SeeBeyond ICAN Suite

e*Way Intelligent Adapter for Teradata User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

Version 20050405202350.

Contents

Chapter 4

Chapter 5

Chapter 7

Chapter 1

Introduction

This chapter provides a brief introduction to the SeeBeyond e*Way Intelligent Adapter for Teradata™. It includes:

- A general overview of the e*Way's functionality.
- A description of the Teradata e*Way Adapter implementation options.
  - TPump™ implementation, which uses Teradata's Parallel Data Pump (TPump) utility to populate a database.
  - FastExport implementation which uses the Teradata FastExport™ utility to export data from a database.
  - JDBC implementation, which inserts, updates, selects or retrieves data in Teradata RDBMS via the JDBC driver and exports data.
- Information regarding error handling.
- Supported operating systems and system requirements.

1.1 Intended Reader

The reader of this document is presumed:

- To be a developer or system administrator with the responsibility of maintaining the e*Gate system.
- To have high-level knowledge of system operations and administration.
- To be thoroughly familiar with Windows and UNIX operations.
- To have high-level knowledge of the operation of the Teradata Relational Database Management System (RDBMS).

1.2 Overview

The Teradata relational database management system (RDBMS) is Teradata's data warehousing and customer relationship management software. NCR Corporation's Teradata division is a leader in business-intelligence applications designed to help companies fully apply data to get fast answers and make better decisions.

Teradata's data warehouse and parallel processing technology allow companies to store tremendous amounts (terabytes) of detailed operational data, and to access specific data quickly in order to answer complex questions. An array of tools and utilities are provided to facilitate the operation, administration and maintenance of the Teradata system.

1.2.1 e*Way Intelligent Adapter for Teradata

The e*Way Intelligent Adapter for Teradata provides a communication interface between e*Gate Integrator and the Teradata RDBMS. The result is an integrated adapter that provides high-volume bulk loading and ad hoc loading of the Teradata database.

The e*Way allows the user to create and run scripts to populate a database using the Teradata TPump utility (see **Database Population Through TPump** on page 9) and to export specified data from a database to an external file using the Teradata FastExport utility (see **Exporting Data Through FastExport** on page 10).

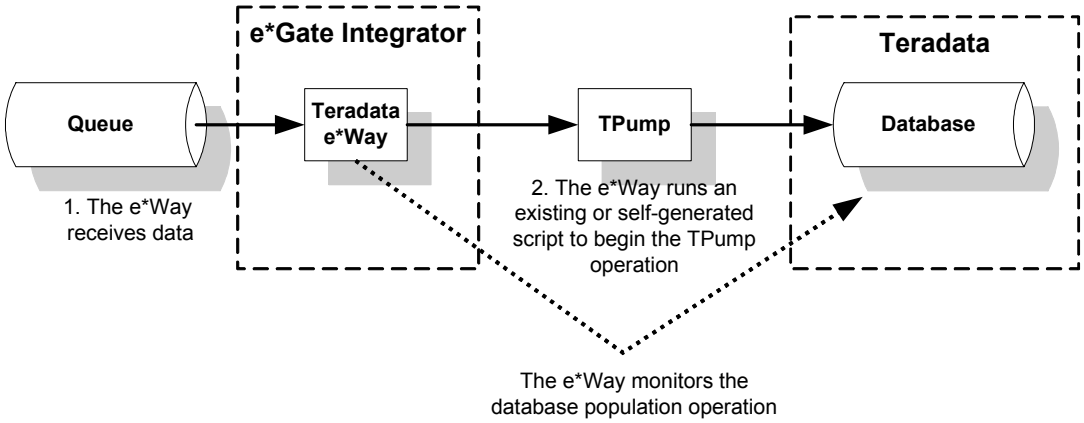
The e*Way monitors the status of these operations and alerts the Schema Manager if for any reason it is unable to start a job. If an error occurs once a job has started, the error information is available through the e*Way's ETD and can be written to a log file.

The Teradata e*Way adapter can also select and insert data from the Teradata RDBMS through the Teradata JDBC driver (see **Data Retrieval via the Teradata (JDBC) e*Way** on page 11).

1.2.2 Database Population Through TPump

The e*Way Intelligent Adapter for Teradata, configured to work with the Teradata TPump utility, receives data from an external source, and initializes and monitors the TPump operation that populates a database with this data. This is illustrated in Figure 1.

Figure 1 Database Population Through TPump



Database population occurs as follows:

1. The e*Way Intelligent Adapter for Teradata receives data from an external data source, or is triggered by a scheduler.
2. The e*Way runs a script to begin the TPump database population operation. The script can already exist, or the e*Way can generate it.

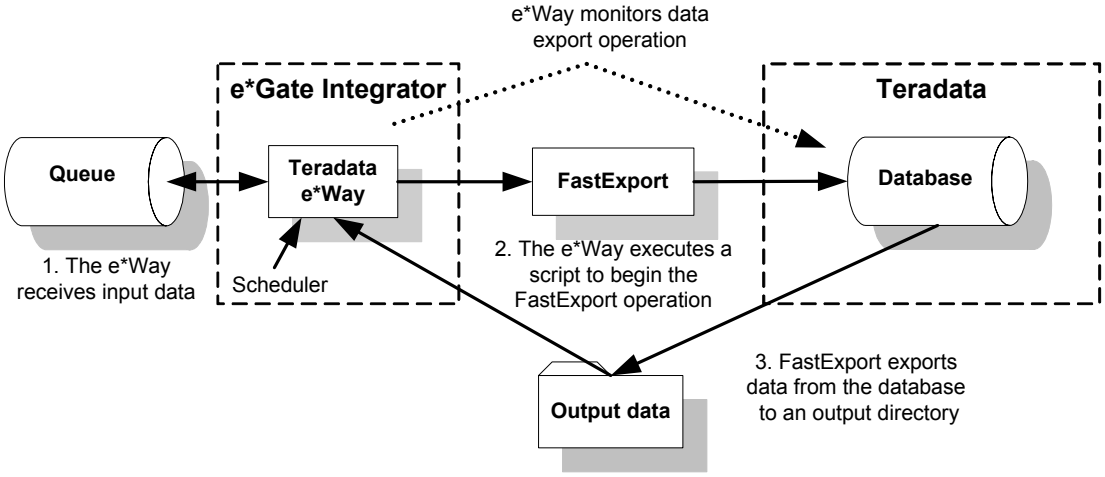
Note: *The user can generate a new TPump script by calling the method* ***generateScript()*** *in the Collaboration Rules. The instructions for doing this are provided in* **Chapter 4** *of this document.*

3. The TPump batch load utility populates the database with the data according to the parameters of the script.
4. The e*Way monitors the operation. If an error is detected during the initiation process, the e*Way alerts the e*Gate monitor. If an error occurs during the TPump job, the e*Way writes a description of the error to the ETD, which may then be written to the e*Way log file (see **Error Handling** on page 11 for more information).

1.2.3 Exporting Data Through FastExport

The e*Way Intelligent Adapter for Teradata, configured to work with the Teradata FastExport utility, receives data from an external source, and initializes and monitors the FastExport operation that retrieves specified data from a Teradata database and exports that data to an external directory. This is illustrated in Figure 2.

Figure 2 Data Export via FastExport



Database export occurs as follows:

1. The e*Way Intelligent Adapter for Teradata receives data from an external source, or is triggered by a scheduler.
2. The e*Way executes a script to initiate the FastExport operation. This script can already exist, or the e*Way can generate it.

Note: *The user can generate a new FastExport script by calling the method* ***generateScript()*** *in the Collaboration Rules. The instructions for doing this are provided in* **Chapter 4** *of this document.*

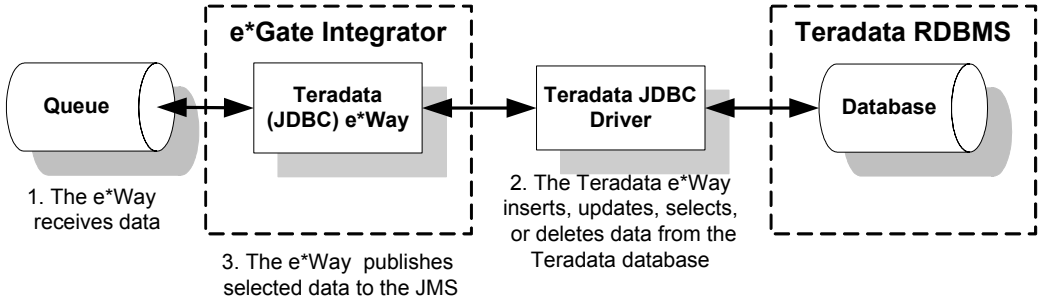
3 The FastExport utility logs onto the Teradata RDBMS, retrieves the specified data, and exports the data to the specified file.

4 The e*Way monitors the operation. If an error is detected during the initiation process, the e*Way alerts the e*Gate monitor. If an error occurs during the FastExport job, the e*Way writes a description of the error to the error log (see **Error Handling** on page 11 for more information).

1.2.4 Retrieving and Publishing Data using the Teradata (JDBC) e*Way

The e*Way Intelligent Adapter for Teradata, configured to work through the Teradata JDBC driver, selects and retrieves data from the Teradata RDBMS and writes the data to an external file. This is illustrated in Figure 3. Given input data, it can also insert data into the Teradata RDBMS.

Figure 3 Data Retrieval via the Teradata (JDBC) e*Way



Data retrieval occurs as follows:

1 The e*Way Intelligent Adapter for Teradata receives data from an external source, or is triggered by a scheduler.

2 The Teradata JDBC e*Way inserts, updates, deletes, or selects data in the Teradata RDBMS via the Teradata JDBC driver.

3 Selected data is published to an external file.

1.2.5 Error Handling

Schema Manager Alert Notifications and Log Files

The e*Gate system continually issues monitoring Events to provide information on how well the overall system is functioning. All major e*Gate components and features issue these Events via internal system operations. The Control Broker converts monitoring Events into notifications (notification Events) and sends them to the e*Gate monitors.

Notifications that indicate problems are called Alert notifications. A message readout of the Alert notifications appears in the Schema Manager GUI, and provides immediate, easy-to-read information on system problems.

The e*Gate system logging facility enables you to trace and store detailed operations information. Log files are generated by various e*Gate components, which include the Control Brokers, Intelligent Queue (IQ) Managers, and e*Way Intelligent Adapters. Each log file is clearly labeled as belonging to its generating module component.

For detailed information on the Schema Manager, Alert notifications, and logging facilities, see the *e*Gate Integrator Alert and Log File Reference Guide* and the *e*Gate Integrator User's Guide.*

Teradata Error Recovery

All or None Importing Data Using TPump

The Teradata e*Way Adapter employs the inherent functionality of Teradata's TPump error recovery mechanism, which provides a roll-back of all imported records in the event of an error. This "All or None" option imports up to a maximum of 300 records per PACK. Currently TPump restricts the maximum number of records to rollback to 300. To ensure that all records are rolled back in case of any error, the error threshold (ERRLIMIT) must be set to 1, and the input file must be limited to 300 records for import, and PACK size be set to 300.

This option is best suited for use with interfaces that must maintain order, in which case, once the a failure occurs the entire interface is suspended until the errors are resolved.

For more information on TPump error recovery functionality (ERRLIMIT and PACK) refer to the BEGIN LOAD command in the *Teradata TPump Reference*.

TeradataETL ETD ErrorOutput

The Teradata e*Way ETD, Teradata ETL (TeradataETL.xsc), provides another error handling option, in which a large number of records can be imported and problem records are identified. This "continuation with error-identification mechanism" allows good records to be committed to the database. Bad records can then be repaired and re-submitted later.

The ErrorOutput node of the ETD contains three subnodes that provide the following:

- ErrorCode: contains the Teradata UTY-Series Message error code (if available).
- ErrorText: contains the text of the error.
- RecordLineNumber: contains the line number of the record in the input file, that caused the error (if available).

The user can query the ErrorOutput nodes of the ETD for the line number and error code of the problem records. With this information the user can extract those records from the file, and either write the records with the error code and text to a separate file, send them to a queue, or place them in an email, to process later.

1.3 Supported Operating Systems

The Teradata e*Way is available for the following operating systems:

- Windows 2000 and Windows Server 2003
- HP-UX 11.0
- IBM AIX 5.1L
- Sun Solaris 8

Note: *The e*Gate Schema Designer runs only on Windows operating systems.*

1.4 System Requirements

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

- An e*Gate Participating Host.
- A TCP/IP network connection.
- The Teradata ODBC driver installed on the e*Gate server. This is required for the Database Builder Wizard only.
- The Teradata JDBC driver version TeraJDBC.02.02.01.07 or higher installed on the e*Gate server.
- The Teradata FastExport and TPump Utilities installed on the e*Gate server.
- Sufficient free disk space on both the Participating Host and the Registry Host to accommodate the e*Way executable, configuration, library, and script files. Additional free disk space is also required to process and queue data. The amount of free space needed varies according to the volume of data processed.

Review the Teradata e*Way **Readme.txt** file for any additional requirements prior to installation. This file is located on the *e*Gate Integrator Installation CD-ROM* in the *\setup\addons\ewteradata* directory.

1.5 External System Requirements

The Teradata e*Way supports the Teradata RDBMS (version V2R4.1.3 or higher) running on a Microsoft Windows 2000 or NCR UNIX MP - RAS r3.02.00 system.

It is assumed any customization of Teradata required for your implementation has been performed previous to the installation of e*Gate. No special configuration of Teradata is required to interface with e*Gate.

Chapter 2

Installation

This chapter describes the requirements and procedures for installing the Teradata e*Way. It contains the following instructions and information:

Note: *Review the **Readme.txt** file (located in the **setup\addons\ewteradata** directory on the installation CD-ROM) for important information regarding this installation.*

2.1 Installing the Teradata e*Way on a Microsoft Windows System

2.1.1 Pre-installation

- Exit all Windows programs before running the setup program, including any antivirus applications.
- Make sure that you are logged into the system with an account that has Administrator privileges.

2.1.2 Installation Procedure

1. Log in as Administrator on the workstation on which you want to install the e*Way.
2. Insert the e*Way installation CD-ROM.
3. Launch the setup program.

Note: *If the CD-ROM drive's Autorun feature is enabled, the setup program should launch automatically. If this does not occur, launch the **setup.exe** file using Windows Explorer or the Control Panel's **Add/Remove Applications** utility.*

4 Follow the setup application instructions until you come to the **Please choose the product to install** dialog box.

5 Select **e*Gate Integrator**, then click **Next**.

6 Follow the on-screen instructions until you come to the second **Please choose the product to install** dialog box.

7 Clear all check boxes except **Add-ons**, then click **Next**.

8 Follow the on-screen instructions until you come to **Select Components**.

9 Select (but do not check) **e*Ways** and click **Change**. The **Select Sub-components** dialog box appears.

10 Select the **Teradata e*Way**.

11 Click **Continue** to return to the **Select Components** dialog box, then click **Next**.

12 Follow the rest of the instructions to complete the installation of the Teradata e*Way.

Important: *Make sure that all the e*Way files are installed in the suggested client installation directories. The installation utility detects and suggests the appropriate directory for installing these files. Unless directed to do so by SeeBeyond support personnel, do not change the suggested installation directory setting.*

2.2 Installing the Teradata e*Way on UNIX

2.2.1 Pre-installation

Root privileges are not required to install this e*Way. Log in under the user name that you wish to own the e*Way files. Be sure that this user has sufficient privileges to create files in the e*Gate directory tree.

2.2.2 Installation Procedure

1 Log onto the workstation containing the CD-ROM drive, and insert the installation CD-ROM.

2 If necessary, mount the CD-ROM drive.

3 At the shell prompt, type

cd /cdrom

4 Start the installation script by typing

setup.sh

5 A menu of options will appear. Select the **Install e*Way** option. Then, follow the additional on-screen directions to completion

Note: *Be sure to install the e*Way files in the suggested* ***client*** *installation directory. The installation utility detects and suggests the appropriate installation directory.* ***Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested "installation directory" setting.***

2.3 Files and Directories Created by the Installation

The Teradata e*Way installer installs the files shown in Table 1 within the e*Gate directory tree. Files are installed within the *egate\client* tree on the Participating Host and committed to the default schema on the Registry Host.

Table 1 Files Created by the Installation

e*Gate Directories	File(s)
	stcewteradata.ctl stcewteradataetl.ctl
\bin	NewDatabaseWizard.dll
\bin\WizardIcons	NewDatabase_Wizard.bmp
\classes	stcjdbcx.jar stcteradataetl.jar
\configs\teradata	Teradata.def
\configs\teradataetl	TeradataETL.def
\etd	teradata.ctl teradataetl.ctl
\etd\teradata	Com_stc_jdbcx_teradatacfg.java Com_stc_jdbcx_teradatacfg.xsc
\etd\teradataetl	TeradataETL.xsc

2.4 Importing the Teradata JDBC Patch and Jar file

The Teradata e*Way requires the installation of the Teradata JDBC driver version **TeraJDBC.02.02.01.07** or higher. The patch includes the Teradata jar file, **teradata.jar**, which must be copied to the e*Gate Client and Server directories. The TeraJDBC patch is available from NCR Teradata support.

To install the TeraJDBC patch and copy the jar file to the e*Gate directories, do the following:

1. Install the **TeraJDBC patch** by double-clicking **teraJDBC.02.02.01.07.exe**.
2. Copy the **teradata.jar** file from:

   ```
   C:\Program Files\NCR\TeraJDBC\classes\teradata.jar
   ```

   (or wherever the TeraJDBC patch is installed) to the following directories:

```
<egate\client>\ThirdParty\NCR\classes
```

where <egate/client> is the location of your e*Gate client directory, and

```
<egate\server>\registry\repository\default\ThirdParty\NCR\classes
```

where <egate/server> is the location of your e*Gate server directory.

Chapter 3

Configuration

This chapter contains the procedures for configuring the e*Way Intelligent Adapter for Teradata. This process includes creating and configuring the **Multi-Mode e*Ways** and the **e*Way Connections**. Network access configuration is also described in this chapter.

This information is found in the following sections:

3.1 Multi-Mode e*Way Configuration

A Multi-Mode e*Way is a multi-threaded component used to route and transform data within e*Gate. Multi-Mode e*Ways can use multiple simultaneous e*Way Connections to communicate with external systems and Intelligent Queues (IQs).

This section provides instructions for using the e*Gate Schema Designer to create and configure a Multi-Mode e*Way.

Additional Information

This document contains basic instructions for creating and configuring a Multi-Mode e*Way. The following resources contain additional information:

- *e*Gate Integrator User's Guide*
- *Standard e*Way Intelligent Adapter User's Guide*
- The e*Way Editor's online Help

3.1.1 Creating a Multi-Mode e*Way

1. On the e*Gate Schema Designer Navigator, click the **Components** tab.
2. Open the host on which you want to create the e*Way.
3. Click the **Create a New e*Way** button.

4 Enter a name for the new e*Way and click **OK**.

5 Right-click the new e*Way and select **Properties**.

The **e*Way Properties** dialog box is displayed.

6 In the **Executable File** field, select **stceway.exe** (located in the *bin* directory) if it is not selected by default.

7 In the **Additional Command Line Arguments** field, enter any additional command line arguments *at the end* of the existing command-line string. Do not change any of the default arguments unless you have a specific need to do so.

8 Under the **Configuration File** field, click one of the following:

- Click **New** to create a new configuration file.
- Click **Find** to select an existing configuration file.

Note: *If a configuration file has already been assigned to this e*Way, you can edit it by clicking* ***Edit****.*

The e*Way Configuration Editor is displayed.

9 Set the parameters of the configuration file.

Note: *Configuration file parameter settings are explained in* **Multi-Mode e*Way Configuration Parameters** *on page 19.*

10 After setting the parameters, click **Save**.

11 Select **Promote to Run Time**.

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

3.2 Multi-Mode e*Way Configuration Parameters

As described in **Creating a Multi-Mode e*Way** on page 18, you can use the e*Way Configuration Editor to set the Multi-Mode e*Way configuration parameters. These parameters are described in this section.

The Multi-Mode e*Way configuration has two sections:

- **JVM Settings** on page 19
- **General Settings** on page 23

3.2.1 JVM Settings

The JVM Settings section contains the following parameters associated with the JVM (Java Virtual Machine):

- **JNI DLL Absolute Pathname** on page 20
- **CLASSPATH Prepend** on page 21

JNI DLL Absolute Pathname

Description

Specifies the absolute pathname to where the JNI DLL installed by the Java 2 SDK 1.3 is located on the Participating Host.

Required Values

A valid pathname.

Additional Information

The JNI DLL name varies on different operating system (OS) platforms, as outlined in Table 2.

Table 2 Java 2 JNI DLL Name by Operating System

Operating System	Java 2 JNI DLL Name
Windows	jvm.dll
Solaris	libjvm.so
HP-UX	libjvm.sl
AIX	libjvm.a

The value assigned can contain a reference to an environment variable. This is done by enclosing the variable name within a pair of percent-sign (%) symbols. For example:

```
%MY_JNIDLL%
```

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

Important: *To ensure that the JNI DLL loads successfully, the Dynamic Load Library search path environment variable must be set appropriately to include all the directories under the Java 2 SDK (or JDK) installation directory that contain shared libraries (UNIX) or DLLs (Windows).*

CLASSPATH Prepend

Description

Specifies 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 will be prepended to the CLASSPATH environment variable.

Existing environment variables may be referenced in this parameter by enclosing the variable name in a pair of percent-sign (%) symbols. For example:

```
%MY_PRECLASSPATH%
```

CLASSPATH Override

Description

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

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

Required Values

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

Additional Information

Existing environment variables maybe referenced in this parameter by enclosing the variable name in a pair of percent-sign (%) 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

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

Initial Heap Size

Description

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

Required Values

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

Maximum Heap Size

Description

Specifies the value for the maximum heap size for native threads in bytes. If set to 0 (zero), the preferred value for the maximum heap size of the JVM will be used.

Required Values

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

Maximum Stack Size for Native Threads

Description

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

Required Values

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

Maximum Stack Size for JVM Threads

Description

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

Required Values

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

Disable JIT

Description

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

Required Values

YES or **NO**.

Note: *This parameter is not supported for Java Release 1.*

Remote debugging port number

Description

Specifies the port number for the remote debugging of the JVM.

Required Values

An integer between 2000 and 65536.

Suspend option for debugging

Description

Specifies whether the option for debugging will be enabled or suspended upon JVM startup.

Required Values

YES or **NO**.

Auxiliary JVM Configuration File

Description

Specifies the relative path to a JVM properties file.

Required Values

A valid relative path name. This parameter is optional.

3.2.2 General Settings

The General Settings section contains the following parameters:

Note: **Rollback Wait Interval** *on page 23*

Rollback Wait Interval

Description

Specifies the period of time (in milliseconds) to wait before rolling back a message (retaining a message and cancelling an unsuccessful transaction).

Required Values

An integer between 0 and 99999999.

Standard IQ FIFO

Description

Specifies whether the e*Way retrieves messages from all STC-standard Intelligent Queues (IQs) in First-In-First-Out (FIFO) order

Required Values

YES or **NO**. **YES** enables FIFO. Subscribing Collaboration retrieve Events for each triggering Event Type/publishing Collaboration to which it subscribes. The Events retrieved are those with the highest priority and the oldest sequence number. The subscribing Collaboration then compares the priorities of all Events it retrieves, and among messages of the highest priority, it publishes the one with the oldest enqueue time.

Selecting **NO** disables FIFO. The Collaboration cycles through a list of triggering Event Type/publishing Collaborations to which it subscribes. The IQ Manager returns the oldest, unread Event (by sequence number) of the highest priority for that Event Type/publishing Collaboration combination. The subscribing Collaboration then processes this Event before retrieving another Event.

3.3 e*Way Connection Configuration

The Teradata e*Way adapter contains two Teradata-specific e*Way Connection types, **Teradata ETL**, and **Teradata (JDBC)**. The parameters for each of these e*Way Connection types are defined in the following sections:

- **Teradata ETL e*Way Connection Parameters** on page 26
- **Teradata (JDBC) e*Way Connection Parameters** on page 34

e*Way Connections are the encoding of access information for specific external connections. The e*Way Connection configuration file contains the parameters necessary for connecting with a specific internal system.

You will use the e*Gate Schema Designer to create a Teradata ETL e*Way Connection and set its configuration parameters, as described in this section.

3.3.1 Creating an e*Way Connection

1. In the Schema Designer's Component editor, select the **e*Way Connections** folder.
2. Click the **Create a New e*Way Connection** button.

   The **New e*Way Connection Component** dialog box appears.
3. Enter a name for the new e*Way Connection and click **OK**.
4. Double-click the new e*Way Connection.

