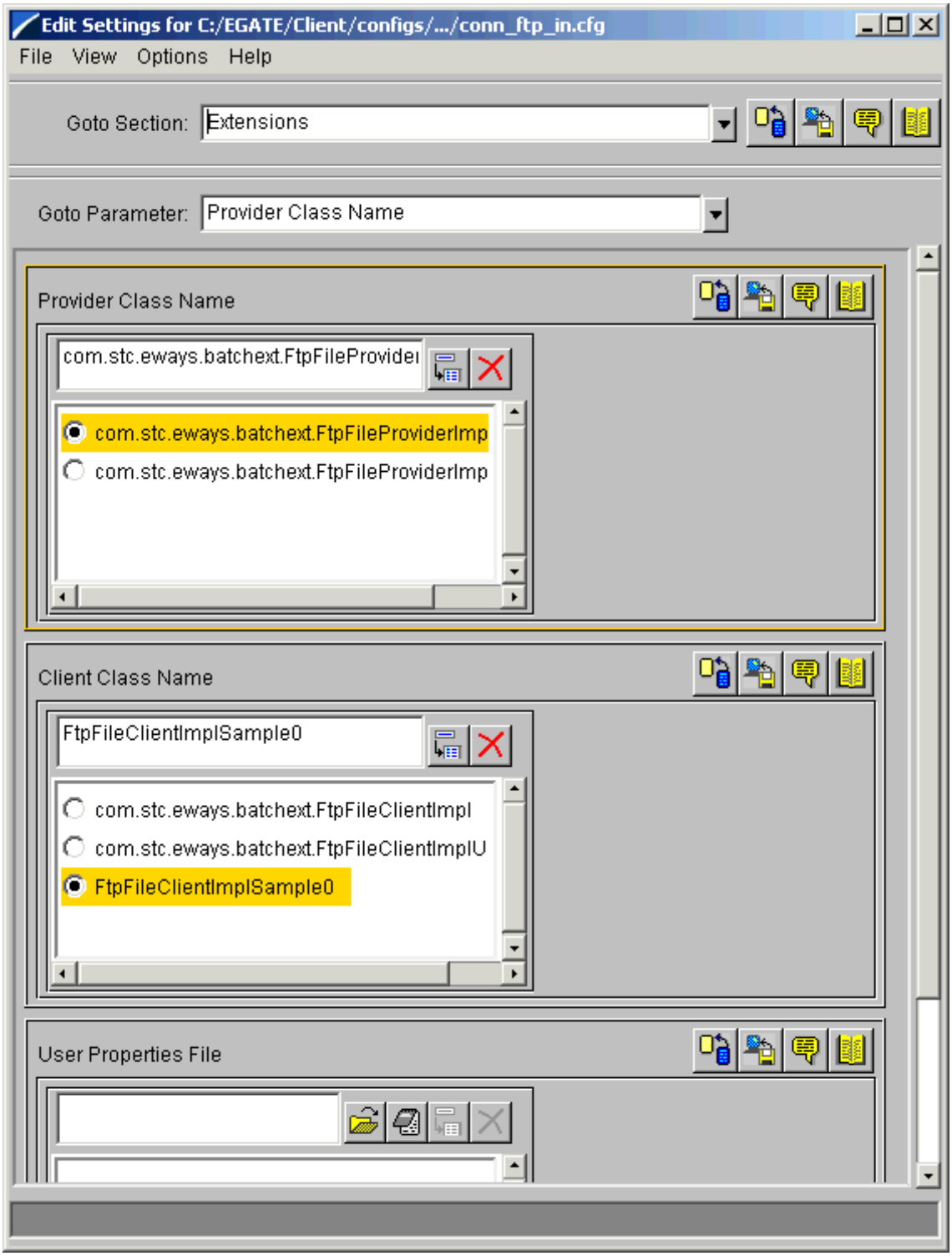
You must enter your system settings for these parameters (see **Figure 117 on page 250**).

Figure 117 e*Way Configuration Editor: conn_ftp_in Target Location Settings



11 In addition to the parameters listed previously, you must also enter parameters for your user-defined extensions. Select the **Extensions** settings (see Figure 118).

Figure 118 e*Way Configuration Editor: conn_ftp_in Extensions Settings



12 Select the **Provider Class Name** as shown in Figure 118 on page 251.

13 Under the **Client Class Name**, enter your user-defined class **FtpFileClientImplSample0**, as shown in Figure 118 on page 251.

14 When you are finished, save the **.cfg** file as the name of the e*Way Connection, close the e*Way Configuration Editor, and promote the file to run time.

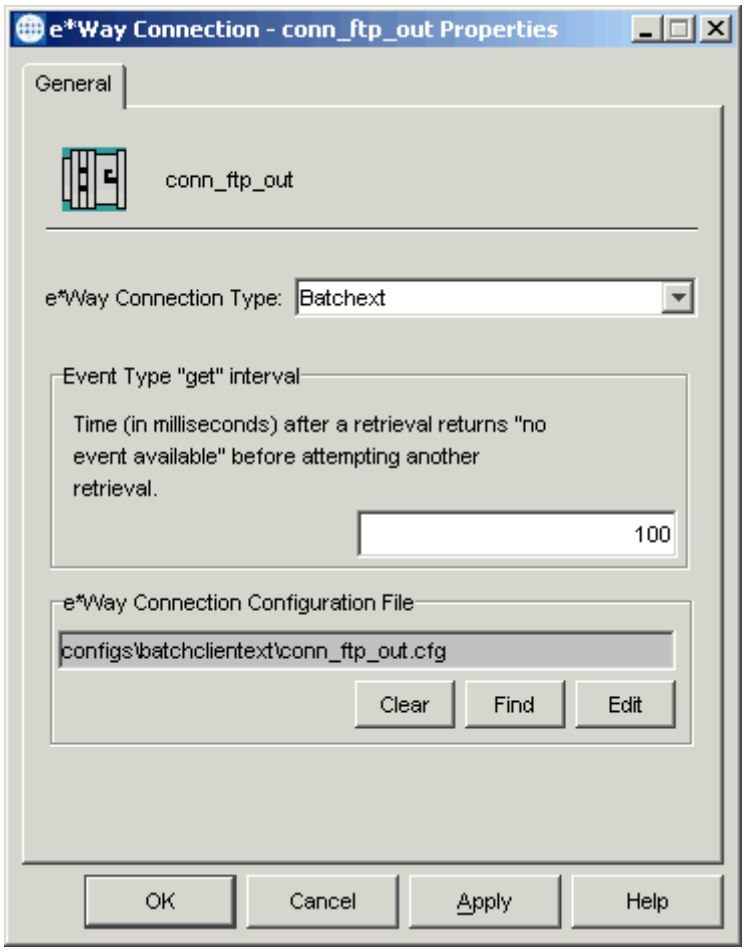
15 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the conn_ftp_out e*Way Connection

1 Access the **e*Way Connection Properties** dialog box for this component in the same way as you did for the previous e*Way Connection.

2 Set the properties as shown in (see Figure 119).

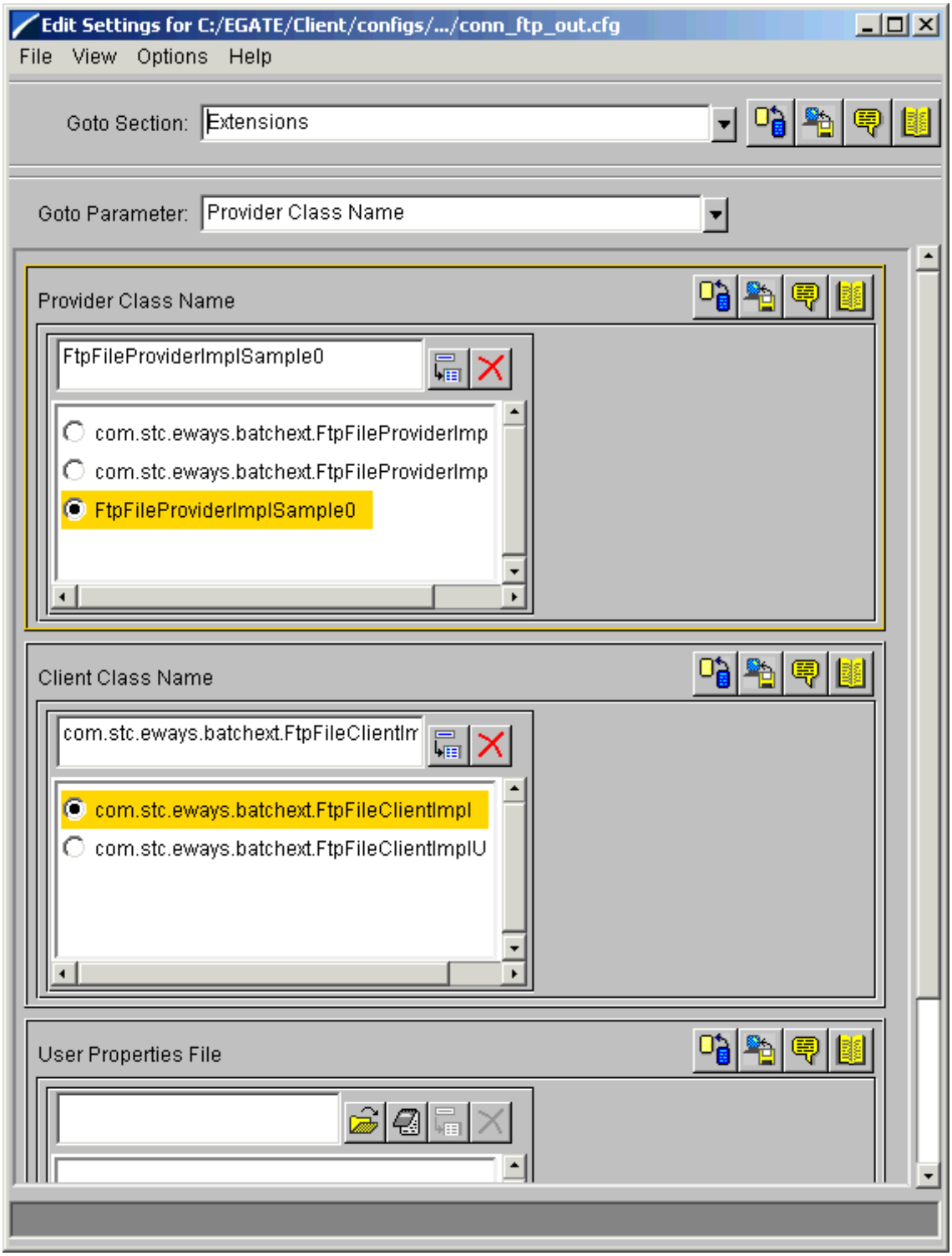
Figure 119 conn_ftp_out e*Way Connection Properties Dialog Box



3 Set the **FTP** and **Target Location** parameters for this e*Way Connection, using the appropriate information from the FTP system you are interfacing with.

4 You must also enter parameters for your user-defined extensions. Select the **Extensions** settings (see Figure 120).

Figure 120 e*Way Configuration Editor: conn_ftp_out Extensions Settings



5 Select the **Client Class Name** as shown in **Figure 120 on page 253**.

6 Under the **Provider Class Name**, enter your user-defined class **FtpFileProviderImplSample0**, as shown in **Figure 120 on page 253**.

7 When you are finished, save the **.cfg** file as the name of the e*Way Connection, close the e*Way Configuration Editor, and promote the file to run time.

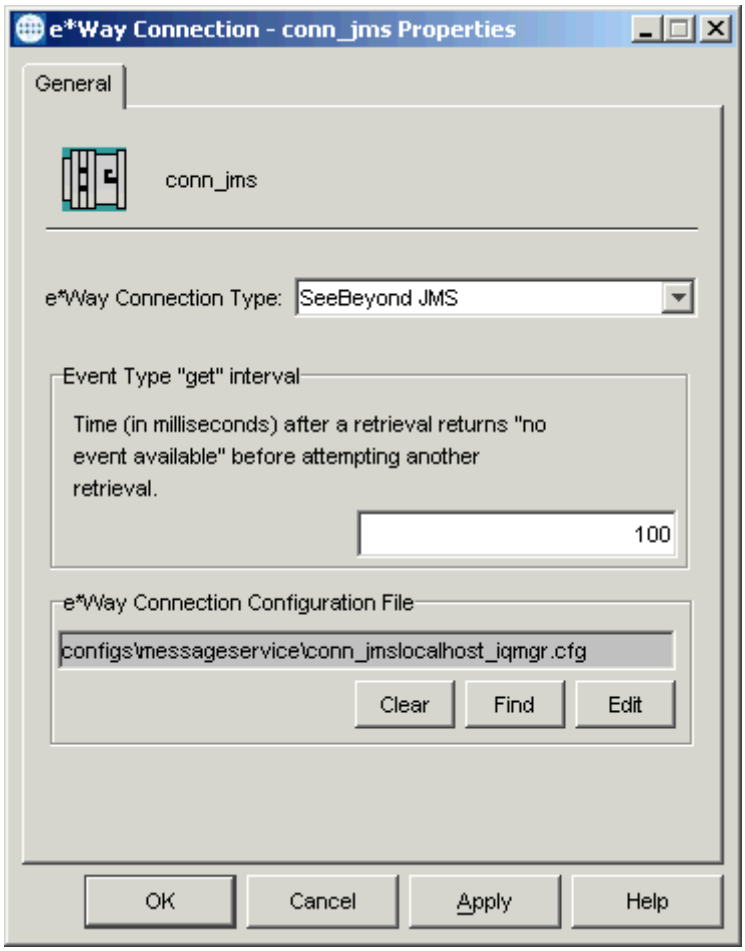
8 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the conn_jms e*Way Connection

1 Access the **e*Way Connection Properties** dialog box for this component in the same way as you did for the previous e*Way Connection.

2 Set the properties as shown in (see Figure 121).

Figure 121 conn_jms e*Way Connection Properties Dialog Box



3 There are some parameters for this component that you must set. See **"To create and configure the recordQueue e*Way Connection" on page 197**, and set the parameters for this component in the same way as they are set for that component, using your own system settings.

Checking the IQ Manager

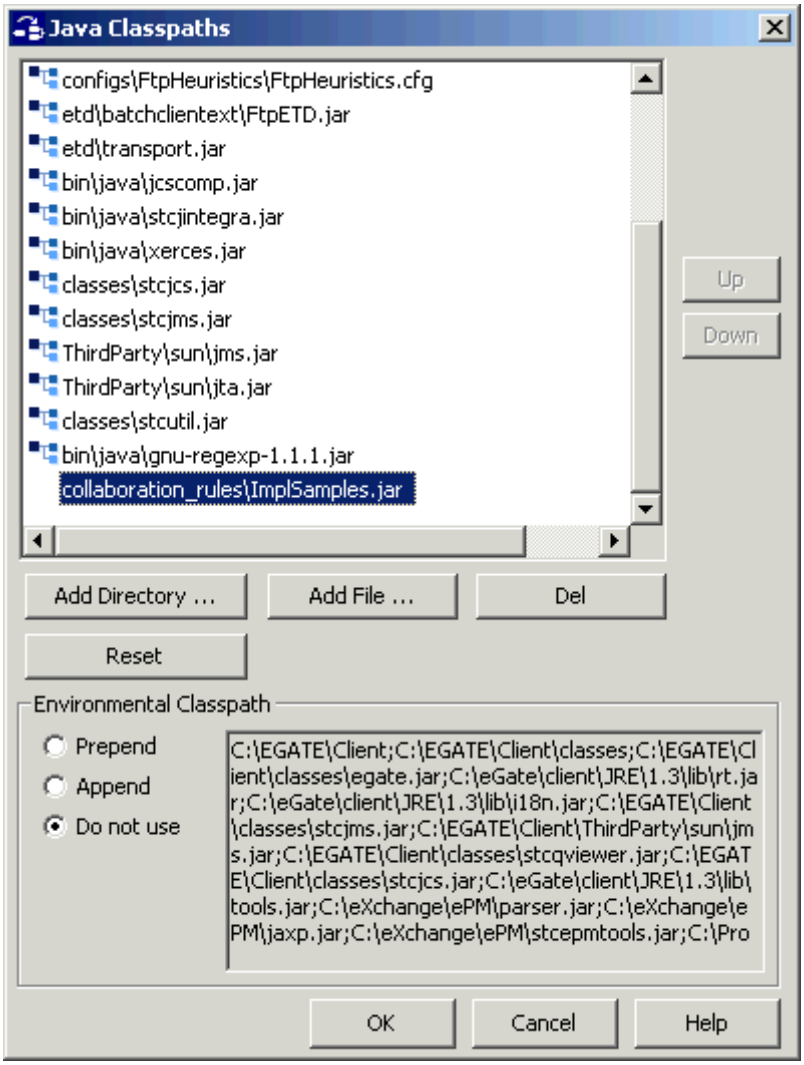
By default, the IQ Manager in your schema is configured to use the Oracle SeeBeyond JMS IQ Manager. See **"Checking the IQ Manager" on page 201** for more information.

Creating Collaboration Rules

Before creating your Collaboration Rules, be sure to check the Java user-defined implementation file, **ImplSamples.jar**, provided with this schema. This file shows how to extend the standard FTP ETD implementation. Ensure that this file is in the Java classpath. To verify its location, open the Collaboration Rules Editor's Main window, click the **Tools** menu, and select **Options** (see **Figure 122 on page 255**).

Note: *For more information, see the Java source programs* ***FtpFileClientImplSample0.java*** *and* ***FtpFileProviderImplSample0.java*** *contained in the* ***ImplSamples.jar*** *file.*

Figure 122 Collaboration Rules Editor: Java Classpaths Dialog Box

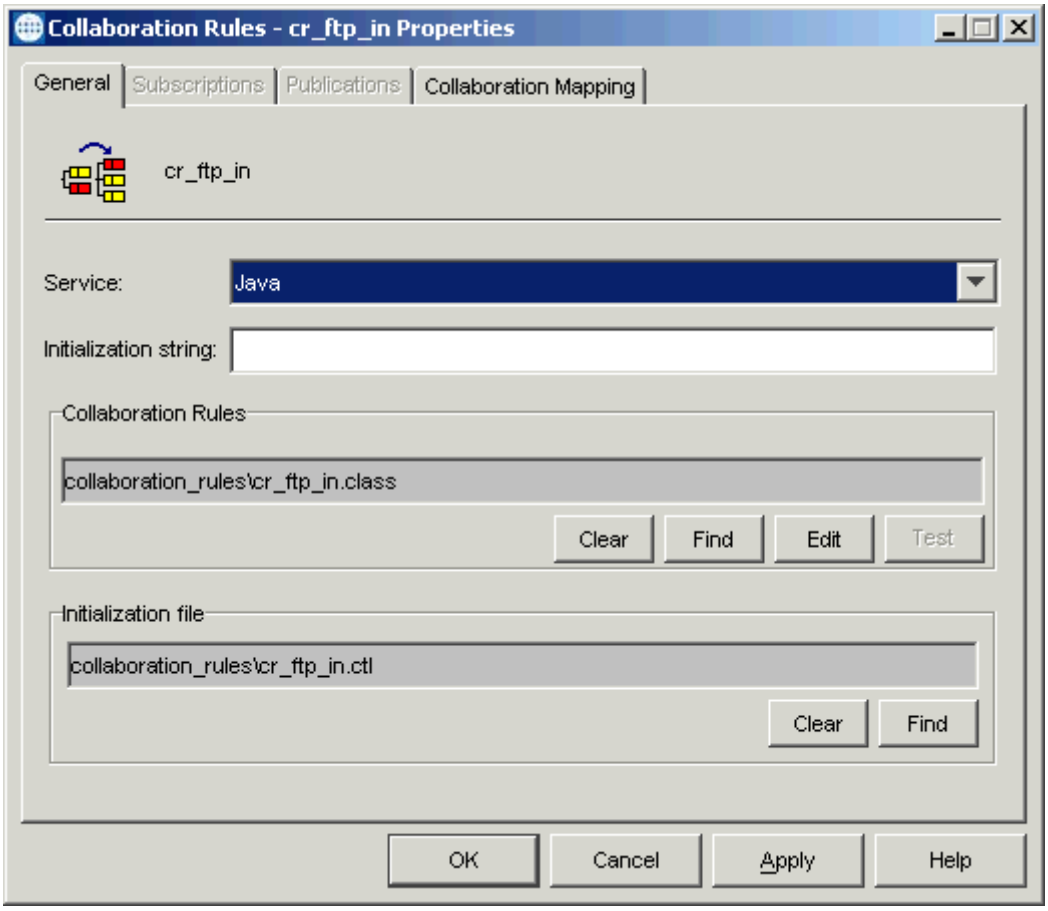


The FtpExtensibilitySample schema uses two Collaboration Rules, one for each of its e*Ways. Create and set up the Collaboration Rules as follows:

To create the cr_ftp_in Collaboration Rules file

1. Select the **Components** tab in the e*Gate Schema Designer.
2. In the Navigation pane, select the **Collaboration Rules** folder.
3. On the palette, click the **Collaboration Rules** icon.
4. Enter the name of the new Collaboration Rule, then click **OK.** Use **cr_ftp_in** for this example, for the **ew_ftp_in** e*Way's Collaboration, **col_ftp_in**.
5. Select the new **Collaboration Rule**, then right-click to edit its properties.
6. The **Collaboration Rules Properties** dialog box appears (see Figure 123).

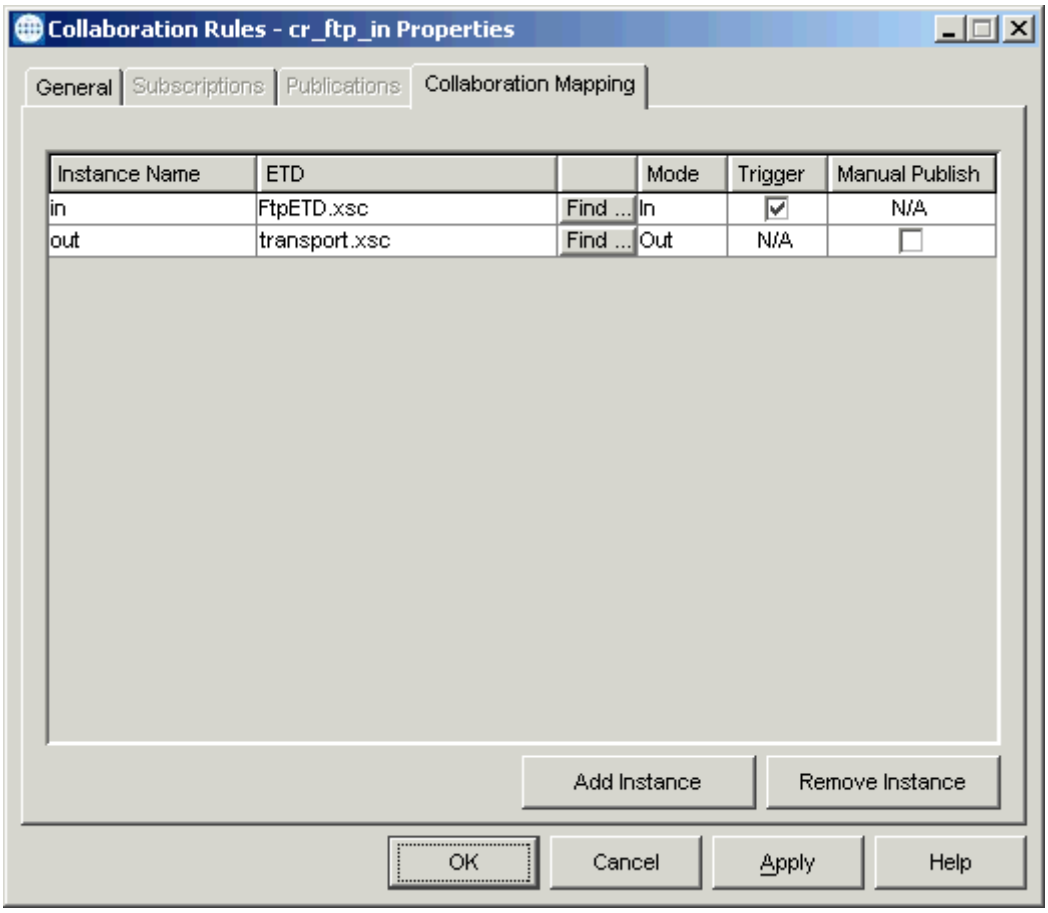
Figure 123 Collaboration Rules Properties Dialog Box for cr_ftp_in: General Tab



7. On the **General** tab in the dialog box select the **Java Collaboration Service**.
8. In the **Initialization String** text box, enter any required initialization string that the Collaboration Service may require. This field can be left blank.
9. Click the **Collaboration Mapping** tab.

10 Using the **Add Instance** button, create instances to coincide with the ETDs (see Figure 124).

Figure 124 Collaboration Rules Properties Dialog Box for cr_ftp_in: Mapping Tab



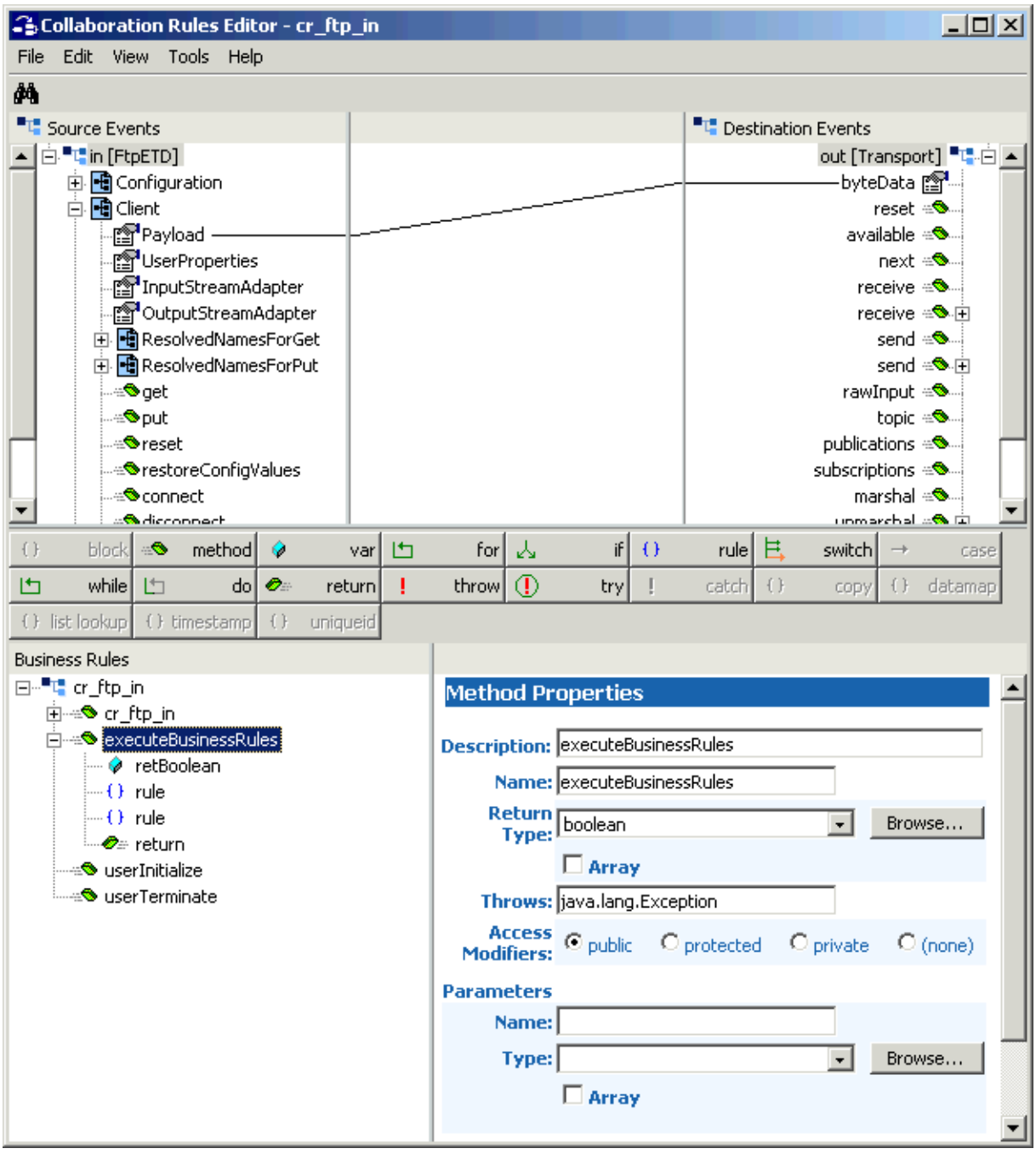
Configure the rest of **cr_ftp_in** as shown in the previous figure.

11 Select the **General** tab again, then click **New** (where the **Edit** button is in **Figure 123 on page 256**).

The Collaboration Rules Editor Main window opens.

12 After the window has opened, expand the source and destination Events, as well as the Business Rules. Figure 125 shows the results.

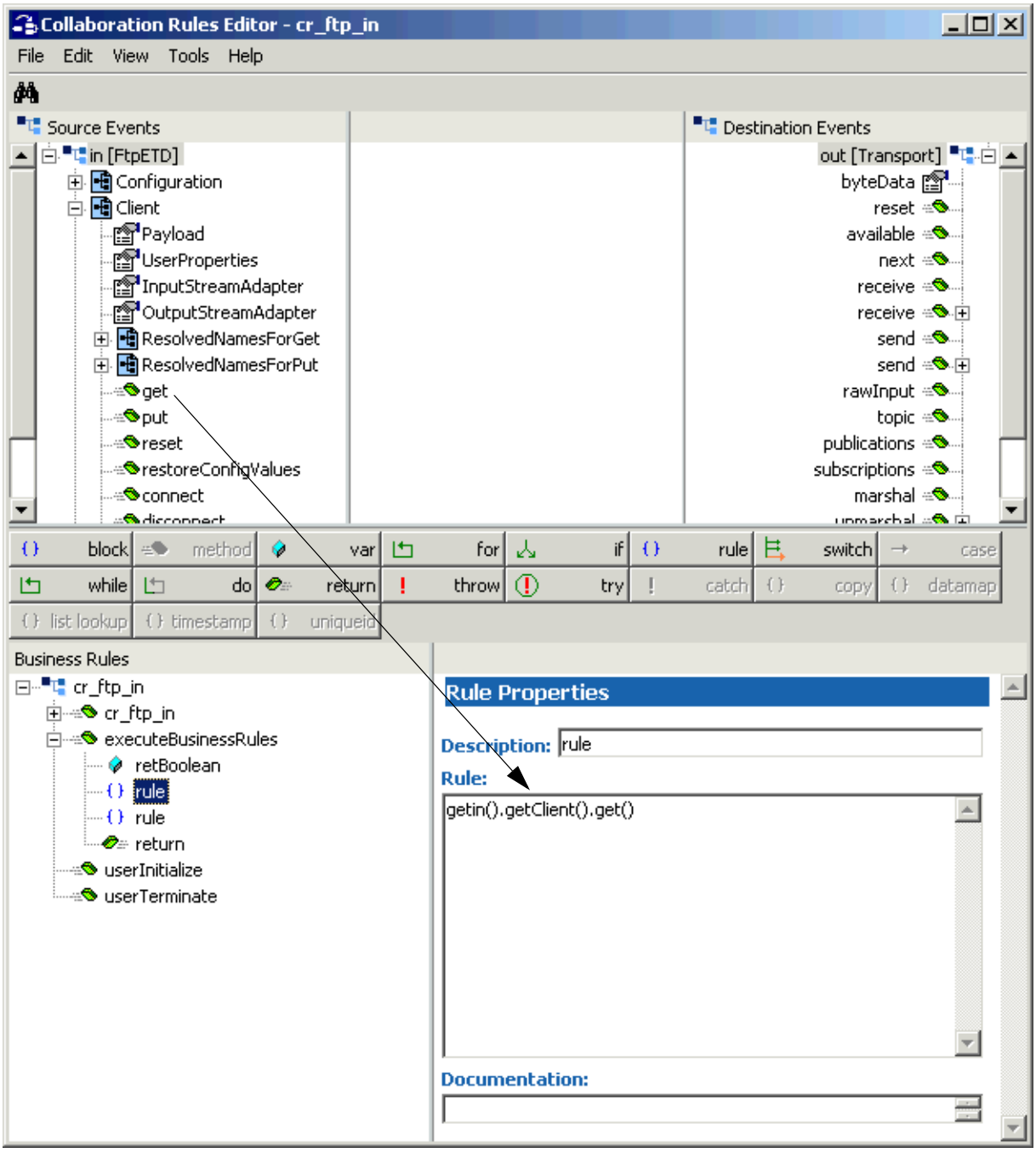
Figure 125 Collaboration Rules Editor: cr_ftp_in Expanded



13 Click **retBoolean** to begin creating a new rule (see **Figure 126 on page 259**).

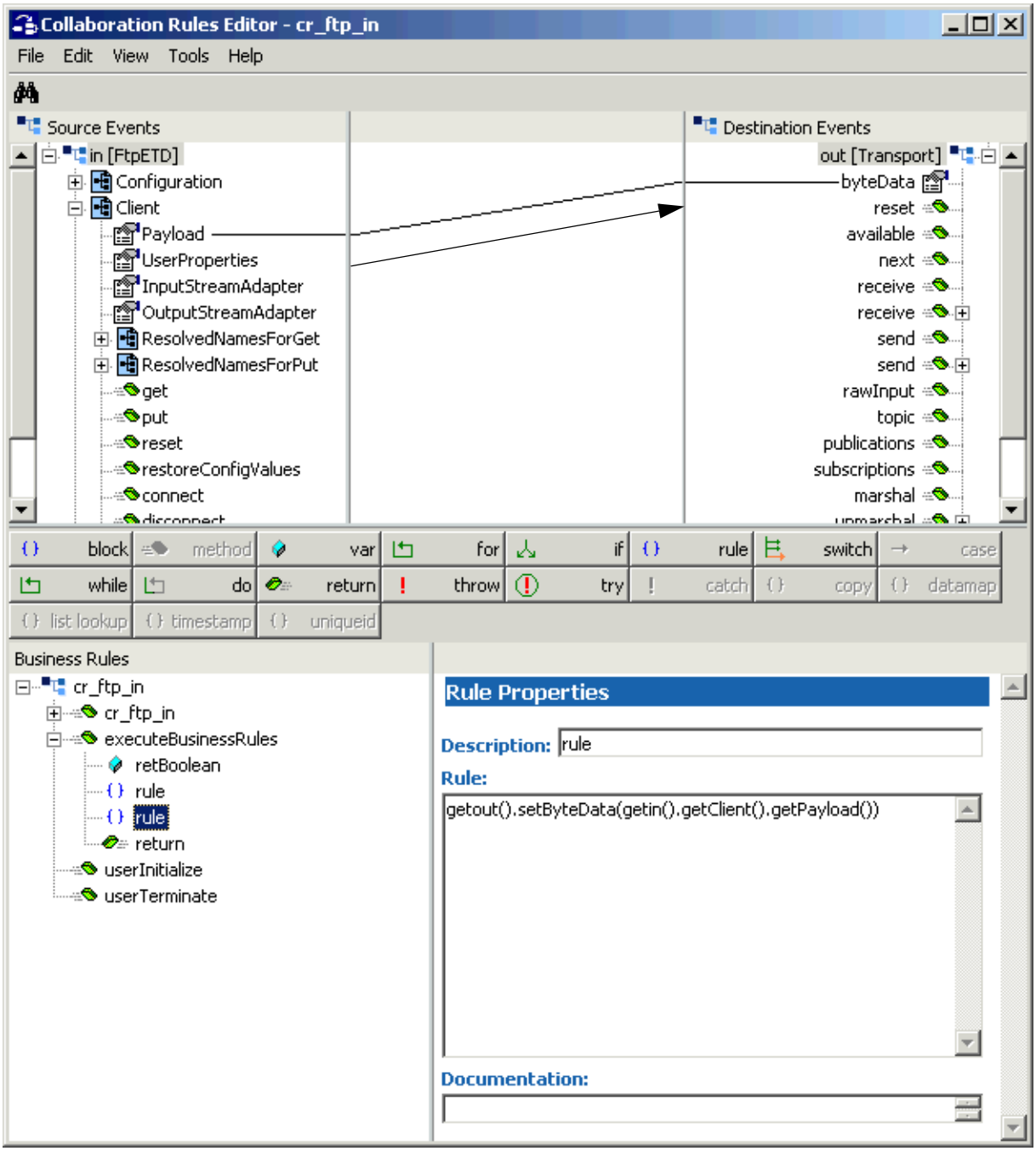
14 Drag the **get** method from the **Source Event** to the **Rule** scroll box in the **Rule Properties** window (see Figure 126). This action creates a new rule.