   The **e*Way Connection Properties** dialog box appears.
5. Select **Teradata** (for the JDBC implementation) or **Teradata ETL** as the e*Way Connection Type from the drop-down box.
6. Enter a value for the **Event Type "get" interval** (the default is 10000 milliseconds).
7. Under **e*Way Connection Configuration File**, click one of the following:
   - Click **New** to create a new configuration file.
   - Click **Find** to select an existing configuration file

Note: *If a configuration file has already been assigned to this e*Way Connection, you can edit it by clicking* ***Edit****.*

   The e*Way Connection Configuration Editor is displayed.
8. Set the parameters of the configuration file, as described in the following configuration settings sections for the Teradata and Teradata ETL e*Way Connection Types.
9. After setting the parameters, click **Save** to save your settings as a .cfg file.
10. Select **Promote to Run Time**.
11. Click **OK** to close the **e*Way Connection Properties** dialog box.

3.4 Teradata ETL e*Way Connection Parameters

As described in **"Creating an e*Way Connection" on page 25**, you can use the e*Way Connection Configuration Editor to set the Teradata ETL e*Way Connection's configuration parameters. These parameters are described in this section.

The Teradata ETL e*Way Connection configuration has five sections:

- **Connector** on page 26
- **ETL Control** on page 27
- **Teradata FastExport** on page 29
- **Teradata TPump** on page 31

3.4.1 Connector

The Connector section contains the following parameters:

- **Type** on page 26
- **Class** on page 26

Important: *Do not change the default values of the parameters in the Connector section.*

Type

Description

Specifies the connector type. This is a required parameter.

Required Values

teradataetl

This value is mandatory and should not be changed.

Class

Description

Specifies the class name of the Teradata connector object. This is a required parameter.

Required Values

com.stc.eways.teradataetl.TeradataETLConnector.

This value is mandatory and should not be changed.

3.4.2 ETL Control

The ETL Control section contains the following parameters:

ETL Utility

Description

Specifies the ETL utility to be used, **FastExport** or **TPump**. This is a required parameter.

Required Values

FastExport or **TPump**.

Script File Name

Description

Specifies the absolute path and name of the FastExport or TPump script file that is accessible to the Teradata server. This is a required parameter.

Required Values

A valid path and file name.

Additional Information

If no path is specified, the e*Way writes to or looks for the file in the *\egate\client* directory. The script file contains the FastExport or TPump commands and statements, and redirects stdin (that is, "< scriptfile"). This file must be accessible to the Teradata Server.

Output Log File Name

Description

Specifies the absolute path and name of the FastExport or TPump output log file. When the FastExport or TPump job executes, this file receives the output of the job which redirects stdout (that is, "> outputfile"). This is a required parameter.

Required Values

A valid path and file name.

Additional Information

If you do not specify an absolute path for this value, the e*Way will write this file to the *\egate\client* directory.

Teradata Error File Name

Description

Specifies the absolute path and name of the FastExport or TPump error file from the Teradata server. This is the "-e" option of the FastExport or TPump command. From e*Gate's perspective, this is the remote error file. When the FastExport or TPump job is executed, the errors associated with that job are written to this file. This is a required parameter.

Required Values

The absolute path and name of the FastExport or TPump error file from the remote server's perspective.

Additional Information

If you do not specify an absolute path for this value, the e*Way will write this file to the *\egate\client* directory.

Note: *The ETD stores up to 100 of the most recent errors from the errorfile.*

Start Command

Description

The -r command to start the FastExport or TPump operation. This is an optional parameter.

Required Values

A valid command in string format. Quotes are not required. For example:

```
.RUN FILE abc.txt
```

Note: *For more information about the FastExport or TPump -r command, see the Teradata FastExport Reference or Teradata TPump Reference.*

Character Set

Description

The FastExport or TPump -c option command, which specifies a character set. This is an optional parameter.

Required Values

A valid command in string format. Quotes are not required. For example:

```
ascii
```

Note: *For more information about the TPump -c command, see the Teradata FastExport Reference or Teradata TPump Reference.*

3.4.3 Teradata FastExport

The Teradata server for FastExport section contains the following parameters:

- **Teradata Server Alias** on page 29
- **Teradata User Name** on page 29
- **Teradata Password** on page 30
- **Path To FastExport Binary** on page 30
- **Brief** on page 30
- **Minimum Sessions** on page 31
- **Maximum Sessions** on page 31

Note: *The first three parameters in the Teradata TPump section,* ***Teradata Server Alias****,* ***Teradata User Name****, and* ***Teradata Password****, should be entered as a group in a single location. These values can be entered in the e*Way Connection configuration (.cfg) file, as described in this section, in the Event Type Definition (see* **Chapter 4***), or in the Collaboration (see* **Chapter 5***).*

Teradata Server Alias

Description

Specifies the Teradata server alias name, from the host file, used in the LOGON support command that establishes a Teradata session between FastExport and the Teradata RDBMS.

Required Values

The alias is the eight, or less, character name of the Teradata server that precedes the appended tag in the name used to configure network access to the Teradata server. For example, the e*Gate server's HOSTS file may refer the Teradata server as follows:

```
10.10.10.1   TERASVRCOP1
```

In this example, **TERASVR** is the alias and **COP1** is the appended tag. **TERASVR** would be entered as the Teradata Server Alias.

Teradata User Name

Description

Specifies the Teradata database login user name used in the LOGON support command which establishes a Teradata session between FastExport and the Teradata RDBMS.

Required Values

A valid database login ID.

Teradata Password

Description

Specifies the Teradata database password used in the LOGON support command which establishes a Teradata session between FastExport and the Teradata RDBMS.

Required Values

A valid database login password.

Note: *You can also enter a value for the **Password** parameter in the Event Type Definition (see **Chapter 4**), or in the Collaboration (see **Chapter 5**). However, it is recommended that you enter this value as an e*Way Connection configuration setting and save it to the **.cfg** file. This ensures encryption of the password. It is also recommended that wherever you enter the values for **Teradata Server Alias**, **Teradata User Name**, and **Teradata Password** parameters, you enter them as a group in a single location. If these values are set in the .cfg (configuration) file they should not be included explicitly when setLOGON is used in the Collaboration Rules.*

Path To FastExport Binary

Description

Specifies the absolute path to the location of the FastExport executable. This is a required parameter.

Required Values

The absolute path for the Teradata server FastExport executable. For example:

```
"C:\Program Files\NCR\Teradata Client\bin"
```

Brief

Description

Specifies whether the -b option is used to limit the utility printout to the least information required to determine success or failure. This parameter is optional.

Required Values

No or **Yes**. Yes indicates that the -b option is used. The configured default is No.

Note: *For more information about the FastExport -b command, see the Teradata FastExport Reference.*

Minimum Sessions

Description

Specifies the -N option that determines the minimum number of sessions required to run the job. This parameter is optional.

Required Values

A number indicating the minimum number of sessions to run the job.

Note: *For more information about the FastExport -N command, see the Teradata FastExport Reference.*

Maximum Sessions

Description

Specifies the -M option that determines the maximum number of sessions allowed to log on. This parameter is optional.

Required Values

A number indicating the maximum number of sessions logged on.

Note: *For more information about the FastExport -M command, see the Teradata FastExport Reference.*

3.4.4 Teradata TPump

The Teradata TPump section contains the following parameters:

- **Teradata Server Alias** on page 32
- **Teradata User Name** on page 32
- **Teradata Password** on page 32
- **Path to TPump Binary** on page 33
- **Keep Macros** on page 33
- **Buffers Per Session** on page 33
- **Periodicity** on page 33

Note: *The first three parameters in the Teradata TPump section,* ***Teradata Server Alias****,* ***Teradata User Name****, and* ***Teradata Password****, should be entered as a group in a single location. These values can be entered in the e*Way Connection configuration (.cfg) file, as described in this section, in the Event Type Definition (see* **Chapter 4***), or in the Collaboration (see* **Chapter 5***).*

Teradata Server Alias

Description

Specifies the Teradata server alias name, from the host file, used in the LOGON support command that establishes a Teradata session between TPump and the Teradata RDBMS.

Required Values

The alias is the eight, or less, character name of the Teradata server that precedes the appended tag in the name used to configure network access to the Teradata server. For example, the e*Gate server's HOSTS file may refer the Teradata server as follows:

```
10.10.10.1   TERASVRCOP1
```

In this example, **TERASVR** is the alias and **COP1** is the appended tag. **TERASVR** would be entered as the Teradata Server Alias.

Teradata User Name

Description

Specifies the Teradata database password used in the LOGON support command which establishes a Teradata session between TPump and the Teradata RDBMS.

Required Values

A valid database login ID.

Teradata Password

Description

Specifies the password of a valid Teradata server account for TPump.

Required Values

A valid database login password.

Note: *You can also enter a value for the **Password** parameter in the Event Type Definition (see **Chapter 4**), or in the Collaboration (see **Chapter 5**). However, it is recommended that you enter this value as an e*Way Connection configuration setting and save it to the **.cfg** file. This ensures encryption of the password. It is also recommended that wherever you enter the values for **Teradata Server Alias**, **Teradata User Name**, and **Teradata Password** parameters, you enter them as a group in a single location. If these values are set in the .cfg (configuration) file they should not be included explicitly when setLOGON is used in the Collaboration Rules.*

Path to TPump Binary

Description

Specifies the absolute path to the location of the TPump executable. This is a required parameter.

Required Values

A valid path.

Keep Macros

Description

Specifies the TPump -m command, which retains macros between jobs. This is an optional parameter.

Required Values

YES or **NO**. **NO** is the default.

Note: *For more information about the TPump -m command, see the Teradata TPump Reference.*

Buffers Per Session

Description

Specifies the TPump -f command, which indicates the number of buffers per session. This is an optional parameter.

Required Values

An integer between 2 and 10.

Note: *For more information about the TPump -f command, see the Teradata TPump Reference.*

Periodicity

Description

Specifies the -d command. This is an optional parameter.

Required Values

An integer between 1 and 30. 4 is the default.

Note: *For more information about the TPump -d command, see the Teradata TPump Reference.*

3.5 Teradata (JDBC) e*Way Connection Parameters

As described in **"Creating an e*Way Connection" on page 25**, you can use the e*Way Connection Configuration Editor to set the Teradata e*Way Connection's configuration parameters. These parameters are described in this section.

The Teradata ETL e*Way Connection configuration has two sections:

- **DataSource** on page 34
- **Connector** on page 35

3.5.1 DataSource

The ETL Control section contains the following parameters:

- **Driver** on page 34
- **JDBC URL** on page 34
- **User Name** on page 34
- **Password** on page 35

Driver

Description

Specifies the class name of the JDBC driver.

Required Values

The class name of the JDBC driver. The default is com.ncr.teradata.TeraDriver.

JDBC URL

Description

Specifies the JDBC URL used to gain access to the Teradata database.

Required Values

A typically example is:

jdbc:teradata://<Teradata Server Alias>

where <Teradata Server Alias> is the Teradata server alias name (see **Teradata Server Alias** on page 29).

User Name

Description

Specifies the case-insensitive user name used to connect to the Teradata database.

Required Values

A valid database login ID.

Password

Description

Specifies the encrypted password used to connect to the database

Required Values

A valid database login password.

3.5.2 Connector

The Connector section contains the following parameters:

Important: *Do not change the default values of the parameters in the Connector section.*

Type

Description

Specifies the connector type for the JDBC connection.

Required Values

DB. Currently this is the only one type.

This value is mandatory and should not be changed.

Class

Description

Specifies the Java class name of the JDBC connector object.

Required Values

com.stc.eways.teradataetl.TeradataETLConnector.

This value is mandatory and should not be changed.

Transaction Mode

Description

Specifies how transactions should be handled. This parameter has two options:

- **Automatic**: eGate takes care of transaction control. Users should not issue commit or rollback.
- **Manual**: Users take care of transaction control by issuing commit or rollback.

Required Values

Automatic or **Manual**. Automatic is the default setting.

Connection Establishment Mode

Description

Specifies how the database connection established and closed. The parameter has three options:

- **Automatic**: the connection is automatically established when the Collaboration is started. The connection remains active as needed.
- **OnDemand**: the connection is established on demand, when Business Rules that require a connection to the external system are performed. The connection is closed when the methods have concluded.
- **Manual**: the user explicitly calls the connection connect and disconnect methods in the Collaboration's Business Rules.

Required Values

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

Connection Inactivity Timeout

Description

Specifies timeout (in milliseconds) for the Automatic connection establishment mode.

- If this parameter is not set, or if it is set to 0, the connection is not closed for inactivity. The connection continues to remain live. If the connection is cut off, it attempts to re-establishing a connection automatically.
- If a non-zero value is specified, the connection manager monitors inactivity. The connection is closed when the specified value (in milliseconds) has been reached.

Required Values

Blank or **0** to continuously maintain a live connection, or specify a **value in milliseconds** (for example, **500** for 5 seconds) to indicate the amount of inactivity allowed before a connection is closed.

Connection Verification Interval

Description

Specifies the regularity (in milliseconds) that the connection status to the database server is monitored. If the connection to the server is detected as down during verification, the Collaboration's **onConnectionDown** method is called. If the connection comes from a previous connection error, the Collaboration's **onConnectionUp** method is called. If no value is specified, a default value of 60000 milliseconds is used.

Required Values

The minimum period of time (in milliseconds) between verification of the connection status to the database server.

3.6 Network Access Configuration

The Teradata e*Way adapter supports Windows and UNIX environments, including cross platform (Windows to UNIX) environments. In order to the TPump and FastExport executables, the TPump and FastExport utilities must be installed on the e*Gate server.

You must supply the Teradata server's IP address and an alias of 8 characters or less, plus an appended tag, typically **COP1,** to the system HOSTS file.

For example:

```
10.10.10.1   TERASVRCOP1
```

The **system alias** must have a length of eight characters or less, plus the appended tag, typically **COP1**. This alias is also used in the e*Way Connection configuration parameters **Teradata Server Alias** for FastExport and TPump.

Chapter 4

Teradata ETL ETD Overview

This chapter provides an overview of the Teradata Extraction, Transformation, Load (ETL) Event Type Definition (ETD) hierarchy structure. It includes information about nodes, available methods, and properties, and their application.

4.1 Overview

The Teradata ETL e*Way uses an Event Type Definition (ETD) to parse, validate, and (if necessary) transform Events.

To clarify, the terms associated with ETDs are defined as follows:

- An **Event** is a packet of data within e*Gate.
- An **Event Type** is a class of Events with common data structure.
- An **Event Type Definition** (ETD) is a programmatic representation of an Event Type that Collaboration Rules can use when parsing, transforming, or routing data.

The e*Gate system packages data within Events and categorizes them into Event Types. Common elements between Events define the Event Type and comprise the ETD.

4.2 Teradata ETL ETD Structure

The Teradata ETL ETD exposes the APIs that e*Gate uses to access Teradata. The Teradata ETL ETD has two components:

- The **TeradataETL.xsc** file, which exposes the structures and methods
- **Java classes**, which implement the structures and methods

The following sections describe the Teradata ETL ETD in detail and provide information for using the ETD to build Java Collaboration Rules to access Teradata.

Default ETD Configuration Values

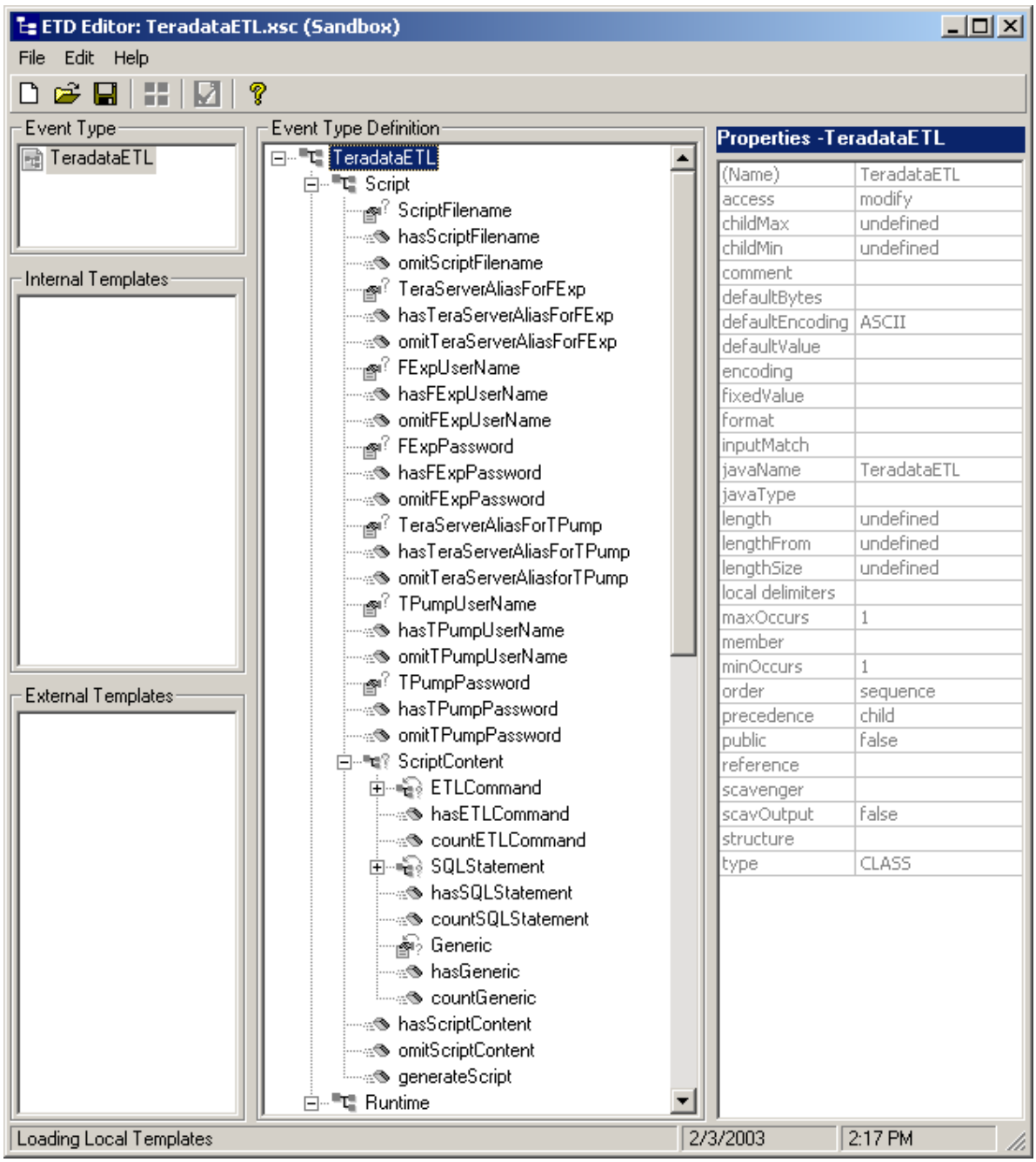
The e*Way Connection's configuration values (see **e*Way Connection Configuration** on page 25) are used to initially populate the Teradata ETL ETD configuration settings.

You can edit any of these values within the ETD itself. The edited values then supersede the e*Way Connection configuration values for the ETD.

e*Gate Schema Designer ETD Editor

The Teradata ETL ETD configuration settings can be edited from the e*Gate ETD Editor (see Figure 4).

Figure 4 TeradataETL.xsc (Teradata ETL ETD) in e*Gate Schema Designer ETD Editor



As shown in Figure 4, TeradataETL is the root node. The root node provides a graphical representation of the interface in the ETD Editor. Expanding the node reveals all of the methods and attributes in the interface, which are themselves represented as nodes. A node representing a method is typically expandable, revealing parameters.

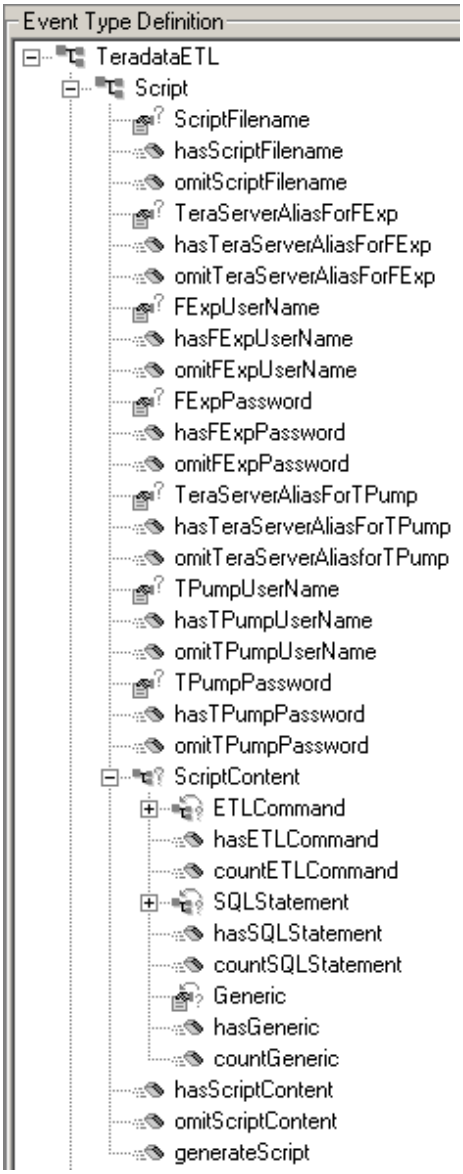
The TeradataETL ETD contains two main nodes:

- **Script Node** on page 41
- **Runtime Node** on page 47

4.2.1 Script Node

The Script node is used to define the script that e*Gate uses to initialize the database population operation. Figure 5 shows the Script node in its expanded form.

Figure 5 Script Node in TeradataETL ETD

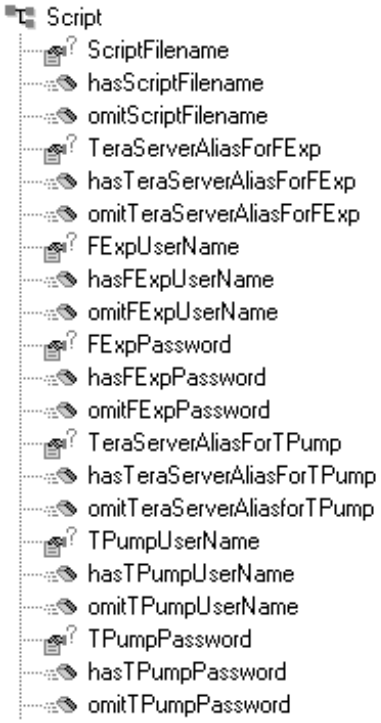


Note: *The **generateScript()** method is only called in the Collaboration if the e*Way is used to generate the script file. If the e*Way uses an existing script file to run the database population operation, then **generateScript()** is not called.*

Attribute Nodes for the Script Node

The Script node contains attribute nodes that define the configuration of the Teradata script file generated by the e*Way.

Figure 6 Script Node Attribute Nodes



Note: *Many of the attribute nodes employ helper methods, such as **"has"**, **"omit"**, and **"count"**. See* **Script Node Helper Methods** *on page 47* *for more information about these methods.*

Table 3 lists the attribute nodes for the Script node.

Table 3 Teradata ETL ETD - Script Node Attribute Nodes

Node Name	Function
ScriptFileName	Specifies the name and location of the script file
TeraServerAliasForFExp	Specifies the alias for the name of the Teradata server for FastExport
FExpUserName	Specifies the user name of a valid Teradata server account for FastExport
FExpPassword	Specifies the password of a valid Teradata server account for FastExport

Table 3 Teradata ETL ETD - Script Node Attribute Nodes

Node Name	Function
TeraServerAliasForTPump	Specifies the alias for the name of the Teradata server for TPump
TPumpUserName	Specifies the user name of a valid Teradata server account for TPump
TPumpPassword	Specifies the password of a valid Teradata user account for TPump

Note: *The values for **TeraServerAliasForFExp, FExpUserName**, and **FExpPassword**, should be entered as a group in a single location. This is also true for **TeraServerAliasForTPump**, **TPumpUserName**, and **TPumpPassword**. These values can be entered in the e*Way Connection configuration (**.cfg**) file (see* Chapter 3*), in the Event Type Definition (as described here), or in the Collaboration (see* Chapter 5*).*

*Note also that the value for the TPump and FastExport passwords are only encrypted if they are entered in the e*Way Connection configuration file. For more information, see* Teradata Password *on page 30 for **FastExport**, or page 32 for **TPump**.*

ScriptContent

The ScriptContent attribute node contains a number of nodes that define TPump, FastExport, and SQL commands.

TPump and FastExport Command Nodes

Table 4 lists the TPump and FastExport command nodes and their functions.

Table 4 TPump and FastExport Command Nodes

Sub-node Name	Function
ACCEPT	Accepts the value and data type of one or more external-source utility variables
BEGIN_EXPORT	Indicates the beginning of an export task and sets the task session's specifications with the Teradata RDBMS
BEGIN_LOAD	Indicates the following: ♦ The beginning of a TPump task ♦ The task execution parameters
DATEFORM	Defines the form of the TPump job's DATE data type specifications
DISPLAY	Writes messages to the destination specified

Table 4 TPump and FastExport Command Nodes

Sub-node Name	Function
DML_LABEL	For the Teradata SQL DML statement(s) following the DML command, defines the following: • Label • Error treatment option *Note: The following are DML statement options:* • *INSERT* • *UPDATE* • *DELETE*
ELSE	If the preceding IF command is false, commands and statements following ELSE are executed.
ENDIF	Exits from conditional expression command sequences (IF or IF/ELSE). ENDIF is followed by commands or statements executed when the program resumes.
END_EXPORT	Indicates the end of an export task and initiates processing by the Teradata RDBMS.
END_LOAD	The final command of a TPump task, END_LOAD indicates the following: • The end of TPump command entries • That the task should begin
EXPORT_OUTFILE	Exports the following information to a file: • The client system destination. • File format specifications for export data retrieved from the Teradata RDBMS. Also, generates a MultiLoad script file that can be used to reload export data into the Teradata RDMBMS.
EXPORT_OUTMOD	Exports the following information to a module: • The client system destination. • File format specifications for export data retrieved from the Teradata RDBMS. Also, generates a MultiLoad script file that can be used to reload export data into the Teradata RDMBMS.
FIELD	Used with the LAYOUT command, FIELD provides a definition of a field in the source record. Fields specified are sent to the Teradata RDBMS.
FILLER	Used with the LAYOUT command, FILLER defines a field in the data source that is not to be sent to the Teradata RDBMS.
IF	If the conditional expression following the IF command is true, the commands and statements that follow it are executed.
IMPORT_INFILE	Reads data from a file and provides the following to the client program: • Data source • Layout • Optional selection criteria

Table 4 TPump and FastExport Command Nodes

Sub-node Name	Function
IMPORT_INMOD	Identifies the following to a module: • Data source • Layout • Optional selection criteria
LAYOUT	Specifies the layout of externally-stored data records that are used in the TPump task. LAYOUT is used together with a sequence of these commands that immediately follows: • FIELD • FILLER • TABLE
LOGOFF	On the client, disconnects all active sessions and terminates the execution of TPump.
LOGON	LOGON provides the following functionality: • On the Teradata RDBMS, establishes a Teradata SQL session • Indicates the LOGON string to be used when connecting sessions for all future functions.
LOGTABLE	In the event of a client or Teradata RDBMS failure, a safe automatic restart of TPump uses the table indicated in LOGTABLE for required journaling checkpoint information.
NAME	Sets the utility variable &SYSJOBNAME with a job name (up to 16 characters).
ROUTE_MESSAGES	Identifies an alternative destination for FastExport report output.
RUN_FILE	Invokes a specified external file, the current source of SQL statements and utility commands.
SET	Assigns a value and data type to a utility variable.
SYSTEM	Used to suspend TPump for the purpose of issuing commands to the local operating system.
TABLE	Used with the LAYOUT command, identifies a table that contains column names and data descriptions. These are used as the names and data descriptions in the input record fields. *Note: This command is used in place of, or in addition to, the FIELD command.*
THEN	Followed by an IF condition.

SQL Command Nodes

Table 5 lists the SQL command nodes and their functions.

Table 5 SQL Command Nodes

Sub-node Name	Function
ALTER_TABLE	Alters the options or column configuration of a table that currently exists.
CHECKPOINT	Adds a checkpoint entry to a journal table.
COLLECT_STATISTICS	For one or more table columns, collects statistical data.
COMMENT	Retrieves or stores a comment string that is associated with a database object.
CREATE_DATABASE	Creates a new database.
CREATE_MACRO	Creates a new macro.
CREATE_TABLE	Creates a new table.
CREATE_VIEW	Creates a new view.
DATABASE	For the current session, indicates a default database.
DELETE	Deletes (removes) rows from a table.
DELETE_DATABASE	Removes the following from a database: ♦ Tables ♦ Views ♦ Macros
DROP_DATABASE	Removes a database from the Teradata RDBMS.
DROP_TABLE	Removes a table from the Teradata RDBMS.
EXECUTE	Specifies a macro created by the user that is used for execution.
GIVE	Transfers database ownership to another user.
GRANT	Grants access privileges to a database object.
INSERT	Inserts new rows in a table.
MODIFY_DATABASE	Changes the options of an existing database.
RENAME	Renames an existing table, view, or macro.
REPLACE_MACRO	Redefines an existing macro.
REPLACE_VIEW	Redefines an existing view.
REVOKE	Revokes access privileges to a database object.
SELECT	Queries information from a table.
SET_SESSION_COLLATION	Overrides the collation specification for the current session.
UPDATE	Changes the column values of an existing row in a table.

generateScript()

The **generateScript()** method is called if the e*Way is to generate the TPump script file. When called, the e*Way takes the content of the input data from the user and creates the script at a location specified in the configuration file or the ETD.

Note: *The **generateScript()** method is typically called by the user in the Collaboration Rules. See* **Creating the tpump_import Collaboration Rules** *on page 67.*

Important: *e*Gate must have a network path to the location of the script file, such as an NFS mounted directory, UNC name, or local directory. If this is not the case, the e*Way must be running on the same system as the TPump utility. The e*Way does not provide File Transfer Protocol (FTP) support for files.*

Script Node Helper Methods

The attribute nodes of the Script node employ a number of helper methods. These methods operate as follows:

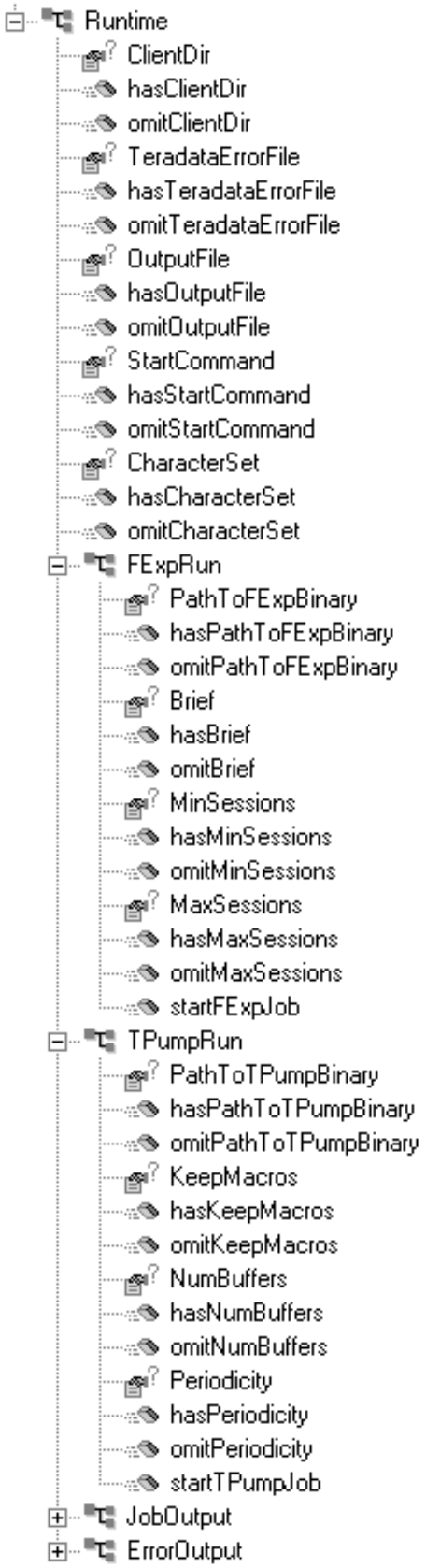
- The **has** method returns true if the field node is set.
- The **omit** method operates such that the **has** method returns false after the **omit** method is called.
- The **count** method returns the number of elements in the field.

4.2.2 Runtime Node

The Runtime node contains attribute nodes that hold the Teradata server information and the output of the TPump and FastExport task.

Figure 7 shows an expanded view of the Runtime node.

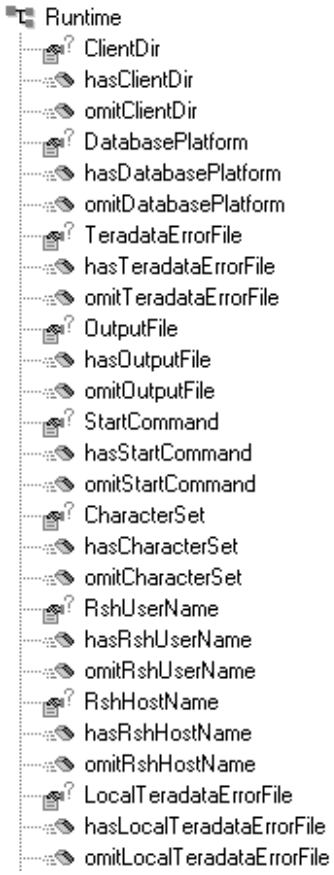
Figure 7 Runtime Node in TeradataETL ETD



Attribute Nodes for the Runtime Node

The Runtime node contains attribute nodes that define the configuration of the Teradata script file generated by the e*Way (see Figure 8).

Figure 8 Runtime Attribute Nodes



Note: *Many of the attribute nodes employ helper methods, such as* ***"has"****,* ***"omit"****, and* ***"count"****. See* **Runtime Node Helper Methods** *on page 50 for more information about these methods.*

Table 6 lists the attribute nodes for the Runtime node.

Table 6 Teradata ETL ETD - Runtime Node Attribute Nodes

Node Name	Description
ClientDir	The name of the e*Gate client directory
ErrorFile	The absolute path and name of the error file on the Teradata server from the perspective of the remote server
OutputFile	The location and name of the FastExport or TPump output file
StartCommand	The command to start the TPump or FastExport operation
CharacterSet	The TPump or FastExport command specifying character set

Note: *For more information about FastExport or TPump commands, see the* ***Teradata FastExport Reference*** *and the* ***Teradata TPump Reference****.*

Runtime Node Helper Methods

The attribute nodes of the Runtime node employ a number of helper methods. These methods operate as follows:

- The **has** method returns true if the field node is set.
- The **omit** method operates such that the **has** method returns false after the **omit** method is called.
- The **count** method returns the number of elements in the field.

FExpRun

FExpRun contains components that facilitate the creation of the command used to execute the data export script (seeFigure 9).

Figure 9 FExpRun

StartFExpJob

FExpRun contains a single method, **startFExpJob**. This method forms a command that's sent to the FastExport utility to execute the script file.

For example, this command appears as follows:

For Windows operating systems:

```
<FastExportPath>\fexp.exe -e <errorfile> > <outputfile> <
    <scriptfile>
```

For UNIX operating systems:

```
<FastExportPath>/fexp -e <errorfile> > <outputfile> < <scriptfile>
```

FExp Attribute Nodes

FExpRun contains four attribute nodes, which are listed in Table 8.

Note: *For more information about FastExport commands, see the Teradata FastExport Reference.*

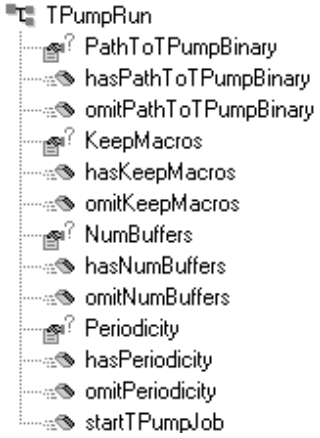
Table 7 FExpRun Attribute Nodes

Attribute Node Name	Description
PathToFExpBinary	Specifies the absolute path to the FastExport executable location
Brief	Specifies the -b option used to limit the utility printout to the least information required to determine success or failure.
MinSessions	Specifies the -N option that determines the minimum number of sessions required to run the job.
MaxSessions	Specifies the -M option that determines the maximum number of sessions allowed to log on.

Note: *The FExpRun attribute nodes employ helper methods, such as **"has"** and **"omit"**. See* **Runtime Node Helper Methods** *on page 50 for more information about these methods.*

Similar to FExpRun, TPumpRun contains components that facilitate the creation of the command used to execute the database population script (see Figure 10).

Figure 10 TPumpRun



startTPumpJob

TPumpRun contains a single method, **startTPumpJob**. This method forms a command that's sent to the TPump utility to execute the script file.

For example, this command appears as follows:

For Windows operating systems:

```
<TPumpPath>\tpump.exe -e <errorfile> > <outputfile> < <scriptfile>
```

For UNIX operating systems:

```
<TPumpPath>/tpump -e <errorfile> > <outputfile> < <scriptfile>
```

TPumpRun Attribute Nodes

TPumpRun also contains attribute nodes, which are listed in Table 8.

Note: *For more information about TPump commands, see the Teradata TPump Reference.*

Table 8 TPumpRun Attribute Nodes

Attribute Node Name	Description
PathToTPumpBinary	Specifies the path to the TPump executable location
KeepMacros	Specifies the TPump -m command, which keeps macros between jobs
NumBuffers	Specifies the TPump -f command, which indicates the number of buffers per session
Periodicity	Specifies the TPump -d command

Note: *Many of the TPumpRun attribute nodes employ helper methods, such as **"has"** and **"omit"**. See* **Runtime Node Helper Methods** *on page 50* *for more information about these methods.*

JobOutput

The JobOutput node contains components that provide information about the database population operation and any errors that may be encountered. See Figure 11.

Figure 11 JobOutput

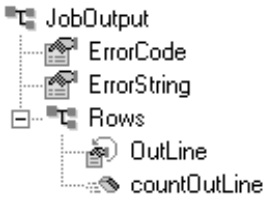


Table 9 defines the components in the JobOutput node.

Table 9 JobOutput Node Components

Component Name	Description
ErrorCode	After the job is complete, ErrorCode displays error codes generated by the e*Way: • **-1** = Job did not start • **0** (zero) = Job completed successfully • **Greater than zero** = Indicates a TPump warning or error
ErrorString	Provides more detailed information about any errors generated by the e*Way.

Rows

The JobOutput node also contains a Rows node, as shown in Figure 11.

Table 10 defines the components of the Rows node.

Table 10 Rows Node Components

Component Name	Description
OutLine	The e*Way inserts the contents of the output file into this node after the completion of the operation.
countOutline	Displays a count of the number of lines in the output file.

Note: *See* **Output Log File Name** *on page 27 for more information about the FastExport and TPump output file.*

ErrorOutput

The ErrorOutput node contains components that provide information about errors generated by Teradata during the database population operations. The ErrorOutput node is shown in Figure 12.

Figure 12 ErrorOutput

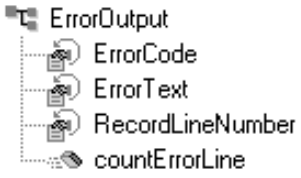


Table 11 defines the ErrorOutput components.

Table 11 ErrorOutput Node Components

Component Name	Description
ErrorCode	Displays the error code. This is the Teradata UTY_Series Message code.
ErrorText	Displays the text of the error.
RecordLineNumber	Displays the line number for the record (if available).
countErrorLine	Displays a count of the number of lines in the error file.

ErrorCodes

Teradata UTY-Series Message codes are captured by the ErrorOutput node of the ETD. A number of these codes include the record number, which provides the RecordLineNumber information. These codes appear in **Table 12 on page 54**.

Table 12 Error Codes Containing RecordLineNumber

UTY-Series Message	Description
UTY1808	The record contains a computational expression that cannot be evaluated.
UTY4007	Contains a description of the input/output error.
UTY4016	The record is to large for the assigned field
UTY4017	The record does not contain enough data for the assigned layout.
UTY4201	The record exceeds the input buffer maximum.
UTY4202	Internal error.
UTY 4203	The record length is too short
UTY4204	The record exceeds the output buffer maximum.
UTY4205	The records input data is smaller that the expected layout size.

RecordLineNumber

When the user calls the get method for **RecordLineNumber** the error log displays the line number of the message at fault for the error, or the default line number "-1" if the record number is not available.

For information regarding all Teradata UTY-Series Messages refer to the *Teradata Messages Reference*.

Note: *See* **Teradata Error File Name** *on page 28 for more information about the TPump or FastExport error file.*

4.2.3 TeradataETL ETD Java Classes

The TeradataETL .xsc structure, described in the previous sections, allows the user to perform drag-and-drop operations on objects to create the Java Collaboration Rules.

Table 13 provides information about the Java classes that implement the Teradata ETL ETD.

Table 13 Teradata ETL ETD Java Classes

Class Name	Description
ScriptFile.java	♦ Creates the script file. ♦ Writes FastExport or TPump commands into the script file.
ETLRun.java	♦ Creates the command to run the FastExport or TPump job. ♦ Runs the FastExport or TPump job. ♦ Writes output and errors into the ETD (TeradataETL.xsc)

Chapter 5

Teradata ETL Implementation

This chapter provides information for implementing the Teradata ETL e*Way in a production environment.

e*Way implementation is demonstrated by configuring the sample schemas included with the e*Way. This chapter provides detailed information about how the sample schemas are created. These procedures can be applied to implementing the Teradata ETL e*Way in your own specific production environment.

5.1 Teradata e*Way Implementation

Implementation of the Teradata e*Way involves completion of the following:

- Activate the Control Broker.
- Use the e*Gate Schema Designer to define and configure the following:
  - The e*Ways
  - The Teradata ETL e*Way Connection
  - The TeradataETL Event Type Definition (ETD)
  - The Intelligent Queue (IQ)
  - Collaboration Rules, which are used to process Events.
  - Collaborations, associated with each e*Way component, to apply the required Collaboration Rules

This chapter describes how to perform each of these steps, using the sample schemas to demonstrate the creation of the e*Way components and their configuration.

- **The TPump Sample Schema** on page 56 provides a step by step creation of the e*Way's components.
- **The FastExport Sample Schema** on page 74 provides an overview of the creation and configuration of the samples components.
- **Copy the fexp_in.dat file from the extracted FExp_Sample files, to the location specified in the set Import Business Rule of the fexp_export Collaboration Rules (see Creating the Business Rules on page 78).** on page 82 provides an overview of the creation and configuration of the samples components.
- **Executing the Schema** on page 82.

5.2 Concerns

The following information should be noted before implementing the Teradata FastExport and TPump sample schemas.

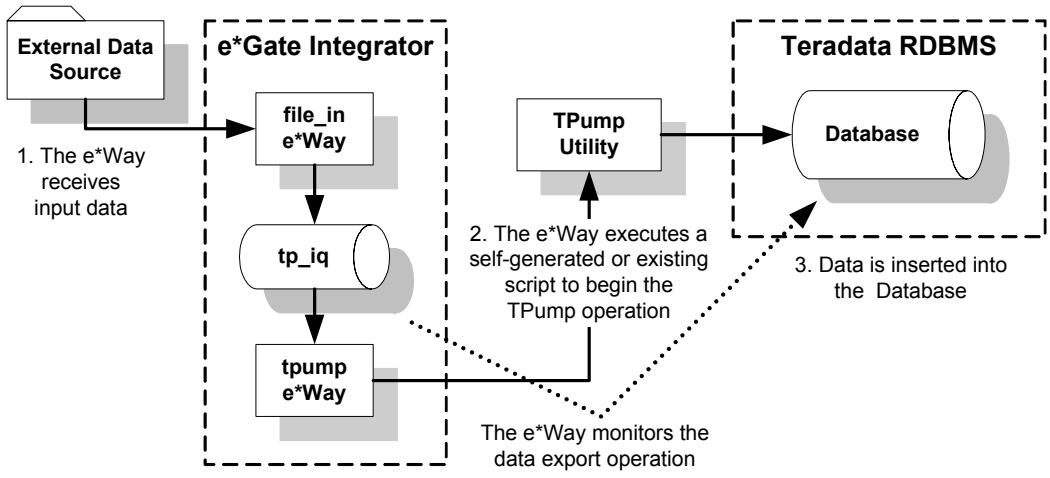
- The Teradata **FastExport** and **TPump** utilities must be installed on the e*Gate server for the Teradata e*Way to use the utilities. For information on installing the FastExport and TPump utilities refer to the *Teradata Tools and Utilities Installation Guide* for your specific system. These documents are available for download from NCR at: **http://www.info.ncr.com/eHome.cfm**
- **Shutting down the Teradata e*Way**: The ExecuteBusinessRules() method in the Collaboration must complete before an e*Way is shutdown. If a JMS rollback occurs in a TPump or FastExport schema, and the user shuts down all modules before the ExecuteBusinessRules() method in the Collaboration has completed, the method will execute again when the modules are restarted, causing TPump or FastExport to re-execute.
- **XA is not supported**: The e*Way Intelligent Adapter for Teradata does not support XA.

Review the Teradata e*Way Readme file for any additional requirements prior to implementation. This file is located on the e*Gate Integrator Installation CD-ROM in the *\setup\addons\ewteradata* directory.

5.3 The TPump Sample Schema

The TPump sample schema, included with the Teradata e*Way, demonstrates how to configure e*Gate to use the TPump utility to populate a Teradata database (see Figure 13.

Figure 13 Teradata e*Way - TPump Sample Schema



The schema's components are nearly complete once a sample schema is imported. In addition, the following sections explain how these components are created manually.

This process involves the following activities, which are explained in this chapter:

- **Importing the TPump Sample Schema** on page 57
- **Creating and Configuring the e*Ways** on page 57
- **Creating and Configuring the Teradata ETL e*Way Connection** on page 59
- **Creating and Configuring Event Type Definitions** on page 61
- **Creating Intelligent Queues** on page 63
- **Creating Collaboration Rules** on page 64
- **Creating Collaborations** on page 71
- **Configuring the Trigger File** on page 73
- **Configuring the Input File** on page 73
- **Executing the Schema** on page 82

Note: *To run the TPump sample schema, your enterprise must contain a functional Teradata server that can be accessed by the e*Gate Integrator server.*

5.3.1 Importing the TPump Sample Schema

Import the TPump sample schema to the e*Gate Integrator as follows:

1. Start **e*Gate Schema Designer**.
2. Enter the **password** associated with the user name displayed.
3. Click **Log In**.
4. In the **Open Schema on Registry Host** dialog box, click **New**.
5. In the **New Schema** dialog box, enter a name for the new schema.
6. Select **Create from Export** and click **Find**.
7. Navigate to the location of the **TPump_Sample.zip** file (on the installation CD-ROM, this file is located in *..\samples\ewteradata\..*).
8. Select the zip file and click **Open**.

   The e*Gate Schema Designer opens with the new schema.

5.3.2 Creating and Configuring the e*Ways

The TPump sample schema uses two e*Ways:

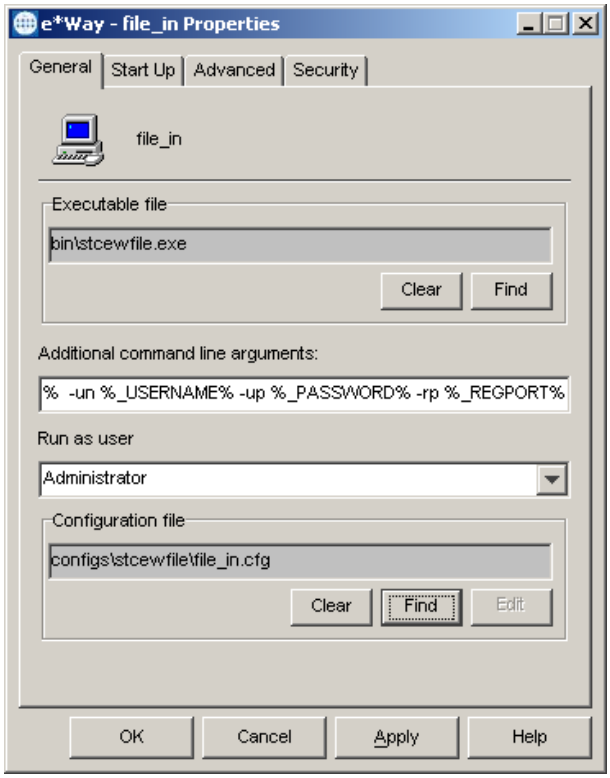
- The **file_in** (batch) e*Way, a component e*Way responsible for polling an external directory for data.
- The **tpump** e*Way, responsible for routing and transforming data within e*Gate.

This section provides instructions for creating and configuring these two e*Ways to operate within the TPump sample schema.