Figure 126 Collaboration Rules Editor: cr_ftp_in First Rule



15 With the previous rule selected, drag the **Payload** node from the **Source Event** onto the **byteData** node of the **Destination Event** (see Figure 127). This action creates another rule.

Figure 127 Collaboration Rules Editor: cr_ftp_in Second Rule



You have now finished creating the Business Rules.

16 You must create a Collaboration Rules class or use one from the sample.

Note: *See the* ***e*Gate Integrator User's Guide*** *for details on this procedure.*

17 To save the Collaboration Rules file, click **Save** on the **File** menu. The **Save** dialog box appears.

18 Provide a name for the **.xpr** file (for this example, use **cr_ftp_in.xpr**) then click **Save**.

19 Before compiling the code, on the **Tools** menu, click **Options** and verify that all necessary **.jar** files are included (see **"Collaboration Rules Editor: Java Classpaths Dialog Box" on page 255**).

20 When you have finished defining all the desired business logic, compile the Java code by selecting **Compile** from the **File** menu.

If the code compiles successfully, the message **Compile Completed** appears. If the outcome is unsuccessful, a Java Compiler error message appears. If there are any Java errors, be sure to correct them.

21 Once the compilation is complete, you can exit the Collaboration Rules Editor.

To create the cr_ftp_out Collaboration Rules file

1 Select the **Components** tab in the e*Gate Schema Designer.

2 In the Navigation pane, select the **Collaboration Rules** folder.

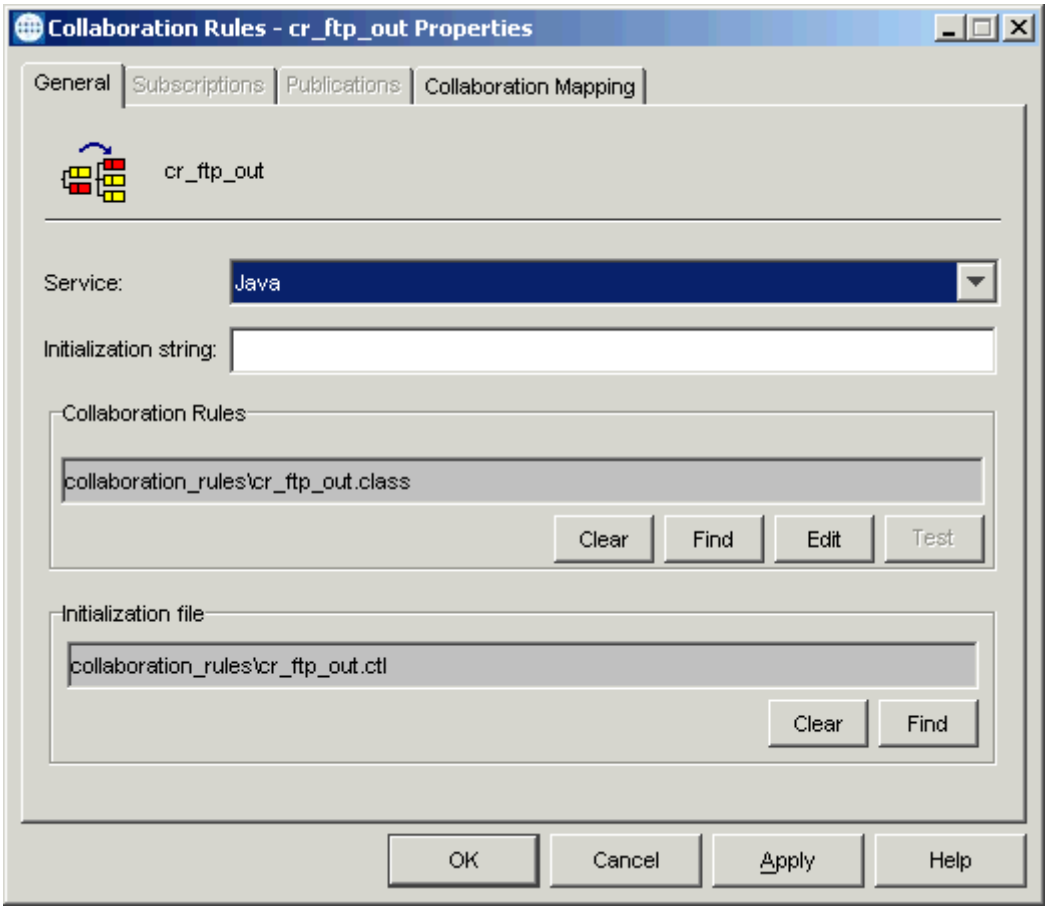
3 On the palette, click the **Collaboration Rules** icon.

4 Enter the name of the new Collaboration Rule, then click **OK.** Use **cr_ftp_out** for this example, for the **ew_ftp_out** e*Way's Collaboration, **col_ftp_out**.

5 Select the new **Collaboration Rule**, then right-click to edit its properties.

6 The **Collaboration Rules Properties** dialog box appears (see Figure 128).

Figure 128 Collaboration Rules Properties Dialog Box for cr_ftp_out: General Tab



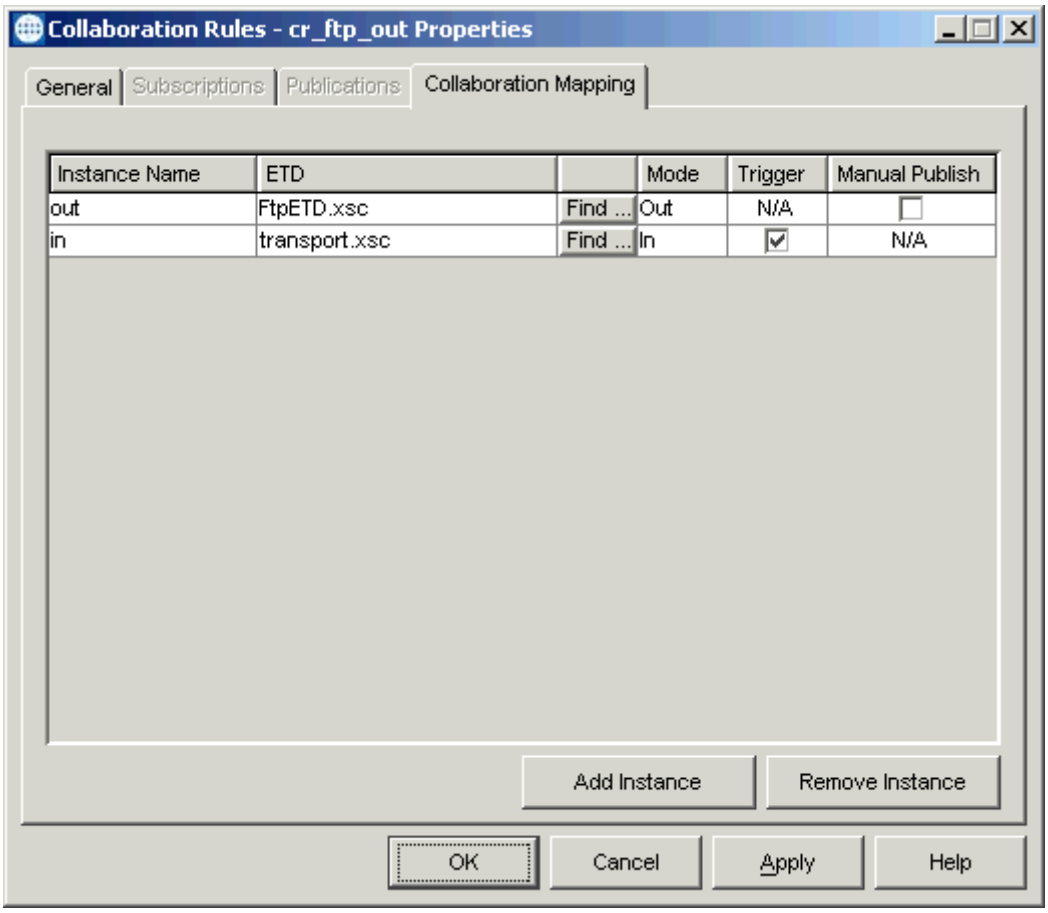
7 On the **General** tab in the dialog box select the **Java Collaboration Service**.

8 In the **Initialization String** text box, enter any required initialization string that the Collaboration Service may require. This field can be left blank.

9 Click the **Collaboration Mapping** tab.

10 Using the **Add Instance** button, create instances to coincide with the ETDs (see Figure 129).

Figure 129 Collaboration Rules Properties Dialog Box for cr_ftp_out: Mapping Tab



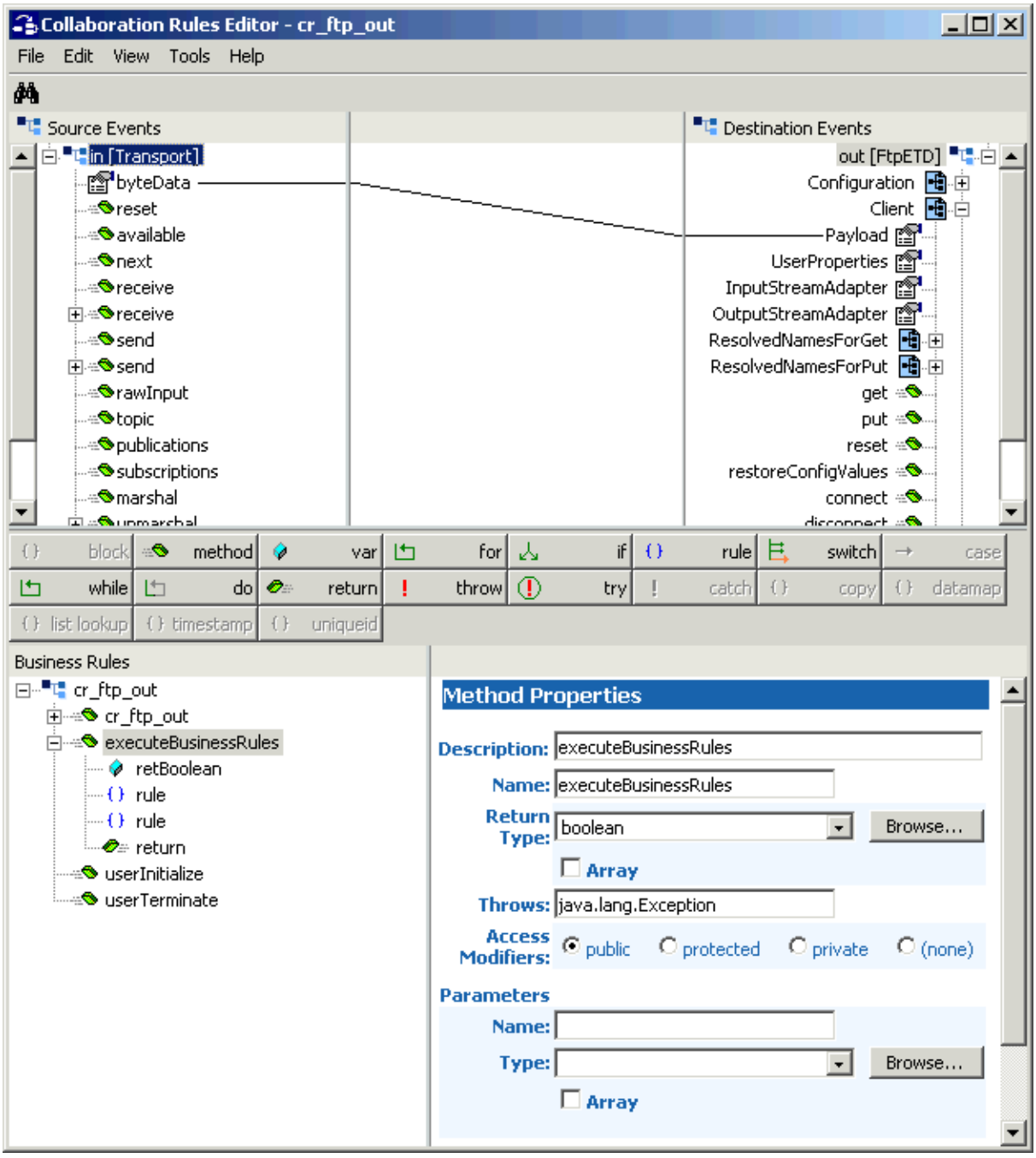
Configure the rest of **cr_ftp_out** as shown in the previous figure.

11 Select the **General** tab again, then click **New** (where the **Edit** button is in **Figure 128 on page 262**).

The Collaboration Rules Editor Main window opens.

12 After the window has opened, expand the source and destination Events, as well as the Business Rules. Figure 130 shows the results.

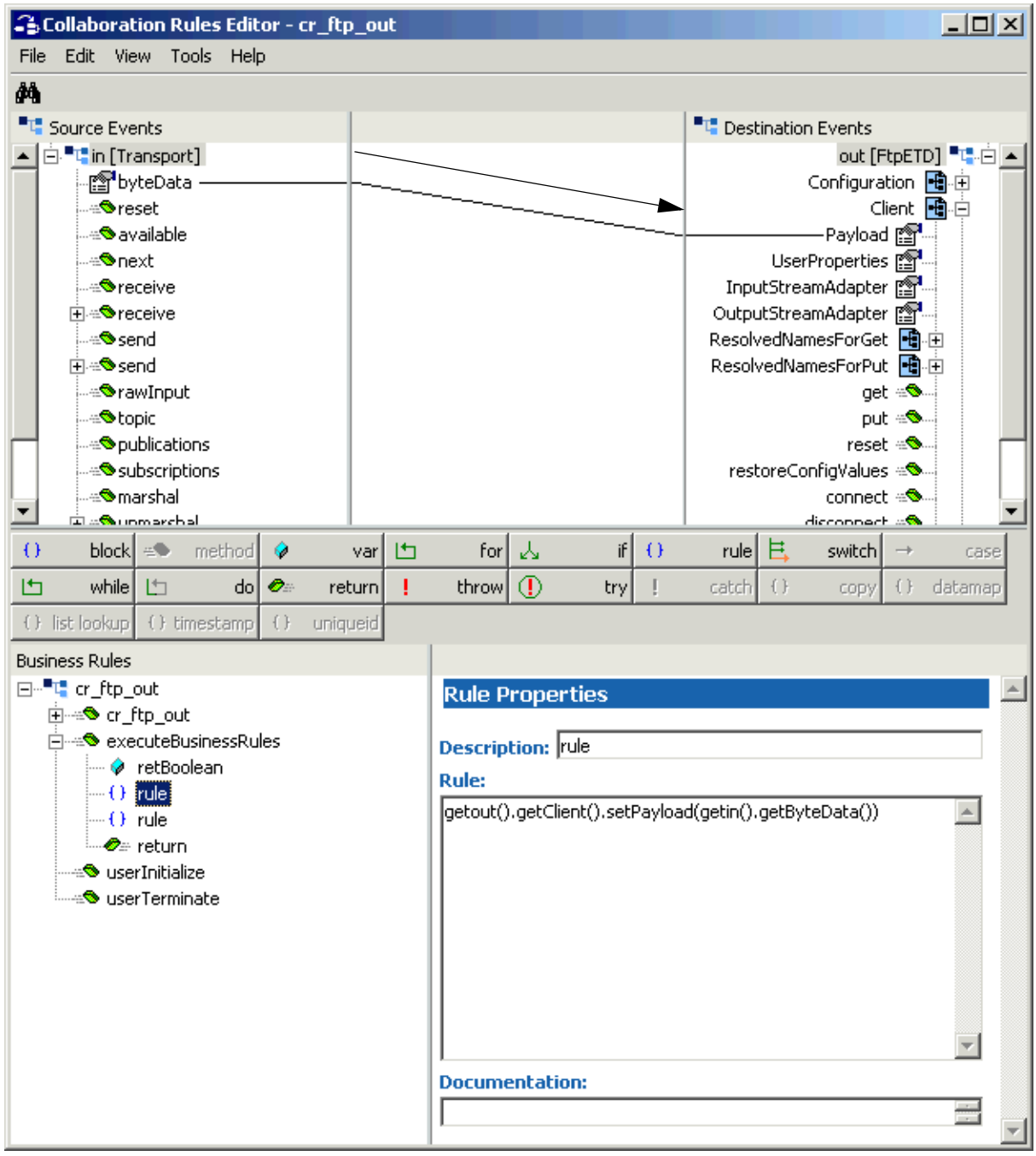
Figure 130 Collaboration Rules Editor: cr_ftp_out Expanded



13 Click **retBoolean** to begin creating a new rule (see **Figure 131 on page 265**).

14 Drag the **byteData** node from the **Source Event** onto the **Payload** node of the **Destination Event** (see Figure 131). This action creates a new rule.

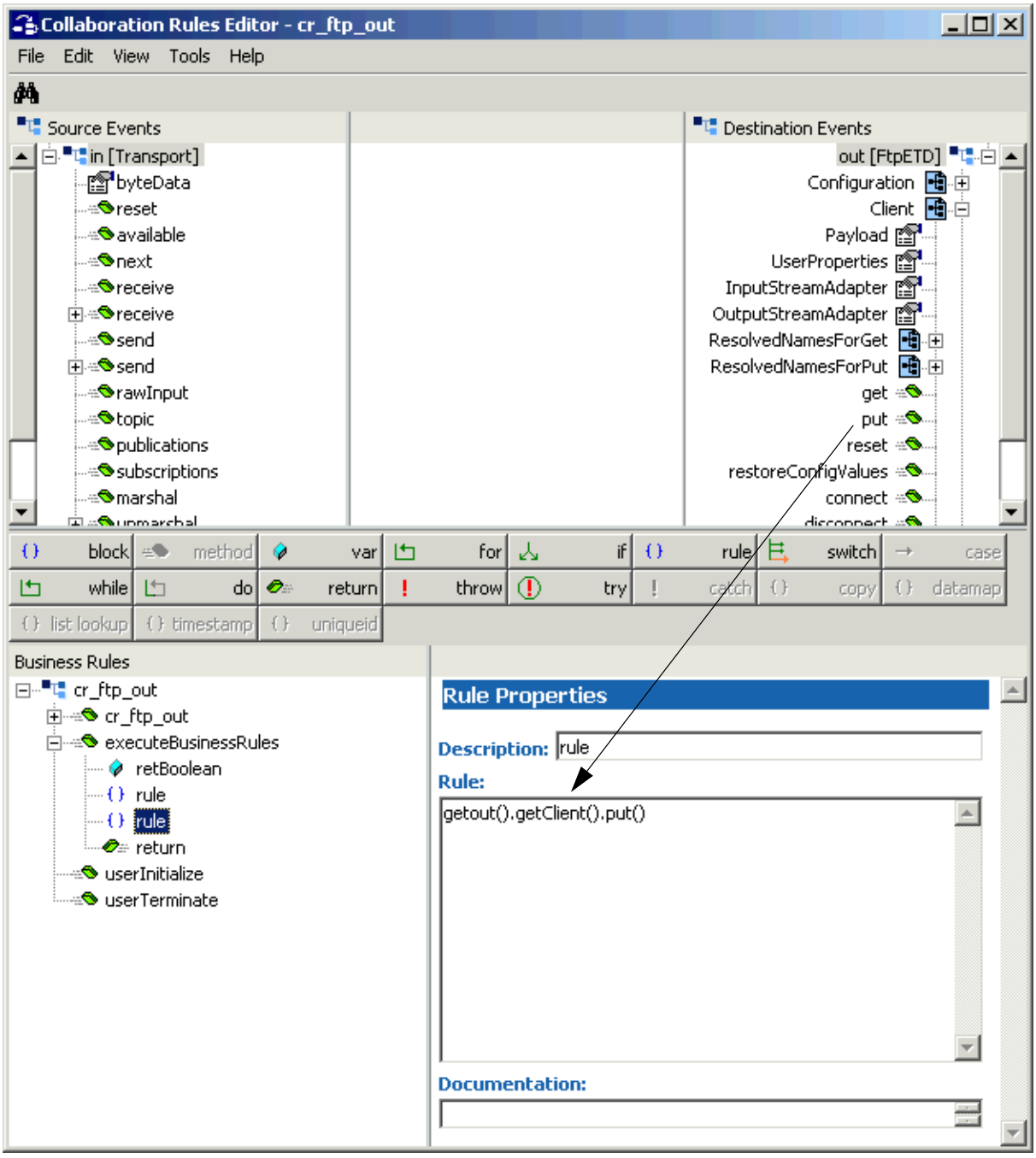
Figure 131 Collaboration Rules Editor: cr_ftp_out First Rule



15 Click the previous rule to begin creating a new rule (see **Figure 132 on page 266**).

16 Drag the **put** method from the **Destination Event** into the **Rule** scroll box in the **Rule Properties** window (see Figure 132). This action creates a new rule.

Figure 132 Collaboration Rules Editor: cr_ftp_out Second Rule



You have now finished creating the Business Rules.

17 You must create a Collaboration Rules class or use one from the sample.

18 To save the Collaboration Rules file, click **Save** on the **File** menu. The **Save** dialog box appears.

19 Provide a name for the **.xpr** file (for this example, use **cr_ftp_out.xpr**) then click **Save**.

20 Compile and save this Collaboration Rule in the same way as you did the previous one.

21 Once the compilation is complete, you can exit the Collaboration Rules Editor.

Creating Collaborations

This section explains how to create the Collaborations for this schema.

To create the Collaborations

1 In the e*Gate Schema Designer, select the Navigator's **Components** tab.

2 Open the host on which you want to create the Collaboration.

3 Select the Control Broker for this schema.

4 Select the **ew_ftp_in** e*Way to assign the Collaboration.

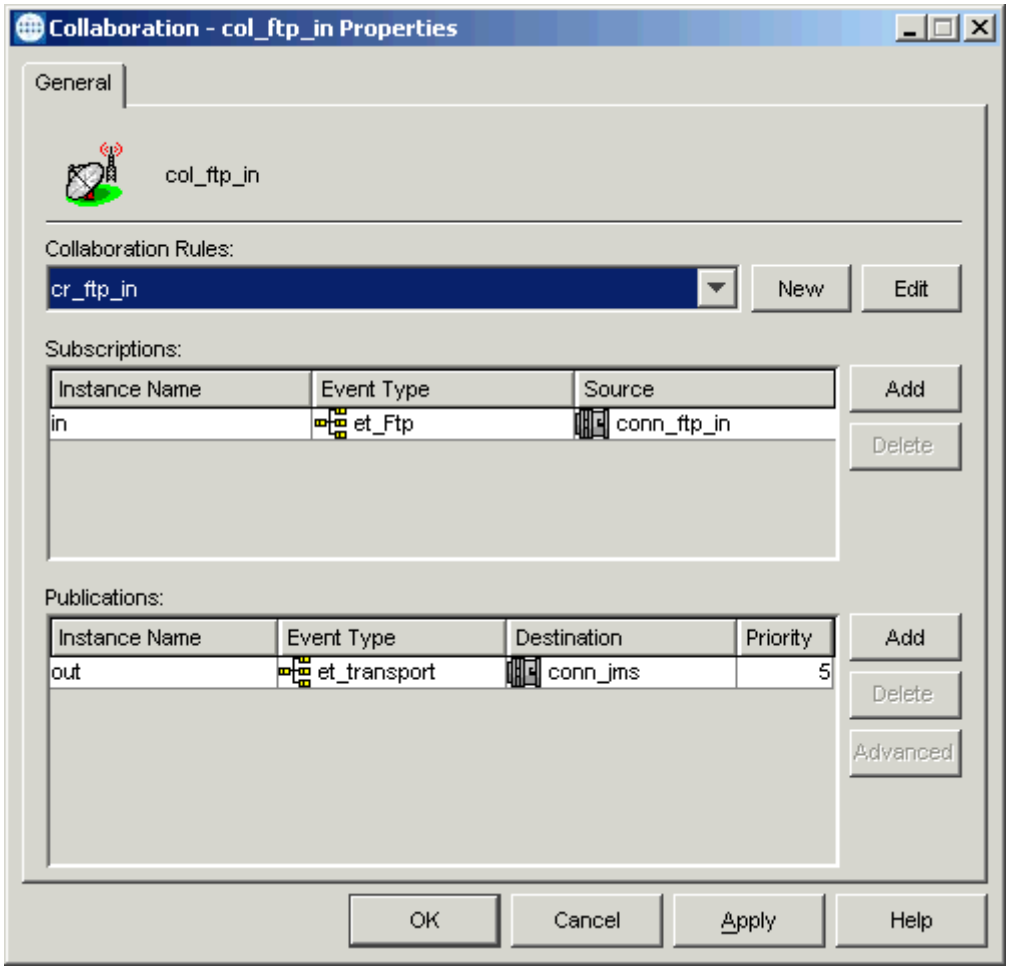
5 On the palette, click the **Collaboration** icon.

6 Enter the name (**col_ftp_in**) of the new Collaboration, then click **OK**.

7 Select the new Collaboration, then right-click to edit its properties.

8 The **Collaboration Properties** dialog box appears (see Figure 133).

Figure 133 col_ftp_in Properties Dialog Box



Configure the appropriate Collaboration properties as shown in the previous figure.

9 From the **Collaboration Rules** list, select the first Collaboration Rule that you created previously (**cr_ftp_in**) for this Collaboration.

10 Click **OK** to close the dialog box and save your changes.

11 Select the **ew_ftp_out** e*Way to assign the next Collaboration.

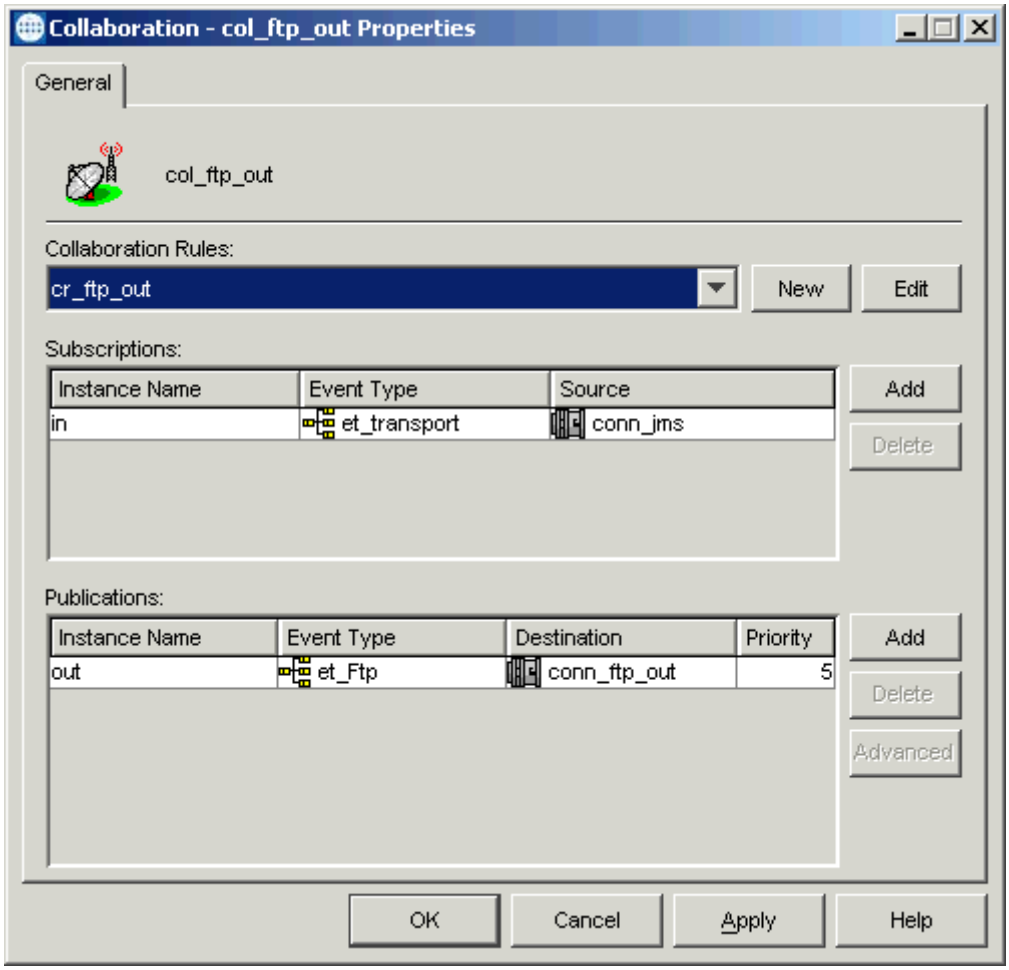
12 On the palette, click the **Collaboration** icon.

13 Enter the name (**col_ftp_out**) of the new Collaboration, then click **OK**.

14 Select the new Collaboration, then right-click to edit its properties.

15 The **Collaboration Properties** dialog box appears (see Figure 134).

Figure 134 col_ftp_out Properties Dialog Box



Configure the appropriate Collaboration properties as shown in the previous figure.

16 From the **Collaboration Rules** list, select the Collaboration Rule that you created previously (**cr_ftp_out**) for this Collaboration.

17 Click **OK** to close the dialog box and save your changes.

Running the Schema

For an explanation of how to run the schema see **"Running the Schema" on page 169**.

7.5 Sample Schema: Using Secure FTP

This section explains how to add secure FTP to the Batch e*Way in a typical e*Gate schema environment.

Note: *For more information on secure FTP and how the e*Way implements it, see* **"Secure FTP and the e*Way" on page 342**.

7.5.1 FtpSecuritySample Schema Overview

This section provides an overview of the sample schema and how it operates. The name of this schema is FtpSecuritySample, and it is contained in the import file **FtpSecuritySample.zip**.

Schema Setup

This schema is set up in the same way as the FtpExtensibilitySample. So you can use **Figure 106 on page 236** to see a diagram of the schema's general architecture. The arrows show the direction of data flow. The remote FTP locations can be the same or two different FTP servers.

Schema Operation

This sample schema has the following input/output setup:

- **Input**: An inbound e*Way retrieves a remote file from a remote FTP location and sends it as an Event to a JMS IQ Manager.
- **Output**: An outbound e*Way gets the Event from the IQ Manager and stores it as another remote FTP file.

This sample schema demonstrates the Batch e*Way's secure FTP feature Secure Shell (SSH) tunneling. It does *not* include the SOCKS feature.

Before you run this schema, make sure you configure the SSH environment properly. For details on how to configure the SSH tunneling configuration parameters, see **"SSH Tunneling Configuration" on page 50**.

You also need to be sure an FTP input file is provided and is accessible to e*Gate. Because no post-transfer command is specified, the input file is transferred continually until you shut down the e*Way.

In this sample schema, the inbound e*Way establishes an SSH channel. The outbound e*Way uses that channel, relying on the inbound. Therefore, the outbound e*Way works properly only when the inbound e*Way has established the SSH channel successfully. If desired, you can configure the outbound e*Way to establish its own SSH channel and avoid this dependency.

Note: *The purpose of this sample schema is only to demonstrate how to implement SSH tunneling, so there is no error-handling logic in the Collaboration Rules. See* **"Sample Schema: Basic FTP With Streaming" on page 129** *for a sample with error-handling logic.*

Schema Components

The FtpSecuritySample schema with the SSH tunneling security implementation consists of the following main e*Gate components:

- **ew_ftp_in**: Inbound Multi-Mode e*Way that brings the file from a remote FTP server into e*Gate.
- **ew_ftp_out**: Outbound Multi-Mode e*Way that sends the file from e*Gate to a remote FTP server.
- **col_ftp_in**: Collaboration for the **ew_ftp_in** e*Way.
  - **cr_ftp_in**: Collaboration Rule for **col_ftp_in**.
- **col_ftp_out**: Collaboration for the **ew_ftp_out** e*Way.
  - **cr_ftp_out**: Collaboration Rule for **col_ftp_out**.
- **localhost_iqmgr**: Oracle SeeBeyond JMS IQ Manager.
- **conn_ftp_in**: e*Way Connection for the **ew_ftp_in** e*Way.
- **conn_jms**: e*Way Connection for the JMS IQ Manager **localhost_iqmgr**.
- **conn_ftp_out**: e*Way Connection for the **ew_ftp_out** e*Way.

7.5.2 Creating the FtpSecuritySample Schema

Because this schema is set up in the same way as the previous FtpExtensibilitySample schema, you can follow those steps for basic schema creating and configuration. See **"Sample Schema: FTP and ETD Extensibility" on page 235** for details.

The exception to this similarity is in setting up the e*Way Connection configuration parameters. Instead of setting the **Extensions** parameters, you must set the **SSH Tunneling** parameters. There are also a few additional parameters that are different. The rest of this section explains how to set all of these parameters.

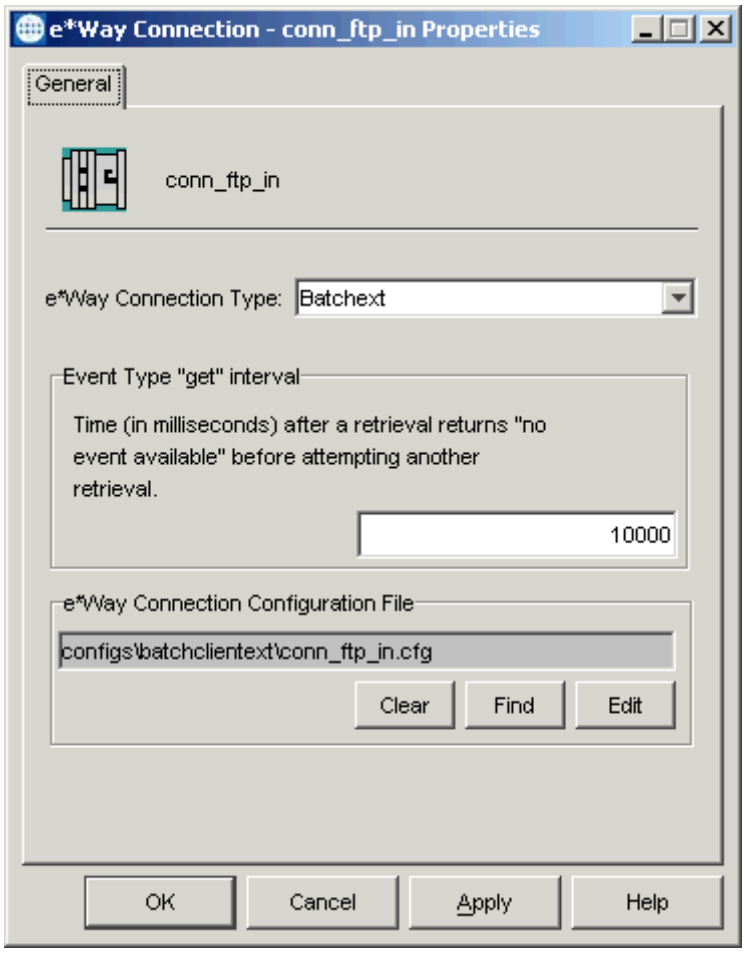
Creating and Configuring e*Way Connections

To create and configure the conn_ftp_in e*Way Connection

1. Highlight the **e*Way Connection** folder on the **Components** tab of the e*Gate Schema Designer.
2. On the palette, click the icon to create a new e*Way Connection.
3. Enter the name of the e*Way Connection (**conn_ftp_in**), then click **OK**.
4. Select the new **e*Way Connection**, then right-click to edit its properties.

5 When the **e*Way Connection Properties** dialog box opens, select **Batchext** from the **e*Way Connection Type** drop-down menu (see Figure 135).

Figure 135 conn_ftp_in e*Way Connection Properties Dialog Box



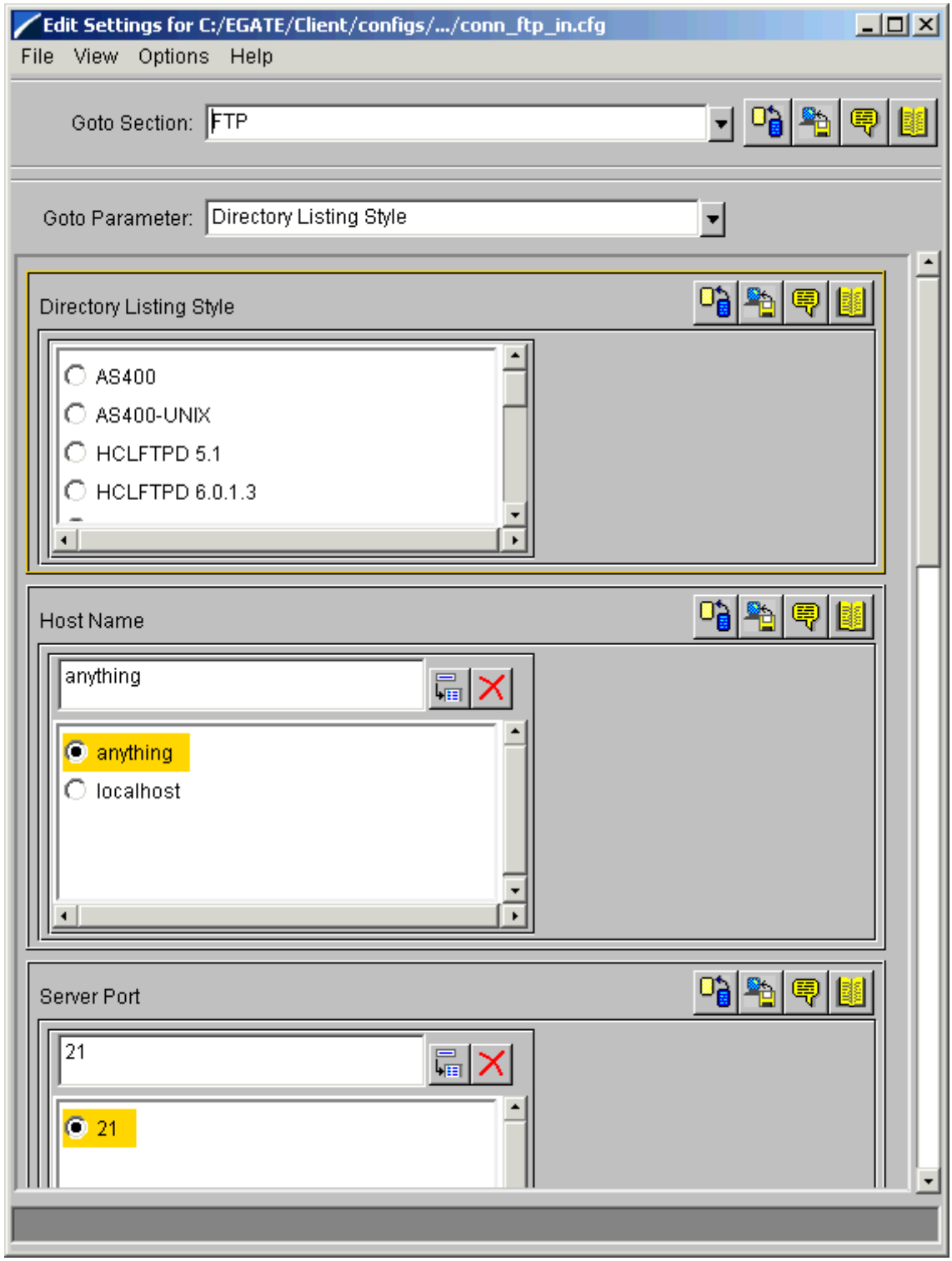
Configure the e*Way Connection properties as shown in the previous figure.

6 Under **e*Way Connection Configuration File**, click **New** (where the **Edit** button is in the previous figure). Select the **FtpETD**.

The e*Way Configuration Editor Main window opens. Use this interface to select the desired parameters, including those that correspond to the remote FTP system you are using. See **"FtpETD: Configuration Parameters" on page 36** for details.

7 Select the **FTP** settings (see **Figure 136 on page 273**).

Figure 136 e*Way Configuration Editor: conn_ftp_in FTP Settings



8 Under the **FTP** settings, accept the default settings for all parameters except for:

- **Directory Listing Style**
- **Host Name**
- **Server Port**
- **User Name**
- **Password**

You must enter your system settings for these parameters.

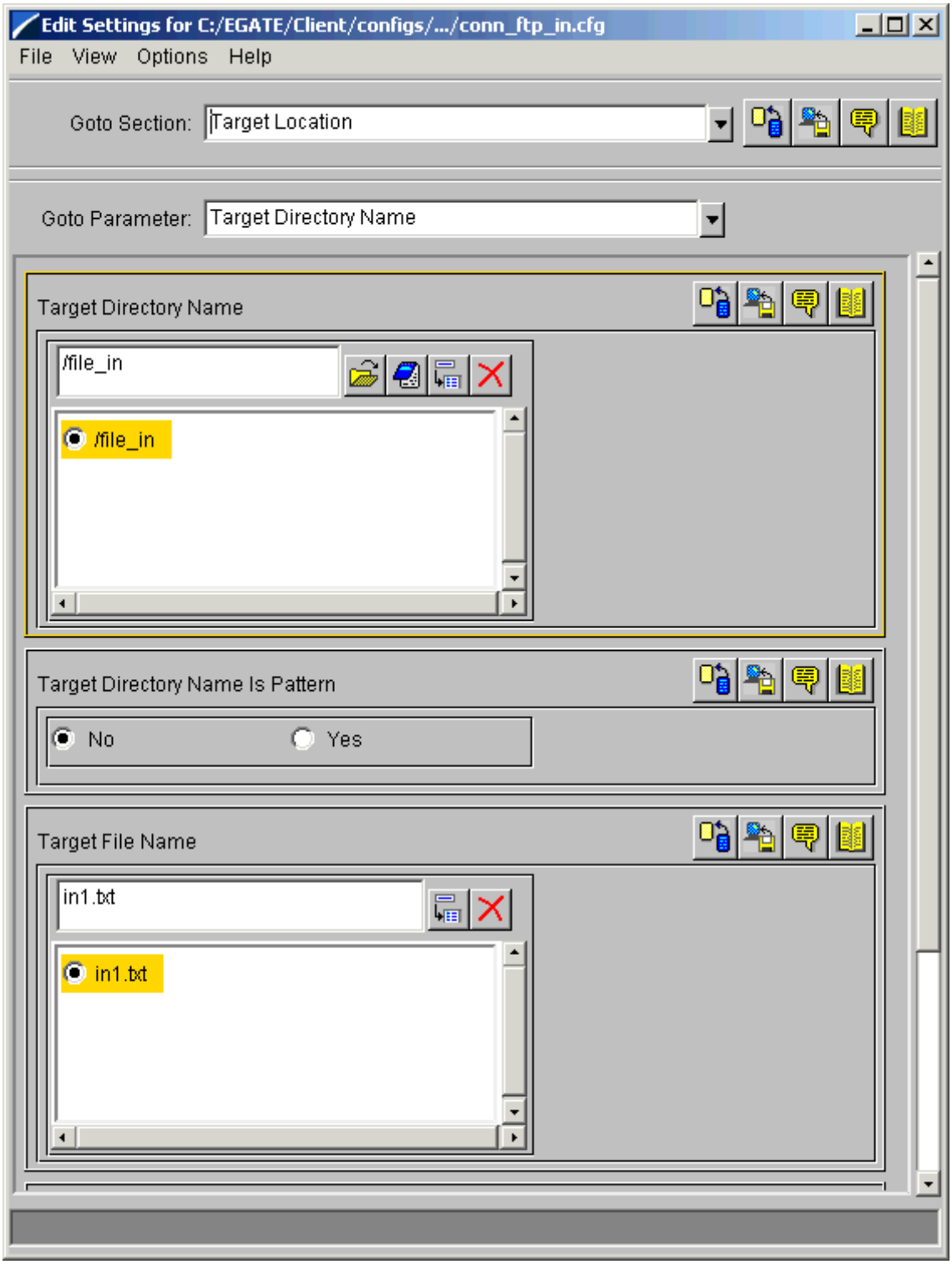
9 Select the **Target Location** settings (see **Figure 137 on page 275**).

10 Under the **Target Location** settings, accept the default settings for all parameters except for:

- **Target Directory Name**
- **Target File Name**

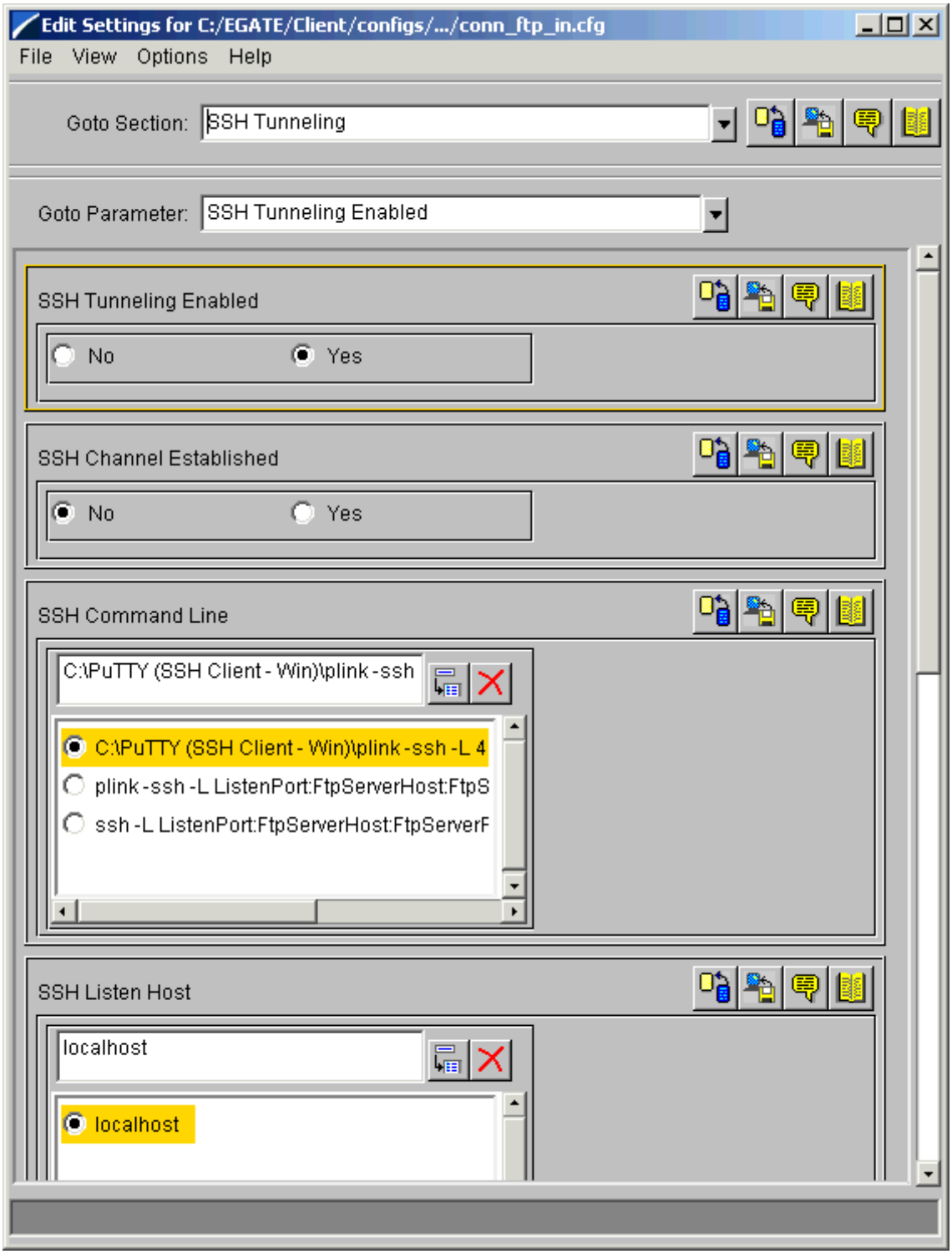
You must enter your system settings for this parameter (see **Figure 137 on page 275**).

Figure 137 e*Way Configuration Editor: conn_ftp_in Target Location Settings



11 In addition to the parameters listed previously, you must also enter parameters for security. Select the **SSH Tunneling** settings (see Figure 138).

Figure 138 e*Way Configuration Editor: conn_ftp_in SSH Tunneling Settings



12 Set the **SSH Tunneling Enabled** to **Yes** and **SSH Channel Established** to **No** as shown in **Figure 138 on page 276**.

13 Set the rest of these parameters as required by your own system.

14 When you are finished, save the .**cfg** file as the name of the e*Way Connection, close the e*Way Configuration Editor, and promote the file to run time.

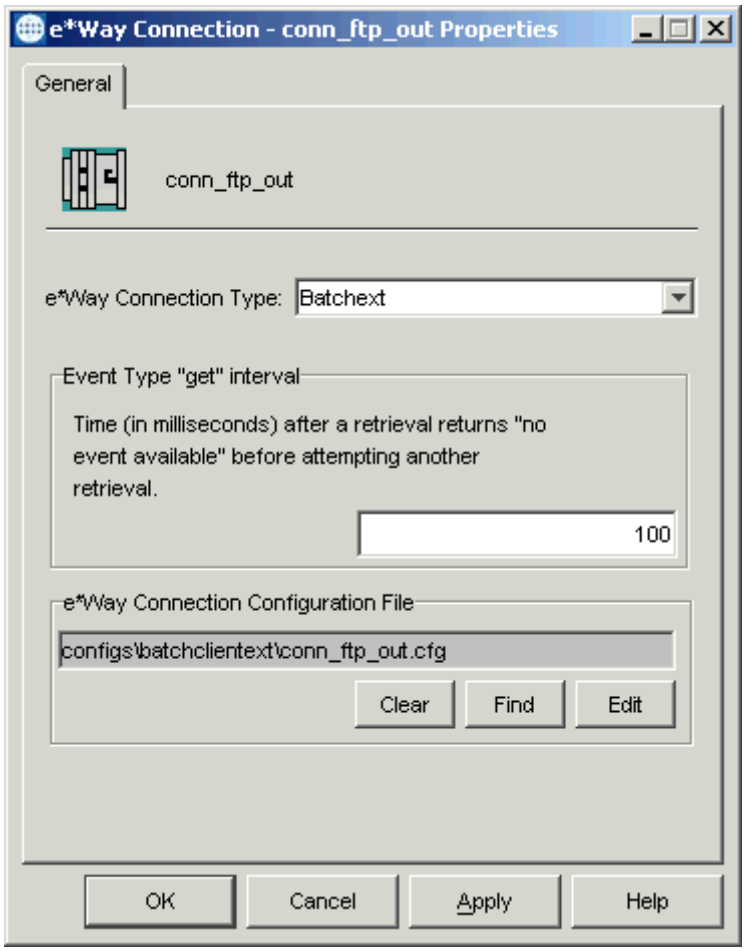
15 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the conn_ftp_out e*Way Connection

1 Access the **e*Way Connection Properties** dialog box for this component in the same way as you did for the previous e*Way Connection.

2 Set the properties as shown in (see Figure 139).

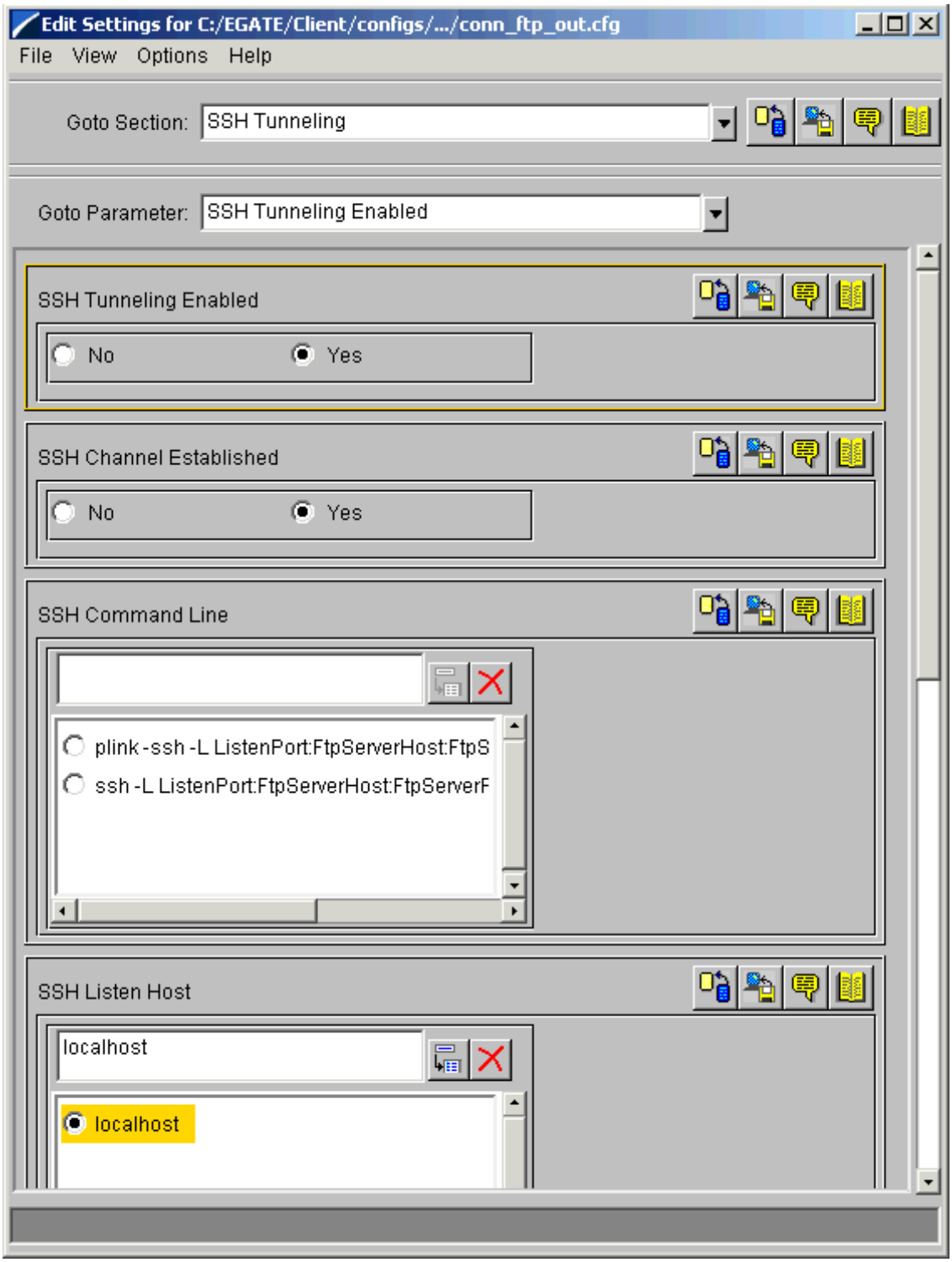
Figure 139 conn_ftp_out e*Way Connection Properties Dialog Box



3 Set the **FTP** and **Target Location** parameters for this e*Way Connection, using the appropriate information from the FTP system you are interfacing with.

4 In addition to the parameters listed previously, you must also enter parameters for security. Select the **SSH Tunneling** settings (see Figure 140).

Figure 140 e*Way Configuration Editor: conn_ftp_out SSH Tunneling Settings



5 Set the **SSH Tunneling Enabled** to **Yes** and **SSH Channel Established** to **Yes** as shown in **Figure 140 on page 278**.

6 Set the rest of these parameters as required by your own system.

7 When you are finished, save the .**cfg** file as the name of the e*Way Connection, close the e*Way Configuration Editor, and promote the file to run time.

8 Click **OK** to close the **e*Way Connection Properties** dialog box.

To create and configure the conn_jms e*Way Connection

See **"To create and configure the conn_jms e*Way Connection" on page 254** for details.

Running the Schema

For an explanation of how to run the schema see **"Running the Schema" on page 169**.

Important Notes:

In order to run this sample schema, you need to perform a few extra steps to ensure SSH tunneling works correctly.

- You can use Plink to verify that port forwarding is working outside of e*Gate (see **"Batch e*Way and SSH Tunneling" on page 49** and **"SSH Tunneling" on page 344** for more information). Run the following command from the `plink` directory:

  ```
  plink -ssh -L <ListenPort>:<FtpServerHost>:<FtpServerPort>
  <FtpServerHost>
  ```

  For example:

  ```
  plink -ssh -L 4567:MyFTPServer:21 MyFTPServer
  ```

- Use the same Windows user account to run the `ssh` command as to run the sample schema because this account is used to cache the FTP server host key.
  - The default e*Gate user account is **Administrator**, so the sample schema runs with no account changes if the Windows account is also Administrator.
  - If the Windows account is not named Administrator, create a new user account in the e*Gate Schema Designer (under **Security** and then **Users**) with the same name as the Windows user account. Select the **Administrator** role for the user account. In the properties window for each schema component, select the new user account in the **Run as User** property.
- After you complete the above steps, start the Control Broker using the following command:

  ```
  stccb -rh <RegistryHost> -rs <SchemaName> -un <UserName> -up
  <Password> -ln <ControlBrokerName>
  ```

For example:

```
stccdb -rh localhost -rs FtpSecuritySample -un Administrator
-up STC -ln localhost_cb
```

Note: *The username and password can be for the e*Gate Administrator user or the current Windows user configured to run the schema.*

This starts the control broker and the schema components, which should process the input data file. When there are no more runtime messages, verify that the output file was written to the remote FTP site in the directory you specified.

Chapter 8

Dynamic Configuration

This chapter explains how to use the Batch e*Way Intelligent Adapter's Dynamic Configuration features, including its message-based operations.

8.1 Dynamic Configuration: Overview

The Batch e*Way's Dynamic Configuration feature allows you to dynamically change an e*Gate Integrator schema's Event Type Definition (ETD) configuration parameters. Using messages based on the Extensible Markup Language (XML), you can change these parameters "on the fly," as desired.

You can base these changes on predefined conditions, depending on the business logic (Business Rules) you create in your e*Gate schemas. Although this business logic is available normally in e*Gate via Collaborations and Collaboration Rules, Dynamic Configuration utilizes the convenience of XML. This feature uses XML messages to give you the flexibility to meet changing conditions.

Using XML messaging, you can allow Dynamic Configuration routines to repeat indefinitely or give them the ability to respond to multiple changes immediately. Again, using this feature's provided XML messages, you can create your own business logic to customize the configuration and combine this logic with any other operation you want your e*Gate schema to do.

Schema Template

Using this feature requires that you import a schema template into e*Gate. See **"Dynamic Configuration Template" on page 293** for details on this template.

8.1.1 General Operation

In its simplest form, the Dynamic Configuration feature is an input-output mechanism. You input XML message-based orders, and the Dynamic Configuration Collaboration in the schema template outputs XML message-based data, error information, or both. The input orders can contain data, or they can be only orders.

The Dynamic Configuration feature allows the following types of messages:

- Order (send and receive)
- Error
- Data

Dynamic Configuration Messages and Files

Use the Document Type Definition (DTD) and e*Gate ETD files for the Dynamic Configuration XML messages, as shown in Table 10. You can find the **.xsc** files in the **eGate\client\monk_scripts\common** directory, along with their corresponding **.jar** files.

Table 10 Dynamic Configuration Messages and Files

Message Type	DTD File	Corresponding ETD File
Order	batch_eway_order.dtd	batch_eway_order.xsc
Error	batch_eway_error.dtd	batch_eway_error.xsc
Data	batch_eway_data.dtd	batch_eway_data.xsc

The rest of this section explains these message types.

Order Messages

Order messages are transmitted to the schema template in either of the following ways:

- Ordering it to send once (to one or more destinations)
- Ordering it to receive once (from one or more destinations)

In either of these cases, the order message, in XML, has the following basic format:

```
<batch_eWay_order>
    <command>             (command)    </command>
    <order_record>
        <file_transfer_method>(method)
        </file_transfer_method>
    <payload>             (DATA or reference to DATA)</payload>
</batch_eWay_order>
```

This message contains the following primary elements:

- **command**: Can only be send or receive.
- **order_record**: Contains the details for sending or retrieving to or from a single source or destination, for which the mandatory element is **file_transfer_method**.
- **payload**: Specifies the data to be sent (for a send command).

The payload data, if it is present (for the **send** command), can come in one of the following forms as specified by the **location** attribute:

- **base64InSitu**: Tells the schema template that the data is Base64-encoded and that it is located in the body of the **payload** element; this is the default value.

  Binary or HTML/XML content (that is, special XML-reserved characters that require encoding), or unicode/multi-byte data must be encoded in the Base64 format.

- **localDir**: Tells the schema template that the payload data is contained in a file in a local directory on the Participating Host. The local directory is specified by a value (a directory name) stored in the **payload** element. In this case the **payload** element has a **location** attribute equal to **localDir** (for more information, see **"More on the localDir Attribute" on page 283**).
- Empty string (" ") or the absence of the **location** attribute.

  The **payload** node can contain unencoded raw data, in which case it has a **location** attribute set to either an empty string (" ") or simply not specified. It is important that the data in this case must *not* contain XML-reserved or binary characters. Base64 encoding must be used with the **base64Insitu location** attribute.

More on the localDir Attribute

Use the **localDir** attribute if you do not want to transport large files through the Intelligent Queues (IQs) for the sole purpose of sending files to an external location.

If you only specify a directory name, the file name is assumed to represent either one of the following:

- The working file, as specified in the remote file name element of the XML order
- The default file named in the e*Way Connection configuration (if the XML order does not overwrite the remote file name)

If the payload data contains a valid file name with full path information, the specified file is read for the **payload** element.

See **"Send or Receive Order Message" on page 285** for the corresponding DTD file and additional information.

Error Messages

If the schema template has problems with an order, it publishes the command message with a corresponding error record. It also publishes the corresponding order record or payload data, as defined in the Dynamic Configuration parameter settings.

Even if the template schema does not encounter a problem, it can still be configured to publish a success status record based on the XML error message.

Also, you can control whether the order record, payload data, or both, are included in the error message, using the appropriate Dynamic Configuration parameter settings in the corresponding e*Way Connection.

See the Dynamic Configuration parameters in **Chapter 4** for more information.

The error message, in XML, has the following basic format:

```
<batch_eWay_order>
     <command>          (command)          </command>
     <order_record>…
</order_record>

     <error_record>
     <error_code>                          </error_code>
     <error_text>                          </error_text>
     <last_action>                         </last_action>
     </error_record>
     <payload>          (DATA)             </payload>
</batch_eWay_order>
```

This message has the following error-related elements:

- **error_record**
- **error_code**
- **error_text**

See **"Error Message" on page 288** for the corresponding DTD file and additional information.

Data Messages

Data messages follow the same general format as outlined in the previous sections and carry Dynamic Configuration output data.

The data message, in XML, has the following basic format:

```
<batch_eWay_data>
     <command>          (command)          </command>
     <order_record>     (order_record)     </order_record>
     <payload>               (DATA)        </payload>
</batch_eWay_data>
```

If the **location** attribute of the **payload** element is **localDir** for the corresponding order message, the XML data message contains the actual content of the data so it can be forwarded to other Collaborations through e*Gate IQs.

See **"Data Message" on page 291** for the corresponding DTD file and additional information.

Configuration Parameters

For an explanation of the e*Way Connection configuration parameters associated with this feature, see the following sections:

FTP ETD

"Dynamic Configuration" on page 57

Local File ETD

"Dynamic Configuration" on page 67

Limitations of the Feature

The Dynamic Configuration processor has the following limitations:

- Does not support XA-mode operations for FTP or local file operations
- Does not support the Resume Reading feature for the local file ETD, because any data transfer is completed within each instance of the business file execution
- Can only process one incoming order per e*Gate Event; when it is sending data with a send order, it does not support the **return_tag** element
- Does not support regular expressions to the extent that all the files that satisfy the expression are operated upon all at once for multiple-file merging, archiving, and so on (see **"Receiving Data with a Receive Order" on page 287**)

Note: *You can create additional regular expression business logic for the schema template yourself, if desired. See* **"Dynamic Configuration Template" on page 293** *for information on the template.*

8.2 Message Descriptions

This section shows the text for each XML DTD message used in the Dynamic Configuration feature.

8.2.1 Send or Receive Order Message

The following DTD file provides details of the XML message that can be used for send orders or receive orders:

```
<!-- batch eway order record format. -->
<!ELEMENT batch_eWay_order (command, (order_record)+, payload?)>
<!ELEMENT command (#PCDATA)>
<!ATTLIST command
    Enumeration (send | receive) "send"
>
<!ELEMENT order_record (external_host_setup?, (subscribe_to_external
| publish_to_external)?, FTP?, SOCKS?)>
<!ELEMENT external_host_setup (host_type?, external_host_name?,
user_name?, encrypted_password?, file_transfer_method?, return_tag?)>
<!ELEMENT host_type (#PCDATA)>
<!ELEMENT external_host_name (#PCDATA)>
<!ELEMENT user_name (#PCDATA)>
<!ELEMENT encrypted_password (#PCDATA)>
<!ELEMENT file_transfer_method (#PCDATA)>
<!ATTLIST file_transfer_method
    Enumeration (ftp | copy) "ftp"
>
<!ELEMENT return_tag (#PCDATA)>
<!ELEMENT subscribe_to_external (remote_directory_name?,
remote_file_regexp?, remote_command_after_transfer?,
remote_rename_or_archive_name?, local_command_after_transfer?,
local_archive_directory?)>
<!ELEMENT remote_directory_name (#PCDATA)>
<!ELEMENT remote_file_regexp (#PCDATA)>
```

```
<!ELEMENT remote_command_after_transfer (#PCDATA)>
<!ATTLIST remote_command_after_transfer
    Enumeration (archive | delete | none | rename) "delete"
>
<!ELEMENT remote_rename_or_archive_name (#PCDATA)>
<!ELEMENT local_command_after_transfer (#PCDATA)>
<!ATTLIST local_command_after_transfer
    Enumeration (archive | delete) "delete"
>
<!ELEMENT local_archive_directory (#PCDATA)>
<!ELEMENT publish_to_external (remote_directory_name?,
remote_file_name?, append_or_overwrite_when_transferring_files?,
remote_command_after_transfer?, remote_rename_or_archive_name?,
local_command_after_transfer?, local_archive_directory?)>
<!ELEMENT remote_file_name (#PCDATA)>
<!ELEMENT append_or_overwrite_when_transferring_files (#PCDATA)>
<!ATTLIST append_or_overwrite_when_transferring_files
    Enumeration (append | overwrite) "append"
>
<!ELEMENT FTP (server_port, mode, Pretransfer_Commands,
Posttransfer_Commands)>
<!ELEMENT server_port (#PCDATA)>
<!ELEMENT mode (#PCDATA)>
<!ELEMENT Pretransfer_Commands (#PCDATA)>
<!ELEMENT Posttransfer_Commands (#PCDATA)>
<!ELEMENT SOCKS (server_host_name, server_port, method, user_name,
encrypted_password)>
<!ELEMENT server_host_name (#PCDATA)>
<!ELEMENT method (#PCDATA)>
<!ELEMENT payload (#PCDATA)>
<!ATTLIST payload
    Location (base64InSitu | localDir) #IMPLIED
>
```

Additional Information: Order Messages

The following elements in the order message are mandatory:

- Command element (**send** or **receive**)
- **file_transfer** method element (**ftp** or **copy**; use lower case)
- For the **send** command, there must be a **payload** element

All other elements of the XML order message are optional. If other elements are present in an XML order message, they are used to overwrite the corresponding default parameters.