Configuring the file_in e*Way

1. On the e*Gate Schema Designer Navigator, click the **Components** tab.
2. Open the host on which you want to create the e*Way and select the **Control Broker** that will manage the e*Way.
3. Click the **Create a New e*Way** button.
4. Enter a name for the new e*Way (for this sample, **file_in**) and click **OK**.
5. Right-click the **file_in** e*Way and select **Properties**. The **e*Way file_in Properties** dialog box appears (see Figure 14).

Figure 14 file_in e*Way Properties



6. In the **Executable File** field, select **stcewfile.exe** (located in the *bin* directory).
7. In the **Additional Command Line Arguments** field, enter any additional command line arguments *at the end* of the existing command-line string. Do not change any of the default arguments unless you have a specific need to do so.
8. Click **New** or **Edit** (depending on whether a configuration file was previously created) beneath the **Configuration File** field. The **Configuration Editor** appears.
9. From the **Goto Section** drop-down menu on the Configuration Editor, select **Poller (inbound) settings**.
10. For the **PollDirectory** parameter, accept the default value (**c:\eGate\client\data\Teradata**), or enter the path to an existing accessible directory.

Note: *If you accept the default value for the PollDirectory parameter, you must create the* ***c:\eGate\client\data\Teradata*** *directory on the system. The trigger.dat file is placed in this directory. See* **Configuring the Trigger File** *on page 73 for more information.*

11 After setting the parameters, from the **File** menu, click **Save** to save your settings as a **.cfg** file, and click **Promote to Run Time**.

12 Click **OK** to close the **Properties** dialog box.

Important: *See* **Configuring the Trigger File** *on page 73 for more information about setting up the directory that the* ***file_in*** *e*Way polls for data. This directory must be configured as directed to successfully run the sample schema.*

Creating and Configuring the tpump e*Way

1 From the **Components** tab of the e*Gate's Navigator pane, expand the appropriate Participating Host, and select the **Control Broker**.

2 Click the **Create a New e*Way** button, enter a name for the new e*Way (for this sample, **tpump**) and click **OK**.

3 Right-click the **tpump** e*Way and select **Properties**. The **e*Way Properties** dialog box appears.

4 In the **Executable File** field, select **stceway.exe** (located in the *bin* directory).

5 In the **Additional Command Line Arguments** field, enter any additional command line arguments *at the end* of the existing command-line string. Do not change any of the default arguments unless you have a specific need to do so.

6 On the **General** tab of the **e*Way - file_in Properties** dialog box, click **New** or **Edit** under the **Configuration File** field. The **Configuration Editor** appears.

7 Set the parameters of the configuration file to accommodate your specific environment.

Note: *Configuration file parameter settings are explained in* **Multi-Mode e*Way Configuration Parameters** *on page 19.*

8 After setting the parameters, from the **File** menu, click **Save** to save your settings as a **.cfg** file, and click **Promote to Run Time.**

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

5.3.3 Creating and Configuring the Teradata ETL e*Way Connection

An e*Way Connection is the encoding of access information for an external connection. The e*Way Connection configuration file contains the setting required for connecting with Teradata.

Creating and Configuring the e*Way Connection

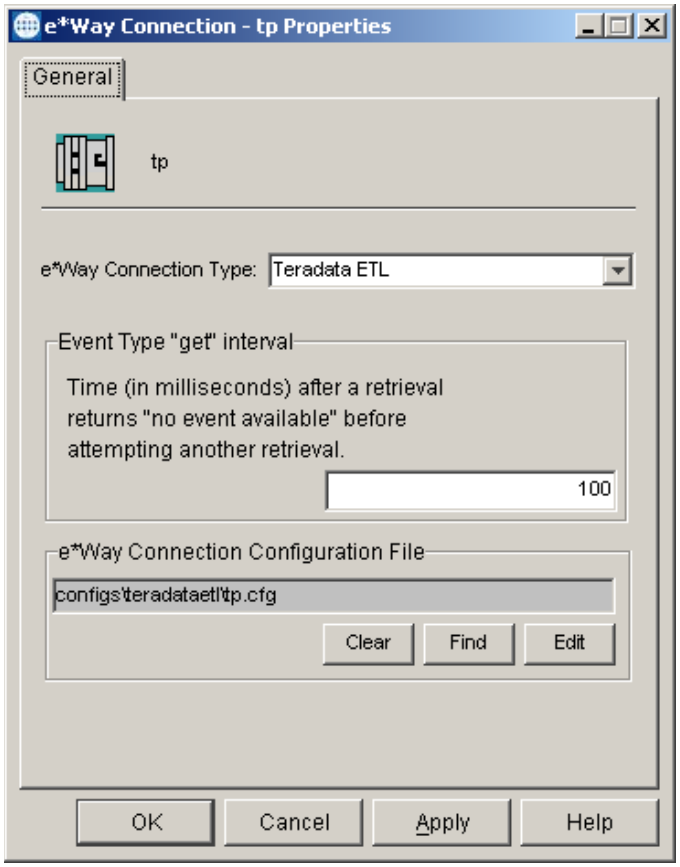
1 From the Schema Designer's Component pane, select the **e*Way Connections** folder.

2 Click the **Create a New e*Way Connection** button. The **New e*Way Connection Component** dialog box appears.

3 Enter a name for the new e*Way Connection (for this sample, **tp**) and click **OK**.

4 Double-click the new e*Way Connection. The **e*Way Connection Properties** dialog box appears (see Figure 15).

Figure 15 e*Way Connection tp Properties



5 Select **Teradata ETL** as the e*Way Connection Type.

6 Enter a value of **100** for the **Event Type "get" interval** (the default is 10000 milliseconds).

7 Click **New** or **Edit** (depending on whether a configuration file was previously created) beneath the **e*Way Connection Configuration File** field. The **Configuration Editor** appears.

8 Set the parameters of the configuration file in accordance with the instructions in Table 14. You must provide valid values for the parameters listed to run the sample

schema successfully. Accept the default settings for those parameters not listed in the Table 14.

Table 14 tp e*Way Connection Parameters

e*Way Connections - TPump Sample Schema	
e*Way Connection Name	tp
e*Way Connection Type	Teradata ETL
Connector Section - Set as directed, otherwise accept the default.	
Connector type	teradataetl
Connector class	com.stc.eways.teradataetl.TeradataETLConnector
ETL Control Section - Set as directed, otherwise accept the default.	
ETL Utility	TPump
Script File Name	The location and name of the TPump script file (for example: C:\TeradataFile\tpumpscript.scr)
Output Log File Name	The location and name of the TPump output log file (for example: C:\TeradataFile\output.out)
Teradata Error File Name	The location and name (-e option) of the TPump error file (for example: C:\TeradataFile\tperror.err)
Teradata TPump Section - Set as directed, otherwise accept the default.	
Teradata Server Alias	The alias for the name of the Teradata server for TPump (see **Teradata Server Alias** on page 32)
Teradata User Name	The user name of a valid Teradata server account
Teradata Password	The user password of a valid Teradata server account
Path to TPump Binary	The location of the TPump binary (for example: C:\Program Files\NCR\Teradata Client\bin)

Note: *Configuration file parameter settings are explained in* **Teradata ETL e*Way Connection Parameters** *on page 26.*

9 After setting the parameters, from the **File** menu, click **Save** to save your settings as a **.cfg** file, and click **Promote to Run Time**.

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

5.3.4 Creating and Configuring Event Type Definitions

The Teradata ETL e*Way uses Event Type Definitions (ETDs) to parse, validate, and (if necessary) transform Events.

To clarify, the terms associated with ETDs are defined as follows:

- An **Event** is a packet of data within e*Gate.
- An **Event Type** is a class of Events with common data structure.

The e*Gate system packages data within Events and categorizes them into Event Types. Common elements between Events define the Event Type and comprise the ETD.

The Teradata ETL e*Way uses two ETDs:

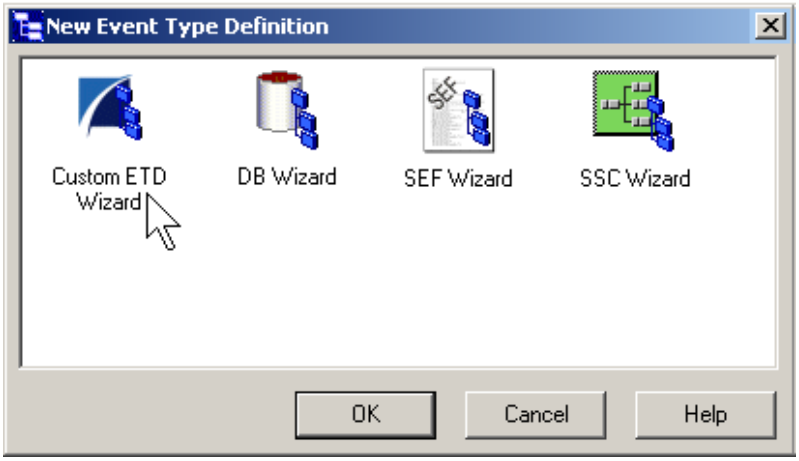
- **generic_in**, which is created using the Custom ETD wizard.
- **teradata**, which is associated with an existing .xsc file.

This section describes the procedures for creating an ETD.

Creating an ETD Using the Custom ETD Wizard

1. Click the **Event Types** folder from e*Gate's **Components** tab.
2. Click the **Create a New Event Type** button.
3. Enter the **name** of the **Event Type** (for this sample, **generic_in**) and click **OK**.
4. Double-click the **new Event Type** to open the Properties dialog box.
5. In the **Event Type Properties** dialog box click **New**. The ETD Editor appears.
6. From the **File** menu, select **New**. The **New Event Type Definition** dialog box appears (see Figure 16).

Figure 16 New Event Type Definition - Custom ETD Wizard



7. Double-click the **Custom ETD Wizard** icon. The Custom ETD Wizard appears.
8. Enter the following:
   A. A Root Node Name (for this sample, **GenericIn**)
   B. A **Package Name** (this is where the ETD Editor places the generated Java classes associated with the newly-created ETD)
9. Click **Next**, then click **Finish**. The ETD Editor now displays the generated ETD.
10. Right-click the root node (**GenericIn**) in the Event Type Definition pane.
11. From the **Add Field** sub-menu, select **As Child Node**. A node named **Field1** is added.
12. Triple-click **Field1**, and rename the node as **data**.
13. Select the **data** node. Selecting a node displays its properties in the **Properties** pane. Change the value of the **structure** property to **fixed**.

14 From the **File** menu, select **Compile and Save**, and **Promote to Run Time**.

15 Close the ETD Editor and the Event Type Properties dialog box.

Creating the Teradata Event Type Using an Existing .xsc File

1 Click the **Event Types** folder from e*Gate's **Components** tab.

2 Click the **Create a New Event Type** button, enter the name of the Event Type (for this sample, **teradata**), and click **OK**.

3 Double-click the new Event Type. The **Event Type Properties** dialog box appears.

4 Under the **Event Type Definition** field, click **Find**. Browse to and select the **.xsc** file to associate with this Event Type (for this sample, ***..\etd\teradataetl\Teradataetl.xsc***).

5 Click **Apply** and **OK**.

5.3.5 Creating Intelligent Queues

Intelligent Queues (IQs) provide the following to the e*Gate system:

- Management of the exchange of data between components within e*Gate.
- Non-volatile storage for data as it passes from one component to another.

IQs use **IQ Services** to transport data. IQ Services provide the following:

- The mechanism for moving Events between IQs.
- Handling of low-level implementation of data exchange (such as system calls to initialize or reorganize a database).

The Teradata e*Way uses the SeeBeyond JMS (Java Message Service) IQ Service.

Note: *For more information about the SeeBeyond JMS IQ Service, see the SeeBeyond JMS Intelligent Queue User's Guide.*

To creating an Intelligent Queue

1 From e*Gate's **Components** tab, expand the **host** on which you want to create the IQ, and the host's **Control Broker**, and select the **IQ Manager** (for this sample, **localhost_iqmgr**).

2 Click the **Create a New IQ** button, enter a name for the new IQ (for this sample, ***tp_iq***) and click **OK**.

3 Double-click the **new IQ**. The **IQ Properties** dialog box appears. The **Service** defaults to **STC_JMS_IQ** and cannot be changed.

4 Set the **Initialization string** and **"get" interval** as needed.

Note: *The default values are satisfactory for the implementation of the sample schema.*

5 On the **Advanced** tab, make sure that **Simple publish/subscribe** is selected under IQ behavior.

6 Click **OK**.

5.3.6 Creating Collaboration Rules

Collaboration Rules extract and process selected information from the Source Event Type according to its associated Collaboration Service.

Creating Collaboration Rules includes the following:

- Assigning the subscription and publication instance name and Event Type.
- Specifying the e*Way Connection as either the source or destination in the Collaboration.

The TPump sample schema uses two Collaboration Rules:

- **pass_thru**
- **tpump_import**

This section provides a description of the process for creating these Collaboration Rules.

Setting the Default Collaboration Rules Editor

1 From e*Gate Schema Designer's **Options** menu, select **Default Editor**.

2 Select **Java**.

Note: *Though the Default Editor can be set to either **Monk** or **Java**, the TPump sample schema requires that the Default Editor be set to **Java**.*

Creating the pass_thru Collaboration Rules

1 From e*Gate's **Components** tab, click the **Collaboration Rules** folder.

2 Click the **Create New Collaboration Rules** button.

3 Enter the **name** of the Collaboration Rules (for this sample, **pass_thru**) and click **OK**.

4 Double-click the **pass_thru** Collaboration Rules. The **Collaboration Rules Properties** dialog box appears.

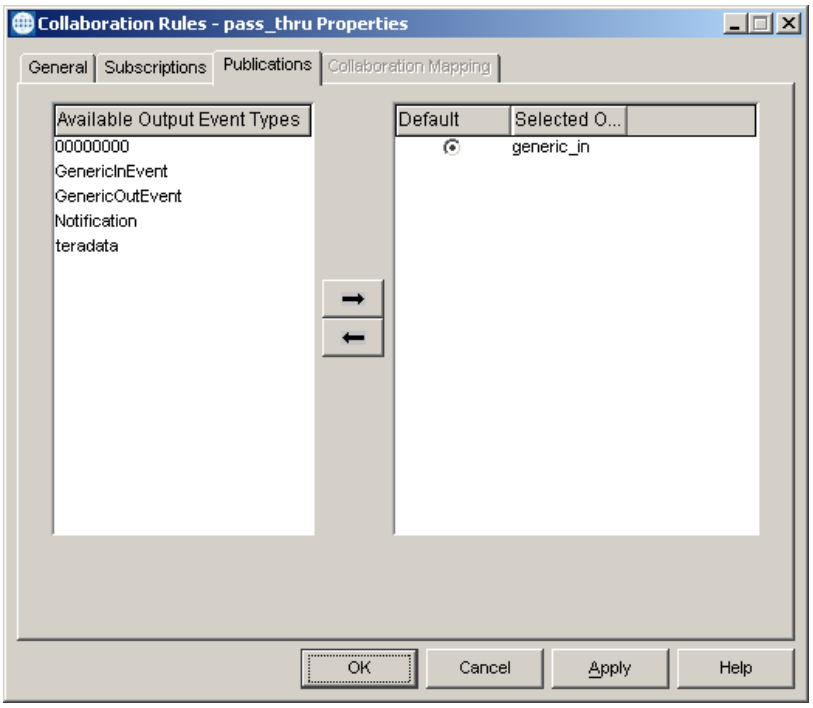
5 Select **Pass Through** as the service.

6 From the **Subscriptions** tab, select **generic_in** under **Available Input Event Types**, and click the right arrow to move it to the **Select Input Event Types**.

7 Make sure that the corresponding **Triggering Event** box is checked.

8 From the **Publications** tab, select **generic_in** under **Available Output Event Types**, and click the right arrow to move it to the **Select Output Event Types** (seeFigure 17).

Figure 17 Collaboration Rules - Pass Through - Publications



9 Make sure that the corresponding **Default** radio button is selected.

10 Click **OK**.

Creating the tpump_import Collaboration Rules

1 From the e*Gate Schema Designer's **Components** tab, click the **Collaboration Rules** folder.

2 Click the **Create New Collaboration Rules** button.

3 Enter the **name** of the Collaboration Rules (for this sample, **tpump_import**) and click **OK**.

4 Double-click the **tpump_import** Collaboration Rules. The **Collaboration Rules Properties** dialog box appears.

5 Select **Java** as the service.

6 Click the **Collaboration Mapping** tab (see Figure 18).

7 Click **Add Instance**.

8 Enter a **name** for the instance (for this sample, **out**).

9 For ETD, click **Find**, navigate to **TeradataETL.xsc** and click **Select**.

10 For Mode, select **Out**.

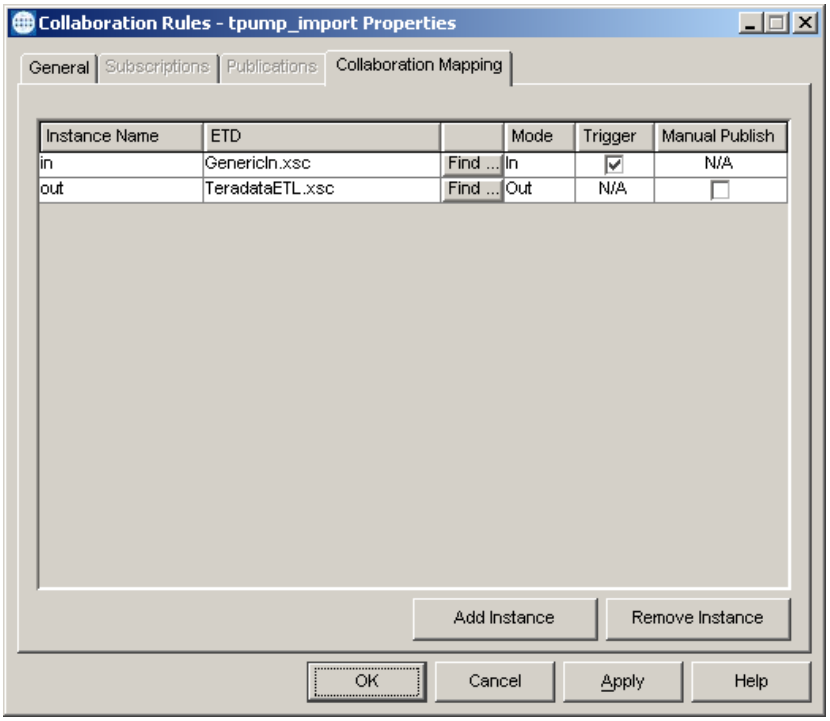
11 Click **Add Instance** again to add another instance.

12 Enter a **name** for the instance (for this sample, **in**).

13 For ETD, click **Find**, navigate to **GenericIn.xsc** and click **Select**.

14 For Mode, select **Out**. The Collaboration Mapping settings should appear similar to those in Figure 18. The order in which the instances are displayed is unimportant.

Figure 18 Collaboration Rules - Java - Collaboration Mapping



15 Click **Apply** and **OK** to close the tpump_import Collaboration Rules at this point, or proceed to the next section to create the Business Rules using the Java Collaboration Rules Editor.

5.3.7 Creating Business Rules Using the Collaboration Rules Editor

The Java Collaboration Rules Editor is the GUI used to create and modify Java Collaboration Rules. The Editor contains a Business Rules toolbar which allows the user to insert common Java programming constructs into the Collaboration.

Teradata e*Way Business Rules

The following section provides an example of how Business Rules are created using the Java Collaboration Rules Editor. The configuration settings used in the sample's rules need to be modified to those of your specific system.

Index Numbers

The Business Rules that generate the **TPump** or **FastExport** scripts require an index number for each node. A node that is used multiple times must be numbered in order of use, starting with 0. For example:

```
getout().getScript().getScriptContent().getETLCommand(0).setFIELD(0, "ACCOUNTNUMBER 1
VARCHAR(005);");
getout().getScript().getScriptContent().getETLCommand(0).setFIELD(1, "BALANCE       *
VARCHAR(009);");
```

```
getout().getScript().getScriptContent().getETLCommand(0).setIMPORT_INFILE(0, "\tester
est\fexp_in.DAT");
```

In this example, the **Field** node is used twice, so the first instance is numbered **0**, the second instance is numbered **1**, and so forth. The Import node has only one instance but is still numbered as **0**. If the **Field** node is used again in the Collaboration Rules, it must continue with the number **2**, and so on.

Semi-colon and Corresponding Text

The Collaboration Rules Editor does not enter semi-colons for you. For information regarding **corresponding text** and **semi-colon requirements** for TPump commands see the **Teradata TPump Reference**. For information regarding FastExport commands, see the **Teradata FastExport Reference**.

e*Gate Error Messages

All e*Gate **error messages** that are generated during the execution of the TPump job or FastExport job are printed to the e*Gate log file.

System Specific Settings

Many of the Business Rules used in the TPump and FastExport sample schemas must be specifically configured to your particular system. These Rules include:

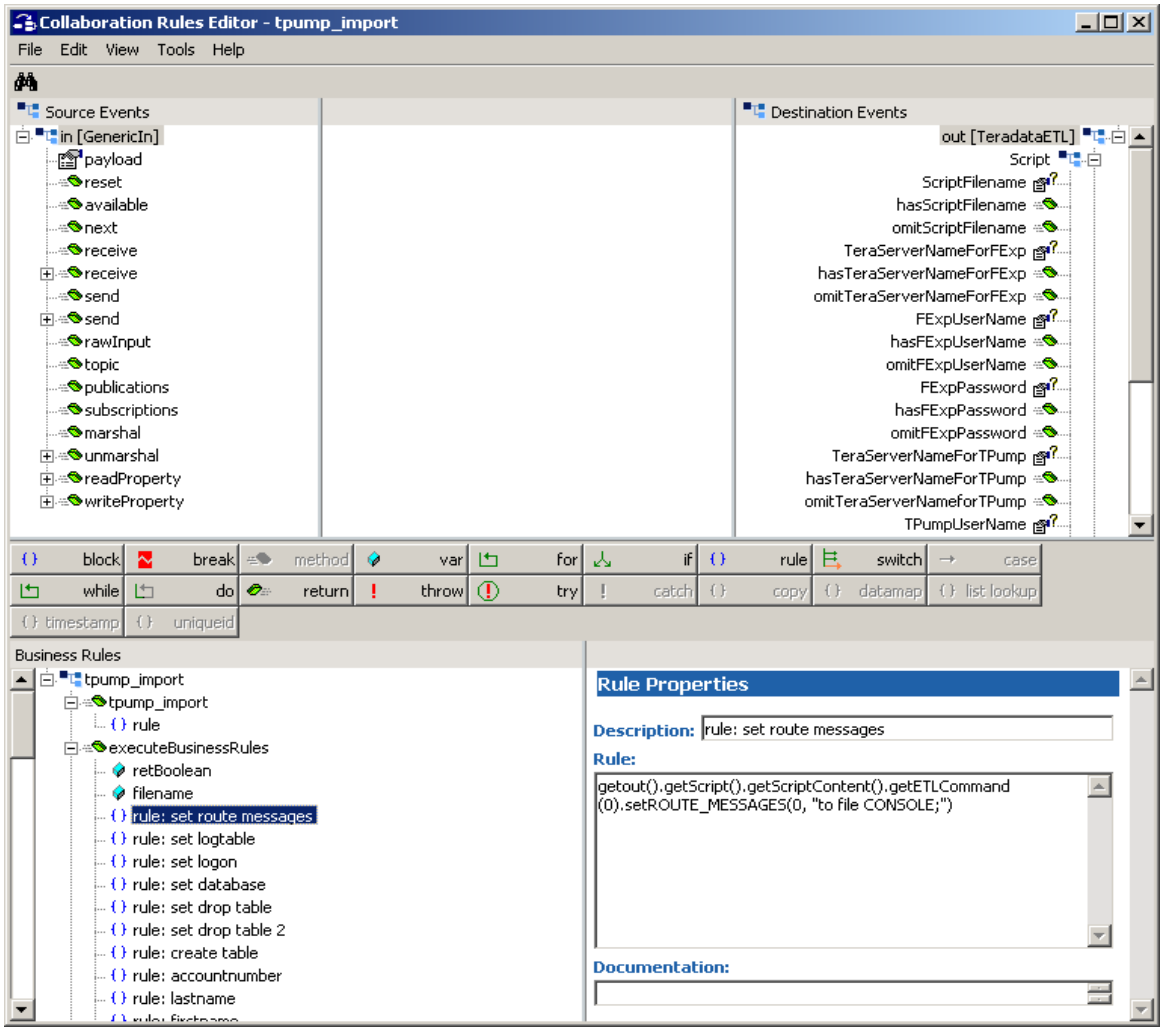
- **LOGTABLE**: The **set Logtable** rule includes journal checkpoint settings that allow for a safe restart in the event of a client or Teradata RDBMS failure.
- **DATABASE**: For the **set Database** rule, you must enter the name of your database before running the sample schema (replace the value **DNSTESTER** with your own database name).
- **IMPORT_INFILE**: For the **set Import_Infile** rule, you must enter the path of the input directory (the location of the **Input.dat** file). The **Input.dat** file, included as part of the TPump_Sample.zip file, is used to populate the database. See **Configuring the Input File** on page 73 for more information about the **Input.dat** file.
- **EXPORT_OUTFILE**: The **set Export_Outfile** rule requires the path to the of the outfile directory. This is the directory to which FastExport exports data.