Sending Data with a Send Order

The following representation shows the structure of an XML send order message:

```
<batch_eWay_order>
    <command>            send             </command>
    <order_record>
    <external_host_setup>
        <host_type>      Unix             </host_type>
        <user_name>      Alincoln         </user_name>
        <encrypted_password> liasdfLIJB   </encrypted_password>
        <file_transfer_method> ftp        </file_transfer_method>
    </external_host_setup>
    <publish_to_external>
```

```
            <remote_directory_name>/usr/home/honest_abe/to
                </remote_directory_name>
            <remote_file_name> X1.tmp          </remote_file_name>
            <append_or_overwrite_when_transferring_files>overwrite
                </append_or_overwrite_when_transferring_files>
            <remote_rename_or_archive name>X1.dat
                </remote_rename_or_archive_name>
        </publish_to_external>
        </order_record>
        <payload>             (DATA)           </payload>
</batch_eWay_order>
```

Note: *The payload* ***(DATA)*** *attribute must be either Base64-encoded data or a file location.*

This example shows the delivery of a file to an external system. The file is sent by the generation of the XML message, **batch_eWay_order**. This message contains a command record, an order record, and finally a single payload message. The order record represents one piece of information on the destination for this payload data.

Receiving Data with a Receive Order

Receiving from a file is similar to sending. The following representation shows the structure of an XML receive order message:

```
<batch_eWay_order>
    <command>             receive          </command>
    <external_host_setup>
        <host_type>       Unix             </host_type>
        <user_name>       Alincoln         </user_name>
        <encrypted_password>liasdfLIJB     </encrypted_password>
        <file_transfer_method> ftp         </file_transfer_method>
        <return_tag>      Factor order     </return_tag>
    </external_host_setup>
    <subscribe_to_external>
        <remote_directory_name> /usr/home/honest_abe/from
            </remote_directory_name>
        <remote_file_regexp> Y*.dat    </remote_file_regexp>
    </subscribe_to_external>
</batch_eWay_order>
```

In this case, the e*Way template schema retrieves the first file in the designated directory that matches the given regular expression. The template then reads the entire content of the file and sends it to the e*Gate component you specified, as a publication.

The following order message elements or attributes require special explanation:

- **encrypted_password**: Used for SOCKS or simple FTP; you can use the method **com.stc.common.utils.ScEncrypt.encrypt()** to encrypt a password in the XML order for a Collaboration.
- **return_tag**: Not used by the template schema because the schema can process only one order at a time.
- **remote_file_regexp**: For the **receive** command, this element can be a regular expression, but only the first matching file is processed. For the **send** command, this element *cannot* be a regular expression; it can only write to one file.
- **remote_command_after_transfer:**
  - **archive**: Translated to **rename** in the e*Way's ETDs because this ETD does not support the **archive** operation. The **remote_rename_or_archive_name** element is instead intrepreted as the archive directory name, and the current working file (expansion of the **remote_file_regexp** element) is used to simulate the archive effect.
  - **rename**: For this element, the **remote_rename_or_archive_name** element is interpreted as the new file name, and the working directory (that is, the **remote_directory_name** element) is used.
- **local_command_after_transfer:**
  - **archive**: For the local command after transfer. If it is **archive**, the element **local_archive_directory** is interpreted as the directory name to which the working file (that is, the expansion of the **remote_file_regexp** element) is archived. If this directory does not exist, it is created by the schema template.
  - **delete**: No operation is carried out because no temporary file is created that requires clean up.
- **payload**: Contains the attribute, **location**, which can have one of the values listed and explained under **"Payload Data" on page 292**.

8.2.2 Error Message

The following DTD file is used for the error-reporting XML message:

```
<!-- batch eway error record format. -->
<!ELEMENT batch_eWay_error (command, (return_tag | order_record)?,
error_record, payload?)>
<!ELEMENT command (#PCDATA)>
<!ATTLIST command
    Enumeration (send | receive) "send"
>
<!ELEMENT order_record (external_host_setup?, (subscribe_to_external
| publish_to_external)?, FTP?, SOCKS?)>
<!ELEMENT external_host_setup (host_type?, external_host_name?,
user_name?, encrypted_password?, file_transfer_method?, return_tag?)>
<!ELEMENT host_type (#PCDATA)>
<!ELEMENT external_host_name (#PCDATA)>
<!ELEMENT user_name (#PCDATA)>
<!ELEMENT encrypted_password (#PCDATA)>
<!ELEMENT file_transfer_method (#PCDATA)>
<!ATTLIST file_transfer_method
```

```
    Enumeration (ftp | copy) "ftp"
>
<!ELEMENT return_tag (#PCDATA)>
<!ELEMENT subscribe_to_external (remote_directory_name?,
remote_file_regexp?, remote_command_after_transfer?,
remote_rename_or_archive_name?, local_command_after_transfer?,
local_archive_directory?)>
<!ELEMENT remote_directory_name (#PCDATA)>
<!ELEMENT remote_file_regexp (#PCDATA)>
<!ELEMENT remote_command_after_transfer (#PCDATA)>
<!ATTLIST remote_command_after_transfer
    Enumeration (archive | delete | none | rename) "delete"
>
<!ELEMENT remote_rename_or_archive_name (#PCDATA)>
<!ELEMENT local_command_after_transfer (#PCDATA)>
<!ATTLIST local_command_after_transfer
    Enumeration (archive | delete) "delete"
>
<!ELEMENT local_archive_directory (#PCDATA)>
<!ELEMENT publish_to_external (remote_directory_name?,
remote_file_name?, append_or_overwrite_when_transferring_files?,
remote_command_after_transfer?, remote_rename_or_archive_name?,
local_command_after_transfer?, local_archive_directory?)>
<!ELEMENT remote_file_name (#PCDATA)>
<!ELEMENT append_or_overwrite_when_transferring_files (#PCDATA)>
<!ATTLIST append_or_overwrite_when_transferring_files
    Enumeration (append | overwrite) "append"
>
<!ELEMENT FTP (server_port, mode, Pretransfer_Commands,
Posttransfer_Commands)>
<!ELEMENT server_port (#PCDATA)>
<!ELEMENT mode (#PCDATA)>
<!ELEMENT Pretransfer_Commands (#PCDATA)>
<!ELEMENT Posttransfer_Commands (#PCDATA)>
<!ELEMENT SOCKS (server_host_name, server_port, method, user_name,
encrypted_password)>
<!ELEMENT server_host_name (#PCDATA)>
<!ELEMENT method (#PCDATA)>
<!ELEMENT payload (#PCDATA)>
<!ATTLIST payload
    Location (base64InSitu | localDir) #IMPLIED
>
<!ELEMENT error_record (error_code, error_text, last_action)>
<!ELEMENT error_code (#PCDATA)>
<!ELEMENT error_text (#PCDATA)>
<!ELEMENT last_action (#PCDATA)>
>
```

Additional Information: Error Messages

The inclusion of the order record, payload data, or both, in an error message is configurable in the corresponding e*Way Connection. See the Dynamic Configuration parameters in **Chapter 4** for more information.

The following require special explanation:

- If there is no error, and the error message is sent as a result of a successful transfer when the **Publish Status Record on Success** parameter is set to **Yes**, the following elements are present:
  - **error_code**: Zero (0).
  - **error_text**: A timestamp showing when the order was successfully processed, for example:

    ```
    Successfully sent on: Fri, 11 Oct 2002 at 14:02:30 PDT
    ```

  - **last_action**: An empty string (reserved for future use).
- If there is an error in processing an order message when the **Publish Status Record on Error** parameter is set to **Yes**, and the **file_transfer** method is **ftp**, the following elements are present:
  - **error_code**: The reply/return code of the last FTP reply command.
  - **error_text**: The exception message as supplied by the corresponding ETD.
  - **last_action**: The entire text of the last FTP server response exactly as it was received.
- If there is an error in processing an order message when the **Publish Status Record on Error** parameter is set to **Yes**, and the **file_transfer** method is **copy**, the following elements are present:
  - **error_code**: -1 to indicate an error.
  - **error_text**: The exception message as supplied by the corresponding ETD.
  - **last_action**: The text **No information** (reserved for future use).
- If there is an error in processing an order message when the **Publish Status Record on Error** parameter is set to **Yes,** and there is another error, for example:
  - An unknown file transfer method (neither **ftp** nor **copy**)
  - A problem converting the output message's data payload to the Base64 format
  - An invalid malformed command action (see **"Malformed Command Actions" on page 322**)

  Then, the following elements are present:
  - **error_code**: -1 to indicate an error.
  - **error_text**: The exception message or explanation.
  - **last_action**: The text **No information** (reserved for future use).

8.2.3 Data Message

The following DTD file provides a data structure, includes a data payload, and is used for transporting data from the schema template:

```
<!-- batch eway data record format. -->
<!ELEMENT batch_eWay_data (command,
                           (return_tag|order_record)?,
                           payload) >
<!ELEMENT command (#PCDATA) >
<!ATTLIST command Enumeration (send|receive) "send" >
<!ELEMENT order_record (external_host_setup?,
                        (subscribe_to_external|publish_to_external)?,
                        FTP?,
                        SOCKS?) >
<!ELEMENT external_host_setup (host_type?,
                               external_host_name?,
                               user_name?,
                               encrypted_password?,
                               file_transfer_method?,
                               return_tag?) >
<!ELEMENT host_type (#PCDATA) >
<!ELEMENT external_host_name (#PCDATA) >
<!ELEMENT user_name (#PCDATA) >
<!ELEMENT encrypted_password (#PCDATA) >
<!ELEMENT file_transfer_method (#PCDATA) >
<!ATTLIST file_transfer_method Enumeration (ftp|copy) "ftp" >
<!ELEMENT return_tag (#PCDATA) >
<!ELEMENT subscribe_to_external (remote_directory_name?,
                                 remote_file_regexp?,
                                 remote_command_after_transfer?,
                                 remote_rename_or_archive_name?,
                                 local_command_after_transfer?,
                                 local_archive_directory?) >
<!ELEMENT remote_directory_name (#PCDATA) >
<!ELEMENT remote_file_regexp (#PCDATA) >
<!ELEMENT remote_command_after_transfer (#PCDATA) >
<!ATTLIST remote_command_after_transfer Enumeration
(archive|delete|none|rename) "delete" >
<!ELEMENT remote_rename_or_archive_name (#PCDATA) >
<!ELEMENT local_command_after_transfer (#PCDATA) >
<!ATTLIST local_command_after_transfer Enumeration (archive|delete)
"delete" >
<!ELEMENT local_archive_directory (#PCDATA) >
<!ELEMENT publish_to_external (remote_directory_name?,
                               remote_file_name?,

append_or_overwrite_when_transferring_files?,
                               remote_command_after_transfer?,
                               remote_rename_or_archive_name?,
                               local_command_after_transfer?,
                               local_archive_directory?) >
<!ELEMENT remote_file_name (#PCDATA) >
<!ELEMENT append_or_overwrite_when_transferring_files (#PCDATA) >
<!ATTLIST append_or_overwrite_when_transferring_files Enumeration
(append|overwrite) "append" >
<!ELEMENT FTP (server_port,
               mode,
               Pretransfer_Commands,
               Posttransfer_Commands) >
```

```
<!ELEMENT server_port (#PCDATA) >
<!ELEMENT mode (#PCDATA) >
<!ELEMENT Pretransfer_Commands (#PCDATA) >
<!ELEMENT Posttransfer_Commands (#PCDATA) >
<!ELEMENT SOCKS
(server_host_name,server_port,method,user_name,encrypted_password) >
<!ELEMENT server_host_name (#PCDATA) >
<!ELEMENT method (#PCDATA) >
<!ELEMENT payload (#PCDATA) >
```

Additional Information: Data Messages

The XML data message, **batch_eWay_data**, sent by the schema template is similar to the XML message receive order that initiated the transfer. The major difference is that, if the processing of a receive order is successful, the message contains a **payload** field with the data received. This data is encoded in the Base64 format, along with the original order record.

As a result of a receive order (if the command is **receive**) a data XML message is published by the schema template. See the following example:

```
<batch_eWay_data>
    <command>            receive          </command>
    <external_host_setup>
        <host_type>      Unix             </host_type>
        <user_name>      Alincoln         </user_name>
        <encrypted_password> liasdfLIJB   </encrypted_password>
        <file_transfer_method> ftp        </file_transfer_method>
    </external_host_setup>
    <subscribe_to_external>
        <remote_directory_name> /usr/home/honest_abe/from
            </remote_directory_name>
        <remote_file_regexp> Y*.dat       </remote_file_regexp>
    </subscribe_to_external>
    <payload>            (DATA encoded in Base64)</payload>
</batch_eWay_data>
```

Caution: *The payload* ***(DATA)*** *attribute must be encoded in the Base64 format.*

Payload Data

Normally, there is no payload data corresponding to a send order unless there is an error condition. If an order fails, the schema template can publish an error record, depending on your setting for the **Publish Status Record on Error** feature. If this parameter is set to **Yes**, the payload data is included in the XML error message.

8.3 Dynamic Configuration Template

This section explains how to import and use the Dynamic Configuration e*Gate schema template for the Batch e*Way. It also provides a general overview of how the template operates.

Note: *See the **e*Gate Integrator User's Guide** and **Creating an End-to-end Scenario With e*Gate Integrator** for more information on how to create an e*Gate schema.*

This e*Way provides a specialized transport component for incorporation into an operational schema. The schema also contains specialized Collaborations, linking different Event Types and IQs. Typically, other types of e*Ways (in addition to the Batch) also can be used as components of a Dynamic Configuration schema.

Using the Dynamic Configuration Template

It is recommended that you import this schema template into e*Gate in order to use the Batch e*Way's Dynamic Configuration feature. You must base all implementations of this feature on the template. The rest of this chapter explains how to implement and use the relevant features of the schema template.

Note: *After you are thoroughly familiar with this template schema, you can change its operation to suit your individual needs, if desired. However, it is recommended that you first gain **complete** familiarity with this schema's operation.*

8.3.1 Importing the Dynamic Configuration Schema Template

For detailed instructions on how to import this schema, see **"Implementation Overview" on page 127**. This section also provides a list of the basic steps in creating an e*Gate schema.

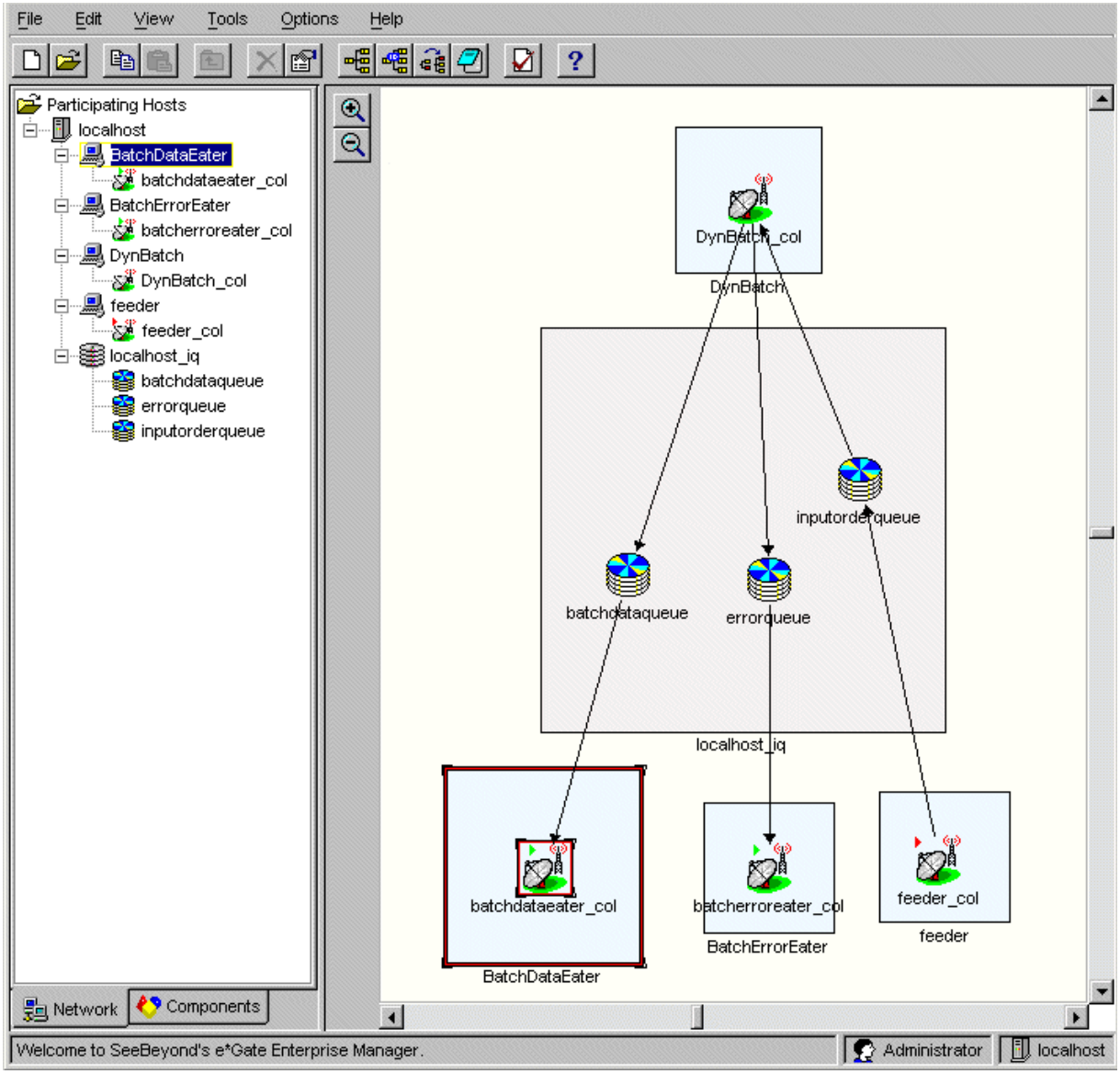
The template is on the installation CD-ROM in the following location:

\samples\ewbatch\Java\DynConfigSampleTemplate.zip

8.3.2 Schema Setup

Figure 141 on page 294 shows a diagram of the schema's general architecture, using the e*Gate Schema Designer's Network View. The arrows show the direction of data flow.

Figure 141 Dynamic Configuration Template Schema: Network View



8.3.3 Schema Operation

This schema template has the following input/output setup:

- **Input**: XML orders, via the **blob** Event Type, to the **feeder** file e*Way (one order per Event); can contain data or orders only.
- **Output**: Data in the form of XML messages (**BatchEwayData** Event Type); can also output error information (**BatchEwayError** Event Type).

Because this schema is a template, it is created with a modular architecture. It contains the basic framework of a Dynamic Configuration setup only. After you import the schema, you can add whatever functionality you want, for example, get and put operations to remote FTP systems or data record processing via the record-processing ETD.

Data enters through the **feeder** e*Way as XML order messages via the **blob** Event Type. The orders are published to the **inputorderqueue** IQ as the **BatchEwayOrder** Event Type.

The **DynBatch** e*Way and its Collaboration pick up this Event Type. The **DynBatch_col** Collaboration (via its Collaboration Rules) does the actual Dynamic Configuration processing and publishes XML-based Events to the **errorqueue** (error information) IQ, **batchdataqueue** (payload data) IQ, or both.

The file e*Ways operate as follows:

- **BatchDataEater** picks up the **BatchEwayData** Events (data messages) and writes out the payload data, if any, to a file in the designated target location.
- **BatchErrorEater** picks up the **BatchEwayError** Events (error messages) and writes out the success or error message.

The **DynBatch_col** Collaboration and its Collaboration Rules **cr_DynBatch** component make up the heart of the schema and do the actual Dynamic Configuration analyzing and processing. The Collaboration Rules component sends out data and error messages, as desired, based on the business logic (Business Rules) you configure in the schema.

Important: *As described previously, you can only input one XML order message per Event into the **DynBatch_col** Collaboration. The Dynamic Configuration processor can only process one order per a single Event.*

8.3.4 Schema Components

The most important Dynamic Configuration components in this schema template are:

- e*Way Connections
- Collaborations
- Collaboration Rules

These are the components that control the Dynamic Configuration feature's operation.

The template contains all the additional components of an operating e*Gate schema, including:

- Event Types
- ETDs
- e*Ways
- IQs and IQ Manager

Table 11 lists the names of the major components in this schema template, along with a brief description of each one.

Table 11 Components in Schema Template

Component	Name in Schema	Description
Event Types	BatchEwayData	For the batch_eway_data.dtd output data XML Events.
	BatchEwayError	For the batch_eway_error.dtd output error XML Events.
	BatchEwayOrder	For the batch_eway_order.dtd input order XML Events.
	FtpETD	For the FTP ETD (for FTP operations).
	LocalFileETD	For the local file ETD (for local file operations.)
	blob	For input XML messages coming into e*Gate to the feeder e*Way.
e*Ways	BatchDataEater	File e*Way contains the batchdataeater_col Collaboration.
	BatchErrorEater	File e*Way contains the batcherroreater_col Collaboration.
	DynBatch	Multi-Mode e*Way contains the DynBatch_col Collaboration.
	feeder	File e*Way contains the feeder_col Collaboration.
e*Way Connections	ConDynFtp	For remote FTP system connections.
	ConDynLocalFile	For local file system connections.
IQ Manager	localhost_iq	Manages the schema's IQs.
IQs	batchdataqueue	Holds BatchEwayData Events from the DynBatch_col Collaboration.
	errorqueue	Holds BatchEwayError Events from the DynBatch_col Collaboration.
	inputorderqueue	Holds BatchEwayOrder Events from the feeder_col Collaboration.
Collaboration Rules	cr_batchdataeater	Sends out Dynamic Configuring messages output data messages.
	cr_batcherroreater	Sends out Dynamic Configuration output error messages.
	cr_DynBatch	Performs the Dynamic Configuration analyzing and processing.
	cr_feeder	Brings messaging XML order data into e*Gate for Dynamic Configuration processing.

Table 11 Components in Schema Template (Continued)

Component	Name in Schema	Description
Collaborations	batchdataeater_col	Executes the cr_batchdataeater Collaboration Rules.
	batcherroreater_col	Executes the cr_batcherroreater Collaboration Rules.
	DynBatch_col	Executes the cr_DynBatch Collaboration Rules.
	feeder_col	Executes the cr_feeder Collaboration Rules.

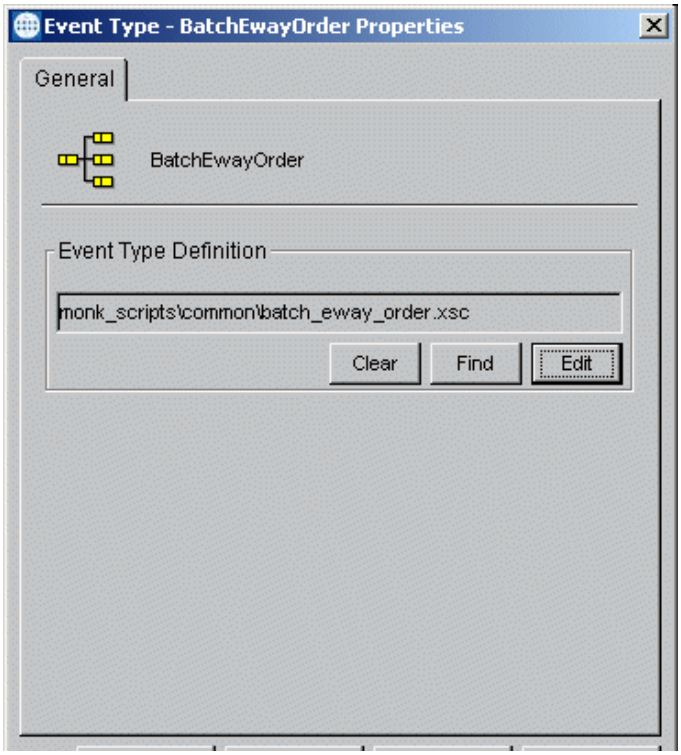
For instructions on how to create and define components in an e*Gate schema with the Batch e*Way, see **"Sample Schema: Basic FTP With Streaming" on page 129**.

The rest of this section explains the operation and definition of the e*Gate components in this schema template, with special emphasis on the e*Way Connections, Collaborations, and Collaboration Rules.

8.3.5 Overview of Event Types and ETDs

As shown in the **Event Type Properties** dialog box example in Figure 142, the schema contains an Event Type for each of its precompiled ETDs. For example, the figure shows the BatchEwayOrder Event Type for the **batchewayorder.xsc** ETD. This ETD corresponds to the **batch_eway_order.dtd** XML message.

Figure 142 BatchEwayOrder Event Type Properties Dialog Box



The following list shows the schema's Event Types with their corresponding ETD and DTD files, along with an explanation of each:

- **BatchEwayData: batchewaydata.xsc**

  Utilizes the precompiled ETD based on the **batch_eway_data.dtd** XML file to carry data messages.

- **BatchEwayError: batchewayerror.xsc**

  Utilizes the precompiled ETD based on the **batch_eway_error.dtd** XML file to carry error messages.

- **BatchEwayOrder: batchewayorder.xsc**

  Utilizes the precompiled ETD based on the **batch_eway_order.dtd** XML file to carry order messages (send and receive).

- **FtpETD: FtpETD.xsc**

  Enables data transactions with remote FTP systems.

Note: *For more information on the FTP ETD, see* **"ETD for FTP Operations" on page 89***.*

- **LocalFileETD: LocalFileETD.xsc**

  Enables data transactions with local file systems.

Note: *For more information on the local file ETD, see* **"ETD for Local File" on page 100***.*

- **blob: blob.xsc**

  Used for raw input data, for example, an XML order.

Note: *This ETD is user-defined. For information on how to use the Custom ETD wizard to create a user-defined ETD, see* **"Creating Event Types and ETDs" on page 173***.*

8.3.6 Configuring the e*Way Connections

This section explains how to configure the e*Way Connections for the Dynamic Configuration feature and schema template. The configuration parameters set in these e*Way Connections serve as the default order message parameter values for the schema template. You can overwrite any one or more of them with the individual XML orders as desired.

Note: *The default configuration parameter settings in the e*Way Connections must be valid values. For example, if the default value of the* ***Host Name*** *parameter is* ***localhost****, an FTP server on the machine* ***localhost*** *must be accessible.*

To configure the ConDynFtp e*Way Connection

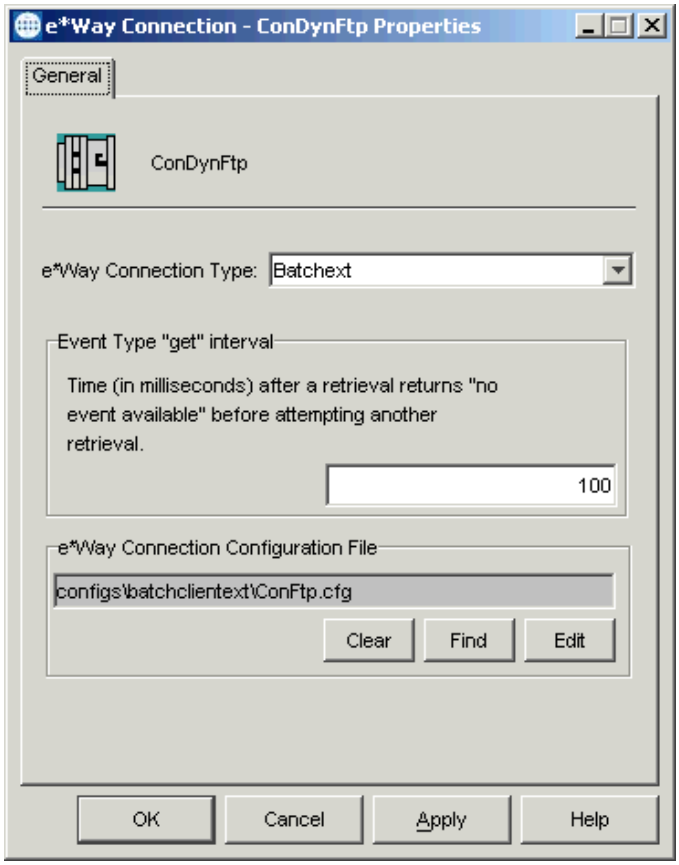
1. Run the e*Gate Schema Designer and go to the Main window.

Note: *If you want to change the name of an e*Way Connection, make sure that you also update the corresponding publication destination as entered in any relevant Collaboration Properties dialog box.*

2 In the **Component** pane, select the **ConDynFtp** e*Way Connection, then right-click to edit and view its properties.

3 When the **e*Way Connection Properties** dialog box opens, you see the defined properties as shown in **Figure 143 on page 299**.

Figure 143 ConDynFtp e*Way Connection Properties Dialog Box



You can adjust the Event Type "get" interval as needed.

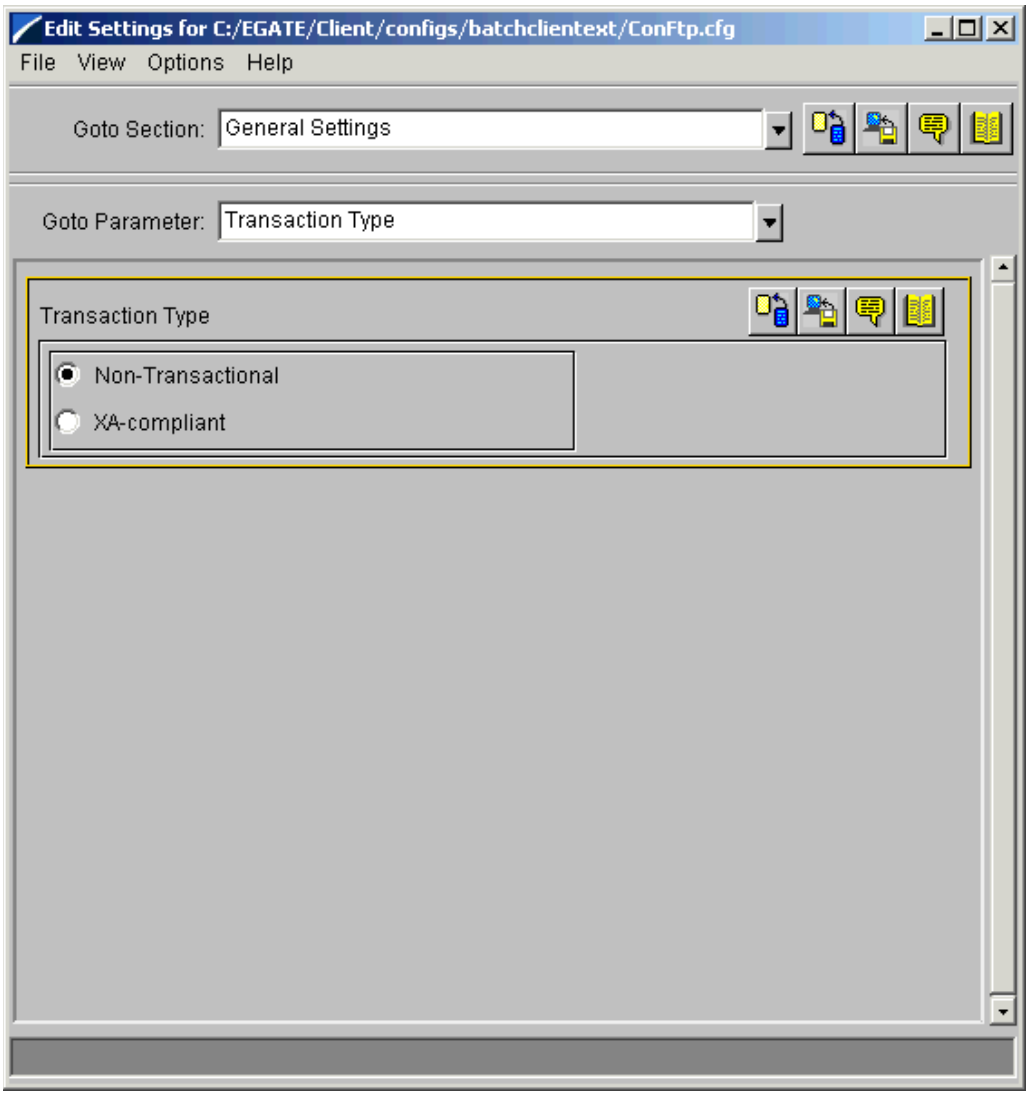
4 Under **e*Way Connection Configuration File**, click **Edit** to edit the FTP ETD's configuration parameters.

The e*Way Configuration Editor Main window opens. Use this interface to select the desired parameters, including those that correspond to the remote FTP system you are using.

Note: *See* **"FtpETD: Configuration Parameters" on page 36** *for details on setting these parameters.*

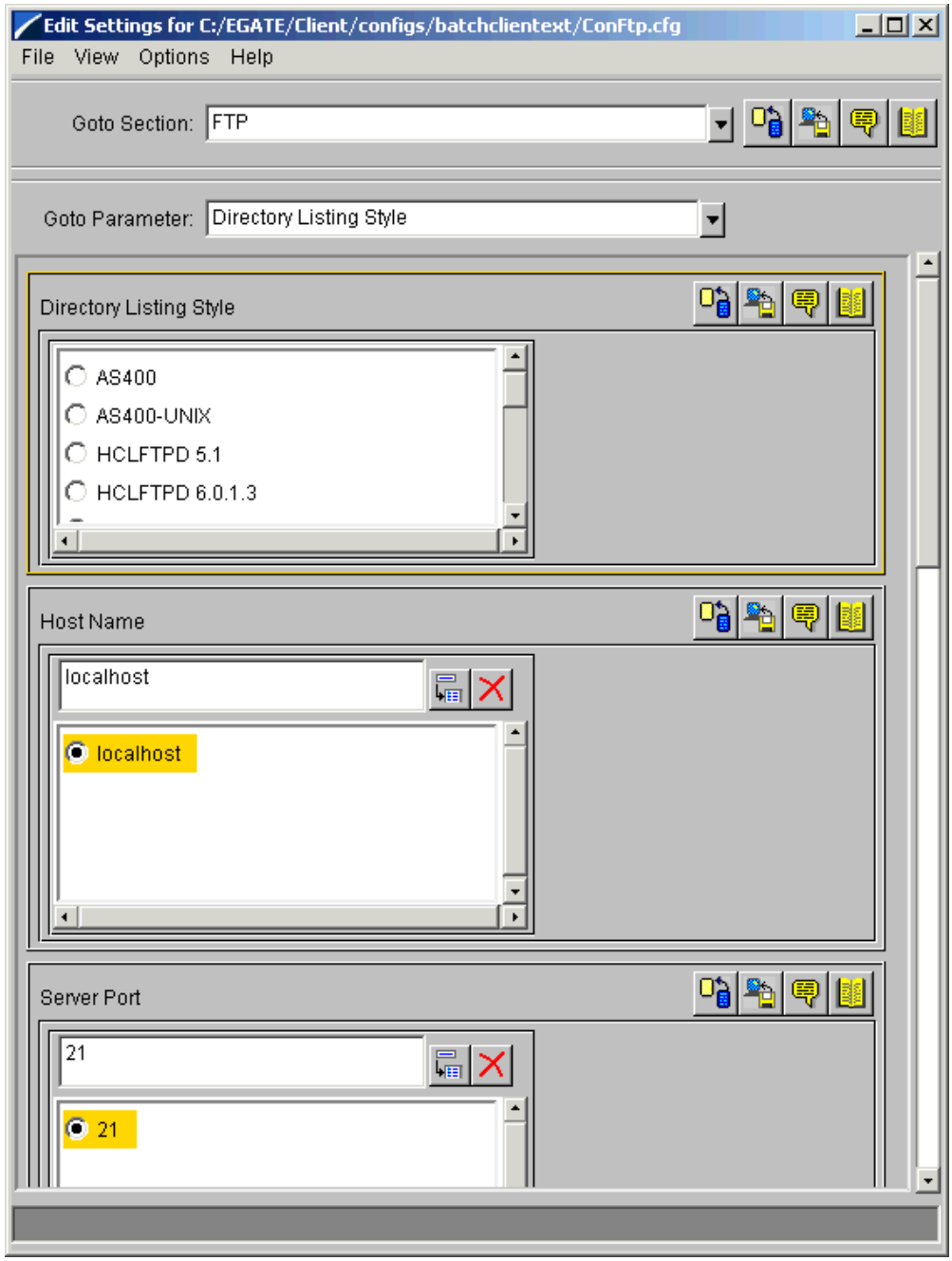
5 Under the **General Settings**, make sure that the XA mode is not enabled (see **Figure 144 on page 300**).

Figure 144 e*Way Configuration Editor: ConDynFtp General Settings



6 Select the **FTP** settings (see Figure 145).

Figure 145 e*Way Configuration Editor: ConDynFtp FTP Settings 1

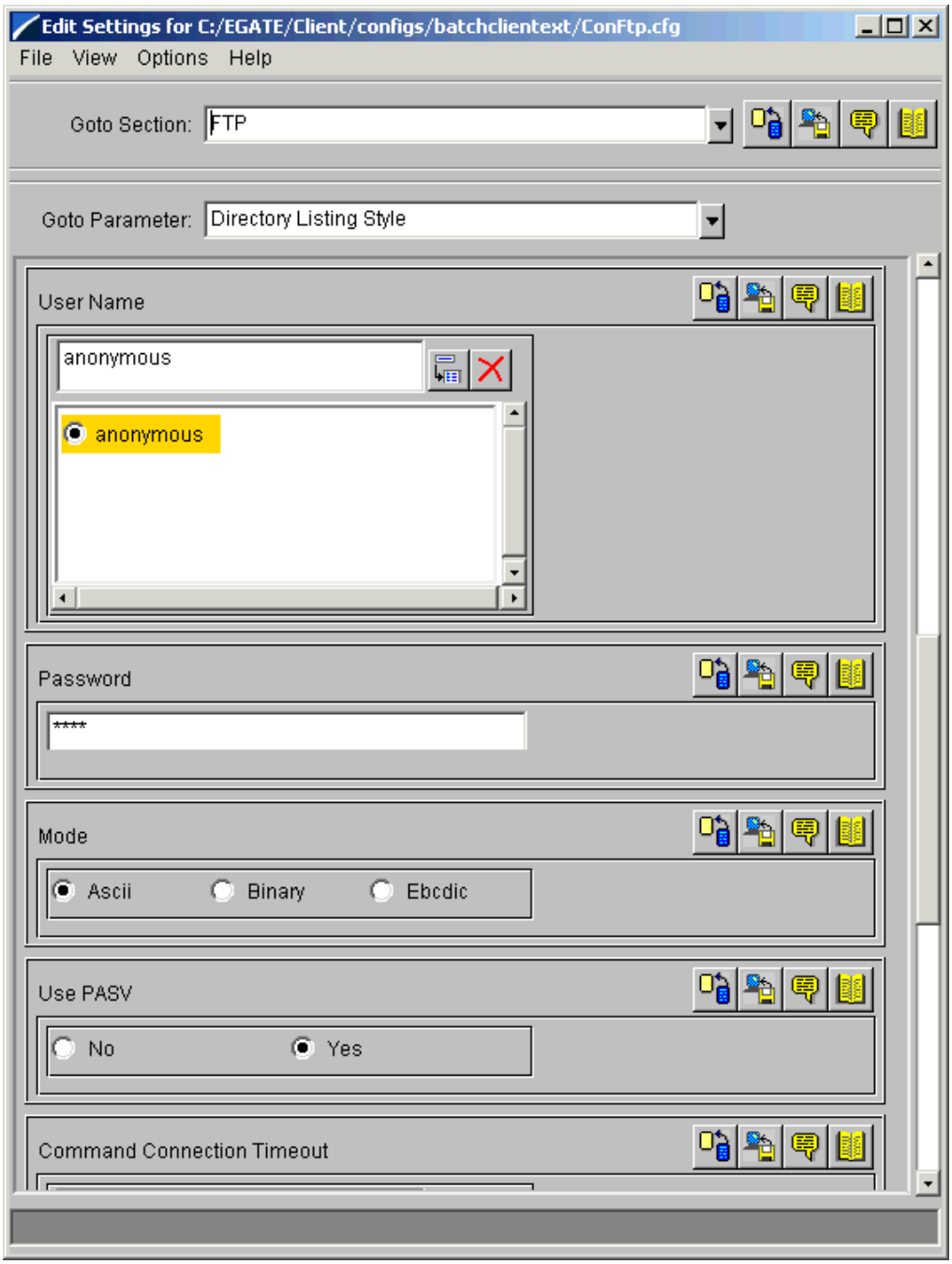


7 Under the **FTP** settings, accept the default settings for all parameters except:

- **Directory Listing Style**
- **Host Name**
- **Server Port**
- **User Name**
- **Password**

You must enter your default system settings for these parameters. Also, see **Figure 146 on page 303**.

Figure 146 e*Way Configuration Editor: ConDynFtp FTP Settings 2



8 Select the **Target Location** settings (see Figure 147 on page 304).

9 Under the **Target Location** settings, you can accept the default settings for all parameters except:

- **Target Directory Name**
- **Target File Name**

You must enter your system default settings for these parameters (see Figure 147).

10 Make sure to set *all* of the pre/post file transfer command parameters as desired (see **Figure 148 on page 305** and **Figure 149 on page 306**). Your desired settings may differ from those in the schema template, depending on your system needs.

Figure 147 e*Way Configuration Editor: ConDynFtp Target Location Settings

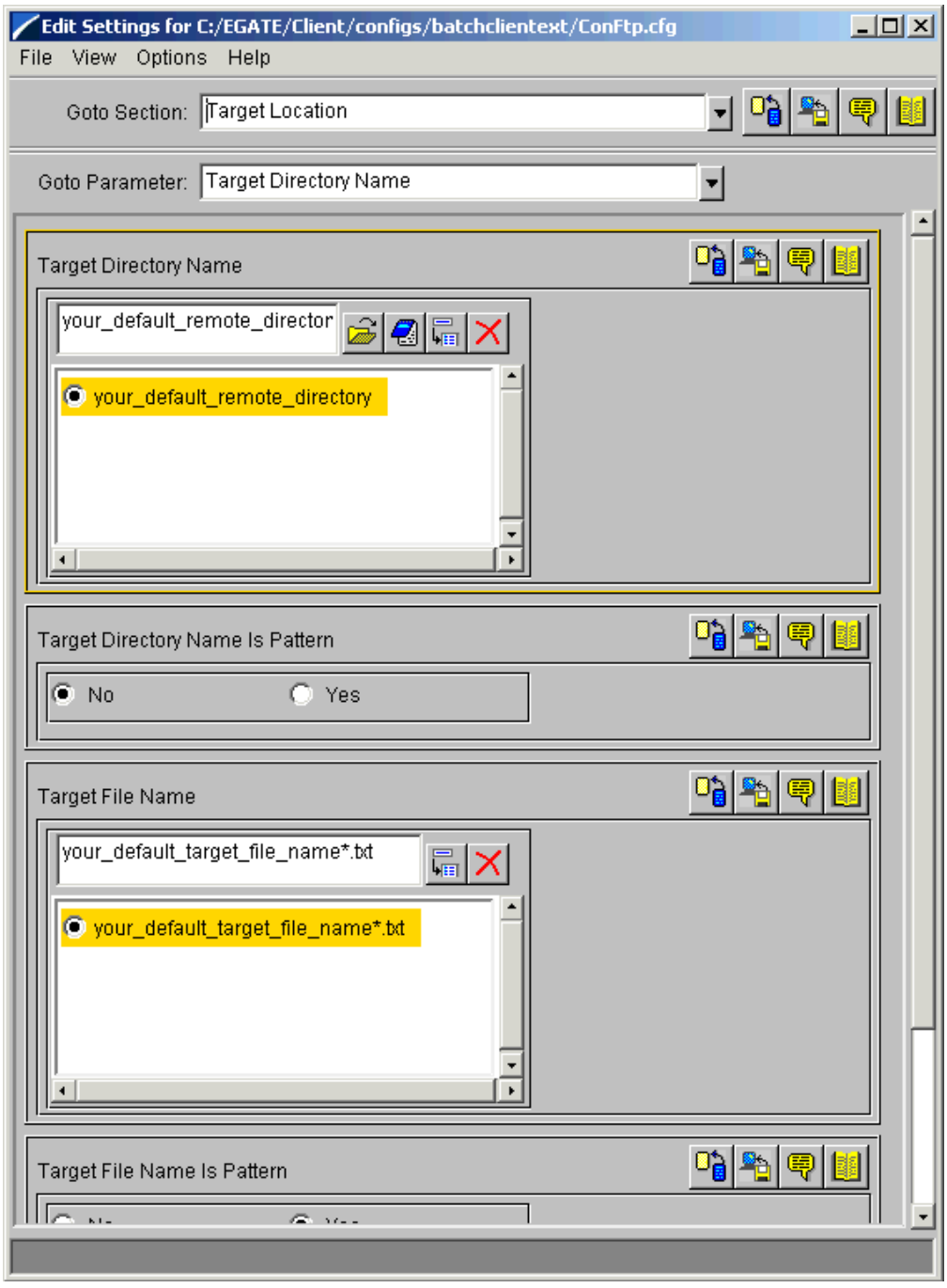


Figure 148 e*Way Configuration Editor: ConDynFtp Pre File Transfer Settings

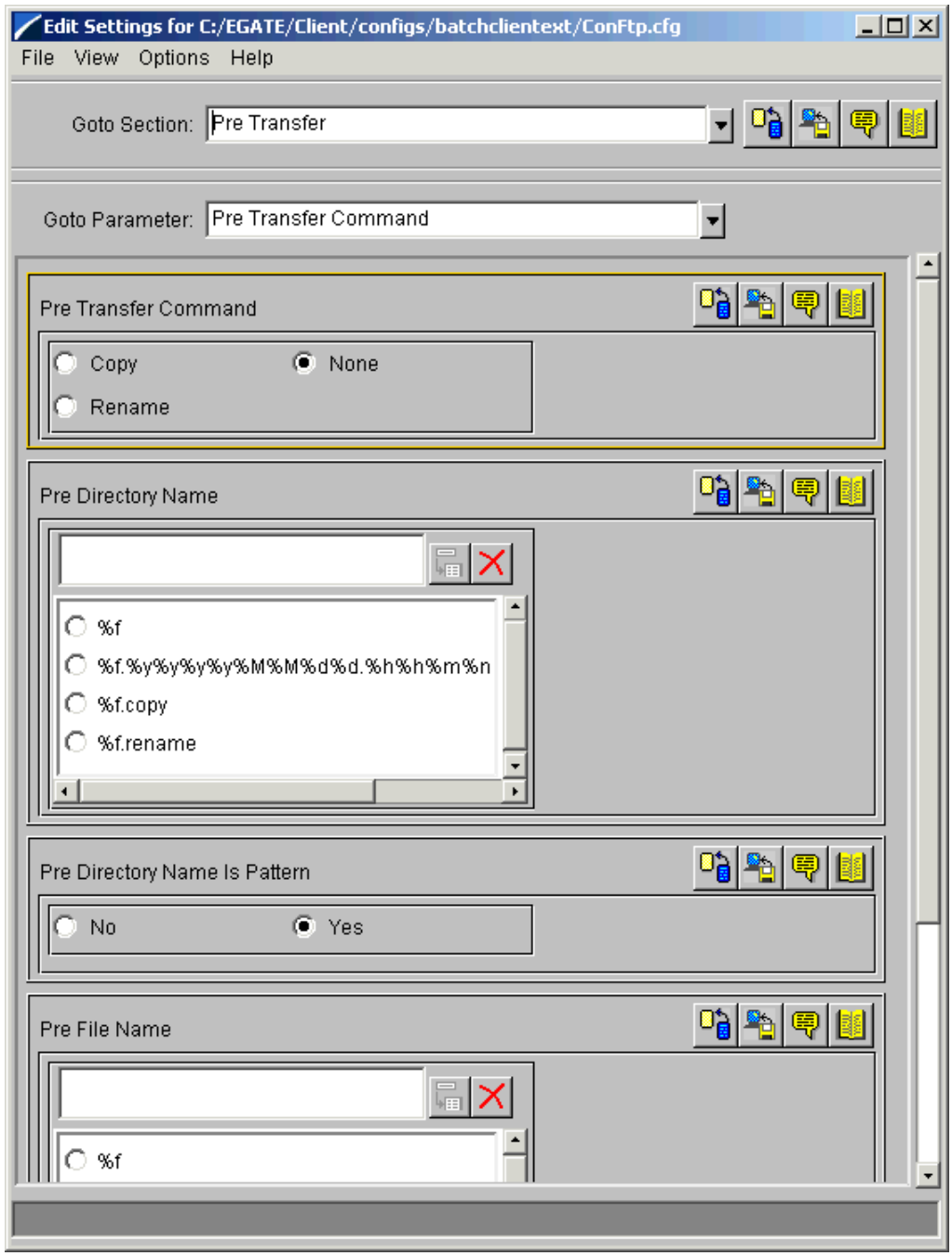
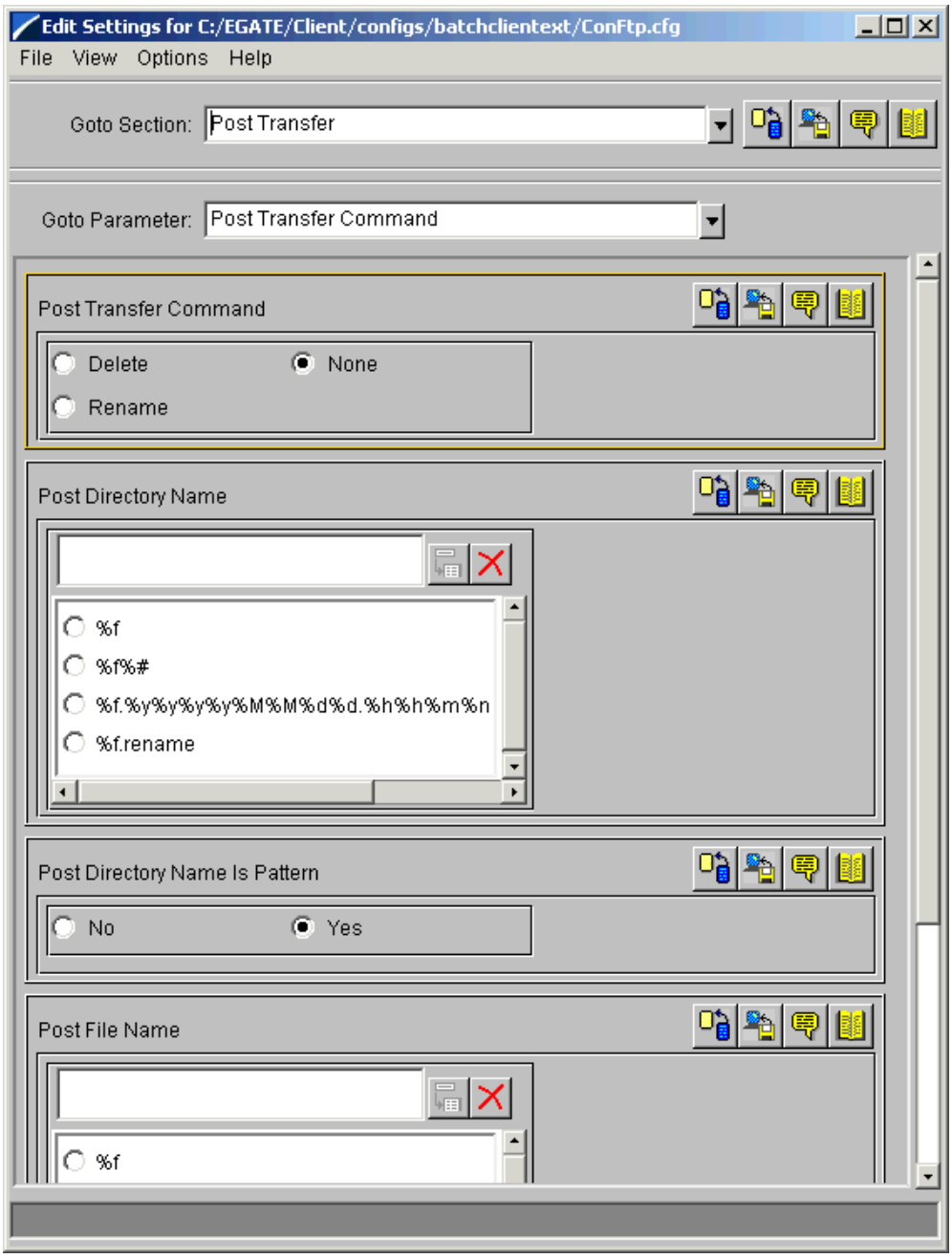
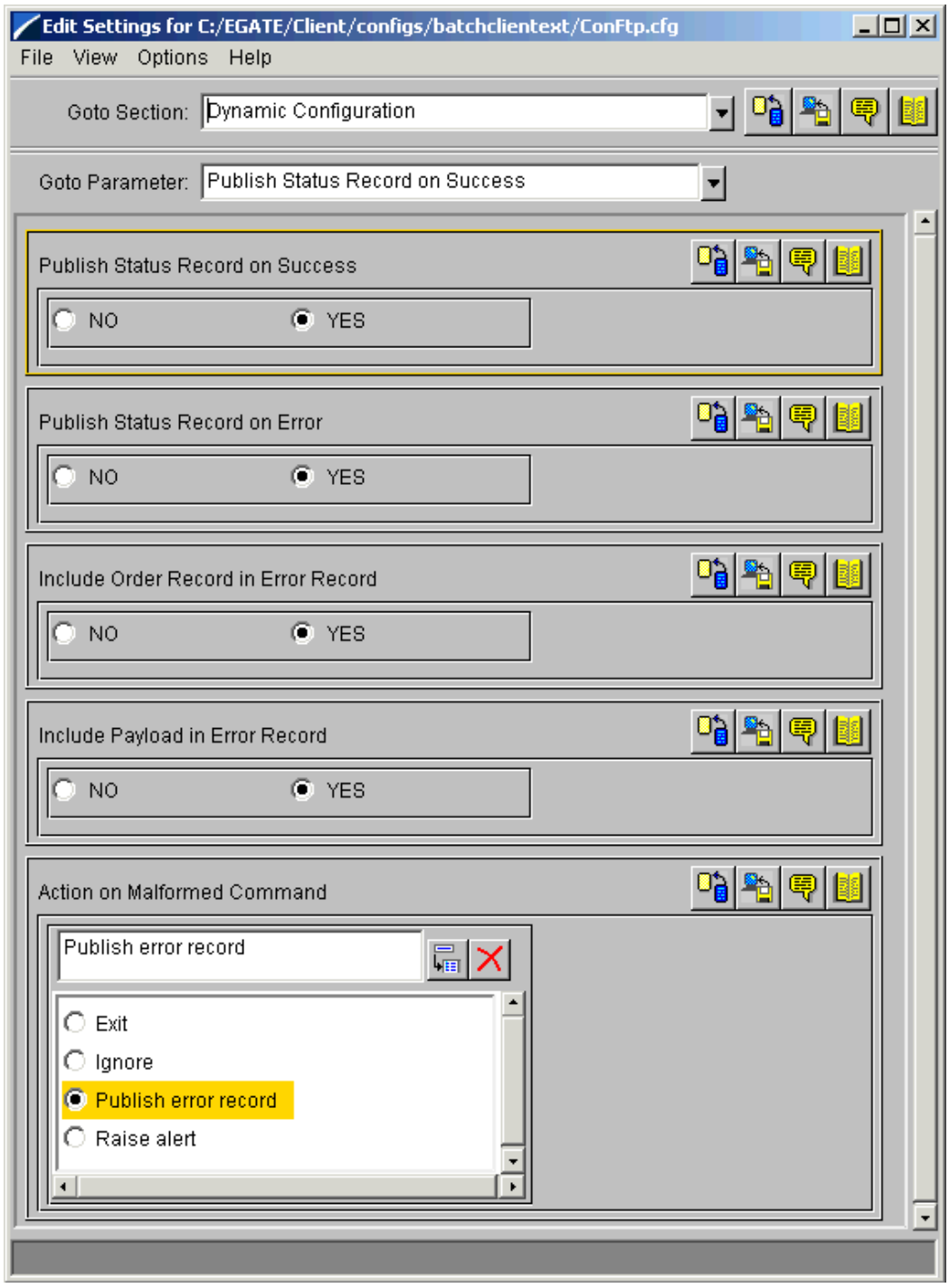


Figure 149 e*Way Configuration Editor: ConDynFtp Post File Transfer Settings



11 In addition to the parameters listed previously, you must also enter parameters for the Dynamic Configuration feature. Select the **Dynamic Configuration** settings (see **Figure 150 on page 307**) and set these parameters as desired.

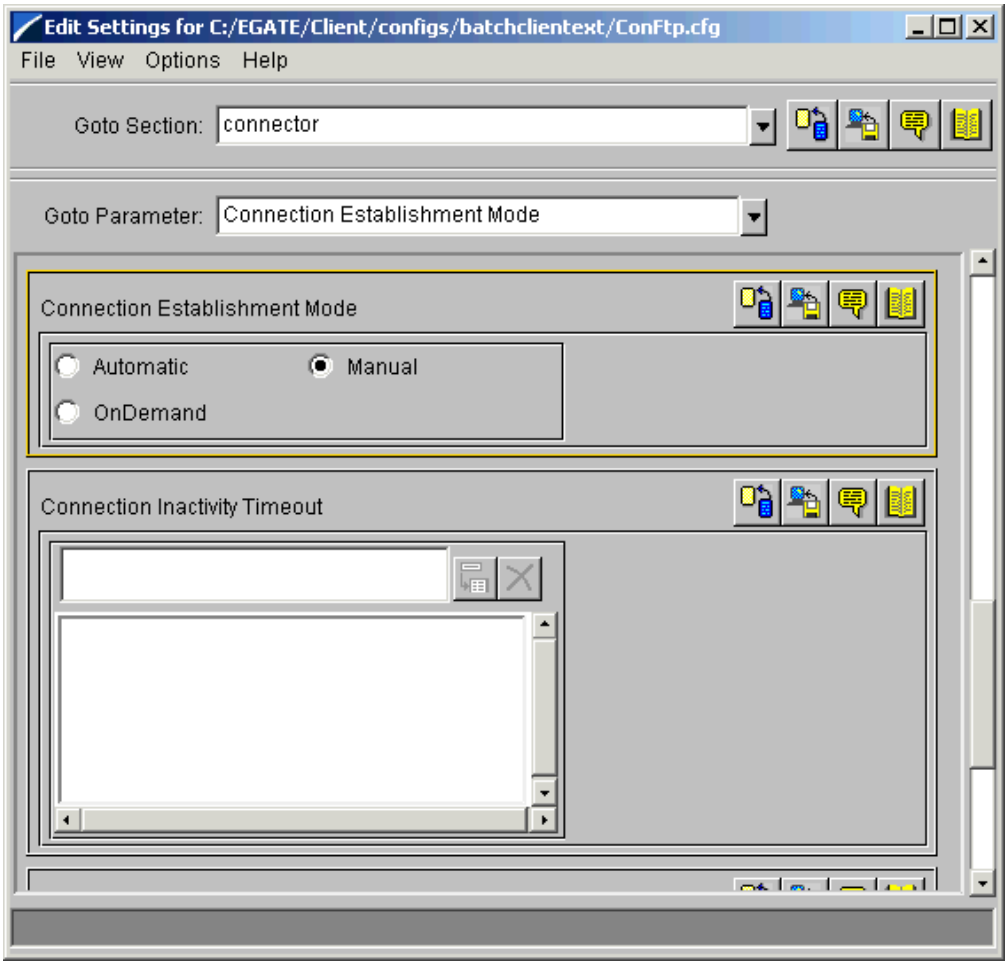
Figure 150 e*Way Configuration Editor: ConDynFtp Dynamic Configuration Settings



Note: *It is recommended that the Dynamic Configuration parameter settings be the same for the corresponding local file ETD e*Way Connection so that the behavior for FTP and local file operations are identical. For more information, see* **"Dynamic Configuration" on page 57**.

12 Make sure that the **Connection Establishment Mode** is set to **Manual** (see Figure 151) under the **connector** configuration settings.

Figure 151 e*Way Configuration Editor: Connector Settings



13 Set the rest of these parameters in the same way either as they are set in the schema template or as required by your own system.

14 When you are finished, save the .**cfg** file, close the e*Way Configuration Editor, and promote the file to run time.

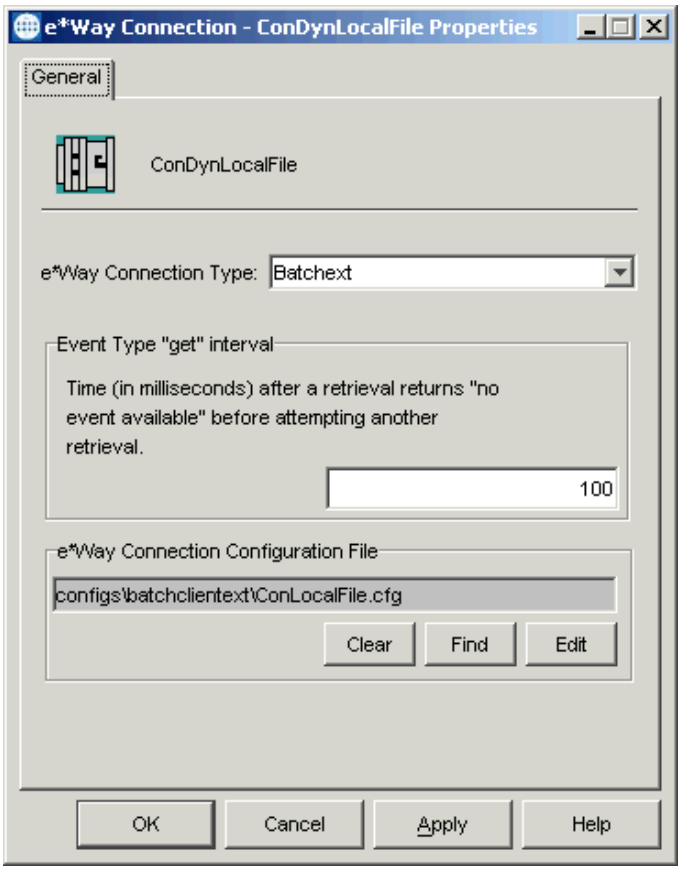
15 Click **OK** to close the **e*Way Connection Properties** dialog box.

To configure the ConDynLocalFile e*Way Connection

1 Access the **e*Way Connection Properties** dialog box for this component in the same way as you did for the previous e*Way Connection. The name of this e*Way Connection in the schema template is **ConDynLocalFile**.

2 Figure 152 shows the schema's default settings. Set the "get" interval as desired.

Figure 152 ConDynLocalFile e*Way Connection Properties Dialog Box

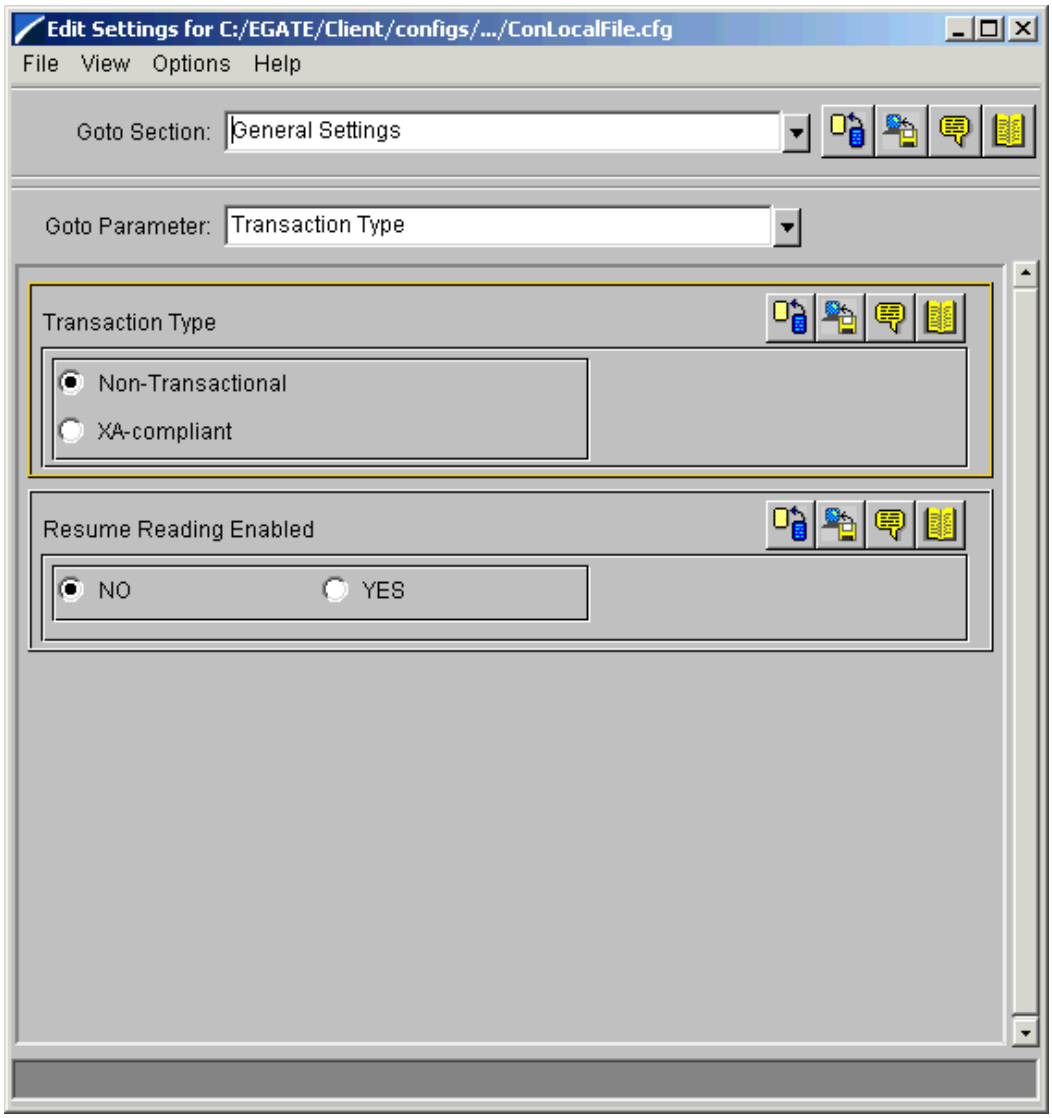


3 Under **e*Way Connection Configuration File**, click **Edit** to edit the local file ETD's configuration parameters.

The e*Way Configuration Editor Main window opens. Use this interface to select the desired parameters, including those that correspond to the local file system you are using. See **"LocalFileETD: Configuration Parameters" on page 59** for details.

4 Under the **General Settings**, make sure that the XA mode is not enabled (see **Figure 153 on page 310**). Also, make sure that **Resume Reading** is not enabled.

Figure 153 e*Way Configuration Editor: ConDynLocalFile General Settings



5 Select the **Target Location** settings (see **Figure 154 on page 311**).

6 Under the **Target Location** settings, you can accept the default settings for all parameters except for:

- **Target Directory Name**
- **Target File Name**

You must enter your system default settings for these parameters (see **Figure 154 on page 311**).

7 Make sure to set *all* of the pre/post file transfer command parameters as desired (see **Figure 155 on page 312** and **Figure 156 on page 313**). Your desired settings may differ from those in the schema template, depending on your system needs.