Creating the tpump_import Collaboration Rules

The **tpump_import** Collaboration Rules contain Business Rules that are used to create and run the database population script. These Business Rules are created as described in the following section.

1. To open the **tpump_import Collaboration Rules**, from the e*Gate's Components tab, click the **Collaboration Rules** folder.
2. From the e*Gate's Editor pane, double-click **tpump_import**. The **tpump_import Collaboration Rules Properties** dialog box appears.
3. From the General tab, click **New** or **Edit** under the Collaboration Rules field. The **Collaboration Rules Editor** appears (see **Figure 19 on page 68**).

Figure 19 Collaboration Rules Editor- tpump_import



4 Expand the Collaboration Rules Editor to the maximum size for optimum viewing, expanding the Source and Destination Events as well.

5 All user-defined variables and rules are added as part of **executeBusinessRules()**.

Initializing the filename Variable

The initialization of the **filename** variable in the schema allows the script file name to appear in the e*Gate log file.

1 Under **executeBusinessRules**, click **retBoolean**.

2 Click the **variable** (**var**) button.

3 In the Variable Properties window, enter **filename** as the **name** and **description** of this variable, and accept the default type (**String**).

4 Enter **null** as the **Initial Value**.

Adding a Rule

1 With the **filename** variable highlighted, click the **rule** button on the Business Rules toolbar.

2 In the **Description** field of the **Rule Properties** pane, enter **rule: set route messages**.

3 Under **Destination Events**, navigate to **Script > ScriptContent > ETLCommand > ROUTE_MESSAGES**.

4 Click-and-drag **ROUTE_MESSAGES** to the **Rule** field in the **Rule Properties** pane.

5 In the **Select Repetition** dialog box, click **OK** to accept the default values.

6 Place the cursor in the final parentheses in the **Rule** field and enter the corresponding text for this rule in quotation marks (for this example, 0, "messages to file CONSOLE;").

7 If necessary, enter a semi-colon after the final right parentheses of the rule.

8 To view the java code in the Business Rules pane, from the **View** menu, select **Display Code**. The completed **set route message** rule appears as follows:

```
rule: set route message:
      getout().getScript().getScriptContent().getETLCommand(0).setROUTE_MESSAGE(0, "messages
      to file CONSOLE;");
```

Creating Business Rules

The following section details the creation of a few of the Business Rules for the tpump_import sample schema.

1 The **set logtable** rule is created by dragging-and-dropping **LOGTABLE** under **out > Script > ScriptContent > ETLCommand**, in the Destination Events pane to the Rule Properties, Rule field. When prompted for the **Select Repetition Instance**, click **OK** to accept the default values. Place the curser in the last set of parentheses and enter the journal checkpoint settings (for this example, **"TLddNT2I;"**) to create the following code:

```
getout().getScript().getScriptContent().getETLCommand(0).setLOGTABLE(0, "TLddNT2I;")
```

2 The **set logon** rule is created by dragging-and-dropping **LOGON** under **out > Script > ScriptContent > ETLCommand**, in the Destination Events pane to the Rule Properties, Rule field. When prompted for the **Select Repetition Instance**, click **OK** to accept the default values. Place the curser in the last set of parentheses and type **";"** to create the following code:

```
getout().getScript().getScriptContent().getETLCommand(0).setLOGON(0, ";")
```

3 To create the **set database** rule, drag-and-drop **DATABASE** under **out > Script > ScriptContent > SQLStatement**, in the Destination Events pane to the Rule Properties, Rule field. When prompted for the **Select Repetition Instance**, click **OK** to accept the default values. Place the curser in the last set of parentheses and type **"DNSTESTER;"** to create the following code:

```
getout().getScript().getScriptContent().getSQLStatement(0).setDATABASE(0, "DNSTESTER;")
```

In addition to the script generation command rules, the sample schema also contains rules using the **filename** command (prints the name of the script file to the e*Gate log file), and the **startTPumpJob** command (starts the TPump job).

The Collaboration Rules conclude with an **if** statement that defines the TPump log files to which the Collaboration prints error messages generated during the TPump job.

The completed Java Business Rules appears as shown in **Figure 20 on page 70**.

Figure 20 Business Rules - tpump_import

```
tpump_import : public class tpump_import extends tpump_importBase implements JCollaboratorExt
  tpump_import : public  tpump_import()
    { } rule : super();
  executeBusinessRules : public boolean executeBusinessRules() throws Exception
    retBoolean : boolean retBoolean = true;
    filename : String filename = null;
    { } rule: set route messages : getout().getScript().getScriptContent().getETLCommand(0).setROUTE_MESSAGES(0, "to file CONSOLE;");
    { } rule: set logtable : getout().getScript().getScriptContent().getETLCommand(0).setLOGTABLE(0, "TLddNT2J;");
    { } rule: set logon : getout().getScript().getScriptContent().getETLCommand(0).setLOGON(0, ";");
    { } rule: set database : getout().getScript().getScriptContent().getSQLStatement(0).setDATABASE(0, "DNSTESTER;");
    { } rule: set drop table : getout().getScript().getScriptContent().getSQLStatement(0).setDROP_TABLE(0, "TBL1T;");
    { } rule: set drop table 2 : getout().getScript().getScriptContent().getSQLStatement(0).setDROP_TABLE(1, "TLNT2ERR;");
    { } rule: create table : getout().getScript().getScriptContent().getSQLStatement(0).setCREATE_TABLE(0, "TBL1T");
    { } rule: accountnumber : getout().getScript().getScriptContent().setGeneric(0, "(ACCOUNTNUMBER    INTEGER NOT NULL,");
    { } rule: lastname : getout().getScript().getScriptContent().setGeneric(1, " LASTNAME        VARCHAR(25),");
    { } rule: firstname : getout().getScript().getScriptContent().setGeneric(2, " FIRSTNAME       VARCHAR(25),");
    { } rule: address : getout().getScript().getScriptContent().setGeneric(3, " ADDRESS         VARCHAR(30),");
    { } rule: city : getout().getScript().getScriptContent().setGeneric(4, " CITY           VARCHAR(20),");
    { } rule: state : getout().getScript().getScriptContent().setGeneric(5, " STATE          CHAR(2),");
    { } rule: zipcode : getout().getScript().getScriptContent().setGeneric(6, " ZIPCODE        CHAR(5),");
    { } rule: balance : getout().getScript().getScriptContent().setGeneric(7, " BALANCE        DECIMAL(9,2))");
    { } rule: primary key : getout().getScript().getScriptContent().setGeneric(8, "UNIQUE PRIMARY INDEX (ACCOUNTNUMBER);");
    { } rule: set begin load : getout().getScript().getScriptContent().getETLCommand(0).setBEGIN_LOAD(0, "SESSIONS 2");
    { } rule: errortable : getout().getScript().getScriptContent().setGeneric(9, " ERRORTABLE TLNT2ERR");
    { } rule: pack : getout().getScript().getScriptContent().setGeneric(10, " PACK 10");
    { } rule: errlimit : getout().getScript().getScriptContent().setGeneric(11, " ERRLIMIT 5");
    { } rule: checkpoint : getout().getScript().getScriptContent().setGeneric(12, " CHECKPOINT 1");
    { } rule: tenacity : getout().getScript().getScriptContent().setGeneric(13, " TENACITY 2");
    { } rule: robust : getout().getScript().getScriptContent().setGeneric(14, " ROBUST OFF;");
    { } rule: set layout : getout().getScript().getScriptContent().getETLCommand(0).setLAYOUT(0, "LAY1A;");
    { } rule: set field : getout().getScript().getScriptContent().getETLCommand(0).setFIELD(0, "ACCOUNTNUMBER 1 VARCHAR(005);");
    { } rule: set field 1 : getout().getScript().getScriptContent().getETLCommand(0).setFIELD(1, "LASTNAME     * VARCHAR(025);");
    { } rule: set field 2 : getout().getScript().getScriptContent().getETLCommand(0).setFIELD(2, "FIRSTNAME    * VARCHAR(025);");
    { } rule: set field 3 : getout().getScript().getScriptContent().getETLCommand(0).setFIELD(3, "ADDRESS      * VARCHAR(030);");
    { } rule: set field 4 : getout().getScript().getScriptContent().getETLCommand(0).setFIELD(4, "CITY         * VARCHAR(020);");
    { } rule: set field 5 : getout().getScript().getScriptContent().getETLCommand(0).setFIELD(5, "STATE        * VARCHAR(002);");
    { } rule: set field 6 : getout().getScript().getScriptContent().getETLCommand(0).setFIELD(6, "ZIPCODE      * VARCHAR(005);");
    { } rule: set field 7 : getout().getScript().getScriptContent().getETLCommand(0).setFIELD(7, "BALANCE      * VARCHAR(009);");
    { } rule: set dml label : getout().getScript().getScriptContent().getETLCommand(0).setDML_LABEL(0, "LABELA IGNORE DUPLICATE ROWS IGNORE
                    MISSING ROWS IGNORE EXTRA ROWS;");
    { } rule: set insert : getout().getScript().getScriptContent().getSQLStatement(0).setINSERT(0, "INTO TBL1T VALUES (");
    { } rule: col accountnumber : getout().getScript().getScriptContent().setGeneric(15, " ACCOUNTNUMBER = :ACCOUNTNUMBER");
    { } rule: col lastname : getout().getScript().getScriptContent().setGeneric(16, " ,LASTNAME    = :LASTNAME");
    { } rule: col firstname : getout().getScript().getScriptContent().setGeneric(17, " ,FIRSTNAME   = :FIRSTNAME");
    { } rule: col address : getout().getScript().getScriptContent().setGeneric(18, " ,ADDRESS     = :ADDRESS");
    { } rule: col city : getout().getScript().getScriptContent().setGeneric(19, " ,CITY        = :CITY");
    { } rule: col state : getout().getScript().getScriptContent().setGeneric(20, " ,STATE       = :STATE");
    { } rule: col zipcode : getout().getScript().getScriptContent().setGeneric(21, " ,ZIPCODE     = :ZIPCODE");
    { } rule: col balance : getout().getScript().getScriptContent().setGeneric(22, " ,BALANCE     = :BALANCE);");
    { } rule: set import infile : getout().getScript().getScriptContent().getETLCommand(0).setIMPORT_INFILE(0, "\\\\tester\\tmp\\test\\IMPORT.DAT");
    { } rule: format : getout().getScript().getScriptContent().setGeneric(23, " FORMAT VARTEXT '|' NOSTOP");
    { } rule: layout : getout().getScript().getScriptContent().setGeneric(24, " LAYOUT LAY1A");
    { } rule: apply : getout().getScript().getScriptContent().setGeneric(25, " APPLY LABELA;");
    { } rule: set end load : getout().getScript().getScriptContent().getETLCommand(0).setEND_LOAD(0, ";");
    { } rule: set logoff : getout().getScript().getScriptContent().getETLCommand(0).setLOGOFF(0, ";");
    { } rule: generate tpump script : filename=getout().getScript().generateScript();
    { } rule: filename : EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMATION,">>>>> File name: " + filename);
    { } rule: run : getout().getRuntime().getTPumpRun().startTPumpJob();
  if: return is not -1 : if (!getout().getRuntime().getJobOutput().getErrorCode().equals("-1"))
    { } then :
      for: errorline : for( int jj=0;jj<getout().getRuntime().getErrorOutput().countErrorLine();jj++)
        { } rule: get errorcode : EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMATION,">>>>>
                    Errcode(" + jj + "): " + getout().getRuntime().getErrorOutput().getErrorCode(jj));
        { } rule: get errortext : EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMATION,">>>>>
                    Errtext(" + jj + "): " + getout().getRuntime().getErrorOutput().getErrorText(jj));
        { } rule: get recordlinenum : EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMATION,">>>>>
                    ErrRecordNum(" + jj + "): " + getout().getRuntime().getErrorOutput().getRecordLineNumber(jj));
      for: outline : for( int ii=0;ii<getout().getRuntime().getJobOutput().getRows().countOutLine();ii++)
        { } rule: get outline : EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMATION,">>>>> Outline:
                    " + getout().getRuntime().getJobOutput().getRows().getOutLine(ii));
    { } else: error : else
      { } rule: get error string : EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMATION,">>>>> Error:
                    <" + getout().getRuntime().getJobOutput().getErrorString() + ">");
  return : return retBoolean;
  userInitialize : public void userInitialize()
  userTerminate : public void userTerminate()
```

Note: *The tpump_import sample Business Rules are displayed as wrapped due to size restrictions. The editor does not wordwrap code in the Business Rules pane.*

5.3.8 Creating Collaborations

Collaborations perform for the following tasks:

- Receive and process Events
- Forward output to other e*Gate components or external sources

Collaborations consist of a Subscriber, which listens for Events of a known type (sometimes from a given source) and a Publisher, which distributes the transformed Event to a specified recipient.

The TPump sample schema contains of two Collaborations:

- **Creating the file_in Collaboration** on page 71
- **The tpump Collaboration** on page 72

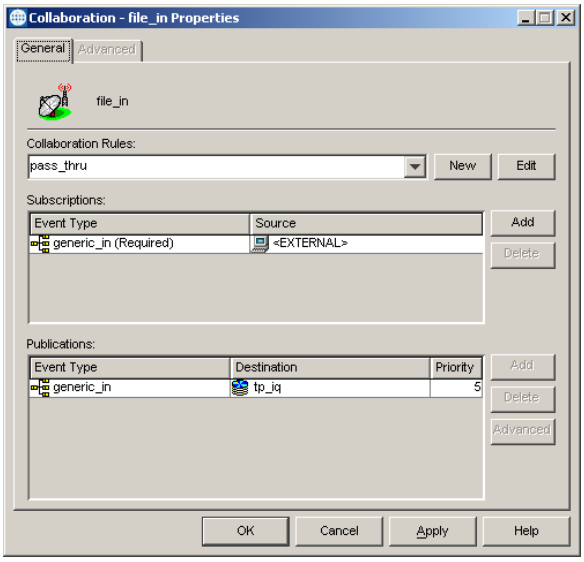
Creating the file_in Collaboration

The **file_in** Collaboration receives data from an external source and forwards it to an IQ (**tp_iq**).

Creating the file_in Collaboration

1. From e*Gate's **Components** tab, expand the **Control Broker**, and select the **file_in** e*Way.
2. Click the **Create a New Collaboration** button.
3. Enter a name for the new Collaboration (for this sample, **file_in**) and click **OK**.
4. Double-click the new Collaboration. The **Collaboration Properties** dialog box appears (see Figure 21).

Figure 21 Collaboration Properties - file-in



5 Under **Subscriptions**, click **Add**.

6 Select **generic_in** as the **Event Type**.

7 Select **<EXTERNAL>** as the **Source**.

8 Under **Publications**, click **Add**.

9 Select **generic_in** as the **Event Type**.

10 Select **tp_iq** as the **Destination**.

11 Click **Apply** and **OK**.

The tpump Collaboration

The tpump Collaboration, associated with the **tpump** e*Way and the **tpump_import** Collaboration Rules, receives data from the IQ and forwards it to Teradata.

Creating the tpump Collaboration

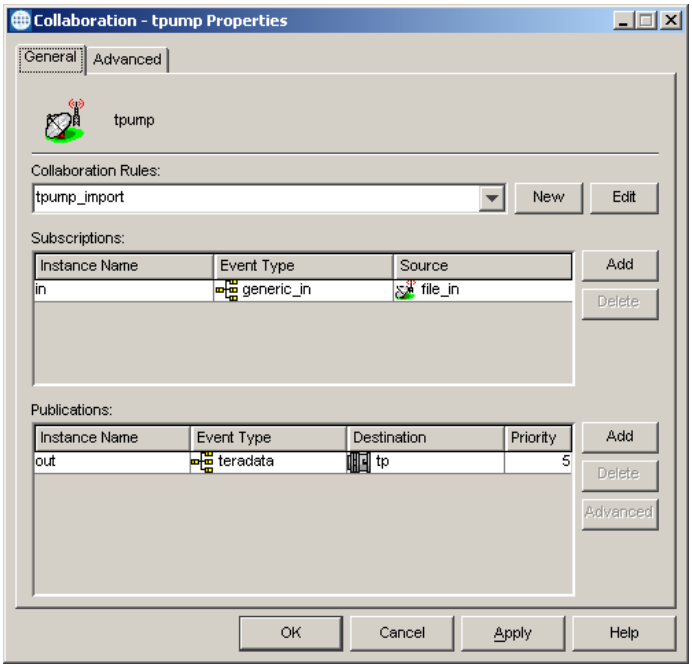
1 From e*Gate's **Components** tab, expand the **Control Broker**, and select the **tpump** e*Way.

2 Click the **Create a New Collaboration** button.

3 Enter a name for the new Collaboration (for this sample, **tpump**) and click **OK**.

4 Double-click the new Collaboration. The **Collaboration Properties** dialog box appears (see Figure 22).

Figure 22 Collaboration Properties - tpump



5 From the **Collaboration Rules** drop-down list box, select **tpump_import**.

6 Under **Subscriptions**, click **Add**.

7 Under **Instance Name**, enter **in**.

8 Select **generic_in** as the **Event Type**.

9 Select **file_in** as the **Source**.

10 Under **Publications**, click **Add**.

11 Under **Instance Name**, enter **out**.

12 Select **teradata** as the **Event Type**.

13 Select **tp** as the **Destination**.

14 Click **Apply** and **OK**.

5.3.9 Configuring the Trigger File

Copy the file that triggers the beginning of the operation (**trigger.dat**) from the **TPump_Sample.zip** file to the location you specified in the e*Way.

Do this as follows:

1 Using WinZip, navigate to the location of the **TPump_Sample.zip** file (on the installation CD, this file is located in *..\samples\ewteradata\..*). Extract the zip file to a temporary directory.

2 Copy the file, **trigger.dat** from the extracted TPump_Sample files, to the location defined as the file_in e*Way's **PollDirectory** parameter (see **Configuring the file_in e*Way** on page 58**The file_in e*Way** on page 75).

5.3.10 Configuring the Input File

Copy the input file (**input.dat**) from the .zip file to a location you specified in the Collaboration.

Do this as follows:

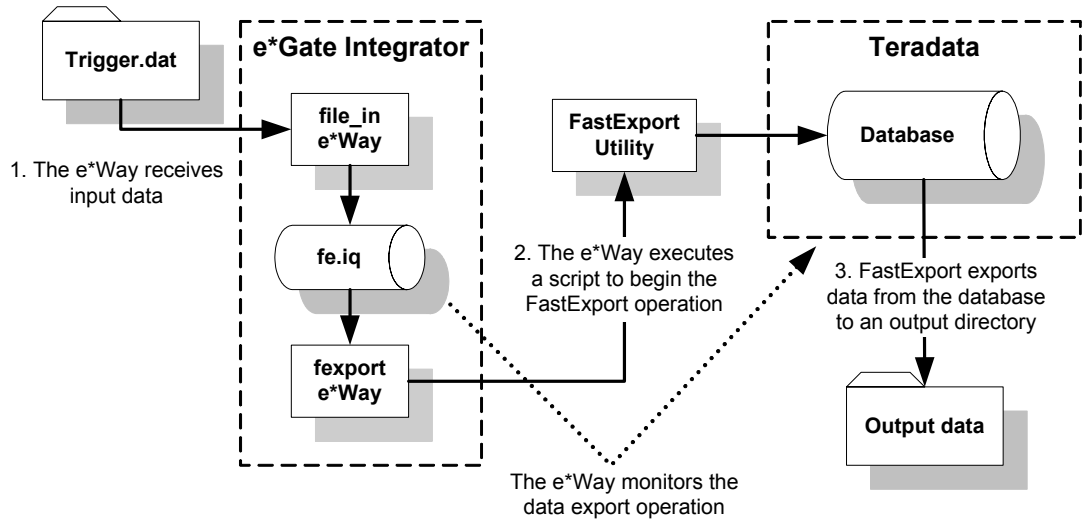
1 Using WinZip, navigate to the location of the **TPump_Sample.zip** file (on the installation CD-ROM, this file is located in *..\samples\ewteradata\..*). Extract TPump_Sample to a temporary file.

2 Copy the **input.dat** file from the extracted TPump_Sample files, to the location specified in the **set Import** Business Rule of the tpump_import Collaboration Rules (see **Creating the tpump_import Collaboration Rules** on page 67).

5.4 The FastExport Sample Schema

The FastExport sample schema, included with the Teradata e*Way, demonstrates how to configure e*Gate to use the FastExport utility to quickly transfer data from the Teradata Relational Database Management System (RDBMS) to an external directory (see Figure 23).

Figure 23 Teradata e*Way - FastExport Sample Schema



The schema's components are nearly complete once the sample schema is imported into the e*Gate Schema Designer. The configuration settings must to be modified for your specific system.

5.4.1 Importing the FastExport Sample Schema

To import the FastExport sample schema to e*Gate, do the following:

1. Start **e*Gate Schema Designer**.
2. Enter the **password** associated with the user name displayed.
3. Click **Log In**.
4. In the **Open Schema on Registry Host** dialog box, click **New**.
5. In the **New Schema** dialog box, enter a name for the new schema.
6. Select **Create from Export** and click **Find**.
7. Navigate to the location of the **FExp_Sample.zip** file (on the installation CD-ROM, this file is located in *..\samples\ewteradata\..*).
8. Select the zip file and click **Open**.

   The e*Gate Schema Designer opens under your new schema.

Note: *Schemas may also be imported from the e*Gate Schema Designer's File menu by selecting New Schema.*

The following sections describes the e*Way components for the FastExport schema. For more information on how to use the e*Gate Integrator to create these components, refer to **The TPump Sample Schema** on page 56, or see the *e*Gate Integrator User's Guide*. To create the components of the FastExport sample schema manually see the following:

- **Creating the e*Ways** on page 75
- **Creating the e*Way Connection** on page 76
- **Creating Event Type Definitions** on page 77
- **Creating Intelligent Queues** on page 77
- **Creating Collaboration Rules** on page 78
- **Creating the Business Rules** on page 78
- **Creating Collaborations** on page 80
- **Executing the Schema** on page 82

Note: *To run the FastExport sample schema, your enterprise must contain a functional Teradata server that can be accessed by the e*Gate Integrator server.*

5.4.2 Creating the e*Ways

The FastExport sample schema uses two e*Ways:

- The **file_in** e*Way, a component e*Way that subscribes to an external directory for data.
- The **fexport** e*Way, a multi-mode e*Way that routes and transforms data within e*Gate.

Any system specific parameters associated with the e*Way must be configured to accommodate for your specific environment.

The file_in e*Way

The **file_in e*Way** uses the **stcewfile.exe** executable selected in the e*Way's properties file. The configuration parameters are set as shown in Table 15.

Table 15 file_in e*Way Parameters

file_in e*Way - Configuration File Parameters	
General settings - Set as directed, otherwise accept default.	
AllowIncoming	YES
AllowOutgoing	NO
PerformanceTesting	NO
Poller (inbound) settings - Set as directed, otherwise accept default.	
PollDirectory	c:\eGate\client\data\Teradata
InputFileMask	trigger.dat

Table 15 file_in e*Way Parameters

file_in e*Way - Configuration File Parameters	
PollMilliseconds	1000
RemoveEOL	YES
MultipleRecordsPerFile	YES
MaxBytesPerLine	4096
BytesPerLineIsFixed	NO
File Records Per eGate Event	1
Performance Testing - Set as directed, otherwise leave as default.	
Performance Testing	100
InboundDuplicates	1

When the configuration settings are complete, save the configuration file, and promoted to run time. From the e*Way Properties Start tab, make sure that **Start automatically** is selected.

The fexport e*Way

The **fexport e*Way** uses the **stceway.exe** executable selected in the e*Way's properties file. The configuration parameters are saved with the default settings. From the e*Way Properties Start tab, make sure that **Start automatically** is selected.

5.4.3 Creating the e*Way Connection

One e*Way Connection, named **fe**, is created for the FastExport sample schema.