Figure 154 e*Way Configuration Editor: ConDynFtp Target Location Settings

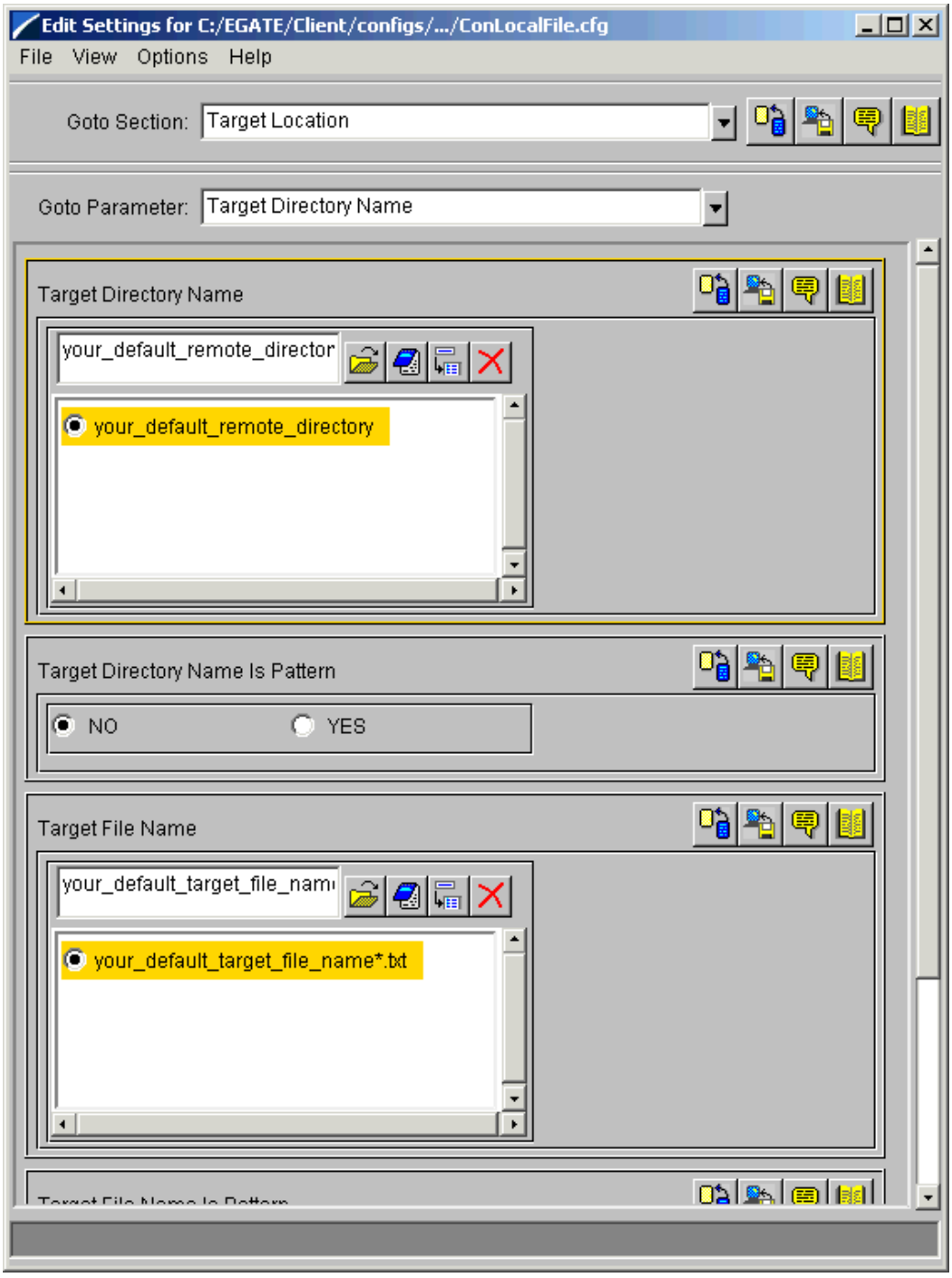


Figure 155 e*Way Configuration Editor: ConDynFtp Pre File Transfer Settings

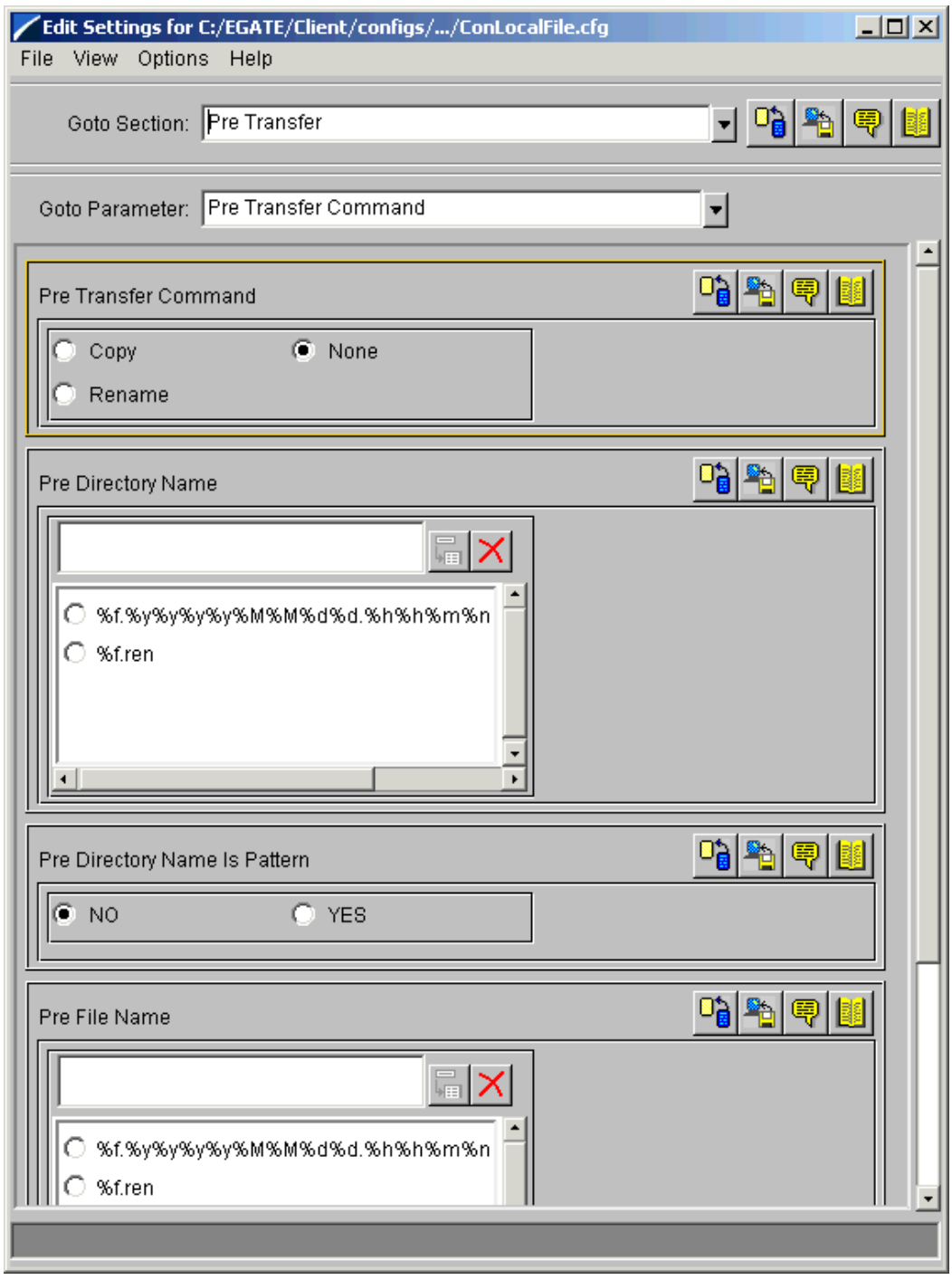
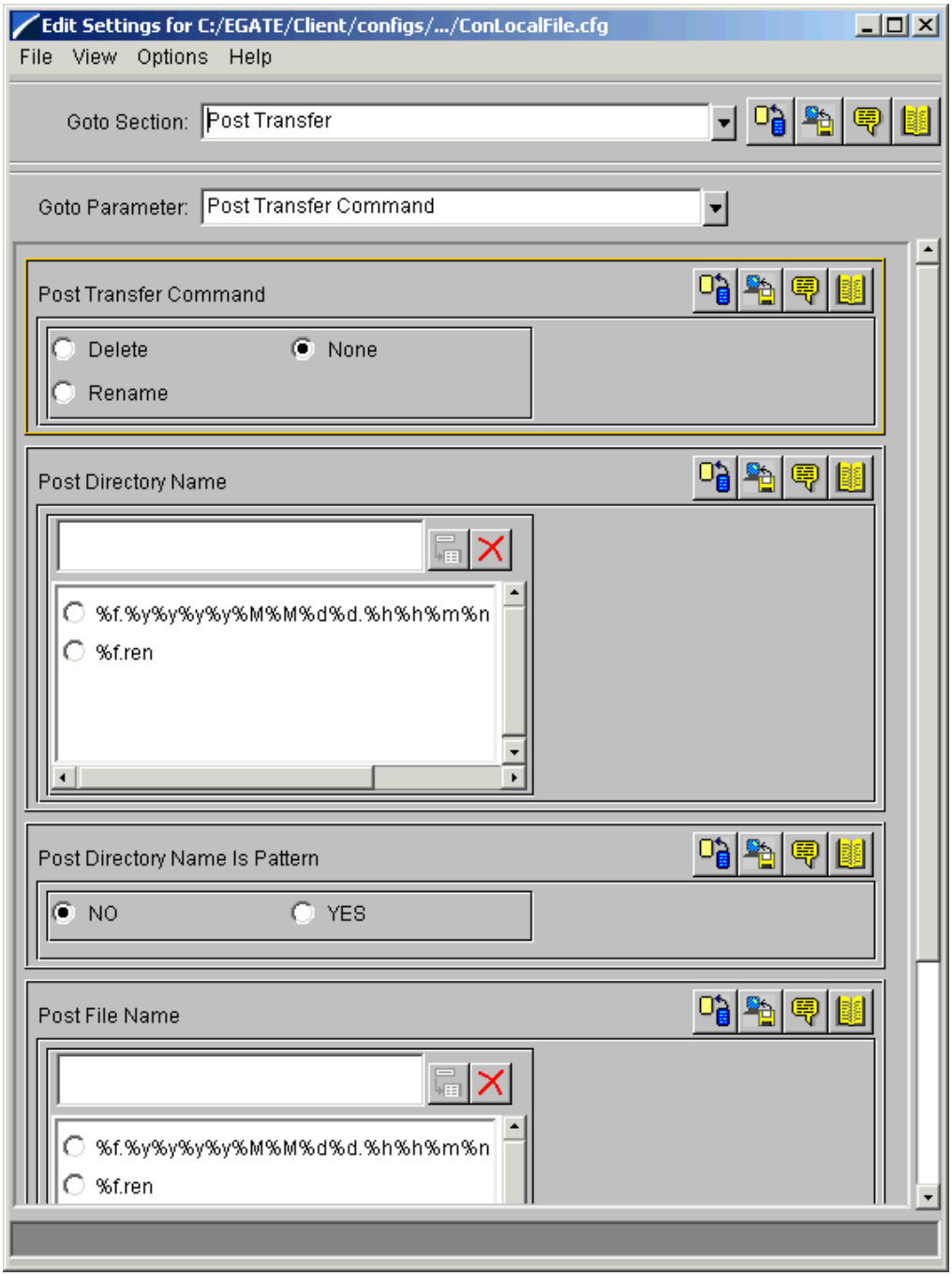
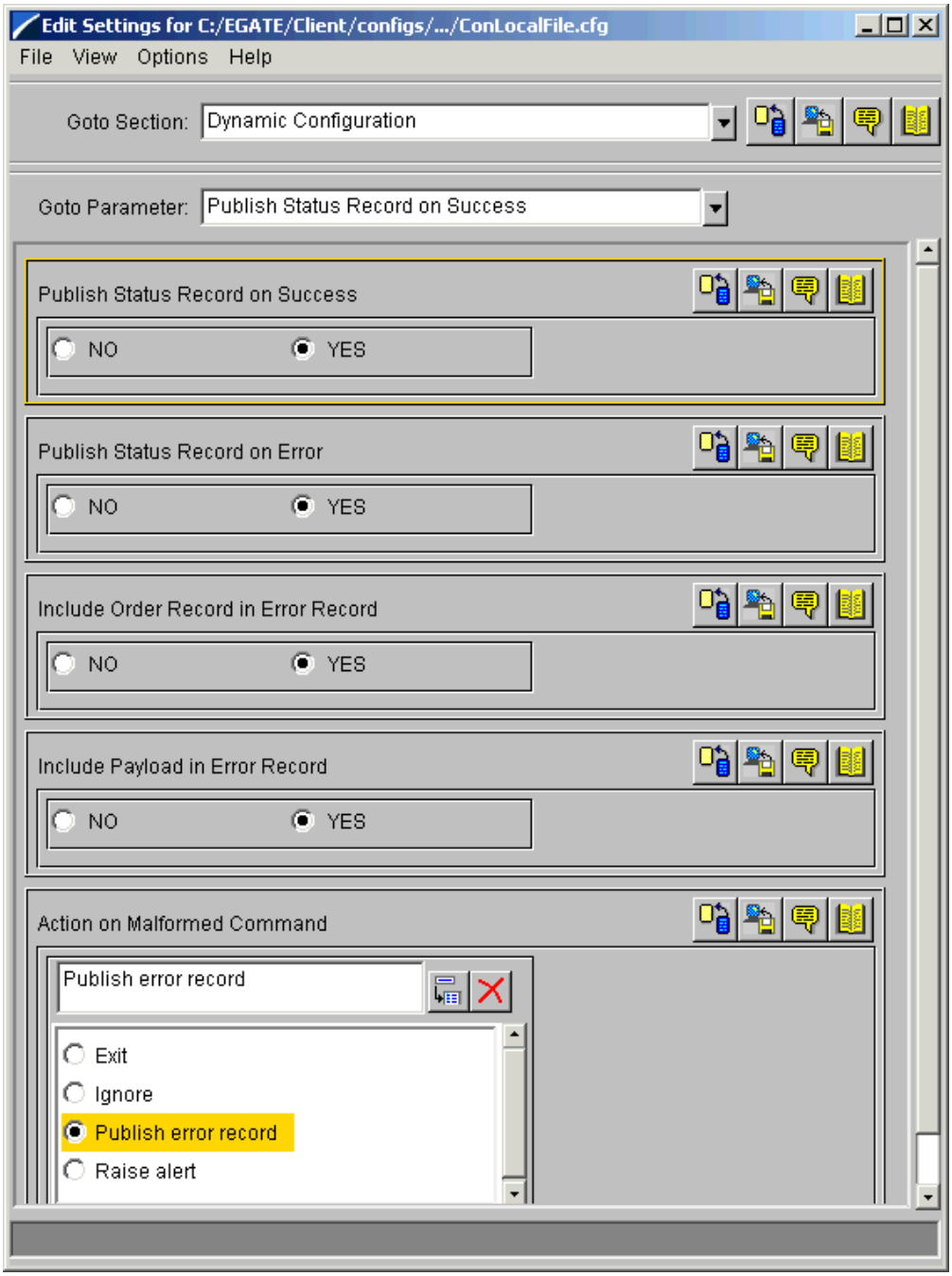


Figure 156 e*Way Configuration Editor: ConDynFtp Post File Transfer Settings



8 In addition to the parameters listed previously, you must also enter parameters for the Dynamic Configuration feature. Select the **Dynamic Configuration** settings (see **Figure 157 on page 314**) and set these parameters as desired.

Figure 157 e*Way Configuration Editor: ConDynFtp Dynamic Configuration Settings



Note: *It is recommended that the Dynamic Configuration parameter settings be the same for the corresponding FTP ETD e*Way Connection so that the behavior for FTP and local file operations are identical. For more information, see* **"Dynamic Configuration" on page 67**.

9 Set the rest of these parameters in the same way either as they are set in the schema template or as required by your own system.

10 When you are finished, save the .**cfg** file, close the e*Way Configuration Editor, and promote the file to run time.

11 Click **OK** to close the **e*Way Connection Properties** dialog box.

8.3.7 Configuring the File e*Ways

Make sure the following configuration parameters (**feeder.cfg** file) are set in the schema template's file e*Ways:

When configuring the feeder file e*Way

- Make sure that the **MultipleRecordsPerFile** parameter is set to **No**. This way, even if the XML order input file contains carriage returns, the feeder Collaboration still accepts the whole file as a single record.
- Increase the **MaxBytesPerLine** parameter setting to account for the largest data size (Base64 or not) and the rest of the XML order message; the default is 64 KB.
- Set the default input directory (**PollDirectory** parameter) as desired.

When configuring the BatchDataEater and BatchErrorEater file e*Ways

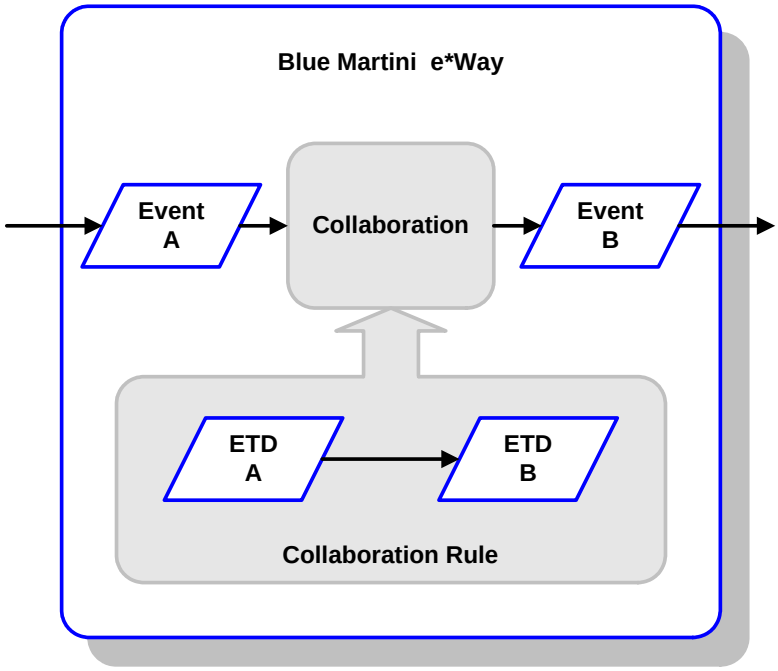
- Make sure that the **MultipleRecordsPerFile** parameter is set to **No** here also, so that a new XML data and error file is generated for each new XML order.
- Set the default output directory (**OutputDirectory** parameter) as desired.

Note: *In adapting this template schema to your needs, keep in mind that the output file becomes an XML ETD published to another Collaboration via an IQ.*

8.3.8 Collaboration Rules and Collaboration Operation

Collaborations transform incoming Events into outgoing Events. A Collaboration is driven by a Collaboration Rules script component, which defines the relationship between the source (incoming) and destination (outgoing) ETDs (see **Figure 158 on page 316**).

Figure 158 Collaboration and Collaboration Rules Operation



Note: *Java Collaboration Rules can have more than one source and more than one destination Event.*

Using the Predefined Collaboration Rules

The current version of the Batch e*Way provides predefined Dynamic Configuration **.class** files in the schema template. These files are based on e*Gate **.xpr** Collaboration Rules files.

While it is possible to create your own Collaboration Rules to control this feature, in most cases the predefined Collaboration Rules component is sufficient.

Collaboration Rules Components: File e*Ways

The schema template contains the following file e*Ways:

- **BatchDataEater**
- **BatchErrorEater**
- **DynBatch**

Through their Collaborations, these e*Ways contain the following Collaboration Rules components:

- **cr_feeder**
- **cr_batchdataeater**
- **cr_batcherroreater**

Collaboration Rules Component: cr_DynBatch

The central Collaboration Rules component in the schema template is **cr_DynBatch** in the **DynBatch_col** Collaboration. This component does the actual Dynamic Configuration message processing. Use its Business Rules to control your implementation of the Dynamic Configuration feature.

Figure 159 and **Figure 160 on page 318** show the properties for the **cr_DynBatch** Collaboration Rule. Note that on the **Collaboration Mapping** tab, under **Mode**, the input is order messages and the output is data and error messages. The FTP and local file ETD outputs allow you to additionally send data using these ETDs.

Figure 159 cr_DynBatch Properties: General Tab

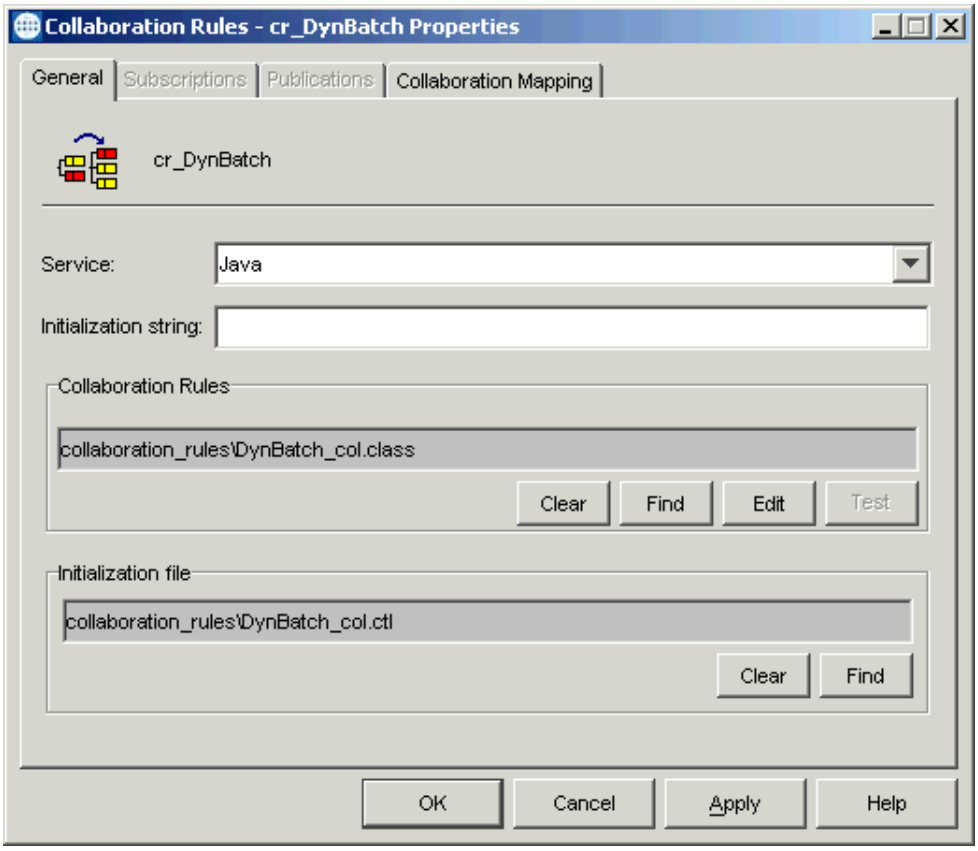
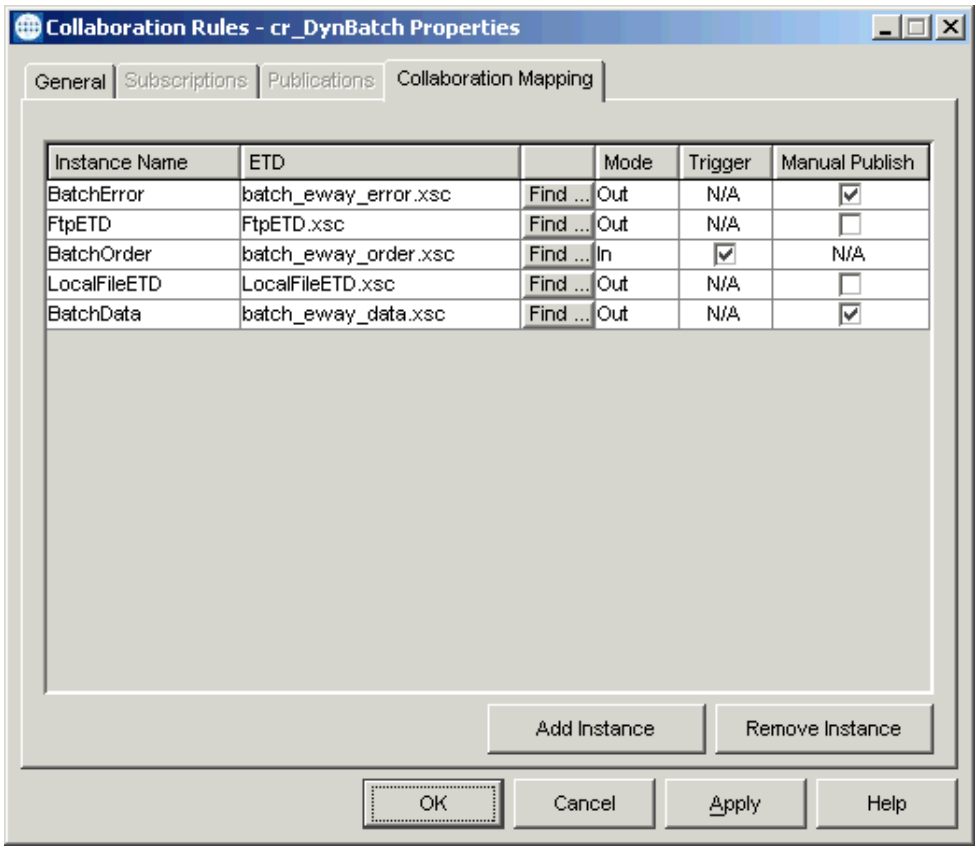


Figure 160 cr_DynBatch Properties: Collaboration Mapping Tab

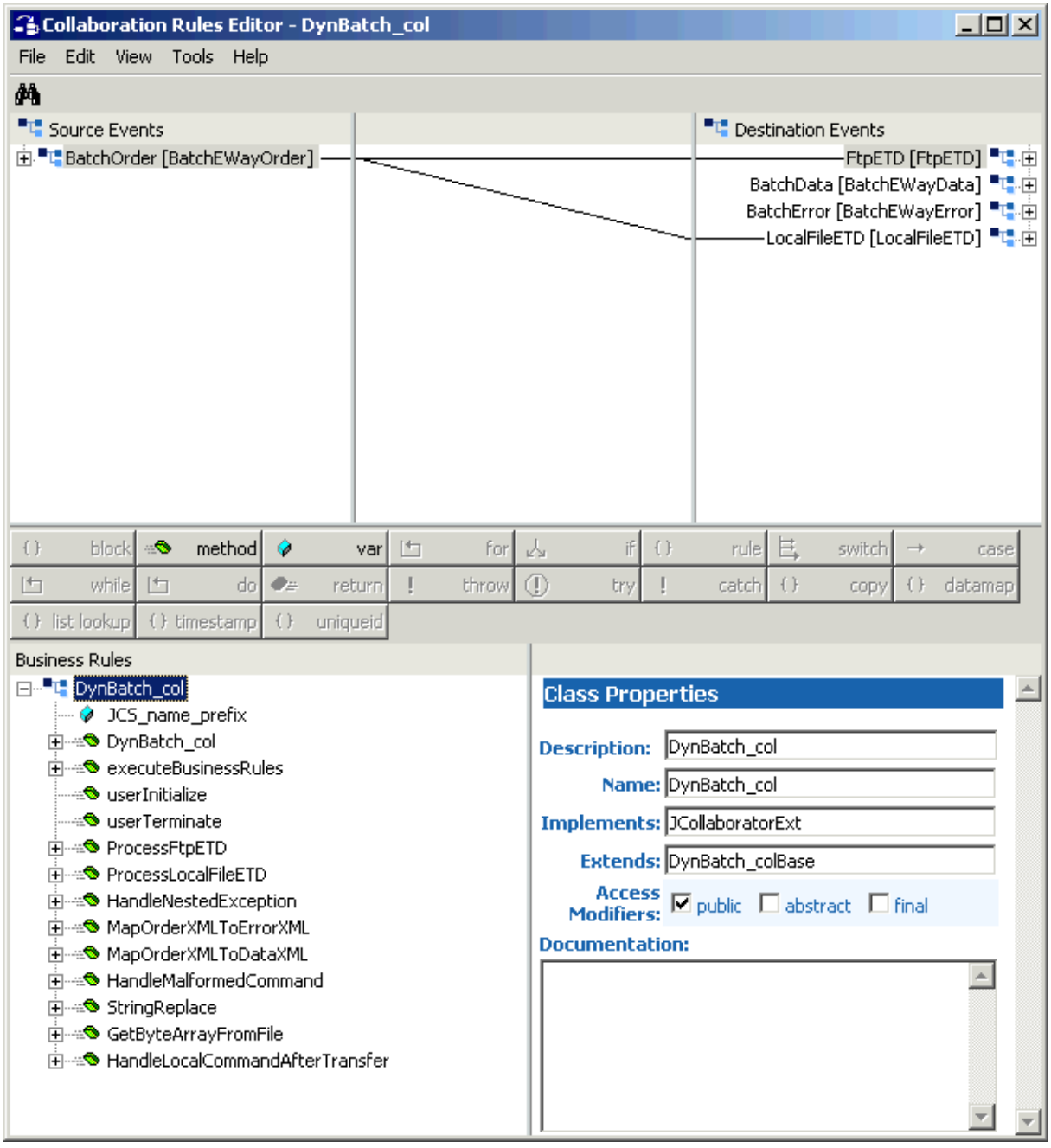


Make sure that the **BatchError** and **BatchData** instances are set to **Manual Publish** (see Figure 160). You must use these settings because the **DynBatch_col** Collaboration may not always automatically generate these two instances.

For example, if the **Publish Status Record on Success** and **Publish Status Record on Error** parameters are both set to **No**, then no **BatchError** Event is generated. However, if all the incoming orders in this case are send orders, there are no outbound XML data messages.

Figure 161 on page 319 shows the **cr_DynBatch** Collaboration Rules file open in the Collaboration Rules Main window. The Business Rules are contracted to show only the top-level Business Rules in the **Business Rules** window.

Figure 161 Collaboration Rules Editor: Collaboration Rules File for cr_DynBatch



This Collaboration Rules component is the primary Dynamic Configuration processor. Its purpose is to analyze the incoming XML orders and map all relevant elements in each order to corresponding parameters of the FTP or local file ETD.

Note: *For the Dynamic Configuration feature, your Collaboration must call* ***connect()*** *inside the* ***executeBusinessRules()*** *method after all the values have been set from the incoming XML message.*

In general, this Collaboration Rules component performs the following tasks:

- Checks the incoming XML order to make sure it is in the appropriate format; if so, the Event is processed, but if not, it is rejected
- In case of rejection, performs malformed command handling logic according to the **Action on Malformed Command** parameter setting; the options in case of an error are:
  - **Exit**: Shut down the e*Way.
  - **Ignore**: The e*Way does nothing.
  - **Raise Alert**: Send an Alert to the Schema Manager.
  - **Publish Error Record**: Publish an XML error status record.

Note: *See* **Chapter 4** *for complete information on the e*Way's configuration parameters.*

- Processes and analyzes the incoming order to map its output to the FTP or local file ETD (must be one or the other)
- Processes and analyzes the incoming order to map its output and generate XML message output, either data or error, or both
- Handles exceptions, if present

You can open the **cr_DynBatch** file in the Collaboration Rules Editor yourself to observe its exact Business Rule structure. Each Business Rule is commented to explain its specific function to the user.

Outline of Operation

The following list provides a high-level outline of how the **cr_DynBatch** Collaboration Rules component operates and what it does:

Main executeBusinessRules Operation

1. Checks the transaction type of the ETD and invokes the malformed command handler if the ETD is set to XA-compliant.
2. Invokes the FTP ETD or local file ETD handler according to the **FileTransferMethod** of the XML order and invokes the malformed command handler if it is invalid.
3. Restores all the original parameter values from the ETD's e*Way Connection.
4. Invokes the **reset()** methods of the ETDs to clear the client to allow for another transfer (not necessary for **cr_DynBatch** but a recommended practice).

ProcessFtpETD Operation

This operation handles the FTP ETD. It also analyzes the XML order message and maps it to the FTP ETD parameters as necessary.

1. Sets the passive mode data connection.
2. Checks to see if it needs to overwrite the various entries in the FTP ETD with any corresponding part in the XML order.

3 Uses the transfer direction of the XML order as follows:

- Determines whether to map any FTP get operation (**SubscribeToExternal** in the XML order) as needed to overwrite the configured parameters. Then it takes the following steps:
  - Performs an FTP get operation for the file specified.
  - Obtains and validates the payload data obtained before transferring the payload to the XML data message.
- Determines whether to map any FTP put operation (**PublishToExternal** in the XML order) as needed to overwrite the configured parameters. Then it takes the following steps:
  - Checks the **location** attribute to validate the data format or source and loads the payload data for the FTP ETD.
  - Performs an FTP put operation for the file specified.

4 For either transfer direction, publishes the status record (XML error message) if configured, performs the **local command after transfer** command if provided, or both.

5 Invokes the nested exception handler for the **FtpFileException** exception as needed.

6 Sends messages of other error exceptions, if any, to the e*Way's log file (if the log level is set to **TRACE** for **EWAY**) and also sends the exception stack trace to the corresponding .**stderr** file.

7 Invokes the malformed command handler as needed.

ProcessLocalFileETD Operation

This operation handles the local file ETD. It also analyzes the XML order message and maps it to the local file ETD parameters as necessary.

1 Disables the Resume Reading feature.

2 Uses the transfer direction of the XML order to determine:

- Whether to map the local file get or read operation (**SubscribeToExternal** in the XML order) as needed to overwrite the configured parameters. Then it takes the following steps:
  - Performs a local file get or read operation for the file specified.
  - Obtains and validates the payload data obtained before sending it to the XML data message.
- Whether to map the local file put or write operation (**PublishToExternal** in the XML order) as needed to overwrite the configured parameters. Then it takes the following steps:
  - Checks the **location** attribute to validate the data format or source and loads the payload data for the local file ETD.
  - Performs a local file put or write operation for the file specified.

3 For either transfer direction, publishes the status record (XML error message) if configured, performs the **local command after transfer** command if provided, or both.

4 Invokes the nested exception handler for the **LocalFileException** exception as needed.

5 Sends messages of other error exceptions, if any, to the e*Way's log file (if the log level is set to **TRACE** for **EWAY**) and also sends the exception stack trace to the corresponding .**stderr** file.

6 Invokes the malformed command handler as needed.

Malformed Command Actions

The following errors cause a malformed command action:

- The configured **Transaction Type** of the FTP ETD is **XA-compliant**, which is not supported by the schema template.
- The configured **Transaction Type** of the local file ETD is **XA-compliant**, which is not supported by the schema template.
- An invalid file **transfer()** method is being used, that is the method is not **ftp** or **copy**. In these cases, an error record is published in the component's error log, and an alert message is sent to the Schema Manager. No attempt is made to get the **Action On Malformed Command** from either of the **configConDynFtp.cfg ConDynLocalFile.cfg** configuration files.
- An invalid command type was used, that is, the type was neither **send** nor **receive**.
- There was no **payload** element for a send order.
- A send order contains a **payload** element with a **location** attribute that is:
  - **invalid** (*not* **base64InSitu**, **localDir**, or an empty string)
  - **localDir** information with an invalid full-path file name
  - Unread because of a problem in reading the file content
- There is a missing entry for the command in the **local command after transfer** element when the local archive directory is not empty.
- There is an invalid archive file name corresponding to the command in the **local command after transfer** element.

Note: *If you have configured* ***Publish Error Record*** *as the malformed command action, the error message published always includes the order message but not the payload data.*

Schema Collaborations

This section shows how the Collaborations are defined in this schema template.

To view the Collaborations' properties

1 Run the e*Gate Schema Designer and go to the Main window.

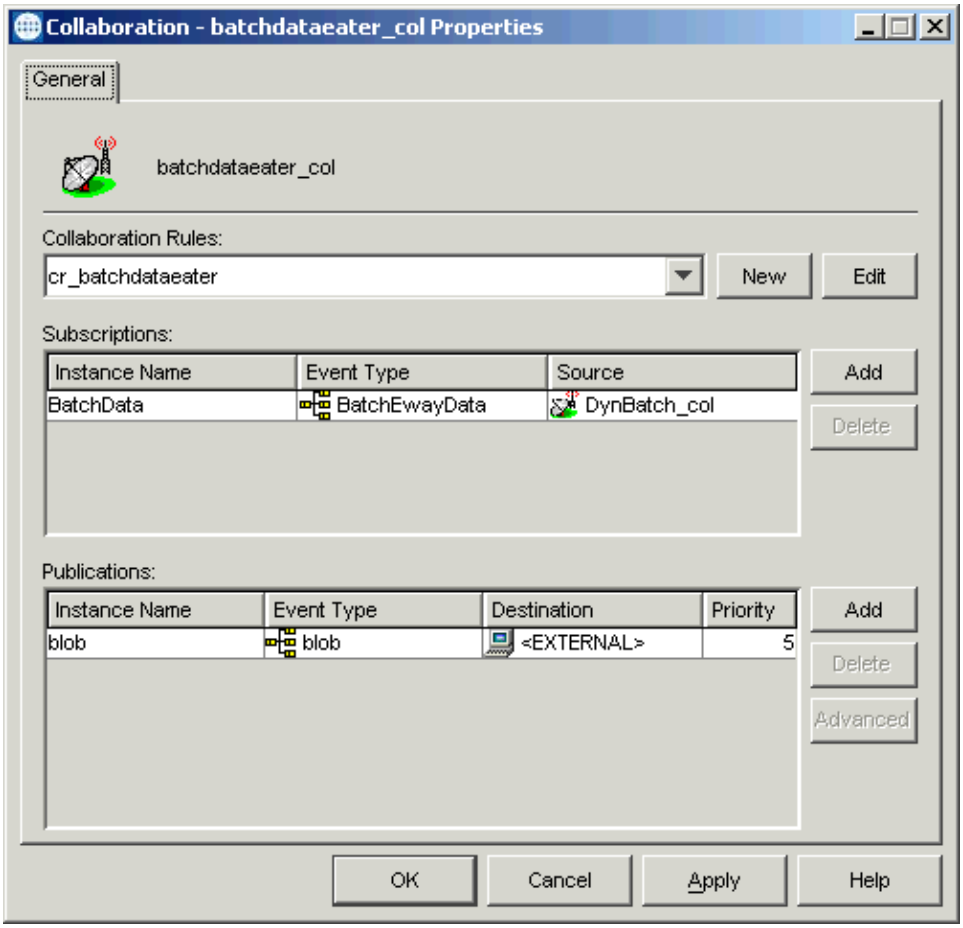
2 In the **Component** pane, open the host where the template schema resides.

3 Select the **BatchDataEater** e*Way.

4 Select the **batchdataeater_col** Collaboration in the right pane, then right-click to edit or view its properties.

5 The **Collaboration Properties** dialog box appears (see **Figure 162 on page 323**).

Figure 162 batchdataeater_col Properties Dialog Box



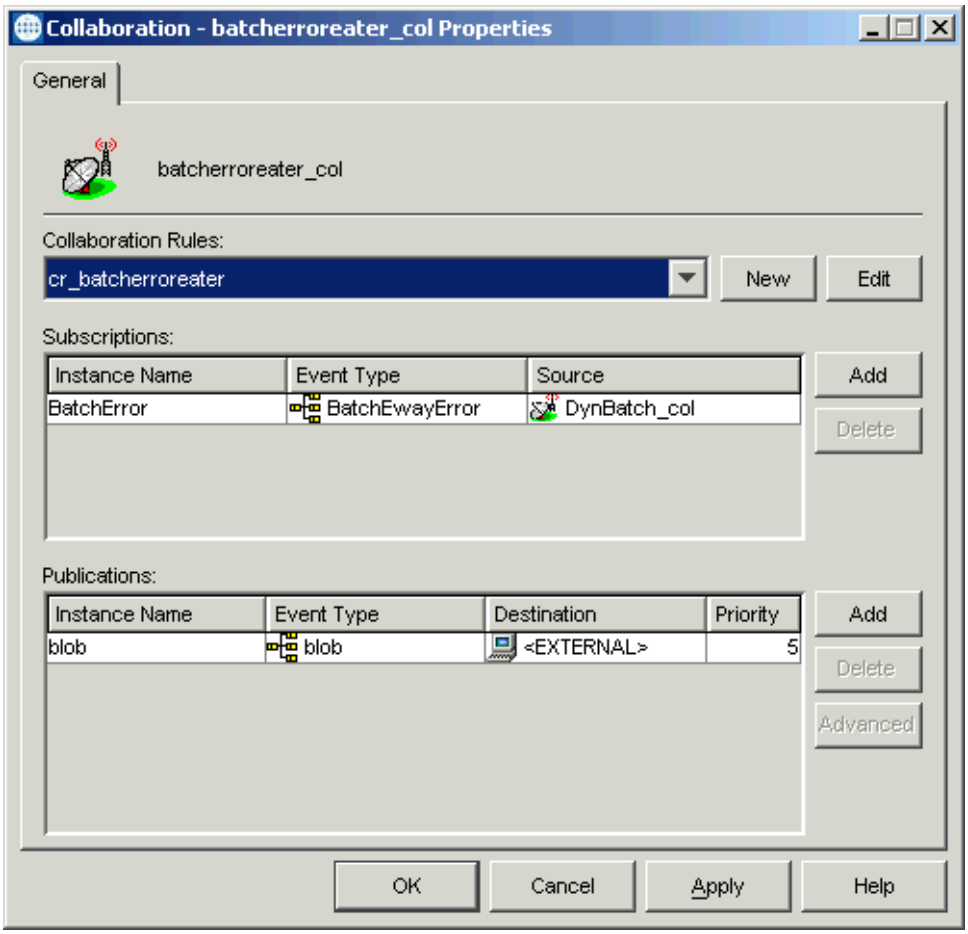
You can configure any Collaboration properties as desired or use the default properties shown in Figure 162.

6 Click **OK** to close the dialog box and save any changes.

7 Select the **BatchErrorEater** e*Way and **batcherroreater_col** Collaboration in the same way as you did the previous e*Way and Collaboration.

8 The **Collaboration Properties** dialog box appears (see **Figure 163 on page 324**).

Figure 163 batcherroreater_col Properties Dialog Box



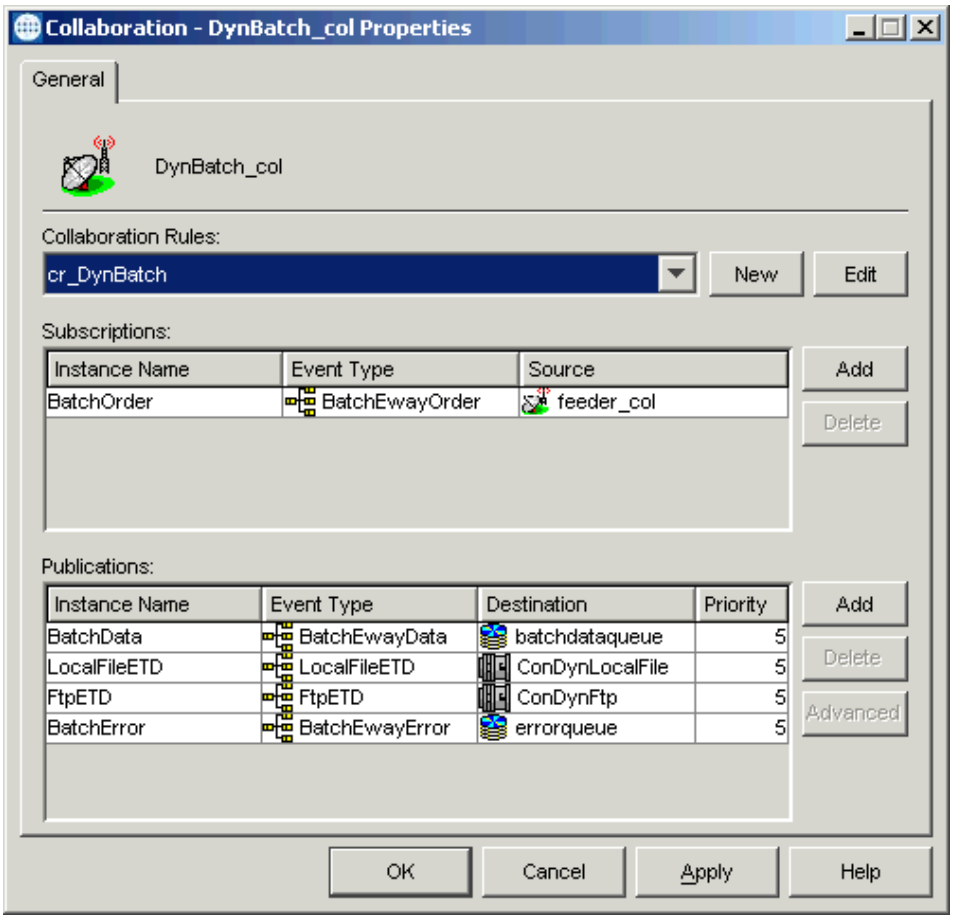
You can configure any Collaboration properties as desired or use the default properties shown in Figure 163.

9 Click **OK** to close the dialog box and save any changes.

10 Select the **DynBatch** e*Way and **DynBatch_col** Collaboration in the same way as you did the previous e*Ways and Collaborations.

11 The **Collaboration Properties** dialog box appears (see **Figure 164 on page 325**).

Figure 164 DynBatch_col Properties Dialog Box



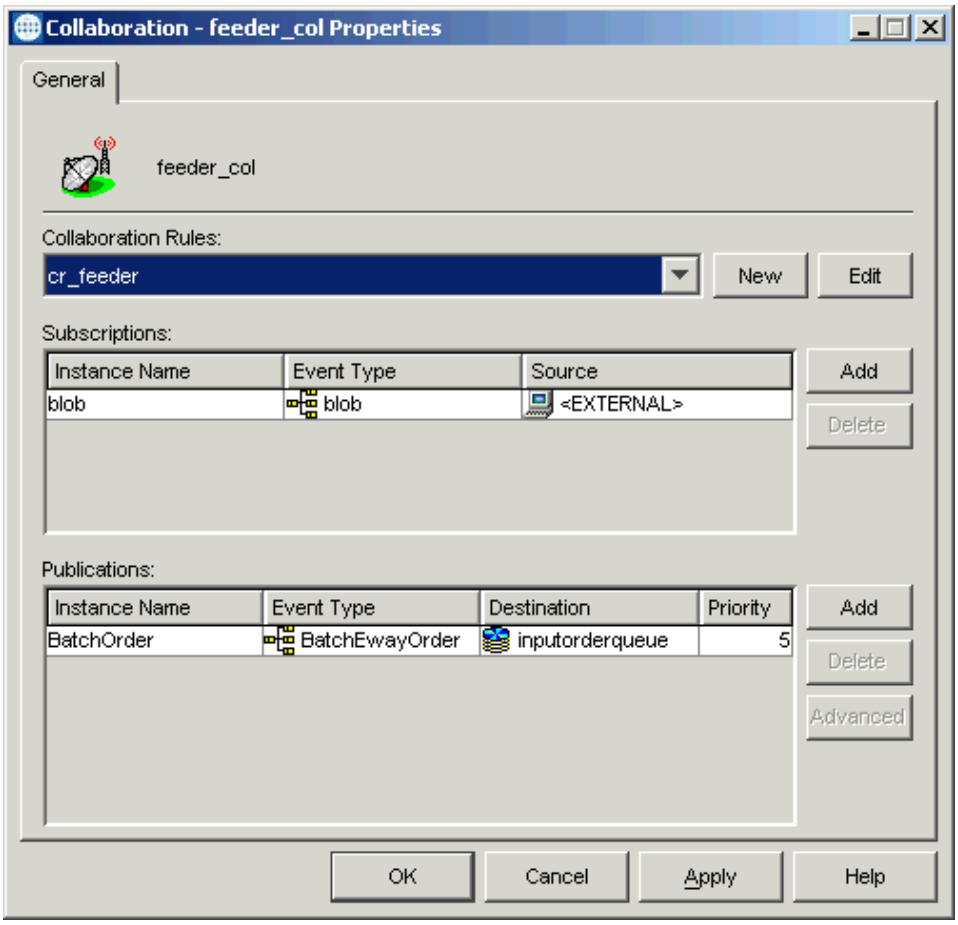
You can configure any Collaboration properties as desired or use the default properties shown in Figure 164.

12 Click **OK** to close the dialog box and save any changes.

13 Select the **feeder** e*Way and **feeder_col** Collaboration in the same way as you did the previous e*Ways and Collaborations.

14 The **Collaboration Properties** dialog box appears (see **Figure 165 on page 326**).

Figure 165 feeder_col Properties Dialog Box



You can configure any Collaboration properties as desired or use the default properties shown in Figure 165.

15 Click **OK** to close the dialog box and save any changes.

Before Running the Schema

Before you run the schema, you must create certain XML order files, as explained under **"Sending Data with a Send Order" on page 286** and **"Receiving Data with a Receive Order" on page 287**. Place these files in the configured input directory for the **feeder** e*Way.

Make sure that the name pattern of the order files conforms to that specified by the **feeder** e*Way's configuration (**feeder.cfg** file). By default, the file name must have a **.fin** extension. Also, ensure that the order files are ready to be consumed by the **feeder** e*Way's Collaboration.

Note: *If you want to easily view the input data file (data XML message) using an XML-enabled Web browser, you can rename the file to* ***.xml*** *(from* ***.~in*** *or* ***.fin****) for this purpose.*

Running the Schema

For instructions on how to run an e*Gate schema, see **"Running the Schema" on page 169**.

After Running the Schema

Observe the following schema template operations:

- The schema generates files for the XML send orders. Verify that an XML error/status record file is also generated in the appropriate output directory as specified by the **BatchErrorEater** e*Way's configuration (**batcherroreater.cfg** file).
- The schema generates an XML data file (encoded in the Base64 format) for an XML receive order. This file is generated in the appropriate output directory as specified by the **BatchDataEater** e*Way's configuration (**batchdataeater.cfg** file). Verify that an XML error/status record file is also generated in the output directory as specified by the **BatchErrorEater** e*Way's configuration (**batcherroreater.cfg** file).

Note: *The exact output directory location depends on the settings for the Dynamic Configuration, as shown in* **Figure 150 on page 307** *and* **Figure 157 on page 314**.

When you are adapting this template schema to your own needs, keep the following schema operations in mind:

- The input XML order files become XML ETDs that need to be generated by the Business Rules of other Collaborations via IQs.
- The output XML files also become XML ETDs and are published to other Collaborations via IQs.

Chapter 9

Additional Features

This chapter explains the following additional features of the Batch e*Way Intelligent Adapter:

- **Data streaming**: The e*Way's ability to stream data between Event Type Definition (ETD) components.
- **Secure FTP:** The e*Way supports SOCKS and SSH tunneling to provide secure FTP data transmission.
- **Guaranteed Exactly Once Delivery (GEOD) of Events**: The XA-related features of the e*Way.

9.1 Streaming Data Between Components

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

9.1.1 Introduction to Data Streaming

Data streaming provides a means for interconnecting any two components of the e*Way by means of a *data stream channel*. This channel provides an alternate way of transferring the data between the Batch e*Way components.

Each ETD component in the e*Way 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 ETD, 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 e*Gate's memory, you can use a *data-stream channel* to provide for streaming the data between them a little at a time, outside of e*Gate.

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

You will use the e*Gate Schema 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.*

9.1.2 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 e*Gate system's memory. Using streaming is not as fast as using in-memory operations, but it is far less resource-intensive.

9.1.3 Using Data Streaming

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

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

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

Data-streaming Operation

Each of the ETDs in the Batch e*Way exposes stream adapter nodes, allowing any ETD 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 e*Gate Schema Designer's Collaboration Rules Editor (see **"Data Streaming Setups" on page 332** for details).

Figure 166 on page 331 shows an example of the local file ETD, as shown in the e*Gate ETD Editor's Main window.

Figure 166 Local File ETD With Data-streaming Nodes

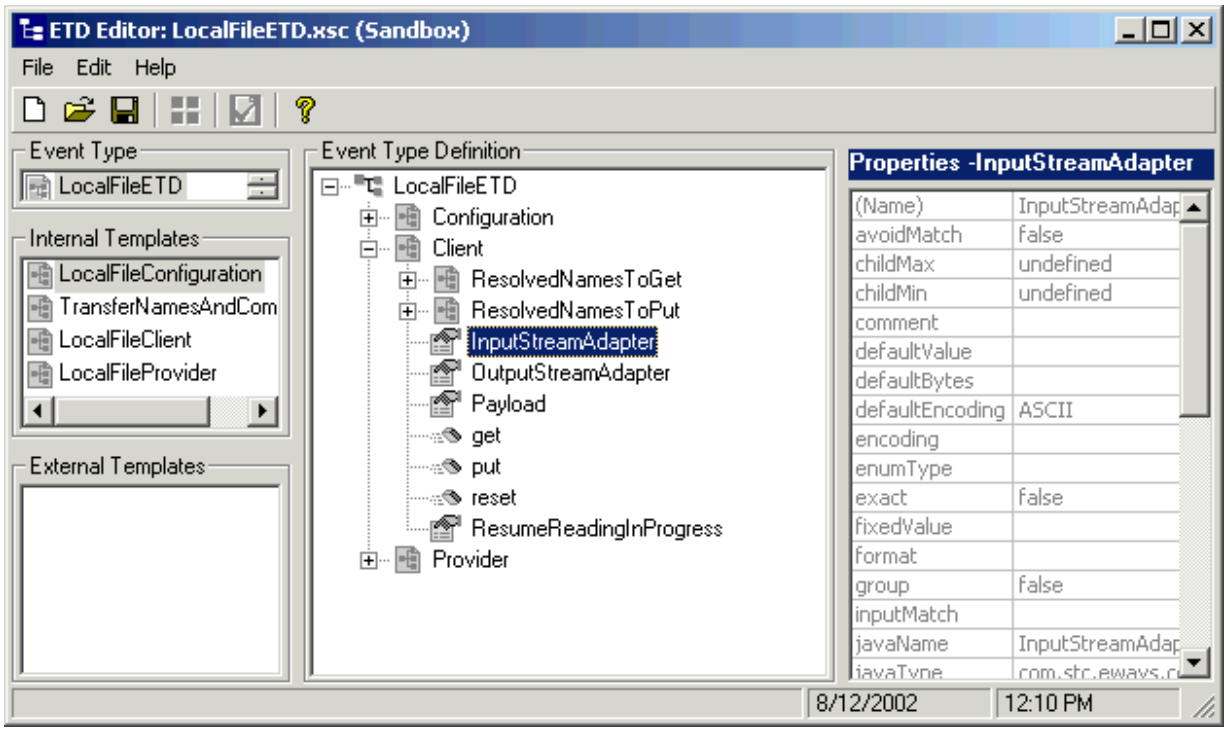


Figure 166 shows the **InputStreamAdapter** (highlighted) and **OutputStreamAdapter** nodes in the ETD, which you use for data streaming. Basically, this feature operates as follows:

- **Stream-adapter consumers**: The FTP and the record-processing ETDs 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 ETD can only *provide* stream adapters, so its stream-adapter nodes are *read-only*. Its node value can only be *retrieved*.

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

Note: *For an explanation of the e*Way's different types of ETDs, see* **Chapter 5**.

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 ETDs. For example, the FTP ETD'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 ETD'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 e*Way 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 ETD)
- From a record-processing setup to a local file system (uses **OutputStreamAdapter** node in ETD)
- From a local file system to a remote FTP server (uses **InputStreamAdapter** node in ETD)
- From a remote FTP server to a local file system (uses **OutputStreamAdapter** node in ETD)

Using the Collaboration Rules Editor

Use the Collaboration Rules Editor to set up data streaming between ETDs. Because the local file ETD is the stream provider and the FTP and record-processing ETD are consumers, when you use the Collaboration Rules Editor, you always drag *from* the input-output ETD to either of the other ETDs.

Note: *For details on how to use the Collaboration Rules Editor, see the* ***e*Gate Integrator User's Guide****.*

This section explains how to create a simple version of each setup in the e*Gate Collaboration Rules Editor and the basics of how each operates. These are general procedures applicable to all of the types of data streaming in the previous list. Which exact procedures you will follow before and after the steps listed in this section are dependent on your specific implementation.

Note: *All the streaming-related interactions are handled by the ETD implementations. See* **Chapter 7** *for complete sample implementations of data streaming.*

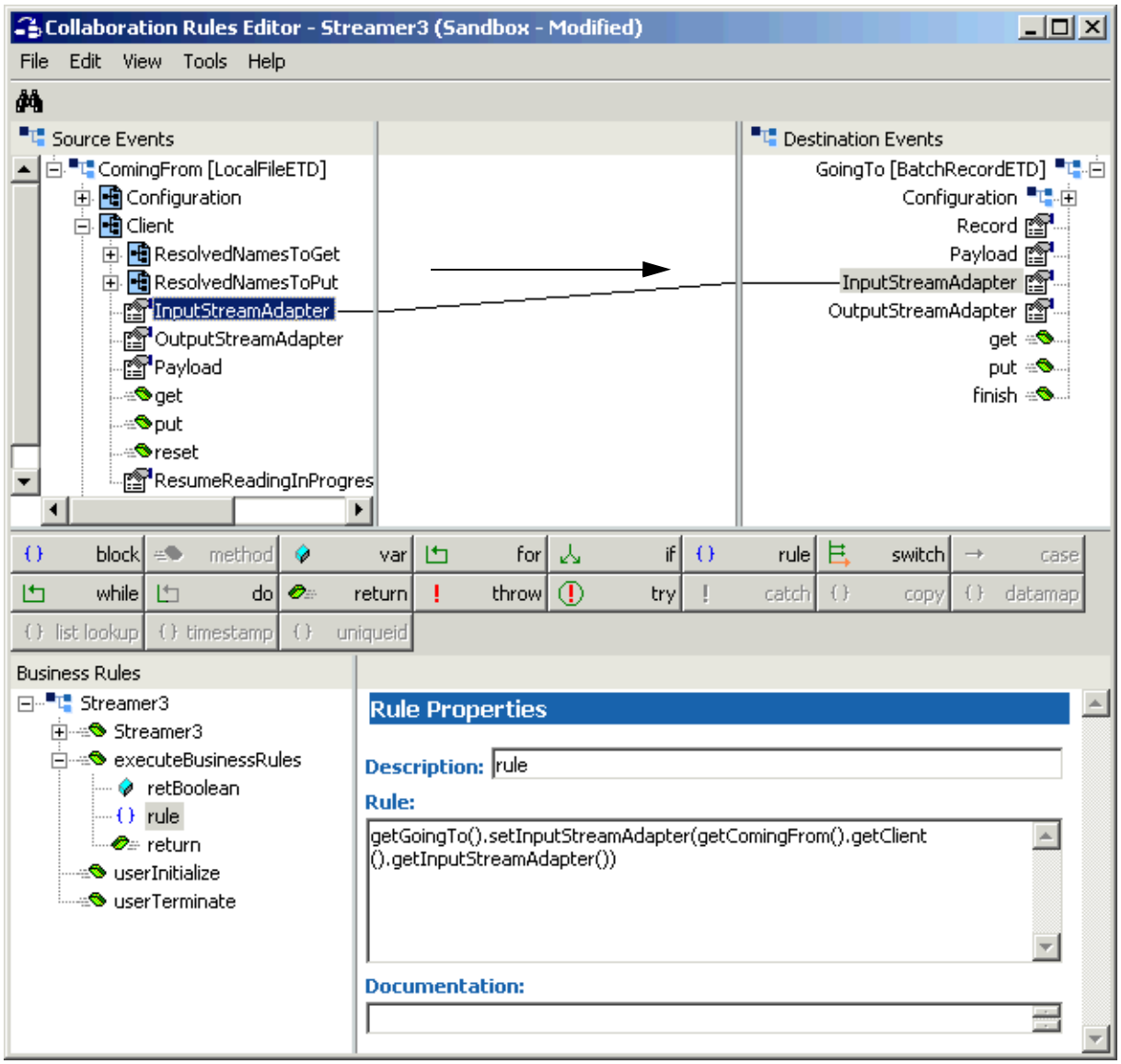
Local File System to Record-processing Setup

Set up this operation as follows:

1. Using the e*Gate Collaboration Rules Editor's Main window, set up the desired local file ETD as the source and a record-processing ETD as the destination.

2 Drag the **InputStreamAdapter** node from the local file ETD to the **InputStreamAdapter** node on the record-processing ETD (see **Figure 167 on page 333**).

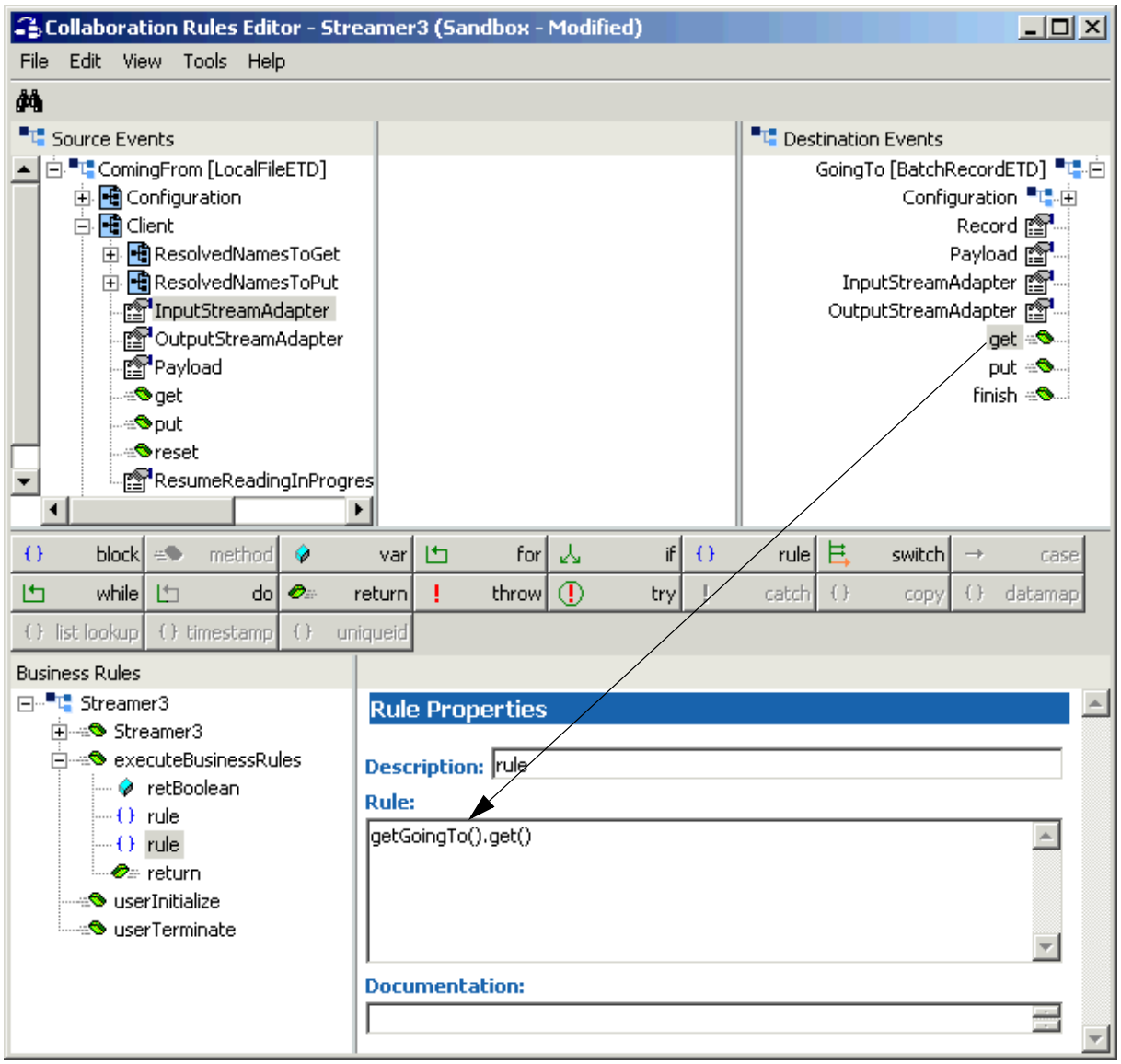
Figure 167 Collaboration Rules Editor Local File to Record Processing: Step 2



3 Create another rule. In this rule, drag the **get()** method from the **Destination Event** to the **Rule** scroll box in the **Rule Properties** window.

This action invokes the **get()** method on the record-processing ETD, which consumes the stream adapter, to perform the transfer (see **Figure 168 on page 334**).

Figure 168 Collaboration Rules Editor Local File to Record Processing: Step 3

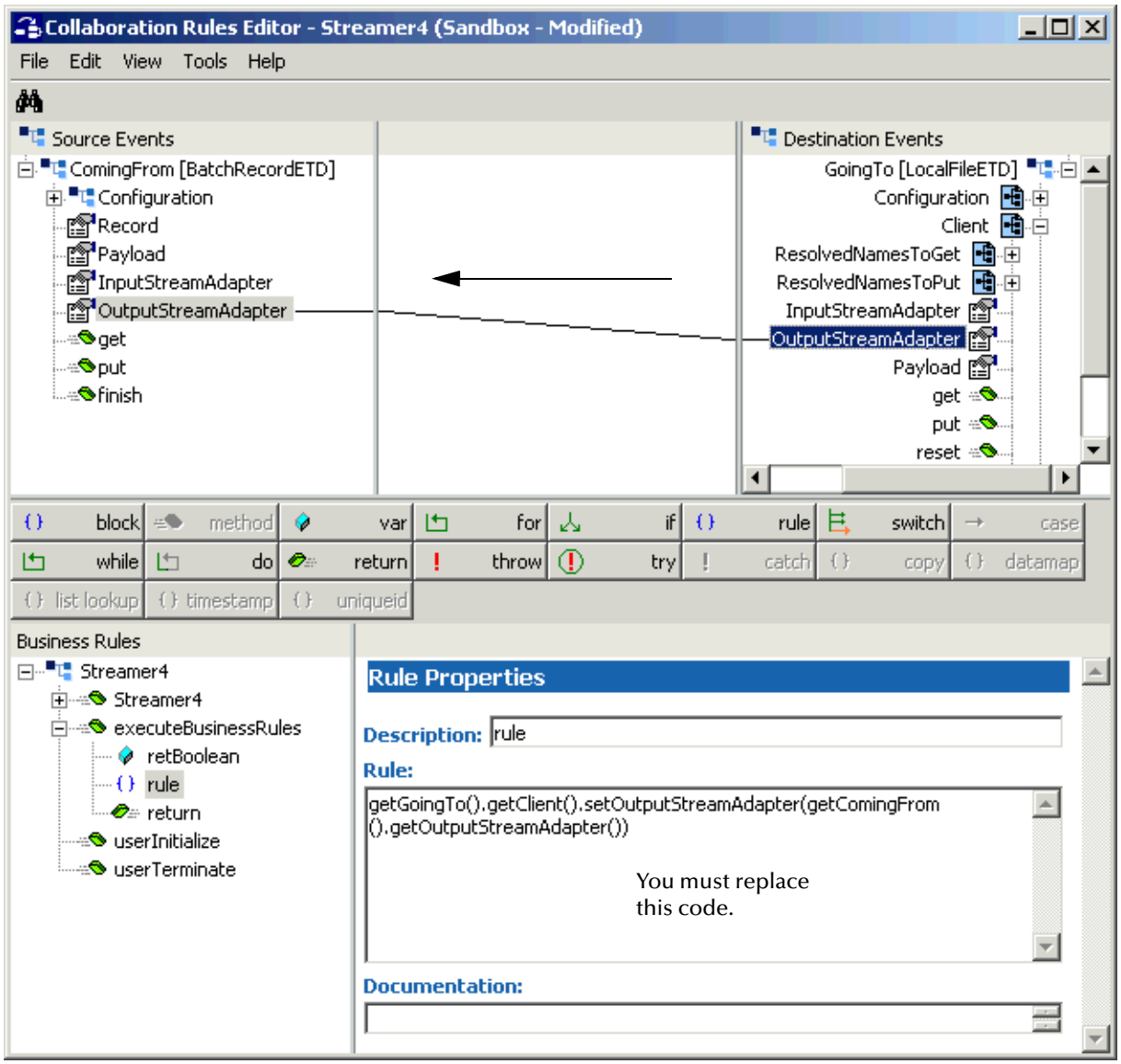


Record-processing Setup to Local File System

Set up this operation as follows:

1. Using the e*Gate Collaboration Rules Editor's Main window, set up the desired record-processing ETD file as the source and a local file ETD as the destination.
2. Drag the **OutputStreamAdapter** node from local file ETD to the **OutputStreamAdapter** node on the record-processing ETD (see **Figure 169 on page 335**).