The fe e*Way Connection

The **fe e*Way Connection** uses the Teradata ETL e*Way Connection Type, selected in the e*Way Connection's properties file. The configuration parameters are set as shown in Table 16. Accept the default settings for those parameters not listed in the table. Accept the default settings for those parameters not listed in the table.

Table 16 fe e*Way Connection Parameters

e*Way Connections - FastExport Sample Schema	
e*Way Connection Name	fe
e*Way Connection Type	Teradata ETL
Connector Section - Set as directed, otherwise accept the default.	
Connector type	teradataetl
Connector class	com.stc.eways.teradataetl.TeradataETLConnector
ETL Control Section - Set as directed, otherwise accept the default.	
ETL Utility	FastExport

Table 16 fe e*Way Connection Parameters

e*Way Connections - FastExport Sample Schema	
Script File Name	The location and name of the FastExport script file (for example: C:\TeradataFile\expexport.scr)
Output Log File Name	The location and name of the FastExport output file (for example: C:\TeradataFile\fexpOutput.out)
Teradata Error File Name	The location and name (-e option) of the FastExport error file (for example: C:\TeradataFile\fexperror.err)
Teradata FastExport Section - Set as directed, otherwise accept the default.	
Teradata Server Alias	The alias for the name of the Teradata server for FastExport (see **Teradata Server Alias** on page 29)
Teradata User Name	The user name of a valid Teradata server account
Teradata Password	The user password of a valid Teradata server account
Path to FastExport Binary	The location of the FastExport binary (for example: C:\Program Files\NCR\Teradata Client\bin)

Save the configuration file and promote to run time.

5.4.4 Creating Event Type Definitions

Two Event Types are used by the FastExport sample schema:

- **generic_in**, which uses **GenericIn.xsc** as the ETD (see **Creating an ETD Using the Custom ETD Wizard** on page 62).
- **teradata**, which uses **TeradataETL.xsc** as the ETD (see **Creating the Teradata Event Type Using an Existing .xsc File** on page 63).

5.4.5 Creating Intelligent Queues

One IQ is created for the FastExport sample schema. To create the JMS IQ do the following:

1. From e*Gate's Navigator pane, expand and select the Control Broker. Create an IQ Manager if one does not already exists, using the **Create a New IQ Manager** button on the Tool Palette, or select the current IQ Manager.
2. Open the IQ Manager Properties and select SeeBeyond JMS as the IQ Manager Type.
3. Click **New** under the Configuration file field, save the default file and promote to run time.
4. Select the **IQ Manager** in the Navigator pane and click the **Create a New IQ** button.
5. Name the IQ (for this sample, fe_iq) and click **OK**. Open the new IQ's Properties dialog box. The Service defaults to STC_JMS_IQ. Click **Apply** and **OK**.

5.4.6 Creating Collaboration Rules

Two Collaboration Rules are created for the FastExport sample:

- **pass_thru**, using the **Pass Through** Service.
- **fexp_export**, using the **Java** Service.

Creating the pass_thru Collaboration Rules

For directions on how to create the pass_thru Collaboration Rules see **Creating the pass_thru Collaboration Rules** on page 64.

Creating the fexp_export Collaboration Rules

To create the **fexp_export** Collaboration Rules do the following:

1. Select **Collaboration Rules** in the e*Gate Navigator pane. Click the **Create New Collaboration Rules** button and name the New Collaboration Rules (for this sample, **fexp_export**).
2. Double-click the new Collaboration Rules. The Collaboration Rules Properties dialog box appears.
3. From the General tab, select **Java** as the Service.
4. From the Collaboration Mapping tab, use the **Add Instance** button to create two instances and enter the values as shown in Figure 24.

Figure 24 Collaboration Mapping - fexp_export

Instance Name	ETD		Mode	Trigger	Manual Publish
out	TeradataETL.xsc	Find ...	Out	N/A	☐
in	GenericIn.xsc	Find ...	In	☑	N/A

5. From the General tab, click **OK** to close the Collaboration Rules Properties dialog box, or click **New** (or **Edit**) to open the Collaboration Rules editor and create the Business Rules.

5.4.7 Creating the Business Rules

The **fexp_export** Collaboration Rules contain Business Rules used to create and run the FastExport data export script. Business Rules for the fexp_export Collaboration Rules are created using the Collaboration Rules Editor.

For more information on creating Business Rules for the Teradata e*Way see **Teradata e*Way Business Rules** on page 66.

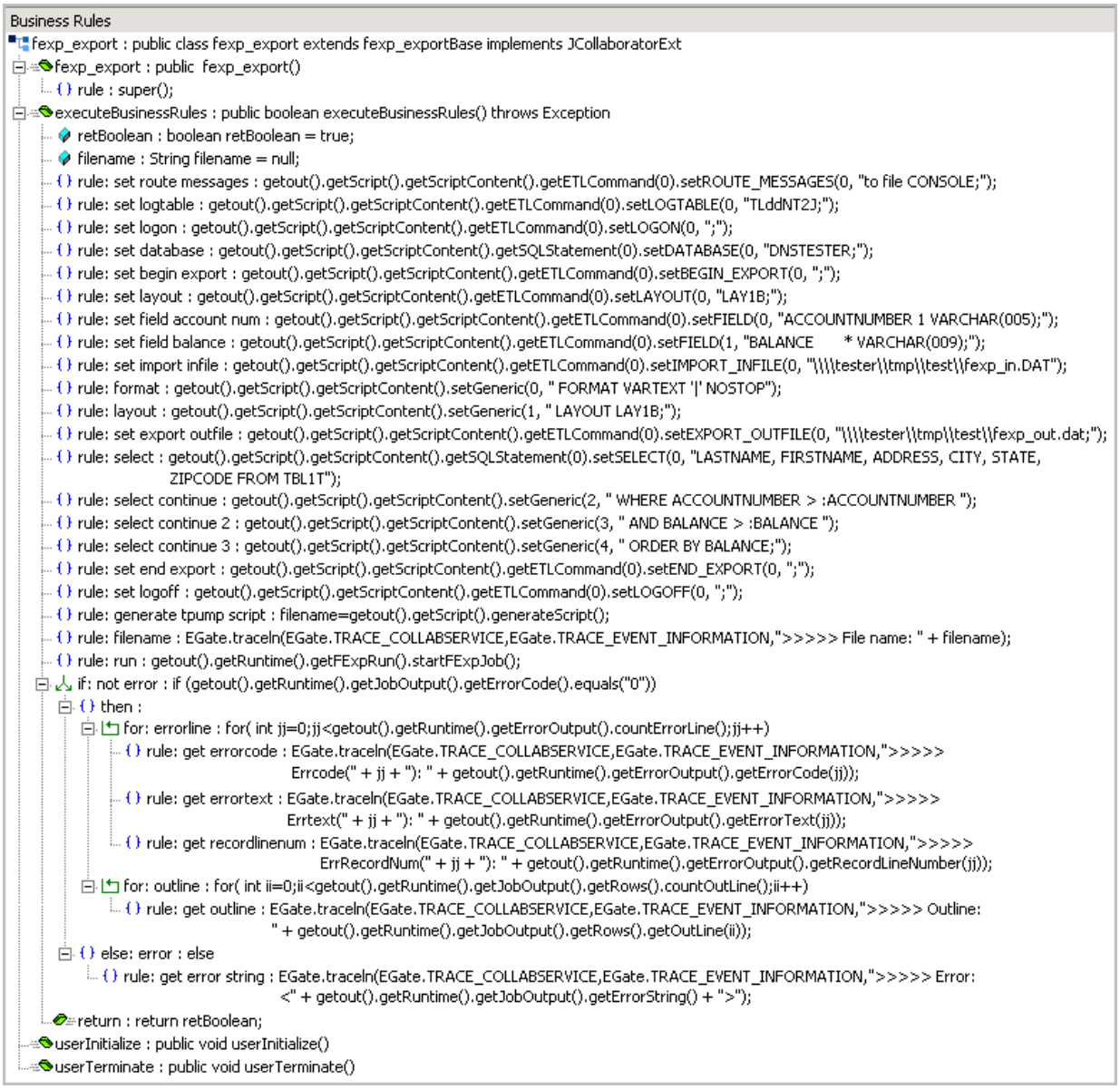
To create the fexp_export Business Rules using the Collaboration Rules Editor do the following:

1. From the General tab of the **fexp_export** Collaboration Rules Properties dialog box, click **New** (or **Edit**) under the Collaboration Rules field. The **Collaboration Rules Editor** appears

2 Expand the Collaboration Rules Editor to the maximum size for optimum viewing, expanding the Source and Destination Events as well. All user-defined variables and rules are added as part of **executeBusinessRules()**.

3 Create the Business Rules as shown in Figure 25.

Figure 25 Collaboration Rules Editor - fexp_export Business Rules



Note: *The FExp_export sample Business Rules are displayed as wrapped due to size restrictions. The editor does not wordwrap code in the Business Rules pane.*

4 When the Business Rules are complete, save and compile the file. If the file fails to compile use the Compiler window to find any errors in the code. Correct any errors and compile again until all the errors have been corrected.

5 Save and promote the completed Collaboration Editor Project.

For more information on using the Collaboration Rules Editor and creating Business Rules see the *e*Gate Integrator User's Guide*.

5.4.8 Creating Collaborations

Two Collaborations are created for the FastExport sample schema:

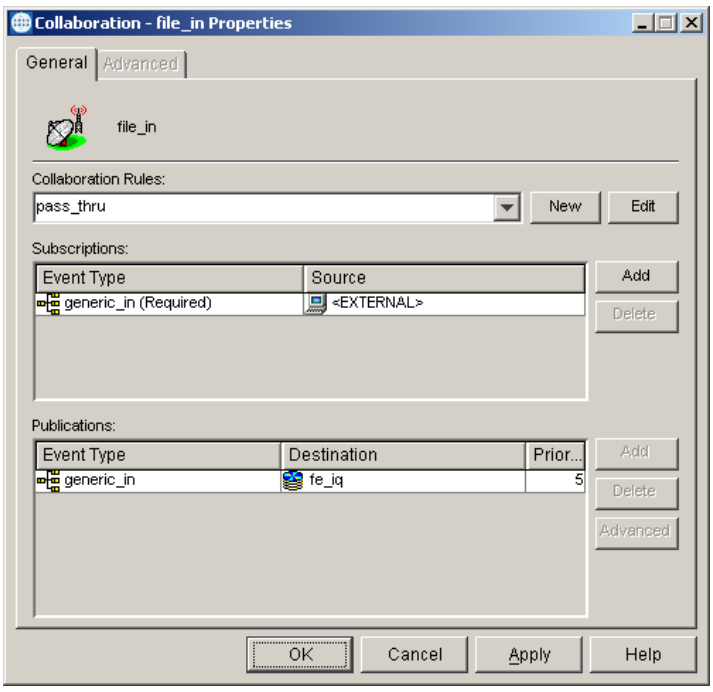
- **file_in**
- **fexport**

Creating the file_in Collaboration

To create the **file_in Collaboration** do the following:

1. From the e*Gate Navigator pane, select the **file_in** e***Way** and click the **Create a New Collaboration** button on the Tool Palette.
2. Name the Collaboration (for this sample, **file_in**).
3. From the e*Gate Editor pane, double-click the new Collaboration. The **Collaboration Properties** dialog box appears.
4. Configure the **file_in** Collaboration Properties to match those in Figure 26.

Figure 26 Collaboration Properties - file_in



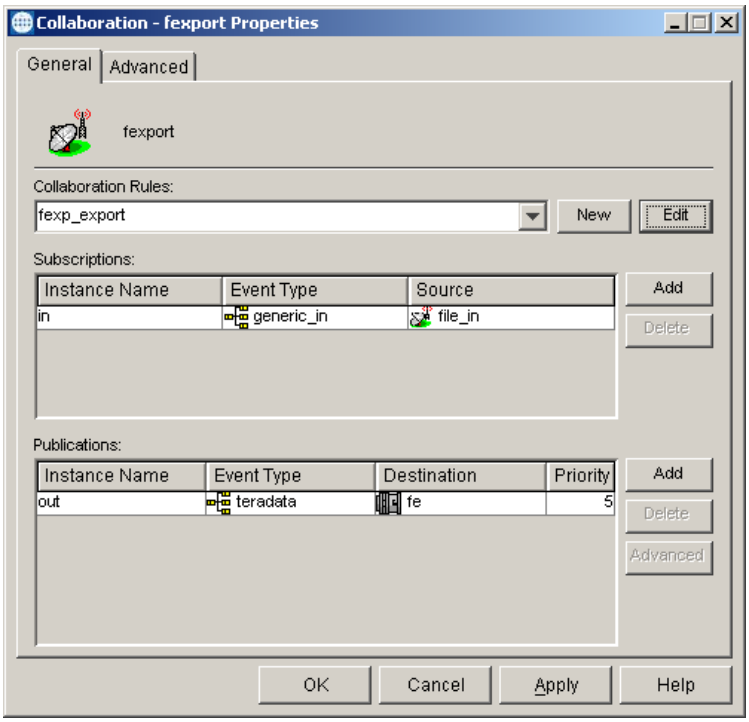
5. When completed, click **Apply** and **OK**.

Creating the fexport Collaboration

To create the **fexport Collaboration** do the following:

1. From the e*Gate Navigator pane, select the **file_in e*Way** and click the **Create a New Collaboration** button on the Tool Palette.
2. Name the Collaboration (for this sample, **file_in**).
3. From the e*Gate Editor pane, double-click the new Collaboration. The **Collaboration Properties** dialog box appears.
4. Configure the **fexport** Collaboration Properties to match those in Figure 27

Figure 27 Collaboration Properties - fexport



5. When completed, click **Apply** and **OK**.

5.4.9 Configuring the Trigger File

Copy the file that triggers the beginning of the operation (**trigger.dat**) from the .zip file to the location you specified in the e*Way.

Do this as follows:

1. Using WinZip, navigate to the location of the **FExp_Sample.zip** file (on the installation CD, this file is located in ..*samples**ewteradata*\..). Extract FExp_Sample to a temporary directory.
2. Copy the file, **trigger.dat** from the extracted TPump_Sample files, to the location defined as the file_in e*Way's **PollDirectory** parameter (see **The file_in e*Way** on page 75).

5.4.10 Configuring the Input File

Copy the input file (**fexp_in.dat**) from the .zip file to the location specified in the Collaboration Rules.

To copy the input file do the following:

1. Using WinZip, navigate to the location of the **FExp_Sample.zip** file (on the installation CD-ROM. This file is located in ..*samples**ewteradata*\..). Extract **FExp_Sample** to a temporary file.
2. Copy the **fexp_in.dat** file from the extracted FExp_Sample files, to the location specified in the **set Import** Business Rule of the **fexp_export Collaboration Rules** (see **Creating the Business Rules** on page 78).

5.5 Executing the Schema

Running the Schema

Run the TPump or FastExport sample schema as follows:

1. At the command prompt, enter the following:

```
stccb -rh hostname -rs schemaname -un username -up user password
    -ln hostname_cb
```

Note: *Substitute the italicized values with the specific values for your schema.*

2. Start the Schema Manager.
3. When prompted, enter the host name which contains the Registry Host started in step 1 above.
4. Select the sample schema.
5. Verify that the Control Broker is connected (the message on the Control tab of the console indicates command succeeded and the status as up).
6. Right-click the IQ Manager and click **Start**.
7. Right-click each e*Way and select **Start**.
8. View the output by copying the **output** file (specified in the Outbound e*Way configuration file) to another location. Open the file in the new location.

Note: *Do not open the destination file while the schema is running. This will cause errors.*

Chapter 6

Teradata JDBC Implementation

This chapter provides information for implementing the Teradata (JDBC) e*Way in a production environment.

e*Way implementation is demonstrated by configuring the Teradata JDBC sample schema included with the e*Way. This chapter provides detailed information about how the Teradata JDBC sample schema is created. These procedures can be applied to implementing the Teradata (JDBC) e*Way in your specific production environment.

6.1 Teradata (JDBC) e*Way Implementation

Implementation of the Teradata (JDBC) e*Way involves completion of the following:

- Activate the Control Broker.
- Use the e*Gate Schema Designer to define and configure the following:
  - The e*Ways
  - The Teradata (JDBC) e*Way Connection
  - The Teradata (JDBC) Event Type Definition (ETD)
  - The Intelligent Queue (IQ)
  - Collaboration Rules, which are used to process Events.
  - Collaborations, associated with each e*Way component, to apply the required Collaboration Rules

This chapter describes how to perform each of these steps, using the sample schemas to demonstrate the creation of the e*Way components and their configuration.

- **The Teradata JDBC Sample Schema** on page 84 provides an overview of the creation and configuration of the samples components.
- **Executing the Schema** on page 104.

6.2 Concerns

The following information should be noted before implementing the Teradata sample schemas.

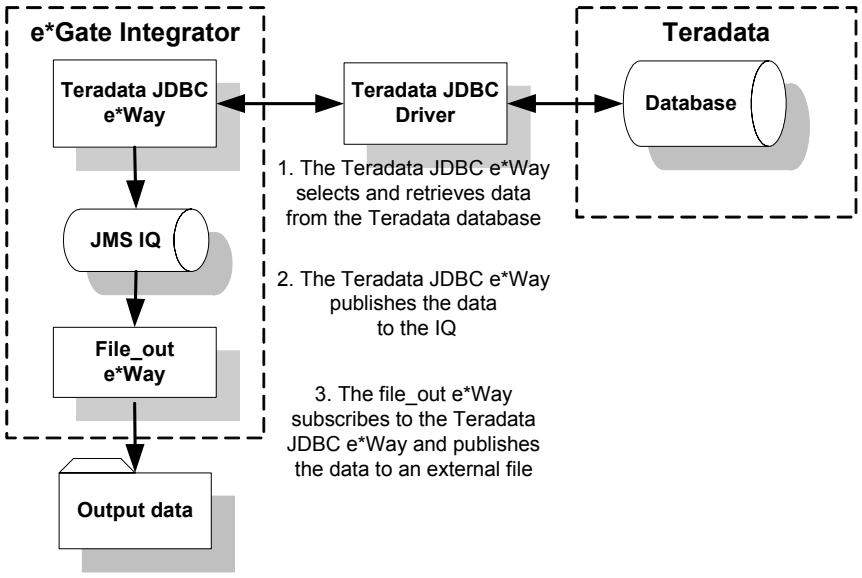
- **ODBC Driver Required for the Teradata JDBC schema:** The Teradata JDBC e*Way requires the installation and configuration of the Teradata ODBC driver on the e*Gate server. The ODBC driver is required for the Database Builder Wizard only. For more information see **ODBC Driver Installation** on page 84.
- **XA is not supported**: The e*Way Intelligent Adapter for Teradata does not support XA.

Review the Teradata e*Way Readme file for any additional requirements prior to implementation. This file is located on the e*Gate Integrator Installation CD-ROM in the *\setup\addons\ewteradata* directory.

6.3 The Teradata JDBC Sample Schema

The Teradata JDBD sample schema (TeradataDB_Select.zip), included with the Teradata e*Way adapter, demonstrates how the e*Way selects and retrieves data from the Teradata RDBMS via the Teradata JDBC driver, and publishes that data to an external file (see Figure 28).

Figure 28 Teradata e*Way - Teradata JDBC Sample Schema



The schema's components are nearly complete once the sample schema is imported into the e*Gate Schema Designer. The configuration settings must to be modified for your specific system.

ODBC Driver Installation

The Teradata JDBC e*Way requires the installation and configuration of the Teradata ODBC driver on the e*Gate server. The ODBC driver is used by the DB Wizard only.

Before you set up the Teradata JDBC e*Way, install the Teradata ODBC driver by doing the following:

1. Open the ODBC Data Source Administrator from Control Panel, Administrative Tools, Data Sources (ODBC).
2. From the ODBC Data Source Administrator, configure the ODBC DSN (Data Source Name) by selecting the Teradata driver.

6.3.1 Importing the Teradata JDBC Sample Schema

To import the Teradata JDBC sample schema to e*Gate, do the following:

1. Start **e*Gate Schema Designer**.
2. Enter the **password** associated with the user name displayed.
3. Click **Log In**.
4. In the **Open Schema on Registry Host** dialog box, click **New**.
5. In the **New Schema** dialog box, enter a name for the new schema (for this sample Teradata JDBC).
6. Select **Create from Export** and click **Find**.
7. Navigate to the location of the **TeradataDB_Select.zip** file (on the installation CD-ROM, this file is located in *..\samples\ewteradata\..*).
8. Select the zip file and click **Open**.

   The e*Gate Schema Designer opens with your new schema.

Note: *Schemas may also be imported from the e*Gate Schema Designer's File menu by selecting New Schema.*

The following sections describes the e*Way components for the Teradata JDBC sample schema. For more information on how to use the e*Gate Integrator to create these components, refer to **The TPump Sample Schema** on page 56, or see the *e*Gate Integrator User's Guide*. To create the components of the Teradata JDBC sample schema manually see the following:

- **Creating the e*Ways** on page 86
- **Creating the e*Way Connection** on page 87
- **Creating Event Type Definitions** on page 87
- **Creating Intelligent Queues** on page 99
- **Creating Collaboration Rules** on page 100
- **Creating the Business Rules** on page 101
- **Creating Collaborations** on page 101
- **Executing the Schema** on page 104

Note: *To run the Teradata JDBC sample schema, your enterprise must contain a functional Teradata server that can be accessed by the e*Gate Integrator server.*

6.3.2 Creating the e*Ways

The Teradata JDBC sample schema uses two e*Ways:

- The **file_out** e*Way, a component e*Way that publishes to an external directory.
- The **db_teradata** e*Way, a multi-mode e*Way that routes and transforms data within e*Gate.

Any system specific parameters associated with the e*Way must be configured to accommodate for your specific environment.

The file_out e*Way

The **file_out e*Way** uses the **stcewfile.exe** executable selected in the e*Way's properties file. The configuration parameters are set as shown in Table 17.

Table 17 file_out e*Way Parameters

file_out e*Way - Configuration File Parameters	
General settings - Set as directed, otherwise accept default.	
AllowIncoming	NO
AllowOutgoing	YES
PerformanceTesting	NO
Outbound (send) settings - Set as directed, otherwise accept default.	
OutputDirectory	data\Teradata
OutputFileName	teradatadb%d.dat
MultipleRecordsPerFile	YES
MaxRecordsPerFile	10000
AddEOL	YES
Performance Testing - Set as directed, otherwise leave as default.	
LogEveryXEvents	100
InboundDuplicates	1

When the configuration settings are complete, save the configuration file, and promoted to run time. From the e*Way Properties Start tab, make sure that **Start automatically** is selected.

The db_teradata e*Way

The **db_teradata e*Way** uses the **stceway.exe** executable selected in the e*Way's properties file. The configuration parameters are saved with the default settings. From the e*Way Properties Start tab, make sure that **Start automatically** is selected.

6.3.3 Creating the e*Way Connection

One e*Way Connection, named **ewc_terajdbc**, is created for the FastExport sample schema.

The ewc_terajdbc e*Way Connection

The Teradata JDBC sample schema contains one e*Way Connection named **ewc_terajdbc.** The **ewc_terajdbc e*Way Connection** uses the **Teradata** e*Way Connection Type, selected in the e*Way Connection's properties file. The configuration parameters are set as shown in Table 18. Accept the default settings for any parameters not listed in the table.

Table 18 e*Way Connection Parameters - ewc_terajdbc

e*Way Connections - ewc_terajdbc	
e*Way Name	ewc_terajdbc
e*Way Connection Type	Teradata
Event Type "get" interval	20000
DataSource Section - Set as directed, otherwise accept the default.	
driver	com.ncr.teradata.TeraDriver
jdbc url	The JDBC URL used to gain access to the Teradata database. A typically example is: jdbc:teradata://<Teradata Server Alias>
user name	The case-insensitive user name used to connect to the Teradata database
password	The password used to connect to the database
connector Section - Set as directed, otherwise accept the default.	
type	DB
class	com.stc.eways.jdbcx.DbConnector
transaction mode	Automatic
Connection Establishment Mode	Automatic

Save the configuration file and promote to runtime.

6.3.4 Creating Event Type Definitions

Two Event Types are used by the Teradata JDBC sample schema:

- **et_Generic**, which uses **GenericEvent.xsc** as the ETD (see **Creating the et_Generic Event Type** on page 88).
- **et_Teradata**, which uses **etdTeradataTbl1t.xsc**, an ETD that is generated using the Database Builder Wizard (see **Creating the et_Teradata Event Type** on page 88).

Creating the et_Generic Event Type

The et_Generic Event Type is created using the Custom ETD Wizard. To create the et_Generic Event Type and the GenericEvent.xsc do the following:

1. Click the **Event Types** folder from e*Gate's **Components** tab.
2. Click the **Create a New Event Type** button.
3. Enter the **name** of the **Event Type** (for this sample, **et_Generic**) and click **OK**.
4. Double-click the **new Event Type** to open the Properties dialog box.
5. In the **Event Type Properties** dialog box click **New**. The ETD Editor appears.
6. From the **File** menu, select **New**. The **New Event Type Definition** dialog box appears.
7. Double-click the **Custom ETD Wizard** icon. The Custom ETD Wizard appears.
8. Enter the following:
   C. A Root Node Name (for this sample, **GenericEvent**)
   D. A **Package Name** (this is where the ETD Editor places the generated Java classes associated with the newly-created ETD)
9. Click **Next**, review your information, then click **Finish**. The ETD Editor now displays the generated ETD.
10. Right-click the root node (**GenericEvent**) in the Event Type Definition pane.
11. From the **Add Field** sub-menu, select **As Child Node**. A node named **Field1** is added.
12. Triple-click **Field1**, and rename the node as **payload**.
13. Select the **payload** node. Selecting a node displays the node's properties in the **Properties** pane. Change the value of the **structure** property to **fixed**.
14. From the **File** menu, select **Compile and Save**, and **Promote to Run Time**.
15. Close the ETD Editor and the Event Type Properties dialog box.

Creating the et_Teradata Event Type

The **et_Teradata** Event Type uses the **etdTeradataTbl1t.xsc.** etdTeradataTbl1t.xsc is generated using the DB (database builder) Wizard.

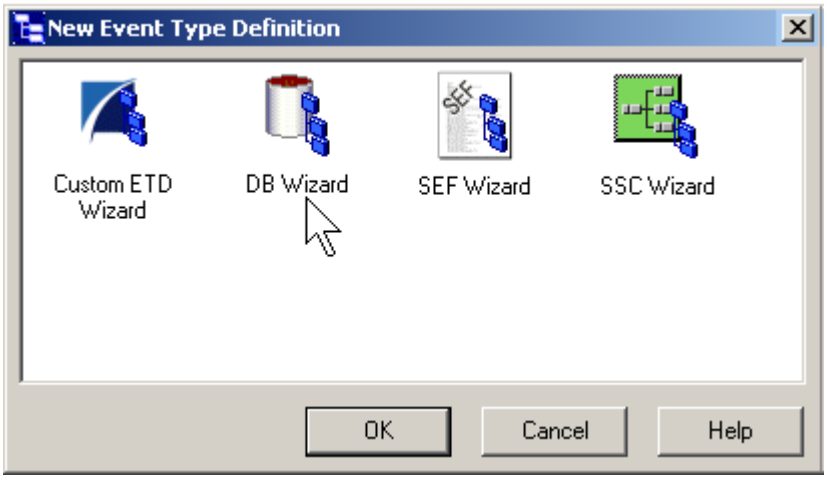
1. Click the **Event Types** folder from e*Gate's **Components** tab.
2. Click the **Create a New Event Type** button.
3. Enter the **name** of the **Event Type** (for this sample, **et_Generic**) and click **OK**.
4. Double-click the **new Event Type** to open the Properties dialog box.
5. The following section, **Creating an ETD Using the DB Wizard** on page 89, describes how the DB Wizard is used to create the ETD, **etdTeradataTbl1t.xsc**.

Creating an ETD Using the DB Wizard

The DB (database builder) Wizard is used to generate Java ETDs. In this sample, the wizard connects to the Teradata RDBMS and generates an ETD that corresponds to the external tables, views, and prepared statements. The created ETD is read-only, which means that the contents of the generated ETD cannot be edited. The following example describes the creation of the ETD, **etdTeradataTbl1t.xsc**.

1. From the **Options** menu of the e*Gate Schema Designer, select **Default Editor**.
2. Verify that **Java** is selected, then click **OK**.
3. Click the **ETD Editor** button to launch the Java ETD Editor.
4. In the ETD Editor, click the **New** button to open the **New Event Type Definition** dialog box.

Figure 29 New Event Type Definition - Database Builder Wizard



5. In the this dialog box, select the **DB Wizard** icon and click **OK** to continue (see Figure 29). The wizard's **Introduction** dialog box appears (see Figure 30).

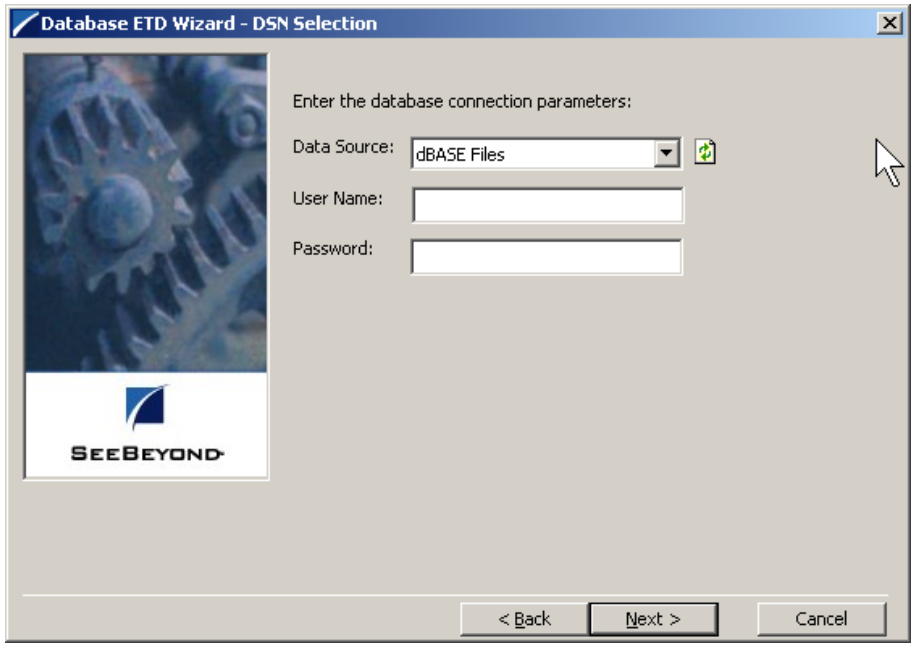
Figure 30 Database Builder Wizard - Introduction

6 Do one of the following actions by selecting the desired option:

- Create a new .**xsc** file.
- Modify an existing .**xsc** file. To select the desired file do the following:
  - Click **Browse** to open the **Registry File Selection** dialog box.
  - Navigate to the files location, select the file, and click **Open**.

7 Click **Next** to continue. The **DSN Selection** dialog box appears (see Figure 31).

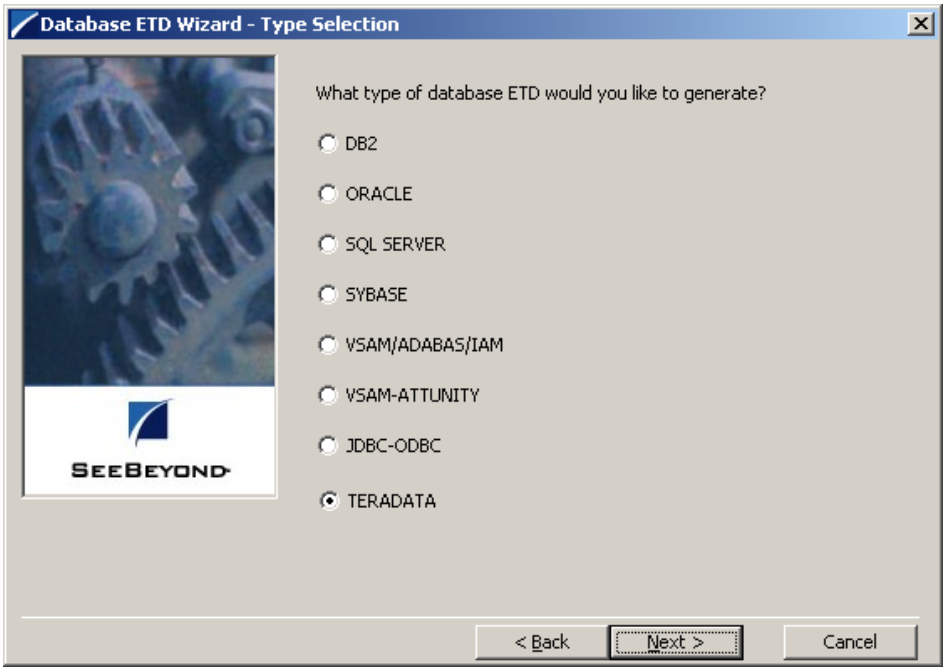
Figure 31 Database Builder Wizard - DSN Selection



8 Select a **Data Source** from the drop-down list and enter a **User Name** and **Password**.

9 Click **Next** to continue. The **Type Selection** dialog box appears (see **Figure 32 on page 91**).

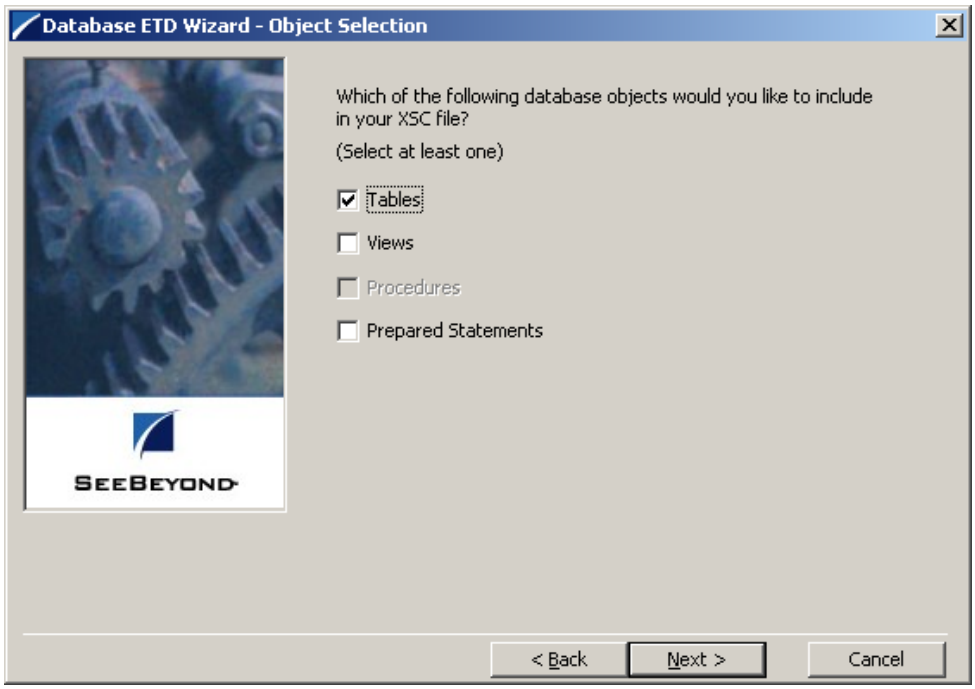
Figure 32 Database Builder Wizard - Type Selection



10 Select the type of database ETD you would like to generate (in this case TERADATA) as shown in Figure 32. The data source you selected in the wizard's **DSN Selection** dialog box becomes the default.

11 Click **Next** to continue. The wizard's **Object Selection** dialog box appears (see Figure 33).

Figure 33 Database Builder Wizard - Object Selection

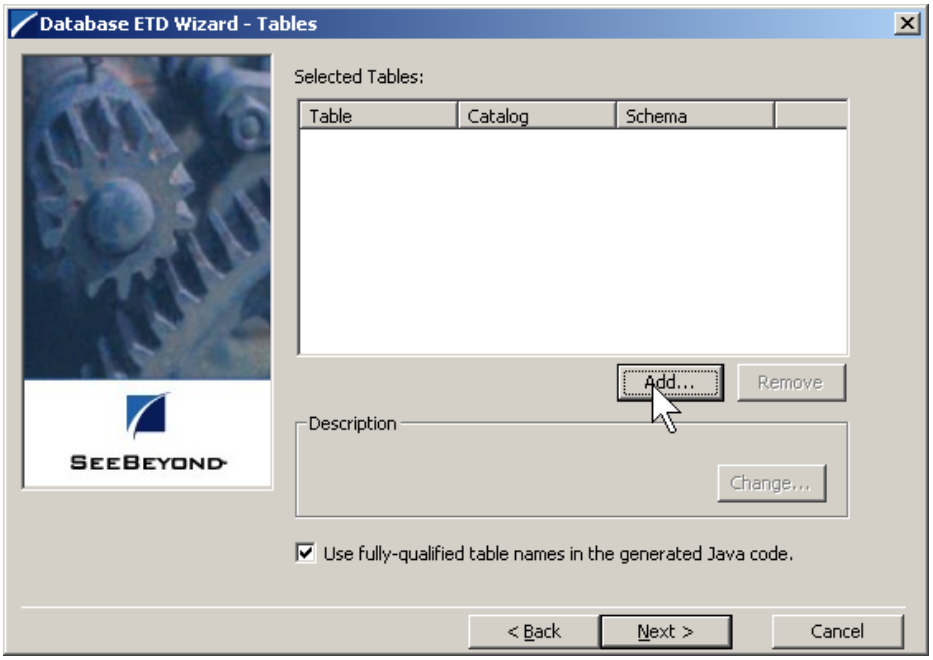


12 In the **Object Selection** dialog box, select **Tables**, **Views**, and/or **Prepared Statements**, depending on which database objects you want to include in the **.xsc** file (for this sample, select **Tables** only).

Note: *The selection **Procedures** is not available and is displayed as disabled. The e*Way Intelligent Adapter for Teradata does not support Stored Procedures*

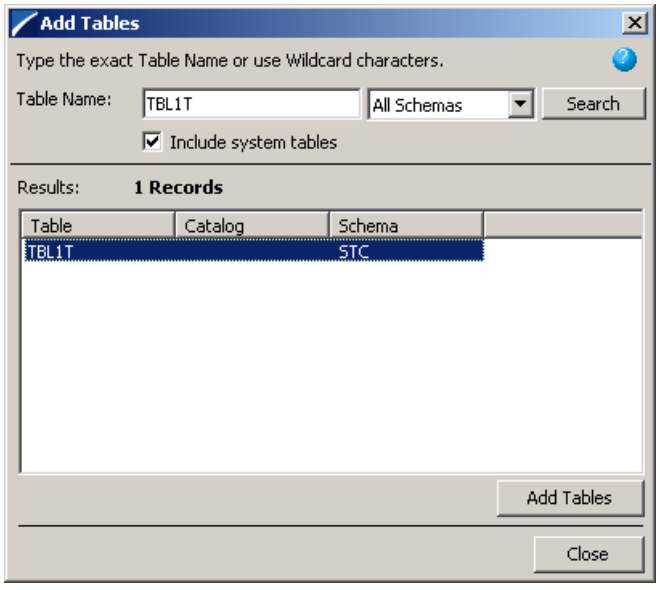
13 Click **Next** to continue. The wizard's **Tables** dialog box appears (see Figure 34). This dialog box allows you to select the tables you want to add to the ETD.

Figure 34 Database Builder Wizard - Tables



14 In this dialog box, click **Add Tables**. The **Add Tables** dialog box appears (see Figure 35).

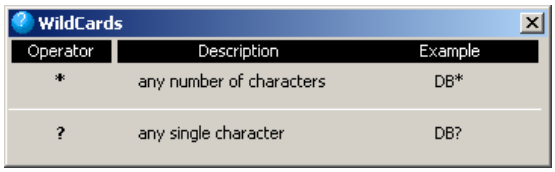
Figure 35 Add Tables dialog box



15 In the **Add Tables** dialog box, type the exact name of the database table or use wildcard characters to return table names.

16 To see a list of valid wildcard characters, click the round ball with a question mark located in its center (see Figure 36).

Figure 36 Add Tables - Wildcards



17 Select the location to search for the specific tables, in the field to the right of the View Name field.

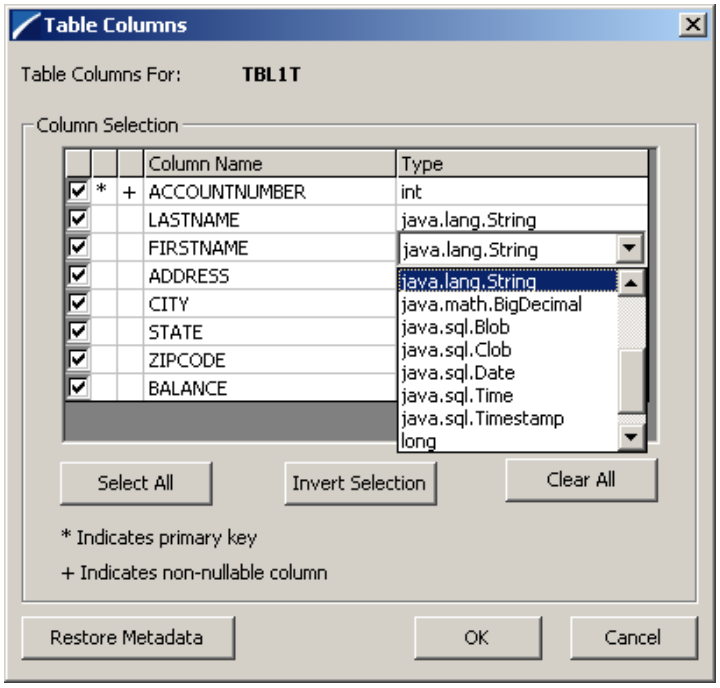
18 Select **Include System Tables** if you wish to include them and click **Search**. If your search was successful, you will see the results in the **Results** dialog box.

19 To select the name of the tables you wish to add to your .**xsc** file, double-click on the table name or highlight the table names (in this case, TBL1T) and click **Add Tables**. You can also use adjacent selections or nonadjacent selections to select multiple table names. When you have finished, click **Close**.

20 In the wizard's **Tables** dialog box, review the tables you have selected. If you would like to change any of the tables you have selected, click **Change**.

21 In the **Columns Selection** dialog box, you can select or de-select your table choices. You can also change the data type for each table by highlighting the data type and selecting a different data type from the drop-down list. For this example all the table choices are retained (see Figure 37).

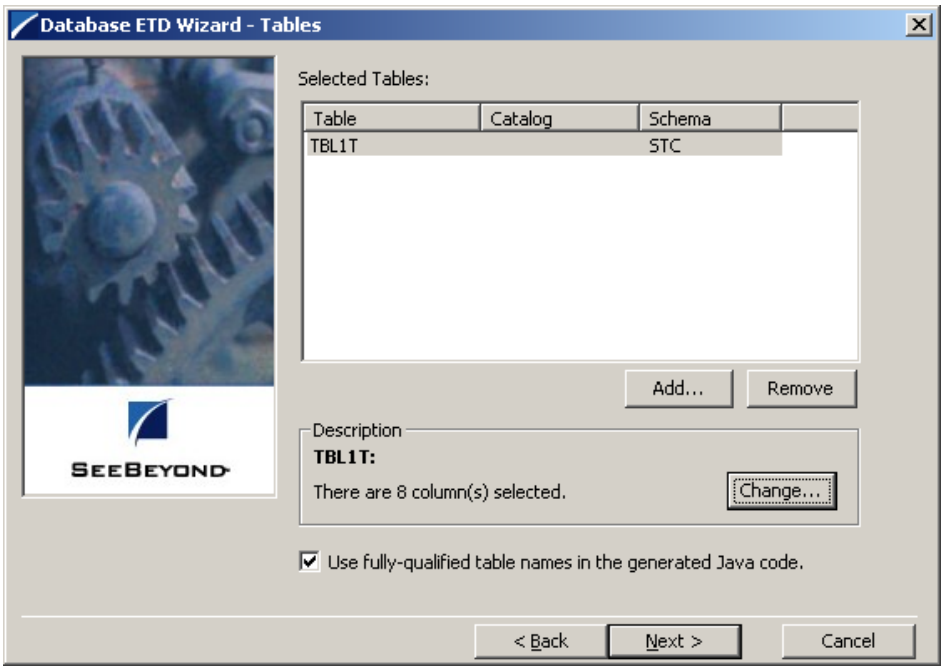
Figure 37 Columns Selection dialog box



22 Once you have completed your choices, click **OK**. The wizard returns you to the **Tables** dialog box.

23 In this dialog box, review the tables you have selected (see Figure 38).

Figure 38 Database Builder Wizard - Tables with Table Names Added



24 To use of fully-qualified table names in the generated Java code, leave the check box selected, or click on the check box to clear it if you do not want to specify this option. When you are finished with this dialog box, click **Next** to continue.

25 If you selected **Views** on the wizard's **Object Selection** dialog box, you are now presented with the wizard's **Views** dialog box. To add views to your **.xsc**. file, complete the following steps:

A Click **Add** to add views. The **Add Views** dialog box opens.

B Enter a view name or use wildcard characters to return view names (see Figure 36) in the **View Name** field.

C Select the location to search for the specific views, in the field to the right of the View Name field.

D Click **Search**. If your search was successful, you will see the results in the **Results** dialog box

E Select the **views** to be used by the statement and click **Add Views**.

F Click **Close** to return to the wizard's **Views** dialog box.

G Review the views you have selected. If you would like to change any of the views you have selected, click **Change**.

H To use of fully-qualified view names in the generated Java code, leave the check box selected, or click on the check box to clear it if you do not want to specify this option. When you are finished with this dialog box, click **Next** to continue.

26 If you selected **Prepared Statements** on the wizard's **Object Selection** dialog box, you are presented with the wizard's **Prepared Statement** dialog box. To add prepared statements to your .**xsc**. file, complete the following steps:

A Click **Add** to add a new prepared statement. The **Add Prepared Statement** dialog box appears.

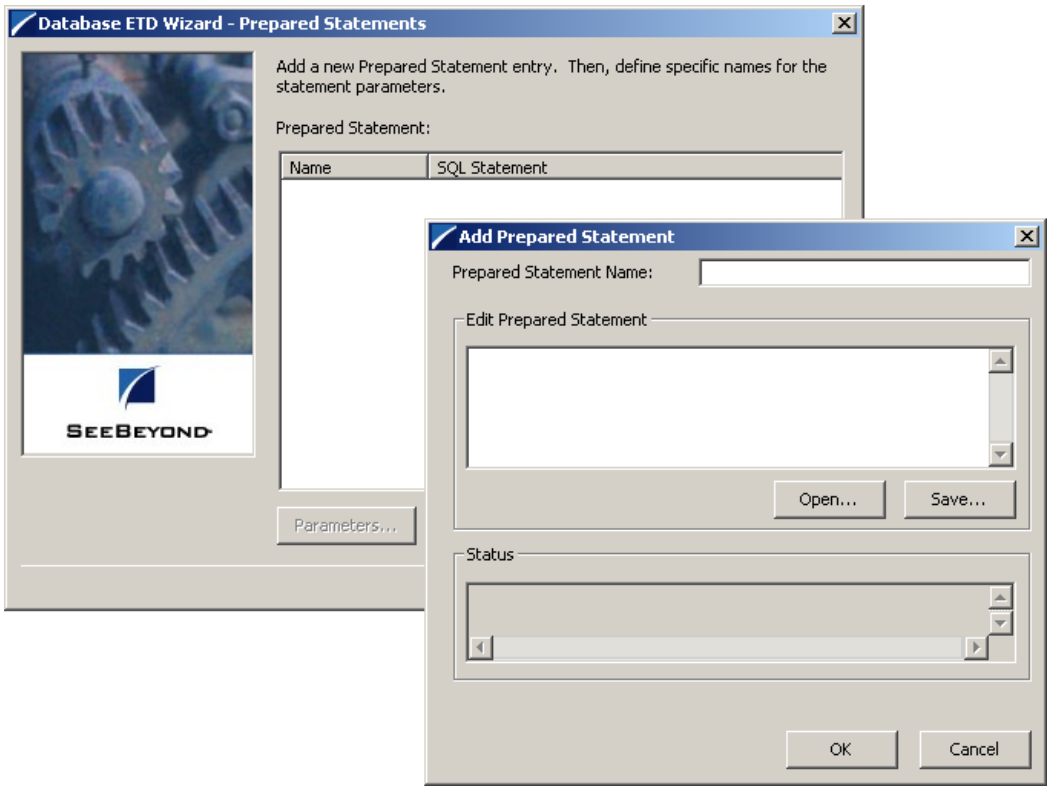
B Enter a name to be used by the statement in the **Prepared Statement Name** field.

C Click **Open** to browse for and open preexisting statements. If necessary, edit the statement and click **Save** to save the statement in the appropriate location.