Figure 169 Collaboration Rules Editor Record Processing to Local File: Step 2

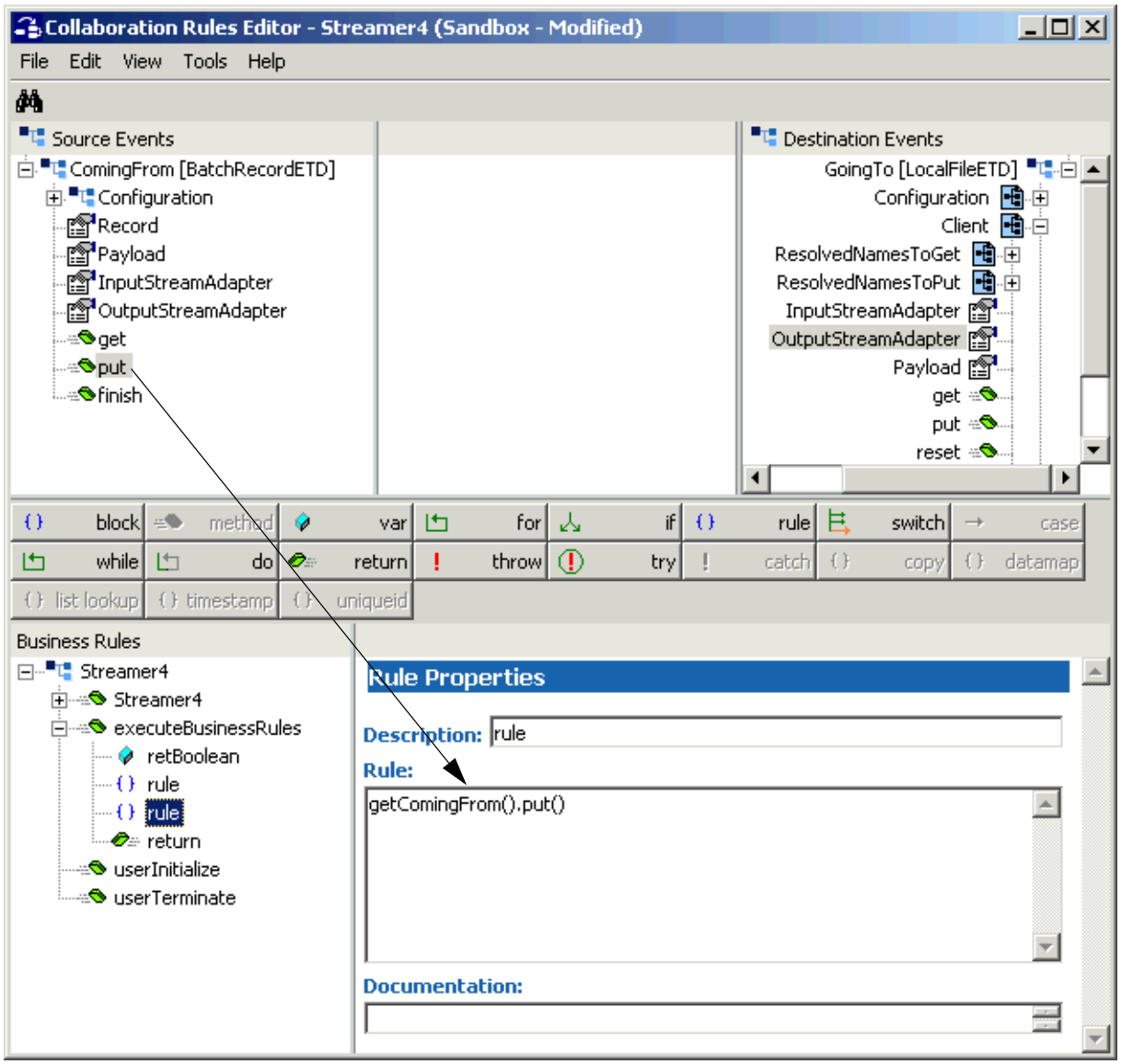


3 To complete this implementation, you must delete the code listed in the **Rule** scroll box in the **Rule Properties** window and enter the following code:

```
getComingFrom().setOutputStreamAdapter(getGoingTo()
    .getClient().getOutputStreamAdapter())
```

4 Create another rule. In this rule, invoke the **put()** method on the record-processing ETD, which consumes the stream adapter, to perform the transfer (see **Figure 170 on page 336**).

Figure 170 Collaboration Rules Editor Record Processing to Local File: Step 4



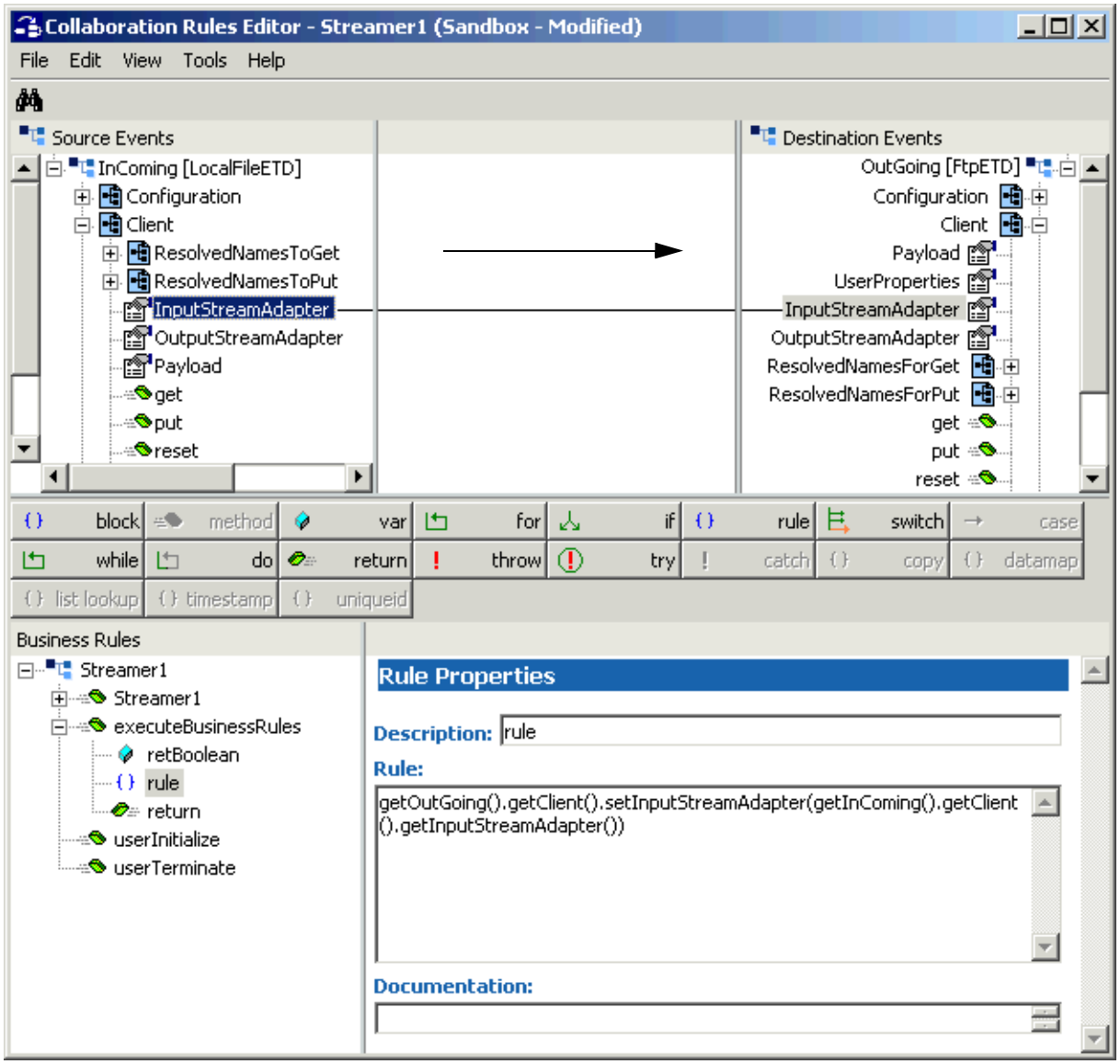
Note: *You may get the* ***FileNotFoundException*** *after a* ***put()*** *transfer, when you are using data streaming, and the local file ETD runs out of files.*

Local File System to a Remote FTP Server

Set up this operation as follows:

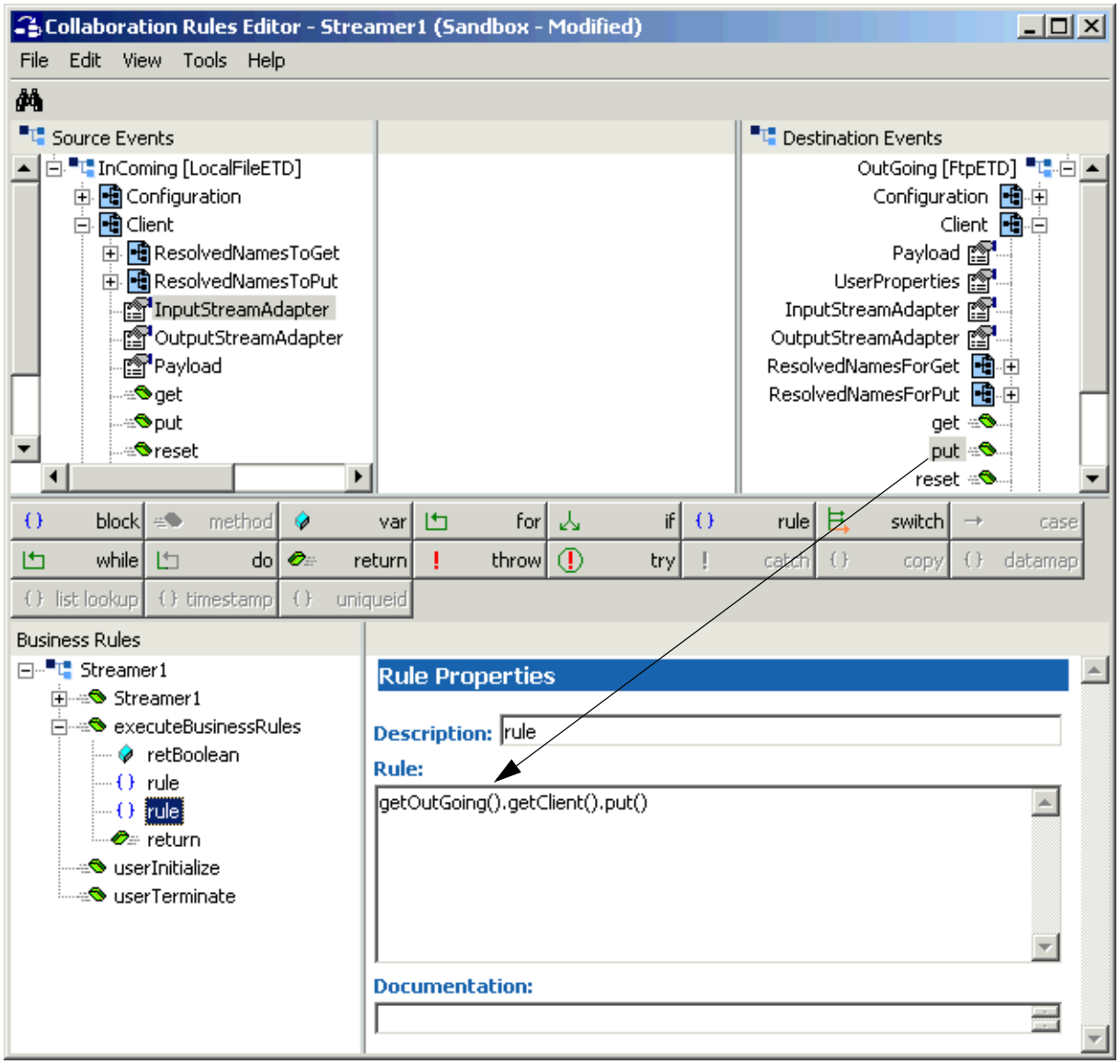
1. Using the e*Gate Collaboration Rules Editor's Main window, set up the desired local file ETD as the source and an FTP ETD as the destination.
2. Drag the **InputStreamAdapter** node from the local file ETD to the **InputStreamAdapter** node on the FTP ETD (see **Figure 171 on page 337**).

Figure 171 Collaboration Rules Editor Local File to FTP: Step 2



3 Create another rule. In this rule, invoke the **put()** method on the FTP ETD, which consumes the stream adapter, to perform the transfer (see **Figure 172 on page 338**).

Figure 172 Collaboration Rules Editor Local File to FTP: Step 3



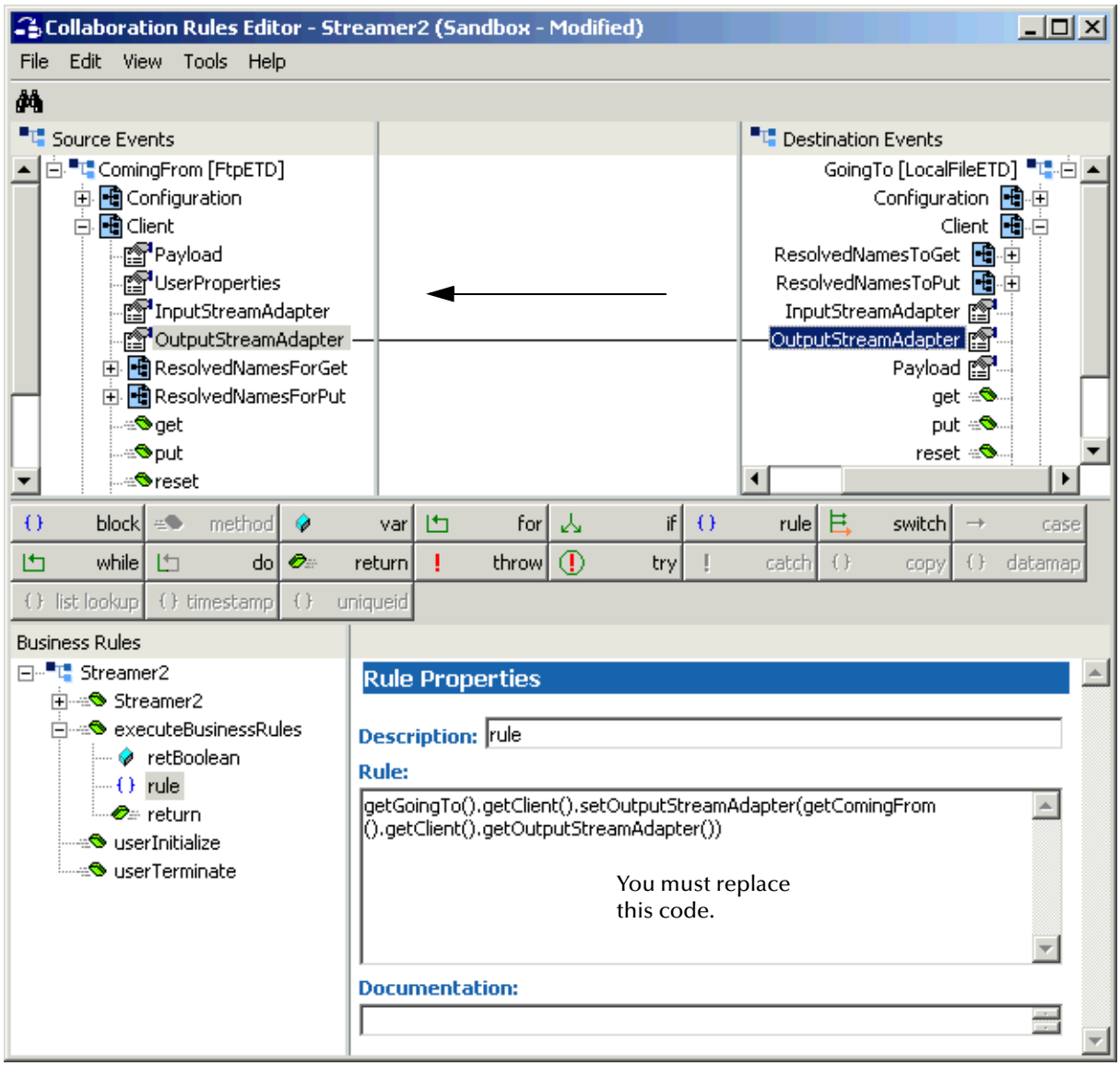
Remote FTP Server to a Local File System

Set up this operation as follows:

1 Using the e*Gate Collaboration Rules Editor's Main window, set up the desired FTP ETD as the source and a local file ETD as the destination.

2 Drag the **OutputStreamAdapter** node from the local file ETD to the **OutputStreamAdapter** node on the FTP ETD (see Figure 173).

Figure 173 Collaboration Rules Editor FTP to Local File: Step 2

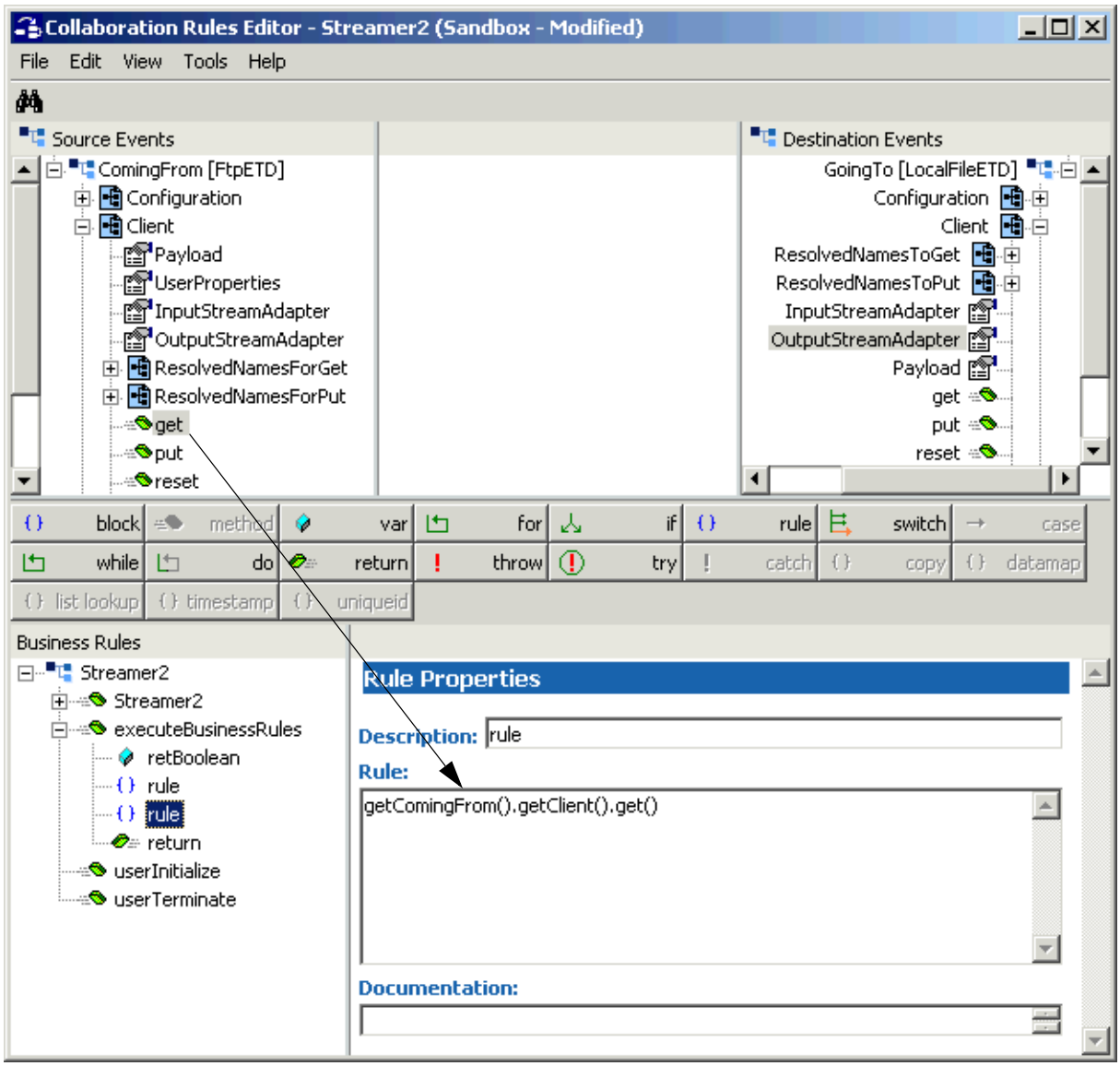


3 To complete this implementation, you must delete the code listed in the **Rule** scroll box in the **Rule Properties** window and enter the following code:

```
getComingFrom().getClient().setOutputStreamAdapter(getGoingTo()
    .getClient().getOutputStreamAdapter())
```

4 Create another rule. In this rule, invoke the **get()** method on the FTP ETD, which consumes the stream adapter, to perform the transfer (see **Figure 174 on page 340**).

Figure 174 Collaboration Rules Editor FTP to Local File: Step 4



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 ETDs support post-transfer commands. The **Success** parameter indicates whether these commands are executed. If the ETD providing the stream adapter is operating in the Guaranteed Exactly Once Delivery (GEOD or XA) mode, it uses the **Success** configuration parameter to determine whether the transaction must be rolled back.

Note: *See* **"Guaranteed Exactly Once Delivery" on page 347** *for details on how to use the e*Way's GEOD feature.*

- Do not close the stream.

9.1.4 Stream-adapter Interfaces

This section provides the Batch e*Way's ETD stream-adapter Java interfaces. This information is only for advanced users familiar with Java programming, who want to provide custom ETD 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;
}
```

9.2 Secure FTP and the e*Way

This section explains the secure FTP features available with the Batch e*Way Intelligent Adapter. The Batch e*Way 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 e*Way uses them.

9.2.1 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 e*Way 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 e*Way supports both SOCKSv4 and SOCKSv5. To enable SOCKS support, the following must be specified in the e*Way 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" on page 48**.

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 e*Way

The e*Way can use SOCKS to provide for secure FTP during data transmission.

Negotiation Methods

The e*Way supports the following negotiation methods:

- No-authentication
- User/password

SOCKS Configuration Parameters

This section lists the SOCKS parameters you must set to configure the e*Way Connection. For more information, see **"SOCKS Configuration" on page 48**.

Socks Enabled

Allows you to specify whether the FTP command connection goes through a SOCKS server.

If you choose **No**, the e*Way 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.

9.2.2 SSH Tunneling

This section explains the Batch e*Way'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 e*Way 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 e*Way makes use of an additional software application for the SSH tunneling. The e*Way 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 e*Way

The e*Way can use SSH tunneling to provide for secure logon IDs and passwords. The e*Way 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 e*Way Connection configuration (see **"SSH Tunneling Configuration" on page 50**). 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 e*Way's use

Using an Existing Channel

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

Using an Internal Channel

If you choose **No**, under the **SSH Channel Established** parameter, the e*Way launches a process within e*Gate 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 e*Way. 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 e*Way Connection.

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

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

Under the e*Way'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 e*Way 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 e*Way Connection. For more information, see **"SSH Tunneling Configuration" on page 50**.

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 e*Way 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 e*Way 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**.

9.3 Guaranteed Exactly Once Delivery

Occasionally, a failure condition can occur in data transfer systems. Data can be lost through system errors. To compound these problems, identical data is often delivered more than once after system restarts.

The Batch e*Way allows you to guarantee that each unit of data is delivered exactly. In e*Gate, this feature is called Guaranteed Exactly Once Delivery (GEOD) of Events. Along with the e*Gate system, the e*Way guarantees exactly once delivery via utilization of the XA protocol, from the X/Open Consortium.

9.3.1 XA Compliance

If cooperating software systems are XA-compliant, they guarantee that for each unit of data transferred between systems:

- No data is lost.
- No unit of data is duplicated.

This vendor-neutral XA protocol was devised to manage transactions between multiple-client application programs and multiple database systems. Data transactions occur as a part of the following process:

- **Prepare**: Transactions are prepared to commit.
- **Commit** or **Rollback**: Transactions are either fully committed or rolled back.

You can enable GEOD in the e*Way by setting the XA-related configuration parameter in the e*Way Connection to **Yes**. This action causes all data transmission through the e*Way Connection to be performed in the e*Way's *XA mode*. See **"Working With GEOD Collaborations" on page 348** for details on how to set this parameter.

Note: *The e*Way cannot perform multiple file transfers within the same Collaboration Rule, while in the XA mode. Also, this mode is not supported with the e*Way's Dynamic Configuration feature. See* **Chapter 8** *for details on this feature.*

9.3.2 Rollback and Commit

In the Batch e*Way, GEOD utilizes this same two-phase commit process. Under normal circumstances, once a unit of data is prepared, it becomes committed. However, in the event of a failure, e*Way data that has been prepared but not committed is rolled back to ensure the non-duplication of data.

GEOD data management features keep track of prepared/uncommitted versus committed data units, as well as managing data recovery. Therefore, when normal system operations return, the e*Way can resume data transactions where they left off. Keep mind that the commit phase completes after the full execution of any Business Rule.

Note: *The e*Way does not support the XA mode with sub-Collaborations.*

9.3.3 Working With GEOD Collaborations

For a Collaboration to be fully GEOD-enabled, all of its sources and destinations must be e*Way Connections that support XA, and XA must be activated in all of its e*Way Connections. If only a subset of the e*Way Connections are XA, then only those e*Way Connections are guaranteed to participate in the two-phase commit process.

Important: *You must make sure there is no manual or other activity in respect to the files transmitted in the XA mode. The transmission **must** be isolated from any outside activity.*

This feature is available with the following ETDs:

- FTP
- Local file

Note: *No concurrent access to a file is allowed (also true for non-XA mode).*

Restrictions

The use of the XA mode with the e*Way has the following restrictions:

FTP ETD Only

- The FTP append operation cannot be used.
- FTP raw commands cannot be used.

FTP and Local File ETDs

- Only one **get()** or **put()** call per ETD can be made during the execution of a Collaboration Rule.
- Direct calls to **Provider** ETD nodes do not participate in any XA-mode transactions. That is, only the **get()** and **put()** methods' operation participates in the XA mode. None of the other methods under the **Provider** node take part in XA transactions.
- All existing destination files are overwritten and not restored during any XA-mode transaction rollback (also valid in non-XA mode). If you set up the ETD to write files to a particular destination, any file there with the same name is overwritten during the e*Way's operation. No backup copy is maintained in either the XA or non-XA mode.

Caution: *If the e*Way stops or does not start, and generates the error message* ***Incorrect State****, you must delete certain state files. In these cases, you can go to the* ***eGate\client\LocalFileETD\<Collaboration UID>*** *or* ***eGate\client\FtpETD\<Collaboration UID>*** *directory and delete all files named in the following format:*
CollaborationName_ETDInstanceName.Extension
Where ***CollaborationName*** *is the name of the Collaboration that generated the error,* ***ETDInstanceName*** *is the name of the current ETD instance, and* ***Extension*** *is the file extension. The files reside in subdirectories named after the Collaboration's UID.*

Behavior With get()

When you are getting data, using the FTP or local file ETD, the XA commit phase is completed after the current file is completely transmitted to e*Gate, and post-transfer commands are performed.

In case of a mid-transfer problem, the e*Way completes rollback as follows:

- The incomplete file is deleted from e*Gate's memory; this action also rolls back any pre-transfer commands.
- When operation resumes, the current file transaction starts over again from the beginning; any pre-transfer commands are repeated.
- When the transaction is done, any post-transfer commands are executed (commit phase complete).

Behavior With put()

When you are sending data, using the FTP or local file ETD, the XA commit phase is completed after the data is transmitted to the remote FTP or local file system, and post-transfer commands are performed.

In case of a problem in mid transfer, the e*Way completes rollback as follows:

- The incomplete file is deleted from the remote or local location; this action also rolls back any pre-transfer commands.
- When operation resumes, the current file transaction starts over again from the beginning; any pre-transfer commands are repeated.
- When the transaction is done, any post-transfer commands are executed (commit phase complete).

Note: *You must use JMS e*Way Connections because these components can be XA-enabled. JMS IQs are not XA-compliant and do not provide XA-mode operation in a GEOD schema.*

9.3.4 Enabling the XA Mode

This section provides general procedures on how to enable the XA mode. For more information on the e*Way's configuration parameters and how to set them, see **Chapter 4**.

For specific information on configuring the e*Way Connection parameters, see:

FTP ETD

- **"Transaction Type" on page 36**

Local File ETD

- **"Transaction Type" on page 60**

To enable the XA Mode in an e*Way Connection

1. In the e*Gate Schema Designer, with the **Components** tab active, open the **e*Way Connections** folder and, in the Editor pane (on the right), double-click the e*Way Connection you want to edit.

   A **Properties** dialog box for the e*Way Connection opens, displaying the **General** tab.

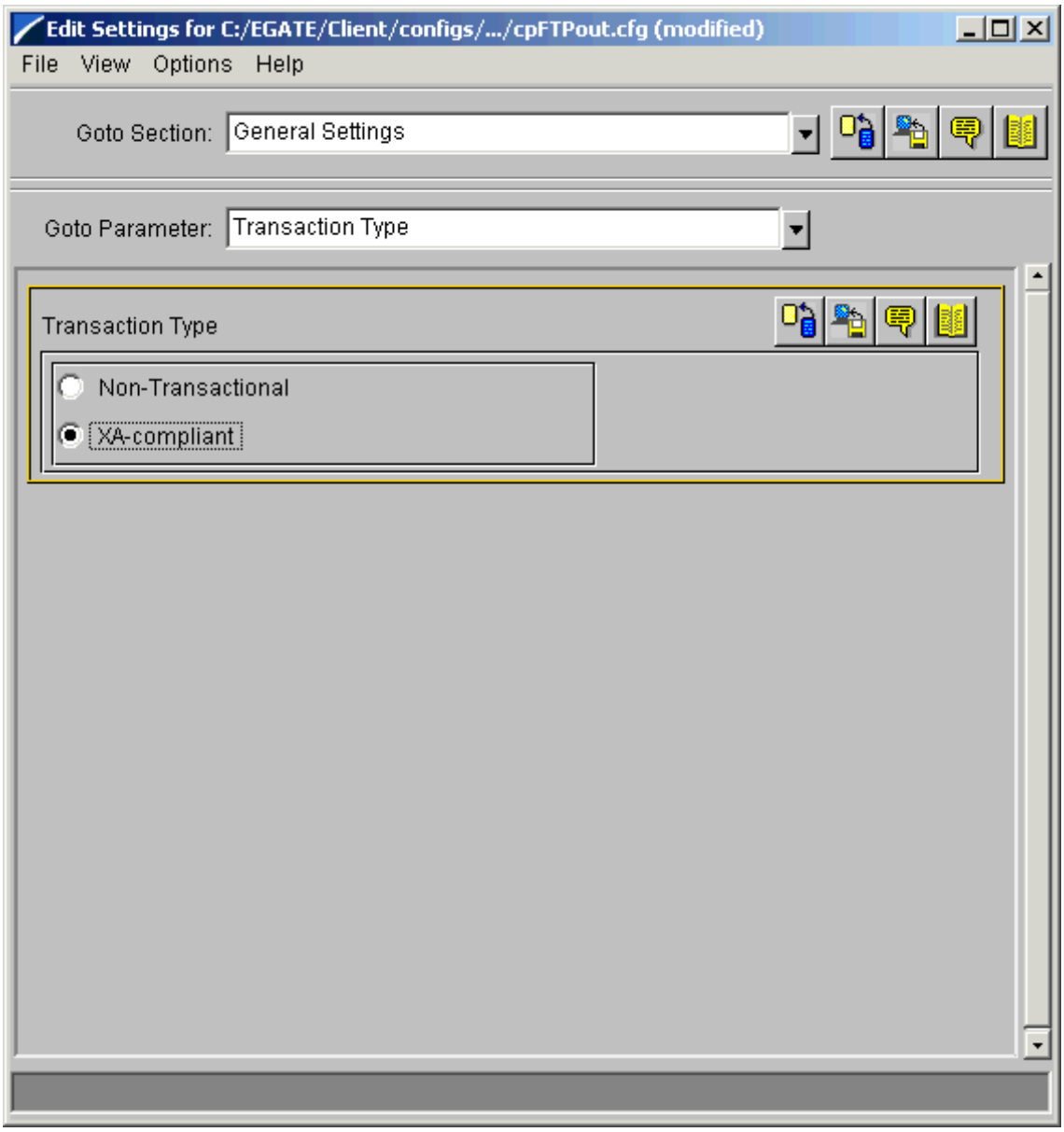
2. In the **e*Way Connection Configuration File** area, click **Edit**.

   The e*Way Configuration Editor window opens. Initially, the **General Settings** section is active.

3. In the **Transaction Type** area, select **XA-compliant** (see **Figure 175 on page 351**).

Note: *Using pre-and post-transfer configuration parameter commands allows you to move a file to a temporary location during an XA-mode transfer. Then, after the transfer, you can take subsequent action from there or from the new location.*

Figure 175 e*Way Configuration Editor: Enabling GEOD



4 After you finish setting parameters for the e*Way Connection, close the editor.

5 In the **Properties** dialog box for the e*Way Connection, click **OK**.

When all sources and destinations for a Collaboration are XA-compliant e*Way Connections, the e*Way and e*Gate take care of the rest of the GEOD process, such as logging, exchanging **prepare, commit** and **rollback** calls, and managing data recovery.

Note: *For more information on e*Gate's GEOD feature and how to use it, see the* ***e*Gate Integrator User's Guide****.*

Chapter 10

e*Way Java Methods

This chapter provides an overview of the Java classes and methods contained in the Batch e*Way Intelligent Adapter. These methods are used to extend the functionality of the e*Way.

10.1 Batch e*Way Methods and Classes: Overview

The Batch e*Way has been enabled by the Java programming language. Java methods have been added to make it easier to set information in the Batch e*Way Event Type Definitions (ETDs), as well as get information from them.

The nature of this data transfer depends on the configuration parameters (see **Chapter 4**) you set for the e*Way in the e*Gate Schema Designer's e*Way Configuration Editor window. These Java methods are organized into related groups or classes.

Note: *For more information on the Batch e*Way ETD structures, their nodes, and attributes (within an e*Gate* ***.xsc*** *file), see* **Chapter 5**.

For the e*Way, the **stceway.exe** file (this file creates a Java-based Multi-Mode e*Way; see **Chapter 3**) is used to communicate between the e*Way and other e*Gate components. A Java Collaboration is utilized to keep the communication open between the e*Way and the external system or network.

10.2 Java Classes

The methods for this e*Way are organized into the following Java classes:

- **BatchException**
- **BatchRecordConfiguration**
- **BatchRecordETD**
- **BatchRecordParser**
- **FtpETD**
- **FtpFileClient**

- **FtpFileConfiguration**
- **FtpFileException**
- **FtpFileProvider**
- **FtpHeuristics**
- **InputStreamAdapter**
- **LocalFileClient**
- **LocalFileConfiguration**
- **LocalFileETD**
- **LocalFileException**
- **NestedException**
- **OutputStreamAdapter**
- **StreamingException**

10.3 Using Java Methods

The Schema Designer's Collaboration Rules Editor window allows you to call Java methods by dragging and dropping an ETD node into the **Rules** scroll box of the **Rules Properties** window.

Note: *The node name can be different from the Java method name.*

After you drag and drop, the actual conversion takes place in the **.xsc** file. To view the **.xsc** file, use the Schema Designer's ETD Editor or Collaboration Rules Editor windows.

For example, if the node name is **TargetFileName**, the associated **javaName** is **TargetFileName**. If you want to get the node value, use the Java method called **getTargetFileName()**. If you want to set the node value, use the Java method called **setTargetFileName()**.

For a complete list of Java methods with an explanation of each, refer to the **Javadoc** at the following location:

Javadocs\Batch_eWay\index.html

Note: *This path is relative to the **eGate\client\docs** installation directory.*

Index

A

B

C

D

E

F

G

H

I

J

L

M

O

P

R

S

T

U

X