D Click **OK** to return to the wizard's **Prepared Statements** dialog box. If is a problem with the prepared statement errors are displayed in the **Status** field.

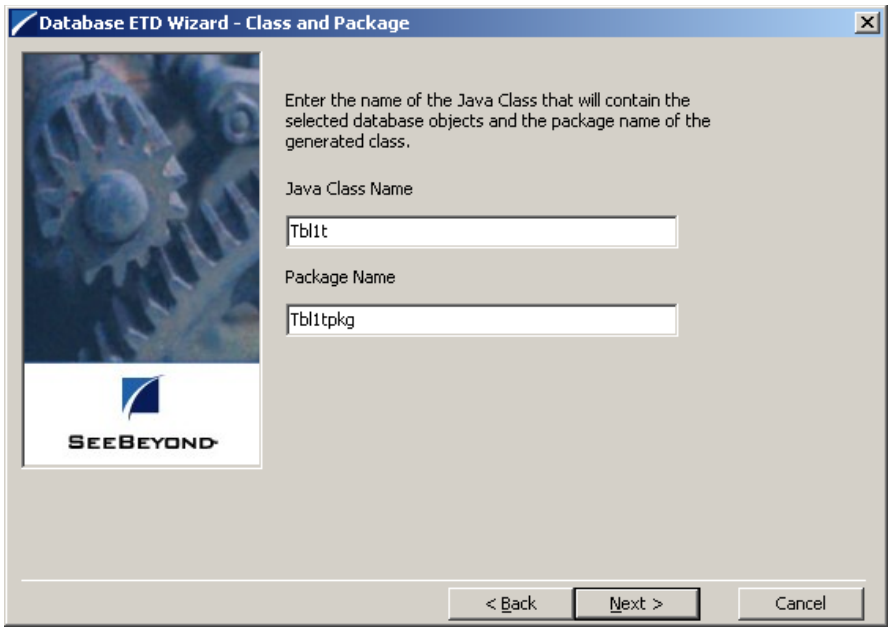
27 Repeat the sub-steps under step 26 to add additional prepared statements (see Figure 39) or click **Next** to continue.

Figure 39 Database Builder Wizard - Prepared Statements



28 The wizard's **Class and Package** dialog box appears (see Figure 40).

Figure 40 Database Builder Wizard: Class and Package

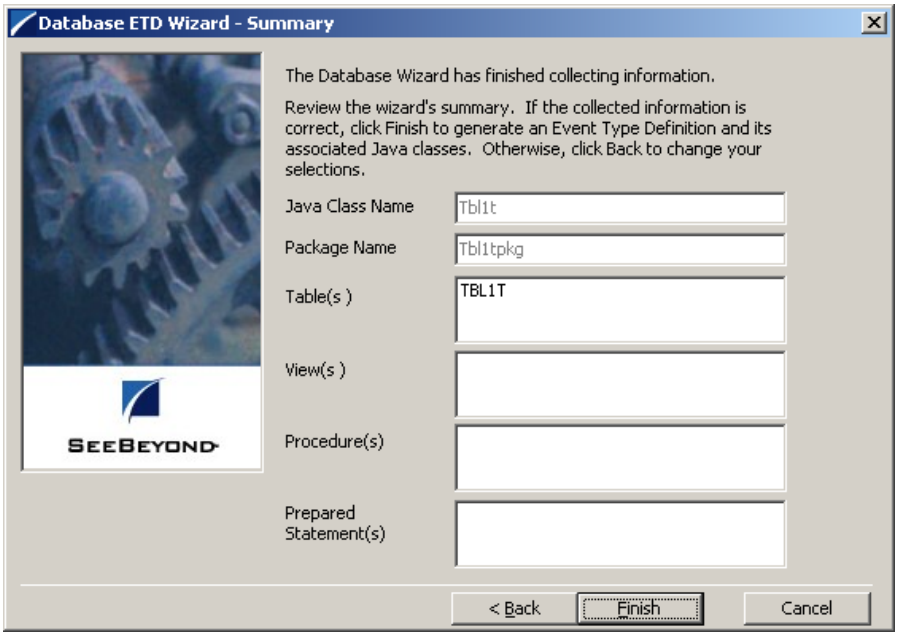


29 Enter the **Java Class Name** that is to contain the selected tables, views, and/or prepared statements.

30 Enter the **Package Name** of the generated classes.

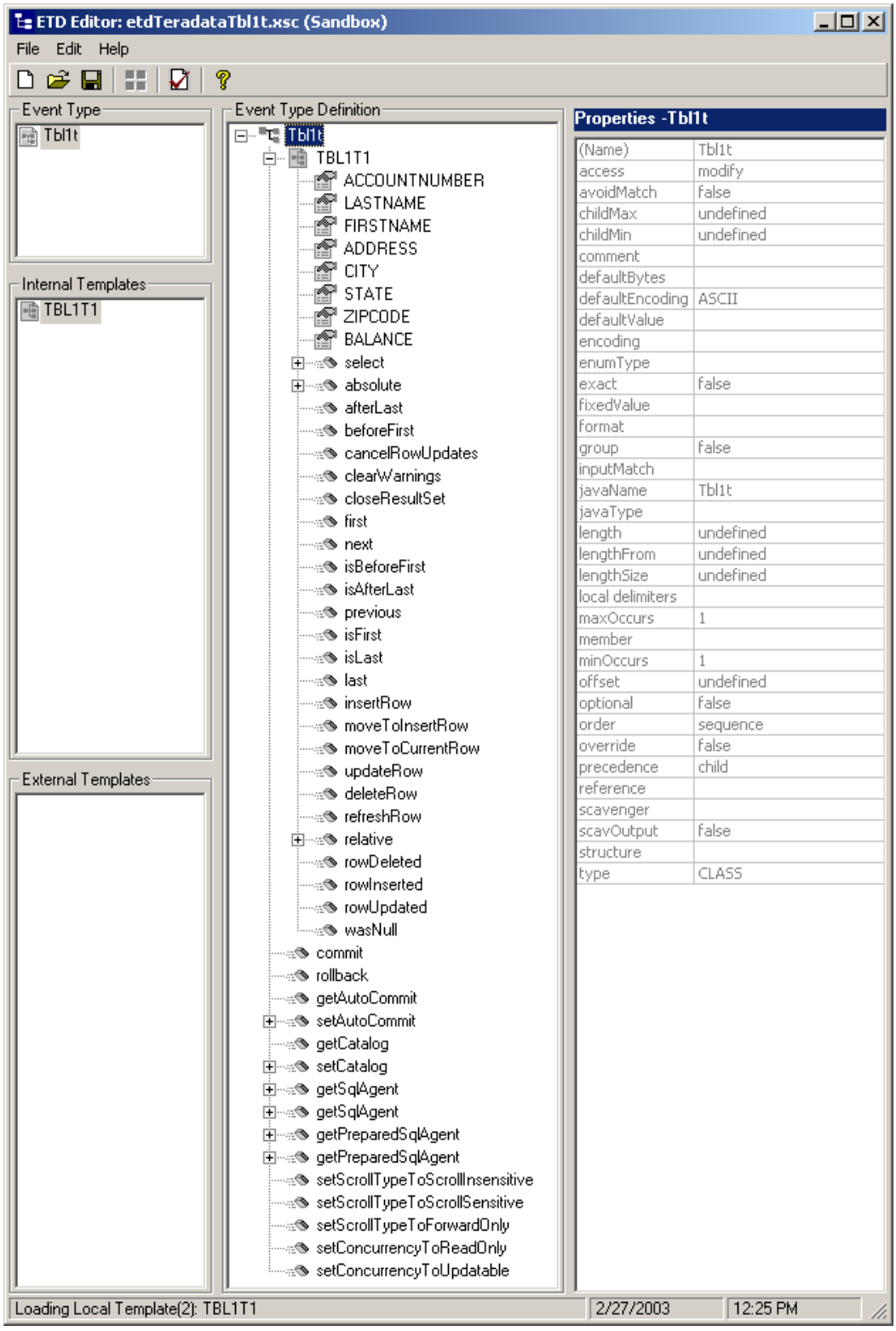
31 Click **Next**. The Database ETD Wizard Summary dialog box appears. Review the information you have entered in the database wizard (see Figure 41).

Figure 41 Database Builder Wizard: Summary



32 Click **Finish** to generate the ETD. The generated ETD's node structure appears in the ETD Editor's Main Window as shown in Figure 42.

Figure 42 ETD Editor Main Window with Generated ETD



33 Click **Save** to save the ETD. Enter the name of the new ETD in the File name field of the **Save** dialog box (for this sample, etdTeradataTbl1t.xsc).

6.3.5 Generated ETDs

The Database Builder wizard can create editable ETDs for the e*Way. These types of ETDs can also be combined with each other. The types of ETDs are:

- **Table**: Contains fields for each of the columns in the selected table, as well as the methods required to exchange data with an external data source. To edit this type of ETD, you need to open the **.xsc** file in the ETD Editor.
- **View**: Contains selected columns from selected tables. This type of ETD is read-only.
- **Prepared Statements**: Contains a result set for a prepared statement. To edit this type of ETD, you need to open the **.xsc** file in the ETD Editor.

6.3.6 Editing an Existing .xsc File

You can use the Database Builder wizard to edit an existing **.xsc** file you have created.

To edit the .xsc file

1 From the **Options** menu of the e*Gate Schema Designer, choose **Default Editor**.

2 Verify that **Java** is selected, then click **OK**.

3 From the **Tools** menu, click **ETD Editor**.

4 From the ETD Editor's **Tools** menu click **File** then **New**.

5 From the **New Event Type Definition** dialog box, select the **DB Wizard** icon and click **OK**.

6 On the Database Builder wizard's **Introduction** dialog box, select **Modify an existing XSC file** and browse to the appropriate **.xsc** file you want to edit.

You are now able to edit the **.xsc** file.

Note: *When you add a new element type to an existing **.xsc** file, you must reselect any preexisting elements or you loose them when the new **.xsc** file is created.*

7 If you attempt to edit an **.xsc** file whose elements no longer exist in the database, you see a warning, and the element is dropped from the ETD.

6.3.7 Creating Intelligent Queues

One IQ is created for the Teradata JDBC sample schema. To create the JMS IQ do the following:

1 From e*Gate's Navigator pane, expand and select the Control Broker. Create an IQ Manager if one does not already exists, using the **Create a New IQ Manager** button on the Tool Palette, or select the current IQ Manager.

2 Open the IQ Manager Properties and select SeeBeyond JMS as the IQ Manager Type.

3 Click **New** under the Configuration file field, save the default file and promote to run time.

4 Select the **IQ Manager** in the Navigator pane and click the **Create a New IQ** button.

5 Name the IQ (for this sample, **jmsiq**) and click **OK**. Open the new IQ's Properties dialog box. The Service defaults to STC_JMS_IQ. Click **Apply** and **OK**.

6.3.8 Creating Collaboration Rules

Two Collaboration Rules are created for the Teradata JDBC sample schema.

- **cr_passthru**, which uses the **Pass Through** Service.
- **cr_teradata**, using the **Java** Service.

Creating the cr_passthru Collaboration Rules

To create the **cr_passthru** Collaboration Rules follow the directions for **Creating the pass_thru Collaboration Rules** on page 64, replacing the Collaboration Rules name, **pass_thru** with **cr_passthru**.

Creating the cr_teradata Collaboration Rules

To create the **cr_teradata** Collaboration Rules do the following:

1 Select **Collaboration Rules** in the e*Gate Navigator pane. Click the **Create New Collaboration Rules** button and name the New Collaboration Rules (for this sample, **cr_teradata**).

2 Double-click the new Collaboration Rules. The Collaboration Rules Properties dialog box appears.

3 From the General tab, select **Java** as the Service.

4 From the Collaboration Mapping tab, use the **Add Instance** button to create two instances and enter the values as shown in Figure 43.

Figure 43 Collaboration Mapping - fexp_export

Instance Name	ETD		Mode	Trigger	Manual Publ...
Generic	GenericEvent.xsc	Find ...	Out	N/A	☑
DBTeradata	etdTeradataTbl1t.xsc	Find ...	In	☑	N/A

5 From the General tab, click **OK** to close the Collaboration Rules Properties dialog box, or click **New** (or **Edit**) to open the Collaboration Rules editor and create the Business Rules.

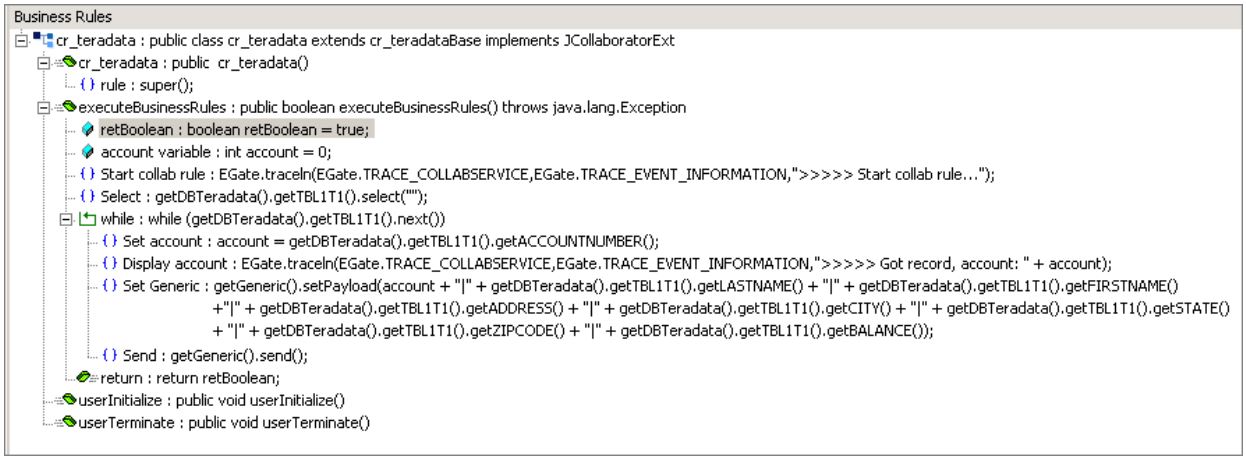
6.3.9 Creating the Business Rules

The **cr_teradata** Collaboration Rules contain Business Rules used to select and retrieve data from the Teradata RDBMS via the Teradata JDBC Driver. Business Rules for the **cr_teradata** Collaboration Rules are created using the Collaboration Rules Editor.

To create the **cr_teradata** Business Rules using the Collaboration Rules Editor do the following:

1. From the General tab of the **cr_teradata** Collaboration Rules Properties dialog box, click **New** (or **Edit**) under the Collaboration Rules field. The **Collaboration Rules Editor** appears
2. Expand the Collaboration Rules Editor to the maximum size for optimum viewing, expanding the Source and Destination Events as well. All user-defined variables and rules are added as part of **executeBusinessRules()**.
3. Create the Business Rules as shown in Figure 44.

Figure 44 Collaboration Rules Editor - cr_teradata Business Rules



Note: *The cr_teradata Business Rules are displayed as wrapped due to size restrictions. The editor does not wordwrap code in the Business Rules pane.*

4. When the Business Rules are complete, save and compile the file. If the file fails to compile use the Compiler window to find any errors in the code. Correct any errors and compile again until all the errors have been corrected.
5. Save and promote the completed Collaboration Editor Project.

For more information on using the Collaboration Rules Editor and creating Business Rules see the *e*Gate Integrator User's Guide*.

6.3.10 Creating Collaborations

Two Collaborations are created for the Teradata JDBC sample schema:

- **col_file**, associated with the file_out e*Way.
- **col_db**, associated with the db_teradata e*Way.

Creating the file_in Collaboration

To create the **col_file** Collaboration do the following:

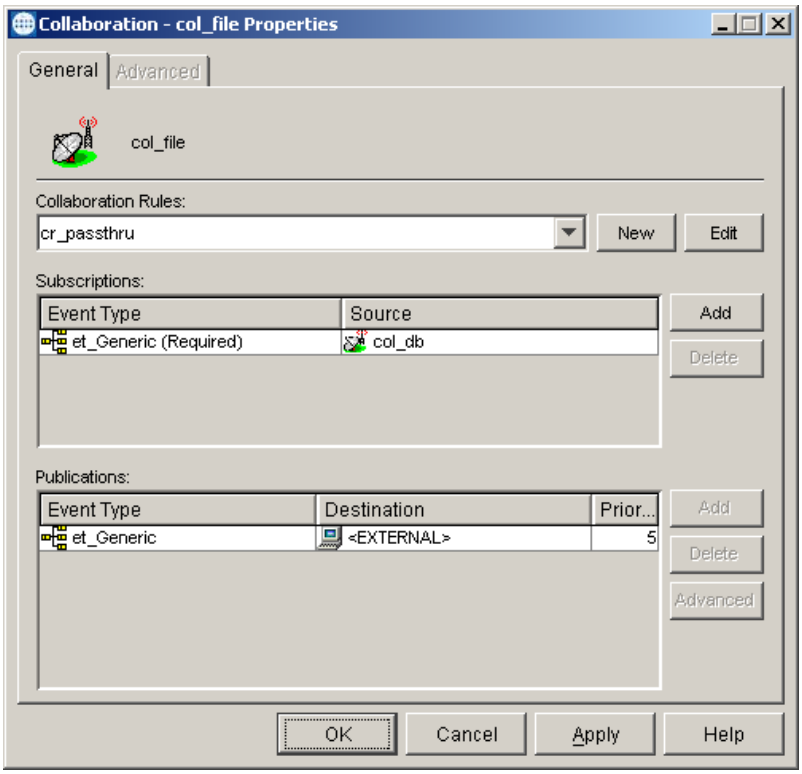
1 From the e*Gate Navigator pane, select the **file_out** e***Way** and click the **Create a New Collaboration** button on the Tool Palette.

2 Name the Collaboration (for this sample, **col_file**).

3 From the e*Gate Editor pane, double-click the new Collaboration. The **Collaboration Properties** dialog box appears.

4 Configure the **col_file** Collaboration Properties to match those in Figure 45.

Figure 45 Collaboration Properties - file_in



5 When completed, click **Apply** and **OK**.

Creating the col_db Collaboration

To create the **col_db** Collaboration do the following:

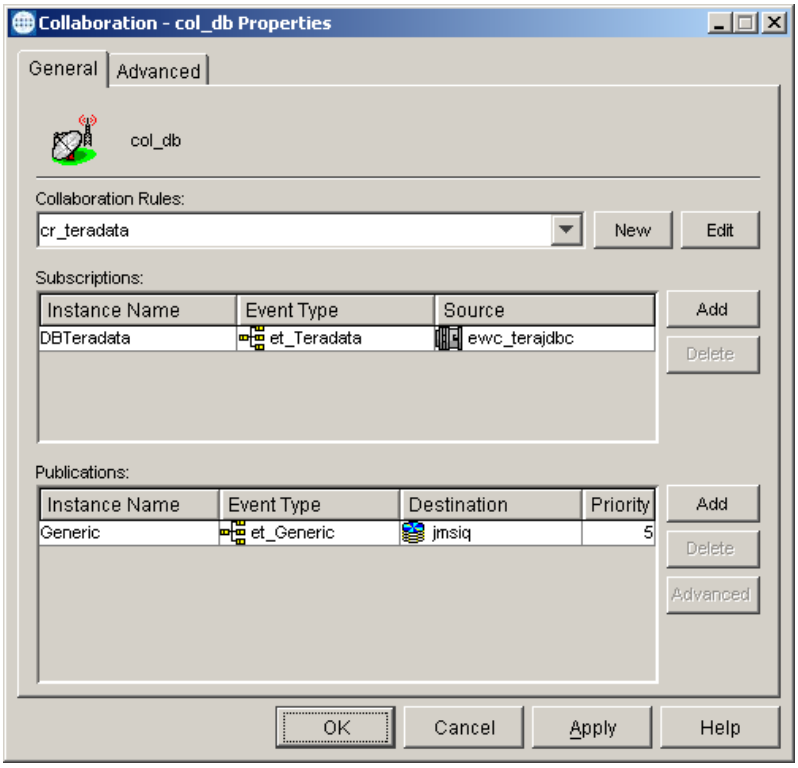
1 From the e*Gate Navigator pane, select the **file_in e*Way** and click the **Create a New Collaboration** button on the Tool Palette.

2 Name the Collaboration (for this sample, **col_db**).

3 From the e*Gate Editor pane, double-click the new Collaboration. The **Collaboration Properties** dialog box appears.

4 Configure the **col_db** Collaboration Properties to match those in **Figure 46 on page 103**

Figure 46 Collaboration Properties - col_db



5 When completed, click **Apply** and **OK**.

6.3.11 Sample Table and Data

To create the sample table use the following information:

```
CREATE TABLE TBL1T
ACCOUNTNUMBER INTEGER NOT NULL,
LASTNAME VARCHAR(25),
FIRSTNAME VARCHAR(25),
ADDRESS VARCHAR(30)
CITY VARCHAR(20)
STATE CHAR(2)
ZIPCODE CHAR(5)
BALANCE DECIMAL(9,2))
UNIQUE PRIMARY INDEX (ACCOUNTNUMBER);
```

Note: The sample table and data for the Teradata JDBC sample schema is also created when the TPump sample schema is run.

6.4 Executing the Schema

Running the Schema

Run the Teradata JDBC sample schema as follows:

1 At the command prompt, enter the following:

```
stccb -rh hostname -rs schemaname -un username -up user password
    -ln hostname_cb
```

Note: *Substitute the italicized values with the specific values for your schema.*

2 Start the Schema Manager.

3 When prompted, enter the host name which contains the Registry Host started in step 1 above.

4 Select the sample schema.

5 Verify that the Control Broker is connected (the message on the Control tab of the console indicates command succeeded and the status as up).

6 Right-click the IQ Manager and click **Start**.

7 Right-click each e*Way and select **Start**.

8 View the output by copying the **output** file (specified in the Outbound e*Way configuration file) to another location. Open the file in the new location.

Note: *Do not open the destination file while the schema is running. This will cause errors.*

Chapter 7

e*Way Classes and Methods

The Teradata e*Way contains a number of Java methods that have been exposed to make it easier for the user to access Teradata RDBMS. These methods are contained in the Teradata ETL Classes and the Teradata JDBC Classes.

7.0.1. Teradata ETL Classes

The Teradata ETL methods are contained in the following classes:

- The **ETLRun** Class
- The **ScriptFile** Class
- The **TeradataETL** Class
- The **TeradataETL.Runtime** Inner Class
- The **TeradataETL.Runtime. ErrorOutput** Inner Class
- The **TeradataETL.Runtime. FExpRun** Inner Class
- The **TeradataETL.Runtime.JobOutput** Inner Class
- The **TeradataETL.Runtime.JobOutput.Rows** Inner Class
- The **TeradataETL.Runtime.TPumpRun** Inner Class
- The **TeradataETL.Script** Inner Class
- The **TeradataETL.Script.ScriptContent** Inner Class
- The **TeradataETL.Script.ScriptContent.ETLCommand** Inner Class
- The **TeradataETL.Script.ScriptContent.SQLStatement** Inner Class
- The **TeradataETLConnector** Class

Teradata ETL Javadoc

For a complete list of the Java methods within the classes listed above, refer to the **Javadoc** at the following location:

<eGate>\client\docs\javadocs\Teradata_eWay\index.html

<eGate> *represents the directory where your e*Gate documentation is installed.*

7.0.2. Teradata JDBC Interfaces and Classes

These Teradata e*Way JDBC methods are contained in the following classes:

Interfaces

- The **ResetEventListener** Interface
- The **SessionEventListener** Interface
- The **StatementEventListener** Interface

Classes

- The **AgentFactory** Class
- The **CallableStatementAgent** Class
- The **DbConnector** Class
- The **PreparedStatementAgent** Class
- The **PreparedStatementResultSet** Class
- The **ResetEvent** Class
- The **ResultSetAgent** Class
- The **Session** Class
- The **SessionEvent** Class
- The **SessionFactory** Class
- The **SqlObjectGroup** Class
- The **SqlObjectGroup.MyXAResource** Class
- The **SqlObjectGroupExt** Class
- The **SqlStatementAgent** Class
- The **StatementAgent** Class
- The **StatementEvent** Class
- The **StoredProcedureAgent** Class
- The **TableResultSet** Class

Teradata JDBC Javadoc

For a complete list of the Java methods within the classes listed above, refer to the **Javadoc** at the following location:

<eGate>\client\docs\javadocs\JDBC\index.html

<eGate> *represents the directory where your e*Gate documentation is installed.*

Index

A

B

C

D

E

F

G

H

I

J

K

M

N

O

P

R

S

T

U

W